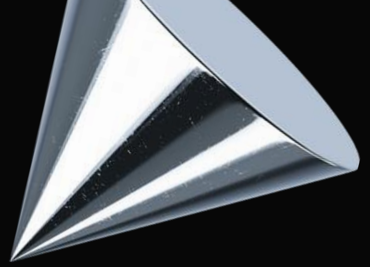
Assignments

Navdeep Kaur

Dockers Lab1



Duration - 20 mins

1. Run a container with below parameters

   Image Info

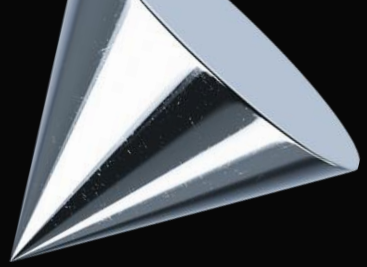
   Image: navjoy220161/python_flask:1.0.0

   Container Info

   Container name: python-c
   Mode: Detached
   Forward 8087 of host -> port 5000 of container

2. Check the output on host:8087
3. Now enter inside the container
   - rm the file main.py
4. Now stop and start the container
5. Access the page localhost:8087
6. Check container logs and status
7. Remove the container
8. Create the container again using the same command as in step1
9. Change the content of your local main.py file to display “Hello Students”
10. Replace the container’s main.py with your local edited main.py and check the page again

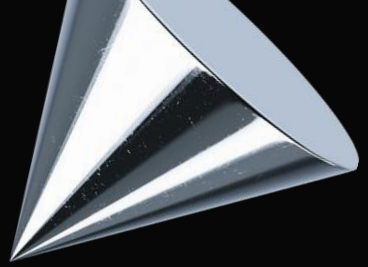
Dockers Lab2



Duration - 20 mins

1. Create a folder mypython-image and copy paste the main.py from python-flask-demo
2. Rename main.py to entry.py
3. Edit the content of entry.py to display “This is my python image”
4. Create a docker file with below steps
   from image navjoy220161/python_flask:1.0.0
   remove main.py from /python-flask
   copy entry.py created by you inside the container
   Install pandas library
   Finally run the command “python entry.py” as an entrypoint of the container
5. Create a DockerHub Account
6. Run docker login
7. Create an image from above docker file with tag mypython:1.0.0
8. Check your repository in dockerhub
9. Remove the image locally
10. Run a container from above image and expose the port 5000 on 8089 on host
11. Check localhost:8089

Pods Assignment



Duration - 20 mins

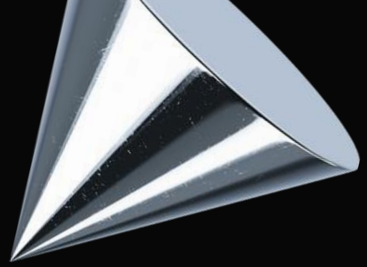
1. Create a new pod with the nginx image and name nginx-pod without using a yaml file.

2. Create a new pod using yaml file with below configuration :
   pod name : nginx-pod-1
   Label: type=loadbalancer, country=us
   Container name: nginx-container
   Container image: nginx, tag 1.21.1-alpine

3. There is a Pod named httpd-pod that is already in your machine. There is some issue while running that pod. Fix the issue and make it in Running state. Also change the Pod label to frontend
(Hint: get yaml file from running pod, delete existing pod, change yaml file and create new pod using kubectl create -f <name>.yaml)

4. There is a pod named myweb-app that is already running in your machine. That pod contains two containers - httpd-container and mongodb-container.
   >> Get the logs of container mongodb-container
   >> Go inside httpd-container and execute the command whoami

Replica Set Assignment



Duration - 20 mins

1. Create below replicaset using yaml file with below configuration :
   Replicaset name: nginx-rs
   Labels: type:loadbalancer-replica

   Container spec
   pod name : nginx-pod
   Label: type=loadbalancer, country=us
   Container name: nginx-container
   Container image: nginx, tag 1.21.1-alpine

