

CKAD Prep, Exam Experience, Cheat Sheet and Few topics for quick review

Friday, June 25, 2021 10:55 PM

I have cleared the CKAD exam at my first attempt! Would like to share my preparation, tips and exam experience. As everyone says practice, practice..., after going through many articles, tips and experience of others I too ended up practice more. Practice is the only key for success. As much as you could practice you will be able to solve the questions faster. When you practice don't just practice the command understand the concepts and know how to verify your solution. Key things to remember while preparing for the exam is **CUP** (Concentrate, Understand and Practice). I am sharing all my notes below, might be too much read. Just sharing hope it helps others.

For those who are planning to take exams in couple of days just read about **section 2. Exam Experience/Tips** because you would have already read many articles/blogs of others.

1. Preparation:



Have shared all my reference links below, it could be too much. When I started my journey with CKAD, my main goal was to read more and understand. My plan was not just certified for CKAD to know more about Kubernetes for development. Hence the list is more. Have categorized the mandatory once for exam. Some of the blog says don't over study, yes don't over study if your exam is very close by.

Exam Simulator - how it look like : https://www.cncf.io/announcements/2021/06/02/linux-foundation-kubernetes-certifications-now-include-exam-simulator/?utm_content=168763729&utm_medium=social&utm_source=linkedin&hss_channel=lcp-12893459

Practice 2-3 times before the exam

<https://www.udemy.com/course/certified-kubernetes-application-developer/>
<https://codeburst.io/kubernetes-ckad-weekly-challenges-overview-and-tips-7282b36a2681>
<https://github.com/dgkanatsios/CKAD-exercises>
<https://medium.com/bb-tutorials-and-thoughts/practice-enough-with-these-questions-for-the-ckad-exam-2f42d1228552>
<https://github.com/ahmetb/kubernetes-network-policy-recipes>

TIPS and Other References:

- Vim Crash Course | How to edit files quickly in CKAD / CKA exam - https://www.youtube.com/watch?v=knyJt8d6C_8
- How to CRUSH the CKAD Exam! - <https://www.youtube.com/watch?v=5cgpFWVD8ds>
- CKAD Practice Challenge in Katacoda - <https://katacoda.com/liptanbiswas/courses/ckad-practice-challenges>
- <https://www.study4exam.com/> - Had not purchased this, had this link as part of my prepy list. But I feel it is worth to practice!

Additional Reference only for more understanding(Not Mandatory):

- <https://kodekloud.com/courses/game-of-pods>
- <https://github.com/bmuschko/ckad-crash-course>
- <https://github.com/bmuschko/ckad-study-guide>
- <https://mingchaoliao.github.io/2020/04/26/ckad-exam-guide.html#what-is-kubernetes>
- The Kubernetes Workshop** by Mohammed Abu Taleb; Zachary Arnold; Sahil Dua; Melony Qin; Faisal Masood; Wei Huang
- Kubernetes in Action** By Marko Lukša

2. Exam Experience/Tips



The auto complete of kubectl, copy and paste all worked well in the exam environment for me. Had given below the alias which I set just before start answering the questions in the exam. I had set these only once during the exam. For every question, there was a kubectl config set-context command given all you need to do is just copy(click on the text) and paste(shift+insert) to execute. Some of them reported saying the auto complete was not working for them I am not sure why. But the below configurations worked fine for me in Windows OS.

Whenever you practice set the below commands and start solving the questions, so that you will not forget to set this during the exam and it helps remember these steps in the exam.

When you start answering the questions always check the weightage and see how long it will take for you to solve. If less weightage and questions is too big flag it and move to the next one. I too got few of them with weightage 2%,3% which I flagged it and revisited it later. I too set myself goal of solving the at least minimum of 11 questions during first half of the session i.e. 60 minutes where as I was able to solve 12-13 questions. Rest of the questions were big took more time to complete. I was able to attempt all of the questions in 120 minutes.

