Redbooks
ibm.com/redbooks

The Virtualization Cookbook for IBM z Systems Volume 4: Ubuntu Server 16.04

Lydia Parziale
Pedro Acosta
Fred Bader
Paul Novak

z Systems

IBM

Redbooks

International Technical Support Organization

The Virtualization Cookbook for IBM z Systems Volume 4: Ubuntu Server 16.04

September 2016

SG24-8354-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (September 2016)

This edition applies to Version 6, Release 3, of IBM z/VM and Ubuntu Server 16.04.

Contents

Notices vii
Trademarks viii

IBM Redbooks promotions ix

Preface xi
Important information before you begin xii
Architectural decisions xii
General criteria xii
Food for thought xiii
Specific examples xiii
Conventions xiv
Font conventions that are used in this cookbook series xiv
Command conventions that are used in this cookbook series xiv
Operating system releases used xiv
Authors xv
With gratitude xv
Now you can become a published author, too xvi
Comments welcome xvii
Stay connected to IBM Redbooks xvii

Part 1. Ubuntu Server 16.04 1
At a glance 2

Chapter 1. Install Ubuntu Server 16.04 on LNXADMIN 3
1.1 Prepare the Ubuntu bootstrap files 4
1.2 Install Ubuntu onto the Linux administration system 6
1.2.1 Stage 1: The initial network configuration 7
1.2.2 Stage 2: The Ubuntu 16.04 installation 12
1.2.3 Partition disks for the LNXADMIN virtual server 14
1.2.4 Finishing the installation 18
1.3 Configure the Ubuntu administration system 20
1.3.1 Enable swap on the virtual disks 20
1.3.2 Set up data DASD disks to a running Linux virtual server 21
1.3.3 Build a local Ubuntu mirror with apt-mirror on the LNXADMIN virtual server 25
1.3.4 Install and configure an apache2 HTTP server 27
1.3.5 Reconfigure the software repositories 28
1.3.6 Install and configure vsftpd FTP server 30
1.3.7 Import files that are associated with this book 31
1.3.8 Download the bootstrap files for Ubuntu 16.04 31
1.3.9 Additional configurations 32
1.3.10 Reboot the system 34
1.3.11 Verify that the system starts correctly 34

Chapter 2. Automating installation 35
2.1 Kickstart 36
2.1.1 Using Kickstart with EDEV 36
2.2 Configure LNXS0031 for Kickstart using FCP devices 37
2.3 How to boot SCSI over FCP (LNXS0031) 40

2.4 Preseeding 44
2.5 Cloning 44
2.5.1 Cloning considerations 44
2.5.2 The cloning process 45
2.6 Create golden image for cloning 45
2.7 Clone the golden image using DirMaint 46
2.8 Send a configuration update to the clone system 47
2.9 Start the clone system 48

Chapter 3. Servicing Linux: Updates and patching 49
3.1 Package management 50
3.1.1 Cross-architecture consistency 50
3.1.2 The dpkg manager and Advanced Package Tool 50
3.1.3 Determining which packages are installed 50
3.1.4 Determining release or version numbers and codenames 52
3.2 Canonical Landscape 53
3.2.1 Setup 54

Part 2. Other topics 55

Chapter 4. Working with disks 57
4.1 Add disk space to virtual machines 58
4.1.1 Make new minidisks or ECKD DASD available in Linux 58
4.1.2 Make a new EDEV available to Linux 59
4.1.3 Make a new zFCP LUN available to Linux 60
4.2 Add a logical volume 62
4.2.1 Create a logical volume and file system 63
4.2.2 Update the file system table and mount the file system 67
4.3 Extend an existing logical volume 68
4.4 Move a physical volume 72

Chapter 5. Workload containers and service orchestration 77
5.1 Run workloads in containers 78
5.1.1 Introducing Docker 78
5.1.2 Installing Docker 79
5.1.3 Updating Docker 79
5.1.4 Uninstalling Docker 79
5.1.5 Docker Universal Control Plane and Docker Cloud 80
5.2 Service orchestration 80
5.2.1 Introducing Juju 80
5.2.2 Installing Juju 2.0 81
5.2.3 Adding and deploying a model 92
5.2.4 Invoking the graphical user interface 93
5.2.5 Other available resources 94

Chapter 6. Working with systemd 95
6.1 Getting started with systemd 96
6.1.1 The systemd unit files 96
6.1.2 Relationship between units 97
6.2 Using systemd units 98
6.2.1 Managing services 98
6.2.2 Managing systemd target units 101
6.3 Working with the systemd journal 102
6.3.1 Getting started with the journal 102

6.3.2 Viewing the journal . . . 105
6.3.3 Filtering the journal . . . 106
6.4 The system boot process . . . 107
6.5 Analyze Linux instances that use systemd . . . 107
6.5.1 Retrieving performance statistics . . . 107
6.5.2 Retrieving information about unit dependencies . . . 109

Chapter 7. Miscellaneous recipes . . . 113
7.1 Rescue a Linux system . . . 114
7.1.1 The initrd shell and systemd targets . . . 114
7.1.2 Rescue Ubuntu 16.04 using systemd targets . . . 114
7.2 Set up Memory Hotplugging . . . 115
7.3 Dynamically add or remove virtual CPUs . . . 118
7.4 Use the cpuplugd service . . . 119
7.4.1 Determine the virtual CPUs being used . . . 120
7.4.2 Generating a workload to see cpuplugd work . . . 122
7.4.3 Setting memory sizes with cpuplugd . . . 124
7.5 Hardware Cryptographic Support for OpenSSH using Ubuntu 16.04 . . . 124
7.6 Use Crypto Express to seed /dev/random . . . 128
7.7 Graphical user interfaces . . . 130
7.7.1 The X Window System . . . 130
7.7.2 Window managers and graphical desktop environments . . . 130
7.7.3 VNC Server . . . 131
7.7.4 X Server on a workstation . . . 132
7.8 Setting up the IUCV Linux Terminal Server . . . 134
7.8.1 IBM z/VM configuration for Linux Terminal Server . . . 134
7.8.2 Ubuntu Linux Enterprise 12 configuration for IUCV consoles . . . 134
7.8.3 SUSE Linux Enterprise Server 12 configuration for IUCV Linux Terminal Server . . . 134
7.9 Issue z/VM CP commands from Linux . . . 135
7.10 Access z/VM CMS disks from Linux . . . 136
7.10.1 Mount a CMS disk using cmsfs-fuse . . . 136
7.11 NFS mounting the LNXADMIN SFS directory from Linux . . . 137
7.12 Customizing default system commands using update-alternatives . . . 137
7.13 Other resources and “recipes” . . . 138

Part 3. Appendixes . . . 139

Appendix A. References, cheat sheets, and blank worksheets . . . 141
IBM z/VM components and related products . . . 142
Cheat sheet for the Linux vi editor . . . 142
Blank planning worksheets . . . 142

Appendix B. Additional material . . . 143
Locating the web material . . . 143
Using the web material . . . 143
Linux code: The Ubuntu clone script . . . 143

Appendix C. Prerequisite network connectivity . . . 157
Connectivity . . . 158
Mirrors . . . 158
Additional information . . . 158
Supplemental information regarding global mirrors . . . 159

Related publications and information 161
IBM Redbooks 161
Other publications 161
Online resources 162
Help from IBM 163
IBM wants your input 163

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

DB2®	IBM z™	WebSphere®
ECKD™	IBM z Systems™	z Systems™
Global Business Services®	Insight™	z/OS®
Global Technology Services®	Redbooks®	z/VM®
IBM®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Inc., and Inc. device are trademarks or registered trademarks of Kenexa, an IBM Company.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get personalized notifications of new content
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android

Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication is Volume 4 of a series of books entitled *The Virtualization Cookbook for IBM z Systems*. The other volumes in the series are:

- *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147
- *The Virtualization Cookbook for IBM z Systems Volume 2: Red Hat Enterprise Linux 7.1 Servers*, SG24-8303
- *The Virtualization Cookbook for IBM z Systems Volume 3: SUSE Linux Enterprise Server 12*, SG24-8890

It is advised that you start with Volume 1 of this series, because the IBM z/VM® Hypervisor is the foundation for installing Linux on IBM z™ Systems.

Volume 1 begins with an introduction, describes planning, and then describes z/VM installation into a two-node single system image (SSI) cluster, configuration, hardening, automation, and servicing. It adopts a cookbook format that provides a concise, repeatable set of procedures for installing and configuring z/VM using the SSI clustering feature and includes a chapter on monitoring z/VM system and Linux guests.

Volumes 2, 3, and 4 assume that you have completed all of the steps that are provided in Volume 1. From that common foundation, these volumes describe how to roll your own Linux virtual servers on IBM z Systems™ hardware under IBM z/VM. The cookbook format continues with installing and customizing Linux.

Volume 4 focuses on Ubuntu 16.04 LTS Server Edition. It consists of the following main chapters:

- Chapter 1, “Install Ubuntu Server 16.04 on LNXADMIN” on page 3. This chapter covers the installation and configuration of the LNXADMIN virtual server, which will function as the Ubuntu Server 16.04 administrative control point for your z/VM single system image (VMSSI) cluster. It includes the following topics:
- Chapter 2, “Automating installation” on page 35. This chapter describes several different methodologies for quickly provisioning additional virtual servers, with an emphasis on automated installation. Anyone who has used Linux with some regularity knows that there are usually many methods of achieving the same result. The automated installation of systems is no exception.
- Chapter 3, “Servicing Linux: Updates and patching” on page 49. This chapter describes the use of the package manager to perform routine service for Ubuntu Linux virtual servers. Essential tasks, such as installation, upgrades, and removals of software, are covered by using examples.
- Chapter 4, “Working with disks” on page 57. This chapter details the processes for working with the two different types of disks available in a z/VM environment, direct access storage device (DASD) extended count key data (IBM ECKD™), and Fibre Channel Protocol/Small Computer System Interface (FCP/SCSI). It concentrates on the tasks that are commonly performed on Linux.
- Chapter 5, “Workload containers and service orchestration” on page 77. This chapter introduces the concept of running workloads in containers, using Docker. Additionally, we cover service orchestration using JuJu. It includes the following topics:
- Chapter 6, “Working with systemd” on page 95. This chapter describes how to work with an Ubuntu Server 16.04 system that uses systemd. On modern Linux installations,

systemd plays a major role. Understanding the concepts and knowing the utilities that are provided by systemd is key to any Linux administrator.

- Chapter 7, “Miscellaneous recipes” on page 113. This chapter contains information that falls under the miscellaneous category. These are topics that help make administration easier, save time, increase functionality, or add capabilities to your systems.

Important information before you begin

It is assumed that you have read and completed all the steps in Volume 1 of this series of books (*The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147). If you have not, stop and do so before proceeding with this volume. Doing so will prevent you from experiencing difficulties following the steps that are provided in this volume.

Your first Ubuntu Linux virtual server requires access to the public Internet. If you have a network infrastructure, such as a firewall or forward proxy that might prevent access to the public Internet, assistance from your network administrator is necessary. Review Appendix C, “Prerequisite network connectivity” on page 157 for complete details.

Architectural decisions

In addition to the decisions made in Volume 1 of this series of books, during the planning and authoring of this current volume, more decision making was required. For many components and elements, options had to be listed and considered. The choices made then resulted in the suggestions and examples shown throughout this volume.

General criteria

The following general criteria were used to evaluate options, layouts, nomenclature, and other fundamentals. These criteria are listed in alphabetical order:

- Clarity, efficiency, practicality, and sensibility
- Community-accepted, de-facto standards and leading practices
- Continuancy, resiliency, and recovery
- Customer environment experience
- Flexibility and scalability
- IBM recommendations (ATS/WSC, Development, Lab Services)
- Industry-accepted standards and standard practices
- Product and technology standards (Request for Comments (RFC), Linux, World Wide Web Consortium (W3C), International Organization for Standardization (ISO), Internet Assigned Numbers Authority (IANA), and so on)
- Production environment experience
- Reliability and consistency
- Simplicity versus complexity

Food for thought

The “recipes” published in this cookbook are meant to teach your brain *how to cook*: Proper techniques, methodologies, safety, and what order things must happen. When you know the *how*, you can more readily understand the *why*. From there, you can choose to tweak and personalize your “menu.” For example, meat, dairy, vegetarian, or vegan choices are up to you.

To put things another way, it is important to realize that the recommendations and examples that are presented to you on the pages of this book are purposefully intended to spark thoughtfulness and meaningful awareness of the entire process. In situations where required, things can and should be adapted to suit your specific needs.

Specific examples

The following list provides specific examples:

- Linux disk planning and layouts
  - Recovering a Linux system using Logical Volume Manager (LVM) for the root file system is a painful ordeal, regardless of which platform is being used.
  - Creating an LVM *system* volume group (VG) gives you the flexibility to make logical volumes (LVs) for `/usr`, `/opt`, and `/srv`. The LVs can be tuned and reworked as needed.
  - A Linux root file system that fills entirely to capacity results in a fully down system. The `/var` directory, or perhaps just `/var/log`, should be an LVM LV that can be expanded.
  - Chargeback models might require you to put data into different LVM groups based on purpose.
  - Massive databases for Oracle or IBM DB2® are good fits for FCP/SCSI where logical units (LUNs) can be sized to match and mounted.
  - Creating an LVM user VG might help make carrying forward LVs for `/home` and `/usr/local` easier in environments where there is high churn, such as Cross-Application Performance Reliability Stress (XPRS) or quality assurance (QA) testing.
- Graphical user interfaces (GUIs)
  - Can be resource-intensive
  - Quickly create security exposures when improperly configured
  - Of limited use in a virtualized headless environment
  - Typically can be needed during installation of certain types of software, and then never or seldom used again

Conventions

The following conventions are used in this book.

Font conventions that are used in this cookbook series

The following font conventions are used in this book:

Monospace and bold	Commands entered by the user on the command line
`monospace`	Linux file, directories, and commands
MONOSPACE CAPITALS	z/VM files, virtual machine and minidisk names, and commands
Monospace bold italics	Values used to test this book, such as Transmission Control Protocol/Internet Protocol (TCP/IP) addresses. This font convention is used to signify that you should replace the ***example value*** with the correct value for your system or enterprise.
Monospace reverse	Keys and key combinations that are used to invoke functions or respond.

Command conventions that are used in this cookbook series

The following command conventions are used in this book:

- z/VM commands are prefixed with **===>** (three Equal signs (=) followed by a Greater than symbol (>))
- z/VM **XEDIT** subcommands are prefixed with **====>** (four Equal signs (=) followed by a Greater than symbol (>)).
- Linux commands that are running as root have a Number sign (#) prefix.
- Linux commands that are running as non-root usually have a Dollar sign ($) prefix.

Operating system releases used

The following releases of operating systems were used in the writing of this book:

z/VM Hypervisor	6.3.0 general availability (GA) code plus Service Level 1601, March 2016
Ubuntu Linux	16.04 Long Term Support (LTS) Xenial Xerus Server Edition GM code, 2016

Authors

This book was produced by a team of specialists from around the world, working at the IBM International Technical Support Organization (ITSO), Poughkeepsie Center.

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management, including software development, project leadership, and strategic planning. Her areas of expertise include Linux on z Systems and database management technologies. Lydia is a Certified IT Specialist with an MBA in Technology Management, and has been employed by IBM for over 25 years in various technology areas.

Pedro Acosta is a Client Technical Specialist on the IBM Washington Systems Center Linux Solutions Team (former Advanced Technical Skills (ATS) Gaithersburg). He serves as a technical consultant to IBM clients, IBM sales teams, and IBM Business Partners. Pedro is an IBM Certified IT Specialist with a degree in electrical engineering. Pedro's background spans circuit design, teaching public school mathematics, and working with IBM mainframe customers and building relationships with clients.

Fred Bader is a Senior IT Specialist and member of the z Systems Applied Technologies team at the IBM Washington Systems Center (former ATS Gaithersburg). Fred has more than 40 years of experience with IBM mainframe technologies, and his areas of expertise include systems project planning, IBM z/OS®, z/VM, and Linux on z Systems. His current focus is demonstrating the advantages of z Systems with new technology test drives and proof of concept projects using z/VM, Linux on z Systems, z/OS, IBM DB2, and IBM WebSphere® based applications. Fred has a BS in Accounting and an MS in Management Information Systems, and has been an IBM employee for over 25 years in various technology roles.

Paul Novak is a Senior Certified IT Specialist working for the IBM Washington Systems Center z/VM and Linux Solutions team in Endicott, New York (former ATS Endicott). Paul has experience in field services, end-user support, software development, enterprise hosting, and enterprise architecture. His areas of expertise are z/VM, Linux, web middleware, and web identity and access management (IAM) solutions. Paul holds a BSBA in Management Information Systems, and is a multi-generation IBM employee with more than 20 years of Linux and Open Source technology experience.

With gratitude

The authors of this book extend our thanks to the following people for their contributions and assistance:

Pete Bertolozzi, Ann Lund
International Technical Support Organization, Poughkeepsie Center

A special thanks to Bob and Dave, without whom these books truly would not be possible.

Bob Haimowitz, Dave Bennin
IBM Global Business Services®, Development Support Team

Alan Altmark, Bill Bitner, Brian Hugenbruch, Emily Kate Hugenbruch, Patty Rando, Tung-Sing Chong, John Franciscovich, Yanique Moffitt, Dan FitzGerald
IBM Systems, Development and Lab Services, Endicott

Bill VanDuzer
IBM Global Technology Services®, Endicott

Mike Cox, Bruce Hayden, Richard Lewis, Ernie Horn, Eduardo Oliveira, Ivan Dobos
IBM Sales and Distribution, Washington Systems Center

Hendrik Brückner, Elisabeth Puritscher, Maria Eisenhaendler, Heinz-Werner Seeck, Steffen Maier, Kate Teplova, Viktor Mihajolvski, Wilhelm Mild, Carl Mayer
IBM Systems, Research & Development Lab, Böblingen

Frank Collura, Clay Harshberger, Heidi Lagares-Greenblatt
IBM Systems, Product Engineering, and New Technology Introduction (NTI), Poughkeepsie

Christian Ehrhardt, Frank Heimes
Canonical, Inc.®

A special thanks to Michael MacIsaac for the original inception of this cookbook and for his dedicated efforts in continually moving the cookbook forward over the years.

Thanks to many others in IBM Endicott and Poughkeepsie and to the many who answered questions on the `Linux-390` and `IBMVM` list servers (see “Online resources” on page 162).

Thanks to the authors of the previous editions of this book:

- Authors of the previous IBM Redbooks publication edition, *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147, last updated 22 February 2011: Lydia Parziale, Marian Gasparovic, Berthold Gunreben, Michael MacIsaac, Filipe Miranda, and Daniel Ruutz
- Authors of the previous IBM Redbooks publication edition, *z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES 11 SP1*, SG24-7931, last updated 22 February 2011: Marian Gasparovic, Michael MacIsaac
- Authors of the previous IBM Redbooks publication edition, *z/VM and Linux on IBM System z: The Virtualization Cookbook for Red Hat Enterprise Linux 6.0*, SG24-7932, last updated 18 February 2011: Brad Hinson, Michael MacIsaac

Now you can become a published author, too

Here’s an opportunity to spotlight your skills, grow your career, and become a published author, all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run two - six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online:

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us.

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments in an email:

  redbooks@us.ibm.com

- Mail your comments:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  ibm.com/redbooks/redbooks.nsf/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

  ibm.com/redbooks/rss.html

Part 1

Ubuntu Server 16.04

“Free software is part of a broader phenomenon, which is a shift toward recognizing the value of shared work. Historically, shared stuff had a very bad name. The reputation was that people always abused shared things, and in the physical world, something that is shared and abused becomes worthless. In the digital world, I think we have the inverse effect, where something that is shared can become more valuable than something that is closely held, as long as it is both shared and contributed to by everybody who is sharing in it.”

— Mark Shuttleworth

This part of the book focuses on the installation, configuration, and servicing of Ubuntu Server 16.04 LTS (Xenial Xerus). Additional topics that are covered include an automated approach to installation, common tools and utilities, and ongoing maintenance.

This part includes the following chapters:

- Chapter 1, “Install Ubuntu Server 16.04 on LNXADMIN” on page 3
- Chapter 2, “Automating installation” on page 35
- Chapter 3, “Servicing Linux: Updates and patching” on page 49

At a glance

Note the following facts about Ubuntu:

- The ancient African word *Ubuntu*, pronounced *OŎ'BOOŎNTOŌ*, means both "*humanity to others"* and *"I am what I am because of who we all are"*. The Ubuntu Linux distribution aims to foster this community-centric philosophy in the world of computing.
- Unlike most other commercial Linux distributions that came before it, Ubuntu chose not to fork into commercial and community versions.
- Ubuntu has a new interim release every six months. Every fourth release is *LTS*. LTS stands for *Long Term Support*, which is five years for Ubuntu Server.
- Hardware enablement updates bring forward certain updates from the interim releases into the actively supported LTS releases.
- Traditionally, releases have been made public in either the fourth or tenth month of a calendar, April and October. This makes identifying the "birth date" of a release quite simple. Ubuntu 16.04 was released in the fourth month of the year 2016. Ubuntu 15.10 was released in October of 2015.

1

Install Ubuntu Server 16.04 on LNXADMIN

"We must never think that what we have today will satisfy the demand ten years from now... Analyze the past, consider the present, and visualize the future."

— Thomas J. Watson, Sr.

Note: The `LNXADMIN` virtual server that is referenced throughout Part 1 of this volume covering Ubuntu Server 16.04 was created using directions that are provided in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

This chapter covers the installation and configuration of the `LNXADMIN` virtual server, which will function as the Ubuntu Server 16.04 administrative control point for your z/VM single system image (VMSSI) cluster. It includes the following topics:

- Prepare the Ubuntu bootstrap files
- Install Ubuntu onto the Linux administration system
- Configure the Ubuntu administration system

The `LNXADMIN` virtual server is a multiconfiguration virtual machine (MCVM), or *IDENTITY*. Compared to a single-configuration virtual machine (SCVM), or *USER*, an MCVM can be logged on to more than one z/VM system in a VMSSI cluster simultaneously. Therefore, migration of an MCVM among cluster members is not possible.

From the perspective of running Ubuntu Linux in the environment that is created by this book, the `LNXADMIN` virtual server provides the following administrative roles:

- The *repository mirror* server is an HTTP-based *local mirror* of the *Ubuntu ports primary installation repository* that is hosted by Canonical. This local mirror serves as the installation and package maintenance source for subsequent Ubuntu Linux virtual servers.
- The *kickstart repository* serves files that are required for automated installations.
- *z/VM Linux Terminal Services* provide quick and easy access to Linux virtual servers through the z/VM Inter-user Communications Vehicle (IUCV). Terminal services make management of the Ubuntu virtual servers in your z/VM environment much easier by

providing a centralized point of access that does not use Transmission Control Protocol/Internet Protocol (TCP/IP).

1.1 Prepare the Ubuntu bootstrap files

To perform the initial program load (IPL), or boot, of an Ubuntu installation system, several bootstrap files, described in Table 1-1, need to be copied to your `LNX` SFS file pool on z/VM.

Table 1-1 Ubuntu bootstrap files and their purposes

File name	Purpose
`UBTU1604.EXEC`	Set-up environment, spool kernel, and parameters to virtual punch
`UBTU1604.KERNEL`	Ubuntu Server 16.04 s390x kernel
`UBTU1604.PARMS`	Default Ubuntu Server 16.04 s390x boot parameters
`UBTU1604.INITRD`	Ubuntu Server 16.04 s390x initial ramdisk

All the files shown in Table 1-1 are included within the compressed `.tar` file that is associated with this cookbook series, which is covered in Chapter B, “Additional material” on page 143 in both this Ubuntu Volume and Volume 1.

You need to download this file from the Internet and decompress it on a local File Transfer Protocol (FTP) server, as described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

In the environment that is used to author these books, the files specific to Ubuntu expands under the following path on the FTP server:

`/ftp/zvm/cookbook/lnxadmin/ubuntu/16.04`

To prepare the Ubuntu bootstrap files:

1. Use your 3270 emulator to log on as `LNXADMIN` on the z/VM member of your choice. The file `PROFILE EXEC` is run and makes the TCP/IP tools, such as NETSTAT and FTP, available. A virtual NIC is created at virtual addresses `600 - 602`, in addition to two virtual disks for swap spaces at virtual addresses `300` and `301`, as shown in Example 1-1.

   Example 1-1 Log on as LNXADMIN

```
LOGON LNXADMIN
00: z/VM Version 6 Release 3.0, Service Level 1601 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0002 RDR,   NO PRT,   NO PUN
00: LOGON AT 09:31:57 EDT MONDAY 2016-05-16
00: Command complete
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2016-05-09 09:04
DMSACP723I T (592) R/O
SWPSWA0012I DIAG swap disk defined at virtual address 0300 with
SWPSWA0012I 64988 4K pages of swap space.
SWPSWA0012I DIAG swap disk defined at virtual address 0301 with
SWPSWA0012I 129980 4K pages of swap space.
```

2. Answer no (N) to the question asking you to IPL Linux from DASD device 100:

```
DO YOU WANT TO IPL LINUX FROM MINIDISK 100? Y/N
n
```

3. Copy the bootstrap files from the local FTP server into the LNXADMIN.UBUNTU directory in the LNX SFS File Pool, as shown in Example 1-2.

Example 1-2 Copy bootstrap files

```
===> ftp 9.60.7.34
   VM TCP/IP FTP Level 630
   Connecting to 9.60.7.34, port 21
   USER (identify yourself to the host):
   ftpuser1
   >>>USER ftpuser1
   331 Please specify the password.
   Password:
   ftppassword
   >>>PASS ********
   230 Login successful.
   Command:
   lcd LNX:LNXADMIN.UBUNTU
   cd /ftp/zvm/cookbook/lnxadmin/ubuntu/16.04
   ascii
   get ubtu1604.parmfile
   get ubtu1604.exec
   locsite fix 80
   binary
   get ubtu1604.kernel
   get ubtu1604.initrd
   quit
```

4. Use the FILELIST command to verify the following information, as shown in Figure 1-1:

```
===> filelist ubutu1604 * a
```

   – The KERNEL and INITRD (RAM disk) must use an 80-column fixed record format (FIXED-80), meaning that the format column shows F, and the *longest record length* (LRECL) column shows a value of 80.
   – The EXEC file is not used by a virtual punch, so there is no concern if this uses a variable length format and is longer than 80 columns.

```
LNXADMIN FILELIST A0  V 169  Trunc=169 Size=4 Line=2 Col=1 Alt=0
Directory = LNX:LNXADMIN.UBUNTU
Cmd Filename Filetype Fm Format Lrecl Records Blocks   Date       Time
    UBTU1604 EXEC     A1 V         69      38      1 2016-05-11 11:11:56
    UBTU1604 PARMFILE A1 F         80       1      1 2016-05-12 15:12:59
    UBTU1604 KERNEL   A1 F         80   46093    886 2016-05-11 12:03:29
    UBTU1604 INITRD   A1 F         80  145477   2842 2016-05-11 12:01:37
```

*Figure 1-1 Listing of UBTU1604 * for file mode A*

After ensuring the proper record formatting, press F3 to quit FILELIST.

5. Verify the contents of the **UBUNTU EXEC** to match the previous list of files. The first several lines are shown in Figure 1-2.

   ===> **xedit UBUNTU EXEC A**

```
/* LOAD EXEC FOR UBUNTU 16.04 VIRT SRVRS -- MOD 2016-05-18  PACOSTA   */
/* LOADS UBUNTU SERVER 16.04 FOR IBM Z SYSTEMS FILES INTO VM READER   */
/**********************************************************************/
'CP SPOOL CONS TO VMLOGS START NAME 'USERID()' CONSLOG' /* CONSLOG ON */
'CP MSG OPERATOR LOGON 'USERID()' TO BUILD UBTU1604 VIRT. SERVER'
SAY 'PUNCHING (LOADING) UBUNTU SERVER 16.04 FILES INTO VIRT. READER...'
'IDENTIFY (ISODATE'                              /* IDENTIFY VIRT SRVR */
'CP CLOSE RDR'                             /* CLOSE THE VIRTUAL READER */
'PURGE RDR ALL'                            /* PURGE ALL V READER FILES */
'SPOOL PUNCH * RDR'                        /* SPOOL PUNCH TO V READER */
'RELEASE 592 (DETACH)'                     /* RELEASE TCP/IP TOOLS DISK */
SAY 'PUNCHING UBUNTU SERVER 16.04 KERNEL...'
'PUNCH UBTU1604 KERNEL   * (NOHEADER'
SAY 'PUNCHING UBUNTU SERVER 16.04 PARAMETERS FILE...'
'PUNCH UBTU1604 PARMFILE * (NOHEADER'
SAY 'PUNCHING UBUNTU SERVER 16.04 INITIAL RAMDISK...'
'PUNCH UBTU1604 INITRD   * (NOHEADER'
'CHANGE RDR ALL KEEP NOHOLD'
'CP IPL 00C'
EXIT
...
```

   Figure 1-2 Excerpt of UBUNTU EXEC contents

6. Quit by typing **FILE** on the input line.

You are now ready to start the installation.

1.2 Install Ubuntu onto the Linux administration system

The Debian installer on s390x runs in netboot mode and has the following stages:

1. An initial stage where the network is configured, usually done on the 3270 z/VM console, or Operating Systems Messages when in LPAR mode.
2. A second stage where the remaining server configuration is completed, done via an Secure Shell (SSH) session.

The process of installing the LNXADMIN virtual server by completing both of these stages is covered in this section. You install Linux onto the LNXADMIN virtual server z/VM single system image (SSI) node. If you have not logged off, you should still be logged in as the LNXADMIN virtual server.

1.2.1 Stage 1: The initial network configuration

Complete the following Stage 1 steps for the initial network configuration:

1. Define the memory size (storage) to 1535 MB (1.5 GB) with the **DEFINE STORAGE** command:

```
===> define storage 1535M
00: STORAGE = 1535M
00: Storage cleared - system reset.
```

2. Restart CMS, as shown in Example 1-3.

```
===> ipl cms
```

Example 1-3 Restart of CMS

```
z/VM V6.3.0    2016-05-09 09:04

ENTER

LNXADMIN AT ITSOZVM4 VIA RSCS 09:44:17 2016-05-16 EDT MONDAY
DMSACP723I T (592) R/O
SWPSWA0012I DIAG swap disk defined at virtual address 0300 with
SWPSWA0012I 64988 4K pages of swap space.
SWPSWA0012I DIAG swap disk defined at virtual address 0301 with
SWPSWA0012I 129980 4K pages of swap space.
DO YOU WANT TO IPL LINUX FROM MINIDISK 100? Y/N

N
```

3. Answer no (N) to the question asking you to start Linux from DASD device 100:

```
DO YOU WANT TO IPL LINUX FROM MINIDISK 100? Y/N

n
```

4. Verify the increased memory size with the `QUERY VIRTUAL STORAGE` command:

```
===> query virtual storage
00: STORAGE = 1535M
```

5. Verify that the disks you are going to install Ubuntu on are available to this virtual server with the `QUERY VIRTUAL DASD` command, as shown in Example 1-4.

Example 1-4 Output from QUERY VIRTUAL DASD

```
===> q v dasd
00: DASD 0100 3390 ZA0L24 R/W      10016 CYL ON DASD  382F SUBCHANNEL = 0000
00: DASD 0190 3390 ZA4RS1 R/O        214 CYL ON DASD  3A31 SUBCHANNEL = 0006
00: DASD 019D 3390 ZA4RS1 R/O        292 CYL ON DASD  3A31 SUBCHANNEL = 0007
00: DASD 019E 3390 ZA4RS1 R/O        500 CYL ON DASD  3A31 SUBCHANNEL = 0008
00: DASD 0200 3390 ZA0L25 R/W      10016 CYL ON DASD  3934 SUBCHANNEL = 0001
00: DASD 0300 9336 (VDSK) R/W     524288 BLK ON DASD  VDSK SUBCHANNEL = 000D
00: DASD 0301 9336 (VDSK) R/W    1048576 BLK ON DASD  VDSK SUBCHANNEL = 000E
```

6. Run the **UBUNTU EXEC** to purge the reader, punch the bootstrap files, and perform an initial program load (IPL) from the reader. Last, you see the Linux RAMdisk being loaded into memory, as shown in Figure 1-3.

```
PUNCHING (LOADING) UBUNTU SERVER 16.04 FILES INTO VIRT. READER...
LNXADMIN AT ITSOZVM4 VIA RSCS 09:44:17 2016-05-16 EDT MONDAY
00: 0000004 FILES PURGED
00: 592 T DETACHED
PUNCHING UBUNTU SERVER 16.04 KERNEL...
00: RDR FILE 0016 SENT FROM LNXADMIN PUN WAS 0016 RECS 046K CPY  001 A
NOHOLD NO KEEP
PUNCHING UBUNTU SERVER 16.04 PARAMETERS FILE...
00: RDR FILE 0017 SENT FROM LNXADMIN PUN WAS 0017 RECS 0001 CPY  001 A
NOHOLD NO KEEP
PUNCHING UBUNTU SERVER 16.04 INITIAL RAMDISK...
00: RDR FILE 0018 SENT FROM LNXADMIN PUN WAS 0018 RECS 145K CPY  001 A
NOHOLD NO KEEP
00: 0000003 FILES CHANGED
00: 0000003 FILES CHANGED
00: Uncompressing Linux...
00: Ok, booting the kernel.
00:
[    0.394217] Initializing cgroup subsys cpuset
[    0.394220] Initializing cgroup subsys cpu
[    0.394221] Initializing cgroup subsys cpuacct
[    0.394225] Linux version 4.4.0-21-generic (buildd@z13-019) (gcc version
5.3.1 20160413 (Ubuntu 5.3.1-14ubuntu2) ) #37-Ubuntu SMP Mon Apr 18 18:31:26
UTC 2016 (Ubuntu 4.4.0-21.37-generic 4.4.6)
...
```

Figure 1-3 Output from the UBTU1604 EXEC

7. At the *Configure the network device* menu, choose option #2 for queued direct input/output (QDIO) Ethernet (`qeth: OSA-Express in QDIO mode / Hipersockets`), as shown in Figure 1-4.

```
Configure the network device
----------------------------

Please choose the type of your primary network interface that you will need
for

installing the Debian system (via NFS or HTTP). Only the listed devices are
supported.
Network device type:
  1: ctc: Channel to Channel (CTC) or ESCON connection,
  2: qeth: OSA-Express in QDIO mode / HiperSockets,
  3: iucv: Inter-User Communication Vehicle - available for VM guests only,
  4: virtio: KVM VirtIO,
Prompt: '?' for help>

> 2
```

Figure 1-4 Configure the network device menu

8. Next, choose option #1 to select the Open Systems Adapter (OSA) triplets for use. In the environment that is used to author this book, this value is ***0.0.0600-0.0.0601-0.0.0602***, as shown in Example 1-5.

Example 1-5 Sample OSA-Express QDIO triplet selection

```
Please select the OSA-Express QDIO / HiperSockets device.
Device:
  1: 0.0.0600-0.0.0601-0.0.0602 [*],
Prompt: '?' for help, default=1>

> 1
```

9. Next, select option #1 to use the QDIO device in *layer2* mode, then input the number zero (**0**) for the *relative port*, as shown in Figure 1-5.

```
By default OSA-Express cards use layer3 mode. In that mode LLC headers are
removed from incoming IPv4 packets. Using the card in layer2 mode will make
it keep the MAC addresses of IPv4 packets.
Use this device in layer2 mode?
  1: Yes [*]  2: No
Prompt: '?' for help, default=1>

> 1

Please enter a relative port for this connection.
Port:
Prompt: '?' for help, default=0>

> 0
```

Figure 1-5 QDIO device mode selection and relative port

Note: You should expect that operating an OSA-Express card in layer 3 (Media Access Control (MAC)) mode will become deprecated as the world moves toward IPv6, virtual local area networks (LANs), and more advanced virtual switch capabilities. The use of layer 3 (MAC) mode is not recommended for any portion of the z/VM and Linux environment.

10. The installer gives you the option to detect your network settings automatically using DHCP. Select option #2 and proceed by entering network settings manually, as shown in Example 1-6.

Example 1-6 Enter the network settings manually

```
Configure the network
---------------------

Networking can be configured either by entering all the information manually,
or by using DHCP (or a variety of IPv6-specific methods) to detect network
settings automatically. If you choose to use autoconfiguration and the
installer is unable to get a working configuration from the network, you will
be given the opportunity to configure the network manually.
Auto-configure networking?
  1: Yes [*]  2: No
```

```
Prompt: '?' for help, default=1>

> 2
```

Important: A configuration error, service interruption, or outage to DHCP can easily become a trigger point for a cascading failure across the environment with catastrophic results.

11. Enter the unique IP address for the `LNXADMIN` virtual server. Example 1-7 uses the value of `9.60.7.36`.

Example 1-7 Input of IP address value for LNXADMIN

```
The IP address is unique to your computer and may be:

 * four numbers separated by periods (IPv4);
 * blocks of hexadecimal characters separated by colons (IPv6).

You can also optionally append a CIDR netmask (such as "/24").

If you don't know what to use here, consult your network administrator.
IP address:
Prompt: '?' for help>

> 9.60.7.36
```

12. Enter the netmask address and then the gateway router for this network segment. Example 1-8 uses the values of `255.255.240.0` and `9.60.7.1`.

Example 1-8 Entering the netmask address and gateway router

```
The netmask is used to determine which machines are local to your network.
Consult your network administrator if you do not know the value.  The netmask
should be entered as four numbers separated by periods.
Netmask:
Prompt: '?' for help, default=255.255.255.0>

> 255.255.240.0

The gateway is an IP address (four numbers separated by periods) that indicates

the gateway router, also known as the default router.  All traffic that goes
outside your LAN (for instance, to the Internet) is sent through this router.

In rare circumstances, you may have no router; in that case, you can leave this
blank.  If you don't know the proper answer to this question, consult your
network administrator.
Gateway:
Prompt: '?' for help, default=9.60.0.1>

> 9.60.7.1
```

13. Enter one or more IP addresses of the *name servers* that are used to resolve host names on the network, separated by a space, such as `9.12.6.6`, `9.12.6.7`, and `9.60.70.80`, as shown in Example 1-9.

Example 1-9 Example name servers

```
The name servers are used to look up host names on the network. Please enter
the IP addresses (not host names) of up to 3 name servers, separated by spaces.

Do not use commas. The first name server in the list will be the first to be
queried. If you don't want to use any name server, just leave this field blank.

Name server addresses:

Prompt: '?' for help, default=9.60.4.1>

> 9.12.6.6 9.12.6.7 9.60.70.80
```

14. Enter the host name that is used to identify this system, for example **`lnxadmin`**, as shown in Example 1-10.

Example 1-10 Host name used to identify this system

```
Detecting link on enc600; please wait...
Please enter the hostname for this system.

The hostname is a single word that identifies your system to the network. If
you don't know what your hostname should be, consult your network
administrator. If you are setting up your own home network, you can make
something up here.
Hostname:
Prompt: '?' for help, default=ubuntu>

> lnxadmin
```

15. Enter the *domain name* that this host name is under, such as `itso.ibm.com`, as shown in Example 1-11.

Example 1-11 Example domain name

```
The domain name is the part of your Internet address to the right of your host
name.  It is often something that ends in .com, .net, .edu, or .org.  If you
are setting up a home network, you can make something up, but make sure you use
the same domain name on all your computers.
Domain name:
Prompt: '?' for help>

> itso.ibm.com
```

16. Choose a password to use for remote access to the Debian installer (as shown in Example 1-12). You are asked to input this password twice. The password can be any value that you want. As stated in the informational text, this password is used only during the installation and then is permanently discarded afterward.

