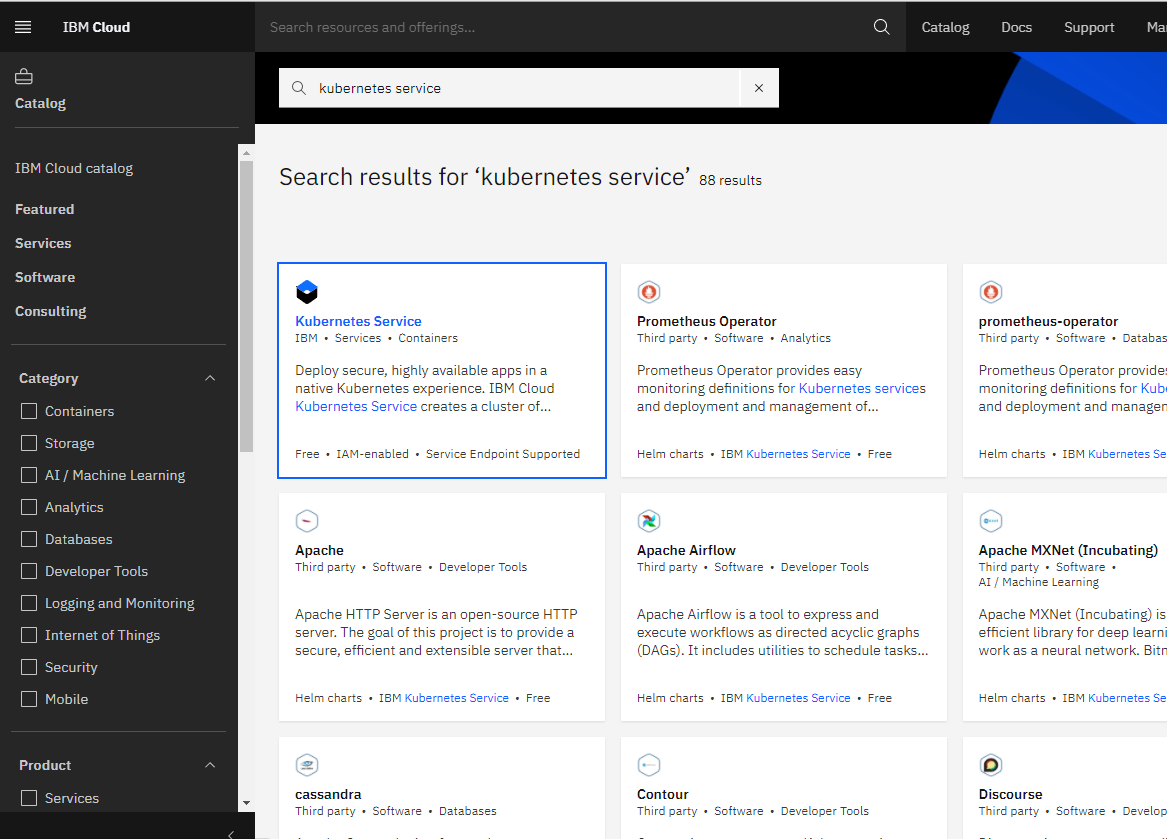
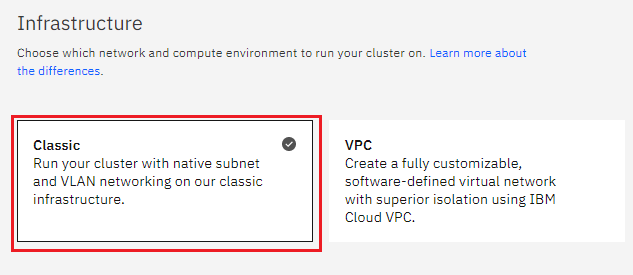
# Installing Kong on IBM Cloud