2.1 Alias to set just before start answering the questions in the exam.

vi ~/.vimrc and enter the below at end of file -- to set line number and convert tab to spaces
set tabstop=2 shiftwidth=2 expandtab

a. vi geetha.conf add the below contents

```
source <(kubectl completion bash) # setup autocomplete in bash into the current shell, bash-completion package should be installed first.
echo "source <(kubectl completion bash)" >> ~/.bashrc # Had copied this from the kubernetes.io cheat sheet.
alias k='kubectl' # Had copied this from the kubernetes.io cheat sheet.
complete -F __start_kubectl k # Had copied this from the kubernetes.io cheat sheet.
alias kcg='k config get-contexts'
alias kcs='k config set-context --current --namespace'
alias ka='k apply -f'
export do='--dry-run=client -o yaml'
export gp='--grace-period=0 -o yaml'
alias
```

b. source geetha.conf

c. vi ~/.vimrc add the below option

```
set tabstop=2 shiftwidth=2 expandtab
```

Once the above configurations done,

To change namespace use below command

```
kcs <namespace>; kcg
```

To use dry-run

```
k run nginx --image=nginx --port 80 $do > nginx.yaml
```

Have used windows OS(laptop) the copy (ctrl+ins), and past (shift+ins) worked well.

3. Cheat Sheet and Few topics for quick review to understand and remember these topics.

Had prepared this cheat sheet while practicing for the exam. Sharing it for reference. Some of the key topics for quick glance whenever is required during preparation or just before the exams.



3.1 Cheat Sheet - Kubectl Commands

Sl.No#	Function	Command
1	Create pod definition template yaml file using command. Dry Run pod nginx and write its spec into a file called pod.yaml	<code>kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.yaml</code>
2.	To modify a running pod, create pod definition yaml file for a running pod.	<code>kubectl get pod <podname> -n <namespace> -o yaml > pod.yaml</code>
3.	Set default name space as different namespace	<code>kubectl config set-context --current --namespace=<namespace-name></code>
4.	Use watch commend instead of running the same to check the pod status or any other command	<code>watch kubectl get pods</code>
5	To change default name space	<code>kubectl config set-context --current --namespace=<insert-namespace-name-here></code> # Validate it <code>kubectl config view --minify grep namespace:</code> Or <code>Kubectl config get-contexts</code>
6	To run more than one (pod) def yaml file	<code>kubectl apply -f <dir-name></code>
7	To create a job yaml definition	<code>kubectl create job throw-dice-job --image kodekloud/throw-dice -o yaml --dry-run=client > job.yaml</code>
8	To create a cronjob yaml definition	<code>kubectl create cronjob throw-dice-cron-job --image=kodekloud/throw-dice --schedule='30 21 * * *' --dry-run=client -o yaml > cron.yaml</code> To generate CronJob schedule expressions, you can also use web tools like crontab.guru Every minute - <code>*****</code> or <code>*/1*****</code> Every Hour - <code>0 */1***</code> Every year on 1st Jan - <code>0 0 1 1 *</code> Every month 1st - <code>0 0 1 */1 *</code>
9	To delete a Kubernetes object immediately.	<code>\$ kubectl delete pod nginx --grace-period=0 --force</code>
10	kubectl provides short names for some of the resources.	<code>\$ kubectl api-resources</code> NAME SHORTNAMES APIGROUP NAMESPACED KIND ... persistentvolumeclaims pvc true PersistentVolumeClaim ...
11	The -C command line option helps with rendering the lines before and after the search term	<code>\$ kubectl describe pods grep -C 10 "author=John Doe"</code> <code>\$ kubectl get pods -o yaml grep -C 5 labels:</code>
12	Delete all pods running in a namespace	You can delete all the pods in a single namespace with this command: <code>kubectl delete --all pods --namespace=foo --force</code>
13	Nslookup <servicename>.<namespace> -> namespace is optional	<code>K run busybox --image=busybox --rm -it --restart Never -- sh</code> <code>nslookup <serviceName></code>
14	Check service reachable or not Reference: https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/	1) <code>k describe svc <service-name></code> IP: 10.110.12.190 Port: <unset> 80/TCP TargetPort: 80/TCP Endpoints: 10.32.0.6:80 2) <code>curl http://10.110.12.190:80</code> or <code>curl http://10.32.0.6:80</code> at masternode 3) <code>k exec secure-pod -- printenv grep SERVICE</code> When a Pod runs on a Node, the kubelet adds a set of environment

