

Cloud Object Storage as a Service

IBM Cloud Object Storage from Theory to Practice

Anil Patil

Deepak Rangarao

Harald Seipp

Maciej Lasota

Reginaldo Marcelo dos Santos

Rob Markovic

Simon Casey

Stephen Bollers

Vasfi Gucer

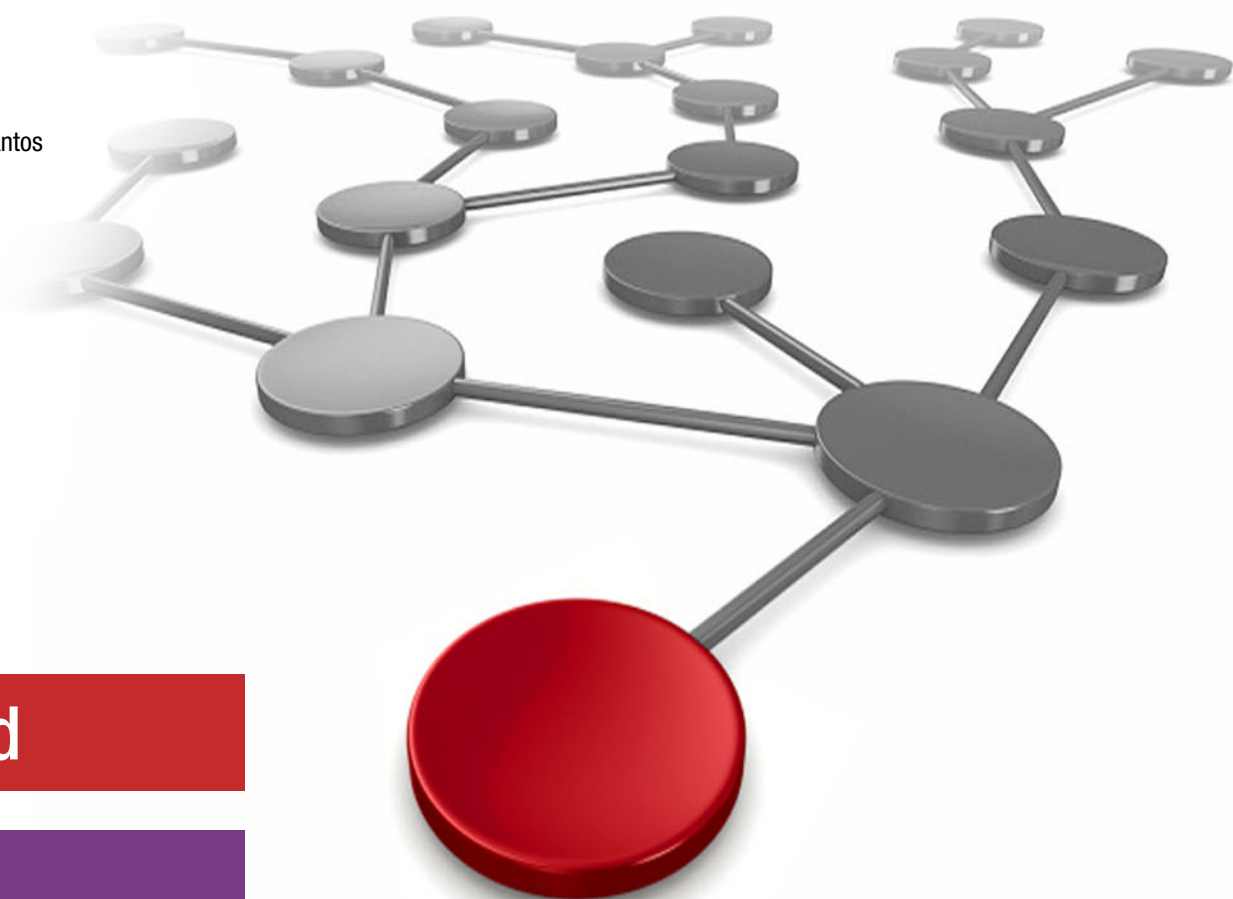
Andy Lin

Casey Richardson

Robert Rios

Ryan VanAlstine

Tim Medlin



 **Cloud**

Storage



International Technical Support Organization

**Cloud Object Storage as a Service: IBM Cloud Object
Storage from Theory to Practice**

March 2017

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

First Edition (March 2017)

This edition applies to IBM Cloud Object Storage public offering Version 1.0).

© Copyright International Business Machines Corporation 2017. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
Authors	x
Now you can become a published author, too!	xiv
Comments welcome	xv
Stay connected to IBM Redbooks	xv
Part 1. IBM Cloud Object Storage concepts	1
Chapter 1. Introduction to object storage	3
1.1 What object storage is	4
1.2 What storage business problems object storage can address	6
1.3 What qualifies business problems for object storage	6
1.4 Access methods for object storage	7
1.4.1 Access to storage through the REST API	7
1.4.2 Access to object storage through other methods	8
1.5 Comparison and positioning to other storage models	9
1.5.1 Block storage	9
1.5.2 File storage	10
1.6 Object storage key concepts	11
1.6.1 Buckets	11
1.6.2 Objects	11
1.6.3 Metadata	11
1.6.4 Access control lists	11
1.6.5 Object data protection	12
Chapter 2. Introducing IBM Cloud Object Store Public Services	13
2.1 Overview	14
2.2 Architecture	15
2.3 IBM Cloud Object Storage offerings	16
2.3.1 On-premises: Software only	16
2.3.2 On-premises: Software and appliance	17
2.3.3 Dedicated cloud-based	17
2.3.4 IBM COS Public Services	18
2.3.5 Comparison of IBM Cloud Object Storage deployment options	18
2.4 IBM COS Public Services capabilities	19
2.5 Programming model and interfaces	24
2.5.1 Simple Storage Service (S3) API	24
2.5.2 IBM COS API (future release)	24
Chapter 3. Getting started with IBM Cloud Object Storage	25
3.1 Setting up your cloud account	26
3.2 Credentials and endpoints	28
3.2.1 Public endpoints	29
3.2.2 Private endpoints	29
3.2.3 Custom Private Addressing (CPA) endpoint	29
3.2.4 Managing credentials	30

3.3 Managing buckets and objects	30
3.3.1 Creating and managing buckets from Bluemix Infrastructure	30
3.3.2 Managing objects from Bluemix Infrastructure	32
3.4 IBM COS S3 API and common functions	32
3.4.1 S3 and IBM COS S3 API differences	33
3.4.2 Supported API operations	34
3.4.3 Authorization and authentication	40
3.5 Clients, CLIs, and SDKs	40
3.5.1 Desktop clients	40
3.5.2 Command-line interfaces (CLIs)	46
3.5.3 Software development kits (SDKs)	50
3.6 Preferred practices for application design	53
3.6.1 Object name	53
3.6.2 Object size	53
3.6.3 Concurrent PUT/GET	53
3.6.4 Handling timeouts	53
3.6.5 Handling overwrites	53
3.6.6 Handling changes	53
Part 2. IBM Cloud Object Storage use cases and scenarios	55
Chapter 4. Typical applications for IBM Cloud Object Storage	57
4.1 IBM Cloud Object Storage workloads	58
4.2 Examples from various industries	58
4.2.1 Example 1: Business and financial management solutions	58
4.2.2 Example 2: Media and entertainment	61
4.2.3 Example 3: Education and research	63
4.2.4 Example 4: Healthcare industry and medical imaging	65
4.2.5 Example 5: Government authority managing road and traffic	68
4.2.6 Example 6: Telecommunications industry	70
4.2.7 Example 7: Insurance industry; cognitive damage assessment	72
4.3 Typical use cases	75
4.3.1 Use case 1: Backup repository	75
4.3.2 Use case 2: Internet of Things (IoT)	77
4.3.3 Use case 3: Active archive	78
4.3.4 Use case 4: Enterprise file services	81
4.3.5 Use case 5: Analytics and cognitive systems	85
4.3.6 Use case 6: Content repository	86
4.3.7 Use case 7: Enterprise collaboration	88
4.4 Integration with other IBM products	91
4.4.1 IBM Spectrum Protect	91
4.4.2 IBM Spectrum Scale	93
4.4.3 IBM Watson IoT	96
4.4.4 IBM Aspera	99
4.4.5 IBM Watson API on Bluemix platform	101
Chapter 5. Building a simple Web Gallery application	105
5.1 Scenario description	106
5.2 Architecture of the scenario	107
5.2.1 High-level architecture	107
5.3 Implementation of the scenario	111
5.3.1 Getting started	111
5.3.2 Ordering IBM COS on IBM Bluemix Infrastructure (SoftLayer)	111
5.3.3 IBM Bluemix account	112

5.3.4	Installing Node.js	112
5.3.5	Installing Git	112
5.3.6	Creating the Web Gallery app on IBM Bluemix.	113
5.3.7	Creating a Bluemix toolchain	118
5.3.8	Setting up IBM Cloud Object Storage credentials.	123
5.3.9	Developing a simple IBM Cloud Object Storage Web Gallery	124
5.3.10	Committing to Git	133
5.4	Verifying the scenario	134
5.5	Conclusion	138
Chapter 6.	Simple gateway scenario	139
6.1	Scenario description	140
6.2	Architecture of the scenario	140
6.2.1	Overview	140
6.2.2	Requirements	141
6.2.3	Planning	141
6.3	Implementation of the scenario	141
6.3.1	Pre-test verification	141
6.3.2	S3FS installation	142
6.3.3	S3FS configuration	146
6.4	Verification of the scenario	147
6.5	Conclusion	149
Chapter 7.	IBM Spectrum Protect integration.	151
7.1	Scenario description	152
7.2	Architecture of the scenario	152
7.2.1	Overview	152
7.2.2	Requirements	153
7.2.3	Planning	154
7.3	Implementation of the scenario	155
7.3.1	Pre-testing setup	156
7.3.2	IBM Spectrum Protect configuration	157
7.4	Verification of the scenario	167
7.5	Conclusion	169
Chapter 8.	VideoRecon Video Analytics	171
8.1	Scenario description	172
8.1.1	Business case	173
8.2	Architecture of the scenario	174
8.3	Implementation of the scenario	176
8.4	Verification of the scenario	180
8.5	Conclusion	182
Chapter 9.	Cognitive Call Center	183
9.1	Scenario description	184
9.1.1	Business case	184
9.2	Architecture of the scenario	185
9.3	Implementation of the scenario	186
9.4	Verification of the scenario	190
9.5	Conclusion	192
Appendix A.	Additional material	193
	Locating the web material	193
	Using the web material.	193

System requirements for downloading the web material	193
Downloading and extracting the web material	194
Related publications	195
IBM Redbooks	195
Online resources	195
Help from IBM	195

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.

AIX®	GPFS™	IBM Watson IoT™
Aspera®	IBM®	Power Systems™
Bluemix®	IBM Spectrum™	Redbooks®
Cloud Object Storage System™	IBM Spectrum Archive™	Redbooks (logo)  ®
Cloudant®	IBM Spectrum Protect™	Watson Analytics™
developerWorks®	IBM Spectrum Scale™	Watson IoT™
FASP®	IBM Watson®	

The following terms are trademarks of other companies:

Accesser, Cleversafe, Slicestor, and Storage Beyond Scale are trademarks or registered trademarks of Cleversafe, Inc., an IBM Company.

SoftLayer, and The Planet are trademarks or registered trademarks of SoftLayer, Inc., an IBM Company.

ITIL is a Registered Trade Mark of AXELOS Limited.

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.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

Preface

The digital enterprise has resulted in an explosion of data, and data volumes are expected to grow in zettabyte scale in the next few years. This explosive growth is largely fueled by unstructured data, such as video, social media, photos, and text. IBM® Cloud Object Storage (previously known as Cleversafe®) provides organizations the flexibility, scalability, and simplicity required to store, manage, and access today's rapidly growing unstructured data.

Cloud Object Storage (COS) provides access to your unstructured data via a self-service portal from anywhere in the world with RESTful APIs, including OpenStack Swift API and S3-compatible API, enterprise availability, and security.

IBM COS is available in the following deployment models:

- ▶ Private on-premises object storage
- ▶ Dedicated object storage (single-tenant)
- ▶ Public object storage (multi-tenant)
- ▶ Hybrid object storage (a mix of on-premises, dedicated or public offerings)

This IBM Redbooks® publication focuses on the IBM COS public offering, *IBM COS Public Services*, and hybrid solutions leveraging this offering.

This book is for solution developers, architects, and IT specialists who are implementing Cloud Object Storage solutions.

Authors

This book was produced by a team of specialists from IBM around the world working at the International Technical Support Organization, San Jose Center and also IBM Business Partner Mark III Systems Team working at their office in Austin/TX.

IBM Team:



Anil Patil is Certified IT Architect at IBM US. He is a Certified Cloud Solution Architect with IT experience in Cognitive Process (BPM/ODM), IBM Bluemix®, IBM Watson® API, and Middleware technologies. His core experience is in Java, WebService, Smarter Process, Integration, Bluemix, and API Development and Solution Architecture. He is currently an IT Architect for various clients in North America. Anil has been a technical contributor for various blogs such as IBM Watson IoT™, Cognitive, and Bluemix. He has also contributed to various IBM training materials such as IBM BPM, Cloud, and Bluemix.

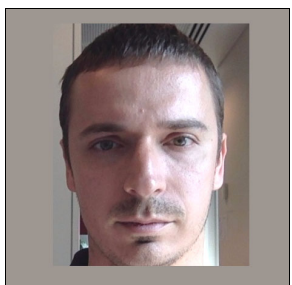


Deepak Rangarao is an Executive IT Specialist with the Global Analytics CTO Office at IBM. He has more than 18 years of experience in the telecommunications, public sector, finance, and insurance industries in both pre-sales and post-sales capacities. Deepak's key consulting experience includes hybrid cloud, data management, data warehousing, advanced analytics, and business intelligence solutions.

Deepak is a member of the IT Certification board at IBM, mentoring IT Specialists and Architects and is also a member of the Global Cloud Expertise Council at IBM, helping the field technical staff and customers understand the value of IBM Cloud. Deepak has a Masters Degree in Information Technology from RMIT, Australia and several developer certifications around Visual Basic, MS SQL Server (Microsoft Certified Professional), IBM Cloud Developer Certification, Apache Spark (O'Reilly Media).



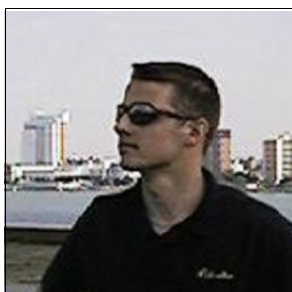
Harald Seipp works for IBM Germany and is the Founder and Technical Leader of the Center of Competence for OpenStack Storage as part of the EMEA Storage Competence Center, leading a virtual team, providing guidance to worldwide IBM teams across organizations, and working with customers and IBM Business Partners across EMEA to create and implement complex storage cloud architectures. Using his strong storage, software development, and open source technology background, he recently created innovative solutions that combine object storage with Bluemix based IoT and Watson technologies. He is working with IBM Research to create an innovative object storage solution that leverages high-latency media to provide a cost-effective cloud-based archive. His more than 25 years of technology experience includes previous job roles as Software Developer, Software Development Leader, Lead Developer, and Architect for successful software products, and co-inventor of an IBM storage product. He holds seven patents on storage and networking technology.



Maciej Lasota is a Client Technical Specialist in IBM Systems for Cloud in Australia. He has 10 years of broad experience in IT architecture and implementation. He holds a degree in Business Information systems from Swinburne University. His areas of expertise focus on enterprise infrastructure spanning Linux, IBM AIX®, SAN, NAS, object storage, and web development. He has been heavily involved in a number of object storage engagements across Australia and New Zealand and is considered an evangelist for object storage technologies.



Reginaldo Marcelo dos Santos is a Senior IT Specialist with IBM Virtual Integrated Organization (VIO) in Brazil. He is a Certified Cloud Platform Application Developer with 4 years of professional experience in IBM Bluemix technologies. He is currently a Cloud Developer, working on a high performance dashboard for internal executives in North America. His core experiences are in application development, mobile first, back-end, and front-end development. Reginaldo has been a technical speaker for IBM Watson APIs at various developer conferences. He is a mentor in the IBM cognitive community.



Rob Markovic works as the Lead System Engineer for Cloud Object Storage in IBM US. Rob brings over 20 years of experience and a focus on delivering innovative business solutions leveraging his industry experience in the fields of automobile, robotics, storage, virtualization, IT transformation and more.



Simon Casey is a Cloud Technical Consultant within IBM Cloud Professional Services with over 15 years of experience of the IBM compute and storage portfolio. Currently working in Cloud Professional Services, he is responsible for technical consultation for IBM and associated partner's cloud technologies such as orchestration, software-defined networking, and software-defined data centers. His core focus of the past 4 years has been in designing and building IaaS and PaaS platforms for public, private, and hybrid clouds across all industries and has been recognized with an Outstanding Innovation Award for his work in these areas.



Stephen Bollers is a Solutions Architect at IBM. He works with customers to provide guidance and translate their complex requirements into the best architectural designs on Bluemix infrastructure (SoftLayer®). He has over 18 years of experience in IT doing front-end and server-side web development, cloud services development, technical leadership, and application support. His past roles include Developer Architect, Business Analyst, Technical Specialist, and Technical Project Manager.



Vasfi Gucer is an IBM Technical Content Services Project Leader with the Digital Services Group. He has more than 20 years of experience in the areas of systems management, networking hardware, and software. He writes extensively and teaches IBM classes worldwide about IBM products. His focus has been primarily on cloud computing for the last 6 years. Vasfi is also an IBM Certified Senior IT Specialist, Project Management Professional (PMP), IT Infrastructure Library (ITIL) V2 Manager, and ITIL V3 Expert.

Mark III Systems Team:



Andy Lin currently serves as Vice President of Strategy at Mark III Systems, a 22-year IBM Platinum Business Partner based in Houston, TX. Andy is currently responsible for helping to drive strategy and execution for key initiatives company-wide, including as Co-founder of Mark III's digital development unit, BlueChasm. Andy is a two-time published IBM Redbooks author and current IBM Champion for Cloud and for Power Systems™. Andy also played a key role in Mark III being awarded the 2017 IBM Beacon Award for Outstanding Solution Developed on Bluemix and Outstanding Storage Solution, both of which involved the use of IBM Cloud Object Storage.



Casey Richardson serves as a Software Developer for BlueChasm, and is based in its Austin, TX headquarters. Having wanted to code ever since he was five years old, Casey wrote his first program in middle school in Visual Basic, and graduated to Java while in high school. Casey has played a huge role in BlueChasm's process of taking VideoRecon and Cognitive Call Center from initial prototypes to enterprise-hardened platforms. He is also the resident data scientist in the BlueChasm office and has a love for all things Scala.



Robert Rios serves as a Software Developer for BlueChasm, and is based in its Austin, TX headquarters. Robert grew up coding, having written his first script in middle school, which was designed to automate tasks on a Microsoft Windows computer. At BlueChasm, Robert has been the inspiration for many of our platforms and prototypes, including creating the first version of VideoRecon. Robert also open-sourced the community's first **npm** package for IBM Cloud Object Storage (Cleversafe) and has been active in writing about how to code efficiently on IBM Bluemix, IBM Cloud Object Storage, and on the IBM stack. Robert is an IBM Champion for Cloud and has led sessions at Watson Developer Conference, World of Watson, and InterConnect.



Ryan VanAlstine serves as CTO and Lead Developer for BlueChasm, the digital development unit of Mark III Systems. At BlueChasm, Ryan is charged with product and technical direction and leads the team of Austin-based developers in building platforms and prototypes, mostly on the IBM Cloud and Bluemix stacks. Ryan was a Cofounder of BlueChasm in late 2014 with Andy and is an IBM Champion for Cloud. He is also an active community member and advocate for Server-side Swift, much of it after serving as an iOS Consultant and working with both large companies and startups, prior to founding BlueChasm.



Tim Medlin currently serves as a Systems Architect for Mark III Systems, a 22-year IBM Platinum Business Partner based in Houston, TX. At Mark III, Tim is responsible for design, architecture, implementation, and ongoing support for Mark III clients across North America. At Mark III, Tim is subject-matter expert on IBM storage technologies like IBM Cloud Object Storage, IBM Spectrum™ Scale, Elastic Storage Server, and many others. He also played the key design and implementation role in the client use case that led to Mark III being awarded the 2017 IBM Beacon Award for Outstanding Storage Solution. In addition, Tim has a passion for code and is one of Mark III's leading experts in DevOps and automation.

Thanks to the following people for their contributions to this project:

Bertrand Dufrasne, Diane Sherman, Wade Wallace, Erica Wazewski
International Technical Support Organization

Riz Amanuddin, Rob Basham, Swami Chandrasekaran, Jeff Harrah, Brittany Heath, Del Hoobler, Nicholas Lange, Wesley Leggette, Michael Lyons, Dawn McKenna, Russ Kennedy, Michael Knieriemen, Anu Khera, Tim Kounadis, Niall McShane, Greg Palinckx, Jeffrey Palmer, Armen Pischdotchian, Arvind Sathi, John Walicki
IBM USA

Olaf Hahn
IBM Austria

Ryan Hay, Mark Jones
IBM Australia

Basil Moshous
IBM Germany

Michael Factor
IBM Israel

The team also would like express special thanks to **Andrew Trice** (Technical Product Manager, IBM Cloud IBM Watson and Cloud Platform) from IBM USA for documenting the scenario in 4.2.7, “Example 7: Insurance industry; cognitive damage assessment” on page 72.

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 from two to 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 at:

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 found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

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:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

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

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Part 1

IBM Cloud Object Storage concepts

This part introduces object storage concepts and discusses IBM Cloud Object Storage concepts.



Introduction to object storage

This chapter describes the concept of object storage and its benefits.

The following topics are covered in this chapter:

- ▶ 1.1, “What object storage is” on page 4
- ▶ 1.2, “What storage business problems object storage can address” on page 6
- ▶ 1.3, “What qualifies business problems for object storage” on page 6
- ▶ 1.4, “Access methods for object storage” on page 7
- ▶ 1.5, “Comparison and positioning to other storage models” on page 9
- ▶ 1.6, “Object storage key concepts” on page 11

1.1 What object storage is

Object storage is a modern storage technology concept and a logical progression from block and file storage. Object storage has been around since the late 1990s, but has gained market acceptance and success over the last 10 years.

Object storage was invented to overcome a number of issues:

- ▶ Managing data at scale by using conventional block and file systems was difficult because these technologies lead to *data islands* due to limitations on various levels of the data management hardware and software stack.
- ▶ Managing namespace at scale resulted in maintaining large and complex hierarchies, which are required to access the data. Limitations in nested structures on traditional block and file storage arrays further contributed to data islands being formed.
- ▶ Providing access security required a combination of technologies, complex security schemes, and significant human involvement in managing these areas.

Object storage, also known as *object-based storage* (OBS) uses a different approach to storing and referencing data. Object data storage concepts include the following three constructs:

- ▶ **Data:** This is the user and application data that requires persistent storage. It can be text, binary formats, multimedia, or any other human- or machine-generated content.
- ▶ **Metadata:** This is the *data about the data*. It includes some predefined attributes such as upload time and size. Object storage allows users to include custom metadata containing any information in key and value pairs. This information typically contains information that is pertinent to the user or application that is storing the data and can be amended at any time. A unique aspect to metadata handling in object storage systems is that metadata is stored with the object.
- ▶ **A universally unique identifier (UUID):** This ID is assigned to every object in an OBS system. This UUID allows the object storage system to differentiate objects from one another and is used to find the data without needing to know the exact physical drive, array, or site where the data is.

Figure 1-1 on page 5 shows these three components of object storage.

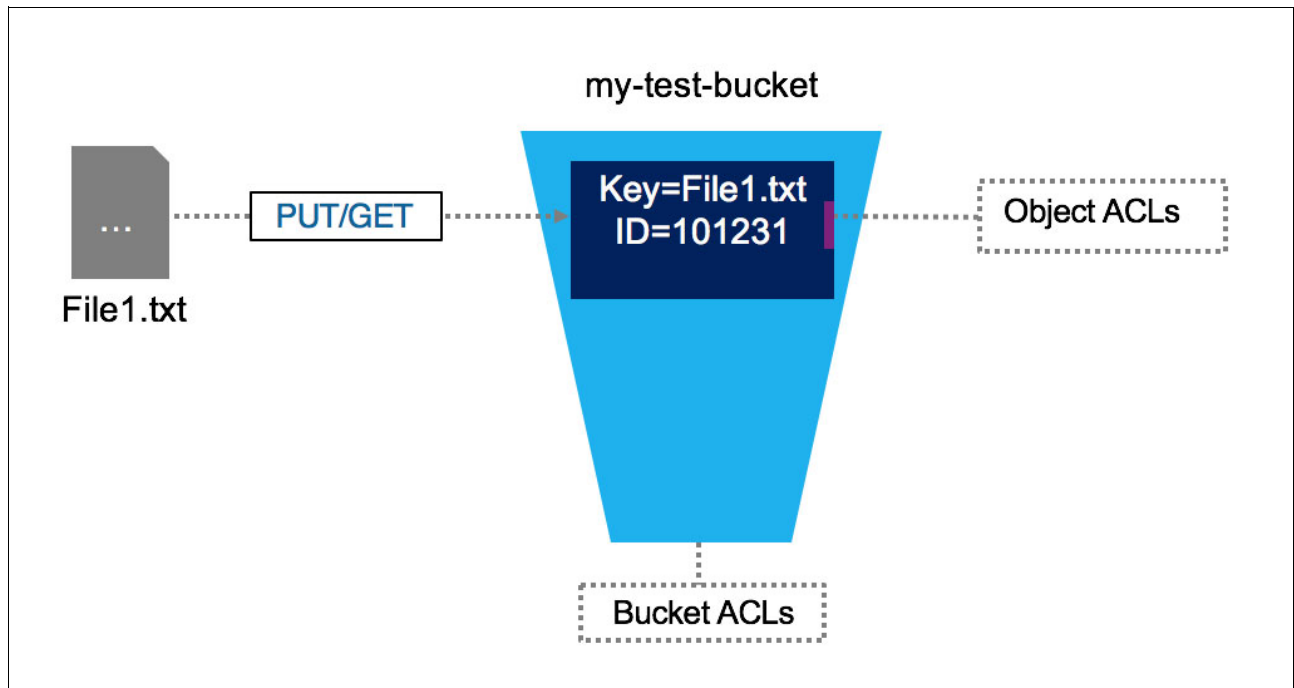


Figure 1-1 Components of object storage

This approach allows object storage to store data in a simple, flat hierarchy, which alleviates the need for large, performance-inhibiting metadata repositories that are a *choke point* for many large, file-based solutions.

Data access is achieved by using a REST interface over the HTTP protocol, which allows anywhere and anytime access simply by referencing the object ID. The file name of the original data is maintained as a key part of the object metadata, so it can also be used for object access in some object implementations.

Auto mechanic shop analogy: To better explain OBS, consider the scenario of going to a large auto mechanic shop. When you drop off your car, the service manager gives you a claim ticket that allows you to retrieve your car when the repairs are complete. While the service manager has your car, he or she might move it around as needed between different mechanics and repair bays and to optimize space in the parking lot. The claim ticket identifies your car, but not a mechanic, repair bay, or parking space. With OBS, an object ID identifies a specific piece of data, but it does *not* specify its location in the storage system. Data can be moved around as needed and the object ID is the *claim ticket that is* used to retrieve the data, wherever it is.

1.2 What storage business problems object storage can address

Here are some of the business problems that are addressed by object storage. For an in-depth description of these use cases and typical usage examples from several industries, see Chapter 4, “Typical applications for IBM Cloud Object Storage” on page 57.

- ▶ Backup repository (See “Use case 1: Backup repository” on page 75)
- ▶ Enterprise storage as a service (See “Enterprise storage as a service” on page 61)
- ▶ Active archive (See “Use case 3: Active archive” on page 78)
- ▶ Analytics and cognitive (See “Use case 5: Analytics and cognitive systems” on page 85)
- ▶ Content repository (“Use case 6: Content repository” on page 86)
- ▶ Enterprise collaboration (See “Use case 7: Enterprise collaboration” on page 88)

1.3 What qualifies business problems for object storage

Modern enterprises are dealing with customer expectations, increasing quantities and size of multimedia content, and statutory requirements for data retention. Data is often used in modern analytics to provide competitive advantage, and it is seen as resource to be used and preserved. As a result of these and many more factors, modern businesses are facing a number of challenges regarding the storage of exponential quantities of data, such as the following examples:

- ▶ Constant replatforming and data migrations that required to move to systems that can deal with growing capacity requirements due to the hard limitations with the current storage infrastructure.
- ▶ Managing rapid capacity growth with static or dropping operational budgets.
- ▶ Storing important, but infrequently accessed, data in a cost-effective and readily accessible manner.
- ▶ Data that is written today should be readable in 10 years time without using antiquated and proprietary technologies.
- ▶ Consumers, regulators, and business partners expect that data that is written will not become corrupt over time. Failure to access the data can result in a loss of customer base, loss of reputation, or fines.
- ▶ A connected world means that consumers and other businesses from around the world might need to access the data from outside the enterprise.

Exponential increase of data: With the rise of cloud, mobility, Internet of Things (IoT), social and analytics, the data explosion is accelerating. New, mostly unstructured data sources emerge constantly, creating an expanding data ecosystem for every organization.

Every day, we create 2.5 quintillion bytes of data, and 90% of the data in the world today was created in the last two years.^a More than 80% of this data is unstructured.^b An estimate states that the year 2020 will see 75 billion Internet-connected devices.^c

a. <http://www.sciencedaily.com/releases/2013/05/130522085217.htm>

b. <https://www.youtube.com/watch?v=DVSiyBXgfzU>

c. <http://www.businessinsider.com/75-billion-devices-will-be-connected-to-the-internet-by-2020-2013-10>

Object storage is inherently geared to better deal with these business challenges:

- ▶ Object architecture means that these systems can scale better than traditional arrays. The limitations are different between products, but most of them can deal with petabytes of data, and some of them can deal with exabytes and beyond of data as a single system with no need for data migrations.
- ▶ Capacity growth is more likely to be seamless because the abstraction of the object from the storage means that data can be easily moved around in the background in an autonomous manner.
- ▶ Data can be stored in a secure and resilient manner because the data protection schemes are geared towards an inherent, multi-site resiliency.
- ▶ Because data is accessible through standard APIs, concerns about future accessibility are alleviated when hardware and software levels change.
- ▶ Data durability is designed into these systems by using various data protection schemes, redundancy checks, error checking, and data scrubbing, ensuring that businesses and consumers can access their data at all points.
- ▶ The REST-based nature of an object means that data can be referenced and accessed from anywhere in the world without replicating data to each access point.

When not to use object storage: Object storage is not optimized for transaction-oriented primary storage for databases and similar workloads that require high I/O processing because of the impact in processing that is associated with slicing and dispersing.

1.4 Access methods for object storage

The underlying storage pool of a dispersed storage system can be shared and is jointly accessible by multiple access protocols. The simple object interface is accessed with a HTTP/REST API. Simple **PUT**, **GET**, **DELETE**, and **LIST** commands allow applications to access digital content, and the resulting object ID is stored directly within the application.

1.4.1 Access to storage through the REST API

REST is a style of software architecture for distributed hypermedia information retrieval systems, such as the World Wide Web. REST style architectures consist of clients and servers. Clients initiate requests to servers. Servers process requests and return associated responses. Requests and responses are built around the transfer of various representations of the resources.

Figure 1-2 shows the REST storage interfaces.

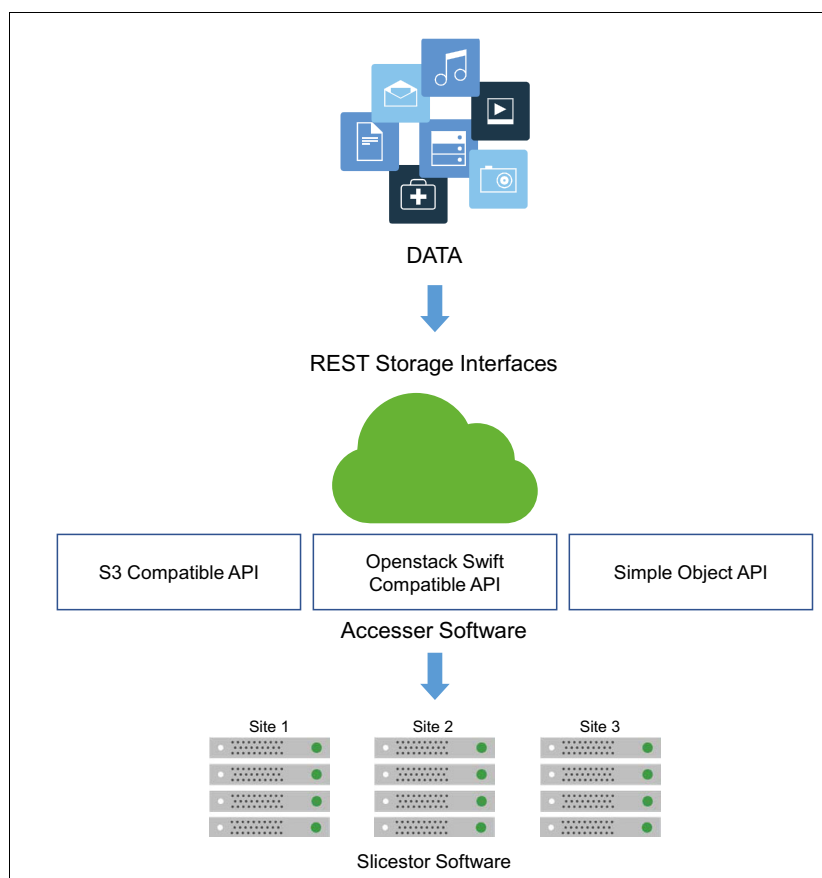


Figure 1-2 REST storage interfaces

The REST API works in way that is similar to retrieving a Universal Resource Locator (URL). But, instead of requesting a web page, the application is referencing an object.

REST API access to storage offers several advantages:

- ▶ Tolerates Internet latency
- ▶ Provides for programmable storage interface
- ▶ Provides efficient global access to large amounts of data

1.4.2 Access to object storage through other methods

You can use other popular access methods, such as gateways, to access an object store by using block- or file-based protocols. The gateways can do various optimizations and caching before data is sent to the back-end object store. You can learn more about gateways in 4.3.4, “Use case 4: Enterprise file services” on page 81.

1.5 Comparison and positioning to other storage models

This section compares object storage with other storage models.

1.5.1 Block storage

Block storage is the basis for a number of technologies and is prevalent in many enterprise workloads. It has various use cases ranging from transactional workloads and high-performance computing to being the base layer below other technologies, such as file storage. With block storage, data is split into evenly sized blocks of data, each with its own address that is written to an underlying medium, such as disk or flash. No additional information (metadata) to provide context for that block of data is recorded; only the location on the physical medium is recorded. The file system or the application accessing the block storage keeps track of the location where which information is written to the block storage.

Block storage is typically direct-attached to a server or workstation or assigned to one from a shared storage array. Block storage that is presented to hosts cannot be shared unless special clustering or locking software is used. Hosts view the disks as local devices and lock and write locally. SCSI reservation is done on the Storage Subsystem level. Data resilience can be achieved by using Redundant Array of Independent Disks (RAID) schemes and replication.

Connections to hosts in an enterprise setting are typically over a storage area network (SAN) that uses the Fibre Channel protocol, or through IP that uses the Internet Small Computer System Interface (iSCSI) or Fibre Channel over IP (FCiP) protocols. Disk assignment to a host requires planning and you must take care when managing the disk and /volume mappings because an incorrect configuration has a severe impact. The end-to-end connection to hosts is stateful and must be uninterrupted or the block device might experience an outage, stop working, or corrupt data.

In comparison, object storage is a higher-level abstraction layer that focuses on scalability, durability, ease of access, and multi-tenancy. Data is not addressed by blocks, but by object IDs rather than blocks and sectors. A file that is uploaded to an object store is stored as a bundle of user data, metadata, and is identified by a unique identifier.

The unique aspect of metadata in the object storage construct, is that it allows user-generated key and value pairs to be uploaded along with the data. This aspect makes object storage powerful and customizable. Examples of metadata can include any recordable details with examples of geographic metrics, application details, audit trails, or content descriptors being common.

The shareable nature of object storage means that it is often used in web-scale and development deployments, and as a result many social media or multimedia streaming services use object storage because of its scalability, resilience, and durability.

Object storage does not allow in-place updates (as block storage does). Objects must be manipulated as a whole unit, requiring the entire object to be loaded to a server or workstation, updated locally, and then rewritten in their entirety. This process can have performance implications for applications that expect to write data as they do with block storage. However, object storage does support partial reads to preserve bandwidth and accelerate access to data.

1.5.2 File storage

File storage is a method of storing computer data as files in a hierarchy of directories and subdirectories. Files can be read and written to in part or in whole. Files can be appended to or edited in place (updated).

File types are managed by the metadata in the file system. The files themselves contain the data without any metadata such as the file name, read/write attributes. For each file, metadata is kept in a special area in the file system. The format of the file is a proprietary or commonly agreed format of the data of the file and has no relation with the file system. Humans have agreed to a convention of naming files with an extension that provides the same hint as to what the file type is. For example, `letter.txt` is a text file and `app.conf` is a configuration file.

Security constructs consist of user and group permissions on files and directories. Shared file storage is typically referred to as network attached storage (NAS), which can mount folders on multiple machines.

File storage typically stores data and metadata separately, with large NAS solutions that use centralized high speed flash aimed to reduce metadata access latency as a performance inhibitor. A common issue for large NAS platforms is that as the number of references in the metadata repository grows, the time that is required to scan or look up metadata increases and performance degrades.

Connections to systems mounting NAS shares are stateful and must send regular packets to maintain access. If for some reason these packets cannot reach their destination, the mounts become stale and data no longer is accessible.

By contrast, object storage does not have a concept of hierarchy and all objects are stored in a flat structure. Objects may have delimiters such as `'/'` in the object name that may be represented as directories on object storage clients. Objects can be read and written to as a whole, can be partially read, but cannot be edited in place. Object storage security is maintained by using credentials that grant access to specific buckets or objects.

Metadata is stored with the object, so there is no need for a central metadata repository alleviating a major pain point in managing NAS systems at scale. In addition, custom metadata can be dynamically added to an object. Unlike NAS, connections to object storage systems are stateless, and are established only when data is being pushed or pulled from the object store.

Table 1-1 summarizes the differences between file storage and object storage.

Table 1-1 Comparison of file storage and object storage

File storage	Object storage
Can store millions of files	Can store hundreds of billions of objects
Has a file system hierarchy	Has one storage pool and object IDs
Complex to scale, and can handle up to millions of files	Scales easily and uniformly, and can handle big data up to petabytes and beyond
TCO increases exponentially	Lowest TCO, which decreases exponentially

1.6 Object storage key concepts

This section introduces object storage key concepts.

1.6.1 Buckets

A bucket is a logical abstraction that is used to provide a data container in object storage. It is not bound to physical arrays or disks and can span multiple sites across multiple regions depending on the object store architecture.

Because of this abstraction from physical storage, buckets do not require any pre-allocation of storage capacity. Segregation of data is achieved through bucket ownership and a combination of public and secret keys that are bound to object store accounts. This security model ensures that data is visible only by users and applications that are authorized for access.

1.6.2 Objects

An object consists of user data that is uploaded to an object store. Typically, it is a file and object metadata that is stored together with the object. Objects are referenced by unique IDs that are known to the entire object storage system.

1.6.3 Metadata

Object metadata contains information about object size, creation time, last modified time, and a hash that is used to confirm the object integrity. In addition, custom metadata in key value pairs can be injected into the object for use by users and applications.

1.6.4 Access control lists

Access control lists (ACLs) are the primary security constructs in object storage. Object ACLs can be set at the object or bucket level and owners can grant permissions based on an account universally unique identifier (UUID), email address, or groups.

Permission granularity extends to read and write access, the ability to read and write ACLs on the object or bucket, or full control, which gives all the privileges just mentioned.

1.6.5 Object data protection

Data protection schemes in object storage typically consist of either replication, erasure coding (depending on the solution, this may be used exclusively), or a combination of the two approaches may be used.

Replication

Replication between two or more sites is one of the methods that is used to ensure data resilience. It includes making multiple copies of the data. In case of disk, system, or site failure, the data may be copied back to the original site or the secondary copy may be promoted to read and write access.

Erasure coding

Erasure coding is a modern data protecting scheme that uses forward error correction codes to pad data so that other data can be rebuilt without needing replication. There are many ways to implement this technology with some vendors choosing in-array erasure coding, site-wide erasure coding, or geographically dispersed erasure coding.



Introducing IBM Cloud Object Store Public Services

This chapter introduces IBM Cloud Object Store (IBM COS) Public Services. It describes the need for innovative storage solutions, and describes IBM COS deployment options, and the features and functions in IBM Cloud Object Store Public Services.

The following topics are covered in this chapter:

- ▶ 2.1, “Overview” on page 14
- ▶ 2.2, “Architecture” on page 15
- ▶ 2.3, “IBM Cloud Object Storage offerings” on page 16
- ▶ 2.4, “IBM COS Public Services capabilities” on page 19
- ▶ 2.5, “Programming model and interfaces” on page 24

2.1 Overview

The digital enterprise has resulted in an explosion of data, both structured and unstructured, and data volumes are expected to grow to the zettabyte scale in the next few years. This situation presents a unique opportunity to harvest information and analytics from data to help improve operational efficiencies and serve customers better.

Data integrity, scalability, manageability, availability, and security are key to data, and the traditional storage solutions simply cannot handle the scale at which the data is growing. A dispersed storage mechanism can be leveraged to handle both structured and unstructured data at scale.

IBM Cloud Object Store (IBM COS) is a cost-effective object store. IBM COS is a dispersed storage mechanism that leverages a cluster of storage nodes to store pieces of the data across the available nodes. IBM COS uses an *Information Dispersal Algorithm (IDA)* to divide files into unrecognizable slices that are then distributed to the storage nodes. No single node has all the data, which makes it safe and less susceptible to data breaches while needing only a subset of the storage nodes to be available to fully retrieve the stored data. This ability to reassemble all the data from a subset of the chunks dramatically increases the tolerance to node and disk failures.

Figure 2-1 shows the characteristics of IBM COS.

Scalability	Reach new cost, capacity and accessibility milestones with petabyte scalability and beyond.
Security	Protect mission-critical data with zero-touch encryption and built-in carrier-grade security.
Availability	Make sure your data is always available—regardless of planned or unplanned downtime.
Efficiency	Simplify management with an intuitive platform that is 15 times more efficient than legacy storage operations.
Economics	Dramatically reduce long-term total cost of ownership with a premium software-based solution that runs on commodity hardware.

Figure 2-1 Characteristics of IBM Cloud Object Storage

IBM COS is available in the following deployment models:

- ▶ Private on-premises
- ▶ Dedicated (single-tenant)
- ▶ Public (multi-tenant)
- ▶ Hybrid (a mix of on-premises, dedicated, or public offerings)

2.2 Architecture

The IBM COS architecture is composed of the following three functional components. Each component runs ClevOS software that can be deployed on compatible, industry-standard hardware:

- ▶ IBM Cloud Object Storage Manager

This component provides an out-of-band management interface that is used for administrative tasks such as system configuration, storage provisioning, and monitoring the health and performance of the system.

- ▶ IBM Cloud Object Storage Accesser®

This component imports and reads data, encrypting/encoding data on import and decrypting/decoding data on read. It is a stateless component that presents the storage interfaces to the client applications and transforms data by using an IDA.

- ▶ IBM Cloud Object Storage Slicestor®

This node is primarily responsible for storage of the data slices. It receives data from the Accesser on import and returns data to the Accesser as required by reads.

Figure 2-2 shows the IBM COS multi-site architecture.

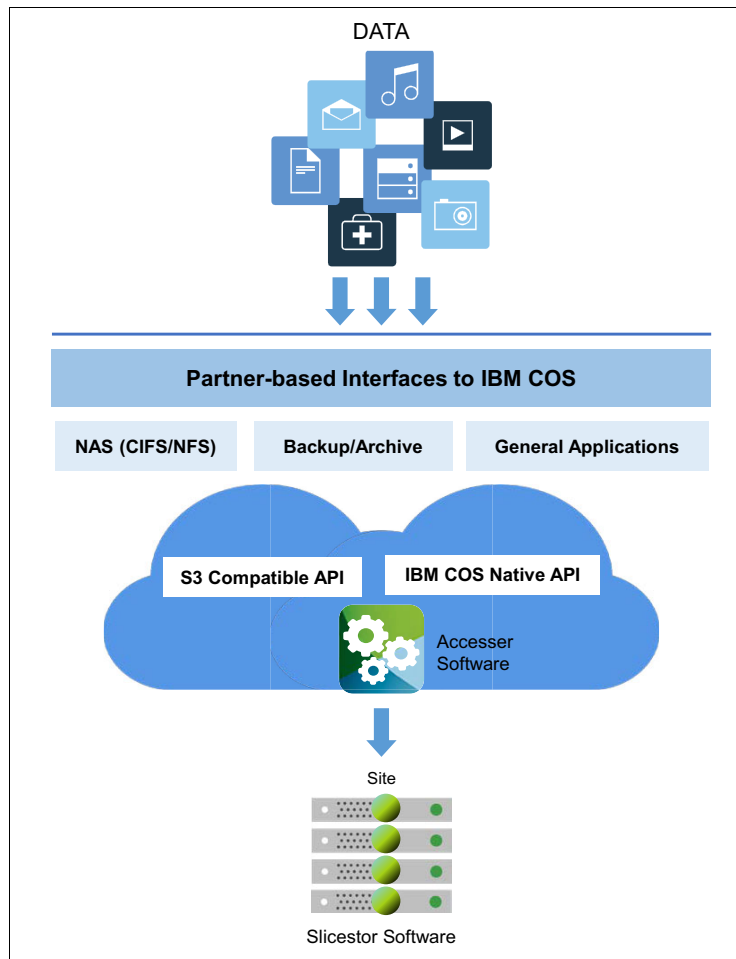


Figure 2-2 IBM Cloud Object Store multi-site architecture

IBM COS API: At the time of writing this book, IBM COS API is not available and expected to be available in 2017. More details are in 2.5.2, “IBM COS API (future release)” on page 24. The final name of the API, and the API features described in this book are subject to change at the time of general availability of the API.

With the IBM COS Public Services, this architecture and the actual implementation of IBM COS is transparent to the users of the service, so we are not going to go into details of these in this book. You can refer to *IBM Cloud Object Storage Concepts and Architecture*. REDP-5435 for a detailed discussion of IBM COS architecture, technology and associated benefits.

2.3 IBM Cloud Object Storage offerings

As the time of writing, IBM COS is available in the configurations that are described in this section. Figure 2-3 shows a summary of these offerings.

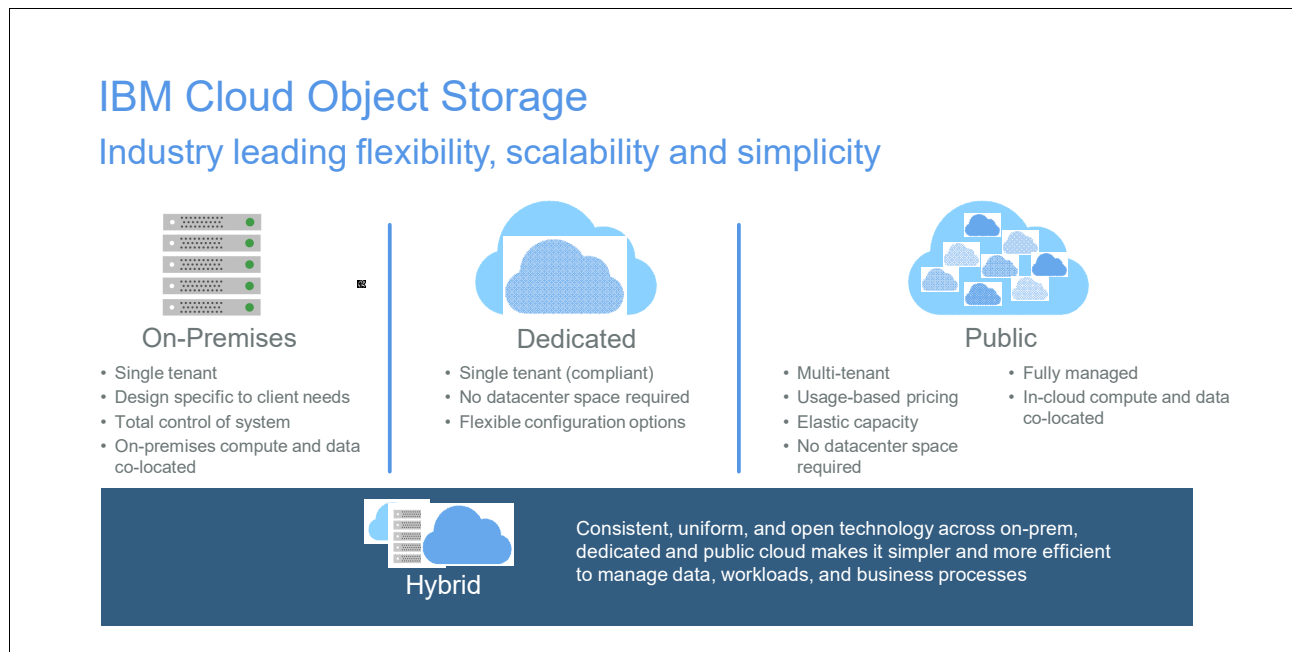


Figure 2-3 IBM Cloud Object Storage offerings

2.3.1 On-premises: Software only

IBM COS is available as a software-only offering. The customer must purchase hardware for the IBM COS Manager, IBM COS Accesser, and IBM COS Slicestor. In addition to IBM appliances, a list of hardware configurations can be obtained from vendors, including HP, Lenovo, Cisco, SuperMicro, Ericsson, Seagate, and others. The COS Accessor software may also be configured as a VM or as a VM on the Slicestor servers.

This offering provides always-on availability and continues to serve storage applications while completing tasks that require scheduled downtime in most traditional storage systems.

2.3.2 On-premises: Software and appliance

IBM COS is available as a preconfigured set of appliances. The customer can directly start using the appliance.

This offering has a shared-nothing architecture for maximum scalability. The software supports a strong data consistency model that allows for data to be stored across multiple time zones. The namespace is virtually unlimited with no centralized metadata management process and no limits on metadata size or the number of attributes.

Note: These appliances are also available as preconfigured bundles for easier and faster deployments.

2.3.3 Dedicated cloud-based

If the need exists for specific performance levels for a cloud-based solution, or a compliance or data isolation requirement exists for a single tenant solution, an IBM COS Dedicated Services offering is available. This provides a single tenant offering in the many IBM Cloud facilities worldwide. Two types of management options are available for the IBM COS Dedicated services:

- ▶ IBM managed
- ▶ Self-managed

The *IBM managed option* includes a dedicated team of IBM specialists to operate your system, providing you a user experience similar to a public cloud. IBM specialists handle day-to-day operations and change management such as provisioning, access control credentials, and health management. The IBM team can be accessed through phone, email, or an online ticketing system.

This service can be deployed across multiple regions based on the specific data access and resiliency requirements of your workload.

What this means for you: Using the IBM managed option frees up your in-house IT personnel to focus on driving innovation in other areas.

With the *self-managed option*, your IT team takes charge of day-to-day management of the IBM COS system with a simpler, easier-to-use web-based portal. This object storage system is deployed on a dedicated, isolated infrastructure in the IBM Cloud. Your team will have full operational control of the system through a simple management portal, including the ability to create reports for current and historical usage. This solution can be deployed across multiple regions based on the specific data access and resiliency requirements of your workload.

What this means for you: With the self-managed option, you have full control over your object storage system using a simple management portal.

So, you might ask: *Which option is best for me?* You can use Table 2-1 to choose the best IBM COS Dedicated Services option to help optimize storage costs and manage your unique workload performance and management control requirements.

Table 2-1 Management features aligned with IBM COS Dedicated Services management options

Management feature	IBM managed	Self-managed
Dedicated onboarding with a team of IBM specialists to help with initial design, deployment and setup	Yes	Yes
Requires power, cooling and rack space infrastructure in your IT facilities ^a	No	N
Isolated storage infrastructure that gives you complete data access control	Yes	Yes
Requires your IT team for day-to-day operations and management	No ^b	Yes
Storage Consumption Monitoring Portal integrated into the IBM Cloud Object Storage System™	Yes	Yes

a. Power, cooling and rack space infrastructure are provided by IBM Cloud facilities.

b. Day-to-day operations and management are provided by IBM.

2.3.4 IBM COS Public Services

Accessible through a self-service portal, IBM COS Public Services offering runs on a shared multitenant cloud infrastructure spanning a global network of IBM Cloud facilities that are managed by IBM. This offering is the main focus of this book and is described in more detail in 2.4, “IBM COS Public Services capabilities” on page 19.

2.3.5 Comparison of IBM Cloud Object Storage deployment options

Depending on the business need, the data sensitivity and application characteristics of any of the available IBM COS deployment options can be used. At a high-level, the benefits of each of the options can be summarized by control and total cost of ownership (TCO) characteristics.

Figure 2-4 on page 19 compares the IBM COS deployment options.

IBM Cloud Object Storage deployment options

Trade-off: Control v. Lower TCO

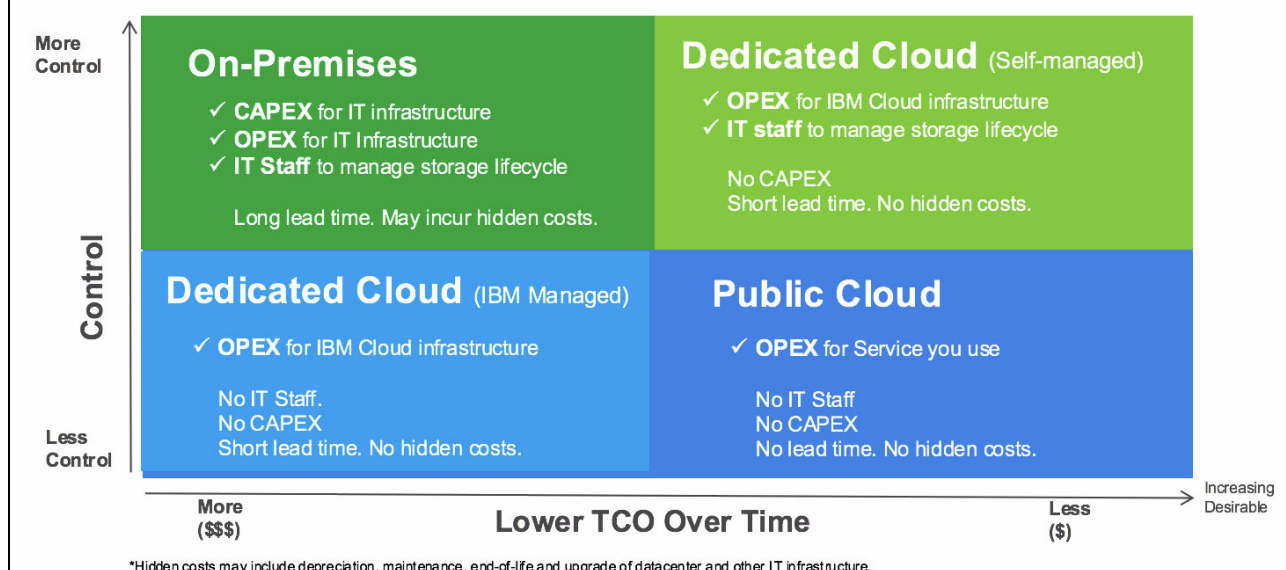


Figure 2-4 IBM Cloud Object Storage deployment options comparison

What about hybrid services? Choosing a hybrid configuration enables you to mix on-premises, dedicated or public offerings to enable a flexible configuration if you need additional data center space and site fault tolerance. The hybrid configuration enables you to take advantage of multiple IBM data center locations to easily expand your footprint and have a fully managed solution from IBM.

If you want to use cloud services, open communities, and emerging technologies to achieve digital transformation (while still retaining control and sovereignty over your data) consider the IBM Cloud Object Storage Hybrid Services options.

2.4 IBM COS Public Services capabilities

With IBM COS Public Services offering, you can consume storage in various locations around the world and with flexible resiliency models. It can accommodate a broad range of workloads, including cloud-native applications and on-premises data, and enables you to pay for storage as you consume it. For those clients that use IBM Cloud facilities for compute, analytics, or other services, IBM COS Public Services offer direct connectivity to object storage. This offering is fully managed by IBM personnel.

Capabilities of this offering are described next.

For more information, see the [IBM Cloud Object Storage](#) web page.

Resiliency options

At time of writing this book, the only resiliency option for the IBM COS Public Services is the *cross-region* option. The *regional* option will be available at a later time, so both options are covered in this section.

Cross-region option

The cross-region option stores the data across three or more geographic regions for business continuity and workload data accessibility from multiple regions, *which provides additional protection from region-wide outages*. The data is available through multiple endpoints across the different regions.

With the cross-region option, IBM COS components are deployed across multiple data centers in multiple geographic locations (for example, cloud facilities in Dallas, San Jose, and Washington DC), as shown in Figure 2-5.

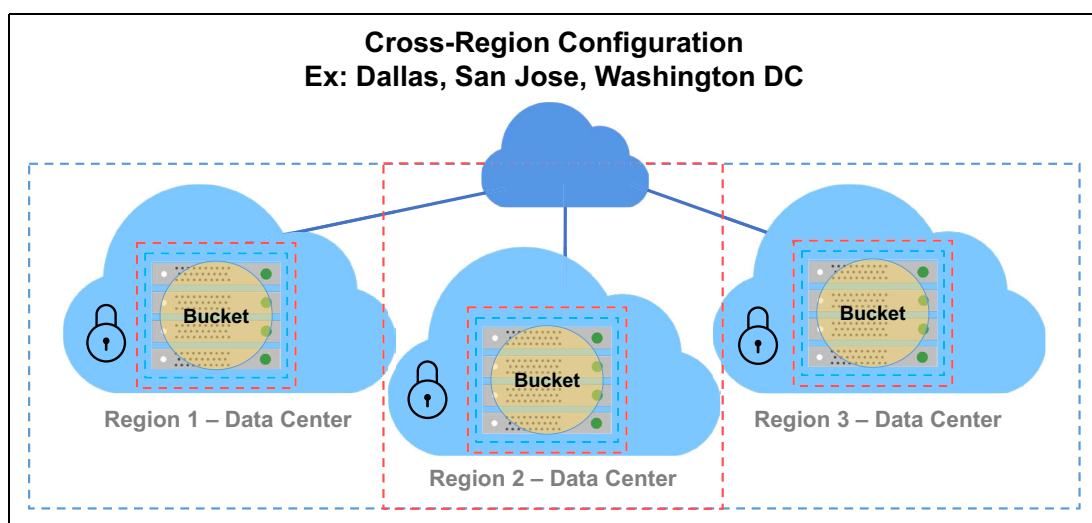


Figure 2-5 IBM Cloud Object Storage cross-region option

Regional option

In the *regional option* the data is stored in multiple IBM Cloud facilities within a single geographic region, which *helps protect data against the most common component failures and power outages*.

With this option, IBM COS components are deployed across multiple locations in a single geographic location (for example, Dallas data center), as shown in Figure 2-6.

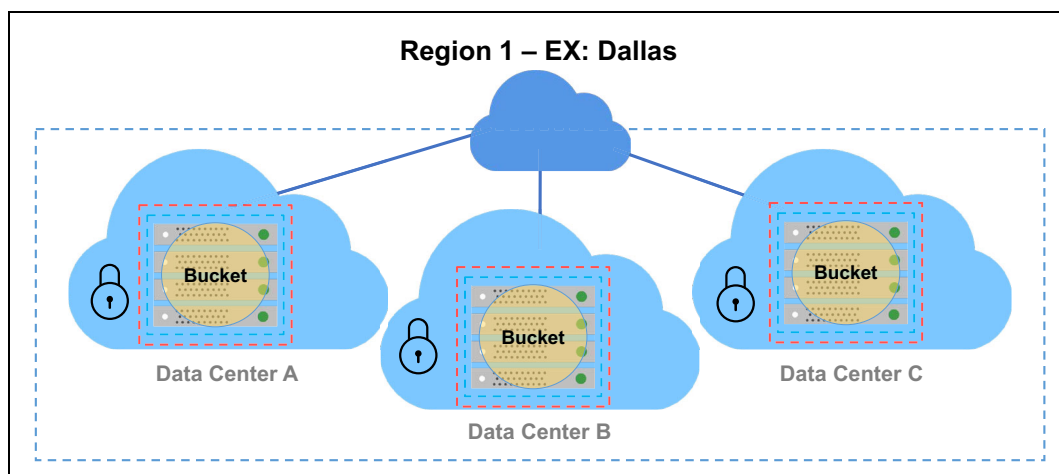


Figure 2-6 IBM Cloud Object Storage regional option

Class options

IBM COS Public Services is available in two classes:

- Standard: This class is ideal for unstructured data that requires frequent data access.
- Vault: This class is better suited for workloads with data that is accessed less often.

How to choose the right service for you

Table 2-2 can help you choose the right service for your workload data access and resiliency requirements. It can help you assess the various options for IBM COS Public Services for your workload data access and resiliency requirements. The sections following the table describe the options.

Table 2-2 Options for IBM COS Public Services

	Standard regional	Standard cross-region	Vault regional	Vault cross-region
Ideal for storing and managing active data such as collaboration and born-on-the-cloud application data	X	X		
Ideal for infrequently accessed data such as compliance and backup application data			X	X
Deployed across multiple IBM Cloud facilities in a single geographic region	X			
Deployed across IBM Cloud facilities located in three or more geographic regions			X	
Protects data from most common component failures and power outages	X	X	X	X
Provides extra protection from region-wide outages, such as natural disasters, to help ensure high availability of mission critical data		X		X

These options are described next. As mentioned in “Resiliency options” on page 19, at time of writing this book, the only resiliency option for the public services is the cross-region option (both standard and vault), but standard regional and vault regional options are expected to be available in the future (currently planned for first half of 2017).

Standard regional option

This option is designed for storing and managing frequently accessed data, providing high performance at low costs for use cases such as DevOps, application testing, big data, analytics, active content repositories, and sync and share applications.

Pricing for this service is primarily based on storage capacity, bandwidth consumed and operational requests made on the data. *You pay for only what you use every month, with no separate charges for deploying your data in multiple IBM Cloud facilities within a single geographic region.*

Standard cross-region option

This option is designed for storing and managing frequently accessed unstructured data that requires the highest level of availability for business continuity. Your data is available through multiple endpoints across three or more geographic regions to help ensure business continuity during region-wide outages due to natural disasters. This option is designed for cloud-hosted mission-critical workloads such as application hosting, worldwide collaboration, active content repositories, big data, and analytics.

Pricing for this service is primarily based on storage capacity, bandwidth consumed and operational requests made on the data. *You pay for only what you use each month with no separate charges for deploying your data in multiple regional.*

Vault regional option

This option is designed for storing and managing your less frequently accessed data such as compliance data, backup and archive data, or content repositories.

