Experiment: Working With Kind

In the Foundation experiment/lab we installed Docker and KIND, validated our Docker installation, and now we'll move to using KIND. KiND/kind/KIND is a development tool for running a Kubernetes cluster as docker containers. This greatly simplifies working with cloud-native application modernization and microservices vs. running a full k8s cluster on your development environment. Because this runs in containers even a computer with 8GB of RAM can be used to effectively develop. KiND is one of two options that we'll use for running Kubernetes in our development environments, and is a tool developed by the Google team.

For MacOS

Start a terminal and cd to ~/Projects/kind

For Windows

Run Git Bash from the c:\Projects\kind folder

For MacOS and Windows

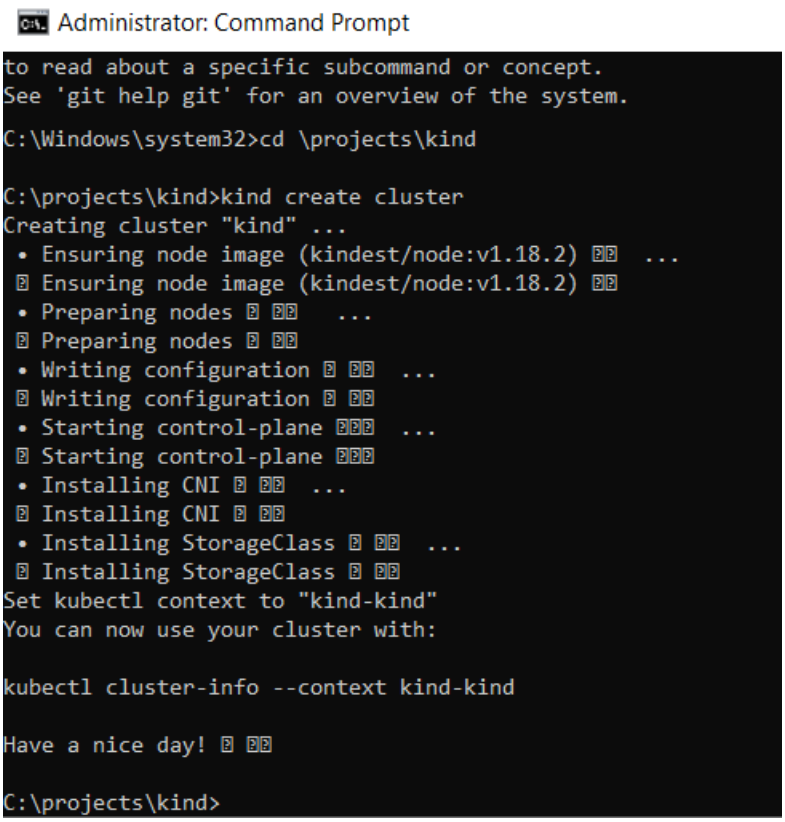
Create our projects folder if that does not already exist

$ mkdir ~/Projects

$ mkdir ~/Projects/kind

$ cd ~/Projects/kind

$ kind create cluster



Super simple, execute “kind create cluster”, and go grab a refill on your favorite beverage while it spins for a few minutes depending on your network speed. Any delay in our initial cluster loading is in grabbing the images we need to build our cluster.

$ **kind get clusters**

kind

Check the kubectl client version

$ **kubectl version --client**

Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.0", GitCommit:"e19964183377d0ec2052d1f1fa930c4d7575bd50", GitTreeState:"clean", BuildDate:"2020-08-26T14:30:33Z", GoVersion:"go1.15", Compiler:"gc", Platform:"windows/amd64"}

Check the kind cluster we just created

$ **kubectl cluster-info --context kind-kind**

Kubernetes master is running at https://127.0.0.1:51089

KubeDNS is running at https://127.0.0.1:51089/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

Notice we need to use the --context or set our default context to kind-kind

The newest installation of KIND automagically creates the ~/.kube/config file targeted at kind-kind, so you won't have to create that file. If your ~/.kube/config file is not created you will have to do that.

$ **cd ~**

$ **mkdir .kube**

$ **cd .kube**

$ **nano config**

The file should look similar to below, and of course the server port reference will be from the information that was returned from your "kubectl cluster-info"

apiVersion: v1

clusters:

- cluster:

certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJd01Ea3dPREF3TWpJME0xb1hEVE13TURrd05qQXdNakkwTTFvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTHJJCk9oYXpaZFJNamx1WFh5aTQ0cFRBbkxPZEZ5QlJVb0FlRmZHUTlzRzd3UkJWMzNMaU5DUDRCMlNvUTZUb20zejMKK2pydUZsUWhZK0grQ1A3UjByRGMwUmozbXdHM0dPckJYcnRBNzB6N2MrTUdMMm5QaEt0OTZ4Q0lNdVIvZ1NUbgpIMVFuY0pkM3I1T3RGd1I2czdGbWxyb1RRNkFoVE5GZktBMTV2UHZ4Mmw2bEV3a0JGRlMrNjZ3dERPT2dJZy9CCklwQkIzcWZ3bzYwbFRFUGorOEczbnBEYTdPNkVTNXNmZFlBZWJGTndlT21DL0cvdnN

2Yks4MXV0anNNM1oyWmgKNUExNkxnUlpNOXk2Rzl2blluYjNlekM2UWNMVG9YVm9
4YU95d0l0MFJIaXBIVkN3U2hKRG9qcjU0ZTBTWXlncQo5MlFpeE1GWEphUndVc1M0
eVU4Q0F3RUFBYU1qTUNFd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd
0VCCi93UUZNQU1CQWY4d0RRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFMVUxZMk
NXTVlNSlJUb0dQZGVWSmQ3MmZoL24KblFGWHRYT3dQbldmMElSR0grYTVwOVcy
ZytKRzQ4QjUwSUtZWFErVEZOTVZ0QUYrMDFWTFJRbzFQUEhudTM5NQppYlpHSn
NybTBhWGVwOHFlSWF2Lzl0MUxHcWk5dThkdFV6cjYvVEkwK2dUK2lEZlZ3bHRYSlZ
iVXN4Q2U1VE5UCmdQTW9GWjBOdDVCTHUzRkdrV0x3OXRNSlkyU0JNckNZZ0ZhZ
Hg5bkVSMjd4L0NsdjEvR3picEhWRmJmYVlubmQKUFNZbVRiWncxYlgyZzJBdC9pQ1
BSbDJGbXYzbXBTdENjaEppVDNkK0U5aEtpV2p4T0kvVDFsN2E5UWxJTXdpdApldjZo
MlJkUTQvZVdkeTFxY2hDYkZ6N0UyZEtvSXRONFFseFhQcVlMQ1VnYzB4TEFGK2kx
dkV1bDM3Zz0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=

    server: https://127.0.0.1:51089

  name: kind-kind

contexts:

- context:

    cluster: kind-kind

    user: kind-kind

  name: kind-kind

current-context: kind-kind

kind: Config

preferences: {}

users:

- name: kind-kind

  user:

    client-certificate-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM4akNDQWRxZ0F3SUJBZ0lJ
U3VyaXQxdGJKNnN3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQ
XhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TURBNU1EZ3dNREl5TkROYUZ3MHlNVE
E1TURnd01ESXlORFJhTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4
wWlhKek1Sa3dGd1lEVlFRREV4QnJkV0psY201bGRHVnpMV0ZrCmJXbHVNSUlCSW
pBTkJna3Foa2lHOXcwQkFRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXo1K1d0L242
RGtEbm5QblcKMFVmb3kxM1ArSldjQi9BMXdBTTU3L2RSREpIZ1g4N1lqOFRibkRDdU
NhcElxVFY0ZjBqdW9JdDRDZmNKQUxMVgpXTWxhYVhncmljSXgxejdBdzlEUDVNRH
M2VkdJcWRKaGc1Z0ROQWVYdTIzZUNuaUgxVDBnUFJxNjdKWHY4YW9VCi9YblY3
UFdXNDVGbDlvd0NwSEc5Vlp4MUNIaVVOd1RVb09pd29oYzhpWmh6QzNHdS9CZ3F

RbDR5K3hMVU0wTDkKWXZEY0NBU2dYdEhCQkU3TWN2SWxIQWFFdmdKVVk0c1
MvMUhZMytZQmZVcCtzUzZjcHBKWFpveXhaTGdaMm51YgpTYmVhQ0J0Wk9wbGR3
YTgvV29SbGtrRkFpK0RURkt3cEZmTC9DYkxqOHd3QTVXdnNtWVdyWkFRZGE2aTB
KQ2k5CmdaMzhCUUlEQVFBQm95Y3dKVEFPQmdOVkhROEJBZjhFQkFNQ0JhQXdF
d1lEVllwbEJBd3dDZ1lJS3dZQkJRVUgKQXdJd0RRWUpLb1pJaHZjTkFRUxCUUFEZ
2dFQkFCOS9FL3o1cnNBeDNoaDVxVlo3ZUswTy95bUtES05SOFVDTApwc21kaVl2dm
1qZEVkc0I2WFhWM2dMeWQ1TmF0SSt2WU5rbS9YODlqNTlURXNVeGhtS2V2SEorQ
mRZWXBPcEQ5CmY3Y3dBNnNIS2lEcGJUMVVuajRoeHJoZyt6VEFXN3hleHZEam8v
dG5xazg1dDR1bzVLdFVrRjRFNEN1QXQ3TFUKUjJaTlBta2UwdTRsUWgzaFU5Slg1bz
Y5N3FHb21EMkZTeDdvb212eXdDMS8rbFBlVkNURk95U2pPbXEya001VApiZCtRbmh
oeVVTRXJheC9sbnV5SDhkUlJYZkZWelk1WElLNS81RnBBMFFUUkltRXJUVExSem5
mcktNQXVaS3VTCjVFL0d4Z083MU1vZGN3ajM2NWZKWE9JT1c0UlZ4TXJabEpnQVZ
Ia0lGUitQSUV0blErRT0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=

client-key-data:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVB
ejUrV3QvbjZEa0RublBuVzBVZm95MTNQK0pXY0IvQTF3QU01Ny9kUkRKSGdYODdZ
Cmo4VGJuREN1Q2FwSXFUVjRmMGp1b0l0NENmY0pBTExWV01sYWFYZ3JpY0l4M
Xo3QXc5RFA1TURzNlZHSXFkSmgKZzVnRE5BZVh1MjNlQ25pSDFUMGdQUnE2N0p
Ydjhhb1UvWG5WN1BXVzQ1Rmw5b3dDcEhHOVZaeDFDSGlVTndUVQpvT2l3b2hjOG
laaHpDM0d1L0JncVFsNHkreExVTTBMOVl2RGNDQVNnWHRIQkJFN01jdklsSEFhRXZ
nSlVZNHNTCi8xSFkzK1lCZlVwK3NTNmNwcEpYWm95eFpMZ1oybnViU2JlYUNCdFp
PcGxkd2E4L1dvUmxra0ZBaStEVEZLd3AKRmZML0NiTGo4d3dBNVd2c21ZV3JaQVFk
YTZpMEpDaTlnWjM4QlFJREFRQUJBb0lCQUVzYmxFNWhvOC9jTXUxYQpoQmVaUit
HcHdqNVBBTzd1T3NPSFowSWoyYklPWTNqRlB4cGpRSDYwTFlGWmxJZUJ6R0Zm
Wk5PM0lHbWFjQ3RNCmhsbGtIY3pocC81aHZkMzcyWWY4MWZnT3dxVjAxVmQ1djh
UM0ROR1puWTQwSklydEoxWkFrcFVJUW02cm51MXgKZGI1c2dMTUQ5TjNHRDNpd
EZaZWZmYnFtcXIreEYvN2tTQ0ZKSEpyR0JoMk9GT3JSTXdxUXREaFBJWS9DTzN6
VQoybVBOTXN2Rk5JQjBoK2lKVXEvWEpwOHAvd3hmYkYva3FuRXBEYTZhaTFXT21
NdzhuQjFtRFZqSjR3TWtZdEkwCjZrTWFTdHYvVVBBeVIvZzFvV0ZTeGFGcGs4OEh6
UUp1ZnpXamZ6c2Z4bDN4ZnNwY1FKNzNZUlZzb29kRFdWTEUKRkwxVFRBRUNnW
UVBODVVWklMcHp3M0lSa21rQkJOT0VLYk5wWWNOZEYyOUNDOGNuYWJRMFdM
NVlqUWEzdjBYMgo5YXdiVllQVXBCck1wZUY0VlNPalhSTWpIazkwZDIvM1dwRlkxdjJV
UjBSTXdrYy9MVXNIVjJ6S2xUczZmanZFCi8vU0VTRm9FS2ZmWmkzbjAzUm81R3gvd
WZxK1YwUnl6L0pNMzlKeDI5N0o5NHVHdE1FSEVCY1VDZ1lFQTJqVXoKK1VzSnRB
Q2NwNkJ3TDNOV1h3U29aMk9nbzZnYjBOclpTUW9YbnNwdGt0UmF0R3NER3JTWG
1CeGJNRUloVklERQpKbDBNTGc3QnFEaDcwVklBbmNJWmQzQkhNd0NYMmgzNDN
6NmJZSTZ1RFBSU1k0amd4WHB0djBaVDdIV1ZDVFgwCk9TWnArYkpuMFpJb3gwSz
BHc0FIcDk2WThIZnRReUoyU2FOendVRUNnWUVBbytzWFVONEluMk01RGdVWnlXe
XEKQ1FJU2pkYll0NzVjZk42VjJGMkx5Smkzc0pmdnVZbFV5emo1NEE5cVh0RW1IUTl
oVWpJOGNwczVTK0YrYUEzeQpNSUdWZm9DQmM0QTBBNTI4bHpkaGhtVFE0Nkp
MRjc0VE1ZZ1VLVGhpaXZlZTcyeXY2c2NGM1FvZERpWU5OUDhTCjVJc1l0Y3dWaDF
VWHdFSE1zYWZnU1YwQ2dZRUFpUmY2RW5zeG1uVHo5MkVXZXNsMUQzZW1zbV
ptcTh2WHhnMXAKakxrWmcvdGNnbTZHbW1uTlpuN2w2M3lOVWpHS0xVUkZlSERuV

