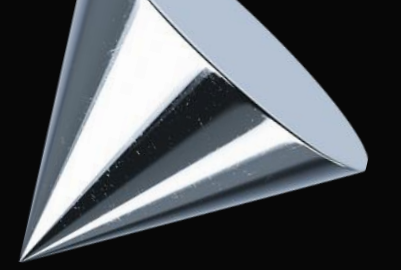


Assignments

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`

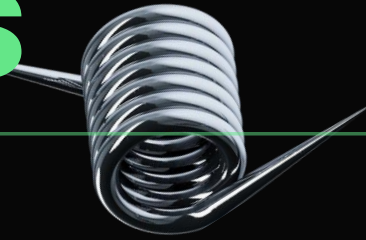
Forward 8087 of host -> port 5000 of container

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

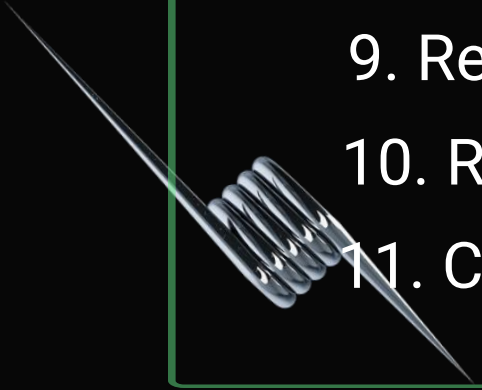
Dockers Lab2



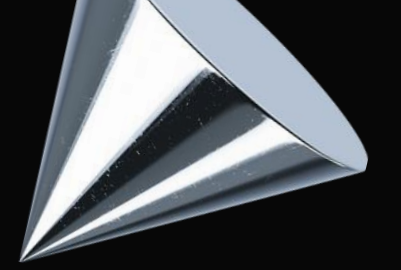
Duration - 20 mins



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 libraryFinally 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


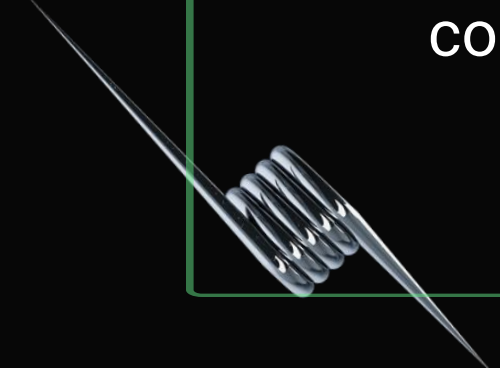


Pods Assignment

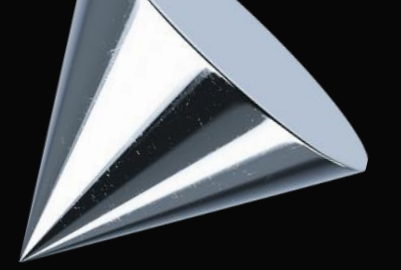




Duration - 20 mins

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

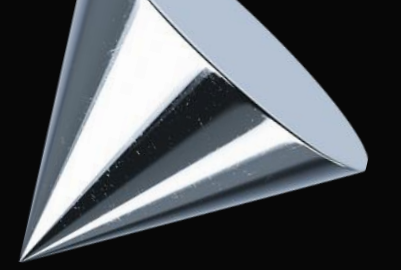
Container name: nginx-container

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.

