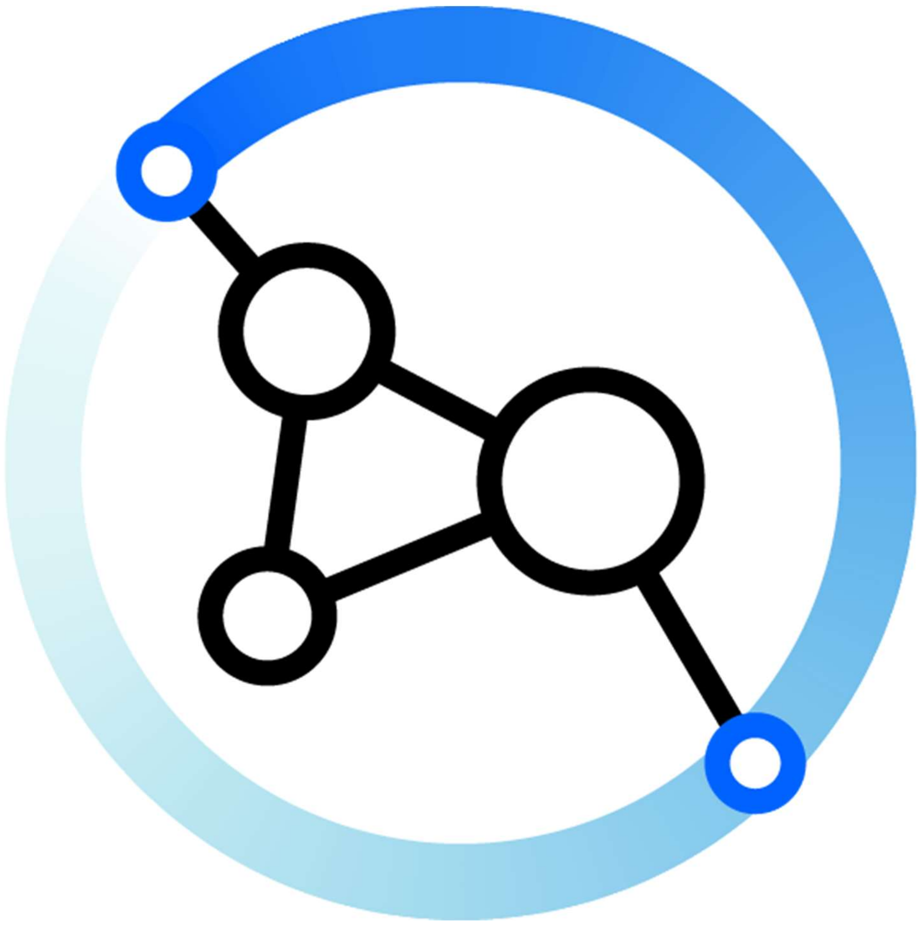
IBM API Connect v10.x Deployment WhitePaper

Author: Chris Phillips



Version: 1.4

Date: 10th Oct 2020

1 Table of Contents

1.1 Table of Figures

1.2 Table of Tables

1.3 Version History

The table below provides a brief description of what has changed between versions of this Whitepaper

VERSION	CHANGE	DATE
1.4	Clarification on Quorums and Block Storage	10/10/2020
1.3	Updated Links	22/07/2020
1.2	Initial Publication	24/06/2020
1.1	Internal Release	23/06/2020
1.0	Internal Release	15/06/2020

Table 1 Document Version History

2 Introduction to the Whitepaper

IBM API Connect is an end-to-end solution that allows users to create, secure, manage, socialize, monetize, and analyze APIs. It provides a powerful set of capabilities for turning backend RESTful, SOAP, or GraphQL services into managed services. This is done by publishing APIs to API Gateways, while enforcing lifecycle and governance controls on those APIs. API Connect enables users to expose APIs, through a developer portal, targeting application developers both inside and outside their organization. Additionally, the solution's analytics tooling helps API providers and API consumers better understand the health and consumption of deployed APIs.

This paper is a technical deep dive on the deployment options of the product. For high level information on API Connect please visit our Marketing pages (https://www.ibm.com/cloud/api-connect) or the Knowledge Center (https://www.ibm.com/support/knowledgecenter/SSMNED_v10/mapfiles/getting_started.html). The sections below cover the major components of API Connect, as well as considerations for configuring different clouds and environments within API Connect so that users can be successful in their API strategies. It targets Solution and Integration Architects.

The supported software and hardware for API Connect deployments is available through the software compatibility reports, by searching for API Connect:
https://www.ibm.com/software/reports/compatibility/clarity/softwareReqsForProduct.html

2.1 Terminology

Term	*Definition*
Target Service	A target service is a running application that connects to systems of record and can provide some level of business logic. Services could be SOAP services, GraphQL, or RESTFUL services.
API	An API is a user focused definition of a backend target service that is deployed to a gateway serving as a proxy to that backend target service.
Cluster	A group of nodes that are running together.
Node	An instance within a cluster.
Worker Node	VM or a physical machine that have some form of compute resources to run the pods for a given service.
Data Center	A data center is a collection of machines that are in the same geographic area.
Availability Zone	An availability zone is one or more data centers that are connected by way of a low latency connection. A Low Latency Connection is required.
Catalog	A catalog is a staging target and behaves as a logical partition of the gateway and the Developer Portal.
Service	The name given to each deployed instance of the API Connect subsystems: API Management Service, API Gateway Service, Analytics Service, and Developer Portal Service.
API Connect Cloud	An API Management service with one or more Portal, Analytics, and Gateway Services. This can be on prem, managed instance, or in a cloud.
High Availability	The ability of the solution to continue to a successful operation without manual intervention in the event of a failure of a subset of the system components.
Disaster Recovery	The process of recovering the successful operation of the solution in the event of a total loss of the current infrastructure. For example, recovering from a backup into a new region.
Active Site	Components are running and serve traffic.
Hot Standby Site	Components that are running and are configured but do not serve traffic. A manual operation is required prior to the components serving traffic.
Warm Standby Site	Components that are running but are not configured, so are unable to serve traffic. Configuration must be loaded prior to the components serving traffic.
Passive/Cold Standby Site	Components that are not running, so are unable to serve traffic.
Very Low Latency Connection	This article defines a Low Latency Connection to be one of less than 6ms as well as less than 1% connection that are unexpectedly dropped due to infrastructure issues.

Table 2 Key Terminology Definitions

2.2 Personas

API Connect has several out of the box personas. These are used in the context of this paper to describe best practices across an enterprise organization:

Persona	***Description***
Shavon, the API Developer	Is in charge of API development for her organization. Her focus is understanding data, services, and information needed from an API and then building out that API based on those requirements. Eventually publishing it to testing, staging, and production environments.
Jason, the API Lifecycle Manager	The lead for the line of business that Shavon works on. He works with Shavon on API requirements and then reviews and manages the approval of new APIs or updates to APIs in the production environments.
Steve, the Provider Organization Owner	Is in charge of coordinating the delivery of APIs across multiple lines of business and development groups. Making sure each group has the resources that they need and that they are publishing their APIs to the appropriate environments.
Will, the Cloud Manager	Is in charge of the IT administration at his organization. He works with his team to deploy the API Connect software, configure the API Connect cloud, and then give Steve access to the resources he needs to get the API initiative off the ground.
Marsha, the Community Manager	Is in charge of Application developer success for her organization. She focuses on making the API portal as straight forward as possible and monitors and manages the API consumer groups.
Andre, the Application Developer	Is in charge of building out new applications and webservices for his organization. To do this he needs to consume APIs. He navigates to the API Developer Portal in order to register his application and consume the APIs needed to build out his app.

Table 3 Personas

These personas might not have a one to one fit with any organization's structure or internal roles. Additionally, individual people at an organization might fill many of these roles and not have a focus on just one role. Regardless, these personas help understand what the targeted task is for the different parts of API Connect.

3 API Connect Subsystems

3.1 Components - Introduction

There are four different Components in API Connect v10.x:

- Management System (API Manager)
- Gateway Service
- Analytics Service
- Developer Portal Service

In this section we list the pods and containers deployed as of v10. However, as the product evolves, the pods and containers that are deployed might change.

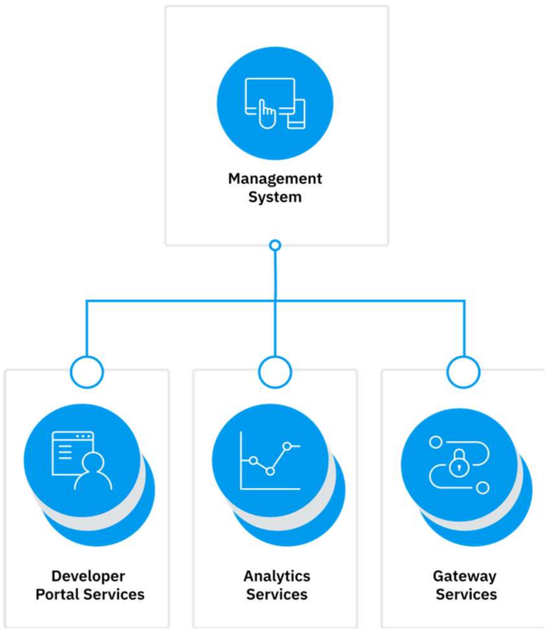


Figure 1: API Connect Components

3.2 Management Service

The Management Service provides two functional roles: the API Manager and the Cloud Manager.

- The Cloud Manager controls the infrastructure of the API Cloud. This is typically accessed by Infrastructure or Operation teams only.
- The API Manager controls the creation, publication, and management of APIs.

To summarize, the Management Service is the central coordinator or "brain" of the whole solution:

- It hosts the Cloud Manager user interface, targeting Will the Cloud Manager and his team.
- It hosts the API Manager user interface, which is leveraged by Steve, Jason, and Shavon.
- Enables API provider lines of business to build and publish APIs.
- It contains a persistent database that is used to store the configuration data about the system.
- It provides a rich set of RESTFul and CLI commands to automate API Management tasks for your organization.
- It maintains and manages the connection to the user registries that validate both providers and consumers of APIs.

3.2.1 Pods in the Management Service

There are multiple different microservices in the API Manager. Shown below is a pod level view of a high availability or standard deployment. Note that there are multiple instances, or replicas (as called by Kubernetes), of the microservices.

```
NAME                                                                  READY   STATUS      RESTART   AGE
<management-identifier>analytics-proxy-5745f87dfd-jawpf               1/1     Running     0         13d
<management-identifier>analytics-proxy-5745f87dfd-acopf               1/1     Running     0         13d
<management-identifier>analytics-proxy-5745f87dfd-jmgpf               1/1     Running     0         13d
<management-identifier>apim-68668d9b67-5sml7                          1/1     Running     0         13d
<management-identifier>apim-68668d9b67-7rn58                          1/1     Running     0         13d
<management-identifier>apim-68668d9b67-iwsrr                          1/1     Running     0         13d
<management-identifier>apim-initschema-9dntc                          0/1     Completed   0         13d
<management-identifier>client-downloads-server-7957d9b795-hjhs6       1/1     Running     0         13d
<management-identifier>client-downloads-server-7957d9b795-pldxn       1/1     Running     0         13d
<management-identifier>client-downloads-server-7957d9b795-v2gf5       1/1     Running     0         13d
<management-identifier>hub-67476964c4-cdasd                           1/1     Running     0         13d
<management-identifier>hub-67476964c4-123da                           1/1     Running     0         13d
<management-identifier>hub-67476964c4-rr8wz                           1/1     Running     0         13d
<management-identifier>juhu-85978fcc54-bw6cf                          1/1     Running     0         13d
<management-identifier>juhu-85978fcc54-cnh8k                          1/1     Running     0         13d
<management-identifier>juhu-85978fcc54-wktk9                          1/1     Running     0         13d
<management-identifier>ldap-76f99dbfbd-jzqgn                          1/1     Running     0         13d
<management-identifier>ldap-76f99dbfbd-qqvqd                          1/1     Running     0         13d
<management-identifier>ldap-76f99dbfbd-xdr8c                          1/1     Running     0         13d
<management-identifier>lur-857466d8db-7chjs                           1/1     Running     0         13d
<management-identifier>lur-857466d8db-bww78                           1/1     Running     0         13d
<management-identifier>lur-857466d8db-fkdmx                           1/1     Running     0         13d
<management-identifier>lur-initschema-p56bt                           0/1     Completed   0         13d
<management-identifier>nats-operator-6f6cbd4855-dpfz7                 1/1     Running     0         13d
<management-identifier>nats-streaming-operator-5bcd777c88-vd666       1/1     Running     0         13d
<management-identifier>natscluster-0                                  1/1     Running     0         13d
<management-identifier>natscluster-1                                  1/1     Running     0         13d
<management-identifier>natscluster-2                                  1/1     Running     0         13d
<management-identifier>stancluster-0                                  1/1     Running     0         13d
<management-identifier>stancluster-1                                  1/1     Running     0         13d
<management-identifier>stancluster-2                                  1/1     Running     0         13d
<management-identifier>portal-proxy-947448799-7fw4z                   1/1     Running     0         13d
<management-identifier>portal-proxy-947448799-w7qgr                   1/1     Running     0         13d
<management-identifier>portal-proxy-947448799-wr497                   1/1     Running     0         13d
<management-identifier>postgres-b5bd65d79-qp996                       2/2     Running     0         13d
<management-identifier>postgres-backrest-shared-repo-b8c5dbf65-xssc2  1/1     Running     0         13d
<management-identifier>postgres-gngl-54779b7f4-hpkb9                  2/2     Running     0         13d
<management-identifier>postgres-hldt-68694b7d94-g4dbn                 2/2     Running     0         13d
<management-identifier>postgres-operator-8975b7446-ghbz6              4/4     Running     1         13d
<management-identifier>postgres-pgbouncer-678ff9b7c5-j2tzt            1/1     Running     0         13d
<management-identifier>postgres-stanza-create-hwl5b                   0/1     Completed   0         13d
<management-identifier>taskmanager-5fdcdd9448-672rr                   1/1     Running     0         13d
<management-identifier>taskmanager-5fdcdd9448-sqwrr                   1/1     Running     0         13d
<management-identifier>taskmanager-5fdcdd9448-vhqgk                   1/1     Running     0         13d
<management-identifier>turnstile-9fbf64698-nk8xc                      1/1     Running     0         13d
<management-identifier>ui-c7b9c4bd7-7dp95                             1/1     Running     0         13d
<management-identifier>ui-c7b9c4bd7-md895                             1/1     Running     0         13d
<management-identifier>ui-c7b9c4bd7-zz5hl                             1/1     Running     0         13d
percona-server-mongodb-operator-568f85969c-8sh7x                      1/1     Running     0         13d
<test-database-identifier>rs0-0                                       2/2     Running     0         13d
<test-database-identifier>rs0-1                                       2/2     Running     0         13d
<test-database-identifier>rs0-2                                       2/2     Running     0         13d
```

Figure 2: High Availability Install of API Manager

Pod or Container	***Description***	***Default No. of Replicas***
apim	The core microservice in the API Management Service. It handles the communication with the other services and is the backend for the UI. Any action taken in API Connect (logging in, publishing a product, creating a Catalog) is coordinated through this microservice. A standard (HA) deployment has at least three pods, deployed across the different worker nodes. This number of pods can be scaled up based on load.	3
ui	This is the graphical User Interface microservice. When accessing the Cloud Management or API Management web console, users are directly interacting with this microservice. A standard deployment has 2 pods, across different worker nodes and this can be scaled up as necessary.	3
postgres	Postgres database pods.	1
postgres-backrest-shared-repo		3
postgres-operator		1
postgres-pgbouncer		1
analytics-proxy	This microservice component handles communication with the analytics-client (See section 3.4.1). This loads the analytics visualizations into the API Management console or handles configuration changes on the Analytics Service (analytics off-loading, decreasing analytics storage length, for example). A standard (HA) deployment has 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
portal-proxy	This microservice component handles requests from the CLI for communication with the Portal Service (See section 3.5.1).	3
juhu	Coordinates user authentication and token management for the API Management service. A standard (HA) deployment has 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
ldap	The LDAP microservice is leveraged to configure a connection to a LDAP registry for user authentication. A standard (HA) deployment has 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
lur	Local user registries are by default configured for the APIC product. These can be leveraged as a user registry for user authentication for the different components (Cloud Manager, API Manager, Portal etc.). A standard (HA) deployment has 3 pods,	3