VJ5V1VURkRvWXhxdExNWk92QkEyMnRZL0lIdQpOWnhwRkc0d0xoVm1tZ0NLYjZmd
XdjaId1MjVZZDVQOVg5YWhxRzZOU1o4Um5iZHIzbzZkZ0x1YXpTSWI0QjBMCndsSSt
UTUVDZ1lCSTFrek5EeHhWUFVucERleXNnWmxlS3QwcDltTzhKWjlGcU5UM044eXY1
OEZnbitkSFU2WmYKWXZUNDdkK3FxdHRYREJEa1ZlWkJ4KzVSdW5kWlgxM3NKUk
g2NUhVeE5QczBkdGVLa3RObnh5Zm9ubEFZTU13QgorWXBlMFVUT2xaK3FOT3d1U
zFkZHM2MU14SVZWRlJxTDVzVGlVZGdkU3VLenU3ZW41R2hIY3c9PQotLS0tLUVOR
CBSU0EgUFJJVkFURSBLRVktLS0tLQo=

Create a folder under your projects or profile folder and switch directory to our new folder

~: kubenerd $ **cd ~**

~: kubenerd $ **mkdir projects/kind**

~: kubenerd $ **cd projects/kind**

For both Windows and MacOS:

~: projects/kind $ **kind create cluster**

~: projects/kind $ **mkdir kind-wp**

~: projects/kind $ **cd kind-wp**

~/projects/kind/kind-wp $ **vi kustomization.yaml**

Note: For folks who are not vi users, use nano editor

~/projects/kind/kind-wp $ **nano kustomization.yaml**

Paste the following content into our file and save or alternatively you can download this from the project GitHub repo here under labs folder or by cloning the repo locally. This uses a Secrets Generator. Since Kubernetes v1.14, kubectl supports managing objects using Kustomize. Kustomize provides resource Generators to create Secrets and ConfigMaps. The Kustomize generators should be specified in a **`kustomization.yaml`** file inside a directory. After generating the Secret, you can create the Secret on the API server in our Kubernetes cluster with **`kubectl apply`**

The following is an example that we'll use for using Kustomize to create a MySQL Database password.

secretGenerator:

```
- name: mysql-pass
  literals:
  - password=VerySecure2020
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

The formatting of a YAML file is key to the stanzas. The indenting in a YAML file is how it groups the information, similar to Ruby coding. This eliminates the need for curly braces that you see in JSON or similar types of groupings in popular programming languages. This file exists in the GitHub repo in the event that folks have issues with the cut/paste formatting.

Download the following two resource files **~/projects/kind/kind-wp** folder

For Windows

If you don't have wget for your Windows environment, you can install with chocolatey

```
D:\projects\kind\kind-wp> choco install wget
Chocolatey v0.10.15
Installing the following packages:
wget
By installing you accept licenses for the packages.
Progress: Downloading Wget 1.21.1.20210323... 100%
```

Use our wget to pull the files to complete this experiment.

For both MacOS and Windows

