

Zero to Cloud-Native with IBM Cloud

Part 1: Introduction and Overview

Kevin Collins

kevincollins@us.ibm.com

Technical Sales Leader

IBM Cloud Enterprise Containers – Americas

Kunal Malhotra

kunal.malhotra3@ibm.com

Cloud Platform Engineer

IBM Cloud MEA

1 - Introduction

With over 200 services available in IBM Cloud Catalog you can develop virtual any cloud native application that you can dream of. But how can you best get started. The purpose of this blog series ‘Zero to Cloud-Native with IBM Cloud’ is to showcase how to get started in developing with the IBM Cloud Catalog. This multi-part series will cover topics such as architecture design and how to select the best services available in the IBM Cloud Catalog based on your architecture and application, how to combine services together and then how to implement day two operations after you have deployed your application.

You can find links to all the blog entries, source code and videos at this GitHub repository:

<https://github.com/kmcolli/zero-to-cloud-native-ibm-cloud>

The main focus of this series will be guiding you through a tutorial in developing a complete cloud native application from the ground up. There will be short blogs and accompanying videos describing a topic while going through the cloud-native development lifecycle. The series is broken down into individual topics that build off each other. Initially the series will go through the following:

1. Introduction – overview and introduction to developing a cloud native application with IBM Cloud. Will provide an overview of the tutorial application and the enterprise application the tutorial is based on.
2. Microservices Design – design considerations for developing a cloud-native application including patterns to follow when developing real time and long duration APIs. This will provide the foundation for the microservices that you will deploy in the tutorial. This section will also go through lessons learned, some the hard way, in developing an enterprise application.
3. Network and Security Configuration – network architecture and security design considerations for a cloud-native application. This will include setting up a Virtual Private Cloud Network, creating and securing a custom domain, storing and securing encryption keys and certificates.
4. RedHat OpenShift Cluster Configuration – deployment considerations and how to design a cluster for development, staging and production environments.

5. Cloud Databases and Messaging – This section will have two parts. First, how to use a managed service as a messaging broker to de-couple microservices and improve performance. The second part will go through examples of selecting a purpose-built database as a service to optimize the application architecture to support different data types.
6. Development Environment – how to setup both an effective command line terminal and development environment. This section will also cover several tools that are very handy when developing a cloud-native application.
7. Preparing to Deploy an Application – before deploying a cloud-native application, there are several preparing tasks that need to be done before you can deploy your code. This section will go through configuring all the services we will leverage in the tutorial to support the tutorial application.
8. Code Structure and Overview – before deploying the application, this section will go through a quick overview of the code that comprises the application. You will learn how to access the code and import it into your development environment we setup in part 6.
9. Deploying the Application – finally the day arrives where we will deploy the application. You will learn how to deploy the application manually, with a 'Classic' CI/CD toolchain and finally with a Tekton toolchain.

Future Planned Sessions

10. Logging – how to leverage logging while developing your cloud-native application and how to leverage logging to support the application while running in a production environment.
11. Monitoring – best practices to design and implement a monitoring dashboard for a cloud-native application.
12. Tuning / Capacity Planning – how to tune and scale your application to make the best use of your cloud resources in a cost-effective manner.
13. Automation – This section will go through how to use IBM Cloud to automate building the entire application environment, including all services, configuration settings, and the application code itself.
14. IBM Cloud Satellite - Deploying the application to a remote location leveraging IBM Cloud Satellite.

This series will be a living series, where I plan to add additional topics as I expand the application. Please drop me a line if there is a certain topic you would like to see in this series.

