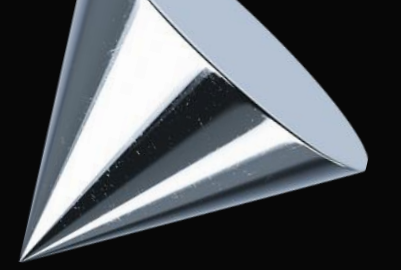


# Assignments


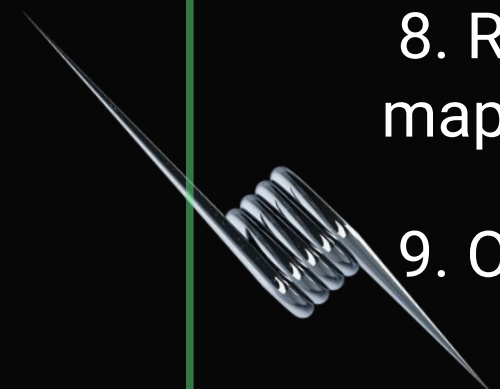
**Navdeep Kaur**

# Dockers Lab1

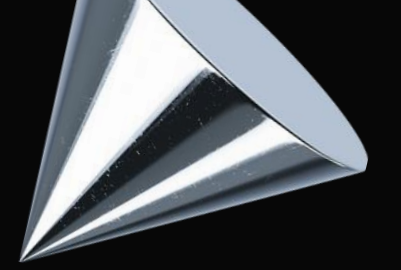




# Duration - 20 mins


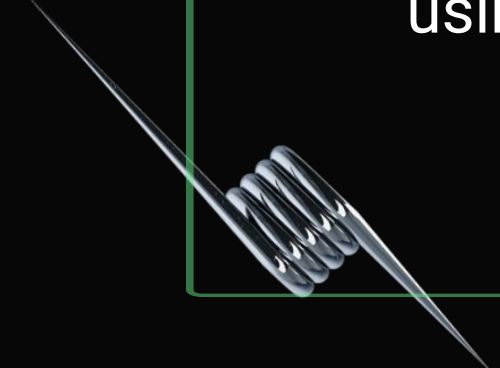
1. `git clone https://github.com/technoavengers/docker_fundamentals.git`
  2. Go to python-flask-demo folder
  3. Create a docker build with a tag "<your repo>/python-flask:1.0.0"
  4. Perform docker login
  5. Now push the above image
  6. Check your repository in dockerhub
  7. Remove the image from your local machine
  8. Run the container from your pushed image, check the exposed port in docker file and map it to some port on your localhost
  9. Open your browser and run localhost:<port>
- 
- 

# Pods Assignment

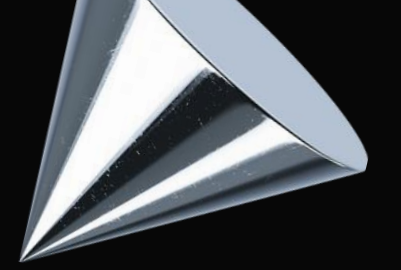




# Duration - 20 mins

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. Create a new pod by running below command  
`kubectl run httpd-pod --image httpd-new`
  4. There is some issue in the above Pod. Fix the issue in above pod and apply the changes.
- (Hint: get yaml file from running pod, delete pod, change yaml file and apply the changes using `kubectl apply -f <name>.yaml`)
- 
- 

# 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

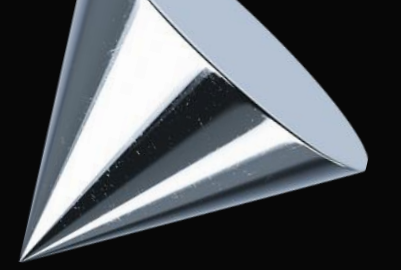
Container name: nginx-container

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. Add one more replica in the replicaset and check the pods



# Deployment Assignment







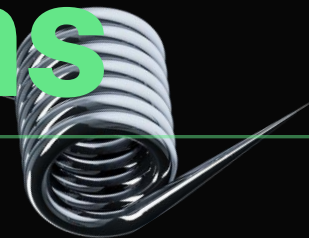
# Duration - 20 mins


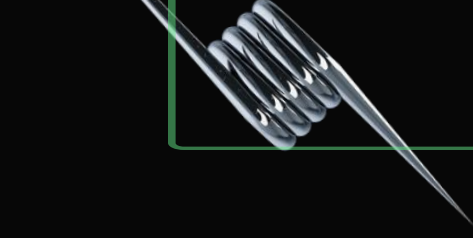
1. Create a new deployment with below configuration :  
deployment name : nginx-deployment  
label: country=us  
Container name: nginx-container  
Container image: nginx, tag 1.21.1-alpine  
Container label: type=loadbalancer  
Replicas: 3
  2. Edit the deployment and change its update Strategy to "Recreate"
  3. In above deployment , set the image to nginx-junk and check the status
  4. Now check the history of all rollouts
  5. Now rollback to previous version and check status
- 
- 

# ConfigMaps

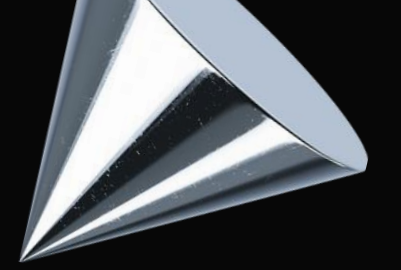


# Duration - 20 mins



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
- 
- 