```
$ wget https://kubernetes.io/examples/application/wordpress/mysql-deployment.yaml
$ wget https://kubernetes.io/examples/application/wordpress/wordpress-deployment.yaml
```

Copy or move the two resource YAML files that we downloaded to the kind-wp folder that you created, if there were not saved there.

From the kind-wp folder run the kubectl apply

$ **kubectl apply -k ./**

Run the following commands to inspect the deployment

$ **kubectl get secrets**
$ **kubectl get pvc**
$ **kubectl get pods**

Once the “get pods” shows status as Running, rather than ContainerCreating we’ll check the services for wordpress

$ **kubectl get services wordpress**

Try accessing the normal default wordpress at the following port.

Open http://localhost:8080/ in your web browser

This will result in a 404, and we’ll correct that defect in our next update

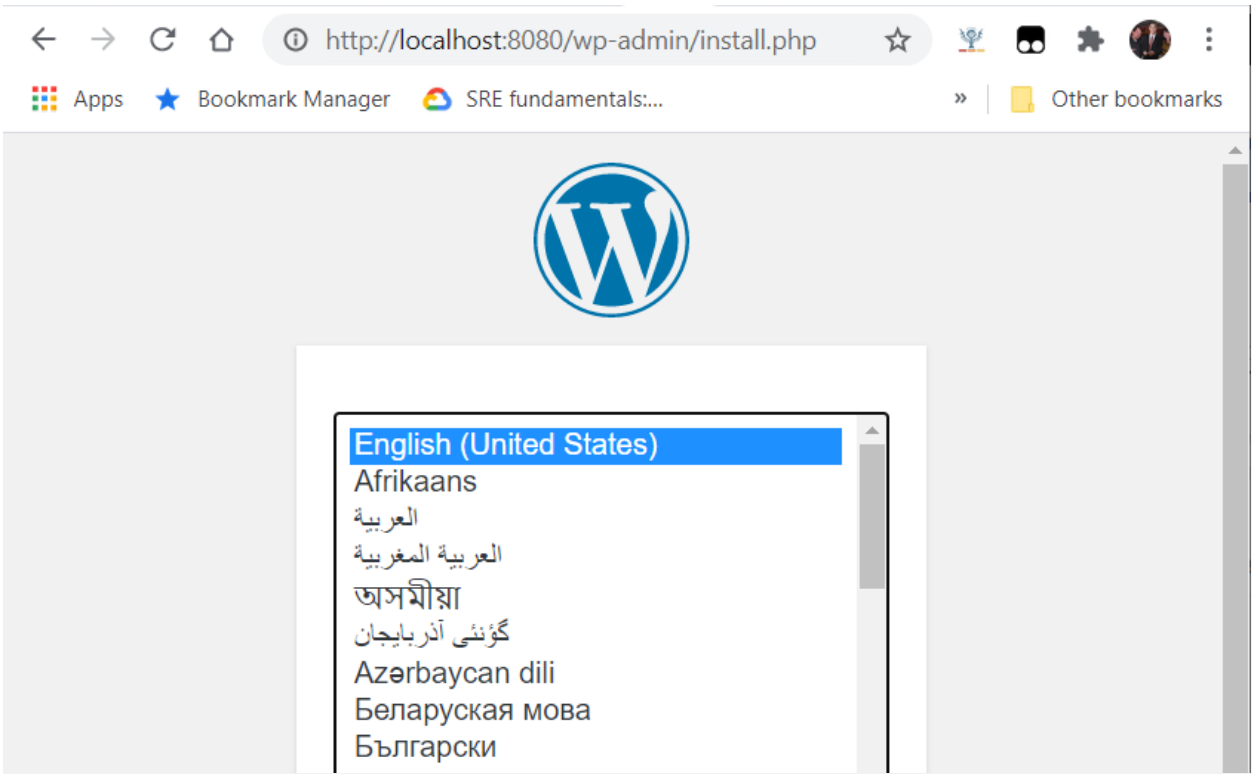
Note: We see that this is refused but recall that we see our service, but have not enabled a port forward to get to that WordPress instance

Create a port forward to allow us to access our Wordpress environment

~/projects/kind/kind-wp$ **kubectl port-forward svc/wordpress 8080:80**

Now reopen or refresh

http://localhost:8080/

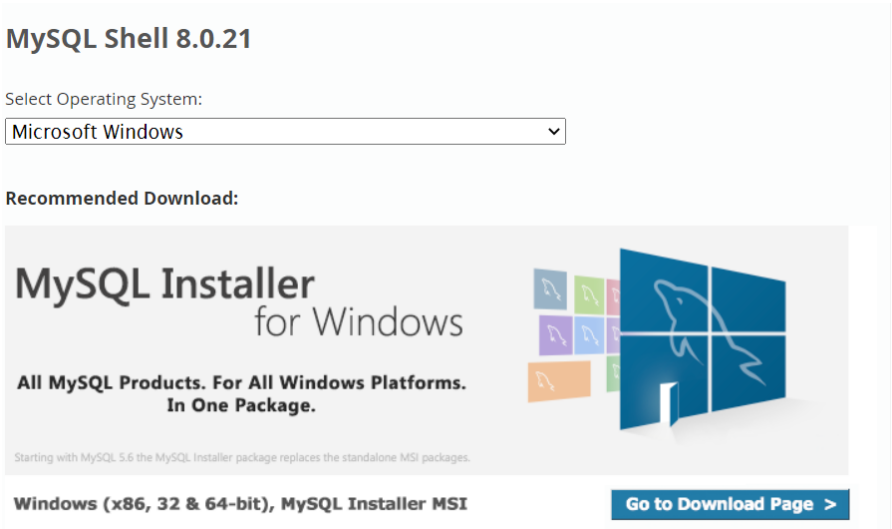


For MacOS:

$ brew install mysql-client