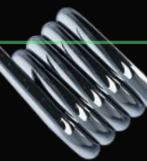
Namespace



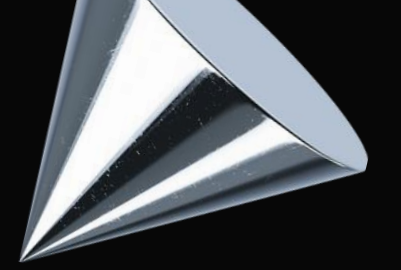


Duration - 10 mins

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


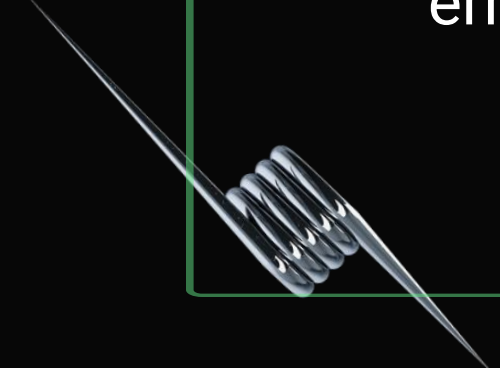


Deployment Assignment





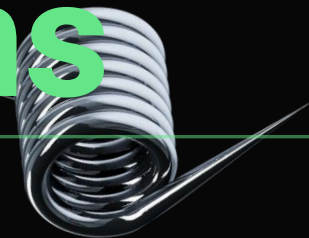
Duration - 20 mins


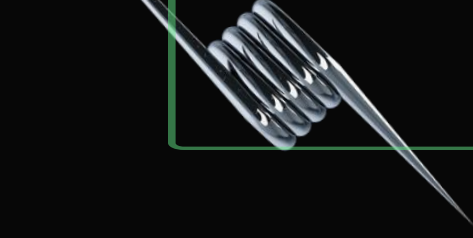
1. Create a new deployment with below configuration :
deployment name : nginx-deployment
label: country=us
Container name: nginx-container
Container image: nginx, tag 1.21.1-alpine
Container label: type=loadbalancer
Replicas: 3
 2. In above deployment , set the image to nginx-junk and check the status
 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

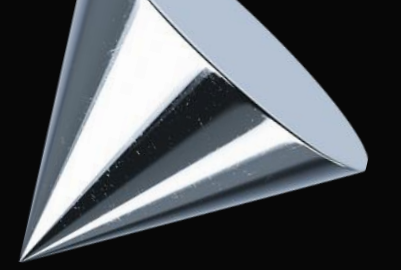


Duration - 20 mins



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 definition 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

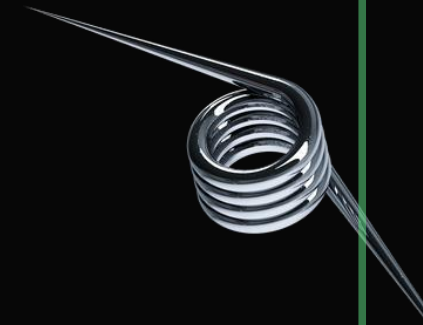
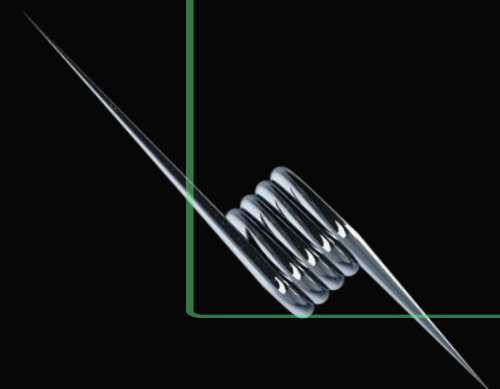
4. Modify the existing pod named httpd-pod and mount the gamesecret as a volume mounted at /opt/secret

Taints & Tolerations/ Node Affinity

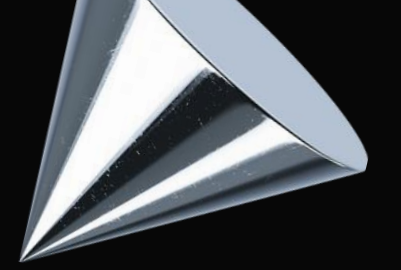




Duration - 25 mins

1. There are two pods httpd-test and httpd-prod.
 2. First make sure that httpd-test does not run on docker-desktop by adding taint on node "env:prod"
 3. Then make sure that httpd-prod does is able to run on docker-desktop by adding toleration for taint "env:prod"
 4. Also label your docker desktop with label "gpu:yes"
 5. Add affinity in httpd-prod for a node that provides gpu capabilitiy
 6. Check the status of both the pods
- 
- 

