1. List all the namespaces in the cluster

2. List all the pods in all namespaces

3. List all the pods in the particular namespace

4. List all the services in the particular namespace

5. List all the pods showing name and namespace with a json path expression

6. Create an nginx pod in a default namespace and verify the pod running

7. generate the yaml for pod called nginx2 & write to /opt/nginx203.yml. DONOT create the pod

8. Output the yaml file of the pod nginx created above & write the output to /opt/nginx.yml

9. Output the yaml file of the pod you just created without the cluster-specific information

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

11. Delete the pod nginx you just created

12. create a pod named alpine with image nginx & Delete the pod created without any delay (force delete)

13. Create the nginx pod with version 1.17.4 and expose it on port 80

14. Change the Image version to 1.15-alpine for the pod you just created and verify the image version is updated

15. Change the Image version back to 1.17.1 for the pod you just updated and observe the changes

16. Check the Image version without the describe command (use jsonpath)

17. Create the nginx pod and execute the simple shell on the pod

18. Get the IP Address of the pod you just created

19. Create a busybox pod and run command ls while creating it and check the logs

20. If pod crashed check the previous logs of the pod

21. Create a busybox pod with command sleep 3600

22. Check the connection of the nginx pod from the busybox pod

23. Create a busybox pod and echo message ‘How are you’ and delete it manually

24. Create a busybox pod and echo message ‘How are you’ and have it deleted immediately

25. Create an nginx pod and list the pod with different levels of verbosity

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

27. List all the pods sorted by name

28. List all the pods sorted by created timestamp

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

30. Check the logs of each container that you just created

31. Check the previous logs of the second container busybox2 if any

32. Run command ls in the third container busybox3 of the above pod

33. Show metrics of the above pod containers and puts them into the file.log and verify

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.

35. Exec into both containers and verify that main.txt exist and query the main.txt from sidecar container with curl localhost

36. Get the pods with label information

37. Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev

38. Verify all the pods are created with correct labels

39. Get the pods with label env=dev

40. Get the pods with label env=dev and also output the labels

41. Get the pods with label env=prod

42. Get the pods with label env=prod and also output the labels

43. Get the pods with label env

44. Get the pods with labels env=dev and env=prod

45. Get the pods with labels env=dev and env=prod and output the labels as well

46. Change the label for one of the pod to env=uat and list all the pods to verify

47. Remove the labels for the pods that we created now and verify all the labels are removed

48. Let’s add the label app=nginx for all the pods and verify

49. Get all the nodes with labels

50. Label the node node1 nodeName=nginxnode

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

52. Verify the pod that it is scheduled with the node selector

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

57. Remove all the pods that we created so far

58. Create a deployment called webapp with image nginx with 5 replicas

59. Get the deployment you just created with labels

60. Output the yaml file of the deployment you just created

61. Get the pods of this deployment

62. Scale the deployment from 5 replicas to 8 replicas and verify

63. Get the deployment rollout status

64. Get the replicaset that created with this deployment

65. Get the yaml of the replicaset and pods of this deployment

66. Delete the deployment you just created and watch all the pods are also being deleted

67. Create a deployment of webapp with image nginx:1.17.1 with container port 80 and verify the image version

68. Update the deployment with the image version 1.17.4 and verify

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

70. Undo the deployment to the previous version 1.17.1 and verify Image has the previous version

71. Update the deployment with the image version 1.16.1 and verify the image and also check the rollout history

72. Update the deployment to the Image 1.17.1 and verify everything is ok

73. Update the deployment with the wrong image version 1.100 and verify something is wrong with the deployment

74. Undo the deployment with the previous version and verify everything is Ok

75. Check the history of the specific revision of that deployment

76. Pause the rollout of the deployment

77. Update the deployment with the image version latest and check the history and verify nothing is going on

78. Resume the rollout of the deployment

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

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

81. Clean the cluster by deleting deployment and hpa you just created

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

83. Get the logs of the job just created

84.Output the yaml file for the Job with the image busybox which echos “Hello I am from job”