For Windows: Download the MySQL Shell if you don't already have it from the below URL

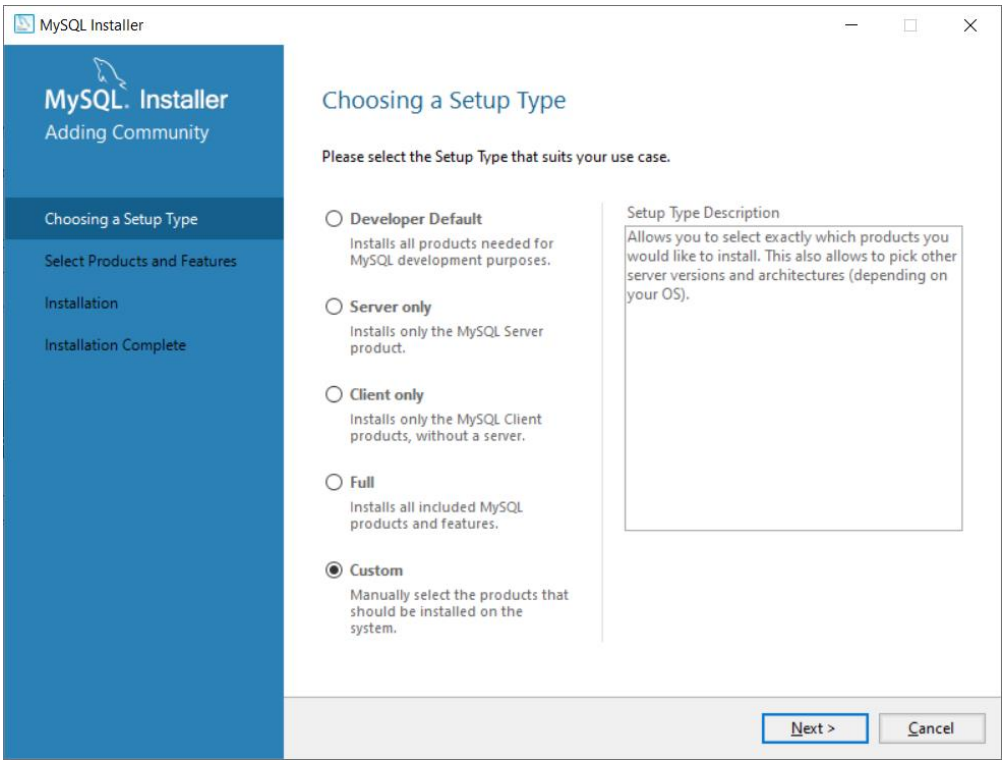
https://dev.mysql.com/downloads/shell/



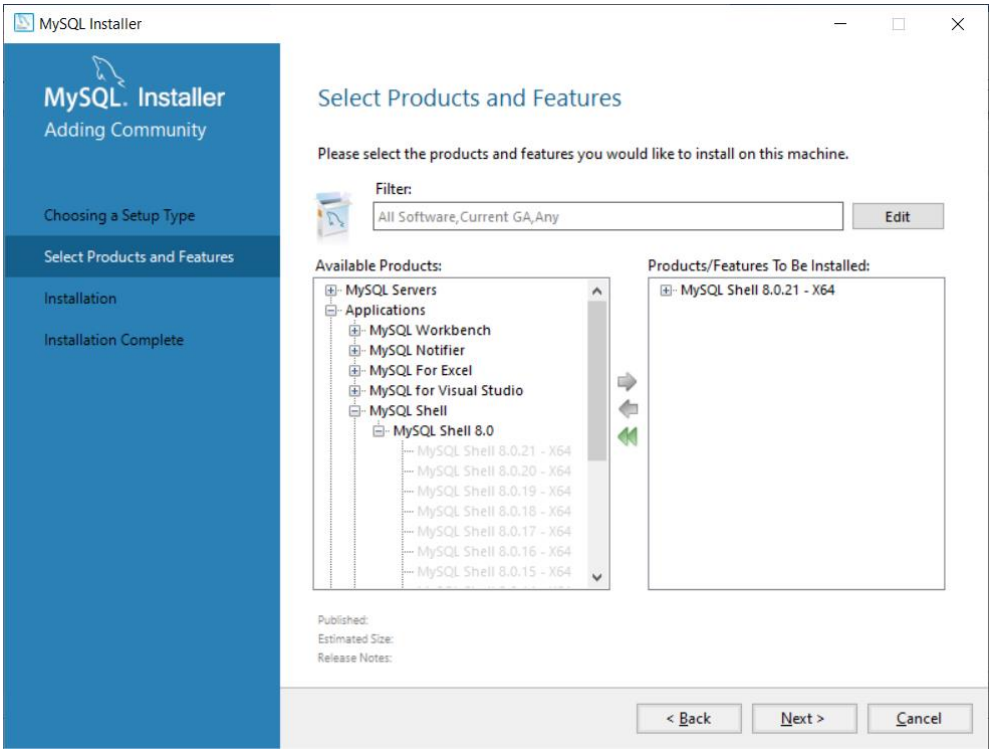
If you have an Oracle account, you can login or you can select “No thanks, just download”

Install the MySQL Shell, by running the downloaded installer

Select the “Custom” unless you prefer to install the server and such locally.

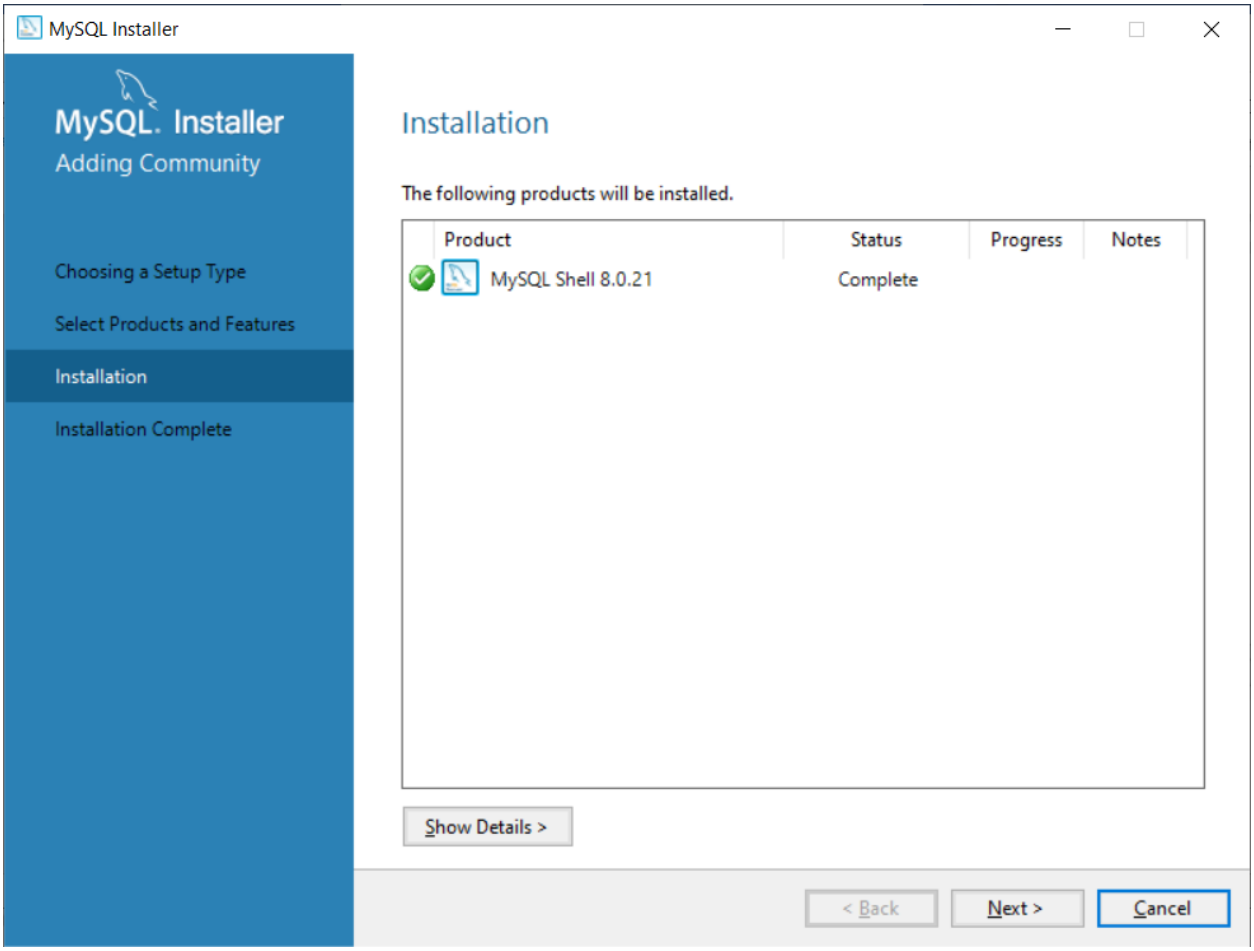


Select “Next”

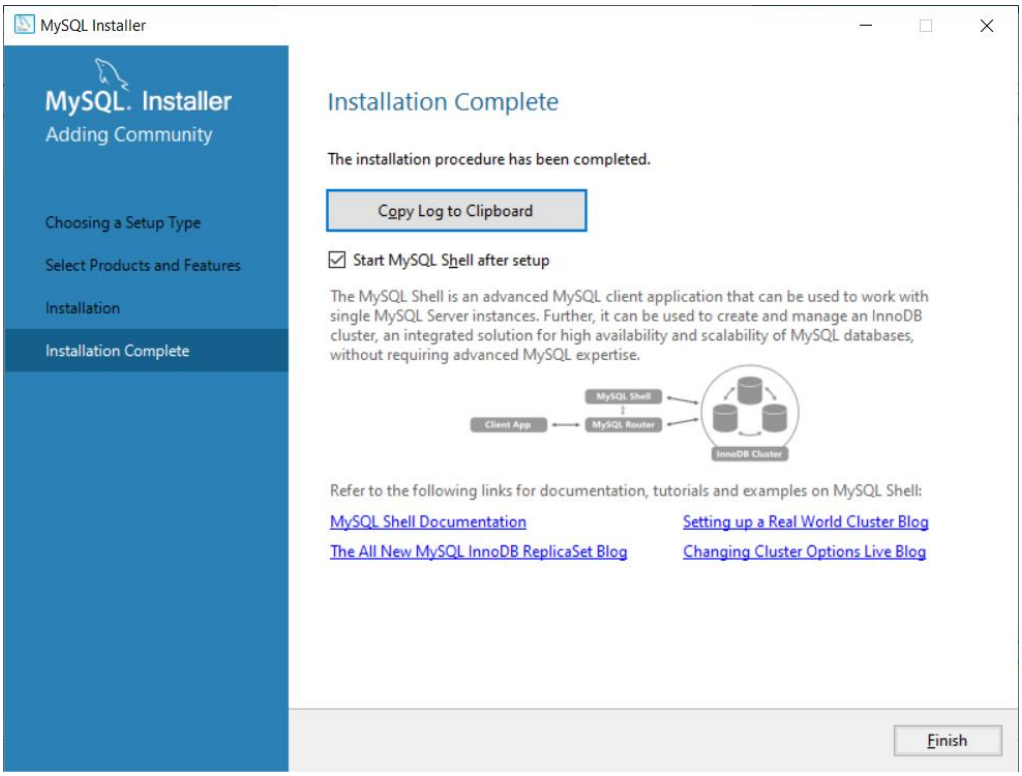


Expand the options under Applications -> MySQL Shell -> MySQL Shell 8.0 and then select MySQL Shell 8.0.21, selecting the top arrow to add that to the Products/Features To Be Installed list.

Select “Execute” to download the MySQL Shell


MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products and Features
Installation
Installation Complete
Installation
The following products will be installed.
Product
Status
Progress
Notes
MySQL Shell 8.0.21
Complete
Show Details >
< Back
Next >
Cancel

Select “Next”



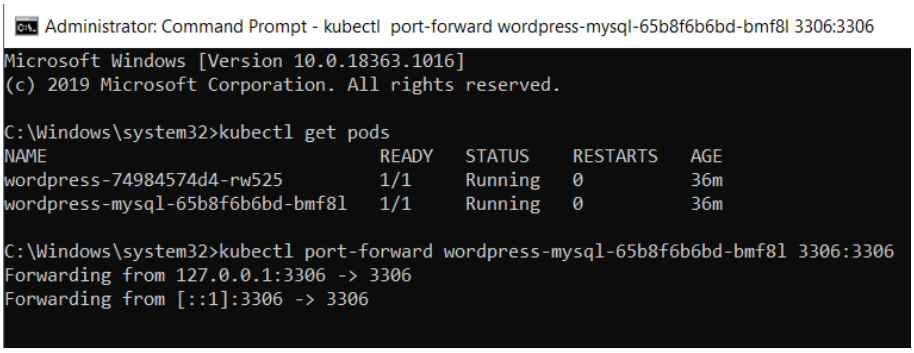
Select “Finish”

Connectivity into our containerized DB

In a command shell check your pods and create a port forward for MySQL to connect to port 3306

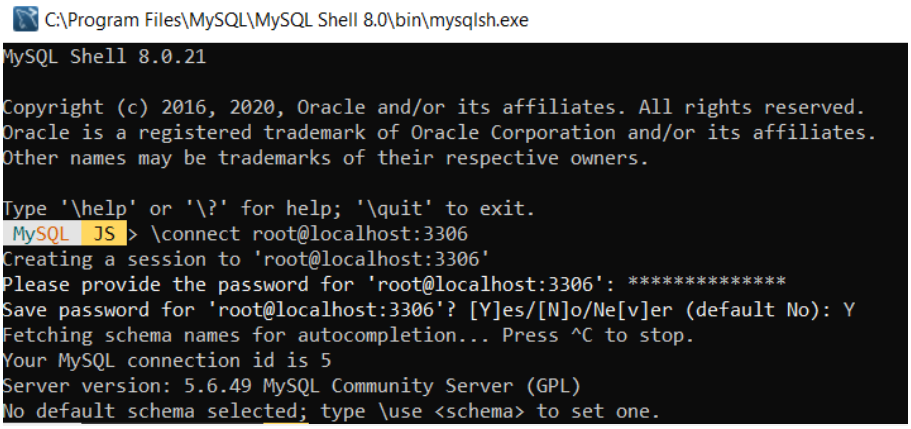
~/projects/kind/kind-wp $ **kubectl get pods**

