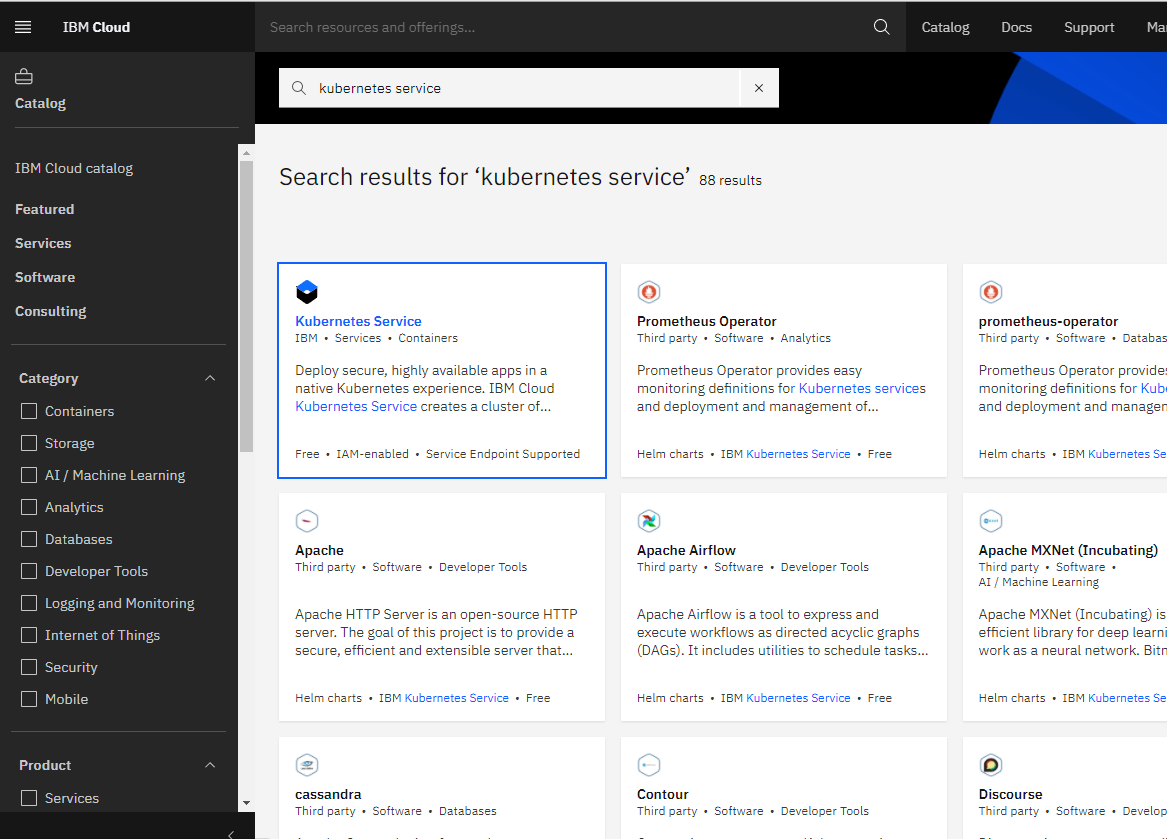
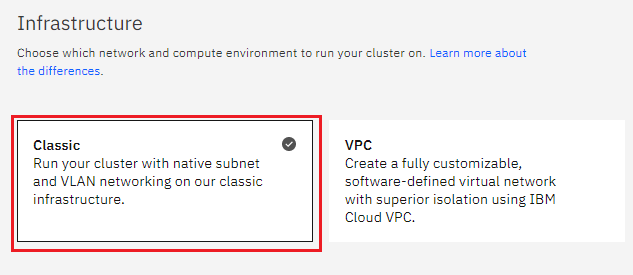
# Installing ASP.NET on IBM Cloud

**Step 1 provision Kubernetes Cluster**

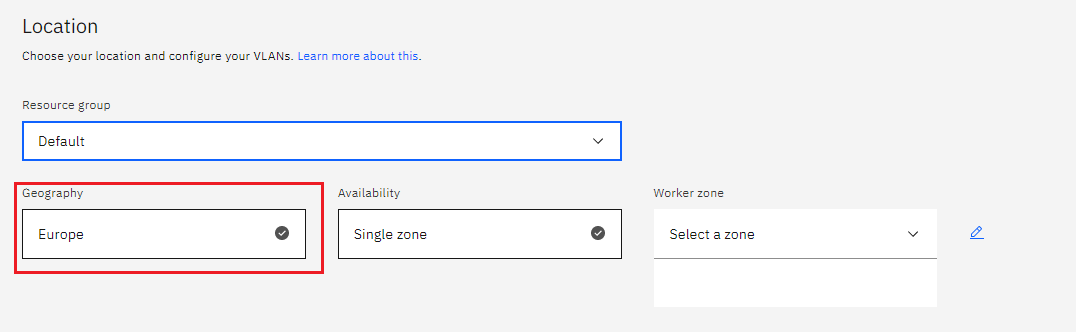
* Click the **Catalog** button on the top
* Select **Service** from the **Catalog**
* Search for **Kubernetes Service** and click on it