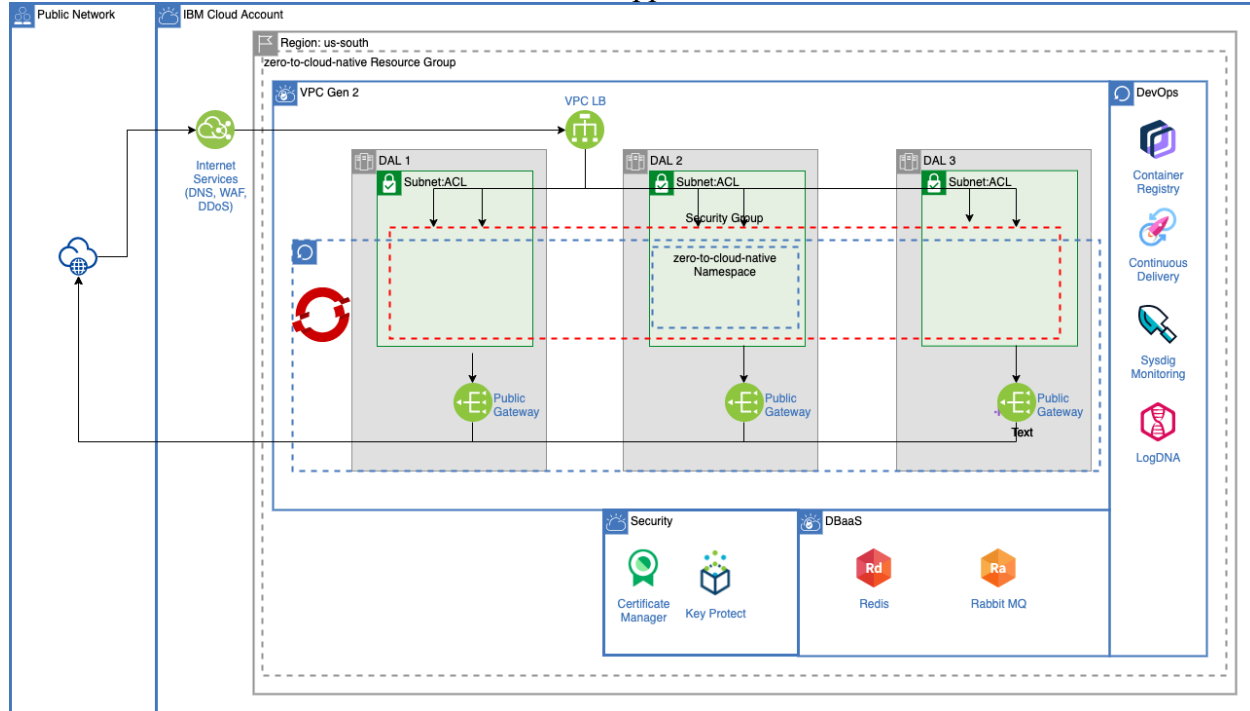
2 - Tutorial Application

The best what to learn, from my perspective, is getting hands on experience. The accompanying tutorial will go through the entire cloud-native development lifecycle. I will walk through step by step to create an entire cloud native application on IBM Cloud leveraging many services in the IBM Cloud Catalog.

The tutorials will go through the following topics:

- Microservices design
- Creating a new isolated resource group
- Creating a custom domain for your application.
- Setting up and configuring a virtual private cloud including configuring subnets, access control lists and security groups.
- Configuration of IBM Cloud Internet Services to secure and provide DNS services for a custom domain
- Leveraging certificate manager to create and managed domain certificates
- Using Key Protect to manage your own certificates across IBM Cloud services
- Provisioning and configuring a managed OpenShift Cluster in a multi-zone region
- Provisioning and configuring Rabbit MQ for messaging
- Provisioning and configuring IBM Cloud Databases for a data layer
- Creating a classic CI/CD pipeline with integration with an enterprise git repository
- Creating a tekton CI/CD pipeline with integration with an enterprise git repository
- Deploying and running the application on OpenShift
- Leveraging IBM Log Analysis with LogDNA to provide logging and troubleshoot both from a development and support perspective.
- Leveraging IBM Systems Monitoring with SysDig to monitor IBM Cloud Platform components, OpenShift cluster and the application itself.
- Tuning your cluster to optimize availability and resources.
- Using IBM Cloud Schematics and Terraform to automate the provisioning of the entire tutorial application and all the resources it leverages.
- Creating a remote location using IBM Cloud Satellite to the deploy the tutorial applications and services.
- New functionality from IBM Cloud to clone and re-instate cluster namespaces

This is the architecture overview of the tutorial application we will create.



The tutorial application will comprise of three APIs:

- 1) **getOCPVersions** – will return a list of OpenShift versions supported by IBM Cloud Managed OpenShift
- 2) **getOCPToken** – will return an OpenShift token to login into your cluster.
- 3) **enableNodeSSH** – will enable your IBM Cloud Managed OpenShift worker nodes to be SSH'd into.

To support these APIs, the following microservices, all written in Python, will be created.

