

# 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
```

Administrator: Command Prompt

```
to read about a specific subcommand or concept.  
See 'git help git' for an overview of the system.
```

```
C:\Windows\system32>cd \projects\kind
```

```
C:\projects\kind>kind create cluster
```

```
Creating cluster "kind" ...
```

```
• Ensuring node image (kindest/node:v1.18.2) ...
```

```
█ Ensuring node image (kindest/node:v1.18.2) █
```

```
• Preparing nodes █ █ ...
```

```
█ Preparing nodes █ █
```

```
• Writing configuration █ █ ...
```

```
█ Writing configuration █ █
```

```
• Starting control-plane █ █ ...
```

```
█ Starting control-plane █ █
```

```
• Installing CNI █ █ ...
```

```
█ Installing CNI █ █
```

```
• Installing StorageClass █ █ ...
```

```
█ Installing StorageClass █ █
```

```
Set kubectrl context to "kind-kind"
```

```
You can now use your cluster with:
```

```
kubectrl cluster-info --context kind-kind
```

```
Have a nice day! █ █
```

```
C:\projects\kind>
```

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 kubectrl client version

## \$ kubectrl 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

## \$ kubectrl cluster-info --context kind-kind

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

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

```
$ cd ~
$ mkdir .kube
$ cd .kube
$ nano config
```

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0I  
CQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcwRXSm  
wKY201bGRHVnpNQjRYFRJd01Ea3dPREF3TWpJME0xb1hEVE13TURrd05qQXdN  
akkwTTFvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdE  
UVlKS29aSWh2Y05BUUVCQlFBRGdnRVBbRENDQVFvQ2dnRUJBTBJJck9oYXpaZ  
FJNamx1WFh5aTQ0cFRBbkxPZEZ5QlJVb0FIRmZHUTlzMzUkZWJWZmZlU5DUDR  
CMINvUTZUb20zejMKK2pydUZsUWWhZK0grQ1A3UjByRGMMwUmozbxXHM0dPckJYcn  
RBNzB6N2MrTUdMMm5QaEt0OTZ4Q0INdVlvZ1NUbgpIMVFuY0pkM3I1T3RGd1I2czd  
GbWxyb1RRNkFoVE5GZktBMTV2UHZ4Mmw2bEV3a0JGRIMrNjZ3dERPT2dJZy9CCkl  
wQklzcWZ3bzYwbFRFUGorOEczbnBEYtTdnPNkVtNXNmZFIbZWJGTndIT21DL0cydnN

2Yks4MXV0anNNM1oyWmgKNUExNkxnUlpNOXk2Rzl2bluYjNlekM2UWNMVG9YVm9  
4YU95d0l0MFJlaXBIVkN3U2hKRG9qcjU0ZTBTWXIncQo5MIFpeE1GWEphUndVc1M0  
eVU4Q0F3RUFBYU1qTUNFd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd  
0VCCi93UUZNQU1CQWY4d0RRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFMVUxZMk  
NXTVINSIJUb0dQZGVWSmQ3MmZoL24KblFGWHRYT3dQblmMEISR0grYTVwOVcy  
ZytKRzQ4QjUwSUtZWFErVEZOTVZ0QUYrMDFWTFJRbzFQUEhudTM5NQppYlpHSn  
NybTBhWGVwOHFISWF2Lzl0MUxHcWk5dThkdFV6cjYvVEkwK2dUK2IEZIZ3bHRYSiZ  
iVXN4Q2U1VE5UCmdQTw9GWjBOdDVCTHUzRkdrV0x3OXRNSlkyU0JNckNZZ0ZhZ  
Hg5bkVSMjd4L0NsdjEvR3picEhWRmJmYVlubmQKUFNZbVRiWncxYlgyZzJBdC9pQ1  
BSbDJGbXYzbXBTdENjaEppVDNkK0U5aEtpV2p4T0kvVDFsN2E5UWxJTXdpdApldjZo  
MIJkUTQvZVdkeTFxY2hDYkZ6N0UyZEtvSXRONFFseFhQcVIMQ1VnYzB4TEFGK2kx  
dkV1bDM3Zz0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=