* 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 Installing ASP.NET**

### **Create a Kubernetes cluster on the IKS environment**

1. Log into your [IBM Cloud](https://cloud.ibm.com/login?cm_sp=ibmdev-_-developer-tutorials-_-cloudreg) account by using:

ibmcloud login

or

ibmcloud login --sso

1. Create the IKS cluster for deployment

* Create a [Kubernetes cluster](https://cloud.ibm.com/containers-kubernetes/overview) by choosing **Cluster Type – Free**. Give a unique name to the cluster and click **Create Cluster**

1. Deploy your containerized application to the Kubernetes cluster. From now on, you’ll use the kubectl command line.
2. Follow the instructions in the **Access** tab to set up your kubectl CLI and get access to your cluster.
3. On running the kubectl get nodes command, you will see something like the following.

NAME STATUS AGE VERSION

10.76.197.43 Ready 1d v1.10.8+IKS

### **Create a Cloudant database in the IKS Cluster**

To create an IBM Cloud Cloudant Database, create a new [Cloudant](https://cloud.ibm.com/catalog/services/cloudant) database instance. Select **Use both legacy credentials and IAM** under **Available authentication methods**.

1. Create new credentials under **Service Credentials** and copy the value of the **url** field
2. Create a Kubernetes secret with your Cloudant credentials.

kubectl create secret generic cloudant --from-literal=url=<URL>

1. You will need this information for the deployment. You can see your secrets by using the following command.

kubectl get secrets

This will display all the secrets you created in their respective clusters.

### **Deploy ASP.NET Core app to an IKS cluster**

The [IBM Cloud Container Registry](https://cloud.ibm.com/kubernetes/catalog/registry) provides a multi-tenant private image registry that you can use to safely store and share your Docker images with users in your IBM Cloud account.

1. Log in to the Container Registry Service to store the Docker image that we created with Docker.

ibmcloud cr login

1. Find your container registry namespace by running the following command.

ibmcloud cr namespaces

1. If you don’t have any, create one by using following command.

ibmcloud cr namespace-add <name>

1. Identify your Container Registry by running the following command.

ibmcloud cr info

1. Build and tag (-t) the Docker image by running the command below, replacing REGISTRY and NAMESPACE with the appropriate values.

docker build . -t <REGISTRY>/<NAMESPACE>/myapp:v1.0.0

It will display the following message in the end.

...

Successfully tagged registry.ng.bluemix.net/aspnetapp-01/app:v1.0.0

1. Push the Docker image to your [Container Registry on IBM Cloud](https://cloud.ibm.com/docs/services/Registry?topic=registry-index#index).

docker push <REGISTRY>/<NAMESPACE>/myapp:v1.0.0

1. Verify that the image was pushed successfully by running the following command.

ibmcloud cr image-lis

### **Deploy your containerized application**.

1. To create a deployment, you will create a folder called **“kubernetes”** and create a [deployment.yaml](https://github.ibm.com/Nidhi-N-Shah/ASP.NET-CORE-App-Deployment-in-IKS/blob/master/Kubernetes/deployment.yaml) file.
2. Create a deployment by using the following command.

kubectl create -f kubernetes/deployment.yaml

The output will display, similar to the following message.

deployment "get-started-aspnet" created

1. By default, the pod is only accessible by its internal IP within the cluster. Create a Kubernetes Service object that external clients can use to access an application running in a cluster. The Service provides load balancing for an application.

Use the NodePort 8080 to expose the deployment.

kubectl expose deployment get-started-aspnet --type NodePort --port 8080 --target-port 8080

### **Access the application**

To verify that your application is running successfully, you need to check the STATUS of your pod. It should be in a state of Running:

kubectl get pods -l app=get-started-aspnet

It will appear like this:

NAME READY STATUS RESTARTS AGE

get-started-aspnet-68d6dc5c4-2trcl 1/1 Running 0 1m

get-started-aspnet-68d6dc5c4-qdbkt 1/1 Running 0 1m

It should also show two instances as we have set two replicas in our deployment.

To access your ASP.Net Core application:

1. Identify your Worker Public IP by using ibmcloud cs workers YOUR\_CLUSTER\_NAME.
2. Identify the Node Port by using kubectl describe service get-started-aspnet.
3. Access your application at http://<WORKER-PUBLIC-IP>:<NODE-PORT>/.

This is how you can deploy and access your application in the IKS environment.

### **Clean up**

Use the following commands to clean up the sample application .

kubectl delete deployment,service -l app=get-started-aspnet

kubectl delete secret cloudant