IBM Storage Solutions for Blockchain Platform
Version 1.2

IBM Storage Team

IBM® Storage
Contents

About this document .......................................................... 1
Executive summary ......................................................... 2
Support for the Blueprint and its configurations ..................... 3
  Support of IBM Spectrum Connect .................................. 3
  Requesting assistance ................................................... 3
Scope ............................................. 3
What's new in Version 1.2 ................................................. 4
Blockchain solution reference architecture ................................ 7
  Sample Configuration .................................................. 7
Solution architecture data paths........................................ 8
Getting started: IBM Storage Solutions for Blockchain Platform on IBM Cloud Private ......................... 8
  IBM Cloud Private on IBM Z installation .......................... 8
  IBM Blockchain Platform on IBM Cloud Private installation .......... 9
  Configuration .......................................................... 11
  Creating a peer ......................................................... 23
  IBM Cloud Object Storage installation ............................... 41
  IBM Blockchain Document Store ..................................... 42
Changing from Cloud COS Storage to Local, Onsite COS Storage for off-chain data .................. 54
Summary ................................................................. 64
For more information ...................................................... 64

Notices ........................................................................ 65
Trademarks ..................................................................... 66
Terms and conditions for product documentation .................... 67
  Applicability ............................................................. 67
  Commercial use .......................................................... 67
  Rights ......................................................................... 67
  Privacy policy considerations ......................................... 67
About this document

This Blueprint is intended to define the infrastructure that is required for a blockchain remote peer and to facilitate the deployment of IBM Blockchain Platform on IBM Cloud Private using that infrastructure. This infrastructure includes the necessary document handler components, such as IBM Blockchain Document Store, and covers the required storage for on-chain and off-chain blockchain data. To complete these tasks, you must have a basic understanding of each of the used components or have access the correct educational material to gain that knowledge.
Executive summary

IBM Blockchain enables more-efficient ledger models, streamlining business processes, and transactions, which reduces risk and increases trust. To gain a competitive advantage for your business, you need a reliable, secure, and flexible IT blockchain environment. This environment enables modern blockchain application enterprise workloads to scale as necessary to fit your needs. It also gives access to users, no matter what kind of endpoint device they are using.

Further, it should allow orchestration to suit your resource consumption requirements and minimize downtime. Your blockchain environment must provide reliable platform-as-a-service capabilities with flexible infrastructure. This means deploying a cloud-service fabric to reliably deliver containerized blockchain applications to your endpoints of choice to meet or exceed service-level expectations.

Organizations must also protect their data, whether for highly regulated industries or when building mission-critical blockchain applications. Getting to market quickly, iterating, and attracting new customers are top-of-mind for executives around the world.

Although cloud computing that uses blockchain is a major force in business innovation, challenges are everywhere. Your blockchain in the cloud is only as private and secure as the technology that protects it. As organizations implement modern blockchain application platforms, they are using technologies to deliver cloud-native workloads, provide stateful data services, and deliver enterprise-critical capabilities; from artificial intelligence and messaging to blockchain applications, DevOps, analytics, and high-performance computing.

To this end, the full-stack IBM Blockchain Platform on IBM Cloud Private cloud solution that uses IBM Blockchain Data Store as described in this Blueprint delivers a private cloud fabric for building and managing on-premises, containerized blockchain applications that can deliver scale, performance, security, and data-protection. They also can extend across hybrid and multicloud environments to fill your most critical application requirements. The possibilities are endless, and real-time decision-making is within reach.

Blockchain architectures require flexibility at all component levels. To maximize the business effectiveness of your blockchain network or peer, you want to take advantage of cloud-based and on-premises storage solutions for off-chain data.

With the heightened concerns over data privacy and controls of Personally Identifiable Information, a hybrid multicloud off-chain storage with adequate data management is the more attractive solution. Therefore, any blockchain solution should consider the needs of a hybrid cloud/onsite storage requirement. The possibilities are endless, and real-time decision-making is within reach.
Support for the Blueprint and its configurations

IBM Storage Solutions for Blockchain provides an integrated support experience for clients. The information in this document (referred to throughout as “the Blueprint”) is distributed on an “as is” basis without any warranty that is either expressed or implied. Support for the underlying components that make up this solution are provided by way of the standard procedures and processes that are available for each of those components, as governed by the support entitlement that is available for those components. For more information about these components, see “Prerequisites” on page 4.

Support of IBM Spectrum Connect

Support assistance for the use of this material is limited to situations where IBM Spectrum Connect support is entitled and where the issues are not specific to a Blueprint implementation. Support of the underlying IBM Spectrum Connect components is entitled and provided as an extension of the related Storage hardware and system software. For more information about how to request assistance and support for the IBM Spectrum Connect components, see the hardware and system software documentation.

Requesting assistance

All components of the solutions are part of this unified support structure. Support assistance of the solution that is described in this Blueprint is available by requesting assistance for any of the components in the solution and is the preferred method. Support assistance can also be requested from the IBM Blockchain Platform when the source of the issue cannot be determined.

Scope

This Blueprint provides the following features:

- A solutions architecture and the related storage endpoint capabilities that interact with the following software and hardware components:
  - IBM Cloud Private 3.1.1
  - IBM LinuxONE
  - IBM FlashSystem 9100
  - IBM Storwize V7000
  - IBM Storwize V5000
  - IBM DS8000
  - IBM Cloud Object Storage
  - IBM Spectrum Connect 3.6.1
  - IBM Storage Enabler for Containers
  - IBM Blockchain platform for IBM Cloud Private 1.1.1
- Detailed technical configuration steps for building an end-to-end blockchain peer on IBM private cloud solution with persistent storage

This technical report does not include the following features:

- Provide performance analysis or metrics for user consumption
- Replace any official manuals and documents that are issued by IBM for related products
- Explain the installation and configuration of VMware vSphere
What's new in Version 1.2

The documentation for the Blueprint configuration, hardware, and software requirements has been updated for the use of:

- IBM Cloud Object Storage on premises
- IBM Blockchain cloud peer for test and validation

Prerequisites
This Blueprint assumes familiarity with and basic knowledge of the following areas:

- IBM Cloud Private 3.1.1 or later
- IBM Spectrum Connect 3.6.1 or later
- IBM Storage Enabler for Containers Version 2.0 or later
- IBM LinuxONE
- IBM z/VM hypervisor
- IBM FlashSystem 9100, IBM Storwize V7000, IBM Storwize V5000, or IBM DS8000
- Linux-Ubuntu OS
- IBM Cloud Object Storage

Next, we highlight key components of the overall architecture. We suggest that you take the time to familiarize yourself with these components before you start the installation process.

IBM Blockchain Platform
The IBM Blockchain Platform for IBM Cloud Private offering is based on Kubernetes, which allows users to deploy Certificate Authorities (CAs), orderers, and peers on x86, LinuxONE, and IBM Z. IBM Blockchain Platform for IBM Cloud Private is based on Hyperledger Fabric v1.2.1 and is deployed by using Kubernetes Helm charts.

IBM Blockchain Platform for IBM Cloud Private delivers the components that you need to run a blockchain network on your own infrastructure through IBM Cloud Private. The components include Hyperledger Fabric, a Certificate Authority (CA), an orderer, and a peer, which you deploy, manage, and set up by using Kubernetes Helm charts.

This offering is intended for customers with advanced Hyperledger Fabric experience. IBM Blockchain Platform for IBM Cloud Private enables blockchain networks to be deployed on a private cloud to address data residency requirements, market regulations, and infrastructure preference. It simplifies the deployment of essential elements of a blockchain network in your own infrastructure through IBM Cloud Private, which is a Kubernetes-based application platform for developing and managing on-premises, containerized applications. Consider the following points:

- Enables clients to manage IBM Blockchain Platform networks with their own infrastructure. A free Community Edition allows customers to run in their own isolated and secure environments, but no support is provided.
- Enables clients to configure Fabric on Kubernetes by using Helm charts and detailed documentation for operations.
- Entitles clients with advanced technical support, unless you use the Community Edition.

IBM Blockchain Platform for IBM Cloud Private is a bundled product for IBM Cloud Private customers to deploy blockchain components in their local environment. After you import the Helm chart, you can find it as an IBM Blockchain Platform tile in the IBM Cloud Private Catalog.
For more information about IBM Blockchain Platform for IBM Cloud Private, see this IBM Cloud web page:


**IBM Blockchain Document Store**
The Blockchain Document Store is an IBM provided cloud service that allows secure sharing of documents across multiple participants on a permissioned-blockchain network. It provides an abstraction for handling documents, such as files (text, PDF, JPG, and so on) and JSON that uses APIs.

It also maintains proof of the existence of the documents by using the immutable property of blockchain and supports verification and secure sharing of these documents. The files are securely stored in the IBM Cloud Object Storage layer. The service, which is a series of APIs that is overlaid on the IBM Blockchain Platform infrastructure, demonstrates the use of IBM Cloud Object Storage as an off-chain storage medium.

**IBM Cloud Private**
Not all application workloads are suitable for the public cloud. In these cases, a private cloud can offer great benefits. A private cloud solution often is chosen for the following reasons:

- Some enterprises cannot tolerate the business disruption of the lengthy refactoring that is often needed to move applications off-premises.
- Many systems of record (traditional database and transactional applications) can include performance characteristics (such as less dynamic resource requirements) that do not benefit as much from cloud economics as do systems of engagement and insight (mobile, social, and analytics applications). Often, these systems feature residency, compliance, or performance needs that require them to run in dedicated on-premises infrastructure.
- Although cloud economics help save money with dynamic applications, applications with steady-state demand can cost more when they are running in public clouds.