Pod or Container	***Description***	***Default No. of Replicas***
	deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	
client-downloads-server	This microservice handles the downloading of the CLI and local API designer from the API Management console. These downloads are for local (laptop) API development and for setting up CI/CD pipelines. A standard deployment has 2 pods, across different worker nodes and this can be scaled up as necessary.	3
taskmanager		3
natscluster		3
nats-operator	The Task Manger manages and regulate internal tasks in API Connect.	1
nats-streaming-operator		1
stancluster		3
Turnstile	Turnstile allows for communication between the Test Pods and the APIM core *(Optional with the Automated API behavior testing).*	1
Hub	The core microservice components for the testing capability *(Optional with the Automated API behavior testing).*	3
percona-server-mongodb-operator	Operator for the mongodb database.	1
rs	MongoDB instance, controlled by the Percona operator *(Optional with the Automated API behavior testing).*	3

Table 4 API Management Microservice Pods

3.2.2 Jobs in the Management Service

Jobs in Kubernetes are used for single tasks. When a job is run it creates the pods that are required and cleans them up. The job can be initiated by a cronjob or a request from kubectl (the Kubernetes CLI). API Connect creates the following jobs:

Jobs	***Description***
postgres-stanza-create	These jobs are used during installation and upgrades. They are used for setting up the database schemas properly.
schema-init-job	

Table 5 API Management

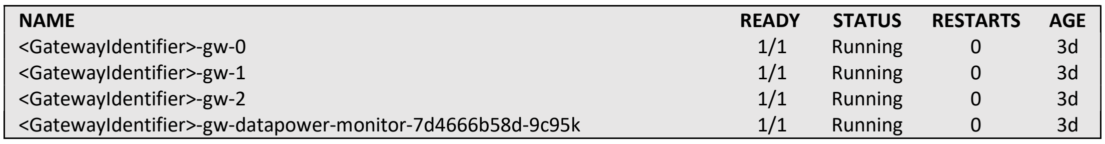
3.3 Gateway Service

The API Gateway Service is the runtime component. It is responsible for responding to incoming API requests from client applications:

- Validating that the application making the API call is permitted to access the API, for example, has an active subscription to a product that contains the API operation.
- Enforcing the security constraints defined by the API such as a requirement for authentication by using protocols such as Basic Authentication or OAuth 2.0.
- Enforcing rate limits so that the calling application cannot invoke the API more frequently than the API provider has specified.
- Invoking the outbound request to the backend service(s) defined in the API implementation, which might involve protocol transformation such as a RESTFul API calling out to a backend SOAP service.
- Aggregating responses from potentially multiple backend service calls and returning the relevant content of those requests to the caller.

In API Connect v5, the API Manager communicated with each individual gateway member. Since the introduction of API Connect v2018 the API Manager communicates with just the gateway ingress allowing for easy deployment to third party clouds. This component is responsible for managing Gateway configuration distribution and persistence across all gateway members in the cluster.

3.3.1 Pods in the Gateway Service



```
NAME                                                       READY   STATUS    RESTARTS   AGE
<GatewayIdentifier>-gw-0                                   1/1     Running   0          3d
<GatewayIdentifier>-gw-1                                   1/1     Running   0          3d
<GatewayIdentifier>-gw-2                                   1/1     Running   0          3d
<GatewayIdentifier>-gw-datapower-monitor-7d4666b58d-9c95k  1/1     Running   0          3d
```

Figure 3 : High Availability Install of Gateway Service

Pod or Container	***Description***	***Default No. of Replicas***
gw	The datapower runtime instance	3
gw-datapower-monitor	The monitoring pod ensures that each DataPower pod's peering information is consistent with the state of the cluster and provides logging for DataPower pod lifecycle events.	1

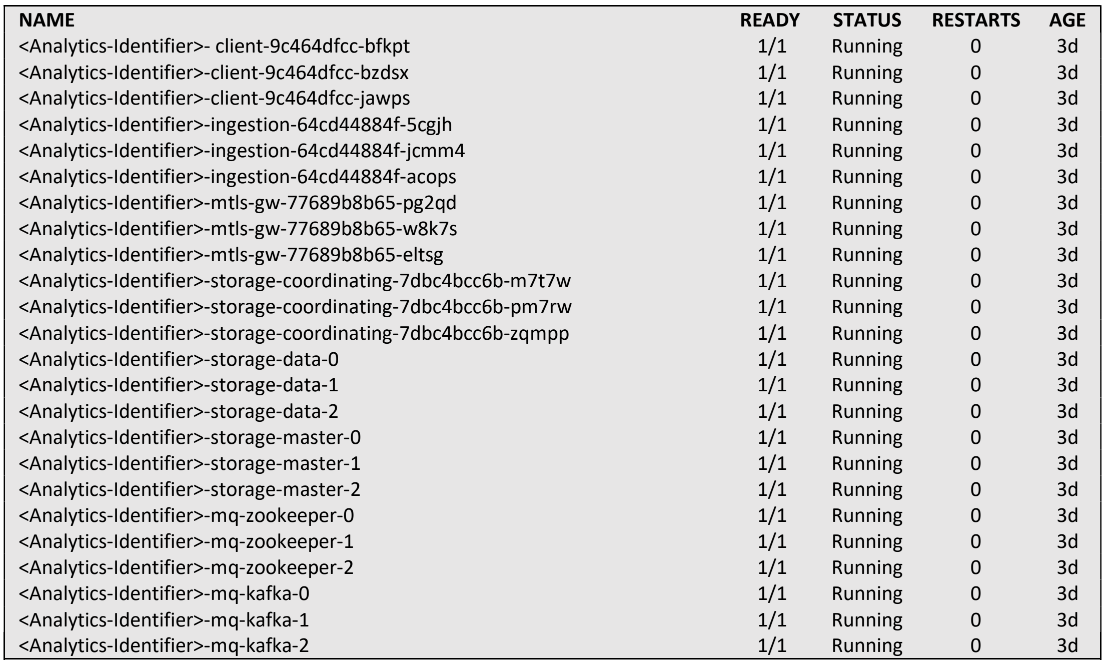
Table 6 List of Pods for the Gateway Service

3.4 Analytics Service

The Analytics Service is deployed separately from the API Manager. The Analytics Service is built on-top of Elastic Stack and performs the following tasks:

- Storing API event logs as they are processed from the Gateway Service.
- Processing API event logs from the gateway.
- providing visualizations of the aggregated metric data from the API events, so that API providers can better understand their APIs' health and consumption.
- Surfacing the API calls' raw log data to help developers debug.
- Off-loading API event records to target locations, for example, Splunk, Syslog, Kafka, and HTTP.

3.4.1 Pods in the Analytics Service



```
NAME                                                          READY   STATUS    RESTARTS   AGE
<Analytics-Identifier>- client-9c464dfcc-bfkpt                  1/1     Running   0          3d
<Analytics-Identifier>-client-9c464dfcc-bzdsx                   1/1     Running   0          3d
<Analytics-Identifier>-client-9c464dfcc-jawps                   1/1     Running   0          3d
<Analytics-Identifier>-ingestion-64cd44884f-5cgjh               1/1     Running   0          3d
<Analytics-Identifier>-ingestion-64cd44884f-jcmm4               1/1     Running   0          3d
<Analytics-Identifier>-ingestion-64cd44884f-acops               1/1     Running   0          3d
<Analytics-Identifier>-mtls-gw-77689b8b65-pg2qd                 1/1     Running   0          3d
<Analytics-Identifier>-mtls-gw-77689b8b65-w8k7s                 1/1     Running   0          3d
<Analytics-Identifier>-mtls-gw-77689b8b65-eltsg                 1/1     Running   0          3d
<Analytics-Identifier>-storage-coordinating-7dbc4bcc6b-m7t7w    1/1     Running   0          3d
<Analytics-Identifier>-storage-coordinating-7dbc4bcc6b-pm7rw    1/1     Running   0          3d
<Analytics-Identifier>-storage-coordinating-7dbc4bcc6b-zqmpp    1/1     Running   0          3d
<Analytics-Identifier>-storage-data-0                           1/1     Running   0          3d
<Analytics-Identifier>-storage-data-1                           1/1     Running   0          3d
<Analytics-Identifier>-storage-data-2                           1/1     Running   0          3d
<Analytics-Identifier>-storage-master-0                         1/1     Running   0          3d
<Analytics-Identifier>-storage-master-1                         1/1     Running   0          3d
<Analytics-Identifier>-storage-master-2                         1/1     Running   0          3d
<Analytics-Identifier>-mq-zookeeper-0                           1/1     Running   0          3d
<Analytics-Identifier>-mq-zookeeper-1                           1/1     Running   0          3d
<Analytics-Identifier>-mq-zookeeper-2                           1/1     Running   0          3d
<Analytics-Identifier>-mq-kafka-0                               1/1     Running   0          3d
<Analytics-Identifier>-mq-kafka-1                               1/1     Running   0          3d
<Analytics-Identifier>-mq-kafka-2                               1/1     Running   0          3d
```

Figure 4: High Availability Install of Analytics Service

Pod or Container	***Description***	***Default No. of Replicas***
client	The analytics client is built on top of Kibana. It provides the layer of access control for retrieving the analytics API event data from the analytics storage nodes. In addition, the pod hosts the analytics webserver for RESTFul APIs and renders the analytics visualization in API Connect. All read access to the analytics data passes through here.	3
ingestion	Analytics ingestion processes the data as it is passed from the gateway, modifying, transforming, and extending the data in the	3

	ingestion pipeline, preparing the data for storage, or off-loading to a third-party system. This is based on logstash.	
mtls-gw	The entry point for communication with the analytics components, API Manager, Gateway, and Portal, serving as an authentication mechanism for analytics communication.	3
storage-coordinating	The storage-coord nodes handle routing of read and write requests for data across the cluster. For example, a search request coming from the analytics client would go through here and be routed to the correct storage-data node. The same goes for writing: the request to write from the ingestion node needs to be routed to the correct storage-data node.	3
storage-data	The storage-data nodes handle reading and writing data. These nodes are going to do all the heavy lifting when it comes to disk I/O and CPU. These nodes perform the aggregations and searches against the data and writing the analytics API event data to the shards. The persistent volumes associated with these nodes hold the actual data, for both primary and replicas.	3
storage-master	The storage-master nodes handle managing the cluster and cluster state. Only one of these nodes is active at a time, the others are for backup in case the main one goes down. This node monitors the cluster, balancing it when needed, moving shards and data for instance. This node also manages the cluster state, so things that are separate from the data like field mapping data types, or the number of the nodes in the cluster, are stored here and propagated to all the other nodes in the cluster.	3
mq-kafka	The analytics message queue is an optional service that provides a reliable ingestion pipeline, especially when offloading analytics to a third-party system. The message queue ensures that the analytics pipeline is not blocked when either analytics storage or the offload endpoint becomes unavailable. Data collection continues internally in the analytics subsystem and is buffered if analytics offload has been configured. With the message queue enabled, data storage continues even if the offload endpoint is unresponsive. Additional benefits include better handling of traffic spikes and a larger buffer to handle Elasticsearch issues such as red indices and mapping issues. *This pod starts only if it is required by the installation.*	3
mq-zookeeper	A stateful set that manages the *mq-kafka* pods. *This pod starts only if it is required by the installation.*	3

Table 7 Analytics Microservice Pods

3.4.2 Jobs in the Analytics Service

Jobs in Kubernetes are used for single tasks. When a job is run it creates the pods that are required and cleans them up. The job can be initiated by a cronjob or a request from kubectl (the Kubernetes CLI). API Connect creates the following jobs:

Jobs	***Description***
cronjob-retention	This job manages the data retention of the historical API event data in the analytics storage datastore. When data is older than the retention period the job truncates the data. The default retention period is 90 days. This job runs at 1:30am system time.
cronjob-rollover	This job manages the index lifecycle of the analytics storage datastore. When run the job creates a new index and points the write alias of the ingestion pipeline to said index (rollover). This runs on the 15th and 45th minute of every hour.

Table 8 Analytics Microservice Jobs

3.5 Developer Portal Service

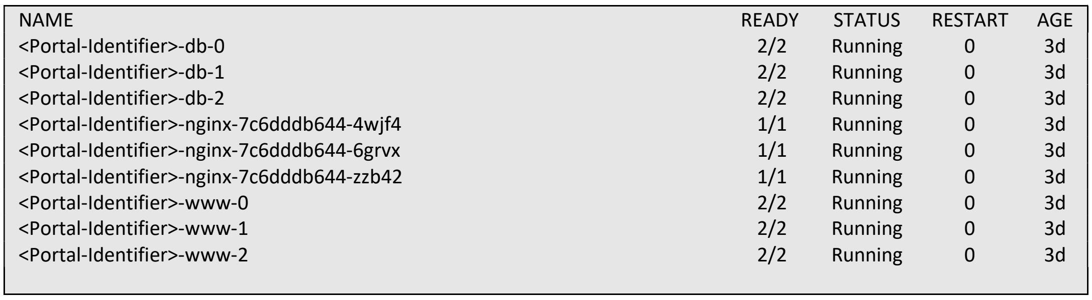
The Developer Portal Service is the component through which APIs are socialized to application developers. The following functionality is provided:

- Discovering the set of APIs and Products that are made available by an API provider.
- Registering applications to API consumer organizations.
  - Creation of an application's credentials is required to authenticate and identify the application when making API calls.
- Host content, communities, and forums on the Portal.

In many cases, the Developer Portal is the public face of an API program. The provider of the Developer Portal can customize the appearance of the Portal so that it matches the corporate branding requirements of the enterprise. It provides customers plenty of flexibility to model the look and feel of the Developer Portal site.

3.5.1 Pods in the Developer Portal Service

The Portal Service has a slightly different deployment strategy than the other API Connect components. The other components have a one to one relationship between microservice or container to pod. The Portal runs multiple containers within a pod and these are called out below. In the image below, there are two types of pods that have multiple containers, indicated by the 2/2.



```
NAME                                          READY   STATUS    RESTART   AGE
<Portal-Identifier>-db-0                      2/2     Running   0         3d
<Portal-Identifier>-db-1                      2/2     Running   0         3d
<Portal-Identifier>-db-2                      2/2     Running   0         3d
<Portal-Identifier>-nginx-7c6dddb644-4wjf4    1/1     Running   0         3d
<Portal-Identifier>-nginx-7c6dddb644-6grvx    1/1     Running   0         3d
<Portal-Identifier>-nginx-7c6dddb644-zzb42    1/1     Running   0         3d
<Portal-Identifier>-www-0                     2/2     Running   0         3d
<Portal-Identifier>-www-1                     2/2     Running   0         3d
<Portal-Identifier>-www-2                     2/2     Running   0         3d
```

Figure 5: High Availability Install of Portal

Pod or Container	***Description***	***Default No. of Replicas***
portal db	This is the database pod which has 2 running containers. • portal db-dbproxy container • portal db-db container These containers are explained below.	3
portal db-dbproxy container	Handles communication with portal db container from the portal www pod.	-
portal db-db container	Hosts the Portal Databases.	-
portal nginx	This pod runs a single container. It is a simple proxy for communication.	3
portal www	This pod hosts the portal sites and contains the admin and web containers.	3
portal www-admin container	Handles communication and integration with the API Manager.	-
portal www-web container	The webserver running the Drupal websites.	-

Table 9 Portal Microservice Pods

4 Deployment Options

4.1 Introduction

API Connect can be deployed in two patterns.

