CLOUDETHIX

Que 1 →

- Create 2 Public Docker Hub registries named cloudethix_master_nginx_yourname & cloudethix_release_nginx_yourname.

- Clone below repository on your system.
  https://github.com/zembutsu/docker-sample-nginx.git

- Initialize a local repository & copy the code from above repo to your local repository in master branch and then create below branches.
  release
  main
  hotfix

- Once code is copied to local repository,from master branch update the index.html and add word "Cloudethix Master Branch Nginx" and build the docker image & add meaningful tags and push to Docker Hub registry cloudethix_master_nginx_yourname.

- Also from release branch update the index.html and add word "Cloudethix Release Branch Nginx" and build the docker image & add meaningful tags and push to Docker Hub registry cloudethix_release_nginx_yourname.

- Once Images are copied to Docker hub registries, switch to the main branch.

- In main branch create directory named kube/clusterIP & inside kube directory create file named master_pod.yaml with pod name master_nginx & with label master_nginx & add image that you have pushed in Docker Hub registry cloudethix_master_nginx_yourname.

- Also create a file release_pod.yaml with pod name release_nginx & with label release_nginx & add image that you have pushed in Docker Hub registry cloudethix_release_nginx_yourname.

- Create a file called cluster_ip-service.yaml with service name cloudethix_clusterip and with Type clusterIP.

- Then, select the pod with label release_nginx in service.

- Create all these three resources in your k8s cluster.
- Now, access master_nginx pod shell & curl the master_nginx pod & check the result.
- Also try to curl release_nginx pod with DNS name & check the result.
- Then curl the clusterip service with its name and check the result.
- Finally, create a GITHUB remote repository named cloudethix-k8s-yourname and push all the branches to the remote repository.
- Take all screenshots and create a well formatted document.

Que 2 →

- In the main branch of your local repository create a directory kube/NodePort.
- Create below files from below url. Please make sure you will create NodePort service with port 30008 instead of loadbalancer.

  https://kubernetes.io/docs/tasks/access-application-cluster/connecting-frontend-backend/.

  backend-deployment.yaml
  backend-service.yaml
  frontend-deployment.yaml
  *frontend-**NodePort**-service.yaml*

- Once files are created , create all the resources in your k8s cluster.
- Access all public ips with port 30008 in the browser and then check the result.
- Finally, push all the latest code to the remote repository.
- Take all screenshots and create a well formatted document.

Que 3 →

- Create any 2 pods and assign them to different worker nodes with nodeName property.

Que 4 →

- Label both worker nodes such as worker-0 node as cloudethix-k8s-00 & worker-1 node as cloudethix-k8s-01.
- Once nodes are labeled, create pod00.yaml file and schedule the pod on worker-0 node with nodeSelector property. Also create one more file named pod01.yaml & schedule the pod on worker-1 node.

Que 5 →

- Clone the below repo locally & create DaemonSet from directory DaemonSet101.

  https://github.com/collabnix/kubelabs

Que 6 →

- Create a static pod with name cloudethix-static in your k8s cluster. Refer below link.

  https://kubernetes.io/docs/tasks/configure-pod-container/static-pod/

Que 7 →

- Install Kubectx & kubens in your k8s cluster.

Que 8 →

- Create 1 Public Docker Hub registry named flask_webapp_yourname.

- Clone below repository on your system.
  https://github.com/mmumshad/simple-webapp-docker.git

- Initialize a local repository & copy the code from above repo to your local repository in your working branch.

- Once code is copied to the local repository, build the docker image & add meaningful tags with version 1 and push to Docker Hub registry.

- Once Images are pushed to Docker hub registries, create a directory named kube. Inside the kube directory create deployement.yaml file with 3 replication , labels app: flask-webapp , containerPort: 8080 and add the image that you have pushed in Docker Hub registry.

- Create one service.yaml file with type nodeport & select flask-webapp with port 8080 & targetPort 8080 with any nodePort between range 30000-32768.

- Once a service is created try accessing the web page in the browser as below. (30011 is nodeport mentioned in service.yaml). Meanwhile open app.py from your code to understand paths & output.

  http://master_ip:30011/
  http://master_ip:30011/how are you

- Now , update the app.py from your code and add below route above if __name__ == "__main__" line

```
@app.route('/Who are you')
def cloudethix():
    return 'Yes, I am cloudethix, and You !!!'
```

- Once the file is updated , rebuild the docker image & add meaningful tags with version 2 and push to Docker Hub registry.