Pricing for this service is primarily based on storage capacity, bandwidth consumed and operational requests made on the data. An additional retrieval charge applies each time data is read for this service. There is a minimum threshold for object size and storage period, consistent with the intended use of this service for colder, less-active data. *You pay for only what you use each month. There are no separate charges for deploying your data in multiple IBM Cloud facilities within a single geographic region.*

Vault cross-region option

This option is designed for data that is accessed less frequently but still requires a high level of data availability to help ensure business continuity during region-wide outages due to natural disasters. Your data is available via multiple end-points across three or more geographic regions.

Common use cases for this service offering include mission critical compliance backup, data archiving, and infrequently accessed content repositories.

Pricing for this service is primarily based on storage capacity, bandwidth consumed, and operational requests made on the data. An additional retrieval charge is applied each time data is read. The service includes a minimum threshold for object size and storage period consistent with the intended use of this service for colder, less-active data. *You pay for only what you use each month, with no separate charges for deploying your data in multiple regional IBM Cloud facilities.*

Summary of options

What this means for you: Continuous availability is inherent in the IBM COS architecture. Regardless of the service option chosen, IBM COS encrypts and slices up data as soon as it comes in, with the slices dispersed across multiple locations automatically. If a write operation is confirmed, data is protected *immediately*. This means that if a location goes down, data can still be delivered from the slices that exist in remaining locations. Applications that rely on that data remain up and running.

Security

As discussed in 2.1, “Overview” on page 14, IBM COS uses an innovative approach for cost-effectively storing large volumes of unstructured data while ensuring security, availability, and reliability. This is accomplished by using Information Dispersal Algorithms (IDAs) to separate data into unrecognizable slices that are distributed across a network of data centers, making transmission and storage of data inherently private and secure. This technology is also used in IBM COS Public Services, because the underlying architecture is the same.

Objects in IBM COS Public Services are encrypted at rest. This technology individually encrypts each object using per-object generated keys. These keys are secured and reliably stored using the same Information Dispersal Algorithms that protect object data using an all-or-nothing transform (AONT), which prevents key data from being disclosed if individual nodes or hard drives are compromised.

Storage can be accessed over HTTPS, and internally storage devices are certified and communicate with each other using Transport Layer Security (TLS).

Note: In IBM COS Public Services, data at rest is encrypted with automatic provider-side Advanced Encryption Standard (AES) 256-bit encryption and an SHA-256 hash. Data in motion is secured with built-in carrier grade Transport Layer Security/Secure Sockets Layer (TLS/SSL) or SNMPv3 with AES encryption. For data access, credentials are generated and managed for individual users, and digital certificates are issued to each node, which helps protect data against unauthorized access and network connections against rogue node attacks. Each object is further sliced with no copy of the data in a single disk, node, or location, so that decoding the data by using algorithmic computation is virtually impossible.

Data deletion

After data is deleted, various mechanisms exist that prevent recovery or reconstruction of the deleted objects. The deletion of an object undergoes various stages, from marking the metadata, indicating the object as deleted, to removing the content regions, to the finalization of the erasure on the drives themselves until the eventual overwriting of the blocks representing that slice data. Depending on whether one compromised the data center or has possession of the physical disks, the time an object becomes unrecoverable depends on the phase of the delete operation. When the metadata object is updated, clients that are external from the data center network can no longer read the object. When a majority of slices representing the content regions have been finalized by the storage devices, accessing the object is not possible.

Tenant isolation

IBM COS Public Services uses a shared infrastructure, multitenant object storage solution. If your workload requires dedicated or isolated storage, you can use one of the on-premises or dedicated options that are described in 2.3, “IBM Cloud Object Storage offerings” on page 16.

Summary of security

What this means for you: IBM COS Public Services is designed with a high level of security in mind to protect your data from security breaches. If security is compromised in a region, the full content will not be exposed. If one region is offline, your applications continue to run without disruption. This is called *always-on availability*, which is major benefit for IBM COS customers.

Scalability

IBM COS software has been tested at web-scale with production deployments that exceed hundreds of petabytes of capacity, and can scale to exabytes. The IBM COS architecture allows for increasing storage capacity and performance by adding Slicestor nodes and also puts no limit on the number of Accesser nodes. With the IBM COS Public Services, you get the same benefits in terms of scalability.

What this means for you: With the IBM COS Public Service you start at an initial capacity with and add or remove capacity as needed, paying only for what you use. The upward scalability is virtually unlimited with this offering.

2.5 Programming model and interfaces

This section describes the IBM COS Public Services programming model and interfaces.

2.5.1 Simple Storage Service (S3) API

Simple Storage Service (S3) is a public cloud platform for storing binary or unstructured data. The S3 application programming interface (API) is based on Representation State Transfer (REST) and delivered over HTTP. Developers use the S3 API to get a large set of tools and SDKs. The IBM COS implementation of the S3 API supports basic I/O and Object ACLs at a storage account/service instance granularity.

Connecting to the IBM COS service with the S3 API requires credentials and an endpoint. Credentials consist of an Access Key and a Secret Key. The Access Key is equivalent to a temporary account ID and the Secret Key is the equivalent of a password.

2.5.2 IBM COS API (future release)

The IBM COS API will be geared toward consistency and integration with the other IBM Cloud features. This API will be consistent with IBM Cloud API guidelines, including cosmetic aspects, such as JSON encoding, and functional items, such as Identity and Access Management (IAM) compliance and OAuth2 authentication for IBM ID.

Objects that are created by using the S3 API can be accessed by using the IBM COS API and vice versa.

Note: The final name of the IBM COS API, and the API features described in this book are subject to change at the time of general availability of the API.



Getting started with IBM Cloud Object Storage

This chapter describes how to get started with IBM Cloud Object Storage Public Services on the IBM public cloud offering, IBM Bluemix Infrastructure.

This chapter introduces the IBM public cloud platform and how to create an account to start consuming IBM Cloud Object Storage. It also provides information about available user interfaces (UIs), software development kits (SDKs) and usage of the application program interfaces (APIs) available for IBM COS Public Services.

The following topics are covered in this chapter:

- ▶ 3.1, “Setting up your cloud account” on page 26
- ▶ 3.2, “Credentials and endpoints” on page 28
- ▶ 3.3, “Managing buckets and objects” on page 30
- ▶ 3.4, “IBM COS S3 API and common functions” on page 32
- ▶ 3.5, “Clients, CLIs, and SDKs” on page 40
- ▶ 3.6, “Preferred practices for application design” on page 53

3.1 Setting up your cloud account

To use IBM COS Public Services a Bluemix Infrastructure (SoftLayer) account is required in order to access the Bluemix Infrastructure portal.

1. You can sign up for an account from the [IBM Cloud Bluemix Object Storage](#) page.
2. After an account is created, log in to the Bluemix Infrastructure portal at [SoftLayer](#).

Note: Bluemix Infrastructure is the new brand name for SoftLayer. Note that you will still see the SoftLayer name in the configuration panels.

3. After logging in to the portal, select **Storage** → **Object Storage** (Figure 3-1).

The screenshot shows the SoftLayer Bluemix Infrastructure portal. The top navigation bar includes links for Devices, Storage, Network, Security, Services, Support, and Account. A search bar is located on the right. The left sidebar contains a notification about unread notifications and an 'Account Summary' section showing a current balance of \$0.00. Below this is an 'Order' section with icons for Devices, Storage, Network, Security, and Add-On Services. The 'Storage' icon is highlighted. The main content area shows a dropdown menu for 'Storage' with options: Block Storage, File Storage, Object Storage (selected), CDN, LockBox, and Backup. The 'Object Storage' option is highlighted in blue. The right sidebar contains a 'Scheduled Maintenance & Updates' section with two calendar views for February 2017 and March 2017. Below this is a 'Bandwidth' section with a message 'No bandwidth data to display'. At the bottom is an 'Account Snapshot' section with links for Users, Devices, Open Tickets, Active Domains, DNS Forward Zones, Private Images, Subnets, and VLANs.

FEBRUARY 2017							MARCH 2017						
S	M	T	W	T	F	S	S	M	T	W	T	F	
29	30	31	1	2	3	4	26	27	28	1	2	3	
5	6	7	8	9	10	11	5	6	7	8	9	10	
12	13	14	15	16	17	18	12	13	14	15	16	17	
19	20	21	22	23	24	25	19	20	21	22	23	24	
26	27	28	1	2	3	4	26	27	28	29	30	31	
5	6	7	8	9	10	11	2	3	4	5	6	7	

Figure 3-1 Bluemix Infrastructure portal Object Storage selection

4. At the object storage page click on **Order Object Storage**.

5. A pop-up panel opens (Figure 3-2). Use it to select either **SWIFT** or **S3 API** based object storage. From the Select Storage Type drop-down, select **S3 API** storage. Click **Continue**.

Order Object Storage

Select Storage Type

Cloud Object Storage - S3 API

* Required field

Storage pricing is based on a rate for storage capacity used, operational requests, and a usage rate for public outbound bandwidth. Inbound and private network bandwidths are included at no additional cost.

Standard Cross Region	\$0.0220 / GB
Vault Cross Region	\$0.0150 / GB
Cold Vault Cross Region Retrieval	\$0.0360 / GB
Cold Vault Cross Region Bandwidth	\$0.0650 / GB
Standard Cross Region Bandwidth	\$0.0650 / GB
Vault Cross Region Retrieval	\$0.0070 / GB
Vault Cross Region Bandwidth	\$0.0650 / GB
Cold Vault Cross Region	\$0.0080 / GB
Standard Cross Region API Request (Class A)	\$0.0044 / 1000 Requests
Vault Cross Region API Request (Class A)	\$0.0090 / 1000 Requests
Standard Cross Region API Request (Class B)	\$0.0036 / 10000 Requests
Vault Cross Region API Request (Class B)	\$0.0090 / 10000 Requests
Cold Vault Cross Region API Request (Class B)	\$0.0180 / 10000 Requests
Cold Vault Cross Region API Request (Class A)	\$0.0180 / 1000 Requests

Cancel

Continue

Figure 3-2 Selecting S3 API based Object Storage

6. The order placement panel opens (Figure 3-3). Review the Master Service Agreement and enter any promotional code, if you have one. Pricing here will show as zero (\$0.00) because pricing is based on a consumption model.

Order Object Storage

Confirm Order

Promo Code

Recalculate

Charges for this order

Charged To

One Time

\$0.00

Initial Payment is \$0.00, no payment required at this time.

Monthly:

\$0.00

Tax

\$0.00

Total

\$0.00

Adjustments

Prorated Total

\$0.00

Initial Charge

\$0.00

Order Review

Item

Cloud Object Storage - S3 API

Quantity

1

☒ I have read the [Master Service Agreement](#) and agree to the terms therein.

Previous

Place Order

Cancel

Figure 3-3 Placing an order of Cloud Object Storage

7. After placing your order, a new COS instance is listed in the Object Storage panel, (Figure 3-4).

SOFTLAYER
an IBM Company

DevicesStorageNetworkSecurityServicesSupportAccount

Search

Object Storage

Order Object StorageView Documentation

You have unread notifications for events which may affect your services

Unplanned Events (1)

Viewing 1 to 3 of 3

Displaying 25 per page

<<<Page 1 of 1>>>

Account Name	API Type	Description (click to edit)
IBMOSC1205109-1	S3	
IBMOSC1205109-2	S3	

Figure 3-4 Object Storage ordered

3.2 Credentials and endpoints

Each COS instance initially has a single set of credentials which consist of an Access Key ID and a Secret Access Key to provide access to the buckets within. At present two sets of credentials can be created per COS instance, this allows credentials to be rotated in applications without interruption.

Current credentials are used at a COS instance level and do not have the ability to grant or restrict access to a specific user.

Bluemix Infrastructure provides three different types of endpoints for connecting clients and applications that use the S3 API to the public cloud instance.

3.2.1 Public endpoints

These endpoints are accessible through the public Internet and should be used for any traffic not originating from a Bluemix Infrastructure data center. Inbound bandwidth is unmetered but outbound bandwidth is charged per gigabyte (GB).

At the time of writing this book, four public endpoints exist:

- ▶ s3-api.us-geo.objectstorage.softlayer.net (US Region)
- ▶ s3-api.dal-us-geo.objectstorage.softlayer.net (Dallas)
- ▶ s3-api.sjc-us-geo.objectstorage.softlayer.net (San Jose)
- ▶ s3-api.wdc-us-geo.objectstorage.softlayer.net (Washington DC)

Note: The number of endpoints is expected to increase in the future as IBM Cloud Object Storage Public Services is deployed at new data centers.

3.2.2 Private endpoints

These endpoints are available only through Bluemix Infrastructure's private network and so are accessible through the private interface on virtual server instances (VSIs), bare metal servers, or a user on the account connected to a Bluemix Infrastructure VPN endpoint. Private endpoints do not incur any charges for outbound or inbound traffic even if the traffic is cross regions or across data centers.

Customers with workloads running in a Bluemix Infrastructure data center in Dallas, San Jose, or Washington DC should use those specific endpoints. Customers with existing workloads running in a remote Bluemix Infrastructure data center should use the US Region endpoint.

Presently four private endpoints exist:

- ▶ s3-api.us-geo.objectstorage.service.networklayer.com (US Region)
- ▶ s3-api.dal-us-geo.objectstorage.service.networklayer.com (Dallas)
- ▶ s3-api.sjc-us-geo.objectstorage.service.networklayer.com (San Jose)
- ▶ s3-api.sjc-us-geo.objectstorage.service.networklayer.com (Washington DC)

3.2.3 Custom Private Addressing (CPA) endpoint

CPA allows customers to provision VSIs, bare metal servers and various services into IP address ranges of their choice. Various restrictions of CPA must be considered: CPA is for new Bluemix Infrastructure accounts only and is available in selected data centers at present. For further information about CPA, talk to a Bluemix Infrastructure Sales representative.

Important: Applications and services running in the Bluemix catalog, such as Watson, containers, and runtimes connect through the public Internet to Bluemix Infrastructure IBM COS public endpoints.

3.2.4 Managing credentials

To view the credentials for the object storage instance, follow these steps:

1. Select the account name, and then click **Show Credentials** at the left side of the panel, and click the **Show** link next to **Credentials**. The Access Key ID and Secret Access Key are displayed (Figure 3-5).

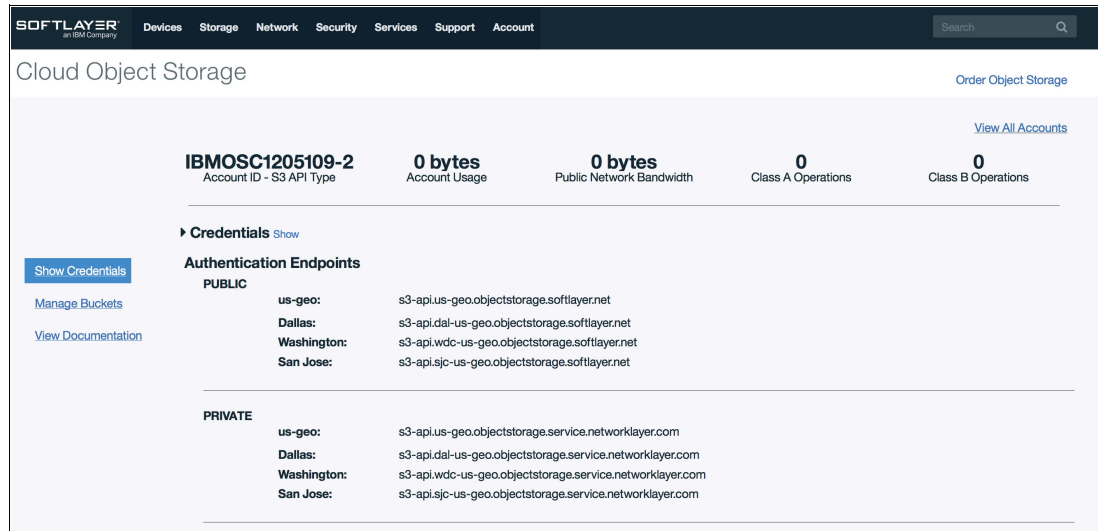


Figure 3-5 Bluemix Infrastructure Portal COS Endpoints and Credentials panel

2. Currently two sets of credentials can be active per IBM COS account. To create the second set of credentials expand and show the current account credentials, and then click the blue **Add** button.

3.3 Managing buckets and objects

Buckets can be created through the Bluemix Infrastructure portal or through various third-party GUI or CLI clients. Primarily the Bluemix Infrastructure portal for object storage provides basic administration functionality for bucket and object management. Most interaction with the object store is focused on API methods through client applications. For information about available APIs for IBM COS see 3.4, “IBM COS S3 API and common functions” on page 32. Further information about clients is in 3.4.2, “Supported API operations” on page 34.

3.3.1 Creating and managing buckets from Bluemix Infrastructure

To create and manage buckets from Bluemix Infrastructure, follow these steps:

1. From the Bluemix Infrastructure object storage panel, click the account you want to create or manage a bucket within. The administration panel for this IBM COS account opens.
The main administration panel shows the name of the account you are viewing, storage currently being consumed, and public bandwidth used so far.
2. At the right, click the blue **Add** symbol to create a new bucket. Because the namespace for buckets is shared globally in public IBM COS, bucket names must comply with DNS and be unique. See Figure 3-6 on page 31.

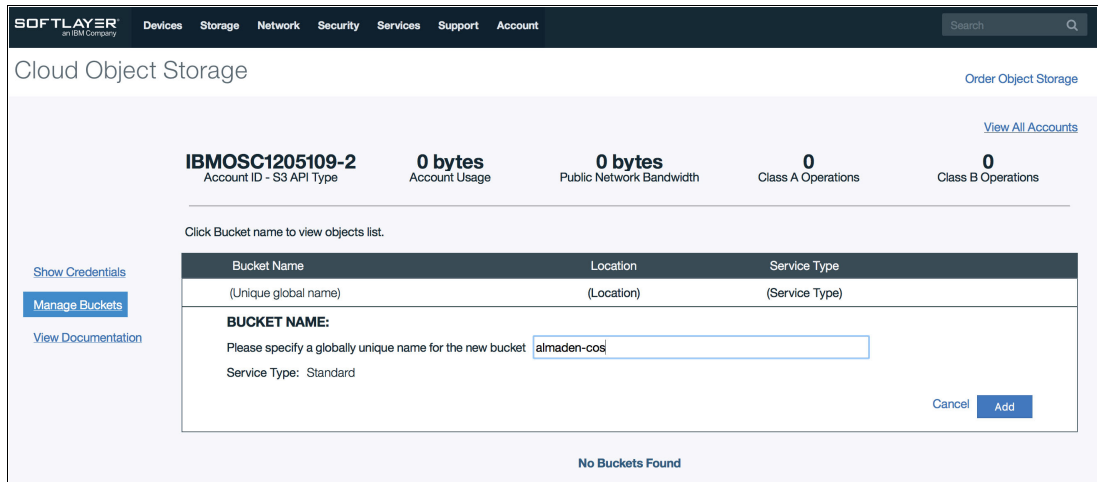


Figure 3-6 Creating an IBM COS bucket

Naming convention: Bucket names must be in the range of 3 - 63 characters and consist of lowercase letters, numbers, and dashes. A common approach to ensure uniqueness is to append a UUID suffix to bucket names.

3. Your bucket is created. To manage objects within the bucket, click the bucket name in the list. See Figure 3-7.

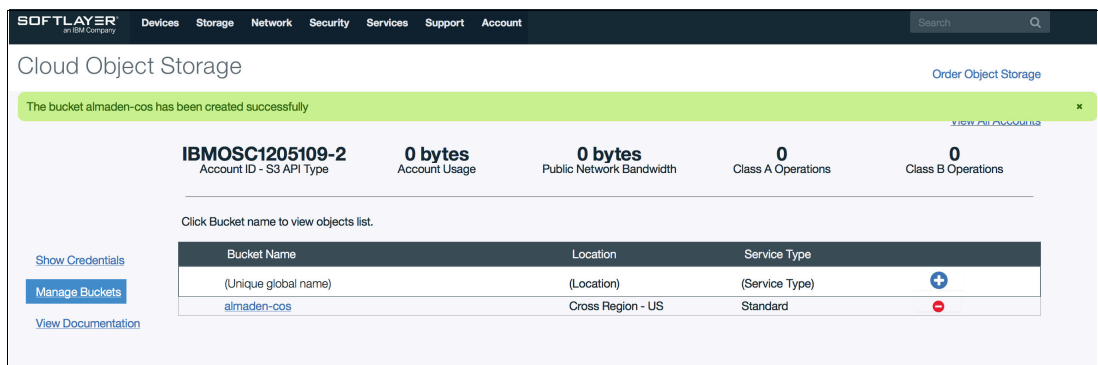


Figure 3-7 IBM COS bucket list

3.3.2 Managing objects from Bluemix Infrastructure

After you click a bucket name, the portal lists the objects within the bucket. From here you can delete and upload new objects to the bucket.

To upload an object to the bucket, click the blue **Add** symbol. You can then select an object to upload and click the **Add** button to upload the object. See Figure 3-8.

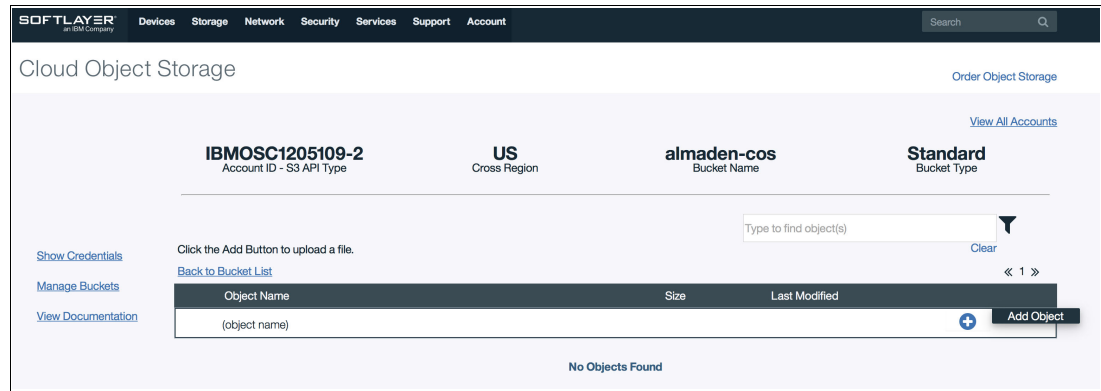


Figure 3-8 Upload a file to a bucket

Within the Bluemix Infrastructure portal, the upload size limit on objects is 20 MB. To upload objects larger than 20 MB a client UI or API call is required. For more details about these methods, see 3.4.2, “Supported API operations” on page 34.

3.4 IBM COS S3 API and common functions

IBM Cloud Object Storage Public Services has three separate APIs for managing and using object storage:

- ▶ Account and Credential administration uses the Bluemix Infrastructure API
- ▶ Interaction with buckets and objects using an implementation of the S3 API
- ▶ Interaction with buckets and objects using the IBM COS S3 API

For more information about using the Bluemix Infrastructure API for credential administration, capacity consumption checks, retrieval of UUIDs, and other account functions, see the [SoftLayer SLN Reference web page](#).

IBM COS Public Services has an implementation of the S3 API that supports the most commonly used subset of Amazon S3 API operations. For a complete list of supported functions see the [IBM Cloud Object Storage API Developer Guide](#).

Notes:

- ▶ At the time of writing this book, the *IBM Cloud Storage Object API Developer Guide* is published on the GitHub location.
- ▶ The *IBM COS S3 API Developer Guide* details the subset of the S3 API that is supported. Any undocumented methods are unsupported.

3.4.1 S3 and IBM COS S3 API differences

Although the IBM COS S3 API is based on the S3 API standard, some features in the IBM COS S3 API are unique to IBM COS. These are described Table 3-1.

Table 3-1 S3 versus CSO API differences

Feature	S3	IBM COS S3
Object Size Limitation	5 TB	No explicit limit for single request uploads. When uploading objects through multipart upload, part size and part count limits are enforced.
Retained Version Count Limitations ^a	No explicit limit	Maximum of 1000 versions allowed per object
Bucket Granular ACL ^a	Read, Write, Read_ACP, Write_ACP, Full_control	Users configured in the Manager Web Interface can be granted read/write, read-only or no-access permissions to any vault. These settings apply to the entire bucket.
Bucket Granular Data Reliability ^a	Allows a storage class to be configured for each object. All objects stored in any bucket share the same reliability characteristics.	Bucket reliability characteristics are determined at creation time.
Traditional Authentication Mechanisms ^a	Uses a custom HTTP scheme based on a keyed-hashed method authentication code (HMAC)	In addition to Access Key authentication, these supported methods are also supported: <ul style="list-style-type: none">▶ HTTP Basic over HTTP and HTTPS▶ PKI over HTTPS▶ Anonymous
Separated Audit and Logging Functions ^a	Performed via AWS CloudTrail	Accesser node collects both access logs and audit trail information but does not expose it through the API.
Encryption and Cryptographic Integrity ^a	<ul style="list-style-type: none">▶ Server-side encryption available▶ Client-side encryption possible	<ul style="list-style-type: none">▶ A bucket can be configured to store information in an encrypted form.▶ Configured at the bucket level.▶ Settings cannot be viewed or edited through API.▶ Request signing is supported.▶ Non-cryptographic.▶ MD5 checksums are calculated and stored with objects.
Lifecycle Configuration	Transitional and expiration actions	N/A

Feature	S3	IBM COS S3
Website Hosting	Set the website configuration for a bucket	N/A
Bucket Location Constraints ^a	Allows buckets to be created with specific location constraints.	Can configure a system to allow data in one bucket to be in a separate geographical location from data on another bucket.
Hard Quota Function	N/A	A hard quota can be configured for an object bucket. HTTP status code 507 is returned for a write request that would cause a hard quota to be exceeded

a. Available for dedicated IBM COS instances only.

3.4.2 Supported API operations

This section describes the complete set of supported operations when using the S3 API to access IBM Cloud Object Storage. For details about using the operations, including examples, see 3.5, “Clients, CLIs, and SDKs” on page 40.

Operations on the account

The only operation that is performed directly at the account level is to GET a list of buckets owned by that account, as shown in Table 3-2. Accounts are currently limited to 100 buckets.

Table 3-2 Account operations

Account operation	Description
GET account	Used to get a list of all buckets belonging to the account

Example 3-1 shows the syntax of this command.

Example 3-1 Syntax of listing a bucket

```
GET https://{endpoint}/
```

Example 3-2 shows how to list all buckets belonging to a specific IBM COS Account through the authorization string.

Example 3-2 List buckets belonging to an account

```
GET / HTTP/1.1
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
X-Amz-Date: 20170209T030815Z
Authorization: {authorization-string}
```


Operations on buckets

The operations, listed in Table 3-3, create, destroy, get information about, and control behavior of buckets.

Table 3-3 Bucket operations

Bucket operation	Description
DELETE bucket	Deletes an empty bucket.
DELETE bucket CORS ^a	Deletes any cross-origin resource sharing configuration set on a bucket.
GET bucket	Lists objects contained in a bucket. Limited to listing 1000 objects at once.
GET bucket ACL	Retrieves the access control list for a bucket.
GET bucket CORS	Retrieves any cross-origin resource sharing configuration set on a bucket.
HEAD bucket	Retrieves a bucket's headers.
GET multipart uploads	Lists multipart uploads that did not complete or were cancelled.
PUT bucket	Creates a bucket.
PUT bucket ACL	Creates an access control list for a bucket.
PUT bucket CORS	Creates a cross-origin resource sharing configuration for a bucket.

a. cross-origin resource sharing (CORS)

Example 3-3 shows the syntax of creating a bucket with the **PUT** command and Example 3-4 shows how to use the command to create a bucket named `images`.

Example 3-3 Syntax of creating a bucket

```
PUT https://{endpoint}/{bucket-name} # path style
PUT https://{bucket-name}.{endpoint} # virtual host style
```

Example 3-4 Creating a bucket named `images`

```
PUT /images HTTP/1.1
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
X-Amz-Date: 20170209T052842Z
Authorization: {authorization-string}
```

Example 3-5 shows creation of a cold vault bucket.

Example 3-5 Creating a cold vault bucket named `frosty`

```
PUT /frosty HTTP/1.1
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
X-Amz-Date: 20170315T124715Z
Authorization: {authorization-string}

<CreateBucketConfiguration>
  <LocationConstraint>us-cold</LocationConstraint>
</CreateBucketConfiguration>
```

Example 3-6 shows the syntax of listing objects in a bucket with the **GET** command.

Example 3-6 Syntax of listing objects in a given bucket

```
GET https://{endpoint}/{bucket-name} # path style
GET https://{bucket-name}.{endpoint} # virtual host style
```

When a **GET** request is given to a specific container, a list of the contents is returned. This listing is limited to the first 1000 objects.

Example 3-7 Listing contents of a bucket named brown

```
GET /brown HTTP/1.1
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
X-Amz-Date: 20170209T225156Z
Authorization: {authorization-string}
```

Creating an access control list for a bucket

A **PUT** command issued to a bucket with the proper parameters creates an access control list (ACL) for that bucket. Access control lists can grant different sets of permissions to different storage accounts using the account's ID, or by using a pre-made ACL.

Note: Credentials are generated for each IBM COS account, not for individual users within Bluemix Infrastructure. As such, ACLs do not have the ability to restrict or grant access at the user level, only to a storage account.

However, `public-read-write` can be set but should be used with extreme caution because it allows any other IBM COS storage account to access the resource, and also the general public through the public endpoints and anyone connected to Bluemix Infrastructure through the private endpoints.

The assumption is that security is maintained in the application.

ACLs can use pre-made permission sets commonly referred to as *canned ACLs*, or be customized in the body of the request. Pre-made ACLs are specified by using the `x-amz-acl` header with `private`, `public-read`, or `public-read-write` as the value. Custom ACLs are specified using XML in the request body and can grant `READ`, `WRITE`, `READ_ACP` (read ACL), `WRITE_ACP` (write ACL) or `FULL_CONTROL` permissions to a given storage account. Example 3-8 sets an ACL on a bucket.

Example 3-8 Setting an ACL on a bucket

```
PUT https://{endpoint}/{bucket-name}?acl= # path style
PUT https://{bucket-name}.{endpoint}?acl= # virtual host style
```

Example 3-9 shows how specifying a pre-made ACL can allow for public-read access to the brown bucket. This allows any storage account to view the bucket's contents and ACL, and to access objects.

Example 3-9 Setting an ACL on a bucket

```
PUT /brown?acl= HTTP/1.1
Host: s3-api.us-geo.objectstorage.softlayer.net
x-amz-date: 20170209T224856Z
x-amz-acl: public-read
Authorization: {authorization-string}
```

Create a cross-origin resource sharing configuration for a bucket

A **PUT** issued to a bucket with the proper parameters creates or replaces a *cross-origin resource sharing* (CORS) configuration for a bucket (Example 3-10).

Example 3-10 Applying CORS to a bucket

```
PUT https://{endpoint}/{bucket-name}?cors= # path style
PUT https://{bucket-name}.{endpoint}?cors= # virtual host style
```

Example 3-11 shows how to set a CORS configuration that allows requests from www.ibm.com to issue **GET**, **PUT**, and **POST** requests to the bucket.

Example 3-11 Setting a cross-origin resource sharing configuration on a bucket

```
GET /brown?cors= HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20170210T194856Z
x-amz-content-sha256:
22938f51643d63c864fdbea618fe71b13579570a86f39da2837c922bae68d72df
Content-MD5: GQmpTNpru0yK6YrxHnpj7g==
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
Content-Length: 237
```

```
<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>http://www.ibm.com</AllowedOrigin>
    <AllowedMethod>GET</AllowedOrigin>
    <AllowedMethod>PUT</AllowedOrigin>
    <AllowedMethod>POST</AllowedOrigin>
  </CORSRule>
</CORSConfiguration>
```

Operations on objects

The operations, listed in Table 3-4, create, destroy, get information about, and control behavior of objects.

Table 3-4 Object operations

Object operation	Description
DELETE object	Deletes an object from a bucket.
DELETE multiple objects	Deletes multiple objects from a bucket.
GET object	Retrieves an object from a bucket.
GET object ACL	Retrieves an object's access control list.
HEAD object	Retrieves an object's headers.
OPTIONS object	Checks CORS configuration to see if a specific request can be sent.
POST object	Adds an object to a bucket using HTML forms.
PUT object	Adds an object to a bucket.
PUT object ACL	Creates an access control list for an object.
PUT object (copy)	Creates a copy of an object.
Initiate multipart upload	Creates an upload ID for a given set of parts to be uploaded.
Upload part	Uploads a part of an object associated with an upload ID.
Upload part (copy)	Uploads a part of an existing object associated with an upload ID.
Complete multipart upload	Assembles an object from parts associated with an upload ID.
Abort multipart upload	Aborts upload and deletes outstanding parts associated with an upload ID.
List parts	Returns a list of parts associated with an upload ID.

Upload an object

A **PUT**, given a path to an object, uploads the request body as an object. A SHA256 hash of the object is a required header. Example 3-12 shows the syntax of this command.

Example 3-12 Syntax to put an object in a bucket

```
PUT https://{endpoint}/{bucket-name}/{object-name}= # path style
PUT https://{bucket-name}.{endpoint}/{object-name}= # virtual host style
```

Example 3-13 shows how an object, queen-bee, is created and a string of text is inserted into that object.

Example 3-13 Uploading an object named queen-bee to the apiary bucket

```
PUT /apiary/queen-bee HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20170210T204856Z
x-amz-content-sha256:09721641329cf441f3fa16ef996cf24a2505f91be3e752ac9411688e34354
Content-Type: text/plain; charset=utf-8
Host: s3-api.us-geo.objectstorage.softlayer.net
Content-Length: 533
```

The ‘queen’ bee is developed from larvae selected by worker bees and fed a substance referred to as ‘royal jelly’ to accelerate sexual maturity. After a short while the ‘queen’ is the mother of nearly every bee in the hive, and the colony will fight fiercely to protect her.

Creating an ACL for an object

As with ACLs for buckets, as discussed in “Creating an access control list for a bucket” on page 36, an ACL can be set on an individual object. Example 3-14 shows the syntax of the command to set an ACL.

Important: There is not the granularity to grant WRITE access at the object level, only at the bucket level.

Example 3-14 Syntax for setting an ACL on an object

```
PUT https://{endpoint}/{bucket-name}/{object-name}?acl= # path style
PUT https://{bucket-name}.{endpoint}/{object-name}?acl= # virtual host style
```

Example 3-15 shows how to use a custom ACL allowing another account to view the ACL for the queen-bee object, but not to access the object itself. Additionally, a third account is given full access to the same object as another element of the same ACL

Example 3-15 Specifying a custom ACL

```
PUT /apiary/queen-bee?acl= HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20170210T114356Z
Content-Type: text/plain
Host: s3-api.us-geo.objectstorage.softlayer.net
Content-Length: 811

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy
xmlns="http://s3-api.us-geo.objectstorage.softlayer.net/skc/2017-02-10/">
  <Owner>
    <ID>{owner-storage-account-uuid}</ID>
    <DisplayName>OwnerDisplayName</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:type="CanonicalUser">
        <ID>{first-grantee-storage-account-uuid}</ID>
        <DisplayName>GranteeDisplayName</DisplayName>
      </Grantee>
      <Permission>READ_ACP</Permission>
    </Grant>
    <Grant>
      <grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:type="CanonicalUser">
        <ID>{second-grantee-storage-account-uuid}</ID>
        <DisplayName>GranteeDisplayName</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

3.4.3 Authorization and authentication

Each request made against IBM COS using the API must be authenticated by using an implementation of the S3 authorization header. IBM COS supports Signature Version 2 and Signature Version 4 authentication methods. Signature Version 4 is considered more secure because it uses a derived signing key rather than the secret access key itself as part of the signature. Using a signature provides in-transit integrity and identity verification, and each signature is tied to the timestamp of the request makes it impossible to reuse authorisation headers.

The authorization header is composed of four components:

- ▶ Algorithm declaration
- ▶ Credential information
- ▶ Signed headers
- ▶ The calculated signature

Example 3-16 shows the format of an authorization header.

Example 3-16 Authorization header format

AWS4-HMAC-SHA256

Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-date;{other-required-headers},Signature={signature}

The date is provided in YYYYMMDD format, with the region for COS cross-region being us-standard. The host and x-amz-date headers are always required and, depending on the request, other headers might also be required (for example x-amz-content-sha256 in the case of requests with payloads). Because of the need to recalculate the signature for every individual request, developers might prefer to use a tool or SDK that will automatically produce the authorization header.

For more information about authorization headers, including creation and example output see the [Authorizat on and authentication](#) web page.

3.5 Clients, CLIs, and SDKs

A range of desktop clients, command-line interface (CLI) tools, and software development kits (SDKs) are available for use with IBM COS and can be used for application integration, automation, or basic manual tasks where required.

3.5.1 Desktop clients

This section describes two desktop clients for IBM COS: CloudBerry and Cyberduck.

CloudBerry

CloudBerry is a flexible backup utility you can use to back up some or all of a local file system to an S3 API-compatible object storage system. It can be configured to be run either manually or scheduled based on need, and can back up separate directories to separate buckets if needed. You can download the utility from the [CloudBerry Lab](#) website.

Cyberduck

Cyberduck is a popular and open source S3 and FTP client that can calculate the authorization signatures that are required to connect to IBM COS. Cyberduck can be downloaded for multiple operating systems from the [Cyberduck](#) website.

To use Cyberduck to create a connection to IBM COS and synchronize a folder of local files to a bucket, follow these steps:

1. Download, install, and start Cyberduck.
2. The main window of the application opens (Figure 3-9), where you can create a connection to IBM COS. Click **Open Connection** to configure a connection to IBM COS.

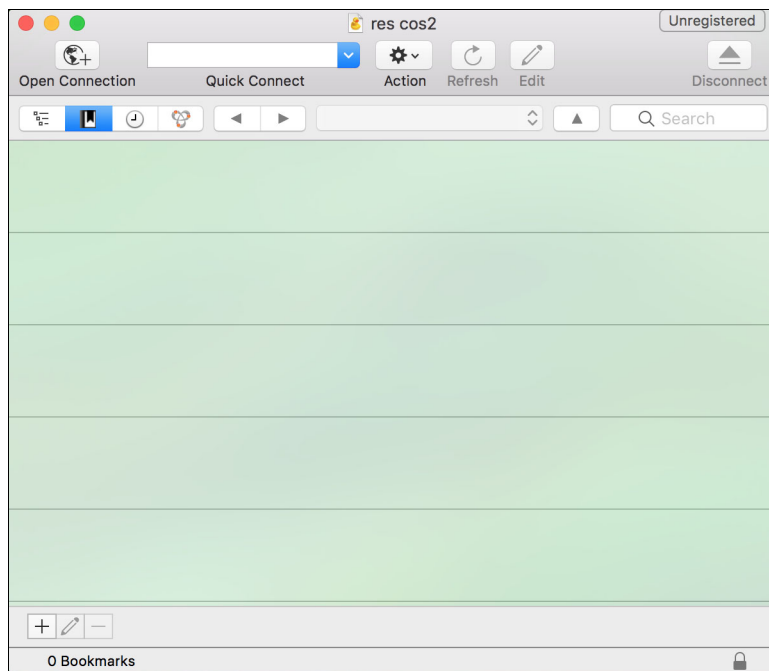


Figure 3-9 Cyberduck main window

3. A pop-up window opens (Figure 3-10 on page 42). From the drop-down menu at the top, select the **S3** storage. Enter information into the following fields, and then click **Connect**:
 - Server: enter the nearest endpoint of IBM COS
 - Access Key ID
 - Secret Access Key

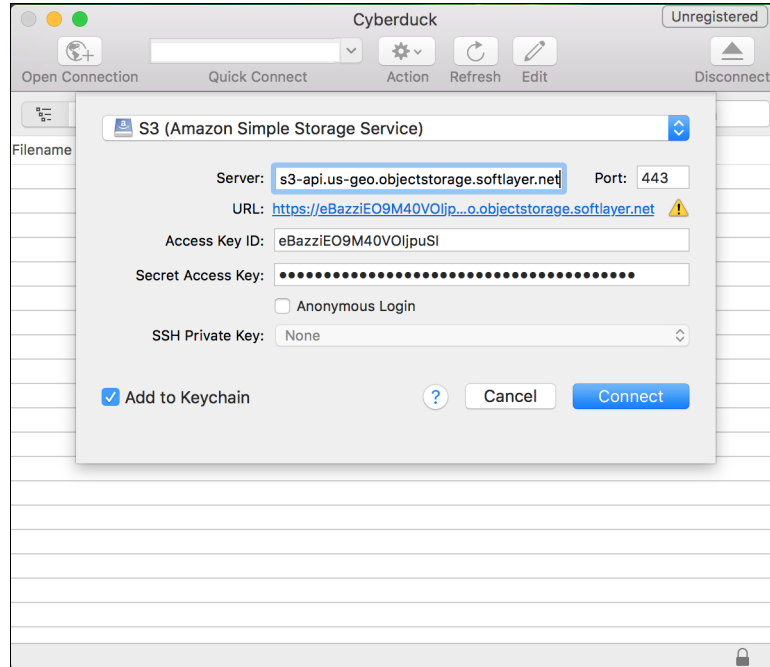


Figure 3-10 Creating connection to IBM COS

4. Cyberduck takes you to the root of the account where buckets can be created. Right-click within the main panel and select **New Folder** (the application deals with many transfer protocols where Folder is the more common container construct).

A dialog opens (Figure 3-11). Enter the bucket name and then click **Create**.

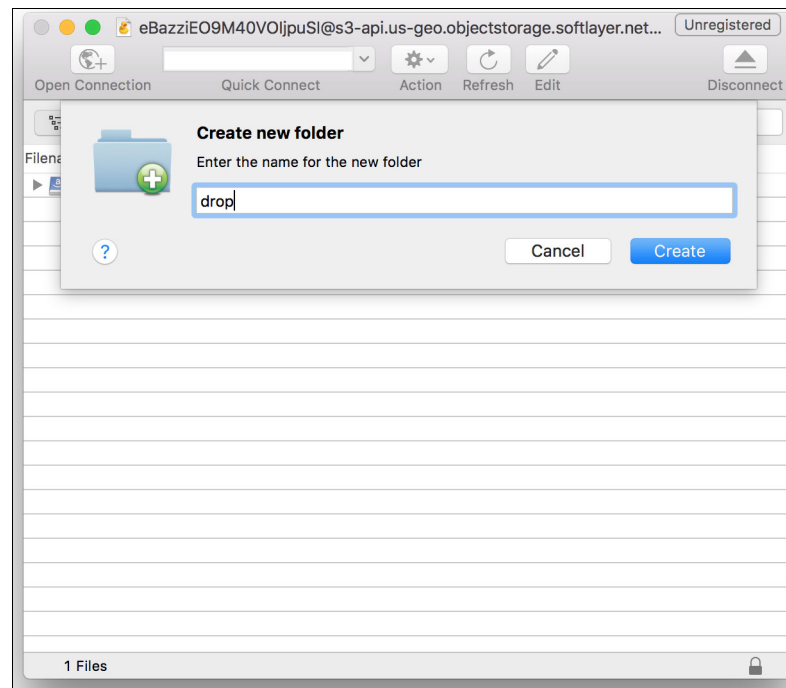


Figure 3-11 Creating a bucket

5. After the bucket is created, double-click the bucket to view it. Within the bucket you can perform various functions such as upload files to the bucket, list bucket contents, download objects from the bucket, synchronize local files to a bucket, synchronize objects to another bucket, or create an archive of a bucket.

6. Right-click within the bucket and select **Synchronize**.

A pop-up panel opens (Figure 3-12) where you can browse to the folder that you want to synchronize to the bucket. Select the folder and click **Choose**.

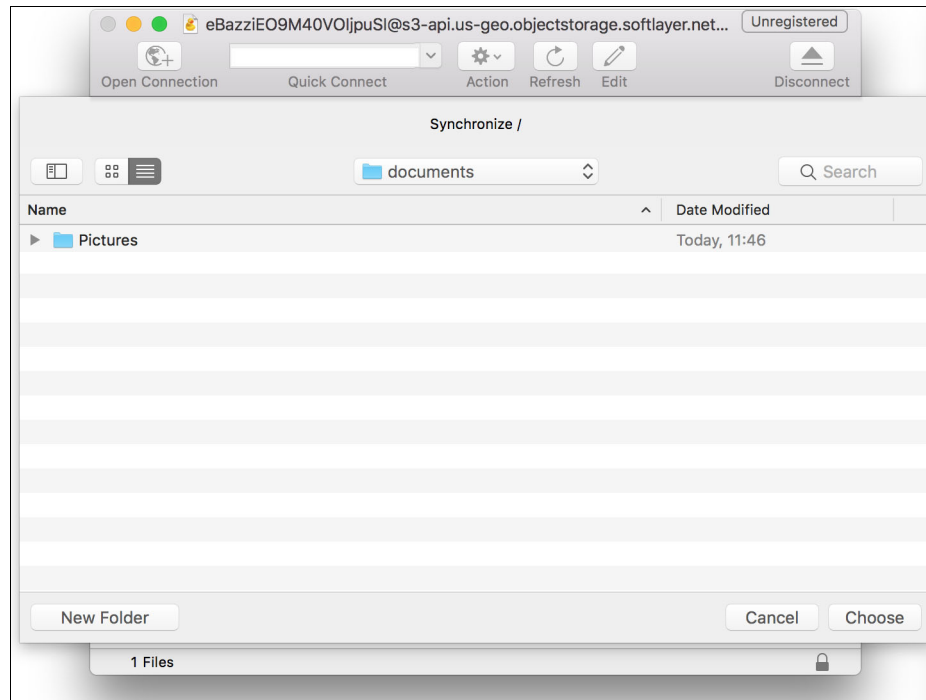


Figure 3-12 Selecting source folder for synchronization

7. After you select the folder, a new pop-up panel opens.

Here, a drop-down menu is available where you select the synchronization operation with the bucket. Three possible synchronize options are available from the menu (Figure 3-13 on page 44):

- Download: This will download changed and missing objects from the bucket.
- Upload: This will upload changed and missing files to the bucket.
- Mirror: This will perform both download and upload operations, ensuring that all new and updated files and objects are synchronized between the local folder and the bucket.

Select **Upload** from the menu and click **Continue** to start the synchronization request.

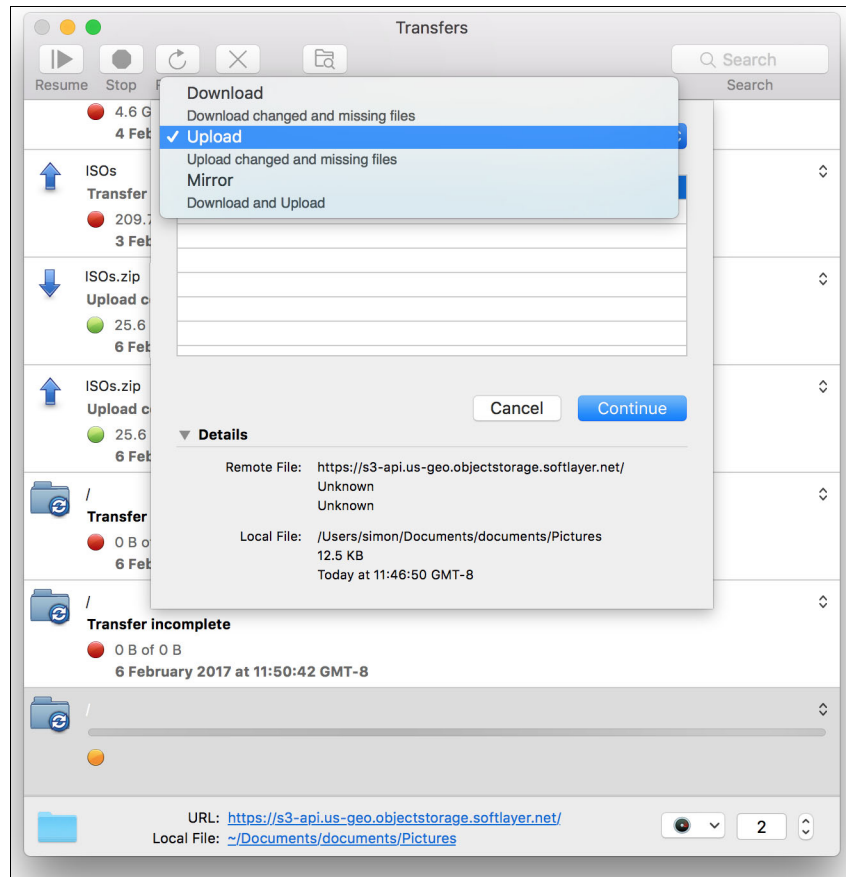


Figure 3-13 Select a synchronization request operation

Another window opens to show active and historical transfer requests (Figure 3-14).

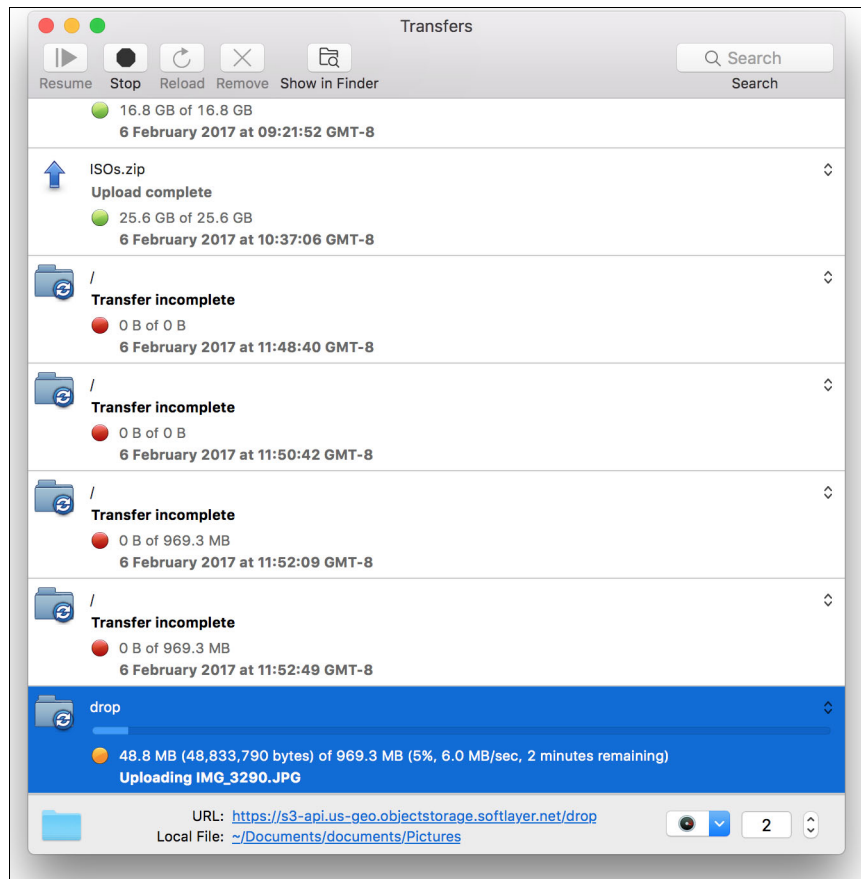


Figure 3-14 Transfer request activity window

After the synchronization request is complete, the main window will perform a list operation on the bucket to reflect updated content in the bucket (Figure 3-15).

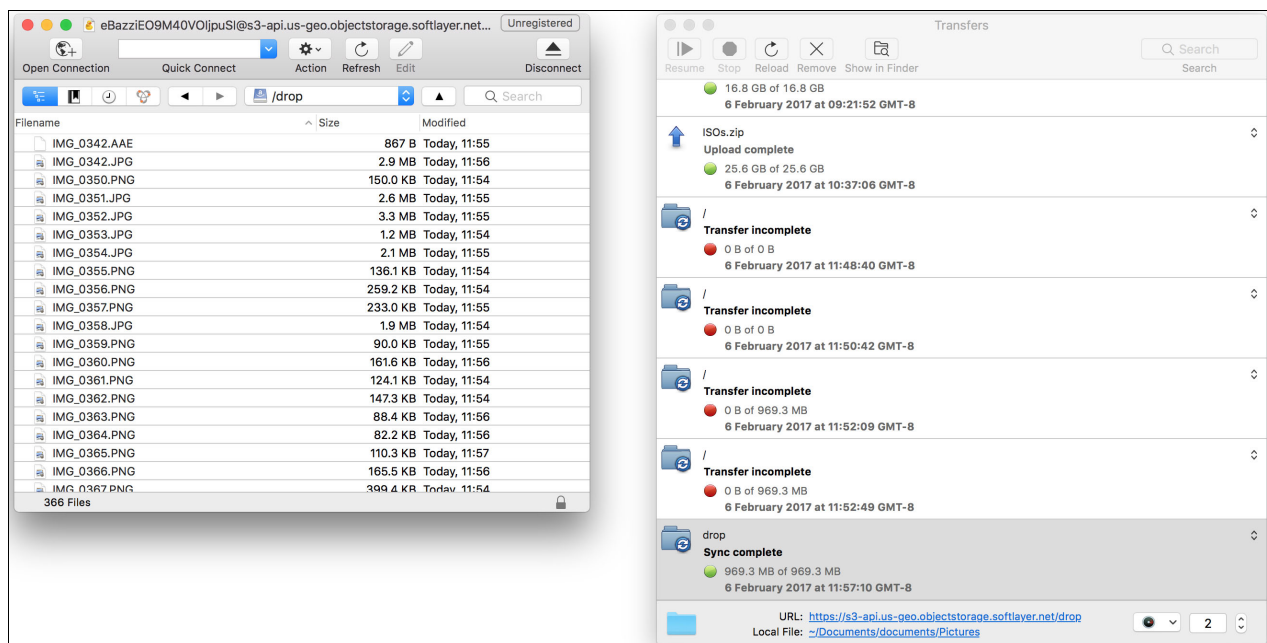


Figure 3-15 Synchronisation complete

3.5.2 Command-line interfaces (CLIs)

The Amazon Web Services (AWS) Command Line Interface (CLI) can be used to control multiple IBM COS accounts from the command line and automate functions through scripts. The AWS CLI Simple Storage Service (S3) tool can be used to access any S3-capable storage platform including any IBM COS systems.

Using the AWS CLI with IBM COS requires a slight modification, which is to set the `--endpoint-url` (Example 3-17) to point to the IBM COS endpoint either through the IP or DNS name.

Example 3-17 Listing buckets in a COS instance from the San Jose endpoint

```
aws --endpoint-url=https://s3-api.sjc-us-geo.objectstorage.softlayer.net s3 ls
```

Consult the *IBM Cloud Storage System API Guide* for details about the functional differences between the S3 API and the IBM Cloud Object Storage System API, and also the subset of S3 API features that are supported.

Note: At the time of writing this book, the IBM Cloud Object Storage System API Guide has not been released.

After the AWS CLI is installed, set up a default profile using the `aws configure` command. This places a file containing your Access Key ID and Secret Access key within the user's home directory. Multiple profiles can be created by using the following command

```
aws configure --profile={profileName}
```

The first credential given in the initial `aws configure` command is set as the default profile.

Additional options can be used in the main AWS CLI command, as shown in Table 3-5. For more information about these options and examples, see the [Using High-Level s3 Commands with the AWS Command Line Interface](#) web page of the Amazon documentation.

Table 3-5 Supported command-line options

Command	Description	Supported	Unsupported options
cp	Copies a local file or IBM COS object to another location locally or in IBM COS.	Yes	--acl Bucket-owner-read option --acl Bucket-owner-full-control --website-redirect --source-region --grants full={emailAddressuserName} --ignore-glacier --force-glacier
ls	Lists IBM COS objects and common prefixes under a prefix or all buckets.	Yes	--output --request-payer
mb	Creates a Bucket.	Yes	
mv	Moves a local file or IBM COS object to another location locally or in COS.	Yes	--acl Bucket-owner-read option --acl Bucket-owner-full-control --website-redirect --source-region --grants full={emailAddressuserName} --ignore-glacier --force-glacier
rb	Deletes a bucket.	Yes	
sync	Synchronizes directories and IBM COS prefixes. Recursively copies new and updated files from the source directory to the destination. Creates folders in the destination only if they contain one or more files.	Yes	--acl Bucket-owner-read option --acl Bucker-owner-full-control --website-redirect --source-region --grants full={emailAddressuserName} --ignore-glacier --force-glacier
website	Sets the website configuration for a bucket.	No	

Examples of AWS CLI uses on IBM COS

This section shows examples of using AWS CLI on IBM COS.

Basic bucket administration

The AWS CLI allows for creating buckets, listing bucket contents, and deleting buckets with the make bucket (**mb**), list bucket (**ls**), and remove bucket (**rb**) commands.

Example 3-18 and Example 3-19 show the creation of buckets.

Example 3-18 Creating a bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mb
s3://sample
```

Example 3-19 Creating a cold-vault bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mb
s3://sample --create-bucket-configuration LocationConstraint=us-cold
```

Example 3-20 and Example 3-21 show the deletion of buckets.

Example 3-20 Deleting a bucket on the private network US geo endpoint

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.networklayer.com s3 rb
s3://sample
```

Example 3-21 Deleting a non-empty bucket

```
aws --endpoint-url=http://s3-api.us-geo.objectstorage.softlayer.net s3 rb
s3://sample --force
```

Example 3-22 shows how to list the contents of a bucket.

Example 3-22 List contents of sample bucket against the San Jose endpoint

```
aws --endpoint-url=http://s3-api.sjc-us-geo.objectstorage.softlayer.net s3 ls
s3://sample
```

Copying Files and objects

The copy (**cp**) operation can upload a file to a bucket, and copy or move an object to another bucket. This operation is shown in Example 3-23 through Example 3-26.

Example 3-23 Copy (upload) a local file to the bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 cp
testFile.txt s3://sample
```

Example 3-24 Copy an object between two buckets

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 cp
s3://sample/testFile.txt s3://sample2/someFile.txt
```

Example 3-25 Copy a series of buckets and objects recursively

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 cp
s3://sample s3://sample2 --recursive
```

Example 3-26 Copy a series of buckets and objects recursively excluding objects that match a pattern

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 cp path
s3://sample/path --recursive --exclude "*.txt"
```

Downloading objects

The copy (**cp**) and move (**mv**) operations download an object, as shown in Example 3-27 and Example 3-28.

Example 3-27 Copy an object to a file

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 cp
s3://sample/testFile.txt retrieve.txt
```

Example 3-28 Move an object to a file

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mv
s3://sample/testFile.txt retrieve.txt
```

Moving objects

The move (**mv**) operation can also be used to move objects within or to other buckets, as shown in Example 3-29 through Example 3-31.

Example 3-29 Move an object between two buckets

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mv  
s3://sample/someFile.txt s3://sample2/someFile.txt
```

Example 3-30 Recursively move all objects between two buckets excluding some objects

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mv  
s3://sample s3://sample2 --recursive --exclude "*.iso"
```

Example 3-31 Move an object between buckets whilst setting the ACL to public-read-write

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 mv  
s3://sample/testFile.txt s3://sample2 --acl public-read-write
```

Deleting (removing) objects

The remove (**rm**) operation deletes an object, as shown in Example 3-32 through Example 3-34.

Example 3-32 Delete a single object

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 rm  
s3://sample/example.txt
```

Example 3-33 Recursively delete all objects in a specified bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 rm  
s3://sample --recursive
```

Example 3-34 Recursively delete objects in a specified bucket while excluding some objects

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 rm  
s3://sample --exclude "*.iso"
```

Synchronizing objects

The sync (**sync**) operation can be used to synchronize a bucket with a local directory or another bucket, as shown in Example 3-35 through Example 3-38 on page 50.