85. Copy the above YAML file to hello-job.yaml file and create the job

86. Verify the job and the associated pod is created and check the logs as well

87. Delete the job we just created

88. Create the same job and make it run 10 times one after one

89. Watch the job that runs 10 times one by one and verify 10 pods are created and delete those after it’s completed

90. Create the same job and make it run 10 times parallel

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

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

93. Output the YAML file of the above cronjob

94. Verify that CronJob creating a separate job and pods for every minute to run and verify the logs of the pod

95. Delete the CronJob and verify all the associated jobs and pods are also deleted.

96. List Persistent Volumes in the cluster

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

98. Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify status is Bound

99. Delete persistent volume and PersistentVolumeClaim we just created

100. Create a Pod with an image Redis and configure a volume that lasts for the lifetime of the Pod

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.

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

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

104. Create an nginx pod with containerPort 80 and with a PersistentVolumeClaim task-pv-claim and has a mouth path "/usr/share/nginx/html"

105. List all the configmaps in the cluster

106. Create a configmap called myconfigmap with literal value appname=myapp

107. Verify the configmap we just created has this data

108. delete the configmap myconfigmap we just created

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

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

111. Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables store it on /opt/keyvalcfgmap-vars.txt. and delete the pod

112. Create an env file file.env with var1=val1 and create a configmap envcfgmap from this env file and verify the configmap

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

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

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.

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.

117. Create pod with an nginx image and configure the pod with capabilities NET_ADMIN and SYS_TIME verify the capabilities

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

119. Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.

120. Create a Pod nginx and specify both CPU, memory requests and limits together and verify.

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

122. Create a secret mysecret with values user=myuser and password=mypassword

123. List the secrets in all namespaces

124. Output the yaml of the secret created above

125. Create an nginx pod which reads username as the environment variable

126. Create an nginx pod which loads the secret as environment variables

127. List all the service accounts in the default namespace

128. List all the service accounts in all namespaces

129. Create a service account called admin

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

131. Create a busybox pod which executes this command sleep 3600 with the service account admin and verify

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.

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.

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.

135. Check what all are the options that we can configure with readiness and liveness probes

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.

137. Create a busybox pod with this command “echo I am from busybox pod; sleep 3600;” and verify the logs.

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

139. List all the events sorted by timestamp and put them into file.log and verify

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.

141. Create the pod with this kubectl create -f https://raw.githubusercontent.com/lerndevops/educka/master/exam-prep/not-running.yml The pod is not in the running state. Debug it.

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://raw.githubusercontent.com/lerndevops/educka/master/exam-prep/problem-pod.yml

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

144. Create an nginx pod with a yaml file with label my-nginx and expose the port 80

145. Create the service for this nginx pod with the pod selector app: my-nginx

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

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

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

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

150. Create a NetworkPolicy which denies all ingress traffic

151. Create a yaml file called db-secret.yaml for a secret called db-user-pass. The secret should have two fields: a username and password. The username should be "superadmin" and the password should be "imamazing".

152. Create a ConfigMap called web-config that contains the following two entries: 'web_port' set to 'localhost:8080' 'external_url' set to 'reddit.com' Run a pod called web-config-pod running nginx, expose the configmap settings as environment variables inside the nginx container.

153. list all the pods with Running status in the default namespace & write the output to /opt/default-pod-status.txt

154. list all the persistant volumes, sort by storage reqeust size & write the output to /opt/pv-size.txt

155. list all namespaces & sort by name

156. Create a job that calculates pi to 2000 decimal points using the container with the image named perl
and the following commands issued to the container: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
Once the job has completed, check the logs to and export the result to pi-result.txt.

157. Create a yaml file called nginx-deploy.yaml for a deployment of three replicas of nginx, listening on the container's port 80. They should have the labels role=webserver and app=nginx. The deployment should be named nginx-deploy. Expose the deployment with a NodePort and use a curl statement on the IP address of the NodeIP
to export the output to a file titled output.txt.