		variables for each active Service.
15	List the taints applied in a node	To get specific node the taints: kubect! describe nodes controlplane grep -i "taint" -A 5 Sample o/p Taints: node-role.kubernetes.io/master:NoSchedule To grep multiple words in get nodes(all nodes in the cluster) yaml o/p - kubect! get nodes -o yaml grep 'taint\ name:' -A 5
16	Imperative command to create deployment yaml definition file	Creates deployment named 'redis' using image and no of replicas: k create deployment redis --image redis:alpine --replicas 1 -o yaml --dry-run=client > dp.yaml
17	Pod.yaml contains nodeName: node01	The pod will be scheduled in node01 irrespective of node01 tainted or label selector applied. nodeName is the highest priority.
18	Expose a deployment/rs	k expose deployment redis -n marketing --port=6379 --target-port=6379 --name=messaging-serviceservice k expose rs redis -n marketing --port=6379 --name=messaging-serviceservice
19	Creating generic secret	k create secret generic db-secret-xxdf --from-literal=DB_Host=sql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
20	Delete all pods that matches the label selector	k delete pod --selector="name=busybox-pod" --grace-period=0 --force
21	Know the api server or kube-proxy up and running	netstat -tulpn netstat -tulpn grep apiserver controlplane \$ netstat -tulpn grep apiserver tcp6 0 0 :::6443 :::* LISTEN 2041/kube-apiserver controlplane \$ netstat -tulpn grep kube-proxy tcp 0 0 127.0.0.1:10249 0.0.0.0:* LISTEN 3005/kube-proxy tcp 0 0 0.0.0.0:30080 0.0.0.0:* LISTEN 3005/kube-proxy tcp 0 0 0.0.0.0:30082 0.0.0.0:* LISTEN 3005/kube-proxy tcp 0 0 0.0.0.0:30083 0.0.0.0:* LISTEN 3005/kube-proxy tcp6 0 0 :::10256 :::* LISTEN 3005/kube-proxy
22	Kube-proxy to kill and restart	controlplane \$ netstat -tulpn grep kube-proxy tcp 0 0 127.0.0.1:10249 0.0.0.0:* LISTEN 3005/kube-proxy Kill -9 3005 controlplane \$ netstat -tulpn grep kube-proxy tcp 0 0 127.0.0.1:10249 0.0.0.0:* LISTEN 3005/kube-proxy The above command will make sure the kube-proxy is re-created. K get events -> to check kube-proxy is running or not.
23	How to run curl image and remove once you verified the pod/service is listening	controlplane \$ k get pods -o wide NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES myapp-pod 1/1 Running 0 14m 10.244.1.2 node01 <none> <none> nginx 1/1 Running 0 13s 10.244.0.4 controlplane <none> <none> controlplane \$ k run curl1 --image=curlimages/curl -i -t --rm --restart=Never -- curl 10.244.0.4 <!DOCTYPE html> <html> Once the command success, will terminate the pod. User --restart=Never other wise if pod fails to schedule will not terminate.
24	Fire up an interactive bash Pod within a Kubernetes cluster	kubect! run my-shell --rm -i --tty --image ubuntu -- bash
25	Josnpath query	controlplane \$ k get pods -o jsonpath="{image \t IP \n}"

		<pre>{range .items[*]}{.spec.containers[*].image}'{t}'{.status.podIP} {'\n'}</pre> <p>Output: image IP nginx 10.244.1.8 redis 10.244.1.9</p>
26	Custom-columns query	K get pods -o custom-columns="POD_NAME:.metadata.name,POD_STATUS:.status.containerStatuses[].state"
27	Verify a service is running or not using wget command	<pre>controlplane \$ k exec -it nginx-c main -- /bin/sh /# wget -S -T 1 nginx Connecting to nginx(10.44.0.7:80) HTTP/1.1 200 OK Server: nginx/1.21.0 Date: Wed, 09 Jun 2021 18:34:08 GMT Content-Type: text/html Content-Length: 8708 Last-Modified: Wed, 09 Jun 2021 18:34:07 GMT Connection: close ETag: "60c1099f-2204" Accept-Ranges: bytes wget: can't open 'index.html': File exists</pre>
28	Sortby- events creation time stamp	<pre>kubect! get events --sort-by=.metadata.creationTimestamp kubect! get events --sort-by=.lastTimestamp</pre>
29	Sortby in reverse order	kubect! get pods --sort-by=.status.startTime awk 'NR == 1; NR > 1 {print \$0 "tac"}'
30	Recent first 3 events	<pre>k get events --sort-by='{.lastTimestamp}' awk 'NR == 1; NR > 1 {print \$0 "tac"}' awk 'NR == 1, NR ==4 {print \$0}' Or k get events --sort-by='{.lastTimestamp}' awk 'NR == 1; NR > 1 {print \$0 "tac"}' head -n 4</pre>
31	Write the logs to a file at exam environment, mostly it will be write protected	K logs nginx sudo tee /var/log/log.txt
32	Events sort by lastTimestamp only last 10	k get events --sort-by='{.lastTimestamp}' tail -n 10
33	Create a resource quota for a namespace	kubect! create quota myrq --hard=cpu=1,memory=1G,pods=2 --dry-run=client -o yaml

