

**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**

**Dehradun**

**APPLICATION CONTAINERISATION**

**Name: Shivani Chauhan**

**Course: B. TECH CSE DevOps (2018-22)**

**Roll no.: R171218096**

**Sapid: 500068969**

**EXPERIMENT: 8**

**AIM-** Introduction to kubernetes

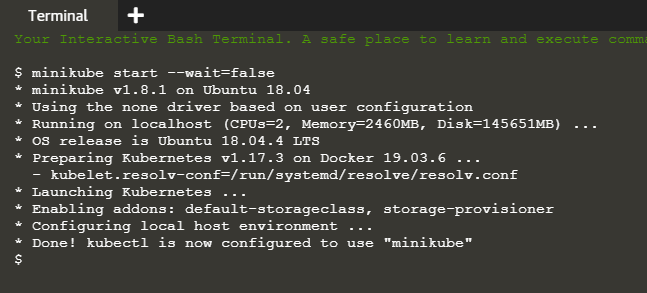
**Step 1 - Launch Cluster**

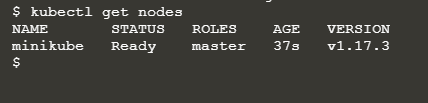
To start we need to launch a Kubernetes cluster.

Execute the command below to start the cluster components and download the Kubectl CLI.

minikube start --wait=false

Wait for the Node to become Ready by checking kubectl get node





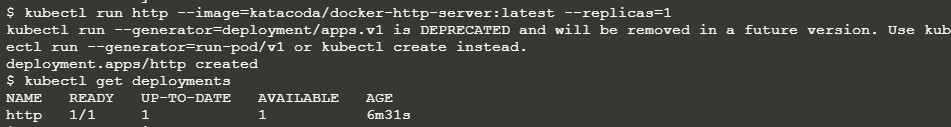
#### Step 2 - Kubectl Run

The following command will launch a deployment called http which will start a container based on the Docker Image katacoda/docker-http-server:latest.

kubectl run http --image=katacoda/docker-http-server:latest --replicas=1

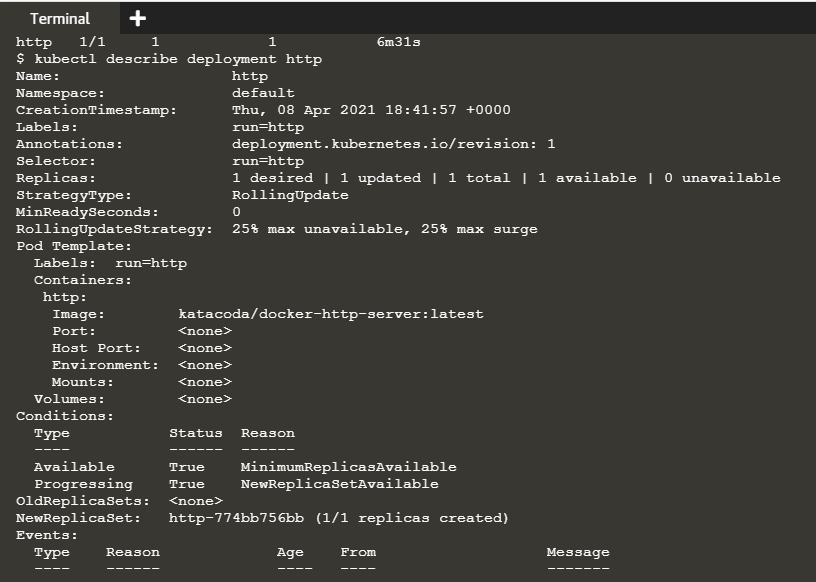
You can then use kubectl to view the status of the deployments

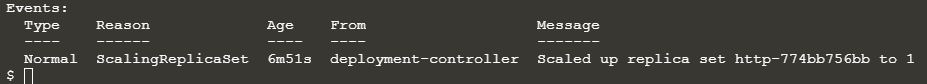
kubectl get deployments



To find out what Kubernetes created you can describe the deployment process.

kubectl describe deployment http





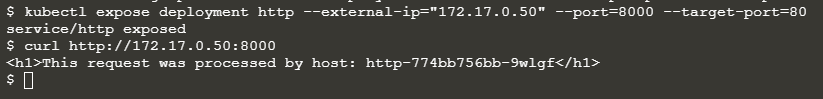
#### Step 3 - Kubectl Expose

Use the following command to expose the container port 80 on the host 8000 binding to the external-ip of the host.

kubectl expose deployment http --external-ip="172.17.0.50" --port=8000 --target-port=80

You will then be able to ping the host and see the result from the HTTP service.

curl http://172.17.0.50:8000

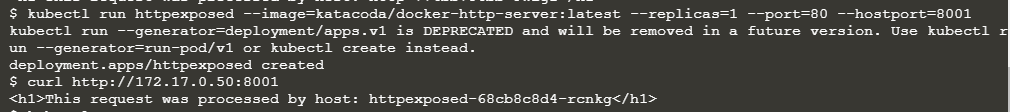


#### Step 4 - Kubectl Run and Expose

Use the command command to create a second http service exposed on port 8001.

