Kubernetes Assignment 01

1. List all the namespaces in the cluster
   kubectl get namespace

2. List all the pods in all namespaces
   kubectl get pods --all-namespaces

3. List all the pods in the particular namespace
   kubectl get pods --namespace <nsname>

4. List all the services in the particular namespace
   kubectl get services --namespace <nsname>

5. List all the pods showing name and namespace with a json path expression
   kubectl get pods -o=jsonpath="{.items[*][‘metadata.name’ , ‘metadata.namespace’]}"

6. Create an nginx pod in a default namespace and verify the pod running
   kubectl -- run nginx --image=nginx -n default

7. Create the same nginx pod with a yaml file
   kubectl create -f nginxpod.yaml

8. Output the yaml file of the pod you just created
   kubectl get deploy nginx-deployment -o yaml

9. Output the yaml file of the pod you just created without the cluster-specific information
   kubectl get pod nginx -o yaml --export (deprecated)

10. Get the complete details of the pod you just created
    kubectl describe pod nginx

11. Delete the pod you just created
    kubectl delete pod nginx

12. Delete the pod you just created without any delay
    kubectl delete pod nginx --grace-period=0 --force

13. Create the nginx pod with version 1.17.4 and expose it on port 80
    kubectl run nginx -- image=nginx-pod -- port=80 - -expose

14. Change the Image version to 1.15-alpine for the pod you just created and verify the image version is updated
    kubectl set image pod/nginx nginx=nginx:1.15-alpine

15. Change the Image version back to 1.17.1 for the pod you just updated and observe the change
    kubectl set image pod/nginx nginx=nginx:1.17.1
    kubectl describe pod nginx
    kubectl get pod nginx -w

16. Check the Image version without the describe command
    kubectl get pod nginx -o jsonpath='{.spec.containers[].image}{"\n"}'

17. Create the nginx pod and execute the simple shell on the pod
    kubectl -- exec -it nginx bash

18. Get the IP Address of the pod you just created
    kubectl describe pod nginx

19. Create a busybox pod and run command ls while creating it and check the logs
    kubectl run busybox --image=nginx -n default
    minikube kubectl logs busybox

20. If pod crashed check the previous logs of the pod
    kubectl logs --container < container_name > --previous=true < pod_name >

21. Create a busybox pod with command sleep 3600
    kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c "sleep 3600"

22. Check the connection of the nginx pod from the busybox pod
    kubectl exec -it busybox -- wget -o- 127.0.0.4

23. Create a busybox pod and echo message ‘How are you’ and delete it manually
    kubectl run busybox --image=nginx --restart=Never -it -- echo "How are you"
    kubectl delete pod busybox

24. Create a busybox pod and echo message ‘How are you’ and have it deleted immediately
    kubectl run busybox --image=nginx --restart=Never -it --rm -- echo "How are you"

25. Create an nginx pod and list the pod with different levels of verbosity
    kubectl run busybox --image=nginx --restart=Never --v=7

26. List the nginx pod with custom columns POD_NAME and POD_STATUS

kubectl get po -o=custom-columns="POD_NAME:.metadata.name,
POD_STATUS:.status.containerStatuses[].state"

27. List all the pods sorted by name
    kubectl get pods --sort-by=.metadata.name

28. List all the pods sorted by created timestamp
    kubectl get po --sort-by=.status.startTime

29. Create a Pod with three busy box containers with commands “ls; sleep 3600;”, “echo Hello World; sleep 3600;” and “echo this is the third container; sleep 3600” respectively and check the status
    Kubectl create -f multic.yaml

30. Check the logs of each container that you just created
    kubectl logs busybox -c busybox1

31. Check the previous logs of the second container busybox2 if any
    kubectl logs busybox -c busybox2 --previous=true

32. Run command ls in the third container busybox3 of the above pod
    kubectl exec busybox -c busybox3 -- ls

33. Show metrics of the above pod containers and puts them into the file.log and verify
    minikube addons enable metrics-server
    kubectl top pod busybox --containers
    kubectl top pod busybox --containers > file.log
    cat file.log