158. Scale the deployment you just made down to 2 replicas

159. Create a Deployment called "haz-docs" with an nginx image listening on port 80. Attach the pod to emptyDir storage, mounted to /tmp in the container. Connect to the pod and create a file with zero bytes in the /tmp directory called my-doc.txt.

160. Label the worker node of your cluster with rack=qa.

161. Create a file called counter.yaml in your home directory and paste the following yaml into it:

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: count
    image: busybox
    args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']
```

162. Create a new namespace called "cloud9". Create a pod running nginx with a liveliness probe that uses httpGet to probe an endpoint path located at / on port 80. The initial delay is 3 seconds and the period is 3.

163. Configure a DaemonSet to run the image k8s.gcr.io/pause:2.0 in the cluster.

164. An app inside a container needs the IP address of the web-dep endpoint to be passed to it as an
environment variable called "ULTIMA". create a deployment with 1 replica using image nginx & assing the environment variable with values. Save the yaml as env-ultima.yaml

165. Figure out a way to create a deployment with 3 replicas using the the nginx container that can have pods deployed on the master node.

166. Copy all Kubernetes scheduler logs into a /opt/schedular-log.txt

165. Run the pod below until the counter in exceeds 30, export the log file into a file called counter-log.txt.

```
apiVersion: v1
kind: Pod
metadata:
  name: counter1
spec:
  containers:
  - name: count
```

```
                    image: busybox
                    args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; echo "$(date) - File - $i"
>> /var/www/countlog; i=$((i+1)); sleep 3; done']
```

166. Create a deployment running nginx, mount a volume called "hostvolume" with a container volume mount at /tmp and mounted to the host at /data. If the directory isn't there make sure it is created in the pod spec at run time.
Go into the container and create an empty file called "my-doc.txt" inside the /tmp directory. On the worker node
that it was scheduled to, go into the /data directory and output a list of the contents to list-output.txt showing
the file exists.

167. Create a namespace called "myns" in your cluster. Create a deployment called "db-deploy" that has one container running mysql image, and one container running nginx:1.7.9 in namespace "myns".
create another deployment called nginx-deploy with a single container running the image nginx:1.9.1 in namespace myns. Export the output of kubectl get pods for the myns namespace into a file called "pod-list.txt" (kubect get pods -n myns > pod-list.txt)

168. disable the scheduling to node1 in your cluster & write the output to /opt/node-SchedulingDisabled.txt

169. now enable the scheduling again to node1 in your cluster & write the output to /opt/node-SchedulingEnabled.txt

170. create a pod named redis using image radis:alpha

171. create a pod using image tomcat & assign label app with value web

172. create a pod named kual203 using image lerndevops/alpine:beta, find the errors in the logs, write the error messages to a file /opt/kual203.txt

173. Deploy a messaging pod using the redis:alpine image with the labels set to tier=msg & app=web

174. list all the pods with label app=web & write the output to a file /opt/webpods.txt

175. deploy pod using
https://raw.githubusercontent.com/lerndevops/educka/master/exam-prep/app-redis.yml
if there are any errors try to fix them and ensure the pod status is running. DO NOT delete the pod.

176. Create a POD in the finance namespace named red-bus with the image redis:alpine

177. deploy a pod with forex name using lerndevops/tomcat:8.5.3 in finance namespace with label env=qa

178. list all pods with running status in finance namespace and write the output to /opt/kufinpods.txt

179. deploy a pod named envpod using image nginx & apply env variables username=dbuser password=dbpass

180. deploy a pod named dbpod using image lerndevops/alpine:sleep with label app=testapp, env variable uname=appuser & set the restart policy to never

181. genereate pod definition file with pod named apppod with application contianers using nginx,tomcat,redis then save the definition file to /opt/apppod.yml. then deploy the pod & ensure it is running state.

ANSWERS:

https://drive.google.com/file/d/1KYvKmZecVzxFISO3wTp1UsXH9GuaJ77V/view?usp=sharing