For these and many other application workloads that operate best on-premises, IBM Cloud Private offers a leading-edge private cloud platform for developing and running workloads locally. It is an integrated environment that enables you to design, develop, deploy, and manage on-premises, containerized cloud applications behind your firewall.

It also accelerates the work of enterprise developers by providing access to valuable data and applications behind the firewall through a flexible container-based architecture and application programming interface (API)- based catalog of services. It includes a private image repository, management console, and monitoring frameworks.

IBM Cloud Private provides control of how and where applications use cloud services. It uses industry-standard open source technologies, such as Kubernetes, Docker, Helm, Terraform, Cloud Foundry, and more than 40 others.

It also provides integrated operational management and developer services, such as IBM MQ messaging for applications in distributed systems, a microservices framework builder, IBM DB2 Developer Edition, an IBM WebSphere Application Server runtime environment, and more. By using these services, enterprises can optimize older applications with cloud and containers for use with DevOps or analytics, create cloud-native applications, and open their data centers to work with cloud services.
IBM Cloud Private integrates various microservices (such as IBM Watson APIs) and middleware capabilities to help form a robust and responsive infrastructure. These capabilities can improve the overall integration and continued deployment of applications, while minimizing risks that are associated with performance bottlenecks and unpredictable scalability.

IBM Cloud Private helps drive enterprise transformation by providing developers with a choice of languages, frameworks, runtimes, and services to build cloud-native applications and microservices so that they can create their own cloud services. It accelerates innovation by facilitating the use of services, such as blockchain tracking and machine learning that developers can infuse into existing or new applications.

As of release 1.4.0, this Blueprint describes a set of other software packages and middleware support that are currently available as listed in Table 1.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Operating systems that are supported by IBM Cloud Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor</td>
<td>Operating system</td>
</tr>
<tr>
<td>Red Hat</td>
<td>Enterprise Linux (RHEL) 7.3, 7.4 and 7.5 (64-bit)</td>
</tr>
<tr>
<td>Canonical</td>
<td>Ubuntu 18.04 LTS and 16.04 LTS</td>
</tr>
<tr>
<td>SUSE</td>
<td>Linux Enterprise Server (SLES) 12 SP3</td>
</tr>
</tbody>
</table>

IBM Spectrum Connect 3.6.0
Today's organizations demand easy and fast integration of storage in multiple cloud environments. IBM Spectrum Connect empowers storage teams and other stakeholders by enabling provisioning, monitoring, automating, and orchestrating IBM block storage in containerized, VMware, and Microsoft PowerShell environments. It offers the same UI for many solutions and environments for a consistent experience. It also helps organizations simplify cloud complexity and is available by entitlement to every IBM block storage customer.

For more information about the supported IBM Storage Systems and respective microcode levels, see IBM Knowledge Center for IBM Spectrum Connect:


IBM Storage Enabler for Containers
IBM Storage Enabler for Containers allows IBM storage systems to be used as persistent volumes for stateful applications that are running in IBM Cloud Private clusters. IBM Storage Enabler for Containers v2.0 extends IBM Spectrum Connect v3.6 for IBM block storage and IBM Spectrum Scale for file storage, respectively, to Kubernetes-orchestrated container environments. For more information about supported operating systems tables, see IBM Storage Enabler for Containers Release Notes.

IBM Cloud Object Storage
IBM Cloud Object Storage (COS) provides a highly flexible set of architectures that allow for local, metro, and geo sharding architectures to be built. Whether you want a local configuration that provides sharding across local storage, a shard datastore that is configured across a metropolitan or campus area, or a full-fledged geo-sharded data store that guarantees data cannot be lost, IBM COS is the solution of choice.
For more information about IBM Cloud Object Storage, see the following website:
https://www.ibm.com/cloud/object-storage

Blockchain solution reference architecture

The solution that is shown in Figure 1 uses Kubernetes containers on IBM Cloud Private on IBM LinuxONE to provide worker nodes in which to install IBM Blockchain Platform. Storage enabler for containers provides creation, attachment, and mounting of storage to containers through interfacing with Spectrum Connect to communicate to the IBM Block Storage. A standard S3 interface is used to access the provided Cloud Object Storage buckets.

Sample Configuration

The architecture features the following supported components:

• Software:
  – IBM Cloud Private (version 3.1.1)
  – IBM Spectrum Connect 3.6.0
  – IBM Storage Enabler for Containers 2.0
  – IBM Blockchain Platform 1.1.1
  – IBM Blockchain Document Store

• Hardware:
  – IBM LinuxONE Rockhopper II (IBM Compute)
  – IBM FlashSystem 9100 (IBM Block Storage)
  – IBM Cloud Object Storage

• Network:
  – 16 Gbps Fibre Channel
  – 40 GB Ethernet
Solution architecture data paths

The complex nature of data flows in the blockchain environment is shown in Figure 2. At the blockchain internal or on-chain level, data flows from the participant, to a node, and to the consensus nodes. After it is approved, a node combines the transaction with other approved transactions into a block, which is crypto-signed and added to the blockchain that is in the cloud and, if the node is distributed, on local storage at the node.

Figure 2   Data paths of the solution architecture

The off-chain or sidedb dataflow goes from the participant to the node and is held there while the transaction goes through consensus. After the transaction is approved, the off-chain data (which might be block or file, depending on the application), is sent to off-chain storage.

Off-chain storage can be Cloud Object Storage in the cloud or local, or block type storage in a distributed database solution. For local storage, S3 compatible connections are provided or Storage Enablement for Containers is used to provide connection through Spectrum Connect.

Getting started: IBM Storage Solutions for Blockchain Platform on IBM Cloud Private

This section describes the end-to-end private cloud solution architecture to facilitate a smooth deployment experience.

IBM Cloud Private on IBM Z installation

To install IBM Cloud Private on IBM Z, follow the instructions in the most recent version of the document that is available at this web page:

Configure two worker nodes for the certificate authority (CA) and peer nodes of the IBM Blockchain Platform. If this peer is a stand-alone peer, you might need to configure another orderer node.

IBM Blockchain Platform on IBM Cloud Private installation

Complete the following steps:

1. Start the IBM Cloud Private console as a user with Cluster Administrator privileges. For example, the IBM Cloud Private Dashboard URL for our test installation is shown in Figure 3 (https://x.xx.xx.xx:8443):

   ![Figure 3 Logging in to IBM Cloud Private](image)

2. From the Console, select Manage → Namespace.

3. Select the Create Namespace option (see Figure 4) to create a namespace to install the IBM Blockchain Platform.

   ![Figure 4 Creating Namespace option](image)
Namespace names must be lowercase. Special characters, such as "." can be used. The namespace must have ibm-privileged-psp privilege or the package does not install. In our test, we installed a namespace that is called `ibp_on_icp`, as shown in Figure 5.

4. Select the **Catalog** option from the upper menu bar. From the right-side menu, select the **Blockchain** option (see Figure 6).
5. From the Blockchain versions that are provided, select the most recently released remote peer option, as shown in Figure 7.

Figure 7  Remote peer option

6. Click the Configuration tab at the top of the panel or click Configure in the lower right corner (see Figure 8).

Note: You can install only one component at a time. If you plan to build a blockchain network with all of these components, you must install a CA before you install an orderer and a peer. For more information about deploying these components, see the IBM Cloud Docs deployment guide Getting Started with IBM Blockchain Platform for IBM Cloud Private.

Figure 8  IBM Blockchain platform

Configuration

Select the component to install and complete the parameter fields. The tables that are described next list the configuration parameters for each component and their default values.
**General and global configuration parameters**
Complete the parameter configurations that are listed in Table 2 for either component to install.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helm release name</td>
<td>The name of your helm release.</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>Target namespace</td>
<td>Choose the Kubernetes namespace to install the Helm Chart.</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>Service account name</td>
<td>Enter the name of the service account that you use to run the pods.</td>
<td>Default</td>
<td>No</td>
</tr>
</tbody>
</table>

**CA configuration parameters**
Complete the parameter configurations that are listed in Table 3 for either component to install.

For more information about the CA configuration parameters, see the IBM Blockchain Platforms documentation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install CA</td>
<td>Select to install a CA</td>
<td>Cleared</td>
<td>No</td>
</tr>
<tr>
<td>CA name</td>
<td>Specify a name to use for the certificate authority.</td>
<td>SampleOrgCA</td>
<td>Yes</td>
</tr>
<tr>
<td>CA worker node architecture</td>
<td>Select your cloud platform architecture (ADM64 or S390X).</td>
<td>AMD64</td>
<td>Yes</td>
</tr>
<tr>
<td>CA database type</td>
<td>The type of database to store CA data. Only SQLite is supported.</td>
<td>SQLite</td>
<td>Yes</td>
</tr>
<tr>
<td>CA data persistence enabled</td>
<td>If checked, data is available when the container restarts. Otherwise, all data is lost if a failover or pod restart occurs.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td>CA use dynamic provisioning</td>
<td>Check to enable dynamic provisioning for storage volumes.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td>CA storage class name</td>
<td>Specify a unique storage class name. Otherwise, the default storage class in the cluster is used.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>CA existing volume claim</td>
<td>Specify the name of a Volume Claim and leave all other fields blank.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>CA selector label</td>
<td>Specify the Selector label for your Persistent Volume Claim (PVC).</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>CA selector value</td>
<td>Specify the Selector value for your PVC.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>CA storage access mode</td>
<td>Specify the storage access mode for the PVC.</td>
<td>ReadWriteMany</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Orderer configuration parameters
Complete the parameter configurations that are listed in Table 4 for either component to install.