1) Directly to an OpenShift or other Kubernetes Environment.
2) Appliance (OVA) install to an existing VMware Estate.

4.2 Kubernetes & OpenShift

API Connect can be installed in a Kubernetes environment running ingress or OpenShift routes that has a persistent storage solution, see Section 9. For a detailed explanation of different types of Kubernetes Environments, see Section 9.

4.3 Appliance (OVA)

The Appliances provide a base Kubernetes with helm, ingress, and local storage preconfigured.

Each Appliance has the containers that are required for their component only. For example, the Management Appliance contains the Management Containers but not the Analytics containers.

The APICUP installer is used to configure the appliances. For more information, see the Knowledge Center.

Note: Appliance OVAs do not run in VMware Fusion or VMware Workstation

5 Terms and Naming Convention

5.1 Introduction

An API Connect Platform is designed to work with a single organization or across multiple organizations within a large enterprise. API Connect provides four levels of segregation for the API Manager. These are as follows:

- API Cloud
- Provider Organization
- Catalog
- Space

Additionally, the deployment of Gateways, Analytics, and Portal Services can be designed for more levels of segregation. In order to ensure the platform is maintainable with minimum effort a consistent naming convention is recommended. This document provides a sample naming convention, although it can be adapted as necessary.

IBM API Connect has a host of free text variable, parameter, profile, and component names. While these are and can be named whatever the customer likes, it is good for uniform naming standards to be established as is common with other products and software across the middleware and service landscape.

In API Connect, most objects have both a Display Name, sometimes referred to as a Title, and a 'name' which refers to the object name. Names are restricted to the following characters a-z, 0-9, and -. Where appropriate, both display and functional names have been given.

Often each new business, value stream, brand, department, and individual have specific ways in which they like to name. This leads to every object being named differently for each new department and team. To alleviate this, the following naming conventions have been created as a standardized selection of IBM Cloud Service preference, based on experience with customers and internal developments.

This section provides some generic naming conventions. It is recommended that each customer maintains their own list of naming conventions that they use.

Note: Where possible, no object in the API Connect Cloud should be left with a default name of 'Default'. This causes confusion, makes email notifications difficult to follow, and can lengthen routine maintenance and management processes as an understanding of what each element does is made apparent.

5.2 API Development

5.2.1 APIs

There are many well documented conventions for naming APIs, which and often follow company standards.

Additionally, there are articles that describe API needs regarding versioning and standard formatting of naming, versions, and URL, and so on. (https://www.ibm.com/developerworks/library/mw-1710-phillips/index.html)

```
https://api.ABank.org/mortgages/v1.0/rates

https://api.sandbox.ABank.org/loan/v1.1/quote

https://developer.ABank.org/v1.0/balance
```

Figure 6: Example API Endpoints

5.2.2 Products

Products are a logical grouping of APIs and Plans. If API security is enabled, APIs are invokable when an Application in a Consumer Organization is subscribed to a plan. If no API security is enabled, then the APIs are invokable immediately after publication.

The purpose of a Product is to tie APIs together for a similar set of use cases. The Product must accurately describe the purpose of the collection of APIs that it contains. Though there is no technical limitation for the number of APIs in a product, from a management perspective it is recommended to have no more than seven APIs and three Plans for each Product. Products can contain many more APIs and Plans, but it creates a challenge to manage, govern, and consume.

Product names should use dashes instead of underscores, and should not contain numbers. Version numbers should not be defined in the Product name, because this causes confusion as it is static against the dynamic version number that displays next to the Product.

If there is a need to differentiate such as brands, these can be appended in parentheses as shown in the following table.

Title	Name
Current Account Information (England)	*current-account-Information-england*
Current Account Information (Scotland)	*current-account-Information-scotland*
Mortgage Offers (England)	*mortgage-offers-england*
Mortgage APR (England)	*mortgage-apr-england*
Mortgage Payments (UKBank)	*mortgage-payments (UKBank)*

Table 10 Example Product Names

5.2.3 Plans

Plans are used for rate limiting of API Calls and are used for monetization of APIs. Developer Organizations subscribe an Application to a Plan, not a Product. Each product can have multiple plans which can have different costs and potentially rate limits. Additionally, different plans might conform to predefined contract agreements between third parties, public, and internal users.

When an Organization has a predefined, well documented, and well understood convention for logical grouping of APIs and their limits, these should be reflected in plan names. This would typically map to a standard selection of upgrades.

Plan 1 - Title	**Plan 1 - Name**	**Plan 2 - Title**	**Plan 2 - Name**	**Plan 3 - Title**	**Plan 3 - Name**
Basic	*basic*	*Standard*	*standard*	*Advanced*	*advanced*
Bronze	*bronze*	*Silver*	*silver*	*Gold*	*gold*
Small	*small*	*Medium*	*medium*	*Large*	*large*

Table 11 Example Plan Names

5.3 API Connect Topology

5.3.1 API Cloud

The API Cloud is a physical segregation point. The API Cloud is made up of one Manager Service, one or more Gateway Services, and zero or more Portal Services and Analytics Services.

The API Cloud is the only way to physically segregate the Management component.

Large enterprises typically have four API Clouds, for the following purposes:
- Infrastructure Sandbox
- Development
- Test
- Production

Though often Development and Test

5.3.2 Provider Organization (pOrg)

A Provider Organization is a logical entity within an API Cloud. An API Cloud can contain multiple Provider Organizations. However, a Provider Organization can exist only in a single API Cloud. Each pOrg has a designated organization owner, which is assigned during pOrg creation from the Cloud Management console. The owner is in charge of the initial configuration and user on-boarding for that pOrg. There is complete isolation from other pOrgs. Each pOrg can be accessed by using the API Manager user interface. Users can belong to multiple pOrgs, with different roles, and corresponding permission sets.

A pOrg provides the following functionality:

- Publishing APIs and Products, as well as managing the lifecycle of those products
- Managing catalogs that contain deployed APIs

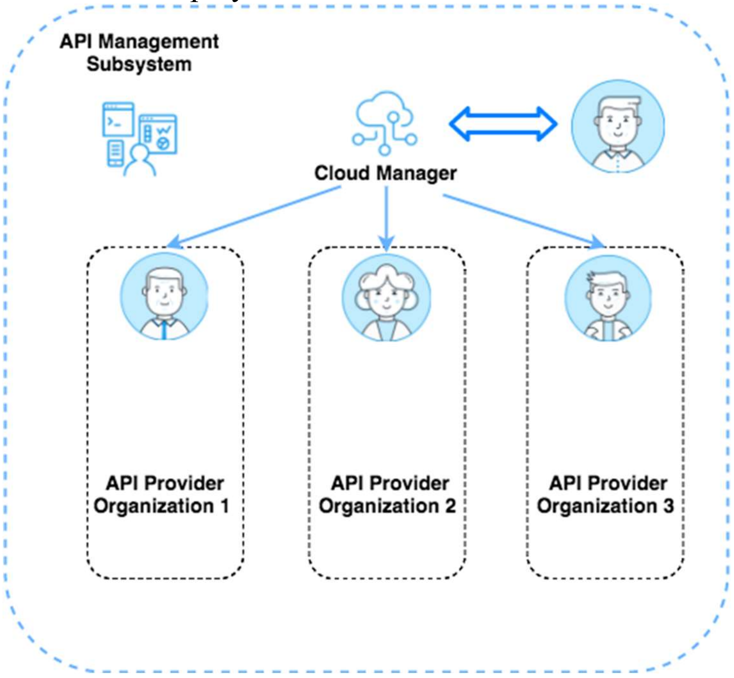

API Management Subsystem
Cloud Manager
API Provider Organization 1
API Provider Organization 2
API Provider Organization 3

Figure 7: Provider Organization (pOrg) Depiction

- Managing API subscriptions and API consumers
- Managing additional resources available to that organization (User Registries, TLS profiles, etc.)

A poorly defined provider organization structure has a negative impact on the management and production of APIs

APIs in API Connect are bundled into Products and published into Catalogs, which are the point of consumption for APIs. There are no restrictions on the number of pOrgs that can be created within an API Connect installation and under pOrgs there is not a restriction on the number of catalogs that can be created. However, a poorly defined provider organization structure has a negative impact on the management and production of APIs. Often Provider

Organizations and Catalogs are used as another method to separate environments in API Connect. Those environments could either be for different working groups or for different release staging environments, for example, dev, test, and qa.

Here are some strategies for large enterprise use cases for leveraging pOrgs to segregate environments:

1. pOrgs to separate release staging environments, for example, dev, test, and qa.
2. pOrgs to separate development groups, or lines of business, that are building out APIs

These two strategies can be leveraged differently and in conjunction with each other depending on needs, size of organization, and size of API initiative.

Provider Organizations might be named after an environment, for example, SIT01 or by an Organization's business unit, for example, UK Sales or UK HR.

An Organization's Display Name might be several words, each beginning with a capital letter. Its logical name must represent the Display Name, it must not contain spaces, should use a dash rather than an underscore. For best display and reference, it is recommended that the Organization not be any more than 2 words and 15 characters. The maximum length is 81 characters.

Where multiple environments might exist of the same type, for example, System Integration Test, the environment's unique identifier should be two numbers after the environments abbreviations beginning at '01' through '99'.

Display Name	Name
OAT01	*oat01*
SIT01	*sit01*
SIT02	*sit02*
LUAT01	*luat01*
UK Sales	*uk-sales*
UK HR	*uk-hr*

Table 12 Example Provider Organization names

5.3.3 Catalog

The Catalog is where API Publishing and Consumption is handled and a Catalog has a one to one relationship with a Portal Site. An API Product is staged to a Catalog, published to the Portal site and then API consumers can begin subscribing and making API calls. Catalogs provide isolated API run-time environments within a pOrg. The Catalog segregation has an impact on the actual API endpoints that are created. API naming the following scheme within API Connect:

```
https://<Load Balancer>/<pOrg-name>/<Catalog>/basepath/path
```

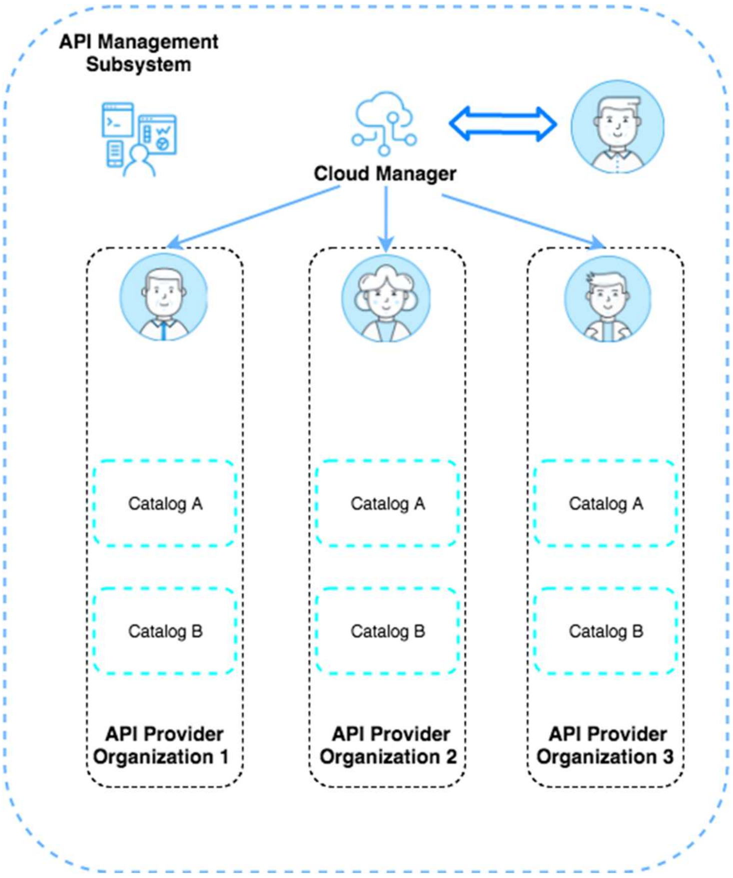


Figure 8: Catalog Depiction

Catalogs can impact API consumption, and also impact logical partitioning on the API Gateway. This is why catalogs are frequently leveraged as a way to separate environments like Development, Test, and QA within a pOrg.

There are permission sets within a pOrg and those permissions can be scoped to the pOrg, Catalog or Space level. Users can belong to multiple pOrgs, Catalogs, and Spaces with different permission sets in each.

Within a Catalog there is the following functionality:

- Configure Gateway Services for the Catalog
- Configuring the Developer Portal Service for API consumers
- Managing API Consumer organizations
- API lifecycle management and approvals (API stage versus API publishing)
- API consumer (Application developer) on-boarding and user registries
- API endpoints - The URL for API calls and the Developer Portal are specific to a particular Catalog
- TLS Client Profiles to be used in the Catalog
- OAuth providers to secure access to APIs in the Catalog
- User defined policies - Each Catalog can also have user defined policies to extend the out of the box policies available to build APIs

Even though there is a one to one relationship with a Catalog to a Portal site there might be more than one API development group, line of business, or organization that want to socialize their APIs to a single Portal Site. Typically, these different development groups want to maintain complete control of their APIs and the governance around them, but provide Application Developers (API consumers) a single marketplace for API consumption. To provide this capability there is the concept of Spaces in API Connect, that provides this isolation.

Catalog names should be a single descriptive word. However, if a double-worded Catalog name is used then it should be dashed and not CamelCase. They should use Sentence Case with spaces for the Display Name.

A Catalog has a Display Name that the user sees, and an internal name. A Catalog's name should describe the catalog's purpose. For ease of maintenance, use the same name with different styles as shown in the following table. Display Names should use Headline Style, and internal names should be lowercase with dashes instead of spaces.

Display Name	**Name**
Mortgages	*mortgages*
Open	*open*
Loans	*loans*
Current Accounts	*current-accounts*

Table 13 Example Catalog names

5.3.4 Space

A Space provides a level of isolation within a Catalog. In API Connect this capability is often referred to as *syndication* to describe the management and control of APIs that can be given to individuals, groups, or lines of business, with all of the APIs centralized to a single Portal

Site for API consumption. This is visible only in the API Manager, the Portal Site and Service does not have any concept of Spaces.

There is no technical limitation to the number of Spaces in a Catalog. However, from a management perspective it is recommended to have no more than seven Spaces per Catalog. We have found that customers with large numbers of Spaces have challenges managing and governing them.

Spaces within a Catalog have the following functionality:

- Ability to define isolated Gateway Services (Third Party Organization socializing APIs can register their own Gateway Service and can be made available to their Space only.)
- Map members to roles and permission sets
- Define custom resources for the Space (TLS profiles, OAuth providers, and so on.)
- Separated Analytics data (members scoped to Space, can view only analytics data within their space)
- Independent lifecycle management of APIs and Products.
- Management of API Consumers, Applications and subscriptions to Products published into that space, but socialized through the common API portal.