**Step 1 provision Kubernetes Cluster**

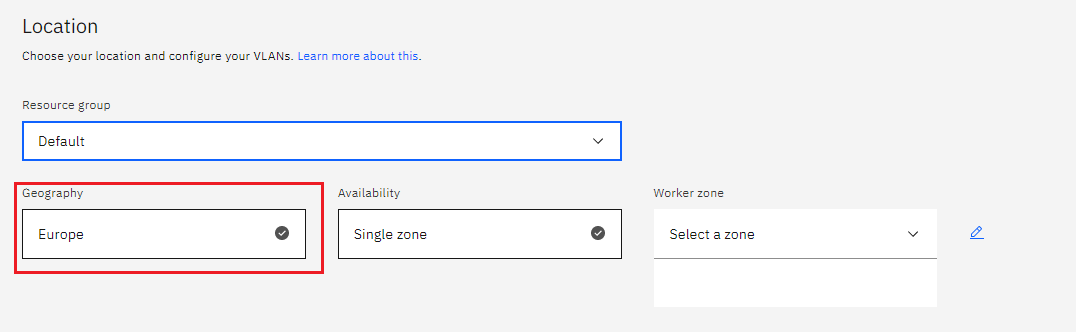
* Click the **Catalog** button on the top
* Select **Service** from the **Catalog**
* Search for **Kubernetes Service** and click on it



* You are now at the Kubernetes deployment page. You need to specify some details about the cluster
* Choose a plan **standard** or **free** , the free plan only has one worker node and no subnet, to provision a standard cluster, you will need to upgrade your account to Pay-As-You-Go
* To upgrade to a Pay-As-You-Go account, complete the following steps:
* In the console, go to Manage > Account.
* Select Account settings; and click Add credit card.
* Enter your payment information, click Next, and submit your information
* Choose **classic** or **VPC** , read the docs and choose the most suitable type for yourself



* Now choose your location settings,
* Choose **Geography** (continent)



* + Choose Single or Multizone, in single zone your data is only kept in on datacenter, on the

other hand with Multizone it is distributed to multiple zones, thus safer in an unforeseen

zone failure

* If you wish to use Multizone please set up your account with[VRF
* If at your current location selection, there is no available Virtual LAN, a new Vlan will be created for you
* Choose a Worker node setup or use the preselected one, set Worker node amount per zone
* Choose **Master Service Endpoint** , In VRF-enabled accounts, you can choose private-only to make your master accessible on the private network or via VPN tunnel. Choose public-only to make your master publicly accessible. When you have a VRF-enabled account, your cluster is set up by default to use both private and public endpoints.  
  Give desired **tags** to your cluster, for more information visit tags
* Click **create**  
  • Wait for your cluster to be provisioned  
  • Your cluster is ready for usage

**Step 2 Deploy IBM Cloud Block Storage plug-in**

The Block Storage plug-in is a persistent, high-performance iSCSI storage that you can add to your apps by using Kubernetes Persistent Volumes (PVs).

* Click the **Catalog** button on the top
* Select **Software** from the catalog
* Search for **IBM Cloud Block Storage plug-in** and click on it  
  • On the application page Click in the dot next to the cluster, you wish to use  
  • Click on Enter or Select Namespace and choose the default Namespace or use a custom one (if you get error please wait 30 minutes for the cluster to finalize)
* Give a **name** to this workspace
* Click **install** and wait for the deployment

**Step 3 : Deploying Kong**

To deploy Kong onto your Kubernetes cluster with Helm, use:

*$ helm repo add kong https://charts.konghq.com*

*$ helm repo update*

*# Helm 2*

*$ helm install kong/kong*

*# Helm 3*

*$ helm install kong/kong --generate-name --set ingressController.installCRDs=false*

**Production configuration**

This will have a value-production.yaml file where some parameters oriented to production configuration in comparison to the regular values.yaml can be find. You can use this file instead of the default file.

* Enable exposing Prometheus metrics:

*- metrics.enabled: false*

*+ metrics.enabled: true*

* Enable Pod Disruption Budget:

*- pdb.enabled: false*

*+ pdb.enabled: true*

* Increase number of replicas to 4:

- replicaCount: 2

+ replicaCount: 4

* Enable exposing Prometheus metrics:

*- metrics.enabled: false*

*+ metrics.enabled: true*

**Database backend**

* Deploy the PostgreSQL sub-chart (default)

*helm install my-release bitnami-ibm/kong*

* Use an external PostgreSQL database

*helm install my-release bitnami-ibm/kong \*

*--set postgresql.enabled=false \*

*--set postgresql.external.host=\_HOST\_OF\_YOUR\_POSTGRESQL\_INSTALLATION\_ \*

*--set postgresql.external.password=\_PASSWORD\_OF\_YOUR\_POSTGRESQL\_INSTALLATION\_ \*

