Task weight: 1%

The DevOps team would like to get the list of all Namespaces in the cluster. Get the list and save it to /opt/course/1/namespaces.

Task weight: 2%

Create a single Pod of image httpd:2.4.41-alpine in Namespace default. The Pod should be named pod1 and the container should be named pod1-container.

Your manager would like to run a command manually on occasion to output the status of that exact Pod. Please write a command that does this into /opt/course/2/pod1-status-command.sh. The command should use kubectl.

Task weight: 2%

Team Neptune needs a Job template located at /opt/course/3/job.yaml. This Job should run image busybox:1.31.0 and execute sleep 2 && echo done. It should be in namespace neptune, run a total of 3 times and should execute 2 runs in parallel.

Start the Job and check its history. Each pod created by the Job should have the label id: awesome-job. The job should be named neb-new-job and the container neb-new-job-container.

*Task weight: 5%*

Team Mercury asked you to perform some operations using Helm, all in *Namespace* mercury:

1. Delete release internal-issue-report-apiv1
2. Upgrade release internal-issue-report-apiv2 to any newer version of chart bitnami/nginx available
3. Install a new release internal-issue-report-apache of chart bitnami/apache. The *Deployment* should have two replicas, set these via Helm-values during install
4. There seems to be a broken release, stuck in pending-install state. Find it and delete it

*Task weight: 3%*

Team Neptune has its own *ServiceAccount* named neptune-sa-v2 in *Namespace* neptune. A coworker needs the token from the *Secret* that belongs to that *ServiceAccount*. Write the base64 decoded token to file /opt/course/5/token.

*Task weight: 7%*

Create a single *Pod* named pod6 in *Namespace* default of image busybox:1.31.0. The *Pod* should have a readiness-probe executing cat /tmp/ready. It should initially wait 5 and periodically wait 10 seconds. This will set the container ready only if the file /tmp/ready exists.

The *Pod* should run the command touch /tmp/ready && sleep 1d, which will create the necessary file to be ready and then idles. Create the *Pod* and confirm it starts.

*Task weight: 4%*

The board of Team Neptune decided to take over control of one e-commerce webserver from Team Saturn. The administrator who once setup this webserver is not part of the organisation any longer. All information you could get was that the e-commerce system is called my-happy-shop.

Search for the correct *Pod* in *Namespace* saturn and move it to *Namespace* neptune. It doesn't matter if you shut it down and spin it up again, it probably hasn't any customers anyways.

*Task weight: 4%*

There is an existing *Deployment* named api-new-c32 in *Namespace* neptune. A developer did make an update to the *Deployment* but the updated version never came online. Check the *Deployment* history and find a revision that works, then rollback to it. Could you tell Team Neptune what the error was so it doesn't happen again?

*Task weight: 5%*

In *Namespace* pluto there is single *Pod* named holy-api. It has been working okay for a while now but Team Pluto needs it to be more reliable. Convert the *Pod* into a *Deployment* with 3 replicas and name holy-api. The raw *Pod* template file is available at /opt/course/9/holy-api-pod.yaml.

In addition, the new *Deployment* should set allowPrivilegeEscalation: false and privileged: false for the security context on container level.

Please create the *Deployment* and save its yaml under /opt/course/9/holy-api-deployment.yaml.

*Task weight: 4%*

Team Pluto needs a new cluster internal *Service*. Create a ClusterIP *Service* named project-plt-6cc-svc in *Namespace* pluto. This *Service* should expose a single *Pod* named project-plt-6cc-api of image nginx:1.17.3-alpine, create that *Pod* as well. The *Pod* should be identified by label project: plt-6cc-api. The *Service* should use tcp port redirection of 3333:80.

Finally use for example curl from a temporary nginx:alpine *Pod* to get the response from the *Service*. Write the response into /opt/course/10/service\_test.html. Also check if the logs of *Pod* project-plt-6cc-api show the request and write those into /opt/course/10/service\_test.log.

*Task weight: 7%*

During the last monthly meeting you mentioned your strong expertise in container technology. Now the Build&Release team of department Sun is in need of your insight knowledge. There are files to build a container image located at /opt/course/11/image. The container will run a Golang application which outputs information to stdout. You're asked to perform the following tasks:

***NOTE:*** Make sure to run all commands as user k8s, for docker use sudo docker

1. Change the Dockerfile. The value of the environment variable SUN\_CIPHER\_ID should be set to the hardcoded value 5b9c1065-e39d-4a43-a04a-e59bcea3e03f
2. Build the image using Docker, named registry.killer.sh:5000/sun-cipher, tagged as latest and v1-docker, push these to the registry
3. Build the image using Podman, named registry.killer.sh:5000/sun-cipher, tagged as v1-podman, push it to the registry
4. Run a container using Podman, which keeps running in the background, named sun-cipher using image registry.killer.sh:5000/sun-cipher:v1-podman. Run the container from k8s@terminal and not root@terminal
5. Write the logs your container sun-cipher produced into /opt/course/11/logs. Then write a list of all running Podman containers into /opt/course/11/containers

*Task weight: 8%*

Create a new *PersistentVolume* named earth-project-earthflower-pv. It should have a capacity of *2Gi*, accessMode *ReadWriteOnce*, hostPath /Volumes/Data and no storageClassName defined.

Next create a new *PersistentVolumeClaim* in *Namespace* earth named earth-project-earthflower-pvc . It should request *2Gi* storage, accessMode *ReadWriteOnce* and should not define a storageClassName. The *PVC* should bound to the *PV* correctly.