3.2 Cheat Sheet - Linux Commands

Sl.No#	Function	Command
1	More command	Use more after the command. Ex: kubect! describe pod <pod-name> more
2.	Delete a word in vi	Use Esc+ dw -> deletes after the cursor one word Use Esc + db -> delete before the cursor one word Use Esc+d\$ to delete after the cursor point in the line Use Esc+d0 to delete before the cursor point in the current line
3.	Search and Replace in current line	/s/<searchText>/<replaceText>
4.	Search and Replace all	%s/<searchText>/<replaceText>/g
5.	Search and Replace all with confirmation	%s/<searchText>/<replaceText>/gc
5.1	Replaces all tab with spaces	%s/\t/ /g Note: number of spaces specified will be replaced with tab
6.	Delete multiple lines	Go VISUAL mode Shift+v Select lines d to delete

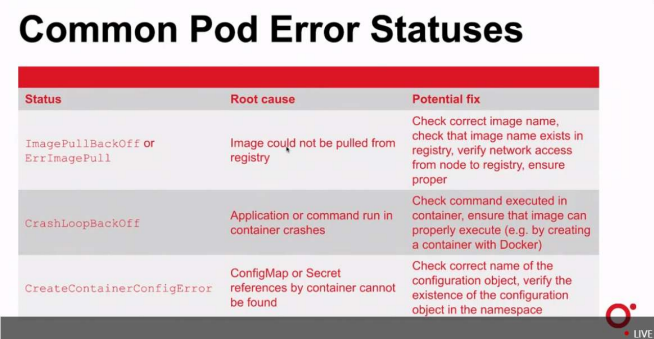
		Or :5d would remove line number 5 Or Esc + 5 dd
7	Grep match line , before and after n number of lines	Grep -i 'text to search' -A <number of lines after> -B <number of lines before matching text> <filepath/filename>
8	Copy and paste lines Cut and Paste line	To copy 5 lines Esc+ 5yy -> for copy and Esc+p -> for paste To cut 5 lines Esc+ 5dd -> cut And Esc + p > for paste
9	Begin of line and End of line	Begin -> Esc + 0 and End of line -> Esc+\$
10	Save and close	ZZ
11	Go to go specific line number while opening the file in vi	error: error parsing netpol2.yaml: error converting YAML to JSON: yaml: line 11: did not find expected key controlplane \$ vi netpol2.yaml +11
12	Run Command 5 Times	for i in {1..5}; do COMMAND-HERE; done
13	Indent aligning when you copy paste the yaml template	To indent multiple lines using vim you should set the shiftwidth using :set shiftwidth=2. Then mark multiple lines using Shift v and the up/down keys. To then indent the marked lines press > or < and to repeat the action press .
14	Vi auto convert tab to space	Vi ~/.vimrc set number Set tabstop=2 shiftwidth=2 expandtab
15	Grep multiple Strings	k describe pods grep -i 'Name\ Namespace'
16	awk	To reverse sorting and print awk 'NR == 1; NR > 1 {print \$0 "tac"}' To List only first 4 rows: awk 'NR == 1, NR ==4 {print \$0}' https://www.geeksforgeeks.org/awk-command-unixlinux-examples/
17	Move one word	Esc + w -> move one word forward Esc + W -> ignore special chars jumps to the next word Esc + b -> one word before Esc + B -> one word ignore special char.
18	Overwrites content to a file Appends to a file	> /opt/index.html >> /opt/index.html # appends to the file.
19	To check logged user and permission	\$ > id / \$ id uid=21(ftp) gid=21(ftp)
20	Vi ~/.vimrc set autoindent	Pasting does not work autoindent :r! cat and then paste (shift + insert) the content, and CTRL+D. Or use :set paste

Other Commands to remember:

Sl.No#	Function	Command
1	Curl with maxtime or timeout	Curl -m 5 http://localhost -m max time limit for 5 seconds
2	Wget to check service or url reachable or not	wget -o- http://google.com

		wget -o- nginx
		wget -S -T 5 nginx --> where nginx is svc name
3	Service output to a file	Wget -O <filepath/name> http://nginx curl -o <filepath/name> http://nginx Or else curl nginx:80 > /var/tmp/out.txt

3.3 Cheat Sheet - Kubernetes Concepts

Sl.No#	Concepts	When to Use	Link												
1	Pod, Multi-container Pod, Init-Container	<p>Multi-container Pod - A pod can have more than one container. There are various pattern available like side car, adaptor, ambassador patterns, these patterns are how the inter-containers communication which is the application running in the container behavior. As per the Kubernetes, we need to create a pod with multiple containers.</p> <p>Init Containers: A pod can have one or more init containers, which are run before the app containers are started. Init containers are exactly like regular containers, except:</p> <ul style="list-style-type: none"> • Init containers always run to completion. • Each init container must complete successfully before the next one starts. <p>Debugging Pod Error Statuses:</p>  <table border="1"> <thead> <tr> <th>Status</th> <th>Root cause</th> <th>Potential fix</th> </tr> </thead> <tbody> <tr> <td>ImagePullBackOff or ErrImagePull</td> <td>Image could not be pulled from registry</td> <td>Check correct image name, check that image name exists in registry, verify network access from node to registry, ensure proper</td> </tr> <tr> <td>CrashLoopBackOff</td> <td>Application or command run in container crashes</td> <td>Check command executed in container, ensure that image can properly execute (e.g. by creating a container with Docker)</td> </tr> <tr> <td>CreateContainerConfigError</td> <td>ConfigMap or Secret references by container cannot be found</td> <td>Check correct name of the configuration object, verify the existence of the configuration object in the namespace</td> </tr> </tbody> </table>	Status	Root cause	Potential fix	ImagePullBackOff or ErrImagePull	Image could not be pulled from registry	Check correct image name, check that image name exists in registry, verify network access from node to registry, ensure proper	CrashLoopBackOff	Application or command run in container crashes	Check command executed in container, ensure that image can properly execute (e.g. by creating a container with Docker)	CreateContainerConfigError	ConfigMap or Secret references by container cannot be found	Check correct name of the configuration object, verify the existence of the configuration object in the namespace	For Multi container pod and see how pod yaml are https://betterprogramming.pub/understanding-kubernetes-multi-container-pod-patterns-577f74690aee
Status	Root cause	Potential fix													
ImagePullBackOff or ErrImagePull	Image could not be pulled from registry	Check correct image name, check that image name exists in registry, verify network access from node to registry, ensure proper													
CrashLoopBackOff	Application or command run in container crashes	Check command executed in container, ensure that image can properly execute (e.g. by creating a container with Docker)													
CreateContainerConfigError	ConfigMap or Secret references by container cannot be found	Check correct name of the configuration object, verify the existence of the configuration object in the namespace													
2	Deployment Rollout pause and Resume	<p>A deployment can be paused while changing it is spec like image, volume etc using the command Kubectrl rollout pause deployment nginx Once all changes are done, it can be resumed using Kubectrl rollout resume deployment nginx</p>													
3	Logging Architecture	<p>Logging can be done in the following ways by the container:</p> <ol style="list-style-type: none"> 1. Basic container level logging When pod is evicted from a node, all logs are 2. Node level logging 3. Cluster level logging - Use a node-level logging agent that runs on every node. Ues Daemonset 4. Using a sidecar container with the logging agent 5. Exposing logs directly from the application 	https://kubernetes.io/docs/concepts/cluster-administration/logging/												
4	DaemonSet (not required for the exam)	<p>A DaemonSet ensures that all (or some) Nodes run a copy of a Pod (to schedule pod in a master node tolerations is used). Key Benefit: As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created. Sample Scenario: Running an logging agent is every node.</p>	https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/												
5	Network Policy	<p>By default any pod can communicate with any other pod in the cluster in same or different namespace. Two types of network policies 1. Ingress (all incoming traffic) 2. Egress (All outgoing traffic). By defining the network policy we can control the pods' ingress and egress traffic. There can be more than one network policy defined , when more than one policy applied for a pod it works based on OR condition i.e., policy1 OR policy2.</p> <p>When egress policy is configured always allow outgoing traffic on UDP/TCP ports 53 for DNS resolution. If this is not configured you will get error response saying bad domain/address because the pod can't resolve the service/DNS. Whereas this is not required for Ingress because it is an incoming request.</p>	https://reuvanharrison.medium.com/an-introduction-to-kubernetes-network-policies-for-security-people-ba92dd4c809d https://github.com/ahmetb/kubernetes-network-policy-recipes https://github.com/Tufin/test-network-policies/tree/master/tests												

A sample Egress policy:

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.

