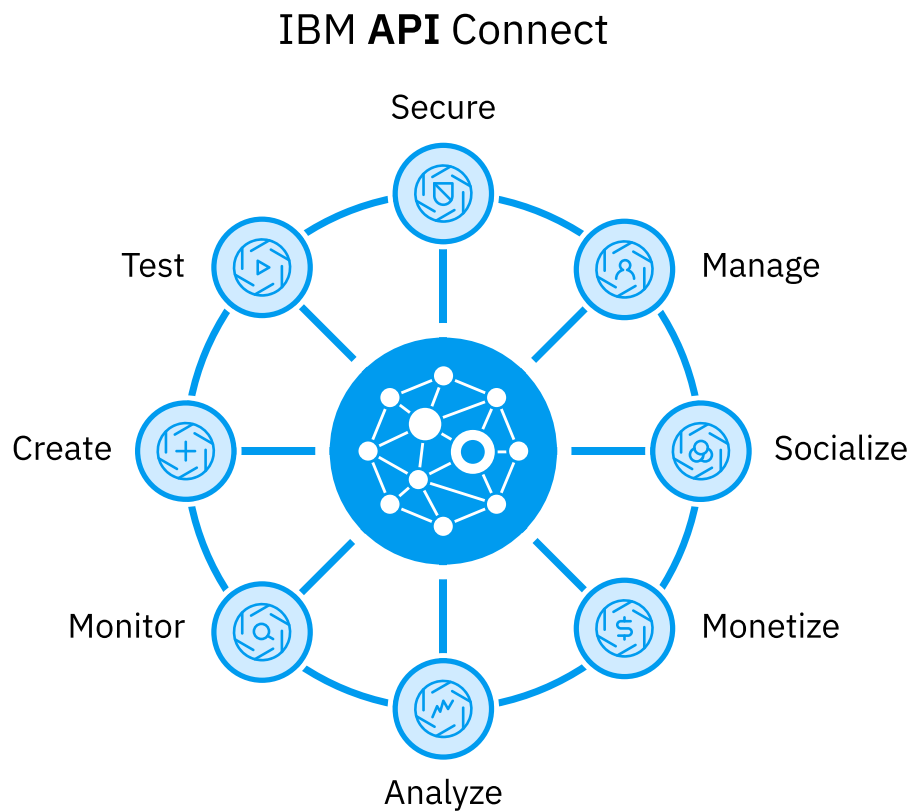


API Connect 2018.4.1.x WhitePaper

Authors: Chris Phillips
Aiden Gallagher



Version: 1.0.9

Date: 11/08/2019

1 Table of Contents

1	TABLE OF CONTENTS	2
1.1	TABLE OF FIGURES.....	3
1.2	TABLE OF TABLES	4
1.3	VERSION HISTORY	6
2	INTRODUCTION TO THE WHITEPAPER.....	7
2.1	TERMINOLOGY.....	8
2.2	PERSONAS	10
3	API CONNECT COMPONENTS	12
3.1	COMPONENTS - INTRODUCTION	13
3.2	MANAGEMENT SERVICE	14
3.3	GATEWAY SERVICE	19
3.4	ANALYTICS SERVICE	20
3.5	DEVELOPER PORTAL SERVICE	23
4	DEPLOYMENT OPTIONS	25
4.1	INTRODUCTION	26
4.2	KUBERNETES	26
4.3	APPLIANCE (OVA)	26
5	TERMS AND NAMING CONVENTION	27
5.1	INTRODUCTION	28
5.2	API DEVELOPMENT	29
5.3	API TOPOLOGY.....	30
5.4	CONSUMER COMPONENTS.....	41
5.5	DEVELOPER PORTAL	42
6	ISOLATION AND ENVIRONMENT SEGREGATION.....	43
6.1	API PROVIDER.....	43
6.2	API CONSUMERS	45
6.3	DATAPOWER GATEWAYS.....	45
6.4	DEVELOPER PORTAL	46

7	API CONNECT HIGH AVAILABILITY	48
7.1	INTRODUCTION.....	48
7.2	HIGH AVAILABILITY WITH A CONTAINER/ KUBERNETES DEPLOYMENT	49
7.3	HIGH AVAILABILITY WITH OVA DEPLOYMENTS.....	51
7.4	QUORUMS.....	52
7.5	AVAILABILITY TABLE.....	55
8	MULTI DATA CENTER DEPLOYMENT PATTERNS	59
8.1	INTRODUCTION.....	59
8.2	ACTIVE-ACTIVE-ACTIVE – SINGLE K8S CLUSTER WITH LOW LATENCY NETWORK.....	61
8.3	ACTIVE-ACTIVE.....	62
8.4	ACTIVE-ACTIVE WITH DATAPOWER NOT IN KUBERNETES.....	68
8.5	ACTIVE/PASSIVE (DR).....	73
8.6	ANTI-PATTERNS.....	76
9	KUBERNETES DEPLOYMENT CONSIDERATIONS	82
10	KUBERNETES STORAGE CONSIDERATIONS	84
10.1	SUMMARY TABLE.....	84
10.2	ON PREMISE.....	85
10.3	CLOUD NATIVE.....	87
11	CONTRIBUTORS	89

1.1 Table of Figures

Figure 1:	API Connect Components.....	13
Figure 2:	High Availability Install of API Manager	15
Figure 3:	High Availability Install of Analytics Service.....	20
Figure 4:	High Availability Install of Portal	23
Figure 5:	Example API Endpoints	29
Figure 6:	Provider Organization (pOrg) Depiction.....	31
Figure 7:	Catalog Depiction.....	34
Figure 8:	Spaces Depiction	36
Figure 9:	API Connect Clouds Depiction	44
Figure 10:	Highly Available Deployment	51
Figure 11:	Example of Cassandra Pods in a standard deployment.....	51
Figure 12:	Failure Tolerance formula	52
Figure 13:	Proof for failure tolerance	53
Figure 14:	Topology for Active Active Active solution	61
Figure 15:	Active Active HA Pattern	63
Figure 16:	Active Active HA Pattern - Failover.....	64
Figure 17:	Active Active HA Pattern	69
Figure 18:	Active Active HA Pattern - Failover.....	70
Figure 19:	Active Passive HA Pattern	74
Figure 20:	Anti-Pattern Active Active.....	77

Figure 21: Anti-Pattern - V5 Deployment Pattern	79
Figure 22: Anti-Pattern - Two Independent Clusters.....	80

1.2 Table of Tables

Table 1 Key Terminology Definitions	9
Table 2 Personas	10
Table 3 API Management Microservice Pods	17
Table 4 API Management Jobs.....	18
Table 5 Analytics Microservice Pods	22
Table 6 Portal Microservice Pods	24
Table 7 Example Product Names	30
Table 8 Example Plan Names	30
Table 9 Example Provider Organization names	33
Table 10 Example Catalog names	35
Table 11 Example Space names	36
Table 12 Example Gateway Service Names.....	37
Table 13 Example Gateway Service Extension Names	37
Table 14 Example User Registry names.....	38
Table 15 Example Role names.....	38
Table 16 Example TLS Profile names.....	39
Table 17 Example API TLS Profile names.....	39
Table 18 Example Availability Zones names.....	40
Table 19 Example Portal, Analytics and Gateway Service names	40
Table 20 Example OAuth Provider Names	41
Table 21 Example Developer Organization Names	41
Table 22 Example Application names.....	42
Table 23 Logical Segregation Examples	44
Table 24 Deep Dive Links for DataStores	49
Table 25 The table above shows for a certain number of nodes how many outages can occur before a Quorum cannot be reached.	53
Table 26 Active Active Availability for Management and Portal	65
Table 27 Active Active Availability for Analytics and Kubernetes Master	66
Table 28 Active Active Availability for Gateway.....	67
Table 29 Active Active with Gateway not in Kubernetes Availability	72
Table 30 Active Passive Availability for Management, Analytics, Portal and Kubernetes Master	75
Table 31 Active Passive Availability for Gateway	75
Table 32 Anti-Pattern Availability for Management, Portal, Analytics and Kubernetes Master	77
Table 33 Anti-Pattern Availability of Gateway	78
Table 34 Kubernetes Storage Type Summary.....	85

1.3 Version History

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

VERSION	CHANGE	DATE
1.0.9	<ul style="list-style-type: none"> • Fixed wording in Section 8 • Corrected description of DataPower internals 	August 2019
1.0.8	<ul style="list-style-type: none"> • Fixed the wording around no of APIs per Product • Fixed the working around no of spaces per catalog • Correct EBS storage information • Highlighted the requirements of Application Optimization module needed for DataPower. 	July 2019
1.0.7	<ul style="list-style-type: none"> • Added Document Version History (This Table) • Language and formatting changes. • Updated Analytics pods 2018.4.1.4 • Updated definition of low latency. • Updated section 8.4.3.3 to include information about manually setting a DataPower device as primary. 	May 2019
1.0.5	<ul style="list-style-type: none"> • Language and formatting changes 	Feb 2018
1.0.3	<ul style="list-style-type: none"> • First Published Version 	-

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 from turning backend RESTFUL or SOAP 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 \(https://www.ibm.com/cloud/api-connect\)](https://www.ibm.com/cloud/api-connect) or [Knowledge Center \(https://www.ibm.com/support/knowledgecenter/en/SSMNED_2018/mapfiles/getting_started.html\)](https://www.ibm.com/support/knowledgecenter/en/SSMNED_2018/mapfiles/getting_started.html). The below sections 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

<i>Term</i>	<i>Definition</i>
<i>Target Service</i>	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 or RESTFUL services
<i>API</i>	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
<i>Cluster</i>	A group of nodes running together
<i>Node</i>	An instance within a cluster
<i>Worker Node</i>	VM or a physical machine that have some form of compute resources to run the pods for a given service
<i>Data Center</i>	A data center is a collection of machines in the same geographic area
<i>Availability Zone</i>	An availability zone is one or more data centers connected via a low latency connection. A Low Latency Connection is required.
<i>Catalog</i>	A catalog is a staging target and behaves as a logical partition of the gateway and the Developer Portal
<i>Service</i>	The name given to each deployed instance of the API Connect subsystems: API Management Service, API Gateway Service, Analytics Service and Developer Portal Service
<i>API Connect Cloud</i>	An API Management service with one or more Portal, Analytics and Gateway Services
<i>High Availability</i>	The ability of the solution to continue successful operation without manual intervention in the event of failure of a subset of the system components
<i>Disaster Recovery</i>	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
<i>Hot Standby</i>	A deployment configuration in which a set of servers are actively running ready to instantaneously take over in the event of a failure, but not actively serving traffic until that failover
<i>Cold Standby</i>	A deployment configuration where the failover servers are in a stopped state until required at the point a failover. Restoration of normal operation using Cold Standby takes longer than Hot Standby
<i>GUID</i>	Globally Unique Identifier
<i>Shard</i>	A database can have it's data split into shards. Analytics has two types of shards Primary and Replica. Primary shards.