Example 1-12 Choose a password

```
Generating SSH host key
Continue installation remotely using SSH
```

```
---------------------------------------

You need to set a password for remote access to the Debian installer. A
malicious or unqualified user with access to the installer can have disastrous

results, so you should take care to choose a password that is not easy to
guess. It should not be a word found in the dictionary, or a word that could be

easily associated with you, like your middle name.

This password is used only by the Debian installer, and will be discarded once
you finish the installation.
Remote installation password:

> XXXXXX

Please enter the same remote installation password again to verify that you
have typed it correctly.
Re-enter password to verify:

> XXXXXX
```

17. Leave the z/VM console open in the background.
18. Start an SSH session to the installation system, and log in as `installer` by using the password that you defined in the previous step, as shown in Example 1-13.

Example 1-13 Log in as installer by using the new password that you set

```
Start SSH

To continue the installation, please use an SSH client to connect to the IP
address 9.60.7.36 fe80::50:dff:fe00:4 and log in as the "installer" user. For
example:

ssh installer@9.60.7.36

The fingerprint of this SSH server's host key is:
SHA256:nzgQbJVXbtNEdQXE82Jg8yG/Ae++JcWLoyDy/U8XvnY

Please check this carefully against the fingerprint reported by your SSH
client.
[Press enter to continue]
```

We are now ready to begin part 2 of the Ubuntu 16.04 installation, where we select the parameters for our base system and define a partitioning scheme to fit our needs.

1.2.2 Stage 2: The Ubuntu 16.04 installation

The Ubuntu installer is self-explanatory and easy to navigate. Make note of the functions provided by the following keys:

- Use the ↕ (UP ↑ and DOWN ↓ arrows) to navigate among the different menus and options.
- The SPACEBAR toggles to select or clear an option.
- The TAB key jumps between input fields.
- The ENTER key submits the selection.

After you establish an SSH session, respond to each prompt using values appropriate for your environment. In the following steps, we use the example values shown in ***monospace italic bold***. You need to replace these example values:

1. Select *Start installer* and press **ENTER**.
2. Scroll to select the *language* of your choice for the installation:

   English
3. Select your *location*:
   a. Continent

      North America

   b. Country, territory, or area

      United States
4. Scroll all the way to the top. For the Ubuntu archive mirror, choose *enter information manually*.
5. For the *hostname* of the mirror that will be used to get the rest of the Ubuntu packages, input the value **ports.ubuntu.com** or the hostname of a local mirror (see section 1.3.3, “Build a local Ubuntu mirror with apt-mirror on the LNXADMIN virtual server” on page 25).
6. Input a single forward slash to indicate use of the server root directory on `ports.ubuntu.com`:

   /
7. If you will need to use a forward proxy to reach `ports.ubuntu.com`, input this information. If a forward proxy is not in use, leave this field blank:

   http://socks5.sdc-northeast.ibm.com:1080

   Note: It is uncommon, and therefore unlikely, that you might encounter a forward proxy on a network segment used for servers. Consult your *network administrator* for more information.
8. Enter the *Full Name* of the account owner that will be used to log in to the system rather than root:

   linuxonz support
9. Choose a *username* for the account that was defined in the previous step:

   support
10. Choose a *password* for this user. Remember this password, or store it in a secure location, because *this will be the only active user account* after the installation is complete. Provide the same password again for verification.
11. Encryption with encryptfs is not yet supported. As such, respond `no` to the prompt about *encrypting your home directory*:

    no
12. The installer now selects a *timezone* based on your physical location. Select `yes` if it is correct, or `no` to change it:

    America/New_York
13. Activate the DASD device that will be used to install your base Ubuntu server:

    0.0.0100
    ENTER

14. Respond `yes` when prompted whether to *format this device*. This takes several minutes to complete:

    yes

15. Select *Finish* to complete the DASD configuration.

1.2.3 Partition disks for the LNXADMIN virtual server

The `LNXADMIN` virtual server serves as the primary repository for future installations and system updates. The partitioning that is selected for this book serves as a starting point from which you can expand in the future. Table 1-2 details the partitioning scheme that we are implementing.

Table 1-2 Partitioning scheme volume and mount definitions

Mount Point	Volume Group	Logical Volume	Size
/boot			200 MB
/			2.5 GB
/var	vgsystem	var_lv	1 GB
/tmp	vgsystem	tmp_lv	800 MB

To complete the partitioning of disks:

1. Select **Manual** as the partitioning method.
2. Select **FREE SPACE** under the DASD device that you want to use. *DASD 0.0.0100* is used as an example in Figure 1-6.

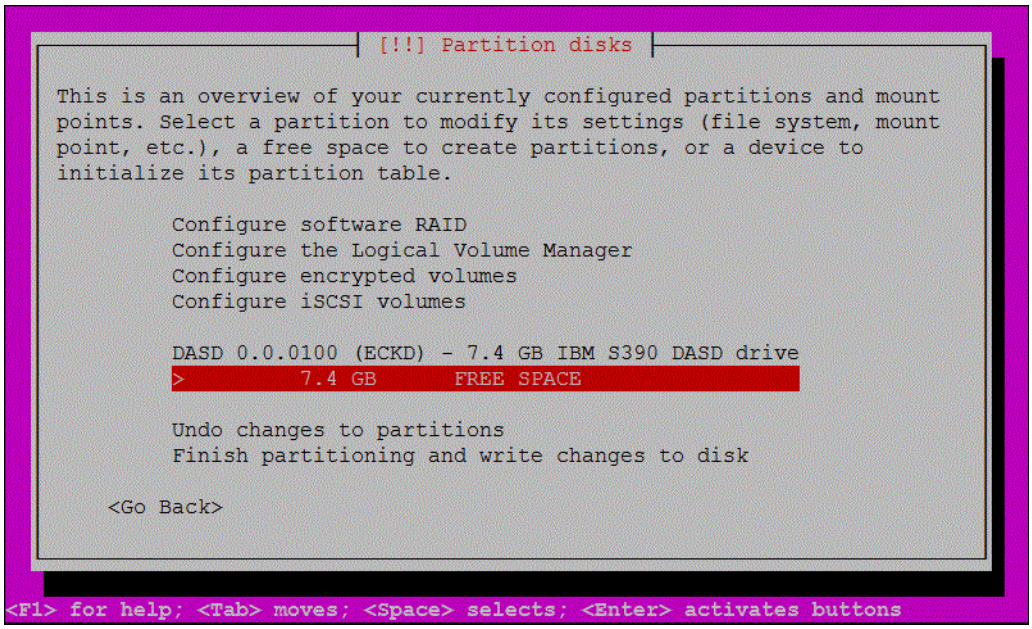


Figure 1-6 Partition disks menu highlighting the available free space

3. A new menu displays, showing options about how to use this free space. Select **Create a new partition**.
4. Enter the partition size. In this example environment, we used a value of ***200 MB***.

5. Choose **Beginning** as the location for the new partition.
6. Select **Use as: xfs journaling file system** by selecting it from the Partition disks menu. You might also choose to use ext2 or ext3 for the `/boot` partition, because it should have a low volume of ongoing change compared to most other system partitions.
7. Select **/boot** as the mount point.
8. Select **Done setting up the partition**.
9. Repeat steps 1-8 to create a 2.5 GB partition and with Forward slash (/) as the mount point. As an alternative to XFS file system, you might want to consider using the IBM z/OS file system (ZFS) file system for the / partition.

   Your partitioning thus far should look like that shown in Figure 1-7.

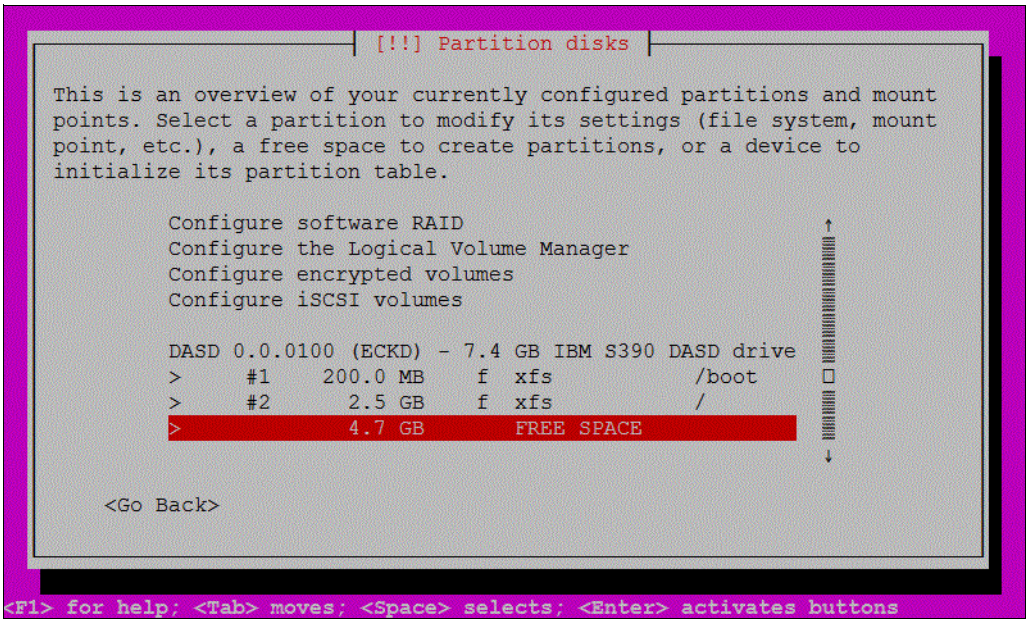


Figure 1-7 Partitioning disks menu showing /boot and / xfs partitions on DASD device 100

   It is now time to create one physical volume for Logical Volume Manager (LVM) with the space that remains on the primary DASD device.

10. Select **FREE SPACE** under *DASD 0.0.0100*.
11. For the partition size, input `max`.
12. Indicate that you want to use this as a **physical volume for LVM**.
13. Select **Done setting up the partition**.

Now that you have a physical volume, you can create a volume group (VG) called `vgsystem` with two logical volumes (LVs):

- `lvvar`, mounted on `/var`
- `lvtmp`, mounted on `/tmp`

Follow these instructions to create the volume group and logical volumes.

1. Back on the partitioning disk overview menu, select **Configure the Logical Volume Manager**.
2. Select **yes** to write the current partitioning scheme to disk.
3. Select **Create volume group** on the LVM configuration action menu.

4. Input `vgsystem` as the **volume group name**.
5. Select the DASD device that will be part of the volume group, and choose **continue**. Figure 1-8 shows using `/dev/dasda3` as an example.

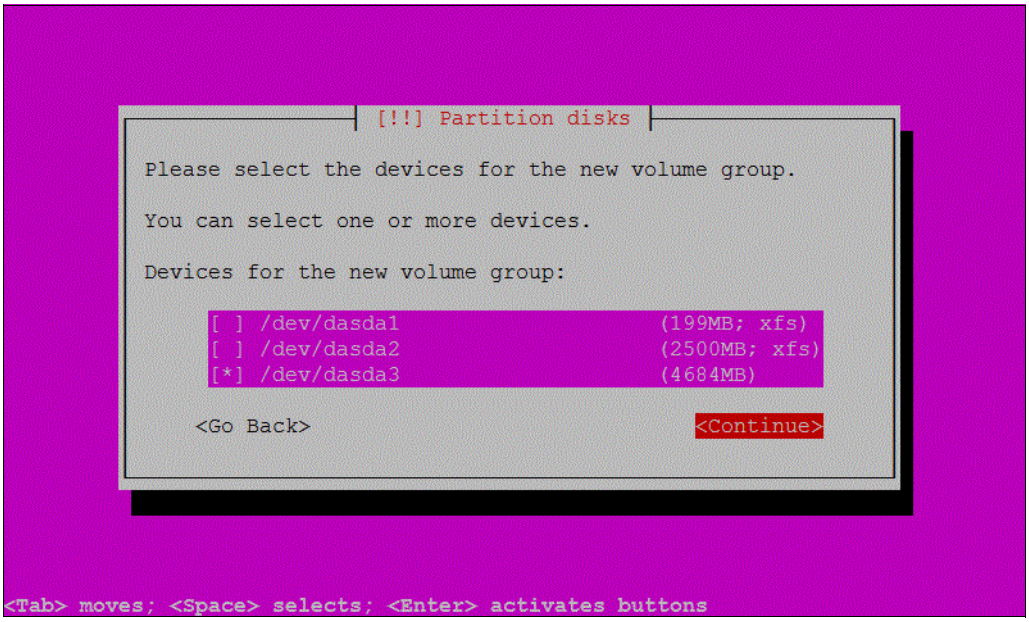


Figure 1-8 Device /dev/dasda3 selected to be part of the vgsystem Volume Group

6. You are shown a summary of the current LVM configuration. If everything looks correct, select **Create logical volume**.
7. For the volume group from which to allocate the logical volume, select **vgsystem**.
8. For the logical volume name, input `lvvar`, then select **Continue**.
9. Input `1 GB` for the logical volume size, then select **Continue**.
10. Repeat steps 6 - 9 to create logical volume `lvtmp` with a size of `800 MB`.

11. Select **Display configuration details** and compare your results with those shown in Figure 1-9.

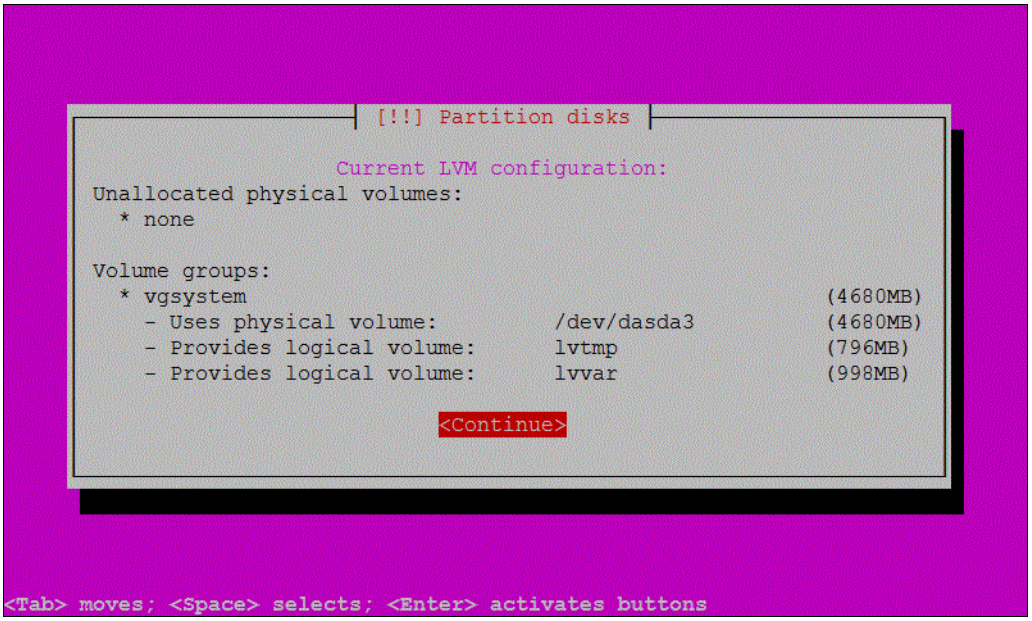


Figure 1-9 Display current configuration menu summarizing current LVM configuration

12. Select **Finish** to conclude the LVM configuration.

The overview of the current configured partitions should now look like that shown in Figure 1-10.

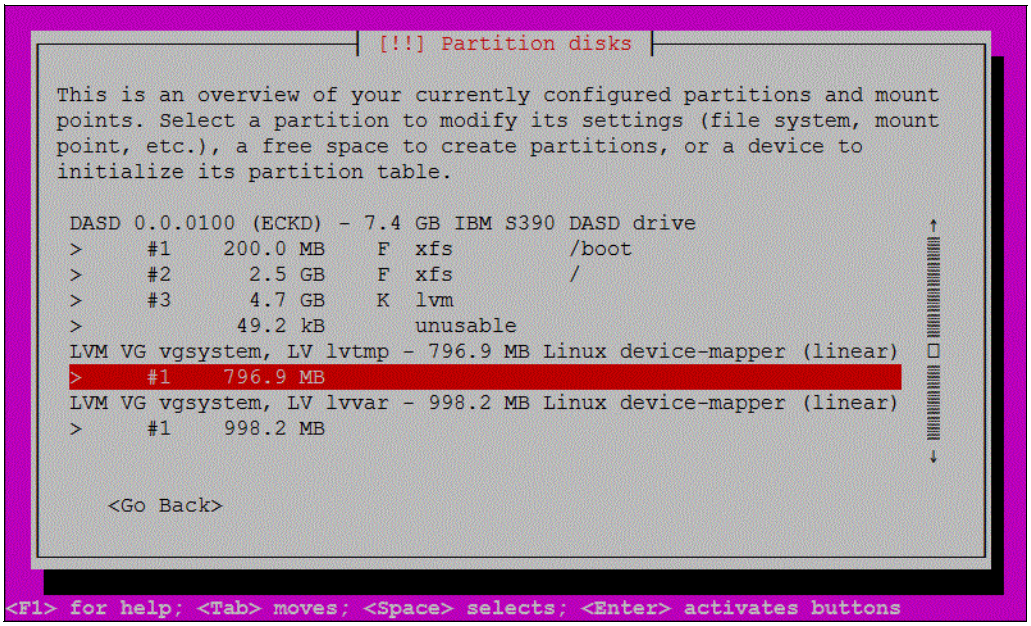


Figure 1-10 Overview of currently configured partitions after LVM configuration

The last thing that you need to do is to mount the logical volumes to their respective directories:

1. Select the `lvtmp` Logical volume and press ENTER.
2. Select **Use as** and select **XFS journaling file system**.
3. Select **Mount point** and scroll to select **/tmp**.
4. Select **Done setting up the partition**.
5. Repeat steps 1-4 for `lvvar` and `/var` as the mount point.
6. Verify your partitioning scheme and mount points with the one shown in Figure 1-11.

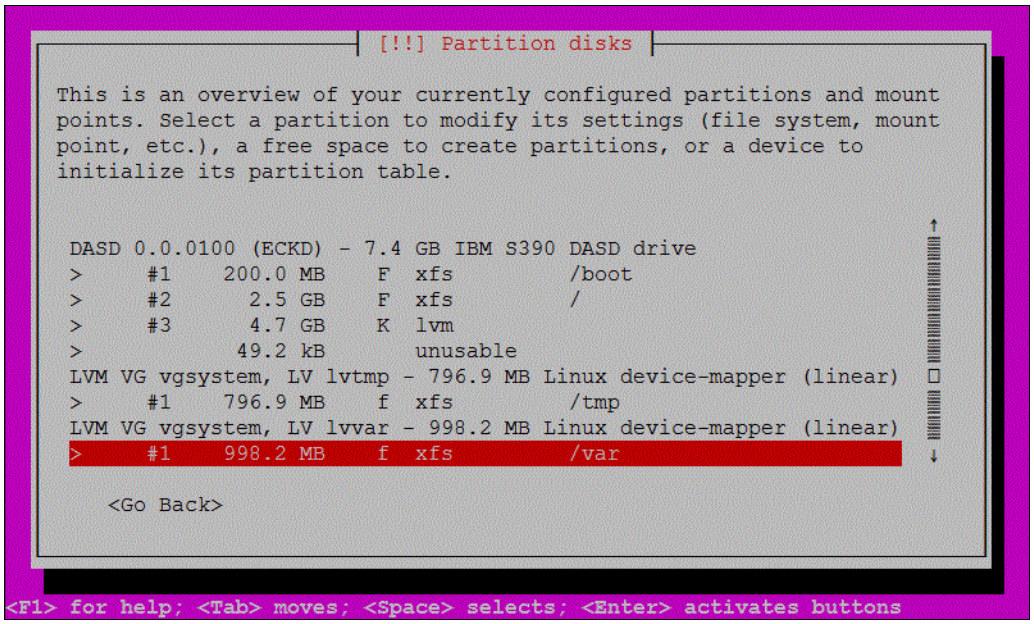


Figure 1-11 Summary of LVM partitioning scheme and corresponding mount points

7. Select **Finish partitioning and write changes to disk**.
8. Enable swap devices *after you load the Linux server*. For now, select no to the prompt about `returning to the partitioning menu`.

   For more information about enablement, refer to 1.3.1, “Enable swap on the virtual disks” on page 20.

   Attention: *Do not enable any swap devices yet.* If you do so, your virtual server will not survive a reboot and will require a complete reinstallation.

9. Select **yes** to write changes to disk concluding the partitioning.

1.2.4 Finishing the installation

The installer now starts installing components for the base system, which can take several minutes to complete. Only a few more steps are required before concluding the installation:

1. Select **No automatic updates** because this requires a direct internet connection to the update servers.

2. At the *Software selection* menu, scroll down to select only the following options, and then select **continue**:
   - OpenSSH server
   - Basic Ubuntu server
3. The installer now downloads and installs all of the necessary components for this server, which can take several minutes to finish. When completed, a panel indicates that it is time to (reboot) the new system, as shown in Figure 1-12. Select **Continue**.

Figure 1-12 Finish the installation panel that indicates the completion of the installation process

4. Return to the 3270 session, where messages indicate that the Linux image is being restarted. You might need to clear the panel several times if you see that the status line indicates `MORE...` or `HOLDING`.
5. An indication that the boot completed successfully is the display of the Linux login prompt:

```
...
Ubuntu 16.04 LTS lnxadmin ttyS0
lnxadmin login:
```

6. When you see the login prompt, disconnect from the terminal using the **`DISCONNECT`** command prefixed by either `%CP` or `#CP`, depending on what you have chosen for your command prefix sign:

```
===> %cp disconnect
```

```
or
```

```
===> #cp disconnect
```

You have now installed the Ubuntu Linux administration system. It should be accessible using SSH with the credentials that are defined during the installation.

1.3 Configure the Ubuntu administration system

Now that you have installed your Ubuntu administration system, you can configure it. This section covers the steps that are involved.

Note: For a system that will become production, you likely require an audit trail from this point forward for any changes made. As such, we suggest that you complete any remaining configuration steps that require elevated privileges or permission using **sudo** before each command.

Using **sudo** to switch user (**su**) to root or using **sudo** to launch a new shell as root is both reckless and dangerous at best, and catastrophic at worst.

1.3.1 Enable swap on the virtual disks

Every time the LNXADMIN virtual server runs the common PROFILE EXEC, two virtual disks are created. On the Linux virtual server, these two in-memory disks can be used as high-speed swap devices.

You might wonder why these were not enabled during installation using the *Configure direct access storage devices (DASD)* menu. The reason is that modern Linux installers refer to these disks by their Universally Unique Identifier (UUID) by default. Because the UUIDs of the virtual disks change every time they are created during IPL, Linux would be unable to access the swap disks and fail to start. Further, recovery from this condition is virtually impossible, because **grub** and **dracut** both generate internal configuration that references these UUIDs.

Thankfully, such a condition is easily avoided entirely by simply omitting swap device configuration during initial installation.

This section shows how to enable swap space using these virtual devices without referring to their UUIDs. To enable swap on the virtual disks:

1. Verify that devices 300 and 301 are visible to the virtual server:

```
support@lnxadmin:~$ sudo lszdev 300,301
TYPE      ID        ON  PERS  NAMES
dasd-fba  0.0.0300  no  no
dasd-fba  0.0.0301  no  no
```

2. Enable and make persistent VDISKS 300 and 301:

```
support@lnxadmin:~$ sudo chzdev -e 300,301
FBA DASD 0.0.0300 configured
FBA DASD 0.0.0301 configured
```

3. View online DASD devices:

```
support@lnxadmin:~$ sudo lsdadsd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
================================================================================
0.0.0100   active      dasda     94:0    ECKD  4096   7042MB    1802880
0.0.0300   active      dasdb     94:4    FBA   512    256MB     524288
0.0.0301   active      dasdc     94:8    FBA   512    512MB     1048576
```

Note: The **SWAPGEN EXEC** that you used to create the VDisks for swap also formats and partitions the disks for swap usage.

4. Activate the swap partition on that disk by using the **swapon** command-line utility:

```
support@lnxadmin:~$ sudo swapon -p 5 /dev/disk/by-path/ccw-0.0.0300-part1
support@lnxadmin:~$ sudo swapon -p 4 /dev/disk/by-path/ccw-0.0.0301-part1
```

5. Verify the activated swap devices:

```
support@lnxadmin:~$ sudo swapon --show
NAME        TYPE        SIZE USED PRIO
/dev/dasdb1 partition 253.9M   0B    5
/dev/dasdc1 partition 507.8M   0B    4
```

6. Make a copy of the /etc/fstab file to serve as a backup:

```
support@lnxadmin:~$ sudo cp /etc/fstab /etc/fstab.orig
```

7. Add the swap disks to the /etc/fstab:

```
support@lnxadmin:~$ sudo echo '/dev/disk/by-path/ccw-0.0.0300-part1 swap swap
pri=5 0 0' >> /etc/fstab
support@lnxadmin:~$ sudo echo '/dev/disk/by-path/ccw-0.0.0301-part1 swap swap
pri=4 0 0' >> /etc/fstab
```

8. Verify the changes to /etc/fstab by using the **cat** command, as shown in Example 1-14.

Example 1-14 Verify the changes using the cat command

```
support@lnxadmin:~$ sudo cat /etc/fstab

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/dasda2 during installation
UUID=074c631c-5c08-4cfa-b6f6-0f10145bb3c3 /              xfs     defaults
0       0
# /boot was on /dev/dasda1 during installation
UUID=ac52f1ec-9bb7-4ec2-a435-50996fb2d5d1 /boot          xfs     defaults
0       0
/dev/mapper/vgsystem-lvtmp /tmp            xfs     defaults        0       0
/dev/mapper/vgsystem-lvvar /var            xfs     defaults        0       0
/dev/disk/by-path/ccw-0.0.0300-part1 swap swap pri=5 0 0
/dev/disk/by-path/ccw-0.0.0301-part1 swap swap pri=4 0 0
```

1.3.2 Set up data DASD disks to a running Linux virtual server

This section describes the process of adding DASD disks to a running Ubuntu Linux system. The goal is to create a LV to be mounted under the /srv directory. This directory space is used in the steps described in 1.3.3, “Build a local Ubuntu mirror with apt-mirror on the LNXADMIN virtual server” on page 25 to hold the files that are used to build other Ubuntu Linux virtual servers. An LVM is used to allow increasing of this space using additional disks without disruption to the virtual server.

To set up the data disk, follow these steps:

1. Verify access to disk 200 using the **lszdev** command:

```
support@lnxadmin:~$ sudo lszdev 200
TYPE       ID        ON  PERS  NAMES
dasd-eckd  0.0.0200  no  no
```

2. Enable and make persistent minidisk 200 with the **chzdev** command:

```
support@lnxadmin:~$ sudo chzdev -e 200
ECKD DASD 0.0.0200 configured
```

3. Use the **lszdev** command again to verify that minidisk 200 is enabled and was made persistent:

```
support@lnxadmin:~$ sudo lszdev 200
TYPE       ID        ON   PERS  NAMES
dasd-eckd  0.0.0200  yes  yes   dasdd
```

4. Format the minidisk using the **dasdfmt** command:

```
support@lnxadmin:~$ sudo dasdfmt -b 4096 -y -d cdl -f
/dev/disk/by-path/ccw-0.0.0200
```

5. Create a partition for your DASD device by using the **fdasd** command, as shown in Example 1-15.

Example 1-15 Create a partition using the fdasd command

```
support@lnxadmin:~$ sudo fdasd -a /dev/disk/by-path/ccw-0.0.0200
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

6. Verify the partition by printing the partition table on DASD device 200, as shown in Example 1-16.

Example 1-16 Verify the partition

```
support@lnxadmin:~$ sudo fdasd -p /dev/disk/by-path/ccw-0.0.0200
reading volume label ..: VOL1
reading vtoc ..........: ok

...

 ------------------------------ tracks -------------------------------
                        Device      start      end   length   Id  System
/dev/disk/by-path/ccw-0.0.02001          2   150239   150238    1  Linux native
exiting...
```

7. Create a re-sizable file system where the server repository files will reside using LVM:

   a. Initialize the partition as an LVM physical volume (PV) by using the **pvcreate** command:

```
support@lnxadmin:~$ sudo pvcreate /dev/disk/by-path/ccw-0.0.0200-part1
Physical volume "/dev/dasdb1" successfully created
```

b. Create a new volume group, `vgdata`, and add the physical volume created previously by using the **`vgcreate`** command:

```
support@lnxadmin:~$ sudo vgcreate vgdata
/dev/disk/by-path/ccw-0.0.0200-part1
Volume group "vgdata" successfully created
```

c. View information about this volume group by using the **`vgdisplay`** command, as shown in Example 1-17.

Example 1-17 View information about this group using the vgdisplay command

```
support@lnxadmin:~$ sudo vgdisplay vgdata
--- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               6.88 GiB
  PE Size               4.00 MiB
  Total PE              1760
  Alloc PE / Size       0 / 0
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               Wzx6et-3qOK-e6hl-8TtA-fK1w-abnO-YCWWrr
```

d. Create a logical volume, `lvdata`, by using the **`lvcreate`** command:

```
support@lnxadmin:~$ sudo lvcreate -l 1760 -n lvdata vgdata
Logical volume "lvdata" created.
```

e. View information about this logical volume by using the **`lvdisplay`** command as shown in Example 1-18.

Example 1-18 View information about this logical volume using the lvdisplay command

```
support@lnxadmin:~$ sudo lvdisplay vgdata
--- Logical volume ---
  LV Path                /dev/vgdata/lvdata
  LV Name                lvdata
  VG Name                vgdata
  LV UUID                uzrCvw-jwDj-yIpy-TNjq-xgHu-Jn3O-CFYL4G
  LV Write Access        read/write
  LV Creation host, time lnxadmin, 2016-05-23 18:45:01 -0400
  LV Status              available
  # open                 0
  LV Size                6.88 GiB
  Current LE             1760
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
```

```
  - currently set to     1024
  Block device           252:2
```

f. Create a file system for the new logical volume by using the **mkfs.xfs** command, as shown in Example 1-19.

Example 1-19 Create a file system

```
support@lnxadmin:~$ sudo mkfs.xfs /dev/vgdata/lvdata
meta-data=/dev/vgdata/lvdata     isize=512    agcount=4, agsize=450560 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=0
data     =                       bsize=4096   blocks=1802240, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

8. Edit the contents of the `/etc/fstab` file to permanently add a mount point for the newly created file system:

   support@lnxadmin:~$ echo "/dev/vgdata/lvdata /srv xfs defaults 0 0" | sudo tee -a /etc/fstab

9. Verify the contents of the `/etc/fstab` file as it stands now, as shown in Example 1-20.

Example 1-20 Verify the contents of the file

```
support@lnxadmin:~$ less /etc/fstab

# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/dasda2 during installation
UUID=074c631c-5c08-4cfa-b6f6-0f10145bb3c3 /              xfs     defaults
0       0
# /boot was on /dev/dasda1 during installation
UUID=ac52f1ec-9bb7-4ec2-a435-50996fb2d5d1 /boot          xfs     defaults
0       0
/dev/mapper/vgsystem-lvtmp /tmp            xfs     defaults        0       0
/dev/mapper/vgsystem-lvvar /var            xfs     defaults        0       0
/dev/disk/by-path/ccw-0.0.0300-part1 swap swap pri=5 0 0
/dev/disk/by-path/ccw-0.0.0301-part1 swap swap pri=4 0 0
/dev/vgdata/lvdata  /srv xfs defaults 0 0
```

10. Reload the `/etc/fstab` configuration by using the **mount** command:

```
support@lnxadmin:~$ sudo mount -a
```

11. Verify that the newly created file system was mounted properly by again using the **mount** command, as shown in Example 1-21.

Example 1-21 Verify that the file system was mounted

```
support@lnxadmin:~$ mount
...
name=systemd on /run/lxcfs/controllers/name=systemd type cgroup
(rw,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=system
d,nsroot=/)
lxcfs on /var/lib/lxcfs type fuse.lxcfs
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

```
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=81156k,mode=700,uid=1000,gid=1000)
/dev/mapper/vgdata-lvdata on /srv type xfs (rw,relatime,attr2,inode64,noquota)
```

1.3.3 Build a local Ubuntu mirror with apt-mirror on the LNXADMIN virtual server

The purpose of the LNXADMIN virtual server is to function as the installation source for other Linux virtual servers. Because the Ubuntu installer requires web access to a mirror server from which it can get most of its required packages, you create a local software repository mirror using the **apt-mirror** utility. Therefore, you do not need to rely on an open connection to the Internet for each subsequent virtual server that you create. This mirror minimizes network bandwidth and achieves better version control.

There are several repositories from which you can find debian packages and binaries, namely *main, restricted, universe*, and *multiverse*. A basic Ubuntu system requires at least access to the *main* and *restricted* repositories. In this section, you set up the **apt-mirror** utility and configure it to get all s390x packages from both the main and restricted software repositories.

Follow these steps:

1. Verify the size of the /srv mount point:

```
support@lnxadmin:~$ sudo df -h /srv
Filesystem                 Size  Used Avail Use% Mounted on
/dev/mapper/vgdata-lvdata   14G   33M   14G   1% /srv
```

Note: As you can see in this example, the /srv partition is about 14 GB in size. As of the time of writing, downloading the contents of the *main* and *restricted* repositories required approximately 9.5 GB of space available. Depending on the repositories that you choose to download, the required available space will vary. Instructions about how to increase the size of the lvdata logical volume to accommodate your needs are provided in Chapter 4, “Working with disks” on page 57.

2. Install the **apt-mirror** utility, as shown in Example 1-22.

Example 1-22 Install the apt-mirror utility

```
support@lnxadmin:~$ sudo apt-get -y install apt-mirror
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  apt-mirror
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 13.5 kB of archives.
After this operation, 64.5 kB of additional disk space will be used.
Get:1 http://ports.ubuntu.com xenial/universe s390x apt-mirror all
0.5.1-1ubuntu
1 [13.5 kB]
Fetched 13.5 kB in 0s (66.3 kB/s)
Selecting previously unselected package apt-mirror.
(Reading database ... 49921 files and directories currently installed.)
Preparing to unpack .../apt-mirror_0.5.1-1ubuntu1_all.deb ...
```

```
Unpacking apt-mirror (0.5.1-1ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up apt-mirror (0.5.1-1ubuntu1) ...
```

3. Make a directory to hold the contents of the repository:

```
support@lnxadmin:~$ sudo mkdir -p /srv/www/ubuntu
```

4. Configuration for the **apt-mirror** utility is done in the `/etc/apt/mirror.list` file. A generic configuration was created during installation. Make the changes shown in Example 1-23 to this file using your favorite text editor.

Example 1-23 Configuration for the apt-mirror utility

```
############# config ##################
#
 set base_path    /srv/www/ubuntu
#
# set mirror_path  $base_path/mirror
# set skel_path    $base_path/skel
# set var_path     $base_path/var
# set cleanscript $var_path/clean.sh
# set defaultarch  <running host architecture>
# set postmirror_script $var_path/postmirror.sh
# set run_postmirror 0
set nthreads     20
set _tilde 0
#
############# end config ##############

deb-s390x http://ports.ubuntu.com xenial main restricted
deb-s390x http://ports.ubuntu.com xenial-security main restricted
deb-s390x http://ports.ubuntu.com xenial-updates main restricted
deb-s390x http://ports.ubuntu.com xenial main/debian-installer
deb-s390x http://ports.ubuntu.com xenial restricted/debian-installer

clean http://ports.ubuntu.com/
```

5. Populate the repository by using the **apt-mirror** command, as shown in Example 1-24.

Example 1-24 Populate the repository

```
# apt-mirror

Downloading 66 index files using 20 threads...
Begin time: Tue May 24 10:49:09 2016
[20]... [19]... [18]... [17]... [16]... [15]... [14]... [13]... [12]... [11]...
[10]... [9]... [8]... [7]... [6]... [5]... [4]... [3]... [2]... [1]... [0]...
End time: Tue May 24 10:51:53 2016

...