- Now we have the latest docker image in repo, It's time to roll out a new image. Roll out the new Image with all three ways one by one.

  1. ***With kubectl set command***
  2. ***With kubectl edit deployment***
  3. ***With deployment.yaml file modification.***

- Run the # kubectl rollout command to check status and history.

- Note:- Once above step 1 is done , run # kubectl rollout undo deployment command to rollback the change and then try a second way of rollout.

- In the browser run all three routes & notice the changes.

  http://master_ip:30011/
  http://master_ip:30011/how are you
  http://master_ip:30011/Who are you

- Once done with all above steps , commit all the changes to the remote repository.

- Capture the snap and prepare a well formatted document.

Que 9 →

- Download and install Lens & access your k8s cluster from Lens.
- Create nginx Pod and Nodeport service. Check the Pod logs from Lens.
- Check the service from lens. Also login to the pod shell using the lens.
- Take snaps and delete the resources that you have just created.

Que 10 →

- Create 1 Public Docker Hub registry named cloudethix_configmap_yourname.

- Clone below repository on your system.
  https://github.com/zembutsu/docker-sample-nginx.git

- Initialize a local repository & copy the code from above repo to your local repository in the working branch.

- Once code is copied , build a docker image from docker file and add meaningful tags and push to docker hub repository.
- Once Images are pushed to Docker hub registries, create a directory named kube. Inside the kube directory create deployement.yaml file with 3

replication , labels app: frontend-webapp , containerPort: 80 and add the image that you have pushed in Docker Hub registry.

- Create one service.yaml file with type nodeport & select frontend-webapp pod with port 80 & targetPort 80 with any nodePort between range 30000-32768.

- Once the service is created try accessing the web page in the browser as below. Notice the changes & take the snap.

- Now create a configmap.yaml file with below data & delete the deployment that you have created.

```
<html>
  <body>
    <h1>  I am Cloudethix Team, Are you ?!! </h1>
    Version: 1.1
  </body>
</html>
```

- Then update the same deployment.yaml file and mount configmap as volume on container using volumeMounts with mountPath /usr/share/nginx/html/

- Now it's time to create configmap & deployment. Once created , try to access the webpage in the browser & confirm that the index page is the same as we have in configmap.

Que 11 →

- Create 1 Public Docker Hub registry named cloudethix_multicontainer_yourname.

- Clone below repository on your system. https://github.com/janakiramm/Kubernetes-multi-container-pod.git

- Initialize a local repository & copy the code from above repo to your local repository in any of your working branches.

- Once code is copied , go to the Build directory and build docker image from docker file and add meaningful tags and push to docker hub repository.
- Now go to the deploy directory and notice the files.

- Here, web-pod-1.yml file will create the pod with two containers (Multi container). Take a note of lables , name of containers and ports. Also, please make sure you will update the python container image that you have pushed to your docker registry.

- Now, open web-svc.yml file and notice service Type , selectors & targetPort. Apply the file.

- Now open db-pod.yml & notice the lables , name , Image, containerPort and apply the file.

- Now open the db-svc.yml file and notice service Type , selectors & targetPort. Apply the file.

- Once above files are applied , Check that the Pods and Services are created using command line or lens.

- Now , from the command line run below urls & notice the changes.

    # curl http://$NODE_IP:$NODE_PORT/init
    Initialize the database with sample schema

- Now it's time to Insert some sample data. Make sure you will use correct $NODE_IP:$NODE_PORT

    # curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"John Doe"}' http://$NODE_IP:$NODE_PORT/users/add

    # curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Jane Doe"}' http://$NODE_IP:$NODE_PORT/users/add

    # curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "3", "user":"Bill Colls"}' http://$NODE_IP:$NODE_PORT/users/add

    # curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "4", "user":"Mike Taylor"}' http://$NODE_IP:$NODE_PORT/users/add

- Now access the data that we have added to database using below command.
    # curl http://$NODE_IP:$NODE_PORT/users/1

- The second time you access the data, it appends '(c)' indicating that it is pulled from the Redis cache.

