CKAD Resources

Table of Contents

CKAD Certification Tips/Resources

🔥 Reference Links

Register: https://www.cncf.io/certification/ckad/
Candidate Handbook: https://www.cncf.io/certification/candidate-handbook
Exam Tips: https://docs.linuxfoundation.org/tc-docs/certification/tips-cka-and-ckad

📋 Articles/Blogs

Basics

- Kubernetes Architecture : https://devopscube.com/kubernetes-architecture-explained/
- DNS in k8s : https://yuminlee2.medium.com/kubernetes-dns-bdca7b7cb868
- Essential Vim for CKAD :
  - https://github.com/jamesbuckett/ckad-tips/blob/main/06-just-enough-vi.md
  - https://www.vim-hero.com/

Troubleshooting

- Ingress/Service traffic flow: https://medium.com/@ManagedKube/kubernetes-troubleshooting-ingress-and-services-traffic-flows-547ea867b120

Courses

> Udemy: Kubernetes Certified Application Developer (CKAD) with Tests
> https://www.udemy.com/course/certified-kubernetes-application-developer

Practice

Resource	Link
Killer.sh Labs 🆓 (with CKAD Purchase)	https://killer.sh/ckad
(Github) CKAD Exercises 🆓	https://github.com/dgkanatsios/CKAD-exercises https://github.com/bmuschko/ckad-prep https://github.com/jamesbuckett/ckad-questions
Kubernetes Challenges 🆓	https://learn.kodekloud.com/courses/kubernetes-challenges
KillerCoda Labs 🆓	https://killercoda.com/killer-shell-ckad (*There are a lot of Community Labs as well: https://killercoda.com/explore?search=ckad&type=profile*)
Interactive Network Policies Tutorial 🆓	https://editor.networkpolicy.io/

CKAD Tricks/Tips

- Get list of all k8s Alias/Resource-Types : `kubectl api-resources` (similar to `Ctrl + A` in k9s)
- Avoid writing .yaml structure from scratch : `--dry-run=client -o yaml`
  - `kubectl -n namespace create deploy app --image=nginx:stable --replicas=2 --dry-run=client -o yaml > app-deploy.yaml`
    OR
    `kubectl -n namespace create deploy app --image=nginx:stable --replicas=2 --dry-run=client -o yaml | vim -`
- Reference Docs during test:
  - https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands
  - https://kubernetes.io/docs/reference/kubectl/quick-reference/
- **Not sure about the syntax of particular command?**
  - `kubectl explain <resource>`
    - Eg. `kubectl explain deployment --recursive` and pick any specific field to get description for it `kubectl explain pods.spec.containers`
- Running a temporary pod to troubleshoot connectivity
  - `kubectl run tmp --image nginx:alpine --restart=Never --rm -it -- /bin/sh`
  - If possible you can use `nicolaka/netshoot` image that contains a huge amount of utilities for troubleshooting instead of `nginx:alpine`

Forgot Something? Help yourself out during Exam Time!

TASK: Create a taint for a node named `node01`

- Let's open the current yaml in VIM to check it's properties: `kubectl get node node01 -o yaml | vim -` (**Use `vim -` to directly open from stdin**)
- Hmm ... no taints are applied, let's apply one ⟹ *Oh no! what's the syntax for taints?*: `kubectl explain node --recursive` OR `kubectl taint -h` (You'll get some example usages as well)
- Okay, so I just need to edit and save the yaml and it will be automatically applied right? *NOPE, not always*, sometimes you gotta force replace the resource with updated yaml manifest: `kubectl replace --force -f /tmp/file/path/stored/by/vim.yaml`

Local `minikube` setup for Practice

Ref. https://medium.com/geekculture/cheatsheet-for-kubernetes-minikube-kubectl-5500ffd2f0d5

```
# Check exiting profiles
minikube profile list

# Start/Stop a minikube cluster
minikube start --driver=docker --profile=minikube --memory 4096 --cpus=4
minikube stop

# Check Status
minikube status

# Install Kubernetes Dashboard
minikube dashboard

# Get minikube IP
minikube ip
```

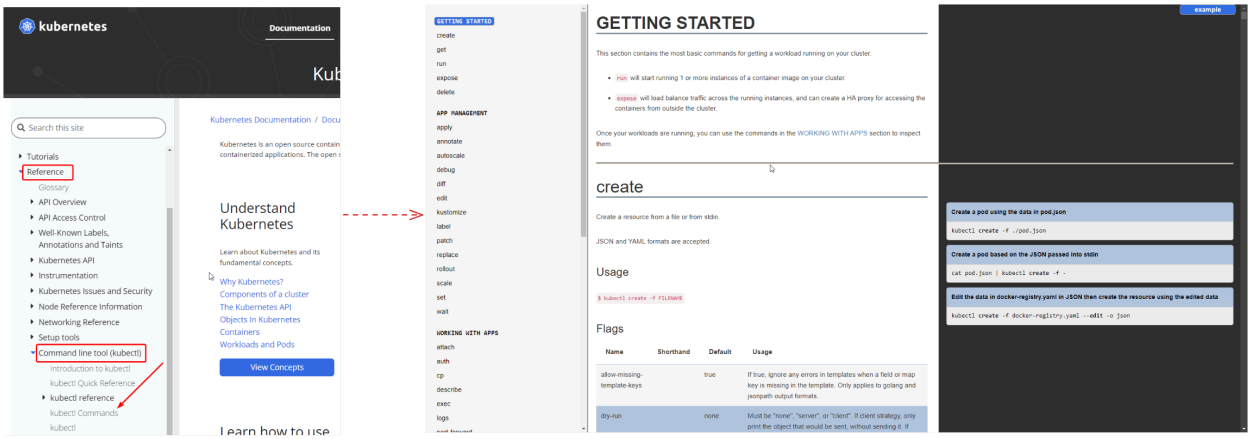
Bootstrap Exam Environment

1. Import following functions into your shell profile i.e. `vim ~/.bashrc`

```
# Please feel free to modify these as per your liking 😉

#############
# Functions #
#############

# It's ABSOLUTELY NECESSARY to use `"` instead of `'`
krmf() {
    kubectl delete "$@" --force --grace-period=0
}

kad() {
    kubectl apply -f "$@" --validate --dry-run=client
}

# Explain recursively a resource properties and open in Vim for easy searching
kexpr() {
    kubectl explain "$@" --recursive | vim -
}