Finally create a new *Deployment* project-earthflower in *Namespace* earth which mounts that volume at /tmp/project-data. The *Pods* of that *Deployment* should be of image httpd:2.4.41-alpine.

*Task weight: 6%*

Team Moonpie, which has the *Namespace* moon, needs more storage. Create a new *PersistentVolumeClaim* named moon-pvc-126 in that namespace. This claim should use a new *StorageClass* moon-retain with the *provisioner* set to moon-retainer and the *reclaimPolicy* set to *Retain*. The claim should request storage of *3Gi*, an *accessMode* of *ReadWriteOnce* and should use the new *StorageClass*.

The provisioner moon-retainer will be created by another team, so it's expected that the *PVC* will not boot yet. Confirm this by writing the log message from the *PVC* into file /opt/course/13/pvc-126-reason.

*Task weight: 4%*

You need to make changes on an existing *Pod* in *Namespace* moon called secret-handler. Create a new *Secret* secret1 which contains user=test and pass=pwd. The *Secret*'s content should be available in *Pod* secret-handler as environment variables SECRET1\_USER and SECRET1\_PASS. The yaml for *Pod* secret-handler is available at /opt/course/14/secret-handler.yaml.

There is existing yaml for another *Secret* at /opt/course/14/secret2.yaml, create this *Secret* and mount it inside the same *Pod* at /tmp/secret2. Your changes should be saved under /opt/course/14/secret-handler-new.yaml. Both *Secrets* should only be available in *Namespace* moon.

*Task weight: 5%*

Team Moonpie has a nginx server *Deployment* called web-moon in *Namespace* moon. Someone started configuring it but it was never completed. To complete please create a *ConfigMap* called configmap-web-moon-html containing the content of file /opt/course/15/web-moon.html under the data key-name index.html.

The *Deployment* web-moon is already configured to work with this *ConfigMap* and serve its content. Test the nginx configuration for example using curl from a temporary nginx:alpine *Pod*.

*Task weight: 6%*

The Tech Lead of Mercury2D decided its time for more logging, to finally fight all these missing data incidents. There is an existing container named cleaner-con in *Deployment* cleaner in *Namespace* mercury. This container mounts a volume and writes logs into a file called cleaner.log.

The yaml for the existing *Deployment* is available at /opt/course/16/cleaner.yaml. Persist your changes at /opt/course/16/cleaner-new.yaml but also make sure the *Deployment* is running.

Create a sidecar container named logger-con, image busybox:1.31.0 , which mounts the same volume and writes the content of cleaner.log to stdout, you can use the tail -f command for this. This way it can be picked up by kubectl logs.

Check if the logs of the new container reveal something about the missing data incidents.

*Task weight: 4%*

Last lunch you told your coworker from department Mars Inc how amazing *InitContainer*s are. Now he would like to see one in action. There is a *Deployment* yaml at /opt/course/17/test-init-container.yaml. This *Deployment* spins up a single *Pod* of image nginx:1.17.3-alpine and serves files from a mounted volume, which is empty right now.

Create an *InitContainer* named init-con which also mounts that volume and creates a file index.html with content check this out! in the root of the mounted volume. For this test we ignore that it doesn't contain valid html.

The *InitContainer* should be using image busybox:1.31.0. Test your implementation for example using curl from a temporary nginx:alpine *Pod*.

*Task weight: 4%*

There seems to be an issue in *Namespace* mars where the ClusterIP service manager-api-svc should make the *Pods* of *Deployment* manager-api-deployment available inside the cluster.

You can test this with curl manager-api-svc.mars:4444 from a temporary nginx:alpine *Pod*. Check for the misconfiguration and apply a fix.

*Task weight: 3%*

In *Namespace* jupiter you'll find an apache *Deployment* (with one replica) named jupiter-crew-deploy and a ClusterIP *Service* called jupiter-crew-svc which exposes it. Change this service to a NodePort one to make it available on all nodes on port 30100.

Test the NodePort *Service* using the internal IP of all available nodes and the port 30100 using curl, you can reach the internal node IPs directly from your main terminal. On which nodes is the *Service* reachable? On which node is the *Pod* running?

*Task weight: 9%*

In *Namespace* venus you'll find two *Deployments* named api and frontend. Both *Deployments* are exposed inside the cluster using *Services*. Create a *NetworkPolicy* named np1 which restricts outgoing tcp connections from *Deployment* frontend and only allows those going to *Deployment* api. Make sure the *NetworkPolicy* still allows outgoing traffic on UDP/TCP ports 53 for DNS resolution.

Test using: wget www.google.com and wget api:2222 from a *Pod* of *Deployment* frontend.

*Task weight: 4%*

Team Neptune needs 3 *Pods* of image httpd:2.4-alpine, create a *Deployment* named neptune-10ab for this. The containers should be named neptune-pod-10ab. Each container should have a memory request of *20Mi* and a memory limit of *50Mi*.

Team Neptune has its own *ServiceAccount* neptune-sa-v2 under which the *Pods* should run. The *Deployment* should be in *Namespace* neptune.

*Task weight: 3%*

Team Sunny needs to identify some of their *Pods* in namespace sun. They ask you to add a new label protected: true to all *Pods* with an existing label type: worker or type: runner. Also add an annotation protected: do not delete this pod to all *Pods* having the new label protected: true.