For more information about the orderer configuration parameters, see the IBM Blockchain Platforms documentation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Orderer</td>
<td>Select to install an orderer.</td>
<td>Cleared</td>
<td>No</td>
</tr>
<tr>
<td>Orderer worker node architecture</td>
<td>Select your IBM Cloud Private worker node architecture (AMD64 or S390X).</td>
<td>Autodetected architecture based on your master node</td>
<td>Yes</td>
</tr>
<tr>
<td>Configuration Parameter</td>
<td>Description</td>
<td>Default</td>
<td>Required</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>Orderer configuration</strong></td>
<td>You can customize the configuration of the orderer. This information overwrites the content in the orderer configuration file; that is, orderer.yaml. Waiting on more instructions from dev.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Organization MSP secret</strong></td>
<td>Specify the name of the secret object that contains organization MSP certificates and keys.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer data persistence enabled</strong></td>
<td>Data is available when the container restarts. If cleared, all data is lost if a failover or pod restart occurs.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer use dynamic provisioning</strong></td>
<td>Check to enable dynamic provisioning for storage volumes.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer image repository</strong></td>
<td>Location of the Orderer Helm Chart. This field is autofilled to the installed path. If you use the Community Edition and do not have internet access, change this field to the location from where you downloaded the Fabric orderer image.</td>
<td>ibmcom/ibp-fabric-orderer</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer Docker image tag</strong></td>
<td>Autofilled to the version of the Orderer image.</td>
<td>1.2.1</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Orderer consensus type</strong></td>
<td>The consensus type of the ordering service.</td>
<td>SOLO</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Orderer organization name</strong></td>
<td>Specify the name that you want to use for the orderer organization.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer Org MSP ID</strong></td>
<td>Specify the name that you want to use for the MSP ID of the orderer organization.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer storage class name</strong></td>
<td>Specify a storage class name for the orderer.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer existing volume claim</strong></td>
<td>Specify the name of a Volume Claim and leave all other fields blank.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer selector label</strong></td>
<td>Specify the <strong>Selector label</strong> for your PVC.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer selector value</strong></td>
<td>Specify the <strong>Selector value</strong> for your PVC.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td><strong>Orderer storage access mode</strong></td>
<td>Specify the storage <strong>access mode</strong> for the PVC.</td>
<td>ReadWriteMany</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Orderer volume claim size</strong></td>
<td>Choose the size of disk to use, which must be at least 2 Gi.</td>
<td>8 Gi</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Peer configuration parameters
Complete the parameter configuration that is listed in Table 5 for either component to install.

For more information about the peer configuration parameters, see the IBM Blockchain Platform documentation.

Table 5  Peer configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Peer</td>
<td>Select to install a peer.</td>
<td>Cleared</td>
<td>No</td>
</tr>
<tr>
<td>Peer worker node architecture</td>
<td>Select your cloud platform architecture (AMD64 or S390x)</td>
<td>AMD64</td>
<td>Yes</td>
</tr>
<tr>
<td>Peer image repository</td>
<td>Location of the Peer Helm Chart. This field is autofilled to the installed path.</td>
<td>ibmcom/ibp-fabric-peer</td>
<td>Yes</td>
</tr>
<tr>
<td>Peer Docker image tag</td>
<td>Autofilled to the version of the Peer image.</td>
<td>1.2.1</td>
<td>Yes</td>
</tr>
<tr>
<td>Peer configuration</td>
<td>You can customize the configuration of the peer. This information overwrites the content in the peer configuration file; that is, core.yaml.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Peer configuration secret (Required)</td>
<td>Name of the Peer configuration secret that you created in IBM Cloud Private.</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>Organization MSP (Required)</td>
<td>This value can be found in Network Monitor (IBP UI) by clicking Remote Peer Configuration in the Overview window. If you are not connecting to an IBP network, you can create an Organization MSP value, such as &quot;org1&quot; or specify an Organization MSP of which the peer is a part.</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Value</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Peer service type</td>
<td>Used to specify whether external ports should be exposed on the peer. Select NodePort to expose the ports externally (recommended), and ClusterIP to not expose the ports. LoadBalancer and ExternalName are not supported in this release.</td>
<td>NodePort</td>
<td>Yes</td>
</tr>
<tr>
<td>State database</td>
<td>The state database that is used to store your channel ledger. The peer must use the same database as your blockchain network.</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB image repository</td>
<td>Applies only if CouchDB is selected as the ledger database. This field is autofilled to the installed path.</td>
<td>ibmcom/ibp-couchdb</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB Docker image tag</td>
<td>Applies only if CouchDB is selected as the ledger database. This field is autofilled to the version of the CouchDB image.</td>
<td>0.4.10</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peer Data persistence enabled</td>
<td>Enable the ability to persist data after cluster restarts or fails (for more information, see storage in Kubernetes). This field is autofilled to the version of the CouchDB image.</td>
<td>Checked</td>
<td>No</td>
</tr>
<tr>
<td>Peer use dynamic provisioning</td>
<td>Check to enable dynamic provisioning for storage volumes.</td>
<td>Checked</td>
<td>No</td>
</tr>
<tr>
<td>Peer persistent volume claim</td>
<td>For new claim only. Enter a name for your new PVC to be created.</td>
<td>my-data-pvc</td>
<td>No</td>
</tr>
<tr>
<td>Peer storage class name</td>
<td>Specify a storage class name for the peer.</td>
<td>Blank if you want to create a new PVC; otherwise, specify the storage class that is associated with the existing PVC.</td>
<td>No</td>
</tr>
<tr>
<td>Peer existing volume claim</td>
<td>Specify the name of an existing Volume Claim and leave all other fields blank.</td>
<td>New claim name</td>
<td>No</td>
</tr>
<tr>
<td>Peer selector label</td>
<td>Specify the Selector label for your PVC.</td>
<td>Default</td>
<td>No</td>
</tr>
<tr>
<td>Peer selector value</td>
<td>Specify the Selector value for your PVC.</td>
<td>Default</td>
<td>No</td>
</tr>
<tr>
<td>Peer storage access mode</td>
<td>Specify the storage access mode for the PVC.</td>
<td>ReadWriteMany</td>
<td>No</td>
</tr>
<tr>
<td>Peer volume claim size</td>
<td>Size of the Volume Claim. This value must be larger than 2 Gi.</td>
<td>8 Gi</td>
<td>Yes</td>
</tr>
<tr>
<td>State database persistent volume claim</td>
<td>For new claim only. Enter a name for your new PVC to be created.</td>
<td>statedb-pvc</td>
<td>No</td>
</tr>
<tr>
<td>State database storage class name</td>
<td>Specify a storage class name for state database.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>State database that is in volume claim</td>
<td>Specify the name of an existing Volume Claim and leave all other fields blank.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>State database selector label</td>
<td>Specify the Selector label for your PVC.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>State database selector value</td>
<td>Specify the Selector value for your PVC.</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>State database storage access mode</td>
<td>Specify the storage access mode for the PVC.</td>
<td>ReadWriteMany</td>
<td>No</td>
</tr>
<tr>
<td>State database volume claim size</td>
<td>Choose the size of disk to use.</td>
<td>8 Gi</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB - Data persistence enabled</td>
<td>For CouchDB container, ledger data is available when the container restarts. <strong>Note:</strong> If cleared, all data is lost if a failover or pod restart occurs.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td>CouchDB - Use dynamic provisioning</td>
<td>For CouchDB container use Kubernetes dynamic storage.</td>
<td>Selected</td>
<td>No</td>
</tr>
<tr>
<td>Peer CPU request</td>
<td>Minimum number of CPUs to allocate to the peer.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Peer CPU limit</td>
<td>Maximum number of CPUs to allocate to the peer.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Peer Memory request</td>
<td>Minimum amount of memory to allocate to the peer.</td>
<td>1 Gi</td>
<td>Yes</td>
</tr>
<tr>
<td>Peer Memory limit</td>
<td>Maximum amount of memory to allocate to the peer.</td>
<td>4 Gi</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB CPU request</td>
<td>Minimum number of CPUs to allocate to CouchDB.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CouchDB CPU limit</td>
<td>Maximum number of CPUs to allocate to CouchDB.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CouchDB Memory request</td>
<td>Minimum amount of memory to allocate to CouchDB.</td>
<td>1 Gi</td>
<td>Yes</td>
</tr>
<tr>
<td>CouchDB Memory limit</td>
<td>Maximum amount of memory to allocate to CouchDB.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Select and configure only one type of node at a time: CA, Orderer, or Peer.
If you do not use x86 architecture for all nodes, dynamic provisioning cannot be used. Because we are installing on the IBM z/Architecture, we cannot use dynamic provisioning.
If you do not specify storage class names, the default cluster storage class is used. If you do not use dynamic provisioning, Persistent Volumes must be created and set up with labels that can be used to refine the Kubernetes PVC bind process.

7. Using the values that you gathered in step 6 on page 11, select the Configuration option on the upper menu bar and complete the values that are required for your installation. All values that are marked as “required” need an entry or the installation fails.