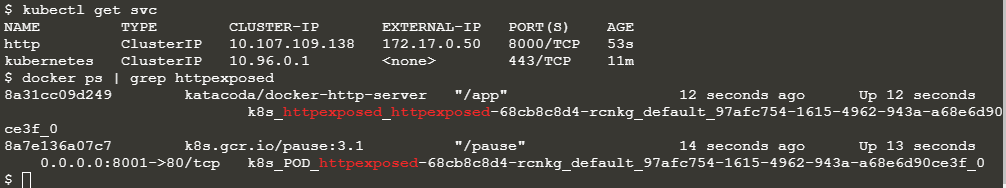
kubectl run httpexposed --image=katacoda/docker-http-server:latest --replicas=1 --port=80 --hostport=8001

You should be able to access it using curl http://172.17.0.50:8001



Under the covers, this exposes the Pod via Docker Port Mapping. As a result, you will not see the service listed using kubectl get svc

To find the details you can use docker ps | grep httpexposed

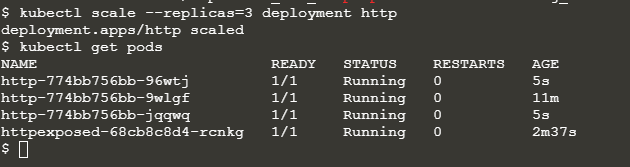


#### Step 5 - Scale Containers

The command kubectl scale allows us to adjust the number of Pods running for a particular deployment or replication controller.

kubectl scale --replicas=3 deployment http

Listing all the pods, you should see three running for the http deployment kubectl get pods

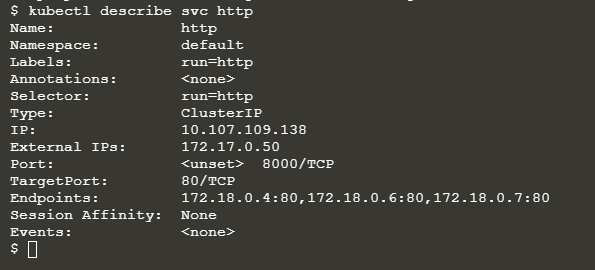


Once each Pod starts it will be added to the load balancer service. By describing the service you can view the endpoint and the associated Pods which are included.

kubectl describe svc http

Making requests to the service will request in different nodes processing the request.

curl http://172.17.0.50:8000



### 

### b**) Launch Single Node Kubernetes Cluster**

#### Step 1 - Start Minikube

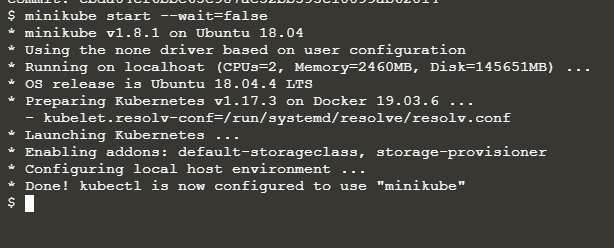
Minikube has been installed and configured in the environment. Check that it is properly installed, by running the minikube version command:

minikube version



Start the cluster, by running the minikube start command:

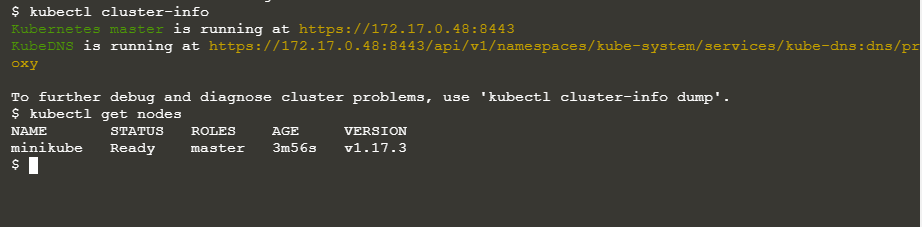
minikube start --wait=false



#### Step 2 - Cluster Info

Details of the cluster and its health status can be discovered via kubectl cluster-info

To view the nodes in the cluster using kubectl get nodes

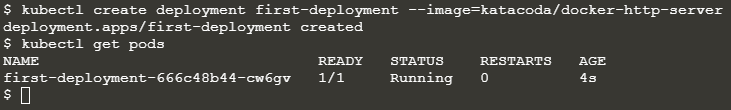


#### Step 3 - Deploy Containers

With a running Kubernetes cluster, containers can now be deployed.

Using kubectl run, it allows containers to be deployed onto the cluster - kubectl create deployment first-deployment --image=katacoda/docker-http-server

The status of the deployment can be discovered via the running Pods - kubectl get pods

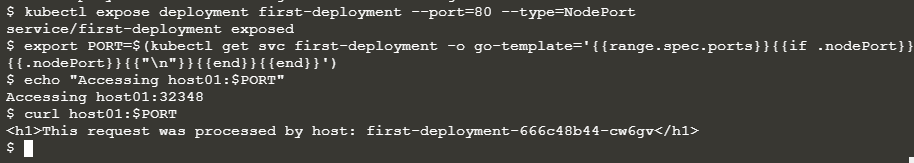


Once the container is running it can be exposed via different networking options, depending on requirements. One possible solution is NodePort, that provides a dynamic port to a container.

kubectl expose deployment first-deployment --port=80 --type=NodePort

The command below finds the allocated port and executes a HTTP request.

export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}{{.nodePort}}{{"\n"}}{{end}}{{end}}') echo "Accessing host01:$PORT" curl host01:$PORT

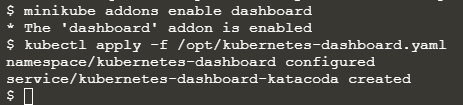


#### Step 4 - Dashboard

Enable the dashboard using Minikube with the command minikube addons enable dashboard

Make the Kubernetes Dashboard available by deploying the following YAML definition. This should only be used on Katacoda.

kubectl apply -f /opt/kubernetes-dashboard.yaml



To see the progress of the Dashboard starting, watch the Pods within the kube-system namespace using kubectl get pods -n kubernetes-dashboard –w