Example 3-35 Sync files from a local directory to a bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 sync  
./upload s3://sample
```

Example 3-36 Sync objects from a source bucket to a target bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 sync  
s3://sample s3://sample2
```

Example 3-37 Sync objects from a source bucket to a target bucket whilst deleting objects in the target bucket that do not exist in the source bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 sync  
s3://sample s3://sample2 --delete
```

Example 3-38 Perform a dry run of syncing a source bucket to a target bucket whilst deleting objects in the target bucket that do not exist in the source bucket

```
aws --endpoint-url=https://s3-api.us-geo.objectstorage.softlayer.net s3 sync  
s3://sample s3://sample2 --delete --dryrun
```

3.5.3 Software development kits (SDKs)

IBM COS Public Services supports multiple SDKs covering .Net, Node-RED, Java, and Python.

SDK guides explain to developers who are familiar with S3 storage how to use the language to port existing applications to IBM COS.

Python SDK for IBM COS

Using the Python SDK with the IBM COS API requires a slight modification of setting the S3 endpoint (details are in 3.5.2, “Command-line interfaces (CLIs)” on page 46). Other minor variations are outlined in the Python SDK Guide.

Python support is provided through the Boto 3 library. Boto can be installed from the Python Package Index through `pip install boto3`. The examples shown here were generated using version 1.4.0 of the boto3 package.

The [Boto SDK API reference](#) provides details about SDK classes and methods.

Note: Existing applications that use Boto 2.x libraries should be compatible too, although Boto 2.x is no longer actively maintained and users are encouraged to migrate to Boto 3.

The boto3 library provides complete access to the S3 API and can source credentials from the following file:

```
~./aws/credentials
```

The IBM COS endpoint must be specified when creating a service resource or low-level client as shown in the following basic examples:

- ▶ “Example service resource script” on page 51
- ▶ “Example low-level client scripts” on page 51

Example service resource script

Creating a service resource provides a greater abstraction for higher level tasks. Example 3-39 is a basic script that fetches the list of buckets owned by an account, and lists the objects in each bucket.

Example 3-39 Listing all buckets and their contents

```
import boto3

endpoint = 'https://s3-api.us-geo.objectstorage.softlayer.net'

s3 = boto3.resource('s3', endpoint_url=endpoint)

for bucket in s3.buckets.all():
    print(bucket.name)
    for obj in bucket.objects.all():
        print("    - %s" % obj.key)
```

Example 3-40 is a response to the script invoked in Example 3-39.

Example 3-40 Script response providing bucket list and contents

```
sample-1
- slc151277-file-00001
- skc090781-file-00002
- cbb170987-file-00003
- bmb130286-file-00004
sample-2
- jcc011177-file-00001
```

Example low-level client scripts

Creating a low-level client allows for considerable more detail and access to metadata. Example 3-41 is a basic script that fetches the list of buckets owned by an account and lists objects in each bucket.

Example 3-41 Script showing buckets with metadata

```
import boto3
import pprint as pp

endpoint = 'https://s3-api.us-geo.objectstorage.softlayer.net'

s3 = boto3.client('s3', endpoint_url=endpoint)

print('These are the buckets in this service account:')
buckets = s3.list_buckets()
pp.pprint(buckets, width=180)

for bucket in buckets['Buckets']:
    name = bucket['name']
    print("Raw output from 'list_buckets()' in %s:" % name)
    objects = s3.list_objects(Bucket=name)
    pp.print(objects)
```

Because considerably more data would be returned than in Example 3-40 on page 51, the `printpp` package is used to increase the readability of the raw output (Example 3-42).

Example 3-42 Creating a cold-vault bucket

```
import boto3
import pprint as pp

endpoint = 'https://s3-api.us-geo.objectstorage.softlayer.net'

s3 = boto3.client('s3', endpoint_url=endpoint)