~/projects/kind/kind-wp $ **kubectl port-forward wordpress-mysql-65b8f6b6bd-bmf81 3306:3306**

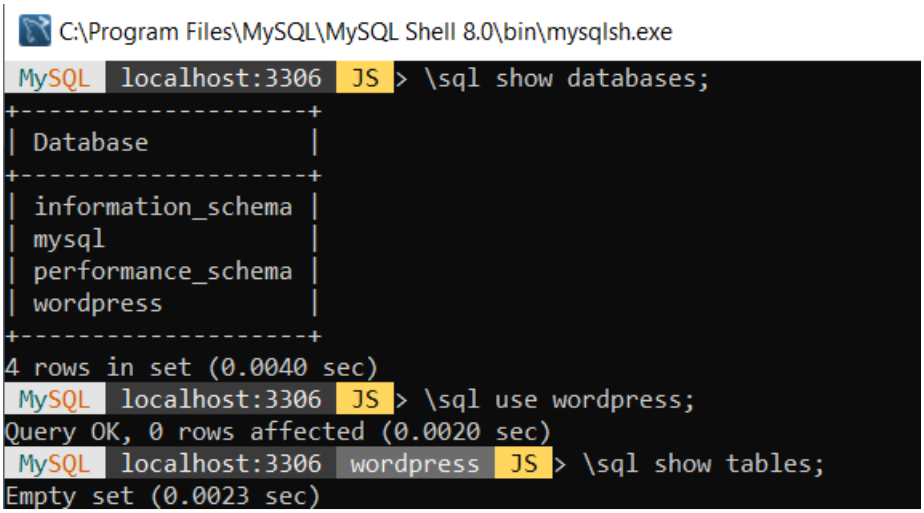


From your MySQL Shell you'll now be able to connect to the database running in the container for the WordPress application

Enter \connect root@localhost:3306



You can review your **kustomization.yaml** to find the MySQL password set or if you used the default noted in this lab, it will be **VerySecure2020**



Enter \sql show databases;

The list of databases in MySQL is listed, including our WP DB

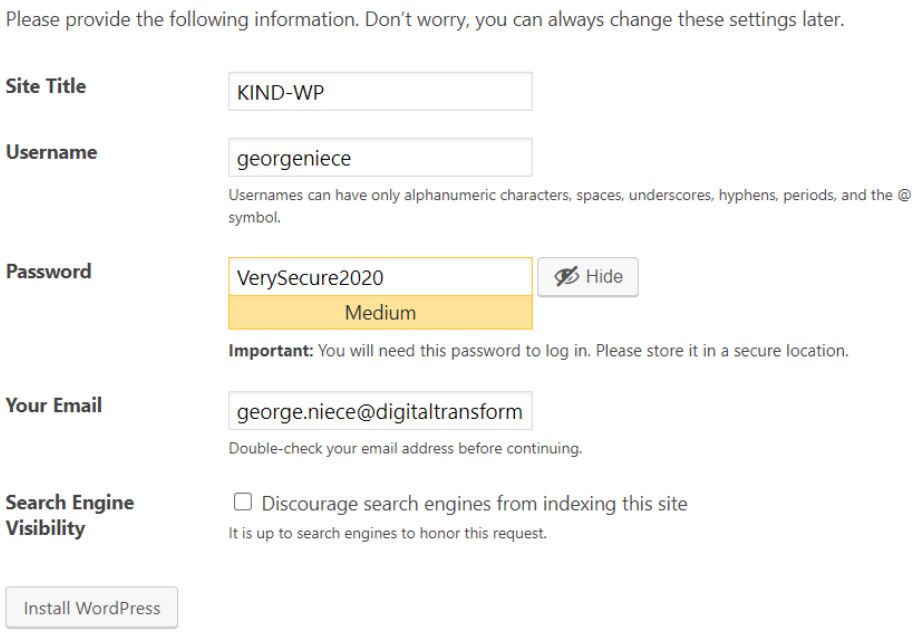
Enter \sql use wordpress;

This selects the WordPress DB that we've created in our deployment

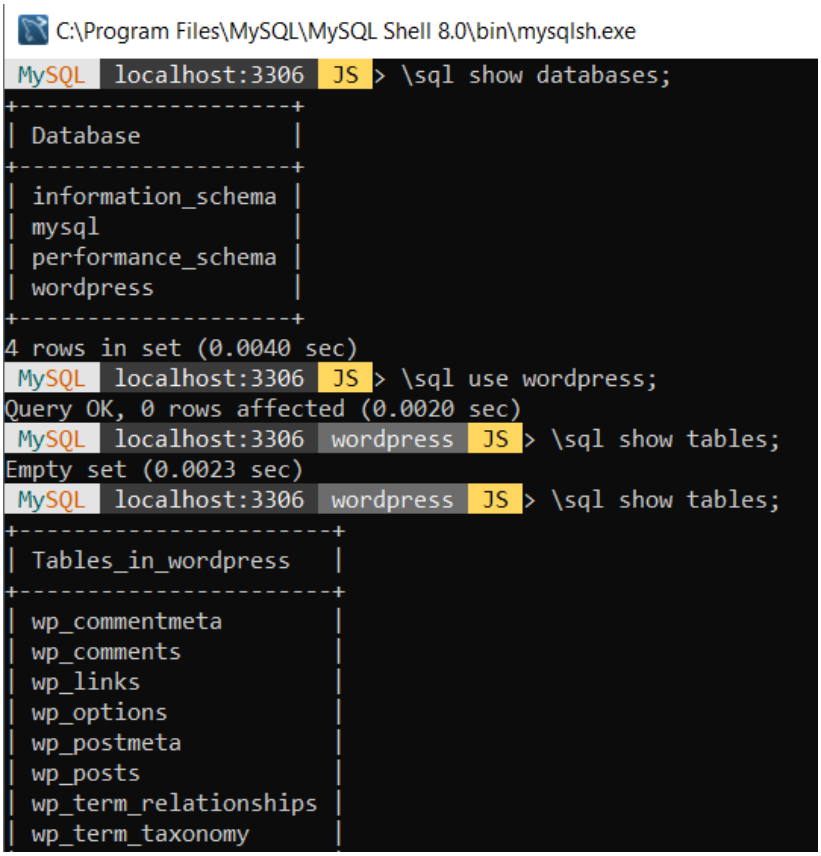
Enter \sql show tables;

You'll notice that no tables are created, since we've not created our WorkPress instance, as yet.

Go back to the WordPress admin screen, fill in the information and select “Install WordPress”



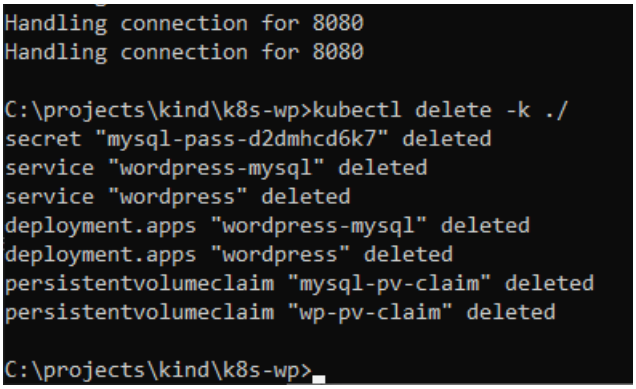
Return to your MySQL Shell and look at the tables again in our WordPress DB



Lab complete, kill the port-forwarding session by Ctrl-C

Run the following command to delete your Secret, Deployments, Services and PersistentVolumeClaims from our kind-wp folder in a command prompt (Windows) or terminal (MacOS)

~/projects/kind/kind-wp $ **kubectl delete -k ./**



~/projects/kind/kind-wp $ **kind help delete**