<i>Low Latency Connection</i>	This article defines a Low Latency Connection to be one of less than 30ms as well as less than 1% connection are unexpectedly dropped due to infrastructure issues.
-------------------------------	---

Table 1 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:







<i>Persona</i>	<i>Description</i>
<i>Shavon, the API Developer</i>	 <p>Is in charge of development of the APIs for her organization. Her focus is understanding data, services or information needed from an API and then building out that API based on those requirements. Eventually publishing it to testing, staging and production environments.</p>
<i>Jason, the API Lifecycle Manager</i>	 <p>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.</p>
<i>Steve, the Provider Organization Owner</i>	 <p>Is in charge of coordinating the delivery of APIs across multiple lines of business and development groups. Making sure each group has the resources needed and are publishing their APIs to the appropriate environments.</p>
<i>Will, the Cloud Manager</i>	 <p>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.</p>
<i>Marsha, the Community Manager</i>	 <p>Is in charge of Application developer success for her organization. She focuses on making the API portal as straight forward as possible and monitors/ manages the API consumer groups.</p>
<i>Andre, the Application Developer</i>	 <p>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.</p>

Table 2 Personas

These personas may not have a one to one fit with any organization's structure or internal roles. Additionally, individual people at an organization may 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 Components

3.1 Components - Introduction

There are four different Components in API Connect 2018.4.1.x:

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

In this section we list the pods and containers deployed as of 2018.4.1, however as the product evolves the pods and containers deployed may change.

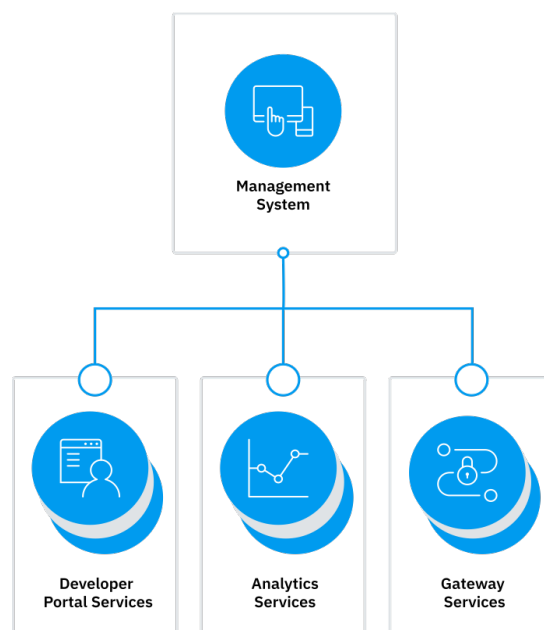


Figure 1: API Connect Components

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 only accessed by Infrastructure or Operation teams.
- 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 contains 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 nine different microservices in the API Manager, shown below is a pod level view of a high availability or standard deployment. It is noted that there are multiple instances or replicas (as called by Kubernetes), of the microservices.

NAME	READY	STATUS	RESTART	AGE
<Management-Identifier>-cassandra-operator-8fcc8c794-t4f29	1/1	Running	0	3d
<Management-Identifier>-a7s-proxy-59cb9f94c9-kwtgn	1/1	Running	0	3d
<Management-Identifier>-a7s-proxy-59cb9f94c9-vnbsr	1/1	Running	0	3d
<Management-Identifier>-a7s-proxy-59cb9f94c9-z9scv	1/1	Running	0	3d
<Management-Identifier>-apiconnect-cc-0	1/1	Running	0	3d
<Management-Identifier>-apiconnect-cc-1	1/1	Running	0	3d
<Management-Identifier>-apiconnect-cc-2	1/1	Running	0	3d
<Management-Identifier>-apim-v2-5678658cb6-5n5kg	1/1	Running	0	3d
<Management-Identifier>-apim-v2-5678658cb6-jzq4c	1/1	Running	0	3d
<Management-Identifier>-apim-v2-5678658cb6-s772n	1/1	Running	0	3d
<Management-Identifier>-client-dl-srv-7df946bd55-dqnm8	1/1	Running	0	3d
<Management-Identifier>-client-dl-srv-7df946bd55-q7lsh	1/1	Running	0	3d
<Management-Identifier>-juhu-7cff8b8dd-rpdkq	1/1	Running	0	3d
<Management-Identifier>-juhu-7cff8b8dd-vvddg	1/1	Running	0	3d
<Management-Identifier>-juhu-7cff8b8dd-vzm22	1/1	Running	0	3d
<Management-Identifier>-ldap-7758ff95dd-2pqcj	1/1	Running	0	3d
<Management-Identifier>-ldap-7758ff95dd-hln5n	1/1	Running	0	3d
<Management-Identifier>-ldap-7758ff95dd-znd7g	1/1	Running	0	3d
<Management-Identifier>-lur-v2-598dc474c-djpbv	1/1	Running	0	3d
<Management-Identifier>-lur-v2-598dc474c-t86kl	1/1	Running	0	3d
<Management-Identifier>-lur-v2-598dc474c-wxbrg	1/1	Running	0	3d
<Management-Identifier>-ui-55d77c6db-6tfb4	1/1	Running	0	3d
<Management-Identifier>-ui-55d77c6db-wttqf	1/1	Running	0	3d

Figure 2: High Availability Install of API Manager

Pod or Container	Description	Default No. of Replicas
<i>apim</i>	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 will have three pods, deployed across the worker nodes. This number of pods can be scaled up based on load.	3
<i>ui</i>	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 will have 2 pods, across different worker nodes and this can be scaled up as necessary.	2

<i>Pod or Container</i>	<i>Description</i>	<i>Default No. of Replicas</i>
<i>cassandra-operator</i>	<p>The Cassandra Operator (COp) is a microservice that handles the individual pods that make up a Cassandra cluster. It uses a Kubernetes Custom Resource to define how the Cassandra cluster is configured. The operator is on a namespace basis, so it will only act upon a custom resource that is defined on the namespace the operator is deployed in. There is only a single Cassandra Operator pod.</p> <p>The operator has the following responsibilities:</p> <ul style="list-style-type: none"> • Deploy one or more Cassandra cluster either through a Statefulset or Daemonset • Scale up or down the number of pods in the Cassandra cluster • Perform ordered upgrade of the pods (to a new Cassandra docker image) • Perform automatic post mortems to of the database when errors occur. • Restore Cassandra database from specified backup • Perform repairs on the Cassandra cluster • Tear down/delete the Cassandra cluster • Perform automatic generation of Cassandra statistics of all the Cassandra pods in a cluster (cron-job) 	1
<i>apiconnect-cc</i>	<p>These Cassandra pods perform the reading and writing of data into storage. A standard (HA) deployment will have multiple pods, deployed across different worker nodes. This number of pods can be scaled up based on load.</p>	3
<i>a7s-proxy</i>	<p>This microservice 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, etc.). A standard (HA) deployment will have 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.</p>	3

<i>Pod or Container</i>	<i>Description</i>	<i>Default No. of Replicas</i>
<i>juhu</i>	Coordinates user authentication and token management for the API Management service. A standard (HA) deployment will have 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
<i>ldap</i>	The LDAP microservice is leveraged to configure a connection to a LDAP registry for user authentication. A standard (HA) deployment will have 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
<i>lur</i>	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 will have 3 pods, deployed across 3 different worker nodes. This number of pods can be scaled up based on load.	3
<i>client-dl-svr</i>	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 will have 2 pods, across different worker nodes and this can be scaled up as necessary	2

Table 3 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:

<i>Jobs</i>	<i>Description</i>
<i>apiconnect-cc-cassandra-stats</i>	This job gathers statistics about the database runtime environment to assist in debugging issues

<i>apiconnect-cc-repair</i>	This job is triggered to keep the data of the Cassandra replicas (cc-0, cc-1, cc-2, etc.) up to date. It goes through the replica and brings all the data up to the newest version, repairing any inconsistencies.
<i>apim-schema-init-job</i>	These jobs are used during install and upgrades. It is used for setting up the database schemas properly.
<i>lur-schema-init-job</i>	

Table 4 API Management Jobs

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 that is making the API call is permitted to access the API (i.e. has an active subscription to a product that contains the API operation)
- Enforcing the security constraints defined by the API such as a requirement to authenticate using a protocol like 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 or services that are defined in the API implementation, which may 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 original caller

In a deployment each Gateway has a pod called the **dynamic-gateway-service**, which runs as a single container. There are different services and ingresses for the gateway pod.

In earlier versions of API Connect, the API Manager communicated with each individual gateway member. Now the API Manager communicates with just the gateway ingress allowing for easy deployment to third party clouds. Configuration data is to a component running on the gateway called the **Gateway Director**. This component is in charge of managing Gateway configuration distribution and persistence across all gateway members in the cluster.