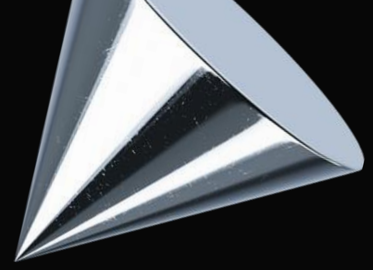
   Replicas: 3

2. Delete one of the pod created by replicaset and check the pod status

3. There is an existing replicaset named web-rs running in your machine. First check how many replicas it is running right now and scale it up to run 6 replicas

4. There is an existing replicaset named loadbalancer-rs running in your machine. Pod started by this replicas are in error state, fix the issue and make the pod in running state.

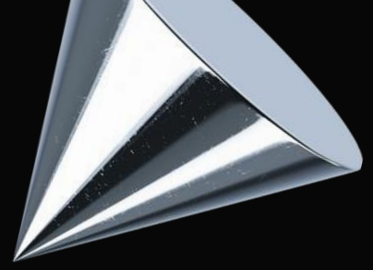
Namespace



Duration - 10 mins

1. Check all existing namespace in your machine
2. Checkout the pods that are running in namespace mars
3. Create a new namespace Jupiter and run a pod inside it with an image busybox
4. Check all the pods running across all the namespace

Deployment Assignment



Duration - 20 mins

1. Create a new deployment with below configuration :
   deployment name : nginx-deployment
   label: country=us
   Container name: nginx-container
   Container image: nginx, tag 1.21.1-alpine
   Container label: type=loadbalancer
   Replicas: 3

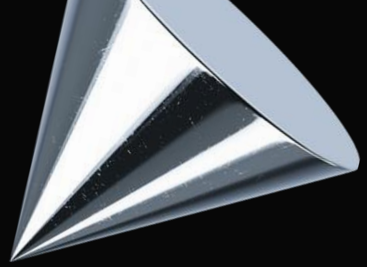
2. In above deployment , set the image to nginx-junk and check the status

3. Now check the history of all rollouts

5. Now rollback to previous version and check status

6. There is an existing deployment in your machine named myweb-dep whose pods are in error state.
   . First checkout all the revision of this deployment
   . Now rollback to a revision where image version was 2.4

ConfigMaps



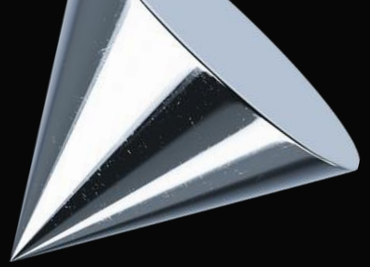
Duration - 20 mins

1. Create a configmap for a gaming app using command with below properties
   name: gameconfigmap
   data:
   enemies=aliens
   lives=3

2. Create a second configmap for the same app but this time using yaml.
   name: uiconfigmap
   data:
   theme=black
   badplayer=red
   goodplayer=purple

3. Create a pod defination and inject the configs from above two configMap in it.
   pod name – httpd-game-pod
   image – httpd
   `ENEMIES_TYPE` -> map config from gameconfigmap with key “enemies”
   `TOTAL_LIVES ->` map config from gameconfigmap with key “lives”
   Also Inject the whole uiconfigmap at one go

Secrets



Duration - 25 mins

1. Create a secret for the same gaming app using command with below properties

   name: dbsecret
   data:
   db_host=mongoDB
   db_user=root
   db_pass=mongo123

3. Now Create a second secret for the same gaming app using yaml file

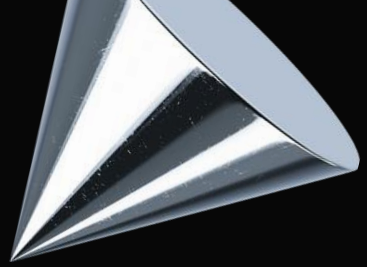
   name: gamesecret
   data:
   game_user= admin
   game_pass=admin123

3. Now create a pod with name httpd-pod and image httpd and inject secret

   MONGO_HOST -> db_host from secret dbsecret
   MONGO_USER -> db_user from secret dbsecret
   MONGO_PASS -> db_pass from secret dbsecret
   Inject gamesecret as it is

