Experiment 04: Getting started with k3d

Install k3d from the binary, build from a tap, or build it custom.

https://github.com/rancher/k3d/releases

For MacOS:

$ brew install k3d

For Windows:

We'd download the binary here:

https://github.com/rancher/k3d/releases/download/v3.0.1/k3d-windows-amd64.exe

Install in c:\k3d or a bin folder for executing, althernatively the %USERPROFILE%\go\bin is commonly used for this executable

Create a project folder for our k3d experiements

C:\> **mkdir k3d**

or

$ **mkdir ~/k3d**

We already installed kubectl with kind, so won't need to reinstall.

C:\k3d> **dir**

```
 Volume in drive C is OS
 Volume Serial Number is 5081-CA53

 Directory of C:\k3d

09/09/2020  12:03 PM    <DIR>          .
09/09/2020  12:03 PM    <DIR>          ..
09/08/2020  10:05 PM         6,284,049 k3d-3.0.1.zip
09/08/2020  10:05 PM        22,014,464 k3d-windows-amd64.exe
               2 File(s)     28,298,513 bytes
               2 Dir(s)  175,237,222,400 bytes free
```

C:\k3d> **move k3d-windows-amd64.exe k3d.exe**

1 file(s) moved.

C:\k3d> **k3d version**

k3d version v3.0.1
k3s version v1.18.6-k3s1 (default)

C:\k3d> **k3d cluster list**

NAME SERVERS AGENTS LOADBALANCER

C:\k3d> **k3d cluster create demo --servers 3 --agents 3**

[36mINFO[0m[0000] Created network 'k3d-demo'
[36mINFO[0m[0000] Created volume 'k3d-demo-images'
[36mINFO[0m[0000] Creating initializing server node
[36mINFO[0m[0000] Creating node 'k3d-demo-server-0'
[36mINFO[0m[0001] Pulling image 'docker.io/rancher/k3s:v1.18.6-k3s1'
[36mINFO[0m[0089] Creating node 'k3d-demo-server-1'
[36mINFO[0m[0090] Creating node 'k3d-demo-server-2'
[36mINFO[0m[0091] Creating node 'k3d-demo-agent-0'
[36mINFO[0m[0092] Creating node 'k3d-demo-agent-1'
[36mINFO[0m[0094] Creating node 'k3d-demo-agent-2'
[36mINFO[0m[0096] Creating LoadBalancer 'k3d-demo-serverlb'
[36mINFO[0m[0097] Pulling image 'docker.io/rancher/k3d-proxy:v3.0.1'
[36mINFO[0m[0158] Cluster 'demo' created successfully!
[36mINFO[0m[0158] You can now use it like this:
kubectl cluster-info

In our example, you'll see that we've setup 3 servers (Kubernetes masters) in our control plane, and 3 agents (Kubernetes nodes) in our data plane.

You'll also see that we have the Load Balancer, k3d-demo-serverlb, which is our containerized Traefik instance running in our cluster.

Kubectl won't know about this cluster until we load and set our KUBECONFIG environment variable.

C:\k3d> **k3d cluster list**

NAME SERVERS AGENTS LOADBALANCER
demo 1/3 2/3 true

C:\k3d> **k3d kubeconfig get demo**

---

apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJWekNCL3FBREFnRUNBZ0VBTUFvR
0NDcUdTTTQ5QkFNQ01DTXhJVEFmQmdOVkJBTU1HR3N6Y3kxelpYSjIKWlhJdFkyRkFNVF

U1T1RZM01URTJOakFlRncweU1EQTVNRGt4TnpBMk1EWmFGdzB6TURBNU1EY3hOekEyT
URaYQpNQ014SVRBZkJnTlZCQU1NR0dzemN5MXpaWEoyWlhJdFkyRkFNVFU1T1RZM01U
RTJOakJaTUJNR0J5cUdTTTQ5CkFnRUdDQ3FHU000OUF3RUhBMElBQkR4cWlSWnI2cVUy
R25GYjQ1UjdTU2ljVmdFSC9RNEY3V3dBTkQxdU9uazUKOFIwVGVNRUh1eTYwN0ZXeWlqaz
VkeFJ3WjBOaUlybjcrSW1EOUVia2FmaWpJekFoTUE0R0ExVWREd0VCL3dRRQpBd0lDcERB
UEJnTlZIUk1CQWY4RUJUQURBUUgvTUFvR0NDcUdTTTQ5QkFNQ0EwZ0FNRVVDSUU2Nn
FaRkVuZ1BuClN3TmE2bU1wN1ZKd1UvN2FValdGM0s0Z1o1OWhzd29CQWlFQTgxY241UjA2
RTEzYndQdXJORjlMTlZXL0l5UzMKeEFEK1EyM2QwVUMvYk1nPQotLS0tLUVORCBDRVJUS
UZJQ0FURS0tLS0tCg==
    server: https://0.0.0.0:53948
  name: k3d-demo
contexts:
- context:
    cluster: k3d-demo
    user: admin@k3d-demo
  name: k3d-demo
current-context: k3d-demo
kind: Config
preferences: {}
users:
- name: admin@k3d-demo
  user:
    password: 6a4ad9aadd405b3dcffc77b5f12c46d5
    username: admin