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 provides the following:

- Storing the API event logs as they are processed from the Gateway Service
- Processing API event logs from the gateway.
- Visualizing 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 the API event records to target locations (e.g. Splunk, Syslog, Kafka, HTTP, etc)

3.4.1 Pods in the Analytics Service

NAME	READY	STATUS	RESTARTS	AGE
<Analytics-Identifier>-analytics-client-9c464dfcc-bfkpt	1/1	Running	0	3d
<Analytics-Identifier>-analytics-client-9c464dfcc-bzdsx	1/1	Running	0	3d
<Analytics-Identifier>-analytics-ingestion-64cd44884f-5cgjh	1/1	Running	0	3d
<Analytics-Identifier>-analytics-ingestion-64cd44884f-jcmm4	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mtls-gw-77689b8b65-pg2qd	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mtls-gw-77689b8b65-w8k7s	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-coordinating-7dbc4bcc6b-m7t7w	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-coordinating-7dbc4bcc6b-pm7rw	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-coordinating-7dbc4bcc6b-zqmp	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-data-0	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-data-1	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-data-2	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-master-0	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-master-1	1/1	Running	0	3d
<Analytics-Identifier>-analytics-storage-master-2	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-zookeeper-0	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-zookeeper-1	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-zookeeper-2	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-kafka-0	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-kafka-1	1/1	Running	0	3d
<Analytics-Identifier>-analytics-mq-kafka-2	1/1	Running	0	3d
<Analytics-Identifier>-analytics-operator-8547bb559f-crt5m	1/1	Running	0	3d

Figure 3: High Availability Install of Analytics Service

Pod or Container	Description	Default No. of Replicas
<i>analytics-client</i>	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. It also, hosts the analytics	2

	webserver for RESTful APIs and rendering the analytics visualization in API Connect. All read access to the analytics data passes through here.	
<i>analytics-ingestion</i>	Analytics ingestion processes the data as it is passed from the gateway modifying, transforming and extending the data in the ingestion pipeline, preparing the data for storage or off-loading to a third-party system. This is based off logstash.	2
<i>analytics-mtls-gw</i>	The entry point for communication with the analytics components, API Manager, Gateway and Portal, serving as authentication mechanism for analytics communication.	2
<i>analytics-storage-coordinating</i>	The analytics-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 analytics-storage-data node. Same goes for writing, the request to write from the analytics-ingestion node needs to be routed to the correct analytics-storage-data node	3
<i>analytics-storage-data</i>	The analytics-storage-data nodes handle performing 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 (primary and replicas).	3
<i>analytics-storage-master</i>	The analytics-storage-master nodes handle managing the cluster and cluster state. Only one of these nodes is active at a time, the others are backup in case the main one goes down. This node will monitor the cluster, balancing it when needed, moving shards/data etc. This node will also manage 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
<i>analytics-mq-kafka</i>	The analytics message queue is an optional service that provides a reliable ingestion pipeline, especially when offloading analytics to a third	3

	<p>party system. The message queue ensures that the analytics pipeline will not be blocked when either analytics storage or the offload endpoint becomes unavailable. Data collection will continue internally in the analytics subsystem and will be buffered if analytics offload has been configured. With the message queue enabled, data storage will continue 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, mapping issues, etc.</p> <p><i>This pod only starts is if it is required by the installation.</i></p>	
<i>analytics-mq-zookeeper</i>	<p>A stateful set that manages the <i>analytics-mq-kafka</i> pods.</p> <p><i>This pod only starts is if it is required by the installation.</i></p>	3
<i>analytics-operator</i>	<p>The Analytics Operator adds serviceability commands to facilitate the managing the analytics service.</p>	1

Table 5 Analytics Microservice Pods

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 of 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 will be 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 is a highly customizable website, built on-top of Drupal. It provides customers plenty of flexibility to model the look and feel of the Developer Portal site.

3.5.1 Microservices in the Developer Portal Service

The portal 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, by the 2/2.

NAME	READY	STATUS	RESTART	AGE
<Portal-Identifier>-apic-portal-db-0	2/2	Running	0	3d
<Portal-Identifier>-apic-portal-db-1	2/2	Running	0	3d
<Portal-Identifier>-apic-portal-db-2	2/2	Running	0	3d
<Portal-Identifier>-apic-portal-nginx-7c6ddb644-4wjf4	1/1	Running	0	3d
<Portal-Identifier>-apic-portal-nginx-7c6ddb644-6grvx	1/1	Running	0	3d
<Portal-Identifier>-apic-portal-nginx-7c6ddb644-zzb42	1/1	Running	0	3d
<Portal-Identifier>-apic-portal-www-0	2/2	Running	0	3d
<Portal-Identifier>-apic-portal-www-1	2/2	Running	0	3d
<Portal-Identifier>-apic-portal-www-2	2/2	Running	0	3d

Figure 4: High Availability Install of Portal

Pod or Container	Description	Default No. of Replicas
<i>portal db</i>	This is the database pod which has 2 running containers.	3

	<ul style="list-style-type: none"> portal db-dbproxy container portal db-db container <p>These containers are explained below.</p>	
<i>portal db-dbproxy container</i>	Handles communication with portal db container from the portal www pod. This routes read/ write request.	-
<i>portal db-db container</i>	Hosts the Portal Database	-
<i>portal nginx</i>	This pod runs just a single container. It is just a simple proxy for communication.	3
<i>portal www</i>	This pod hosts the portal sites and contains the admin and web containers	3
<i>portal www-admin container</i>	Handles communication and integration with the API Manager.	-
<i>portal www-web container</i>	The webserver running the Drupal websites.	-

Table 6 Portal Microservice Pods

4 Deployment Options

4.1 Introduction

IBM API Connect can be deployed in two patterns.

- 1) Directly to a Kubernetes Environment
- 2) Appliance (OVA) install to an existing VMWare Estate

4.2 Kubernetes

API Connect can be installed to a Kubernetes environment that is running helm (including tiller), ingress and has a persistent storage solution. (See Section 9). The APICUP utility enables the installation by creating a configuration file that is then used with the deployment. For a detailed explanation of different types of Kubernetes Environments please 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 required for their component only. i.e. the Management Appliance contain the Management Containers but not the Analytics containers.

The APICUP installer is used to configure the appliances. For more information please review the Knowledge Center.

Please note the 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, though it can be adapted as needed.

IBM API Connect has a host of free text variable, parameter, profile and component names. Whilst 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 people will 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 will provide some generic naming conventions. It is recommended that each customer maintains their own list of naming conventions 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 and will often follow company preference.

Additionally, there are articles that describe API needs around versioning and standard formatting of naming, versioning and URL etc.

(<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 5: Example API Endpoints

5.2.2 Products

Products are a logical grouping of APIs and Plans. If API security is enabled APIs will be invocable once a Consumer Organization has subscribed to a plan. If no API security is enabled, then the APIs are invocable 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, 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 per Product. Products can contain many more APIs and Plans, but it creates a challenge to manage and govern.

Product names should include dashes and not underscores and should not contain numbers. Version numbers should not be defined in the Product name, 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 between brands these are appended in brackets at the end of the product.

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

Table 7 Example Product Names

5.2.3 Plans

Plans are used for rate limiting of API Calls and will be used for monetization of APIs. Developer Organization subscribe an Application to a Plan, not a Product. Plans have the additional functionality of burst limits which is designed to protect both the api infrastructure and downstream application which can then be configured per api within the same product. Each product can have multiple plans which will have different costs and potentially rate limits. Additionally, different plans may 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
<i>Basic</i>	<i>basic</i>	<i>Standard</i>	<i>standard</i>	<i>Advanced</i>	<i>advanced</i>
<i>Bronze</i>	<i>bronze</i>	<i>Silver</i>	<i>silver</i>	<i>Gold</i>	<i>gold</i>
<i>Small</i>	<i>small</i>	<i>Medium</i>	<i>medium</i>	<i>Large</i>	<i>large</i>

Table 8 Example Plan Names

5.3 API 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 three API Clouds, for the following environments:

- Development
- Test (also known as Pre-Production or Non-Production)
- Production

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 only exist 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 using the API Manager user interface. Users are able to belong to multiple pOrgs, with different roles and corresponding permission sets.

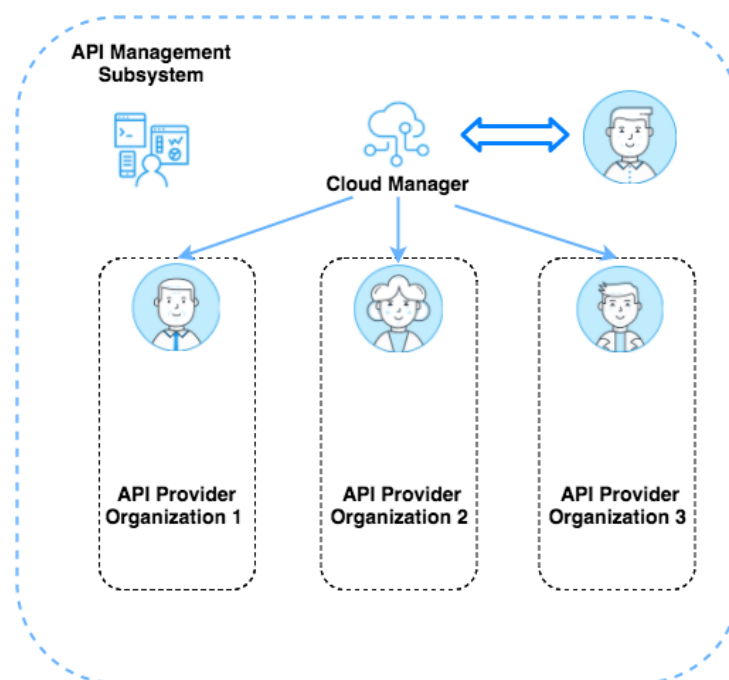


Figure 6: Provider Organization (pOrg) Depiction

A pOrg provides the following functionality:

- Publishing APIs and Products, as well as managing the lifecycle of those products
- Managing catalogs, which contain deployed APIs
- Managing API subscriptions and API consumers
- Managing additional resources available to that organization (User Registries, TLS profiles, etc.)

A poorly defined provider organization structure will have 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 will have 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 (dev, test, qa, etc.) Here are some strategies for large enterprise use cases for leveraging pOrgs to segregate environments:

1. pOrgs to separate release staging environments (dev, stage, qa, etc.)
2. pOrgs to separate development groups who 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 e.g. SIT01 or by an Organization's business unit e.g. UK Sales, UK HR.

An Organization's display name may be several words, each beginning with a capital letter. Its logical name must represent the display name, it must not contain spaces but instead 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 (Maximum length: 81 characters).