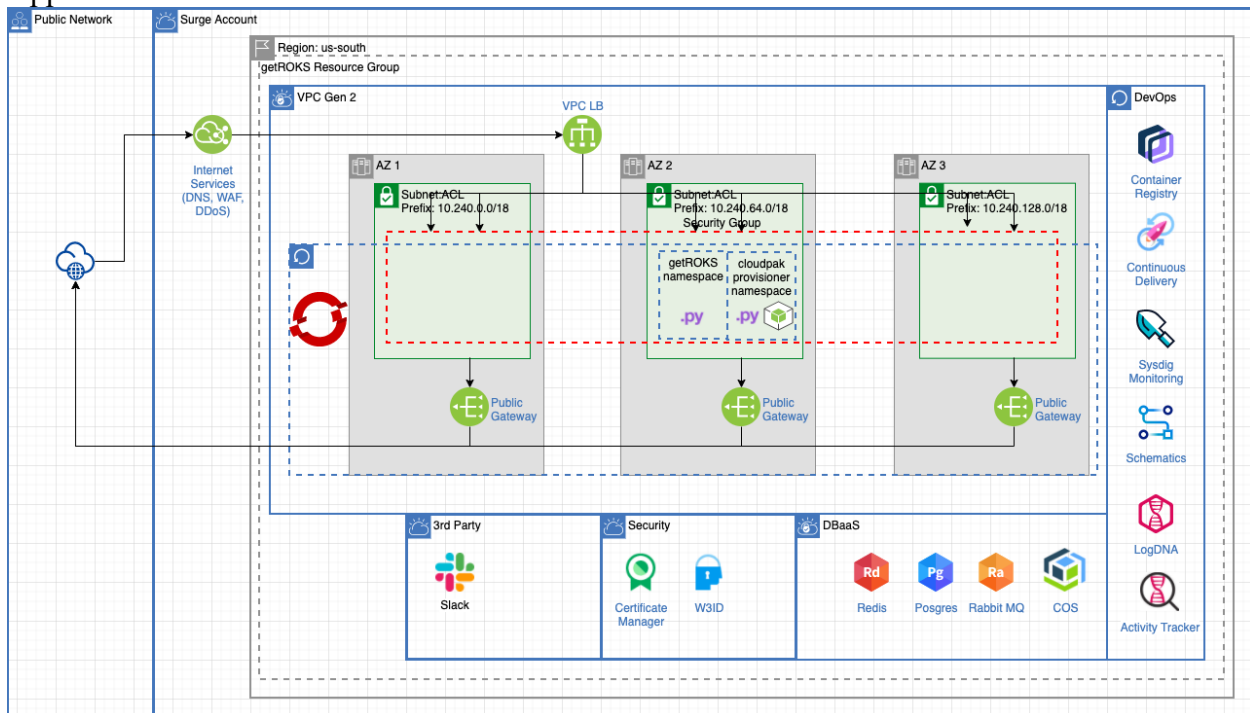
- **frontend-web-02cn.py** – HTML front end to invoke the three public facing APIs
- **frontend-api-02cn.py** – API interface to call the public facing APIs
- **ocp-realtime-02cn.py** – implementation of the getOCPVersions and getOCPToken APIs
- **loadOCPVersions-02cn.py** – Kubernetes cronjob that will load current versions of OCP supported by IBM Cloud into a Redis database.
- **enableNodeSSH-02cn.py** – implementation of the 'long' running enableNodeSSH API

3 - Example Enterprise Application

To help guide the discussion and share lesson learned, I'm going to step through a real-world cloud-native application that I helped develop that is being used across IBM. The tutorial is actually a subset of enterprise application that I will discuss. The reason for both a tutorial and a real-world application is because many times a tutorial is too simplistic. When you are developing a real-world enterprise ready application there are many issues you face that a tutorial doesn't cover. I'll point out some of the additional complexities to consider while you are developing an enterprise level application. The tutorial is meant to get hands on experience in developing a complete cloud-native application. The tutorial application will provide a foundation that you can extend and make it your own.

The cloud native application that I will go through the development lifecycle of actually has two applications one called getROKS and other called CloudPak Provisioner. The getROKS application was developed to provide an automated web application that allows IBMers to request and provision an IBM Cloud Managed OpenShift cluster to use for a proof of concept with our clients. CloudPak Provisioner is an API based application that provides easy to use APIs to simplify and automate common tasks that are performed with RedHat OpenShift and IBM Cloud Paks running on IBM Cloud. Both applications leverage many different services from the IBM Catalog that each section of this series will go through. Each topic will include the architecture decisions on why certain services were used, implementation details on how to use the IBM Cloud Services and lessons learned. You will notice that the tutorial is based on the CloudPak Provisioner application.