8. Complete the following steps to install the certificate authority (for more information, see this IBM Developer web page):
   a. Attach the FlashSystem 9100 to the master node and install NFS server. Set up the other nodes in the cluster to use the master node as NFS server to which to mount.
      The following nodes are used in the installation:
      
      ```
      ordererca_user=ord-ca-admin
      namespace=ibp-on-icp
      ```
   b. Create persistent volumes that use the following names:
      
      ```
      blockchain-pv01
      blockchain-pv02
      blockchain-pv03
      blockchain-pv04
      blockchain-pv05
      blockchain-pv06
      blockchain-pv07
      ```
      Your volumes should follow your naming conventions.
   c. Run the following shell commands:
      
      ```
      root # ===>
      export ibp4icp_install_dir=$HOME/fabric-ca-client
      root # ===>
      echo "export ibp4icp_install_dir=$HOME/fabric-ca-client" >> ~/.bashrc
      root # ===>
      echo $ibp4icp_install_dir
      /root/fabric-ca-client
      root # ===>
      export ordererca_user=ord-ca-admin
      root # ===>
      export ordererca_password=secure_password
      ```
      (secure_password should be replaced by something actually secure)
   d. Log in to the cloud control console:
      
      ```
      root # ===> cloudctl login -a
      Username> username@xx.ibm.com
      Password> ********
      Authenticating...
      OK
      Targeted account ibp-icp-blueprint Account (id-ibp-icp-blueprint-account)
      ```
   e. Select a namespace:
      i. cert-manager
      ii. default
      iii. ibmcom
      iv. ibp-on-icp
      v. istio-system
      vi. kube-public
vii. kube-system

viii. platform

ix. services

Enter a number: 4

Targeted namespace ibp-on-icp

Configuring kubectl ...

Property "clusters.ibp-icp-blueprint" unset.

Property "users.ibp-icp-blueprint-user" unset.

Property "contexts.ibp-icp-blueprint-context" unset.

Cluster "ibp-icp-blueprint" set.

User "ibp-icp-blueprint-user" set.

Context "ibp-icp-blueprint-context" created.

Switched to context "ibp-icp-blueprint-context".

OK

f. Configure helm:

/root/.helm

OK

g. Log in to kubectl:

root # ===> kubectl config view --minify | grep namespace

namespace: ibp-on-icp

h. Create a secret for CA:

root # ===> kubectl create secret generic ibp4icp-orderer-ca

--from-literal=ca-admin-name=$ordererca_user

--from-literal=ca-admin-password=$ordererca_password

secret/ibp4icp-orderer-ca created

i. Get the proxy for the CA node:

root # ===> kubectl get nodes -l "proxy=true" -o

jsonpath="{.items[0].status.addresses[0].address}"

9.60.87.24

j. Issue the shell commands to configure logicals for:

root # ===> export release=ibp4icp-orderer-ca

root # ===> helm get values $release --tls

k. Enter the configuration values for ca in the GUI window:

name: OrdererCA

enabled: true

proxyIP: 9.60.87.24

app:

arch: s390x

tlsca:

name: orderer-tlsca

cname: orderer-tlsca-common

dataPVC:

existingClaimName: blockchain-pv01

da:

caAdminSecret: ibp4icp-orderer-ca

license: accept

peer:

enabled: false

orderer:
enabled: false
global:
  multiarch: false

I. Issue the following root commands:

root # ===>
export NODE_IP=9.60.87.24
root # ===>
export ord_caname=OrdererCA
root # ===>
export ord_tlscaname=orderer-tlsca
root # ===>
helm status $release --tls

Re-creating as Table
The following values were used for the Blueprint CA:

• Service account name: Default
• CA Name: OrdererCA
• CA Storage class name: Local-storage
  Storage classes are used mostly for dynamic provisioning. Dynamic provisioning is supported under GlusterFS on AMD64/i86_84 only; therefore, it is not a consideration for use on s390x. Although the use of dynapro with NFS works, it is not suggested because it is not supported.
• CA Existing volume claim: ibp4icp-orderer-ca-pvc
  (mapped to persistent volume blockchain-pv01)
• CA worker node architecture: s390x
• CA Selector label: {leave blank}
• CA selector value: {leave blank}
• CA Secret: Required field. Enter the name of the Kubernetes secret object that you created for your `ca-admin-name` and `ca-admin-password` ibp4icp-orderer-ca.
• CA TLS Instance Name: orderer-tlsca
• CSR Common Name: orderer-tlsca-common
• ProxyIP: 9.60.87.24
The parameters are entered in the configuration windows that are shown in Figure 9, Figure 10, Figure 11 on page 22, and Figure 12 on page 22.

Figure 9  First Configuration window

Figure 10  Second Configuration window
9. After all of the configuration parameters are entered, click **Install**.
You can create your peer after the CA is configured.

Creating a peer

Complete the following steps to create a peer:

1. Log in to the cloudctl CLI:

```
[root@ibpzms03 ~]# cloudctl login -a https://ibp-icp-blueprint.wsclab.endicott.ibm.com:8443 --skip-ssl-validation
Username> mrault@us.ibm.com
Password> xxxxxxxxxxx

Authenticating...
OK
Targeted account ibp-icp-blueprint Account (id-ibp-icp-blueprint-account)
```

2. Select the same namespace that is used for the CA (ibp-on-icp):
   - cert-manager
   - default
   - ibm-blockchain-platform
   - ibmcom
   - ibp-on-icp
   - istio-system
   - kube-public
- kube-system
- platform
- services
  Enter a number> 5
  Targeted namespace ibp-on-icp
  Configuring kubectl ...
  Property "clusters.ibp-icp-blueprint" unset.
  Property "users.ibp-icp-blueprint-user" unset.
  Property "contexts.ibp-icp-blueprint-context" unset.
  Cluster "ibp-icp-blueprint" set.
  User "ibp-icp-blueprint-user" set.
  Context "ibp-icp-blueprint-context" created.
  Switched to context "ibp-icp-blueprint-context".
  OK
  Configuring helm: /root/.helm
  OK

3. Verify your name:

   [root@ibpzms03 ~]# helm ls -m 10 -dr --tls
   next: metering

   NAME       REVISION UPDATED               STATUS  CHART
   NAMESPACE
   ibp4icp-orderer-ca 1   Thu Feb 21 12:24:56 2019DEPLOYEDibm-blockchain-platform-prod-1.0.1 ibp-on-icp

4. Move your certification into the tls.pem file:

   root@ibpzms03 ~]# kubectl exec $POD_NAME – cat
   /etc/hyperledger/fabric-ca-server/ca-cert.pem > tls.pem && cat

5. Create the needed directories and perform needed exports:

   [root@ibpzms03 ~]# mkdir fabric-ca-client/catls/
   [root@ibpzms03 ~]# mkdir fabric-ca-client/ca_admin/
   [root@ibpzms03 ~]# export $HOME/fabric-ca-client/ca_admin/
   [root@ibpzms03 ~]# export
   FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca-client/ca_admin/

6. Use kubectl to get your service details; you need the port (in this example, 30722):

   [root@ibpzms03 ~]# kubectl get service
   NAME                     TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)
   AGE
   ibp4icp-orderer-ca-ca   NodePort   10.0.0.162   <none>        7054:30722/TCP
   11d

7. Create your base64 cert (it must be in base64 or it does not work):

   [root@ibpzms03 ~]# export POD_NAME=$(kubectl get pods --namespace ibp-on-icp -l "app=ibm-ibp, release=ibp4icp-orderer-ca" -o
   jsonpath="{.items[0].metadata.name}")
   [root@ibpzms03 catls]# kubectl exec $POD_NAME -- cat
   /etc/hyperledger/fabric-ca-server/ca-cert.pem > tls.pem && cat tls.pem | base64
   $FLAG
   LS0tLS1CRUdJTiBDRUVJQSUzQ0FQSU0tLS0tCK1JSUNGakNDQWlyzF03S3UJBZO1VU1JtMjRlZjRo
   d00nRINkRQaWpGSRHQV8bOnnU1L1pJemowRUF3SxKURFTEmI8a0dBMVFQmhNQ1ZWTXhG
   ekFwQmdOVkJBZ1REazV2Y25sbOFTmhzjTlzYVc1aE1SUxdFZI1ETVlFRSwFpFd3RJZVhCGrNteGa
   R2RsY2pUHE1BMEdBMVFQ3hNR1JtRm1jbWxqTVrdoZ3MURUWVFhChV1XSn1hV0I0C1kyRXRj
8. If not yet not done so, download and install the needed fabric binaries:

```bash
[root@ibpzms03 ~]# mkdir $HOME/fabric-ca-client

[root@ibpzms03 ~]# cd $HOME/fabric-ca-client

[root@ibpzms03 fabric-ca-client]# mkdir fabric-binaries
[root@ibpzms03 fabric-ca-client]# cd fabric-binaries

[root@ibpzms03 fabric-binaries]# curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.1
```

9. Add the path to the binaries to your $PATH logical so that the binaries are searchable:

```bash
export PATH=$PATH:$HOME/fabric-ca-client/fabric-binaries/bin
```

10. Enroll the certificate authority. You need the following information:

   - **Common Name:** The name of your CA secret user: `ord-ca-admin` (Step 9 c of installing the CA, page 18).
   - **Password:** The password for that user: `secure_password` (Step 9 c of installing the CA, page 18).
   - **IP Address:** The IP address of the CA node: `9.60.87.24` (Step 9 i of installing the CA, page 19).
   - **Port:** The port of the CA: `30722` (Step 6 page 24).
   - **TSL Certificate:** The full directory of the tls.pem file (step 4 page 24).
   - **Common Name:** The name of the CA: `OrdererCA` (Step 9 j of installing the CA, page 19):

```bash
[root@ibpzms03 ~]# cd $HOME/fabric-ca-client

[root@ibpzms03 fabric-ca-client]# mkdir peer-admin
[root@ibpzms03 fabric-ca-client]# mkdir tls-ibp
[root@ibpzms03 fabric-ca-client]# export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca-client/peer-admin
[root@ibpzms03 fabric-ca-client]# cd fabric-binaries

[root@ibpzms03 fabric-binaries]# fabric-ca-client enroll -u https://ord-ca-admin:secure_password@9.60.87.24:30722 --caname OrdererCA -tls.certfiles $ibp4icp_install_dir/catls/tls.pem
```

```
2019/03/05 11:27:42 [INFO] TLS Enabled
2019/03/05 11:27:42 [INFO] encoded CSR
```

```
```
At this point, the certificate authority is installed and running, you can now deploy a peer.