*--set postgresql.external.user=\_USER\_OF\_YOUR\_POSTGRESQL\_INSTALLATION\_*

* Deploy the Cassandra sub-chart

*helm install my-release bitnami-ibm/kong \*

*--set database=cassandra \*

*--set postgresql.enabled=false \*

*--set cassandra.enabled=true*

* Use an existing Cassandra installation

*helm install my-release bitnami-ibm/kong \*

*--set database=cassandra \*

*--set postgresql.enabled=false \*

*--set cassandra.enabled=false \*

*--set cassandra.external.hosts[0]=\_CONTACT\_POINT\_0\_OF\_YOUR\_CASSANDRA\_CLUSTER\_ \*

*--set cassandra.external.hosts[1]=\_CONTACT\_POINT\_1\_OF\_YOUR\_CASSANDRA\_CLUSTER\_ \*

*...*

*--set cassandra.external.user=\_USER\_OF\_YOUR\_CASSANDRA\_INSTALLATION\_ \*

*--set cassandra.external.password=\_PASSWORD\_OF\_YOUR\_CASSANDRA\_INSTALLATION\_*

**Sidecars and Init Containers**

If you have a need for additional containers to run within the same pod as Kong (e.g. an additional metrics or logging exporter), you can do so via the sidecars config parameter. Simply define your container according to the Kubernetes container spec.

*sidecars:*

*- name: your-image-name*

*image: your-image*

*imagePullPolicy: Always*

*ports:*

*- name: portname*

*containerPort: 1234*

Similarly, you can add extra init containers using the initContainers parameter.

*initContainers:*

*- name: your-image-name*

*image: your-image*

*imagePullPolicy: Always*

*ports:*

*- name: portname*

*containerPort: 1234*

**Adding extra environment variables**

In case you want to add extra environment variables (useful for advanced operations like custom init scripts), you can use the kong.extraEnvVars property.

kong:

extraEnvVars:

- name: KONG\_LOG\_LEVEL

value: error

Alternatively, you can use a ConfigMap or a Secret with the environment variables. To do so, use the kong.extraEnvVarsCM or the kong.extraEnvVarsSecret values.

The Kong Ingress Controller and the Kong Migration job also allow this kind of configuration the ingressController.extraEnvVars, ingressController.extraEnvVarsCM, ingressController.extraEnvVarsSecret, migration.extraEnvVars, migration.extraEnvVarsCM and migration.extraEnvVarsSecret values.

**Using custom init scripts**

For advanced operations, the Bitnami Kong charts allows using custom init scripts that will be mounted in /docker-entrypoint.init-db. You can use a ConfigMap or a Secret (in case of sensitive data) for mounting these extra scripts. Then use the kong.initScriptsCM and kong.initScriptsSecret values.

elasticsearch.hosts[0]=elasticsearch-host

elasticsearch.port=9200

initScriptsCM=special-scripts

initScriptsSecret=special-scripts-sensitive

**Deploying extra resources**

There are cases where you may want to deploy extra objects, such as KongPlugins, KongConsumers, amongst others. For covering this case, the chart allows adding the full specification of other objects using the extraDeploy parameter. The following example would activate a plugin at deployment time.

## Extra objects to deploy (value evaluated as a template)

##

extraDeploy: |-

- apiVersion: configuration.konghq.com/v1

kind: KongPlugin

metadata:

name: {{ include "common.names.fullname" . }}-plugin-correlation

namespace: {{ .Release.Namespace }}

labels: {{- include "common.labels.standard" . | nindent 6 }}

config:

header\_name: my-request-id

plugin: correlation-id

**Setting Pod's affinity**

This chart allows you to set your custom affinity using the affinity parameter.

As an alternative, you can use of the preset configurations for pod affinity, pod anti-affinity, and node affinity available at the [bitnami/common](https://github.com/bitnami/charts/tree/master/bitnami/common" \l "affinities" \t "_blank) chart. To do so, set the podAffinityPreset, podAntiAffinityPreset, or nodeAffinityPreset parameters.