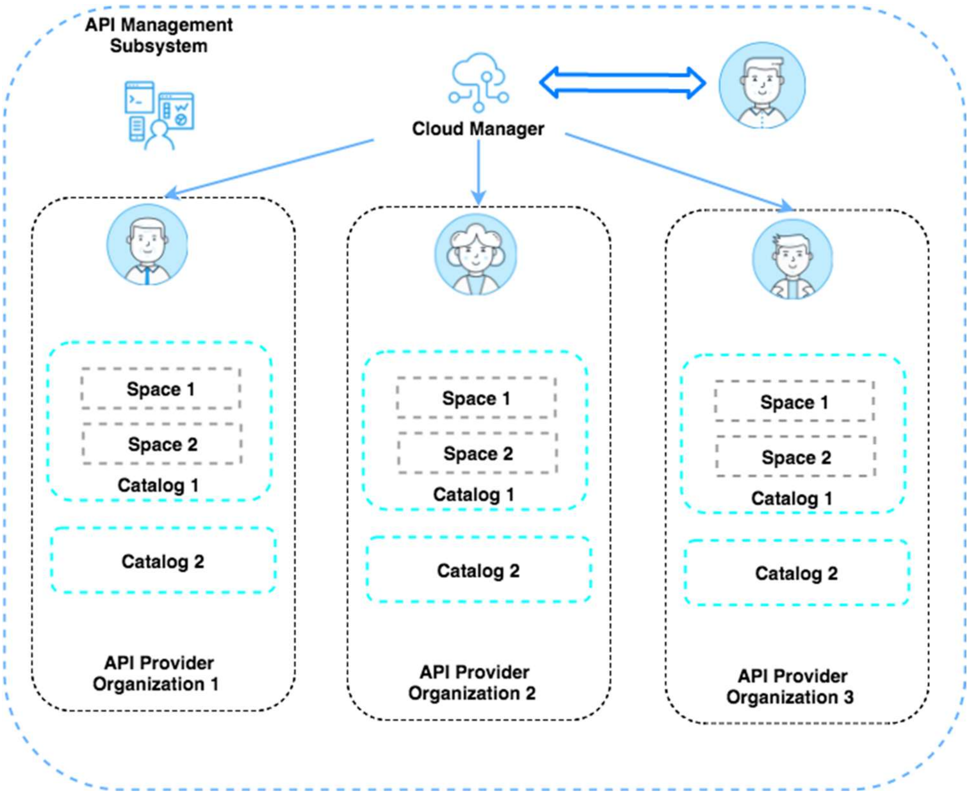


Figure 9: Spaces Depiction

Space names should be single word and descriptive. However, if a double-worded Space name is used then it should be dashed and not CamelCase and should use Sentence Case with spaces for the Display Name.

Display Name	Name
Balance	*balance*
Payments	*payments*
England Stores	*england-stores*
England Regulations	*england-regulations*

Table 14 Example Space names

5.3.5 Gateway Services

Gateways enforce runtime policies to secure and control API traffic, provide the endpoints that expose APIs to calling applications, and provide assembly functions that enable APIs to integrate with various endpoints. They also log and report all API interactions to the API Connect analytics engine, for real-time and historical analytics and reporting.

The Gateway Service is one or more gateway instances that are self-managed. This allows them to share quota enforcement information, revocation tokens, and new APIs.

5.3.5.1 Gateway Service Name

Services should be named relating to the Organization they are being used by and should be named in a descriptive form of the domain they are representing. This service is used by catalogs and spaces to define the Gateway Services that they will use, so a descriptive name is important.

Title
SIT01 Mortgages DP Service
OAT01 Current Account Payments DP Service
PROD01 Open APIs

Table 15 Example Gateway Service Names

5.3.6 User Registries

A User Registry should be easily identifiable to anyone looking at the system. Typically, the name describes what component the registry applies to, where it is based, and the fact that it is a user registry. API Connect can provide a local user registry or work with an existing LDAP or AD user registry.

The name cannot contain spaces, should be entirely in lowercase but should be shorter than the Title/Display Name. The registry location and “user registry” should be condensed to just the first letter of each.

Note: It is recommended to provide separate user registries for the API Manager, the Cloud Manager, and the Portal Service

Display Name	Name
API Manager Local User Registry	*api-manager-lur*
Cloud Manager Docker User Registry	*cloud-manager-lur*
Portal Service User Registry	*portal-lur*

Table 16 Example User Registry names

5.3.7 Roles

Each role within API Manager can have multiple words for its Title/Display Name to a maximum of 81 characters. The name should describe the type of permission being given.

The name must be lowercase and must use dashes instead of spaces.

Display Name	Name
Community Viewer	*community-viewer*
Role Administrator	*role-administrator*

Table 17 Example Role names

5.3.8 TLS Profile

TLS profiles can be used for communication among the Cloud Manager, the API Manager, and the Gateway, or for a TLS handshake with a load balancer. These cloud-based TLS Profiles are defined in the Cloud Manager.

Additionally, TLS profiles can be defined in the API Manager to be used by APIs to either call other APIs on the same gateway, or some form of implementation layer downstream such as an internal load balancer or microservice.

5.3.8.1 APIs

TLS Profiles to be used by APIs for handshakes to endpoints should have the following format; *purpose*-tls-*location*. The purpose should show the intended use of the profile and the location should describe the endpoint where the handshake occurs. The Display Name/Title is the same but in lowercase letters with spaces instead of dashes.

Display Name	Name
Authorise API TLS IHS	*authorise-API-tls-ihs*

Table 18 Example API TLS Profile names

5.3.9 Availability Zones

Availability zones organize API Connect operations based on your business needs. Availability zones are sets of logical or physical data centers containing one or more API Connect services. Multiple availability zones in your API Cloud provide redundancy and failover in the event of network issues.

The Default Availability Zone is created during the installation process. It contains the Management service that was configured by Install Assist. You register one or more gateway, analytics, and Portal Services in the availability zones to configure your API Connect cloud topology.

Availability Zones names should be descriptive and are organized by regions, global separation, or by the physical/logical separation of data centers. The Title can use spaces, capitals, lowercase and so on, but a short one or two-word title is preferred. An abbreviation can be used.

Title	Name
US Availability Zone	*us-availability-zone*
Hampshire Availability Zone	*hants-availability-zone*
Newport Availability Zone	*newport-availability-zone*
LDN South Availability Zone	*london-south-availability-zone*

Table 19 Example Availability Zones names

5.3.10 Naming of the Portal, Analytics and Gateway Services

Service naming should describe the type of service and match the location for the availability zone. There should be no spaces and each new word should start with a capital letter. Abbreviations should match those in the Availability Zone.

Title	Name
USPortal	*us-portal*
USGateway	*us-gateway*
HampshireAnalytics	*hants-analytics*
LDNSouthPortal	*ldn-south-portal*

Table 20 Example Portal, Analytics and Gateway Service names

5.3.11 OAuth Provider

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information. In API Connect, you can secure an API with OAuth.

In Cloud Manager, you configure both Native and Third-party OAuth providers that can be made visible to selected Provider organizations. The OAuth Provider configuration is based on the OAuth 2.0 Specification, which is available at https://tools.ietf.org/html/rfc6749. Knowledge of the OAuth 2.0 specification is required to implement an OAuth Provider in API Connect. For more information, see https://www.ibm.com/support/knowledgecenter/SSMNED_v10/com.ibm.apic.cmc.doc/capic_cmc_oauth_concepts.html.

Title	Name
Zendesk	*zendesk*
Ubuntu One	*Ubuntu-one*

Table 21 Example OAuth Provider Names

5.4 Consumer Organizations

Consumer organizations are a means to share applications and their credentials. There is no hard limit on the number of members of a consumer organization. However, members must be in a consumer organization only if they are sharing application credentials.

Consumer Organizations can be created both on a Portal site and on the management server. Typically, the Portal Site would be explored by external entities either to the business itself or external to Provider teams, brands, and value streams. There is no way to enforce naming standards on external consumers. These guidelines here are provided to assist internal consumer organizations. For security reasons there is no uniqueness constraint on Consumer Org titles.

The naming convention to be applied here should follow existing internal naming solutions for applications. When no existing convention exists, the format should follow the format of <Organization >-<ValueStream>-<TeamName>.

Developer Organization
UKBank-HR-Compensation
UKBank-Mortgages-Test
UKBank-Mortgages-Development
UKBank-Mortgages-View

Table 22 Example Developer Organization Names

5.5 Applications

Each application should be named to describe its goal and purpose. The name of the application should be human readable and clearly state the purpose of the application. There is no enforcement by the Developer Portal but it is standard practice to have unique application names in a consumer organization.

Title
HR employee payment
HR employee illness
mortgage-test ivt001
mortgage-test-ivt002
mortgage-test-ivt003
mortgage-test-endtoend

Table 23 Example Application names

5.6 Developer Portal

The developer portal is a server which stores all the information about each of the API Cloud's portal sites. There are limited objects that require naming conventions within the Portal such as Roles, Page Names, and Drupal Modules.

5.6.1 Roles

Roles should follow the rules already defined earlier in the document: Roles.

5.6.2 Page Names

The name of the page should be human readable and clearly state the purpose of the page. There is no enforcement by the Developer Portal but it is standard practice to have unique page names throughout a Drupal site.

5.6.3 Drupal Modules

The recommendation is to follow Drupal's conventions, these can be found here.
https://www.drupal.org/docs/creating-custom-modules/naming-and-placing-your-drupal-module

6 Isolation and Environment Segregation

6.1 API Provider

It is a good practice for customers to have at least two separate API Clouds. These allow the segregation of Production and Non-Production work loads. Industry Standards recommend that enterprises have three API Clouds, allowing for additional segregation between development and other non-production (staging) workloads. Each of these installations have complete separation and isolation, from a physical infrastructure level, software level, and data level. Users belong to these environments separately and might be given different roles with different permission sets in each.

There are two major reasons for this setup. The first is that development aims to be fast and iterative, with reduced governance around building, publishing, and testing of APIs. Production is typically a low touch environment, with complex governance controls around publishing and updating APIs. Secondly, it is good practice to have multiple environments to allow for upgrades to be validated prior to being applied to production.

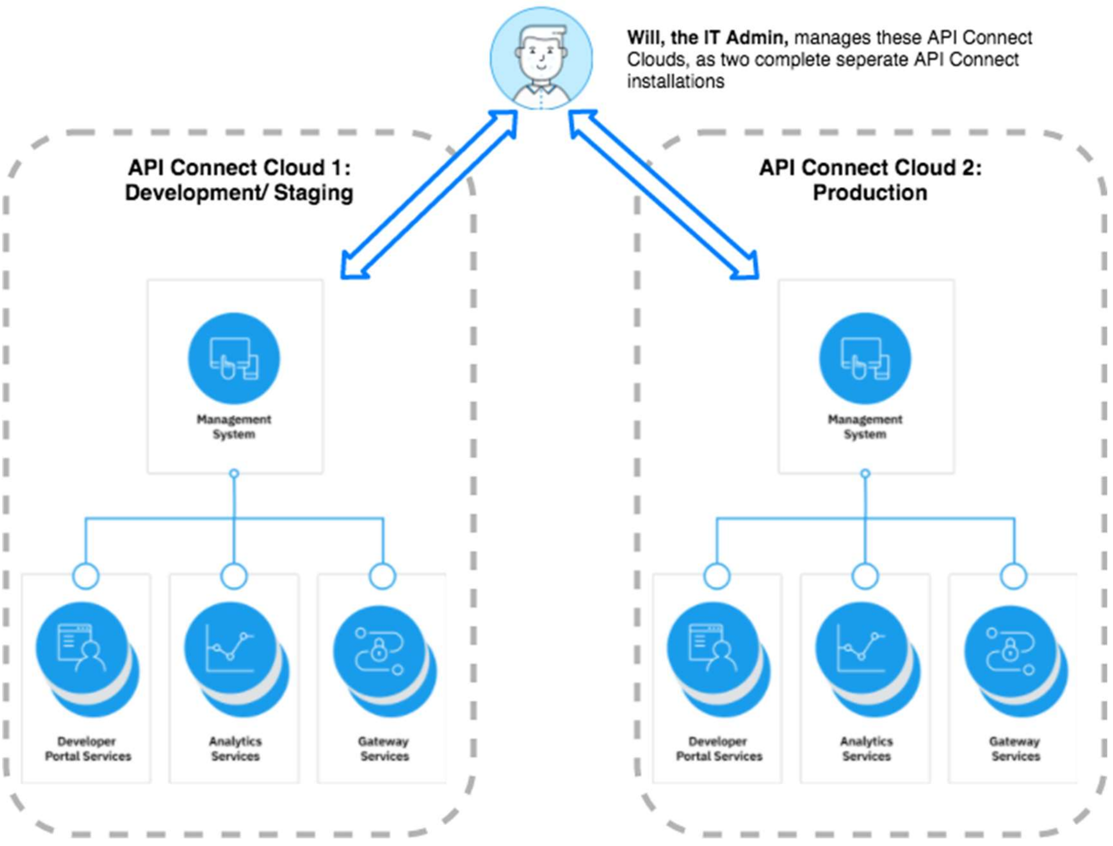

Will, the IT Admin, manages these API Connect Clouds, as two complete seperate API Connect installations
API Connect Cloud 1: Development/ Staging
API Connect Cloud 2: Production
Management System
Developer Portal Services
Analytics Services
Gateway Services
Management System
Developer Portal Services
Analytics Services
Gateway Services

Figure 10: API Connect Clouds Depiction

There are three common strategies for subdividing an API Cloud.

Scenario	API Cloud	Provider Organization	Catalog	Space
Multiple Lines of Business with Multiple Channels	Physical Isolation	Environment	Channel	Line of Business
Single Line of Business with Multiple Channels	Physical Isolation	Environment	Channel	-n/a-
Multiple Lines of Business with a single channel	Physical Isolation	Line of Business	Environment	-n/a-

Table 24 Logical Segregation Examples

It is recommended to use the same strategy across all API Clouds to reduce the risk of contamination. If the organization has one group of developers, it is recommended to separate each provider organization by environment. If each line of business has its own Development Team, it is recommended to split provider organizations by line of business. If multiple lines of business are required to be published to the same Catalog, it is recommended that strategy one is used with the line of business segregation occurring in spaces in each Catalog.

It is recommended that a Dev Ops Pipeline is used to deploy between each environment to ensure the process is the same and reduce risks.

6.2 API Consumers

There are often multiple API Consumer groups for API Projects. These are mapped on to Consumer Organizations in API Connect.

An API Consumer Organization is made up of API Consumers. These API Consumers create and manage the Applications and Subscriptions to Products. Within a Consumer org there are different roles and permission settings, that are mapped to users. Additionally, Products that are published to a Portal can be shared with specific API Consumer Organizations only. This is done by controlling the visibility settings during publication of the Product.

In Production, or exposed, environments the most common way to segregate the API Consumers Organizations are as follows:

- Internal or External Lines of Business
- External Enterprises

In Non-Production Environments these are segregated by one of the following strategies:

- Internal Testing Role
- Internal Testing Team
- Internal Tester

6.3 DataPower Gateways

6.3.1 Gateways and DataPower Instances

If DataPower is being deployed in Kubernetes, then it is recommended there is one DataPower Gateway Service for each Instance.

For other deployments of DataPower it is recommended that Production and Non-Functional Test environments have dedicated DataPower Clusters for the Gateway Service. For all other environments multiple Gateway Services can be deployed to a single DataPower Cluster in the same API Cloud.

6.3.2 Multi Cloud Topology

API Connect support Multi Cloud Topologies. This means that API Connect can have each component deployed to different locations. For example, Gateways and Analytics components might be deployed to the IBM Cloud, and the Portal and the Manager might be deployed on premise.

A single component must not span multiple clouds or Kubernetes Clusters.

6.3.3 Gateway Services and Catalogs

Gateway Services have many to many relationships with Catalogs. For example:

- A Single DataPower Gateway can be attached to multiple catalogs
- A Single Catalog can have multiple DataPower Gateways

A Single DataPower Gateway can be attached to multiple catalogs.

A Single Catalog can have multiple DataPower Gateways

In order to allow our v5 customers to run their existing APIs with no changes to their APIs in v10, a v5 Compatibility Mode gateway has been made available alongside the new high performant API Connect Gateway. Though a single Catalog can have Gateways in Compatibility Mode and the new API Gateway at the same time, it is recommended to associate only Gateways of the same type to a Catalog. By doing this you can ensure that all APIs that are published to a Catalog are in turn published to all Gateways associated to the Catalog. This makes the contents of each Catalog simpler to maintain as all APIs are in all Gateways.