**Deploying the peer**

Complete the following steps to deploy the peer:

1. Complete the CA portion of your JSON document by using the following required entries:
   - **CAname** (Step 9 j of installing the CA, page 19).
   - **CAPort** (Step 6 of Installing the CA, page 24).
   - **CAHost** (Step 9 i of installing the CA, page 19).
   - **CACert** (see next section).

2. Complete the following steps to generate a CACert:
   a. Go to your Starter or Enterprise IBC Console, select **Overview → Connection Profile**, as shown in Figure 14.

   ![Figure 14 Connections Profile](image)

   b. Click **Raw JSON** from the JSON connection profile that is displayed. Select the appropriate cert and insert it into the following command:

   ```bash
   echo -e 'paste in Certificate Authority (CA) TLS Certificate' | base64 $FLAG
   ```

   For example:

   ```bash
   [root@ibpzms03 dev]# echo -e ?-----BEGIN CERTIFICATE-----
   MIIFajCCBFKgAwIBAgISA4zRdubZCc/b8B7dxjGDFE0JMAOGCSqGSIbDQ
   EBCwUA
   mEoCzASjgNVBAITYA1TVMRyFAYDVQK
   EwIMZXQncyBFbmYxEBOMSMwIYDvQQD1nEx
   pMZXQncyBFbmYxEBOIEF1dGhcvm10eSBYMzAeFw0xO0EYMDYiMTI0NDFA
   MCA0CiXjAgBgNVAMMGsouMjWmNySiBGja2NoYWliuMlibsS5jInb2O0wgeEIAOOGCSqGSIbDQ
   EBAQUA41iDwAgEIKoAIBQDjKzoRlmg23pK1fJxjnlJe+J0TgqtOM3ZrU6gcbBtSR3sXbLT
   N5v4mAOHPr/TR2AgGEW/BEt57f9tgUR86v4n4Uu8s2AMx8XEpo/HsYGfexIF4gu6T8j3://KttUEY
   8+h5ZJ1u12U71J3T1/GEOM5/nxebzG6PSLQE+f6BO08r6gOsQ+Tx1buTKY/wUPnxvEh9n9/M+i
   7b1cXweKNTcsc/zwgb7sVBVTcnYyUuXvXcEPgxryXwccyC9JGg9P9/n+1fjxUAF2eNfZ38k1
   OKi1nwhFt0rpcbEmhxcxSxyMSVhmU0KpWsVInZ1UC9PWJBsUHBNjRuIxnkaJcaTYxINcYnAg
   MBAAQ96jggJuIIICajA0B6gNHQ8AFBEBAMCBaAwHQYDVQRI0B8YWFAIKwYB8QUHAwEgCCsGAQF
   UbMCMAwGAU1dEWE/wQCMAAwHQYDVOR05BBYEFc65dBQy8k0cQL5fBie3KH1RsuqBNMMDGA1Ud1w
   QYMAAfkKnaMeFdd25t6ES5Z2zFZe+q0YhNGGSogCScsGAQUFbEB8GWthYTAUggrBgEBFQcwAYYi
   anHROcDovL29j3cUua5W50LxgLmw1CndHHnbnMx80Lm9yZzAvBgrBgEBFQcwA0jYaJHR0CDoV
   LNcnQuaW50LxgLmw1CndHHnbnMx80Lm9yZzAvBgrBgEBFQcwA0jYaJHR0CDoV
   ```

   Note: Replace the `BEGIN CERTIFICATE` and `END CERTIFICATE` with your actual certificate content.
ajJ0Y015bzVqTWQ2am1lV1VISzhby9qb1dVb0hPVWd3dWYFbMVfZei5ZH6a0RXtA0Zmts
eE33WFd2zWw51h6U1w1TI1BDdM5eGluZG1qa1c4Bed5KIFZw5mHb11mWHtH11o2adtdamVt
MFkraVd51lJyVbEyvKWHm3Jm9q5cF0dYK5eVRFw4M4tSTyKE3VUSkkRTST21zJZu5o9x3Ft
NTdUUSDMZVeskUrZ5o5i19ETKZ1MFnPtIU50tL51FTKqOQsvE15GSUNBvEu5L50tLW4tL50t
LUJfF01015ENV1JR1KQVFR5L05tL50tLUJfF01015ENV1JR1KQVFR5L05tL50tLUJfF01015ENV1JR1KQVFR5L05tL50tLUJfF01015ENV1JR1KQVFR5L05tL50tLUJfF01015ENV1JR1KQVFR5L05tL50tLUJfF01015ENV1JR1KQVFR5L05tL50t

The bolded part is the new Base64 certificate to insert into the JSON connection profile.

The JSON connection profile should now look like the following example:

```json
{
  "enrollment": {
    "component": {
      "cahost": "9.60.87.24",
      "caport": "30722",
      "caname": "OrdererCA",
      "catls": {
        "cacert": "Py0tLS0tQkVHSU4gQ0VSVElGSUNBVEUtLS0tLT8KQWdJU0ErRkFZOGtxY241ZkIxZTNLSEhsUnN1a0JXTUI4R0ExVWRJd1FZTUJhQUZLaEthbU1FZmQyNjV0RTV0NlpGWmUvenFPeWhNRzhHbkNDc0dBUVVGQndFQkJHTXdZVEF1QmdnckJnRUZCUWN3QVlZaWFIUjBjRG92TDI5amMzQXVhVzUwTFhnekxteGxuZEhObGJtTnllWEIwTG05eVp6QXZCZ2dyQmdFRkJRy3dBb1lqYUhSMGNEb3ZMMk5sY25RdWFXNTBMWGd6TG14bG5kSE5sYm1OeWVYQjBMbTl5Wnk4d0pBWURWU35d3zSA1fZrwPnmzJowyL11LN3aQ05UUVFlu5L05tLS1STKq0QOSVYESUNBvEU5L05tLT8K
    }
  }
}
```

The new certificate is inserted into the `catls` object of the `enrollment` component.
3. Register your peer by completing the following steps:

a. Go to the Cloud console for your starter or enterprise account to the dashboard.

b. From the left-side menu, select **Certificate Authority**. On the CA, click **Add Use**, as shown in Figure 15 on page 32.
c. Complete the values for your new peer (in this case, mikespeer, mikespeerpw, and the type should be set to peer, as shown in Figure 16). Ensure that all values are correct because after you submit the user, you cannot delete or change it. Click **Submit** to register your peer.
4. The same values that are used for the peer user should be added to the JSON connection profile. The file section for the peer now looks like the following example:

```
"enrollid": "mikespeer",
"enrollsecret": "mikespeerpw",
"admincerts": [""
```

5. Create a place to store information and certs for the peer and reset your $HOME logical, as shown in the following example:

```
[root@ibpzms03 ~]# cd $HOME/fabric-ca-client
[root@ibpzms03 fabric-ca-client]# mkdir peer-admin
[root@ibpzms03 fabric-ca-client]# mkdir tls-ibp
[root@ibpzms03 fabric-ca-client]# export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca-client/peer-admin
```

6. Download your root cert for your Starter or Enterprise Plan and copy it into your tls-ibp directory, as shown in the following example:

```
[root@ibpzms03 ~]# cp us07.blockchain.ibm.com.cert $HOME/fabric-ca-client/tls-ibp/tls.pem
```

**Note:** Download the TLS certs from IBM Cloud depending on the service plan, location, and cluster that you use. You can find your cluster based on the domain name of your certificate authority URL as stored in the JSON file from the IBP console connection profile JSON file; for example: us01.blockchain.ibm.com:31011 or us02.blockchain.ibm.com:31011.

Then, the root cert is at loc.blockchain.ibm.com.cert.

Where loc is the location code; for example, us01 - us08. For our example, the value is us07.blockchain.ibm.com:31001.

However, the address is different based on whether yours is a starter or enterprise plan or your system is on an enterprise cluster or local node.