```

```
#########
# Alias #
#########

alias k='kubectl'
alias kg='kubectl get'
alias kd='kubectl describe'
alias krm='kubectl delete'
alias kexp='kubectl explain'
alias kexec='kubectl exec -it'
alias kmock='kubectl "$@" --dry-run=client -oyaml'
alias ka='kubectl apply -f'
alias krepl='kubectl replace -f "$@" --force'
alias kedit='kubectl edit'
alias kubens='kubectl config set-context --current --namespace'
alias kubectx='kubectl config use-context'

# Eg. k run pod --image=nginx $dry
export dry="--dry-run=client"
```

2. Open the `kubectl` imperative commands reference



3. Configure Vim environment

```
# Create or open 'vim ~/.vimrc'
set nu      # show line numbers
set et      # expandtab (use space character when tab key used) *
set ts=2    # tabstop *
set sw=2    # shiftwidth *
set sts=2   # softtabstop (Let backspace delete indent) **

set ignorecase      # ignorecase when searching in vim
set smartcase

# Indentation
set ai      # autoindent (Indent at the same level of the previous line)
set si      # smart indent
```

```
14
15  # Highlighting
16  set hls     # highlightsearch (Highlight search terms)
17  syntax on   # syntax highlighting
18
19  # While editing files
20  # Incase of tab errors
21  :retab
22
23  # * preconfigured in the exam environment
24  # ** minimum nice to have
```

4. Transfer your `~/.vimrc` & `~/alias` files to each SSH server easily

```
1   # Add your vim settings to `~/.vimrc` & alias to `~/alias`
2   # Add the following function to your .bashrc on Host Node
3   copy() {
4       NODE=$2
5
6       scp ~/.vimrc ~/alias "$NODE":~
7       ssh "$NODE" 'cat alias >> ~/.bashrc && source ~/.bashrc'
8   }
9
10  # Since CKAD allows you to directly copy 'ssh ckad5206' text from the
    questions
11  # Just simply execute the above for each different server as follows
12  copy ssh ckad5206
13
14  # Then just paste the copied command again and all your config will
    just work!
15  ssh ckad5206
```

5. Useful Vim keyboard commands:

```
1   hjkl
2   ESC
3   i, a
4   I, A || 0, $
5   w, e, b
6   x, s, r
7   f, F || /, n, N
8   d, dw, D, dd
9
10  yy, p, P
11  v ⟶ y, p, P
12
13  ??autocmd FileType yaml setlocal ts=2 sts=2 sw=2 expandtab
```

```
14
15  Ctrl + G # Display filename
```

Docker

> Cheatsheets
>
> Commands
> A-Z reference

Helm Cheatsheet

List

- List installed releases:
  `helm list` *(add* `-n <namespace>` *for specific namespaces)*
  Example: `helm list -n my-namespace`
- Show release history (revisions):
  `helm history <release_name> [-n <namespace>]`
- Get post-install notes:
  `helm get notes <release_name> [-n <namespace>]`

Search for Charts

- Search within configured local repositories:
  `helm search repo <chart_name>`
- Search Artifact Hub (remote aggregator):
  `helm search hub <chart_name>`
  (Note: `helm search hub` *uses Artifact Hub, but requires web access and isn't configurable like* `repo` *)*
- View multiple versions of a chart (only for `search repo`):
  `helm search repo <chart_name> --versions | grep <version>`

> ⚠ `helm search hub` does **not** support the `--versions` flag.

Repository Management

- Add a new chart repository:
  `helm repo add <repo_name> <repo_url>`
- List added repositories:
  `helm repo list`

- Remove a repository:
  `helm repo remove <repo_name>`
- Update chart index across all repos:
  `helm repo update`

Install / Pull / Uninstall / Upgrade / Rollback Charts

- Install a chart:
  `helm install <release_name> <repo/chart>`
  or
  `helm install <release_name> ./<chart_folder>`

  Examples:
  - Basic install:
    `helm install my-nginx bitnami/nginx`
  - Dry run with debug info:
    `helm install my-nginx bitnami/nginx --dry-run --debug`
  - Specify chart version and override values:

```
helm install my-nginx bitnami/nginx \
  --version "13.2.34" \
  --set key1=value1 --set key2=value2 \
  --values custom-values.yaml
```

- Pull chart files locally (without installing):
  `helm pull <repo/chart> --untar`
  Example: `helm pull bitnami/nginx --untar`
- Install from local chart directory:
  `helm install <release_name> ./<chart_folder>`
- Uninstall a release:
  `helm uninstall <release_name> [-n <namespace>]`
- Upgrade a release:
  `helm upgrade <release_name> <repo/chart> [--version <ver>] [--values file.yaml]`
- Rollback to previous revision:
  `helm rollback <release_name> <revision_number> [-n <namespace>]`
  (Use `helm history <release_name>` to view revisions)