Application Architecture:



4 - CloudPak Provisioner

Cloud Pak Provisioner is a collection of APIs that make it easy to perform common tasks for managing and configuring ROKS clusters that is available for anyone to use as it is exposed on the Internet. Many of the APIs are focused on IBM CloudPaks but can be extended to any deployment of Managed OpenShift on IBM Cloud. You are more than welcome to try out any of the CloudPak Provisioner APIs.

If you would like to try out the CloudPak Provisioner APIs, I have created a Postman Collection that you can import.

Postman Collection: <https://www.getpostman.com/collections/bc63c30083ec77b2c3c5>

You can also view more detailed documentation on using the APIs here:

<https://github.com/kmcolli/CloudPakProvisioner>

API Reference

There are seven categories of APIs:

- IBM Cloud Object Storage
- IBM CloudPak for Data
- VPC Infrastructure
- General Utility
- Classic Infrastructure
- Portworx
- OpenShift

IBM Cloud Object Storage.

Retrieve a list of COS Buckets

Retrieves a list of buckets including metadata for the bucket and the size of the bucket. Search criteria includes being able to search by name and metadata.

Delete COS Bucket and all Objects

Single API call that deletes all objects in a bucket and the bucket itself.

Add Metadata to COS Bucket

Adds metadata to a bucket by creating a file called README file in the bucket that contains key value pairs. Metadata given to the API call will overwrite the existing metadata.

Retrieve a List of COS Metadata for a Bucket

Retrieves a list of metadata for a bucket by reading a file name README containing key value pairs.

Copy a COS Bucket to Another Bucket

Single API that directly copies one bucket to another in the same COS instance.

Create a COS PVC on a ROKS Cluster

This api will do the following:

- Install the ibm-object-storage helm chart on a ROKS Cluster
 - Create a COS bucket if not already created
 - Create a secret in ROKS to access the COS Bucket
 - Create a pvc attached to the ROKS cluster
-

CloudPak for Data.

Add Certificates to Cloud Pak for Data

- Creates a dns entry in cloud internet services mapping cp4d-
<cluster_name>.ibmcloudroks.net to the cp4d console route that references the cluster ingress subdomain.
- updates CP4D nginx pods with ibmcloudroks.net certificates
- updates the cp4d route to point to the secured URL

Update Certificates for Cloud Pak for Data

- updates the CP4D nginx pods with ibmcloudroks.net certificates

Run CP4D Pre-installation Script

- runs the cp4d pre-installation script before you can install cp4d with IBM Cloud Schematics
 - optimizes CP4D by setting linux worker node parameters no root squash and setting the kernel semaphore.
-

VPC.

Attach raw block devices to ROKS Worker Nodes on a VPC

- creates raw block devices on VPC Gen 2 in each zone the worker nodes are in
- attaches the created block devices to ROKS worker nodes
- works in both MZR and SZR

Delete raw block devices to ROKS Worker Nodes on a VPC

- unattaches block devices created with the *attachBlockVPC* api.
- deletes block devices

Get list of block devices attached to ROKS Worker Nodes on a VPC

- retrieves a list of block devices attached to worker nodes on a VPC.

Classic Infrastructure.

Remove unused block volumes on classic infrastructure

- removes all block volumes not attached to a Kubernetes cluster
- *warning - this is destructive and will immediate delete all block volumes that are not being used by Kubernetes. Only run this if you are sure you are not using block volumes outside of Kuberenetes

Portworx.

Provision portworx using the IBM Cloud Catalog

- Creates a portworx instance for a given cluster
- Portworx instance is created in the same resource group as the cluster
- Assumes raw block storage devices are already created.

Create Block Storage on Classic

- Installs the IBM Block Attacher Helm chart into the given cluster
- For each worker node in the cluster the following is performed:
 - orders a block device using the maximum IOPS for the given size
 - authorizes the worker node to use the block device
 - attaches the block device to the worker node
 - creates a PV for the block device

Utility

Check if ROKS Cluster is ready

Checks a ROKS cluster to see if the ingress subdomain has been set for the cluster and the cluster details populate the resource group name. After the cluster is provisioned, it can take up to 30 minutes for the cluster to recognize the resource group name it was create in.

OpenShift.

Get ROKS Versions

Using a local Redis cache, this api quickly returns a list of versions available in ROKS.

Get ROKS Token

Retrieves an ROKS token and login command by providing your API key and cluster name.