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

Display Name	Name
<i>OAT01</i>	<i>oat01</i>
<i>SIT01</i>	<i>sit01</i>
<i>SIT02</i>	<i>sit02</i>
<i>LUAT01</i>	<i>luat01</i>
<i>UK Sales</i>	<i>uk-sales</i>
<i>UK HR</i>	<i>uk-hr</i>

Table 9 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 follows the following scheme within API Connect:

```
https://<Gateway-Host>/<pOrg-name>/<Catalog>/basepath/path
```

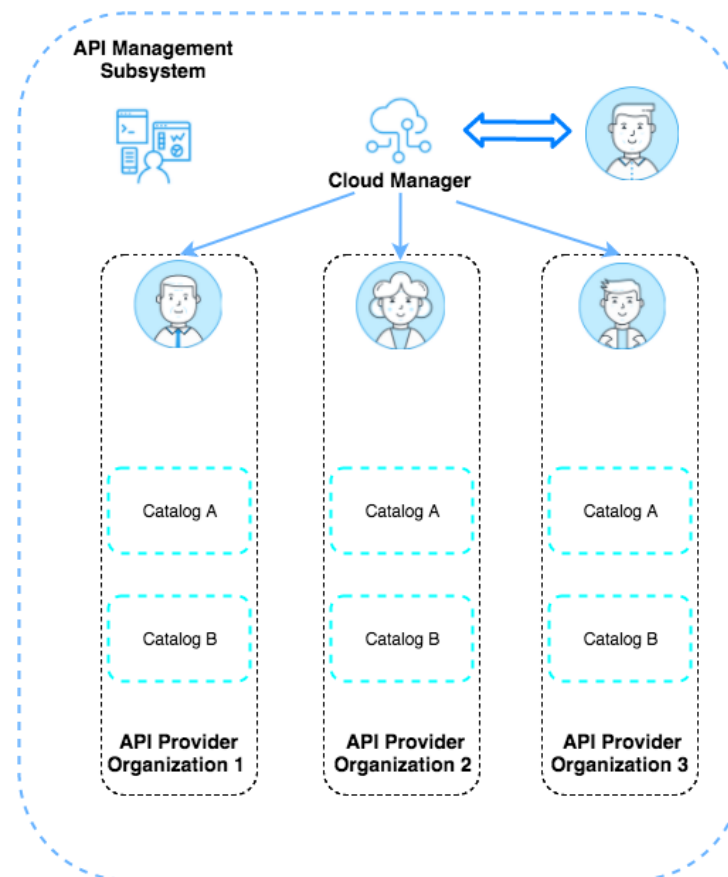


Figure 7: Catalog Depiction

Thus, not only do Catalogs impact API consumption, but also impact logical portioning 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 are 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 vs. 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 may be more than 1 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, which provides this isolation.

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

Display Name	Name
<i>Mortgages</i>	<i>mortgages</i>
<i>Open</i>	<i>open</i>
<i>Loans</i>	<i>loans</i>
<i>Current Accounts</i>	<i>current-accounts</i>

Table 10 Example Catalog names

5.3.4 Space

A Space provides a level of isolation below a Catalog. In API Connect this capability is often referred to as syndication. This is used to describe that the management and control of APIs can be given to individuals, groups or lines of business, but all the APIs end up centralized to a single portal site for API consumption. This is only visible in the API Manager; the portal 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 not recommended to have no more than seven Spaces per catalog. We have found customers with large numbers of space have challenging 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 only their Space in a Catalog)
- Map members to roles and permission sets
- Define custom resources for the Space (TLS profiles, OAuth providers etc.)

- Separated Analytics data (members scoped to Space, can only view 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.

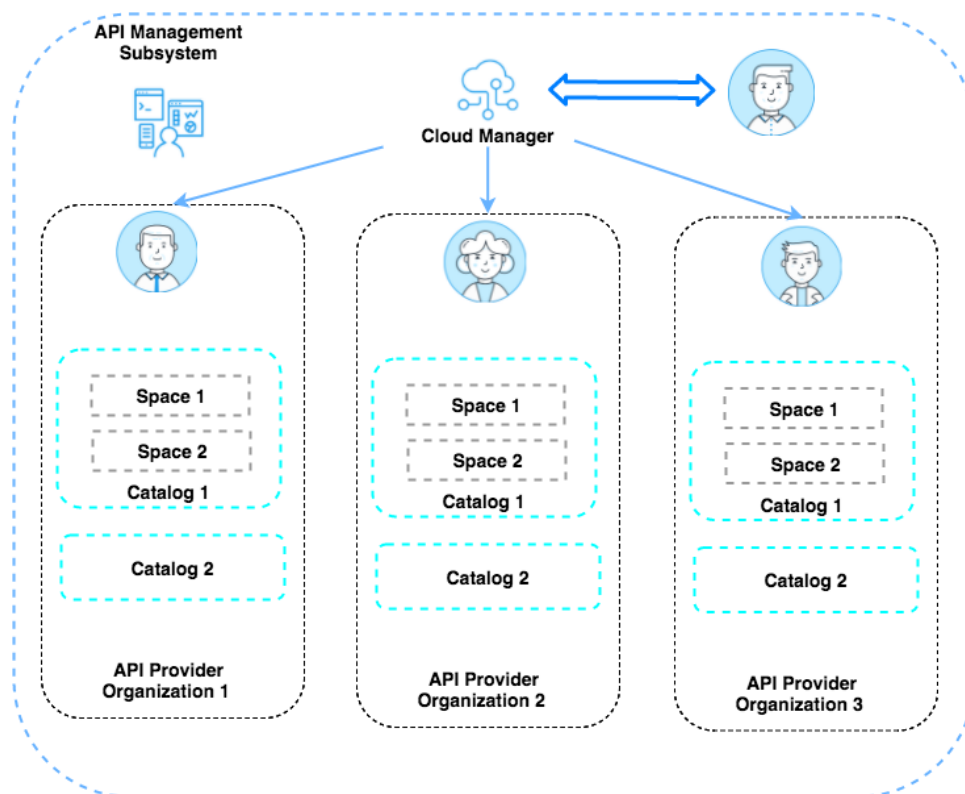


Figure 8: Spaces Depiction

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

Display Name	Name
<i>Balance</i>	<i>balance</i>
<i>Payments</i>	<i>payments</i>
<i>England Stores</i>	<i>england-stores</i>
<i>England Regulations</i>	<i>england-regulations</i>

Table 11 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 the 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 instance that are self managed. This allows them to share quota enforcement information, revocation tokens and new APIs.

FL

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 will be used by catalogs and spaces to define the Gateway Services they will use, so a descriptive name is important.

Title
<i>SIT01 Mortgages DP Service</i>
<i>OAT01 Current Account Payments DP Service</i>
<i>PROD01 Open APIs</i>

Table 12 Example Gateway Service Names

5.3.5.2 Gateway Service Extensions

Extensions are deployed to a single domain. They should include a version number which can be incremented as and when changes are made. Alternatively, or in conjunction with the version number, a date stamp may be appended to the end of the string to further describe when the extension was compiled. Names should be descriptive of the function being performed, should not contain spaces but instead dashes. The file should use full words and not abbreviations.

Filename
<i>api-certificate-updates-v1.0.zip</i>
<i>api-certificate-updates-20180101-0323.zip</i>
<i>api-certificate-updates-v1.0-20180101-0323.zip</i>

Table 13 Example Gateway Service Extension Names

5.3.6 User Registries

A User Registry should be easily identifiable to anyone looking at the system. Typically, it would describe what component it is for, where the registry is

based and then the fact 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.

Display Name	Name
<i>API Manager Local User Registry</i>	<i>api-manager-lur</i>
<i>Cloud Manager Docker User Registry</i>	<i>cloud-manager-dur</i>

Table 14 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 (to be represented equivalently in the name). These should describe the type of permission being given.

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

Display Name	Name
<i>Community Viewer</i>	<i>community-viewer</i>
<i>Role Administrator</i>	<i>role-administrator</i>

Table 15 Example Role names

5.3.8 TLS Profile

TLS profiles can be used for communication with the API Connect Cloud 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 Cloud Console

The name given for Cloud based TLS Profiles should have the following format; *component-tls-side-profile*, component relates to where the profile is used and side is whether it is a server or client profile. The Display Name/Title is the same but with spaces instead of dashes.

Display Name	Name
<i>Default TLS Server Profile</i>	<i>default-tls-server-profile</i>
<i>SNI TLS Client Profile</i>	<i>sni-tls-client-profile</i>

Table 16 Example TLS Profile names

5.3.8.2 APIs

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

Display Name	Name
<i>Authorise API TLS IHS</i>	<i>authorise-API-tls-ihs</i>

Table 17 Example API TLS Profile names

5.3.9 Availability Zones

Availability zones organize API Connect operations based upon 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 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 should be descriptive and are organized by regions, global separation, or for physical/logical separation of data centers. The Title can use spaces, capitals, lowercase etc. but a short one or two-word title is preferred. Abbreviation may be used

Title	Name
<i>US Availability Zone</i>	<i>us-availability-zone</i>

<i>Hampshire Availability Zone</i>	<i>hants-availability-zone</i>
<i>Newport Availability Zone</i>	<i>newport-availability-zone</i>
<i>LDN South Availability Zone</i>	<i>london-south-availability-zone</i>

Table 18 Example Availability Zones names

5.3.10 Portal, Analytics and Gateway Services

Services 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
<i>USPortal</i>	<i>us-portal</i>
<i>USGateway</i>	<i>us-gateway</i>
<i>HampshireAnalytics</i>	<i>hants-analytics</i>
<i>LDNSouthPortal</i>	<i>ldn-south-portal</i>

Table 19 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 please read here

https://www.ibm.com/support/knowledgecenter/en/SSMNE2_2018/com.ibm.apic.cmc.doc/capic_cmc_oauth_concepts.html

OAuth Title should match the OAuth Provider being used exactly.

Title	Name
<i>Zendesk</i>	<i>zendesk</i>
<i>Ubuntu One</i>	<i>Ubuntu-one</i>

Table 20 Example OAuth Provider Names

5.4 Consumer Components

5.4.1 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 only be in a consumer organization if they are sharing application credentials. I.e. More than ten people in a consumer organization is probably wrong.

Consumer Organizations can be created both on a Portal site and also on the management server. Typically, the Portal Site would be explored by external entities either external 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.