34. Create a Pod with main container busybox and which executes this “while true; do echo ‘Hi I am from Main container’ >> /var/log/index.html; sleep 5; done” and with sidecar container with nginx image which exposes on port 80. Use emptyDir Volume and mount this volume on path /var/log for busybox and on path /usr/share/nginx/html for nginx container. Verify both containers are running.
    kubectl create -f Q35.yaml
    Kubectl get pods
    kubectl exec -it multi-cont-pod -c main-container -- sh
    cat /var/log/index.html
    kubectl exec -it multi-cont-pod -c sidecar-container -- sh
    cat /usr/share/nginx/html/index.html

35. Exec into both containers and verify that main.txt exist and query the main.txt from sidecar container with curl localhost
    kubectl exec -it multi-cont-pod -c sidecar-container -- /bin/bash

curl localhost

36. Get the pods with label information
    kubectl get pods --show-labels

37. Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev
    kubectl run nginx-prod1 --image=nginx --restart=Never --labels=env=prod
    kubectl run nginx-prod2 --image=nginx --restart=Never --labels=env=prod
    kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
    kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev
    kubectl run nginx-dev3 --image=nginx --restart=Never --labels=env=dev

38. Verify all the pods are created with correct labels
    kubectl get pods --show-labels

39. Get the pods with label env=dev
    kubectl get pods -l env=dev

40. Get the pods with label env=dev and also output the labels
    kubectl get pods -l env=dev --show-labels

41. Get the pods with label env=prod
    kubectl get pods -l env=prod

42. Get the pods with label env=prod and also output the labels
    kubectl get pods -l env=prod --show-labels

43. Get the pods with label env
    kubectl get pods -l env

44. Get the pods with labels env=dev and env=prod
    kubectl get pods -l 'env in (dev,prod)'

45. Get the pods with labels env=dev and env=prod and output the labels as well
    kubectl get pods -l 'env in (dev,prod)' --show-labels

46. Change the label for one of the pod to env=uat and list all the pods to verify
    kubectl label pod/nginx-dev1 env=uat --overwrite

47. Remove the labels for the pods that we created now and verify all the labels are removed
    kubectl label pod nginx-dev{1..3} env-
    kubectl label pod nginx-prod{1..2} env-

kubectl get pods --show-labels

48. Let's add the label app=nginx for all the pods and verify
    kubectl label pod nginx-dev{1..3} app=nginx
    kubectl label pod nginx-prod{1..2} app=nginx
    kubectl get pod --show-labels

49. Get all the nodes with labels (if using minikube you would get only master node)
    kubectl get nodes --show-labels

50. Label the node (minikube if you are using) nodeName=nginxnode
    kubectl label node minikube nodeName=nginxnode

51. Create a Pod that will be deployed on this node with the label nodeName=nginxnode
    Bbyaml.yaml

52. Verify the pod that it is scheduled with the node selector
    kubectl describe pod nginx | grep Node-Selectors

53. Verify the pod nginx that we just created has this label
    kubectl describe pod nginx

54. Annotate the pods with name=webapp
    kubectl annotate pod nginx-dev{1..3} name=webapp
    kubectl annotate pod nginx-prod{1..2} name=webapp

55. Verify the pods that have been annotated correctly
    kubectl describe pod nginx-dev{1..3} | grep -i annotations
    kubectl describe pod nginx-prod{1..2} | grep -i annotations

56. Remove the annotations on the pods and verify
    kubectl annotate pod nginx-dev{1..3} name-
    kubectl annotate pod nginx-prod{1..2} name-
    kubectl describe po nginx-dev{1..3} | grep -i annotations
    kubectl describe po nginx-prod{1..2} | grep -i annotations

57. Remove all the pods that we created so far
    kubectl delete pod --all

58. Create a deployment called webapp with image nginx with 5 replicas
    kubectl create -f Q58.yaml

59. Get the deployment you just created with labels
    kubectl get deploy webapp --show-labels

60. Output the yaml file of the deployment you just created
    kubectl get deploy webapp -o yaml

61. Get the pods of this deployment
    kubectl get pod -l app=webapp

62. Scale the deployment from 5 replicas to 20 replicas and verify
    kubectl scale deploy webapp --replicas=20
    kubectl get rs -l app=webapp