9.5 GiB will be downloaded into archive.
Downloading 7238 archive files using 20 threads...
Begin time: Tue May 24 10:57:00 2016
[20]...
...
```

Note: Depending on your Internet connection speeds and the repositories selected, this process can take up to several hours to complete. Each subsequent time that you run this command after this initial download should take less time to complete, because only incremental updates are downloaded.

While you wait, you can learn more about the available repositories at:

https://help.ubuntu.com/community/Repositories

You should now have a local copy of the repositories. In the next session, you install and configure an HTTP server to make this mirror available to other Ubuntu Linux virtual servers on the local network.

You now have the power to determine how often you want to download the latest updates from the remote server. You can manually issue the **apt-mirror** command or schedule a **cron** job to update the repositories automatically on a regular and frequent basis.

You can learn more about defining **cron** jobs at:

https://help.ubuntu.com/community/CronHowto

Important: Carefully consider the implications and risks to your enterprise from falling too far behind, especially with fixes and patches that are released for security reasons.

1.3.4 Install and configure an apache2 HTTP server

To set up the Ubuntu Linux administration server as an HTTP server:

1. Update the repository information:

   ```
   support@lnxadmin:~$ sudo apt-get update
   ```

2. Instal Apache2 server:

   ```
   support@lnxadmin:~$ sudo apt-get install apache2
   ```

3. Enable the apache2 service to start with every reboot of the virtual server:

   ```
   support@lnxadmin:~$ sudo systemctl enable apache2.service
   apache2.service is not a native service, redirecting to systemd-sysv-install
   Executing /lib/systemd/systemd-sysv-install enable apache2
   ```

4. Check the status of the newly installed Apache server, as shown in Example 1-25.

Example 1-25 Status of the Apache server

```
support@lnxadmin:~$ sudo systemctl status apache2.service
* apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           `-apache2-systemd.conf
   Active: active (running) since Tue 2016-05-24 11:00:45 EDT; 2min 54s ago
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/apache2.service
           |-3448 /usr/sbin/apache2 -k start
           |-3450 /usr/sbin/apache2 -k start
           `-3451 /usr/sbin/apache2 -k start

May 24 11:00:45 lnxadmin systemd[1]: Starting LSB: Apache2 web server...
```

```
May 24 11:00:45 lnxadmin apache2[3423]:  * Starting Apache httpd web server apa
May 24 11:00:45 lnxadmin apache2[3423]:  *
May 24 11:00:45 lnxadmin systemd[1]: Started LSB: Apache2 web server.
```

5. Verify that the server is running correctly by pointing your web browser to your server's IP address. You should see a page such as that shown in Figure 1-13.

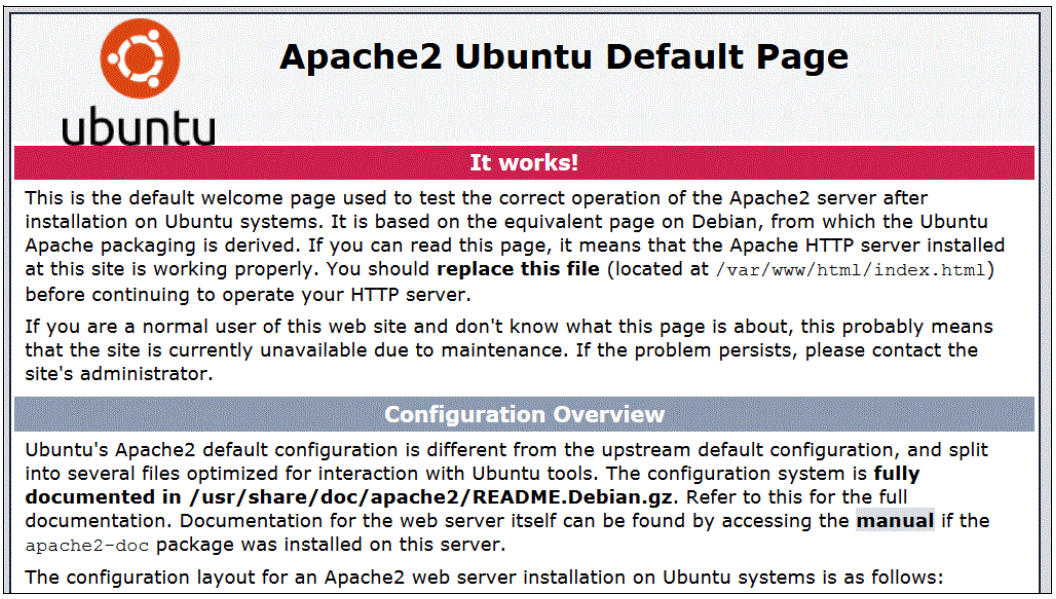


Figure 1-13 Apache2 Ubuntu Default Page verifying that the HTTP server works

6. Create a symbolic link to the downloaded repositories to `/var/www/html/Ubuntu`:

   `support@lnxadmin:~$ sudo ln -s /srv/www/ubuntu/mirror/ports.ubuntu.com/ /var/www/html/ubuntu`

   Any requests for `<ip address>/ubuntu` on the server should now display the contents of `/srv/www/ubuntu/mirror/ports.ubuntu.com/`.

The **`apache2`** HTTP server should now be configured and running on the `LNXADMIN` virtual server. You can further modify this server's configuration according to your company's security policy.

1.3.5 Reconfigure the software repositories

Ubuntu and its package manager (`apt`) remember the location of the installation source and uses it as the source for installing new packages. Now that you have set up the local software repository, you can change the source location to point to itself, which can provide greater control of the packages that are installed on your system and might further reduce bandwidth consumption.

To configure the software repository for the Linux administration system, choose one of the following options:

- Use the local HTTP server
- Use the path to the local repository
- Use FTP (installed later in this chapter)

To modify the current `sources.list` file to point to the local HTTP server:

1. Copy the current sources.list file to keep it as a backup:

   ```
   support@lnxadmin:~$ sudo cp -p /etc/apt/sources.list /etc/apt/sources.list.orig
   ```

2. Create a new `sources.list` file with the new definitions, as shown in Example 1-26.

   Example 1-26 New sources.list file with new definitions

   ```
   support@lnxadmin:~$ sudo editor sources.list

   ## Ubuntu's primary repositories to support installation and other
   ## package installations.
   deb http://9.60.7.36/ubuntu xenial main restricted

   ## Major bug fix updates produced after the final release of the
   ## distribution.
   deb http://9.60.7.36/ubuntu xenial-updates main restricted

   ## Security patches
   deb http://9.60.7.36/ubuntu xenial-security main restricted
   ```

   Note: If you used a proxy during installation, you might need to remove the following line from the file `/etc/apt/apt.conf`:

   ```
   Acquire::http::Proxy "http://proxy.mycompany.com:1234";
   ```

3. Save and close the file using **CTRL**+**X**, then **Y** and **Enter** to confirm the name.

4. Update the `apt` database using the **`apt-get update`** command:

   ```
   # apt-get update
   ```

You now have the Linux administration system (the `LNXADMIN` virtual server) independent from the external Ubuntu servers that are used to install this system. You now have access to only the repositories and components of your choosing. This repository can be used to install and update other Ubuntu servers on your environment without the need to access the public network.

1.3.6 Install and configure vsftpd FTP server

To make the rest of the files that are used in this cookbook series, configure a local FTP server using `vsftpd`. You can install and configure this FTP server by following these steps:

1. Install `vsftpd` from the local repository:

```
support@lnxadmin:~$ sudo apt-get -y install vsftpd
```

> **Note:** As part of the installation, an FTP user is created. This user is a name for an anonymous FTP user. You can verify this by using the tail command:
>
> ```
> $ tail /etc/passwd
> ...
> ftp:x:112:121:ftp daemon,,,:/srv/ftp:/bin/false
> ```
>
> This user has a home directory, `/srv/ftp`, which we will use to host the files provided as part of this book.

2. Edit the `vsftp` configuration file, `/etc/vsftpd.conf`, and make the following changes:

```
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=NO
```

3. Still on the `/etc/vsftpd.conf` file uncomment the following lines:

```
ascii_upload_enable=YES
ascii_download_enable=YES
```

4. Save the file, then restart the `vsftpd` service to reload its configuration, and verify the current status, as shown in Example 1-27.

Example 1-27 Verify the current status

```
support@lnxadmin:~$ sudo systemctl restart vsftpd.service
support@lnxadmin:~$ sudo systemctl status vsftpd.service

* vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset:
enabled)
   Active: active (running) since Tue 2016-05-24 11:25:22 EDT; 1min 32s ago
 Main PID: 2867 (vsftpd)
   CGroup: /system.slice/vsftpd.service
           `-2867 /usr/sbin/vsftpd /etc/vsftpd.conf

May 24 11:25:22 ubuntuadm systemd[1]: Starting vsftpd FTP server...
May 24 11:25:22 ubuntuadm systemd[1]: Started vsftpd FTP server.
```

5. Enable the `vsftpd` to start after system reboot:

```
support@lnxadmin:~$ sudo systemctl enable vsftpd.service
Synchronizing state of vsftpd.service with SysV init with
/lib/systemd/systemd-sysv-install...
Executing /lib/systemd/systemd-sysv-install enable vsftpd
```

The vsftpd FTP server should now be configured and running on the LNXADMIN virtual server. The next section explains how to import the contents of the files that are associated with this book.

1.3.7 Import files that are associated with this book

In this section, you download and import all of the files that you staged on the local FTP server to the LNXADMIN virtual server directory in the LNX SFS pool, which enables you to eliminate the dependency with this server. This step is optional but can be useful and perhaps even necessary in some installations.

To download the files that are associated with this book:

1. Change directory to /srv/ftp, if you are not already there:

   ```
   support@lnxadmin:~$ cd /srv/ftp
   ```

2. Obtain the latest version of the files that are associated with this cookbook series:

   a. Go to the following website to verify that you are downloading the current version of the associated files:

      http://www.vm.ibm.com/pubs/redbooks/sg248147/

   b. The following example shows how to use the **wget** command to download a compressed tape archive (TAR) named *example*.tgz into the current directory:

      ```
      support@lnxadmin:~$ sudo wget -v
      www.ibm.com/vm/pubs/redbooks/sg248147/files/example.tgz
      ...
      ```

      Note: Alternatively, you can import these files from the FTP server that is set up in the section about configuring an FTP server in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

3. Expand the .tar file with the **tar** command and the flags for decompression, expansion, and verbosity (**-zxv**):

   ```
   support@lnxadmin:~$ sudo tar -zxvf example.tgz
   ```

These files should now be available under the /srv/ftp/SG24-8147-01/ directory.

1.3.8 Download the bootstrap files for Ubuntu 16.04

You can choose to use the bootstrap files that are provided with the compressed archive that accompanies this series of books, or you can choose to download the files from Ubuntu.com.

Important: If you are using the bootstrap files that are provided with this book, skip this section.

Just like the files that are associated with the book, which are downloaded in the previous section, you can store the bootstrap files that are used to start an installation locally. Check these files regularly for updates. You might also consider the use of a **cron** job to check the files.

Follow these steps to download the bootstrap files from Ubuntu's site. You can use **rsync** rather than **wget** if you so choose:

1. Make a new directory to store the bootstrap files and set the ownership and permissions:

```
support@lnxadmin:~$ sudo mkdir -p /srv/ftp/ubuntu16.04
support@lnxadmin:~$ sudo chgrp -R support srv/ftp/ubuntu16.04
support@lnxadmin:~$ sudo chmod g+rwxs /srv/ftp/ubuntu16.04
```

2. Change the current working directory to `/srv/ftp/ubuntu16.04`:

```
support@lnxadmin:~$ cd /srv/ftp/ubuntu16.04/
```

3. Download the bootstrap files from Ubuntu's site using the **wget** command, as shown in Example 1-28.

Example 1-28 Download the bootstrap files

```
support@lnxadmin:/srv/ftp/ubuntu16.04 $ sudo wget -v
http://ports.ubuntu.com/ubuntu-ports/dists/xenial/main/installer-s390x/current/
images/generic/initrd.ubuntu

support@lnxadmin:/srv/ftp/ubuntu16.04 $ sudo wget -v
http://ports.ubuntu.com/ubuntu-ports/dists/xenial/main/installer-s390x/current/
images/generic/kernel.ubuntu

support@lnxadmin:/srv/ftp/ubuntu16.04 $ sudo wget -v
http://ports.ubuntu.com/ubuntu-ports/dists/xenial/main/installer-s390x/current/
images/generic/parmfile.ubuntu
```

You should now have the bootstrap files to punch into the VM Reader in order to start a new installation.

1.3.9 Additional configurations

This section describes the following topics:

- Modifications to how DASD devices are mounted from the Linux File System Table (fstab)
- Enablement of Linux Terminal Services using IUCV
- Options for scaling
- Options for enhanced security

DASD mount parameters

Modify the `/etc/fstab` file to ensure that DASD devices are being mounted *by path* and not by UUID, which can be useful if you ever need to rescue your Linux system.

To modify the `/etc/fstab` file to ensure that DASD devices are being mounted *by path* and not by UUID:

1. Make a copy of the current `/etc/fstab` file:

```
support@lnxadmin:~$ sudo cp -p /etc/fstab /etc/fstab.works
```

2. Use your favorite text editor to modify the `/etc/fstab` file as follows:
   a. Remove the information about `blkid`, because this does not apply when using DASD.
   b. For each line that contains a UUID string:
      - Duplicate the line containing the UUID string directly underneath the original.

- Comment the original line by inserting the Number sign (#) character as the first character of the line.
- Modify the duplicated line and replace the UUID with the correct path to the DASD.

Example 1-29 shows the file before editing, and Example 1-30 on page 33 shows the file after editing.

Example 1-29 The /etc/fstab file before editing

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>              <dump>  <pass>
# / was on /dev/dasda2 during installation
UUID=074c631c-5c08-4cfa-b6f6-0f10145bb3c3       /        xfs  defaults  0  0
# /boot was on /dev/dasda1 during installation
UUID=ac52f1ec-9bb7-4ec2-a435-50996fb2d5d1       /boot    xfs  defaults  0  0
```

Example 1-30 The /etc/fstab file after editing is complete

```
# /etc/fstab: static file system information.     See fstab(5).
##############################################################################
# <file system> <mount point>   <type>  <options>              <dump>  <pass>
##############################################################################
# / was on /dev/dasda2 during installation
# UUID=074c631c-5c08-4cfa-b6f6-0f10145bb3c3       /       xfs  defaults  0  0
/dev/disk/by-path/ccw-0.0.0100-part2              /       xfs  defaults  0  0
# /boot was on /dev/dasda1 during installation
# UUID=ac52f1ec-9bb7-4ec2-a435-50996fb2d5d1       /boot   xfs  defaults  0  0
/dev/disk/by-path/ccw-0.0.0100-part1              /boot   xfs  defaults  0  0
```

3. Save your changes and close the file.
4. Use the **mount -a** command to ensure that you do not get any errors:

```
support@lnxadmin:~$ sudo mount -a
```

Enable Linux Terminal Services

This particular virtual server, LNXADMIN, is the basis for the Linux Terminal Services environment. Many IBM customers who run Linux under z/VM consider the implementation of Linux Terminal Services to be a fundamental requirement for a Linux Virtual Server to be eligible for classification as a production system in their environment.

Additionally, it is based on a character mode interface that enables the use of traditional Linux full-screen tools, such as **vi**, **nano**, **emacs**, and so on, that do not function when using a line-mode terminal, such as a 3270 emulator.

The IUCV Linux Terminal Server can be your “lifeboat” in a time of crisis, and you will be glad that you took the time to enable it. Go to 7.8, “Setting up the IUCV Linux Terminal Server” on page 134 to complete the tasks necessary for setup.

Other items to consider before your first reboot

The following links provide other topics for your consideration before your first reboot:

- 7.2, “Set up Memory Hotplugging” on page 115
- 7.3, “Dynamically add or remove virtual CPUs” on page 118
- 7.4, “Use the cpuplugd service” on page 119
- 7.5, “Hardware Cryptographic Support for OpenSSH using Ubuntu 16.04” on page 124
- 7.6, “Use Crypto Express to seed /dev/random” on page 128
- 7.7.1, “The X Window System” on page 130

1.3.10 Reboot the system

Now reboot the system to test the changes:

```
support@lnxadmin:~$ sudo reboot
Connection to 9.60.7.36 closed by remote host.
Connection to 9.60.7.36 closed.
```

Your system should be back up in a few minutes. You are now done customizing the Linux administration system Linux image.

1.3.11 Verify that the system starts correctly

To verify if the system starts correctly:

1. Start an SSH session as `root` to the Linux administration system.
2. Verify that all of the swap spaces are operational:

```
support@lnxadmin:~$ sudo swapon --show
NAME        TYPE        SIZE USED PRIO
/dev/dasde1 partition 507.8M   0B    4
/dev/dasdb1 partition 253.9M   0B    5
```

3. Verify that the HTTP service is running:

```
support@lnxadmin:~$ sudo systemctl status apache2
* apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           `-apache2-systemd.conf
   Active: active (running) since Tue 2016-05-24 11:52:18 EDT; 48s ago
```

4. Verify that the FTP service is running:

```
support@lnxadmin:~$ sudo systemctl status vsftpd
* vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset:
enabled)
   Active: active (running) since Tue 2016-05-24 11:52:18 EDT; 3min 40s ago
```

Congratulations

You have installed and configured an Ubuntu 16.04 Linux system onto the Linux administration system. In the next chapter, you will learn about automating the installation process for future deployments of Ubuntu.

2

Automating installation

“Please cut off a nanosecond and send it over to me.”

— Grace Hopper

This chapter describes several different methodologies for quickly provisioning additional virtual servers, with an emphasis on automated installation. Anyone who has used Linux with some regularity knows that there are usually many methods of achieving the same result. The automated installation of systems is no exception.

It includes the following topics:

- Kickstart
- Configure LNXS0031 for Kickstart using FCP devices
- Preseeding
- Cloning
- Create golden image for cloning
- Clone the golden image using DirMaint
- Send a configuration update to the clone system
- Start the clone system

2.1 Kickstart

Kickstart was chosen as the preferred method of provisioning in this book, based on its popularity among the Ubuntu user community.

Kickstart provides an automated way of pre-answering questions that the Ubuntu Linux installer normally prompts for. This method makes installations fast and easy, and frees you from hours of answering the same questions over and over.

After the creation of an answer file, Kickstart now knows how to answer all of the questions that will be asked by the installation program. Deployment of Ubuntu onto a virtual server with Kickstart gives you flexibility when installing multiple Linux systems by enabling you to tailor configuration parameters, as well as pre-installation and post-installation scripting.

This section shows you how to create two example virtual servers, `LNXS0030` and `LNXS0031`, using Kickstart. Using the Linux administration virtual server, `LNXADMIN`, we build upon previous work from step 1.3.4, “Install and configure an apache2 HTTP server” on page 27 to enable automated installations over the network.

2.1.1 Using Kickstart with EDEV

You can use a new virtual server with the `LNXS0030` ID for Kickstart using EDEV devices. Use these steps:

1. Use *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147 of this series of books to complete the following tasks:
   a. Create the new virtual server.
   b. Enroll the user in the LNX file pool, and create the necessary aliases.
2. Log on as `LNXS0030` from the 3270 console. The `PROFILE EXEC` runs and prompts you for a response whether you want to boot Linux from a minidisk, as shown in example Example 2-1. Indicate no by typing the letter **`n`** and pressing Enter:

   ```
   ===> n
   ```

Example 2-1 Running PROFILE EXEC

```
LOGON LNXS0030
00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0003 RDR,   NO PRT,   NO PUN
00: LOGON AT 13:39:17 EDT THURSDAY 2016-05-26
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2015-04-09 09:04

DMSVML2060I TCPMAINT 592 linked as 0120 file mode Z
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
Do you want to IPL Linux from minidisk 100? y/n

n
```

3. Verify that the shared file system directories in the LNX: pool are visible, as shown in Example 2-2:
   - Mode A should be LNX:LNXSO030
   - Mode D should be LNX:LNXADMIN.UBUNTU
   - Mode E should be LNX:LNXADMIN.SWAPGEN

Example 2-2 Query file system directories

```
===> query accessed
Mode  Stat     Files  Vdev  Label/Directory
A      R/W         1  DIR   LNX:LNXSO030.
B      R/W         1  300   LXSWAP
C      R/W         1  301   LXSWAP
D      R/O        10  DIR   LNX:LNXADMIN.UBUNTU
E      R/O        18  DIR   LNX:LNXADMIN.SWAPGEN
S      R/O       698  190   MNT190
Y/S    R/O      1123  19E   MNT19E
Z      R/O       892  120   TCM592
Ready; T=0.01/0.01 17:02:35
```

4. Set the 3270 console to automatically clear messages and then run **ubuntu exec** to initiate the Kickstart. You see some initial kernel messages, followed by the rest of the installation process:

```
===> cp term more 0 0
===> ubuntu exec
```

You have now installed Ubuntu Server onto the virtual server using Kickstart. You can repeat this process for other Linux virtual servers.

2.2 Configure LNXS0031 for Kickstart using FCP devices

This section provides the steps to configure a virtual server for Kickstart using FCP devices:

1. Log on as LNXSO031 from the 3270 console. You notice that the REXX **EXEC** loads as part of the PROFILE EXEC and presents a message prompting to boot disk 100, as shown in Example 2-3. For this step, answer n for no.

Example 2-3 LNX0031 logon

```
LOGON LNXSO031
OO: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
OO: built on IBM Virtualization Technology
OO: There is no logmsg data
OO: FILES: 0002 RDR,   NO PRT,   NO PUN
OO: LOGON AT 10:02:17 EDT FRIDAY 04/17/15
OO: Command complete
OO: NIC 0600 is created; devices 0600-0602 defined
OO: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2015-04-09 09:04

DMSVML2060I TCPMAINT 592 linked as 0120 file mode Z
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
```

```
DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
Do you want to IPL Linux from minidisk 100? y/n
n
```

2. Verify that the shared file system pool, LNX:LNXSO031 and LNX:LNXADMIN, is available to LNXSO031, as shown in Example 2-4.

Example 2-4 Shared system pool is available

```
===> q accessed
Mode  Stat     Files  Vdev  Label/Directory
A      R/W         3  DIR   LNX:LNXSO031.
B      R/W         1  300   LXSWAP
C      R/W         1  301   LXSWAP
D      R/O        10  DIR   LNX:LNXADMIN.UBUNTU
E      R/O        18  DIR   LNX:LNXADMIN.SWAPGEN
S      R/O       698  190   MNT190
Y/S    R/O      1123  19E   MNT19E
Z      R/O       892  120   TCM592
```

3. Copy GENERIC PRM from the Shared disk to your local A disk:

```
===> copyfile generic prm d = = a
===> listfiles * * a
PROFILE  EXEC     A1
GENERIC  PRM      A1
PROFILE  XEDIT    A1
Ready; T=0.01/0.01 14:46:17
```

4. Edit the GENERIC PRM file, all changes are in **bold**:

```
ro ramdisk_size=40000 cio_ignore=all,!condev
ip=9.60.7.99::9.12.4.1:20:vmlnx2-4.itso.ibm.com:enccw0.0.0600:none
rd.znet=qeth,0.0.0600,0.0.0601,0.0.0602,layer2=1
nameserver=9.12.6.7 nameserver=9.12.6.6
rd.zfcp=0.0.fc00,0x500507630500c74c,0x4010401800000000
rd.zfcp=0.0.fd00,0x500507630510c74c,0x4010401800000000
inst.repo=ftp://9.60.7.96/pub/UbuntuLNXSO03071
ks=ftp://9.60.7.96/pub/kickstart/LNXSO031-ks.cfg
inst.cmdline
```

Note: rd.zfcp parameter has the following parts:

- The virtual device, for example, fc00
- The WWPN of the storage, for example, 500507630500c74c
- The LUN, for example, 4010401800000000

5. Set the terminal to automatically clear, and then run **UBUNTU EXEC** to initiate the kickstart. You see some initial kernel messages, followed by the remainder of the installation process:

```
===> cp term more 0 0
===> UBUNTU exec
```

6. After the installation is successful, log on to LNXSO030 using an SSH client:

```
# ssh root@9.60.7.99
root@9.60.7.99's password:
```

7. Verify the FCP configuration by using the **lsluns** command, as shown in Example 2-5.

Example 2-5 Verify the FCP configuration

```
# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507630500c74c:
                0x4010401800000000
        at port 0x50050763050bc74c:
                0x4010401800000000
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507630510c74c:
                0x4010401800000000
        at port 0x50050763051bc74c:
                0x4010401800000000
```

Even though only one path is provided in the `GENERIC PRM` CMS file, the FCP autoscan feature enabled all paths to the wanted LUN automatically during the installation process.

Note: If the output of the **lsluns** command is as shown in the following code, load the kernel module `sg` by using the **modprobe sg** command:

```
# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507630500c74c:
                Cannot attach WLUN / LUN0 for scanning.
        at port 0x50050763050bc74c:
                Cannot attach WLUN / LUN0 for scanning.
lsluns: Error: Please load/configure SCSI Generic (sg) to use lsluns.
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507630510c74c:
                Cannot attach WLUN / LUN0 for scanning.
        at port 0x50050763051bc74c:
                Cannot attach WLUN / LUN0 for scanning.

# modprobe sg
# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507630500c74c:
                0x4010401800000000
        at port 0x50050763050bc74c:
                0x4010401800000000
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507630510c74c:
                0x4010401800000000
        at port 0x50050763051bc74c:
                0x4010401800000000
```

You have now installed Ubuntu Server onto the virtual server using Kickstart. You can repeat this process for other Linux virtual servers.

Note: On step 4 you modified the `GENERIC PRM` file and added the FCP devices. It is also possible to install Ubuntu using FCP devices by specifying them only in the Kickstart file rather than the `GENERIC PRM` CMS file:

1. Edit the `GENERIC PRM` file, note that in this case you do not specify the FCP devices in this file:

```
ro ramdisk_size=40000 cio_ignore=all,!condev
ip=9.60.7.99::9.12.4.1:20:vmlnx2-4.itso.ibm.com:enccw0.0.0600:none
rd.znet=qeth,0.0.0600,0.0.0601,0.0.0602,layer2=1
nameserver=9.12.6.7 nameserver=9.12.6.6
inst.repo=ftp://9.60.7.96/pub/Ubuntu71
ks=ftp://9.60.7.96/pub/kickstart/LNXS0031-ks.cfg
inst.cmdline
```

2. Edit the `LNXS0031-ks.cfg` file and add the following entries:

```
...
# Use text mode install
text
zfcp --devnum=fc00 --wwpn=500507630500c74c --fcplun=0x4010401800000000
zfcp --devnum=fd00 --wwpn=500507630510c74c --fcplun=0x4010401800000000
# Keyboard layouts
keyboard --vckeymap=us --xlayouts='us'
# System language
...
```

2.3 How to boot SCSI over FCP (LNXS0031)

`LNXS0031` was installed directly on SCSI over FCP to boot the Linux virtual server installed on SCSI is a different process. Follow this process:

1. Log on to `LNXS0031` using a 3270 terminal and answer **`n`** to the `PROFILE EXEC` REXX **`EXEC`**, as shown in Example 2-6.

Example 2-6 Log on to LNXS0031

```
LOGON LNXS0031
00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0003 RDR,   NO PRT,   NO PUN
00: LOGON AT 14:40:24 EDT WEDNESDAY 04/22/15
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
DMSACC724I 19E replaces Y (19E)
DMSACP723I Y (19E) R/O
z/VM V6.3.0    2015-04-09 09:04
DMSWSP100W Shared S-STAT not available
DMSWSP100W Shared Y-STAT not available
DMSVML2060I TCPMAINT 592 linked RR as 0592 file mode Z
LNXS0031   AT ITSOZVM1 VIA RSCS     2015-04-22 14:40:24 EDT      WEDNESDAY
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
```

```
Do you want to IPL Linux from minidisk 100? y/n
n
```

2. Type the commands shown in Example 2-7 to start the virtual FCP device, where *portname* is the WWPN of the storage port for `fc00` and LUN is the `lun` definition. However, note that they are separated in blocks of eight digits.

Example 2-7 Start the virtual FCP device

```
===> cp set loaddev portname 50050763 0500c74c lun 40104018 00000000
===> cp ipl fc00

cp ipl fc00
00: HCPLDI2816I Acquiring the machine loader from the processor controller.
00: HCPLDI2817I Load completed from the processor controller.
00: HCPLDI2817I Now starting the machine loader.
01: HCPGSP2630I The virtual machine is placed in CP mode due to a SIGP stop and
store status from CPU 00.
00: MLOEVL012I: Machine loader up and running (version v2.4.5).
00: MLOPDM003I: Machine loader finished, moving data to final storage location.
00: Uncompressing Linux...
00: Ok, booting the kernel.
00:
...

Ubuntu Server Server 16.04 LTS (Xenial Xerus)
...

LNXS0031 login:
```

To automate the boot of SCSI over FCP, use the `PROFFCP EXEC` REXX **`EXEC`** from `LNX:LNXADMIN.` as your `PROFILE EXEC` on your local disk. To do so, use the following instructions:

1. Log on to `LNXS0031` from a 3270 terminal, as shown in Example 2-8.

Example 2-8 Log on to LNXS0031

```
LOGON LNXS0031
00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0003 RDR,   NO PRT,   NO PUN
00: LOGON AT 14:40:24 EDT WEDNESDAY 04/22/15
00: Command complete
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
DMSACC724I 19E replaces Y (19E)
DMSACP723I Y (19E) R/O
z/VM V6.3.0    2015-04-09 09:04
DMSWSP100W Shared S-STAT not available
DMSWSP100W Shared Y-STAT not available
DMSVML2060I TCPMAINT 592 linked RR as 0592 file mode Z
LNXS0031   AT ITSOZVM1 VIA RSCS     2015-04-22 14:40:24 EDT      WEDNESDAY
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 300 (64988 4K pages of swap space)
DIAG swap disk defined at virtual address 301 (129980 4K pages of swap space)
```

```
Do you want to IPL Linux from minidisk 100? y/n
n
```

2. Overwrite the `PROFILE EXEC` on your **`LNX:LNXS0031.`** using the `PROFCP EXEC` file from **`LNX:LNXADMIN.UBUNTU`**:

```
===> copyfile proffcp exec d profile exec a (replace
```

3. Edit the `profile exec` at **`LNX:LNXS0031`** and modify the REXX **`EXEC`** variables `STOWWPN`, `LKUPNUM`, and `IPLUNIT` according to your Linux system, as shown in Example 2-9.

Example 2-9 Edit and modify the profile variables

```
===> xedit profile exec a

/* PROFILE EXEC FOR LINUX VIRTUAL SERVERS -- MOD 2015-04-09  PWNOVAK */
/* BOOTING FROM FBA/FCP/SCSI DISKS                                   */
/*********************************************************************/
/* --- MODIFY STORAGE WWPN, LUN, IPLU VARS FOR EACH LINUX VSM ------- */
      STOWWPN = '50050763 0500C74C'        /* 8 CHAR + SPACE + 8 CHAR */
      LKUPNUM = '40104018 00000000'        /* 8 CHAR + SPACE + 8 CHAR */
      IPLUNIT = 'FC00'                     /* IPL UNIT ADDRESS        */
/****************** NO CHANGES BELOW THIS LINE ***********************/
'CP SP CONS TO LNXADMIN START NAME 'USERID()' CONSLOG'  /* CONSLOG ON */
'CP SET RUN ON'                                  /* RUN DISCONNECTED */
'CP SET PF11 RETRIEVE FORWARD'                   /* RETRIEVE CMD FWD */
'CP SET PF12 RETRIEVE'                           /* RETRIEVE CMD BKW */
'IDENTIFY (ISODATE'                              /* IDENTIFY VSM ID */
'PIPE CP QUERY' USERID() '| VAR USER'            /* DETERMINE USERID */
 PARSE VALUE USER WITH ID . DSC .                /* CHECK IF DISCOED */

 IF ( ID <> 'LNXADMIN' ) THEN              /* IF USER IS NOT LNXADMIN */
  DO                                       /* IF USER IS NOT LNXADMIN */
   'ACCESS LNX:LNXADMIN. D'                /* IF USER IS NOT LNXADMIN */
  END                                      /* IF USER IS NOT LNXADMIN */

'SWAPGEN 0300 0524288'             /* MAKE 256M LNXSWAP VDISK AT 0300 */
'SWAPGEN 0301 1048576'             /* MAKE 512M LNXSWAP VDISK AT 0301 */
IF (DSC = 'DSC') THEN                     /* IF USER IS DISCONNECTED */
  DO
   'CP QUERY TERMINAL'
   'CP QUERY CONSOLE'
   'CP SPOOL CONSOLE STOP'                                /* CONSLOG OFF */
   'CP SET LOADDEV CLEAR PORT 'STOWWPN' LUN 'LKUPNUM   /* SET FCP VAR */
   'CP QUERY LOADDEV'                                     /* Q LOADDEV   */
   'CP IPL FC00'                                          /* BOOT LINUX  */
  END
 ELSE                                /* USER IS INTERACTIVE SO PROMPT */
  DO
   SAY 'DO YOU WANT TO IPL LINUX FROM 'IPLUNIT' AS'
   SAY 'STORAGE WWPN 'STOWWPN ' AT LOOKUP NUMBER 'LKUPNUM'? Y/N'
   PARSE UPPER PULL ANSWER .
   IF (ANSWER = 'Y') THEN
    DO
     'CP QUERY TERMINAL'
     'CP QUERY CONSOLE'
     'CP SPOOL CONSOLE STOP'                              /* CONSLOG OFF */
```

```
    'CP SET LOADDEV CLEAR PORT 'STOWWPN' LUN 'LKUPNUM /* SET FCP VAR */
    'CP QUERY LOADDEV'                               /* Q LOADDEV   */
    'CP IPL FC00'                                    /* BOOT LINUX  */
   END
  'VMLINK TCPMAINT 592 < 592 T RR > ( NONAMES'   /* ACCESS TCP TOOLS */
 END
```

4. Verify the changes to the `PROFILE EXEC` REXX **`EXEC`** file, as shown in Example 2-10.

Example 2-10 Verify the changes to the file

```
===> ipl cms
DMSACC724I 19E replaces Y (19E)
DMSACP723I Y (19E) R/O
z/VM V6.3.0    2015-04-09 09:04

DMSWSP100W Shared S-STAT not available
DMSWSP100W Shared Y-STAT not available
LNXS0031   AT ITSOZVM1 VIA RSCS     2015-04-22 18:30:12 EDT      WEDNESDAY
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 0300 (64988 4K pages of swap space)
DIAG swap disk defined at virtual address 0301 (129980 4K pages of swap space)
DO YOU WANT TO IPL LINUX FROM FC00 AS
STORAGE WWPN 50050763 0500C74C  AT LOOKUP NUMBER 40104018 00000000? Y/N
Y

00: LINEND  # , LINEDEL OFF, CHARDEL OFF, ESCAPE  " , TABCHAR OFF
00: LINESIZE 080, ATTN OFF, APL OFF, TEXT OFF, MODE VM, HILIGHT OFF
00: CONMODE 3215, BREAKIN   IMMED , BRKKEY PA1  , SCRNSAVE OFF
00: AUTOCR ON , MORE 050 010, HOLD OFF, TIMESTAMP OFF, SYS3270 OFF
00: CONS 0009 ON LDEV L0005   TERM START HOST TCP/IP    FROM 9.12.5.134
00:      0009 CL T NOCONT NOHOLD COPY 001    READY FORM STANDARD
00:      0009 TO LNXADMIN RDR DIST LNXS0031    FLASHC 000 DEST OFF
00:      0009 FLASH       CHAR       MDFY       0 FCB       LPP OFF
00:      0009 3215   NOEOF OPEN 0297 NOKEEP NOMSG   NAME LNXS0031   CONSLOG
00:      0009 SUBCHANNEL = 0003
PORTNAME 50050763 0500C74C    LUN  40104018 00000000    BOOTPROG 0
BR_LBA   00000000 00000000
00: HCPLDI2816I Acquiring the machine loader from the processor controller.
00: HCPLDI2817I Load completed from the processor controller.
00: HCPLDI2817I Now starting the machine loader.
...

Ubuntu Server Server 16.04 LTS (Xenial Xerus)
...

LNXS0031 login:
```

2.4 Preseeding

Preseeding enables you to create a file that provides answers automatically while running an install. It enables you to automate your installation. For more information about preseeding an Ubuntu installation on z Systems environments, see the following website:

https://wiki.ubuntu.com/S390X/InstallationGuide/AutomatedInstallsWithPreseed

2.5 Cloning

Cloning has been one of the hallmarks of running Linux virtual servers under the z/VM Hypervisor for many years. In the context of this book, the term *cloning* refers to the creation of one or more new Linux virtual servers, which are identical copies of an existing *source master* virtual server system. The source master is also frequently referred to as a *golden image*.

Over time, the Linux operating system is evolving to meet the demands and expectations of those users who rely on it for everything from mobile devices to premier enterprise servers. Although this evolution enabled new functionality and enabled progress, it also introduced inherently disruptive cloud technologies and brought about increasing complexity.

2.5.1 Cloning considerations

This section covers the special considerations that you must consider when you are evaluating whether cloning will be a part of your Linux deployment strategy.

Universally Unique Identifiers

Nearly all modern Linux distributions across virtually all hardware platforms have widespread and growing adoption of components and software that generate *Universally Unique Identifiers*, more commonly known as *UUIDs*. One of the many possible examples where UUIDs are in use is integrated system bus architectures, such as the *systemd* framework, described in Chapter 6, “Working with systemd” on page 95.

Consider the following aspects of UUIDs:

- The process of generating a UUID is typically done in such a way that the possibility of more than one system or component sharing the same UUID is highly improbable.
- On a cloned system, each UUID must be regenerated or otherwise altered from the value of the golden image.
- Each installed package, system component, software configuration, or other element that relies on a UUID presents special challenges. The challenges exist because you must have an in-depth knowledge of your golden image to know where a UUID is in use.
- Failure to update all UUIDs could cause unforeseen future consequences, such as data corruption or security exposures in the event of a hash collision.

Workload containers

As container solutions become more popular, the ability to create a highly standardized environment for applications and services is increasingly shifting toward using containers to accomplish this. Consider the following aspects of using containers:

- As a result of the container trend, much of the complexity surrounding deployments shifts from the operating system level to the container level instead.

- The requirement to clone an entire virtual server to manage the complexity of creating a bespoken environment required to install certain software might no longer be necessary.
- Container solutions, such as Docker, rely on kernel cgroup and namespacing via Linux Containers (LXC), enabling much more flexibility for the types of work that can be combined in one virtual server.
- Kernel support for cgroup namespaces is likely to keep expanding, which is likely to result in connectivity across multiple systems using systemd. Should this become a reality, the UUID hash collision mentioned previously becomes much more of a focus item.

2.5.2 The cloning process

This section describes an example of the cloning process. A basic cleanup procedure has been supplied on a best-effort basis. The cleanup shell script should be reviewed by the user, and it requires updates whenever a more specialized golden image is prepared.

After the cleanup, the configuration update of the target system has been prepared as an example in this section. Depending on the services used, this also needs updates from the administrator, and should be reviewed in the first place.

2.6 Create golden image for cloning

A golden image is a special type of Linux installation that is used as a baseline to generate new Linux systems. Special care must be taken to prepare the golden image. All UUIDs must be updated for the resulting new Linux system, known as the cloned system.

Important: Before starting, be sure that you have the files that are associated with this book available. For information about how to obtain these files, refer to Appendix B, “Additional material” on page 143.

The user name of the z/VM guest in this book is assumed to be `GOLD`. Be sure to make all needed changes that you want to have for your template system before proceeding.

To prepare the Linux system as the golden image:

1. Copy the prepare script to the system. The script removes files specific to the one installation:

   ```
   # scp prepare_system.sh golden.itso.ibm.com:/usr/local/bin
   ```

2. Log on to the golden image and make sure that the following scripts from `/usr/local/bin` are executable:

   ```
   # ssh golden.itso.ibm.com
   # chmod 755 /usr/local/bin/*
   ```

3. Shut down the image by calling the **`prepare_system.sh`** script, as shown in Example 2-11.

   Example 2-11 Shut down the image

   ```
   # /usr/local/bin/prepare_system.sh

   Warning: This utility will prepare the root disk for cloning by removing some
            configuration files. After it has finished it will halt this systemd.

   press Ctrl+C within the next 15 seconds to abort
   ```

```
ln -s '/etc/systemd/system/clone.service'
'/etc/systemd/system/network.target.wants/clone.service'
Removing SSH keys
Removing the network configuration
Removing this script (/usr/local/bin/prepare_system.sh)
halt the system
```

4. Make a backup of the image.

2.7 Clone the golden image using DirMaint

To copy the golden image to a new user, **dirmaint** is used.