4. Modify the existing pod named httpd-pod and mount the gamesecret as a volume mounted at /opt/secret

Node-Mongo Project



Duration - 15 mins

Before doing this assignment, make sure your cluster is clean

1. Go to below location

   /Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo

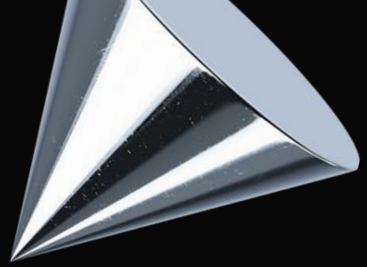
2. Run below command

```
kubectl create –f .
Kubectl rollout restart deployment mywebapp
```

3. Now explore your cluster for below answers

   1. How many deployment objects are there?
   2. Checkout webapp deployment- how many replicas, what is the image used?
   3. Which Service is exposing webapp deployment - what is the type, name and matching label of that service?
   4. Checkout mongodb deployment- how many replicas, what is the image used?
   5. Which Service is exposing mongodb deployment - what is the type, name and matching label of that service?
   6. Also check out configmap that got created and what is Mongo_host referred too here?
   7. In which deployment, configmap is injected?
   8. Is APP_port of configmap and webapp-service matching?

Persistent Volumes



Duration - 15 mins

1. Check out what is default storage class in your cluster.

   ```
   kubectl get storageclass
   ```

2. Describe storage class “gp2”

   ```
   kubectl describe storageclass gp2
   ```

3. Create a persistent volume claim using the storage class “gp2” and claim 10Gi. You can change below yaml file

   ```
   /Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo-persistence/mongo-pvc.yaml
   ```

4. Check your pvc

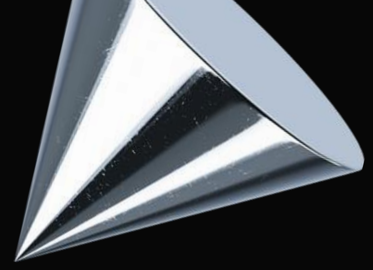
   ```
   kubectl get pvc
   ```

5. Now create mongodb-deployment using above persistent volume claim

   ```
   /Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo-persistence/mongodb-deployment.yaml
   ```

6. Describe your mongo pod check whether it is using above pvc?

   ```
   kubectl describe pod mongo-58c79fcb66-kqmwv
   ```



DaemonSet

Duration - 15 mins

1. Go to below location

   /Kubernetes_Fundamentals/k8s_operations/DaemonSet

2. Check out the dameonset.yaml
3. Did you notice there is no replicas in the spec part

4. Create the Daemonset

```
Kubectl create –f daemonset.yaml
```

5. Check out the pod

```
Kubectl get pod
```

7. Did you noticed, you got exactly two replicas because if you check you have two worker machine

8 . Check out NODE in result of above command

9 . Check out node of your cluster

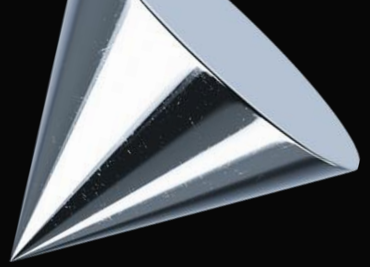
```
Kubectl get node
```

10 . Check out pod and node on which they are allocated

```
Kubectl get pod
```

11. Did you noticed, you got exactly two replicas on two different nodes

Canary Deployment



Duration - 15 mins

1. Create your first deployment with below configuration :
   deployment name : webserver-deployment
   label:
      app: frontend
      version: v1
   image: httpd
   Replicas: 3