7. Generate certificates for our Peer Admin that we registered by using the following commands for fabric-ca-client enroll:

```
-u https://<admin peer name>:<admin peer secret>@<CA URL with Port>
--caname <CA Name in Connection Profile>
--tls.certfiles <path to tls-ibp/tls.pem>
```

```
[root@ibpzms03 fabric-ca-client]# cd fabric-binaries
[root@ibpzms03 fabric-binaries]# fabric-ca-client enroll
-u https://mikespeer:mikespeerpw@9.60.87.24:30722
--caname OrdererCA
--tls.certfiles $HOME/fabric-ca-client/tls-ibp/tls.pem
```

2019/03/05 21:30:38 [INFO] TLS Enabled
2019/03/05 21:30:38 [INFO] generating key: &{A:ecdsa S:256}
2019/03/05 21:30:38 [INFO] encoded CSR
8. Convert the peer-admin’s cert into a BASE64 cert so that we can continue to complete our configuration. The following basic command is used:

```
cat $HOME/fabric-ca-client/peer-admin/msp/signcerts/cert.pem | base64 $FLAG
```

After the command is run, take the resulting certificate and place it in the admincert portion of our configuration file:

```
"enrollid": "mikespeer",
"enrollsecret": "mikespeerpw",
"admincerts": ["LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5ekNDQW5LZ0F3SUJBZ01VQkJKXU3RQSDZN"
```

We continue to complete our configuration file with more information. All of this information falls under the tls section, as shown in the following example:

```
cahost": "9.60.87.24" # Your CA URL without its port
caport": "30727" # Your CA port
caname": "OrdererCA" # Your CA name from your CA deployment
```

9. Get our certificate by issuing the following command (with your variables):

```
[root@ibpzms03 fabric-ca-client]# cat $HOME/fabric-ca-client/catls/tls.pem | base64 $FLAG
```

Place the output into the tls section of the configuration file:

```
"tls": {
  "cahost": "9.60.87.24",
  "caport": "30727",
  "caname": "OrdererCA",
  "catls": {
    "cacert": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5ekNDQW5LZ0F3SUJBZ01VQkJKXU3RQSDZN"
```

10. Create a directory and export its logical:
```
[root@ibpzms03 ~]# cd $HOME/fabric-ca-client
[root@ibpzms03 fabric-ca-client]# mkdir tlsca-admin
[root@ibpzms03 fabric-ca-client]# export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca-client/tlsca-admin
```

11. Generate the certificates of our TLS CA Admin by using the following syntax:
```
fabric-ca-client enroll
-u https://<Username for CA secret>:<Password for CA secret>@<Your CA Deployment with URL>
--caname <Your Deployed CA Name>
--tls.certfiles >Path to catls/tls.pem file
```

```
[root@ibpzms03 fabric-ca-client]# fabric-ca-client enroll -u https://ord-ca-admin:secure_password@9.60.87.24:30722
--caname OrdererCA
```
12. Determine what your affiliation by using the following syntax:

```
fabric-ca-client affiliation list
--caname <CA caname>
--tls.certfiles <Path to /catls/tls.pem file>
```

```
[root@ibpzms03 fabric-ca-client]# fabric-ca-client affiliation list --caname OrdererCA
--tls.certfiles $HOME/fabric-ca-client/catls/tls.pem
affiliation: .
affiliation: org2
  affiliation: org2.department1
affiliation: org1
  affiliation: org1.department1
  affiliation: org1.department2
```

13. Register our peer by using the following syntax:

```
fabric-ca-client register --caname <Your CA Deployed CA name>
--id.affiliation <Your affiliation>
--id.name <Peer name>
--id.secret <Peer secret>
--id.type peer
--tls.certfiles <Path to /catls/tls.pem file>
```

For example:
```
[root@ibpzms03 fabric-ca-client]# fabric-ca-client register
--caname OrdererCA
--id.affiliation org1.department1
--id.name mikestlspeer
--id.secret mikestlspeerpw
--id.type peer
--tls.certfiles /root/fabric-ca-client/catls/tls.pem
```

14. For the CSR section of the configuration file, add your proxy node IP address and then what you are going to call your peer helm chart, as shown in the following example:

```
"csr": {
  "hosts": [ "9.60.87.24",
```
The configuration process for JSON is completed. You can now create your configuration file (secret.json).

Encode your secret.json file into base64 format to put it in IBM Cloud Private, as shown in the following example:

```
[root@ibpzms03 fabric-ca-client]# cat secret.json | base64 $FLAG
```

Optionally, you can encode your CouchDB information that is used later, as shown in the following example:

```
[root@ibpzms03 fabric-ca-client]# echo -n 'admin' | base64 $FLAG
```

15. Log on to IBM Cloud Private and create your Peer’s secret. Also, create the secret that is required to enable CouchDB as your state database. Select Configuration → Secrets. Then, click Create Secret, as shown in Figure 17.

![Figure 17: Creating Secrets](image)
The Create Secret pop-up window opens, as shown in Figure 18. Enter the secret name, select the correct namespace for your peer, and enter the type of secret as opaque. Click the Data menu item.

![Create Secret](image)

**Figure 18 Creating a secret name**

In the data area, we add three data items: one for the peer and two for the CouchDB. The peer value is the `secret.json` file and points to its contents. The CouchDB has a user and password that point to the CouchDB base64 value that was created in the optional section of step 13 on page 35. After you complete copying in the certifications, click Create to create the secret (see Figure 19).

![Create Secret](image)

**Figure 19 Adding Data item secrets**

After the peer and CouchDB secrets are established, you have all of the required data to complete the configuration of the peer and install it. Return to the Cloud Private Console, click Catalog, choose Blockchain.
In the helm configuration chart, complete the peer section with the values that are listed in Table 6.

**Table 6  Values for Peer section**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Peer</td>
<td>Select to install a peer.</td>
<td>Selected</td>
</tr>
<tr>
<td>Peer worker node architecture</td>
<td>Select your cloud platform architecture (AMD64 or S390x).</td>
<td>S390x</td>
</tr>
<tr>
<td>Peer image repository</td>
<td>Location of the Peer Helm Chart. This field is autofilled to the installed path.</td>
<td>ibmcom/ibp-fabric-peer</td>
</tr>
<tr>
<td>Peer Docker image tag</td>
<td>Autofilled to the version of the Peer image.</td>
<td>1.2.1</td>
</tr>
<tr>
<td>Peer configuration</td>
<td>You can customize the configuration of the peer. This information overwrite the content in the peer configuration file; that is, core.yaml.</td>
<td>None</td>
</tr>
<tr>
<td>Peer configuration secret (Required)</td>
<td>Name of the Peer configuration secret you created in IBM Cloud Private.</td>
<td>mikes_peer_secret</td>
</tr>
<tr>
<td>Organization MSP (Required)</td>
<td>This value can be found in Network Monitor (IBP UI) by clicking Remote Peer Configuration in the Overview window. If you are not connecting to an IBP network, you can create an Organization MSP value, such as &quot;org1&quot; or specify an Organization MSP of which the peer will be a part.</td>
<td>Org1</td>
</tr>
<tr>
<td>Peer service type</td>
<td>Used to specify whether external ports should be exposed on the peer. Select NodePort to expose the ports externally (recommended), and ClusterIP to not expose the ports. LoadBalancer and ExternalName are not supported in this release.</td>
<td>NodePort</td>
</tr>
<tr>
<td>State database</td>
<td>The state database that is used to store your channel ledger. The peer must use the same database as your blockchain network.</td>
<td>couchdb</td>
</tr>
<tr>
<td>CouchDB image repository</td>
<td>Applies only if CouchDB is selected as the ledger database. This field is autofilled to the installed path.</td>
<td>ibmcom/ibp-couchdb</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>CouchDB Docker image tag</td>
<td>Applies only if CouchDB is selected as the ledger database. Autofilfilled to the version of the CouchDB image.</td>
<td>0.4.10</td>
</tr>
<tr>
<td>Peer Data persistence enabled</td>
<td>Enable the ability to persist data after cluster restarts or fails. For more information, see storage in Kubernetes. Note: If not selected, all data is lost if a failover or pod restart occurs.</td>
<td>Selected</td>
</tr>
<tr>
<td>Peer use dynamic provisioning</td>
<td>Select to enable dynamic provisioning for storage volumes.</td>
<td>Not selected</td>
</tr>
<tr>
<td>Peer persistent volume claim</td>
<td>For new claim only. Enter a name for your new Persistent Volume Claim (PVC) to be created.</td>
<td>Blockchain_PV06</td>
</tr>
<tr>
<td>Peer storage class name</td>
<td>Specify a storage class name for the peer.</td>
<td>Blank if you want to create a new PVC; otherwise, specify the storage class that is associated with the PVC.</td>
</tr>
<tr>
<td>Peer existing volume claim</td>
<td>Specify the name of a Volume Claim and leave all other fields blank.</td>
<td>New claim name</td>
</tr>
<tr>
<td>Peer selector label</td>
<td>Specify the Selector label for your PVC.</td>
<td>Default</td>
</tr>
<tr>
<td>Peer selector value</td>
<td>Specify the Selector value for your PVC.</td>
<td>Default</td>
</tr>
<tr>
<td>Peer storage access mode</td>
<td>Specify the storage access mode for the PVC.</td>
<td>ReadWriteMany</td>
</tr>
<tr>
<td>Peer volume claim size</td>
<td>Size of the Volume Claim. This value must be larger than 2 Gi.</td>
<td>8 Gi</td>
</tr>
<tr>
<td>State database persistent volume claim</td>
<td>For new claim only. Enter a name for your new PVC to be created.</td>
<td>Blockchain_PV07</td>
</tr>
<tr>
<td>State database storage class name</td>
<td>Specify a storage class name for state database.</td>
<td>None</td>
</tr>
<tr>
<td>State database that is in volume claim</td>
<td>Specify the name of an existing Volume Claim and leave all other fields blank.</td>
<td>None</td>
</tr>
<tr>
<td>State database selector label</td>
<td>Specify the Selector label for your PVC.</td>
<td>None</td>
</tr>
<tr>
<td>State database selector value</td>
<td>Specify the Selector value for your PVC.</td>
<td>None</td>
</tr>
</tbody>
</table>
16. Confirm that your Peer is working by reviewing the logs of the init container, by using the following syntax:

```
kubectl logs <Your Peer's Pod> -c init | grep EXIT
```