63. Get the deployment rollout status
    kubectl rollout status deploy webapp

64. Get the replicaset that created with this deployment
    kubectl get rs -l app=webapp

65. Get the yaml of the replicaset and pods of this deployment
    kubectl get rs -l app=webapp -o yaml
    kubectl get pod -l app=webapp -o yaml

66. Delete the deployment you just created and watch all the pods are also being deleted
    kubectl delete deploy webapp
    kubectl get pod -l app=webapp -w

67. Create a deployment of webapp with image nginx:1.17.1 with container port 80 and verify the image version
    kubectl create -f wapp.yaml
    kubectl describe deploy webapp | grep Image

68. Update the deployment with the image version 1.17.4 and verify
    kubectl set image deploy/webapp nginx=nginx:1.17.4
    kubectl describe deploy webapp | grep Image

69. Check the rollout history and make sure everything is ok after the update
    kubectl rollout history deploy webapp

70. Undo the deployment to the previous version 1.17.1 and verify Image has the previous version
    kubectl rollout undo deploy/webapp
    kubectl describe deploy webapp | grep Image

71. Update the deployment with the image version 1.16.1 and verify the image and also check the rollout history
    kubectl set image deploy/webapp nginx=nginx:1.16.1

kubectl describe deploy webapp | grep Image
kubectl rollout history deploy webapp

72. Update the deployment to the Image 1.17.1 and verify everything is ok
    kubectl rollout undo deploy webapp --to-revision=3

73. Update the deployment with the wrong image version 1.100 and verify something is wrong with the deployment
    kubectl set image deploy/webapp nginx=nginx:1.100 (imagepullbackoff status)
    kubectl rollout status deploy webapp

74. Undo the deployment with the previous version and verify everything is Ok
    kubectl rollout undo deploy webapp
    kubectl rollout status deploy webapp
    kubectl get pod -l app=webapp

75. Check the history of the specific revision of that deployment
    kubectl rollout history deploy webapp --revision=7

76. Pause the rollout of the deployment
    kubectl rollout pause deploy webapp

77. Update the deployment with the image version latest and check the history and verify nothing is going on
    kubectl set image deploy webapp nginx=nginx:latest
    kubectl rollout history deploy webapp

78. Resume the rollout of the deployment
    kubectl rollout resume deploy webapp

79. Check the rollout history and verify it has the new version
    kubectl rollout history deploy webapp

80. Apply the autoscaling to this deployment with minimum 10 and maximum 20 replicas and target CPU of 85% and verify hpa is created and replicas are increased to 10 from 1
    kubectl autoscale deploy webapp --min=1 --max=10 --cpu-percent=85
    kubectl get hpa

81. Clean the cluster by deleting deployment and hpa you just created
    kubectl delete deploy webapp
    kubectl delete hpa webapp

82. Create a Job with an image node which prints node version and also verifies there is a pod created for this job

kubectl create job myjob --image=node -- node -v
kubectl get pod

83. Get the logs of the job just created
    kubectl logs myjob-knmr2

84. Output the yaml file for the Job with the image busybox which echos “Hello I am from job”
    kubectl create job hello-job --image=busybox --dry-run -o yaml -- echo "Hello I am from job"

85. Copy the above YAML file to hello-job.yaml file and create the job
    kubectl create job job2 --image=busybox --dry-run=client -o yaml -- echo "Hey there" > hello-job.yaml
    kubectl create -f hello-job.yaml

86. Verify the job and the associated pod is created and check the logs as well
    kubectl get job
    kubectl get po
    kubectl logs job2-kj8q4

87. Delete the job we just created
    kubectl delete job job2

88. Create the same job and make it run 10 times one after one
    kubectl create -f hello-job.yaml

89. Watch the job that runs 10 times one by one and verify 10 pods are created and delete those after it’s completed
    kubectl get job -w
    Kubectl get pods
    kubectl delete job job2