Answer

First we get an overview:

→ k -n venus get all

```
NAME                READY STATUS RESTARTS AGE
pod/api-5979b95578-gktxp 1/1 Running 0      57s
pod/api-5979b95578-lhcl5 1/1 Running 0      57s
pod/frontend-789cbdc677-c9v8h 1/1 Running 0      57s
pod/frontend-789cbdc677-npk2m 1/1 Running 0      57s
pod/frontend-789cbdc677-pl67g 1/1 Running 0      57s
pod/frontend-789cbdc677-rjt5r 1/1 Running 0      57s
pod/frontend-789cbdc677-xgf5n 1/1 Running 0      57s
```

```
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)  AGE
service/api         ClusterIP   10.3.255.137 <none>       2222/TCP  37s
service/frontend   ClusterIP   10.3.255.135 <none>       80/TCP    57s
...
```

(Optional) This is not necessary but we could check if the Services are working inside the cluster:

→ k -n venus run tmp --restart=Never --rm -i --image=busybox -i -- wget -O- frontend:80

Connecting to frontend:80 (10.3.245.9:80)

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

→ k -n venus run tmp --restart=Never --rm --image=busybox -i -- wget -O- api:2222

Connecting to api:2222 (10.3.250.233:2222)

```
<html><body><h1>It works!</h1></body></html>
```

Then we use any frontend Pod and check if it can reach external names and the api Service:

→ k -n venus exec frontend-789cbdc677-c9v8h -- wget -O- www.google.com

Connecting to www.google.com (216.58.205.227:80)

```
- 100% |*****| 12955 0:00:00 ETA
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
lang="en"><head>
...
```

→ k -n venus exec frontend-789cbdc677-c9v8h -- wget -O- api:2222

```
<html><body><h1>It works!</h1></body></html>
```

Connecting to api:2222 (10.3.255.137:2222)

```
- 100% |*****| 45 0:00:00 ETA
...
```

We see Pods of frontend can reach the api and external names.

vim 20_np1.yaml

Now we head to <https://kubernetes.io/docs>, search for NetworkPolicy, copy the example code and adjust it to:

```
# 20_np1.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: np1
  namespace: venus
spec:
```



```

podSelector:
  matchLabels:
    id: frontend    # label of the pods this policy should be applied on
policyTypes:
- Egress          # we only want to control egress
egress:
- to:             # 1st egress rule
  - podSelector:  # allow egress only to pods with api label
    matchLabels:
      id: api
- ports:         # 2nd egress rule
  - port: 53     # allow DNS UDP
    protocol: UDP
  - port: 53     # allow DNS TCP
    protocol: TCP

```

Notice that we specify two egress rules in the yaml above. If we specify multiple egress rules then these are connected using a logical OR. So in the example above we do:

allow outgoing traffic if
(destination pod has label id:api) OR ((port is 53 UDP) OR (port is 53 TCP))

Let's have a look at example code which wouldn't work in our case:

```

# this example does not work in our case
...
egress:
- to:             # 1st AND ONLY egress rule
  - podSelector:  # allow egress only to pods with api label
    matchLabels:
      id: api
  ports:         # STILL THE SAME RULE but just an additional selector
  - port: 53     # allow DNS UDP
    protocol: UDP
  - port: 53     # allow DNS TCP
    protocol: TCP

```

In the yaml above we only specify one egress rule with two selectors. It can be translated into:

```

allow outgoing traffic if
(destination pod has label id:api) AND ((port is 53 UDP) OR (port is 53 TCP))