bucket = s3.create_bucket(Bucket='samepl',
                          CreateBucketConfiguration={
                              'LocationConstraint': 'us-cold'})
```

AWS Java SDK for IBM COS

The IBM COS S3 API supports AWS Signature v2 and v4 authentication on IBM COS.

The AWS Signature v4 signing specification describes how to add authentication information to S3 requests.

Requests using AWS authentication must be signed using the requesting user's access key ID and secret access key.

The order of precedence for using credentials is as follows:

1. Credentials passed as `BasicAWSCredentials` instance parameters
2. Credentials set as environment variables
3. Credentials set as JVM system properties
4. Credentials set in the `AwsCredentials.properties` file
5. Credentials set in the shared credentials file

The AWS Java SDK requires that you use the software versions listed in Table 3-6.

Table 3-6 Software Version Requirements

Software	Software Version
Open JDK	7
AWS SDK for Java	1.10.2

For more details of how to get started with the AWS Java SDK see the [AWS SDK for Java](#) web page.

3.6 Preferred practices for application design

This section describes the considerations for application design when storing and accessing data from the IBM COS public service.

3.6.1 Object name

Object naming must be unique within a bucket. To name an object, append or prepend the object name with a random string or time stamp.

3.6.2 Object size

Object size has certain considerations:

- ▶ Considerations for small objects

The **PUT** operation impact is the same regardless of the size of the object, so small objects still have the same impact as larger objects. When possible, try to merge multiple small objects into larger objects for performance and cost reasons.

- ▶ Considerations for large objects

Larger objects take more time to upload and download, so applications should be designed to cater to network throughput and latency constraints. IBM COS supports the following optimizations:

- Multi-part uploads, which help mitigate upload speeds
- Partial reads that use byte offsets reduce the download size by downloading only small chunks of the object as required

3.6.3 Concurrent PUT/GET

Object storage does not manage object-level locking. The application design must manage object locks for object-level changes that might occur as a result of concurrent **PUT/GET** operations on the same object across multiple users of the application.

3.6.4 Handling timeouts

Timeouts can occur as a result of network latency, connectivity issues, or the size of objects. Such timeouts must be managed by the application tier.

3.6.5 Handling overwrites

Object storage does not stop object overwrites. The potential exists for concurrent users to overwrite each other's version of the object. The application design must manage the locking and version control in such situations.

3.6.6 Handling changes

Object storage does not allow in-place changes to parts of the objects, so any changes regardless of the size of the change must be performed at the application tier and pushed to IBM COS.



Part 2

IBM Cloud Object Storage use cases and scenarios

This part introduces some of the use cases and scenarios that are commonly used for IBM Cloud Object Storage.



Typical applications for IBM Cloud Object Storage

This chapter describes several example IBM Cloud Object Storage (COS) deployments across various industry verticals, provides use cases for IBM COS, and outlines IBM COS integration with other IBM offerings.

The aim of this chapter is to provide to system and application owners, architects, and administrators an outline of how IBM COS can fit into or enhance their business.

The following topics are covered in this chapter:

- ▶ 4.1, “IBM Cloud Object Storage workloads” on page 58
- ▶ 4.2, “Examples from various industries” on page 58
- ▶ 4.3, “Typical use cases” on page 75
- ▶ 4.4, “Integration with other IBM products” on page 91

4.1 IBM Cloud Object Storage workloads

IBM Cloud Object Storage workloads and use cases can be grouped as shown in Figure 4-1.

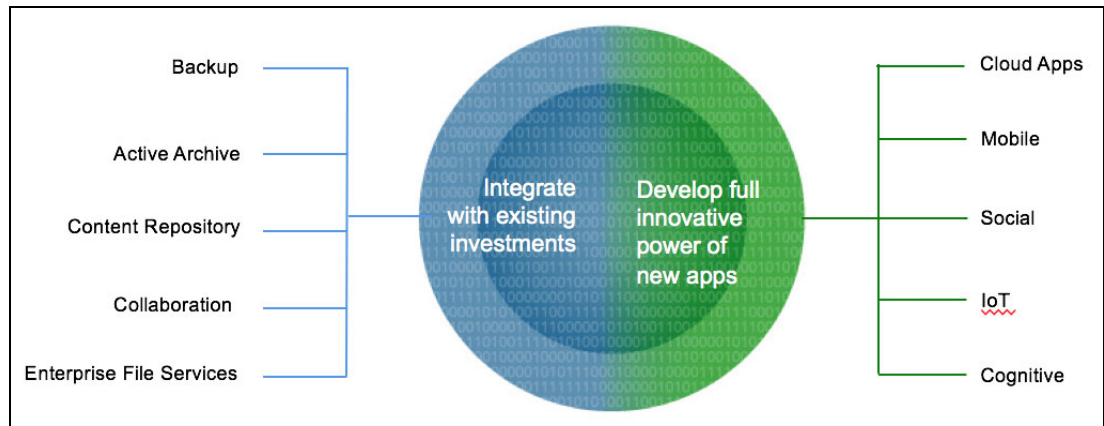


Figure 4-1 IBM Cloud Object Storage workloads

The figure shows two broad categories:

- ▶ The first category targets known and mature workloads that were traditionally serviced by file, block, or tape storage systems.
- ▶ The second category targets “new world” workloads, which are newer applications and solutions that are more programmatically flexible and are more likely to directly access and derive insights from data and expose them to users.

Key point: IBM Cloud Object Storage can service existing workloads and also new world applications by using the same back-end service.

4.2 Examples from various industries

This section discusses industry-specific examples of IBM COS solutions and what business outcomes were achieved.

4.2.1 Example 1: Business and financial management solutions

Most modern businesses, such as those in the finance industry, run one or many transactional database systems that hold critical data. IBM COS can be an excellent solution for some of the data that these environments generate.

Modernize Oracle database protection

A business and financial management provider dealing with small and medium businesses was looking to innovate its legacy database backup solution.

Oracle Recovery Manager (RMAN) was the database backup management layer; in recent versions, this layer supports backups and replication directly over S3 protocol as opposed to streaming to a backup tool or file system.

A mixture of deduplication devices and tape was in use to store the data depending on retention, criticality, and data type.

Figure 4-2 shows the current Oracle backup environment.

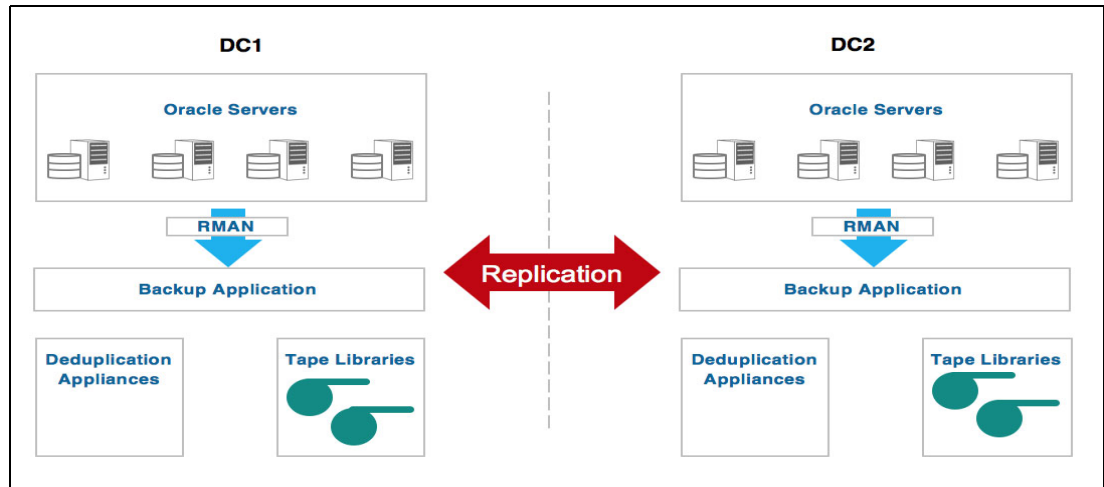


Figure 4-2 Oracle backup

Business problems

The existing backup solution, although functional, had a number of issues affecting business and operations:

- ▶ Scalability:
 - Issues scaling tape infrastructure
 - Issues scaling multiple deduplication appliances
- ▶ Management complexity:
 - Multiple vendors to manage
 - Multiple teams to manage multiple solutions
 - Bad experience with tape
 - Replication management
- ▶ Cost:
 - Desire to reduce costs
 - Desire to reduce tape infrastructure costs

Solution

The solution implemented replaced the legacy backup, tape, and deduplication infrastructure with a single IBM Cloud Object Storage system being written to directly by Oracle RMAN.

Figure 4-3 shows the environment after the solution was implemented.

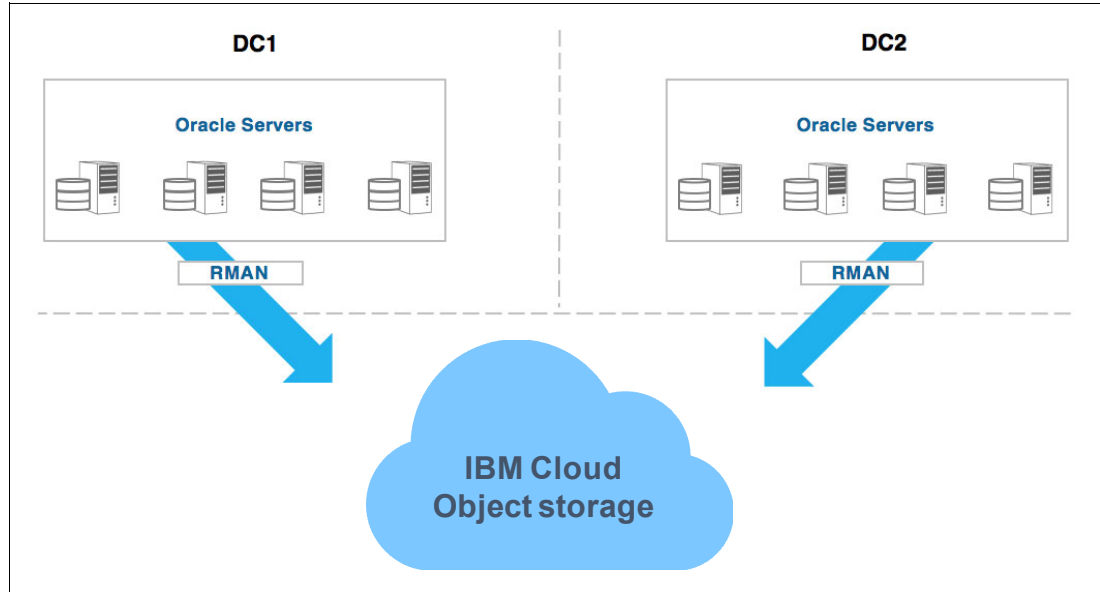


Figure 4-3 Direct to IBM COS Oracle backup

Benefits

The solution has the following benefits:

- ▶ Scalability:
 - Ability to scale to petabytes without management of multiple storage arrays and tape libraries
 - Multiple RMAN streams to almost limitless storage
- ▶ Management complexity:
 - Management of a simple vault URL instead of multiple devices
 - Simple S3 API
 - No SAN to manage
- ▶ Cost:
 - Eliminated spending on legacy tape and disk infrastructure
 - Eliminated spending on backup software
 - Reduced capacity and costs associated with storing multiple copies of data

Key point: IBM COS can reduce operational complexity and total cost of ownership for database protection workloads.

4.2.2 Example 2: Media and entertainment

The media and entertainment industry produces content viewed and heard by billions of people around the world. This content typically consists of large multimedia files and is a natural fit for IBM COS.

Enterprise storage as a service

A large content producer and Pay TV provider used a number of disparate storage systems such as tape for archive and backup, network attached storage (NAS) for corporate applications and user services, and block storage for high-speed access.

Archive data was held on offline storage and if required for broadcast, the data would be recalled back onto disk storage and content was only broadcast via the PayTV service.

Figure 4-4 shows the current environment.

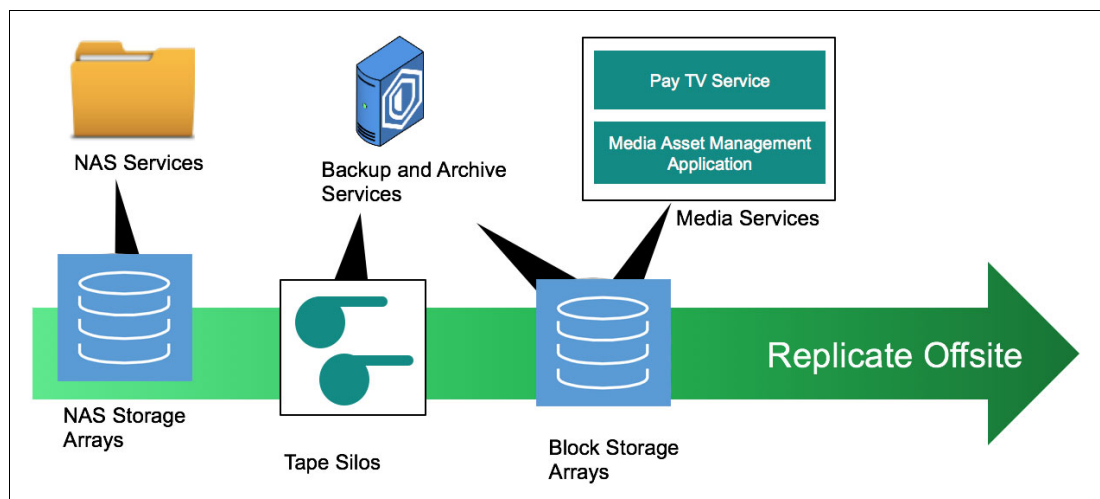


Figure 4-4 Multiple storage silos in media and entertainment

Business problems

The existing systems were an inhibitor to business growth and profitability because of several issues:

- ▶ Scalability:
 - Rapid data growth challenged IT capability to service the business need.
- ▶ Inefficient use of data assets:
 - Inability to easily monetize archive data as tape recall time for large media files meant that online streaming of this data was not practical.
- ▶ Resiliency concerns:
 - Manual intervention during disaster recovery (DR) exposed business to human error and increased recovery times.
- ▶ Cost:
 - Desire to reduce costs.

Solution

The solution that was implemented replaced the legacy backup, tape, and deduplication infrastructure, legacy NAS heads used by applications with cloud NAS gateways, and provided new file sync and share and NAS services for user services and IBM COS integrated file sync and share solution.

Media content is now stored directly through the object interface and is accessible to the conventional media delivery services, and new video on-demand services are accessed by customers.

Figure 4-5 shows the environment after the solution was implemented.

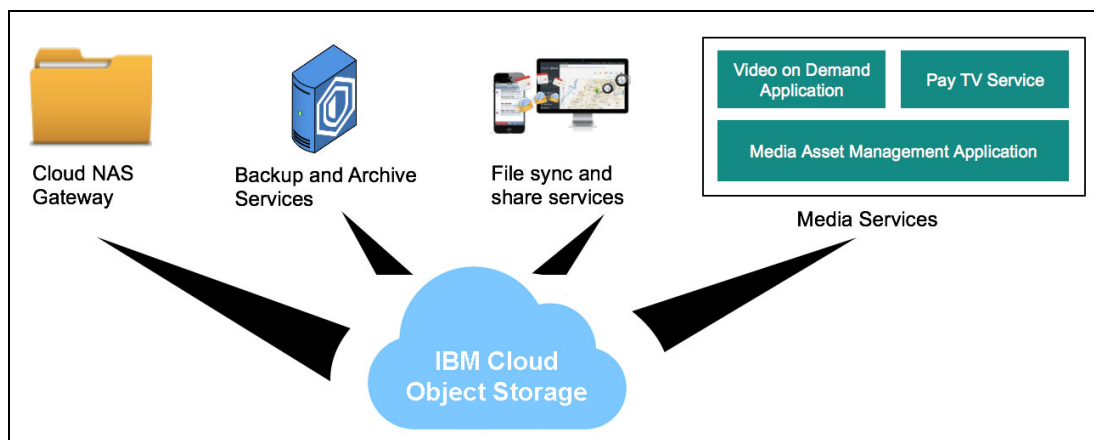


Figure 4-5 Consolidated storage in multimedia and entertainment

Benefits

The solution has the following benefits:

- ▶ Scalability:
 - Agile and simple scalability means that after migration to IBM COS, the business can increase its capacity without concerns of re-platforming, data migration between arrays, and the outages that these can involve. All this can be achieved because the IBM COS service provides *limitless* capacity from a user perspective, while the back end can seamlessly scale in capacity, performance, and resilience without any interruption of service.
- ▶ Efficient use of data assets:
 - Video on-demand services further leverage existing and new media assets.
 - Content can be used directly from IBM COS as opposed to waiting for recall from cold storage.
- ▶ Resiliency:
 - The nature of IBM COS transparent geographically dispersed architecture ensures that data was available across multiple sites and regions regardless of site failure and without a need for a complex DR procedure.
- ▶ Cost:
 - Reduced spending on various hardware platforms through the use of a common back end for all services
 - Operational efficiencies mean that IT staff can spend more time on value-added activities that can further monetize the existing assets.

Key point: IBM COS allows media organizations to more effectively use their media content for on-demand streaming services. Furthermore, the same backend can be used to eliminate storage sprawl and islands of storage that are traditionally seen across large organizations.

4.2.3 Example 3: Education and research

Education and research institutions must grapple with servicing student and researcher expectations within tightly constrained budgets and, in an Internet of Things (IoT) world, a huge quantity of sensor data is used by researchers.

Reducing data sprawl with a hybrid cloud solution

A large education and research institution was looking to upgrade its current active and offline data solution. This included a high-performance computing (HPC) environment that was using the highly parallel IBM Spectrum Scale™ file system, which tiered old data to tape, and also keeping multiple backup copies on tape using IBM Spectrum Protect™.

Figure 4-6 shows the current environment.

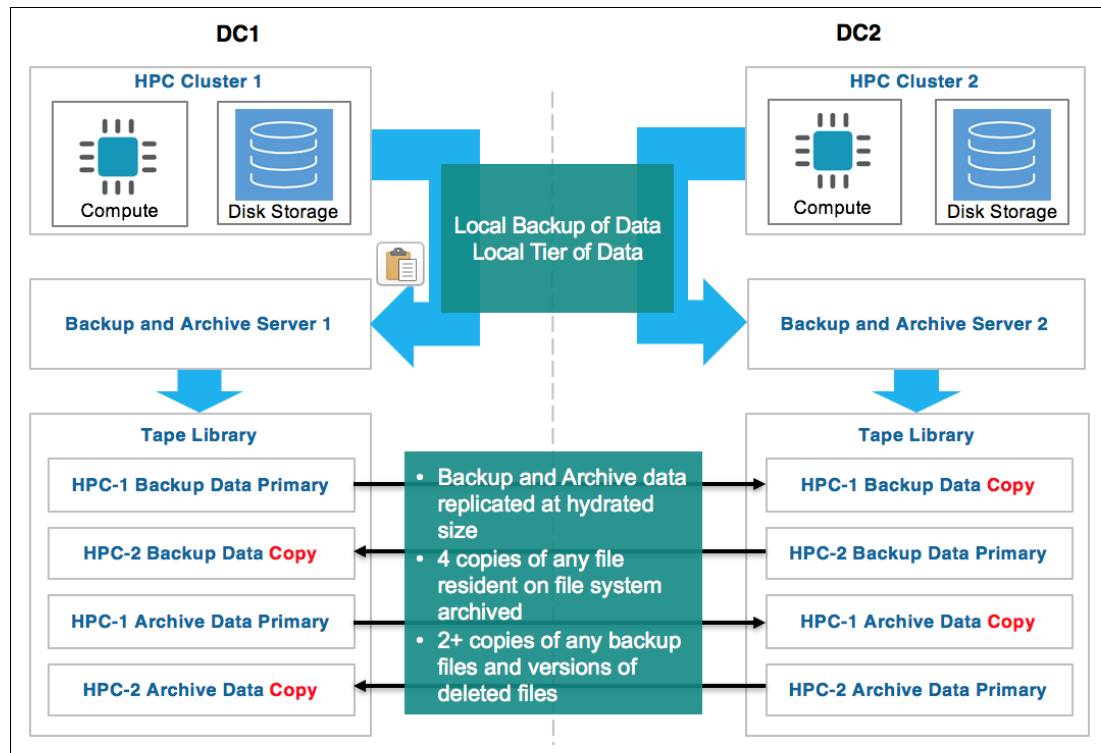


Figure 4-6 Multiple data copies in education and research

Business problems

The current solution and continuing on the traditional path forward had a number of issues:

- ▶ Data sprawl:
 - Making multiple copies of files and keeping them on tape meant most files were stored at least four times across all sites.
 - Advanced data reduction like deduplication not being available on tape further exacerbated the cost.
- ▶ Data center constraints:
 - Inability to fit the projected tape library expansions because of space constraints
- ▶ Recovery concerns:
 - Data replication for all data sets took 8-12 hours meaning disaster recovery prior to this replication could mean losing over 24 hours of data.

Solution

The IBM COS solution consisted of using IBM Spectrum Protect Server native S3 integration to store the data in a single ICOS pool for each HPC system instance.

Figure 4-7 shows the environment after the solution was implemented.

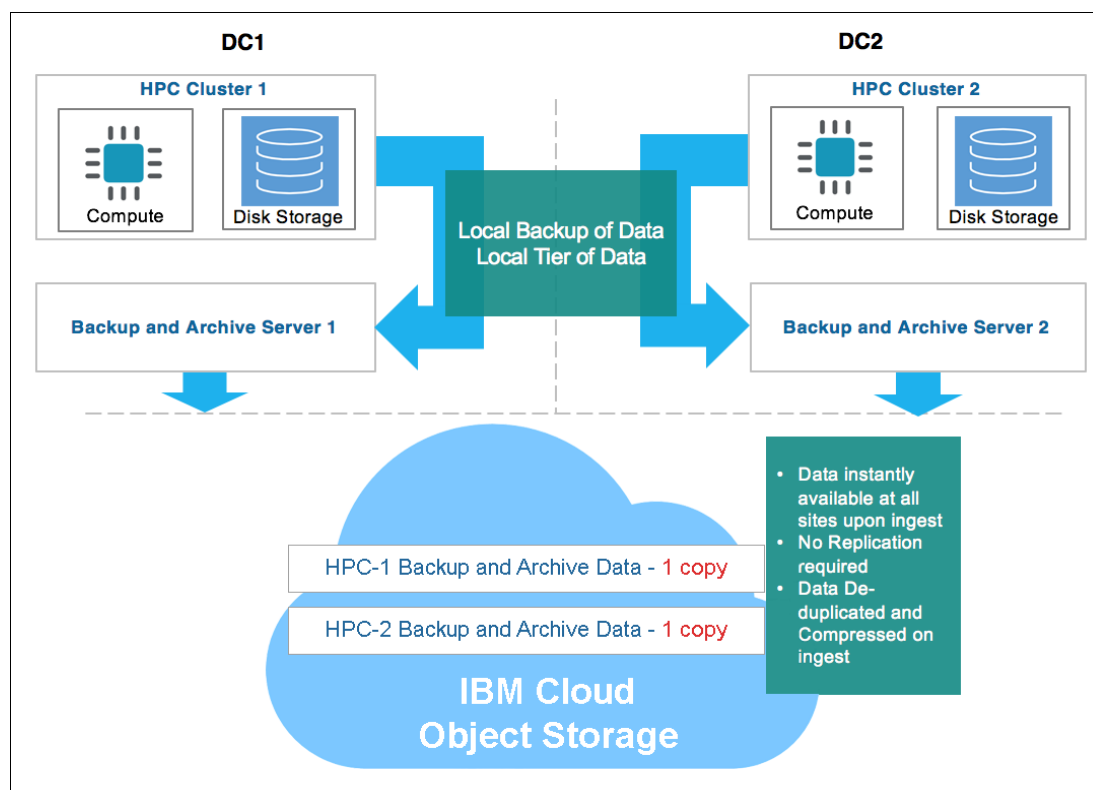


Figure 4-7 Elimination of data copies with IBM COS in education and research

Benefits

The solution has the following benefits:

- ▶ Data sprawl:
 - Number of data copies were reduced to one for archive and backup data.
 - Physical data footprint further is reduced with IBM Spectrum Protect Data deduplication and compression obtaining an average 4:1 data reduction.
- ▶ Data center constraints:
 - Physical Infrastructure requirements were moved to IBM Cloud Object Storage, which freed up data center space for high value HPC equipment core to the research unit operations.
- ▶ Recovery concerns:
 - Backup and archive data instantly is available at the other site, improving the recovery point to be no further back than the latest data backup or archive.

Key point: IBM COS can be effectively used as an archive and data protection back end in many environments including research and HPC.

4.2.4 Example 4: Healthcare industry and medical imaging

The healthcare industry, specifically medical imaging, is growing at an incredible rate. An estimate is that more than 1 billion medical images are stored in the US and that number is growing rapidly each year.

As image sizes increase, aging storage archives and capacity are causing concerns for health systems across the world.

Improving outcomes in healthcare

This organization needed to ensure business continuity to remain in compliance and protect medical records, and also offer new capabilities for providers to collaborate across the enterprise and care continuum in a secure manner on mobile devices.

The system in place was a conventional Patient Archiving and Communications System (PACS) application ingesting data into a local file store as below.

Figure 4-8 on page 66 shows the current environment.

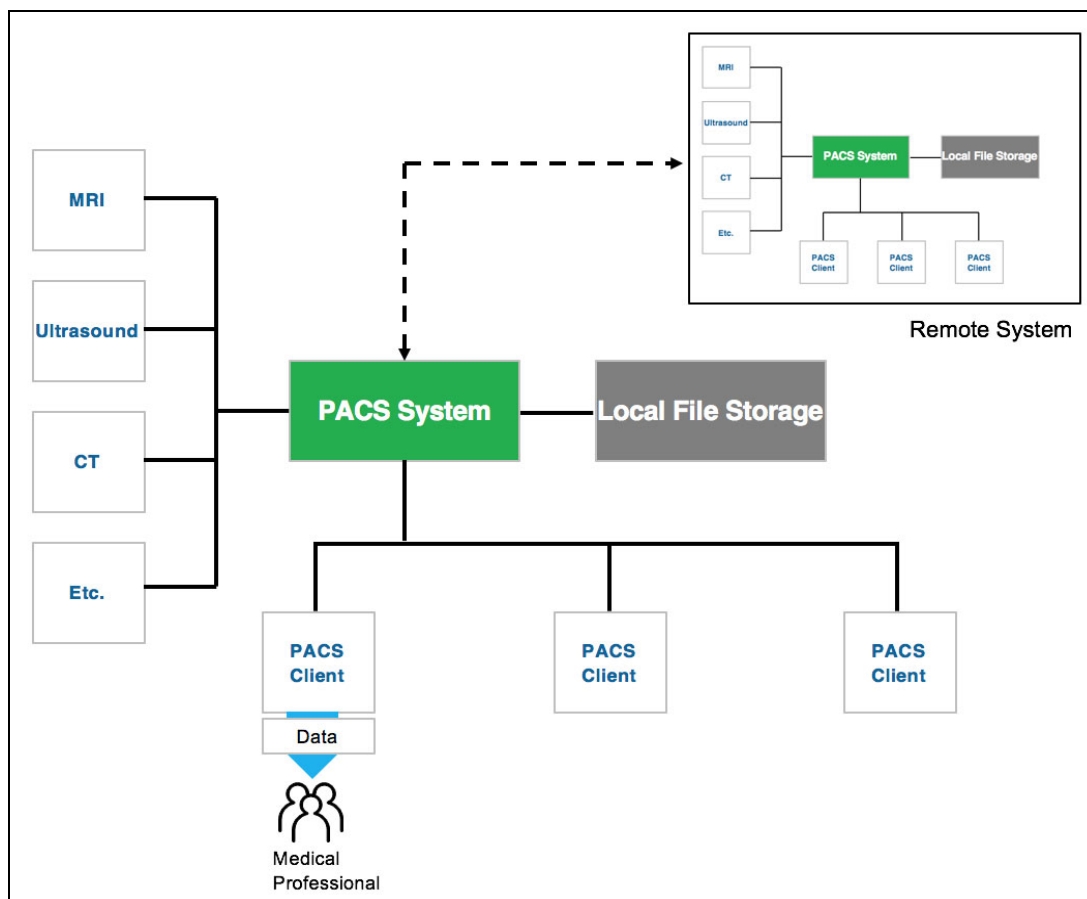


Figure 4-8 Traditional Healthcare systems

Business problems

This healthcare provider had a number of issues and goals to solve in the proposed solution:

- ▶ Compliance and business continuity:
 - Difficulties in providing required data resilience and uptime.
- ▶ Inability to collaborate:
 - Difficulties in sharing patient data securely among multiple practitioners at multiple sites.
- ▶ Scalability concerns:
 - Rapid data growth due to the increased resolutions in medical imaging was an issue with legacy systems scalability limits.
- ▶ Improving patient outcomes:
 - Ensuring each patient has the best outcome possible.

Solution

The solution consists of IBM Cloud Object storage integrated with sensors, IBM Watson, and cognitive analytics as shown in Figure 4-9 on page 67.

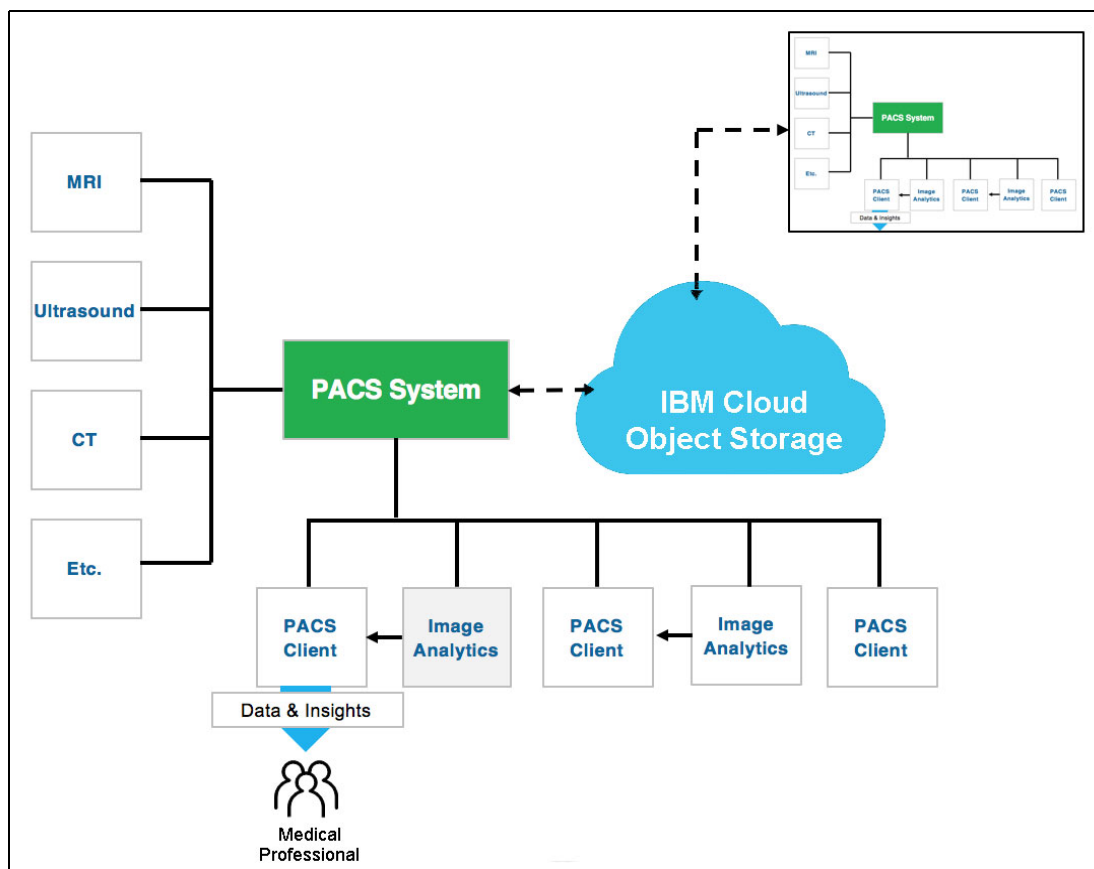


Figure 4-9 Modernized IBM COS integrated healthcare systems

This solution provides the following capabilities:

- ▶ Cloud technologies will transform healthcare in previously inaccessible regions because physical infrastructure can be accessed from anywhere.
- ▶ Players across the healthcare value chain will focus on the continuum of healthcare delivery for patients (single patient and family view).
- ▶ Instantaneous access to digitized patient health records through cloud to specialists anytime and anywhere.
- ▶ Insights based on analytics that are integrated with mobile devices, or smart sensors will be used to improve clinical outcomes.

Benefits

The solution has the following benefits:

- ▶ Compliance and business continuity:
 - Instant multi-site access to medical image archive on-premises, in IBM Cloud or as hybrid.
- ▶ Collaboration:
 - Ability to centralize image archiving across the enterprise.
 - Drive consistent imaging workflow hospital-wide.
 - Support image exchange across affiliated hospital.
 - Securely store and share patient-centric information to speed diagnosis.

- ▶ Scalability:
 - Ability to seamlessly scale without creating data islands.
- ▶ Improved patient outcomes:
 - Improved diagnosis and monitoring

This provider improved its patient quality measurements and outcomes by identifying high-risk, high-cost patients: provided faster and more intuitive training for its employees; and made vital information available to physicians on demand.
 - Innovative service delivery

A provider of healthcare services in the US facilitates consistent delivery of patient care management, decreases dependency on pharmaceutical interventions, and can better measure therapeutic efficacy.
 - Reduce cost and increase value

Key point: IBM COS can be combined with Watson Analytics™ and cognitive analytics. This combination can improve medical outcomes and service quality for healthcare providers.

4.2.5 Example 5: Government authority managing road and traffic

Keeping logistics of a large city flowing is a responsibility of road and traffic authorities in many cities and is a common problem around the world.

Keeping the traffic flowing

This government organization was responsible for staffing control rooms where employees manually analyze city traffic sensor output to maintain traffic flows and project future road and traffic enhancements.

Figure 4-10 shows the initial environment.

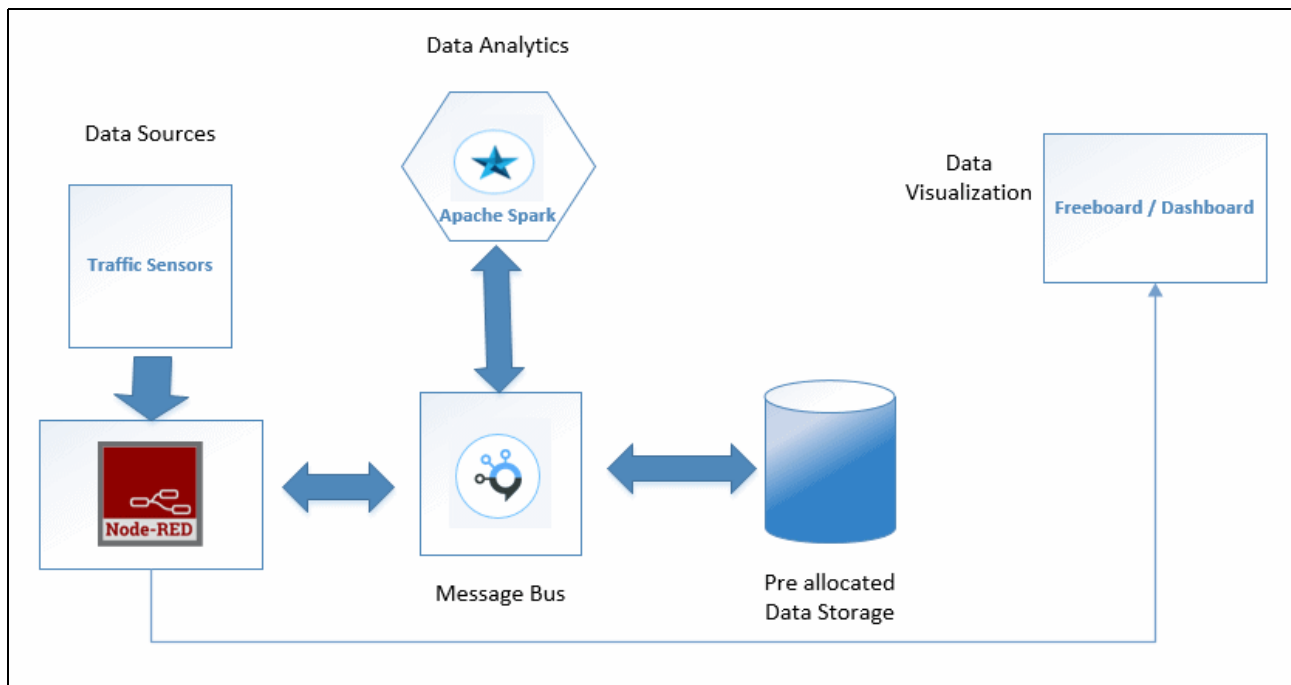


Figure 4-10 Keeping the traffic flowing

Business problems

This entity had to grapple with a number of issues and goals:

- ▶ Inefficiencies in manual analysis:
 - This was time consuming and prone to human error
- ▶ Storing large amounts of sensor data:
 - Storing data on multiple disparate storage islands meant it was difficult to leverage this data for predictive analytics.

Solution

The solution collects historical time series data from IoT devices using data source (traffic sensors), this data is pushed as a message to Kafka through Node-RED and the IoT data will eventually be stored in IBM Cloud Object Storage. These components are described in Table 4-1 on page 70.

The system analyzes traffic conditions recorded in IBM COS and uses a complex event processing (CEP) approach which has these characteristics:

- ▶ Is rule based.
- ▶ Processes event record by record.
- ▶ Automates this process and makes it smarter.

The solution is described in Figure 4-11.

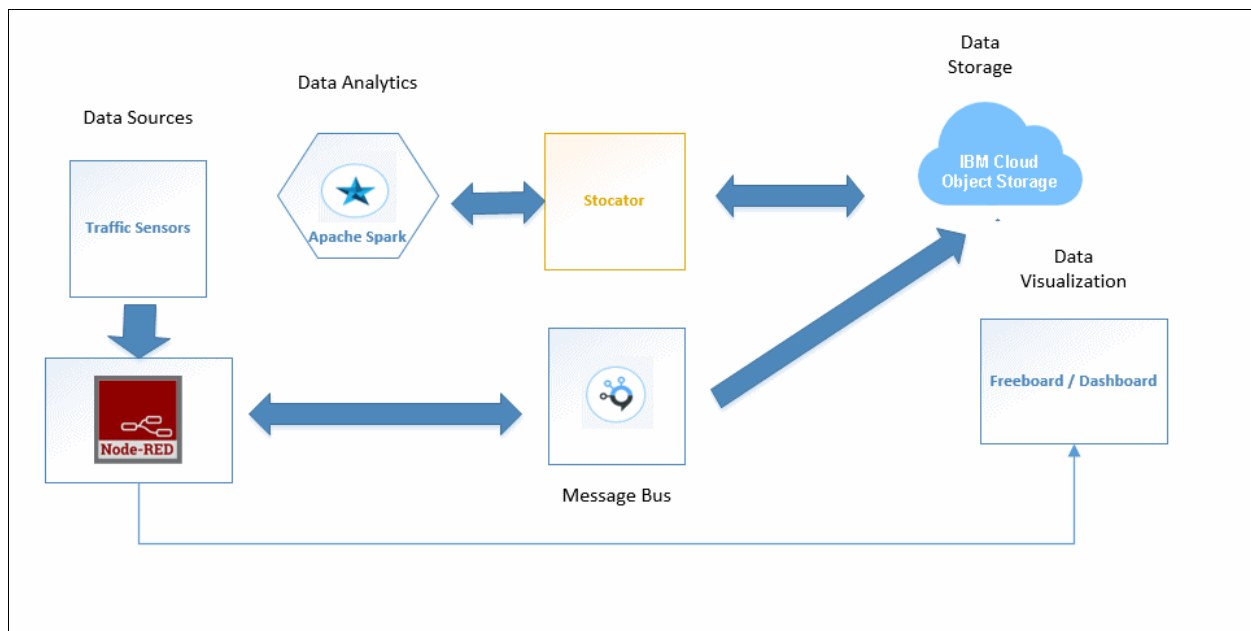


Figure 4-11 Keeping the traffic flowing - IBM COS solution

Table 4-1 describes the solution components.

Table 4-1 The solution components:

Component	Purpose
Node-RED	Node-RED is a visual programming tool for wiring the Internet of Things (IoT) devices. The IoT service allows you to register and connect various types of devices by using MQTT.
Message Hub	A messaging queue to pass real-time data to IBM Cloud Object Store.
Stocator (IBM SPARK S3 connector)	This is considered a part of the Message Hub in terms of functionality.
Cloud Object Storage on Bluemix	Stores IoT data in cloud object storage on the Bluemix platform.
Apache Spark	Apache Spark helps to increase in-memory processing speed. Optimizes predicate pushdown.
Dashboard	Visualizes information to enable proactive response to predicted events such as traffic congestion.

Benefits

The solution has the following benefits:

- ▶ Inefficiencies in manual analysis:
 - Predictive analytics based on historical data stored in IBM COS ensure that traffic flow can be modeled more accurately and consumed more quickly.
- ▶ Storing large amounts of sensor data:
 - IBM COS enabled the use of a single centralized data store with the required resilience features and 'limitless' (from a customer perspective) storage capacity.
 - Use of industry standard Object APIs meant that data was accessible from anywhere as a URL.

Key point: IBM COS combined with analytics can provide to government entities the solutions to better predict traffic flows.

4.2.6 Example 6: Telecommunications industry

The telecommunications industry is responsible for managing large phone and data networks around the world. The breadth of services required by users and businesses means that significant infrastructure is required to support their business operations.

Mobile content and voicemail backup using IBM Cloud Object Storage

A telecommunications provider has millions of users and mobile consumers who generate billions of photos and videos and use the carrier's voicemail services, all of which need to be stored in some way.

This carrier was looking for a way to cost effectively store that data and mobile device backups as a new service for its subscribers.

Business problems

This large mobile provider was challenged to manage huge and massive real-time data around the following areas:

- ▶ Scalability:
 - Difficulty in safely and securely storing large amount of unstructured data.
 - Managing the complexity of growing without distribution using legacy storage platforms.
- ▶ Security:
 - Dealing with data encryption requirements borne from regulatory authorities and risk of litigation.
- ▶ Availability:
 - No tolerance to downtime, which might impact millions of users and lead to loss of reputation and customer base.
 - Data access required anywhere and anytime: recall from cold storage was not an option.
- ▶ Economics:
 - The storage solution needs to solve all of these challenges at a lower cost than traditional block and file storage platforms.

Solution

The IBM Cloud solution is based on an S3-compliant phone backup application, which protects cell phone content by copying data to IBM Cloud Object Storage.

The carrier's phone application performs regular backup recovery of mobile content, such as image, file, photos, video, documents, and pictures and available to users at any time. Its highly scalable, secure, and available 24x7 to customers when needed.

Figure 4-12 shows the environment after the solution was implemented.

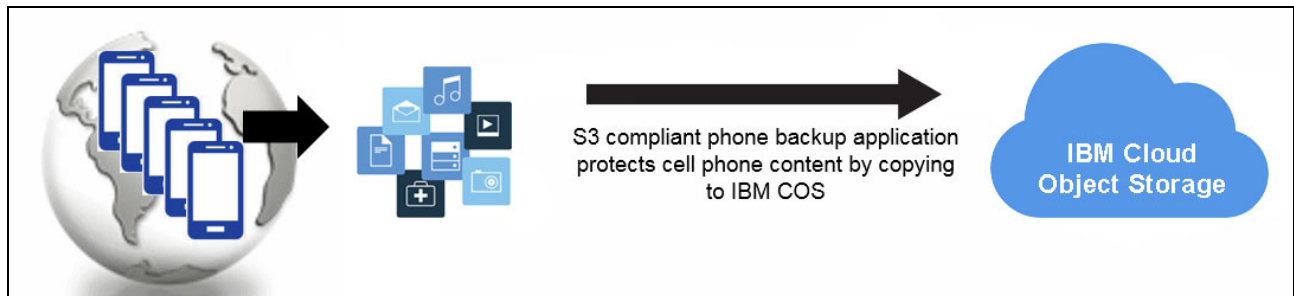


Figure 4-12 IBM COS enabling mobile device backup

Benefits

The solution has the following benefits:

- ▶ Scalability:
 - Tens of petabytes serving millions of mobile users; service is now expanding to deliver enterprise services for mail and storage on demand.
- ▶ Security:
 - Main driver and is a requirement for the customer.

- ▶ Availability:
 - 24x7 availability.
- ▶ Economics:
 - Reliable 80% savings means that the carrier can expand its business profitably.

Key point: IBM Cloud Object Storage is built for cloud-scale and is able to address any unstructured data storage (from petabyte to exabyte with reliability, security, availability, and disaster recovery without replication) making IBM COS extremely durable at scale.

4.2.7 Example 7: Insurance industry; cognitive damage assessment

In order to handle home insurance claims, insurance companies need to make a damage assessment first. In most cases this is a manual process, where an inspector is sent to the house for which the claim was made to evaluate the damage.

Cognitive damage assessment using the IBM Watson Visual Recognition service and IBM Cloud Object Storage

An insurance company was looking for a way to use commercial drone aircrafts to streamline the process for handling home insurance claims, especially for houses in rural areas or areas where it is difficult to send an inspector.

Business problems

This company had to grapple with a number of issues and goals:

- ▶ There are inefficiencies in manual damage assessment, especially in rural areas or areas where it is difficult or not cost effective to send an inspector.
- ▶ In the case of a natural or human-made disaster, it might not be possible for an insurance inspector to safely reach a scene.
- ▶ The assessments need to be documented promptly to process the claims as soon as possible. Any delays could cause customer dissatisfaction.

Solution

In this use case, the insurance company analyzes photographs and imagery collected by small, commercial drone aircraft to streamline the process for handling home insurance claims.

Rather than sending an inspector up on top of the home to inspect the structure, a small drone aircraft flies over the home in specific patterns, capturing imagery that will then be used in analysis. That imagery is then analyzed for damage using the IBM Watson Visual Recognition service, which has been trained to recognize the presence of hail damage on shingle roofs.

Figure 4-13 on page 73 shows a sample imagery that is analyzed for damage by the IBM Watson Visual Recognition service.



Figure 4-13 Sample imagery

In parallel, the images are also run through a photogrammetry process to create geo-referenced 3D models and 3D point cloud representations of the structure. These models are then further analyzed to calculate measurements and structural diagrams of the roof (including square footage, and roof pitch), which are then used to calculate the replacement value of the insurance claim.

Where, why, and how does IBM Cloud Object Storage fit into the solution?

To answer that question, consider these numbers:

- ▶ The current generation of consumer drones capture 12 megapixel images, averaging in size about 5 MB per image.
- ▶ The latest emerging generation of drones captures 20 megapixel images, averaging about 20 MB per image.
- ▶ A scan of each property requires anywhere from 50 to 100 images.

With image data alone, not including any generated 3D modelling data, this means a range of 250 MB to 2 GB of data per property. Generated 3D modelling information also adds anywhere from 100 to 500 MB per property, depending on the level of complexity.

By that logic, a single scan of a single property can range from 350 MB to 2.5 GB, and can sometimes be even larger when scanning much larger properties. If you consider that a region could have anywhere from tens of thousands to millions of insured properties, these numbers grow dramatically. If you think about 4D usage, or 3D scanning a property over time for historical records, these numbers grow even larger.

This is where IBM Cloud Object Storage solves the need for massively scalable, and highly-available storage. As new media is captured and models are generated, they are saved into Cloud Object Storage, which automatically scales based on the solution needs.

This solution uses many of the offerings within the IBM Bluemix platform; each of these components was specifically chosen because of specific needs for the scenario: IBM Cloud Object Storage, IBM Watson, Cloud Foundry runtimes, bare metal servers, and IBM Cloudant®.

A real solution: This is actually a real solution that is being used by a few insurance companies. The solution was implemented with several teams coming together within IBM to share ideas and this technology came as a result of such collaboration. IBM Cloud Object Storage is well positioned for this solution because of the sheer amount of data being collected and analyzed.

How do all of the pieces of this solution work together on Bluemix?

Figure 4-14 shows the architecture of the solution.

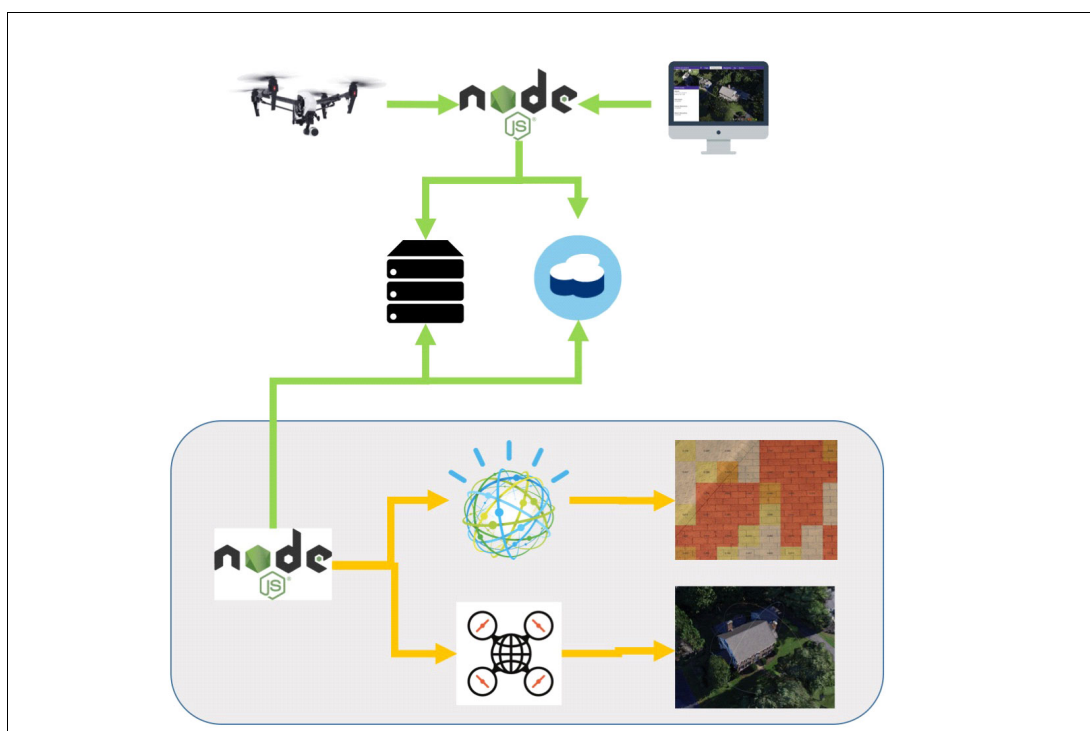


Figure 4-14 Architecture of the solution

All user interactions within the system take place through a web browser. This is where users can create new records, upload imagery for analysis, and view analyzed properties. Serving the web user interface is a Node.js CloudFoundry runtime on Bluemix. CloudFoundry runtimes on Bluemix allow for rapid development, ease of deployment, and scalability.

After the user uploads images through the web interface, the Node.js application saves records of all media in IBM Cloudant (the IBM NoSQL database as a service). It also saves all image media to the Cloud Object Storage bin. After all of the media is uploaded for a property, that property is ready for analysis.

The media analysis takes place on a bare metal server on Bluemix. Bare metal servers are privately provisioned physical hardware running within the cloud infrastructure, essentially on-demand dedicated cloud hardware. On the bare metal server is a separate Node.js server which monitors for new processing jobs. After a new processing job is created, the server does two operations in parallel: the photogrammetry operation and image analysis using the

Watson Visual Recognition service. The photogrammetry operation is based on a third-party solution. For the Watson Visual Recognition analysis, the Node.js service first chops each image into much smaller tiles, and then each tile is analyzed by the Watson service individually for the presence of damage. This approach allows the solution to determine locality of damage within the larger image. After all Watson and photogrammetry analysis is complete, all generated models are saved back into Cloud Object Storage, and the analysis results are made available back in the web interface. A bare metal server is used in this scenario, because it allows the solution to have dedicated high performance computing hardware tailored specifically for the photogrammetry operations, which are very computationally expensive.

Benefits

This solution has a lot of benefits and is transformative for the industry for many reasons, most obviously these reasons:

- ▶ It makes the process of inspecting a home safer: For example, nobody has to climb on the roof of the home.
- ▶ It makes the claim less prone to fraudulent activity: The Watson Visual Recognition service can determine whether there are indications of damage automatically, and can calculate square footage to ensure the claim matches the size of the insured property.
- ▶ It makes the claim faster to process: inspections process can be handled the same day, where traditionally, they would have taken weeks to process.

Key point: All of these components come together to create a solution that streamlines business process, increases safety, improves productivity, which in turn improves customer satisfaction. At the heart of it, IBM Cloud Object Storage provides the scalable storage infrastructure that makes the capture, analysis, and persistence of all of this information possible in the cloud and distributed computing environment.

However, do not be confined to think that only cases like this are where you could possibly leverage IBM Cloud Object Storage; the best approach is to think of the Cloud Object Storage solution as a massively scalable and highly available repository for any kind of unstructured file or data storage.

4.3 Typical use cases

The purpose of this section is to discuss typical use cases for IBM COS solutions. A use case is a specific purpose application of IBM COS and is not associated to a specific industry.

4.3.1 Use case 1: Backup repository

A *backup repository* is a storage system configured to store backup data. Traditionally disk, file, and tape systems are used for this purpose. Managing these involves capacity management, managing space reclamation, and data replication.

Significant SAN/IP bandwidth is typically required due to replication requirements inherent to most traditional storage systems. In some cases, manual vaulting using couriers and physical tape shipping is still used for offsite data storage.

Sample architecture 1: Backup software to IBM COS

Figure 4-15 shows a backup server.

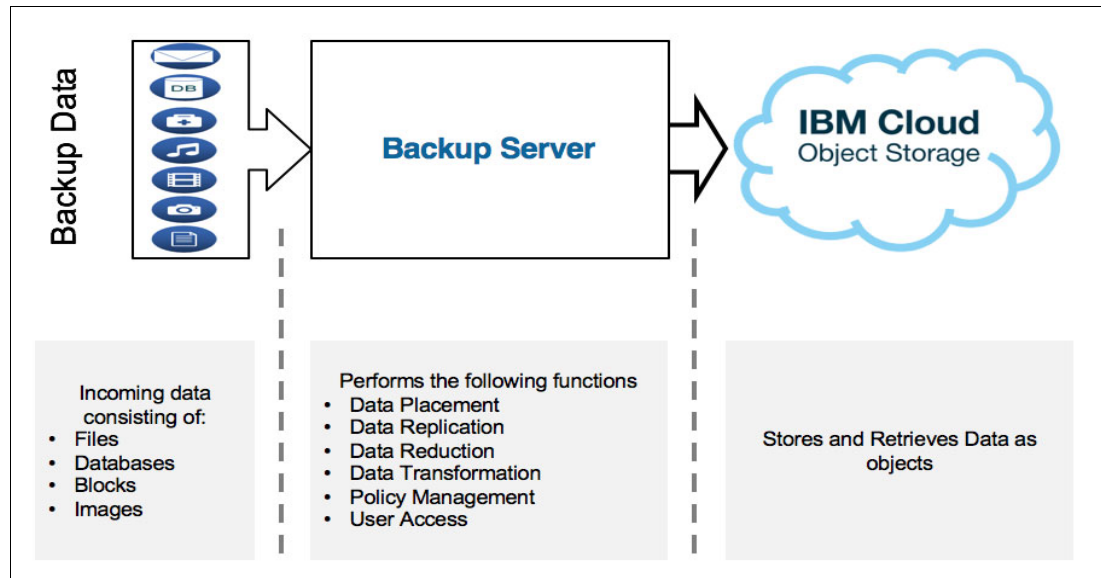


Figure 4-15 Backup server using IBM COS as a back end

In this use case, a backup server ingests data and then processes it for movement to IBM COS. Depending on the backup product in question, the architecture varies between products because some can store only a tertiary copy of the data in the cloud, some can store the primary copy, and some can use this as the only copy given the resilience of IBM COS.

An important note is that network bandwidth to IBM COS and also IBM COS system load can impact performance in this type of architecture. Therefore, service level agreements must be examined to determine whether this solution is appropriate for some or all of the data.

Tip: Disk staging in some solutions can lessen the network impact for backup.

Sample architecture 2: Direct application to IBM COS backup

Figure 4-16 shows IBM Cloud Object Storage ingesting backups from applications.

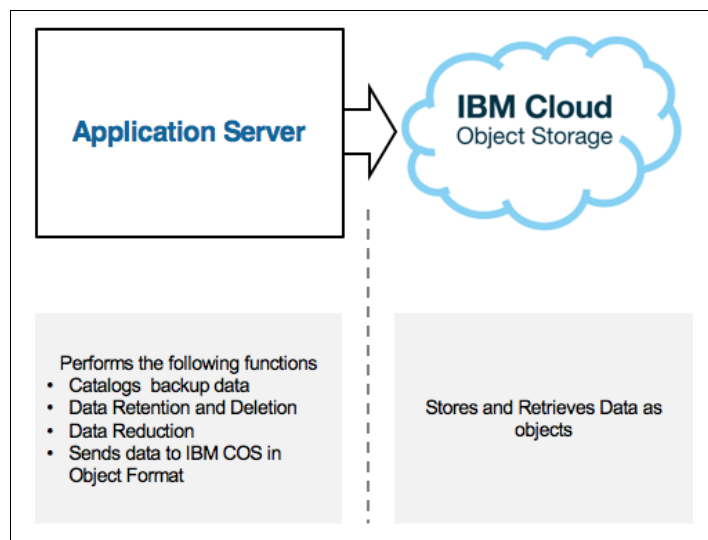


Figure 4-16 IBM COS being ingesting backups from application directly

In this use case, the application has native object APIs and performs the backups directly using IBM COS, bypassing the need to run and maintain a backup server. This means that software and hardware costs associated with this infrastructure are removed, however it does place responsibility on the application administrators to manage and delete their data. A number of enterprise applications, such as Oracle, have this functionality.

Why IBM COS

IBM COS allows enterprises to move away from managing large costly storage arrays and tape silos and instead deploy storage-thin backup servers that store the data in IBM COS, or backup to IBM COS directly.

The resilience features allow businesses to move away from replication (reducing storage requirements), or use IBM COS as a low cost tertiary copy of the data depending on the business requirements and backup software capability.

The inherent “endless” pool of storage means that capacity management is no longer an arduous and time-consuming task; the data deduplication capability of most major backup vendors reduces the physical capacity requirements to a fraction of the fully hydrated data set.

Managing tape reclamation, rotation, offsiteing, and transport all become redundant in an object architecture, freeing up backup administrators to perform value-added work instead of managing and monitoring these resource-intensive tasks.

4.3.2 Use case 2: Internet of Things (IoT)

The *Internet of Things (IoT)* is a system of interrelated computing devices, mechanical and digital machines, objects, and people, which are each provided with a unique identifier and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Different types of IoT device data are transformed through IoT platforms into long-term data storage, live dashboards, and real-time presentations.

Sample architecture

Different devices, sensors, mobile, applications, web, and social networks (system of engagement) produce different events that pass to the IoT platform. Data connection and secure transformation is processed through a connection and pipeline framework like REST API, MQTT, and Kafka, which transfer data through Spark to Cloud Object Storage (COS).

Figure 4-17 shows IoT with IBM COS.

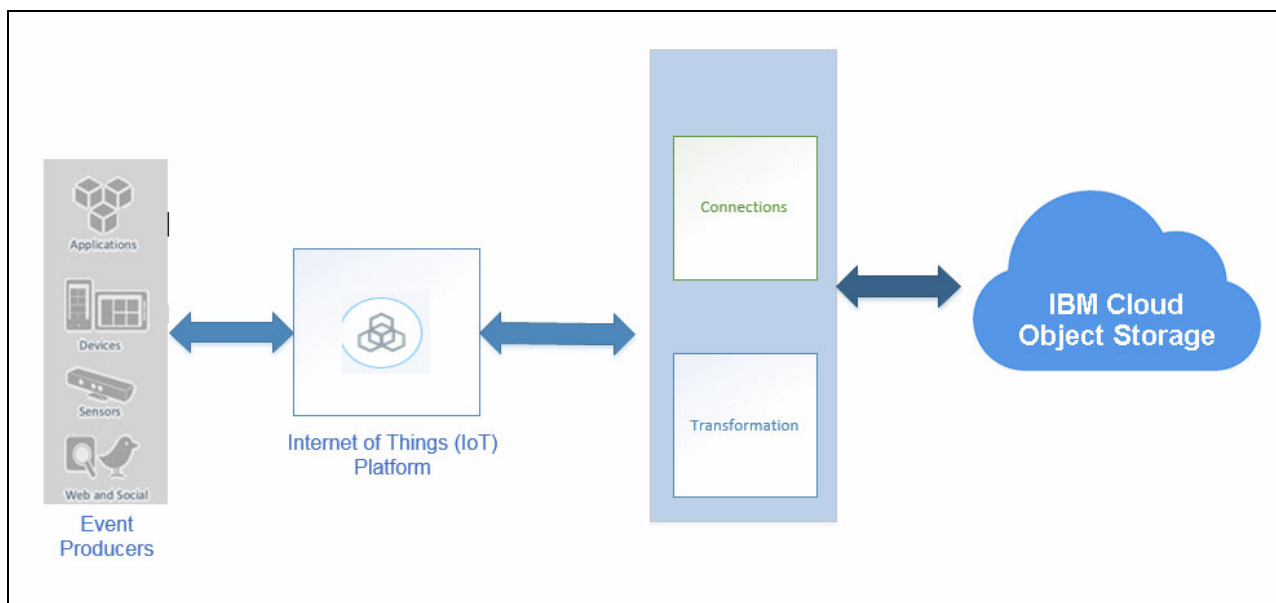


Figure 4-17 Internet of Things (IoT) with IBM COS

These systems of engagement (SOE) are connected to back-end systems of record (SOR) where real-time data is stored, retrieved and archived through IBM Cloud Object Storage.

These solutions help users in various industries, home, and cities to monitor activities and control devices by using the IoT platform.

Why IBM COS

IBM COS is well suited for IoT solutions because it is capable of storing IoT content repository, access real-time data, large image files, and video.

IBM COS stored data is accessible anywhere, anytime for IoT devices, applications, and front-end consumers.

IBM COS works with on-premises and public cloud data, visualizes data lineage, and feeds applications with data while integrating with IoT platforms.

4.3.3 Use case 3: Active archive

Active archive is a data store that is designed to transparently store important, but infrequently accessed data. Some examples of data requiring these systems are legal documentation, finance documents, and others.

Because of these two seemingly conflicting requirements, this data store must be cost effective, but resilient because data might be required at any point in time, in many cases months or years after ingest. However the data must be near and quickly accessible unlike deep archive storage systems.

Traditional active archive systems are constructed of a mix of inexpensive disk systems and tape, with replication of data between sites being done by the active archive application or the underlying storage layer.

Figure 4-18 shows an established active archive system.

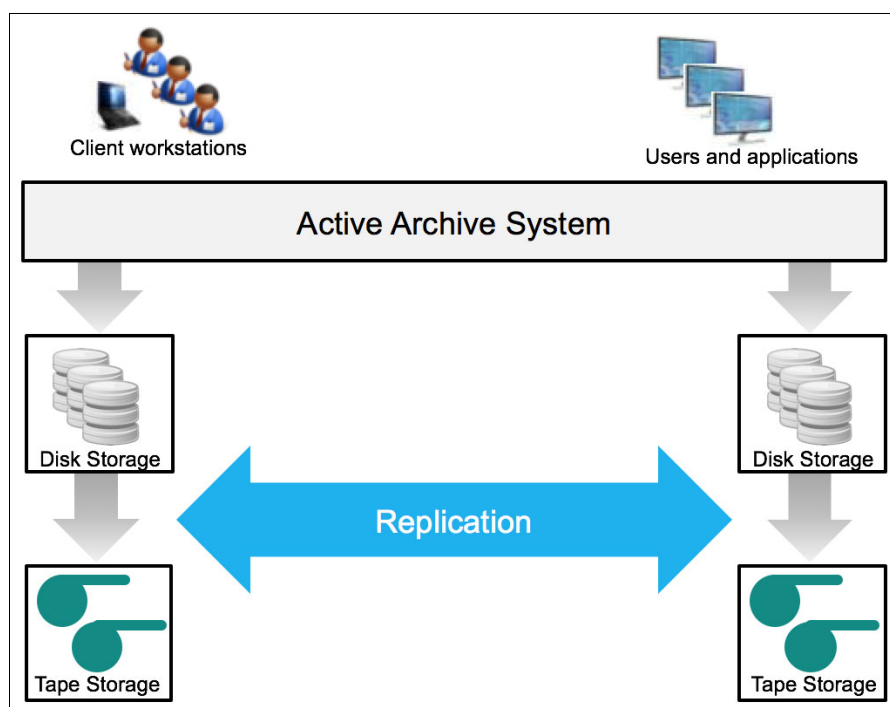


Figure 4-18 Legacy active archive system

The requirements for resilience, availability, and low cost are key to this use case and typically mandate this replicated approach. The need for replication results in at least two data copies even before backup copies are counted. Data replication must be carefully monitored because failure to do so reduces system resiliency.

Where tape is a part of the solution, architects must ensure that adequate disk staging space for initial ingest and recall is available, and sufficient tape drives to service these must be included. In addition, the application must have some capability to deal with extended recall times that a busy tape system might experience.

Sample architecture

An IBM COS based active archive system at a high-level looks like the diagram in Figure 4-19.

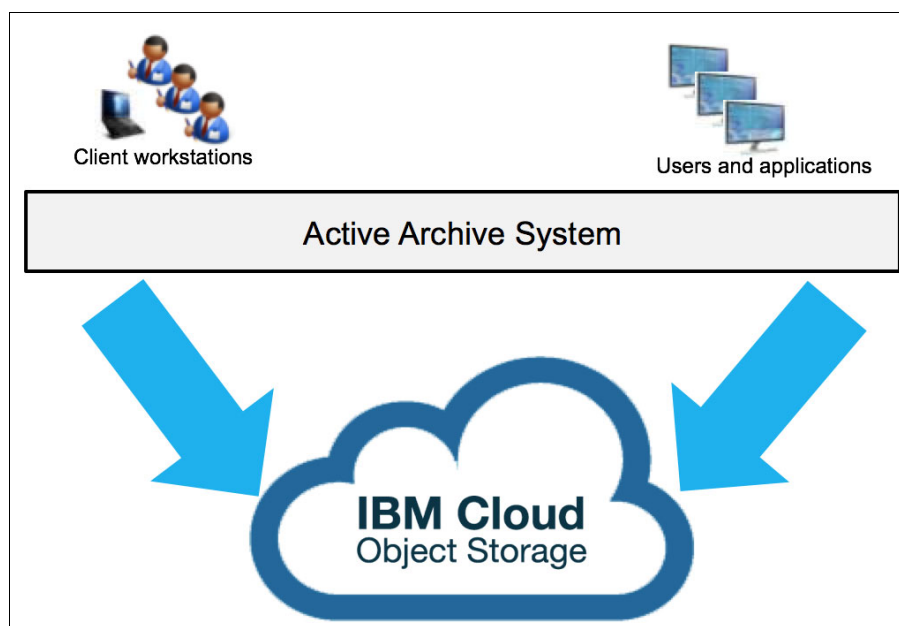


Figure 4-19 IBM COS enabled active archive system

In an IBM COS architecture, the data tiers are either reduced or eliminated so data can be directly pushed or retrieved from the IBM COS repository. Site loss resilience is achieved through geographically dispersed erasure coding rather than replication.

Why IBM COS

IBM COS, serving as an archive repository back end, solves a number of key issues present in traditional archive architecture.

The geographically dispersed nature IBM COS provides two key benefits:

- ▶ Site fault loss resilience without multiple copies greatly reduces the capacity requirements and removes the need to monitor replication.
- ▶ Removal of the primary and secondary (read only) copy concept means a disaster involving a loss of a site, or loss of access to a site, does not require any traditional high availability processes (for example, failover to secondary, promote secondary to read/write access, reverse replication, demote primary to read only, or the reverse on fail back).

The flattening of the data tiers allows for simplified administration and uniform file access, which are expectations of users, meaning the data access speed is the same regardless of file age or type.

The need to manage a large landing space is typically removed or drastically reduced, because the data can be read streamed directly from IBM COS to the user, who can then view or edit the file locally and then upload through the Active Archive system.

Capacity limitations present in traditional storage systems are largely removed as applications have access to a “limitless” pool of storage. So, while capacity usage must be

measured for fiscal planning, management and design of the storage infrastructure are no longer a burden for the application, system administrators, or architects.

The improved operational simplicity, reduced capacity, and increased resilience typically result in a lower total cost of ownership for an active archive system when compared with traditional on-premises or replication-based cloud-based deployments.

4.3.4 Use case 4: Enterprise file services

File access protocols have existed for decades, with initial implementations being constrained to local machine access. As enterprise dependence on IT grew, the need for centralized file services was apparent and a number of vendors released various devices to satisfy these requirements. Today, the most common ways to expose shared file data are through NFS protocol for UNIX and Linux based systems, CIFS/SMB for Windows based systems. Recently, additional services composed of web portals and local clients providing file sync and share services have also become ubiquitous.

At a high-level, file access systems structure data into directories, subdirectories, and files that are shared among many users, workstations, servers, security based on user and group concepts. Typically a metadata repository stores the relationship between these components.

Array resiliency is achieved through RAID schemes on the underlying block storage, and multi-site resiliency is achieved through replication.

Figure 4-20 shows a silo approach.

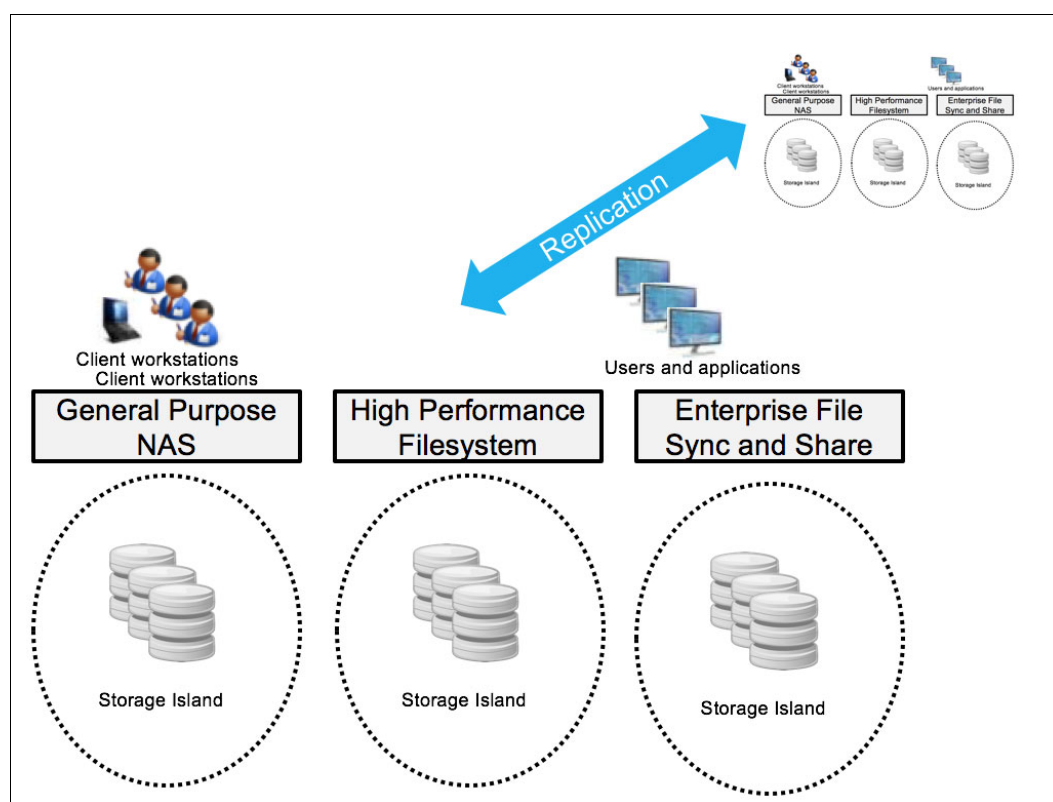


Figure 4-20 Legacy Enterprise File Services: A silo approach

Although NAS systems in particular are mature and well understood, the underlying architecture of these systems poses a number of challenges because of the immense amounts of data being generated by users, applications, and sensors.

The controller-enclosure scaling approach means that rigid limits exist regarding the number of NAS heads a cluster can contain, therefore limiting the capacity that is available in a single name space, creating a number of discrete storage islands within enterprises. This is the exact problem that modern storage systems were designed to solve.

The dependency on metadata access for file operations can place an enormous load on this component and it can often be a choke point on large file-based systems, especially if they contain millions to billions of objects.

Depending on RAID schemes for large scale data protection has become cumbersome, especially on large capacity spinning disk drives. This means that more disks are required for data protection to ensure resiliency and require significant administrator overhead to manage because loss of one or two disks in the same RAID stripe can result in data loss.

Multi-site data resilience requires data replication, using up significant network bandwidth and needing at least 2x the capacity of the actual data being stored.

These factors all add up to power, cooling, and operational costs that enterprises are looking to reduce.

An important aspect to note is that IBM COS is not a NAS or file sync and share solution. Although many workloads, such as those that are mentioned in this publication (active archive, content repository, backup repository, etc.) can directly replace NAS solutions using direct IBM COS S3 API access, a number of users and applications are still not familiar or compatible with these APIs. In many cases organizations are locked into access through NAS protocols by legacy applications. For these instances, a number of gateway solutions can provide file services while using a common object back end.

At a high-level, three use cases for file services are available:

- ▶ General purpose NAS. These solutions provide user file and application access through standard protocols like SMB/CIFS and NFS.
- ▶ High-performance computing. These systems provide extreme performance and scalability using parallelism and very high-speed network interconnect over protocols implicitly designed for high performance such as IBM Spectrum Scale.
- ▶ File sync and share. These systems provide user file sharing capabilities using web interfaces, agent software on workstations and mobile devices, and sometimes tie in with NAS services.

Because workloads can vary significantly among these use cases, no single solution exists for all workloads. This section provides a high-level overview of gateways working with IBM COS and their “best fit” characteristics at the time of publication. An important note is that these are not mutually exclusive and enterprises can, and do, use a combination of these solutions to fulfill their requirements.

Table 4-2 lists several products that are currently tested and working with IBM COS. Because of the rapid change in this industry segment, the items in the table will change. Also note that other products that are capable of using the S3 API should be able to use IBM COS.

Table 4-2 IBM COS gateway options

Gateway product	Description
IBM Spectrum Scale	IBM Spectrum Scale is a software-defined parallel file system with a rich HPC heritage. It can expose native IBM GPFS™ protocol, and also CIFS/SMB, NFS, and OpenStack Swift. IBM Spectrum Scale can use IBM COS as an external storage pool and move inactive data to this tier.
IBM Aspera®	IBM Aspera is a software suite designed for high performance data transfer achieved through a protocol stack that replaces TCP with the IBM FASP® protocol. IBM Aspera Direct-to-Cloud provides the capability to expose a file system interface to users and applications through a web interface or through client software installed on the server/workstation or mobile device and can provide file sync and share capability. Data ingested through Aspera can be read as an object because one file is uploaded as one object to IBM COS.
Avere FXT Filer	Avere consists of a hardware or virtual offering that exposes CIFS/NFS for users and applications. The architecture is caching so active, recent, and hot data is cached locally while the entire data set is kept in IBM COS. Data reduction in the form of compression is available and the customer use cases include I/O intensive workloads such as rendering and transcoding.
Nasuni Filer	Nasuni filer is a software or hardware solution that provides general purpose NAS capability through SMB/CIFS and NFS. The architecture is caching so active, recent, and hot data is cached locally while the entire data set is kept in IBM COS. The suite provides file sync and share capability and also a global name space. Data reduction in the form of deduplication and compression is included.
Panzura	Panzura is a software or hardware solution that provides general purpose NAS capability through SMB/CIFS and NFS. The architecture is caching so active, recent, and hot data is cached locally while the entire data set is kept in IBM COS. Data reduction in the form of deduplication and compression is included.
Ctera	Ctera is a hardware and software solution focused on file sync and share capability. The architecture caches the data set onsite with the master copy retained in IBM COS. Client software can be installed on workstations or mobile devices and data can also be exposed through SMB/CIFS and NFS. Data reduction in the form of deduplication and compression is included.
Storage Made Easy (SME)	SME provides File Sync, a software based file sync and share solution with mobile and workstation clients and a focus on inter-cloud compatibility including IBM COS.
CloudBerry Explorer	CloudBerry Explorer is a software-based object storage client that allows users to directly interact with an IBM COS. Data ingested through CloudBerry Explorer can be read as an object because one file is uploaded as one object to IBM COS.
Seven10 Storfirst	Seven10 Storfirst is a software based SMB/CIFS and NFS gateway offering with capability to talk to IBM COS as well as legacy tape and VTLs.

Sample architecture

An IBM COS based architecture for file services can include one or more of these gateway offerings because any given enterprise can have a mixture of requirements all of which can use a common IBM COS back end.

Figure 4-21 shows a consolidated approach.

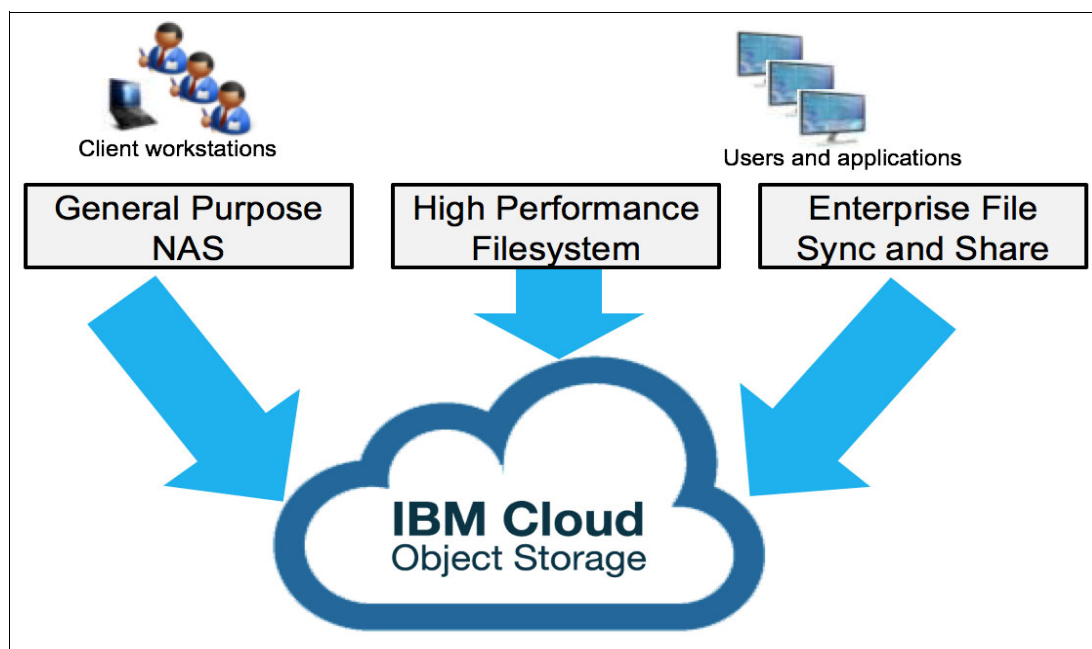


Figure 4-21 IBM COS enabled Enterprise File Services: A consolidated approach

Why IBM COS

IBM COS acting as a single unified back end for file based access, offloads the enterprise from a number of management and operational costs by removing the legacy storage silo approach.

The geographically dispersed erasure coding data protection, unique to IBM COS, means that data is instantly available at multiple sites, eliminating RAID management, replication management, and the additional capacity this requires.

Depending on the solution, the metadata for the file access may be stored in the cloud, replicated between gateway devices, or stored on the user's local host in case of agent-based solutions. This approach preserves the security constructs that are necessary for file-based access, meaning no recording of applications is necessary.

The best fit solution can be used for each workload so that enterprises are not locked into one NAS solution; the economies of scale are achieved at the underlying IBM COS layer as opposed to the NAS heads, which are tied to deploying very large on-premises NAS systems with hundreds to thousands of disk spindles dedicated to one storage silo.

Key point: Workloads tied to legacy NAS protocols can be serviced by IBM COS when integrated with a cloud gateway solution. This approach greatly reduces the physical on-premises footprint by leveraging the IBM COS economies of scale.

4.3.5 Use case 5: Analytics and cognitive systems

Cognitive systems learn and reason based on interactions with humans and experiences with the environment. They do not give predetermined responses but make probabilistic prediction based on these learnings. Cognitive systems generate hypotheses, reasoned arguments, and recommendations, and can scale to keep pace with the complexity and unpredictability of information in the modern world.

Cognitive systems can understand unstructured information such as the imagery, natural language, and sounds in books, email, tweets, journals, blogs, images, sound, and videos. They unlock meaning because they can reason through it, giving us new contexts to weigh and consider. Cognitive systems also learn continually, honing our own expertise so we can immediately take more informed actions.

As organizations begin to embed data and analytics into every business process and every customer experience, they are finding that their existing IT infrastructure was not designed to drive performance for systems. Cognitive workloads will require a re-imagined IT infrastructure, one that can synthesize massive amounts of data quickly, accelerate analytics, and be available anytime, anywhere with trust.

A cognitive solution explains cognitive analytics, operations, and engagement, and how it interacts with customers, users, and ecosystem partners.

These systems need to store large quantities of unstructured data for the ongoing analysis required to build their understanding and cognitive capabilities.

Figure 4-22 shows an analytics and cognitive system.

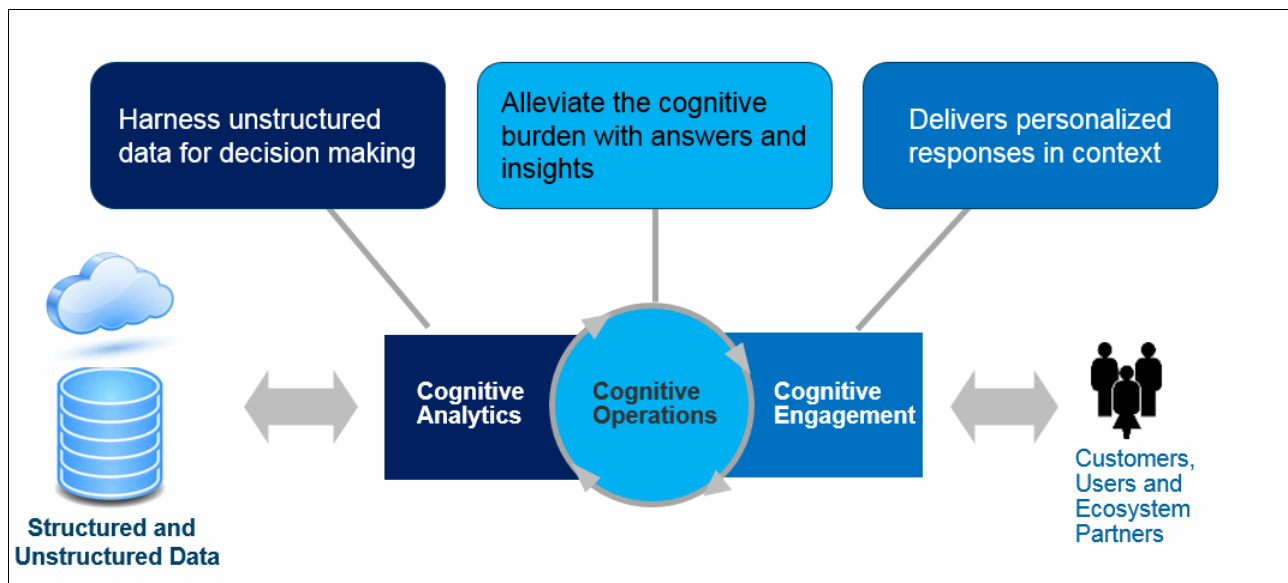


Figure 4-22 Analytics and Cognitive System

Sample architecture

Cognitive Solution helps to build industry leading solution that addresses a faster time to market, a simple approach, flexibility, and delivers accurate, evidence-based information.

With analytics, you get key insights from data, but with cognitive systems, you can turn those key insights into knowledge.

Figure 4-23 shows a system with IBM COS.

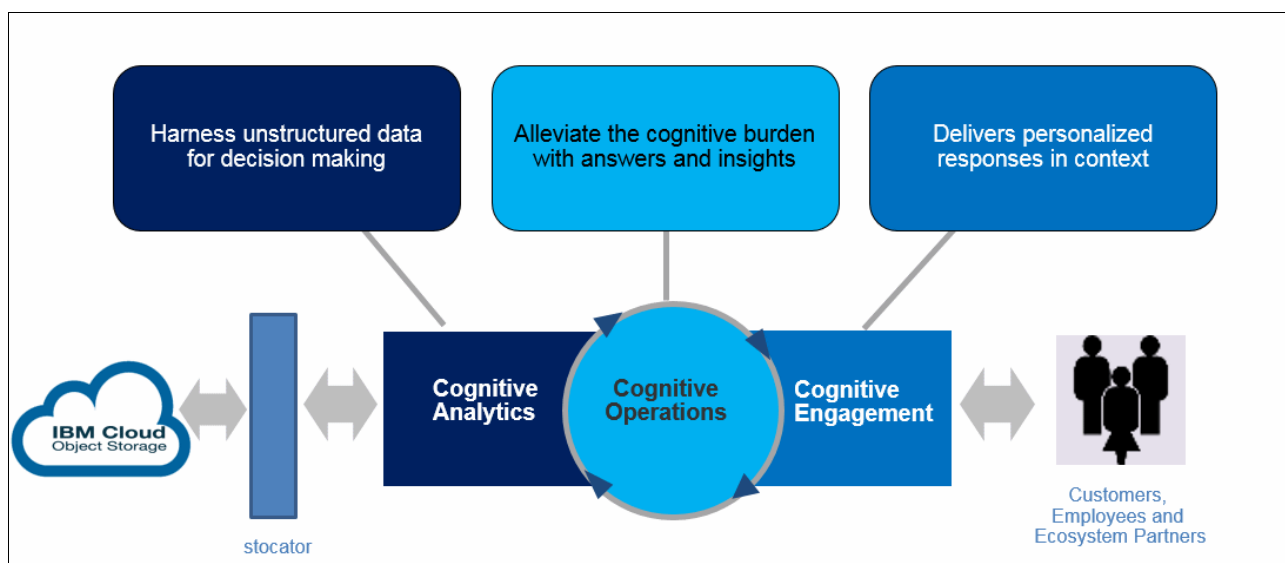


Figure 4-23 Analytics and cognitive system with IBM COS solution

Different applications have evolved to store very large volumes of unstructured data. More and more machine data is being created and needs to be stored and protected for further analysis and for potential cognitive uses. Software applications that create this type of data work with IBM Cloud Object Storage to deliver carrier grade system reliability and data security with the ability to scale to petabytes and larger in a cost-effective way.

Cognitive systems store and process large volumes of data using IBM COS and integrate with other analytics services.

Why IBM COS

IBM COS is well suited for cognitive infrastructure solutions because it can store large amounts of historical data, archive data, and backup data without the need for many discrete storage solutions for these data types.

IBM COS works with on-premises and public cloud data, with cognitive solution that is deployed on a cognitive infrastructure.

IBM COS benefits are as follows:

- ▶ Is cost effective.
- ▶ Scales to 100's of petabytes, exabytes, and beyond.
- ▶ Stores a single digital copy of digital assets; no content replication is necessary.
- ▶ Efficiently manages content.
- ▶ Has carrier grade system reliability.

4.3.6 Use case 6: Content repository

Content repository systems manage large catalogs of digital content such as document, images, videos, and more. These systems differ from active archive in that they provide comprehensive search and file management functionalities that include, but are not limited to, access control, checking in and checking out documents, file locking, and file tiering.

These systems typically store a mix of data types and need to provide resilient services to their users who can use the data in these systems for their daily workflow. The locking features ensure that duplicate or forked versions are not unintentionally created, or data lost when users accidentally overwrite by saving old files over an updated version.

Figure 4-24 shows a content repository system.

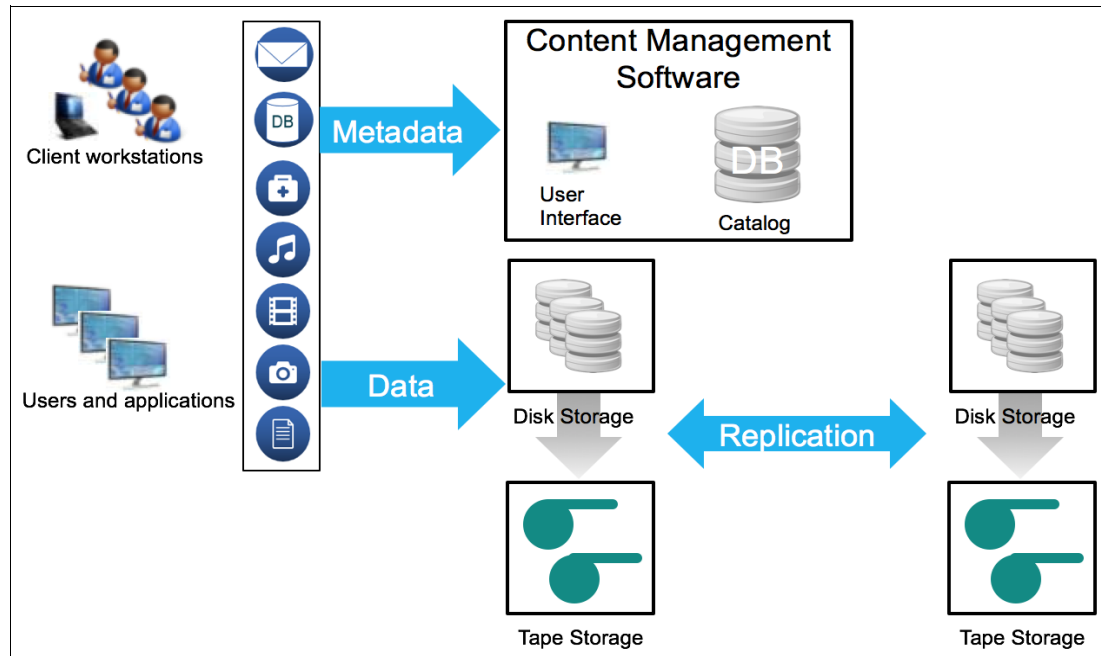


Figure 4-24 Legacy content repository system

Resiliency is typically provided through replication, and the data tiering can be managed by the content repository system, or it might sit on top of an active archive system performing the underlying data protection.

Content repository systems are typically constructed from a mix of inexpensive disk and tape and are not performance- sensitive. However, user experience and productivity can be compromised by extended wait times if data needs to be recalled from cold storage.

Disaster recovery often includes traditional High availability (HA) procedures of promoting and demoting copies, and failing back is not possible until new data is replicated to original sites.

Sample architecture

Figure 4-25 shows a content repository solution constructed around IBM COS.

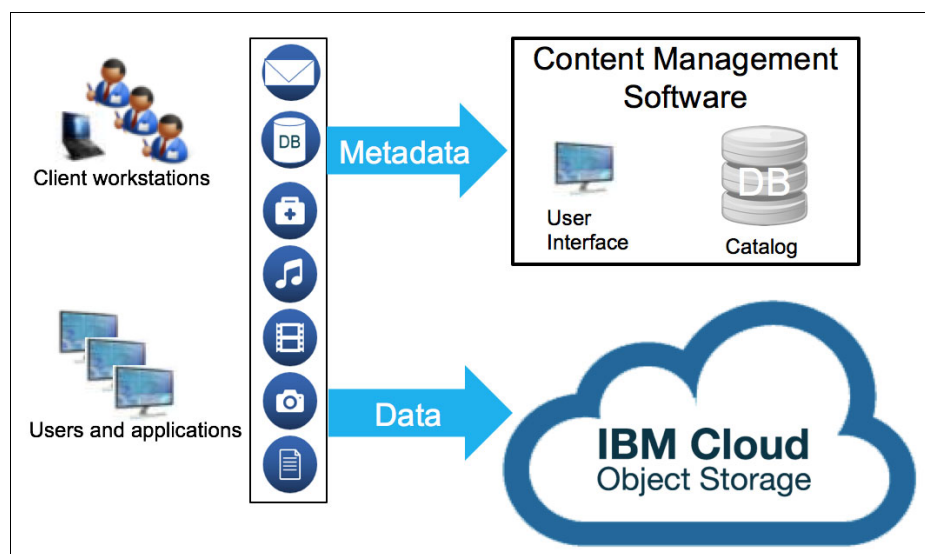


Figure 4-25 IBM COS enabled Content Repository systems

In this instance, the application sits on traditional storage that hosts the catalog of data browsed by users, while the data itself is held on IBM COS.

Why IBM COS

IBM COS, used as the back-end storage for a content repository and management system, provides a cost-effective solution that can solve a number of issues in traditional architectures.

Removal of cold storage ensures that users can access data instantly as opposed to waiting for extended recall times for old data.

System administrators benefit from the geographically dispersed nature of IBM COS because it removes the overheads of managing data replication and eliminates complex failover of the data store that is used by the repository.

4.3.7 Use case 7: Enterprise collaboration

Collaboration systems provide functionalities that allow teams of people to communicate and share content in a streamlined manner. These systems can encompass workflow management, file sync and share capabilities and social collaboration like instant messaging. This wide variety of applications shares a number of common needs:

- ▶ Resiliency
- ▶ Anywhere, anytime availability
- ▶ Instant content access

Traditionally these platforms are constructed from block and file storage platforms because of their instant access characteristics that are not possible with cold storage technologies like tape. More recently, many of these platforms are cloud-enabled and many can now use industry standard object interfaces for their data storage.

Figure 4-26 shows a legacy collaboration system.

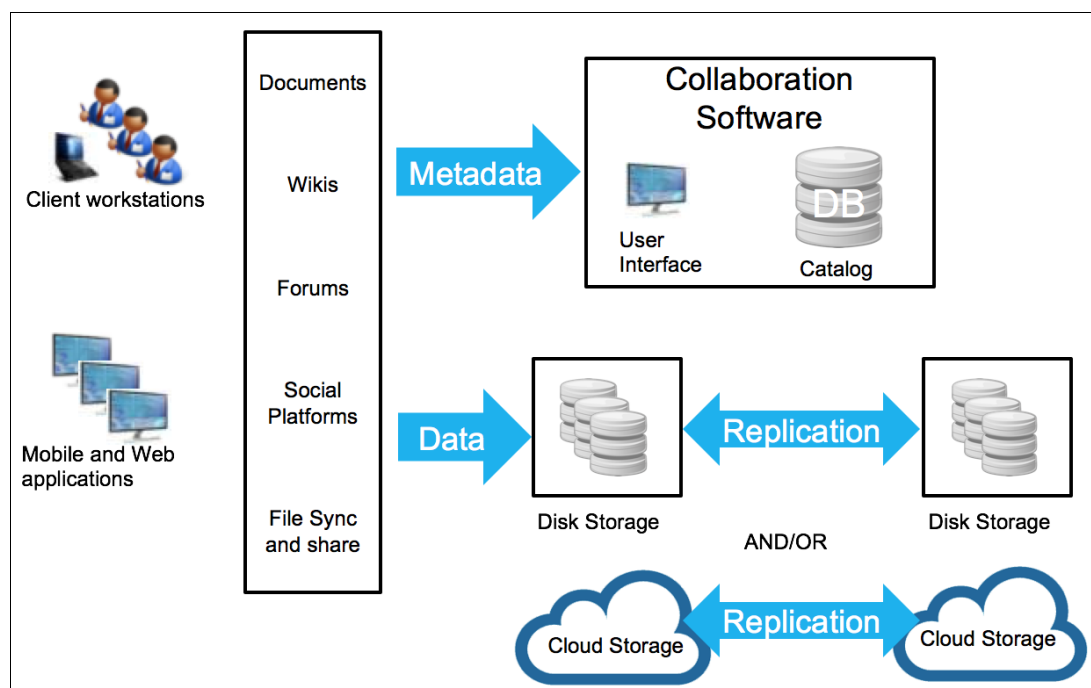


Figure 4-26 Legacy collaboration system

Resilience is typically provided through replication at the application layer or replication of the disk or cloud storage.

External access through mobile devices is provided through access applications sitting on top of file systems storing data.

Accessing the data from anywhere is achieved by URLs or application clients, pointing at the central data copy, with read access to tertiary copies in some cases.

Instant access is achieved by using disk or cloud storage for centrally accessed data. In the case of file sync and share solutions, selected sets of data are replicated to user workstations or mobile devices and synchronized back to the central data source upon update.

Sample architecture

Figure 4-27 illustrates an IBM COS based collaboration solution.

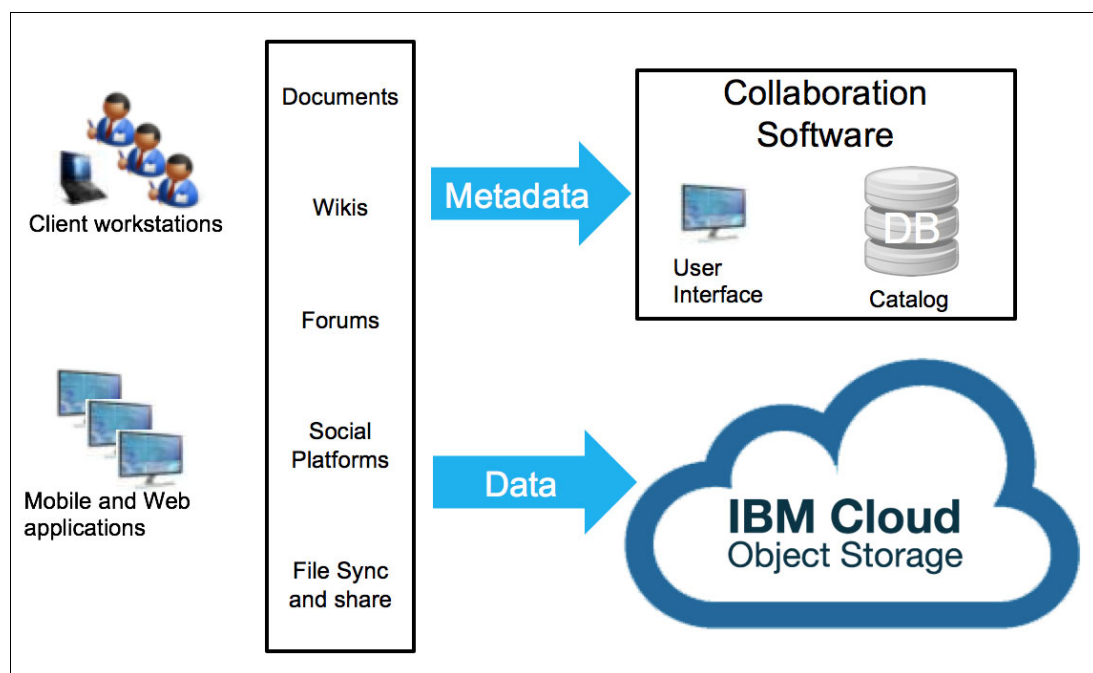


Figure 4-27 IBM COS enabled collaboration system

In this architecture, the data is stored on IBM COS instead of replicated storage. Catalogs and applications orchestrate the front-end user experience and data movements between the IBM COS back end, and user clients and agent applications.

Why IBM COS

The key benefits to IBM COS based collaboration solutions are that the geographically dispersed nature of the data answers the four key requirements of collaboration systems:

- ▶ Resiliency is no longer a concern, without the need for additional capacity, network and cost of traditional replication.
- ▶ Data locality. As data is spread across multiple locations, users are more likely to access at least some portion of their data locally, improving response times.
- ▶ Availability. System is more likely to be available because no data store failover procedures are necessary from the application or data storage perspective.
- ▶ Instant content access for applications is available via standard object APIs.

4.4 Integration with other IBM products

This section covers specific integration options with products in the IBM portfolio that are optimized to work with the IBM COS offering.

4.4.1 IBM Spectrum Protect

IBM Spectrum Protect is the eminent IBM enterprise backup and restore (EBR) tool. It is consistently in the Gartner magic quadrant and is considered a leader in the EBR Market. A key differentiator for IBM Spectrum Protect is that it performs a progressive incremental forever backup, which means that only a single full backup is required. From this point onward, all subsequent backups contain only incremental changes. This approach means that IBM Spectrum Protect solutions require less storage, less compute, and less network infrastructure when compared to other EBR solutions.

Product architecture

A typical IBM Spectrum Protect environment consists of one or more servers that either copy data to each other over the IP for disk-based solutions or perform tape-to-tape copies over storage area network (SAN). Tertiary copies of data are typically used for disaster recovery purposes and all data placement and movement is managed by the IBM Spectrum Protect server.

Figure 4-28 shows an IBM Spectrum Protect traditional environment with *electronic* vaulting.

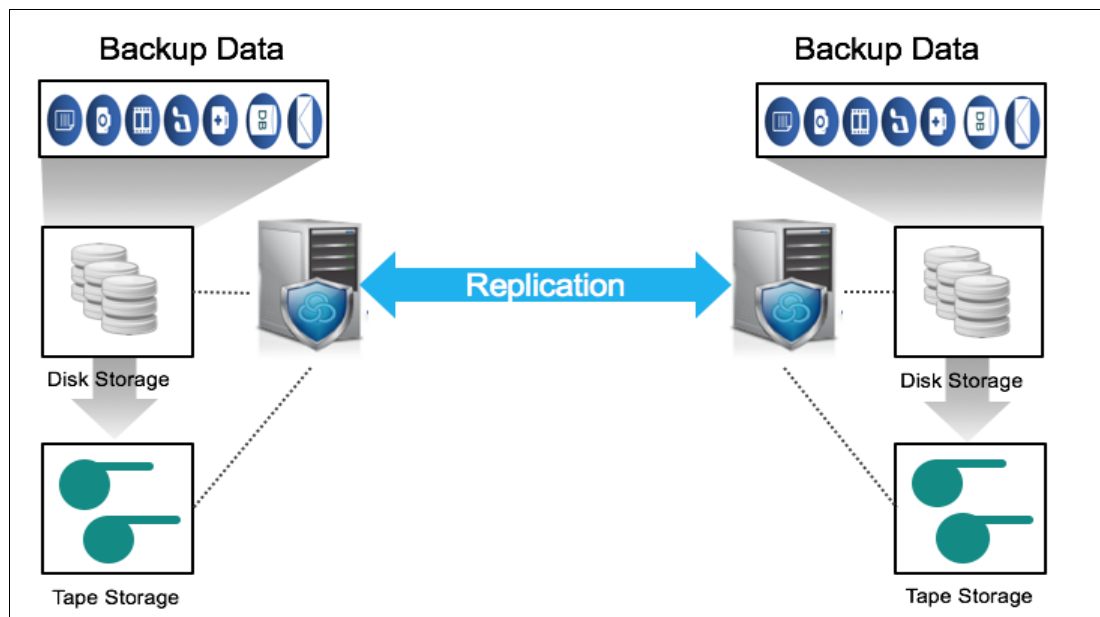


Figure 4-28 Traditional IBM Spectrum Protect architecture with electronic vaulting

Figure 4-29 shows an IBM Spectrum Protect traditional environment with *manual* offsite vaulting.

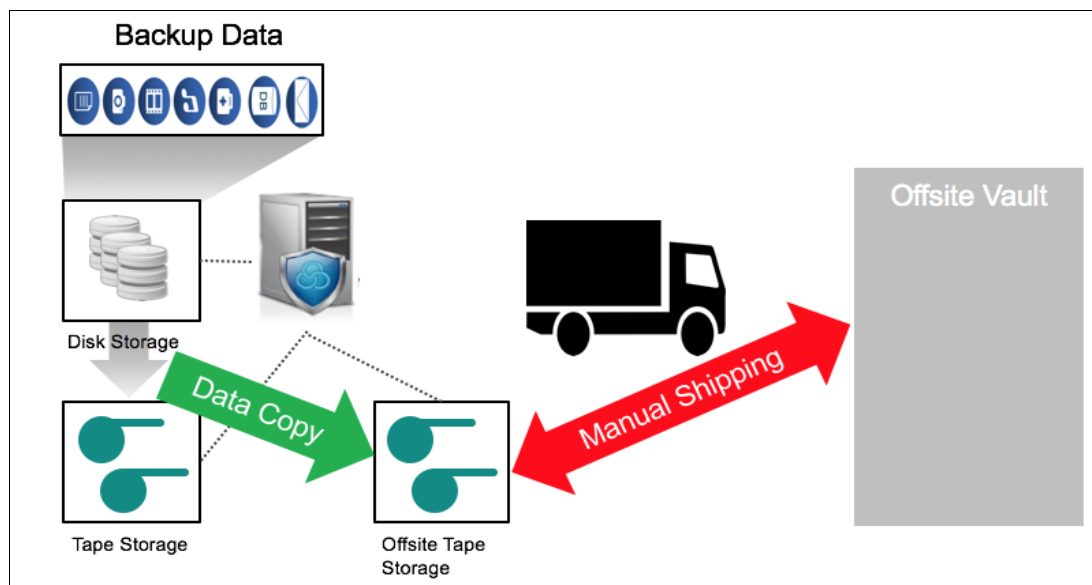


Figure 4-29 Traditional IBM Spectrum Protect architecture with manual offsite vaulting

Key factors that typically require administrator attention in these IBM Spectrum Protect environments are capacity management, data replication management, tape reclamation, and tape management.

IBM COS Integration with IBM Spectrum Protect

IBM Spectrum Protect integration with IBM COS is implemented as a storage pool replacement for conventional disk or tape storage pools. Several possible deployment options are available with IBM COS, which are further explained next.

Note: The primary and tertiary IBM Spectrum Protect instances depicted next can be on-premises or in the Bluemix infrastructure cloud. This has no bearing on the architecture other than the network link considerations between the IBM Spectrum Protect instance and IBM COS.

Figure 4-30 shows IBM COS enabled IBM Spectrum Protect architectures.

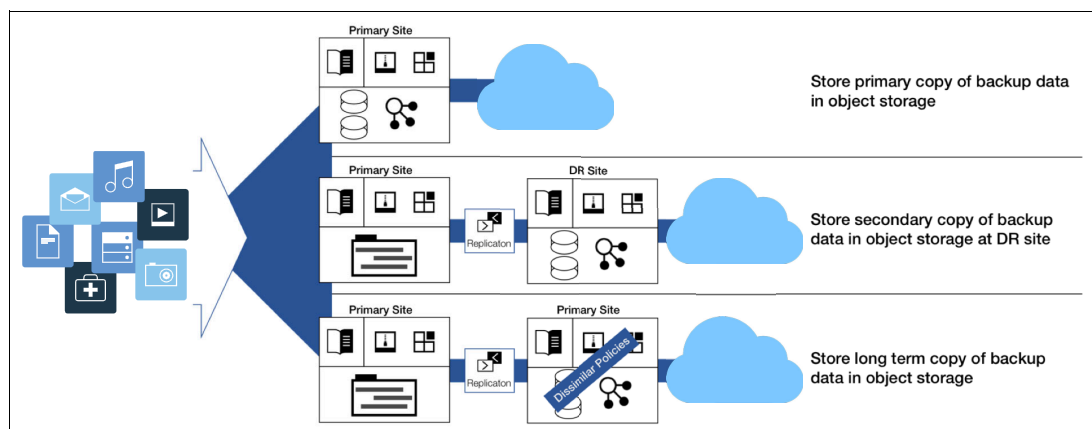


Figure 4-30 IBM COS enabled Spectrum Protect architectures

All three deployment options achieve significant improvements over traditional storage. The first option is the most cost-effective because only a single copy of the data is kept and all replication management and offsiteing are eliminated from the workflow.

This is possible only because of the resilience features of IBM COS. This means that tertiary copies are not required from a data durability perspective. This architecture should be considered where multiple data copies are not mandated by company or government regulations.

The geographically dispersed access anywhere quality means that if a primary IBM Spectrum Protect server becomes unavailable, another server can be quickly configured to use the IBM COS repository without the need to manage copy pool volumes tapes, and so on.

Good backup and restore performance is attained with the use of a small accelerator cache, which assists with initial data ingest and data reduction, and transforms the data into a format optimal for cloud transfer.

Key point: The capability to use cloud as a primary tier of data is unique to IBM Spectrum Protect; other backup and archive offerings use this as only a tertiary and archive tier.

Options two and three use a dual server setup, whereby the data is ingested onto block or file storage and then replicated to a standby server, which can be used to ingest new data and restore systems at the DR site in case of disaster, and to eventually recover the IBM Spectrum Protect server at the primary site. In the context of this architecture the tertiary Spectrum Protect server uses IBM COS for data storage.

This architecture provides excellent backup and restore performance at the primary site while providing a tertiary backup server, which may be required in case of disaster recovery or data corruption at the primary site.

Disaster recovery testing becomes a non-event because the secondary server can service restore activities without the need for complex recovery procedures.

Key point: IBM Spectrum Protect can be tightly integrated with IBM COS, providing gains in operational simplicity, while consuming less storage and using less infrastructure, all of which lead to a lower total cost of ownership.

4.4.2 IBM Spectrum Scale

IBM Spectrum Scale is a highly parallel POSIX compliant file system. At its core, the product is geared to providing high-performance file access through the General Parallel File System (GPFS) protocol, and providing access to other platforms through Network File System (NFS) and Common Internet File System (CIFS).

A key feature of IBM Spectrum Scale is the information lifecycle management (ILM) and data placement engine, which control the initial placement of data and eventual movement of data between different storage tiers.

IBM Spectrum Scale ILM has the ability to tier data off to low-cost storage. Traditionally this was tape through hierarchical storage management (HSM) features with IBM Spectrum Protect or IBM Spectrum Archive™.

Product architecture

IBM Spectrum Scale deployments consist of a number of Network Shared Disk (NSD) servers that have local or SAN disk. These NSD servers stripe data across all disk media and expose the native IBM Spectrum Scale protocol, which can be accessed by IBM Spectrum Scale clients directly. Alternatively, clients that are not using IBM Spectrum Scale can access the data through an additional protocol software layer that can expose CIFS/SMB, NFS, or Swift Object.

Data is typically initially placed on the most optimal tier depending on file name, type, extension, size, or any other file attribute. As the file attributes, access patterns, or ages change, data will tier down to lower cost forms of storage, and at some point it might be migrated out to a low-cost external storage pool but remain transparently visible to users who will trigger a recall when accessing this file. Traditionally this lowest tier used tape technologies.

Figure 4-31 shows an integration view.

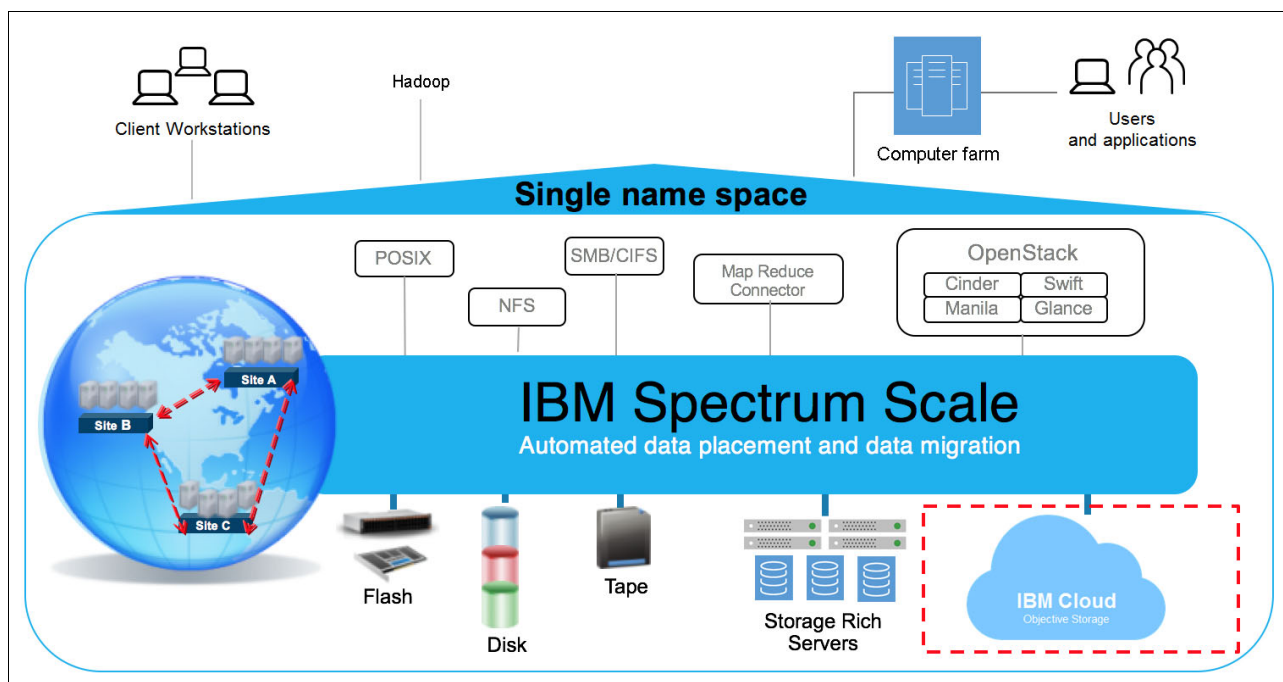


Figure 4-31 IBM Spectrum Scale overall storage and application integration view

Although tape is still one of the lowest cost storage solutions available, it does require an onsite investment, management of the tape infrastructure, and the SAN connectivity. A pool of spare tape media needs to be maintained, and the streaming characteristics of tape mean that, although performance for large files might meet or exceed the rated speeds of up to 400 MBps, small file performance is a fraction of this. Also, the number of parallel access streams is limited to the number of physical tape drives, meaning that at busy times, users and applications can face extended wait times to access their files.

IBM COS integration with IBM Spectrum Scale options

Three options of integrating IBM Spectrum Scale with IBM COS are available:

- ▶ IBM Spectrum Scale HSM
- ▶ Transparent Cloud Tiering
- ▶ Cloud data sharing

IBM Spectrum Scale HSM

The first integration option is to use the IBM Spectrum Scale HSM integration with IBM Spectrum Protect. In this case, the IBM Spectrum Scale cluster is not aware of whether the data is in an object store or tape or disk. In effect, this use case is the same as in “IBM COS Integration with IBM Spectrum Protect” on page 92, so will not be described further here; the details and benefits are the same as in that section.

Transparent Cloud Tiering

The second integration option is the native Transparent Cloud Tiering (TCT) feature in IBM Spectrum Scale. In this option, the IBM Spectrum Scale cluster is object-aware and transfers the data to IBM COS directly without the need for any HSM through IBM Spectrum Archive or IBM Spectrum Protect software.

Figure 4-32 shows Transparent Cloud Tiering.

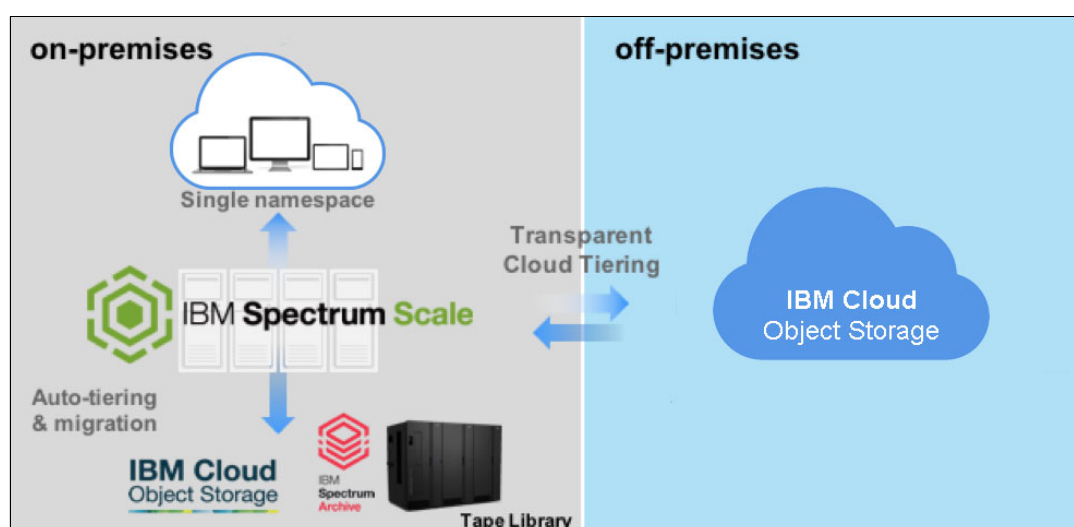


Figure 4-32 Transparent Cloud Tiering overview

IBM Spectrum Scale maintains local stubs and metadata of the objects migrated, and access to files is transparent to users. When designing a TCT solution, consider file size and network throughput in order to ensure a good user experience.

This architecture removes a number of intermediary software and hardware components from the data path, simplifying the solution stack. The result is in greater per-stream performance to the back-end object store at the cost of the data reduction features that might be present if you use the IBM Spectrum Protect HSM integration.

Data streamed to IBM COS by using this feature is not accessible directly through object APIs.

Key point: IBM COS as a low-cost tier for IBM Spectrum Scale inactive data frees up high performance infrastructure and capacity so that system owners can spend less time and money managing space, and can therefore redirect these resources into further enhancing their IBM Spectrum Scale deployments performance capabilities.

Cloud data sharing

The final integration option is cloud data sharing, which differs significantly from the tiering construct because IBM COS is used as a central data store and data distribution point as opposed to a tier of data.

Figure 4-33 shows the cloud data sharing option.

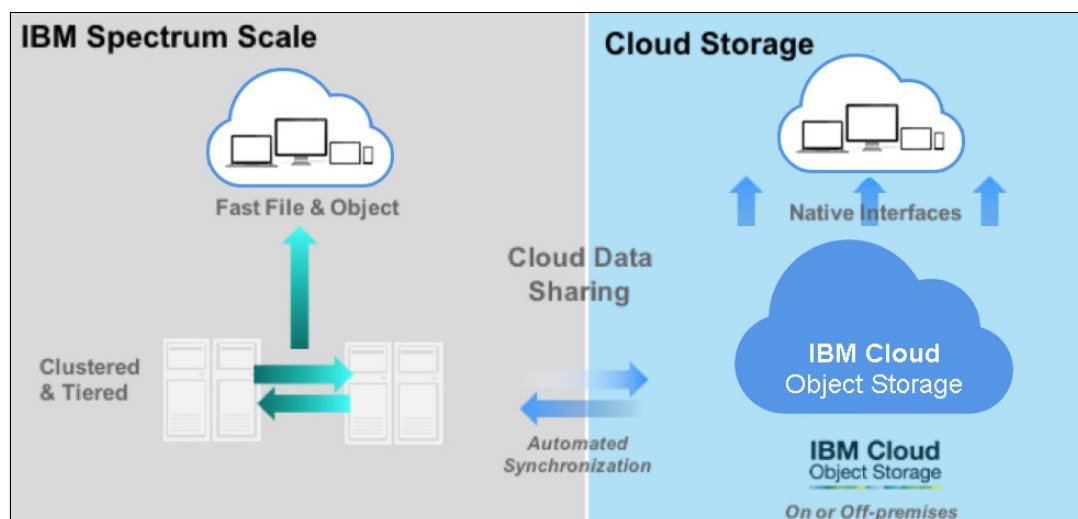


Figure 4-33 Cloud Data Sharing overview

The key difference in this architecture is that the data is not tiered, but rather it is copied to the cloud through object APIs. This means that IBM COS becomes the central data repository and *data distribution point*, with one or more IBM Spectrum Scale clusters pushing data into the cloud. These clusters can pull data on demand so they can leverage the local high performance characteristics of IBM Spectrum Scale as required, with flexibility to retain or delete the data after it has been pushed to the central IBM COS repository.

Data pushed and deleted remains in IBM COS, meaning it is accessible and retrievable when required, however file system tracking spaces (inodes) are no longer consumed.

In addition, unlike many other file-to-object integration solutions, the data in IBM COS can be directly referenced and accessed through object storage clients or applications using IBM COS S3 APIs.

Further details are in *Cloud Data Sharing with IBM Spectrum Scale*, REDP-5419.

Key point: IBM COS can serve as a central data distribution point for enterprises that want to share file-based data across regions.

4.4.3 IBM Watson IoT

The IoT is transforming how businesses compete. The digitization of the physical world is powering a radical reimagining of how organizations and individuals learn, and decide and act on opportunity. It is shifting the ways they come together, creating new ecosystems that drive competitive advantage. The IoT is an essential component of today's business strategy because it can assist with the following business needs:

- ▶ Creating a platform for exploration and invocation
- ▶ Tapping into new data types to solve business challenges
- ▶ Starting a transformative journey with cognitive IoT

IBM offers a broad range of technologies, software, services, and expertise to help organizations succeed with IoT and cognitive computing.

IBM Watson platform for cognitive IoT enables you to quickly connect devices, infuse intelligence into your applications and services, and capitalize on data to realize emerging opportunities.

IBM helps line-of-business leaders across roles and industries achieve more, drawing on a comprehensive portfolio for IoT applications.

Game changing IoT and cognitive technologies offer a clear path forward for organizations seeking to capitalize on new insight.

IBM Watson technology represents the leading edge of cognitive computing. Using natural language and other means of communication to converse with people, Watson can reduce the need for users to know complex coding or data science or to have specialized IT expertise.

IBM Watson IoT architecture

Several significant challenges are unique to the IoT architecture:

- ▶ Innovating and seizing IoT growth opportunities by using the insight from IoT data.
- ▶ Automating smart processes using strength in cognitive, analytics, security, and cloud to catalyze and monetize the transformation of global technology.
- ▶ Improving engagement by providing a rich programming platform and exploring new business models with new revenue opportunities.

Figure 4-34 shows the architecture and data flows for an IBM Watson IoT application.

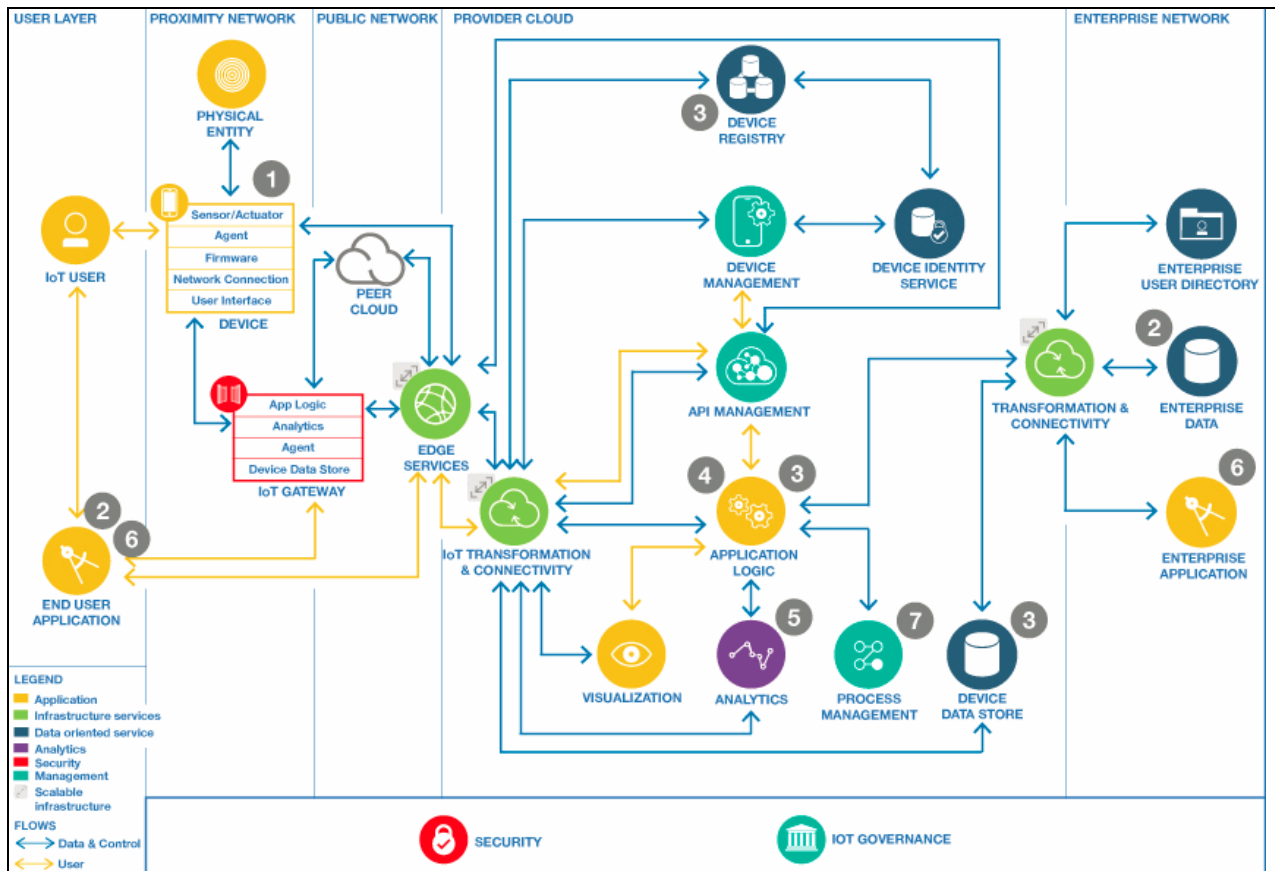


Figure 4-34 IBM Watson IoT architecture

Smart homes with connected devices and sensors enable insurance companies to improve service for their policy holders while providing insight into risks in the home. By connecting home ecosystem partners, insurers, and services such as weather reporting, the IoT for Insurance solution leverages key components of the Cloud Reference Architecture:

1. Sensors and actuators are deployed in the home and attached to the device maker's cloud service. The sensors can include water leak detection, water flow, and temperature, and the actuators can include automatic water shutoff valves.
2. The homeowner logs in to the insurance mobile application and authorizes the insurance service to access the device maker's (peer) cloud and device data. The mobile application sends the authorization token and insurance company identifier to the cloud service.
3. This information is used to map the user, devices, and insurance policy within the cloud service. The device cloud service is used because the device makers already deployed into their cloud and then own the lifecycle of the device and the user experience with the devices.
4. The insurance service receives information such as authorization, device details, and insurance ID from the insurance mobile application and processes this in several nodes (application logic, device registry, and device data store). The devices are registered with the device registry and data mapping is updated in the application logic component.
5. The insurance service application connects to the device maker (peer) cloud using the authorization token and requests the data. The application is set up to pull data on a configured interval. In addition to device data, the application can be configured to access other data sources such as a weather data service for use in analysis.
6. Data from devices and other sources such as the weather service are continually updated and sent to analytics to determine if a potential risk threshold was exceeded. This data is analyzed to determine if a potential exists for damage to the home (including water damage, freeze potential, and more). After a determination is made that a problem exists, notifications are sent to the homeowner and to the insurance company using the analysis from step 5. The homeowner can then take an action to respond to the notification and determine whether damage occurred, and the insurance company can initiate a claim process.
7. If damage occurred, the insurance business process of claims management is initiated. The insurance business processes can be accomplished in the cloud service, their enterprise applications, or their mobile applications. This is dependent on how and where the insurance company decides to perform the business logic.

IBM COS integration with Watson IoT

Figure 4-35 shows the integration.

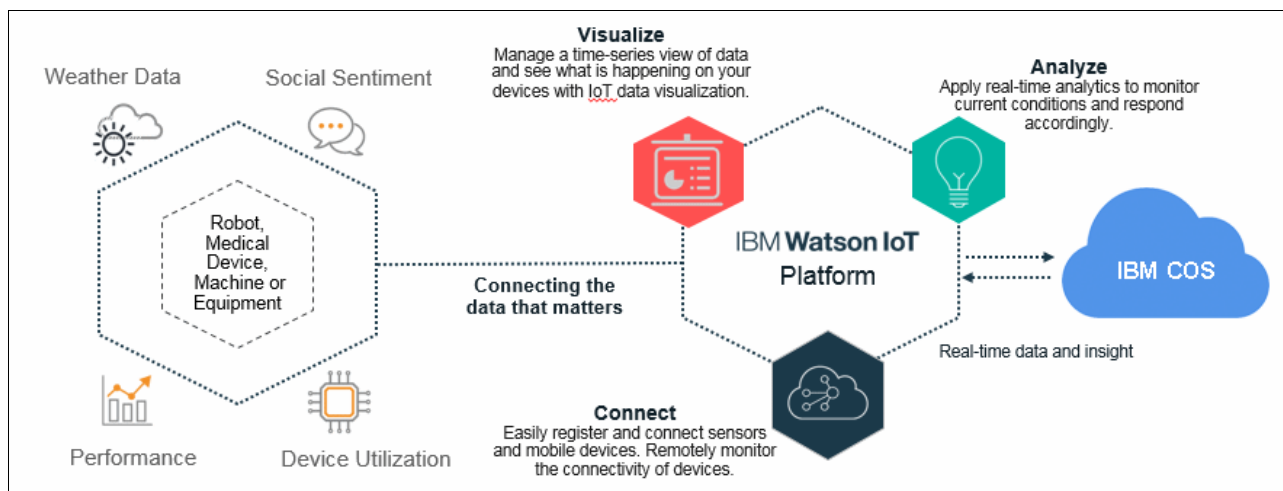


Figure 4-35 IBM Watson IoT integration with IBM COS

Millions of sensors provide the information about equipment, robots, machines, or any devices. These types of unstructured data feeds to the IBM Watson IoT platform.

Data is easily registered and connect sensors and mobile devices. Remotely monitor the connectivity of different devices. Manage a time-series view of data and see what is happening on your devices with IoT data visualization in real time.

Apply real-time analysis to monitor current conditions and respond accordingly. Machine learning models continuously sweep for non-obvious patterns. The size of data is increasing daily from terabytes to petabytes; maintaining this data is a challenge. This large data, files, video, and images are stored in IBM COS and are available to users, customers, and devices when needed, anywhere and anytime.

IBM COS integrated with IBM Watson IoT Platform solves these complex challenges and store real-time data and insights. Watson Explorer harvests insights from documents and Watson IoT Equipment Advisor assists with interactive diagnostics and manual data mining.

Watson IoT Equipment Advisor collects response effectiveness to enhance the corpus and machine learning model.

Key point: IBM Cloud Object Storage can play a major role in storing large data, video, files, and images in a readily accessible manner. IBM COS is a cost effective cloud storage for virtualization, security, backup storage, and file repository in IBM Cloud Platform.

4.4.4 IBM Aspera

IBM Aspera is a suite of products designed for data transfer and data sharing over long distances at high speeds and not feasible using traditional network protocols. The Aspera suite includes user capabilities such as file sync and share, “FTP-like” data access, web-based browser access, and local file access when the Aspera client is installed.

IBM Aspera architecture

Aspera technology is built on a foundation of the Fast and Secure Protocol (FASP) that is geared toward superior flow control and link utilization when compared with TCP, allowing data transfers to reach line speed even on high latency links with significant packet loss.

Data transfers between sites are facilitated by Aspera client and server software; the data endpoints were traditionally block or file storage. However, the shareable nature and low-cost characteristics of object solutions such as IBM COS are attractive to organizations that want to store data in a centralized and cost-effective manner.

Figure 4-36 illustrates FASP performance.

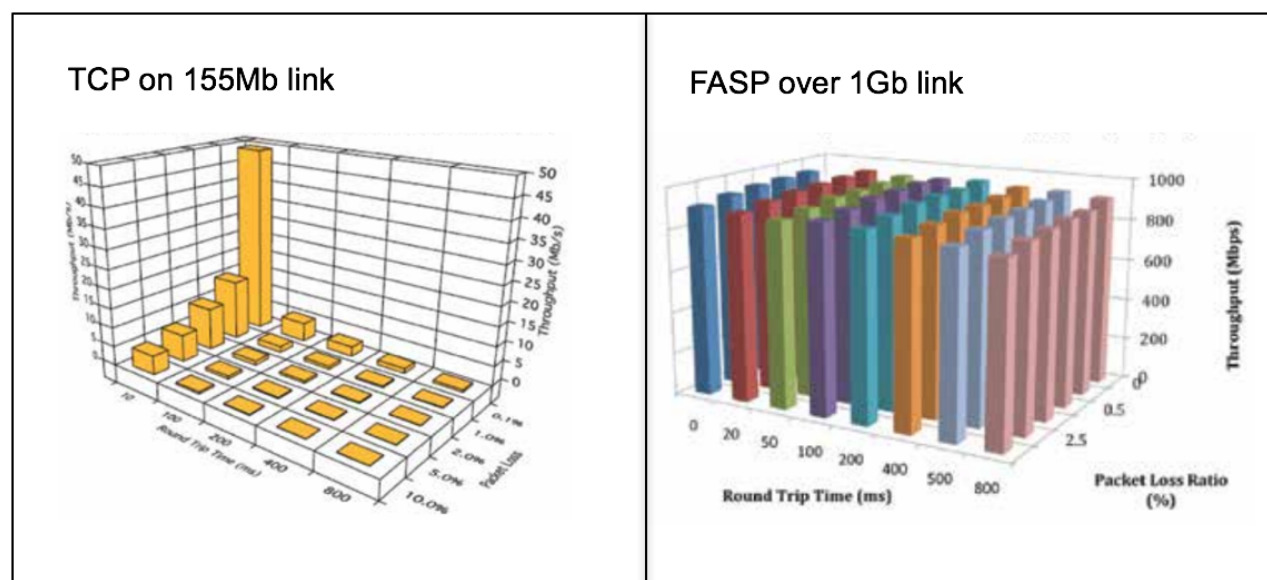


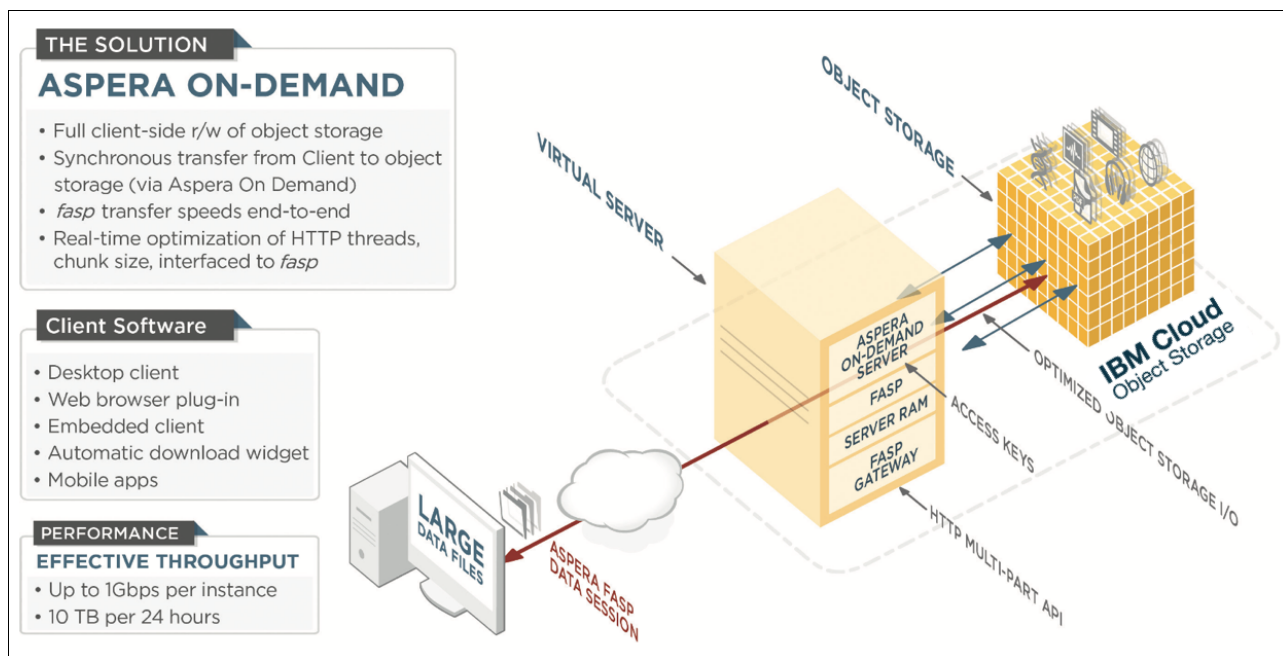
Figure 4-36 Aspera FASP performance

Important: The TCP can saturate the link to only one-third of its capacity, whereas using FASP can reach nearline speed regardless of latency and with far less impact from packet loss.

This performance differential over long distance and impaired links is the key differentiator for the Aspera FASP technology.

IBM COS integration with Aspera

Figure 4-37 illustrates the Aspera integration with IBM COS.



The Aspera solution consists of client software installed on a workstation or server sending encrypted data over the WAN/Internet to the Aspera On-Demand server located in the IBM Bluemix infrastructure cloud.

The Aspera-on-demand server then intelligently chunks the data in real time and leverages the multipart upload capability of the object protocol to ensure that the last hop to IBM COS is not a bottleneck to achieving maximum end to end throughput.

The speed of the service available to consumers can be managed by adding an extra Aspera On-Demand server, which can be done either manually or as an automated spin-up and spin-down, as mandated by system workload.

The combination of Aspera high-speed ingest capability, with the cost-effective resilient nature of IBM COS can provide a number of opportunities:

- ▶ Increased throughput to IBM COS using existing links.
- ▶ Reduced link bandwidth if additional performance exceeds business need.
- ▶ Facilitated data migration to the cloud by providing high speed predictable ingest speeds.

Key point: Aspera high speed ingest capability can be a key enabler for businesses to predictably and cost effectively move data into the IBM Cloud Object Storage.

4.4.5 IBM Watson API on Bluemix platform

IBM Cognitive Solution is implemented using Watson API on Bluemix platform. Most of Watson Cognitive Services can be used as unbound Bluemix services. No need to create a Bluemix application. Select a Cognitive application on the Bluemix platform and decide whether to expose the Managed API through business operations (API Connect).

Watson allows for getting deeper insight and discovery patterns from unstructured data. Watson API on Bluemix platform leverages deep content analysis and evidence based reasoning to accelerate and improve decisions, reduce operational costs and optimize business outcome.

It continuously ingests and analyzes unstructured data. Watson does not prepare answers, but determines its answers and associated confidence scores based on acquired knowledge. IBM Watson API exposes as APIs on the Bluemix platform to build innovative client apps and integrate with other applications.

Figure 4-38 illustrates IBM Watson Cognitive System.

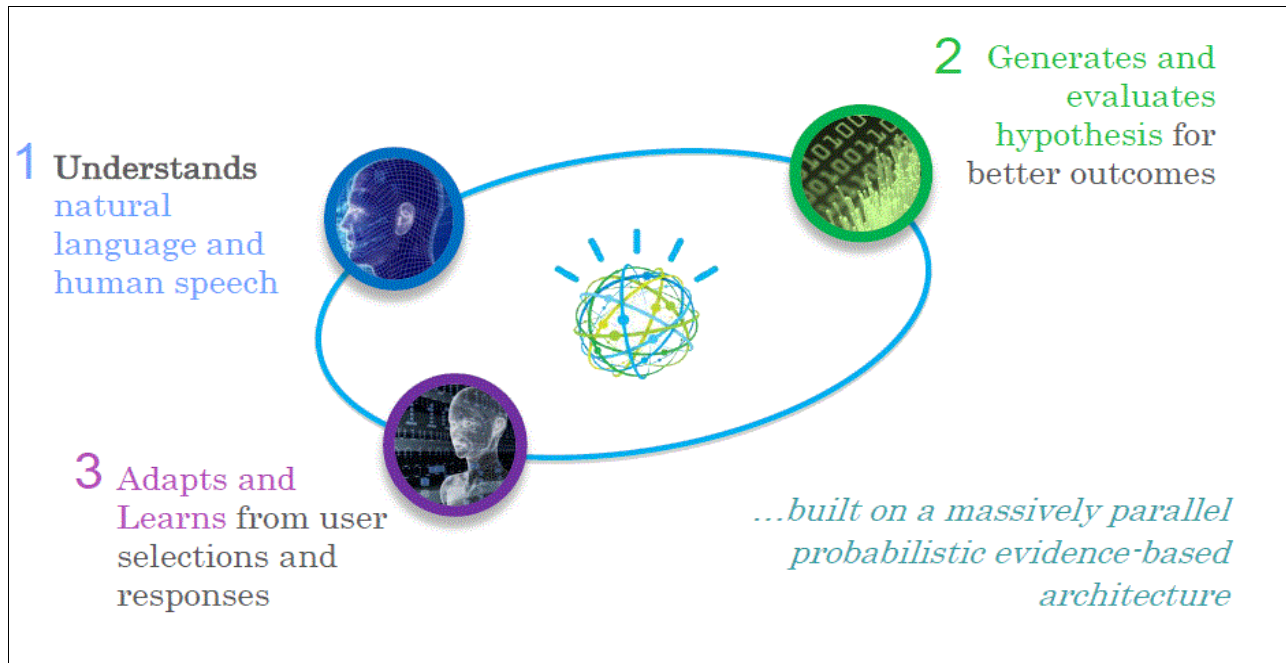


Figure 4-38 IBM Watson Cognitive System

IBM Watson API services are available on the Bluemix platform:

- ▶ Language: Alchemy Language, Conversation, Dialog, Document Conversion, Language Translator, Natural Language Classifier, Personality Insights, Retrieve and Rank, and Tone Analyzer.
- ▶ Speech: Speech to Text, Text to Speech
- ▶ Vision: Visual Recognition
- ▶ Data insights: Alchemy Data News, Discovery, Trade Off Analytics
- ▶ Embodied cognition: Project Intu
- ▶ IBM Watson API: Conversation architecture

Watson Conversation allows you to create applications that understand natural-language input and use machine learning to respond to customers in a way that simulates a conversation between human beings.

Watson API combines a number of cognitive techniques to help you build and train a bot: defining intents and entities and crafting dialog to simulate conversation.

The system can then be further refined with supplementary Watson APIs and other technologies to make the system more human-like or to give it a higher chance of returning the correct answer.

Watson API allows you to deploy a range of bots through many channels, from simple narrowly focused bots to much more sophisticated, full-blown virtual agents across mobile devices, messaging platforms like Slack, or even through a physical robot.

Users send data from various sources (systems of engagement) to an application. Different API services are enabled through applications, for example Watson Conversation and other Watson services. These API and applications connect to back-end services (systems of record) that store large amounts of data.

Figure 4-39 shows the Conversation architecture.

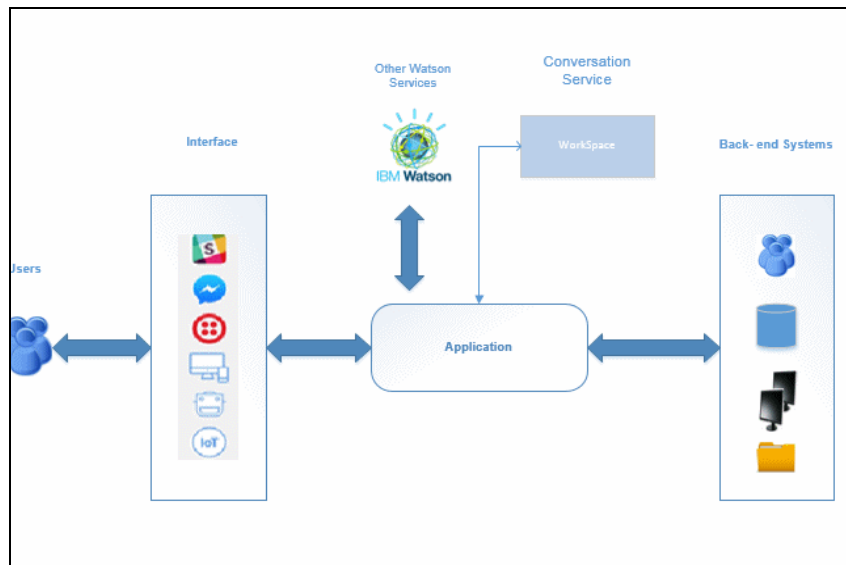


Figure 4-39 IBM Watson API: Conversation architecture

IBM COS integration with Watson API architecture

IBM Watson API Question and Answer (QAAP) service is integrated with IBM COS to deliver robust applications on the Bluemix platform. A large amount of data, file repository, and complex data are stored in IBM Cloud Object Storage (COS) and are accessed through API and applications to service within Watson API on the Bluemix platform.

The Watson QAAP is a REST styled service interface that allows applications to interact with Watson.

Using this API, one can pose questions to Watson, retrieve responses, and submit feedback on those responses. It is a JSON based payload and provides transparency with its conclusions.

IBM COS is integrated with Watson API for different cognitive solutions on the Bluemix platform to store large complex structured and unstructured data.

Figure 4-40 shows a Watson API and Bluemix integration.

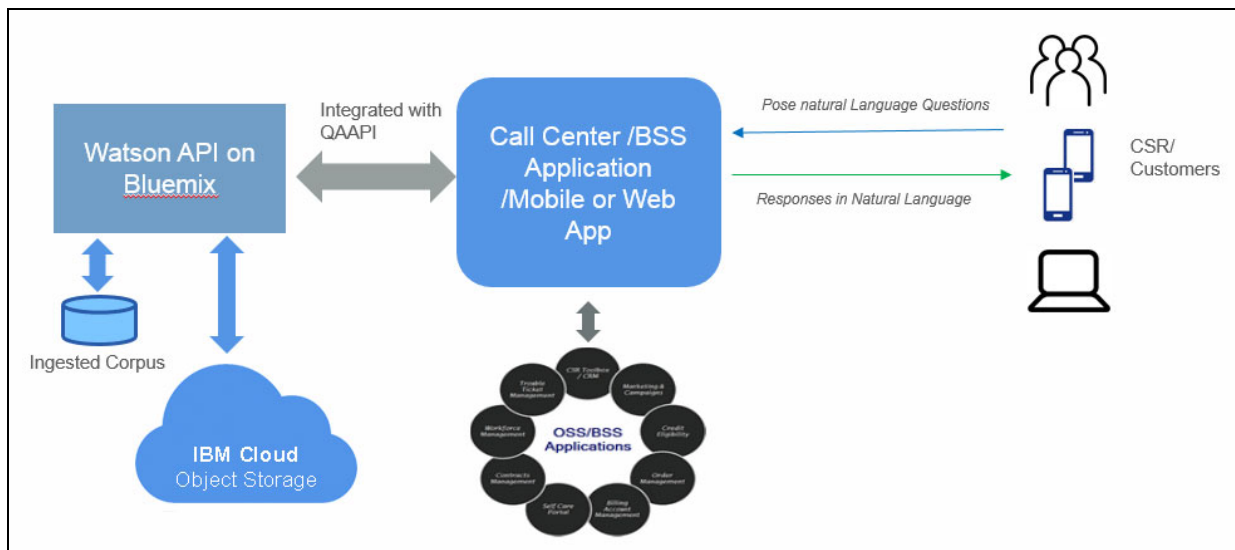


Figure 4-40 IBM Watson API integrated with IBM COS on Bluemix platform

Key point: IBM can deliver a cost-effective, scalable, and manageable solution for a compute-centric data-intensive solution using IBM Watson API and COS on the Bluemix platform.



Building a simple Web Gallery application

This chapter serves as a basic example of how you might use the IBM Bluemix development platform to build a simple web application that interacts with IBM Cloud Object Storage (IBM COS). This chapter can be a starting point for creating a user interface (UI) for the content repository use case that is described in 4.3.6, “Use case 6: Content repository” on page 86. You might want to build a more elaborate application to manage the unstructured data you or your company stores in IBM COS. We cover some of the basics and show you how to get started.

The following topics are covered in this chapter:

- ▶ 5.1, “Scenario description” on page 106
- ▶ 5.2, “Architecture of the scenario” on page 107
- ▶ 5.3, “Implementation of the scenario” on page 111
- ▶ 5.4, “Verifying the scenario” on page 134
- ▶ 5.5, “Conclusion” on page 138

5.1 Scenario description

This scenario shows you how to build a simple Web Gallery application, step by step. You create an IBM COS instance, create, and deploy an app on IBM Bluemix, create a GitHub repository, and set it up for continuous delivery on Bluemix. Then, you build an IBM COS Web Gallery to show you working examples of code that reads and writes data to IBM COS. The goal is to simplify and make accessible the uploading, retrieving, and displaying of digital assets that are stored in IBM COS. This scenario can help you jumpstart your development on IBM COS and empowers you to get the most value from this advanced storage technology.

The source code for the scenario: The two ways to download the source code used in this scenario are as follows:

- ▶ From the [Code tab on GitHub](#)
- ▶ From the IBM Redbooks web server. Download instructions are in Appendix A, “Additional material” on page 193.

5.2 Architecture of the scenario

This section describes the high-level architecture, and the application and operational architectures of the scenario.

5.2.1 High-level architecture

The Web Gallery app is hosted at the IBM Bluemix cloud and consists of two main components (Figure 5-1): a Cloud Foundry Node.js app and an IBM COS instance running on IBM Bluemix Infrastructure. The app user interface (UI) is accessible to users who have a web browser that is installed on a local device.

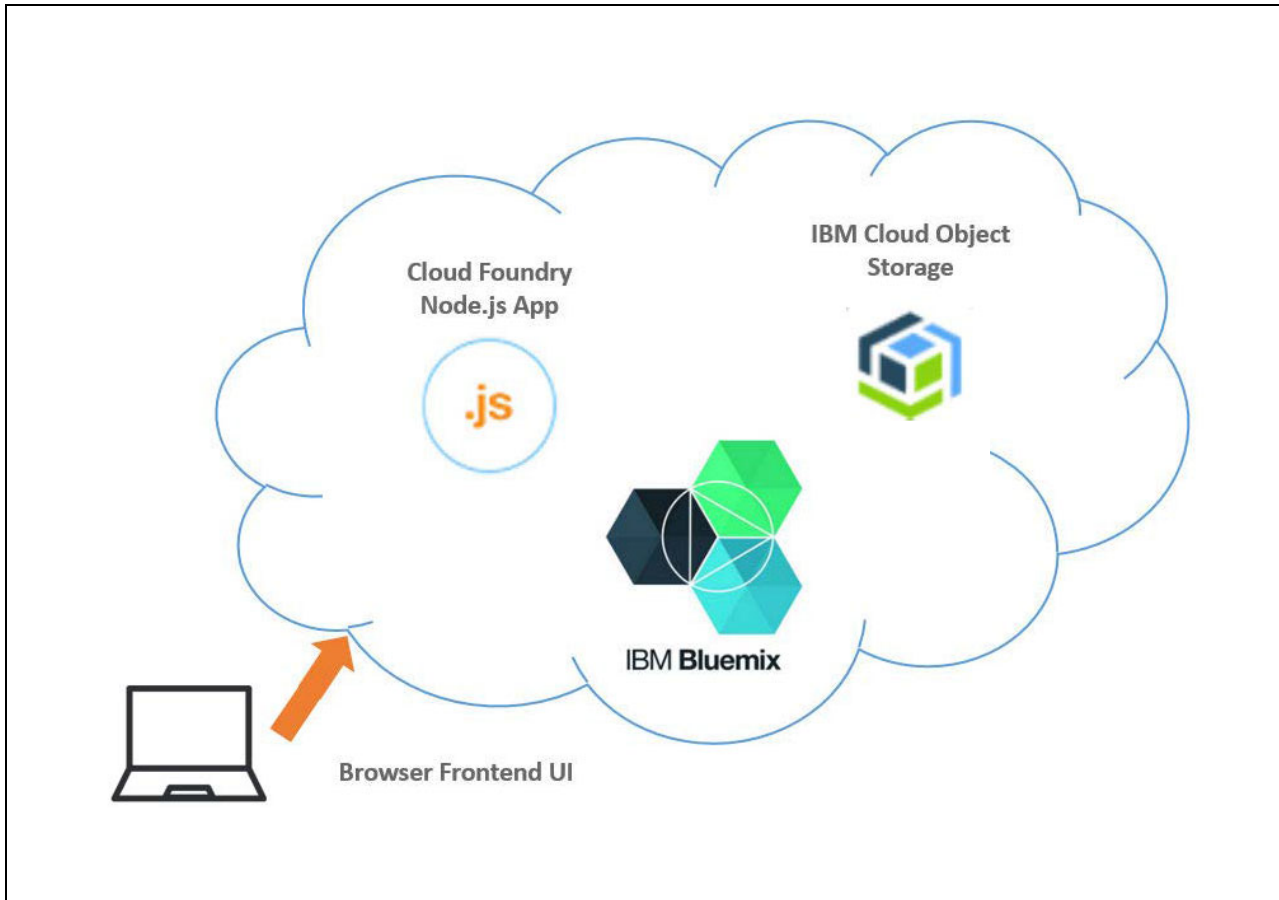


Figure 5-1 Web Gallery architecture context

Application architecture

Considering the key components, using a Model–View–Controller (MVC) architecture for the application makes sense, as shown in Figure 5-2. At a high level, the major building blocks that the architecture uses are HTML with CSS and JavaScript to generate the view on the presentation layer, JavaScript running on Node.js as the controller on the business layer, and IBM COS on the storage layer becomes our model.

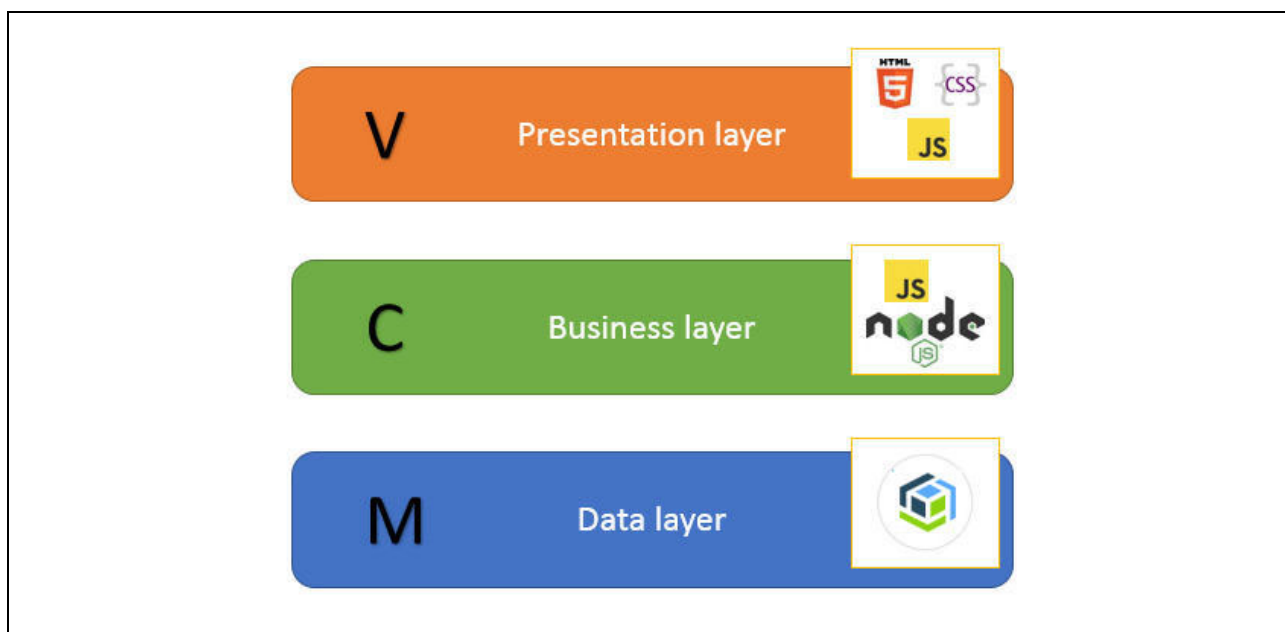


Figure 5-2 High-level application architecture

Because Node.js drives the business logic, you want to take advantage of some Node.js packages to help accomplish your goal. The Express framework keeps the server-side MVC code simple, and Embedded JavaScript (EJS) as the templating engine makes dynamically rendering the view easy. All the S3 API calls to the IBM COS data layer use the AWS SDK for JavaScript. These are the key packages that you use along with several others.

Operational architecture

Figure 5-3 shows a high-level overview of using a local browser to open the image upload form and send a **POST** to the Node.js controller, which after authenticating with the IBM COS credentials allows the app to make an S3 API call to **PUT** the image into the IBM COS destination bucket.

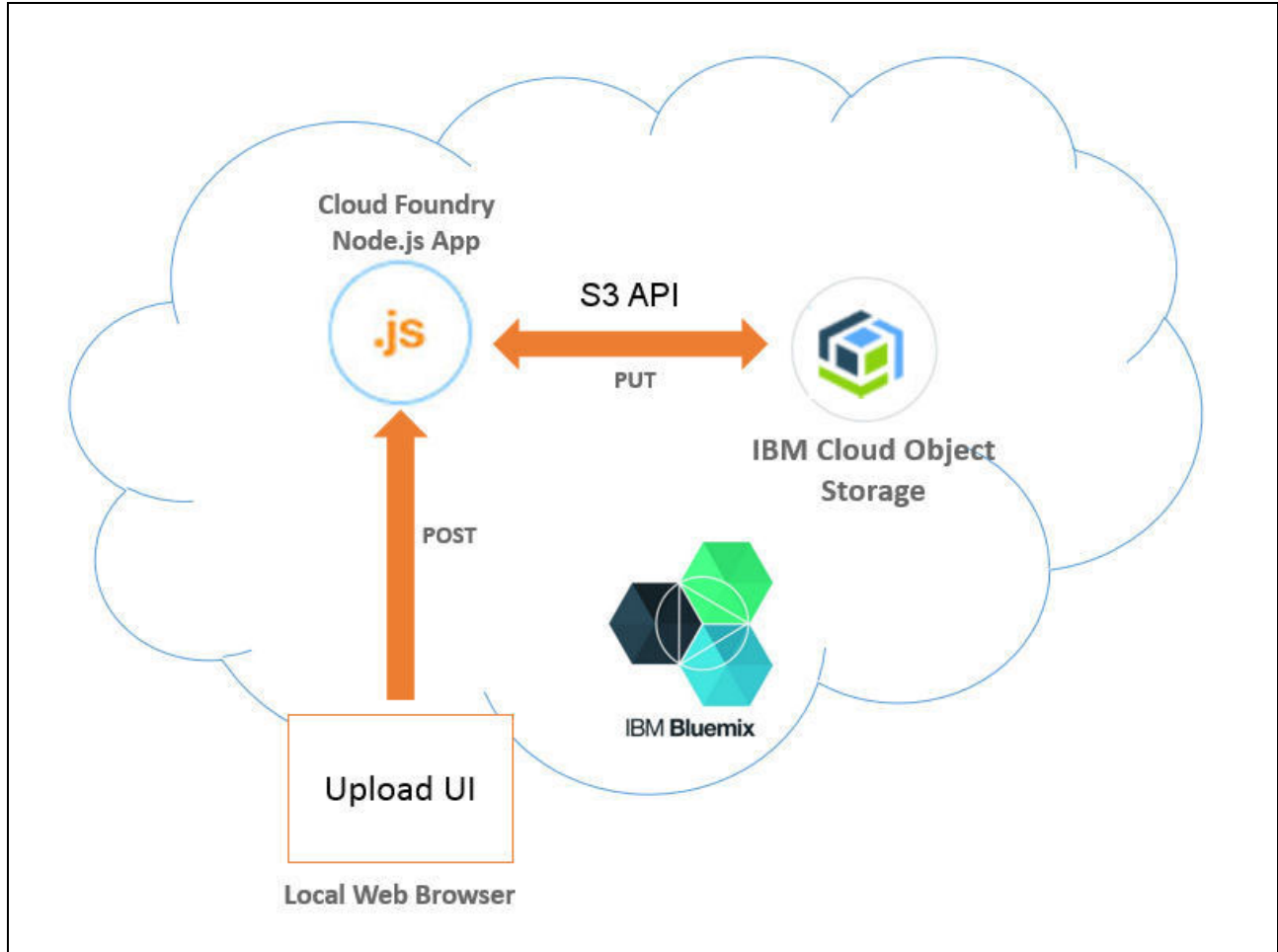


Figure 5-3 Operational architecture: Upload images to the Web Gallery

In Figure 5-4, the user clicks the **Gallery** button, which issues a **GET** from the browser to open the gallery view. This request is routed to the controller, which performs a logic call that sends an S3 API call to **GET** a list of URLs from the IBM COS bucket for the Web Gallery view. After receiving the image URLs, the Web Gallery view is rendered for the user.

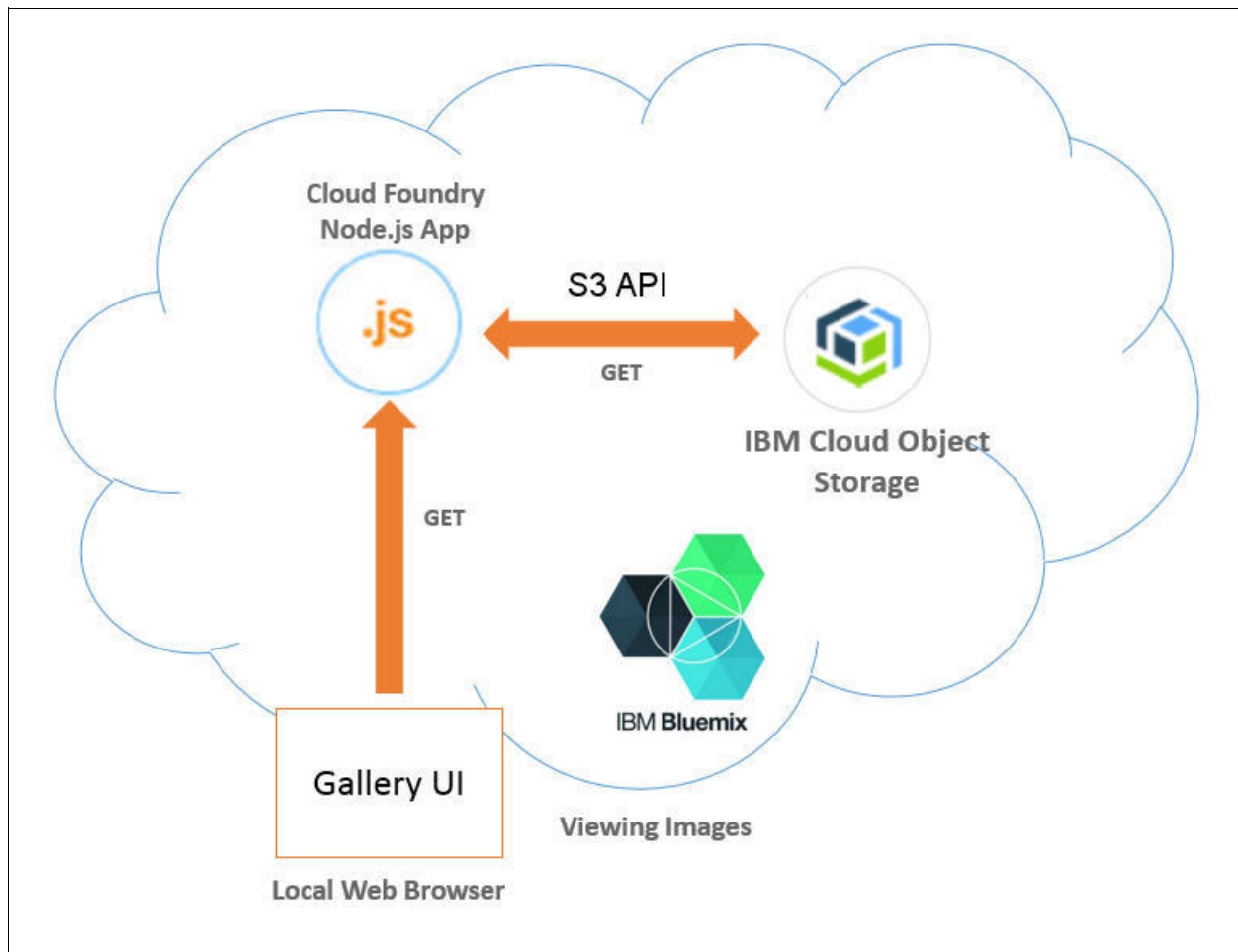


Figure 5-4 Operational architecture: Display images in the Web Gallery

5.3 Implementation of the scenario

This section covers the step by step implementation of this scenario.

5.3.1 Getting started

Before getting started with writing any code, you must ensure that you have the following items set up:

- ▶ IBM Bluemix Infrastructure account with IBM COS
- ▶ IBM Bluemix account
- ▶ Cloud Foundry
- ▶ Node.js
- ▶ Git

The steps show how to set up each of these items.

5.3.2 Ordering IBM COS on IBM Bluemix Infrastructure (SoftLayer)

The *IBM Bluemix Infrastructure portal* is currently not the same as the *IBM Bluemix portal*, so do not bypass this step.

Before you can order IBM COS, you need a Bluemix Infrastructure account because the IBM COS service runs on that account. If you do not yet have an IBM Bluemix Infrastructure (SoftLayer) account, your account will be created after you order IBM COS. To place your order, go to the [Object Storage](#) page of the Bluemix website and be sure to select the **Cloud Object Storage - S3 API** option during the ordering process (see Figure 5-5). For more information about how to order IBM COS if you already have an IBM Bluemix Infrastructure (SoftLayer) account set up without IBM COS, see 3.1, “Setting up your cloud account” on page 26.

Order Object Storage

▼ Object Storage Configuration

Object Storage

☒ Cloud Object Storage - S3 API [Pay As You Go]

This option is for frequently accessed data that requires the highest level of availability. Your data is available through multiple endpoints across three or more geographic zones, helping ensure business continuity.

IBM Cloud Object Storage pricing is based on a rate for storage capacity used, operational requests, and a usage rate for public outbound bandwidth. Inbound and private network bandwidths are included at no additional cost.

Standard Cross Region	\$0.0300/GB
Standard Cross Region Bandwidth	\$0.0900/GB
Standard Cross Region API Request (Class A)	\$0.0060/1000 requests
Standard Cross Region API Request (Class B)	\$0.0050/10000 requests

Object Storage

☐ Cloud Object Storage - Standard Regional Swift API [Pay As You Go]

Summary

\$0.00 /mo.

+ \$0.00 in one-time fees

ADD TO ORDER

Configuration

Object Storage	
Cloud Object Storage - S3 API	\$0.00 /mo.
Object Storage	\$0.00 /mo.

Figure 5-5 Order the IBM Cloud Object Storage S3 API

5.3.3 IBM Bluemix account

If you are new to IBM Bluemix and have not yet created your Bluemix account, do that now. You can try it out at no charge for 30 days. Go to the [Bluemix website](#) and click **Sign-up**. After you complete the form and click **Create Account**, and you are sent an email asking you to confirm your account. Click **Confirm Account** and sign-in to your Bluemix account.

Four steps remain until your account setup is complete:

1. Accept the terms and conditions.
2. Create your organization by specifying an organization name.
3. Create a space in your organization to do your development.
4. Review the summary, note your org name and space name, and then click **I'm Ready**.

You are taken to your My IBM page. Find IBM Bluemix under Products and services, and click **Launch** (see Figure 5-6). Your IBM Bluemix account is set up. Now, set up a few more things locally and you are ready to create the app.

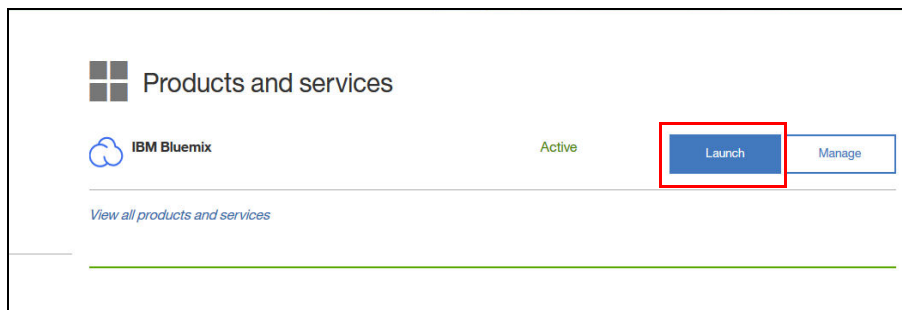


Figure 5-6 Launch IBM Bluemix from My IBM

5.3.4 Installing Node.js

The app uses Node.js as the server-side JavaScript engine to run the JavaScript code. You must install node locally so that you can use the node package manager (NPM). It is also helpful to have Node.js installed so that you can test your code locally. Go to the [Node.js releases](#) web page and download the Long Term Support (LTS) Version of Node.js, which matches the latest version supported by the SDK for Node.js buildpack on IBM Bluemix. At the time of this writing the latest buildpack is Version 3.10, and it supports Node.js Version 6.9.4. You can find information about the latest Bluemix SDK for Node.js buildpack on the [SDK for Nodejs latest updates](#) page. Run the Node.js installer to set up Node.js and NPM on your system.

5.3.5 Installing Git

Git is the most widely used source code versioning system in the industry. We use Git later when we create a toolchain in Bluemix for continuous delivery. If you do development regularly, you probably are already familiar with Git. If you do not have a GitHub account, create a free public personal account at the [Github](#) website; otherwise, feel free to use any other account you might have.

Go to the [Github Desktop](#) page to download GitHub Desktop, and then run the installer. When the installer finishes, you are prompted to log in to GitHub with your account.

In the Log in window (Figure 5-7), enter the name and email you want displayed publicly (assuming you have a public account) for any commits to your repository.

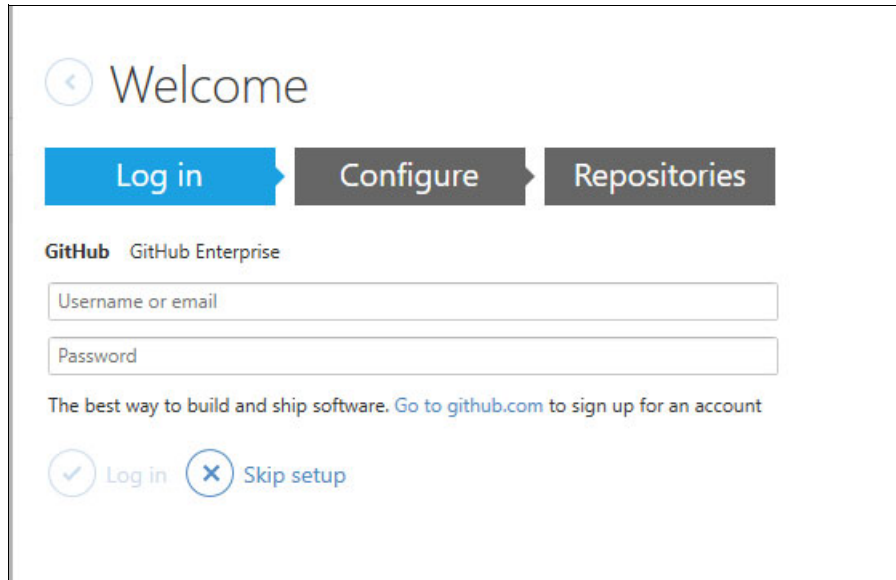


Figure 5-7 GitHub Desktop post install configuration

You do not have to create any repositories yet. You might notice a repository named Tutorial that is included with GitHub Desktop to help familiarize you with the flow. Feel free to experiment with it.

5.3.6 Creating the Web Gallery app on IBM Bluemix

To create a Cloud Foundry app, log in to [IBM Bluemix](#) and click **Create App** (Figure 5-8).

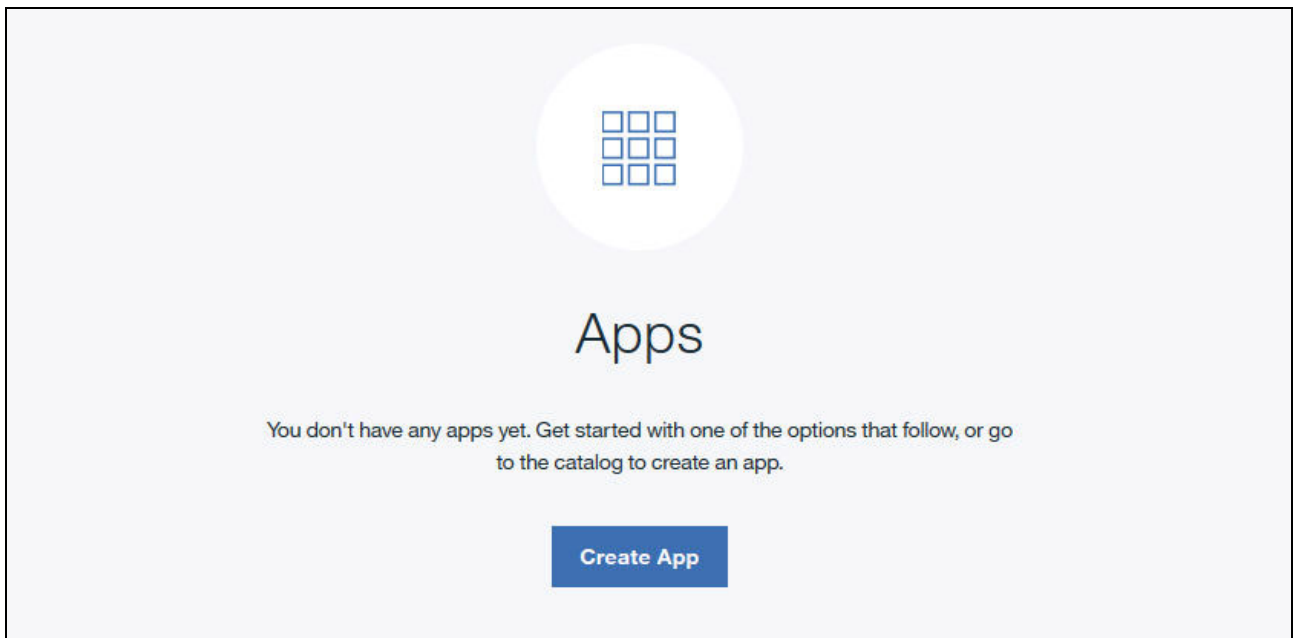


Figure 5-8 Create an app in IBM Bluemix

Then, under Cloud Foundry Apps, select **SDK for Node.js** (Figure 5-9).

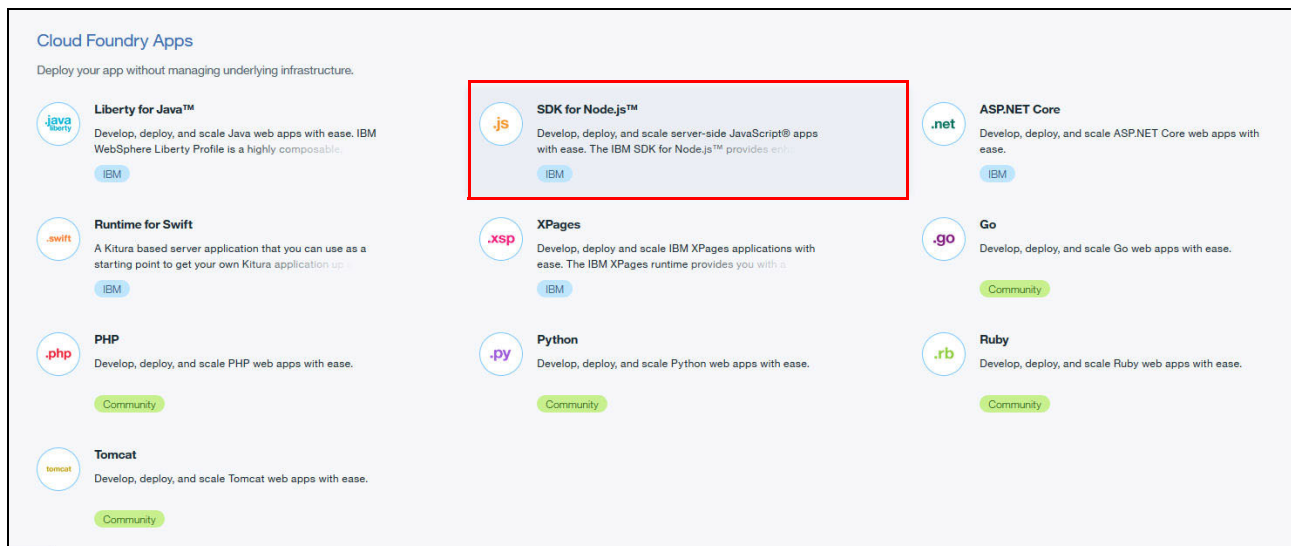


Figure 5-9 Select Cloud Foundry app SDK for Node.js

Figure 5-10 shows the app creation page where you give a name to the app. Call it something descriptive, such as COS-WebGallery. The Host name is populated automatically to reflect the App name field. The Host name with the Domain name becomes the address that you use to view the app on Bluemix. The Host name can be updated later if necessary, so accept the default and click **Create**. Bluemix creates a starter app, deploys and starts it for us.

App name:

COS-WebGallery

Host name:

COS-WebGallery

Domain:

mybluemix.net

Pricing Plans

Monthly prices shown are for country or region: [United States](#)

PLAN	FEATURES	PRICING
✓ Default	Run one or more apps free for 30 days (375 GB-hours free). This is a service plan for the IBM Bluemix Platform runtime.	\$0.07 USD/GB-Hour

[Terms](#)

\$0.07 USD/GB-Hour **Create**

Figure 5-10 Name and create the IBM Cloud Object Storage Web Gallery app

Now that the app is created and running, click **View App** from the app's **Getting Started** page to see it in a new browser window. It was created with a basic Hello World starter app as a placeholder (Figure 5-11).

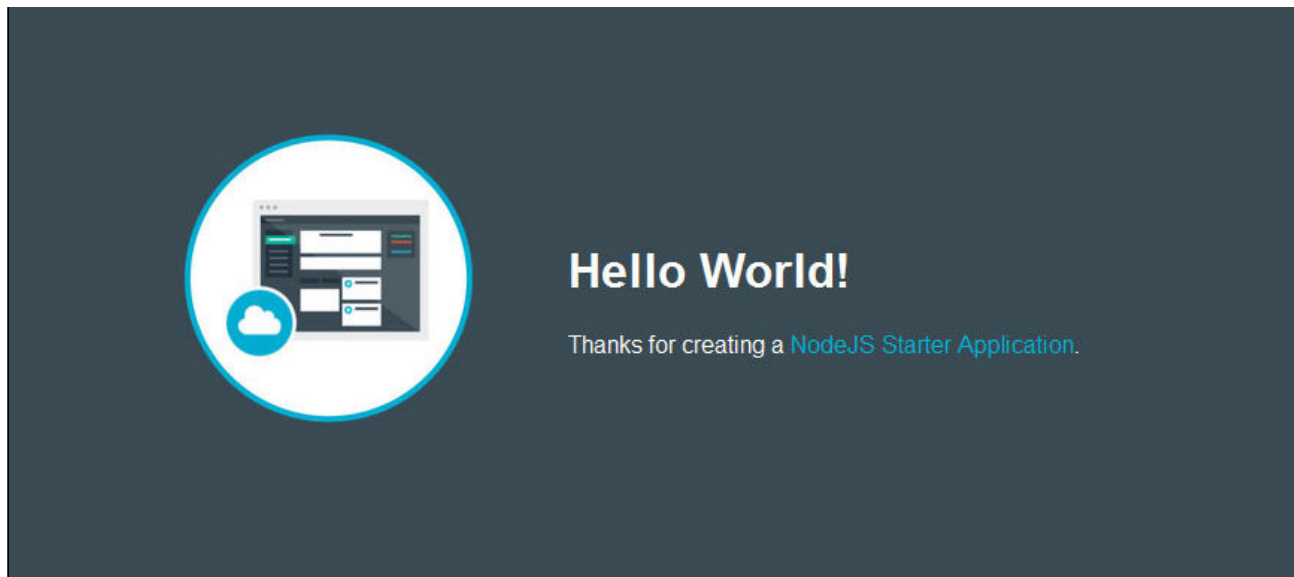


Figure 5-11 Initial Node.js Hello World starter app deployed after app creation

Notice back on the Getting Started page the prerequisites that you need in for developing a Node.js app on Bluemix are listed. You already created your Bluemix account, and installed Node.js. Download the [Cloud Foundry \(CF\) command-line interface \(CLI\) tool](#). Then run the installer. You can use the tool to log in to Bluemix and interact directly with your account from your local environment. This tool puts many powerful commands at your disposal that you do not use in this scenario. More information is at the Cloud Foundry CLI [commands index](#) page.

The next item listed as a prerequisite is the Git command line client. We will use Github Desktop in most this scenario, but you could also the Git command line client to complete the same tasks. We will only use it to clone a starter template for the app. If you do not have Git installed, download it from [Git](#) and run the installer accepting the default options.

Follow these steps:

1. Clone the repo. Download the template for your app on your local development environment using Git. Rather than cloning the sample app from Bluemix, use the command in Example 5-1 to clone the starter template for the IBM COS Web Gallery app. After cloning the repo you will find the starter app in the `COS-WebGalleryStart` directory. Open a Git CMD window and change to a directory where you want to clone Github repo. Use the command shown in Example 5-1.

Example 5-1 Clone the COS Web Gallery repo to get the starter app