Make sure that the needed resources, such as disks are already prepared. The needed tasks are described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

To clone the golden image to a user with user name LNXSO016:

1. Log off of the source system (golden).
2. Log on as MAINT.
3. If the z/VM guest hasn't been created yet, use **dirmaint** to do so:

   ===> dirmaint add LNXSO016 like LNXPROTO pw <NewPassword>

4. Ensure that the source and target systems are logged off, as shown in Example 2-12.

Example 2-12 Log off the source and target systems

```
===> query LNXSO015 #query LNXSO016
query LNXSO015
query LNXSO016
HCPCQU045E LNXSO015 not logged on
Ready(00045); T=0.01/0.01 10:11:14
HCPCQU045E LNXSO016 not logged on
Ready(00045); T=0.01/0.01 10:11:14
```

5. Copy the disk 0100 from the source to the target system, as shown in Example 2-13.

Example 2-13 Copy the disk from the source to the target system

```
===> dirmaint for LNXSO016 clonedisk 0100 LNXSO015 0100
dirmaint for LNXSO016 clonedisk 0100 LNXSO015 0100
DVHXMT1191I Your CLONEDISK request has been sent for processing to
DVHXMT1191I DIRMAINT at ITSOZVM1 via DIRMSAT2.
Ready; T=0.01/0.01 08:41:24
 DVHREQ2288I Your CLONEDISK request for LNXSO016 at
 DVHREQ2288I * has been accepted.
...
 DVHBIU3428I Changes made to directory entry
 DVHBIU3428I DATAMOV2 have been placed online.
 DVHSHN3430I CLONEDISK operation for LNXSO016
 DVHSHN3430I address 0100 has finished (WUCF DVHSHN3430I 01085901).
```

6. Log off MAINT.

2.8 Send a configuration update to the clone system

Before being operational, the system needs some information about its configuration, which is achieved by sending a configuration script to the READER of the target system, LNXSO016. For Ubuntu, it is called **sndcfgub.sh**.

To send a configuration:

1. Create a small configuration file similar to that shown in Example 2-14.

   Example 2-14 Configuration file to update network configuration of LNXS0016

   ```
   GENERATE_MACHINEID="1"
   Hostname="LNXSO016"
   HostIP="9.60.7.103/20"
   Gateway="9.12.4.1"
   ReadChannel=0.0.0600
   WriteChannel=0.0.0601
   DataChannel=0.0.0602
   Domain=itso.ibm.com
   Nameserver=9.12.6.6
   Layer2=1
   ```

2. Log in as root on LNXADMIN.
3. Copy the **sndcfgub.sh.sh** shell script to /usr/local/bin:

   ```
   # cp sndcfgub.sh /usr/local/bin
   ```

4. Make sure that the script is executable:

   ```
   # chmod +x /usr/local/bin/send_config*
   ```

5. Create a configuration (cnf) subdirectory to /root:

   ```
   # cd /root
   # mkdir /root/cnf
   ```

6. Create and adapt the additional configuration script, as shown in Example 2-14. The filename must match the USERID of the cloned system:

   ```
   # vi /root/cnf/LNXSO016
   ```

7. Run the **send_config** script with the target USERID as an argument:

   Ubuntu: `# sndcfgub.sh LNXSO016`

An update script is sent to the reader of LNXSO016. This process has the following characteristics:

- The file name sent to the reader is CLONE.
- The file mode sent to the reader is SYSCTL.
- The file must be sent by LNXADMIN.
- The lines of the script must not exceed 80 characters.

The script sent to LNXSO016 is then run during the first boot up.

2.9 Start the clone system

To activate the clone system, just start it. You can log on or, if the system is set up with a common `PROFILE EXEC` that boots the system disk, you can log on as a remote user with `XAUTOLOG`:

```
===> ipl 100
```

During the IPL, the systemd **`clone.service`** reads and runs the **`clone.sysclt`** from the reader.

The **`clone.sysctl`** script contains the information that is needed to finalize the target system.

Finally, reboot the new system that has undergone IPL to make sure that the new `machine-id` is used consistently with the cloned system.

3

Servicing Linux: Updates and patching

This chapter describes the use of the package manager to perform routine service for Ubuntu Linux virtual servers. Essential tasks, such as installation, upgrades, and removals of software, are covered by using examples.

It includes the following topics:

- Package management
- Canonical Landscape

3.1 Package management

A derivative of the package manager that is used by the Debian GNU and Linux distribution, the Ubuntu package management facility has a base of over 45,000 software packages and several different options for use.

3.1.1 Cross-architecture consistency

Often times, a misconception exists that Linux on one architectural platform is somehow drastically different from another. The truth is, the modern 64-bit Linux code base is virtually identical across the different platforms, whether they be x86_64, s390x, or arm. In sections where there are variations, integration and testing of the code is done in a way that helps assure maximum consistency.

When the first enterprise Linux distributions for IBM mainframes came about, a new concept was needed to provide updates and patches in a defined and predictable way. The following overall principles were largely agreed upon by most distributions at that time:

- Corrections for code errors should be done in the same package version that is distributed as the initial, general availability (GA) level.
- The product management must decide if an update is truly required, and only then perform it.
- Programming interfaces must remain stable among major releases.
- The source code for all architectures must be consistent, meaning that a package that fails quality assurance (QA) or integration testing on *any architecture* results in blocking the release of that same patch for *all other architectures*.

3.1.2 The dpkg manager and Advanced Package Tool

Linux distributions that are built around the dpkg and Advanced Package Tool (APT) framework are different by design. Unlike some others, every component of the system is part of a package, which includes everything from core fundamentals, such as the Linux kernel itself, all the way through to lightweight tools and utilities.

Solid package management means a reliable system inventory and configuration management experience that is highly functional and consistent.

The process that is used to apply fixes on Ubuntu Linux is different from most other commercial Linux distributions because it uses Advanced Package Tool (APT). APT calculates dependencies, automatically resolves them, and then retrieves the required packages. From this point, APT then begins a transaction with the actual package management system, dpkg, commonly referred to as *d-package*.

3.1.3 Determining which packages are installed

The commands that are shown in this section can be run by any user. They do not require any elevated privileges. So you do not need to use sudo:

- To produce an alphabetically sorted list of the packages that are installed on a particular Ubuntu Linux system, issue the following command:

```
$ apt --installed list
```

Figure 3-1 shows the first 10 lines of output from this command.

```
support@lnxadmin:~$ apt --installed list | head -10

WARNING: apt does not have a stable CLI interface. Use with caution in
scripts.

Listing...
accountsservice/xenial-updates,now 0.6.40-2ubuntu11.1 s390x
[installed,automatic]
acl/xenial,now 2.2.52-3 s390x [installed]
acpid/xenial,now 1:2.0.26-1ubuntu2 s390x [installed]
adduser/xenial,now 3.113+nmu3ubuntu4 all [installed]
apparmor/xenial,now 2.10.95-0ubuntu2 s390x [installed]
apport/xenial,now 2.20.1-0ubuntu2 all [installed]
apport-symptoms/xenial,now 0.20 all [installed]
apt/xenial,now 1.2.10ubuntu1 s390x [installed]
apt-transport-https/xenial,now 1.2.10ubuntu1 s390x [installed]
```

Figure 3-1 The first 10 lines of terminal output from the apt --installed list command

- To determine whether a specific package is installed, such as *s390-tools*, use one of the following commands, as shown in Figure 3-2:
  - For status only, use the following command:

    ```
    $ dpkg --get-selections | grep -i s390-tools
    ```

  - For additional information, including version, architecture, and description, run the following command:

    ```
    $ dpkg -l | grep -i s390-tools
    ```

```
support@lnxadmin:~$ dpkg --get-selections | grep -i s390-tools
s390-tools                  install

support@lnxadmin:~$ dpkg -l | grep -i s390-tools
ii  s390-tools               1.34.0-0ubuntu8      s390x        fundamental
utilities for Linux on z Systems
```

Figure 3-2 Terminal view showing both options for determining package status

- To receive verbose information about a package, including installation status, run the following command (Figure 3-3):

  ```
  $ dpkg -s vim
  ```

```
support@lnxadmin:~$ dpkg -s vim
Package: vim
Status: install ok installed
Priority: optional
Section: editors
Installed-Size: 2563
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: s390x
Version: 2:7.4.1689-3ubuntu1
Provides: editor
Depends: vim-common (= 2:7.4.1689-3ubuntu1), vim-runtime (=
2:7.4.1689-3ubuntu1), libacl1 (>= 2.2.51-8), libc6 (>= 2.15), libgpm2 (>=
1.20.4), libselinux1 (>= 1.32), libtinfo5 (>= 6)
Suggests: ctags, vim-doc, vim-scripts
Description: Vi IMproved - enhanced vi editor
 Vim is an almost compatible version of the UNIX editor Vi.
 .
 Many new features have been added: multi level undo, syntax highlighting,
command line history, on-line help, filename completion, block operations,
folding, Unicode support, etc.
 .
 This package contains a version of vim compiled with a rather standard set
of features.  This package does not provide a GUI version of Vim.  See the
other vim-* packages if you need more (or less).
Homepage: http://www.vim.org/
Original-Maintainer: Debian Vim Maintainers
<pkg-vim-maintainers@lists.alioth.debian.org>
```

Figure 3-3 Detailed results of installed package vim

3.1.4 Determining release or version numbers and codenames

There are several different reasons that you might find yourself wanting to determine the release, version, and codename of a particular system:

- There might be a situation where you want to install a specific package and find that a repository or mirror that you want to source from is organized by using the codenames for releases.
- As your Ubuntu workloads expand and grow, you might have different virtual servers running different releases. Are you about to install or upgrade on the wrong system?

If you need to determine what the version number and codename is for a particular system, one or more of the following commands can help you quickly find what you need to know. None of these require elevated privileges. So the use of sudo is not required.

```
uname -vp
cat /etc/issue.net (or /etc/issue)
cat /etc/os-release
cat /etc/motd
```

Example 3-1 determines that the current system is running 12.04.5 LTS, code named **Precise Pangolin**. Also notice from the MOTD that it is time to upgrade this system.

Example 3-1 Terminal output from version identification commands

```
support@lnxadmin:~$ uname -vp
#139~precise1-Ubuntu Wed Jun 29 21:30:37 UTC 2016 s390x

support@lnxadmin:~$ cat /etc/issue.net
Ubuntu 12.04.5 LTS

support@lnxadmin:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="12.04.5 LTS, Precise Pangolin"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu precise (12.04.5 LTS)"
VERSION_ID="12.04"

support@lnxadmin:~$ cat /etc/motd
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-92-generic s390x)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2017.
```

3.2 Canonical Landscape

For businesses that run many systems, having a management framework in place is a must. *Landscape*, the Ubuntu systems management framework, covers desktop, server, and public cloud deployments. Landscape also enables you to build and manage private cloud deployments that are OpenStack enabled. It is a cost-effective method to support a large or growing Ubuntu footprint through automation and intelligent monitoring.

Highlights of the Landscape service include alerting, automation, and patching for up to 40,000 systems from one instance of Landscape. An API is also available to provide further customization and integration.

Landscape is available as either a stand-alone product, or in software as a service (SaaS) form. Commercial support is also available directly from Canonical as part of the *Ubuntu Advantage* package.

Landscape at a glance

Consider the following important points regarding the Canonical Landscape:

- Manage up to 40,000 desktop, server, and cloud systems from one instance.
- Create custom profiles and policies for different system types and classifications.
- Package management: Install, update, downgrade, and uninstall.
- IBM Insight™ for security and compliance through policies for automated updates, security patches, and custom reports.
- Configure users and groups.
- Elimination of the requirement to produce a *golden image* or a *cloning master* preproduction system.

Why APIs

A developer who has the ability to craft bespoken systems of engagement through the combination of multiple APIs might be a glimpse into the future of overall systems management across all platforms and architectures. Symmetry with other infrastructure components in your enterprise, such as IBM Wave for z/VM, means virtually limitless possibilities as APIs continue to grow in popularity and capabilities.

3.2.1 Setup

Setup of a z/VM Virtual Server for Landscape is a simple process. Simply run the following commands:

```
sudo add-apt-repository ppa:landscape/16.03
sudo apt update
sudo apt install landscape-server-quickstart
```

For complete details about the free trial offer for Landscape, detailed system resource requirements, and Ubuntu Advantage, refer to:

https://landscape.canonical.com/try-landscape
https://www.ubuntu.com/management/ubuntu-advantage

Part 2

Other topics

This part of the book includes the following chapters:

- Chapter 4, “Working with disks” on page 57
- Chapter 5, “Workload containers and service orchestration” on page 77
- Chapter 6, “Working with systemd” on page 95
- Chapter 7, “Miscellaneous recipes” on page 113

4

Working with disks

“An ounce of application is worth a ton of abstraction.”

— Booker T. Washington

This chapter details the processes for working with the two different types of disks available in a z/VM environment, direct access storage device (DASD) *extended count key data* (IBM ECKD™), and Fibre Channel Protocol/Small Computer System Interface (FCP/SCSI). It concentrates on the tasks that are commonly performed on Linux.

It includes the following topics:

- Add disk space to virtual machines
- Add a logical volume
- Extend an existing logical volume
- Move a physical volume

For the z/VM perspective, see *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

4.1 Add disk space to virtual machines

This section describes how to add more disk space to a Linux virtual machine. This disk space could come from different types of disks. The respective types are described in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

Tip: If you add minidisks to the user directory for a certain virtual machine, it is possible to attach them to a running Linux system without "bouncing" it (that is, shut down Linux, log off z/VM, log on to z/VM, and then boot Linux).

For example, if you added new minidisks at virtual addresses `0200 and 0200`, you can use the following commands to link to the disk:

```
# vmcp 'link * 200 200 mr'
# vmcp 'link * 201 201 mr'
```

4.1.1 Make new minidisks or ECKD DASD available in Linux

After getting new minidisks or ECKD DASD at addresses 0.0.0200 and 0.0.0201 accessible to your virtual server, make the new disks available by completing the following steps:

1. Remove the disks from the device driver blacklist (that is, make the disks visible) with the command **`cio_ignore`**:

   ```
   # cio_ignore -r 200-201
   ```

2. Use the **`chzdev`** command to enable minidisks 0200 and 0201:

   ```
   # chzdev -e 200-201
   ECKD DASD 0.0.0200 configured
   ECKD DASD 0.0.0201 configured
   ```

3. Use the **`lsdasd`** command to view the available disks:

   ```
   # lsdasd
   Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
   ==============================================================================
   0.0.0100   active      dasda     94:0    ECKD  4096   7042MB    1802880
   0.0.0200   n/f         dasdb     94:4    ECKD
   0.0.0201   n/f         dasdc     94:8    ECKD
   ```

   A status of `n/f` indicates *not formatted*, meaning the volumes have not been formatted for Linux.

4. Format the disks in parallel with the **`dasdfmt`** command (running in the background) using a bash **`for`** loop, as shown in Example 4-1.

   Example 4-1 Format the disks in parallel with the dasdfmt command

   ```
   # for DISK in b c
   > do
   >   dasdfmt -b 4096 -d cdl -y -f /dev/dasd$DISK &
   > done
   [1] 1230
   [2] 1231
   # Finished formatting the device.
   Rereading the partition table... ok
   Finished formatting the device.
   Rereading the partition table... ok
   ```

```
[1]-  Done                    dasdfmt -b 4096 -d cdl -y -f /dev/dasd$i
[2]+  Done                    dasdfmt -b 4096 -d cdl -y -f /dev/dasd$i
```

5. View the available disks again with the **lsdasd** command to see that they are formatted and that each is about 7 GB, as shown in Example 4-2.

Example 4-2 Output from the lsdasd command after formatting

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0100   active      dasda     94:0    ECKD  4096   7042MB    1802880
0.0.0200   active      dasdb     94:4    ECKD  4096   7042MB    1802880
0.0.0201   active      dasdc     94:8    ECKD  4096   7042MB    1802880
```

6. Create one partition on each of the disks using a bash **for** loop and the **fdasd -a** command, as shown in Example 4-3.

Example 4-3 Using a bash for loop and the fdasd -a command

```
# for DISK in b c
> do
>   fdasd -a /dev/dasd$DISK
> done
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
reading volume label ..: VOL1
reading vtoc ..........: ok

auto-creating one partition for the whole disk...
writing volume label...
writing VTOC...
rereading partition table...
```

The two new minidisks are now low-level formatted, partitioned, and configured to be active at boot time:

- If you are creating a new logical volume, see 4.2.1, “Create a logical volume and file system” on page 63.
- If you are extending an existing logical volume, skip to 4.3, “Extend an existing logical volume” on page 68.

4.1.2 Make a new EDEV available to Linux

After getting the new EDEV at address 0.0.0150 dedicated to your system, the procedure to integrate them into Linux is similar to what was done with ECKD DASD. The main difference is that you use other tools to format and partition these disks.

Complete the following steps:

1. Make the disk visible with the command **cio_ignore**:

   ```
   # cio_ignore -r 0150
   ```

2. Use the **chzdev** command to enable the DASD:

   ```
   # chzdev 0150 -e
   FBA DASD 0.0.0150 configured
   ```

3. Confirm the disk is now visible by issuing the **lsdasd** command to view the list of available disks:

   ```
   # lsdasd
   Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
   ==============================================================================
   0.0.0301   active      dasda     94:0    FBA   512    512MB     1048576
   0.0.0300   active      dasdb     94:4    FBA   512    256MB     524288
   0.0.0100   active      dasdc     94:8    ECKD  4096   3521MB    901440
   0.0.0101   active      dasdd     94:12   ECKD  4096   3521MB    901440
   0.0.0150   active      dasde     94:16   FBA   512    5120MB    10485760
   ```

4. Create a partition on the disk using the **parted** command. In the following example, we are using the device name ***dasde***, with a device path of ***/dev/dasde***:

   ```
   # parted -s /dev/dasde mklabel msdos mkpart primary 0% 100%
   ```

The new DASD is now partitioned and configured to be active at boot time:

- If you are creating a new logical volume, see 4.2.1, “Create a logical volume and file system” on page 63.
- If you are extending an existing logical volume, skip to 4.3, “Extend an existing logical volume” on page 68.

4.1.3 Make a new zFCP LUN available to Linux

Current distributions of Linux perform automatic LUN detection. Therefore, it is sufficient to just configure the host adapters and let Linux use the multipathed device for LUN discovery and disk configuration.

The steps outlined in this section assume that no previous zFCP was available.

In this example, you create two FCP adapters at the addresses ***0.0.fc00*** and ***0.0.fd00***. The necessary setup for z/VM is described in detail in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

Complete these steps:

1. Start a Secure Shell (SSH) session to the target system.
2. Check to see that the `zfcp` driver is loaded and available by issuing the **lszfcp** command. If you get an error stating “No `zfcp` support available,” use the **modprobe** command to load the `zfcp` module:

   ```
   # lszfcp
   Error: No zfcp support available.
   Load the zfcp module or compile
   the kernel with zfcp support.
   # modprobe zfcp
   # lszfcp
   Error: No fcp devices found.
   ```

3. Check that two FCP devices are available by using the **CP QUERY FCP** command, as shown in Example 4-4.

Example 4-4 Check the FCP devices using the CP QUERY FCP command

```
#  vmcp q v fcp
FCP  FC00 ON FCP   B602 CHPID 76 SUBCHANNEL = 0012
     FC00                        TOKEN      = 00000007F7A65300
     FC00 DEVTYPE FCP         VIRTUAL CHPID FF FCP REAL CHPID 76
     FC00 QDIO ACTIVE         QIOASSIST ACTIVE        QEBSM
     FC00
     FC00 INP + 01 IOCNT = 00000306  ADP = 128 PROG = 000 UNAVAIL = 000
     FC00          BYTES = 0000000000000000
     FC00 OUT + 01 IOCNT = 00000496  ADP = 000 PROG = 128 UNAVAIL = 000
     FC00          BYTES = 00000000004FC2C6
     FC00 DATA ROUTER ACTIVE
     WWPN C05076DD90002528
FCP  FD00 ON FCP   B702 CHPID 77 SUBCHANNEL = 0013
     FD00                        TOKEN      = 00000007F7A65480
     FD00 DEVTYPE FCP         VIRTUAL CHPID 77 FCP REAL CHPID 77
     FD00 QDIO ACTIVE         QIOASSIST ACTIVE        QEBSM
     FD00
     FD00 INP + 01 IOCNT = 00000337  ADP = 128 PROG = 000 UNAVAIL = 000
     FD00          BYTES = 0000000000000000
     FD00 OUT + 01 IOCNT = 00000494  ADP = 000 PROG = 128 UNAVAIL = 000
     FD00          BYTES = 00000000004FA2C6
     FD00 DATA ROUTER ACTIVE
     WWPN C05076DD90002748
```

4. Make the disks visible with the **cio_ignore** command:

```
# cio_ignore -r fc00
# cio_ignore -r fd00
```

5. Enable the FCP adapters with the **chzdev** command:

```
# chzdev fc00,fd00 -e
FCP device 0.0.fc00 configured
FCP device 0.0.fd00 configured
```

6. Now that the FCP adapters are enabled, they show up as hosts when you issue the **lszfcp** command:

```
# lszfcp
0.0.fc00 host0
0.0.fd00 host1
```

7. Verify that the auto LUN scan feature detected all of the paths to the LUNs by issuing the **lszluns** command, as shown in Example 4-5.

Example 4-5 Verify that all paths to the LUNs are detected

```
# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507680120bb91:
                0x0000000000000000
        at port 0x500507680120bc24:
                0x0000000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
```

```
        at port 0x500507680130bc24:
                0x0000000000000000
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507680120bb91:
                0x0000000000000000
        at port 0x500507680120bc24:
                0x0000000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
        at port 0x500507680130bc24:
                0x0000000000000000
```

8. Set up a multipath configuration, if not already configured:

   a. Make sure `multipath-tools` is installed with the following **`apt-get`** command:

   ```
   # apt-get install multipath-tools
   ```

   b. Although multipath can run without `/etc/multipath.conf`, it is useful to create one and set the option to enable “user friendly names” (for example, `/dev/mapper/mpatha`). Use the **`vi`** command, and create `/etc/multipath.conf` with the following contents:

   ```
   defaults {
          user_friendly_names     yes
   }
   ```

   c. Enable and start the multipath daemon:

   ```
   # systemctl enable multipathd
   # systemctl start multipathd
   ```

   d. Create a partition on the disk using the **`parted`** command:

   ```
   # parted -s /dev/mapper/mpatha mklabel msdos mkpart primary 0% 100%
   ```

   e. Verify the paths are active using the **`multipath -ll`** command, as shown in Example 4-6.

   Example 4-6 Verify that the paths are active

```
# multipath -ll
mpatha (360050768018305e1200000000000013c) dm-4 IBM,2145
size=25G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 1:0:0:0 sde 8:64  active ready running
| |- 1:0:3:0 sdf 8:80  active ready running
| |- 0:0:0:0 sda 8:0   active ready running
| `- 0:0:1:0 sdb 8:16  active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 1:0:4:0 sdg 8:96  active ready running
  |- 1:0:5:0 sdh 8:112 active ready running
  |- 0:0:2:0 sdc 8:32  active ready running
  `- 0:0:3:0 sdd 8:48  active ready running
```

4.2 Add a logical volume

There are times when you require more disk space than a single direct access storage device (DASD) volume provides. For example, if you want to have a shared data directory, it needs to be of sufficient size to contain all of the users’ data. Another example is, if you need to create

a repository that will contain many files that will be served to other Linux virtual servers, it needs to have enough space to hold all of the files. When this is the case, you can use the Logical Volume Manager (LVM) to combine multiple DASD volumes into one logical volume.

This example creates a logical volume using the 0200 and 0201 disks that were just added to Linux. Because each of those disks has a capacity of about 7 GB, if this example repository requires 10 GB of space, we'll combine those two disks to create a logical volume with a capacity of about 14 GB.

The following sections describe creating a logical volume with additional DASD that have already been added to Linux virtual server. For information about adding disks to a z/VM virtual server, see *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

4.2.1 Create a logical volume and file system

The following overall steps are involved in creating a logical volume:

1. Create physical volumes from the newly added disks.
2. Create a single volume group from the physical volumes.
3. Create a single logical volume using free space in the volume group.
4. Make a file system from the logical volume.

Figure 4-1 shows a block diagram of the LVM.

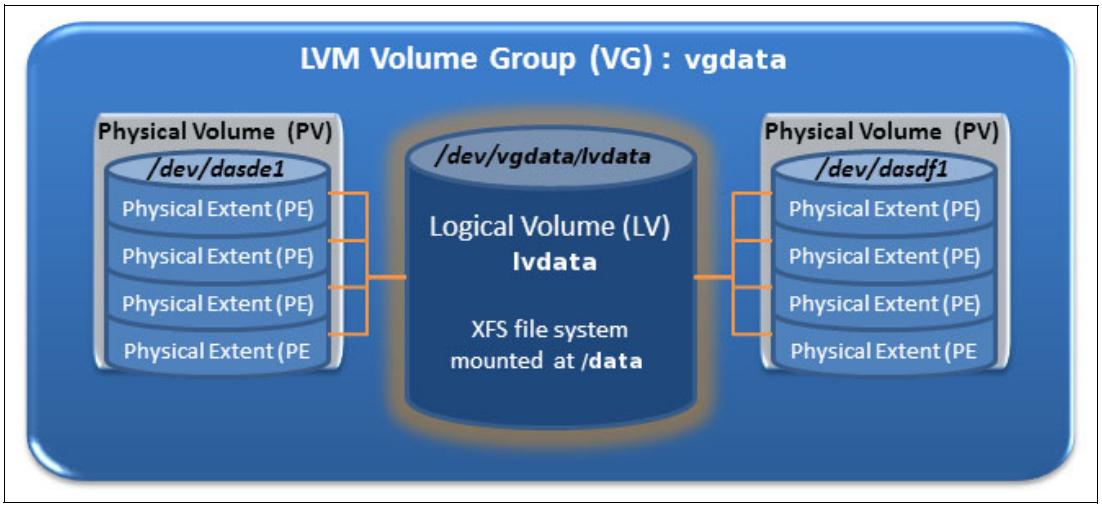


Figure 4-1 Volume Group block diagram

Create physical volumes from minidisks

To create physical volumes from the new minidisks at virtual device addresses ***0200*** and ***0201***, complete the following steps:

1. Check the devices on your system with the **`lsdasd`** command. This command shows you the details for all visible DASDs, including *device names* associated with a particular DASD address. In Example 4-7, the name for DASD `0200` is `dasdb` (`/dev/dasdb`), and the name for DASD `0201` is `dasdc` (`/dev/dasdc`).

Example 4-7 Checking the devices with the lsdasd command

```
# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
```

```
0.0.0100   active      dasda     94:0    ECKD  4096   7042MB    1802880
0.0.0200   active      dasdb     94:4    ECKD  4096   7042MB    1802880
0.0.0201   active      dasdc     94:8    ECKD  4096   7042MB    1802880
```

Note: The **lsdasd** command shows *devices* such as /dev/dasdb; however, *physical volumes* are created from numbered partitions such as /dev/dasdb1.

2. The **pvcreate** command initializes partitions for use as physical volumes by LVM. Initialize the partitions on DASD devices ***0200*** and ***0201*** as physical volumes:

```
# pvcreate /dev/dasdb1 /dev/dasdc1
  Physical volume "/dev/dasdb1" successfully created
  Physical volume "/dev/dasdc1" successfully created
```

3. Use the **pvdisplay** command to verify that the physical volumes were created. The output should look somewhat similar to Example 4-8.

Example 4-8 Sample output from the pvdisplay command

```
# pvdisplay /dev/dasdb1 /dev/dasdc1
  "/dev/dasdc1" is a new physical volume of "6.88 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/dasdc1
  VG Name
  PV Size               6.88 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               J8Vs3f-qZTS-ZIi2-opM1-AtQu-HSyl-KkZUC9

  "/dev/dasdb1" is a new physical volume of "6.88 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/dasdb1
  VG Name
  PV Size               6.88 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               i3GkTj-TPHx-gND6-lgAV-y6Bh-GdGd-XnHqBo
```

Create a single volume group

To create a volume group from the two newly created physical volumes, complete the following steps:

1. Use the **vgcreate** command to create a volume group named vgdata from the two newly created physical volumes:

```
# vgcreate vgdata /dev/dasdb1 /dev/dasdc1
  Volume group "vgdata" successfully created
```

2. Use the **vgdisplay** command to verify that the volume group was created. In Example 4-9, the volume group is 13.75 GB (VG Size) and has a total of 3520 physical

extents (Total PE). Of that total, 0 physical extents (0 GB) are currently allocated (Alloc PE / Size), and all 3520 physical extents (13.75 GB) are free (Free PE / Size).

Example 4-9 Output from vgdisplay command for the vgdata volume group

```
# vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       0 / 0
  Free  PE / Size       3520 / 13.75 GiB
  VG UUID               jmO2Oz-6k4e-NXSR-YOWV-R5EC-9DOt-De4EA4
```

Create a single logical volume

Use the **lvcreate** command to create a logical volume by allocating space from the free physical extents of a volume group. The **lvcreate** command has many options that won't be covered here (see the *man* pages for details); however, you need to specify the following basic options when creating a logical volume:

- The name of the new logical volume (**-n**).
- Amount of space to be allocated to the logical volume (**-l** or **-L**).
- The name of the volume group that will contain the logical volume.

Specifying the amount of space to be allocated for the new logical volume can be done in terms of *extents* (**-l**) or *size* (**-L**). The quantity of space represents the size of the new logical volume.

In this example, there are 3520 free extents in the volume group. So the maximum value that you can specify is **-l 3520**. Rather than specifying the number of extents, it can be expressed as a percentage of the total space in the volume group (**%VG**) or the free space in the volume group (**%FREE**). In this example, the volume group is completely empty, so if we want to give all of the space to the new logical volume, we can specify **-l 100%VG** or **-l 100%FREE**.

Because this example repository requires 10 GB of space, it might be easier to specify the size of the new logical volume rather than the number of extents. Size can be specified in different units, including bytes (B), kilobytes (K), megabytes (M) and gigabytes (G). In this example, we can specify **-L 10G** (10 gigabytes) as the size of the new logical volume.

To create a logical volume from free space in the newly created volume group, complete the following steps:

1. Use the **lvcreate** command to create the logical volume. The **-n lvdata** specifies the name of the new logical volume. We could specify **-L 10G** as the size of the logical volume,

but that would leave some unused free space (about 3.75 GB) in the volume group. Rather than leave unused space, we'll specify **-l 100%FREE** to use all of the free space in the volume group. The last argument **vgdata** specifies the name of the volume group from which the logical volume will be created:

```
# lvcreate -n lvdata -l 100%FREE vgdata
  Logical volume "lvdata" created.
```

2. Use the **lvdisplay** command to verify the logical volume is created and is the correct size, as shown in Example 4-10. When issuing the **lvdisplay** command, you must specify the full path name of the logical volume including the volume group name (that is, /dev/vgdata/lvdata), not just the logical volume name.

Example 4-10 Verify the logical volume is created and the correct size

```
# lvdisplay /dev/vgdata/lvdata
  --- Logical volume ---
  LV Path                /dev/vgdata/lvdata
  LV Name                lvdata
  VG Name                vgdata
  LV UUID                da9RcL-N5Qh-WWOt-A1vW-miJe-VaQh-SCIkq8
  LV Write Access        read/write
  LV Creation host, time lnxs0038, 2016-05-24 11:29:22 -0400
  LV Status              available
  # open                 0
  LV Size                13.75 GiB
  Current LE             3520
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           252:2
```

Optionally, you can use the **vgdisplay** command shown in Example 4-11 to look at the **vgdata** volume group and now you see that all of the space is allocated and that there is no free space available.

Example 4-11 The vgdisplay command showing that no free space is available

```
# vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       3520 / 13.75 GiB
```

```
Free  PE / Size       0 / 0
VG UUID               jmO2Oz-6k4e-NXSR-YOWV-R5EC-9DOt-De4EA4
```

Make a file system on the logical volume

Create a file system on the new logical volume. At the time that this book was written, XFS is the suggested file system for data. Use the following command to make the file system:

```
# mkfs.xfs /dev/vgdata/lvdata
...
```

The newly created file system on the logical volume is now ready to be mounted.

4.2.2 Update the file system table and mount the file system

At this point, you can mount the file system manually. However, if you add the entry to the file system table file, `/etc/fstab`, you can effectively test the change by using the **mount** command and specifying only the mount point.

To add the entry for the new file system, complete the following steps:

1. Make a backup copy of the `/etc/fstab` file:

   ```
   # cp -p /etc/fstab /etc/fstab_'date +%Y%m%d'.bak
   ```

2. Add one line to the `/etc/fstab` file:

   ```
   # vi /etc/fstab
   /dev/mapper/vgdata-lvdata  /data           xfs     defaults        0       2
   ...
   ```

3. Before mounting the file system, make sure that the mount point `/data` exists. Ideally, the directory should be completely empty:

   – If the directory does not exist, create it using the **mkdir** command:

     ```
     # mkdir /data
     ```

   – If the directory exists and there are already files in it, when the new file system is mounted over it, the contents of the directory are *hidden*.

   In this example, the directory already exists and it is empty:

   ```
   # ls -la /data
   total 0
   drwxr-xr-x  2 root root   6 May 14 13:02 .
   drwxr-xr-x 23 root root 247 May 14 13:02 ..
   ```

4. Mount the `new` file system by specifying only the mount point. By using just the mount point, the **mount** command gets the other information that it needs (for example, file system name, file system type) from `/etc/fstab`. This tests the new entry that we added to `/etc/fstab`. Example 4-12 shows mounting the `/data` file system, followed by the use of the **df -hT** command to verify that it is mounted correctly. In this case, also verify that it contains at least 10 GB available space.

   Example 4-12 Mounting file system /data

   ```
   # mount /data
   # df -hT
   Filesystem                Type      Size  Used Avail Use% Mounted on
   udev                      devtmpfs  391M     0  391M   0% /dev
   tmpfs                     tmpfs      80M  2.5M   77M   4% /run
   /dev/dasda2               xfs       2.4G  860M  1.5G  37% /
   ```

```
tmpfs                       tmpfs     397M     0  397M   0% /dev/shm
tmpfs                       tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs                       tmpfs     397M     0  397M   0% /sys/fs/cgroup
/dev/dasda1                 xfs       185M   27M  158M  15% /boot
/dev/mapper/vgsystem-lvvar xfs       946M  198M  749M  21% /var
/dev/mapper/vgsystem-lvtmp xfs       754M   33M  722M   5% /tmp
/dev/mapper/vgdata-lvdata  xfs        14G   33M   14G   1% /data
```

5. Reboot the system to verify that the logical volume and its new file system are automatically mounted following an IPL of the system:

   # **reboot**

   When the system comes back, you should see the file system on the new logical volume mounted at /data, similar to Example 4-13.

Example 4-13 Verifying the logical volume is mounted

```
# df -hT
Filesystem                 Type      Size  Used Avail Use% Mounted on
udev                       devtmpfs  391M     0  391M   0% /dev
tmpfs                      tmpfs      80M  1.5M   78M   2% /run
/dev/dasda2                xfs       2.4G  860M  1.5G  37% /
tmpfs                      tmpfs     397M     0  397M   0% /dev/shm
tmpfs                      tmpfs     5.0M     0  5.0M   0% /run/lock
tmpfs                      tmpfs     397M     0  397M   0% /sys/fs/cgroup
/dev/dasda1                xfs       185M   27M  158M  15% /boot
/dev/mapper/vgdata-lvdata  xfs        14G   33M   14G   1% /data
/dev/mapper/vgsystem-lvvar xfs       946M  198M  749M  21% /var
/dev/mapper/vgsystem-lvtmp xfs       754M   33M  722M   5% /tmp
```

4.3 Extend an existing logical volume

This section describes the process of adding a new minidisk to an existing logical volume. This is useful when your logical volume is running out of space. In this example, the output from the **df** command shows that the /data file system has used 95% of its space and there is less than 750 MB available:

```
# df -hT /data
Filesystem                Type  Size  Used Avail Use% Mounted on
/dev/mapper/vgdata-lvdata xfs    14G   14G  726M  95% /data
```

A minidisk at address 0202 was added to the Linux virtual server, and you completed the steps in 4.1.1, “Make new minidisks or ECKD DASD available in Linux” on page 58 to make the disk accessible, format it, and create a partition on it. In addition, you completed the steps that are described in “Create physical volumes from minidisks” on page 63 to prepare the physical volume for use.

To extend the logical volume using the physical volume that is created from the additional minidisk:

1. Use the **vgdisplay** command to determine the amount of free space is available in the volume group, as shown in Example 4-14.

Example 4-14 Determine the amount of free space

```
# vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       3520 / 13.75 GiB
  Free  PE / Size       0 / 0
  VG UUID               jmO2Oz-6k4e-NXSR-YOWV-R5EC-9DOt-De4EA4
```

The `Free PE / Size` in the output indicates that there is no free space in the volume group.

2. Use the **pvscan** command to display a list of all of the physical volumes:

```
# pvscan
  PV /dev/dasda3   VG vgsystem        lvm2 [4.36 GiB / 2.69 GiB free]
  PV /dev/dasdb1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdc1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdd1                      lvm2 [6.88 GiB]
  Total: 4 [24.99 GiB] / in use: 3 [18.11 GiB] / in no VG: 1 [6.88 GiB]
```

This shows that physical volume **/dev/dasdd1** is not associated with any volume group and has 6.88 GB of space.

3. Use the **vgextend** command to add the physical volume to the volume group:

```
# vgextend vgdata /dev/dasdd1
  Volume group "vgdata" successfully extended
```

4. Use the **vgdisplay** command to show the free space in the extended volume group, as shown in Example 4-15.

Example 4-15 Show the free space in the extended volume group

```
# vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        3
```

```
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               20.62 GiB
  PE Size               4.00 MiB
  Total PE              5280
  Alloc PE / Size       3520 / 13.75 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               jmO2Oz-6k4e-NXSR-YOWV-R5EC-9DOt-De4EA4
```

Now you can see that there are 1760 extents / 6.88 GB free in the volume group.

5. The **lvextend** command is used to extend a logical volume by adding space from the free physical extents of the volume group. Specifying the amount of space to be allocated for the extended logical volume can be done in terms of “extents” (**-l**) or “size” (**-L**). The quantity of space represents either the new size of the extended logical volume (for example, **-l 5280** or **-L 20G**), or the quantity of space to be added to the logical volume (for example, **-l +1760** or **-L +5G**).

   In this example, there are 1760 free extents in the volume group. So the maximum value you can specify is **-l 5280** (3520 current extents + 1760 free extents). Rather than specifying the number of extents, it can be expressed as a percentage of the total space in the volume group (**%VG**) or the free space in the volume group (**%FREE**).

   In this example, the volume group has 1760 free extents, so if we want to give all of the free space to the extended logical volume, we can specify **-l 100%VG** (that is, the new size of the logical volume is all of the extents in the volume group) or **-l +100%FREE** (that is, the current size of the logical volume is to be increased by the number of free extents in the volume group).

   Because this example repository now requires 20 GB of space. it might be easier to specify the new size of the extended logical volume rather than the number of extents. Size can be specified in different units, including bytes (B), kilobytes (K), megabytes (M) and gigabytes (G). In this example, we can specify **-L 20G** (20 gigabytes) as the new size of the extended logical volume, or **-L +5G** (5 gigabytes) as the amount of space to add to the logical volume.

   Use the **lvextend** command to extend the logical volume. If you specify **-L 20G** as the new size of the logical volume, some unused free space (about 630 MB) would remain in the volume group. Rather than leave unused space, specify **-l +100%FREE** to use all of the free extents in the volume group to extend the logical volume:

```
# lvextend -l +100%FREE /dev/vgdata/lvdata
  Size of logical volume vgdata/lvdata changed from 13.75 GiB (3520 extents) to
20.62 GiB (5280 extents).
  Logical volume lvdata successfully resized.
```

Optionally, you can use the **vgdisplay** command to look at the **vgdata** volume group, as shown in Example 4-16, and see that all of the space has been allocated and that there is no free space available.

Example 4-16 Using the vgdisplay command to see that no free space is available

```
# vgdisplay vgdata
  --- Volume group ---
  VG Name               vgdata
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               20.62 GiB
  PE Size               4.00 MiB
  Total PE              5280
  Alloc PE / Size       5280 / 20.62 GiB
  Free  PE / Size       0 / 0
  VG UUID               jmO2Oz-6k4e-NXSR-YOWV-R5EC-9DOt-De4EA4
```

6. Use the **lvdisplay** command to verify that the logical volume is extended and is the correct size, as shown in Example 4-17. When issuing the **lvdisplay** command, you must specify the full path name of the logical volume including the volume group name (that is, /dev/vgdata/lvdata), not just the logical volume name.

Example 4-17 Show that the logical volume is extended and is the correct size

```
# lvdisplay /dev/vgdata/lvdata
  --- Logical volume ---
  LV Path                /dev/vgdata/lvdata
  LV Name                lvdata
  VG Name                vgdata
  LV UUID                da9RcL-N5Qh-WWOt-A1vW-miJe-VaQh-SCIkq8
  LV Write Access        read/write
  LV Creation host, time lnxsO038, 2016-05-24 11:29:22 -0400
  LV Status              available
  # open                 1
  LV Size                20.62 GiB
  Current LE             5280
  Segments               3
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           252:2
```

7. Use the **xfs_growfs** command to increase the size of the XFS file system on the newly extended logical volume, as shown in Example 4-18. The file system must be mounted and the mount point of the file system is given as the argument to the **xfs_growfs** command.

Example 4-18 Increase the size of the XFS file system

```
# xfs_growfs /data
meta-data=/dev/mapper/vgdata-lvdata isize=512    agcount=4, agsize=901120 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=1        finobt=1 spinodes=0
data     =                       bsize=4096   blocks=3604480, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=2560, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 3604480 to 5406720
```

8. Use the **df** command to show the size of the file system on the extended logical volume:

```
# df -hT /data
Filesystem                Type  Size  Used Avail Use% Mounted on
/dev/mapper/vgdata-lvdata xfs    21G   14G  7.6G  64% /data
```

When compared to the output from the **df** command we issued before extending the logical volume (see 4.3, “Extend an existing logical volume” on page 68), the `/data` file system has grown from 14 GB to 21 GB, available space increased from 726 MB to 7.6 GB, and utilization of the space in the file system decreased from 95% to 64%.

4.4 Move a physical volume

This section shows how to move data from one physical volume to another without taking the file system offline.

In addition to file systems getting larger, you might need to move data off one or more volumes onto another or target set of volumes. If your data is in LVM, the **pvmove** and **vgreduce** commands were designed for this, and can be used with the file system online.

In this example, two minidisks have been added to the Linux virtual server at addresses 0400 and 0401. You completed the steps in 4.1.1, “Make new minidisks or ECKD DASD available in Linux” on page 58 to make the disks accessible, format them, and create a partition on each of them. In addition, you completed the steps discussed in “Create physical volumes from minidisks” on page 63 to prepare the physical volume for use.

These two physical volumes are `/dev/dasde1` and `/dev/dasdf1`. Use the **pvscan** command to display a list of all physical volumes:

```
# pvscan
  PV /dev/dasda3   VG vgsystem        lvm2 [4.36 GiB / 2.69 GiB free]
  PV /dev/dasdb1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdd1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdc1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdf1                      lvm2 [6.88 GiB]
  PV /dev/dasde1                      lvm2 [6.88 GiB]
  Total: 6 [38.74 GiB] / in use: 4 [24.98 GiB] / in no VG: 2 [13.75 GiB]
```

The output shows that physical volumes /dev/dasde1 and /dev/dasdf1 are not associated with any volume group and each one has 6.88 GB of space. Data is placed on the first volume, and later moved to the second. This is done while the file system is online.

To complete this test:

1. Create volume group vgtest from the first logical volume:

```
# vgcreate vgtest /dev/dasde1
  Volume group "vgtest" successfully created
```

2. Determine the number of total, allocated, and free physical extents in the new volume group:

```
# vgdisplay vgtest | grep -i -e "total" -e "alloc" -e "free"
  Total PE              1760
  Alloc PE / Size       0 / 0
  Free  PE / Size       1760 / 6.88 GiB
```

3. Create a logical volume named lvtest which uses all of the space in volume group vgtest:

```
# lvcreate -n lvtest -l 100%VG vgtest
  Logical volume "lvtest" created.
```

   Optionally, use the **vgdisplay** command again and see that all of the space has been allocated and that there is no free space available:

```
# vgdisplay vgtest | grep -i -e "total" -e "alloc" -e "free"
  Total PE              1760
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       0 / 0
```

4. Create an ext4 file system on the logical volume:

```
# mkfs.ext4 /dev/vgtest/lvtest
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 1802240 4k blocks and 450560 inodes
Filesystem UUID: b573c264-cd86-4563-9812-8fbce0d03f94
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

5. Create mount point /test, add the new file system to /etc/fstab, mount it, and check the space that it is using:

```
# mkdir /test
# cp /etc/fstab /etc/fstab.old
# vi /etc/fstab
/dev/mapper/vgtest-lvtest  /test           ext4    defaults        0       0
...
# mount /test
# df -hT /test
Filesystem                Type  Size  Used Avail Use% Mounted on
/dev/mapper/vgtest-lvtest ext4  6.7G   16M  6.3G   1% /test
```

6. Use the **dd** command to create a fairly large file on it with the **dd** command and recheck file system usage:

```
# dd if=/dev/zero of=/test/bigfile bs=1M count=2500
2500+0 records in
2500+0 records out
2621440000 bytes (2.6 GB, 2.4 GiB) copied, 15.4097 s, 170 MB/s
# df -hT /test
Filesystem               Type  Size  Used Avail Use% Mounted on
/dev/mapper/vgtest-lvtest ext4  6.7G  2.5G  3.9G  40% /test
```

7. Show the volume group usage with the **vgdisplay** command, as shown in Example 4-19.

Example 4-19 Show the volume group usage

```
# vgdisplay vgtest
  --- Volume group ---
  VG Name               vgtest
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               6.88 GiB
  PE Size               4.00 MiB
  Total PE              1760
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       0 / 0
  VG UUID               dze3cy-i2Hm-U2kA-Q2Dd-iVZW-i9HF-Wa26ML
```

This output shows that the volume group contains one physical volume. The size of the volume group is 6.88 GB, and all physical extents in the volume group are currently allocated.

8. Use the **pvscan** command to display a list of all physical volumes. Note that physical volume /dev/dasdf1 is not associated with any volume group and has 6.88 GB of space:

```
# pvscan
  PV /dev/dasde1   VG vgtest          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasda3   VG vgsystem        lvm2 [4.36 GiB / 2.69 GiB free]
  PV /dev/dasdb1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdd1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdc1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdf1                      lvm2 [6.88 GiB]
  Total: 6 [38.74 GiB] / in use: 5 [31.86 GiB] / in no VG: 1 [6.88 GiB]
```

9. Use the **vgextend** command to add physical volume /dev/dasdf1 (which is the target of the move) to volume group vgtest:

```
# vgextend vgtest /dev/dasdf1
  Volume group "vgtest" successfully extended
```

10. Use the **pvscan** command to look at all physical volumes that are associated with volume group `vgtest`. You see that physical volume `/dev/dasdf1` is now associated with the volume group. It still has 6.88 GB of free space because it isn't used yet:

```
# pvscan | grep vgtest
  PV /dev/dasde1   VG vgtest          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdf1   VG vgtest          lvm2 [6.88 GiB / 6.88 GiB free]
```

11. Use the **vgdisplay** command again to recheck volume group usage, as shown in Example 4-20.

Example 4-20 Recheck the volume group usage

```
# vgdisplay vgtest
  --- Volume group ---
  VG Name               vgtest
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  5
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               dze3cy-i2Hm-U2kA-Q2Dd-iVZW-i9HF-Wa26ML
```

The output shows that the volume group has doubled in size. It now contains two physical volumes, the size of the volume group is now 13.75 GB, and there are an equal number of allocated and free extents.

12. The **pvmove** command moves allocated physical extents from one source physical volume to free physical extents (PEs) on one or more other physical volumes (PVs) in the same volume group (VG). There are options to select specific PEs, select PEs only associated with a specific logical volume (LV), or select a destination PV (see the *man* pages for details).

In this example, move all of the allocated PEs from PV `/dev/dasde1` to free PEs on other PVs in the VG:

```
# pvmove /dev/dasde1
  /dev/dasde1: Moved: 0.1%
  /dev/dasde1: Moved: 20.9%
  /dev/dasde1: Moved: 42.0%
  /dev/dasde1: Moved: 63.0%
  /dev/dasde1: Moved: 84.1%
  /dev/dasde1: Moved: 100.0%
```

13. Use the **vgdisplay** command again to check volume group usage, as shown in Example 4-21.

Example 4-21 Check the volume group usage again

```
# vgdisplay vgtest
  --- Volume group ---
  VG Name               vgtest
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  8
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               13.75 GiB
  PE Size               4.00 MiB
  Total PE              3520
  Alloc PE / Size       1760 / 6.88 GiB
  Free  PE / Size       1760 / 6.88 GiB
  VG UUID               dze3cy-i2Hm-U2kA-Q2Dd-iVZW-i9HF-Wa26ML
```

As you expect, this output shows that the volume group is still the same size and there are still an equal number of allocated and free extents. The data has been moved between PVs, but is still in the same VG.

14. Use the **pvscan** command again to look at all PVs associated with VG **vgtest**. If you compare it to the output before using **pmove**, you will see that PV /dev/dasde1 now has 6.88 GB of free space and /dev/dasdf1 now has no free space:

```
# pvscan | grep vgtest
  PV /dev/dasde1   VG vgtest          lvm2 [6.88 GiB / 6.88 GiB free]
  PV /dev/dasdf1   VG vgtest          lvm2 [6.88 GiB / 0    free]
```

15. The **vgreduce** command removes one of more physical volumes from a volume group. Remove the empty source PV from the VG:

```
# vgreduce vgtest /dev/dasde1
  Removed "/dev/dasde1" from volume group "vgtest"
```

16. Use the pvscan command to see that /dev/dasde1 is no longer associated with any volume group:

```
# pvscan
  PV /dev/dasdf1   VG vgtest          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasda3   VG vgsystem        lvm2 [4.36 GiB / 2.69 GiB free]
  PV /dev/dasdb1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdd1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasdc1   VG vgdata          lvm2 [6.88 GiB / 0    free]
  PV /dev/dasde1                      lvm2 [6.88 GiB]
  Total: 6 [38.74 GiB] / in use: 5 [31.86 GiB] / in no VG: 1 [6.88 GiB]
```

Physical volume /dev/dasde1 is now ready for reassignment to another volume group, or if it is no longer needed to be a resource under the LVM, you can use the **pvremove** command to remove it.

5

Workload containers and service orchestration

“Be a yardstick of quality. Some people aren’t used to an environment where excellence is expected.”

— Steve Jobs

This chapter introduces the concept of running workloads in *containers*, using *Docker*. Additionally, we cover *service orchestration* using *JuJu*. It includes the following topics:

- Run workloads in containers
- Service orchestration

5.1 Run workloads in containers

As container solutions become more popular, creating a highly standardized environment for applications and services is increasingly shifting toward using these container solutions to accomplish this. Container solutions, such as Docker, which this section covers in detail, function much in the same way that z/VM does. Major components of the base operating system are accessed through shared memory segments. Incredible efficiency and high workload density result from not instantiating an entire copy of an OS for each container.

The following aspects are hallmarks of containers:

- *Portability*. Much of the complexity that surrounds deployments shifts from the operating system level to the container level instead.
- *Agility*. Having to clone an entire virtual server to manage the complexity of creating a bespoken environment that is required to install certain software might no longer be necessary.
- *Flexibility*. Container solutions, such as Docker, rely on kernel cgroup and namespacing via LXC, enabling much more flexibility for the types of work that can be combined in one virtual server
- *Scalable*. Deployment of scalable services with a secure and reliable pattern across a wide variety of platforms.

5.1.1 Introducing Docker

Docker container is an open source platform where distributed application workloads can be built, bundled, and run. Docker provides the following key benefits:

- Each container uses the same kernel as the host and the same file system. You can use Dockerfile with the **`docker build`** command to handle the provisioning and configuration of your container.
- Docker brings standardized control by enabling developers to own all of the code from the infrastructure level all the way down through the actual application level. This level of control paves the way for creation of a highly-standardized, scalable, and secured operating environment with a consistent user interface that meets many different needs.
- Docker provides language and platform mobility through composition of applications from microservices and allows freedom from inconsistencies within adjacent or even foreign environments.

5.1.2 Installing Docker

To install Docker:

1. On any system where you want to use Docker, use the commands shown in Example 5-1 to perform the installation.

   Installation note: This example uses **sudo** for the commands that require root privileges instead of **sudo su -** to root.

   Example 5-1 installing Docker

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
$ echo "deb https://apt.dockerproject.org/repo ubuntu-xenial main" | sudo tee
/etc/apt/sources.list.d/docker.list
$ sudo apt-get update
$ sudo apt-get purge lxc-docker
$ sudo apt-get install -y docker-engine
$ sudo apt-cache policy docker-engine
```

2. Configure **systemd** to auto-start the Docker service during system boot:

```
sudo systemctl enable docker
```

5.1.3 Updating Docker

To update Docker to the latest available version, issue the following commands:

```
sudo apt-get update
sudo apt-get upgrade docker-engine
```

5.1.4 Uninstalling Docker

If for some reason you want to uninstall Docker, issue the following commands:

```
sudo systemctl stop docker
sudo systemctl disable docker
sudo apt-get purge docker-engine
sudo apt-get autoremove docker-engine --purge docker-engine
```

Important: Configuration files and other data specific to Docker is stored under `/var/lib/docker` on the `/var` file system. You can *optionally* choose to remove these files as well *after carefully checking the entire path to make sure that there is nothing that you might still need*.

There is no undo for a forced, recursive deletion performed by the following command:

```
sudo rm -Rf /var/lib/docker
```

5.1.5 Docker Universal Control Plane and Docker Cloud

This section describes the Docker Universal Control Plane and the Docker Cloud.

Docker Universal Control Plane (UCP)

UCP is the enterprise cluster management solution from Docker. UCP is installed behind your firewall, and helps manage your whole cluster from a single interface. UCP also integrates with Docker Trusted Registry and Docker Content Trust. This is a key feature that makes it so attractive for enterprise use. This enables you to keep your Docker images stored behind your firewall, where they are under your control. It also enables image signing to ensure that the images that you deploy have not been altered.

For additional details about UCP, visit the following site:

https://docs.docker.com/ucp/installation/plan-production-install/

Docker Cloud

Docker Cloud is a SaaS offering that provides a Registry with build and test capabilities for:

- Images, Builds, and Testing
- Infrastructure management
- Services, Stacks, and Applications
- Deployment tools to help automate deployment of images to your infrastructure

You can find additional details about Docker Cloud at:

https://docs.docker.com/docker-cloud/overview/

5.2 Service orchestration

Juju, ICO, Puppet, Chef, Landscape, Cassandra, Fabric, and a host of other remote configuration and management tools are helpful for servers and can automate many different tasks.

5.2.1 Introducing Juju

Juju is the service orchestration solution from Canonical. Juju is supported to operate with a wide range of cloud providers, with the idea that it should be easy to model and then deploy any service you want into the cloud environment of your choice.

The most current release as of the time this book was authored is Juju 2.0. Juju 2.0 includes LXD support for local deployments, as well as co-located services on a cloud instance or physical machine.

This section focuses on the local use case, going through the experience of an LXD user without any pre-existing Juju experience.

5.2.2 Installing Juju 2.0

Example 5-2 demonstrates the installation of Juju 2.0 on a new Linux virtual server, LNXS0101.

Example 5-2 Commands used for the installation of Juju

```
support@lnxs0101:~$ sudo apt update
support@lnxs0101:~$ sudo apt upgrade
support@lnxs0101:~$ sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
support@lnxs0101:~$ sudo apt install juju zfsutils-linux
Reading package lists... Done
Building dependency tree
Reading state information... Done

The following additional packages will be installed:
  acl distro-info dns-root-data dnsmasq-base juju-2.0 liblxc1 liblzo2-2
  libnetfilter-conntrack3 libnvpair1linux libuutil1linux libzfs2linux
  libzpool2linux lxc-common lxcfs lxd lxd-client squashfs-tools uidmap
  xz-utils zfs-doc zfs-zed
Suggested packages:
  shunit2 juju-core criu lxd-tools default-mta | mail-transport-agent
  samba-common-bin nfs-kernel-server zfs-initramfs
The following NEW packages will be installed:
  acl distro-info dns-root-data dnsmasq-base juju juju-2.0 liblxc1 liblzo2-2
  libnetfilter-conntrack3 libnvpair1linux libuutil1linux libzfs2linux
  libzpool2linux lxc-common lxcfs lxd lxd-client squashfs-tools uidmap
  xz-utils zfs-doc zfs-zed zfsutils-linux
0 upgraded, 23 newly installed, 0 to remove and 0 not upgraded.
Need to get 49.6 MB of archives.
After this operation, 395 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Y

Get:1 http://ports.ubuntu.com xenial/main s390x acl s390x 2.2.52-3 [38.0 kB]
Get:2 http://ports.ubuntu.com xenial/main s390x libnetfilter-conntrack3 s390x
1.0.5-1 [34.9 kB]
Get:3 http://ports.ubuntu.com xenial-updates/main s390x dnsmasq-base s390x
2.75-1ubuntu0.16.04.1 [281 kB]
Get:4 http://ports.ubuntu.com xenial-updates/main s390x lxc-common s390x
2.0.3-0ubuntu1~ubuntu16.04.1 [27.2 kB]
Get:5 http://ports.ubuntu.com xenial-updates/main s390x liblxc1 s390x
2.0.3-0ubuntu1~ubuntu16.04.1 [184 kB]
Get:6 http://ports.ubuntu.com xenial-updates/main s390x lxcfs s390x
2.0.2-0ubuntu1~ubuntu16.04.1 [32.4 kB]
Get:7 http://ports.ubuntu.com xenial/main s390x liblzo2-2 s390x 2.08-1.2 [44.6 kB]
Get:8 http://ports.ubuntu.com xenial/main s390x squashfs-tools s390x
1:4.3-3ubuntu2 [106 kB]
Get:9 http://ports.ubuntu.com xenial/main s390x uidmap s390x 1:4.2-3.1ubuntu5
[64.8 kB]
Get:10 http://ports.ubuntu.com xenial/main s390x xz-utils s390x
5.1.1alpha+20120614-2ubuntu2 [78.4 kB]
Get:11 http://ports.ubuntu.com xenial-updates/main s390x lxd s390x
2.0.3-0ubuntu1~ubuntu16.04.2 [3908 kB]
Get:12 http://ports.ubuntu.com xenial/main s390x distro-info s390x 0.14build1
[19.8 kB]
```

```
Get:13 http://ports.ubuntu.com xenial/main s390x dns-root-data all 2015052300+h+1
[15.0 kB]
Get:14 http://ports.ubuntu.com xenial-updates/main s390x lxd-client s390x
2.0.3-0ubuntu1~ubuntu16.04.2 [1930 kB]
Get:15 http://ports.ubuntu.com xenial-updates/main s390x juju-2.0 s390x
2.0~beta7-0ubuntu1.16.04.1 [42.0 MB]
Get:16 http://ports.ubuntu.com xenial-updates/main s390x juju all
2.0~beta7-0ubuntu1.16.04.1 [9556 B]
Get:17 http://ports.ubuntu.com xenial-updates/main s390x zfs-doc all
0.6.5.6-0ubuntu10 [49.4 kB]
Get:18 http://ports.ubuntu.com xenial-updates/main s390x libuutil1linux s390x
0.6.5.6-0ubuntu10 [34.5 kB]
Get:19 http://ports.ubuntu.com xenial-updates/main s390x libnvpair1linux s390x
0.6.5.6-0ubuntu10 [22.1 kB]
Get:20 http://ports.ubuntu.com xenial-updates/main s390x libzpool2linux s390x
0.6.5.6-0ubuntu10 [360 kB]
Get:21 http://ports.ubuntu.com xenial-updates/main s390x libzfs2linux s390x
0.6.5.6-0ubuntu10 [97.2 kB]
Get:22 http://ports.ubuntu.com xenial-updates/main s390x zfsutils-linux s390x
0.6.5.6-0ubuntu10 [254 kB]
Get:23 http://ports.ubuntu.com xenial-updates/main s390x zfs-zed s390x
0.6.5.6-0ubuntu10 [29.0 kB]
Fetched 49.6 MB in 6s (7833 kB/s)
Preconfiguring packages ...
Selecting previously unselected package acl.
(Reading database ... 63684 files and directories currently installed.)
Preparing to unpack .../acl_2.2.52-3_s390x.deb ...
Unpacking acl (2.2.52-3) ...
Selecting previously unselected package libnetfilter-conntrack3:s390x.
Preparing to unpack .../libnetfilter-conntrack3_1.0.5-1_s390x.deb ...
Unpacking libnetfilter-conntrack3:s390x (1.0.5-1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../dnsmasq-base_2.75-1ubuntu0.16.04.1_s390x.deb ...
Unpacking dnsmasq-base (2.75-1ubuntu0.16.04.1) ...
Selecting previously unselected package lxc-common.
Preparing to unpack .../lxc-common_2.0.3-0ubuntu1~ubuntu16.04.1_s390x.deb ...
Unpacking lxc-common (2.0.3-0ubuntu1~ubuntu16.04.1) ...
Selecting previously unselected package liblxc1.
Preparing to unpack .../liblxc1_2.0.3-0ubuntu1~ubuntu16.04.1_s390x.deb ...
Unpacking liblxc1 (2.0.3-0ubuntu1~ubuntu16.04.1) ...
Selecting previously unselected package lxcfs.
Preparing to unpack .../lxcfs_2.0.2-0ubuntu1~ubuntu16.04.1_s390x.deb ...
Unpacking lxcfs (2.0.2-0ubuntu1~ubuntu16.04.1) ...
Selecting previously unselected package liblzo2-2:s390x.
Preparing to unpack .../liblzo2-2_2.08-1.2_s390x.deb ...
Unpacking liblzo2-2:s390x (2.08-1.2) ...
Selecting previously unselected package squashfs-tools.
Preparing to unpack .../squashfs-tools_1%3a4.3-3ubuntu2_s390x.deb ...
Unpacking squashfs-tools (1:4.3-3ubuntu2) ...
Selecting previously unselected package uidmap.
Preparing to unpack .../uidmap_1%3a4.2-3.1ubuntu5_s390x.deb ...
Unpacking uidmap (1:4.2-3.1ubuntu5) ...
Selecting previously unselected package xz-utils.
Preparing to unpack .../xz-utils_5.1.1alpha+20120614-2ubuntu2_s390x.deb ...
Unpacking xz-utils (5.1.1alpha+20120614-2ubuntu2) ...
```

```
Selecting previously unselected package lxd.
Preparing to unpack .../lxd_2.0.3-0ubuntu1~ubuntu16.04.2_s390x.deb ...
Adding system user `lxd' (UID 109) ...
Adding new user `lxd' (UID 109) with group `nogroup' ...
Creating home directory `/var/lib/lxd/' ...
Adding group `lxd' (GID 118) ...
Done.
Unpacking lxd (2.0.3-0ubuntu1~ubuntu16.04.2) ...
Selecting previously unselected package distro-info.
Preparing to unpack .../distro-info_0.14build1_s390x.deb ...
Unpacking distro-info (0.14build1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../dns-root-data_2015052300+h+1_all.deb ...
Unpacking dns-root-data (2015052300+h+1) ...
Selecting previously unselected package lxd-client.
Preparing to unpack .../lxd-client_2.0.3-0ubuntu1~ubuntu16.04.2_s390x.deb ...
Unpacking lxd-client (2.0.3-0ubuntu1~ubuntu16.04.2) ...
Selecting previously unselected package juju-2.0.
Preparing to unpack .../juju-2.0_2.0~beta7-0ubuntu1.16.04.1_s390x.deb ...
Unpacking juju-2.0 (2.0~beta7-0ubuntu1.16.04.1) ...
Selecting previously unselected package juju.
Preparing to unpack .../juju_2.0~beta7-0ubuntu1.16.04.1_all.deb ...
Unpacking juju (2.0~beta7-0ubuntu1.16.04.1) ...
Selecting previously unselected package zfs-doc.
Preparing to unpack .../zfs-doc_0.6.5.6-0ubuntu10_all.deb ...
Unpacking zfs-doc (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package libuutil1linux.
Preparing to unpack .../libuutil1linux_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking libuutil1linux (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package libnvpair1linux.
Preparing to unpack .../libnvpair1linux_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking libnvpair1linux (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package libzpool2linux.
Preparing to unpack .../libzpool2linux_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking libzpool2linux (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package libzfs2linux.
Preparing to unpack .../libzfs2linux_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking libzfs2linux (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package zfsutils-linux.
Preparing to unpack .../zfsutils-linux_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking zfsutils-linux (0.6.5.6-0ubuntu10) ...
Selecting previously unselected package zfs-zed.
Preparing to unpack .../zfs-zed_0.6.5.6-0ubuntu10_s390x.deb ...
Unpacking zfs-zed (0.6.5.6-0ubuntu10) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for dbus (1.10.6-1ubuntu3) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (229-4ubuntu7) ...
Processing triggers for initramfs-tools (0.122ubuntu8.1) ...
update-initramfs: Generating /boot/initrd.img-4.4.0-31-generic
Using config file '/etc/zipl.conf'
Building bootmap in '/boot'
Building menu 'menu'
Adding #1: IPL section 'ubuntu' (default)
```

```
Adding #2: IPL section 'old'
Preparing boot device: dasda (1628).
Done.
Setting up acl (2.2.52-3) ...
Setting up libnetfilter-conntrack3:s390x (1.0.5-1) ...
Setting up dnsmasq-base (2.75-1ubuntu0.16.04.1) ...
Setting up lxcfs (2.0.2-0ubuntu1~ubuntu16.04.1) ...
Setting up liblzo2-2:s390x (2.08-1.2) ...
Setting up squashfs-tools (1:4.3-3ubuntu2) ...
Setting up uidmap (1:4.2-3.1ubuntu5) ...
Setting up xz-utils (5.1.1alpha+20120614-2ubuntu2) ...
update-alternatives: using /usr/bin/xz to provide /usr/bin/lzma (lzma) in auto
mode
Setting up distro-info (0.14build1) ...
Setting up dns-root-data (2015052300+h+1) ...
Setting up lxd-client (2.0.3-0ubuntu1~ubuntu16.04.2) ...
Setting up juju-2.0 (2.0~beta7-0ubuntu1.16.04.1) ...
Setting up juju (2.0~beta7-0ubuntu1.16.04.1) ...
Setting up zfs-doc (0.6.5.6-0ubuntu10) ...
Setting up libuutil1linux (0.6.5.6-0ubuntu10) ...
Setting up libnvpair1linux (0.6.5.6-0ubuntu10) ...
Setting up libzpool2linux (0.6.5.6-0ubuntu10) ...
Setting up libzfs2linux (0.6.5.6-0ubuntu10) ...
Setting up zfsutils-linux (0.6.5.6-0ubuntu10) ...
zfs-import-cache.service is a disabled or a static unit, not starting it.
zfs-import-scan.service is a disabled or a static unit, not starting it.
zfs-mount.service is a disabled or a static unit, not starting it.
Processing triggers for initramfs-tools (0.122ubuntu8.1) ...
update-initramfs: Generating /boot/initrd.img-4.4.0-31-generic
Using config file '/etc/zipl.conf'
Building bootmap in '/boot'
Building menu 'menu'
Adding #1: IPL section 'ubuntu' (default)
Adding #2: IPL section 'old'
Preparing boot device: dasda (1628).
Done.
Setting up zfs-zed (0.6.5.6-0ubuntu10) ...
zed.service is a disabled or a static unit, not starting it.
Setting up liblxc1 (2.0.3-0ubuntu1~ubuntu16.04.1) ...
Setting up lxd (2.0.3-0ubuntu1~ubuntu16.04.2) ...
locale: Cannot set LC_ALL to default locale: No such file or directory

The default LXD bridge, lxdbr0, comes unconfigured by default.
Only limited HTTP connectivity through a PROXY will be available.
To go through the initial LXD configuration, run: lxd init

Setting up lxc-common (2.0.3-0ubuntu1~ubuntu16.04.1) ...
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for dbus (1.10.6-1ubuntu3) ...
Processing triggers for systemd (229-4ubuntu7) ...
Processing triggers for ureadahead (0.100.0-19) ...
support@lnxs0101:~$
```

The default user needs to be in the group named `lxd`. In this case, add a new user named `jujuman` and ensure that it's added to the `sudo` and `lxd` groups. Set the password to be something secure of your choosing. Use the commands shown in Example 5-3 to set the password.

Example 5-3 Setting the Juju password

```
support@lnxs0101:~$ sudo useradd -G sudo,lxd jujuman

support@lnxs0101:~$ sudo passwd jujuman
Enter new UNIX password: **********
Retype new UNIX password: **********
passwd: password updated successfully

support@lnxs0101:~$ exit
logout
Connection to lnxs0101 closed.
```

Now, log in as the new user, as shown in Example 5-4.

Example 5-4 Logging in as the new user

```
$ ssh -l jujuman lnxs0101
jujuman@lnxs0101's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-25-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Tue Jul 19 06:39:51 2016 from 10.172.194.66
jujuman@lnxs0101:~$ groups
jujuman adm cdrom sudo dip plugdev cpacfstats lpadmin sambashare lxd
support@lnxs0101:~$ sudo lxd init
[sudo] password for jujuman:
Name of the storage backend to use (dir or zfs): dir
Would you like LXD to be available over the network (yes/no)? no
Do you want to configure the LXD bridge (yes/no)? yes
Package configuration

 ............................ Configuring lxd ............................
 .                                                                       .
 . Containers need a bridge to connect them together and to the host for .
 . outside network connectivity.                                         .
 .                                                                       .
 . Choosing this option will let you configure the default LXD bridge to .
 . your liking.                                                          .
 .                                                                       .
 . If you would rather not have LXD do this for you, then you will be asked .
 . whether you want to use an existing bridge or just do everything      .
 . manually.                                                             .
 .                                                                       .
 . Would you like to setup a network bridge for LXD containers now?      .
 .                                                                       .
 .                    <Yes>                        <No>                  .
 .                                                                       .
 .........................................................................
```

```
==> Yes

Package configuration

               ·················· Configuring lxd ··················
               · A valid network interface name (e.g. lxdbr0).  ·
               ·                                                 ·
               · Bridge interface name:                          ·
               ·                                                 ·
               · lxdbr0_____________________________________     ·
               ·                                                 ·
               ·                     <Ok>                        ·
               ·                                                 ·
               ·................................................·

==> Ok

Package configuration

     ·························· Configuring lxd ···························
     ·                                                                    ·
     · This is needed to provide IPv4 connectivity for your containers.   ·
     ·                                                                    ·
     · Do you want to setup an IPv4 subnet?                               ·
     ·                                                                    ·
     ·                  <Yes>                    <No>                     ·
     ·                                                                    ·
     ·····················································================

==> Yes

Package configuration

  ····························· Configuring lxd ·····························
  ·                                                                       ·
  · A random subnet was selected for you                                  ·
  ·                                                                       ·
  · This subnet was selected for your convenience and the next questions  ·
  · have been pre-answered accordingly.                                   ·
  ·                                                                       ·
  · Please make sure this subnet isn't already in use somewhere on your   ·
  · network, if it is, change it to one which isn't.                      ·
  ·                                                                       ·
  · If you later notice network connectivity issues, re-configure lxd and ·
  · pick a different subnet.                                              ·
  ·                                                                       ·
  ·                                <Ok>                                   ·
  ·                                                                       ·
  ·························================================================

==> Ok
```

```
Package configuration

                  ··········· Configuring lxd ···········
                  · A valid IPv4 address. (e.g. 10.0.8.1) ·
                  ·                                       ·
                  · IPv4 address:                         ·
                  ·                                       ·
                  · 10.45.134.1_________________________  ·
                  ·                                       ·
                  ·                <Ok>                   ·
                  ·                                       ·
                  ·········································

==> 0k

Package configuration

                      ········ Configuring lxd ········
                      · A valid CIDR mask. (e.g. 24)  ·
                      ·                               ·
                      · IPv4 CIDR mask:               ·
                      ·                               ·
                      · 24_____________________________ ·
                      ·                               ·
                      ·            <Ok>               ·
                      ·                               ·
                      ·································

==> 0k

Package configuration

       ····················· Configuring lxd ·····················
       · The first address to be handed out over DHCP (e.g. 10.0.8.2)  ·
       ·                                                               ·
       · First DHCP address:                                           ·
       ·                                                               ·
       · 10.45.134.2_________________________________________________  ·
       ·                                                               ·
       ·                           <Ok>                                ·
       ·                                                               ·
       ·················································************

==> 0k

Package configuration
```

```
        ....................... Configuring lxd .......................
        . The last address to be handed out over DHCP (e.g. 10.0.8.254)  .
        .                                                                .
        . Last DHCP address:                                             .
        .                                                                .
        . 10.45.134.254_____________________________________________     .
        .                                                                .
        .                             <Ok>                               .
        .                                                                .
        ....................................................................
```

==> **Ok**

Package configuration

```
    ......................... Configuring lxd ..........................
    . The maximum number of DHCP leases that can be obtained. (e.g. 250)  .
    .                                                                     .
    . Max number of DHCP clients:                                         .
    .                                                                     .
    . 252_____________________________________________________________    .
    .                                                                     .
    .                               <Ok>                                  .
    .                                                                     .
    .......................................................................
```

== **Ok**

Package configuration

```
         ...................... Configuring lxd .......................
         .                                                             .
         . This is needed unless you are using a routed IPv4 subnet.  .
         .                                                             .
         . Do you want to NAT the IPv4 traffic?                        .
         .                                                             .
         .               <Yes>                 <No>                    .
         .                                                             .
         .............................................................
```

== **Yes**

Package configuration

```
························· Configuring lxd ·························
·                                                                 ·
· This is needed to provide IPv6 connectivity for your containers.  ·
·                                                                 ·
· Do you want to setup an IPv6 subnet?                            ·
·                                                                 ·
·                <Yes>                    <No>                    ·
·                                                                 ·
···································································


==> No

jujuman@lnxs0101:~$ sudo lxd init
Name of the storage backend to use (dir or zfs): dir
Would you like LXD to be available over the network (yes/no)? no
Do you want to configure the LXD bridge (yes/no)? yes
Warning: Stopping lxd.service, but it can still be activated by:
  lxd.socket
LXD has been successfully configured.
jujuman@lnxs0101:~
```

Creating the Juju controller

Next, create the Juju controller, as shown in Example 5-5.

Example 5-5 Creating the Juju controller

```
jujuman@lnxs0101:~$ juju bootstrap lxd-test localhost
Creating Juju controller "local.lxd-test" on localhost/localhost
Bootstrapping model "admin"
Starting new instance for initial controller
Launching instance
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
```

```
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
```

```
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:
copying image for ubuntu-xenial from https://cloud-images.ubuntu.com/releases/:  -
juju-356ff5bd-b6ba-4ff9-89bb-c4d464823dbe-machine-0
Installing Juju agent on bootstrap instance
Preparing for Juju GUI 2.1.8 release installation
Waiting for address
Attempting to connect to 10.45.134.191:22
Logging to /var/log/cloud-init-output.log on remote host
Running apt-get update
Running apt-get upgrade
Installing package: curl
Installing package: cpu-checker
Installing package: bridge-utils
Installing package: cloud-utils
Installing package: cloud-image-utils
Installing package: tmux
Fetching tools: curl -sSfw 'tools from %{url_effective} downloaded: HTTP
%{http_code}; time %{time_total}s; size %{size_download} bytes; speed
%{speed_download} bytes/s ' --retry 10 -o $bin/tools.tar.gz
<[https://streams.canonical.com/juju/tools/agent/2.0-beta7/juju-2.0-beta7-xenial-s
390x.tgz]>
Bootstrapping Juju machine agent
Starting Juju machine agent (jujud-machine-0)
Bootstrap agent installed
Waiting for API to become available: upgrade in progress (upgrade in progress)
Waiting for API to become available: upgrade in progress (upgrade in progress)
Waiting for API to become available: upgrade in progress (upgrade in progress)
Waiting for API to become available: upgrade in progress (upgrade in progress)
Bootstrap complete, local.lxd-test now available.
jujuman@lnxs0101:~$
```

Listing the controllers, switches, and status

Finally, list the controllers, switches, and status, as shown in Example 5-6.

Example 5-6 List the controllers, switches, and status

```
jujuman@lnxs0101:~$ juju list-controllers
CONTROLLER       MODEL    USER         SERVER
local.lxd-test*  default  admin@local  10.45.134.191:17070
jujuman@lnxs0101:~$ juju switch
local.lxd-test:default
jujuman@lnxs0101:~$ juju status
[Services]
NAME       STATUS EXPOSED CHARM

[Units]
ID      WORKLOAD-STATUS JUJU-STATUS VERSION MACHINE PORTS PUBLIC-ADDRESS MESSAGE

[Machines]
ID         STATE DNS INS-ID SERIES AZ
```

5.2.3 Adding and deploying a model

Add a model and deploy it, as shown in Example 5-7.

Example 5-7 Add and deploy a model

```
jujuman@lnxs0101:~$ juju add-model z-demo
added model "z-demo"

jujuman@lnxs0101:~$ juju list-models
NAME     OWNER        STATUS     LAST CONNECTION
admin    admin@local  available  never connected
z-demo*  admin@local  available  never connected

jujuman@lnxs0101:~$ juju deploy cs:~bigdata-dev/mariadb
Added charm "cs:~bigdata-dev/xenial/mariadb-2" to the model.
Deploying charm "cs:~bigdata-dev/xenial/mariadb-2" with the charm series "xenial".

jujuman@lnxs0101:~$ juju deploy cs:xenial/ghost
Added charm "cs:ghost-0" to the model.
Deploying charm "cs:ghost-0" with the default charm metadata series "xenial".

jujuman@lnxs0101:~$ juju add-relation ghost mariadb

jujuman@lnxs0101:~$ juju expose ghost

jujuman@lnxs0101:~$ juju status
# or:
jujuman@lnxs0101:~$ watch juju status

jujuman@lnxs0101:~$ juju status
[Services]
NAME       STATUS      EXPOSED CHARM
ghost      maintenance true    cs:ghost-0
mariadb    unknown     false   cs:~bigdata-dev/xenial/mariadb-2
```

```
[Relations]
SERVICE1    SERVICE2 RELATION TYPE
ghost       mariadb  db       regular
mariadb     mariadb  cluster  peer