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

Developer Organization
<i>UKBank-HR-Compensation</i>
<i>UKBank-Mortgages-Test</i>
<i>UKBank-Mortgages-Development</i>
<i>UKBank-Mortgages-View</i>

Table 21 Example Developer Organization Names

5.4.2 Applications

Applications should be named with its goal and purpose in mind. The name of these applications should be concise, should all be lower case and obvious what the application does.

Title
<i>hr-employee-payment</i>
<i>hr-employee-illness</i>
<i>mortgage-test-ivt001</i>
<i>mortgage-test-ivt002</i>
<i>mortgage-test-ivt003</i>
<i>mortgage-test-endtoend</i>

Table 22 Example Application names

5.5 Developer Portal

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

5.5.1 Roles

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

5.5.2 Developer Portal Modules

Modules within the Developer Portal allow additional functionality with the Portal Site such as configuration of LDAP, live chat functions, OpenID Connect and many more.

Module names within the Developer Portal should follow Drupal Standards.

<https://www.drupal.org/docs/8/creating-custom-modules/naming-and-placing-your-drupal-8-module>

6 Isolation and Environment Segregation

6.1 API Provider

It is 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 will have complete separation and isolation, from a physical level infrastructure level, software level and data level. Users will belong to these environments separately and may 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.

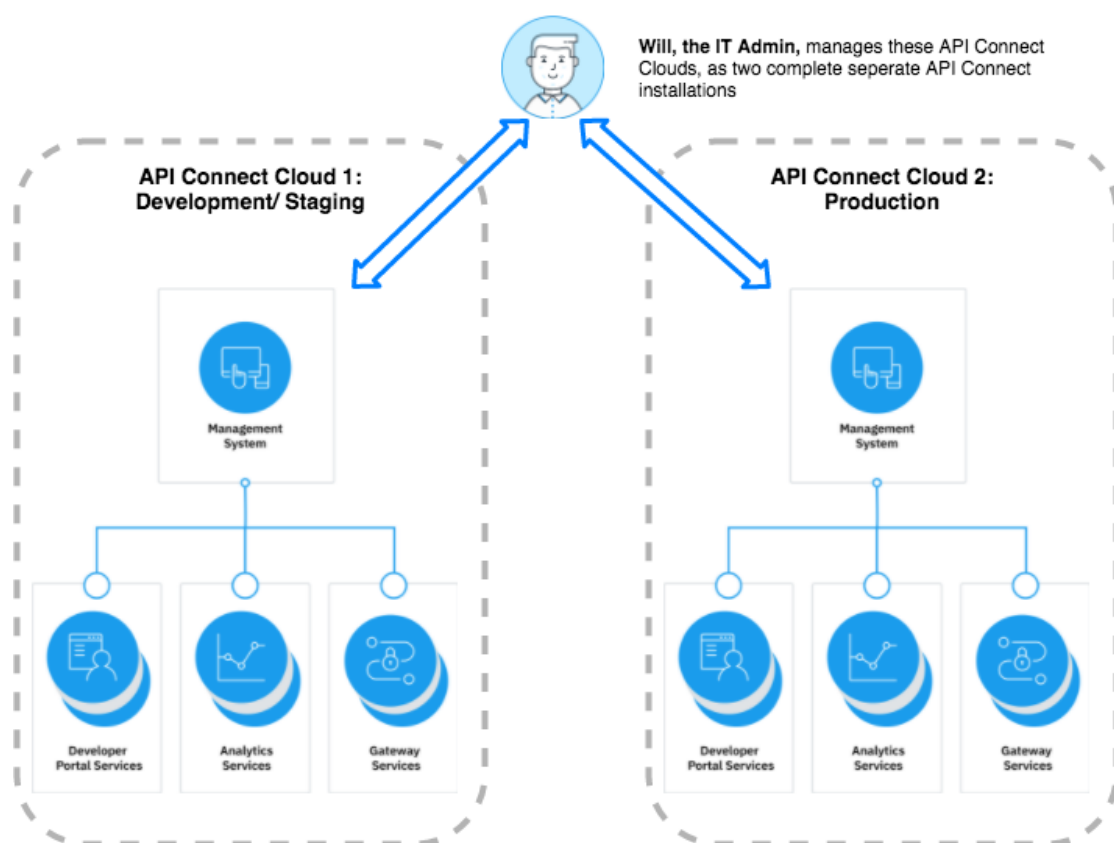


Figure 9: API Connect Clouds Depiction

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 23 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. In the event that 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

Often for API Projects, there are multiple API Consumer groups. These are mapped on to Consumer Organizations in API Connect.

The 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 only specific API Consumer Organizations. 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 per DataPower Instances

If DataPower is being deployed in Kubernetes, then it is recommended there is one DataPower Gateway Service per 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 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 a different location. For example, Gateways and Analytics components may be deployed to the IBM Cloud and the Portal and the Manager may be deployed on premise.

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

6.3.3 DataPowers with Catalogs

DataPower Gateways have many to many relationships with Catalogs. i.e.

- 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 2018 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 only associate 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. Once 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 isolate portals for different channels, e.g. internal, external, business partners that is ok.

6.4.1 Multi Cloud Topology

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

Components must not span multiple clouds.

7 API Connect High Availability

7.1 Introduction

High Availability is the ability to withstand the loss of one or more nodes without the service being interrupted. All of the API Connect services in 2018.4.x are designed for high availability.

The API Connect containers are enabled for High Availability by the use of 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 of the underlying component database systems rely on distributed database replication logic in order to make sure all of 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, which are listed below.

If you want to deep dive into each of these components there are links to additional documentation:

<i>API Manager</i>	Cassandra https://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlConfigConsistency.html
<i>Analytics</i>	Elastic Search https://www.elastic.co/guide/en/elasticsearch/reference/5.6/index.html Log Stash https://www.elastic.co/guide/en/logstash/6.5/index.html
<i>Portal</i>	Portal https://www.percona.com/doc/percona-xtradb-cluster/LATEST/features/multimaster-replication.html
<i>Kubernetes Master Nodes</i>	ETCD https://coreos.com/etcd/docs/latest/v2/admin_guide.html

Table 24 Deep Dive Links for DataStores

7.2 High Availability with a Container/ Kubernetes Deployment

In Kubernetes containers are deployed in Pods. Each Pod may contain one or more containers. If a Pod terminates unexpectedly then it is automatically restarted. I.e. 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 will be 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 please look at the Kubernetes website. <https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/>

When building a Kubernetes cluster for API Connect there are 2 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 please follow the links below.

- https://www.ibm.com/support/knowledgecenter/en/SSYGOH_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 will be multiple replicas of the database pods for each of the Services and by default, these replicas will be 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 and more can be read on this from 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 is required to provide High Availability, but a customer could use more
- A minimum of 3 master nodes is 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)

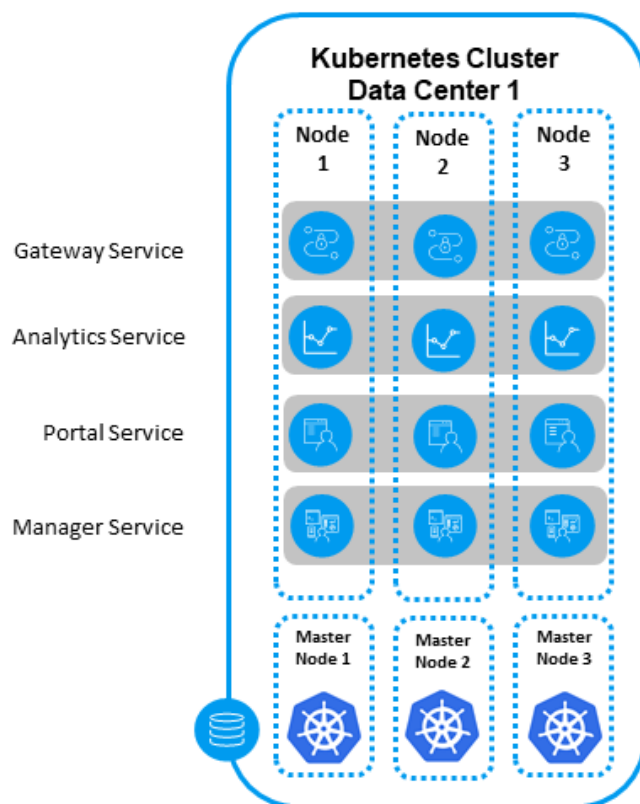


Figure 10: Highly Available Deployment

Referencing back to Figure 2: High Availability Install of API Manager, which shows the pods for a high availability API Management install, the following pods are observed:

NAME	READY	STATUS	RESTARTS	AGE
<version-identifier>-apiconnect-cc-0	1/1	Running	0	5d
<version-identifier>-apiconnect-cc-1	1/1	Running	0	5d
<version-identifier>-apiconnect-cc-2	1/1	Running	0	5d

Figure 11: Example of Cassandra Pods in a standard deployment

These pods are the Cassandra pods, forming the Statefulset that was defined as part of the installation process. ReplicaSets can be increased or decreased to scale based on load.

7.3 High Availability with OVA Deployments

Highly available OVA deployments for IBM API Connect 2018.4.1.x will follow similar recommendation to those described in 6.1 above. 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 is not true 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. Thus, if a connection is dropped or a VM fails or the machine that the VM was running on fails, etcd loses quorum and that VM stops serving requests. Thus, 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 presented above as underlying component technology. The next section will aim to simplify the quorum discussion.

In OVAs the Cassandra pods are in a Daemon Set instead of a stateful set. This allows for a pod to be deployed to each node that is added to the cluster automatically.

7.4 Quorums

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

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

Figure 12: Failure Tolerance formula

** Failure Tolerance is # of cluster members that can fail and the cluster still maintain regular function

**The number is rounded down when the number of members is even in a cluster

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 segment below shows the proof for failure tolerance for a give node.

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

Round down to 0

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

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

Round down to 1

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

And so, on and so forth....