Node-Mongo Project



Duration - 15 mins



Before doing this assignment, make sure your cluster is clean

1. Go to below location

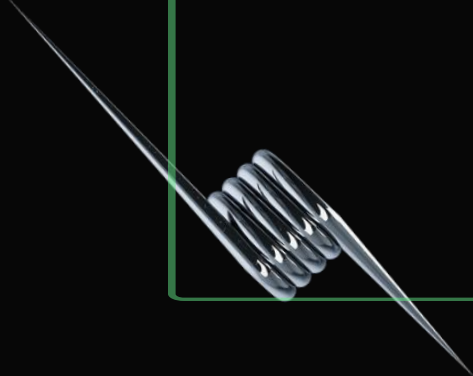

`/Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo`

2. Run below command

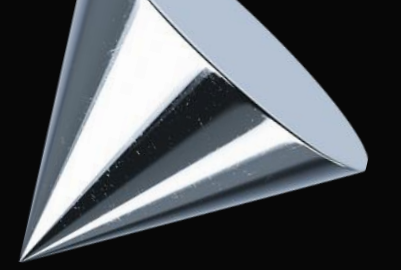
`kubectl create -f .`

`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

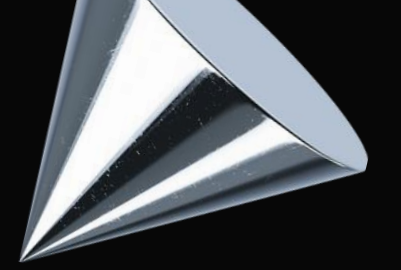
```
/Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo-persistence/mongodb-deployment.yaml
```

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

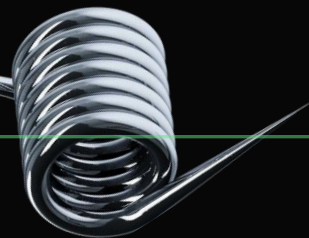
```
kubectl describe pod mongo-58c79fcb66-kqmwv
```



Blue-Green Deployment



Duration - 15 mins



1. Go to below location

[/Kubernetes_Fundamentals/k8s_operations/deployment-strategy/blue-green](#)

2. There is a start script to run node-mongo application.

`chmod + x start-script.sh`

`./start-script.sh`

3. Above script has created a service of type loadbalancer

`kubectl get svc`

4. Copy paste the External IP of above loadbalancer and open your browser and checks how application looks like (it may take 2-3 mins)

5. Now create a new deployment with newer version. Use below deployment file

[/Kubernetes_Fundamentals/k8s_operations/deployment-strategy/blue-green/blue-deployment.yaml](#)

6. You can expose above deployment using a new service and give it to some team for analysis

[/Kubernetes_Fundamentals/k8s_operations/deployment-strategy/blue-green/blue-service.yaml](#)

7. Everyone is happy with newer version. Go ahead and change the label of your production loadbalancer to forward the traffic to blue deployment.

[/Kubernetes_Fundamentals/k8s_operations/deployment-strategy/blue-green/green-service.yaml](#)

Hint: Match Labels of your blue-deployment and green service

8. Open Browser and check the production load balancer again. Did you see the change?

Canary Deployment



Duration - 15 mins



1. Go to below location

`/Kubernetes_Fundamentals/k8s_operations/deployment-strategy/canary`

2. There is a start script to run node-mongo application. Before running below script change service type of webapp-service to LoadBalancer

`chmod + x start-script.sh`

`./start-script.sh`

3. Above script has created a service of type loadbalancer

`kubectl get svc`

4. Now create a new deployment using the same label as service label, set replicas to 2

`Kubernetes_Fundamentals/k8s_operations/deployment-strategy/canary/new-web-deployment.yaml`

5. Scale down the old deployment to replicas 2


6. Open the browser and check your loadbalancer. Half of the traffic should go to new deployment and half of the traffic should go to old deployment

StatefulSets



Duration - 15 mins

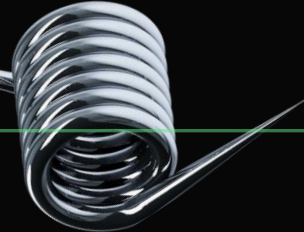


1. Go to below location
[/Kubernetes_Fundamentals/k8s_operations/deployment/node-mongo-stateful](#)
 2. Check out mongodb-headless-service.yaml and create using kubectl create statement
 2. Run storage_pvc.yaml to create the storage class used in pvc template of statetful Set
[Kubectl cr](#)
 3. Checkout the mongodb-statefulset.yaml and create using kubectl create statement
 - Check out pvc that is tied to each pod
 - Delete mongo pod and see whether name and pvc is attached back
 - Check out mongo pod name for sticky identity
- 

DeamonSet



Duration - 15 mins



1. Go to below location

[/Kubernetes_Fundamentals/k8s_operations/DaemonSet](#)

2. Check out the daemonset.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