[Units]
ID        WORKLOAD-STATUS JUJU-STATUS VERSION   MACHINE PORTS PUBLIC-ADDRESS
MESSAGE
ghost/0   maintenance     executing   2.0-beta7 1             10.45.134.8
(update-status) installing NPM dependencies for /srv/app
mariadb/0 unknown    idle    2.0-beta7 0      10.45.134.161

[Machines]
ID         STATE   DNS           INS-ID
SERIES AZ
0          started 10.45.134.161
juju-372a591f-1a8b-429f-8c27-6a8df9102931-machine-0 xenial
1          started 10.45.134.8
juju-372a591f-1a8b-429f-8c27-6a8df9102931-machine-1 xenial
```

5.2.4 Invoking the graphical user interface

Invoke the graphical user interface (GUI) as shown in Example 5-8.

Example 5-8 Invoke the GUI

```
jujuman@lnxs0101:~$ juju gui --no-browser --show-credentials
https://10.45.134.191:17070/gui/372a591f-1a8b-429f-8c27-6a8df9102931/
Username: admin@local
Password: db8abb97de0ba71ea07d638666a5329e
# access Juju GUI via web browser
# and use username and password from the output of "juju gui --no-browser
--show-credentials"
```

Assuming that you have connectivity, you can now launch an Internet browser from your local workstation as follows:

```
sysadmin@controlstation10:~$ firefox
https://10.45.134.191:17070/gui/372a591f-1a8b-429f-8c27-6a8df9102931/
```

You might require a proxy to access your new Juju network, for which `sshuttle` is a helpful solution. Assuming that you are also running Ubuntu on your local workstation, you can install `sshuttle` as shown in Example 5-9.

Example 5-9 Installing sshuttle

```
sysadmin@controlstation10:~$ sudo apt-get install sshuttle
sysadmin@controlstation10:~$ sshuttle -r jujuman@lnxs0101 10.45.134.191
10.0.3.0/24
The authenticity of host 'lnxs0101 (10.245.236.12)' can't be established.
ECDSA key fingerprint is SHA256:p9I7eDkDxCEV+uC2z+tq5VihCIqo9nZvOK4yjwL1rAQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'lnxs0101' (ECDSA) to the list of known hosts.
Warning: the ECDSA host key for 'lnxs0101' differs from the key for the IP address
'10.245.236.12'
```

```
Offending key for IP in /home/sysadmin/.ssh/known_hosts:30
Are you sure you want to continue connecting (yes/no)? yes
jujuman@lnxs0101's password:
client: Connected.
server: warning: closed channel 1 got cmd=TCP_STOP_SENDING len=0
server: warning: closed channel 7 got cmd=TCP_STOP_SENDING len=0
server: warning: closed channel 7 got cmd=TCP_EOF len=0
server: warning: closed channel 9 got cmd=TCP_STOP_SENDING len=0
^Cclient:
client: Keyboard interrupt: exiting.
sysadmin@controlstation10:~$

sysadmin@controlstation10:~$ firefox
https://10.45.134.191:17070/gui/372a591f-1a8b-429f-8c27-6a8df9102931/
# click 'Advanced'
# click 'Proceed to 10.45.134.191 (unsafe)'
# Login:
# Username: admin@local
# Password: db8abb97de0ba71ea07d638666a5329e
sysadmin@controlstation10:~$
# you are now ready to use the Juju GUI ...
```

5.2.5 Other available resources

For more information, refer to:

- https://jujucharms.com/docs/devel/getting-started
- https://jujucharms.com/docs/devel/developer-getting-started
- https://jujucharms.com/u/bigdata-dev/mariadb
- https://jujucharms.com/docs/devel/charms

6

Working with systemd

“The greatest impediment to progress is the phrase ‘but, we’ve always done it that way’.”

— Grace Hopper

This chapter describes how to work with an Ubuntu Server 16.04 system that uses **systemd**. On modern Linux installations, **systemd** plays a major role. Understanding the concepts and knowing the utilities that are provided by **systemd** is key to any Linux administrator.

It includes the following topics:

- Getting started with systemd
- Using systemd units
- Working with the systemd journal
- The system boot process
- Analyze Linux instances that use systemd

6.1 Getting started with systemd

As a system and services manager, **systemd** replaced the SysV init system. It is the first user space process that the kernel starts when booting. This process is responsible for starting all of the services and their dependencies that allow the system to act as a server. A *unit* is a resource that the system knows how to manage and operate, and is the primary object that **systemd** tools deal with. There are different *unit types*.

Table 6-1 lists the types of **systemd** unit files and their purpose.

Table 6-1 Types of systemd unit files

Type	Unit file extension	Purpose
Automount	`.automount`	File system automount point, supercedes `/etc/fstab`
Device	`.device`	Trigger reactions for devices as they appear or disappear
Mount	`.mount`	Control mount points
Path	`.path`	File or directory in a file system
Service	`.service`	Used to start services
Socket	`.socket`	Allow socket-based activations
Scope	`.scope`	Externally created process
Swap	`.swap`	A swap device or file
Target	`.target`	Allow grouping of units to act as synchronization points
Timer	`.timer`	A systemd timer

6.1.1 The systemd unit files

These resources are defined in configuration files that **systemd** calls *unit files*. These text files contain a representation of resources that can be managed by daemons and controlled by utilities. Unit files are located in several different places. System unit files are usually located in `/lib/systemd/system`, and when software is installed, any associated unit files are placed there by default. Files in that directory should not be edited.

If you need to change a unit file, it should be done in `/etc/systemd/system`. Unit files that are located in that directory take precedence over unit files with the same name that are located elsewhere. If you want to modify a unit file, you should copy it from `/lib/systemd/system` to `/etc/systemd/system` and make your changes there.

The **systemd** process can dynamically create unit files at run time, and these are stored in `/run/systemd/system`. Changes made to unit files that are located in that directory are only effective during the duration of the session, and are lost when the system is rebooted. Unit files in this directory take precedence over those in `/lib/systemd/system`, but are still lower in precedence to those in `/etc/systemd/system`.

Some unit files are generated dynamically by **systemd** unit generators. For example, the **systemd-fstab-generator** parses `/etc/fstab`, and creates mount units for the entries if necessary.

Unit files are organized into sections. A section begins with the section name between a pair of square brackets ([and]), for example, `[Unit]`, `[Service]`, and `[Install]`. The section extends until the start of another section or the end of the file. Section names are case-sensitive. The behavior of the unit and associated metadata are defined using key-value pairs that are separated by an Equals sign (=), for example, `FirstKeyword=value1`, `SecondKeyword=value2`, and so on.

For example, the command shown in Example 6-1 provides the unit file for the **`ssh.service`**.

Example 6-1 Sample ssh.service command that provides the unit file

```
# cat /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Example 6-2 shows the unit file for the **`rc-local.service`**, which provides compatibility for `/etc/rc.local` from SysV init.

Example 6-2 The unit file for rc-local.service

```
# cat /lib/systemd/system/rc-local.service
...
# This unit gets pulled automatically into multi-user.target by
# systemd-rc-local-generator if /etc/rc.local is executable.
[Unit]
Description=/etc/rc.local Compatibility
ConditionFileIsExecutable=/etc/rc.local
After=network.target

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
RemainAfterExit=yes
```

6.1.2 Relationship between units

Information in unit files describes how **`systemd`** manage a service or application, including how to start and stop it; when and if it should automatically be started or stopped; and what dependency might exist with other services or applications. For example, before the

local-fs.target is reached, the local mount units need to be activated, and they are dependent on the device units. By default, the system boots into the **default.target** that pulls in all of the dependencies necessary to start the system. You can find more details about the boot process in 6.4, “The system boot process” on page 107.

6.2 Using systemd units

This section describes how to manage services and isolate **systemd** targets. The **systemctl** command is used to control the **systemd** system and service manager. The **systemctl** command has many options that are not covered here (see the man pages for details). Instead, we provide examples of using the command to perform many common tasks.

6.2.1 Managing services

Because **systemd** starts all of the services, one of the first things you want to know is how to get a list of the available services. Use **systemctl** to control services and inspect their state.

Example 6-3 shows the command for active service units (running services).

Example 6-3 Active service units (running services)

```
# systemctl -t service
UNIT                                LOAD   ACTIVE SUB     DESCRIPTION
accounts-daemon.service             loaded active running Accounts Service
acpid.service                       loaded active running ACPI event daemon
apparmor.service                    loaded active exited  LSB: AppArmor initialization
apport.service                      loaded active exited  LSB: automatic crash report
generation
atd.service                         loaded active running Deferred execution scheduler
cron.service                        loaded active running Regular background program
processing daemon
dbus.service                        loaded active running D-Bus System Message Bus
ifup@enc600.service                 loaded active exited  ifup for enc600
...
systemd-udevd.service               loaded active running udev Kernel Device Manager
systemd-update-utmp.service         loaded active exited  Update UTMP about System
Boot/Shutdown
systemd-user-sessions.service       loaded active exited  Permit User Sessions
ufw.service                         loaded active exited  Uncomplicated firewall
user@1001.service                   loaded active running User Manager for UID 1001

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

43 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Note: The **systemctl** command attempts to fit the contents of its output into your terminal window. If the output has more lines than your terminal window is capable of displaying, it pipes the output into `less`, the default pager. You can change that behavior by adding the `--no-pager` option. In case the width of your terminal window is too narrow to display the full contents, you can add the `--full` or `-l` option to show the full contents. Of course, the `--no-pager` and the `--full` options can be combined if wanted.

The following command lists failed service units:

```
# systemctl -t service --state=failed
0 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

To query the status of a service unit, use the command shown in Example 6-4.

Example 6-4 Query the status of the service unit

```
# systemctl status sshd.service
* ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2016-05-26 21:19:55 EDT; 15h ago
 Main PID: 982 (sshd)
    Tasks: 1
   Memory: 6.7M
      CPU: 574ms
   CGroup: /system.slice/ssh.service
           `-982 /usr/sbin/sshd -D

May 26 21:19:55 lnxs0038 systemd[1]: Starting OpenBSD Secure Shell server...
May 26 21:19:55 lnxs0038 sshd[982]: Server listening on 0.0.0.0 port 22.
May 26 21:19:55 lnxs0038 sshd[982]: Server listening on :: port 22.
May 26 21:19:55 lnxs0038 systemd[1]: Started OpenBSD Secure Shell server.
May 26 21:20:05 lnxs0038 sshd[1070]: Accepted password for bader from 9.12.5.134 port 65172 ssh2
May 26 21:20:05 lnxs0038 sshd[1070]: pam_unix(sshd:session): session opened for user bader by (uid=0)
May 26 21:31:06 lnxs0038 sshd[1244]: Accepted password for bader from 9.12.5.134 port 65235 ssh2
May 26 21:31:06 lnxs0038 sshd[1244]: pam_unix(sshd:session): session opened for user bader by (uid=0)
May 27 10:24:12 lnxs0038 sshd[2034]: Accepted password for bader from 9.12.5.134 port 58309 ssh2
May 27 10:24:12 lnxs0038 sshd[2034]: pam_unix(sshd:session): session opened for user bader by (uid=0)
```

In addition to the information you expect, the output also shows the last 10 log messages from the journal. If you want to know if a service unit is active or not, enter the following command:

```
# systemctl is-active sshd.service
active
```

To stop, start, and restart a service unit, use the following commands:

```
# systemctl stop vsftpd.service
# systemctl start vsftpd.service
# systemctl restart vsftpd.service
```

You can omit the `.service` suffix, and you can also specify multiple units on one command:

```
# systemctl restart vsftpd sshd
```

As a Linux administrator you might change the configuration of a running service and want to have it to reload its configuration without restarting the service. Service unit files can use the **ExecReload=** directive to specify a command line (such as sending a signal) that triggers the reload of the configuration. If your service supports it, you can trigger reload using:

```
# systemctl reload sshd
```

However, not all services support that:

```
# systemctl reload vsftpd
Failed to reload vsftpd.service: Job type reload is not applicable for unit
vsftpd.service.
```

To reload the configuration of services that support it, and restart services that do not, type the following command:

```
# systemctl reload-or-restart sshd vsftpd
```

To list installed service units, use the following command:

```
# systemctl list-unit-files -t service
UNIT FILE                                  STATE
accounts-daemon.service                    enabled
acpid.service                              disabled
...
uuidd.service                              indirect
x11-common.service                         masked
165 unit files listed.
```

To list the service units that are wanted or required by another unit (such as the `multi-user.target`), use the following command:

```
# systemctl list-unit-files -t service --state=enabled
UNIT FILE                   STATE
accounts-daemon.service     enabled
atd.service                 enabled
...
unattended-upgrades.service enabled
ureadahead.service          enabled
23 unit files listed.
```

To disable a service, use the following command:

```
# systemctl disable vsftpd
```

To enable a service, use the following command:

```
# systemctl enable vsftpd
```

Note: Issuing the **`enable`** command for a service unit doesn't trigger the activation of the service unit. It doesn't start the service. It merely creates a dependency on another unit, for example the `multi-user.target` unit. This dependency causes the start of the service whenever the unit the service depends on is activated, for example the `multi-user.target`. Disabling the service unit removes the dependency.

To check which socket units activate which services, enter the command shown in Example 6-5.

Example 6-5 Which socket units activate which services

```
# systemctl list-sockets
LISTEN                          UNIT                           ACTIVATES
/run/acpid.socket               acpid.socket                   acpid.service
/run/dmeventd-client            dm-event.socket                dm-event.service
/run/dmeventd-server            dm-event.socket                dm-event.service
...
audit 1                         systemd-journald-audit.socket  systemd-journald.service
kobject-uevent 1                systemd-udevd-kernel.socket    systemd-udevd.service

17 sockets listed.
Pass --all to see loaded but inactive sockets, too.
```

6.2.2 Managing systemd target units

Target units act as synchronization points for groups of units. The concept of **systemd** *targets* is somewhat similar to the concept of *runlevels* in the SysV init system. However, there are a couple of differences. Rather than using runlevel numbers, the **systemd** targets have names because they are units. Also, with **systemd**, there is usually more than one target active, because one target unit might depend on another target unit.

To list active targets, use the command shown in Example 6-6.

Example 6-6 List active targets

```
# systemctl -t target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Encrypted Volumes
...
time-sync.target       loaded active active System Time Synchronized
timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

19 loaded units listed.
Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

To list all targets, use the command shown in Example 6-7.

Example 6-7 List all targets

```
# systemctl -t target --all
UNIT                        LOAD   ACTIVE   SUB    DESCRIPTION
basic.target                loaded active   active Basic System
cryptsetup.target           loaded active   active Encrypted Volumes
emergency.target            loaded inactive dead   Emergency Mode
final.target                loaded inactive dead   Final Step
getty.target                loaded active   active Login Prompts
graphical.target            loaded active   active Graphical Interface
halt.target                 loaded inactive dead   Halt
...
time-sync.target            loaded active   active System Time Synchronized
timers.target               loaded active   active Timers
umount.target               loaded inactive dead   Unmount All Filesystems

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

28 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

To query the default target that the system boots into, use the command shown in Example 6-8.

Example 6-8 Query the default target for boot

```
# systemctl get-default
graphical.target
```

To set the default target that the system should boot into, use the command shown in Example 6-9.

Example 6-9 Set the default target for boot

```
# systemctl set-default multi-user.target
Created symlink from /etc/systemd/system/default.target to
/lib/systemd/system/multi-user.target.
# systemctl set-default graphical.target
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to
/lib/systemd/system/graphical.target.
```

To switch to a target, use the following command:

```
# systemctl isolate multi-user.target
```

The **isolate** command activates the specified target and all of its dependent units. All other units are stopped. By default, only a few targets are able to be isolated, to prevent unusable system states. Table 6-2 maps important **systemd** targets to SysV runlevels.

Table 6-2 Targets and runlevels

systemd Target	SysV Runlevel	Notes
poweroff.target	0	Halts the system
rescue.target	1, s, single	Single user mode that provides a base system and a rescue shell
multi-user.target	2,3,4	Multi-user, non-graphical but with Network and Services running
graphical.target	5	Multi-user, Graphical
reboot.target	6	Reboot the system
emergency.target	emergency	Emergency shell. This is a special systemd target unit that can be specified as a kernel command-line argument: `systemd.unit=emergency.target`

6.3 Working with the systemd journal

This section describes how to enable persistent journal data and how to view the journal.

6.3.1 Getting started with the journal

The journal is part of **systemd** and provides a modern logging mechanism. It allows capturing Kernel log messages, regular syslog messages, `stdout`/`stderr` written by services, and

messages from the early boot stages. Along with the log message text, the journal stores metadata, such as PID, UID, GID, the executable, and so on. For more details, see the `man` page for **`systemd.journal-fields`**. All of the information in the journal is indexed, and can be queried by the administrator.

Management of the journal is handled by the **`systemd-journald`** service. You can view the status of the journal service using the command shown in Example 6-10.

Example 6-10 View the status of the journal service

```
# systemctl status systemd-journald
* systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset: enabled)
   Active: active (running) since Fri 2016-05-27 14:20:59 EDT; 16min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 441 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1
   Memory: 2.2M
      CPU: 21ms
   CGroup: /system.slice/systemd-journald.service
           `-441 /lib/systemd/systemd-journald

May 27 14:20:59 lnxs0038 systemd-journald[441]: Runtime journal (/run/log/journal/) is 1016.0K,
max 7.9M, 6.9M free.
May 27 14:20:59 lnxs0038 systemd-journald[441]: Journal started
Warning: Journal has been rotated since unit was started. Log output is incomplete or
unavailable.
```

Note the message "`Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable`." at the end of the output. By default in Ubuntu 16.04 LTS, the data of the journal is not stored permanently. The **`systemd-journald`** service is configured to store its data onto a small in-memory disk, `/run/log/journal`, which is a TMPFS. Periodically, that file is rotated and previous entries are lost.

Therefore only the latest log data can be queried from the journal directly. To change the **`systemd-journald`** to store its data persistently on disk, simply create the `/var/log/journal` directory. When the **`journald`** service gets restarted, it detects the existence of this directory and start to store the system journal permanently on disk.

To enable persistent journal data at `/var/log/journal`, use the command shown in Example 6-11.

Example 6-11 Enable persistent journal data

```
# mkdir /var/log/journal
# systemctl restart systemd-journald
# systemctl status systemd-journald
* systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset: enabled)
   Active: active (running) since Fri 2016-05-27 14:59:57 EDT; 6s ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 1260 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1
```

```
   Memory: 1.7M
      CPU: 6ms
   CGroup: /system.slice/systemd-journald.service
           `-1260 /lib/systemd/systemd-journald

May 27 14:59:57 lnxs0038 systemd-journald[1260]: System journal (/var/log/journal/) is 8.0M, max
94.5M, 86.5M free.
May 27 14:59:57 lnxs0038 systemd-journald[1260]: Runtime journal (/run/log/journal/) is 1016.0K,
max 7.9M, 6.9M free.
May 27 14:59:57 lnxs0038 systemd-journald[1260]: Time spent on flushing to /var is 4.347ms for
464 entries.
May 27 14:59:57 lnxs0038 systemd-journald[1260]: Journal started
Warning: Journal has been rotated since unit was started. Log output is incomplete or
unavailable.
```

The **systemd-journald** service is now recording the system journal to the `/var/log/journal` directory.

Disable and stop the traditional log daemon (optional)

This step is optional. If wanted, you can also use both the **systemd-journald** command and a traditional syslog daemon. If you don't rely on the traditional syslog, it can be disabled and stopped by using the command shown in Example 6-12.

Example 6-12 Disable and stop the traditional log daemon

```
# systemctl status rsyslog
* rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2016-05-27 15:21:33 EDT; 26s ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 1212 (rsyslogd)
    Tasks: 4
   Memory: 532.0K
      CPU: 1ms
...
# systemctl disable rsyslog
Synchronizing state of rsyslog.service with SysV init with
/lib/systemd/systemd-sysv-install...
Executing /lib/systemd/systemd-sysv-install disable rsyslog
Removed symlink /etc/systemd/system/syslog.service.
# systemctl stop rsyslog
# systemctl status rsyslog
* rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
```

6.3.2 Viewing the journal

Use the **journalctl** utility to view and filter the journal. If invoked without any parameters, it shows the entire journal in a format that is similar to traditional log files. The output from **journalctl** is piped into `less`, the default pager, as shown in Example 6-13.

Example 6-13 The journalctl utility

```
# journalctl
-- Logs begin at Fri 2016-05-27 14:59:57 EDT, end at Fri 2016-05-27 15:25:37 EDT. --
May 27 14:59:57 lnxs0038 systemd-journald[1260]: System journal (/var/log/journal/) is 8.0M, max
94.5M, 86.5M free.
May 27 14:20:59 lnxs0038 systemd-journald[441]: Runtime journal (/run/log/journal/) is 1016.0K,
max 7.9M, 6.9M free.
May 27 14:20:59 lnxs0038 kernel: Initializing cgroup subsys cpuset
May 27 14:20:59 lnxs0038 kernel: Initializing cgroup subsys cpu
May 27 14:20:59 lnxs0038 kernel: Initializing cgroup subsys cpuacct
May 27 14:20:59 lnxs0038 kernel: Linux version 4.4.0-22-generic (buildd@z13-019) (gcc version
5.3.1 20160413 (Ubuntu 5.3.1-14ubuntu2) ) #40-Ubuntu SMP Th
May 27 14:20:59 lnxs0038 kernel: setup: Linux is running as a z/VM guest operating system in
64-bit mode
May 27 14:20:59 lnxs0038 kernel: setup: Max memory size: 2048MB
...
May 27 14:20:59 lnxs0038 kernel: RCU: Adjusting geometry for rcu_fanout_leaf=64, nr_cpu_ids=64
May 27 14:20:59 lnxs0038 kernel: NR_IRQS:259
May 27 14:20:59 lnxs0038 kernel: clocksource: tod: mask: 0xffffffffffffffff max_cycles:
0x3b0a9be803b0a9, max_idle_ns: 1805497147909793 ns
May 27 14:20:59 lnxs0038 kernel: console [ttyS0] enabled
May 27 14:20:59 lnxs0038 kernel: Calibrating delay loop (skipped)... 20325.00 BogoMIPS preset
lines 1-52
```

View the journal and tell the `less` pager to jump to the end of the log, as shown in Example 6-14.

Example 6-14 Jump to end of the log

```
# journalctl -e
May 27 15:22:10 lnxs0038 systemd[1]: Started ACPI event daemon.
May 27 15:22:35 lnxs0038 systemd[1]: Stopping System Logging Service...
May 27 15:22:35 lnxs0038 systemd[1]: Stopped System Logging Service.
May 27 15:25:37 lnxs0038 systemd[1]: Starting Cleanup of Temporary Directories...
May 27 15:25:37 lnxs0038 systemd-tmpfiles[1280]: [/usr/lib/tmpfiles.d/var.conf:14] Duplicate
line for path "/var/log", ignoring.
May 27 15:25:37 lnxs0038 systemd[1]: Started Cleanup of Temporary Directories.
lines 953-1004/1004 (END)
```

Show the last 10 entries from the log (similar to **tail -n 10**), as shown in Example 6-15.

Example 6-15 Show the last 10 entries

```
# journalctl -n 10
-- Logs begin at Fri 2016-05-27 14:59:57 EDT, end at Fri 2016-05-27 15:25:37 EDT. --
May 27 15:22:10 lnxs0038 systemd[1]: apt-daily.timer: Adding 6h 44min 5.316782s random time.
May 27 15:22:10 lnxs0038 systemd[1]: Started ACPI event daemon.
May 27 15:22:10 lnxs0038 systemd[1]: Reloading.
May 27 15:22:10 lnxs0038 systemd[1]: apt-daily.timer: Adding 8h 9min 22.386085s random time.
```

```
May 27 15:22:10 lnxs0038 systemd[1]: Started ACPI event daemon.
May 27 15:22:35 lnxs0038 systemd[1]: Stopping System Logging Service...
May 27 15:22:35 lnxs0038 systemd[1]: Stopped System Logging Service.
May 27 15:25:37 lnxs0038 systemd[1]: Starting Cleanup of Temporary Directories...
May 27 15:25:37 lnxs0038 systemd-tmpfiles[1280]: [/usr/lib/tmpfiles.d/var.conf:14] Duplicate
line for path "/var/log", ignoring.
May 27 15:25:37 lnxs0038 systemd[1]: Started Cleanup of Temporary Directories.
```

A live view of the journal can be obtained using the **-f** option. It shows the last 10 lines, like **tail -f**, and then the display updates as soon as new log messages are being added to the journal:

```
# journalctl -f
```

The live view can be terminated using `Control-C`, which sends a **SIGINT** signal to the **journalctl** process and causes it to end.

The default log output is intended to be read by people. There are other output formats that are intended to be machine-readable, and they are specified with the **-o** option. For details, see the `man` page for **journalctl**.

6.3.3 Filtering the journal

This section lists the various commands that are used to manage the journal.

To show the log messages of the current boot (filters out the messages from previous boots), use the following command:

```
# journalctl -b
```

To show today's log messages, use the following command:

```
# journalctl --since today
```

To show kernel messages of the current boot only, use the following command:

```
# journalctl -b -k
```

To show only errors, use the following command:

```
# journalctl -p err
```

To show the log messages of a specific unit (like the `sshd.service` unit), use the following command:

```
# journalctl -u sshd.service
```

To show the messages logged by a specific executable file, use the following command:

```
# journalctl /usr/sbin/sshd
```

To show the messages logged by a specific PID, user ID, or group ID, use the following command:

```
# journalctl _PID=0
# journalctl _UID=0
# journalctl _GID=0
```

For more fields that can be specified, see the `man` page for **systemd.journal-fields**.

To show the disk space that is being used by journal data, use the following command:

```
# journalctl --disk-usage
Archived and active journals take up 16.0M on disk.
```

6.4 The system boot process

This section provides an overview of the boot process on Linux instances with **systemd**.

The boot loader

The **zipl** boot loader loads the Linux kernel with the `initramfs` and the **boot** parameters into memory and starts the kernel. Then, the kernel unpacks the `initramfs` and starts `/sbin/init` provided by the `initramfs`, which is usually a symlink to **systemd** these days.

The initramfs

The `initramfs` contains the user space that is supposed to bring the disk that contains the root file system online and switch over to it. For example, if your `rootfs` is on a logical volume it needs to activate the volume group. After the `rootfs` is mounted to `/sysroot`, it is used as the new root directory. This is done by the `initrd-switch-root.service`. It runs **systemctl switch-root /sysroot**, which switches the root directory and starts the **systemd** as PID 1. The job of the `initramfs` is done.

The systemd

From now on, **systemd** is responsible to start your system. Its ultimate goal is to reach the default target. First it needs to load all of the unit files. Some of the unit files are generated dynamically by generators. For example, the **systemd-fstab-generator** creates mount units for the entries found in the `/etc/fstab`. After **systemd** has loaded the unit files that it knows the dependency tree and activates all of the units necessary for the default target.

6.5 Analyze Linux instances that use systemd

This section shows a few commands to retrieve performance statistics and information about the dependencies between **systemd** units.

6.5.1 Retrieving performance statistics

The **systemd-analyze** command can be used to analyze system boot-up performance. For information about what statistics can be obtained, see the `man` page for **systemd-analyze**.

To show the time that is spent in the various stages of the boot process, after the boot loader started the kernel, use the following command:

```
# systemd-analyze time
Startup finished in 3.747s (kernel) + 1.521s (userspace) = 5.268s
```

To display a list of units and the time that it took for each one to initialize, use the following command. The list is sorted in descending order by time:

```
# systemd-analyze blame
           479ms lvm2-monitor.service
           245ms dev-dasda2.device
           214ms ssh.service
```

```
          179ms lxd-containers.service
          171ms accounts-daemon.service
          140ms systemd-modules-load.service
          120ms var.mount
          ...
```

Note: Although this information can be useful to help optimize the boot-up process, it can also be misleading. A unit might appear to take a long time to initialize because it has to wait for the initialization of another service to complete.

The command shown in Example 6-16 displays a tree that shows the time-critical chain of units for a specific service or target unit.

Example 6-16 Tree that shows the time-critical chain of units

```
# systemd-analyze critical-chain default.target
The time after the unit is active or started is printed after the "@" character.
The time the unit takes to start is printed after the "+" character.

graphical.target @1.489s
`-multi-user.target @1.489s
  `-ssh.service @1.274s +214ms
    `-network.target @1.264s
      `-networking.service @1.173s +91ms
        `-apparmor.service @1.076s +95ms
          `-local-fs.target @1.075s
            `-run-lxcfs-controllers-blkio.mount @1.235s
              `-run-lxcfs-controllers.mount @1.233s
                `-local-fs-pre.target @658ms
                  `-lvm2-monitor.service @178ms +479ms
                    `-lvm2-lvmetad.service @258ms
                      `-lvm2-lvmetad.socket @178ms
                        `--.mount @142ms
                          `-system.slice @143ms
                            `--.slice @141ms
```

The following command generates a Scalable Vector Graphics (SVG) image file that shows the details about the boot process (as shown in Figure 6-1):

```
# systemd-analyze plot > plot.svg
```

Figure 6-1 A systemd-analyze plot

Note: SVG files can easily be viewed using your internet browser.

6.5.2 Retrieving information about unit dependencies

With **systemd**, there are two different types of dependencies between units:

- Dependencies that affect the activation of units (Requires/Wants/Conflicts).
- Dependencies that affect the order of units (After/Before).

The activation perspective (Requires/Wants/Conflicts)

To show units that the specified unit (for example, `sshd.service`) requires or wants, use the command shown in Example 6-17.

Example 6-17 Show units that the specified unit requires

```
# systemctl list-dependencies sshd.service
sshd.service
* |-system.slice
* `-sysinit.target
*   |-apparmor.service
*   |-dev-hugepages.mount
*   |-dev-mqueue.mount
*   |-friendly-recovery.service
*   |-iscsid.service
  ...
*   |-local-fs.target
*   | |-.mount
*   | |-boot.mount
*   | |-data.mount
*   | |-systemd-remount-fs.service
*   | |-test.mount
*   | |-tmp.mount
*   | `-var.mount
*   `-swap.target
```

To show units that require or want the specified unit (`for example`, `sshd.service`), use the following command:

```
# systemctl list-dependencies --reverse sshd.service
sshd.service
* `-multi-user.target
*   `-graphical.target
```

The order perspective (After/Before)

To list the units that the specified unit (for example, `sshd.service`) has an “after” dependency with, use the command shown in Example 6-18. In other words, the specified unit can’t start until *after* the units listed below it have started.

Example 6-18 List the order perspective (after)

```
# systemctl list-dependencies --after sshd.service
sshd.service
* |-auditd.service
* |-system.slice
* |-systemd-journald.socket
* |-basic.target
* | |-.mount
...
```

To list the units that the specified unit (for example, `sshd.service`) has a “before” dependency with, use the command shown in Example 6-19. In other words, the specified unit must start *before* the units listed below it can start.

Example 6-19 List the order perspective (before)

```
# systemctl list-dependencies --before sshd.service
sshd.service
* |-multi-user.target
* | |-systemd-update-utmp-runlevel.service
* | `-graphical.target
* |   |-systemd-update-utmp-runlevel.service
* |   |-ureadahead-stop.service
* |   `-ureadahead-stop.timer
* `-shutdown.target
*   |-systemd-halt.service
*   |-systemd-reboot.service
*   `-final.target
*     |-systemd-halt.service
*     `-systemd-reboot.service
```

7

Miscellaneous recipes

"*Try not to become a man of success, but rather try to become a man of value.*"

— Albert Einstein

This chapter contains information that falls under the *miscellaneous* category. These are topics that help make administration easier, save time, increase functionality, or add capabilities to your systems.

It includes the following topics:

- Rescue a Linux system
- Set up Memory Hotplugging
- Dynamically add or remove virtual CPUs
- Use the cpuplugd service
- Hardware Cryptographic Support for OpenSSH using Ubuntu 16.04
- Use Crypto Express to seed /dev/random
- Graphical user interfaces
- Setting up the IUCV Linux Terminal Server
- Issue z/VM CP commands from Linux
- Access z/VM CMS disks from Linux
- NFS mounting the LNXADMIN SFS directory from Linux
- Customizing default system commands using update-alternatives
- Other resources and "recipes"

7.1 Rescue a Linux system

This section describes how to boot your Linux server into different modes for troubleshooting purposes. It covers booting Linux into single user mode, and also entering a rescue environment when you require more advanced troubleshooting.

7.1.1 The initrd shell and systemd targets

When using previous versions, Linux SysV offered special run levels that could be used for special tasks for example, the single user mode, or the emergency mode. The **systemd** introduces a new concept, called *targets*, which offers the same functionality but in a different way. For more information, review Chapter 6, “Working with systemd” on page 95.

7.1.2 Rescue Ubuntu 16.04 using systemd targets

The procedure to enter rescue mode using **systemd** targets in Ubuntu 16.04 is similar to that of other **systemd** systems. Follow these steps to access rescue mode:

1. Log on to the troubled guest via the 3270 console.
2. Answer no (N) to the question prompting you to IPL Linux from 100:

```
Do you want to IPL Linux from minidisk 100? y/n
==> N
```

3. Perform an IPL from the boot device with the PARM systemd.unit=rescue.target, as shown in Example 7-1.

Example 7-1 An IPL from the book device

```
==> ipl 100 PARM systemd.unit=rescue.target

...
[    0.397687] Built 1 zonelists in Node order, mobility grouping on.  Total
pag
es: 208656
[    0.397688] Policy zone: DMA
[    0.397690] Kernel command line:
root=UUID=f34d4b70-1fd7-4861-afd3-2453ce5f3d
69 crashkernel=196M BOOT_IMAGE=0 systemd.unit=rescue.target
...

Welcome to rescue mode! After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or ^D to
boot into default mode.
Press Enter for maintenance
(or press Control-D to continue):

ENTER (May need to be entered twice)
```

You can now attempt to fix the problems that are preventing your system from starting successfully. When you are done, you can exit single user mode by using the **systemctl default** command to continue to boot normally to the default systemd target in your system.

Alternatively, you can use the **systemctl reboot** command to reboot the system and remain in rescue mode.

Note: In `rescue.target` mode, all of the file systems in `/etc/fstab` are mounted, but networking is not started.

To enter a different **systemd** target, from the IPL command, type the target that you want in the **systemd.unit=** parameter.

7.2 Set up Memory Hotplugging

Linux Memory Hotplug allows the amount of memory in a Linux system to be increased or decreased without a reboot. You must first have standby memory that is defined to the virtual machine in which Linux is running. You can issue the **CP DEFINE STORAGE** command to configure standby memory (storage).

To set up standby storage for Linux Memory Hotplug, using, for example, `LINUX01` as the virtual machine, complete the following steps:

1. To give the virtual machine an additional 1 GB of standby memory, this change can be done either in the directory level or per specific virtual machine using CMS command-line mode. The following examples show how to change a specific Linux virtual machine to define a standby memory using the CMS command. In addition, the second example shows you how to make the changes in the directory, and then how the Linux guest loads the changes:
   - If you want this to be a temporary change for this bootup of Linux only, or for some other reason decide not to change your virtual machine's directory entry, then from a 3270 terminal type the **DEFINE** statement, as shown in Example 7-2.

     Example 7-2 Use the DEFINE statement

     ```
     ===> define storage 1GB standby 1GB
     00: HCPZPM003E Invalid option - 1GB
     Ready(00003); T=0.01/0.01 14:53:04
     define storage 1G  standby 1G
     00: STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0
     00: Storage cleared - system reset.
     ===> ipl 100
     ```

   - If you want to make this a permanent change, you must update the user directory entry for this virtual machine. To do so, see *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147. After you make directory changes, log on to `LINUX1`. You should see the standby memory reported, as shown in Example 7-3.

     Example 7-3 Standby memory reported

     ```
     LOGON LNXS0008
     00: z/VM Version 6 Release 3.0, Service Level 1501 (64-bit),
     00: built on IBM Virtualization Technology
     00: There is no logmsg data
     00: FILES: 0003 RDR,   NO PRT,   NO PUN
     00: LOGON AT 15:27:29 EDT MONDAY 04/27/15
     00: STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0
     00: Storage cleared - system reset.
     00: Command complete
     ```

```
00: NIC 0600 is created; devices 0600-0602 defined
00: NIC 0600 is connected to VSWITCH SYSTEM VSW1
z/VM V6.3.0    2015-04-09 09:04
LINUX1   AT ITSOZVM1 VIA RSCS     2015-04-27 15:27:30 EDT      MONDAY
DMSACR723I D (LNX:LNXADMIN.) R/O
DIAG swap disk defined at virtual address 0300 (64988 4K pages of swap
space)
DIAG swap disk defined at virtual address 0301 (129980 4K pages of swap
space)
DO YOU WANT TO IPL LINUX FROM MINIDISK 100? Y/N
Y
```

2. Log in temporarily as root:

   $ sudo -i

3. Start an SSH session as root and view the memory in the /sys/ file system. Change directory to /sys/devices/system/memory/ and list the files, as shown in Example 7-4.

   Example 7-4 List the files

```
# lsmem
Address Range                           Size (MB)  State    Removable  Device
==========================================================================
0x0000000000000000-0x000000000fffffff        256  online   no         0-127
0x0000000010000000-0x000000001fffffff        256  online   yes        128-255
0x0000000020000000-0x000000003fffffff        512  online   no         256-511
0x0000000040000000-0x000000007fffffff       1024  offline  -          512-1023

Memory device size  : 2 MB
Memory block size   : 256 MB
Total online memory : 1024 MB
Total offline memory: 1024 MB

Here you see that we have 1024MB of offline memory and memory block size of
256MB. The Total offline memory indicates what we have available to be enabled.
The memory block size represent the increments on which we can add or remove
memory from our running system.
```

4. You can also show information about memory with the **free -m** command:

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:            792          77         596           1         119         672
Swap:           761           0         761
```

   This shows 596 MB of free memory available (some of the memory is used internally by Linux).

5. You can turn on memory by using the **chmem** command. Turn on an extra 512 MB of memory with the following commands:

```
# chmem -e 512M
```

6. Show the current online memory again using the **lsmem** command, as shown in Example 7-5.

Example 7-5 Show the current online memory (using the lsmem command)

```
# lsmem
Address Range                           Size (MB)  State    Removable  Device
==========================================================================
0x0000000000000000-0x000000000fffffff        256  online   no         0-127
0x0000000010000000-0x000000001fffffff        256  online   yes        128-255
0x0000000020000000-0x000000003fffffff        512  online   no         256-511
0x0000000040000000-0x000000005fffffff        512  online   yes        512-767
0x0000000060000000-0x000000007fffffff        512  offline  -          768-1023

Memory device size  : 2 MB
Memory block size   : 256 MB
Total online memory : 1536 MB
Total offline memory: 512 MB
```

7. Again, confirm with the **free -m** command, which shows that 1104 MB of free memory is now available:

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:           1304          77        1104           1         121        1179
Swap:           761           0         761
```

8. You can also give back memory that is removable again with the **chmem** command:

```
# chmem -d 256M
```

9. Verify that the memory is returned, as shown in Example 7-6.

Example 7-6 Verify that memory is returned

```
# lsmem
Address Range                           Size (MB)  State    Removable  Device
==========================================================================
0x0000000000000000-0x000000000fffffff        256  online   no         0-127
0x0000000010000000-0x000000001fffffff        256  online   yes        128-255
0x0000000020000000-0x000000003fffffff        512  online   no         256-511
0x0000000040000000-0x000000004fffffff        256  online   yes        512-639
0x0000000050000000-0x000000007fffffff        768  offline  -          640-1023

Memory device size  : 2 MB
Memory block size   : 256 MB
Total online memory : 1280 MB
Total offline memory: 768 MB

# free -m
              total        used        free      shared  buff/cache   available
Mem:           1048          77         849           1         121         926
Swap:           761           0         761
```

This section has shown how to configure virtual machines with standby memory and how to “hot-plug” the memory from Linux. This capability requires some advance planning on the part of the system administrator. By using this capability, you can increase your system’s performance and availability.