Figure 13: Proof for failure tolerance

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

The other way to read the quorum equation from above, is from the perspective of individual cluster members. Individual cluster members are continuously syncing with their peers and will continue to operate as long as they can communicate with greater than 50% of members in the cluster (including them self). When a member stops being able to communicate with a quorum of cluster members, quorum is lost and that member will stop 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 Elastic Search 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 2018.4.x+

Therefore, a minimum of 3 cluster members is required for High Availability deployments of API Connect 2018.4.x+

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 2018.4.1.x components. This is a change from API Connect version 5.0.x and below.

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

The quorum requirements for these 3 components are fairly 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 will stop handling both write requests to their databases, rendering the services unavailable. The exception to this is the Analytics component, only the master nodes are impacted by losing quorum. If the data nodes loose quorum data is still read/write able.

Each of the systems can have the behavior fine-tuned, but this is only for more advanced users and to do this requires interacting with the databases directly. As an example, the default consistency levels for Cassandra in API Connect are "Quorum", however there are other consistency levels that could be utilized for a customer use case. For more information please read the following

<https://docs.datastax.com/en/cassandra/3.0/cassandra/dml/dmlConfigConsistency.html>

7.5 Availability Table

The table below shows the availability of the API Connect Gateway 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

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

7.5.1 Implications for API Gateway Service

The API Connect Gateway Service for IBM API Connect is IBM DataPower. IBM DataPower stores for 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 of these functions are where quorum is important to the Gateway Service. If quorum cannot be established, then the failover process will not work. This is important to prevent against split brain scenarios. A quorum Sentinel instances is required to handle failover, the original quorum equation from section 7.4 applies.

This means that a minimum of 3 gateway instances is 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 fail, quorum is lost . The following tasks will not take place.

- API Publications
- Onboarding cannot take place, i.e. 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. Once recovered the API Gateway Service will behave as if master is up and handle both configuration and application traffic normally.

If there are only two DataPower Devices this does not allow the API Connect Gateway Service to 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 (GW1 can't communicate with GW2)
- 3) Device failure (GW1 or GW2 fail)

The best option for establishing quorum for a two DataPower device configuration is the following:

- 1) GW1 with tenant or domain 1.a and 1.b
- 2) GW2 with tenant or domain 2.a

This protects against the failure scenarios 1, 2 and partially 3. This is because this setup can handle GW2 failure but can't handle GW1 failure. If GW1 were to fail, GW2 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.

7.5.1.1 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	Node4	Node5	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
4	Active	Active	Active	Active		4	Yes
4	Active	Active	Active	Down		3	Yes
4	Active	Active	Down	Down		2	No
4	Active	Down	Down	Down		1	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	Active	Active	Down	Down	Down	2	No
5	Active	Down	Down	Down	Down	1	No

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 may run. It is unlikely that there will be 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 2018 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 one and 2.x are in site two. 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 will contain 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
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, may be planned or unplanned.
No Quorum	(Gateway Only) The Node is not in Quorum but is processing API Traffic

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 are needing to apply maintenance to the nodes listed as Up and that node will need 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.

In each diagram in these sections we draw the four key components. Please note that each component signifies where pods for that component may run. These pods are deployed depending on the Affinity and Anti Affinity rules provided in Kubernetes. For more information on this please read <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 will be running in one or two of the components but not all three as only two are required.

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

8.2.1 Scenario

This scenario is the recommended topology for deploying API Connect 2018. There is a single Kubernetes cluster spread over three or more worker pools. Each worker pool can be a site, data hall or group of nodes. There must be a Low Latency Connection between the worker nodes.



8.2.2 Topology

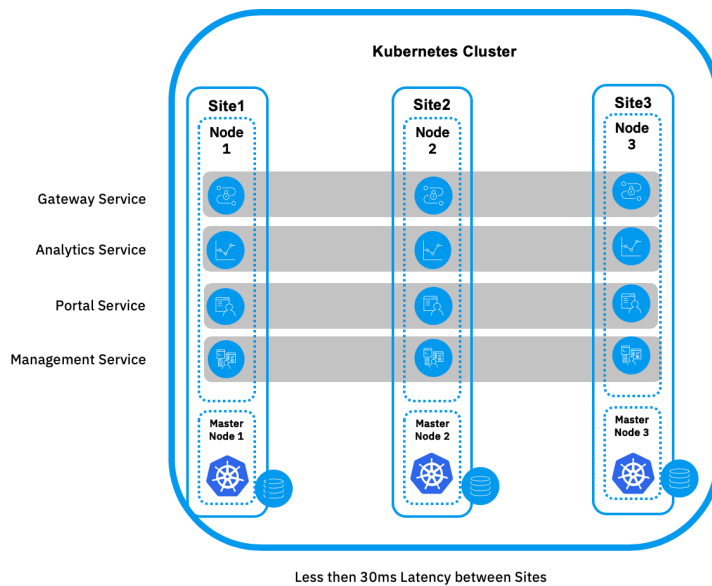


Figure 14: Topology for Active Active Active solution

8.2.3 Availability

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

8.2.3.1 Management, 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

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

8.2.4 Advantages

- Can cope with a single site outage

8.2.5 Concerns

- A low latency network connection must be available between each worker node.
- Can cope with a site outage or a node outage, but not both. I.e. Does not provide double jeopardy support.
 - Unable to apply maintenance during an outage

8.3 Active-Active

The Active-Active Scenario is used when the client has two data centers and they wish to be able to process API Traffic in either data center.

8.3.1 Scenario

The following statements describe the scenario for this pattern.

- The customer wants an Active - Active deployments in two sites.
- They are unable or unwilling to provide a third site.
- The customer is taking regular back-ups of the API Management and Portal and
- These backups are replicated to the second site.

8.3.2 Topology

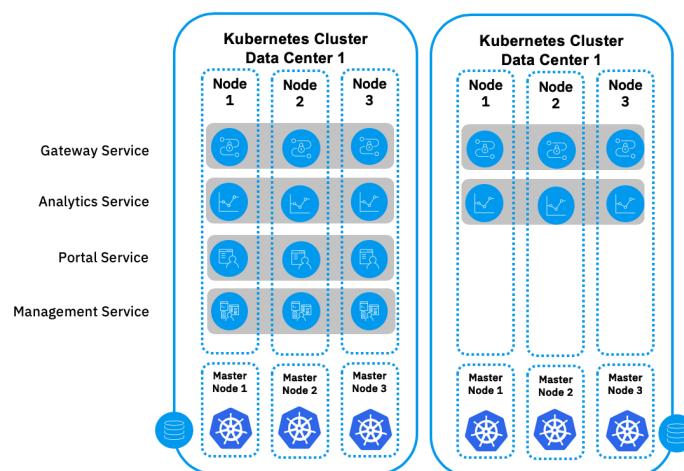


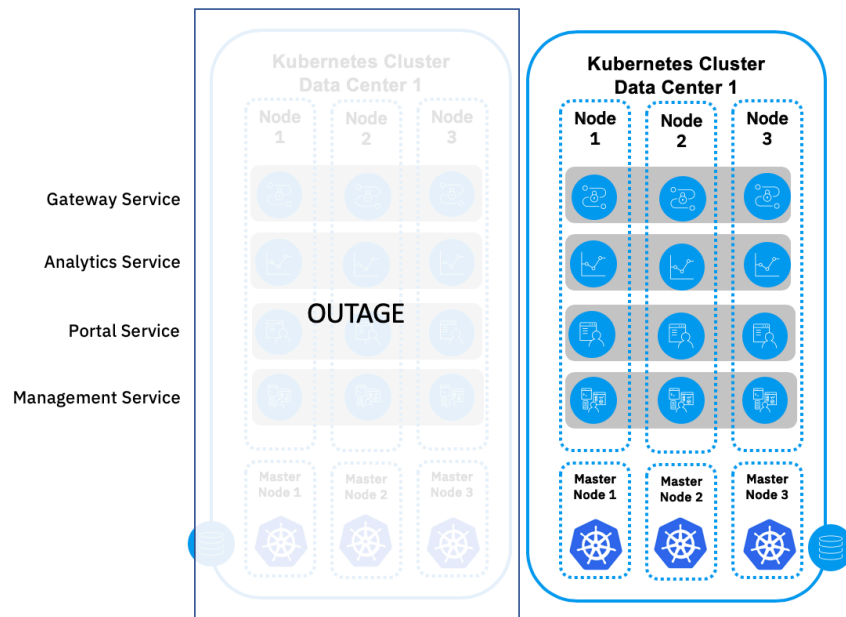
Figure 15: Active Active HA Pattern

The topology described here requires separate Kubernetes clusters 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 will no longer be 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.

Each API Connect component cannot 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 one goes down the other site is still serving the API Requests. Analytics Data is preserved for the site that has not gone down.

The Management and Portal are deployed in an Active-Passive Pattern. If site two goes down there is no Impact to the Management and Portal. If site one goes down, then the Management and Portal components are unavailable. A single Management Instance does not support running across multiple Management Components.



Manual Operation to bring API Manager and Portal up in Site 2.

Figure 16: Active Active HA Pattern - Failover

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

8.3.1 Availability

8.3.1.1 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 will not be able to be registered.

During a site outage a decision to invoke a Disaster Recovery may 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 26 Active Active Availability for Management and Portal

8.3.1.2 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 will be 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 27 Active Active Availability for Analytics and Kubernetes Master

8.3.1.3 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 28 Active Active Availability for Gateway

8.3.2 Advantages

- Supports Double Jeopardy i.e. can apply maintenance even during a site outage.
- Does not require three sites.
- Gateway and Analytics work in Active Active

8.3.3 Concerns

8.3.3.1 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 will have its own Quota enforcements

8.3.3.2 Disassociation

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

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

8.4 Active-Active with DataPower not in Kubernetes

This Scenario builds on 8.3. By removing DataPower from Kubernetes we can run a single DataPower service across two sites.

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

8.4.1 Scenario

The following statements describe the scenario for this pattern.

- The customer wants an Active - Active deployments in two sites.
- They are unable or unwilling to provide a third site.
- The customer is taking regular back-ups of the API Management and Portal and
- These backups are replicated to the second site.

- A low latency network connection between the Data Centers is required.