6.4 Developer Portal

Each Catalog can have a single portal site associated to it. The site can be deployed to a single Portal Service. When it is deployed it cannot be moved to a different service. A portal service can host multiple portal sites that each represent a different catalog.

The recommended pattern is to have one Developer Portal Service per cluster. However, if a customer wants to physically isolate portals for different channels, for example, internal, external, and business partners, then different portal services are required.

6.4.1 Multi Cloud Topology

If there is a requirement to deploy to multiple clouds, each cloud can have its own Portal Service. Components must not span multiple clouds. However, note that a Catalog can be deployed to a single portal site only.

Components cannot span multiple clouds.

---

7 API Connect High Availability

7.1 Introduction

High Availability is the ability to withstand the loss of one or mode nodes without the service being interrupted. All the API Connect services in v10.x are designed for high availability.

The API Connect containers are enabled for High Availability by using Kubernetes. Kubernetes is a framework for Containers to allow them to be automatically scaled, restarted, and migrated between cluster nodes.

To support the high availability of the datastores all the underlying component database systems rely on distributed database replication logic in order to make sure all the members in the cluster are in sync, and carry the same data. The underlying component technologies use a quorum methodology to protect against outages, whether planned or unplanned.

The quorum logic comes from the underlying persistent datastores used by the API Connect components.

7.2 High Availability with a Container/ Kubernetes Deployment

In Kubernetes, containers are deployed in Pods. Each Pod can contain one or more containers. If a Pod terminates unexpectedly then it is automatically restarted. For example, if a container abends then a new Pod is deployed or, if a Worker Node goes down then all of the Pods running on it are restarted on a worker node with sufficient capacity, if possible.

Cluster size is dictated by the number of ReplicaSets defined for a pod in a given service. For a standard (highly available) install of API Connect, there are multiple replica pods forming a ReplicaSet for a given APIC component's microservices. This is most important for the database pods of each API Connect Service as explained in the Quorum Section below. For more information about ReplicaSets, see the Kubernetes documentation https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/ .

When you build a Kubernetes cluster for API Connect there are two main types of nodes; Master nodes and Worker nodes. The Master nodes are where the main Kubernetes System components run. The API Connect Components are deployed to the worker nodes. At a minimum for high availability, there must be 3 Kubernetes Master Nodes and three Worker Nodes. For more information, see these links.

- https://www.ibm.com/support/knowledgecenter/en/SSYGQH_6.0.0/admin/install/r_Orient_Me_CFC_HA.html
- https://kubernetes.io/docs/setup/independent/high-availability/

For a standard deployment of API Connect there are multiple replicas of the database pods for each of the Services and by default, these replicas are distributed across the worker nodes. This is dictated by affinity rules setup for an API Connect deployment to promote high availability configurations. These affinity rules are configurable for the Gateway but not for other components. For more information, see the Kubernetes documentation https://kubernetes.io/docs/concepts/configuration/assign-pod-node/.

Shown below is a diagram depicting high availability with Kubernetes deployments:

- An instance represents a single Replica Set of pods.
- A minimum of 3 worker nodes are required to provide High Availability, but a customer could use more.
- A minimum of 3 master nodes are required to provide High Availability, but a customer could use more.
- If a low latency network is available between locations these nodes could be spread across DCs, or in the case of public cloud deployments (AZs).

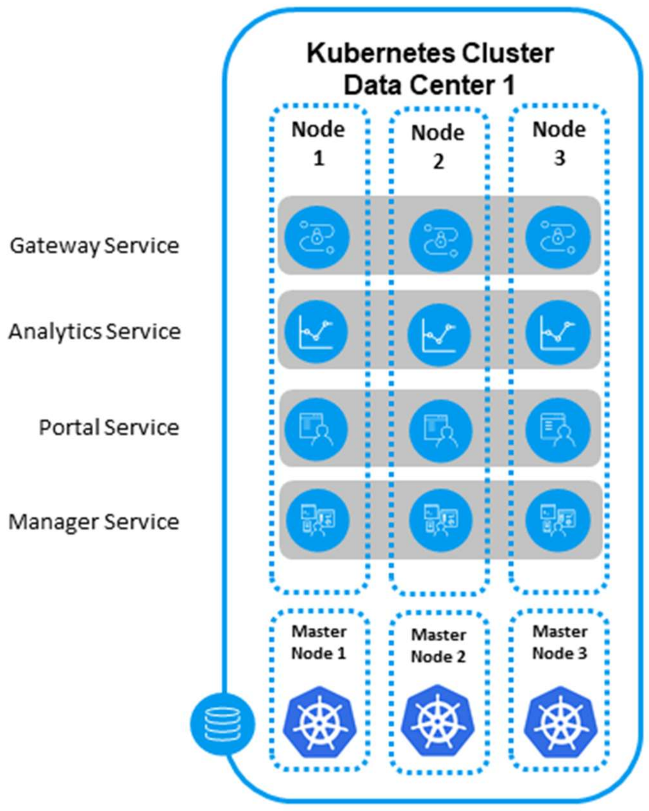


Figure 11: Highly Available Deployment

7.3 High Availability with OVA Deployments

Highly available OVA deployments for IBM API Connect v10.x follow similar recommendations to those described in section 7.1. Inside each OVA there is a running Kubernetes cluster with the required API Connect container images. In each VM of an OVA deployment, there is a Kubernetes master node (running the Kubernetes master node pods) and a single ReplicaSet for each type of pods for the API Connect component deployed. This is true for API Manager, Analytics, and Portal, *but not for DataPower Gateway VMs.*

The ETCD component of the Kubernetes Master nodes dictates high availability for the OVA deployment and has a quorum dependency. When quorum is lost for etcd pods across the cluster, etcd tells the API Server on the master node to stop serving requests, rendering that VM as unavailable. So, if a connection is dropped or a VM fails or the machine that the VM was running on fails, etcd loses quorum as well and that VM stops serving requests. In OVA deployments, for API Management, Analytics, and Portal Services, quorum is dominated by the etcd requirements. Since the Gateway does not have this same dependency its quorum behavior is dictated by the internal services as underlying component technology. The next section aims to simplify the quorum discussion.

7.4 Quorums

IBM's API Connect utilizes the same definition as Kubernetes for quorum:

$$\mathit{Failure\ Tolerance} = \frac{N-1}{2}\ ;\mathit{where\ N\ is\ the\ number\ of\ Nodes\ in\ the\ cluster}$$

Figure 12: Failure Tolerance formula

Failure Tolerance is # of cluster members that can fail and the cluster still maintain regular function. The Failure Tolerance is rounded down when the number of members in a cluster is even

Total Number of Nodes	Node1	Node2	Node3	Node4	Node5	# of Down Nodes	Cluster has a Quorum?
2	Active	Active				0	Yes
2	Active	Down				1	No
3	Active	Active	Active			0	Yes
3	Active	Active	Down			1	Yes
3	Active	Down	Down			2	No
4	Active	Active	Active	Active		0	Yes
4	Active	Active	Active	Down		1	Yes
4	Active	Active	Down	Down		2	No
4	Active	Down	Down	Down		3	No
5	Active	Active	Active	Active	Active	0	Yes
5	Active	Active	Active	Active	Down	1	Yes
5	Active	Active	Active	Down	Down	2	Yes
5	Active	Active	Down	Down	Down	3	No
5	Active	Down	Down	Down	Down	4	No

Table 25 The table above shows for a certain number of nodes how many outages can occur before a Quorum cannot be reached.

The table above assumes that the pods on a node are not automatically restarted on a different node.

The example below shows the failure tolerance for a given node.

$$N = 2\,; Failure\ Tolerance = \frac{2-1}{2} = .5$$

$$Round\ down\ to\ 0$$

$$N = 3\,; Failure\ Tolerance = \frac{3-1}{2} = 1$$

$$N = 4\,; Failure\ Tolerance = \frac{4-1}{2} = 1.5$$

$$Round\ down\ to\ 1$$

$$N = 5\,; Failure\ Tolerance = \frac{5-1}{2} = 2$$

$$And\ so\ on\ and\ so\ forth\ldots.$$

Figure 13: Example for failure tolerance

When a cluster has lost more members than the failure tolerance allows, that cluster loses quorum and loses the corresponding functionality. The different API Connect components have different behaviors when quorum is lost and these are be discussed in the following sections.

The other way to read the equation from above, is from the perspective of individual cluster members. Individual cluster members are continuously syncing with their peers and continue to operate as long as they can communicate with greater than 50% of members in the cluster (including themselves). When a member stops being able to communicate with a quorum of cluster members, quorum is lost and that member stops handling transactions. Additionally, many of the components rely on a primary/secondary relationship between members of the cluster, with the primary member coordinating the read/write requests or settling disputes when data gets out of sync. DataPower and Elasticsearch both leverage a primary/secondary relationship between cluster members, although the implementation is different.

In order to provide High Availability, a minimum of 3 cluster members is required to give some level of failure tolerance. Therefore, a minimum of 3 cluster members is required for High Availability deployments of API Connect v10.

A minimum of 3 cluster members is required for High Availability deployments of API Connect v10

This is of note as there can be no automatic failover with a deployment that has 2 or less cluster members. This is true for all IBM API Connect v10.x and v2018 components. This is a change from API Connect version 5.0.x and earlier.

7.4.1 Implications for API Manager, API Analytics and Developer Portal Services

The quorum requirements for these 3 components are straight forward and do not have a lot of options for customization. All three systems require three or more instances for high availability and to establish quorum. If quorum is ever lost for any of these components then the service instances stops handling write requests to their databases, rendering the services unavailable. The exception to this is the Analytics component, because the master nodes only are impacted by losing quorum. If the data nodes lose quorum, they can still read and write data.

Each of the systems can have the behavior fine-tuned, but this is for more advanced users only and to do this requires interacting with the databases directly.

7.4.1.1 Availability Table

The table below shows the availability of the API Connect Management, Analytics, and Portal Components. Each row says for a given number of nodes, what happens in various outage scenarios. The Table key is as follows:

Active	Node is taking requests and functioning as expected
Unavailable	Node is unable to function as it is not in Quorum
Down	Node is suffering an outage

Table 26 Availability Tables Key

Total Number of Nodes	Node1	Node2	Node3	Node4	Node5	No of Available Nodes	Cluster has a Quorum?
2	Active	Active				2	Yes
2	Unavailable	Down				0	No
3	Active	Active	Active			3	Yes
3	Active	Active	Down			2	Yes
3	Unavailable	Down	Down			0	No
4	Active	Active	Active	Active		4	Yes
4	Active	Active	Active	Down		3	Yes
4	Unavailable	Unavailable	Down	Down		0	No
4	Unavailable	Down	Down	Down		0	No
5	Active	Active	Active	Active	Active	5	Yes
5	Active	Active	Active	Active	Down	4	Yes

5	Active	Active	Active	Down	Down	3	Yes
5	Unavailable	Unavailable	Down	Down	Down	0	No
5	Unavailable	Down	Down	Down	Down	0	No

Table 27 Availability Table for up to five nodes for the Portal Service, Analytics Service or API Management System.

When a node is listed as unavailable, there is still a small number of functions that can be provided. These are listed below:

Manager
- API Manager continues to serve traffic and supports all read transactions
- New creation of resources and deployment is not be possible, for example:
    - No API Publication
    - Unable to create Applications

Analytics
- Continue to view analytics data that is already captured
- New ingestion data from the gateway is not be possible

Portal
- Portal is not accessible
- Unable to Register external consumers
- Unable to Register new Applications

7.4.2 Implications for API Gateway Service

The Gateway Service for IBM API Connect is IBM DataPower. IBM DataPower stores API configuration management, quota enforcement, and token or subscription management data.

Within a gateway cluster the API Connect Gateway Service instances utilize a primary/ secondary relationship, to reduce the risk of stale data. Customers can fine-tune the behavior, based on the network topology and their business requirements.

There are two main functions of Sentinel instances across the gateway cluster: the first is detecting primary node failure and the second is coordinating the failover process. Both these functions are where quorum is important to the Gateway Service. If quorum cannot be established, then the failover process does not work. This is important to prevent split-brain scenarios. A quorum of instances is required to handle failover, and the failure tolerance equation from section 7.4 applies.

This means that a minimum of 3 gateway instances are needed for high availability. These 3 Gateway instances can be separate application domains, separate tenants, or separate physical devices.

The API Connect Gateway Service orchestrates configuration management between gateways. In the event of half or more gateway nodes failing, quorum is lost. APIs are still processed in the Gateway Service but the following tasks do not take place.
- API Publications
- Onboarding cannot take place, for example, no new Applications or Subscriptions
- Rate Limiting (Quota Enforcement)

- Configuration Changes
- Token Revocation

In the event of a Quorum failure the API Gateway provides the option of a manual intervention, allowing a remaining gateway member to manually be marked as the primary and recover the gateway cluster. When recovered, the API Gateway Service behaves as if the master is up and handles both configuration and application traffic normally.

If there are only two DataPower Devices the API Connect Gateway Service cannot be highly available.

If there are only two DataPower devices this does not allow the API Connect Gateway Service to be highly available. When diagnosing a two DataPower device configuration, there are three failure scenarios:
1) Single tenant failure
2) Device disconnection (DataPower 1 can't communicate with DataPower 2)
3) Device failure (DataPower 1 or DataPower 2 fails)

The best option for establishing quorum for a two DataPower device configuration is the following:
1) DataPower 1 with tenant or domain 1.a and 1.b
2) DataPower 2 with tenant or domain 2.a

This protects against the failure scenarios 1, 2 and partially 3. This is because this setup can handle DataPower 2 failure but can't handle DataPower 1 failure. If DataPower 1 were to fail, DataPower 2 could continue to serve API traffic, but would not accept new API configuration data. Manual intervention would be needed with an admin manually setting the remaining gateway member as the primary and recovering the gateway cluster from there.

7.4.2.1 Availability Table

The table below shows the scenarios where a Quorum is lost and how many nodes are available.

Total Number of Nodes	Node1	Node2	Node3	No of Available Nodes	Cluster has a Quorum?
2	Active	Active		2	Yes
2	Active	Down		1	No
3	Active	Active	Active	3	Yes
3	Active	Active	Down	2	Yes
3	Active	Down	Down	1	No

Table 28 Availability Table for up to thrree nodes for the Gateway Service

When the gateway has lost quorum it is still able to serve APIs

When the gateway has lost quorum, it is able to serve APIs; however, it cannot receive new APIs, store tokens, or process rate limiting.

Though quorums can have more then three nodes it is not recommended for the API Manager, Analytics or Portal because of the over head of replicating data between the nodes

8 Multi Data Center Deployment Patterns

8.1 Introduction

In each diagram in these sections we draw the four key components. Please note that each component signifies where pods for that component can run. It is unlikely that there is a pod in every location for each component.

This section covers Patterns and Anti-Patterns for deploying to multiple Data Centers.

In each pattern description for each component of API Connect v10 there is a table dictating the number of components in a node available for a series of scenarios. The nodes listed as 1.x are in Site 1 and 2.x are in Site 2. Please note we have not included scenarios that mirror the ones below. For the Portal and Management Server there is an assumption that there is a human decision on when to initiate Disaster Recovery.

Anti-Patterns are common deployments that IBM does not recommend. There are several reasons for a Pattern to be considered an Anti-Pattern. These include

- Single Point of Failure
- Data Stores cannot be kept in sync
- Components can get out of sync

The table contains one of three statuses for each component in a node. These are explained below.

Status	Definition
Up	Component is operational on this node and accepting requests
Up Replication Only	Component is replicating data from the first site but is not serving requests.
Down	Component is in an unplanned outage on this node and not accepting requests
Offline	Component is in a planned outage on this node, or down because of insufficient available nodes in the quorum and not accepting requests
Down or Offline	Component is in an outage, might be planned or unplanned.
No Quorum	(Gateway Only) The Node is not in Quorum but is processing API traffic

Table 29 - Extended key for Availability Table

In the tables below, we assume the worst-case scenario and the pod cannot be restarted on another node during an outage.