7.3 Dynamically add or remove virtual CPUs

Just as you did for memory, you can add or remove CPUs dynamically to a running Linux guest:

1. Query the number of virtual CPUs available to the Linux instance:

```
# vmcp q v cpus
CPU 01  ID  FF1E32C728278000 CP   CPUAFF ON
CPU 00  ID  FF1E32C728278000 (BASE) CP   CPUAFF ON
```

   This shows that there are currently two virtual CPUs defined to the Linux guest; CPUs 00 and 01.

2. Display the number of virtual CPUs (vCPUs) currently online to Linux using the **lscpu** command, as shown in Example 7-7.

Example 7-7 Display the number of vCPUs currently online

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               2
NUMA node(s):          1
Vendor ID:             IBM/S390
BogoMIPS:              18115.00
Hypervisor:            z/VM 6.2.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             96K
L1i cache:             64K
L2d cache:             1024K
L2i cache:             1024K
NUMA node0 CPU(s):     0-63
Flags:                 esan3 zarch stfle msa ldisp eimm dfp etf3eh highgprs

# lscpu -ae
CPU NODE BOOK SOCKET CORE L1d:L1i:L2d:L2i ONLINE CONFIGURED POLARIZATION ADDRESS
0   0    0    0      0    0:0:0:0         yes    yes        horizontal   0
1   0    1    1      1    1:1:1:1         yes    yes        horizontal   1
```

   This command confirms that there are two vCPUs enabled from a Linux point of view.

3. Dynamically define additional vCPUs to the Linux guest:

```
# vmcp define cpu 02 type cp
CPU 02 defined
```

4. Again, query the vCPUs available to the Linux guest:

```
# vmcp q v cpus
CPU 00  ID  FF1E32C728278000 (BASE) CP   CPUAFF ON
CPU 01  ID  FF1E32C728278000 CP   CPUAFF ON
CPU 02  ID  FF1E32C728278000 STOPPED CP   CPUAFF ON
```

The previous command shows that the new virtual CP is in a STOPPED state. This is because this CP has not been “turned on” from a Linux perspective. In order to do this, we must first do a rescan for CPUs.

5. Rescan for vCPUs available using the **chcpu** command:

```
# chcpu -r
Triggered rescan of CPUs
```

6. Verify that the new CPU has been discovered:

```
# lscpu -ae
CPU NODE BOOK SOCKET CORE L1d:L1i:L2d:L2i ONLINE CONFIGURED POLARIZATION ADDRESS
0   0    0    0      0    0:0:0:0         yes    yes        horizontal   0
1   0    1    1      1    1:1:1:1         yes    yes        horizontal   1
2   0    -    -      -    :::             no     yes        horizontal   2
```

Here we can see that CPU 02 has been configured but is not online.

7. Enable CPU 02 by using the **chcpu -e** command:

```
# chcpu -e 2
CPU 2 enabled

# lscpu -ae
CPU NODE BOOK SOCKET CORE L1d:L1i:L2d:L2i ONLINE CONFIGURED POLARIZATION ADDRESS
0   0    0    0      0    0:0:0:0         yes    yes        horizontal   0
1   0    1    1      1    1:1:1:1         yes    yes        horizontal   1
2   0    2    2      2    2:2:2:2         yes    yes        horizontal   2
```

This confirms that CPU 02 has been successfully brought online.

8. If you want to disable a vCPU, you can again use the **chcpu -d** command:

```
# chcpu -d 2
CPU 2 disabled

# lscpu -ae
CPU NODE BOOK SOCKET CORE L1d:L1i:L2d:L2i ONLINE CONFIGURED POLARIZATION ADDRESS
0   0    0    0      0    0:0:0:0         yes    yes        horizontal   0
1   0    1    1      1    1:1:1:1         yes    yes        horizontal   1
2   0    -    -      -    :::             no     yes        horizontal   2
```

This confirms that CPU 02 has been successfully taken offline.

Following this process allows you to manually add or remove virtual CPUs dynamically to a running system.

7.4 Use the cpuplugd service

The **cpuplugd** service allows Linux to enable or disable CPUs and memory, based on a set of rules. It can improve performance by setting the correct number of processors and amount of memory for Linux systems, depending on their current load. It can also prevent the Linux scheduler from queue balancing in partial load situations.

7.4.1 Determine the virtual CPUs being used

To start working with **cpuplugd**:

1. Determine whether the **cpuplugd** package is installed, as shown in Example 7-8.

Example 7-8 Determine whether the cpuplugd package is installed

```
# dpkg -l *cpuplugd*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                          Version             Architecture        Description
+++-=============================-===================-===================-=======================================
un  s390-tools-cpuplugd          <none>              <none>              (no description available)
```

The un prefix indicates that the package is not installed. So use **apt-get** to install it, as shown in Example 7-9.

Example 7-9 Install the package

```
# apt-get install s390-tools-cpuplugd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  s390-tools-cpuplugd
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 27.1 kB of archives.
After this operation, 91.1 kB of additional disk space will be used.
Get:1 http://us.ports.ubuntu.com/ubuntu-ports xenial/universe s390x
s390-tools-cpuplugd s390x 1.34.0-0ubuntu8 [27.1 kB]
Fetched 27.1 kB in 0s (364 kB/s)
Selecting previously unselected package s390-tools-cpuplugd.
(Reading database ... 46997 files and directories currently installed.)
Preparing to unpack .../s390-tools-cpuplugd_1.34.0-0ubuntu8_s390x.deb ...
Unpacking s390-tools-cpuplugd (1.34.0-0ubuntu8) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up s390-tools-cpuplugd (1.34.0-0ubuntu8) ...
```

Check again to see if it got installed, as shown in Example 7-10.

Example 7-10 Verify that the package is installed

```
# dpkg -l *cpuplugd*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                    Version          Architecture      Description
+++-=======================-================-================-=================================================
ii  s390-tools-cpuplugd     1.34.0-0ubuntu8  s390x             cpuplugd utility for Linux on z Systems
```

The ii prefix indicates that the package is installed.

2. Check the status of the **cpuplugd** service, as shown in Example 7-11.

Example 7-11 Check the status of the service

```
# systemctl status cpuplugd
* cpuplugd.service - CPU and memory hotplug daemon for Linux on z Systems
   Loaded: loaded (/lib/systemd/system/cpuplugd.service; enabled; vendor
preset: enabled)
```

```
   Active: active (running) since Tue 2016-06-14 12:40:04 EDT; 10min ago
 Main PID: 3061 (cpuplugd)
    Tasks: 1
   Memory: 160.0K
      CPU: 187ms
   CGroup: /system.slice/cpuplugd.service
           `-3061 /usr/sbin/cpuplugd -f -c /etc/cpuplugd.conf
```

3. Wait a few minutes and run the **lscpu** script again, as shown in Example 7-12.

Example 7-12 Run the lscpu script again

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0
Off-line CPU(s) list:  1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               1
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             128K
L1i cache:             96K
L2d cache:             2048K
L2i cache:             2048K
```

The output shows that now only one of the two virtual CPUs are active. The **cpuplugd** service turned off the other one.

4. The **cpuplugd** configuration file is `/etc/sysconfig/cpuplugd`. Some middleware products recommend a minimum of two virtual processors. If most of your Linux servers will be running a workload that recommends two processors, change the default for `CPU_MIN` to 2. An exception would be when only a single physical processor is available.

   View the non-comments and lines that are not blank in the configuration file with the command shown in Example 7-13.

Example 7-13 View the non-comments and lines that are not blank in the configuration file

```
# cd /etc/sysconfig
# egrep -v '^$|^#' cpuplugd
CPU_MIN="1"
CPU_MAX="0"
UPDATE="1"
CMM_MIN="0"
CMM_MAX="131072"        # 512 MB
pgscan_d="vmstat.pgscan_direct_dma[0] + vmstat.pgscan_direct_normal[0] +
vmstat.       pgscan_direct_movable[0]"
pgscan_d1="vmstat.pgscan_direct_dma[1] + vmstat.pgscan_direct_normal[1] +
vmstat       .pgscan_direct_movable[1]"
```

```
pgscanrate="(pgscan_d - pgscan_d1) / (cpustat.total_ticks[0] -
cpustat.total_tic       ks[1])"
avail_cache="meminfo.Cached - meminfo.Shmem"
user_0="(cpustat.user[0] - cpustat.user[1])"
nice_0="(cpustat.nice[0] - cpustat.nice[1])"
system_0="(cpustat.system[0] - cpustat.system[1])"
user_2="(cpustat.user[2] - cpustat.user[3])"
nice_2="(cpustat.nice[2] - cpustat.nice[3])"
system_2="(cpustat.system[2] - cpustat.system[3])"
CP_Active0="(user_0 + nice_0 + system_0) / (cpustat.total_ticks[0] -
cpustat.tot       al_ticks[1])"
CP_Active2="(user_2 + nice_2 + system_2) / (cpustat.total_ticks[2] -
cpustat.tot       al_ticks[3])"
CP_ActiveAVG="(CP_Active0+CP_Active2) / 2"
idle_0="(cpustat.idle[0] - cpustat.idle[1])"
iowait_0="(cpustat.iowait[0] - cpustat.iowait[1])"
idle_2="(cpustat.idle[2] - cpustat.idle[3])"
iowait_2="(cpustat.iowait[2] - cpustat.iowait[3])"
CP_idle0="(idle_0 + iowait_0) / (cpustat.total_ticks[0] -
cpustat.total_ticks[1]       )"
CP_idle2="(idle_2 + iowait_2) / (cpustat.total_ticks[2] -
cpustat.total_ticks[3]       )"
CP_idleAVG="(CP_idle0 + CP_idle2) / 2"
CMM_INC="meminfo.MemFree / 40"
CMM_DEC="meminfo.MemTotal / 40"
HOTPLUG="((1 - CP_ActiveAVG) * onumcpus) < 0.08"
HOTUNPLUG="(CP_idleAVG * onumcpus) > 1.15"
MEMPLUG="0"
MEMUNPLUG="0"
```

The default rules for the plugging and unplugging of CPUs in the configuration file is as follows:

```
HOTPLUG="((1 - CP_ActiveAVG) * onumcpus) < 0.08"
HOTUNPLUG="(CP_idleAVG * onumcpus) > 1.15"
```

In this case, the variables in the statements have the following meaning:

`loadavg`	The current average CPU load.
`onumcpus`	The number of CPUs that are online.
`runable_proc`	The current number of processes that can be run.
`idle`	The current idle percentage.

These CPU hot plugging and unplugging values will be used in the next section. In the default setup, **`cpuplugd`** only makes changes to the virtual processor configuration. The auto adaptive adjustment of the memory using the `cmm` feature (module) is deactivated by default, and is also not available when running in a native LPAR environment.

7.4.2 Generating a workload to see cpuplugd work

You can now generate a workload to show how the **`cpuplugd`** turns on CPUs.

Important: Running the following command generates significant CPU use. Verify that there is not a mission-critical workload that is running on this z/VM LPAR, because this test might affect it. Also, be sure to kill the processes after seeing **`cpuplugd`** in action.

Complete the following steps:

1. Put 10 looping jobs in the background with the following **for** loop:

```
# for i in `seq 1 10`
> do
>   bash -c "cat /dev/zero > /dev/null" &
> done
[1] 2441
[2] 2442
[3] 2443
[4] 2444
[5] 2445
[6] 2446
[7] 2447
[8] 2448
[9] 2449
[10] 2453
```

2. See that the jobs are running (you can also use the **top** command):

```
# pstree -G | grep cat
    ··sshd···sshd···bash···50*[bash···cat]
```

3. Run the **lscpu** command. The following example shows that, after a minute or so, **cpuplugd** has started the other spare processor:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               2
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             128K
L1i cache:             96K
L2d cache:             2048K
L2i cache:             2048K
```

   After a few more minutes, all of the CPUs should be activated.

4. Stop the workloads created before using the **killall** command:

```
# killall cat
[1]   Exit 143                bash -c "cat /dev/zero > /dev/null"
[2]   Exit 143                bash -c "cat /dev/zero > /dev/null"
[3]   Exit 143                bash -c "cat /dev/zero > /dev/null"
[4]   Exit 143                bash -c "cat /dev/zero > /dev/null"
[5]   Exit 143                bash -c "cat /dev/zero > /dev/null"
...
[48]   Exit 143                bash -c "cat /dev/zero > /dev/null"
[49]-  Exit 143                bash -c "cat /dev/zero > /dev/null"
[50]+  Exit 143                bash -c "cat /dev/zero > /dev/null"
...
```

5. Run the **lscpu** command. The following example shows that, after a minute or so, **cpuplugd** has stopped the other processor:

```
# lscpu
Architecture:          s390x
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Big Endian
CPU(s):                2
On-line CPU(s) list:   0
Off-line CPU(s) list:  1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s) per book:    1
Book(s):               1
Vendor ID:             IBM/S390
BogoMIPS:              20325.00
Hypervisor:            z/VM 6.3.0
Hypervisor vendor:     IBM
Virtualization type:   full
Dispatching mode:      horizontal
L1d cache:             128K
L1i cache:             96K
L2d cache:             2048K
L2i cache:             2048K
```

7.4.3 Setting memory sizes with cpuplugd

Memory sizes can also be set by the **cpuplugd** service. However, unlike CPUs, there is no good generic default value. The following example is in the *Device Drivers* book:

```
MEMPLUG = "swaprate > freemem+10 & freemem+10 < apcr"
MEMUNPLUG = "swaprate < freemem + 10000"
```

However, this is just a starting point to explain the syntactical structure of a rule. Do not use this configuration in production. You should test any setting that you want to implement against a representative workload that your Linux systems will be running. Details are beyond the scope of this section.

You can find more information about **cpuplugd** in the manual *Device Drivers, Features, and Commands (Kernel 4.4),* SC33-8411 for Ubuntu at:

http://public.dhe.ibm.com/software/dw/linux390/docu/l4n4dd29.pdf

7.5 Hardware Cryptographic Support for OpenSSH using Ubuntu 16.04

This section shows how to copy a test file with OpenSSH, first without any crypto acceleration. Then, crypto acceleration for OpenSSH is enabled, and the same file is copied again. A much higher throughput rate should be observed. The prerequisite for using hardware cryptography is to have a firmware level of LIC 3863 installed on your z Systems CEC.

This section is based on the white paper *First experiences with hardware Cryptographic Support for OpenSSH with Linux for z Systems*, by Manfred Gnirss, Winfried Münch, Klaus Werner, and Arthur Winterling, at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101690

This section shows only a single example of crypto acceleration. For a much more complete and detailed analysis, see the referenced white paper.

To test copying a file with and without cryptographic acceleration, complete the following steps:

1. Start an SSH session as root to any Linux guest.
2. Create a 1 GB test file for copying in the /tmp/ directory:

```
# cd /tmp
# dd if=/dev/zero of=testdata.txt bs=1048576 count=1000
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 13.5595 s, 77.3 MB/s
```

3. Copy the file locally with the **scp** command, two times with specific encryption algorithms and one time without them, prefixing all with the **time** command:

```
# time scp -c 3des-cbc testdata.txt localhost:/dev/null
testdata.txt                                  100% 1000MB  27.8MB/s   00:36

real    0m39.799s
user    0m32.102s
sys     0m0.829s
```

4. Determine if the necessary cryptographic-related Debian packages (DEBs) are installed:

```
# dpkg -| grep openssl-ibmca
```

   No output shows that they are not installed.

You can then proceed with the following steps:

1. Install the DEBs with the **apt-get install** command, as shown in Example 7-14.

Example 7-14 Install the DEBs

```
# apt-get install openssl-ibmca libica2
Reading package lists... Done
Building dependency tree
Reading state information... Done
libica2 is already the newest version (2.6.1-1ubuntu1).
The following NEW packages will be installed:
  openssl-ibmca
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 16.9 kB of archives.
After this operation, 79.9 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ports.ubuntu.com xenial/universe s390x openssl-ibmca s390x
1.3.0-0ubuntu2 [16.9 kB]
Fetched 16.9 kB in 0s (138 kB/s)
Selecting previously unselected package openssl-ibmca.
(Reading database ... 55023 files and directories currently installed.)
Preparing to unpack .../openssl-ibmca_1.3.0-0ubuntu2_s390x.deb ...
Unpacking openssl-ibmca (1.3.0-0ubuntu2) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up openssl-ibmca (1.3.0-0ubuntu2) ...
```

2. Show the new DEBs with the command shown in Example 7-15.

Example 7-15 Show the new DEBs

```
# dpkg -l | egrep "libica|ibmca"
ii  libica-dev:s390x                2.6.1-1ubuntu1                s390x
hardware cryptography support for IBM System z hardware (dev package)
ii  libica-utils                    2.6.1-1ubuntu1                s390x
hardware cryptography support for Linux on z Systems (utils)
ri  libica2:s390x                  2.6.1-1ubuntu1                s390x
hardware cryptography support for IBM System z hardware
ii  openssl-ibmca                   1.3.0-0ubuntu2                s390x
libica based hardware acceleration engine for OpenSSL
```

3. Verify that CP Assist for Cryptographic Function (CPACF) operations are supported, as shown in Example 7-16.

Example 7-16 Verify CP Assist for CPACF operations

```
# icainfo
The following CP Assist for Cryptographic Function (CPACF)
operations are supported by libica on this system:
 function      | # hardware | #software
---------------+------------+-------------
         SHA-1 |    yes     |     yes
       SHA-224 |    yes     |     yes
       SHA-256 |    yes     |     yes
       SHA-384 |    yes     |     yes
       SHA-512 |    yes     |     yes
         GHASH |    yes     |      no
         P_RNG |    yes     |     yes
  DRBG-SHA-512 |    yes     |     yes
        RSA ME |     no     |     yes
       RSA CRT |     no     |     yes
       DES ECB |    yes     |     yes
       DES CBC |    yes     |     yes
       DES OFB |    yes     |      no
       DES CFB |    yes     |      no
       DES CTR |    yes     |      no
      DES CMAC |    yes     |      no
      3DES ECB |    yes     |     yes
      3DES CBC |    yes     |     yes
      3DES OFB |    yes     |      no
      3DES CFB |    yes     |      no
      3DES CTR |    yes     |      no
     3DES CMAC |    yes     |      no
       AES ECB |    yes     |     yes
       AES CBC |    yes     |     yes
       AES OFB |    yes     |      no
       AES CFB |    yes     |      no
       AES CTR |    yes     |      no
      AES CMAC |    yes     |      no
       AES XTS |    yes     |      no
```

You should now have the cryptographic packages installed on Ubuntu.

To configure cryptography on Ubuntu, complete the following steps:

1. Make a backup of the SSL configuration file, `/etc/ssl/openssl.cnf`:

```
# cd /etc/ssl
# cp openssl.cnf openssl.cnf.orig
```

2. Append the sample SSL configuration file in the package documentation to the actual SSL configuration file, `/etc/openssl.cnf`:

```
# cat /usr/share/doc/packages/openssl-ibmca/openssl.cnf.sample >> openssl.cnf
```

3. Edit the appended file and search for the line with the **openssl_conf** variable. Move that line from the bottom to the top and save the file, as shown in Example 7-17.

Example 7-17 Edit the appended file and search for the openssl_conf variable

```
# vi openssl.cnf
/openssl_conf
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd
openssl_conf = openssl_def
...
```

4. Rerun the same **scp** commands, as shown in Example 7-18.

Example 7-18 Rerun the scp commands

```
# cd /tmp
# time scp -c 3des-cbc testdata.txt localhost:/dev/null
testdata.txt                                  100% 1000MB 250.0MB/s   00:04

real    0m3.987s
user    0m3.189s
sys     0m0.646s
# time scp -c aes128-cbc testdata.txt localhost:/dev/null
testdata.txt                                  100% 1000MB 333.3MB/s   00:03

real    0m3.019s
user    0m2.194s
sys     0m0.648s
```

5. To check the usage of the hardware functions, run the **icastats** command shown in Example 7-19.

Example 7-19 Check the use of the hardware functions

```
# icastats
function     |          # hardware      |       # software
-------------+--------------------------+-------------------------
             |       ENC    CRYPT   DEC |        ENC    CRYPT   DEC
-------------+--------------------------+-------------------------
        SHA-1 |              0           |                0
```

```
      SHA-224 |               0          |               0
      SHA-256 |               0          |               0
      SHA-384 |               0          |               0
      SHA-512 |               0          |               0
        GHASH |               0          |               0
        P_RNG |               0          |               0
DRBG-SHA-512 |            3425          |               0
       RSA-ME |               0          |               1
      RSA-CRT |               0          |               1
      DES ECB |         0             0  |         0             0
      DES CBC |         0             0  |         0             0
      DES OFB |         0             0  |         0             0
      DES CFB |         0             0  |         0             0
      DES CTR |         0             0  |         0             0
     DES CMAC |         0             0  |         0             0
     3DES ECB |         0             0  |         0             0
     3DES CBC |        13             7  |         0             0
     3DES OFB |         0             0  |         0             0
     3DES CFB |         0             0  |         0             0
     3DES CTR |         0             0  |         0             0
    3DES CMAC |         0             0  |         0             0
      AES ECB |         0             0  |         0             0
      AES CBC |         2             2  |         0             0
      AES OFB |         0             0  |         0             0
      AES CFB |         0             0  |         0             0
      AES CTR |         0             0  |         0             0
     AES CMAC |         0             0  |         0             0
      AES XTS |         0             0  |         0             0
```

6. To set a preferred cipher for the connections, edit the `/etc/ssh/sshd_config` file and add the following line at the end of the configuration file:

```
# cd /etc/ssh
# vi sshd_config
...
Ciphers aes128-cbc
```

7. The changed default for the cipher improves the throughput:

```
# time scp testdata.txt localhost:/dev/null
testdata.txt                                  100% 1000MB 250.0MB/s   00:04

real    0m3.472s
user    0m2.655s
sys     0m0.656s
You should see an improved throughput as a result of using the cryptographic
hardware.
```

8. Delete the test file:

```
# rm /tmp/testdata.txt
```

7.6 Use Crypto Express to seed /dev/random

Linux has a difficult time getting enough entropy from low interrupts and keyboard events on mainframe computers in the same way as on newer distributed servers. Largely, this comes

from the fact that the upper interrupts are not used to seed the entropy pool, and usually there is no keyboard or mouse attached to the system. With Ubuntu Linux Enterprise Server, there is an entropy generator installed by default that increases entropy when needed.

To check the availability of entropy on a server, complete the following steps:

1. Log on to the server:

   ```
   # ssh LINUX4
   ```

2. Check the current entropy:

   ```
   # cat /proc/sys/kernel/random/entropy_avail
   3965
   ```

The values of the available entropy are always kept 128 - 4096. Therefore, 3965 is a good value. However, although **haveged** is likely to produce good entropy, some environments might want to go with the hardware random generator from the Crypto Express card. To check what happens without entropy generator, complete the following steps:

1. Start an SSH session as `root` to a Linux system.
2. Disable the **haveged** service:

   ```
   # systemctl stop haveged
   Shutting down haveged daemon                                                done
   ```

3. Run a small loop to see how the entropy develops over time:

   ```
   # for i in {1..20}; do cat /proc/sys/kernel/random/entropy_avail ; done
   2985
   2857
   ...
   681
   553
   ```

   The numbers can be explained with the page randomization of the Linux kernel. For each new process, the Linux kernel uses a small amount of entropy, which leads to a decrease of the available entropy. As entropy decreases, programs, such as **sshd** and web servers with SSL, might have issues.

To enable the hardware random number generator from the Crypto Express card, you need to create a dedicated crypto domain:

1. Log in as `MAINT`.
2. To check for an available crypto domain, complete the following steps:

   ```
   ===> query crypto ap
   AP 00 CEX4C  Domain 06 available    shared                unspecified
   ```

3. Dedicate a crypto domain to one Linux guest:

   ```
   ===> dirmaint for LINUX4 CRYPTO DOMAIN 6 APDED 0
   ```

4. Query the crypto device again:

   ```
   ===> query crypto ap
   AP 000 CEX5C  Domain 006 available    dedicated to LINUX4   dedication
   ```

5. Log on to the virtual machine and start Linux.
6. Start an SSH session as `root`.
7. Turn the **haveged** service off:

   ```
   # systemctl disable haveged
   # systemctl stop haveged
   ```

8. Make sure that the `libica2` package is installed:

```
# apt-get install libica2
```

9. Enable the service **`z90crypt`**:

```
# systemctl enable z90crypt
# systemctl start z90crypt
```

10. Check the availability of entropy with an endless loop:

```
# watch -n 0.1 cat /proc/sys/kernel/random/entropy_avail
```

11. Stop the program by pressing Ctrl+C.

This section described how to use the cryptographic hardware to increase entropy to generate numerous random bytes.

7.7 Graphical user interfaces

Although this book specifically covers server environments, it is helpful to understand the different considerations for both workstations and servers with respect to *graphical user interfaces* (GUIs). *Window managers*, and *graphical desktop environments* both fall under the broader category of GUI.

Window managers and graphical desktop environments are both defined and described in this section. However, it is important that you remember for the purposes of this discussion, the term *graphical user interface*, or *GUI*, refers to the entire category inclusively.

7.7.1 The X Window System

For many years, UNIX and UNIX-like operating systems, such as Linux, have been using the X Window System (commonly referred to as simply “X”). This system was designed to provide a client/server-based graphical environment that is hardware-independent and network-enabled. Linux systems currently use X.Org, which is an open source implementation of the mainstream X Window System.

The X communication protocol by its nature is not secure at all. For this reason, it is often used together with SSH protocol, which tunnels X11 traffic using encrypted (and therefore secure) communications.

7.7.2 Window managers and graphical desktop environments

X11 itself provides the ability to display graphics on raster display, nothing more. If the user wants to be able to move, resize, and otherwise manage windows, a *window manager*, such as CDE or TWM, is needed. There are many window managers available; some are lightweight where some are more robust. Window managers are typically quite basic and often rudimentary. Many times, especially in a workstation environment, people have an expectation of interacting with a full *graphical desktop environment* complete with menus, icons, taskbars, and so on. A graphical desktop environment is a much more feature-rich version of a window manager.

Workstations

When you have Linux installed on a workstation, a window manager is probably not enough, so using a graphical desktop environment is a good idea because it provides the functionality users have come to expect from a GUI-based environment. You might see a graphical

desktop environment, such as XFCE, Gnome, KDE, or BusyBox, to name just a few of the available options.

Servers

Regardless of platform, Linux distribution, or preferred GUI option, the authors of this book strongly discourage and caution against the installation of any window manager or graphical desktop environment on any production server, or a server that might become production. There are numerous reasons for this, but the following list includes a few important ones to consider:

- *Security*. The more packages you install typically translates to more packages that you must then keep updated to prevent security vulnerabilities. Furthermore, some GUIs create services and daemons that listen on well-known ports. These can easily become exposure vectors for exploitation.
- *Performance*. Servers are normally sized and tuned to run specific workloads. GUIs and graphical desktop environments are resource-intensive, and can easily create performance problems by starving your production workloads of needed system resources.
- *Agility*. Production systems should always use the *just enough* principle. Install only the packages that are absolutely vital to running your production work.
- *Pitfalls*. Window managers, graphical desktop environments, and GUIs have dependencies, such as:
  - The `apt-get` handles this for you during installation, but you might find that uninstalling is not so easy, and in some cases can be virtually impossible.
  - There might be unintended results that arise from packages that are installed to satisfy a dependency for the GUI or graphical desktop environments.

7.7.3 VNC Server

As mentioned earlier, the X Server is run where the mouse, keyboard, and monitor are located, which is on the workstation. In a nutshell, VNC Server provides a virtual workstation with all these peripherals (virtual). The VNC server starts an embedded X Server. Then, any X-based application can send its output to this X Server, regardless of if the applications are local or remote to the X Server.

One big advantage of VNC is that it is session-oriented. If communication to the VNC server is lost, a new connection is reestablished to the session as it was. Also, applications in a disconnected VNC session still continue to run.

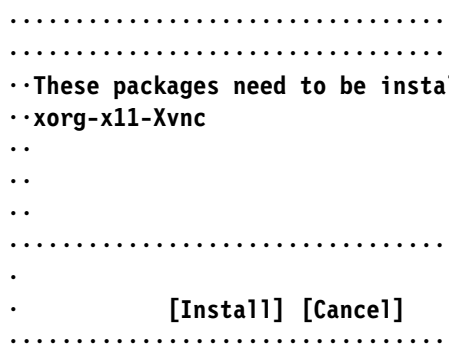
Setting up a VNC Server

Commonly it is not recommended to run X on a server machine. However, sometimes programs require an X Server to run. A VNC server can be set up with YaST:

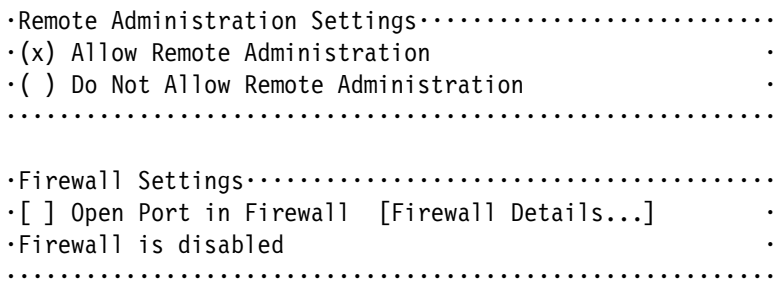
1. Start the yast remote module:

```
# yast
```

2. Change to **Network Services** → **Remote Administration (VNC)**:



3. Click Allow Remote Administration:



4. Click **OK**.
5. Change to **System -> /etc/sysconfig Editor**.
6. Select **Desktop -> Display manager**.
7. Make sure that `DISPLAYMANAGER_ROOT_LOGIN_REMOTE` is set to `yes`.
8. Confirm by clicking **Finish**.
9. Select **System -> Services Manager** and enable **`xinetd`**.

   Edit `/etc/xinet.d/vnc` and remove the line **`disable = yes`** for the entry `vnc1`. The file then looks as follows:

```
service vnc1
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = nobody
        server          = /usr/bin/Xvnc
        server_args     = -noreset -inetd -once -query localhost -geometry
1024x768 -securitytypes none
        type            = UNLISTED
        port            = 5901
}
```

7.7.4 X Server on a workstation

If for some reason VNC is not acceptable, it is possible to use a standard X Server on a workstation. Because Linux users usually know the X Window System, an X Server running on Microsoft Windows is described in this section.

Using embedded SSH to forward X11 with Ubuntu LTS

Support for X11 forwarding with SSH on Ubuntu should already be included in the installation. To verify that X11 forwarding is enabled on your Ubuntu Linux guest, run the command that is shown in Example 7-20 and ensure that the results are the same:

Example 7-20 Checking for X11 forwarding

```
lydiap@lnxub1:~$ which xauth
/usr/bin/xauth
```

If **xauth** is not installed, run the command shown in Example 7-21.

Example 7-21 Install X11 forwarding

```
lydiap@lnxub1:~$sudo apt-get install xauth
```

X11 forwarding needs to be enabled on both the client side and the server side:

- On the client side, `set X11 Forwarding yes` must specified in `~/.ssh/config`.
- On the server side, `set X11 Forwarding yes` must specified in `/etc/ssh/sshd_config`.

Using PuTTY

To use PuTTY for X11 forwarding, select **Enable X11 forwarding**, as shown in Figure 7-1.

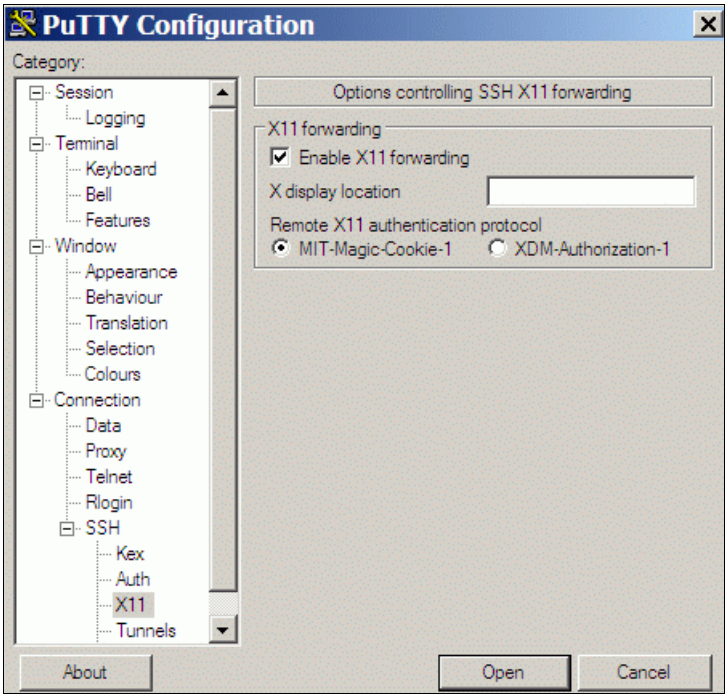


Figure 7-1 Enable X11 forwarding in PuTTY

When connected to a remote Linux system with X11 forwarding enabled, the `DISPLAY` environment variable contains the special value of **`localhost:10.0`**, which tells PuTTY to forward X11 protocol over SSH to an SSH client address. Remember that in order for PuTTY to work, you need an X Window Server running on your Microsoft Windows workstation.

Many commercial and free X Window servers are available for Microsoft Windows, which provides a free X Server that is based on Cygwin.

There are also many ways to achieve the same results. It is up to you to choose a solution that suits the purpose best.

7.8 Setting up the IUCV Linux Terminal Server

Implementation of a Linux Terminal Server based on z/VM inter-user communication vehicle (IUCV) allows access to the Linux console without a functioning TCP/IP stack on Linux.

Many IBM customers who run Linux under z/VM consider the implementation of this to be a fundamental requirement for a Linux Virtual Server to be eligible for classification as a production system in their environment. Additionally, it is based on a character mode interface, which enables the use of traditional Linux full-screen tools such as **vi**.

The official documentation for setup is on IBM.com at the following URL:

http://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_zvm_ht.html

Where the documentation provides many different options as to how the IUCV LTS can be set up, this section of the cookbook covers implementation using the **iucvtty** command.

Implementation of the LTS includes topics that involve changes on both z/VM and Linux. Topics in this section are as follows. Disregard any topics that do not apply to your Linux distribution:

- IBM z/VM configuration for Linux Terminal Server
- Ubuntu Linux Enterprise 12 configuration for IUCV consoles
- SUSE Linux Enterprise Server 12 configuration for IUCV Linux Terminal Server

7.8.1 IBM z/VM configuration for Linux Terminal Server

The **IUCV ALLOW** line allows virtual machines to connect to other virtual machines, such as the Linux Terminal Server, by using IUCV. The **IUCV ALLOW** line was included in the LNXPDFLT PROFILE entry, covered in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

7.8.2 Ubuntu Linux Enterprise 12 configuration for IUCV consoles

A login-protected console IUCV service is already configured by default. No further actions are needed to enable this.

7.8.3 SUSE Linux Enterprise Server 12 configuration for IUCV Linux Terminal Server

On SUSE Linux Enterprise Server 12, the IUCV HVC consoles are configured by default in the system. Therefore, it is only necessary to configure the terminal server. No changes are required on the systems that need to be connected to.

To configure Linux Terminal Server on SUSE Linux Enterprise Server 12:

1. Start an SSH session as root to LNXADMIN.
2. Run **yast** → **Network Services** → **IUCV Terminal Server**.
3. Select **z/VM IDs**, then press Enter and Tab to switch to the z/VM IDs input field.

4. Add all of the machines that LNXADMIN should connect to.
5. Click the IUCVConn tab.
6. Select **Enable IUCVConn on Login**.
7. Enter a password for the IUCVCONN user and confirm it.
8. Click **OK** to finish the configuration.
9. Start a new SSH session to LNXADMIN. Use one of the machines that were configured and is running Linux at that moment. The password for the connection is just the iucvconn password that was entered before:

```
# ssh linux4@lnxadmin.itso.ibm.com
login as: linux4
Using keyboard-interactive authentication.
Password:
iucvconn_on_login: Connecting to linux4 (terminal ID: lnxhvc0)
```

10. You are now connected to the IUCV terminal hvc0 of the z/VM guest LINUX4, and are prompted for a login. Note, that no network connection to that guest is needed at this time.
11. To end the connection, a special key sequence is required. The default for this is Ctrl+Shift+_+. (in words, this is the key combination Control+(Shift)+Underscore followed by a Period).

7.9 Issue z/VM CP commands from Linux

The **vmcp** command enables z/VM CP commands to be issued from Linux, as shown in the examples includes in Example 7-22.

Example 7-22 Examples of vmcp commands

```
# vmcp query v dasd
DASD 0100 3390 VV1569 R/W      10016 CYL ON DASD  1569 SUBCHANNEL = 0000
DASD 0120 3390 VV1560 R/O        140 CYL ON DASD  1560 SUBCHANNEL = 000D
DASD 0190 3390 VV1560 R/O        214 CYL ON DASD  1560 SUBCHANNEL = 0006
DASD 0191 3390 VV1560 R/O        500 CYL ON DASD  1560 SUBCHANNEL = 0009
DASD 019D 3390 VV1560 R/O        292 CYL ON DASD  1560 SUBCHANNEL = 0007
DASD 019E 3390 VV1560 R/O        500 CYL ON DASD  1560 SUBCHANNEL = 0008
DASD 0200 3390 VV156B R/W      10016 CYL ON DASD  156B SUBCHANNEL = 0001
DASD 0300 9336 (VDSK) R/W     524288 BLK ON DASD  VDSK SUBCHANNEL = 000E
DASD 0301 9336 (VDSK) R/W    1048576 BLK ON DASD  VDSK SUBCHANNEL = 000F

# vmcp query names
LINUX3   - SSI , LINUX4   - SSI , DIRMSAT2 - SSI
DATAMOVE - DSC , MAINT630 -LOOO6, LNXSERV1 - DSC , DIRMAINT - DSC
FTPSERVE - DSC , SSLSERVE - DSC , TCP/IP    - DSC , ZHCP     - DSC
XCAT     - DSC , VSMEVSRV - DSC , VSMREQIU - DSC , VSMREQI6 - DSC
VSMREQIN - DSC , DTCSMAPI - DSC , PERSMAPI - DSC , VSMWORK3 - DSC
VSMWORK2 - DSC , VSMWORK1 - DSC , VSMGUARD - DSC , LNXADMIN -LOOO7
DTCVSW2  - DSC , DTCVSW1  - DSC , VMSERVP  - DSC , VMSERVR  - DSC
VMSERVU  - DSC , VMSERVS  - DSC , OPERSYMP - DSC , DISKACNT - DSC
EREP     - DSC , OPERATOR - DSC , MAINT    -LOOO5, LINUX2   -LOOO9
VSM     - TCP/IP

# vmcp query v storage
STORAGE = 1G MAX = 2G INC = 2M STANDBY = 1G RESERVED = 0
```

```
# vmcp query vswitch details
VSWITCH SYSTEM VSW1     Type: QDIO    Connected: 3    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-0A-00-00-01    MAC Protection: Unspecified
...
```

You can find more information about CP commands in the *z/VM v6.3 CP Commands and Utilities Reference*, which is available at:

http://www.vm.ibm.com/library/zvmpdf.html

7.10 Access z/VM CMS disks from Linux

Data about z/VM CMS disks can be accessed using the CMS file system tools. This section describes how to mount the CMS file system tools using **cmsfs-fuse**.

7.10.1 Mount a CMS disk using cmsfs-fuse

To mount the CMS file system tools, install the following packages:

```
$ sudo apt-get install fuse s390-tools
```

Follow these steps:

1. Set the device with the CMS file system online:

```
# cio_ignore -r 0.0.0190
# chccwdev -e 0.0.0190
Setting device 0.0.0190 online
Done
# lsdasd 0.0.0120
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0190   active      dasdc     94:8    ECKD  4096   150MB     38520
```

2. Mount the CMS disk to `/mnt`:

```
# cmsfs-fuse -a -o ro /dev/disk/by-path/ccw-0.0.0190 /mnt
```

3. Access the data:

```
# ls /mnt/EDIT.EXEC
/mnt/EDIT.EXEC

# tail -n5 /mnt/EDIT.EXEC
&LOOP 3 &K
&IF &&I = LRECL  &&I = WIDTH
&IF &&I = NODISP &&I = NOSCREEN
&I = &I + 1
&GOTO -GO
```

4. Unmount the CMS disk:

```
# fusermount -u /mnt
```

5. Set the device with the CMS file system offline:

```
# chccwdev -d 0.0.0190
Setting device 0.0.0190 offline
Done
```

7.11 NFS mounting the LNXADMIN SFS directory from Linux

Although this would seldom be something that you might typically do, the authors of this book are including it for reference. If you do take advantage of this, it is advised that you unmount as soon as it is no longer required so that you do not affect SFS performance by having a litany of unnecessary NFS mounts open to the file pool.

The following code is an example entry that you might include in `/etc/fstab`:

```
itsovm1.itso.ibm.com:lnx:lnxadmin,lines=nl,trans=yes,userid=linux1,password=XXXXXX
XX /mnt/vmlnx/ nfs noauto,rsize=8192,wsize=8192,nosuid,timeo=14,soft,intr 0 0
```

7.12 Customizing default system commands using update-alternatives

Some people might prefer to use a text editor other than `nano`, which is the Ubuntu default. This is done using the **`update-alternatives`** command. The command can be run two different ways to produce the same result. The process of changing the default text editor from **`nano`** to **`vim`** is detailed using both methods as follows:

- Example 7-23 on page 137 shows how to perform the update interactively.
- Example 7-24 on page 138 shows how to perform the update non-interactively.

In Example 7-23, you might notice that there is an asterisk (*) to the left of selection `0` (zero). This indicates the current selection in use. Additionally, selection 0 specifically denotes the original default under most circumstances. We selected `vim.basic` by typing the number **3** then pressing the **ENTER** key.

Example 7-23 Changing the default text editor interactively

```
support@lnxadmin:~$ sudo update-alternatives --config editor
There are 4 choices for the alternative editor (providing /usr/bin/editor).