8.4.2 Topology

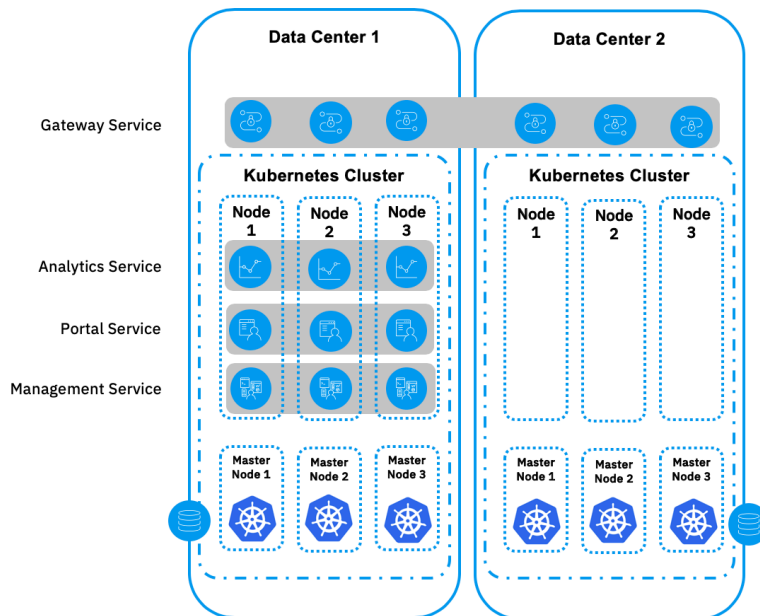


Figure 17: Active Active HA Pattern

The Manager, Analytics and Portal follow the same principles as 8.3. The DataPower Service crosses the Data Centers and so is able to sync revocation tokens and quota enforcement.

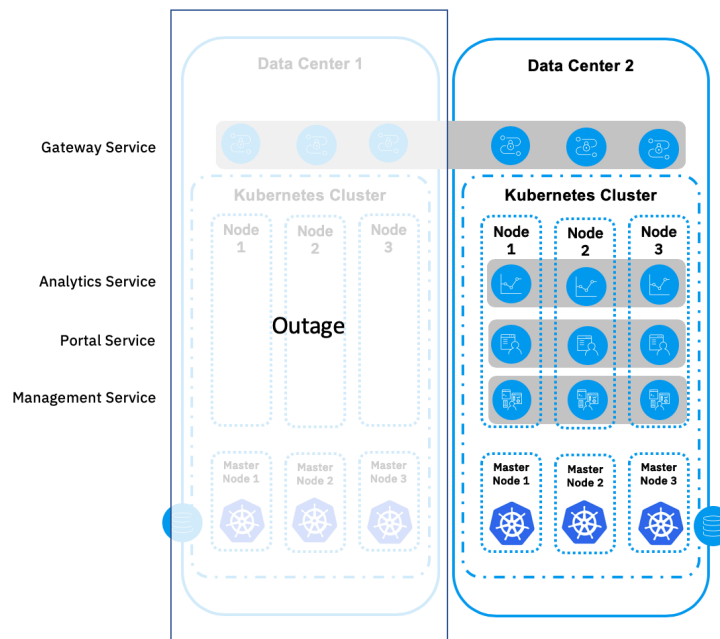


Figure 18: Active Active HA Pattern - Failover

8.4.3 Availability

8.4.3.1 Analytics Management and Portal

See 8.3.1

8.4.3.2 Kubernetes Master

See 8.3.1.2

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.

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 29 Active Active with Gateway not in Kubernetes Availability

8.4.4 Advantages

- Supports Double Jeopardy i.e. can 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.

8.4.5 Concerns

- Gateway cannot run in a Kubernetes cluster

8.5 Active/Passive (DR)

Active Passive is a common topology. The idea is that a single site will serve the traffic, however if an outage occurs a manual action will cause all traffic to be routed to the second site. The challenge here is that the data must be updated regularly to site two 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 it is not serving traffic.

8.5.1 Scenario

- The customer is content to have only a single Active site 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

8.5.2 Topology

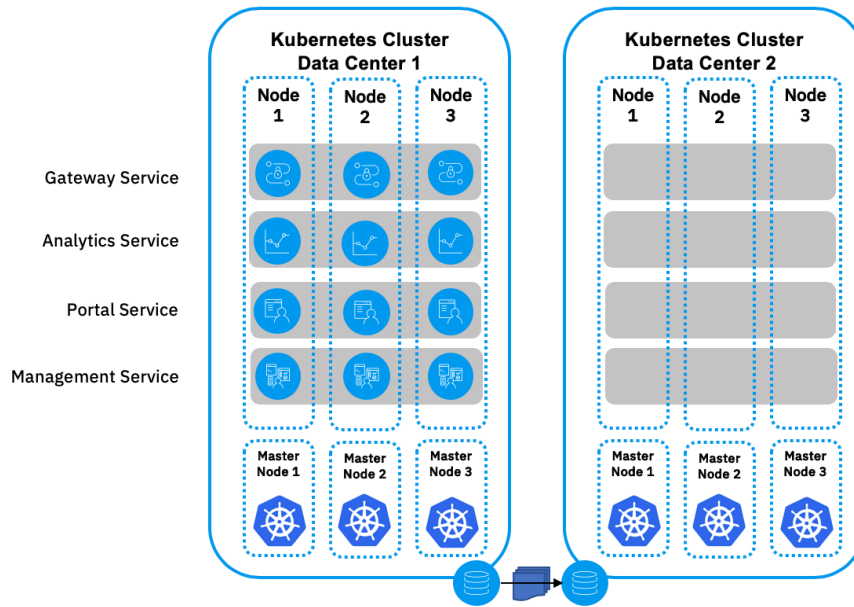


Figure 19: Active Passive HA Pattern

8.5.3 Availability

8.5.3.1 Management, Analytics, Portal and Kubernetes Master

	Node	Node	Node		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site,	
	1.1	1.2	1.3		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 30 Active Passive Availability for Management, Analytics, Portal and Kubernetes Master

8.5.3.2 Gateway

	Node	Node	Node		Node 2.1	Node 2.2	Node 2.3	No of Available Nodes	Can Apply Maintenance in Site,	
	1.1	1.2	1.3		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 31 Active Passive Availability for Gateway

8.5.4 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

8.5.5 Concerns

- In the event of a site outage all data will be lost that was written since the last backup was transferred to site 2.
 - Regular backups mitigate this concern.
- Outage will occur during site two 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 may be no license costs to consider.
 - If the Second site is a hot standby there may be license costs to consider.

8.6 Anti-Patterns

Anti-Patterns are solutions that have a significant risk.

These are NOT recommended.

8.6.1 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 resulting in a full DR scenario requiring mirrored backups and redeployment of Kubernetes new sites.

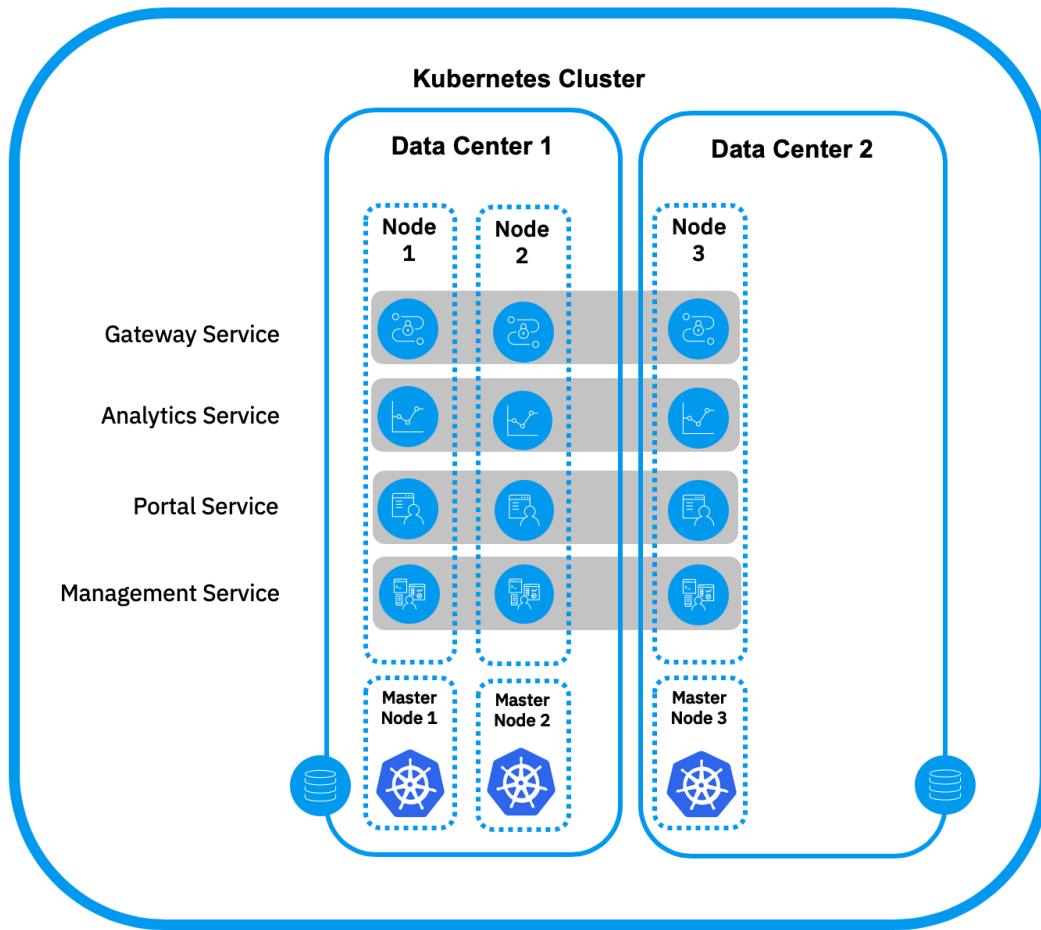


Figure 20: Anti-Pattern Active Active

8.6.1.1 Availability

8.6.1.1.1 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 32 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 has 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, update any applications in the cluster.