Finally, we show if maintenance can be applied without requiring a complete outage of the running system. In this calculation we assume that we need to apply maintenance to the nodes listed as in an Up status and that the node needs to be taken down. In the Maintenance Column the color coding means the following:

Status	Definition
Yes	Maintenance can be applied without an outage.
No	Maintenance cannot be applied without creating an outage.

Table 30 Maintenance Key for Availability Table

In each diagram in these sections we draw the four key components. Please note that each component signifies where pods for that component can run. These pods are deployed depending on the Affinity and Anti-Affinity rules provided in Kubernetes. For more information, see https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity.

For example: In an HA environment there are two UI pods in the management node. Therefore, these two pods are running in one or two of the nodes but not all three as only two are required.

8.2 Active-Active-Active – Single K8s Cluster with very low latency network

Recommended Topology

8.2.1 Use Case

This scenario is the recommended topology for deploying API Connect v10. There is a single Kubernetes cluster spread over three or more worker nodes. Each worker pool can be a site, data hall, or group of nodes. There must be a very Low Latency Connection between the worker nodes. The latency between sites must be less than 6ms.

8.2.2 Topology

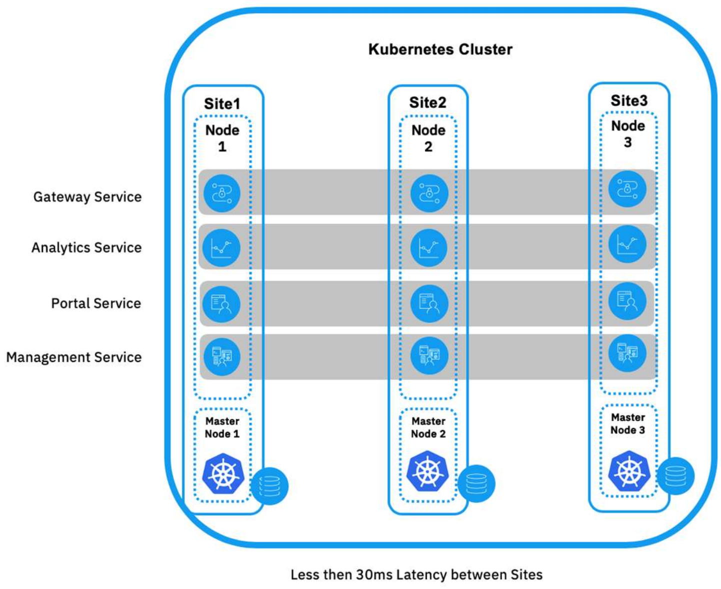


Figure 14: Topology for Active Active Active solution

8.2.3 Availability

The tables below describe the worst-case scenario for availability of each component.

8.2.3.1 API Manager, Analytics, Portal, and Kubernetes Master

	Node 1.1		Node 2.1		Node 3.1	Number of Nodes available	Cluster is available	Can Apply Maintenance
Normal Operations	Up		Up		Up	3	Yes	Yes
One Node Down	Up		Up		Down	2	Yes	No
Two Node Down	Offline*		Down		Down	0	No	No

Table 31 - Active-Active-Active API Manger, Analytics and Portal Availability Table

8.2.3.2 Gateway

	Node 1.1		Node 2.1		Node 3.1	Number of Nodes available	Cluster is available	Can Apply Maintenance
Normal Operations	Up		Up		Up	3	Yes	Yes
One Node Down	Up		Up		Down	2	Yes	No
Two Node Down	No Quorum		Down		Down	1	Yes	No

Table 32 - Active-Active-Active Gateway

8.2.4 Advantages of Active-Active-Active

- Can cope with a single site outage

8.2.5 Concerns of Active-Active-Active

- A low latency network connection must be available between all worker nodes.
- Can cope with a site outage or a node outage, but not a scenario where there is a site outage and a node outage at the same time.
- Unable to apply maintenance during an outage.

8.3 Active-Active (Hot Standby for API Manager and Portal)

NOTE: This is due to be delivered in a future fix pack of V10

This scenario has the Gateway and Analytics in Active-Active and the API Manager and Portal as Active–Hot Standby. This is similar to API Connect v5.

8.3.1 Use Case

The following statements describe the scenario for this pattern.

- The customer wants an Active-Active across two sites where the latency is less than 80ms.
- Unable or unwilling to provide a third site.

8.3.2 Topology

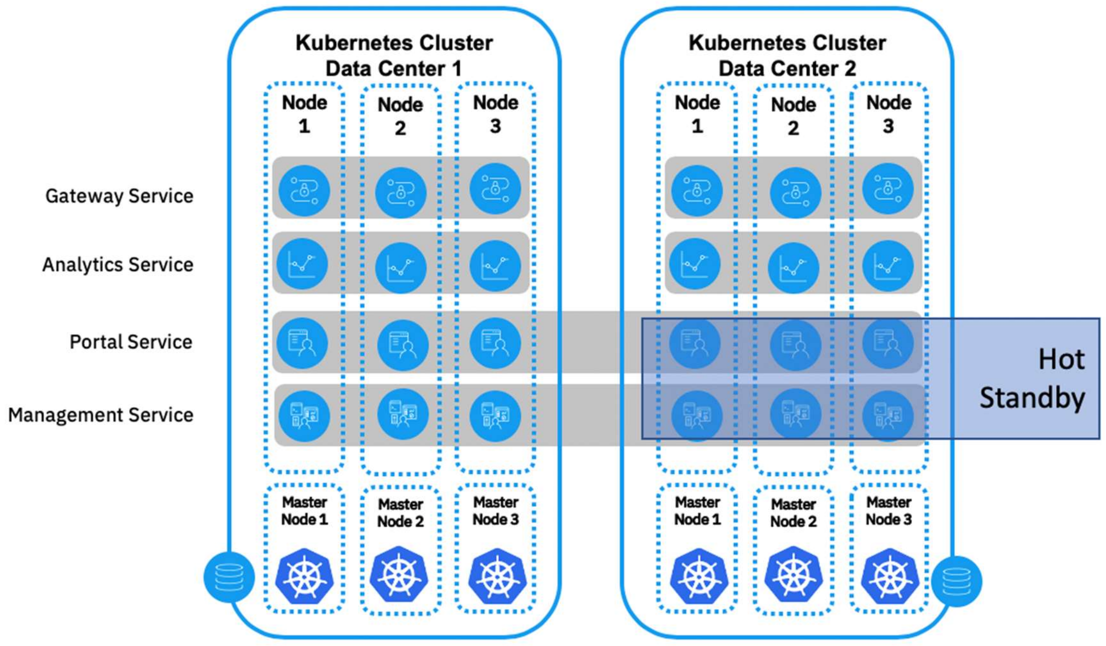


Figure 15: 8.3 HA Pattern

The topology described here requires a separate Kubernetes cluster in each site. The Kubernetes cluster is ***not*** expected to cross between the sites. This is because Kubernetes requires at least three master nodes to form a Quorum. If Kubernetes loses two master nodes it is no longer able to administer or manage the cluster. This means that maintenance cannot be applied, and components cannot cleanly be shut down, scaled up, or started. If the Master nodes cannot be recovered there is no clear way to rescue the cluster. By using a Kubernetes cluster on each site, if one site goes down there is no impact to the managing and running of the Kubernetes on the second site.

No API Connect component can span multiple Kubernetes clusters. All components are deployed to both Kubernetes clusters.

The Gateway and the Analytics services work independently and are able to provide Active-Active availability. This requires that APIs are published to both Gateways by the API Manager. If Site 1 goes down the other site is still serving the API requests. Analytics data is preserved for the site that did not go down.

The API Manager and Portal service are deployed in an Active-Hot Standby pattern similar to API Connect v5. At deploy time, the API Manager and Portal service in Site 1 are configured to replicate their database to their counterpart in Site 2.

- If Site 2 goes down there is no impact to the API Manager and Portal.
- If Site 1 goes down, then manual intervention must enable the API Manager and Portal components.
- If there is a disconnect between sites, then only Site 1 processes data.

8.3.3 Availability

8.3.3.1 Management and Portal

The Management node is used to manage, maintain, and create APIs. The portal is used to assist external developers in consuming APIs. If there is a complete outage of the portal, the runtime (API Gateway) is not impacted. However, new Developers and Applications are not be able to be registered.

A site outage occurs when two or more nodes in the same site are unavailable. During a site outage a decision to invoke a Disaster Recovery might be taken. This enables the components on the second site to become operational.

	Node 1.1	**Node 1.2**	**Node 1.3**		**Node 2.1**	**Node 2.2**	**Node 2.3**	**No of Available Nodes**	**Can Apply Maintenance in Site,**	
									S1	S2
Normal Operations	Up	Up	Up		Up Replication Only	Up Replication Only	Up Replication Only	3	Yes	Yes
One Node Down	Down	Up	Up		Up Replication Only	Up Replication Only	Up Replication Only	2	No	Yes
Two Node Down	Down	Down	Offline		Up Replication Only	Up Replication Only	Up Replication Only	0	n/a	n/a
DR (After manual intervention)	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	No
DR (After manual intervention). with Node Down	Down or Offline	Down or Offline	Down or Offline		Down	Up	Up	2	n/a	No
DR (After manual intervention) with Two Nodes Down	Down or Offline	Down or Offline	Down or Offline		Down	Down	Offline	0	n/a	n/a

Table 33 Active Hot Standby Availability for Management and Portal

8.3.3.2 Analytics and Kubernetes Master

Analytics is not critical for runtime operations. However, if the Analytics Service associated with a Gateway is not available then no analytics data is stored. Each Analytics Component would be registered with the API Gateway Component in the same site.

The Kubernetes Master is required for managing, creating, and scaling deployments. If the Master becomes unavailable, then the upgrades cannot be applied.

In the event of a complete site outage the Analytics Service in the second site would continue. Note: If you have more than one Analytics Service the portal is unable to show analytics data.

	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site	
									S1	S2
Normal Operations	Up	Up	Up		Up	Up	Up	6	Yes	Yes
One Node Down	Down	Up	Up		Up	Up	Up	5	No	Yes
Two Node Down	Down	Down	Offline		Up	Up	Up	3	n/a	Yes
On Node Down in Each Site	Down	Up	Up		Down	Up	Up	4	No	No
Two Node Down in one site and one node down in the other site	Down	Down	Offline		Down	Up	Up	2	n/a	No
Site 1 outage	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes

Table 34 Active-Active Availability for Analytics and Kubernetes Master

8.3.3.3 Gateway

If the Gateway service has an outage, then the APIs are not accepting requests. In the proposed solution there is a gateway in each site that is associated to the Analytics component in the same site. In the event of a Gateway or site outage then the Gateway on the second site should continue working unaffected. If a node is listed as No Quorum it means that it can accept requests but is not part of a quorum.

Note: Each Gateway service is independent and does not synchronize with the other Gateway service.

Gateway	**Node 1.1**	**Node 1.2**	**Node 1.3**		**Node 2.1**	**Node 2.2**	**Node 2.3**	**No of Available Nodes**	**Can Apply Maintenance in Site**	
									S1	**S2**
Normal Operations	Up	Up	Up		Up	Up	Up	6	Yes	Yes
One Node Down	Down	Up	Up		Up	Up	Up	5	Yes	Yes
Two Node Down	Down	Down	No Quorum		Up	Up	Up	4	No	Yes
On Node Down in Each Site	Down	Up	Up		Down	Up	Up	4	Yes	Yes
Two Node Down in one site and one node down in the other site	Down	Down	No Quorum		Down	Up	Up	3	No	Yes
Site 1 outage	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes

Table 35 Active-Active Availability for Gateway

8.3.4 Advantages of Active-Active (Hot Standby for API Manager and Portal)

- Does not require three sites.
- Gateway and Analytics services can serve traffic with no loss of service during a site outage.
- In the event of an outage on Site 1 the recovery process requires a manual operation that takes minutes to complete.

8.3.5 Concerns of Active-Active (Hot Standby for API Manager and Portal)

8.3.5.1 Maintenance

- During the maintenance process both sites are unavailable so that the database schema is updated in both locations at the same time.

8.3.5.2 No syncing between Gateway Clusters

- When the gateways are created the OAuth Secret must be the same. This is to allow both Gateway Services to enforce all OAuth Tokens.
- Token Revocation is not synced between the gateways.
- Rate Limit quotas are not synced between the gateways.
  - Each Cluster has its own Quota enforcements.

8.3.5.3 Disassociation

In the event that a network outage occurs between Site 1 and Site 2, then the following happens

- Artifacts including APIs, Applications, and Products can be updated only on Site 1, Site 2 continues serving the previous version.
- Analytics data from Site 2 is not accessible as the Management Node is not able to reach the Analytics Component on Site 2.

8.4 Active-Active (Cold Standby for API Manager and Portal)

The Active-Active high latency scenario is adopted when the client has two data centers with 80ms or greater latency and they want to be able to process API Traffic in either data center.

8.4.1 Use Case

The following statements describe the scenario for this pattern.

- The customer wants an Active-Active across two sites where the latency is greater than 100ms.
- They are unable or unwilling to provide a third site.
- The customer is taking regular back-ups of the API Manager and Portal
- These backups are replicated to the second site.

8.4.2 Topology

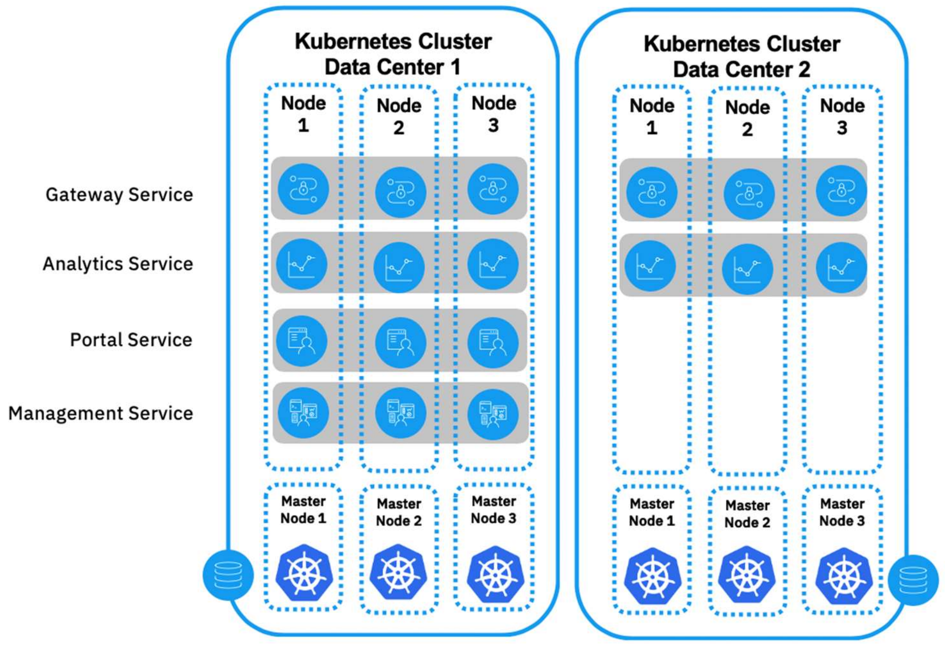


Figure 16: 8.4 HA Pattern

The topology described here requires a separate Kubernetes cluster in each site. The Kubernetes cluster is not expected to cross between the sites. This is because Kubernetes requires at least three master nodes to form a Quorum. If Kubernetes loses two master nodes it is no longer able to administer or manage the cluster. This means that maintenance cannot be applied, components cannot cleanly be shut down, scaled up, or started. If the Master nodes cannot be recovered there is no clear way to rescue the cluster. By using a Kubernetes cluster on each site, if one site goes down there is no impact to the managing and running of the Kubernetes on the second site.