```

Apply the correct policy:

```

k -f 20_np1.yaml create

```

And try again, external is not working any longer:

```

→ k -n venus exec frontend-789cbdc677-c9v8h -- wget -O- www.google.de
Connecting to www.google.de:2222 (216.58.207.67:80)
^C

→ k -n venus exec frontend-789cbdc677-c9v8h -- wget -O- -T 5 www.google.de:80
Connecting to www.google.com (172.217.203.104:80)
wget: download timed out
command terminated with exit code 1
Internal connection to api work as before:

→ k -n venus exec frontend-789cbdc677-c9v8h -- wget -O- api:2222
<html><body><h1>It works!</h1></body></html>
Connecting to api:2222 (10.3.255.137:2222)
-          100% |*****| 45 0:00:00 ETA

```

6	Labels and Selector		https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
7	Node Selector, Node Affinity, Pod Affinity and Pod Anti-Affinity	<p>Node Selector: is the simplest recommended form of node selection constraint. It specifies a map of key-value pairs.</p> <p>Node Affinity: Allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node.</p> <p>There are currently two types of node affinity, called <code>requiredDuringSchedulingIgnoredDuringExecution</code> (rules that must be met for a pod to be scheduled onto a node) and <code>preferredDuringSchedulingIgnoredDuringExecution</code></p>	https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#node-affinity

		<p>(preferences that the scheduler will try to enforce but will not guarantee.)</p> <p>The pod affinity and anti-affinity allow you to constrain which nodes your pod is eligible to be scheduled based on labels on pods that are already running on the node rather than based on labels on nodes.</p> <p>Pod Affinity -says that the pod should be scheduled onto a node if only if that node has already running a pod with matching labels/matchingExpressions.</p> <p>Pod anti-affinity - says that the pod should not be scheduled onto a node if that node has already running a pod with matching labels.</p> <p>As with node affinity, there are currently two types of pod affinity and anti-affinity, called <code>requiredDuringSchedulingIgnoredDuringExecution</code> and <code>preferredDuringSchedulingIgnoredDuringExecution</code> which denote "hard" vs. "soft" requirements.</p>	
8	Volumes Vs Persistent Volumes and Storage Class	<p>How is Local Persistent Volume different from a HostPath Volume?</p> <p>To better understand the benefits of a Local Persistent Volume, it is useful to compare it to a HostPath volume. HostPath volumes mount a file or directory from the host node's filesystem into a Pod. Similarly a Local Persistent Volume mounts a local disk or partition into a Pod.</p> <p>The biggest difference is that the Kubernetes scheduler understands which node a Local Persistent Volume belongs to. With HostPath volumes, a pod referencing a HostPath volume may be moved by the scheduler to a different node resulting in data loss. But with Local Persistent Volumes, the Kubernetes scheduler ensures that a pod using a Local Persistent Volume is always scheduled to the same node.</p> <p>While HostPath volumes may be referenced via a Persistent Volume Claim (PVC) or directly inline in a pod definition, Local Persistent Volumes can only be referenced via a PVC. This provides additional security benefits since Persistent Volume objects are managed by the administrator, preventing Pods from being able to access any path on the host.</p> <p>Additional benefits include support for formatting of block devices during mount, and volume ownership using <code>fsGroup</code></p> <p>-----</p> <p>This storageclass mode instructs Kubernetes to wait to bind a PVC until a Pod using it is scheduled.</p> <pre> kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: local-storage provisioner: kubernetes.io/no-provisioner volumeBindingMode: WaitForFirstConsumer </pre>	https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolume
9	Probes - Liveness, Readiness, Startup	<p>The kubelet uses liveness probes to know when to restart a container.</p> <p>The kubelet uses readiness probes to know when a container is ready to start accepting traffic.</p> <p>The kubelet uses startup probes to know when a container application has started. If such a probe is configured, it disables liveness and readiness checks until it succeeds, making sure those probes don't interfere with the application startup. This can be used to adopt liveness checks on slow starting containers, avoiding them getting killed by the kubelet before they are up and running.</p> <pre> startupProbe: httpGet: path: /healthz port: liveness-port failureThreshold: 30 periodSeconds: 10 </pre> <p>Thanks to the startup probe, the application will have a maximum of 5 minutes (30 * 10 = 300s) to finish its startup. Once the startup probe has succeeded once, the liveness probe takes over to provide a fast response to container deadlocks. If the startup probe never succeeds, the container is killed after 300s and subject to the pod's restartPolicy</p>	
10	The Downward API (<i>not required for</i>	<p>There are two ways to expose Pod and Container fields like <code>metadata.name</code>, <code>metadata.namespace</code>, <code>spec.serviceAccountName</code> and <code>spec.nodeName</code> etc to a running</p>	https://kubernetes.io/docs/tasks/inject-data-application/print/#the-downward-api