8.6.1.1.2 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 33 Anti-Pattern Availability of Gateway

8.6.2 V5 Deployment Pattern

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

8.6.2.1 Topology

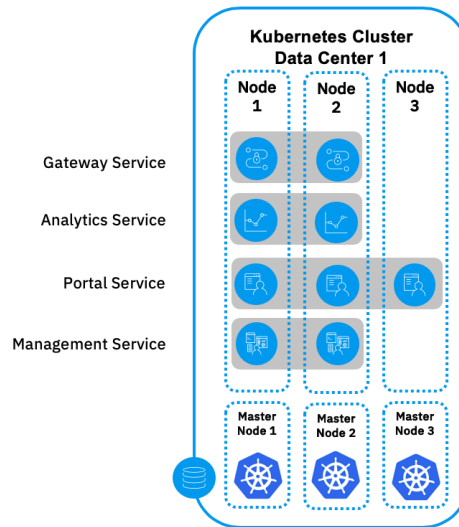


Figure 21: Anti-Pattern - V5 Deployment Pattern

8.6.2.2 Concerns

- Manager, 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

8.6.2.3 Mitigation

If this pattern is deployed with 50% additional capacity, the risk is mitigated. With the additional capacity available when a node is lost Kubernetes will automatically begin to deploy pods to nodes with available capacity. This will cause a short outage (measured in minutes) but the system will recover with no interaction.

8.6.3 Active / Active - Two Independent clusters across two Clusters

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

8.6.3.1 Topology

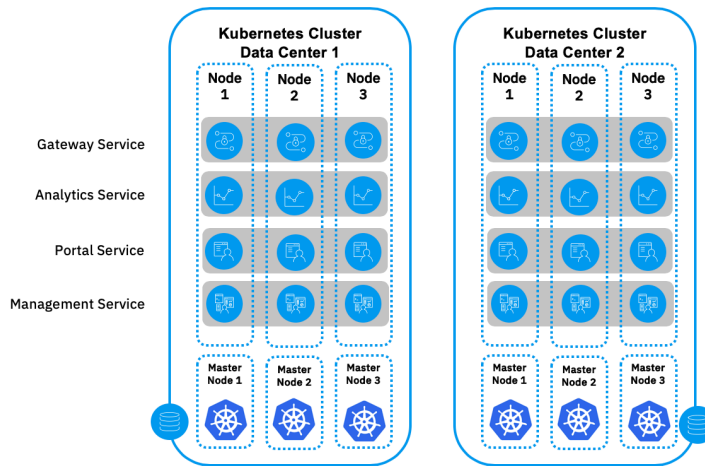


Figure 22: Anti-Pattern - Two Independent Clusters

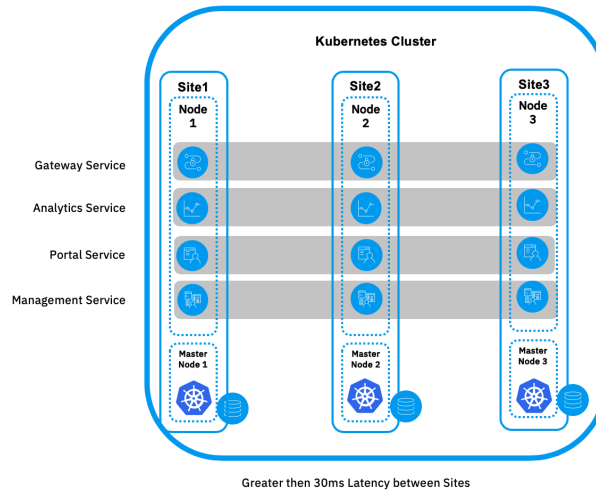
8.6.3.2 Concerns

- No single point of truth, I.e. 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 to 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 will get out of sync.
- Prone to human error

8.6.4 Active/Active/Active – Single K8s Cluster with a non low latency network

This is similar to pattern 1 (Active-Active-Active – Single K8s Cluster with low latency network) however the latency is over 30ms or more connections are routinely unexpectedly dropped.

8.6.4.1 Topology



8.6.4.2 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 off line 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 is true as of 18th January 2019.

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

The table below shows recommendations for each of the major Kubernetes deployment targets. If an entry does not exist in this table, please assume it is Not Tested.

Recommendation	Description
Recommended	IBM Tests this system as part of their test runs.
Reported Working but Not Tested	Customers have reported that this system is working but it is not tested as part of IBM's testing strategy
Known Limitation / Known Limitation but Not Tested	Either IBM has found limitations or customers have reported limitations. This solution may work with some caveats, see section below.
Not Supported	IBM has tested it is not possible to use.
Not Tested	IBM has not tested it and customers have not reported success or failure.

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

<i>OnPrem / Cloud Native</i>	<i>Deployment</i>	<i>Recommendation</i>
<i>On Premise</i>	Kubernetes	Recommended
<i>On Premise</i>	IBM Cloud Private	Recommended
<i>On Premise</i>	RedHat Open Shift	Recommended
<i>Cloud</i>	AWS - EKS	Reported Working but Not Tested
<i>Cloud</i>	AWS - Bare Metal	Recommended
<i>Cloud</i>	Azure – AKS	Reported Working but Not Tested
<i>Cloud</i>	Azure – Bare Metal	Not Tested
<i>Cloud</i>	GCP	Reported Working but Not Tested
<i>Cloud</i>	IBM Cloud – IKS	Known Limitation Note: A community Ingress must be deployed to IKS in order for analytics to work.
<i>Cloud</i>	IBM Cloud – Bare Metal	Recommended

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 assume it is Not Tested.

Recommendation	Description
Recommended	IBM Tests this system as part of their test runs.
Reported Working but Not Tested	Customers have reported that this system is working but it is not tested as part of IBM's testing strategy
Known Limitation / Known Limitation but Not Tested	Either IBM has found limitations or customers have reported limitations. This solution may work with some caveats, see section below.
Not Supported	IBM has tested it is not possible to use.
Not Tested	IBM has not tested it and customers have not reported success or failure.

<i>On Premise or Cloud Native</i>	<i>Storage Class</i>	<i>Recommended For</i>			
		Management	Analytics	Portal	Gateway
<i>On Premise</i>	Ceph RBD	Recommended	Recommended	Recommended	Recommended
	Ceph FS	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested
	Gluster FS	Not Supported	Not Supported	Not Supported	Not Supported
	NFS	Not Supported	Not Supported	Not Supported	Not Supported
	Host Path	Known Limitation in HA Environments	Known Limitation in HA Environments	Known Limitation in HA Environments	Known Limitation in HA Environments
<i>Cloud Native</i>	IBM Block Storage	Recommended	Recommended	Recommended	Recommended
	AWS Elastic Block Storage	Recommended	Recommended	Recommended	Recommended
	Azure Premium Storage	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested
	Azure Storage	Reported Limitation but Not Tested	Reported Limitation but Not Tested	Reported Limitation but Not Tested	Reported Limitation but Not Tested
	GCE Persistent Disk	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested	Reported Working but Not Tested

Table 34 Kubernetes Storage Type Summary

10.2 On Premise

This section provides information on storage solutions that can be deployed On Premise or any public cloud.

10.2.1 CephRBD

- Status: Recommended
- Tested by API Connect Development Team
- Extensive Testing

Ceph RBD is the primary storage solution in the development of API Connect. Ceph RBD can be deployed outside of Kubernetes or inside using Rook. **It is the recommended solution.**

For simple instructions on how to configure this for a POT or demo environment please read <https://medium.com/@cminion/how-to-use-rook-ceph-with-kubernetes-424a7a65173e>

10.2.2 CephFS

- Status: Reported Working but Not Tested
- **Not** tested by API Connect Development Team

10.2.3 GlusterFS

- Status: Not Supported

Gluster FS is designed to keep data sources in sync where large files are used. This works fine for the API Manager but does not work for the Portal or Analytics which are made up of multiple smaller files. When using Gluster FS we have had reported problems with consistencies between the replicas.

10.2.4 NFS

- Status: Not Supported

NFS is not supported at this time

10.2.5 Host Path

- Status: Known Limitation in Production
- **Significant** testing by API Connect Development Team for the OVA release.

The API Connect v2018 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 that if a node becomes unavailable the persistent storage is also lost for those pods.

10.3 Cloud Native

Each public cloud instance has their own storage solution. This section goes through the most commonly used Public Cloud offerings and the storage solution they provide.

10.3.1 IBM Block Storage

- Status: Recommended
- Tested by API Connect Development Team

10.3.2 AWS Elastic Block Storage (EBS)

- Status: Recommended
- Tested by API Connect Development Team
- Reported Working by one or more API Connect Customer

This topology is formally tested by the IBM team.

Please Note: In AWS the Pods can only bind EBS from the availability zone they are in. If the Kubernetes cluster is running across availability zones when a POD is restarted it may start in a new Availability Zone. This means it will lose the Persistent Volume. If multiple PODs in the same deployment restart at the same time this will cause an outage as quorum will be lost.

10.3.3 Azure Storage

Azure has two tiers of storage Premium and Regular.

10.3.3.1 Azure Premium Storage

- Status: Reported Working but Not Tested
- **Not** Tested by API Connect Development Team
- Reported Working by one or more API Connect Customer

Azure premium storage provides a faster write and read access to the disk. This has been used by API Connect platforms in Highly Available topologies

successfully however it is not tested as part of the Development Process of API Connect.

10.3.3.2 Azure Storage

- Status: **Reported Limitation but Not Tested**
- **Not** Tested by API Connect Development Team
- Reported not working by one or more API Connect Customers

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.

10.3.4 GCE Persistent Disk

- Status: **Reported Working but Not Tested**
- **Not** Tested by API Connect Development Team
- Reported Working by one or more API Connect Customer

Though a customer has reported that this is working IBM has not attempted to deploy to this storage solution.

11 Contributors

Thank you to **Jake Leonard** and **Rengan Sundararaman** for contributing to the content of this document and to the following people for proof reading and validating the content.

- Quentin Presley
- Chris Dudley
- Simon Tierny
- Frank Delporte
- Ivan Pryanichnikov
- Johan Thole
- Murali Sitaraman
- Nick Cawood
- Tom Van Oppens
- Peter Brabec
- Pierre Richell
- Adrian Osadcenco
- Anne Redwood
- Izzy Godard
- Taha Ben Masaud
- Jahid Zaynal
- Andy Garratt
- Sajan Sankaran
- Sebastian Sutter
- James Hedley

And many others