In this lab, You will deploy web application and database on **Kubernetes Cluster** on IBM Cloud using the package manager for Kubernetes which is called **"Helm"**.

Pre-req:

Lab : Helm runs as a CLI client, so is installed on your laptop

Install Helm from the link –
https://docs.helm.sh/using_helm/#installing-helm

Options for installing Helm
1. Download the release, including the binary
− https://github.com/kubernetes/helm/releases
2. Homebrew on MacOS
− brew install kubernetes-helm
3. Installer script
− curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > get_helm.sh

STEP 1: Login to IBM Cloud and initialise the Helm.

1.1 Login to IBM cloud, Execute "**bx login** "on your local system terminal window to login. Note if you are using IBM federated email then execute with –sso option

1.2 After successful login to IBM cloud, Initialize the Container plugin by executing the below command.
Execute "**bx cs init**"

1.3 Make sure kubectl is installed properly and it can communicate to the cluster.
Execute "**kubectl version**"

the output of this command should return both server and client version as follows:
Client Version: version.Info{Major:"1", Minor:"7", GitVersion:"v1.7.3", GitCommit:"2c2fe6e8278a5db2d15a013987b53968c743f2a1", GitTreeState:"clean", BuildDate:"2017-08-03T07:00:21Z", GoVersion:"go1.8.3", Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"8+", GitVersion:"v1.8.8-2+9d6e0610086578", GitCommit:"9d6e06100865789613cbac936edce948f0710a2f", GitTreeState:"clean", BuildDate:"2018-02-23T08:20:09Z", GoVersion:"go1.8.3", Compiler:"gc", Platform:"linux/amd64"}

1.3.1 This step is required only if kubectl client is not able to connect to your cluster or if server version is not returned.

Execute "**bx cs cluster**" to get the list of clusters available in your account. If there is no cluster reported then create one by login to IBM Cloud console

- Execute "**bx cs cluster-config <cluster name>**",
- Replace <cluster name > with your cluster name, received from step 1.3.1, Output of this command return the cluster name and environment setting need to be execute.
- Execute the export command in the output of cluster-config command.

Example: export KUBECONFIG=/Users/ragdeshp/.bluemix/plugins/container-service/clusters/mycluster/kube-config-hou02-mycluster.yml

STEP 2: Prepare the Helm chart for web tier

Execute the "helm create web" from parent directory of web.

This will create following folder structures:
web

- ├── Chart.yaml
- ├── charts
- ├── templates
- │ ├── NOTES.txt
- │ ├── _helpers.tpl
- │ ├── deployment.yaml
- │ ├── ingress.yaml
- │ └── service.yaml
- └── values.yaml

The above files need to be modified as per our requirement.
Replace the values.yaml, deployment.yaml and service.yaml with the one provided to you in web folder.

STEP 3: Test and install the "web" helm chart

Once the deployment and service yamls are updated as per your application, you can test your charts

3.1 Execute the following command to package your charts.
"helm package web"
This will package your charts and that be released to release to your repos

3.2 Execute "helm install –debug –dry-run web" to test your charts

3.3 Execute "helm install web" to install your package on your cluster

STEP 4: Prepare the Helm chart for "db"

Execute the "helm create db" from parent directory of db.

This will create following folder structures:
db:

- ├── Chart.yaml
- ├── charts
- ├── templates
- │ ├── NOTES.txt
- │ ├── _helpers.tpl
- │ ├── deployment.yaml

- │   ├── ingress.yaml
- │   └── service.yaml
- └── values.yaml

The above files need to be modified as per our requirement.
Replace the values.yaml, deployment.yaml and service.yaml with the one provided to you in db folder.

STEP 5: Test and install the "db" helm chart

Once the deployment and service yamls are updated as per your application, you can test your charts

3.1 Execute the following command to package your charts.
"helm package db"
This will package your charts and that be released to release to your repos

3.2 Execute "helm install –debug –dry-run db" to test your charts

3.3 Execute "helm install db" to install your package on your cluster

STEP 6: List the helm releases

Execute "helm list" to display the list of helm releases one for "web" and another for "db"

STEP 7: Get the Public IP of the cluster through which we you can route the traffic

Execute "helm cs workers <<my cluster name>>" and you should see the following output as below:

Get the Public IP address of the cluster.

STEP 8: View the Pods and Services created as part of Helm install

Execute "kubectl get pods" to list the number of pods created
Execute "kubectl get svc" to list the number of services created

STEP 9: Run the curl command to send requests to the web application:

```
$ curl <<Cluster public IP>>:31000
Hello Container World from web-829031562-jsm60! I have been seen 1 times.

$ curl <<Cluster public IP>>:31000
Hello Container World from web-829031562-k9r38! I have been seen 2 times.
```

Each call is handled by one of the instances of the web service we created.

Congratulations!! You could successfully create Kubernetes resources using the Helm package manager.