```
Deletes one of [cluster]

Usage:
  kind delete [command]

Available Commands:
  cluster     Deletes a cluster
  clusters    Deletes one or more clusters

Flags:
  -h, --help   help for delete

Global Flags:
      --loglevel string   DEPRECATED: see -v instead
  -q, --quiet             silence all stderr output
  -v, --verbosity int32   info log verbosity

Use "kind delete [command] --help" for more information about a command.
```

~/projects/kind/kind-wp $ **kind delete cluster --help**

```
Deletes a resource

Usage:
  kind delete cluster [flags]

Flags:
  -h, --help                help for cluster
      --kubeconfig string   sets kubeconfig path instead of $KUBECONFIG or
$HOME/.kube/config
      --name string         the cluster name (default "kind")

Global Flags:
      --loglevel string   DEPRECATED: see -v instead
  -q, --quiet             silence all stderr output
  -v, --verbosity int32   info log verbosity

C:\projects\kind\kind-wp> kind delete cluster --name Kind
Deleting cluster "Kind" ...
```

Now we can create a multi node cluster by inputting a config flag into our “kind create cluster” using the config flag and modifying the name with the name flag.

First we create a new configuration yaml file using your favorite text editor

$ vi grogu.yaml

Insert the following

six nodes (three workers & three leaders) cluster config

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: control-plane
- role: control-plane
- role: worker
- role: worker
- role: worker
#end of file
```

Save our file and now run that using the kind create with optional flags

$ kind create cluster --name grogu --config grogu.yaml

```
Creating cluster "grogu" ...
 • Ensuring node image (kindest/node:v1.21.1) 🖼  ...
 ✓ Ensuring node image (kindest/node:v1.21.1) 🖼
 • Preparing nodes 📦 📦 📦 📦 📦 📦  ...
 ✓ Preparing nodes 📦 📦 📦 📦 📦 📦
 • Configuring the external load balancer ⚖️  ...
 ✓ Configuring the external load balancer ⚖️
 • Writing configuration 📜  ...
 ✓ Writing configuration 📜
 • Starting control-plane 🕹️  ...
 ✓ Starting control-plane 🕹️
 • Installing CNI 🔌  ...
 ✓ Installing CNI 🔌
```

• Installing StorageClass 💾 ...

✓ Installing StorageClass 💾

• Joining more control-plane nodes 🎮 ...

✓ Joining more control-plane nodes 🎮

• Joining worker nodes 🚜 ...