- Also, try to access mysql shell i.e db pod & run select * from the users table. check app.py for DB related information.

- Prepare proper documentation in brief & write start to end flow. Refer below link if you face any issues.
  https://github.com/janakiramm/Kubernetes-multi-container-pod

Que 12 →

- Create 1 Public Docker Hub registry named cloudethix_Initcontainer_yourname.

- Clone below repository on your system.
  https://github.com/janakiramm/simpleapp.git

- Initialize a local repository & copy the code from above repo to your local repository in any of your working branch.

- Once code is copied , go to the Build directory and build docker image from docker file and add meaningful tags and push to docker hub repository.

- Once Images are pushed to Docker hub registries, create a directory named kube. Inside the kube directory create deployement.yaml file with 3 replication , label app: simpleapp-webapp , containerPort: 80 and add the image that you have pushed in Docker Hub registry.

- Create one service.yaml file with type nodeport & select simpleapp-webapp pod with port 80 & targetPort 80 with any nodePort between range 30000-32768.

- Open the webpage in the browser and notice the changes and capture the snap.

- Then delete the deployment that you have just created.

- Update the deployment.yaml file and add volumeMounts with mountPath /usr/share/nginx/html from emptyDir: {} volume.

- Once above changes are added, add initContainers block with below parameters. Also add volumeMounts for Init Container with mountPath "/work-dir" from emptyDir: {} volume.

```
initContainers:
- name: install
  image: busybox:1.28
  command:
  - wget
  - "-O"
  - "/work-dir/index.html"
  - http://info.cern.ch
  volumeMounts:
  - name: workdir
    mountPath: "/work-dir"
```

- Add volumes with emptyDir: {} in deployment.yaml file.

- Once the deployment.yaml file is ready, create the deployment & access the page in the browser and notice the changes.

- Prepare a well formatted document and write your understanding step by step.

Que 13 →

- Create 1 Public Docker Hub registry named cloudethix_hpa_yourname.

- Clone below repository on your system.
  https://github.com/vivekamin/kubernetes-hpa-example.git

- Initialize a local repository & copy the code from above repo to your local repository in any of your working branch.

- Once code is copied , build a docker image from the docker file and add meaningful tags and push to the docker hub repository.

- Once the image is pushed, go to k8s directory and update deployment.yaml file with image name from your repo. And then create it.

- Open service.yml and change the type to nodePort and apply the same.

- Open the HPA.yaml file, notice it and then apply the same.
- Open the browser, and access the webpage.
- Now it's time to test the HPA working with the below command.

  # kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://NODE_PORT_SERVICE_NAME; done"

- Check the HPA from kubectl command and also check if the deployment is scaling up.
- Take the snap , prepare a well formatted doc and write your understanding.

Que 14 →

- Create 1 Public Docker Hub registry named cloudethix_cronjob_yourname.
- Initialize a local repository & copy below code (three files) to your local repository in any of your working branch.
- Once code is copied, build the docker image from Dockerfile , add meaningful tags and then push the docker image to Docker hub registry.
- Now update the pythoncronjob.yml file to change the image name that you have just pushed to docker hub registry.
- Now create a cron job using pythoncronjob.yml file. Check with kubectl command if the cron job is created.
- Check the Job name which is created by cronjob from command line or lens.
- Then check the pod logs which are created by the job and capture the output.
- Prepare well formatted documents and write your understanding.

 # vim helloworld.py

```
#!/usr/local/bin/python3
import datetime
x = datetime.datetime.now()
print("Welcome to the Cloudethix World")
print("Today is")
print(x)
```

 # vim Dockerfile

```
FROM python:3.7-alpine
#add user group and ass user to that group
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
#creates work dir
WORKDIR /app
#copy python script to the container folder app
COPY helloworld.py /app/helloworld.py
RUN chmod +x /app/helloworld.py
#user is appuser
USER appuser
ENTRYPOINT ["python", "/app/helloworld.py"]
```

 # vim pythoncronjob.yml

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: python-helloworld
spec:
  schedule: "*/1 * * * *""
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: python-helloworld
            image: python-helloworld
            command: [/app/helloworld.py]
          restartPolicy: OnFailure
```