Example:

```
[root@ibpzms03 fabric-ca-client]# kubectl logs mikespeer-74b89b485f-bmfs9 -c init | grep EXIT
EXIT WITH RC=0 #
```

An RC=0 is a normal entry that indicates that the peer is working normally.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>State database storage access mode</td>
<td>Specify the storage access mode for the PVC.</td>
<td>ReadWriteMany</td>
</tr>
<tr>
<td>State database volume claim size</td>
<td>Choose the size of disk to use.</td>
<td>8 Gi</td>
</tr>
<tr>
<td>CouchDB - Data persistence enabled</td>
<td>For CouchDB container, ledger data will be available when the container restarts. If cleared, all data is lost if a failover or pod restart occurs.</td>
<td>Selected</td>
</tr>
<tr>
<td>CouchDB - Use dynamic provisioning</td>
<td>For CouchDB container use Kubernetes dynamic storage.</td>
<td>Not selected</td>
</tr>
<tr>
<td>Peer CPU request</td>
<td>Minimum number of CPUs to allocate to the peer.</td>
<td></td>
</tr>
<tr>
<td>Peer CPU limit</td>
<td>Maximum number of CPUs to allocate to the peer.</td>
<td></td>
</tr>
<tr>
<td>Peer Memory request</td>
<td>Minimum amount of memory to allocate to the peer.</td>
<td>1 Gi</td>
</tr>
<tr>
<td>Peer Memory limit</td>
<td>Maximum amount of memory to allocate to the peer.</td>
<td>4 Gi</td>
</tr>
<tr>
<td>CouchDB CPU request</td>
<td>Minimum number of CPUs to allocate to CouchDB.</td>
<td></td>
</tr>
<tr>
<td>CouchDB CPU limit</td>
<td>Maximum number of CPUs to allocate to CouchDB.</td>
<td></td>
</tr>
<tr>
<td>CouchDB Memory request</td>
<td>Minimum amount of memory to allocate to CouchDB.</td>
<td>1 Gi</td>
</tr>
<tr>
<td>CouchDB Memory limit</td>
<td>Maximum amount of memory to allocate to CouchDB.</td>
<td>4 Gi</td>
</tr>
</tbody>
</table>
IBM Cloud Object Storage installation

With over 600 technology patents, IBM Cloud Object Storage is a software-defined storage platform that stores massive amounts of data with efficiency, reliability, simplicity, and cloud native accessibility to transform the enterprise for multiple use cases. IBM Cloud Object Storage breaks down barriers for storing massive amounts of data by using an Information Dispersal Algorithm (IDA) and flexible configurations to spread data across multiple nodes by using IBM’s patented technologies. Our proven solutions can turn storage challenges into business advantages.

The on-premises IBM Cloud Object Storage System is a breakthrough platform for storing large amounts of unstructured data. It provides scalability, availability, security with simplicity, and lower total cost of ownership (TCO). It is available as an integrated storage system or as a software-only solution. In addition, IBM Cloud Object Storage is available as a public cloud service in the IBM Cloud. IBM Cloud Object Storage is ideal for use cases, such as remote file collaboration, backup or archive repository, and as a content repository for images, video, and voice.

IBM Cloud Object Storage can integrate with analytics workloads and now offers a new metadata management and insight software with IBM Spectrum Discover. This feature makes it an ideal candidate for blockchain applications. One of the advantages for customers with the IBM Cloud Object Storage architecture is that as more use cases are put on the system, more benefits can be realized.

Clients can start with as few as three commodity x86 server nodes or as little as 72 TB and grow to exabytes of usable storage without ever losing access to the data. By combining a single copy of protected data and the ability to lock down data by using policy-based WORM storage, IBM Cloud Object Storage is quickly becoming the choice for many industries, such as finance, healthcare, and government, that have compliance or other data retention requirements.

For more information about IBM Cloud Object Storage see the following website:
https://www.ibm.com/cloud/object-storage

You must obtain from your Cloud Object Storage administrator what is known as a bucket (a place to store objects). By using the bucket identifier, access, secret keys, user, and password data, we connect the BDS instance that we create to the Cloud Object Storage bucket for off-chain storage. The Bucket data resembles the information that is listed in Table 7.

<table>
<thead>
<tr>
<th>Vault (Bucket) properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vault (Bucket) Name</td>
<td>Vault name</td>
</tr>
<tr>
<td>Endpoint</td>
<td>Endpoint URL</td>
</tr>
<tr>
<td>Vault (Bucket) Description</td>
<td>For IBM Blockchain</td>
</tr>
<tr>
<td>Vault (Bucket) User</td>
<td>email/id</td>
</tr>
<tr>
<td>Name Index Enabled</td>
<td>True</td>
</tr>
<tr>
<td>Recovery Listing Enabled</td>
<td>False</td>
</tr>
<tr>
<td>SecureSlice Enabled</td>
<td>True</td>
</tr>
<tr>
<td>Versioning</td>
<td>False</td>
</tr>
<tr>
<td>Compliance Enabled</td>
<td>False</td>
</tr>
</tbody>
</table>
The following prerequisites must be met:

- Have an IBM ID and IBM Cloud account.
- Have a starter or enterprise level blockchain account with blockchain installed, an organization, and channel created.
- Be on the Blockchain Document Store (BDS) whitelist.

After you complete the prerequisites, you can continue the installation process.

**Note:** To install this platform on a remote peer, you need IBM support to add the remote peer connection certification to the IBP in the cloud's connection JSON.

To install BDS Utilize, see the following website (log in required):


**Note:** The Blockchain Document Store is a whitelist product, which means that you cannot access it unless you are on the whitelist for BDS. After you are on the Whitelist the link (https://console.bluemix.net/catalog/services/blockchain-document-store), you can access the BDS.

<table>
<thead>
<tr>
<th>Expiration Date</th>
<th>Expiration Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secret Key Authentication</strong></td>
<td><strong>Secret Key Authentication</strong></td>
</tr>
<tr>
<td>Access Key ID</td>
<td>Access Key ID</td>
</tr>
<tr>
<td>Secret Access Key</td>
<td>Secret Access Key</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Virtual Host Access</strong></th>
<th><strong>Virtual Host Access</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be an accessible URL</td>
<td>Can be an accessible URL</td>
</tr>
</tbody>
</table>
Complete the following steps:

1. Browse to the following ULR to see the window that is shown in Figure 20:
   
   [https://console.bluemix.net/catalog/services/blockchain-document-store](https://console.bluemix.net/catalog/services/blockchain-document-store)

2. From this window, select the link to the catalog. You are taken to the Blockchain Document Store installation package. If you are not taken to the package, you are not on the whitelist. You must be on the whitelist to proceed. If the connection is unsuccessful, you see the standard catalog. If the connection is successful, you see the menu that is shown in Figure 21.
3. Click Create to create an instance of BDS in your blockchain.

4. After the instance is created, you must configure it. Open your blockchain services dashboard. The window that is shown in Figure 22 is displayed.

![Blockchain service dashboard](image1.png)

**Figure 22** Blockchain service dashboard

5. From the listed services, select the BDS service and then, click Create Instance.

After the instance is created, you must configure it so it can connect to your peer node network, which requires a JSON network credential.

6. Select the Blockchain service and then, select Monitor from the window. Select the APIs tab, as shown in Figure 23.

![APIs](image2.png)

**Figure 23** APIs
7. Return to the overview window and select C to see the required JSON script (see Figure 24). Select Raw JSON and copy the JSON script, or download it to a file so that you can upload it later.

![Figure 24 Obtaining the JSON script](image1.png)

8. Return to the services and select the BDS service. Select Manage and the configuration window that is shown in Figure 25 opens.

![Figure 25 Configure the services](image2.png)
9. Using the information that was gathered in the previous steps, complete the necessary values.

10. Complete the following steps to authorize the service:
   a. You need the administrator certificate from the IBM Blockchain Solutions Manager. Browse to the following URL to get to the service:
      
      \[
      \text{https://pbsa-prod.us-south.containers.mybluemix.net/1e627852-f65e-48e6-98b0-002de67ff533/onboarding/v1/logins}
      \]
      
      This URL is the base URL for the BDS service plus the path to the onboarding service, as shown in Figure 26.

   b. From this window, select **Continue as Solution Admin of default**, as shown in Figure 27.

   ![Figure 26 Top-level IBM Blockchain Solution Manager window](image1)

   ![Figure 27 User Details](image2)
c. Select **Copy**. Return to the Authorize window and paste the certificate into the field that is circled in Figure 28.

![Figure 28  Token Details](image)

**Figure 28  Token Details**

d. Click **Submit** to submit the new certificate (see Figure 29).

![Figure 29  Adding a new certificate](image)
e. Restart the peers so that they recognize the new certificate by clicking **Restart** (see Figure 30).

![Restart peers window](image)

**Figure 30** Restart peers window

11. Specify the network channels to which the service is connected by clicking the **Members** menu on the IBP console. Verify that the certificate was added and that the Members are correct, as shown in Figure 31.

![Members menu](image)

**Figure 31** Members menu
12. From the Blockchain console, use the right-side menu in the channel section to open the channel area of the Blockchain console (see Figure 32).

![Figure 32 Blockchain console, channel area](image)

13. Select the correct channel and use the right-side menu link to select **Sync Certificate** (see Figure 33).

![Figure 33 Defining the Channels](image)
The certificate sync occurs automatically, as shown in Figure 34.