✓ Joining worker nodes 🚜

Set kubectl context to "kind-grogu"

You can now use your cluster with:

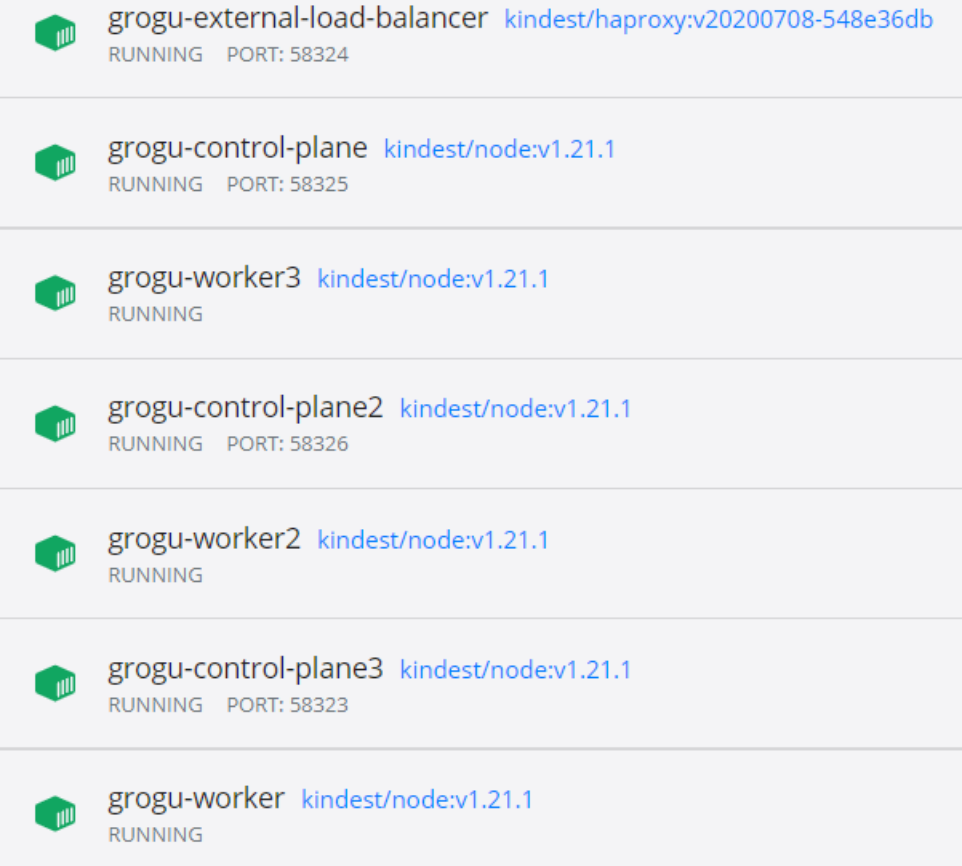
kubectl cluster-info --context kind-grogu

Have a nice day! 👋

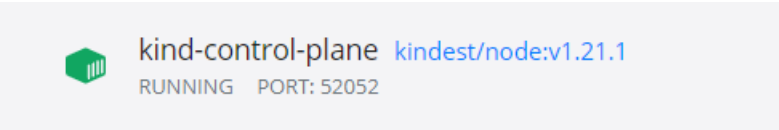
~/projects/kind $ **kubectl get nodes**

NAME	STATUS	ROLES	AGE	VERSION
grogu-control-plane	Ready	control-plane,master	3m20s	v1.21.1
grogu-control-plane2	Ready	control-plane,master	2m16s	v1.21.1
grogu-control-plane3	Ready	control-plane,master	77s	v1.21.1
grogu-worker	Ready	<none>	45s	v1.21.1
grogu-worker2	Ready	<none>	48s	v1.21.1
grogu-worker3	Ready	<none>	45s	v1.21.1

Looking at this environment in Docker Desktop we see



For the original single node cluster, created initially, we'd only see the single node

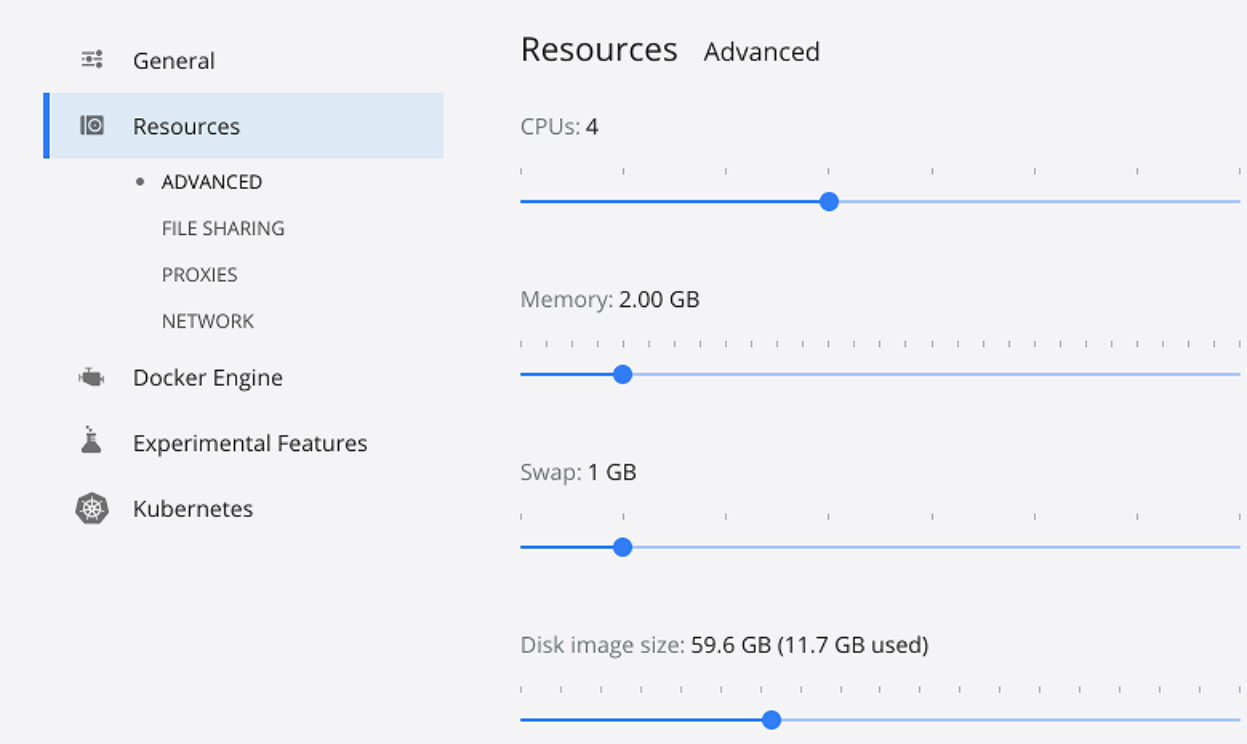


Now we'll delete the additional cluster with the following

~/projects/kind/ $ **kind delete cluster --name grogu**

Deleting cluster "grogu" ...

For updating our resource settings in Kind running in Docker Desktop for MacOS we'd go to the Settings -> Resources -> Advanced. These settings are not exposed in Docker Desktop for Windows and you to adjust those settings by setting priority on the Launch for Docker Desktop. For some task like building images the default settings won't work, i.e. building images typically requires 6GB RAM allocation.


General
Resources
ADVANCED
FILE SHARING
PROXIES
NETWORK
Docker Engine
Experimental Features
Kubernetes
Resources Advanced
CPUs: 4
Memory: 2.00 GB
Swap: 1 GB
Disk image size: 59.6 GB (11.7 GB used)