	<i>the exam)</i>	<p>Container as:</p> <p>1. Environment variables env: - name: MY_NODE_NAME valueFrom: fieldRef: fieldPath: spec.nodeName</p> <p>2. Volume Files volumes: - name: podinfo downwardAPI: items: - path: "labels" fieldRef: fieldPath: metadata.labels</p> <p>Together, these two ways of exposing Pod and Container fields are called the Downward API.</p>	
11	Projected Volume Storage(<i>not required for the exam)</i>	<p>A projected volume maps several existing volume sources into the same directory.</p> <p>Currently, the following types of volume sources can be projected:</p> <ul style="list-style-type: none"> • secret • downwardAPI • configMap • serviceAccountToken <p>All sources are required to be in the same namespace as the Pod.</p>	https://kubernetes.io/docs/concepts/storage/volumes/#projected
12	Job	<p>A Job creates one or more Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate. When a specified number of successful completions is reached, the task (ie, Job) is complete. Deleting a Job will clean up the Pods it created. Suspending a Job will delete its active Pods until the Job is resumed again.</p> <p>Important Fields To Remember .spec.backoffLimit to specify the number of retries before considering a Job as failed. The back-off limit is set by default to 6. For a fixed completion count Job, you should set .spec.completions to the number of completions needed. .spec.parallelism - specify number of pods running at any instant.</p> <p>The activeDeadlineSeconds applies to the duration of the job, no matter how many Pods are created. spec.activeDeadlineSeconds takes precedence over its .spec.backoffLimit. Therefore, a Job that is retrying one or more failed Pods will not deploy additional Pods once it reaches the time limit specified by activeDeadlineSeconds, even if the backoffLimit is not yet reached.</p> <p>Another way to clean up finished Jobs (either Complete or Failed) automatically is to use .spec.ttlSecondsAfterFinished, it will delete the Job cascadingly, i.e. delete its dependent objects, such as Pods, together with the Job. If .spec.ttlSecondsAfterFinished: 100 the job automatically deleted, 100 seconds after it finishes. If the field is set to 0, the Job will be eligible to be automatically deleted immediately after it finishes. If the field is unset, this Job won't be cleaned up by the TTL controller after it finishes.</p>	https://kubernetes.io/docs/concepts/workloads/controllers/job/
13	Understanding of cron and schedule expressions	<p>CronJobs are useful for creating periodic and recurring tasks(job), like running backups or sending emails.</p> <p>Important Fields To Remember spec.schedule: <code>"*/1 * * * *"</code> spec.jobTemplate:</p>	https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

Cron schedule syntax

```

# |----- minute (0 - 59)
# |----- hour (0 - 23)
# |----- day of the month (1 - 31)
# |----- month (1 - 12)
# |----- day of the week (0 - 6) (Sunday to Saturday;
# |                               7 is also Sunday on some systems)
# |-----
# * * * * *
  
```

Entry	Description	Equivalent to
@yearly (or @annually)	Run once a year at midnight of 1 January	0 0 1 1 *
@monthly	Run once a month at midnight of the first day of the month	0 0 1 * *
@weekly	Run once a week at midnight on Sunday morning	0 0 * * 0
@daily (or @midnight)	Run once a day at midnight	0 * * * *
@hourly	Run once an hour at the beginning of the hour	0 * * * *

To generate CronJob schedule expressions, you can also use web tools like <https://crontab.guru/>

Every minute -> * * * * * or */1 * * * * *

Every Hour -> 0 */1 * * *

Every day at 9:30 -> 30 9 */1 * * *

Every month 1st -> 0 0 1 */1 *

At 00:00 on every 15th day-of-month. -> 0 0 */15 * *

Every year on 1st Feb-> 0 0 1 2 *

14 Patching
Kubernetes
Resources (*not
required for the
exam*)

<https://www.runlev4.com/kubernetes/patching-kubernetes-resources>

<https://kubernetes.io/docs/tasks/manage-kubernetes-objects/update-api-object-kubectl-patch/>

Thank you for reading it till the end. Best wishes for your preparation, practice and the exam!!!!