No API Connect component can span multiple Kubernetes clusters. However, all components can be deployed to the same Kubernetes cluster. In order to achieve Active-Active, we need to deploy components to both data centers.

The Gateway and the Analytics work in an Active-Active pattern (See diagram above). If Site 1 goes down the other site is still serving the API Requests. Analytics Data is preserved for the site that has not gone down.

The API Management and Portal are deployed in an Active-Cold Standby Pattern. If Site 2 goes down there is no impact to the Management and Portal. If Site 1 goes down, then the API Management and Portal components are unavailable. A single API Management instance does not support running across multiple API Management Components.

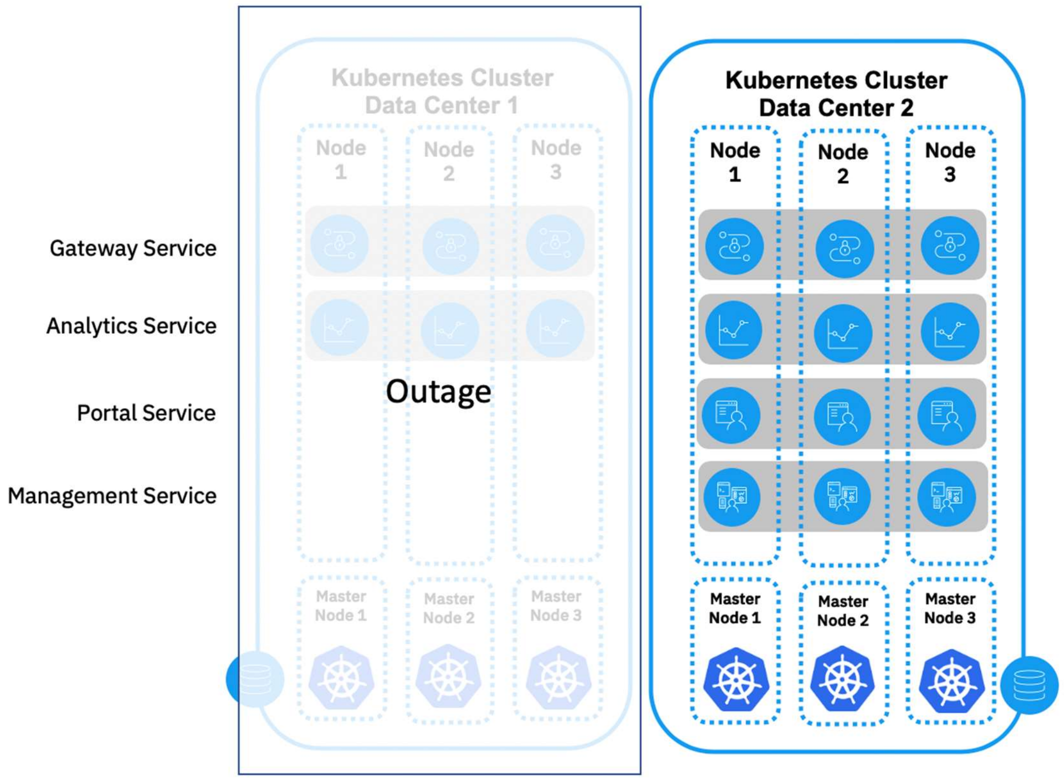


Figure 17: 8.4 HA Pattern - Failover

If a decision to trigger DR is made, then the Management and Portal pods are redeployed in Site 2. It is recommended that this process is not automated so that in the event of a disassociation there is no risk of multiple Mangers and Portals working independently. When the deployment has completed, the back-up artifacts of the Portal and Management are applied to the new deployment in Site 2.

8.4.3 Availability

8.4.3.1 Management and Portal

The Management node is used to manage, maintain, and create APIs. The portal is used to assist external developers in consuming APIs. If there is a complete outage of the portal the runtime (API Gateway) is not impacted. However, new Developers and Applications are not able to be registered.

During a site outage a decision to invoke a Disaster Recovery might be taken. If this happens then the API Manager is deployed into the second site and the most recent back up is applied.

	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site,	
									S1	S2
Normal Operations	Up	Up	Up		Offline	Offline	Offline	3	Yes	n/a
One Node Down	Down	Up	Up		Offline	Offline	Offline	2	No	n/a
Two Node Down	Down	Down	Offline		Offline	Offline	Offline	0	n/a	n/a
DR	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes
DR with Node Down	Down or Offline	Down or Offline	Down or Offline		Down	Up	Up	2	n/a	No
DR with Two Nodes Down	Down or Offline	Down or Offline	Down or Offline		Down	Down	Offline	0	n/a	n/a

Table 36 Active - Cold Standby - Availability for Management and Portal

8.4.3.2 Analytics and Kubernetes Master

The analytics is not critical for runtime operations. However, if the Analytics Service associated to a Gateway is not available then no analytics data is stored. Each Analytics Components would be registered with the API Gateway Component in the same site.

The Kubernetes Master is required for managing, creating, and scaling deployments. If the Master becomes unavailable, then the upgrades cannot be applied.

In the event of a complete site outage the Analytics Service in the second site would continue.

	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site	
									S1	**S2**
Normal Operations	Up	Up	Up		Up	Up	Up	6	Yes	Yes
One Node Down	Down	Up	Up		Up	Up	Up	5	No	Yes
Two Node Down	Down	Down	Offline		Up	Up	Up	3	n/a	Yes
On Node Down in Each Site	Down	Up	Up		Down	Up	Up	4	No	No
Two Node Down in one site and one node down in the other site	Down	Down	Offline		Down	Up	Up	2	n/a	No
Site 1 outage	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes

Table 37 Active-Active Availability for Analytics and Kubernetes Master

8.4.3.3 *Gateway*

If the Gateway has an outage, then the APIs are not accepting requests. In the proposed solution there is a gateway in each site that is associated to the Analytics component in the same site. In the event of a Gateway or site outage then the Gateway on the second site should continue working unaffected. If a node is listed as No Quorum it means that it can accept requests but is not part of a quorum.

Gateway	**Node 1.1**	**Node 1.2**	**Node 1.3**		**Node 2.1**	**Node 2.2**	**Node 2.3**	**No of Available Nodes**	**Can Apply Maintenance in Site**	
									S1	**S2**
Normal Operations	Up	Up	Up		Up	Up	Up	6	Yes	Yes
One Node Down	Down	Up	Up		Up	Up	Up	5	Yes	Yes
Two Node Down	Down	Down	No Quorum		Up	Up	Up	4	No	Yes
On Node Down in Each Site	Down	Up	Up		Down	Up	Up	4	Yes	Yes
Two Node Down in one site and one node down in the other site	Down	Down	No Quorum		Down	Up	Up	3	No	Yes
Site 1 outage	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes

Table 38 Active-Active Availability for Gateway

8.4.4 Advantages of Active-Active (Cold Standby for API Manager and Portal)

- Supports Double Jeopardy, for example, able to apply maintenance even during a site outage.
- Does not require three sites.
- Gateway and Analytics work in Active-Active.

8.4.5 Concerns of Active-Active (Cold Standby for API Manager and Portal)

8.4.5.1 Syncing between Gateway Clusters

- When the gateways are created the OAuth Secret must be the same. This is to allow both DataPowers to enforce all OAuth Tokens.
- Token Revocation is not synced between the gateways.
- Quota Enforcement is not synced between the gateways.
    - Each Cluster has its own Quota enforcements

8.4.5.2 Disassociation

In the event that a network outage occurs between Site 1 and Site 2 then the following will happen

- No Business Change; for example, APIs cannot be published or removed
    - If APIs are published, they will only go to Site 1 not Site 2.
- Analytics data from Site 2 are not accessible as the Management Node will not be able to reach the Analytics Component on Site 2.

8.5 Active-Active with DataPower not in Kubernetes (Cold Standby for API Manager and Portal)

This Scenario builds on 8.4. By removing DataPower from Kubernetes, we can run a single DataPower service across two sites.

Note: DataPower requires the Application Optimization License for the Analytics Component to successfully register.

It is possible to mix physical DataPower and virtual DataPower, whether Virtual Appliance or RPM install, in this scenario.

8.5.1 Use Case

The following statements describe the scenario for this pattern.

- The customer wants an Active-Active deployment in two sites.
- They are unable or unwilling to provide a third site.
- The customer is taking regular back-ups of the API Manager and Portal.

- These backups are replicated to the second site.
- A low latency network connection between the Data Centers is required.

8.5.2 Topology

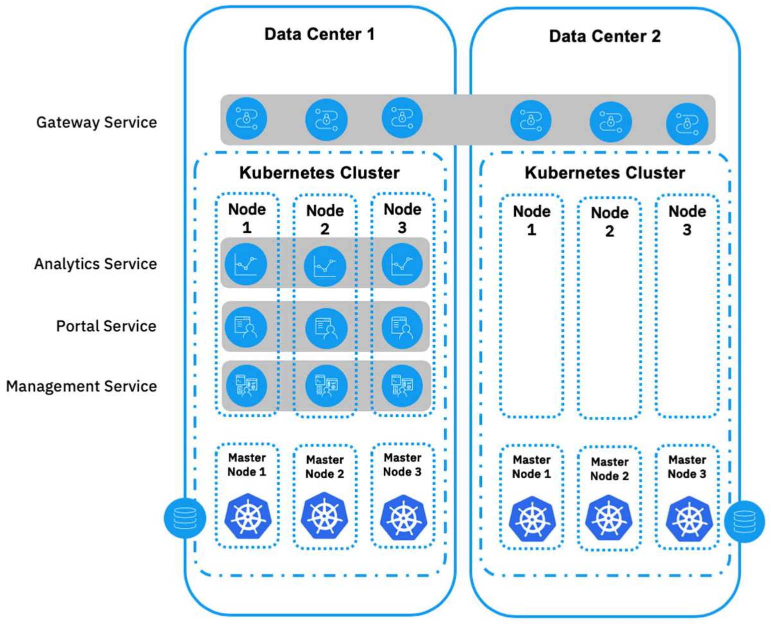


Figure 18: 8.5 HA Pattern

The Manager, Analytics, and Portal follow the same principles as 8.4. The DataPower Service crosses the Data Centers and can sync revocation tokens and quota enforcement.

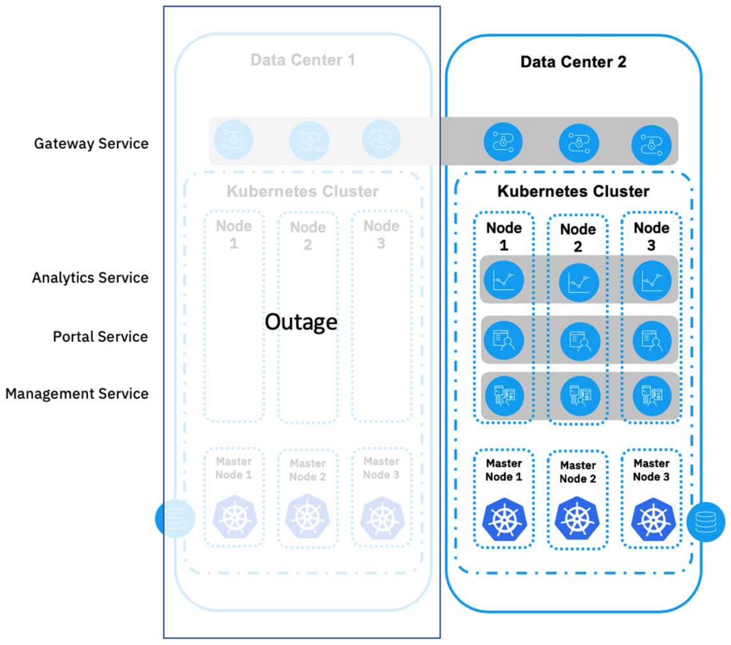


Figure 19 8.5 Pattern - Failover

8.5.3 Availability

8.5.3.1 Analytics Management and Portal

See 8.4.3.1

8.5.3.2 Kubernetes Master

See 8.4.3.2

8.5.3.3 Gateway

If the Gateway has an outage, then the APIs are not accepting requests. In the proposed solution there is a gateway in each site that is associated to the Analytics component in the same site. In the event of a Gateway or site outage then the Gateway on the second site should continue working unaffected. If a node is listed as No Quorum it means that it can accept requests but is not part of a quorum.

In the event of a quorum being down it can be restarted by manually selecting which node should be the master in the DataPower configuration. This should never be done to both sites at the same time.

Gateway	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site	
									S1	S2
Normal Operations	Up	Up	Up		Up	Up	Up	6	Yes	Yes
One Node Down	Down	Up	Up		Up	Up	Up	5	Yes	Yes
Two Node Down	Down	Down	Up		Up	Up	Up	4	Yes	Yes
On Node Down in Each Site	Down	Up	Up		Down	Up	Up	4	Yes	Yes
Two Node Down in one site and one node down in the other site	Down	Down	No Quorum		Down	No Quorum	No Quorum	3	n/a	n/a
Site 1 outage	Down	Down	Down		No Quorum	No Quorum	No Quorum	3	n/a	n/a
Site 1 outage after manual intervention	Down	Down	Down		Up	Up	Up	3	n/a	Yes
No Connectivity between sites	No Quorum	No Quorum	No Quorum		No Quorum	No Quorum	No Quorum	6	n/a	n/a
No Connectivity between sites after manual intervention	No Quorum	No Quorum	No Quorum		Up	Up	Up	6	n/a	Yes

Table 39 Active-Active with Gateway not in Kubernetes Availability

8.5.4 Advantages

- Supports Double Jeopardy, for example, able to apply maintenance even during a site outage.
- Does not require three sites.
- Gateway works in Active-Active.
- Gateway can share tokens and quota enforcement between sites while there is a quorum.

8.5.5 Concerns

- Gateway cannot run in a Kubernetes cluster.

8.6 Active/Passive (DR)

Active Passive is a common topology. The idea is that a single site serves the traffic. However, if an outage occurs a manual action causes all traffic to be routed to the second site. The challenge here is that the data must be updated regularly to Site 2 otherwise if an outage occurs then all data since the last backup would be lost.

This solution is expensive as it requires a separate instance to be available, but not serving traffic.