**Figure 34  Sync the certificate**

14. Instantiate the chain code automatically, as shown by Figure 35. This process can take several seconds to complete.

**Figure 35  Instantiate the chain code**
15. After the code is instantiated, the system displays the needed API links to use when an application is attached to the API (see Figure 36).

![Figure 36 API links](image)

16. To use the BDS, an application must be interfaced. IBM provides the Swagger API to provide a basic interface to the BDS. To instantiate the Swagger API, you must obtain a service ID. A service ID is created by clicking the Cloud Control Console **Identity & Access** option in the Service ID submenu. Selecting that option and then **Create** opens to window that is shown in Figure 37.

![Figure 37 Entering values or creating a service ID](image)
17. Enter the correct values and select **Create**. Successful service ID creation results are displayed in the window that is shown **Figure 38**.

![Figure 38 Creating the service ID](image)

The Service ID option now shows the created ID. Clicking the ID returns to the display that is shown in **Figure 39**.

![Figure 39 Service ID display](image)

From the display of the service ID, you need the service ID, as shown in the following example:

```
ServiceId-58ce2fd0-e2f8-433b-947d-5c06b2005ca3
```

18. Assign the service ID to an access group after an appropriate group is created if such a group does not exist. To create a group, browse to the **Access Groups** menu in the **Identity & Access** menu and click **Create** (see **Figure 40**).

![Figure 40 Creating an access group](image)
19. Enter the necessary information for your group and select **Create**.

20. Add the service ID to the group you created (see Figure 41).

![Figure 41: Adding service IDS to Documents Groups](image1)

21. Create the platform API key by selecting **Platform API Key** and clicking **Create**.

22. Select the created API Key to copy its value. In our example, the following key was created (see Figure 42):

   MkSR3Q8jaHDOjJg7hT9l7ta75w9eqHrmQ9GTA0-KSHvl

![Figure 42: API key value](image2)

23. By using the keys that were generated, you can quickly authorize other applications to access your BDS by using the on and off boarding JSON that is provided in step 16 page 51.

   Use the following URL to use you are when accessing the BDS:
   
   https://console.bluemix.net/dashboard/apps

24. Log in by using your ID. The BDS instance is displayed. Click the instance to access the needed swagger URL.

25. To install this on a remote peer, you need Support to add the remote peer connection certification to the IBP in the cloud’s connection JSON.

   The BDS is installed in each peer that requires access to the BDS.
Changing from Cloud COS Storage to Local, Onsite COS Storage for off-chain data

Many clients might not want to use IBM or other cloud storage. Instead, they want to use their own onsite storage. Blockchain Document Store uses IBM Cloud COS as a default. In this section, we review this feature. To begin, go to your BDS instance’s solution manager website to get an organization admin token for the organization that had BDS installed (login required):

https://pbsa-prod.us-south.containers.mybluemix.net/703c9b46-3443-4a77-8cbf-7d5b555dbf37/onboarding/v1/logins?responseMode=undefined

After you enter in the URL for your instance, you see the main login window for the solution manager, as shown in Figure 43.

![Figure 43](image1.jpg)

Figure 43 Example login window for IBM Blockchain Solution Manager

From the main login window, complete the following steps:

1. Select **Sign in with IBMID**. The standard IBM login page opens in which you enter your IBM login ID and your password.

   After you are authorized, the IBM Blockchain Solution Manager menu is displayed, as shown in Figure 44.

![Figure 44](image2.jpg)

Figure 44 IBM Blockchain Solution Manager menu
2. For BDS use, select **Continue as an Organization Admin of Organization1**. The user token that is needed for the Swagger application that controls the low-level access to the BDS APIs is displayed, as shown in Figure 45.

![IBM Blockchain Solution Manager](image)

**Figure 45  User Token window**

3. Select **Copy** from the User Token menu to copy the JWT token. Start your BDS instance Swagger application by using the provided URL. The following URL was used in our example:

https://pbsa-prod.us-south.containers.mybluemix.net/703c9b46-3443-4a77-8cbf-7d5b555dbf37/docstore/swagger-ui.html#/
Successful startup of the Swagger application is shown in Figure 46.

Figure 46  Swagger application for BDS APIs

4. Copy the JWT token into the space after the word “Bearer”. Do not press Enter, or you must copy the token again.
5. To test if we are attached to the Cloud COS, we upload a file. Select the **Upload** option. Then, select the option for Uploading a single document. A window opens, as shown in the example, in Figure 47.

![Figure 47 Swagger BDS Upload display](image)

6. To upload a file, you need a unique document identifier (because this document is the first document, any identifier is unique), the document type, the channel ID, and knowledge where the document is stored. The entries should not have quotes around them.
Select **Try it out** after entering the correct information. If you are successful, you receive a 200 series response, as shown in Figure 48.

```bash
Curl
curl -X POST --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' --header 'Authorization: Bearer eyJ...

Request URL
https://poba-prod.us-south.containers.mybluemix.net:443/78x96b4-3443-4a7f-8c8f-78d0555d8f37/docstore/v1/docstore/defaultchannel/

Request Headers
{
   "Accept": "application/json"
}

Response Body
{
   "status": 200,
   "response": {
      "correlationId": "401c78b0-501a-4f05-b408-54ec3f089e5"
   }
}

Response Code
200

Response Headers
{
   "pragma": "no-cache",
   "date": "Tue, 19 May 2019 21:38:41 GMT",
   "content-type": "application/octet-stream",
   "content-length": "383",
   "connection": "keep-alive",
   "content-type": "application/json; charset=UTF-8",
   "cache-control": "no-cache, no-store, must-revalidate",
   "transfer-encoding": "chunked",
   "strict-transport-security": "max-age=15300000 ; includeSubdomains",
   "x-frame-options": "denied",
   "x-content-type-options": "nosniff",
   "expires": "0"
}

Figure 48   Successful file upload
The easiest way to verify the document is in the Cloud COS is to use the **Download Document** function, shown in Figure 49.

**Figure 49  Download single document**
7. Enter the document ID that you assigned and the channel name. Then, select **Try it out**.

If the document is successfully uploaded, you receive a successful download message, as shown in Figure 50.

![Response Body](image)

This message demonstrates that Cloud-based COS for BDS is working.

But, what about local?

We must edit the underlying json document that provides the storage location information for BDS. This process is done by BDS Support through a Support ticket. The new vault is in an onsite COS environment and uses the vault information that is provided by the COS administrator. That information is sent to the BDS support team.

So as not to leave orphan records, we delete the record we created. Because this is blockchain, the path of creation, listing, and deletion is still recorded in the blockchain; however, the off-chain record is deleted and the blockchain is updated to reflect its deleted status.
To delete a document, use the Delete set of APIs (the single document deletion), as shown in Figure 51.

![Figure 51  Delete Document API](image)

After a document is deleted, any further requests result in a 410 status that says the document or file is deleted. The status check still indicates a 200 status because a record of the file is still available, but the file no longer exists in the BDS.

To test the capability to use a new onsite COS linkage, a COS vault was established on a remote COS appliance and the login information was sent to the BDS team. The BDS team re-pointed the internal credentials and address of the vault being used by the BDS instance in our Hyperledger blockchain.
What we expect to see is that it does not find the document we tested with the cloud-based COS, but does allow us to load a new document. Figure 52 shows our request to upload a new document.

![Figure 52](image1.png)  
**Figure 52** Attempt to upload into the new COS onsite storage

The successful upload results are shown in Figure 53.

![Figure 53](image2.png)  
**Figure 53** Successful document upload
As you can see by the 200 series Response Code, the file was successfully uploaded to the new onsite location, just as it is if we were pointed to the cloud COS instead. To verify it was loaded, we download it. Figure 54 shows the download request.

![Figure 54 Download of single document or file](image)

The download results are shown in Figure 55.

![Figure 55 Successful download of the uploaded document](image)

As you can see in Figure 55, the first few lines of our PDF file are shown in the Response Body and the Response Code shows a 200 which shows success.
In this section, we showed that Hyperledger blockchain can be used successfully with off-chain storage in the cloud or with local onsite storage. As with any type of blockchain on or off-chain storage, you must be sure that it is globally accessible to all members of the blockchain network that might need to see the data that is stored there.

Summary

This Blueprint delivers an end-to-end blockchain infrastructure that is ready for any blockchain implementation.

Clients are not locked into one version of an application stack because the solution uses open industry standards. Instead, clients can pick and choose the open source Hyperledger-based solution that is best for their environment.

IBM Blockchain Protocol on IBM Cloud Private with the IBM Blockchain Document Store provides clients with an enterprise-grade on-premises cloud stack that is enabled by IBM compute and storage infrastructure. With this IBM Storage Solution for Blockchain, clients can rest easy knowing that their data is within their control and that their solution allows them to use blockchain related services and manage operational expenses within the confines of their environment.

For more information

For more information, see the following resources:

- How to get the benefits of cloud behind your firewall: IBM Cloud Private:
  https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=KUW12527USEN
- IBM FlashSystem 9100:
- IBM Redbooks: Implementing the IBM Storwize V7000 and IBM Spectrum Virtualize V7.8:
- IBM Redbooks: VersaStack Solution for File Storage Using IBM Storwize V5030 and Windows Server 2016:
- IBM Spectrum Connect:
- IBM Blockchain:
  https://www.ibm.com/blockchain
- IBM Cloud Object Storage:
  https://www.ibm.com/cloud/object-storage
- IBM DS8880 and IBM Z Synergy:
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.

The following terms are trademarks of other companies:

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.

Other company, product, or service names may be trademarks or service marks of others.
Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.