2. Above deployment is exposed on port 80. Add a NodePort service on nodePort 30000 to access above deployment

3. Let's hit the service multiple times

```
for ((i=1;i<=10;i++)); do curl 192.168.49.2:30000; done
```

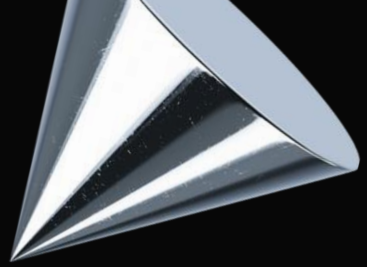
4. Now you want to deploy a new version with different image. You want to deploy it using canary approach.
   Create a second deployment such that 25% of traffic comes to the new version and the rest 75% goes to older version v1
   Image : nginx
   Replicas: 1

5. Let's hit the service multiple times again

```
for ((i=1;i<=10;i++)); do curl 192.168.49.2:30000; done
```

Hint: Use labels in such a way that existing service starts sending traffic to new deployment as well

Blue-Green Deployment



Duration - 15 mins

1. Create your first deployment with below configuration :
   deployment name : webserver-deployment-blue
   label:
       version: blue
   image: httpd
   Replicas: 3

2. Above deployment is exposed on port 80. Add a NodePort service on nodePort 30000 to access above deployment

3. Let’s hit the service multiple times

```
for ((i=1;i<=10;i++)); do curl 192.168.49.2:30000; done
```

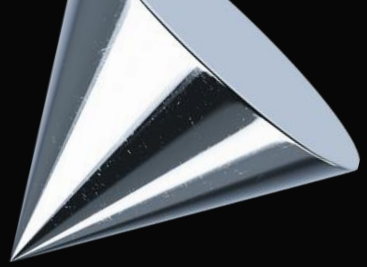
4. Now you want to deploy a new version. You want to deploy it using blue/green approach.
   Create a second deployment with same number of replicas as your existing deployment
   deployment name : webserver-deployment-green
   label:
       version: green
   image: nginx
   Replicas: 3

5. Adjust the labels of your existing service so that it start moving traffic at once to new deployment

6. Let’s hit the service multiple times again

```
for ((i=1;i<=10;i++)); do curl 192.168.49.2:30000; done
```

Taints & Tolerations



Duration - 15 mins

1. Run a pod on your machine using below command & make sure pod comes in running state

   ```
   kubectl run nginx-pod - -image nginx
   ```

2. Now taint your machine with below taint

   ```
   kubectl taint node minikube env=prod:NoExecute
   ```

3. Now check out the status of pod that you ran in step1

   ```
   kubectl get pod
   ```

4. You might have noticed that pod is in pending status. Check out why by looking at the event section.

   ```
   kubectl describe pod nginx-pod
   ```

5. Now, run a new pod with image “httpd” using yaml file and add matching toleration to run it on tainted machine.

6. Check out the status of above pod. It should be in running status.

7. Let’s add a new untainted machine(node) in your minikube cluster so that pod that was missing the toleration should also start running on this new machine

   ```
   minikube node add
   ```

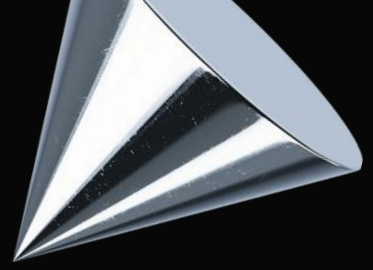
8. Check the status of that nginx-pod that was in pending state. Did it start running, check out on which machine

   ```
   kubectl get pod –o wide
   ```

9. Finally don’t forget to untaint your machine

   ```
   kubectl taint node minikube env=prod:NoExecute-
   ```

Node Affinity / Anti-Affinity



Duration – 15 mins

Please note: In the last assignment, you already added one more machine in your minikube cluster. Make sure you have two machines and none of them is tainted.

1. Create a pod with an image mongo. Add a node affinity section to make it run on a machine that has label “size=large”

2. Check out the pod status and describe and look out for the reason , why are they in pending state?

3. You might find out that it must be searching for machine with the matching label. Let’s label our only machine

```
kubectl label node minikube size=large
```

4. Check out the pod status and did they start running? Also check did they started on the labelled machine?

```
kubectl get pod –o wide
```

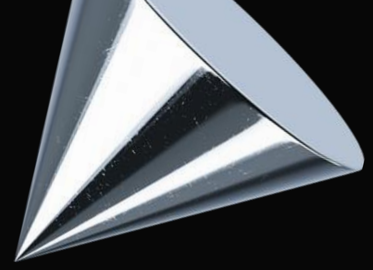
5. What will happen to a pod that does not have node affinity added? Will it run on labelled or unlabelled machine? Try this out, create a deployment with 6 replicas , figure out where those 6 replicas are started?