8.6.1 Use Case

- The customer is content to have a single Active site only at any one time.
- The customer is taking regular back-ups of all components.
- The customer is transferring the back up to the second site at regular intervals

8.6.2 Topology

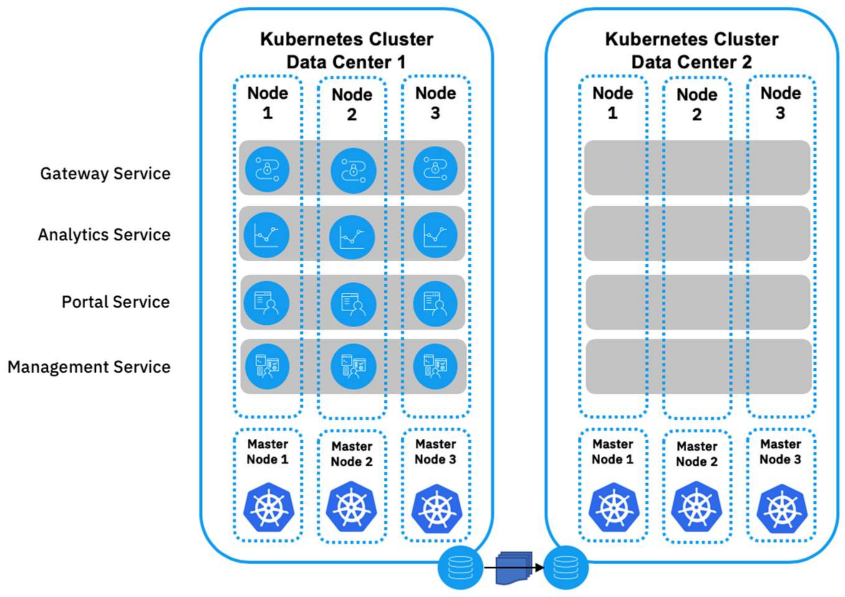


Figure 20: Active Passive HA Pattern

8.6.3 Availability

8.6.3.1 Management, Analytics, Portal and Kubernetes Master

	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site,	
									S1	S2
Normal Operations	Up	Up	Up		Offline	Offline	Offline	3	Yes	n/a
One Node Down	Down	Up	Up		Offline	Offline	Offline	2	No	n/a
Two Node Down	Down	Down	Offline		Offline	Offline	Offline	0	n/a	n/a
DR	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes
DR with Node Down	Down or Offline	Down or Offline	Down or Offline		Down	Up	Up	2	n/a	No
DR with Two Nodes Down	Down or Offline	Down or Offline	Down or Offline		Down	Down	Offline	0	n/a	n/a

Table 40 Active Passive Availability for Management, Analytics, Portal and Kubernetes Master

8.6.3.2 Gateway

	Node 1.1	Node 1.2	Node 1.3		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site,	
									S1	S2
Normal Operations	Up	Up	Up		Offline	Offline	Offline	3	Yes	n/a
One Node Down	Down	Up	Up		Offline	Offline	Offline	2	No	n/a
Two Node Down	Down	Down	No Quorum		Offline	Offline	Offline	1	No	n/a
DR	Down or Offline	Down or Offline	Down or Offline		Up	Up	Up	3	n/a	Yes
DR with Node Down	Down or Offline	Down or Offline	Down or Offline		Down	Up	Up	2	n/a	No
DR with Two Nodes Down	Down or Offline	Down or Offline	Down or Offline		Down	Down	No Quorum	1	n/a	No

Table 41 Active Passive Availability for Gateway

8.6.4 Advantages

- Data does not have to be synced between the sites in real time.
- Simple for the operations team to maintain.
- No risk of disassociation.

8.6.5 Concerns

- In the event of a site outage all data is lost that was written since the last backup was transferred to Site 2.
    - Regular backups mitigate this concern.
- Outage occurs during Site 2 activation.
    - Set Site 2 to be a hot standby to mitigate this concern.
- Quota Enforcement and Revoked Tokens are not backed up.
- Expensive, as a second set of infrastructures is required but is not actively used.
    - If the Second Site is a cold standby there might be no license costs to consider.
    - If the Second site is a hot standby there might be license costs to consider.

8.7 Anti-Patterns

Anti-Patterns are solutions that have a significant risk.

These are NOT recommended.

8.7.1 Active / Active - Single Kubernetes Cluster

Though this pattern provides three instances of each API Connect Components, it has a significant flaw because two of the Kubernetes Masters Nodes are in the same datacenter. If DC1 goes down, then the Kubernetes Master is unable to function because there is only one instance. When the Kubernetes Master is unavailable the infrastructure cannot be scaled, managed, or maintained. In addition, if DC1 cannot be recovered there is no way to recover the Kubernetes master as quorum cannot be reformed. This results in a full DR scenario requiring mirrored backups and redeployment of new Kubernetes sites.

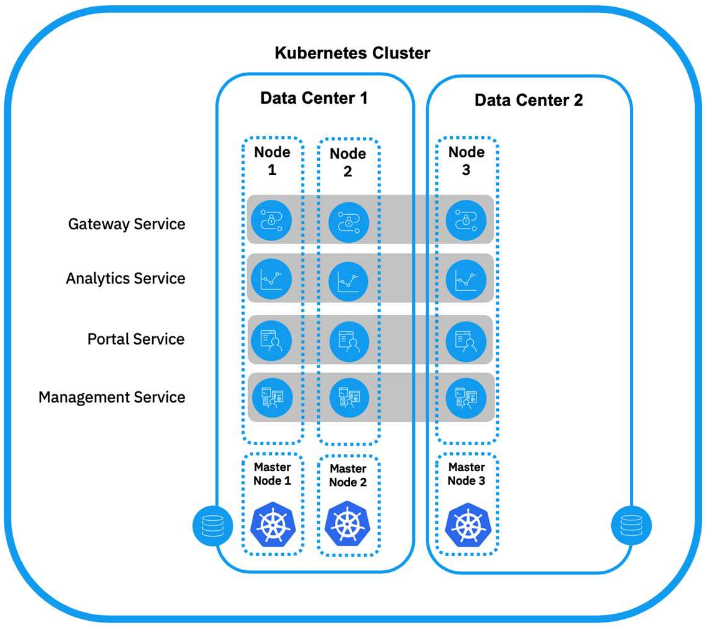


Figure 21: Anti-Pattern Active-Active

8.7.1.1 Availability

8.7.1.1.1 Management, Portal, Analytics and Kubernetes Master

	Node 1.1	Node 1.2		Node 2.1	No of Available Nodes	Can Apply Maintenance
Normal Operations	Up	Up		Up	3	Yes
One Node Down	Down	Up		Up	2	No
Site 1 Outage*	Down	Down		Offline	0	No
Site 2 Outage	Up	Up		Down	2	No

Table 42 Anti-Pattern Availability for Management, Portal, Analytics and Kubernetes Master

** in the event of a Site 1 outage the Kubernetes Master Cluster cannot be used until Site 1 has been recovered.*

In the event of a Site 1 outage, the Kubernetes Master Cluster cannot be used until Site 1 is recovered. If Site 1 is not recoverable the Kubernetes Master Cluster cannot be recovered. When the Kubernetes Master Cluster is not available Kubernetes is unable to scale, create, manage, or update any applications in the cluster.

8.7.1.1.2 Gateway

	Node 1.1	Node 1.2		Node 2.1	No of Available Nodes	Can Apply Maintenance
Normal Operations	Up	Up		Up	3	Yes
One Node Down	Down	Up		Up	2	No
Site 1 Outage	Down	Down		No Quorum	0	No
Site 2 Outage	Up	Up		Down	2	No

Table 43 Anti-Pattern Availability of Gateway

8.7.2 V5 Deployment Pattern

This pattern is similar to deployments recommended for API Connect V5.

8.7.2.1 Topology

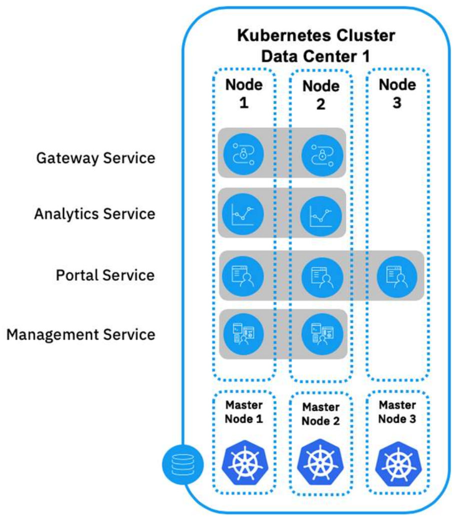


Figure 22: Anti-Pattern - V5 Deployment Pattern

8.7.2.2 Concerns

- Manager and Analytics both have two single points of failure. If either of the pods go down this component becomes unavailable as there is no quorum.
- When a single Analytics Component is lost no new analytics data can be stored.
- When a single API Management Component is lost APIs cannot be published, created, or retired. Applications, Subscriptions, and Consumer Organizations cannot be created or managed in the Portal or the API Manager.

8.7.2.3 Mitigation

If this pattern is deployed with 50% additional capacity, the risk is mitigated. With the additional capacity available, then when a node is lost Kubernetes automatically begins to deploy pods to nodes with available capacity. This causes a short outage, measured in minutes, but the system recovers with no interaction.

8.7.3 Active / Active - Two Independent clusters across two Clusters

This pattern allows for two distinct API Connect Platforms with their own, Management, Gateway, Portal, and Analytics. The platforms are kept in sync by way of scripts or human interaction.

8.7.3.1 Topology

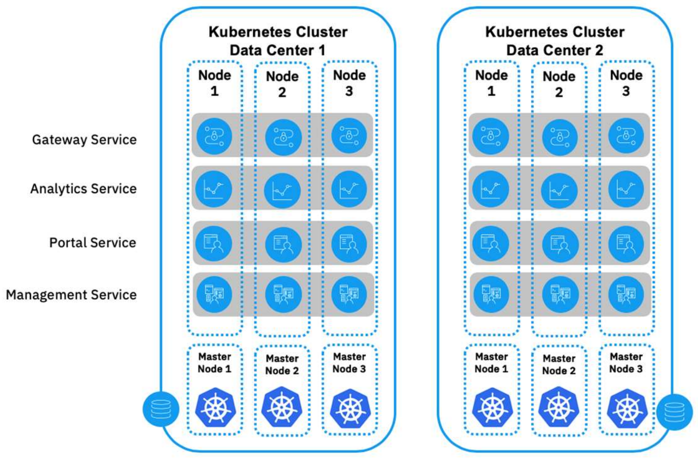


Figure 23: Anti-Pattern - Two Independent Clusters

8.7.3.2 Concerns

- No single point of truth. For example, if the management servers get out of sync, which one is correct?
- Analytics cannot be viewed in a single dashboard unless they are off loaded.
- Portals cannot be used for user registration without significant work.
    - Even with significant work the user experience would be impacted.
- Gateways can have the same APIs published at different times and so they become out of sync.
- Applications need to be created by script to ensure they are in sync between the API Managers.
- Manager is a single point of failure. If one manager goes down, no business change can be allowed to happen in either platform as the published APIs get out of sync.
- Prone to human error.

8.7.4 Active/Active/Active – Single K8s Cluster with a non-very low latency network

This is similar to pattern 1 (Active-Active-Active – Single K8s Cluster with very low latency network). However, if the latency is over 6ms or more, connections are routinely and unexpectedly dropped.

8.7.4.1 Topology

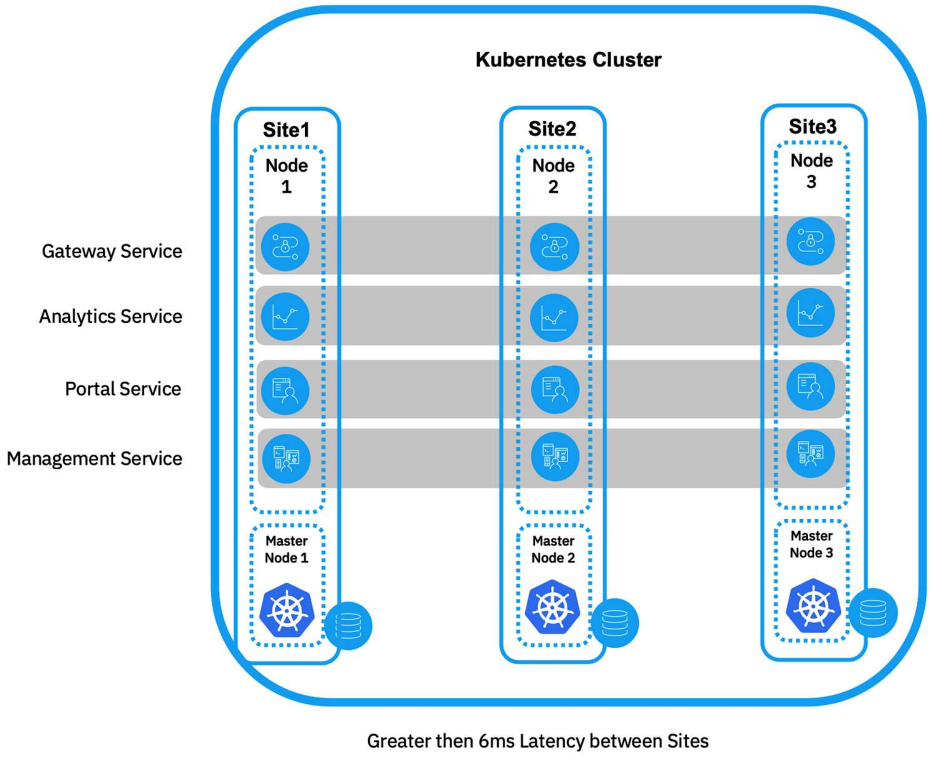


Figure 24 Antipattern - 8.7 – high latency three data centers

8.7.4.2 Concerns

- As the latency increases, the risk of a believed outage increases. If a node is unable to respond in a timely manner, then it is considered offline and the solution is no longer Highly Available.
- If all nodes are unable to communicate in a timely manner, then the quorum is considered lost and the solution becomes unavailable.

9 Kubernetes Deployment Considerations

This section covers the different Kubernetes Deployment options and known implications.

The table below shows recommendations for each of the Kubernetes environments. If an entry does not exist in this table, please raise a PMR on APIC for an up to date answer.

Recommendation	**Description**
Supported	IBM will take PMRs on solutions using this platform
Supported with a Known Limitation	IBM will take PMRs on solutions using this platform, however there are known limitations
Not Supported	IBM will not take PMRs on solutions using this platform

Table 44 Key for Kubernetes Deployment Targets

This table does NOT cover enterprise readiness of the Deployment target.

Deployment	***Recommendation***
Kubernetes	Supported
RedHat OpenShift	Supported
AWS - EKS	Supported
AWS - Bare Metal	Supported
Azure – AKS	Supported
Azure – Bare Metal	Supported
GCP	Supported
IBM Cloud – IKS	Known Limitation Note: A community Ingress must be deployed to IKS in order for analytics to work.
IBM Cloud – Bare Metal	Supported
*VMWare OVA**	Supported
Other	Please Contact IBM

Table 45 -Deployment Targets

**Though the OVA deployment of API Connect is not strictly a Kubernetes environment it does include Kubernetes inside of each component.*

10 Kubernetes Storage Considerations

10.1 Summary Table

The table below shows recommendations for each of the major storage types. If an entry does not exist in this table, please raise a PMR on APIC for an up to date answer.

Recommendation	**Description**
Supported	IBM will take PMRs on solutions using this platform
Supported with a Known Limitation	IBM will take PMRs on solutions using this platform, however there are known limitations
Not Supported	IBM will not take PMRs on solutions using this platform

Table 46 Key for Kubernetes Storage

Storage Class	**All Components**
Ceph RBD	Supported
Ceph FS	Supported
Gluster FS	Not Supported
NFS	Not Supported
Host Path	Supported with a Known Limitation
OpenShift Container Storage	Supported
IBM Spectrum Scale with Block storage	Supported
IBM Block Storage	Supported
AWS Elastic Block Storage	Supported
Azure Premium Storage	Supported
Azure Storage	Supported with a Known Limitation
GCE Persistent Disk	Supported
Others	Please contact IBM

Table 47 Kubernetes Storage Type Summary

10.2 Details for Supported with a known limitation entries

This section provides additional details to the none supported lines in Table 47 Kubernetes Storage Type Summary

10.2.1 Gluster FS

- Status: Not Supported

Gluster FS is designed to keep data sources in sync where large files are used. When using Gluster FS we have had reported problems with inconsistencies between the replicas.

10.2.2 NFS

- Status: Not Supported

NFS is not supported at this time

10.2.3 Host Path

- Status: Known Limitation in Production

The API Connect v10 OVA releases use Host Path storage. Host Path storage is where the pods write their persistent storage to the local OS disk. This has one critical weakness: if a node becomes unavailable, then the persistent storage is also lost for those pods.

10.2.3.1 Azure Storage

- Status: Reported Limitation but Not Tested

Azure storage has a slower write and read speed than premium storage. Customers have reported that the Developer Portal, Manager, and Analytics have consistency issues as well as other intermittent problems. The recommendation is to use Azure Premium Storage over this.

11 Contributors

Thanks to the following people for making the updates to this document possible

- Michael O'Sullivan
- Dan Whitacre
- John Bellessa
- Dave Whiteley
- Kai Zhang
- Chris Dudley
- Alex Butcher
- Elizabeth Bowling
- Jacqueline Clarke
- Cesar Cavazos