```
git clone https://github.com/IBMRedbooks/IBMRedbooks-SG248385-Cloud-Object-Storage-as-a-Service.git
```

2. Run the app locally. Open a CLI and change your working directory to the `COS-WebGalleryStart` directory. Notice the Node.js dependencies listed in the `package.json` file. Download them using the command shown in Example 5-2.

Example 5-2 Install app dependencies

```
npm install
```

Run the app using the command shown in Example 5-3.

Example 5-3 Starting the app locally

```
npm start
```

Open a browser and view your app on the address and port that is output to the console, `http://localhost:3000`.

Tip: To restart the app locally, kill the node process (Ctrl+C) to stop it, and use **npm start** again. Using nodemon to restart your app when it detects a change saves you time. Install nodemon globally like this: **npm install -g nodemon**. Then run it from the command line in your app directory using: **nodemon**, to have nodemon start your app.

3. Prepare the app for deployment. Update the application name property value in the `manifest.yml` file from COS-WebGallery, to the name you entered for your app on Bluemix. The COS-WebGallery `manifest.yml` looks like Example 5-4. Also update the `package.json` file located in the app root directory for your app to reflect the name of your app, and your name as the author.

Example 5-4 Update manifest.yml

```
---
applications:
- name: COS-WebGalery
  random-route: true
  memory: 256M
```

4. Deploy the app to Bluemix. To get the starter app with your changes to Bluemix, deploy it using the Cloud Foundry CLI:
 - a. Set the API Endpoint for your region by using the **api** command (Example 5-5). if you do not know your regional API endpoint URL see the **Getting Started** page.

Example 5-5 Set Bluemix API endpoint

```
cf api <API Endpoint>
```

- b. Log in to Bluemix by using the **login** command (Example 5-6). You can specify optional parameters if you want: your organization with option **-o**, and the space with option **-s** (Example 5-8).

Example 5-6 Login to Bluemix

```
cf login
```

- c. Deploy the app to Bluemix with the **push** command (Example 5-7).

Example 5-7 Deploy the app from the app directory

```
cf push
```

Example 5-8 shows what we used to deploy the COS-WebGallery app.

Example 5-8 Commands used to deploy COS-WebGallery