   Did you noticed, replicas are distributed on both machines

6. Modify the above deployment to add anti affinity rules against the minkube master node that is labelled “size=large”. Restart the deployment and check out the pod again, where are they started?

   Did you noticed, replicas are placed on machine which is not labelled

Pod Affinity/Anti-Affinity

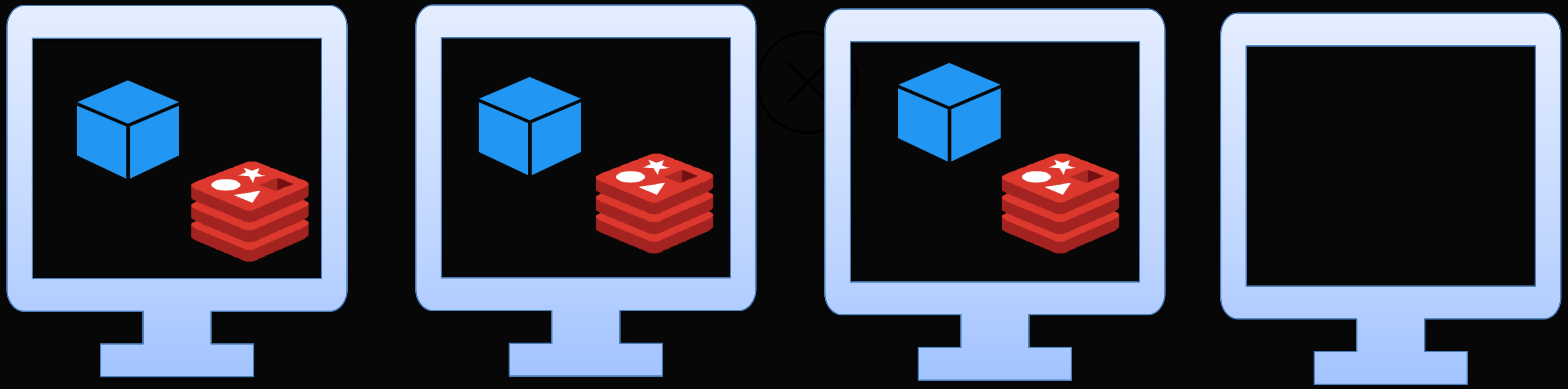


Duration - 15 mins

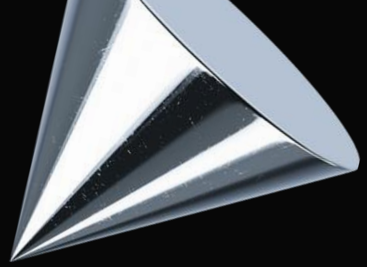
1. We have a frontend app with image “httpd” and cache app with image “redis”

2. frontend pod should not be co-located and neither redis-pods should be co-located.

3. However, Frontend should be deployed alongside redis pod to reduce latency



Stateful Sets



Duration - 15 mins

1. Create a StatefulSet for Mysql database with 3 replicas

2. In your stateful set, use persistence volume claim template to create pvc for 1 GB each using default storage class

3. Run your Stateful set, check the status of running pods

4. Check out what all things are created in cluster – Pods, replicasets, services and pvc that is tied to each pod

5 Delete first pod and see whether name and pvc is attached back

6 Try scaling up replicas to 5 , check out the order in which they are started, also check out corresponding pvc

7 Try scaling down replicas to 6 , check out the order in which they are scaled down and also check corresponding pvc

StatefulSets

Using Kubernetes Operator/ Helm

Duration - 15 mins

Create a StatefulSet for promotheus/Grafana stack using Kubernetes Operator/Helm