  Selection    Path                Priority   Status
------------------------------------------------------------
* 0            /bin/nano            40        auto mode
  1            /bin/ed             -100       manual mode
  2            /bin/nano            40        manual mode
  3            /usr/bin/vim.basic   30        manual mode
  4            /usr/bin/vim.tiny    10        manual mode

Press enter to keep the current choice[*], or type selection number: 3
ENTER
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/editor (editor)
in manual mode
support@lnxadmin:~$
```

Example 7-24 shows how to perform the same change shown in Example 7-23, but non-interactively, which can be helpful when including a shell script or when using service orchestration.

Example 7-24 Changing the default text editor non-interactively

```
support@lnxadmin:~$ sudo update-alternatives --set editor /usr/bin/vim.basic
pdate-alternatives: using /usr/bin/vim.basic to provide /usr/bin/editor (editor)
in manual mode

support@lnxadmin:~$
```

7.13 Other resources and “recipes”

You can find miscellaneous recipes that are either specific to z/VM or are generalized in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

Part 3

Appendixes

This section consists of the following appendixes:

- Appendix A, “References, cheat sheets, and blank worksheets” on page 141
- Appendix B, “Additional material” on page 143
- Appendix C, “Prerequisite network connectivity” on page 157

A

References, cheat sheets, and blank worksheets

This appendix refers to additional materials described or referred to in this book. Information about how to locate these reference materials is also provided.

IBM z/VM components and related products

See the appendix for *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

Cheat sheet for the Linux vi editor

The following list includes a small subset of the most commonly used **vi** commands. The **vi** editor has the following modes:

- *Input mode*. Select the Insert key, i, o (lowercase “o”, add a line below), O (uppercase “O”, add a line above), and other commands put you in this mode where you can type text into the file. When you are in this mode, you see the text `--INSERT--` in the last line.
- *Command mode*. Selecting Esc gets you out of input mode and into command mode. You can issue the following commands:

i	Brings you back to input mode
dd	Deletes a line and puts it in the buffer
<n>dd	Delete <n> lines
x	Delete a character
dw	Delete a word
p	Add the buffer past the current location
P	Add the buffer before the current location
o	Add a line and go into insert mode
/string	Search for string
n	Do the last command again; this command can be useful.
jkl;	Cursor movement
A	Add text at the end of the line
<nn>G	Go to line <*nn*>
G	Go to the last line in the file
yy	Yank a line (copy into buffer)
<n>yy	Yank n lines

- *Command line mode*. Pressing the colon (**:**) key brings you to this mode at the bottom of the window. You can issue the following commands:

:wq	Save (write and quit)
:q!	Quit and discard changes
:<nn>	Go to line number <*nn*>
:r <file>	Read <*file*> into the current file
:1,$s/old/new/g	Globally replace <*old*> with <*new*>
:help	Give help

Blank planning worksheets

For blank planning worksheets, see the sections “Planning for VMSSI with LGR” and “Appendixes” in *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

B

Additional material

This section refers to additional materials that can be downloaded from the Internet.

Locating the web material

The web material associated with this book is available on the Internet. You can find this at the following URL:

`http://www.vm.ibm.com/pubs/redbooks/sg248147/`

Using the web material

The files associated with this series of books are in a GNU compressed tar file (`.tgz`). To find information about the files included and how to extract and use them, consult *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

Linux code: The Ubuntu clone script

Example B-1 lists the code for the `/usr/sbin/clone` script that clones from an Ubuntu golden Linux image to a target virtual machine. It is contained in `clone-1.0-11.s390x.rpm`.

Example B-1 The Ubuntu clone script

```
#!/bin/sh
#
# clone.sh is a script that clones Linux images.  It makes use of vmcp to
# relay messages to the z/VM system and configuration files to modify
# the new image once it has been cloned.
#
# The script reads in /etc/sysconfig/clone for user setting customizations.
#
```

```
# For details on how this script works see the book:
#   "z/VM and Linux on IBM System z: The Virtualization Cookbook for
UbuntuUbuntu4"
#   on the Web at: http://www.redbooks.ibm.com/abstracts/sg247272.html
#
# ------------------------------------------------------------------------
# THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY
# WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY
# OR FITNESS FOR A PARTICULAR PURPOSE.
# NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY
# OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR
# DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED
# HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES
# ------------------------------------------------------------------------

# These MUST be lower case!
MASTER_LINK=fffe
CLONE_LINK=ffff

#+-------------------------------------------------------------------------+
function help
# give help
#+-------------------------------------------------------------------------+
 {
  echo "Usage: clone [-v] sourceID targetID [rootMinidisk [minidisk1
minidisk2..]]"
  echo "    Switches"
  echo "       -v Verbose output"
  echo "    Required"
  echo "       sourceID the z/VM user id you want to clone from"
  echo "       targetID  the z/VM user id you want to clone to"
  echo "    Optional"
  echo "       rootMinidisk the minidisk address that contains the root
filesystem"
  echo "       minidisk1..n additional minidisks that should be copied"
  exit
 }

#+-------------------------------------------------------------------------+
function cp_cmd
# echo a CP command and invoke it via cp_cmd
#   Arg1-n: the z/VM command to issue
#   Return: the z/VM command's return code
#+-------------------------------------------------------------------------+
 {
  [ -n "$VERBOSE" ] && echo "Invoking CP command: $@"
  out=$(vmcp $@ 2>&1)
  rc=$?

  # Pull the z/VM error code from the output
  if [ $rc -ne 0 ] ; then
```

```
   rc=$(echo $out | grep Error | sed s/.*#//g)
   [ -z "$rc" ] && rc=1
  fi
  return $rc
 }

#+-------------------------------------------------------------------------+
function copy_key
# If the host has a id_dsa.pub file then append that to the clone's
# authorized_keys file.
#+-------------------------------------------------------------------------+
 {
  if [ -e /root/.ssh/id_dsa.pub ] ; then
    [ ! -d /mnt/clone/root/.ssh/ ] && mkdir -p /mnt/clone/root/.ssh/
    echo "# LNXINST" >> /mnt/clone/root/.ssh/authorized_keys
    cat /root/.ssh/id_dsa.pub >> /mnt/clone/root/.ssh/authorized_keys
    chmod 600 /mnt/clone/root/.ssh/authorized_keys
  fi
 }

#+-------------------------------------------------------------------------+
function abort
# Exit the script and clean up
#+-------------------------------------------------------------------------+
 {
  umount_cloned_image

  set_offline $CLONE_LINK
  set_offline $MASTER_LINK

  unlink_one $CLONE_LINK
  unlink_one $MASTER_LINK

  exit $1
 }

#+-------------------------------------------------------------------------+
function get_target_info
# Get the TCP/IP and DNS info for the Linux ID to clone to.  This function
# will check both the shared.conf file and the specific target id's conf
# file.  If values are still missing then the user will be prompted to
# supply them.
#+-------------------------------------------------------------------------+
 {
  unset HOSTNAME
  [ -f /etc/clone/shared.conf ] && . /etc/clone/shared.conf
  [ -f /etc/clone/${target_linux_id}.conf ] && .
/etc/clone/${target_linux_id}.conf

  shift # drop the MasterGuestID
  shift # drop the CloneGuestID

  # If there are still command line arguments then the user must have specified
DASD
```

```
  # on the command line.  Unset whatever we have in DASD (from the config files)
and
  # set DASD equal to the rest of the arguments.
  [ $# -gt 0 ] && DASD="$@" && unset DASD_ROOT

  # Loop through all of the values that we require and double check that they have
  # values.  If they don't then we will prompt the user to fill them in.
  for v in HOSTNAME IPADDR DNS GATEWAY NETMASK MTU SUBCHANNELS SEARCHDNS NETTYPE
DASD
  do
   if [ -z "$(eval echo \$$v)" ]; then
    [ "$PROMPT" != "y" ] && echo "Error: missing required value for $v" && exit 1
    [ -z "$first" ] && echo "Please enter $target_linux_id's value for: " &&
first=1
    echo -n "$v: "
    read in
    eval $(echo $v=\"$in\")
    export $v
    echo "$v=$in" >> /etc/clone/${target_linux_id}.conf
   fi
  done

  # Expand DASD ranges if they have been defined
  if [ -n "$DASD" ] ; then
   split=$(echo $DASD | tr ',' ' ')
   DASD=""
   for s in $split
   do
    out=$(echo $s | grep \-)
    rc=$?
    [ $rc -eq 0 ] && DASD=${DASD}$(seq -s" " $(echo $s | tr '-' ' ' | tr '\n' '
'))
    [ $rc -ne 0 ] && DASD=${DASD}$(echo -n "$s ")
   done
   [ -n "$DASD_ROOT" ] && DASD=$(echo $DASD | sed "s/$DASD_ROOT//")
   DASD="$DASD_ROOT $DASD"
   # Assuming that if no DASD_ROOT is specified then the first DASD device will be
   # take as root
   if [ -z "$DASD_ROOT" ] ; then
     DASD_ROOT=$(echo $DASD | awk -F" " '{print $1}')
   fi
   export DASD
  fi

  # Grab just the hostname with out any DNS suffixes from the FQDN
  target_host=$(echo $target_fqhost | awk -F. '{print $1}')
 }


#+---------------------------------------------------------------------------+
function dd_copy
# Use the dd command to copy one disk to another
#   Arg 1: Source minidisk - assumed to be online
#   Arg 2: Target minidisk - must be brought online and dasdfmt'd
#+---------------------------------------------------------------------------+
```

```
 {
  ret_val=0

  source_mdisk=$1
  target_mdisk=$2

  # Bring the source and target devices online
  set_online $source_mdisk
  set_online $target_mdisk

  target_dev_node=`cat /proc/dasd/devices | grep "$target_mdisk(ECKD)" | awk '{
print $7 }'`
  source_dev_node=`cat /proc/dasd/devices | grep "$source_mdisk(ECKD)" | awk '{
print $7 }'`

  wait_for_device /dev/$target_dev_node
  ret_val=$?

  if [ $ret_val -eq 0 ] ; then
    [ -n "$VERBOSE" ] && echo "Invoking Linux command: dasdfmt -p -b 4096 -y -F -f
/dev/$target_dev_node"
    [ -n "$VERBOSE" ] && progress="-p"
    dasdfmt $progress -b 4096 -y -F -f /dev/$target_dev_node
    [ $? -ne 0 ] && echo "Error: dasdfmt failed" && ret_val=1
  fi

  if [ $ret_val -eq 0 ] ; then
    wait_for_device /dev/$source_dev_node
    ret_val=$?
  fi

  if [ $ret_val -eq 0 ] ; then
    nblks=`cat /proc/dasd/devices | grep $target_dev_node | awk '{ print $13 }'`
    [ -n "$VERBOSE" ] && \
    echo "Invoking Linux command: dd bs=4096 count=$nblks if=/dev/$source_dev_node
of=/dev/$target_dev_node"
    dd bs=4096 count=$nblks if=/dev/$source_dev_node of=/dev/$target_dev_node
>/dev/null
    [ $? -ne 0 ] && echo "Error: dd failed" && ret_val=1
  fi

  # Put the source and target devices offline
  set_offline $target_mdisk
  set_offline $source_mdisk

  return $ret_val
 }

#+---------------------------------------------------------------------------+
function link_one
# This will link one minidisk from another user id as the target minidisk
# address on the current z/VM user id with a link mode indicated by the
# 4th argument.
#
#   Arg1: Source z/VM ID
```

```
#   Arg2: Source minidisk virtual address
#   Arg3: Target minidisk virtual address
#   Arg4: Link mode (rr/w)
#+-------------------------------------------------------------------------+
 {
  source_id=$1
  source_mdisk=$2
  target_mdisk=$3
  link_mode=$4

  cp_cmd QUERY VIRTUAL $target_mdisk
  if [ $? != 40 ]; then
    cp_cmd DETACH $target_mdisk
  fi

  cp_cmd LINK $source_id $source_mdisk $target_mdisk $link_mode $LINK_PASSWD
  if [ $? != 0 ]; then
    echo "cp_cmd link $source_id $source_mdisk $target_mdisk $link_mode failed -
exiting"
    abort 1
  fi
  }

#+-------------------------------------------------------------------------+
function unlink_one
# This will unlink a minidisk from the current z/VM user id.
#   Arg1: The target minidisk to unlink
#+-------------------------------------------------------------------------+
 {
  cp_cmd DETACH $1
  return $?
 }

#+-------------------------------------------------------------------------+
function copy_one
# Try to use z/VM FLASHCOPY to copy one disk to another. If that fails,
#    call dd_copy() to fall back to the Linux DD command
#   Arg 1: Source minidisk
#   Arg 2: Target minidisk
#+-------------------------------------------------------------------------+
 {
  source_mdisk=$1
  target_mdisk=$2

  if [ "$CLONE_METHOD" == "AUTO" -o "$CLONE_METHOD" == "auto" ] ; then
    cp_cmd FLASHCOPY $source_mdisk 0 END $target_mdisk 0 END
    rc=$?
    if [ $rc -ne 0 ]; then # FLASHCOPY failed
      [ -n "$VERBOSE" ] && echo "FLASHCOPY $source_mdisk $target_mdisk failed with
$rc - using Linux dd"
    else
     return 0
    fi
  fi
```

```
  dd_copy $source_mdisk $target_mdisk
  [ $? -ne 0 ] && return 1
 }

#+-------------------------------------------------------------------------+
function copy_disks
# Call copy_one to copy each disk passed in as an argument.
#   Arg1-n: The minidisk address to copy
#+-------------------------------------------------------------------------+
 {
  [ -n "$VERBOSE" ] && echo "Copying minidisks..."
  while [ $# -gt 0 ]; do
    link_one $source_linux_id $1 $MASTER_LINK RR
    link_one $target_linux_id $1 $CLONE_LINK W
    copy_one $MASTER_LINK $CLONE_LINK
    [ $? -eq 0 ] && echo "$1 disk copied ..."
    unlink_one $MASTER_LINK
    unlink_one $CLONE_LINK
    shift
  done
 }

#+-------------------------------------------------------------------------+
function link_disks
# Call link_one to link each disk passed in as an argument.
#   Arg1-n: The minidisk address to link
#+-------------------------------------------------------------------------+
 {
  [ -n "$VERBOSE" ] && echo "Linking minidisks for LVM..."
  while [ $# -gt 0 ]; do
    link_one $target_linux_id $1 400$# W
    set_online 400$#
    [ $? -eq 0 ] && echo "$1 disk linked ..."
    shift
  done
 }

#+-------------------------------------------------------------------------+
function unlink_disks
# Call unlink_one to unlink each disk passed in as an argument.
#   Arg1-n: The minidisk address to unlink
#+-------------------------------------------------------------------------+
 {
  [ -n "$VERBOSE" ] && echo "Unlinking minidisks ..."
  while [ $# -gt 0 ]; do
    set_offline 400$#
    unlink_one 400$#
    [ $? -eq 0 ] && echo "$1 disk unlinked ..."
    shift
  done
 }

#+-------------------------------------------------------------------------+
function ask_are_you_sure
# Ask "Are you sure?" - if not, then exit
```

```
#+-------------------------------------------------------------------------+
 {
  echo ""
  echo "This will copy disks from $source_linux_id to $target_linux_id"
  echo "Host name will be:  $HOSTNAME"
  echo "IP address will be: $IPADDR"
  echo -n "Do you want to continue? (y/n): "
  read ans
  if [ $ans != "y" ]; then
    abort 1
  fi
 }

#+-------------------------------------------------------------------------+
function check_logged_off
# Verify the user ID exists and is logged off
#   Arg1: The user id to query if it is logged on or not
#+-------------------------------------------------------------------------+
 {
  cp_cmd QUERY $1
  case $? in
    0)  # user ID is logged on or disconnected
      echo "$1 user ID must be logged off"
      exit 2
      ;;
    3)  # user ID does not exist
      echo "$1 user ID does not exist"
      exit 3
      ;;
   45) # user ID is logged off - this is correct
      ;;
   *) # unexpected
      echo "$1 user ID must exist and be logged off"
      exit 4
  esac
 }

#+-------------------------------------------------------------------------+
function modify_cloned_image
# Modify the networking information in appropriate files under /etc
# Regenerate SSH keys in golden image's /etc/ssh/ directory and change root pw
#+-------------------------------------------------------------------------+
 {
  source_ipaddr=$(grep IPADDR
$CLONE_MNT_PT/etc/sysconfig/network-scripts/ifcfg-eth0 \
        | awk -F= '{print $2}')
  source_hostname=$(grep HOSTNAME $CLONE_MNT_PT/etc/sysconfig/network \
        | awk -F= '{print $2}')
  source_host=$(echo $source_hostname| awk -F. '{print $1}')

  [ ! -d $CLONE_MNT_PT/etc ] && echo "Error: no $CLONE_MNT_PT/etc found" && abort
1

  [ -n "$VERBOSE" ] && echo "Modifying networking info under $CLONE_MNT_PT..."
  sed -i \
```

```
    -e "s/$source_ipaddr/$IPADDR/g" \
    -e "s/$source_hostname/$HOSTNAME/g" \
    -e "s/$source_host/$target_host/g" \
  $CLONE_MNT_PT/etc/hosts

  sed -i \
    -e "s/HOSTNAME=.*/HOSTNAME=$HOSTNAME/g"\
    -e "s/GATEWAY=.*/GATEWAY=$GATEWAY/g"\
  $CLONE_MNT_PT/etc/sysconfig/network

  sed -i \
    -e "s/IPADDR=.*/IPADDR=$IPADDR/g"\
    -e "s/MTU=.*/MTU=$MTU/g"\
    -e "s/NETMASK=.*/NETMASK=$NETMASK/g"\
    -e "s/SUBCHANNELS=.*/SUBCHANNELS=$SUBCHANNELS/g"\
    -e "s/NETTYPE=.*/NETTYPE=$NETTYPE/g"\
  $CLONE_MNT_PT/etc/sysconfig/network-scripts/ifcfg-eth0

  # Modify MACADDR/HWADDR if specified (optional)
  [ -n "$MACADDR" ] && sed -i -e "s/MACADDR=.*/MACADDR=$MACADDR/g" \
  $CLONE_MNT_PT/etc/sysconfig/network-scripts/ifcfg-eth0

  [ -n "$HWADDR" ] && sed -i -e "s/HWADDR=.*/HWADDR=$HWADDR/g" \
  $CLONE_MNT_PT/etc/sysconfig/network-scripts/ifcfg-eth0

  # Regenerate the SSH keys on the new clone's root filesystem
  [ -n "$VERBOSE" ] && echo "Regenerating SSH keys in $CLONE_MNT_PT/etc/ssh/ ..."
  rm -f $CLONE_MNT_PT/etc/ssh/ssh_host*
  ssh-keygen -t rsa -N "" -q -f $CLONE_MNT_PT/etc/ssh/ssh_host_rsa_key
  ssh-keygen -t dsa -N "" -q -f $CLONE_MNT_PT/etc/ssh/ssh_host_dsa_key
  ssh-keygen -t rsa1 -N "" -q -f $CLONE_MNT_PT/etc/ssh/ssh_host_key

  copy_key
 }


#+-----------------------------------------------------------------------------+
function set_online
# This will set online the target minidisk.
#   Arg1 - Minidisk virtual address to set online
#+-----------------------------------------------------------------------------+
 {
  local target_mdisk=$(echo $1 | tr 'A-Z' 'a-z')
  chccwdev -e 0.0.$target_mdisk >/dev/null
  rc=$?
  if [ $rc != 0 ]; then
    echo "Error: chccwdev -e 0.0.$target_mdisk failed with $rc - exiting"
    abort 1
  fi

  local target_dev_node=`cat /proc/dasd/devices | grep "$target_mdisk(ECKD)" | awk
'{ print $7 }'`
  if [ "$target_dev_node" = "" ]; then
    echo "Error: can't find $target_mdisk(ECKD) in /proc/dasd/devices - exiting"
    set_offline $target_mdisk
```

```
    abort 1
  fi
 }

#+-------------------------------------------------------------------------+
function set_offline
# This will set offline the target minidisk.
#   Arg1 - Minidisk virtual address to set offline
#+-------------------------------------------------------------------------+
 {
   target_mdisk=$(echo $1 | tr 'A-Z' 'a-z')
   chccwdev -d 0.0.$target_mdisk > /dev/null 2>&1
   rc=$?
   #if [ $rc -ne 0 ]; then
   # echo "Error: chccwdev -d 0.0.$1 failed with $rc - ignoring"
   #fi

   return $rc
 }

#+-------------------------------------------------------------------------+
function mount_cloned_image
# This will mount the cloned root filesystem.  It will pair a minidisk
# address to a device file and then mount the first partition.
#   Arg1: The minidisk address to mount
#+-------------------------------------------------------------------------+
 {
  target_mdisk=$1

  target_dev_node=`cat /proc/dasd/devices | grep "$target_mdisk(ECKD)" | awk '{
print $7 }'`

  wait_for_device /dev/${target_dev_node}1
  [ $? -ne 0 ] && echo "Error: timed out waiting for /dev/${target_dev_node}1" &&
abort 1

  /bin/mount /dev/${target_dev_node}1 $CLONE_MNT_PT
  [ $? -ne 0 ] && echo "Error: unable to mount cloned image" && abort 1

  /bin/mount | grep /dev/${target_dev_node}1 >/dev/null 2>&1
  [ $? -ne 0 ] && echo "Error: unable to mount cloned image" && abort 1

 }

#+-------------------------------------------------------------------------+
function mount_cloned_image_lvm
# This will mount the cloned root filesystem.  It will pair a minidisk
# address to a device file and then mount the first partition.
#   Arg1: The minidisk address to mount
#+-------------------------------------------------------------------------+
 {
  target_mdisk=$1

  /bin/mount /dev/$VG_NAME/$LV_ROOT $CLONE_MNT_PT
  [ $? -ne 0 ] && echo "Error: unable to mount cloned image" && abort 1
```

```
  /bin/mount | grep $LV_ROOT >/dev/null 2>&1
  [ $? -ne 0 ] && echo "Error: unable to mount cloned image" && abort 1

 }

#+-------------------------------------------------------------------------+
function umount_cloned_image
#   Unmount the cloned root filesystem
#+-------------------------------------------------------------------------+
 {
  /bin/umount $CLONE_MNT_PT >/dev/null 2>&1

  return $?
 }

#+-------------------------------------------------------------------------+
function check_for_conf
# Check that the configuration file exists for the ID that we are cloning to.
#+-------------------------------------------------------------------------+
 {
  if [ ! -f /etc/clone/${target_linux_id}.conf -a "$PROMPT" != "y" ]; then
   echo "Error: /etc/clone/${target_linux_id}.conf not found. Exiting"
   exit
  fi
 }

#+-------------------------------------------------------------------------+
function check_for_vmcp
# Check that the vmcp module is loaded and the vmcp binary is installed.
#+-------------------------------------------------------------------------+
 {
   # Check that vmcp exists and is executable
   [ ! -x /sbin/vmcp ] && echo "Error: can't find /sbin/vmcp" && exit

   # Load the vmcp kernel module if not already loaded
   if ! /sbin/lsmod | grep vmcp > /dev/null 2>&1 ; then
     if ! /sbin/modprobe vmcp > /dev/null 2>&1 ; then
       echo "Error: unable to load module vmcp, check kernel version"
       exit
     fi
   fi

  wait_for_device /dev/vmcp
  [ $? -ne 0 ] && echo "Error: timed out waiting for /dev/vmcp" && exit
 }

#+-------------------------------------------------------------------------+
function wait_for_device
# Sleep until a certain file exists
#   Arg1: The path of the file to sleep on.
#+-------------------------------------------------------------------------+
 {
  device=$1
```

```
  sleep 2
  for t in $(seq 1 20)
  do
    [ -e $device ] && return 0
    sleep 1
  done
  return 1
}

#+-------------------------------------------------------------------------+
function autolog
# Issue an XAUTOLOG command to bring up the new cloned image.
#+-------------------------------------------------------------------------+
 {
  cp_cmd XAUTOLOG $target_linux_id
  rc=$?
  if [ $? != 0 ]; then
    echo "xautolog $target_linux_id failed with $rc"
    return 0
  fi
  echo "Booting $target_linux_id"
 }

#+-------------------------------------------------------------------------+
# main()

# Only root can run this script
[ $(id -u) != "0" ] && echo "Error: you must be root" && exit

# Check if the user has defined any clone.sh configurations
[ -f /etc/sysconfig/clone ] && . /etc/sysconfig/clone

# Set defaults for clone.sh configurations
[ -z "$PROMPT" ] && PROMPT="y"
[ -z "$CLONE_MNT_PT" ] && CLONE_MNT_PT="/mnt/clone"

# If the clone mount point does not exist then we'll create it for you
[ ! -d $CLONE_MNT_PT ] && mkdir -p $CLONE_MNT_PT

# Check if -v was specified on the command line
if [ "$1" = "-v" ] ; then
  VERBOSE=1
  shift
fi

# If no command line options were provided show the help message
[ $# -eq 0 ] && help

# If one comand line option was provided show the help message
if [ $# -lt 2 ]; then
  echo "Error: incorrect number of arguments"
  help
fi

# Check that vmcp exists and the module is loaded
```

```
check_for_vmcp

# Allow UPPER or lower case source, target, blacklist entries.
# Convert all to lower case for consistency.
source_linux_id=$(echo $1 | tr "[:upper:]" "[:lower:]")
target_linux_id=$(echo $2 | tr "[:upper:]" "[:lower:]")

# Check the blacklist, which prevents using the master image as a target.
if [ -f /etc/clone/blacklist.conf ]; then
  . /etc/clone/blacklist.conf
  BlackList=$(echo ${BLACKLIST} | tr "[:upper:]" "[:lower:]")
  for Target in ${BlackList}
  do
    if [ "${Target}" == "${target_linux_id}" ]; then
      echo "${target_linux_id} is blacklisted!  Exiting!"
      exit
    fi
  done
fi

# Check that the master and clone z/VM IDs are logged off.
check_logged_off $source_linux_id
check_logged_off $target_linux_id

# Check that the clone's configuration file exists
check_for_conf

# Collect information from the clone's configuration file
get_target_info $@
[ "$PROMPT" = "y" ] && ask_are_you_sure

echo "Cloning $source_linux_id to $target_linux_id ..."
[ -z "$DASD" ] && echo "Error: no DASD defined in
/etc/clone/${target_linux_id}.conf" && exit
copy_disks $DASD

# Update the newly cloned image locally, so link, set online then mount the
# clone's root filesystem.  Then call modify_cloned_image to update
# configuration files with the proper settings.  Finally unmount,
# set offline and unlink the disk.
echo "Updating cloned image ..."
if [ -n "$VG_NAME" ]; then
  link_disks $DASD
  # FIXME wait for disks
  sleep 2
  /sbin/vgscan
  # FIXME wait for vgscan
  sleep 2
  /sbin/vgchange -a y $VG_NAME
  mount_cloned_image_lvm $CLONE_LINK
else
  link_one $target_linux_id $DASD_ROOT $CLONE_LINK W
  set_online $CLONE_LINK
  mount_cloned_image $CLONE_LINK
fi
```

```
modify_cloned_image
umount_cloned_image
if [ -n "$VG_NAME" ]; then
  /sbin/vgchange -a n $VG_NAME
  unlink_disks $DASD
else
  set_offline $CLONE_LINK
  unlink_one $CLONE_LINK
fi

# Autolog the clone unless AUTOLOG has been set to "n"
[ "$AUTOLOG" = "y" ] && autolog

echo "Successfully cloned $source_linux_id to $target_linux_id"
```

C

Prerequisite network connectivity

This appendix refers to fundamental network requirements that you will need in order to perform the installation steps referred to in this book. It includes the following topics:

- Connectivity
- Mirrors
- Additional information

Connectivity

When completing your planning worksheets, you determine the IP addresses to be used by the LNXADMIN virtual servers on each node of your cluster. At least one instance of LNXADMIN requires the ability to access several targets on the public Internet, more specifically, the ability to reach several different Canonical servers, as listed in Table C-1.

Table C-1 Mirror information

Purpose	Fully-qualified host & domain name	IPv4 IP address(es)
CD Image server	cdimage.ubuntu.com	91.189.88.39
Mirroring information management	launchpad.net	91.189.89.222 91.189.89.223
Archive (packages) source	ports.ubuntu.com	91.189.88.151

Although it might not always be the case, the authors of this book assume that your z Systems CPC is inside of a corporate network, with one or more firewalled zones separating the subnet that your OSA cards are cabled to from the public Internet.

Tip: Determine who your Network Administrator is and engage them early on in the planning of your z/VM deployment project. This is covered in further detail *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147.

If firewalls are in place, you will require assistance to create the correct openings. Regardless of firewalls, it is prudent to engage your network administrator. Not only can you draw upon their expertise, but you can prevent potential problems later on by ensuring that they agree with any configuration parameters, such as those for your VSWITCHes.

Mirrors

LNXADMIN requires the ability to *initiate* an outgoing TCP/IP connection on TCP/80 and TCP/443 to the following targets. LNXADMIN requires the ability to then receive incoming data that was requested with the initial outbound request.

The transaction that initially populates the local mirror is a long-running process. After that, transactions are much shorter, because only delta data is received.

Additional information

For more information about mirroring and release images, visit the following URLs:

- http://www.ubuntu.com/download/alternative-downloads
- http://wiki.ubuntu.com/Mirrors

Supplemental information regarding global mirrors

As of the time this book was authored, the *Ubuntu Server for IBM System/390x architecture* is not yet among the mainstream archive mirrors. It is our hope that as it gains popularity, it will eventually be. As of now, the following information is presented for anyone who may want to learn more about the global mirroring process.

https://launchpad.net/ubuntu/+archivemirrors

Know that not all mirrors contain all architectures or all packages, so you have to check:

- Look for a `Contents-s390x.gz` file under either of the following paths. Note that both paths may not exist:
  - `/ubuntu-ports/dists/xenial/`
  - `/ubuntu/dists/xenial/`
- Avoid choosing a mirror that is marked as being more than one day behind or unknown.
- Choose the closest mirror to you geographically with the fastest posted bandwidth speed.

Related publications and information

The publications and web sites listed in this section are considered particularly suitable for a more detailed discussion of topics covered in this series of books and/or this specific volume of the book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only:

- *Fibre Channel Protocol for Linux and z/VM on IBM System z*, SG24-7266
- *Security on z/VM*, SG24-7471
- *Sharing and maintaining Linux under z/VM*, REDP-4322
- *An Introduction to z/VM Single System Image (SSI) and Live Guest Relocation (LGR)*, SG24-8006
- *IBM Wave for z/VM Installation, Implementation, and Exploitation*, SG24-8192
- *Accounting and Monitoring for z/VM Linux guest machines*, REDP-3818
- *Systems Management APIs for z/VM*, REDP-3882
- *Introduction to the New Mainframe: z/VM Basics*, SG24-7316
- *Using z/VM for Test and Development Environments: A Roundup*, SG24-7355
- *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059
- *z/VM and Linux on IBM System z*, SG24-7492
- *Printing with Linux on zSeries Using CUPS and Samba*, REDP-3864

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *z/VM Performance Toolkit Guide*, SC24-6156-00
- *z/VM Performance Toolkit Reference*, SC24-6157-00

Online resources

These websites are also relevant as further information sources:

- IBM z Systems:

  ibm.com/systems/z

- IBM LinuxONE:

  ibm.com/systems/linuxone

Peer collaboration and shared community knowledge

- Linux for z Systems: The *Linux/390 project* website and associated wiki:

  http://linuxvm.org/
  http://wiki.linuxvm.org/

- List servers; including IBMVM, Linux-390, VM-UTILS, and more:

  ibm.com/vm/techinfo/listserv.html

IBM z/VM

- Installation

  ibm.com/vm/install

- Publications:

  ibm.com/vm/pubs

- Technical library:

  ibm.com/vm/library

- Security

  ibm.com/vm/security

- Performance:

  ibm.com/vm/perf

- Performance tips:

  ibm.com/vm/perf/tips

IBM Techdocs

- IBM Techdocs technical sales library:

  ibm.com/support/techdocs

Linux distributions

The following paragraphs describe various Linux distributions.

Red Hat

- Documentation for z Systems Linux Development stream:

  ibm.com/developerworks/linux/linux390/documentation_red_hat.html

- General information:

  http://www.redhat.com/en/resources/red-hat-enterprise-linux-ibm-system-z

- No-charge evaluation download for IBM z Systems:

  http://www.redhat.com/en/technologies/linux-platforms/enterprise-linux

SUSE

- Documentation for z Systems Linux Development stream:

  ibm.com/developerworks/linux/linux390/documentation_suse.html

- General information and evaluation download:

  http://www.suse.com/products/systemz

- SLES on IBM z Systems forum:

  http://forums.suse.com/forumdisplay.php?42-SLES-for-System-Z

Ubuntu

- Documentation for z Systems Linux Development stream:

  ibm.com/developerworks/linux/linux390/documentation_ubuntu.html

- Ubuntu for IBM z Systems and LinuxONE wikis:

  http://wiki.ubuntu.com/z
  http://wiki.ubuntu.com/S390X

- Ubuntu Linux Server Guide (LTS releases only):

  http://help.ubuntu.com/lts/serverguide

- No-charge download for IBM z Systems and LinuxONE:

  http://www.ubuntu.com/download/server/linuxone

Help from IBM

IBM Support Portal

ibm.com/support

IBM Global Services

ibm.com/services

IBM wants your input

Do you have a suggestion about how z/VM or a related product could be made even better? Is there a specific feature you would like to have? Would you like the opportunity to collaborate directly with the IBM product development teams and other product users? The RFE Community is the place.

What is RFE Community?

The RFE community is a place where you can collaborate directly with product management teams and other product users through your ability to search, view, comment on, submit, and track product *Requests For Enhancement* (RFEs).

How can the RFE community help you?

Using the RFE Community as the method for enhancement submission provides the following improvements:

- Eliminates multiple touch points that slow down communication in the enhancement process
- Increases the transparency in the development process for you
- Provides predictable response times for enhancement requests
- Empowers you to influence product direction and roadmaps and improve communication between end-users and development
- Provides a better tool for you to review other product enhancement ideas and cast a vote for your favorites
- Avoids the need to open a Problem Management Report (PMR) for a simple enhancement request
- Avoids the need to call Customer Support to determine the status of a specific RFE
- Provides the ability to comment on, and provide workarounds for RFEs that are created by other customers
- Provides the ability to see comments on RFEs from other customers

What are some of the popular features of the RFE community?

The following list includes some popular RFE community features:

- Online Submission of RFEs
- Browse RFEs by product
- Top 20 Watched RFEs
- Top 20 Voted RFEs
- Planned RFEs
- Delivered RFEs
- Online Searching for RFEs
- Vote on RFEs
- Watch RFEs
- Set Email or RSS feed notifications
- Comment on RFEs
- Start or Join a group to discuss RFEs

Join today

Visit the RFE Community on the developerWorks section of ibm.com at the following URL:

`https://www.ibm.com/developerworks/rfe/execute?use_case=changeRequestLanding`

Redbooks

The Virtualization Cookbook for IBM z Systems Volume 4: Ubuntu Server 16.04

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8354-00

ISBN 0738442003

Printed in U.S.A.