```
cf api https://api.ng.bluemix.net
cf login -u myaccount@us.ibm.com -o "IBM Redbooks" -s scenarios
cf push
```

If successful, Cloud Foundry reports that the app was uploaded, successfully deployed, and started. If you are logged in to the Bluemix web console, you are notified there also of the status of your app (Figure 5-12).

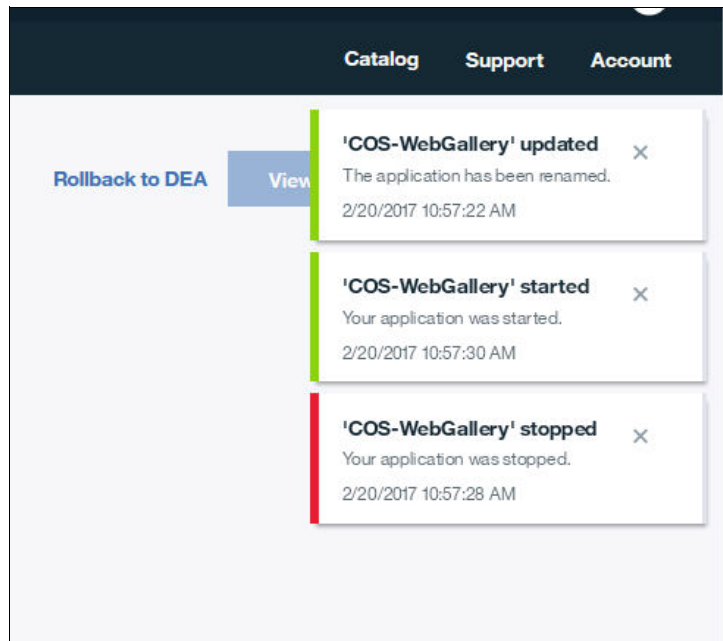


Figure 5-12 Notifications from the Bluemix Console

You can verify that the app was deployed by visiting the app URL reported by Cloud Foundry with a browser, or from the Bluemix web console by clicking **View App** button. The sample

5. Test the app. The visible change from the default app template that was deployed at creation (Figure 5-11 on page 115) to the starter app shown in Figure 5-13 proves that deploying the app to Bluemix was successful.

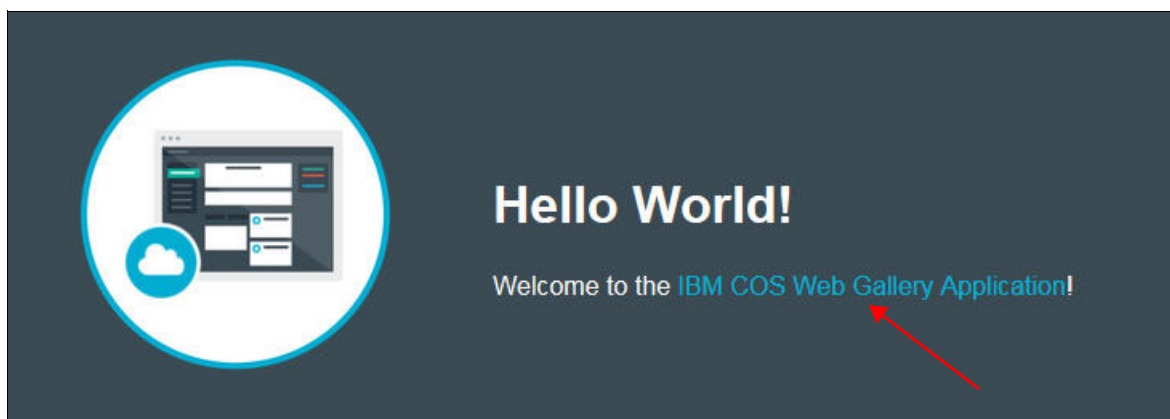


Figure 5-13 The redeployed COS Web Gallery app with the changes

5.3.7 Creating a Bluemix toolchain

You are almost ready to start working on the Web Gallery app, but before you start coding you must have a source repository for the code. It must be accessible from both Bluemix and the local development environment. We will want to both push changes to Bluemix, and pull down the changes made in the Bluemix cloud to our local development environment. To do so, create a Delivery Pipeline for the app in Bluemix by completing the following steps:

1. After signing in to IBM Bluemix, select the COS-WebGallery app, and from the app Overview window, scroll to **Continuous delivery** and click **Enable** (Figure 5-14).

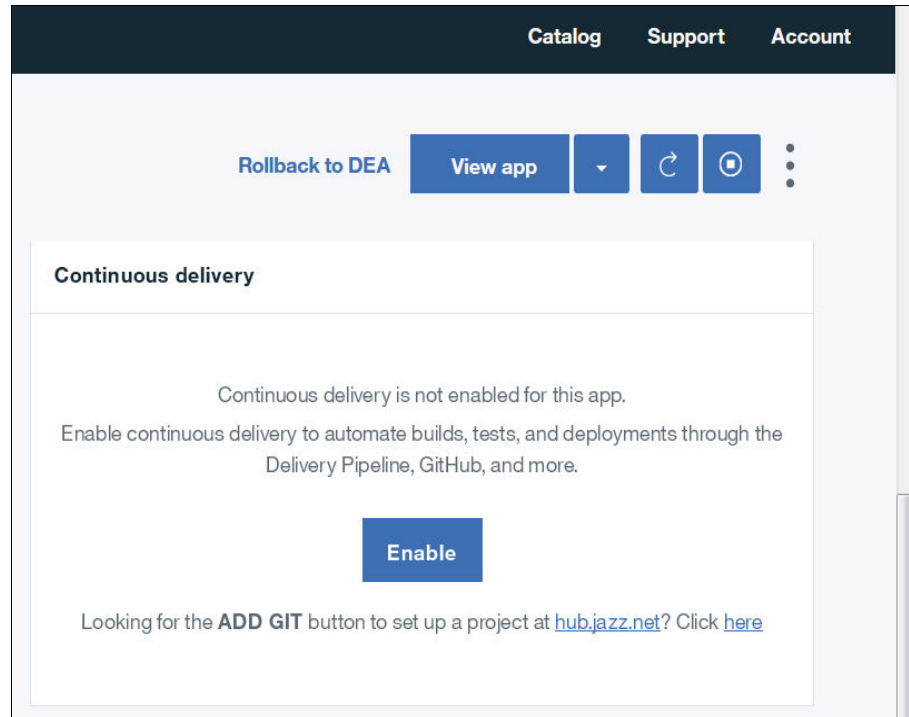


Figure 5-14 Enable Continuous delivery for COS-WebGallery

2. Set up the Toolchain Integrations. Scroll down to see the information that is shown in Figure 5-15. The Continuous Delivery Toolchain template creates a GitHub repository, Delivery Pipeline, and integrates Eclipse Orion Web IDE to allow the editing of your app code in the cloud. Everything is populated with the necessary values to create the toolchain.

Click **Create**.

Organization: IBM Redbooks / Toolchain Name: COS-WebGallery

Tool Integrations

GitHub

Eclipse Orion Web IDE

Delivery Pipeline

Store your source code in a new or existing repository on GitHub.com and engage in social coding through wikis, issue tracking, and pull requests.

Repository type:
New

Create an empty repository:
☐

New repository name:
COS-WebGallery

☒ Enable GitHub Issues ⓘ

Create

Figure 5-15 Create the toolchain

The Bluemix Toolchain is now set up (Figure 5-16 on page 120).

Tip: If you do not have an active GitHub session, you are prompted to log in. Click **Authorize Application** to allow Bluemix to access your GitHub account. If you have an active GitHub session but you have not entered your password recently, you might be prompted to enter your GitHub password to confirm. If Bluemix cannot access the GitHub repo, the Build Stage of your Delivery Pipeline will be unable to use it as input.

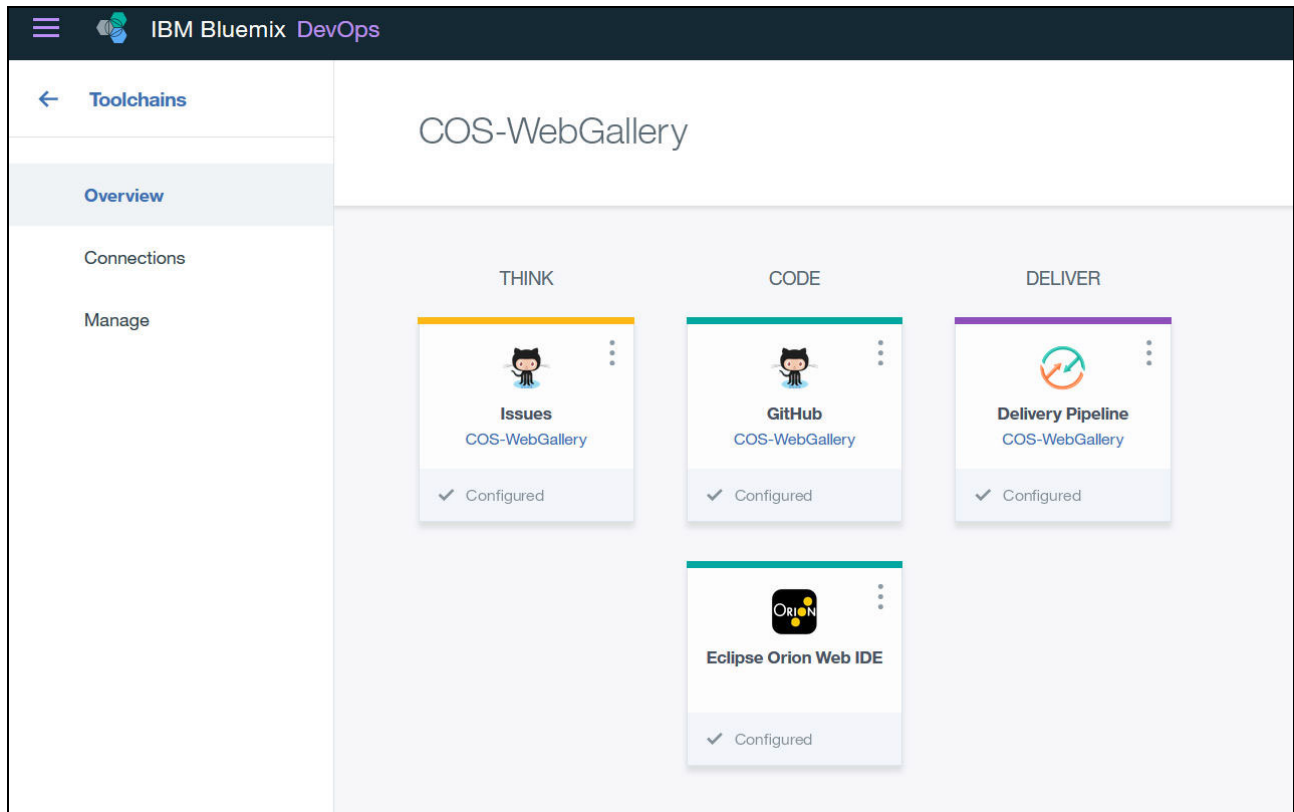


Figure 5-16 IBM Cloud Object Storage Web Gallery app toolchain

3. Click **GitHub** Tool to open the new repo created by your toolchain on GitHub.

4. It is currently empty so you must add app code from your local development environment, but first you need to clone the empty repo. To do so, you have options as shown in Figure 5-17.

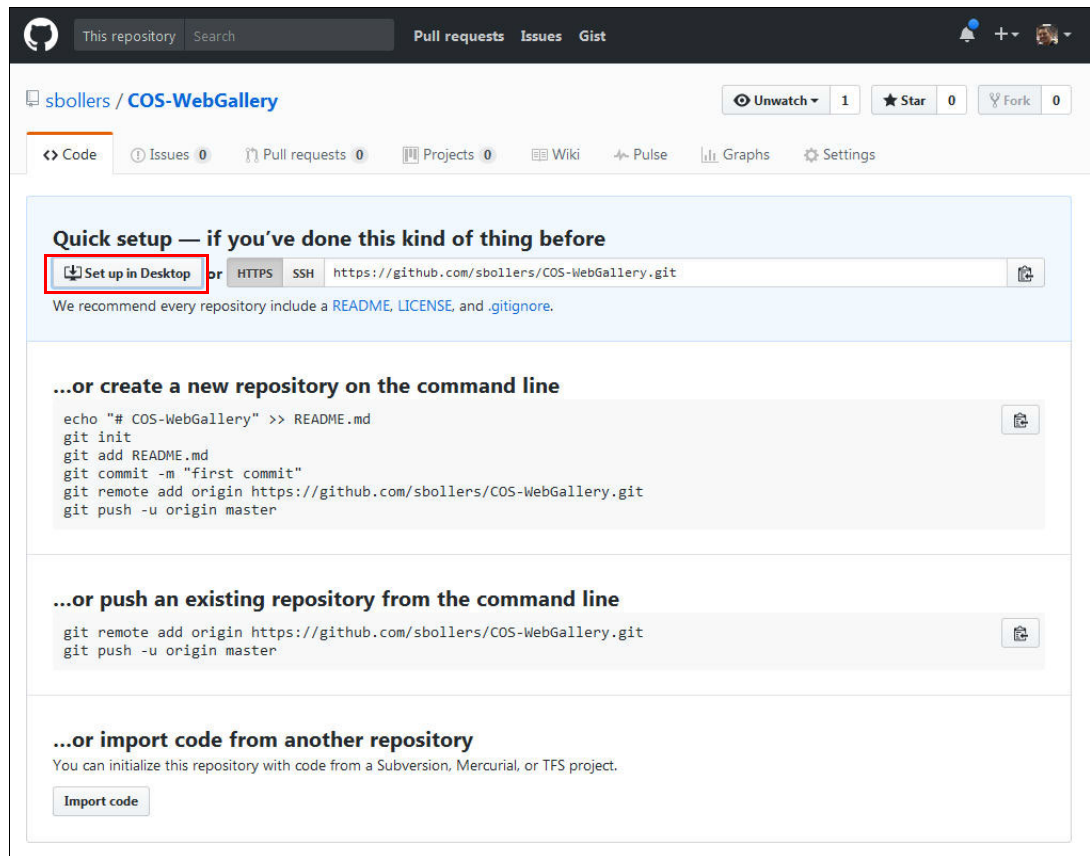


Figure 5-17 New GitHub created by Toolchain setup

This example uses the Quick setup option. Click **Set up in Desktop**. Allow GitHub desktop to open the link, and select an empty directory as the location for your new local repo. You now have a directory named the same as your app with nothing except the .git directory inside. In this example, it is named COS-WebGallery.

5. Copy the files and directories from the starter app you modified in step 3 on page 116. It looks like Figure 5-18 on page 122.

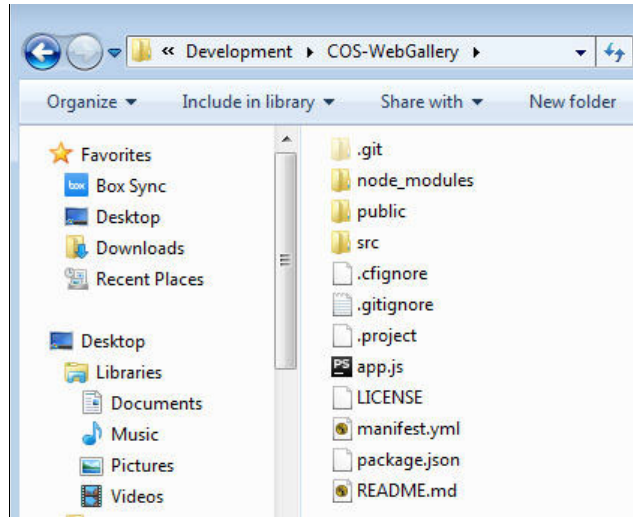


Figure 5-18 Local repo directories

- Return to GitHub Desktop and notice it detected what you added to the repo directory (Figure 5-19). Type `initial commit` into the summary field, and click **Commit to master**.

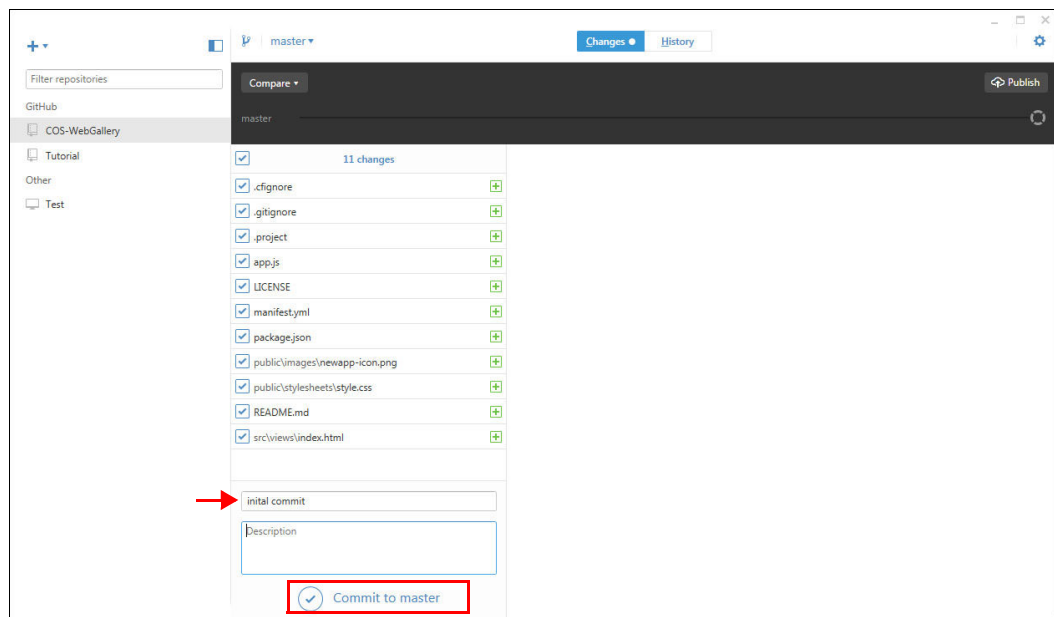


Figure 5-19 Commit initial changes to GitHub

Create a Git branch

Now, you need to create a branch for the local development environment to use for your Bluemix Delivery Pipeline Build Stage:

- Click the **branch icon**; you are prompted to enter a name for the branch (Figure 5-20 on page 123). This example uses `Local-dev` as the name.

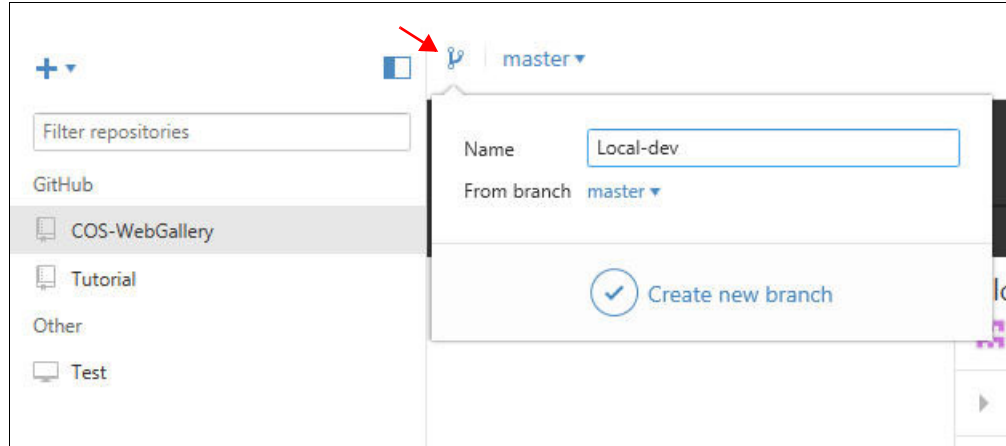


Figure 5-20 Create a local development branch

2. After you create the branch, GitHub compares the local files on the Local-dev branch with the files in the repository on the master branch and reports No local changes. You can now click **Publish** to add the branch you created on your local repo to your GitHub repo (Figure 5-21).

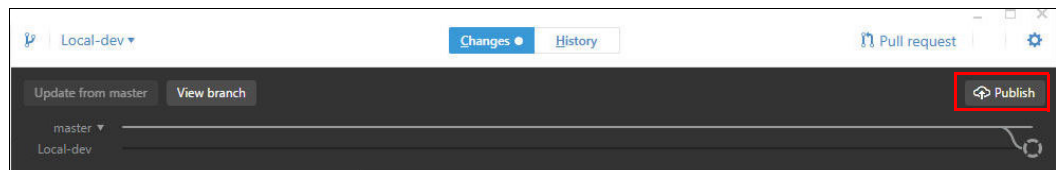


Figure 5-21 Publish Local-dev branch

Now that the Local-dev branch is published to the GitHub repo in your toolchain, the build stage of your Bluemix Delivery Pipeline will be triggered followed by the deploy stage anytime you push a commit to it. Deploying the app from the Cloud Foundry CLI will no longer be necessary.

5.3.8 Setting up IBM Cloud Object Storage credentials

You need to setup IBM COS credentials for the application, and a bucket where it will store and retrieve images. For a reminder of how to add credentials, see 3.2, “Credentials and endpoints” on page 28. After you have them, complete the following steps:

1. On the local development environment, place the credentials in the Windows path %USERPROFILE%\aws\credentials (for Mac/Linux, they go into ~/.aws/credentials).

Example 5-9 shows the contents of a credentials file.

Example 5-9 IBM Cloud Object Storage local credentials file

```
[default]
aws_access_key_id = XXXXXXXXXXXXXXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXX
```

2. In Bluemix, set up the credentials as environment variables by logging in to Bluemix, and under Cloud Foundry Apps, select the app **COS-WebGallery**. From the app menu, click **Runtime**.

3. In the Runtime window, click **Environment variables** at the top of the page and scroll to the User-defined section, which allows you to add the variables.
4. Add two variables: `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. These variables and their respective values are what the app uses to authenticate to the IBM COS instance when running on Bluemix (Figure 5-22). When you finish with the entries, click **Save**, and Bluemix restarts the app.

NAME	VALUE	ACTION
AWS_ACCESS_KEY_ID	XXXXXXXXXXXXXXXXXXXX	
AWS_SECRET_ACCESS_KEY	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	

Figure 5-22 Add IBM COS keys as variables

Add a bucket to IBM COS to contain your images (see 3.3.1, “Creating and managing buckets from Bluemix Infrastructure” on page 30). This scenario uses the bucket named `web-images`.

5.3.9 Developing a simple IBM Cloud Object Storage Web Gallery

Because this examples uses an MVC architecture, adjusting the directory structure to reflect architecture is necessary. The directory structure has a `views` directory to contain the EJS view templates, a `routes` directory to contain the express routes, and a `controllers` directory as the place to put the controller logic. Place these items under a source directory that is named `src` (Figure 5-23).

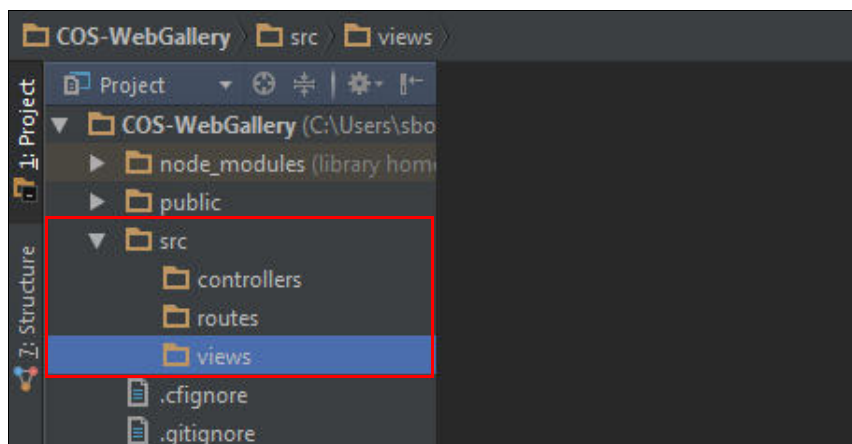


Figure 5-23 New directory structure

Tip: The repo you cloned in 5.3.6, “Creating the Web Gallery app on IBM Bluemix” on page 113 contains a directory named `COS-WebGalleryEnd`. Viewing the source code in your preferred editor might be helpful as you follow the next steps. This will be the version of the `COS-WebGallery` app that is committed and deployed to Bluemix in 5.3.10, “Committing to Git” on page 133.

Designing the app

These are the two main tasks that a user should be able to do with the simple web app:

- ▶ Upload images from a web browser to the IBM COS bucket.
- ▶ View the images in the IBM COS bucket in a web browser.

The next steps focus on how to accomplish these two basic functions in a simple fashion rather than building a fully developed production grade app.

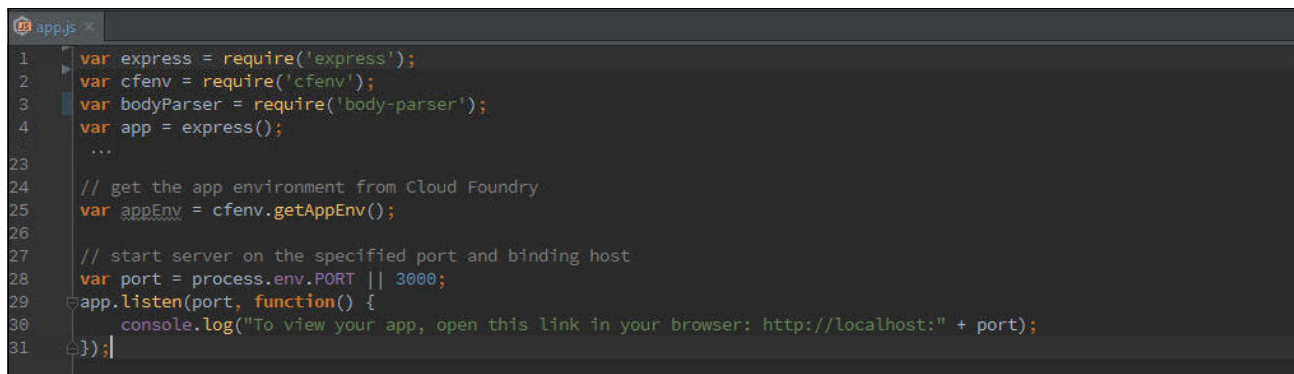
Developing the app

Look at the main application file, which is `app.js`. This is the code that we have told Node.js to process first when you start your app with the `npm start` command (or `nodemon`). In the `package.json` file, inside the `scripts` object, you see how "start" is defined (Example 5-10). This file is what Bluemix uses to tell node to run `app.js` each time the app starts. Also use it when testing the app locally.

Example 5-10 Package.json scripts object

```
...  
"scripts": {  
  "start": "node app.js"  
},  
...
```

Figure 5-24 shows the beginnings for the application in `app.js`. Lines 1 - 3 tell the node to load modules that are need to get started. Line 4 creates the express app by using the express module. Line 25 gets the Cloud Foundry environment object. Lines 28 - 32 tell the express app to listen on the port that is assigned to the port property. We print a message with the server URL to the console.



```
1  var express = require('express');  
2  var cfenv = require('cfenv');  
3  var bodyParser = require('body-parser');  
4  var app = express();  
5  ...  
23  
24  // get the app environment from Cloud Foundry  
25  var appEnv = cfenv.getAppEnv();  
26  
27  // start server on the specified port and binding host  
28  var port = process.env.PORT || 3000;  
29  app.listen(port, function() {  
30    console.log("To view your app, open this link in your browser: http://localhost:" + port);  
31  });
```

Figure 5-24 Beginning and ending of `app.js`

Figure 5-25 on page 126 shows how to define a path and views. Line 7 tells the express app to use the `public` directory to serve our static files, which include any static images and style sheets we use. Lines 8 - 9 tells the express app where to find the view templates for our views in the `src/views` directory, and set our view engine to be EJS. Line 10 tells express to use the `body-parser` middleware to expose incoming request data to the app as JSON. In lines 12 - 16, the express app responds to all incoming `GET` requests to our app URL by rendering the `index.ejs` view template.

```

6 // serve the files out of ./public as our main files
7 app.use(express.static('public'));
8 app.set('views', './src/views');
9 app.set('view engine', 'ejs');
10 app.use(bodyParser.json());
11
12 var title = 'Simple COS Web Gallery';
13 // Serve index.ejs
14 app.get('/', function (req, res) {
15   res.render('index', {status: '', title: title});
16 });
17

```

Figure 5-25 Define a static file path and views to be served by the app

Figure 5-26 shows what the index view template when rendered and sent to the browser.

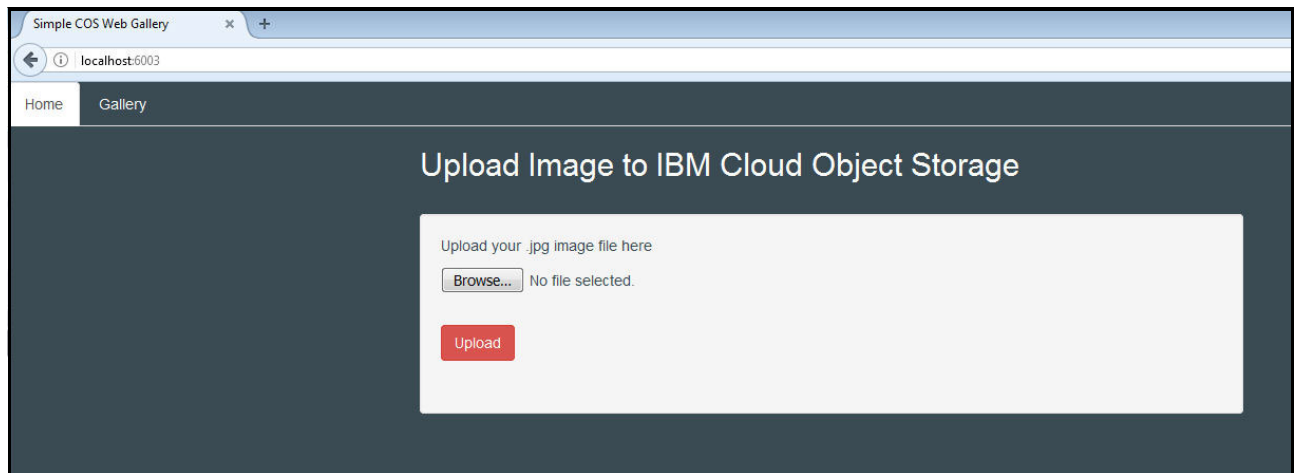


Figure 5-26 Rendered index view

In this example, our view templates share HTML code between the `<head>...</head>` tags, so we placed it into a separate `include` template (Figure 5-27). This template (`head-inc.ejs`) contains a scriptlet for the page title on line 1. The title variable is being set in `app.js` on line 12, and passed in as data for our view template on line 15. Otherwise, we are simply using some CDN addresses to pull in Bootstrap CSS, Bootstrap JavaScript, and JQuery. We use a static `styles.css` file from our `pubic/style` sheets directory.

```

1 <title><%=title%></title>
2 <meta charset="utf-8">
3 <meta http-equiv="X-UA-Compatible" content="IE=edge">
4 <meta name="viewport" content="width=device-width, initial-scale=1">
5 <!-- Latest compiled and minified CSS -->
6 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
7   integrity="sha384-BVYiISIFeK1dGmJRAKycuHAHRg320mUcw7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
8   crossorigin="anonymous">
9 <script src="https://code.jquery.com/jquery-3.1.1.min.js"
10   integrity="sha384-hVVnYaiADRTO2PzUGmuLJR8BLUSjGIZsDYGMiJLv2b8="
11   crossorigin="anonymous">
12 </script>
13 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
14   integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWvXZxUPnCJA7L2mCWNIPG9mGCD8wGNICPD7Txa"
15   crossorigin="anonymous">
16 </script>
17
18 <link rel="stylesheet" href="stylesheets/style.css">

```

Figure 5-27 The `head-inc.view` template

The body of the index view (Figure 5-28), contains our bootstrap styled navigation tabs, and our upload form in a basic bootstrap. Consider these two notes:

- ▶ We set our form method to **POST** and the form-data encoding type as `multipart/form-data` on line 24. For the form action, we send the data from our form to the app to the app route `/`. Later we do additional work in our router logic to handle **POST** requests to that route.
- ▶ We want to display feedback about the status of the attempted file upload to the user. This feedback is passed to our view in a variable named `"status"`, and is displayed below the upload form on line 31.

```

4  <head>
5    <%- include('head-inc'); %>
6  </head>
7
8  <body>
9    <ul class="nav nav-tabs">
10     <li role="presentation" class="active"><a href="/">Home</a></li>
11     <li role="presentation"><a href="/gallery">Gallery</a></li>
12   </ul>
13   <div class="container">
14     <h2>Upload Image to IBM Cloud Object Storage</h2>
15     <div class="row">
16       <div class="col-md-12">
17         <div class="container" style="...">
18           <div class="row">
19             <div class="col-lg-8 col-md-8 well">
20               <p class="wellText">Upload your .jpg image file here</p>
21               <form method="post" enctype="multipart/form-data" action="/">
22                 <p><input class="wellText" type="file" size="100px" name="img-file" /></p>
23                 <br/>
24                 <p><input class="btn btn-danger" type="submit" value="Upload" /></p>
25               </form>
26               <br/>
27               <span class="notice"><%=status%></span>
28             </div>
29           </div>
30         </div>
31       </div>
32     </div>
33   </div>
34 </body>

```

Figure 5-28 Body of the index view template

Figure 5-29 returns to `app.js`. Lines 18 - 19 sets up express routes to handle additional requests that will be made to our app. The code for these routers will be in two files under the `./src/routes` directory:

- ▶ `imageUploadRoutes.js`: This file handles what happens when the user selects an image and clicks **Upload**.
- ▶ `galleryRoutes.js`: This file handles requests when the user clicks the **Gallery** tab to request the `imageGallery` view.

```

17
18 var imageUploadRouter = require('./src/routes/imageUploadRoutes')(title);
19 var galleryRouter = require('./src/routes/galleryRoutes')(title);
20
21 app.use('/gallery', galleryRouter);
22 app.use('/', imageUploadRouter);
23

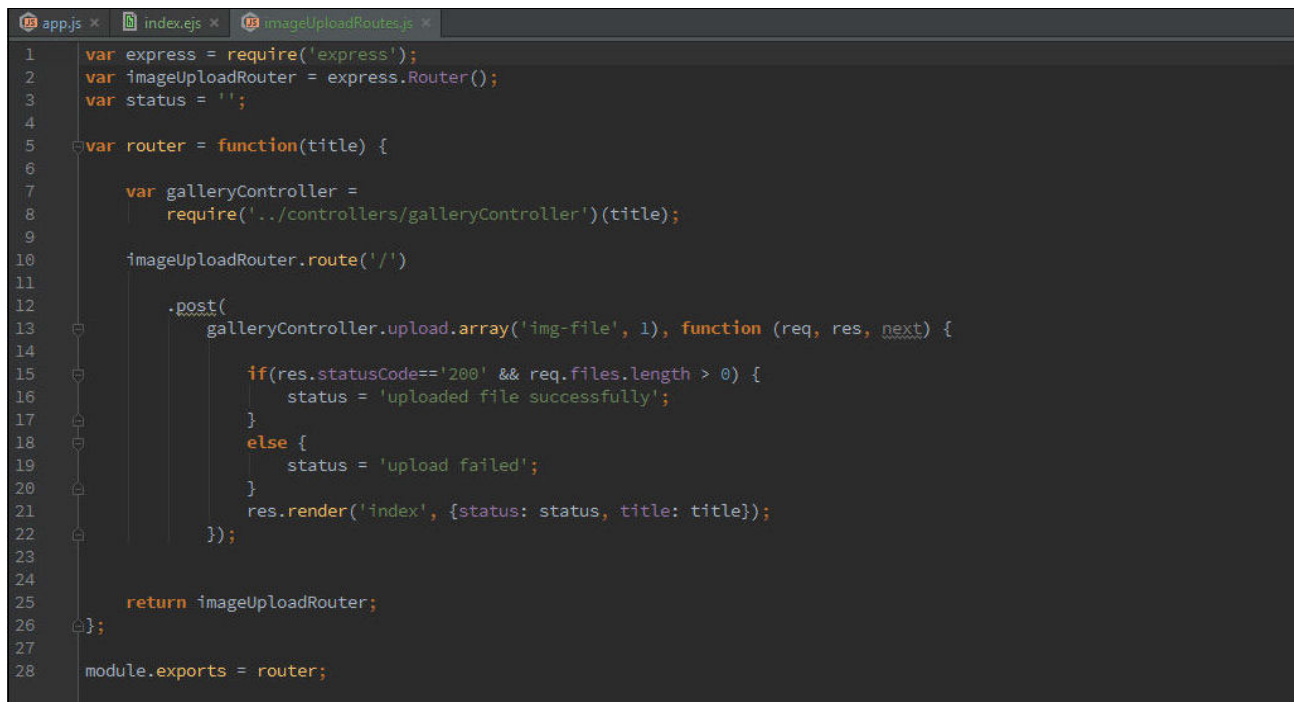
```

Figure 5-29 Define which router to use for each path

Image upload

See `imageUploadRoutes.js` in Figure 5-30. We must create an instance of a new express router and name it `imageUploadRouter` in lines 1 - 2. Then, on line 5, we create a function that returns `imageUploadRouter`, and assign it to a variable called "router". We export the function in "router" on line 28 to make it accessible to `app.js`. On line 7, we require a file named `galleryController.js`. Because some logic is dedicated to controlling how we upload our images, we put that logic in this function and save it in our `./src/controllers` directory.

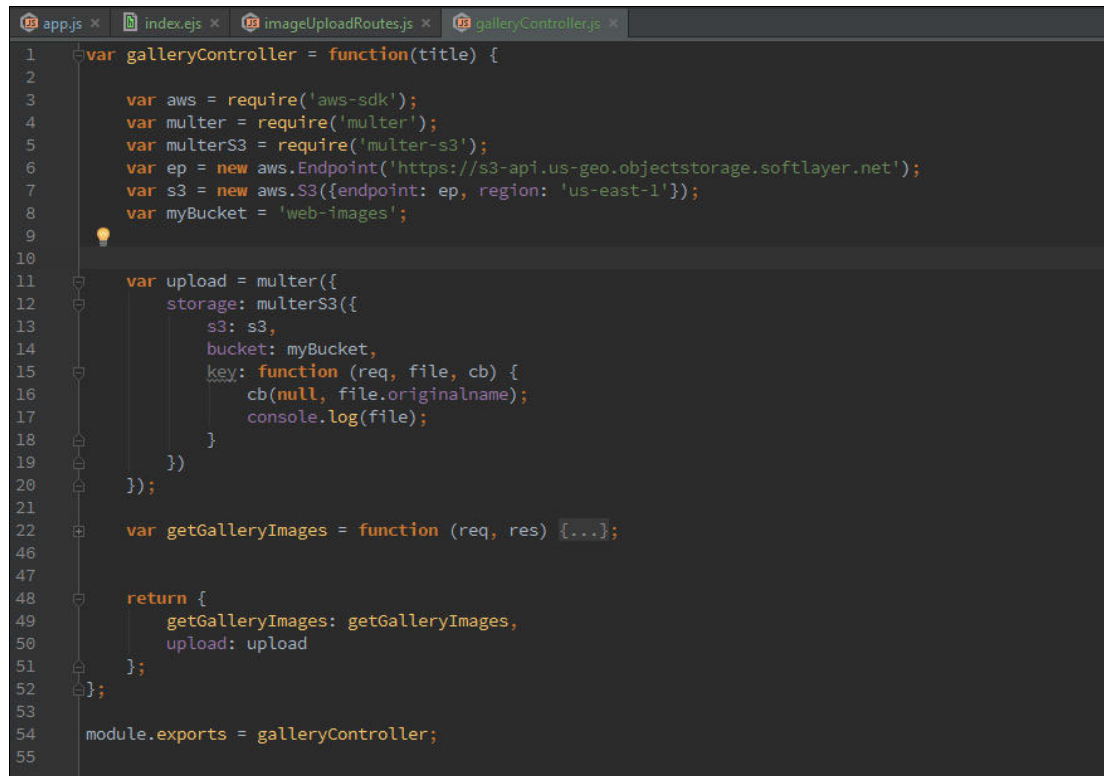
Line 12 is where our `imageUploadRouter` is told to route requests for the root app route ("/") when the HTTP **POST** method is used. Inside the `post` function of our `imageUploadRouter`, we use middleware from the `multer` and `multer-s3` modules which is exposed by the `galleryController` as `upload`. The middleware takes the data and file from our Upload form **POST**, processes it, and runs a callback function. In the callback function on line 13 - 22, we check that we get an HTTP status code of 200, and that we had at least one file in our request object to upload. Based on those conditions, we set the feedback in our status variable and render the index view template with the new status.



```
1  var express = require('express');
2  var imageUploadRouter = express.Router();
3  var status = '';
4
5  var router = function(title) {
6
7      var galleryController =
8          require('../controllers/galleryController')(title);
9
10     imageUploadRouter.route('/')
11
12         .post(
13             galleryController.upload.array('img-file', 1), function (req, res, next) {
14
15                 if(res.statusCode==='200' && req.files.length > 0) {
16                     status = 'uploaded file successfully';
17                 }
18                 else {
19                     status = 'upload failed';
20                 }
21                 res.render('index', {status: status, title: title});
22             });
23
24
25     return imageUploadRouter;
26 };
27
28 module.exports = router;
```

Figure 5-30 Express route for image upload form POST

Look at how we set up the `multer` upload in Figure 5-31 on page 129. We require modules `aws-sdk`, `multer`, and `multer-s3`. Lines 6 - 7 show how to configure an S3 object that points to an IBM COS server endpoint. We are statically setting values such as the endpoint address, region, and bucket for simplicity, but they could easily be referenced from an environment variable or JSON configuration file.



```

1  var galleryController = function(title) {
2
3      var aws = require('aws-sdk');
4      var multer = require('multer');
5      var multerS3 = require('multer-s3');
6      var ep = new aws.Endpoint('https://s3-api.us-gso.objectstorage.softlayer.net');
7      var s3 = new aws.S3({endpoint: ep, region: 'us-east-1'});
8      var myBucket = 'web-images';
9
10
11     var upload = multer({
12         storage: multerS3({
13             s3: s3,
14             bucket: myBucket,
15             key: function (req, file, cb) {
16                 cb(null, file.originalname);
17                 console.log(file);
18             }
19         })
20     });
21
22     var getGalleryImages = function (req, res) {...};
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48     return {
49         getGalleryImages: getGalleryImages,
50         upload: upload
51     };
52 };
53
54 module.exports = galleryController;
55

```

Figure 5-31 Upload function in the gallery controller

We define upload used by imageUploadRouter on line 11 by creating a new multer instance with a storage property on line 12. This property tells multer where to send the file from our multipart/form-data. Since IBM COS uses an implementation of the S3 API, we set storage to be an s3-multer object. This s3-multer object contains an s3 property that we have assigned to our s3 object from line 7, and a bucket property that we have assigned the myBucket variable from line 8, which is assigned a value of “web-images”. The s3-multer object now has all the data necessary to connect and upload files to our IBM COS bucket when it receives data from the upload form. The name or key of the uploaded object will be the original file name taken from the file object when it is stored in our IBM COS “web-images” bucket. For local testing, a helpful task is to print the file object to the console, on line 17.

We perform a local test of the Upload form and the output from the console log of the file (Example 5-11).

Example 5-11 Output from our console

```

{ fieldname: 'img-file',
  originalname: 'Chrysanthemum.jpg',
  encoding: '7bit',
  mimetype: 'image/jpeg' }

```

Figure 5-32 shows that feedback from our callback saying it was a successful upload.

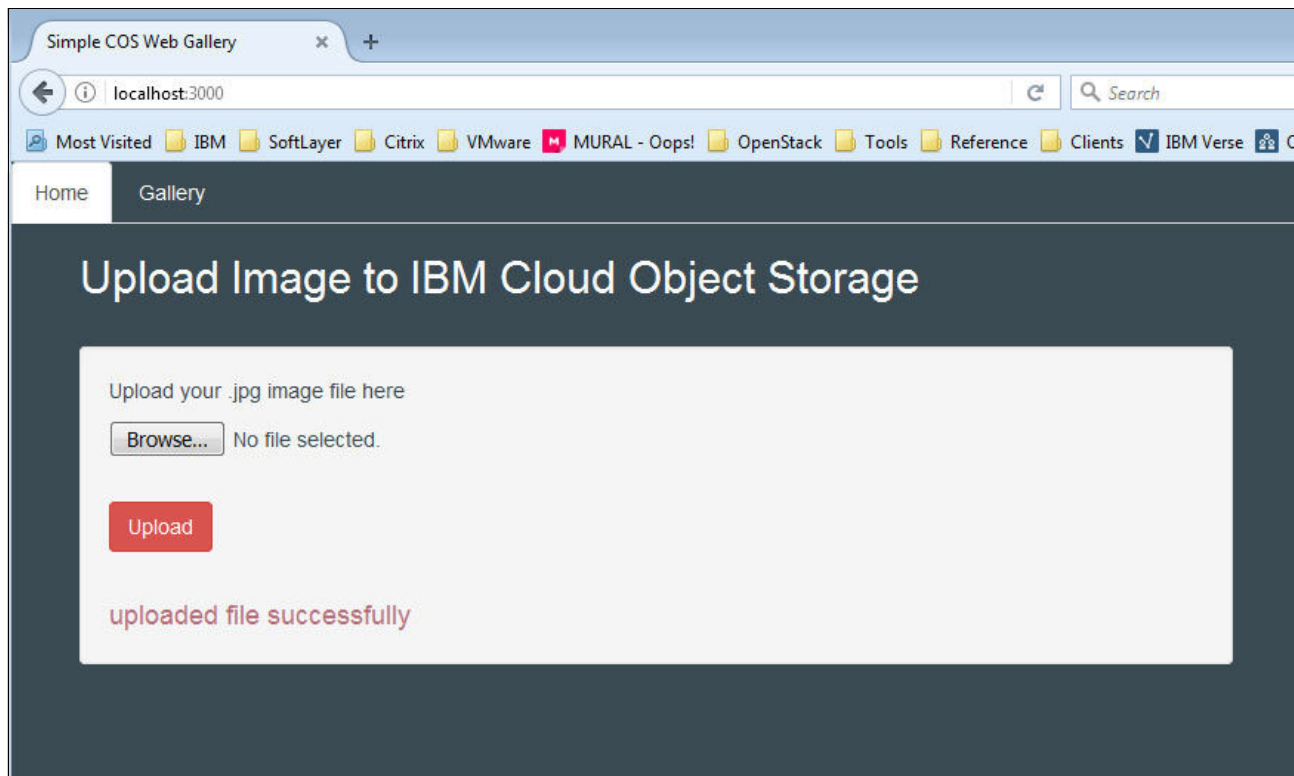


Figure 5-32 Local Upload form test

Image retrieval and display

Figure 5-33 refers to `app.js`. Line 19 creates `galleryRouter`, and tells express to use it when the `"/gallery"` route is requested. Look at the `galleryRoutes.js` file that is used to define `galleryRouter`.

```
17
18 var imageUploadRouter = require('./src/routes/imageUploadRoutes')(title);
19 var galleryRouter = require('./src/routes/galleryRoutes')(title);
20
21 app.use('/gallery', galleryRouter);
22 app.use('/', imageUploadRouter);
23
```

Figure 5-33 Defining which router to use for each path

Figure 5-34 on page 131 shows `galleryRoutes.js` where we define the express router assigned to `galleryRouter` on line 2. We follow the same pattern that we did with `imageUploadRouter` and require `galleryController` on lines 6 - 7, then set up our route on line 9. The main difference is we are routing HTTP `GET` requests rather than `POST`, and sending all the output in the response from `getGalleryImages`, which is exposed by the `galleryController` on line 10.


```

1  var express = require('express');
2  var galleryRouter = express.Router();
3
4  var router = function(title) {
5
6      var galleryController =
7          require('../controllers/galleryController')(title);
8
9      galleryRouter.route('/')
10         .get(galleryController.getGalleryImages);
11
12      return galleryRouter;
13  };
14  module.exports = router;

```

Figure 5-34 Routing requests for /gallery/ to galleryController

Referring to `galleryController.js` (Figure 5-35), we define the `getGalleryImages` function we just saw on line 22. Using the same S3 object that we set up for our image upload function, we call a function that named `listObjectsV2` on line 26. This function returns data of the objects in our bucket. To display images, we need an image URL for each JPEG image in our web-images bucket to display in our view template. The content on line 28 is an array map from the data object returned by `listObjectsV2` containing metadata about each object in our bucket. We loop the content and search for any object key ending in `.jpg`, and create a parameter to pass to the S3 `getSignedUrl` function. This function returns a signed URL for any object when we pass it the object's bucket name and key. In the callback function we save each URL in an array, and pass it `res.render` as `imageUrls`.

```

21
22  var getGalleryImages = function (req, res) {
23
24      var imageUrlList = [];
25      var params = {Bucket: myBucket};
26      s3.listObjectsV2(params, function (err, data) {
27          if(data) {
28              var bucketContents = data.Contents;
29
30              for (var i = 0; i < bucketContents.length; i++) {
31                  if(bucketContents[i].Key.search(/.jpg/i) > -1) {
32                      var urlParams = {Bucket: myBucket, Key: bucketContents[i].Key};
33
34                      s3.getSignedUrl('getObject', urlParams, function (err, url) {
35                          imageUrlList[i] = url;
36                      });
37                  }
38              }
39              res.render('galleryView', {
40                  title: title,
41                  imageUrls: imageUrlList
42              });
43          });
44      });
45  };
46
47  return {
48      getGalleryImages: getGalleryImages,
49      upload: upload
50  };
51  };
52

```

Figure 5-35 Retrieve the .jpg image URLs from IBM Cloud Object Storage

Figure 5-36 shows the galleryView EJS template body. We get the `imageUrls` array from the `res.render()` method and iterate over a pair of nested `<div></div>` tags where the image URL will make a **GET** request for the image when the `/gallery` route is requested.

```
app.js x index.ejs x imageUploadRoutes.js x galleryController.js x galleryView.ejs x
html body
1 <!DOCTYPE html>
2 <html>
3
4 <head...>
22
23 <body>
24 <ul class="nav nav-tabs">
25 <li role="presentation"><a href="/">Home</a></li>
26 <li role="presentation" class="active"><a href="/gallery">Gallery</a></li>
27 </ul>
28 <div class="container">
29 <h2>IBM COS Image Gallery</h2>
30
31 <div class="row">
32 <% for(var i=0; i < imageUrls.length; i++) { %>
33 <div class="col-md-4">
34 <div class="thumbnail">
35 
36 </div>
37 </div>
38 <% } %>
39 </div>
40 </div>
41 </body>
42
43 </html>
44
```

Figure 5-36 Iterate over the `imageUrls` array in the `galleryView` template

We test it locally from `http://localhost:3000/gallery` and see our image in Figure 5-37.

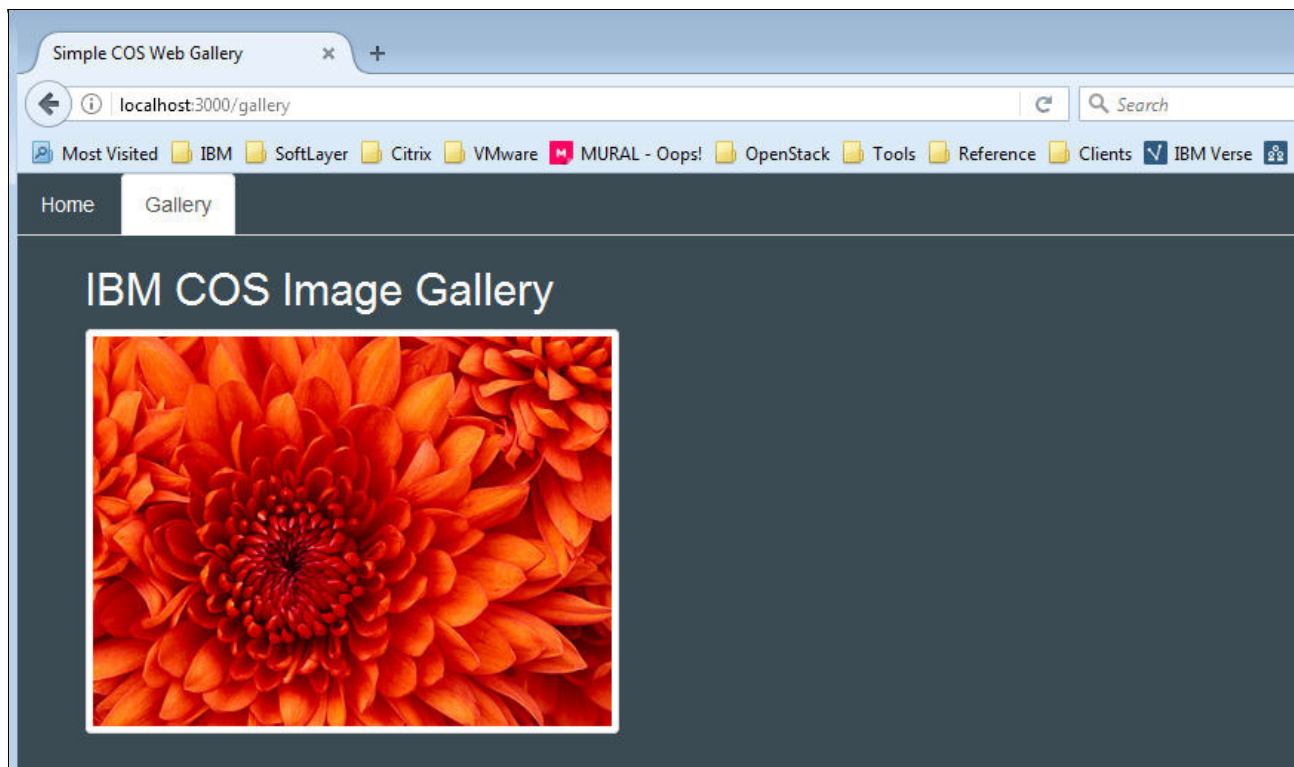


Figure 5-37 Local Gallery view test

5.3.10 Committing to Git

Now that the basic features of the app are working, we commit our code to our local repo, and push it to GitHub. Back in GitHub Desktop, we click **Changes**, type a summary of the changes in the Summary field, and then click **Commit to Local-dev** (Figure 5-38). When we click **Sync**, our commit is sent to the remote Local-dev branch that we published to GitHub, and this action starts the Build Stage followed by the Deploy Stage in our Delivery Pipeline (Figure 5-39 on page 134).

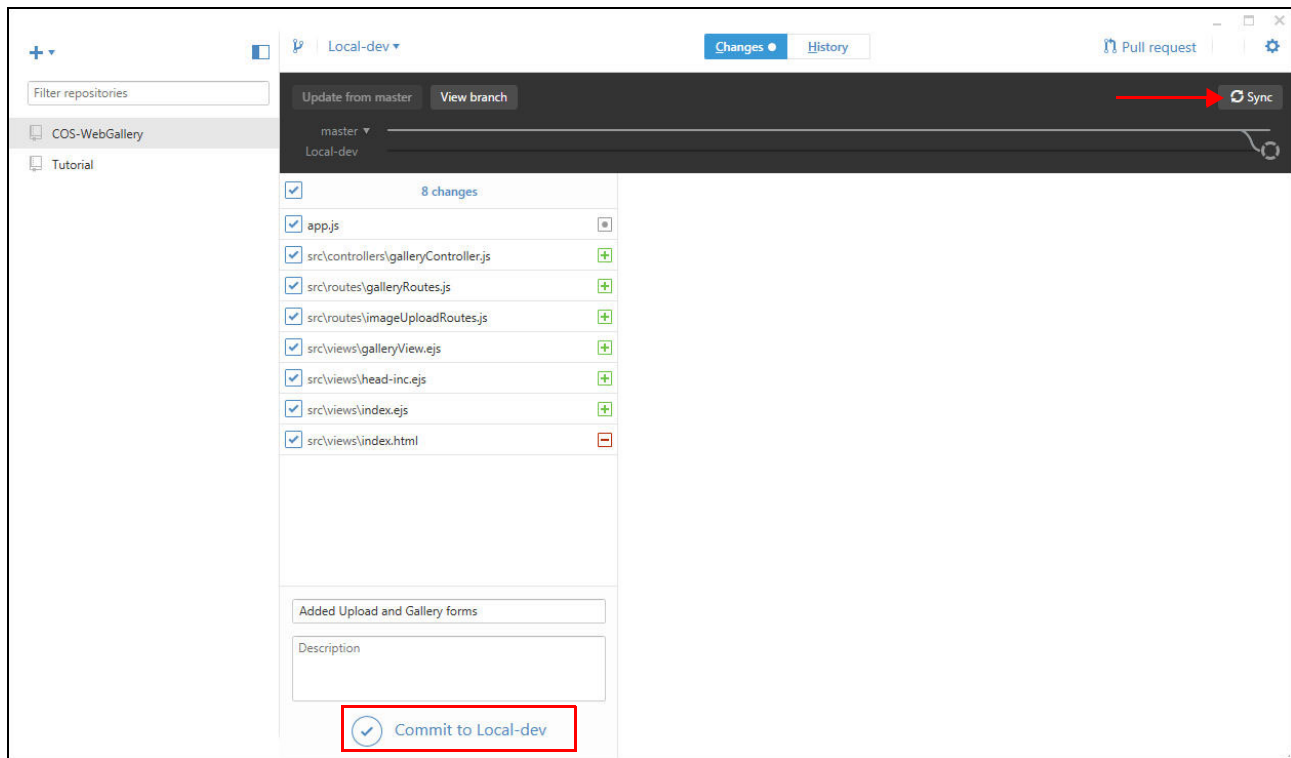


Figure 5-38 Commit changes to Local-dev repo

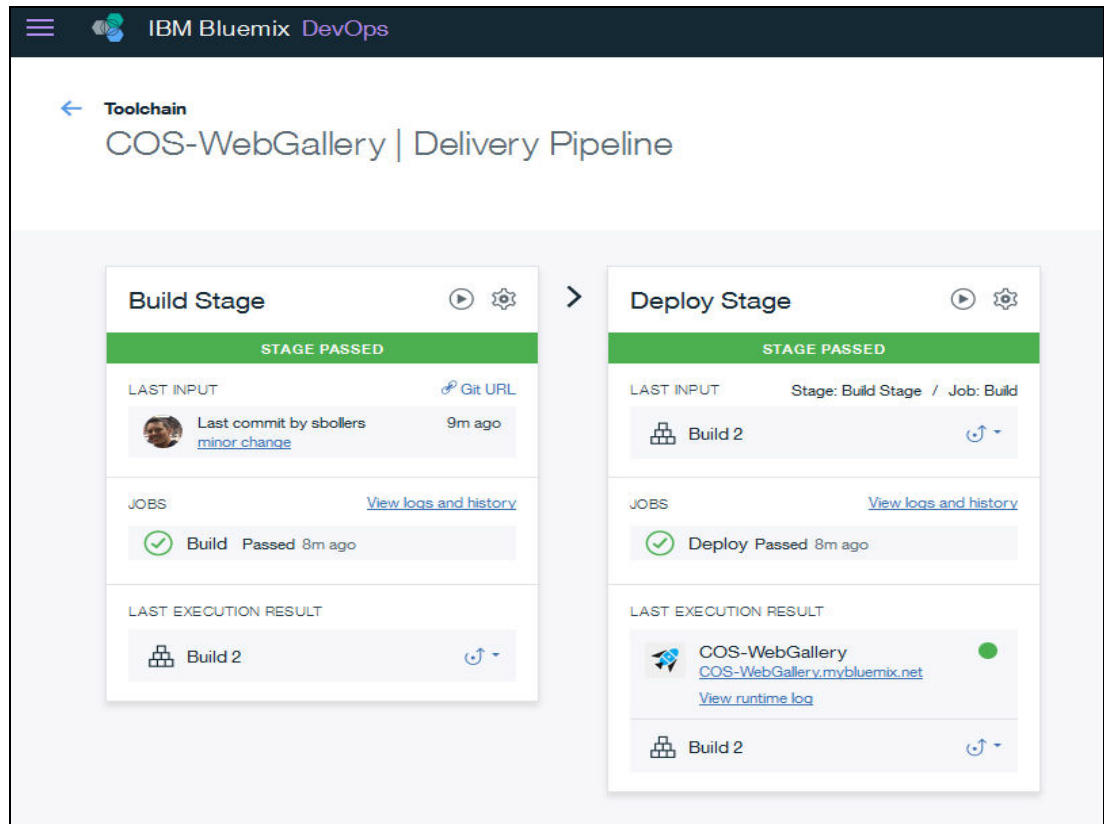


Figure 5-39 Stages of pipeline triggered after remote branch Sync

5.4 Verifying the scenario

This section describes a test run of the scenario to verify that it is working correctly.

Testing the app

The Updated COS-WebGallery app is now running on Bluemix. We verify that it can do both tasks that we designed it to do. After logging in to Bluemix, click **COS-WebGallery** under Cloud Foundry Apps and click **View app**. The upload form is displayed.

Image file upload test

Figure 5-40 on page 135 shows the file upload form in our COS-WebGallery index view when we request the app at <http://cos-webgallery.mybluemix.net>.

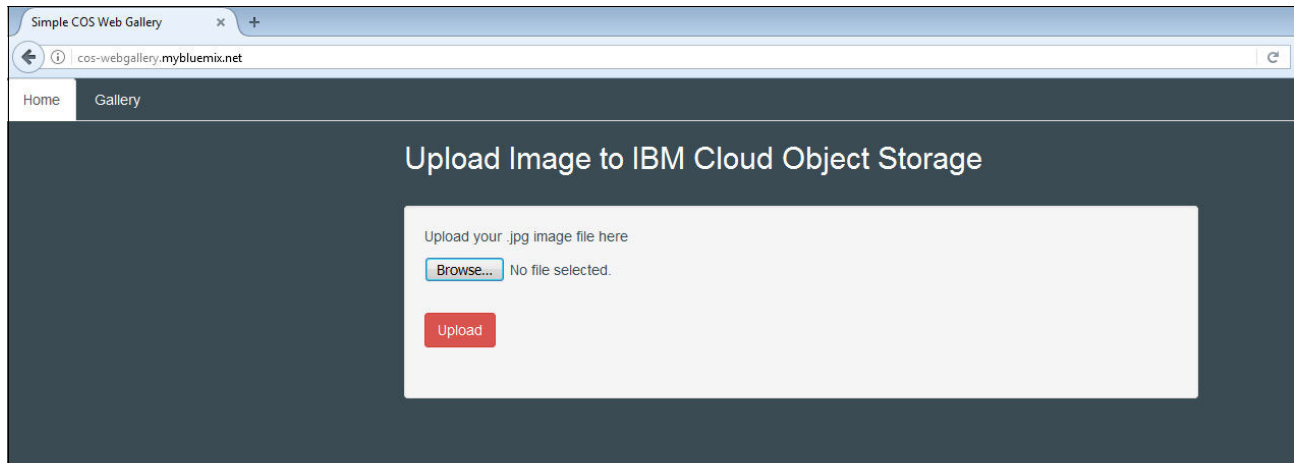


Figure 5-40 Upload form

Complete the following steps:

1. After you click **Upload**, select an image from the browser File Upload explorer (Figure 5-41).

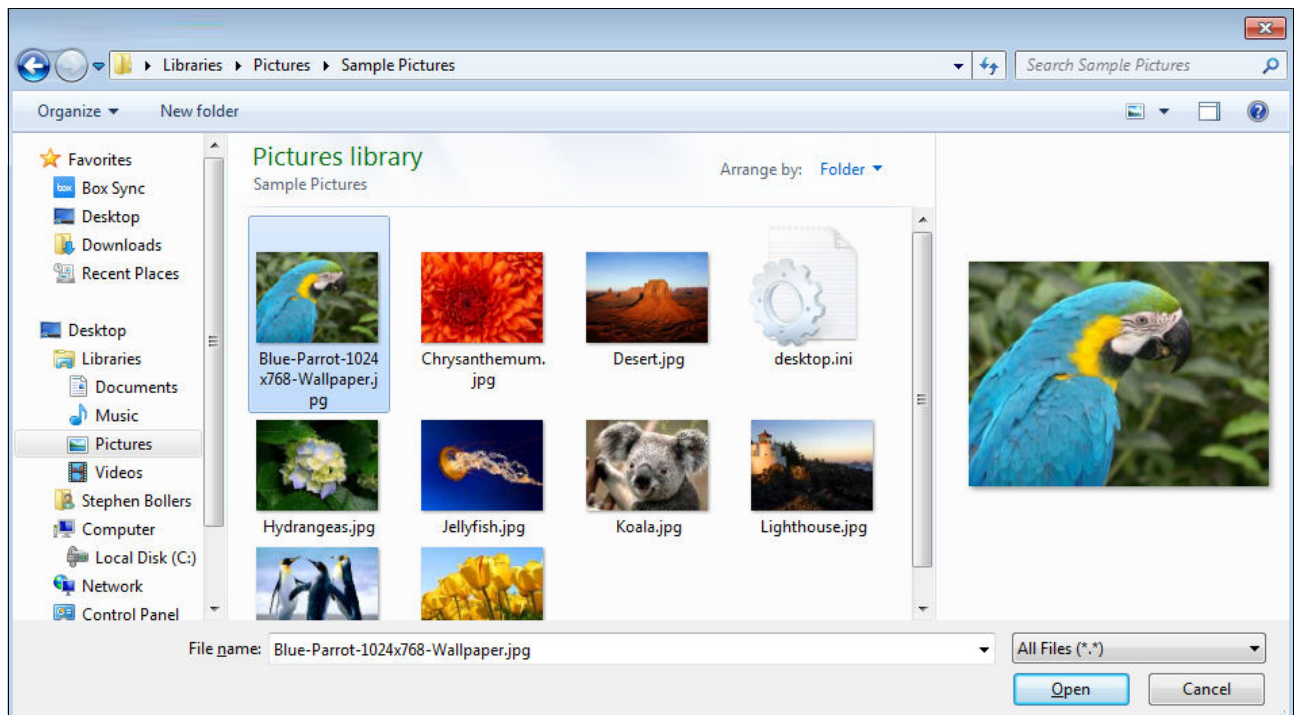


Figure 5-41 Select a JPEG image with File Upload explorer

Figure 5-42 on page 136 shows that the image is selected with the form and is ready to upload.

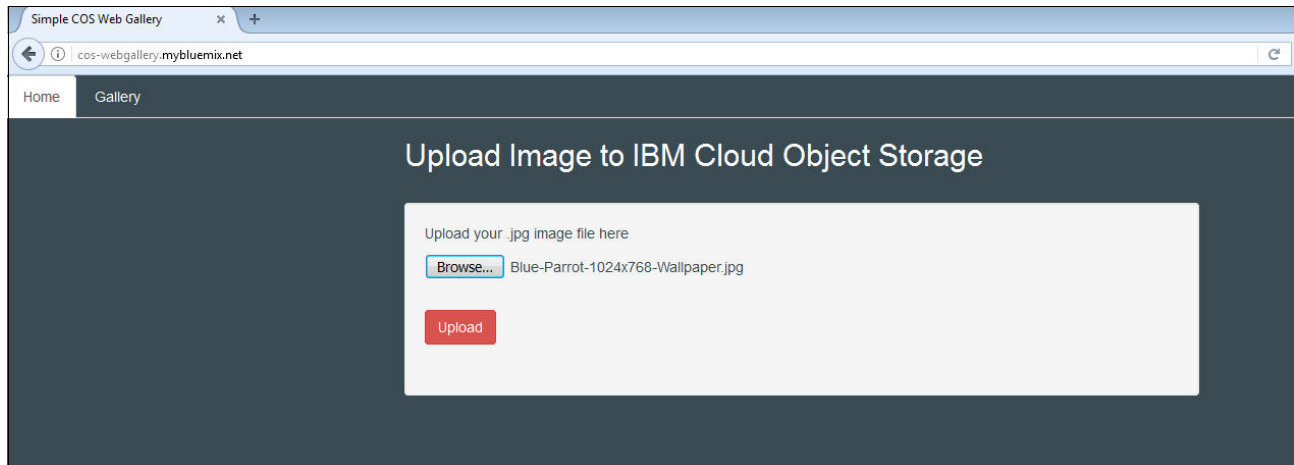


Figure 5-42 Image that is selected for upload

2. After clicking **Upload**, feedback indicates that the app uploaded the selected file successfully (Figure 5-43),

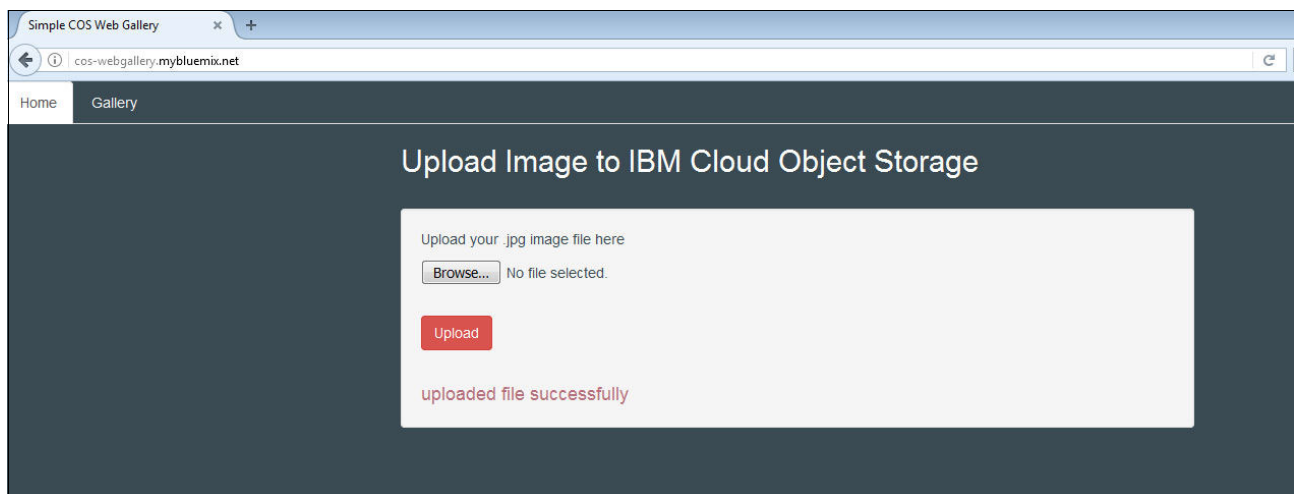


Figure 5-43 Feedback that is displayed after successful upload

3. Log in to Bluemix Infrastructure and click **Storage** → **Object Storage**. Click the COS Account Name, and then the bucket that is named web-images. Figure 5-44 shows the list of images in that bucket. The first image, Blue-Parrot-1024x768-Wallpaper.jpg, is the image that just uploaded in this test.

[Back to Bucket List](#) « 1 »

Object Name	Size	Last Modified	
(object name)			+
Blue-Parrot-1024x768-Wallpaper.jpg	203.25 KB	2017-02-24 01:23 am	+
Chrysanthemum.jpg	858.78 KB	2017-02-22 05:45 pm	+
Desert.jpg	826.11 KB	2017-02-24 01:17 am	+
Hydrangeas.jpg	581.33 KB	2017-02-16 09:45 pm	+
Jellyfish.jpg	757.52 KB	2017-02-17 11:33 am	+
Koala.jpg	762.53 KB	2017-02-16 04:37 pm	+
Lighthouse.jpg	548.12 KB	2017-02-24 01:21 am	+
Penguins.jpg	759.60 KB	2017-02-17 11:49 am	+
Tulips.jpg	606.34 KB	2017-02-22 05:17 pm	+

Figure 5-44 IBM Cloud Object Storage web-images bucket object listing

Gallery View test

Figure 5-45 shows the images in our IBM COS bucket that are displayed in the galleryView when we request <http://cos-webgallery.mybluemix.net/gallery>.

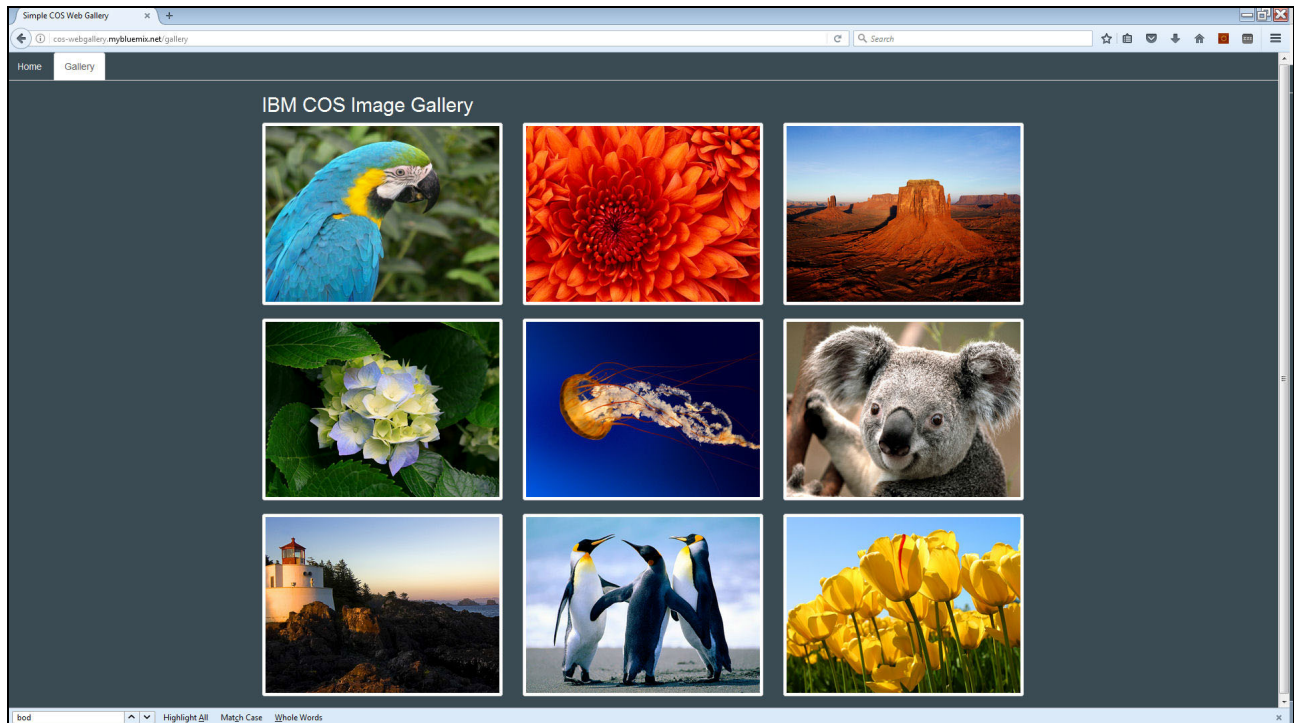


Figure 5-45 Rendered galleryView displaying images from IBM COS

Upon closer inspection of the HTML source of Figure 5-46 on page 138, you can see that the `img src` value is the following string:

```
https://web-images.s3-api.us-geo.objectstorage.softlayer.net/Chrysanthemum.jpg?AWS  
AccessKeyId=MjWD11Y0eH9cugeNaCiz&Expires=1487921976&Signature=qItMGrxRVbtIMmYfCxCu  
EIj5XkM%3D.
```

We retrieved the image object from our IBM COS web-images bucket.

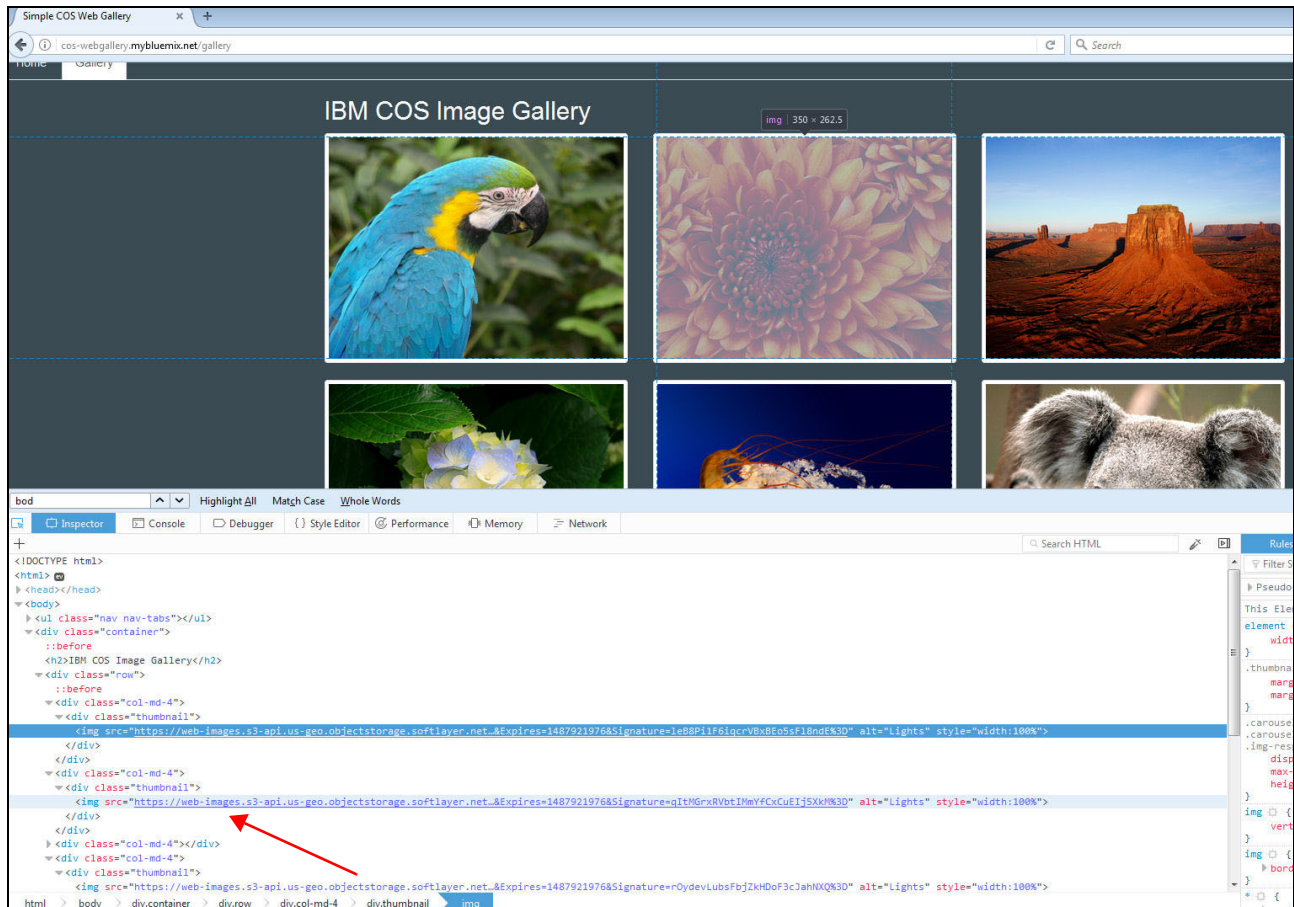


Figure 5-46 Inspect the IBM Cloud Object Storage Web Gallery image URL

This chapter described these topics:



Simple gateway scenario

This chapter describes the configuration steps and architecture to configure a basic file interface on a commodity Linux system to push and pull data from IBM COS.

The following topics are covered in this chapter:

- ▶ 6.1, “Scenario description” on page 140
- ▶ 6.2, “Architecture of the scenario” on page 140
- ▶ 6.3, “Implementation of the scenario” on page 141
- ▶ 6.4, “Verification of the scenario” on page 147
- ▶ 6.5, “Conclusion” on page 149

6.1 Scenario description

Many businesses and applications rely on file system interfaces as a means to store data. In this scenario, a Linux server is set up to mount the IBM COS bucket as a file system directly. This type of solution is appropriate where the following conditions are met:

- ▶ Organizations that are comfortable with managing Linux and addressing any issues through online communities.
- ▶ Data set being dumped does not contain thousands of small files, or very large files.
- ▶ Application will not be impacted by speed or latency of this approach.
- ▶ Application cannot integrate with one of the object utilities that push and pull data from IBM COS such as those mentioned in 3.5, “Clients, CLIs, and SDKs” on page 40.
- ▶ Limited or no budget exists for commercial gateway offerings.
- ▶ Data must remain transparently readable via direct IBM COS S3 access.

6.2 Architecture of the scenario

This section describes the architecture and components of this scenario.

6.2.1 Overview

A number of methods exist for connecting to and pushing data to IBM COS including a number of robust and rich featured commercial solutions from IBM and gateway partners as described in 4.3.4, “Use case 4: Enterprise file services” on page 81.

For the scenario described here, the technical requirements can be met by a number of these methods, however the commercial constraints might mandate a simple low-cost solution.

To achieve file-based access in this scenario (Figure 6-1), the system will be configured to mount an IBM COS bucket as a file system using an open source project named S3FS.

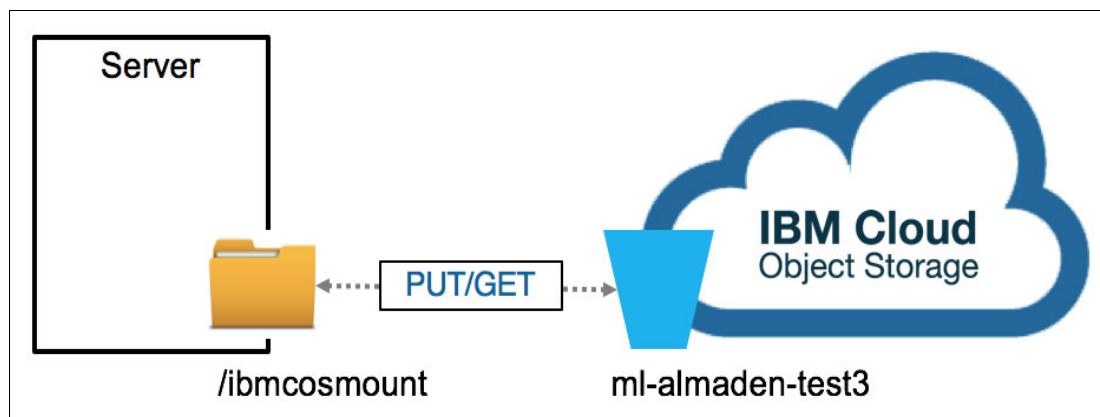


Figure 6-1 Simple file access scenario

This solution is limited because of the back-end nature of how data is uploaded (UNIX curl), it is not geared to high performance. Also because it is not a commercial offering, this setup does require self-support. It is, however essentially free, and does give a basic file interface that will allow data to be offloaded to IBM COS.

6.2.2 Requirements

Requirements are as follows:

- ▶ A Linux server with outward Internet connectivity. Although the instructions described here were compiled with CentOS, the steps should work on similar platforms like Fedora or Red Hat.
- ▶ An IBM COS account, including the public and secret key.
- ▶ An IBM COS bucket in which to store data.
- ▶ In this example, the application dumps a 1x100 MB file per hour so the solution must be able to handle this without timing out.

6.2.3 Planning

To prepare for this scenario, record several items for reference during implementation. These are listed in Table 6-1.

Table 6-1 Simple file access IBM COS implementation planning

Component	Value	Comment
IBM COS access point	s3-api.sjc-us-geo.objectstorage.sofflayer.net	Use an end point that is geographically appropriate
public-key-string	ASuJzEYSXA6tQnBhEoBy	As per Bluemix Portal
secret-key-string	opvhChJbm1x4tnlVT8aFXW7uXJ0XDIV2jhMGN25s	As per Bluemix Portal
Bucket name	ml-almaden-test3	Pre-created in Bluemix Portal
Local password file	/etc/ibmcoss.pwd	File containing bucket credentials
IBM COS mount point	/ibmcosmount	Directory to mount IBM COS bucket
Operating system	CentOS	Open Source variant of Red Hat
Operating system level	CentOS Linux release 7.2.1511 (Core)	Tested level

Note: If replicating this scenario, amend values to those matching your environment.

6.3 Implementation of the scenario

The assumption in this section is that a working Centos Linux server was deployed and is accessible through Secure Shell (SSH).

6.3.1 Pre-test verification

Run a connectivity test and software level check.

IBM COS connectivity test

Run a basic test, as shown in Example 6-1, to make sure that the server has visibility of the selected IBM COS endpoint.

Example 6-1 IBM COS ping test

```
[parallels@centos-linux-7 ~]$ ping -c 1
s3-api.sjc-us-geo.objectstorage.softlayer.net
PING s3-api.sjc-us-geo.objectstorage.softlayer.net (169.45.118.101) 56(84) bytes
of data.
64 bytes from 65.76.2da9.ip4.static.sl-reverse.com (169.45.118.101): icmp_seq=1
ttl=128 time=15.9 ms

--- s3-api.sjc-us-geo.objectstorage.softlayer.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.933/15.933/15.933/0.000 ms
```

Minimum software level check

Check whether the fuse package is already installed and at minimum levels (2.8.4). Use the command in Example 6-2.

Example 6-2 Verify fuse package level

```
[parallels@centos-linux-7 ~]$ rpm -qa|grep fuse
fuse-devel-2.9.2-7.el7.x86_64
glusterfs-fuse-3.7.1-16.0.1.el7.centos.x86_64
fuse-2.9.2-7.el7.x86_64
gvfs-fuse-1.22.4-6.el7.x86_64
fuse-libs-2.9.2-7.el7.x86_64
[parallels@centos-linux-7 ~]$
[parallels@centos-linux-7 ~]$ cat /etc/centos-release
CentOS Linux release 7.2.1511 (Core)
```

In this example, the fuse levels are at an adequate level and do not need to be reinstalled to be compatible with IBM COS.

6.3.2 S3FS installation

Installation involves these steps:

1. Install dependencies.
2. Download S3FS.
3. Compile and install S3FS.

Install dependencies

Install prerequisite packages needed by S3FS on this platform. The command used in this process is shown in Example 6-3.

Example 6-3 Installation of pre-requisite packages

```
[root@centos-linux-7 ~]# yum install gcc libstdc++-devel gcc-c++ fuse fuse-devel
curl-devel libxml2-devel mailcap git automake openssl-libs openssl-devel
```

After the prerequisites are installed, you can work with S3FS.

Download S3FS

Download S3FS from the online repository, as shown in Example 6-4.

Example 6-4 Download s3fs from online repository

```
[root@centos-linux-7 ~]# git clone https://github.com/s3fs-fuse/s3fs-fuse
Cloning into 's3fs-fuse'...
remote: Counting objects: 3242, done.
remote: Total 3242 (delta 0), reused 0 (delta 0), pack-reused 3242
Receiving objects: 100% (3242/3242), 1.67 MiB | 714.00 KiB/s, done.
Resolving deltas: 100% (2205/2205), done.
```

Compile and install S3FS

Use the standard Linux compile process. This process is shown in Example 6-5 through Example 6-8 on page 145.

Example 6-5 Use autogen

```
[root@centos-linux-7 ~]# cd s3fs-fuse
[root@centos-linux-7 s3fs-fuse]# ./autogen.sh
--- Make commit hash file -----
--- Finished commit hash file ---
--- Start autotools -----
configure.ac:26: installing './config.guess'
configure.ac:26: installing './config.sub'
configure.ac:27: installing './install-sh'
configure.ac:27: installing './missing'
src/Makefile.am: installing './depcomp'
parallel-tests: installing './test-driver'
--- Finished autotools -----
```

Example 6-6 Use configure

```
[root@centos-linux-7 s3fs-fuse]# ./configure --prefix=/usr --with-openssl
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for g++... g++
checking whether the C++ compiler works... yes
checking for C++ compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking for style of include used by make... GNU
checking dependency style of g++... gcc3
checking for gcc... gcc
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
```

```

checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking sys/xattr.h usability... yes
checking sys/xattr.h presence... yes
checking for sys/xattr.h... yes
checking attr/xattr.h usability... no
checking attr/xattr.h presence... no
checking for attr/xattr.h... no
checking sys/extattr.h usability... no
checking sys/extattr.h presence... no
checking for sys/extattr.h... no
checking s3fs build with nettle(GnuTLS)... no
checking s3fs build with OpenSSL... yes
checking s3fs build with GnuTLS... no
checking s3fs build with NSS... no
checking for pkg-config... /bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for common_lib_checking... yes
checking compile s3fs with... OpenSSL
checking for DEPS... yes
checking for malloc_trim... yes
checking for library containing clock_gettime... none required
checking for clock_gettime... yes
checking pthread mutex recursive... PTHREAD_MUTEX_RECURSIVE
checking for git... yes
checking for .git... yes
checking github short commit hash... d40da2c
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating test/Makefile
config.status: creating doc/Makefile
config.status: creating config.h
config.status: executing depfiles commands

```

Example 6-7 Use make

```

[root@centos-linux-7 s3fs-fuse]# make
make all-recursive
make[1]: Entering directory `/root/s3fs-fuse'
Making all in src
make[2]: Entering directory `/root/s3fs-fuse/src'
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT s3fs.o -MD -MP -MF
.deps/s3fs.Tpo -c -o s3fs.o s3fs.cpp
mv -f .deps/s3fs.Tpo .deps/s3fs.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT curl.o -MD -MP -MF
.deps/curl.Tpo -c -o curl.o curl.cpp

```

```

mv -f .deps/curl.Tpo .deps/curl.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT cache.o -MD -MP -MF
.deps/cache.Tpo -c -o cache.o cache.cpp
mv -f .deps/cache.Tpo .deps/cache.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT string_util.o -MD -MP
-MF .deps/string_util.Tpo -c -o string_util.o string_util.cpp
mv -f .deps/string_util.Tpo .deps/string_util.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT s3fs_util.o -MD -MP -MF
.deps/s3fs_util.Tpo -c -o s3fs_util.o s3fs_util.cpp
mv -f .deps/s3fs_util.Tpo .deps/s3fs_util.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT fdcache.o -MD -MP -MF
.deps/fdcache.Tpo -c -o fdcache.o fdcache.cpp
mv -f .deps/fdcache.Tpo .deps/fdcache.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT common_auth.o -MD -MP
-MF .deps/common_auth.Tpo -c -o common_auth.o common_auth.cpp
mv -f .deps/common_auth.Tpo .deps/common_auth.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT addhead.o -MD -MP -MF
.deps/addhead.Tpo -c -o addhead.o addhead.cpp
mv -f .deps/addhead.Tpo .deps/addhead.Po
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT openssl_auth.o -MD -MP
-MF .deps/openssl_auth.Tpo -c -o openssl_auth.o openssl_auth.cpp
mv -f .deps/openssl_auth.Tpo .deps/openssl_auth.Po
g++ -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -o s3fs s3fs.o curl.o cache.o string_util.o
s3fs_util.o fdcache.o common_auth.o addhead.o openssl_auth.o -pthread -lfuse -lcurl
-lxml2 -lcrypto
g++ -DHAVE_CONFIG_H -I. -I.. -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse
-I/usr/include/libxml2 -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -MT test_string_util.o -MD
-MP -MF .deps/test_string_util.Tpo -c -o test_string_util.o test_string_util.cpp
mv -f .deps/test_string_util.Tpo .deps/test_string_util.Po
g++ -g -O2 -Wall -D_FILE_OFFSET_BITS=64 -o test_string_util string_util.o
test_string_util.o
make[2]: Leaving directory `/root/s3fs-fuse/src'
Making all in test
make[2]: Entering directory `/root/s3fs-fuse/test'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory `/root/s3fs-fuse/test'
Making all in doc
make[2]: Entering directory `/root/s3fs-fuse/doc'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory `/root/s3fs-fuse/doc'
make[2]: Entering directory `/root/s3fs-fuse'
make[2]: Leaving directory `/root/s3fs-fuse'
make[1]: Leaving directory `/root/s3fs-fuse'

```

Example 6-8 Use make install

```

[root@centos-linux-7 s3fs-fuse]# make install
Making install in src
make[1]: Entering directory `/root/s3fs-fuse/src'
make[2]: Entering directory `/root/s3fs-fuse/src'
/bin/mkdir -p '/usr/bin'
/bin/install -c s3fs '/usr/bin'
make[2]: Nothing to be done for `install-data-am'.

```

```

make[2]: Leaving directory `/root/s3fs-fuse/src'
make[1]: Leaving directory `/root/s3fs-fuse/src'
Making install in test
make[1]: Entering directory `/root/s3fs-fuse/test'
make[2]: Entering directory `/root/s3fs-fuse/test'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/root/s3fs-fuse/test'
make[1]: Leaving directory `/root/s3fs-fuse/test'
Making install in doc
make[1]: Entering directory `/root/s3fs-fuse/doc'
make[2]: Entering directory `/root/s3fs-fuse/doc'
make[2]: Nothing to be done for `install-exec-am'.
/bin/mkdir -p '/usr/share/man/man1'
/bin/install -c -m 644 man/s3fs.1 '/usr/share/man/man1'
make[2]: Leaving directory `/root/s3fs-fuse/doc'
make[1]: Leaving directory `/root/s3fs-fuse/doc'
make[1]: Entering directory `/root/s3fs-fuse'
make[2]: Entering directory `/root/s3fs-fuse'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/root/s3fs-fuse'
make[1]: Leaving directory `/root/s3fs-fuse'

```

S3FS is now installed and ready to be configured to mount IBM COS.

6.3.3 S3FS configuration

Configuration involves these steps:

- ▶ Set up environment.
- ▶ Configure password file.
- ▶ Mount IBM COS bucket.

Set up environment

For guidance, see Example 6-9 and Example 6-10.

Example 6-9 Post Install s3fs system configuration

```

[root@centos-linux-7 s3fs-fuse]# export
PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/lib64/pkgconfig:/usr/local/lib/pkgconfig
[root@centos-linux-7 s3fs-fuse]# modprobe fuse
[root@centos-linux-7 s3fs-fuse]# echo "/usr/local/lib" >> /etc/ld.so.conf
[root@centos-linux-7 s3fs-fuse]# ldconfig

```

Example 6-10 Confirm fuse is installed and is detecting the correct level

```

[root@centos-linux-7 s3fs-fuse]# pkg-config --modversion fuse
2.9.2

```

Configure password file

Create a credentials file (Example 6-11), with a colon-delimited public and secret key in the file, and then secure the file with permissions so that only the user has read/write access.

Example 6-11 Create credentials file

```
[root@centos-linux-7 s3fs-fuse]# vi /etc/ibmcos.pwd

APuJzEYSXA6tQnBhEoBy:opvhChJbm1x4tn1VT8aFXW7uXJ0XD1V2jhMGN25s

[root@centos-linux-7 s3fs-fuse]# chmod 600 /etc/ibmcos.pwd
```

Mount IBM COS bucket

Create a mount directory (Example 6-12) and mount the IBM COS bucket (Example 6-13).

Example 6-12 Create mount directory

```
[root@centos-linux-7 s3fs-fuse] mkdir /ibmcosmount
```

Example 6-13 Mount IBM COS bucket

```
[root@centos-linux-7 s3fs-fuse] s3fs ml-almaden-test3 /ibmcosmount/ -o nocopyapi -o
use_path_request_style -o nomultipart -o sigv2 -o
url=http://s3-api.sjc-us-geo.objectstorage.softlayer.net/ -d -f -o curldbg -of2 -o
allow_other -o passwd_file=/etc/ibmcos.pwd
```

The bucket is now mounted directly over a directory and ready for access.

6.4 Verification of the scenario

Verification involves these steps:

1. Verify the mount
2. Write a file to the mounted directory,
3. Test file visibility, access, and deletion.

Verify mount

Confirm mount by listing all file system mounts and looking for s3fs.

Example 6-14 mount output

```
[parallels@centos-linux-7 /]$ mount | grep s3fs
s3fs on /ibmcosmount type fuse.s3fs
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

Write to IBM COS

Write a test file to the mounted directory by using the **dd** command (Example 6-15).

Example 6-15 Write 100 MB test file to /ibmcosmount using dd command

```
[parallels@centos-linux-7 ibmcosmount]$ dd if=/dev/zero of=testfile.out bs=102400
count=1024
1024+0 records in
1024+0 records out
104857600 bytes (105 MB) copied, 157.275 s, 667 kB/s
```

Depending on the network link, current IBM COS load, and local system load the results will vary, but the access pattern should look similar to Figure 6-2.

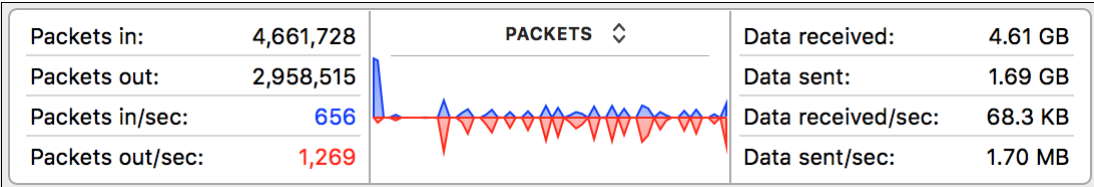


Figure 6-2 Network throughput observations

Test file visibility

List files on the system by using the `ls` command (Example 6-16).

Example 6-16 List files on system

```
[root@centos-linux-7 ibmcosmount]# ls -al
total 102405
drwxrwxrwx. 1 root root      0 Dec 31  1969 .
dr-xr-xr-x. 19 root root 4096 Feb 13 16:29 ..
-rw-r--r--. 1 root root 104857600 Feb 13 16:37 testfile.out
```

List files on IBM COS through the Bluemix portal by selecting **Storage** → **Object Storage** → **Manage Buckets** → **ml-almaden-test3**.

As illustrated in Figure 6-3, the object is visible with the same file size as the local listing.

Object Name	Size	Last Modified	
(object name)			+
testfile.out	100.00 MB	2017-02-13 04:37 pm	-

Figure 6-3 Bluemix object storage portal

Test file access

Because reading 100 MB of zeros is impractical, the file can be scanned by using the word count (`wc`) command (Example 6-17).

Example 6-17 Line count of test file

```
[root@centos-linux-7 ibmcosmount]# wc -l testfile.out
1 testfile.out
```

Because the file must be accessed locally, the entire file is temporarily copied back. This takes approximately one minute on the test system and is visible on a network monitor of your choice. See Figure 6-4.

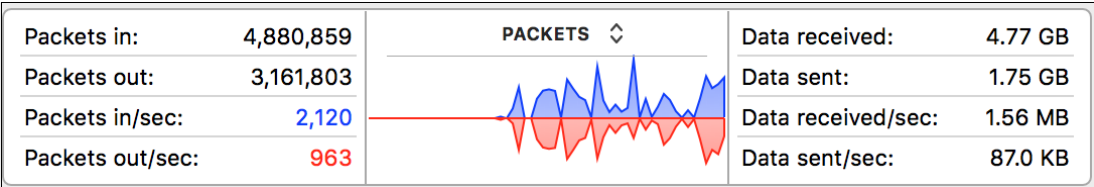


Figure 6-4 IBM COS retrieve using S3FS

Test file deletion

Verify that the file can be deleted by using a standard utility (Example 6-18).

Example 6-18 File deletion with standard linux rm command

```
[root@centos-linux-7 ibmcosmount]# rm testfile.out
rm: remove regular file 'testfile.out'? y
```

List the files by using a standard Linux utility (Example 6-19).

Example 6-19 List file locally using ls -al

```
[root@centos-linux-7 ibmcosmount]# ls -al
total 5
drwxrwxrwx. 1 root root 0 Dec 31 1969 .
dr-xr-xr-x. 19 root root 4096 Feb 13 16:29 ..
```

Confirm the object is deleted on IBM COS through the Bluemix portal by selecting **Storage** → **Object Storage** → **Manage Buckets** → **ml-almaden-test3**.


Object Name	Size	Last Modified
(object name)		
No Objects Found		

Figure 6-5 Bluemix portal object listing in test bucket

Important: Be sure that the data being worked on is not a local copy but the bucket directly. This means deleting an object through the local Linux command instantly deletes it on the IBM COS back end.

6.5 Conclusion

Most modern platforms such as Linux have the capability to integrate with IBM COS. In this scenario, Linux was integrated with IBM COS using an simple open source library to mount the public bucket and access it as a file system.

Data ingested using this method is transparently available, so if the user application is updated to integrate with IBM COS S3 directly, data migration might not be required.

This method is not intended for complex workloads, systems dumping thousands of files, or systems that need a fast response time or high throughput.

For those workloads, you should consider one of the commercial offerings from IBM and the partner community that best suits your needs.

Key point: Organizations can use IBM COS cloud economics on a range of budgets.



IBM Spectrum Protect integration

This chapter describes the configuration steps and architecture that are required to configure IBM Spectrum Protect to use IBM Cloud Object Storage (IBM COS) as its back-end storage repository.

The following topics are covered in this chapter:

- ▶ 7.1, “Scenario description” on page 152
- ▶ 7.2, “Architecture of the scenario” on page 152
- ▶ 7.3, “Implementation of the scenario” on page 155
- ▶ 7.4, “Verification of the scenario” on page 167
- ▶ 7.5, “Conclusion” on page 169

7.1 Scenario description

In this scenario, IBM Spectrum Protect is configured to use IBM COS as the back-end storage repository for all backup data. The environment that is described in this scenario is appropriate for organizations with the following attributes:

- ▶ A need to reduce the capacity that is required for storing backup data and the associated costs.
- ▶ No regulatory requirements mandating multiple data copies.
- ▶ Sufficient network bandwidth to support the required logical data rates that are required for backup and restore (especially on-premises IBM Spectrum Protect to IBM COS public)
- ▶ Those clients that are deploying IBM Spectrum Protect in a Bluemix infrastructure to avoid the costs that are associated with pre-allocated block or file storage.

Note: The logical data rate is the rate of data transfer, including any data reduction benefits. For example, data that achieves a 4:1 reduction from data deduplication and compression that is being transferred at 1 Gbps achieves a 4 Gbps logical data rate.

7.2 Architecture of the scenario

This section covers the architecture of this scenario.

7.2.1 Overview

A number of possible IBM COS integration architectures are possible. However, for this scenario, a single IBM Spectrum Protect instance is configured to use IBM COS as its primary data store, as shown in Figure 7-1.

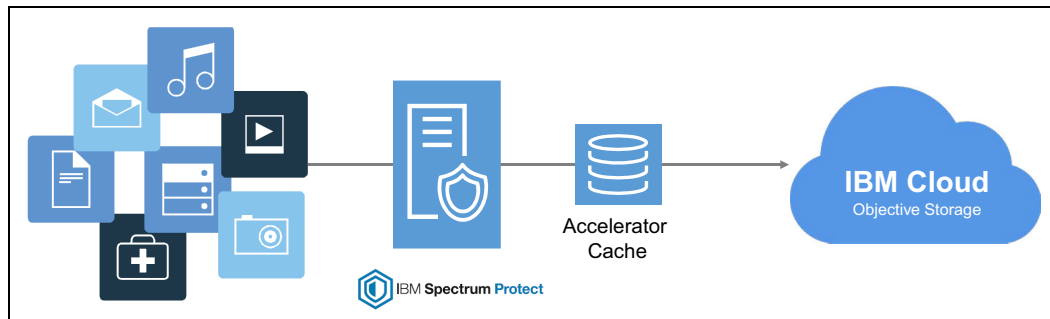


Figure 7-1 IBM Spectrum Protect IBM COS scenario

As described in 4.4.1, “IBM Spectrum Protect” on page 91, there are many deployment options. Although the restore and disaster recovery (DR) implications of these options differ, the implementation of these additional architectures adds IBM Spectrum Protect replication to IBM COS integration.

IBM Spectrum Protect replication implementation and planning are not described in this book but IBM Knowledge Center does document the following information:

- ▶ [Replicating client data to another server](#)
- ▶ [Checklist for node replication](#)

For this scenario, the IBM Spectrum Protect PoC virtual appliance, which is available at the IBM developerWorks®, is being used. This appliance is available for download for anyone with an IBM ID, and can be deployed on an on-premises hypervisor or on a Bluemix Infrastructure cloud.

Readers with existing IBM Spectrum Protect instances can use their existing infrastructure or use their software entitlements to install a new IBM Spectrum Protect instance from scratch.

IBM developerWorks website

The IBM Spectrum Protect virtual appliance is available at the following web page and is referred to throughout this section:

<https://www.ibm.com/developerworks/community/groups/community/TSMVirtualAppliance>

7.2.2 Requirements

You need a hypervisor with public network connectivity and sufficient resources to host the linux VM running IBM Spectrum Protect. The tested hypervisor is VMWare. Instructions about the conversion of the image files to other formats are available at the “IBM developerWorks website” with the download images.

Here are the minimum resource requirements for using the appliance area:

- ▶ Space: 13 GB of space if you deploy a thin-provisioned resource. Additional space is required for backups, a database, and logs.
- ▶ Memory: 24 GB of memory that is allocated to the VM.
- ▶ Cores: 8 CPU cores that are allocated to the VM.
- ▶ Network: A public-facing network for IBM COS access.

Data store layout

Although it is not mandatory, it is a preferred practice to place the database logs and storage pool volumes on different data stores to avoid resource contention and improve performance, as shown in Figure 7-2 on page 154.

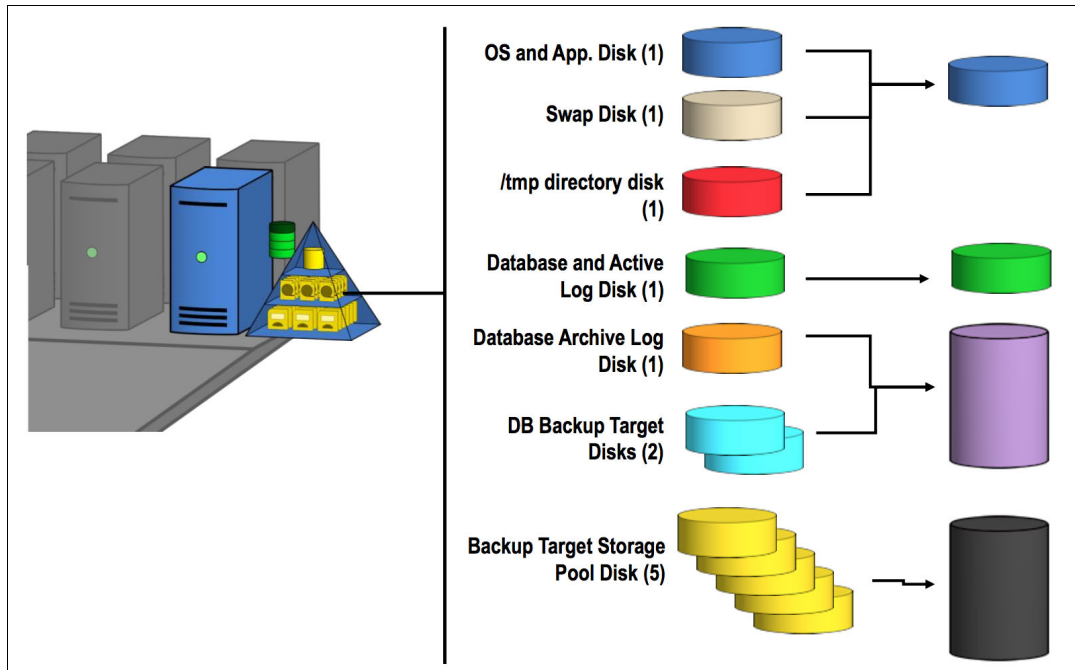


Figure 7-2 Balanced IBM Spectrum Protect storage layout

If possible, an ideal approach is to place the database and active log disk on the fastest possible data store if one is available.

IBM Cloud Object Storage

An IBM COS account, a pre-created bucket, and access credentials are required for this scenario.

7.2.3 Planning

The details of the demonstration environment are shown in Table 7-1.

Table 7-1 IBM Spectrum Protect and IBM COS planning details

Component	Value	Comments
Host name	m1sp810_demo	
Host IP	10.1.102.7	
Network mask	255.255.252.0	
Gateway	10.1.100.1	
DNS server	10.1.1.2	
IBM Spectrum Protect instance name	tsminst1	Default name.
Storage pool name	IBM_COS	
IBM COS access point	s3-api.sjc-us-geo.objectstorage.softlayer.net	Use the endpoint that is geographically appropriate.
public-key-string	ASuJzEYSXA6tQnBhEoBy	As Bluemix Portal.

Component	Value	Comments
secret-key-string	opvhChJbm1x4tn1VT8aFXW7uXJ 0XD1V2jhMGN25s	As Bluemix Portal.
Bucket name	ml-almaden-test	Pre-created in Bluemix Portal.
Hypervisor version	5.5	
Storage provisioning type	Thin	Do not use thin provisioning for performance testing.

Note: If replicating this scenario, change the values to the values that match your environment.

7.3 Implementation of the scenario

This section assumes that an IBM Spectrum Protect server is deployed and users can log on to this host through SSH.

An IBM Spectrum Protect demonstration appliance and detailed installation instructions are available at the “IBM developerWorks website” on page 153

Once the virtual image is deployed, and network details supplied, the system will start up with the below splash screen, as shown in Figure 7-3.



Figure 7-3 IBM Spectrum Protect virtual appliance startup

Note: If you use the demonstration appliance from developerWorks, be aware that it is not intended for production use, and the licenses expire after 60 days.

Allow the boot to complete. After booting, the virtual console will list all URLs that are required for managing this IBM Spectrum Protect environment (Figure 7-4).

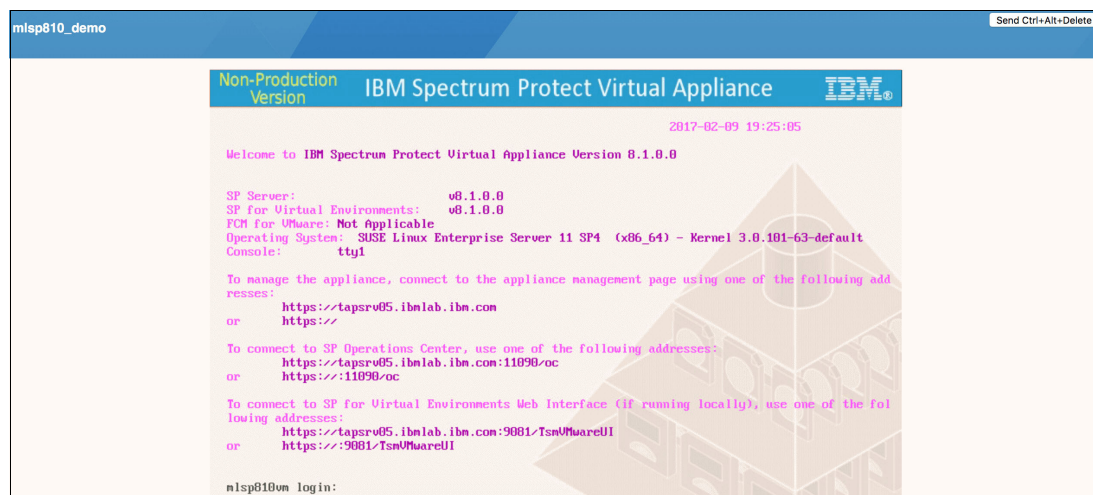


Figure 7-4 IBM Spectrum Protect virtual appliance URL details screen

7.3.1 Pre-testing setup

A GUI is not installed on the Linux OS that is used, so the first steps are performed through the CLI.

Testing the IBM COS connectivity

To ensure that basic outward connectivity exists, ping the selected IBM COS access point, as shown in Example 7-1.

Example 7-1 IBM COS connectivity test

```
mlsp810vm:~ # ping -c 1 s3-api.sjc-us-geo.objectstorage.softlayer.net
PING s3-api.sjc-us-geo.objectstorage.softlayer.net (169.45.118.101) 56(84) bytes of
data.
64 bytes from 65.76.2da9.ip4.static.sl-reverse.com (169.45.118.101): icmp_seq=1
ttl=235 time=181 ms
```

```
--- s3-api.sjc-us-geo.objectstorage.softlayer.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 181.494/181.494/181.494/0.000 ms
```

Creating a test file

Create a 1 GB test file for backup by running the commands that are shown in Example 7-2.

Example 7-2 Test file creation

```
mlsp810vm:~ # dd if=/dev/urandom of=sample.file bs=64M count=16
16+0 records in
16+0 records out
1073741824 bytes (1.1 GB) copied, 147.78 s, 7.3 MB/s
```

Note: The lab environment that is required to perform meaningful testing is set up and the IBM COS and IBM Spectrum Protect integration may begin.

7.3.2 IBM Spectrum Protect configuration

IBM Spectrum Protect can be managed through the CLI or through the Operations Center web interface. Both configuration methods are documented in this section.

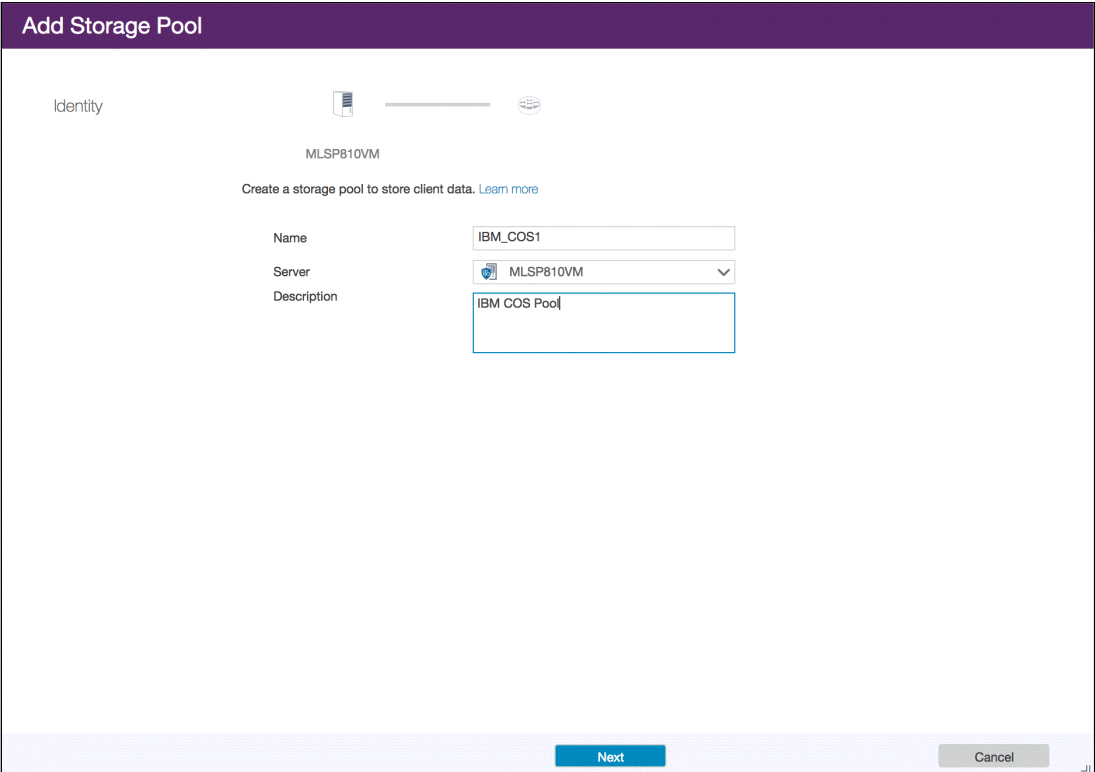
GUI setup

This section describes the GUI setup.

Creating a storage pool

To create storage pool, complete the following steps:

1. Log in to IBM Spectrum Protect Operations Center.
2. Start the new storage pool wizard by clicking **Storage** → **Storage Pools** → **+Storage Pools**.
3. Enter the preferred storage pool name and description (Figure 7-5), and click **Next**.



The screenshot shows the 'Add Storage Pool' wizard in the IBM Spectrum Protect Operations Center. The wizard is in the 'Identity' step, indicated by a progress bar. Below the progress bar, the text 'MLSP810VM' is displayed, followed by the instruction 'Create a storage pool to store client data. Learn more'. The form contains three fields: 'Name' with the value 'IBM_COS1', 'Server' with a dropdown menu showing 'MLSP810VM', and 'Description' with the value 'IBM COS Pool'. At the bottom of the form, there are 'Next' and 'Cancel' buttons.

Figure 7-5 Storage pool naming

4. Select **Offsite cloud storage pool type** and click **Next** (Figure 7-6).

Add Storage Pool

Type

MLSP810VM

IBM_COS2

Choose the type of pool that best supports your business goals. [Learn more](#)

To copy data from an existing directory-container pool, cancel the wizard, select the pool, and click **More > Add Container-copy Pool**.

Container-based storage

Traditional volume-based storage

Directory

File-based storage on disk with optional copy pools

On-premises cloud

Object-based storage that is managed by internal IT staff in your data center

Off-premises cloud

Storage in vendor-managed repositories, using IBM SoftLayer, OpenStack Swift or Amazon S3

Disk (primary)

Storage on disk or in a mountable deduplicating appliance

Tape (primary)

Storage on tape or in a deduplicating VTL

Tape (copy)

Copies of primary storage on tape or in a VTL

Back

Next

Cancel

Figure 7-6 Storage pool type selection

158

Cloud Object Storage as a Service: IBM Cloud Object Storage from Theory to Practice

5. Select **S3** as the cloud type, enter the credentials, and select **Other** for the Region. Enter the closest geographical endpoint and click **Next** (Figure 7-7).

Add Storage Pool

Credentials

MLSP810VM

IBM_COS2

Select the cloud type and enter connection information for accessing the cloud. [Learn more](#)

Pool type

Off-premises cloud

Encryption

☒ Enable

Cloud type

Amazon S3

Access key ID

TEUJzEYSXA6tQnBhEoBy

Password

.....

Region

Other (specify)

Bucket name

ml-almaden-test2

URL

http://s3-apl.sjc-us-geo.objectstorage.softlay

Back

Next

Cancel

Figure 7-7 Cloud details entry

6. The GUI provides an option to configure an encryption key to be stored in the IBM Spectrum Protect database to be used for encrypting all data (Figure 7-8).

Add Storage Pool

Protect Encryption Keys

MLSP810VM

IBM_COS2

Data in this storage pool will be encrypted and the encryption keys will be included in database backups. Specify a password to protect these encryption keys.

⚠ Store this password in a secure location. You cannot recover the server database without it. [Learn more](#)

Password

Confirm password

Back

Next

Cancel

Figure 7-8 Cloud encryption key setup

7. Select the directory to use as the accelerator cache (Figure 7-9).

Add Storage Pool

Local Storage

MLSP810VM

IBM_COS2

Specify at least one existing directory where IBM_COS2 can temporarily store data before it is transferred to the cloud. Type a directory path and press **Enter**. [Learn more](#)

Directories

— /tsminst1/TSMfile00/

Back Add Storage Pool Cancel

Figure 7-9 Accelerator cache setup

Note: Using an accelerator cache directory is strongly preferred because it results in significantly faster ingest and destage to cloud as data is transformed and chunked together into larger objects.

8. Review details and proceed to the Policy Editor, as shown in Figure 7-10.

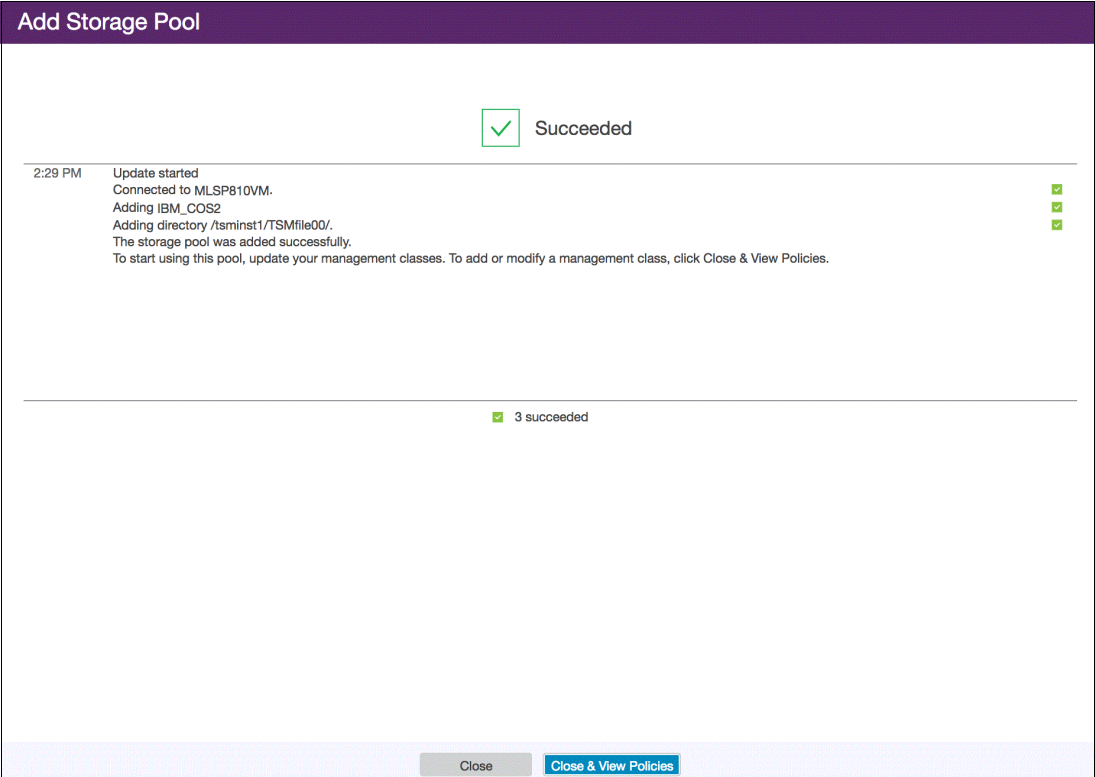


Figure 7-10 Storage pool definition complete

9. Right-click a **Policy Domain** for the demonstration machine and select **Details** (Figure 7-11).

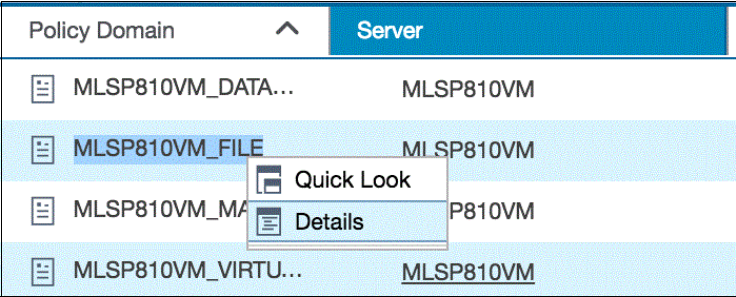


Figure 7-11 Policy selection

10. Enable configuration mode by selecting the toggle on right side of the window, as shown in Figure 7-12.

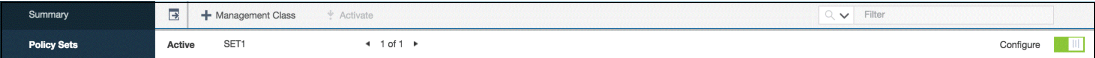


Figure 7-12 Enable configuration

Add Management Class

Management classes define the data protection services that are available for clients to use. [Learn more](#)

☐ Make default

☒ Backup

Destination

While the file or directory is c

Keep extra versions for

Archive

Destination

Keep archives for

Migrate

Destination

Technique

TOC Destination

Select

2 months

1 year

None

Requires backup

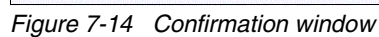
IBM_COS2

Name	Status	Capacity
(None)		
ARCHIVEPOOL	Normal	No capacity
BACKUPPOOL	Normal	No capacity
DEDUPPOOL	Normal	0 TB 8.3 TB
IBM_COS1	Normal	No capacity
IBM_COS2	Normal	No capacity
SPACEMGPOOL	Normal	No capacity

Cancel

Add

12. Review the details in the confirmation window (Figure 7-14), and click **Close**.




13. Activate the policy set, as shown in Figure 7-15.



Figure 7-15 Activate a policy

14. Accept the warning message and proceed. This warning notifies you that policy changes in IBM Spectrum Protect allow retrospective changes and should be reviewed carefully in production environments. See Figure 7-16.

Activate SET1

 **Changing the policy set can cause data loss!** [Learn more](#)

The following updates will be made:

Management Class	Changes	Default	Backup Destination	Backups	Keep Backups	Deleted Backups	Keep Deleted Backups	Archive Des
1WEEK		No	(None)	(None)	(None)	(None)	(None)	(None)
	To be added	Yes	IBM_COS2	No limit	7	No limit	7	IBM_COS2
BACKUP_DISK_KEEP30DAYS_ARCH730		Yes						
	Pending Activation	No						

☒ I understand that these updates can cause data deletion.

CancelActivate

Figure 7-16 Activate policy confirmation screen

Note: Storage pool and policy configuration through the Operations Center GUI is now complete.

CLI setup

Log in to the command line by running the `dsmdmnc` command and administrative credentials.

Defining the IBM_COS storage pool

Run the commands in Example 7-3.

Example 7-3 Define IBM_COS storage pool

```
Protect: MLSP810VM>DEFINE STGPOOL IBM_COS1 STGTYPE=CLOUD
IDENTITY=ASuJzEYSXA6tQnBhEoBy PASSWORD=opvhChJbm1x4tn1VT8aFXW7uXJ0XD1V2jhMGN25s
CLOUDLOCATION=OFFPREMISE ENCRYPT=No CLOUDTYPE=S3
CLOUDURL=http://s3-api.sjc-us-geo.objectstorage.softlayer.net
bucketname=ml-almaden-test
ANR2249I Storage pool IBM_COS is defined.
```

Defining the accelerator cache directory

Because we are reusing an existing directory, we delete an existing directory definition from the IBM Spectrum Protect virtual appliance DEDUPPOOL (Example 7-4) and use the same directory as accelerator cache.

Example 7-4 Identify an existing storage pool directory for reuse: SP demonstration VM only

```
Protect: MLSP810VM>q stgpooldir
```

Storage Pool Name	Directory	Access
-----	-----	-----
DEDUPPOOL	/tsminst1/TSMfile01	Read/Write
DEDUPPOOL	/tsminst1/TSMfile02	Read/Write
DEDUPPOOL	/tsminst1/TSMfile03	Read/Write
DEDUPPOOL	/tsminst1/TSMfile04	Read/Write
DEDUPPOOL	/tsminst1/TSMfile05	Read/Write
DEDUPPOOL	/tsminst1/TSMfile06	Read/Write
DEDUPPOOL	/tsminst1/TSMfile07	Read/Write
DEDUPPOOL	/tsminst1/TSMfile08	Read/Write
DEDUPPOOL	/tsminst1/TSMfile09	Read/Write
DEDUPPOOL	/tsminst1/TSMfile00	Read/Write

```
Protect: MLSP810VM>del stgpooldir DEDUPPOOL /tsminst1/TSMfile00
ANR3255I Storage pool directory /tsminst1/TSMfile00 deleted.
```

Note: Readers can create a new directory or mountpoint and skip the above step if they have adequate space in their data store.

Example 7-5 shows how to define an accelerator cache directory.

Example 7-5 Define accelerator cache directory

```
Protect: MLSP810VM>def stgpooldir IBM_COS1 /tsminst1/TSMfile00
ANR3254I Storage pool directory /tsminst1/TSMfile00 was defined in storage pool
IBM_COS1.
```

Amending a policy domain to use the IBM_COS pool

To amend a policy domain to use the IBM_COS pool, complete the following steps:

1. Identify the domain that is used (Example 7-6).

Example 7-6 Query Node

```
Protect: MLSP810VM>
```

```
Protect: MLSP810VM>q node MLSP810VM
```

Node Name	Platform	Policy Domain Name	Days Since Last Access	Days Since Password Set	Locked?
-----	-----	-----	-----	-----	-----
MLSP810VM	TDP VMware	MLSP810VM_FILE	<1	1	No

2. Identify the policy that is set to edit (Example 7-7).

Example 7-7 Query policy

```
Protect: MLSP810VM>q pol MLSP810VM_FILE
```

Policy Domain Name	Policy Set Name	Default Management Class Name	Description
-----	-----	-----	-----
MLSP810VM_FILE	ACTIVE	BACKUP_DISK_KEEP3 ODAYS_ARCHIVE730	Container for policy management classes
MLSP810VM_FILE	SET1	BACKUP_DISK_KEEP3 ODAYS_ARCHIVE730	Container for policy management classes

3. Define backup policies by using an IBM_COS pool as the target. To define the management class, assign it as the default, and activate the policy, run the commands that are shown in Example 7-8.

Example 7-8 Define a management class, assign it as the default, and activate the policy

```
Protect: MLSP810VM>def mgmtclass MLSP810VM_FILE SET1 1WEEK
```

```
ANR1520I Management class 1WEEK defined in policy domain MLSP810VM_FILE, set SET1.
```

```
Protect: MLSP810VM>define copygroup MLSP810VM_FILE SET1 1WEEK type=backup
```

```
destination=IBM_COS1 verexist=nolimit verdeleted=nolimit retextra=7 retonly=7
```

```
ANR1530I Backup copy group STANDARD defined in policy domain MLSP810VM_FILE, set SET1,  
management class 1WEEK.
```

```
Protect: MLSP810VM>assign defmgmtclass MLSP810VM_FILE SET1 1WEEK
```

```
ANR1538I Default management class set to 1WEEK for policy domain MLSP810VM_FILE, set SET1.
```

```
Protect: MLSP810VM>activate policyset MLSP810VM_FILE SET1
```

```
Do you wish to proceed? (Yes (Y)/No (N)) y
```

```
ANR1514I Policy set SET1 activated in policy domain MLSP810VM_FILE.
```

Note: The storage pool and policy configuration through the CLI is now complete.

7.4 Verification of the scenario

To verify the setup, complete a backup to the IBM_COS1 pool to initiate data transfer to the cloud by completing the following steps:

1. Initiate the backup, as shown in Example 7-9.

Example 7-9 Perform a backup by using the CLI

```
mlsp810vm:~ # /opt/tivoli/tsm/client/ba/bin/dsmc selective sample.file
IBM Spectrum Protect
Command Line Backup-Archive Client Interface
  Client Version 8, Release 1, Level 0.0
  Client date/time: 10/02/17   13:16:29
(c) Copyright by IBM Corporation and other(s) 1990, 2016. All Rights Reserved.
```

```
Node Name: MLSP810VM
Session established with server MLSP810VM: Linux/x86_64
  Server Version 8, Release 1, Level 0.0
  Server date/time: 10/02/17   13:16:29  Last access: 10/02/17   13:16:04
```

Selective Backup function invoked.

```
Directory-->          4,096 /root [Sent]
Normal File-->      1,073,741,824 /root/sample.file [Sent]
Selective Backup processing of '/root/sample.file' finished without failure.
```

```
Total number of objects inspected:          2
Total number of objects backed up:           2
Total number of objects updated:             0
Total number of objects rebound:            0
Total number of objects deleted:             0
Total number of objects expired:             0
Total number of objects failed:              0
Total number of objects encrypted:           0
Total objects deduplicated:                  1
Total number of objects grew:                0
Total number of retries:                     0
Total number of bytes inspected:             1.00 GB
Total number of bytes processed:             1.00 GB
Total bytes before deduplication:            1.00 GB
Total bytes after deduplication:             1.00 GB
Total number of bytes transferred:          1.00 GB
Data transfer time:                          11.45 sec
Network data transfer rate:                  91,946.00 KB/sec
Aggregate data transfer rate:                59,473.10 KB/sec
Objects compressed by:                       0%
Deduplication reduction:                     0.00%
Total data reduction ratio:                  0.00%
Elapsed processing time:                     00:00:17
```

2. Verify container location; initially the data is placed into the accelerator cache, as shown in Example 7-10 (/tsminst1/TSMfile00 is the Local accelerator cache).