C:\k3d> **k3d node list**

NAME	ROLE	CLUSTER	STATUS
k3d-demo-agent-0	agent	demo	running
k3d-demo-agent-1	agent	demo	running
k3d-demo-agent-2	agent	demo	exited
k3d-demo-server-0	server	demo	exited
k3d-demo-server-1	server	demo	exited
k3d-demo-server-2	server	demo	running
k3d-demo-serverlb	loadbalancer	demo	running

C:\k3d> **mkdir .kube**

C:\k3d> **cd .kube**

On Windows:

C:\k3d> **set KUBECONFIG_FILE=C:\k3d\.kube\demo**

C:\k3d> **k3d kubeconfig get demo > %KUBECONFIG_FILE%**

C:\k3d> **set KUBECONFIG=%KUBECONFIG_FILE%**

On MacOS or Linux

~/k3d/.kube $ **export KUBECONFIG_FILE=~/.kube/demo**

~/k3d/.kube $ **k3d kubeconfig get demo > $KUBECONFIG_FILE**

~/k3d/.kube $ **export KUBECONFIG=$KUBECONFIG_FILE**

Verify we have our file set correctly, remember this is relative and requires us to execute commands from the “k3d” folder to be effective

C:\k3d\.kube **set | grep KUBE**

KUBECONFIG_FILE=.\.kube\demo

C:\k3d> **k3d cluster list**

NAME SERVERS AGENTS LOADBALANCER
demo 1/1 3/3 true

For MacOS:

~/k3d $ **cat $KUBECONFIG_FILE**

For Windows:

C:\k3d> **type %KUBECONFIG_FILE%**

---

apiVersion: v1
clusters:
- cluster:
certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJWekNCL3FBREFnRUNBZ0VBTUFvR
0NDcUdTTTQ5QkFNQ01DTXhJVEFmQmdOVkJBTU1HR3N6Y3kxelpYSjIKWlhJdFkyRkFNVF
U1T1RZM01qUTRPVEFlRncweU1EQTVNRGt4TnpJNE1EbGFGdzB6TURBNU1EY3hOekk0TU
RsYQpNQ014SVRBZkJnTlZCQU1NR0dzemN5MXpaWEoyWlhJdFkyRkFNVFU1T1RZM01qUT
RPVEJaTUJNR0J5cUdTTTQ5CkFnRUdDQ3FHU000OUF3RUhBMElBQkdmRm53RUtycFVtbV
h3ckVFUFdaYSsxZWdYQWhPV2ZUZEorZU94UWo4U3kKUDgzSTJQbDYrTUQ4OUNMTlRTb
E1Ebk5pM3FvS1N0ZHdGZFRhOFRHQUxTS2pJekFoTUE0R0ExVWREd0VCL3dRRQpBd0lDc
ERBUEJnTlZIUk1CQWY4RUJUQURBUUgvTUFvR0NDcUdTTTQ5QkFNQ0EwZ0FNRVVDSUF
VOGpaQ0RORkhMCkpDVkdOd2I2UXhxS0xPekp1NUtYV2JNdGZ0VVB4Ymc4QWlFQXNkQXF
JRm90R2JPcVk4OUxudU45eStrTU44M1AKU1pPWWRGMEIyNUV2dXgwPQotLS0tLUVORCB
DRVJUSUZJQ0FURS0tLS0tCg==
server: https://0.0.0.0:6550
name: k3d-demo
contexts:

```
- context:
    cluster: k3d-demo
    user: admin@k3d-demo
  name: k3d-k3d-rancher
current-context: k3d-demo
kind: Config
preferences: {}
users:
- name: admin@k3d-demo
  user:
    password: dd79f910ebe64a30855bcd38b7425b98
    username: admin
```

C:\k3d> **kubectl cluster-info**

```
Kubernetes master is running at https://0.0.0.0:6550
CoreDNS is running at https://0.0.0.0:6550/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://0.0.0.0:6550/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy
```

C:\k3d> **k3d cluster delete demo**

```
[36mINFO[0m[0000] Deleting cluster 'demo'
[36mINFO[0m[0001] Deleted k3d-demo-serverlb
[36mINFO[0m[0001] Deleted k3d-demo-agent-2
[36mINFO[0m[0002] Deleted k3d-demo-agent-1
[36mINFO[0m[0003] Deleted k3d-demo-agent-0
[36mINFO[0m[0003] Deleted k3d-demo-server-2
[36mINFO[0m[0003] Deleted k3d-demo-server-1
[36mINFO[0m[0003] Deleted k3d-demo-server-0
[36mINFO[0m[0003] Deleting cluster network '7f899c3403da533a8429f782ed2d5e1090d8eaaa605a886cba48c4d36ecc4413'
[36mINFO[0m[0003] Deleting image volume 'k3d-demo-images'
[36mINFO[0m[0003] Removing cluster details from default kubeconfig...
[36mINFO[0m[0003] Removing standalone kubeconfig file (if there is one)...
[36mINFO[0m[0003] Successfully deleted cluster demo!
```