90. Create the same job and make it run 10 times parallel
    Add parallelism: 10 in spec of yaml

91. Watch the job that runs 10 times parallelly and verify 10 pods are created and delete those after it’s completed
    kubectl get job -w
    kubectl delete job job2

92. Create a Cronjob with busybox image that prints date and hello from kubernetes cluster message for every minute

kubectl create cronjob democronjob --image=busybox --schedule="*/1 * * * *" -- bin/sh -c "date; echo Hello from kubernetes cluster"

93. Output the YAML file of the above cronjob
    kubectl get cj democronjob -o yaml

94. Verify that CronJob creating a separate job and pods for every minute to run and verify the logs of the pod
    Kubectl get pod
    kubectl logs democronjob-1613996340-lhn54

95. Delete the CronJob and verify all the associated jobs and pods are also deleted.
    kubectl delete cj democronjob
    Kubectl get pod
    Kubectl get job

96. List Persistent Volumes in the cluster
    kubectl get pv

97. Create a hostPath PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteOnce, storageClassName manual, and volume at /mnt/data and verify
    kubectl create -f task-pv-volume.yaml
    kubectl get pv

98. Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify status is Bound
    kubectl create -f task-pv-claim.yaml
    kubectl get pvc

99. Delete persistent volume and PersistentVolumeClaim we just created
    Kubectl delete pv --all
    Kubectl delete pvc --all

100. Create a Pod with an image Redis and configure a volume that lasts for the lifetime of the Pod
    kubectl create -f Q100.yaml

101. Exec into the above pod and create a file named file.txt with the text ‘This is called the file’ in the path /data/redis and open another tab and exec again with the same pod and verifies file exist in the same path.
    kubectl exec -it redis-pod -- bash
    cd /data/redis
    Touch file.txt
    echo ‘This is called the file’ > file.txt

New tab --> kubectl exec -it redis-pod -- bash
cat /data/redis/file.txt

102. Delete the above pod and create again from the same yaml file and verifies there is no file.txt in the path /data/redis
kubectl delete pod redis-pod
kubectl create -f Q100.yaml
kubectl exec -it redis-pod -- bash
cat /data/redis/file.txt

103. Create PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteOnce, storageClassName manual, and volume at /mnt/data and Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify status is Bound
kubectl create -f demopv.yaml
kubectl create -f Q98.yaml

104. Create an nginx pod with containerPort 80 and with a PersistentVolumeClaim task-pv-claim and has a mount path "/usr/share/nginx/html"
kubectl create -f Q104.yaml

105. List all the configmaps in the cluster
kubectl get cm

106. Create a configmap called myconfigmap with literal value appname=myapp
kubectl create cm myconfigmap --from-literal=appname=myapp

107. Verify the configmap we just created has this data
kubectl get cm -o yaml

108. delete the configmap myconfigmap we just created
Kubectl delete cm myconfigmap

109. Create a file called config.txt with two values key1=value1 and key2=value2 and verify the file
cat >> config.txt << EOF
key1=value1
key2=value2
EOF
cat config.txt

110. Create a configmap named keyvalcfgmap and read data from the file config.txt and verify that configmap is created correctly
kubectl create cm keyvalcfgmap --from-file=config.txt

kubectl get cm keyvalcfgmap -o yaml

111. Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables and delete the pod
kubectl create -f Q111.yaml
kubectl exec -it nginx-pod -- env
kubectl delete pod nginx-pod

112. Create an env file file.env with var1=val1 and create a configmap envcfgmap from this env file and verify the configmap
kubectl create cm envcfgmap --from-env-file=file.env
kubectl get cm envcfgmap -o yaml

113. Create an nginx pod and load environment values from the above configmap envcfgmap and exec into the pod and verify the environment variables and delete the pod
kubectl create -f Q113.yaml
kubectl exec -it nginx -- env
kubectl delete pod nginx

114. Create a configmap called cfgvolume with values var1=val1, var2=val2 and create an nginx pod with volume nginx-volume which reads data from this configmap cfgvolume and put it on the path /etc/cfg
kubectl create cm cfgvolume --from-literal=var1=val1 --from-literal=var2=val2
kubectl describe cm cfgvolume
kubectl create -f Q114.yaml
Terminal: kubectl exec -it nginxpod -- bin/sh
cd /etc/cfg
ls
var1 var2