Example 7-10 Query container before destaging

Protect: MLSP810VM>q container

Container	Storage Pool Name	Container Type	State
-----	-----	-----	-----
/tsminst1/TSMfile00/00/0000000000000006.ncf	IBM_COS1	Non Dedup	Available
/tsminst1/TSMfile00/00/0000000000000007.dcf	IBM_COS1	Dedup	Available
/tsminst1/TSMfile00/00/0000000000000008.dcf	IBM_COS1	Dedup	Available
/tsminst1/TSMfile00/00/0000000000000009.dcf	IBM_COS1	Dedup	Available

3. Destage to IBM COS (automatic).

After a period, an automated process destages the data to the IBM COS bucket that is configured on this storage pool. No user interaction is required here because IBM Spectrum Protect automatically manages the process, but progress can be queried by running the **query process** command.

Example 7-11 Destage to IBM COS

Protect: MLSP810VM>q process

Process Number	Process Description	Process Status
-----	-----	-----
14	Local to Cloud Transfer	Local disk to Cloud transfer for directory-container storage pool IBM_COS1. 1 container(s) processed. 4,555 bytes in 1 data extent(s) transferred. Elapsed time: 0 Days, 0 Hours, 2 Minutes.

4. Verify that the data destaged.

The query container should show that data is now in cloud-based containers in IBM COS, as shown in Example 7-12 (m1-almaden-test is the IBM COS bucket name).

Example 7-12 Query container after destaging

Protect: MLSP810VM>query container

Container	Storage Pool Name	Container Type	State
-----	-----	-----	-----
m1-almaden-test/006-48dd4e0130efe6118cb7005056a38f63-L/0000000000000006.ncf	IBM_COS1	Cloud	
m1-almaden-test/007-48dd4e0130efe6118cb7005056a38f63-L/0000000000000007.dcf	IBM_COS1	Cloud	
m1-almaden-test/008-48dd4e0130efe6118cb7005056a38f63-L/0000000000000008.dcf	IBM_COS1	Cloud	
m1-almaden-test/009-48dd4e0130efe6118cb7005056a38f63-L/0000000000000009.dcf	IBM_COS1	Cloud	

5. Verify that the data is visible in the Bluemix Infrastructure portal.

Log in to the Bluemix Infrastructure portal and drill down to the relevant bucket. The containers with data are visible in IBM COS, as shown in Figure 7-17.

Object Name	Size	Last Modified	
(object name) local accelerator cache			+
001-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	16.00 MB	2017-02-09 06:18 pm	-
002-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	16.00 MB	2017-02-09 06:09 pm	-
003-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	117.93 MB	2017-02-09 06:23 pm	-
004-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	78.62 MB	2017-02-09 06:23 pm	-
005-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	62.85 MB	2017-02-09 06:22 pm	-
006-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	16.00 MB	2017-02-10 11:20 am	-
007-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	463.58 MB	2017-02-10 11:25 am	-
008-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	332.70 MB	2017-02-10 11:26 am	-
009-48dd4e0130efe6118cb7005056a38f63-L/00000000000000...	238.66 MB	2017-02-10 11:26 am	-

Figure 7-17 IBM Spectrum Protect data in IBM COS

Note: Data import into IBM Spectrum Protect and into IBM COS has been verified.

7.5 Conclusion

IBM Spectrum Protect can be easily integrated with IBM COS in a matter of minutes for existing deployments. Testing on a new environment can be easily done by using the available developerWorks virtual image, which can be set up in under an hour.

Key point: Using IBM COS enables businesses to derive benefits from reduced infrastructure costs due to reduced storage requirements, operational simplicity, and the shift to cloud economics.



VideoRecon Video Analytics

This chapter describes a solution, named *VideoRecon Video Analytics*, that provides actionable cognitive insights on the video content.

The following topics are covered in this chapter:

- ▶ 8.1, “Scenario description” on page 172
- ▶ 8.2, “Architecture of the scenario” on page 174
- ▶ 8.3, “Implementation of the scenario” on page 176
- ▶ 8.4, “Verification of the scenario” on page 180

8.1 Scenario description

This scenario covers the architecture of the prototype of the IBM BlueChasm [VideoRecon](#) Video Analytics platform.

BlueChasm: BlueChasm is the digital development unit of Mark III Systems, a long-time IBM Platinum Business Partner and winner of both the 2017 IBM Beacon Award for Outstanding Solution Developed on Bluemix and Outstanding Storage Solution categories (anchored by IBM Cloud Object Storage). For more information about the enterprise version of VideoRecon or information of how VideoRecon can be used to demonstrate the power of using the IBM stack (regardless of use case), contact [BlueChasm](#).

VideoRecon enables enterprises, service providers, institutions, and software as a service (SaaS) and software partners to gain actionable cognitive insights on their video content, whether the video content is a repository of historical videos or new video that generated from new services, applications, or devices. This case study examines, at a high-level, why video analytics matter relative to digital initiatives, how the prototype to VideoRecon was built using IBM Cloud Object Storage and IBM Bluemix, and some of the basic steps of how this platform and its microservices use Cloud Object Storage to store key data.

Figure 8-1 shows the VideoRecon Demo and Enterprise Edition, which is accessible from [VideoRecon](#).

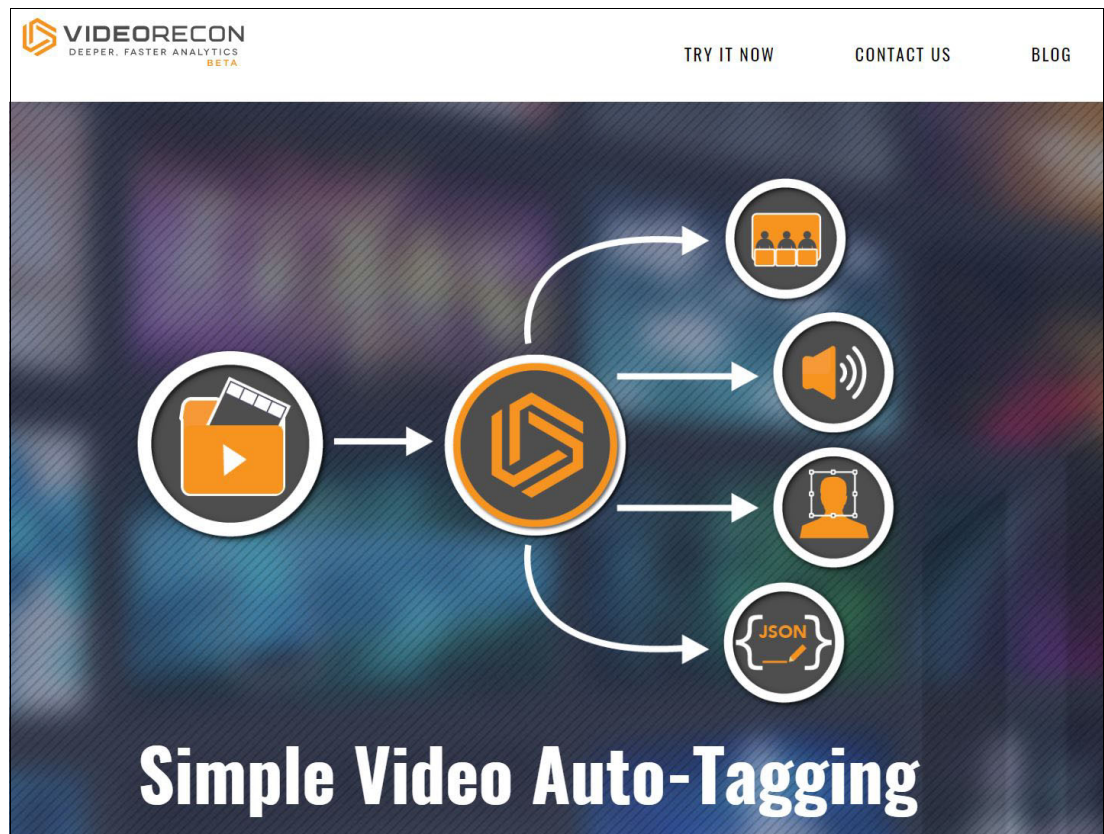


Figure 8-1 VideoRecon Demo and Enterprise Edition accessible at [videorecon.io](#)

8.1.1 Business case

VideoRecon is a cloud-native, video analytics platform that enables enterprises and service providers to use cognitive computing, anchored by IBM Watson, to automatically see, listen, and understand video content. VideoRecon is currently served from the IBM Cloud on IBM Bluemix and is available for consumption by enterprises, service providers, institutions, and SaaS and software partners via the VideoRecon open API, the VideoRecon enterprise portal, or one of VideoRecon's third-party integration packs (such as Box.com).

VideoRecon was designed to solve several major business challenges behind the harnessing of video content for enterprise use. Video represents a huge opportunity for enterprises moving forward, because approximately 80% of content on the Internet today is video and it represents the fastest growing unstructured data type. Although enterprises have accumulated hundreds, if not thousands or tens of thousands of videos, it represents only a small amount of the overall content that will likely be generated in the future, as new use cases spurred by technology advances generate even more content, including secular trends like IoT, drones, self-driving vehicles, and general-purpose computer vision.

Beyond just adequately storing the videos, which can sometimes be challenging, many challenges exist that prevent these videos from being fully analyzed, understood, and acted upon for business advantage or used to create new digital use cases or engagement models for enterprises.

This list describes some of those unsolved challenges for video:

- ▶ Sheer volume of existing and constant newly generated video content stored with inaccurate, minimal, or no tags to describe them
- ▶ Metadata and tagging processes today that are manual, with manpower required to watch videos and tag them with non-uniform human judgement between taggers
- ▶ Long turnaround times for attaching metadata to videos for near real time or future analysis, because humans must currently take the time to watch videos for tagging
- ▶ High cost of paid services either internal or with a third-party to watch and analyze videos
- ▶ Lack of manpower to watch and classify videos when many videos must be watched or analyzed in parallel, either in near real time or for a quick turnaround
- ▶ Lack of accuracy of metadata tags and recognition of key objects and events in videos due to natural human error, fatigue, and motivation, especially in the context of constantly and rapidly changing business requirements

For those organizations that are able to take advantage of video analytics, the following benefits can be derived:

- ▶ Significantly faster speed-to-market and iterations of use cases that involve video content, as in these three examples:
 - Speedier video production enabled by faster search, filtering, and analytics
 - More relevant targeted marketing videos served to customers
 - Faster availability of transcripts and summaries from training videos, interviews, or lectures and classes
- ▶ More accurate metadata tags for videos that can augment and supplement human judgement:
 - Trained to look for things that humans are either not tasked with or have difficulty seeing
 - Look for patterns and actionable trends across a large number of videos

- ▶ New use cases that involve both real-time video and stored video content, including use cases that involve computer vision and IoT, as in these examples:
 - Video interview insights of employees and external stakeholders
 - Visual identification of people and inventory
- ▶ Greater cost-effectiveness in analyzing and tagging videos, which will allow video analytics to be more widely accessible to more enterprises and their growing use cases

8.2 Architecture of the scenario

The VideoRecon prototype is a cloud-native platform built on the IBM Cloud with a microservices approach. Each microservice is an application that is housed in its own IBM Container on Bluemix and can be deployed, iterated on, and scaled independently, using all the advantages of Docker with some value-added features from IBM.

VideoRecon processes and analyzes both video (visual) and audio as part of its platform. From an architectural standpoint, video and audio are analyzed separately with different microservices, although their architectural design and use of IBM Cloud Object Storage is somewhat similar as is described next. Note that although Cloud Object Storage is part of the default architecture, users do have the ability to opt out of storing files altogether (even temporarily). Alternatively, users can also choose to use Cloud Object Storage to store these files for an extended period of time.

For visual processing, relevant video and images enter our platform through the API in this example. This API is accessible for consumption by third-party SaaS/software applications, our own VideoRecon enterprise portal, or third-party integrations (such as Box, Dropbox). After video and associated images are ingested by our primary VideoRecon app service, the data is then sent to VideoRecon's file processor service, and then funneled into IBM Cloud Object Storage. After the video and images are in Cloud Object Storage, they can then be analyzed in different combinations by the other microservices in VideoRecon, including services accessible through IBM Watson APIs. If you want to ensure that the files are not corrupted prior to storage, you can pass the files to IBM Watson or other analytics services first before storing in IBM Cloud Object Storage. The URL and metadata of the stored videos and images are then stored in a Cloudant NoSQL database for easy access, in addition to the metadata tags being sent back through the API to the original requester. The files and metadata tags are then deleted from VideoRecon, unless the user specifically opts in to a certain time window where their data is to be stored on VideoRecon.

Figure 8-2 shows the architectural diagram of visual analytics within VideoRecon.

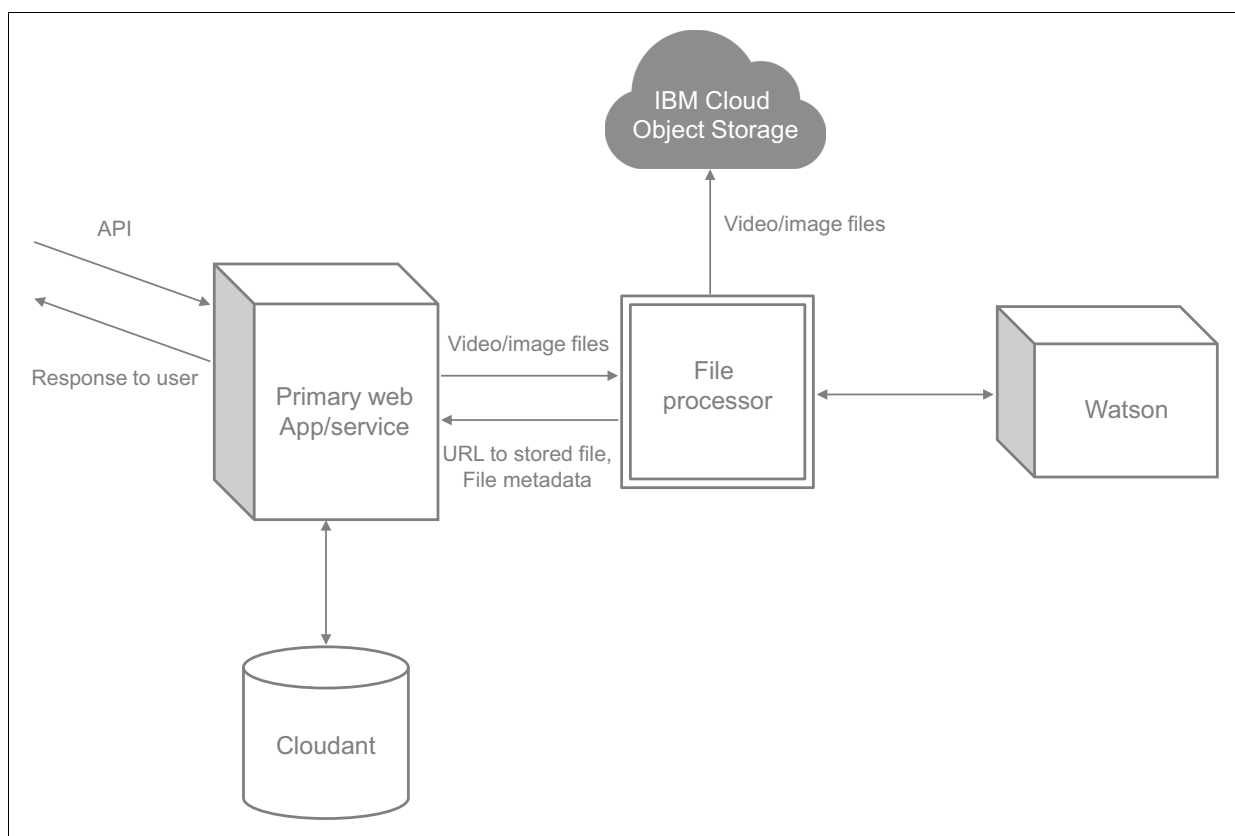


Figure 8-2 Architectural diagram of visual analytics within VideoRecon

For audio analytics in VideoRecon, the platform uses a similar microservices approach and flow. First, audio files are passed into VideoRecon through our API, but this time in a .zip file format, if using batch upload for multiple files. The primary VideoRecon app service then takes the .zip file and sends it to the file processor microservice. The file processor then extracts the audio files and stores the native audio files on IBM Cloud Object Storage, where it can be accessed by the other microservices, including IBM Watson. IBM Watson Speech to Text can be used before or after being stored in IBM Cloud Object Storage, along with other Watson services, as requested. Like with visual analytics, we typically pass the audio files to Watson first in order to ensure that there isn't corruption before storing in IBM Cloud Object Storage. The URL to the audio file in IBM Cloud Object Storage and metadata are then passed back and stored temporarily on Cloudant. In parallel, the metadata tags for the audio file are then passed back through an API to the user.

Figure 8-3 shows the architectural diagram of audio analytics within VideoRecon.

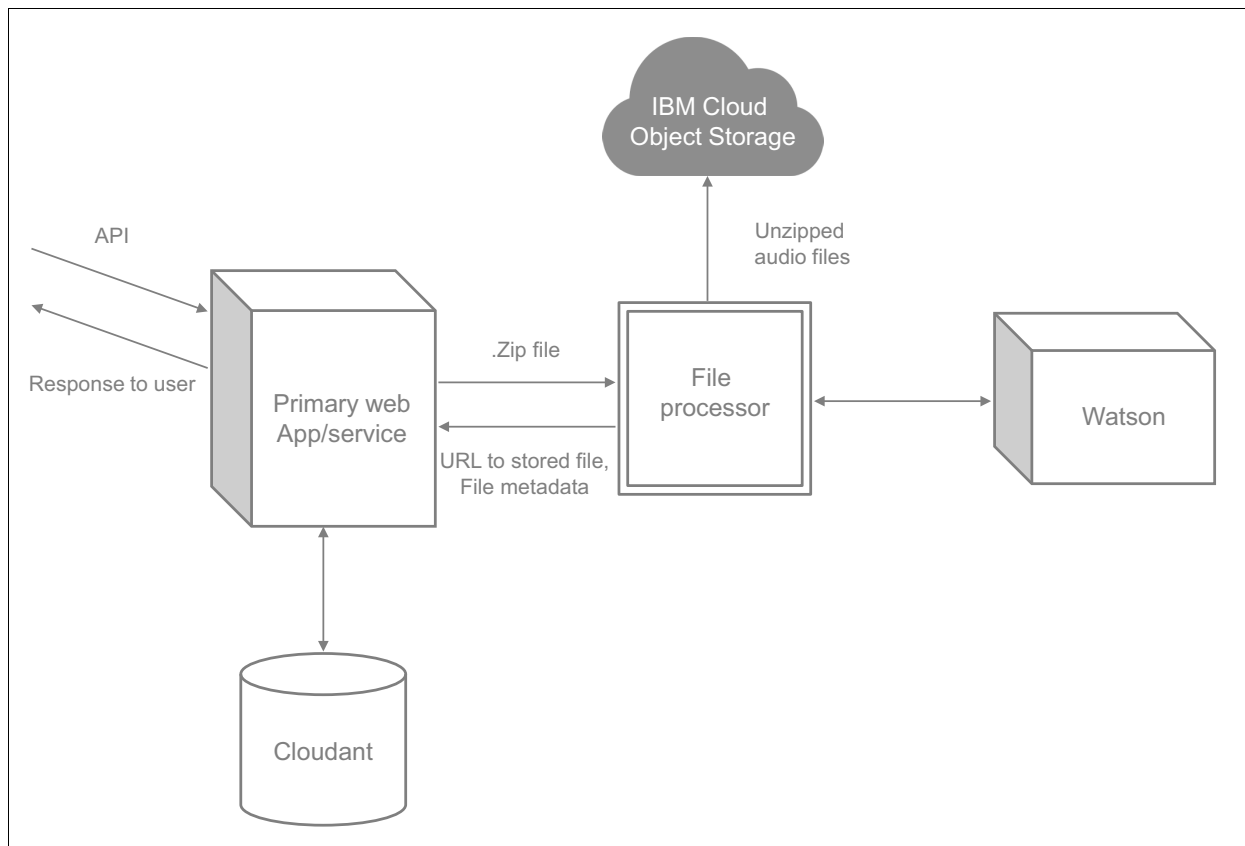


Figure 8-3 Architectural diagram of audio analytics within VideoRecon

8.3 Implementation of the scenario

As an example, code snippets from the audio analytics portion of VideoRecon are included. These code snippets show the steps in which the VideoRecon prototype will take audio files, extract them, and then upload them in native format (.wav files in this particular case) into the IBM Cloud Object Storage bucket called *audio*.

Example 8-1 shows the code snippets from the original prototype built in Node.js.

Example 8-1 Code snippets built in Node.js

```
1. var s3 = require('s3') var client = s3.createClient({
2.   maxAsyncS3: 20,
3.   s3RetryCount: 3,
4.   s3RetryDelay: 1000,
5.   multipartUploadThreshold: 20971520,
6.   multipartUploadSize: 15728640,
7.   s3options: {
8.     accessKeyId: 'XXXXXXXXXX',
9.     secretAccessKey: 'XXXXXXXXXXXXXXXXXXXX',
10.    region: 'us-standard',
11.    endpoint: 's3-api.us-geo.objectstorage.softlayer.net',
12.    sslEnabled: true
13.  }
```

```

14.});
15.var clientS3 = client.s3;
16.
17.function unzipFile(f, complete) {
18.    var zip = new admzip(f);
19.    var zipEntries = zip.getEntries();
20.    zip.extractAllTo(storeDir, true);
21.    count = 0;
22.    fs.readdirSync(storeDir).forEach(function(file) {
23.        var filepath = path.join(storeDir, file);
24.        fs.stat(filepath, function(error, stat) {
25.            if (error) {
26.                console.error('Error stating file: ' + error);
27.                return;
28.            }
29.            if (stat.isFile() && isSupportedAudioFile(filepath)) {
30.                count += 1;
31.                getFileParameters(filepath, (res) => {
32.                    var result = res.hasOwnProperty("error") if (result) {
33.                        complete(res);
34.                        return
35.                    }
36.                    var stats = fs.statSync(filepath).size;
37.                    var mimeType = audioMimeType(filepath) res['filename'] =
file;
38.                    complete(res);
39.                    var params = {
40.                        localFile: filepath,
41.                        ACL: 'public-read-write',
42.                        s3Params: {
43.                            Bucket: 'audio',
44.                            Key: file
45.                        }
46.                    };
47.                    var uploader = client.uploadFile(params);
48.                    uploader.on('error', function(err) {
49.                        console.error("unable to upload:", err.stack);
50.                        res["error"] = err;
51.                        complete(res);
52.                    });
53.                    uploader.on('progress', function() {
54.                        console.log("progress");
55.                    });
56.                    uploader.on('end', function() {
57.                        res['url'] = file complete(res);
58.                    });
59.                });
60.            }
61.        });
62.    });
63.}

```

Also, Example 8-2 shows the code snippets from the optimized service that performs the same function, but this time written in Server-side Swift.

Example 8-2 Code snippets written in Server-side Swift

```
import Foundation
import RestKit      // Found in IBM's Kitura samples
import S3SignerIBM  // Can be retrieved at
https://github.com/BlueChasm/S3SignerIBM.git

func retrieveFromCloudObjectStore(name: String,
                                   container: String,
                                   completionHandler: @escaping (AudioFile?, JSON?)
-> Void) {

    let (s3Signer, endpoint, errorString) =
getS3AuthorizationHeaderWithContainer(container: container, name: name)

    if let error = errorString {
        print(error)
        return
    }

    guard let signer = s3Signer,
          let url = endpoint else {
        let error = "Unable to get cloud object authorization header"
        print(error)
        return
    }

    do {
        let headers = try signer.authHeaderV4(httpMethod: .get,
                                              urlString: url,
                                              headers: [:],
                                              payload: .none)

        let request = RestRequest(
            method: .GET,
            url: url,
            headerParameters: headers
        )

        request.responseData { resp in
            switch resp {
            case .success(let data):
                let audioFile = AudioFile(name: name, data: data)
                completionHandler(audioFile, nil)
            case .failure(let error):
                print(error)
                var errorResponse = JSON([:])
                errorResponse["error"].stringValue = "Error getting audio file
parameters."
                completionHandler(nil, errorResponse)
            }
        }
    }
}
```



```

    } catch (let error) {
        print(error)
    }
}

func deleteFromCloudObjectStore(name: String,
                                container: String,
                                completionHandler: @escaping (Bool) -> Void) {

    let (s3Signer, endpoint, errorString) =
getS3AuthorizationHeaderWithContainer(container: container, name: name)

    if let _ = errorString {
        completionHandler(false)
        return
    }

    guard let signer = s3Signer,
        let url = endpoint else {
        completionHandler(false)
        return
    }

    do {
        let headers = try signer.authHeaderV4(httpMethod: .delete,
                                              urlString: url,
                                              headers: [:],
                                              payload: .none)

        let request = RestRequest(
            method: .DELETE,
            url: url,
            headerParameters: headers
        )

        request.responseJSON { resp in
            switch resp {
            case .success(_):
                completionHandler(true)
            case .failure(_):
                completionHandler(false)
            }
        }

    } catch (let error) {
        print(error)
    }
}

private func getS3AuthorizationHeaderWithContainer(container: String,
                                                  name: String) ->
(S3SignerIBM?, String?, String?) {

    guard let accessKey = cloudObjectStorageProps["accessKey"],

```

```

    let secretKey = cloudObjectStorageProps["secretKey"],
    let region = cloudObjectStorageProps["region"] else {
        return (nil, nil, "Error getting cloud object storage parameters.")
    }

    guard let cloudRegion = Region(rawValue: region) else {
        return (nil, nil, "Error getting cloud object storage region.")
    }

    let host = cloudRegion.host
    let endpoint = "https://\(host)/\(container)/\(name)"

    let s3Signer = S3SignerIBM(accessKey: accessKey,
                               secretKey: secretKey,
                               region: cloudRegion)

    return (s3Signer, endpoint, nil)
}

```

8.4 Verification of the scenario

To test the results of the code, a sample run verifies that these sample files are uploaded correctly in IBM Cloud Object Storage. Complete these steps:

1. Launch the Bluemix Catalog and select the **Cloud Object Storage** module. An overview of the service open in the Bluemix Catalog dashboard (Figure 8-4).

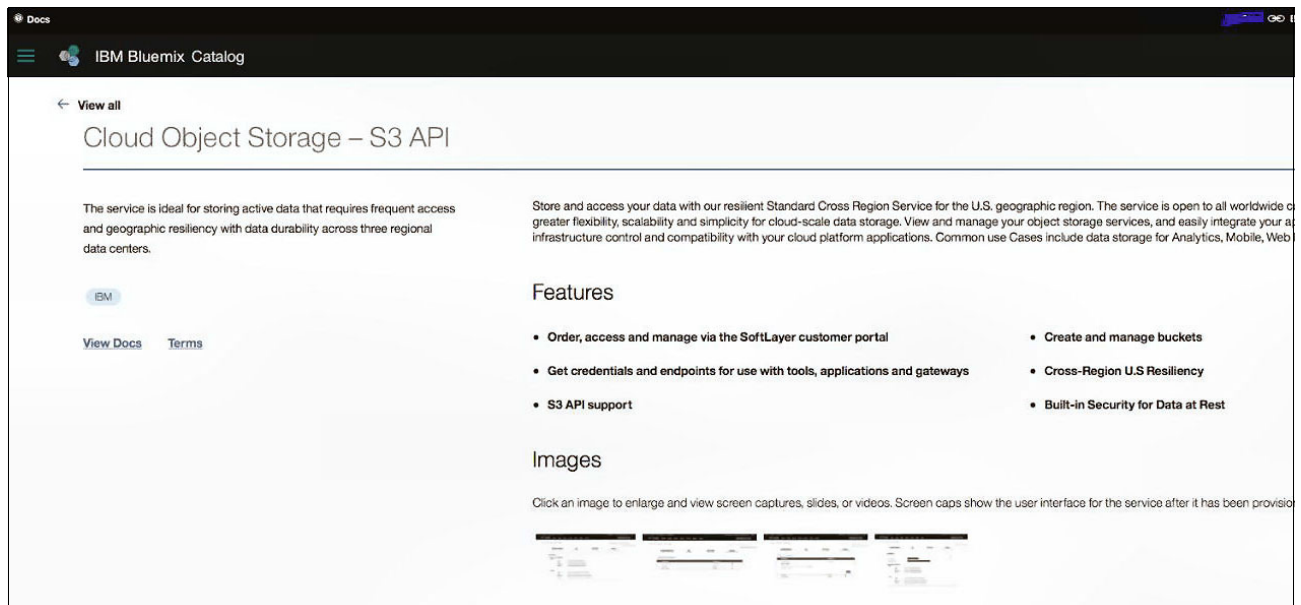


Figure 8-4 IBM Cloud Object Storage in the Bluemix Catalog dashboard

- Under Bluemix Infrastructure, select the **Storage** option under the Dashboard menu (Figure 8-5).

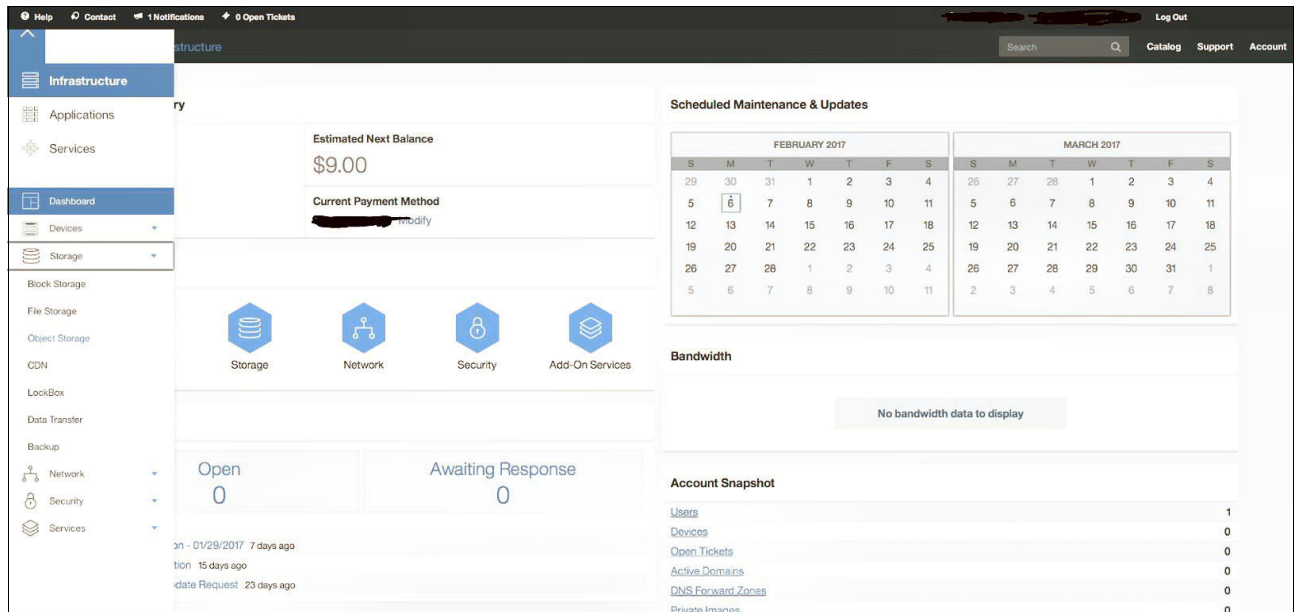


Figure 8-5 Storage tab under Dashboard in Bluemix Infrastructure

- Under Object Storage, click the account that was created previously (Figure 8-6).



Figure 8-6 Cloud Object Storage account name

- The Cloud Object Storage dashboard shows the bucket **audio** that was created (Figure 8-7). A *bucket* is a logical grouping of objects that are organized according to the preference of the user. In this case, the audio bucket will hold the native audio files within this platform. Click the **audio** bucket

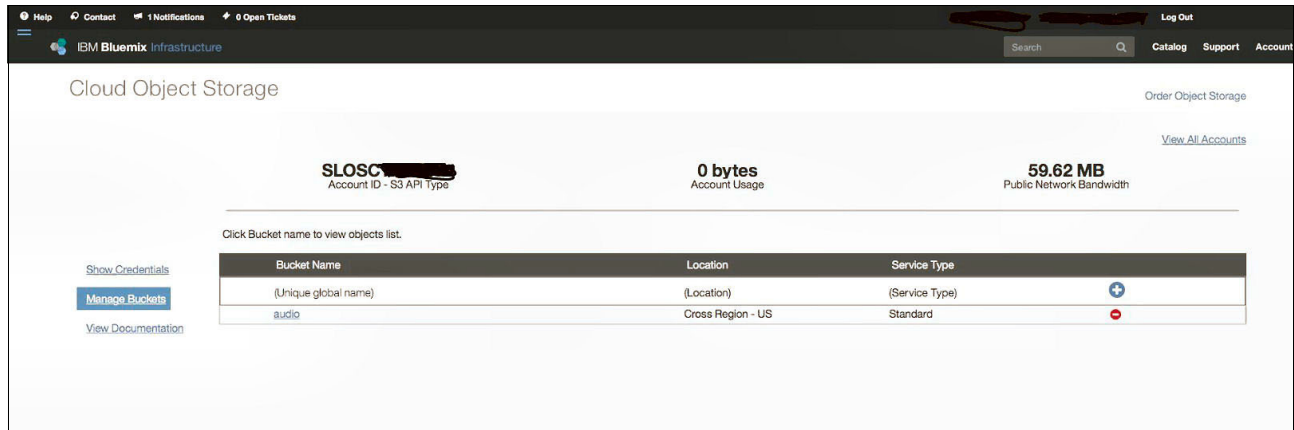


Figure 8-7 Cloud Object Storage buckets

- You see three .wav files, which are the native audio files that VideoRecon will then use for its cognitive analytics (Figure 8-8).

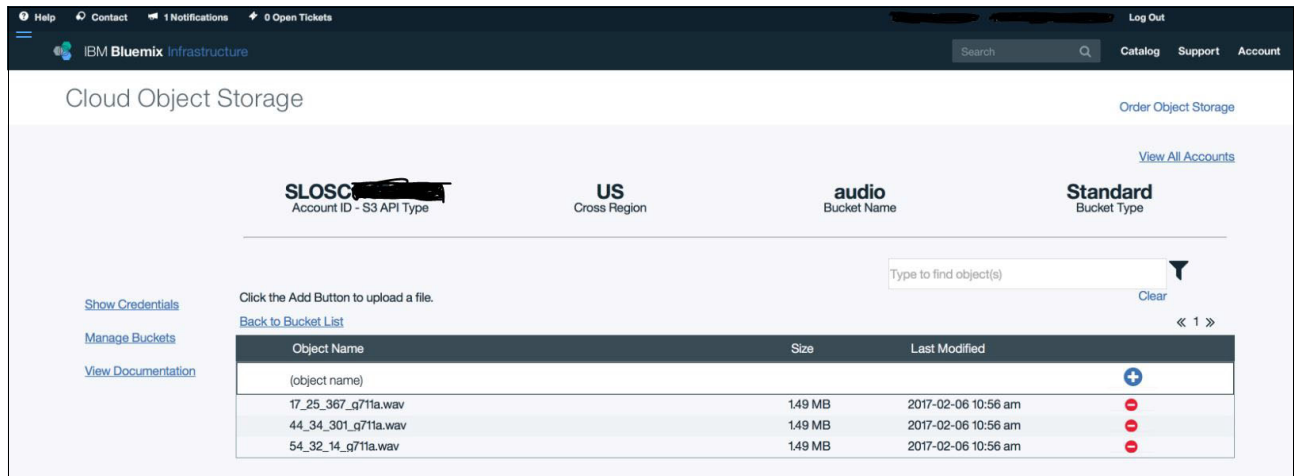


Figure 8-8 Audio files stored in the Cloud Object Storage audio bucket

This verifies that Cloud Object Storage has successfully stored files in it that were sent to it from the file processor microservice.

8.5 Conclusion

In this chapter we presented the VideoRecon Video Analytics solution that enables enterprises, service providers, institutions, and SaaS and software partners to gain actionable cognitive insights on their video content. We discussed why video analytics matter relative to digital initiatives, how the prototype to VideoRecon was built using IBM Cloud Object Storage and IBM Bluemix, and some of the basic steps of how this platform and its microservices use Cloud Object Storage to store key data.



Cognitive Call Center

This scenario describes the prototype of the Cognitive Call Center platform from BlueChasm, an IBM Business Partner. This platform enables companies to gain actionable cognitive insights on their recorded telephone calls.

The following topics are covered in this chapter:

- ▶ 9.1, “Scenario description” on page 184
- ▶ 9.2, “Architecture of the scenario” on page 185
- ▶ 9.3, “Implementation of the scenario” on page 186
- ▶ 9.4, “Verification of the scenario” on page 190
- ▶ 9.5, “Conclusion” on page 192

9.1 Scenario description

The BlueChasm Cognitive Call Center enables enterprises, service providers, institutions, and software as a service (SaaS) and software partners to gain actionable cognitive insights on their recorded telephone calls, whether the audio content is a repository of recorded phone calls or near real-time phone calls that were recently recorded. This case study describes, at a high-level, why call center cognitive analytics matter relative to digital initiatives, how this prototype to Cognitive Call Center was built using IBM Cloud Object Storage and IBM Bluemix, and some of the basic steps of how this platform and its microservices use Cloud Object Storage to store key data.

9.1.1 Business case

Cognitive Call Center is a cloud-native, audio analytics platform that enables enterprises and service providers to use cognitive computing, anchored by IBM Watson, to transcribe and offer cognitive insights on recorded phone calls. Cognitive Call Center is currently served from the IBM Cloud on IBM Bluemix and is available for consumption by enterprises, service providers, institutions, and SaaS and software partners through the Cognitive Call Center's open API.

Cognitive Call Center was designed from the beginning to solve some of the major business challenges behind extracting insights and business value from recorded phone calls. Most enterprises today have a call or contact center that they use to interact with customers, partners, suppliers, and other major stakeholders. Even for those that do not have formal call centers, most take advantage of recorded phone calls for some degree of business benefit, whether it be for training, compliance, audit, or other related purpose. Most enterprises today that record phone calls do not employ more than just a basic layer of analytics around those calls and/or just simply store them for future use.

Beyond just adequately storing the calls, which can be challenging, a number of other challenges prevent these calls from being fully analyzed, understood, and acted upon for business advantage or used to create new digital business models for enterprises.

The following list describes some of those unsolved challenges for analyzing calls:

- ▶ Large volume of existing and constantly newly generated audio content and phone calls, which do not have much metadata or many tags associated with them that could be used for the search, filter, or analysis of phone calls.
- ▶ Metadata and tagging processes today for calls that are manually handled, with manpower required to listen to calls and tag or label them with non-uniform human judgement between taggers.
- ▶ Long turnaround times for attaching metadata to phone calls for analysis, because humans must currently take the time to listen to the phone calls for tagging.
- ▶ Extra cost of paid services either internal or with a third-party to listen to and analyze calls.
- ▶ Lack of manpower to listen to and analyze recorded calls when many calls must be listened to or analyzed in parallel, either in near real-time or for a relatively quick turnaround.
- ▶ Lack of accuracy of metadata tags and recognition of key words or themes in calls due to natural human error, fatigue, and motivation, especially in the context of constantly and rapidly changing business requirements.

Organizations that are able to take advantage of call center cognitive analytics are able to derive the following benefits:

- ▶ Significantly faster speed-to-market and iterations of use cases that involve recorded phone calls and actions that can result in cognitive insights, as in these three examples:
 - Better training for call center employees based on their tone or demeanor in working with customers
 - A 90% or more improvement in time-to-value of the ability to search and filter through phone calls, due to the call transcription from Cognitive Call Center
 - Improved capabilities regarding search, audit, and compliance use cases where cognitive analytics can help more quickly guide humans to content within calls
- ▶ More accurate metadata tags for recorded calls that can augment and supplement human judgement.
- ▶ Look for complex patterns within calls or across large numbers of calls.
- ▶ Greater cost-effectiveness in analyzing and tagging recorded calls, which will allow cognitive call analytics to be more widely accessible to more enterprises and their growing use cases.

9.2 Architecture of the scenario

The Cognitive Call Center prototype is a cloud-native platform built on the IBM Cloud with a microservices approach. Each microservice is an application that is housed in its own IBM Container on Bluemix and can be deployed, iterated on, and scaled independently, using all the advantages of Docker with some value-added features from IBM.

Cognitive Call Center processes and analyzes audio content through its API as part of its platform. From an architectural perspective, audio files are ingested either in single file format or in a batch file with multiple audio files and are analyzed with different microservices.

Note: Although Cloud Object Storage is part of the default architecture, users do have the ability to opt out of storing files on the platform (even temporarily). However, users can also choose to use Cloud Object Storage to store these files for an extended period of time.

For audio analytics of recorded phone calls in Cognitive Call Center, the platform uses an API-driven approach to ingest the data and uses a microservices design and flow. First, audio files are passed into Cognitive Call Center through an API, in a .zip file format, if using batch upload for multiple files. If not using batch upload, then a single audio file is passed in. This primary Cognitive Call Center app service then takes the .zip file and sends it to the file processor microservice. The file processor then extracts the audio files and stores the native audio files on IBM Cloud Object Storage, where it can be accessed by the other microservices, including IBM Watson. IBM Watson Speech to Text can be used before or after being stored in IBM Cloud Object Storage, along with other Watson services, as requested. As with visual analytics (such as the VideoRecon platform described in Chapter 8, “VideoRecon Video Analytics” on page 171), the audio files typically are passed to Watson first, in order to ensure that corruption does not exist, before storing in IBM Cloud Object Storage. The URL to the audio file in IBM Cloud Object Storage and metadata are then passed back and stored temporarily on Cloudant. In parallel, the metadata tags for the audio file are then passed back through an API to the user.

Figure 9-1 shows the architecture of this scenario.

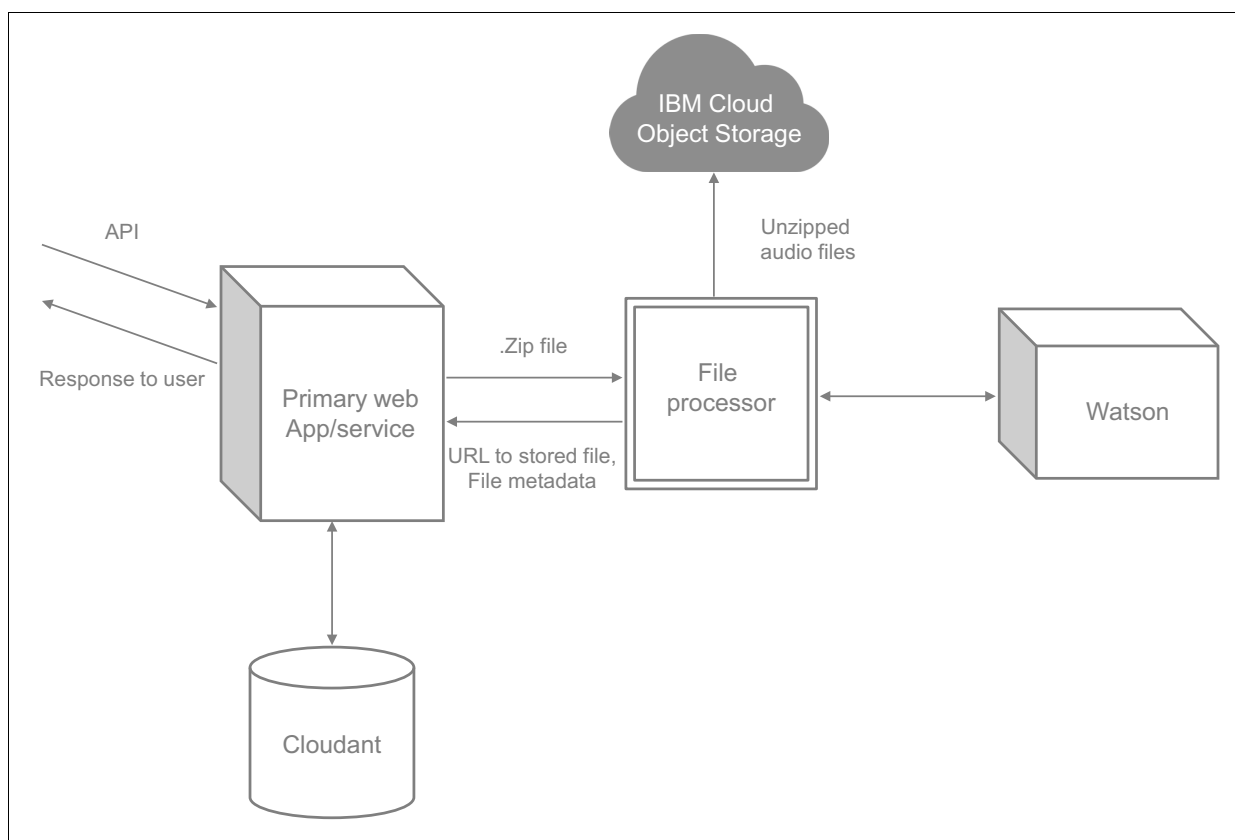


Figure 9-1 Architecture of the scenario

The prototype was originally created in Node.js, but was then optimized with Server-side Swift, as BlueChasm moved to a commercial enterprise product. For the purposes of this publication, example code is included as part of the service that interfaces with IBM Cloud Object Storage, specifically in both Node.js and Server-side Swift.

Audio: The audio design and most of the code used for Cognitive Call Center is similar to the audio analytics service that is used in VideoRecon, our video analytics platform and what we document in Chapter 8, “VideoRecon Video Analytics” on page 171.

9.3 Implementation of the scenario

As an example, code snippets from the prototype version of Cognitive Call Center are included here.

Note: the code used in this commercial enterprise offering for Cognitive Call Center does differ from what you see here. These code snippets show the steps in which the Cognitive Call Center prototype will take audio files, extract (unzip) them, and then upload them in native format (.wav files in this particular case) into the IBM Cloud Object Storage bucket named 'audio' as shown in the example.

Example 9-1 shows the code snippets from the original prototype that was built in Node.js.

Example 9-1 Code snippets from our original prototype built in Node.js

```
1. var s3 = require('s3') var client = s3.createClient({
2.   maxAsyncS3: 20,
3.   s3RetryCount: 3,
4.   s3RetryDelay: 1000,
5.   multipartUploadThreshold: 20971520,
6.   multipartUploadSize: 15728640,
7.   s3options: {
8.     accessKeyId: 'XXXXXXXXXX',
9.     secretAccessKey: 'XXXXXXXXXXXXXXXXXXXX',
10.    region: 'us-standard',
11.    endpoint: 's3-api.us-geo.objectstorage.softlayer.net',
12.    sslEnabled: true
13.  }
14.});
15.var clientS3 = client.s3;
16.
17.function unzipFile(f, complete) {
18.  var zip = new admzip(f);
19.  var zipEntries = zip.getEntries();
20.  zip.extractAllTo(storeDir, true);
21.  count = 0;
22.  fs.readdirSync(storeDir).forEach(function(file) {
23.    var filepath = path.join(storeDir, file);
24.    fs.stat(filepath, function(error, stat) {
25.      if (error) {
26.        console.error('Error stating file: ' + error);
27.        return;
28.      }
29.      if (stat.isFile() && isSupportedAudioFile(filepath)) {
30.        count += 1;
31.        getFileParameters(filepath, (res) => {
32.          var result = res.hasOwnProperty("error") if (result) {
33.            complete(res);
34.            return
35.          }
36.          var stats = fs.statSync(filepath).size;
37.          var mimeType = audioMimeType(filepath) res['filename'] = file;
38.          complete(res);
39.          var params = {
40.            localFile: filepath,
41.            ACL: 'public-read-write',
42.            s3Params: {
43.              Bucket: 'audio',
44.              Key: file
45.            }
46.          };
47.          var uploader = client.uploadFile(params);
48.          uploader.on('error', function(err) {
49.            console.error("unable to upload:", err.stack);
50.            res["error"] = err;
51.            complete(res);
52.          });
```

```

53.         uploader.on('progress', function() {
54.             console.log("progress");
55.         });
56.         uploader.on('end', function() {
57.             res['url'] = file complete(res);
58.         });
59.     });
60. }
61. });
62. });
63.}

```

Example 9-2 shows the code snippets from the optimized prototype that performs the same function, but this time written in Server-side Swift.

Example 9-2 Code snippets that performs the same function, but this time written in Server-side Swift

```

import Foundation
import RestKit      // Found in IBM's Kitura samples
import S3SignerIBM  // Can be retrieved at https://github.com/BlueChasm/S3SignerIBM.git

func retrieveFromCloudObjectStore(name: String,
                                   container: String,
                                   completionHandler: @escaping (AudioFile?, JSON?) -> Void) {

    let (s3Signer, endpoint, errorString) =
    getS3AuthorizationHeaderWithContainer(container: container, name: name)

    if let error = errorString {
        print(error)
        return
    }

    guard let signer = s3Signer,
          let url = endpoint else {
        let error = "Unable to get cloud object authorization header"
        print(error)
        return
    }

    do {
        let headers = try signer.authHeaderV4(httpMethod: .get,
                                              urlString: url,
                                              headers: [:],
                                              payload: .none)

        let request = RestRequest(
            method: .GET,
            url: url,
            headerParameters: headers
        )

        request.responseData { resp in
            switch resp {
            case .success(let data):
                let audioFile = AudioFile(name: name, data: data)
                completionHandler(audioFile, nil)
            case .failure(let error):
                print(error)
                var errorResponse = JSON([:])
            }
        }
    }
}

```

```

        errorResponse["error"].stringValue = "Error getting audio file parameters."
        completionHandler(nil, errorResponse)
    }
}

} catch (let error) {
    print(error)
}
}

func deleteFromCloudObjectStore(name: String,
                                container: String,
                                completionHandler: @escaping (Bool) -> Void) {

    let (s3Signer, endpoint, errorString) =
getS3AuthorizationHeaderWithContainer(container: container, name: name)

    if let _ = errorString {
        completionHandler(false)
        return
    }

    guard let signer = s3Signer,
          let url = endpoint else {
        completionHandler(false)
        return
    }

    do {
        let headers = try signer.authHeaderV4(httpMethod: .delete,
                                              urlString: url,
                                              headers: [:],
                                              payload: .none)

        let request = RestRequest(
            method: .DELETE,
            url: url,
            headerParameters: headers
        )

        request.responseJSON { resp in
            switch resp {
            case .success(_):
                completionHandler(true)
            case .failure(_):
                completionHandler(false)
            }
        }

    } catch (let error) {
        print(error)
    }
}

private func getS3AuthorizationHeaderWithContainer(container: String,
                                                    name: String) ->
(S3SignerIBM?, String?, String?) {

```

```

guard let accessKey = cloudObjectStorageProps["accessKey"],
    let secretKey = cloudObjectStorageProps["secretKey"],
    let region = cloudObjectStorageProps["region"] else {
    return (nil, nil, "Error getting cloud object storage parameters.")
}

guard let cloudRegion = Region(rawValue: region) else {
    return (nil, nil, "Error getting cloud object storage region.")
}

let host = cloudRegion.host
let endpoint = "https://\(host)/\(container)/\(name)"

let s3Signer = S3SignerIBM(accessKey: accessKey,
                           secretKey: secretKey,
                           region: cloudRegion)

return (s3Signer, endpoint, nil)
}

```

9.4 Verification of the scenario

To test the results of the code, the following steps show a sample run and verification in IBM Cloud Object Storage in the Bluemix console, indicating that these sample files uploaded correctly.

1. Launch the Bluemix Catalog and select the Cloud Object Storage module for an overview of the service on Bluemix (Figure 9-2).

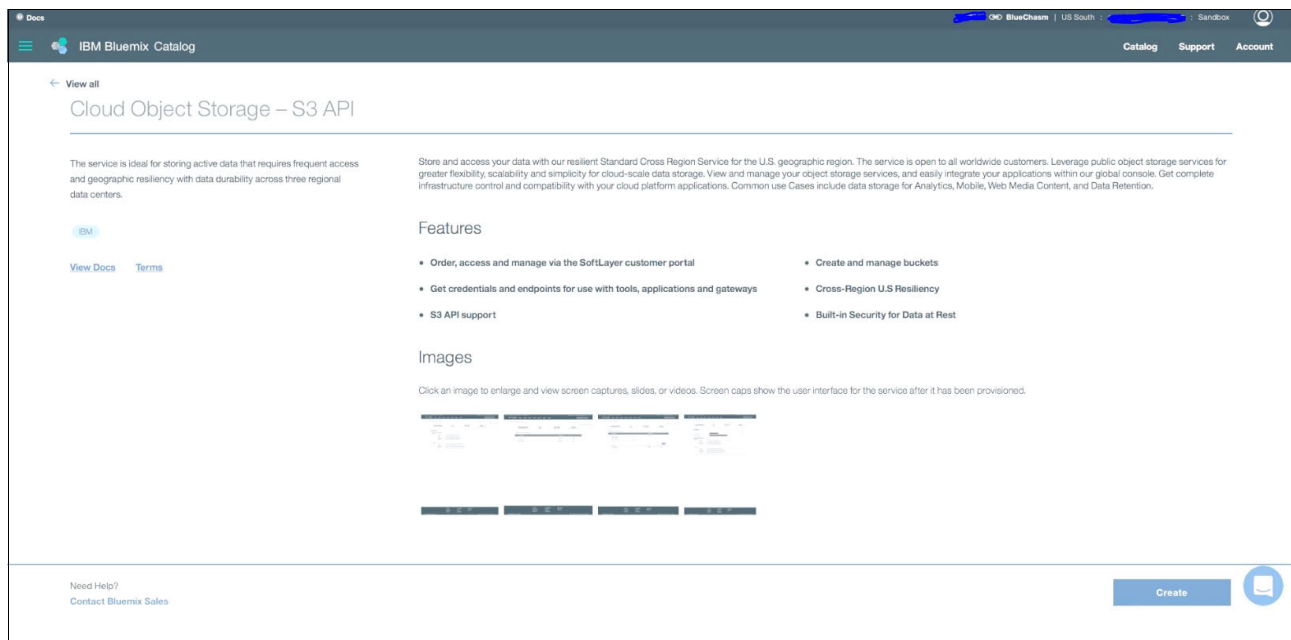


Figure 9-2 IBM Cloud Object Storage in the Bluemix Catalog dashboard

2. Under Bluemix Infrastructure, select the **Storage** option under the Dashboard tab (Figure 9-3).

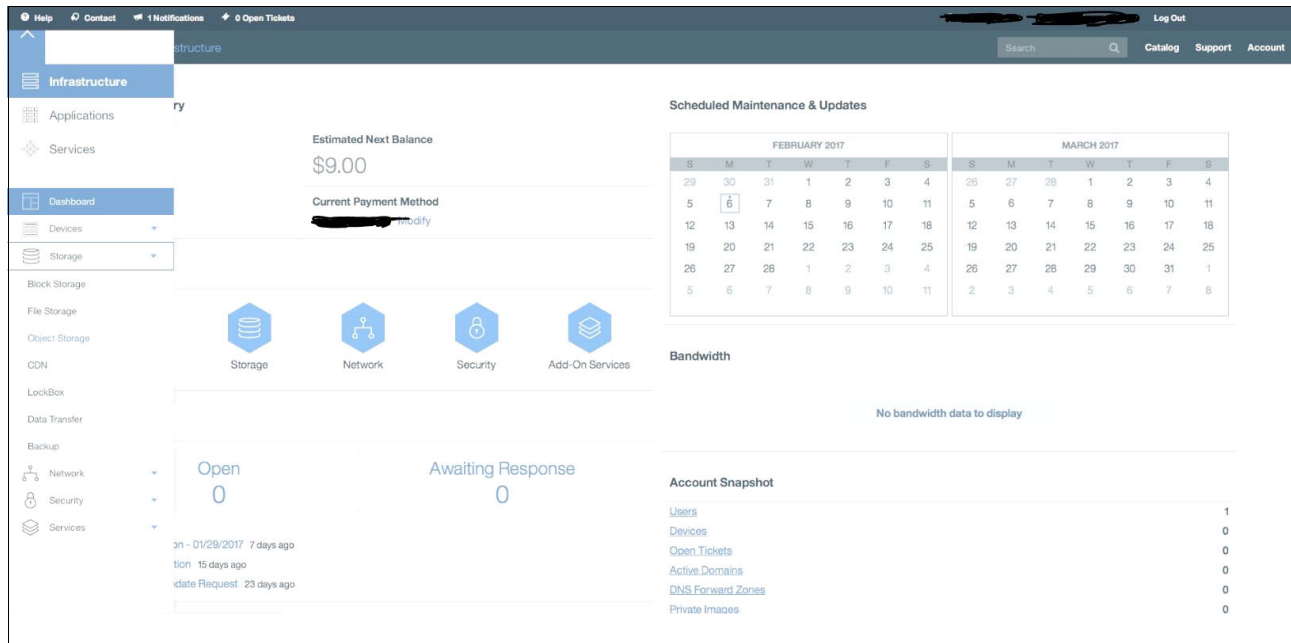


Figure 9-3 Storage tab under Dashboard in Bluemix Infrastructure

3. Under Object Storage, find and click the account that was created previously (Figure 9-4).



Figure 9-4 Cloud Object Storage account name

- The Cloud Object Storage dashboard then lists the bucket named **audio** that was created (Figure 9-5). A *bucket* is a logical grouping of objects that are organized according to the preference of the user. In this case, the audio bucket will hold the native audio files within the platform. Click the **audio** bucket.

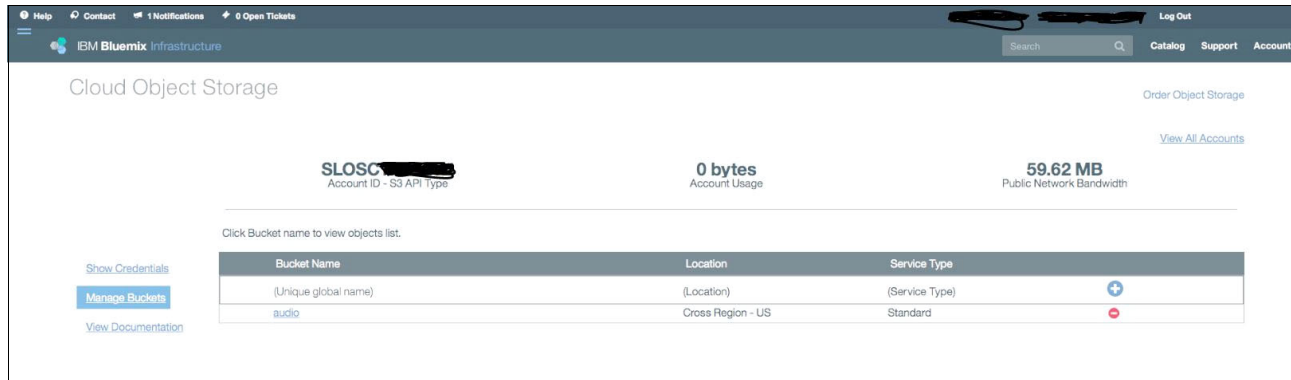


Figure 9-5 Cloud Object Storage buckets

- Note the three .wav files that are then listed (Figure 9-6). These files are the native audio files that Cognitive Call Center will use for its cognitive analytics. This verifies that Cloud Object Storage successfully stored files that were sent to it from the file processor microservice.

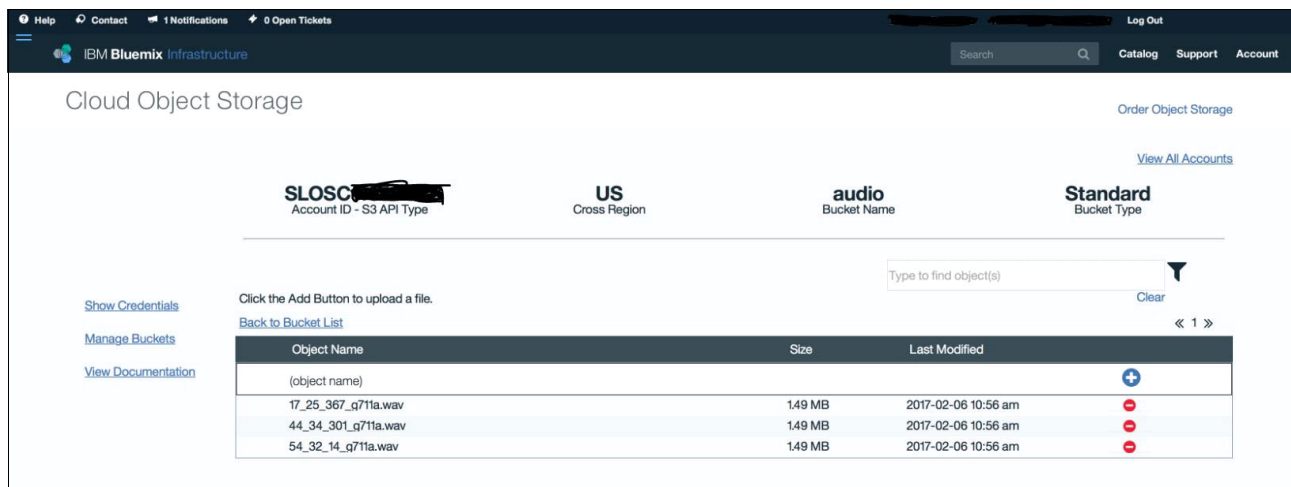


Figure 9-6 Audio files stored in the Cloud Object Storage "audio" bucket

9.5 Conclusion

This chapter presented the prototype of the Cognitive Call Center platform, which enables enterprises, service providers, institutions, and SaaS/software partners to gain actionable cognitive insights on their recorded telephone calls, whether the audio content is a repository of recorded phone calls or near real-time phone calls that were just recently recorded.



A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server:

<ftp://www.redbooks.ibm.com/redbooks/SG248385>

Alternatively, you can go to the IBM Redbooks website:

ibm.com/redbooks

Search for SG248385, select the title, and then click **Additional materials** to open the directory that corresponds with the IBM Redbooks form number, SG24-8385.

Using the web material

The additional web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG248385.zip	Compressed code examples for the scenarios described in the book

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space:	10 MB minimum
Operating system:	Windows, Linux

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material .zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Note that some publications listed here might be available in softcopy only.

- ▶ *Cloud Data Sharing with IBM Spectrum Scale*, REDP-5419
- ▶ *IBM Cloud Object Storage Concepts and Architecture*. REDP-5435

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

You can find more information online:

- ▶ Public IBM Cloud Object Storage documentation:
<https://ibm-public-cos.github.io/crs-docs/>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Cloud Object Storage as a Service: IBM Cloud Object Storage from Theory to Practice

SG24-8385-00

ISBN 0738442453



(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



SG24-8385-00

ISBN 0738442453

Printed in U.S.A.

Get connected