115. Create a pod called secbusybox with the image busybox which executes command sleep 3600 and makes sure any Containers in the Pod, all processes run with user ID 1000 and with group id 2000 and verify.
kubectl create -f Q115.yaml
kubectl run secbusybox --image=busybox --restart=Never --dry-run -o yaml -- /bin/sh -c "sleep 3600;" > Q115.yaml
kubectl exec -it secbusybox -- sh
$id
Output : uid=1000 gid=2000 groups=2000

116. Create the same pod as above this time set the securityContext for the container as well and verify that the securityContext of container overrides the Pod level securityContext.

kubectl create -f Q116.yaml
kubectl exec -it secbusybox1 -- sh
$id → uid:2000

117. Create pod with an nginx image and configure the pod with capabilities NET_ADMIN and SYS_TIME verify the capabilities
kubectl create -f Q117.yaml

```
[ayrastog@ayrastog nginxpod]$ kubectl exec -it nginxpodd -- sh
# cd /proc/1
# cat status
Name:   nginx
Umask:  0022
State:  S (sleeping)
Tgid:   1
Ngid:   0
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 64
Groups:
NStgid: 1
NSpid:  1
NSpgid: 1
NSsid:  1
VmPeak:    10664 kB
VmSize:    10636 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      5876 kB
VmRSS:      5876 kB
RssAnon:             804 kB
RssFile:            5072 kB
RssShmem:              0 kB
VmData:      980 kB
VmStk:       132 kB
VmExe:       980 kB
VmLib:      3800 kB
VmPTE:        60 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
Threads:        1
SigQ:   0/22437
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000040001000
SigCgt: 0000000198016a07
CapInh: 00000000aa0435fb
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
CapBnd: 00000000aa0435fb
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        0
Speculation_Store_Bypass:       thread vulnerable
Cpus_allowed:   3
Cpus_allowed_list:      0-1
```

118. Create a Pod nginx and specify a memory request and a memory limit of 100Mi and 200Mi respectively.

kubectl create -f Q118.yaml

119. Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.
kubectl create -f Q119.yaml

120. Create a Pod nginx and specify both CPU, memory requests and limits together and verify.
kubectl create -f Q120.yaml
kubectl top pod

121. Create a Pod nginx and specify a memory request and a memory limit of 100Gi and 200Gi respectively which is too big for the nodes and verify pod fails to start because of insufficient memory
kubectl create -f Q121.yml
kubectl describe pod nginx - pending state

122. Create a secret mysecret with values user=myuser and password=mypassword
kubectl create secret generic my-secret --from-literal=user=myuser --from-literal=password=mypassword

123. List the secrets in all namespaces
kubectl get secret --all-namespaces

124. Output the yaml of the secret created above
kubectl get secret my-secret -o yaml

125. Create an nginx pod which reads username as the environment variable
kubectl create -f Q125.yaml

126. Create an nginx pod which loads the secret as environment variables
kubectl create -f Q126.yaml
Kubectl exec -it nginx -- env

127. List all the service accounts in the default namespace
kubectl get sa

128. List all the service accounts in all namespaces
kubectl get sa --all-namespaces

129. Create a service account called admin
kubectl create sa admin

130. Output the YAML file for the service account we just created

Kubectl get sa admin -o yaml

131. Create a busybox pod which executes this command sleep 3600 with the service account admin and verify
kubectl create -f Q131.yaml
kubectl describe pod busyboxpod

132. Create an nginx pod with containerPort 80 and it should only receive traffic only it checks the endpoint / on port 80 and verify and delete the pod.
kubectl create -f nginx-get.yaml
kubectl describe pod nginx | grep -i readiness
kubectl delete pod nginx

133. Create an nginx pod with containerPort 80 and it should check the pod running at endpoint / healthz on port 80 and verify and delete the pod.
Change path to /healthz

134. Create an nginx pod with containerPort 80 and it should check the pod running at endpoint /healthz on port 80 and it should only receive traffic only it checks the endpoint / on port 80. verify the pod.
Add readinessProbe

135. Check what all are the options that we can configure with readiness and liveness probes
kubectl explain Pod.spec.containers.livenessProbe
kubectl explain Pod.spec.containers.readinessProbe

136. Create the pod nginx with the above liveness and readiness probes so that it should wait for 20 seconds before it checks liveness and readiness probes and it should check every 25 seconds.
kubectl create -f Q136.yaml

137. Create a busybox pod with this command “echo I am from busybox pod; sleep 3600;” and verify the logs.
kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c "echo I am from busybox pod; sleep 3600;"
kubectl logs busybox

138. copy the logs of the above pod to the busybox-logs.txt and verify
kubectl logs busybox > busybox-logs.txt
cat busybox-logs.txt

139. List all the events sorted by timestamp and put them into file.log and verify
kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get events --sort-by=.metadata.creationTimestamp > file.log
cat file.log

140. Create a pod with an image alpine which executes this command ”while true; do echo ‘Hi I am from alpine’; sleep 5; done” and verify and follow the logs of the pod.
kubectl run hello --image=alpine --restart=Never -- /bin/sh -c "while true; do echo 'Hi I am from Alpine'; sleep 5;done"
kubectl logs --follow hello

141. Create the pod with this kubectl create -f https://gist.githubusercontent.com/bbachi/212168375b39e36e2e2984c097167b00/raw/1fd63509c3ae3a3d3da844640fb4cca744543c1c/not-running.yml. The pod is not in the running state. Debug it.
kubectl create -f https://gist.githubusercontent.com/bbachi/212168375b39e36e2e2984c097167b00/raw/1fd63509c3ae3a3d3da844640fb4cca744543c1c/not-running.yml
kubectl get pod not-running
kubectl describe pod not-running - ImagePullBackOff
kubectl set image pod/not-running not-running=nginx

142. This following yaml creates 4 namespaces and 4 pods. One of the pod in one of the namespaces are not in the running state. Debug and fix it. https://gist.githubusercontent.com/bbachi/1f001f10337234d46806929d12245397/raw/84b7295fb077f15de979fec5b3f7a13fc69c6d83/problem-pod.yaml.
kubectl create -f https://gist.githubusercontent.com/bbachi/1f001f10337234d46806929d12245397/raw/84b7295fb077f15de979fec5b3f7a13fc69c6d83/problem-pod.yaml
kubectl get po --all-namespaces
kubectl get po -n namespace2
kubectl set image pod/pod2 pod2=nginx -n namespace2
kubectl get po -n namespace2

143. Get the memory and CPU usage of all the pods and find out top 3 pods which have the highest usage and put them into the cpu-usage.txt file
kubectl top pod --all-namespaces | sort --reverse --key 3 --numeric | head -3 > cpu-usage.txt
cat cpu-usage.txt

144. Create an nginx pod with a yaml file with label my-nginx and expose the port 80
kubectl run nginx --image=nginx --restart=Never --port=80 --dry-run -o yaml > Q144.yaml
kubectl create -f Q144.yaml

145. Create the service for this nginx pod with the pod selector app: my-nginx
kubectl create -f Q145.yaml

146. Find out the label of the pod and verify the service has the same label

```
kubectl get pod nginx --show-labels
kubectl get svc my-service -o wide
```

147. Delete the service and create the service with kubectl expose command and verify the label

```
kubectl delete svc my-service
kubectl expose po nginx --port=80 --target-port=9376
kubectl get svc -l app=my-nginx
```

148. Delete the service and create the service again with type NodePort

```
kubectl delete svc nginx
kubectl expose po nginx --port=80 --type=NodePort
```

149. Create the temporary busybox pod and hit the service. Verify the service that it should return the nginx page index.html.

```
kubectl get svc nginx -o wide
kubectl run busybox --image=busybox --restart=Never -it --rm -- wget -o- <Cluster IP>:80
```

150. Create a NetworkPolicy which denies all ingress traffic

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
Metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```