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:

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUM4akNDQWRxZ0F3SUJBZ0lJ  
U3VyaXQxdGJKNnN3RFFZSkvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQ  
XhNS2EzVmlaWEp1WlhSbGN6QWVGdZB5TURBNU1EZ3dNREI5TkROYUZ3MHINVE  
E1TURnd01ESXIORFJhTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4  
wWlhKek1Sa3dGd1IEVIFRREV4QnJkV0psY201bGRHVnpMV0ZrCmJXbHVNSUICSW  
pBTkNa3Foa2lHOXcwQkFRRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXo1K1d0L242  
RGtEbm5QblcKMmFmb3kxM1ArSldjQi9BMXdBTUU3L2RSREplZ1g4N1lqOFRibkRDdU  
NhcElxVFY0ZjBqdW9JdDRDZmNKQUxMVgpXTWxhYVhncmljSXgxexjdBdzIEUDVNRH  
M2VkdJcWRKaGc1Z0ROQWVYdTIzZUNuaUgxVDBnUFJxNjdKWHY4YW9Vci9YbIY3  
UFdXNDVGbDlvd0NwSEc5Vlp4MUNlVVOd1RVb09pd29oYzhpWmh6QzNHdS9CZ3F

RbDR5K3hMVU0wTDkKWXZEY0NBU2dYdEhCQkU3TWN2SWxIQWFFdmdKVVk0c1  
MvMUhZMytZQmZVcCtzUzZjcHBKWFpveXhaTGdaMm51YgpTYmVhQ0J0Wk9wbGR3  
YTgvV29SbGtrRkFpK0RURkt3cEZmTC9DYkxqOHd3QTVXdnNtWVdyWkFRZGE2aTB  
KQ2k5CmdaMzhCUUIEQVFBQm95Y3dKVEFPQmdOVkhROEJBZjhFQkFNQ0JhQXdF  
d1IEVIlwbEJBd3dDZ1IJS3dZQkJRvUgKQXkJd0RRWUpLb1pJaHZjTkFRRUxCUUFEZ  
2dFQkFCOS9FL3o1cnNBeDNoaDVxVlo3ZUswTy95bUtES05SOFVDTApwc21kaVI2dm  
1qZEVkc0I2WFhWM2dMeWQ1TmF0SSt2WU5rbS9YODIqNTIURXNveGhtS2V2SEorQ  
mRZWXBPCeQ5CmY3Y3dBNnNIS2IEcGJUMVUajRoeHJoZyt6VEFXN3hleHZEam8v  
dG5xazg1dDR1bzVLdFVrRjRFNEN1QXQ3TFUKUjJaTIBta2UwdTRsUWgzaFU5Slg1bz  
Y5N3FHb21EMkZTeDdvb212eXdDMS8rbFBIVkNURk95U2pPbXEya001VApiZCtRbmh  
oeVVTRXJheC9sbnV5SDhkUIJYZkZWelk1WEILNS81RnBBMFFUUKltRXJUVEExSem5  
mcktNQXVaS3VTCjVFL0d4Z083MU1vZGN3ajM2NWZKWE9JT1c0UIZ4TXJabEpnQVZ  
la0IGUitQSUV0bIERRT0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=

client-key-data:

LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUIFcEFJQkFBS0NBUUVB  
ejUrV3QvbjZEa0RublBuVzBVZm95MTNQK0pXY0lvQTF3QU01Ny9kUkRKSGdYODdZ  
Cmo4VGJuREN1Q2FwSXFUVjRmMGp1b0I0NENmY0pBTExWV01sYWZY3JpY0I4M  
Xo3QXc5RFA1TURzNIZHSXFKSmgKZzVnRE5BZVh1MjNIQ25pSDFUMGdQUeE2N0p  
Ydjhhb1UvWG5WN1BXVzQ1Rmw5b3dDcEhHOVZaeDFDSGIVTndUVQpvT2I3b2hjOG  
laaHpDM0d1L0JncVFfSnhkreExVTTBMOVI2RGNDQVNnWHRIQkJFN01jdklsSEFhRXZ  
nSIVZNHNTCi8xSFkzK1ICZIVwK3NTNmNwcEpYWm95eFpMZ1oybnViU2JIYUNCdFp  
PcGxkd2E4L1dvUmxra0ZBaStEVEZLd3AKRmZML0NiTGo4d3dBNVd2c21ZV3JaQVFk  
YTZpMEpDaTlnWjM4QIFJREFRQUJBb0lCQUVzYmxFNWhvOC9jTXUxYQpoQmVaUit  
HcHdqNVBBTzd1T3NPSFowSWoyYkIPWTNqRIB4cGpRSDYwTFIGWmxJZUJ6R0Zm  
Wk5PM0IHbWfjQ3RNCmhsbGtlY3pocC81aHZkMzcyWWY4MWZnT3dxVjAxVmQ1djh  
UM0ROR1puWTQwSklydEoxWkFrcFVJUW02cm51MXgKZGI1c2dMTUQ5TjNHRDNpd  
EZaZWZmYnFtcXlreEYvN2tTQ0ZKSEpyR0JoMk9GT3JSTXdxUXREaFBJWS9DTzN6  
VQoybVBOTXN2Rk5JQjBoK2IKVXEvWEpwOHAvd3hmYkYva3FuRXBEYTZhaTFXT21  
NdzhUjFtRFZqSjR3TWtZdEkwCjZrTWFTdHYvVVBBeVlvZzFvV0ZTeGFGcGs4OEh6  
UUp1ZnpXamZ6c2Z4bDN4ZnNwY1FKNzNZUIZzb29kRFdWTEUKRkwxVFRBRUNnW  
UVBODVWwklMcHp3M0ISa21rQkJOT0VLYk5wWWNOZEYyOUNDOGNuYWJRMFdm  
NVIqUWEzdzBYMgo5YXdiVlIQVXBCck1wZUY0VINPalhSTWplazkwZDlvM1dwRlkxdjJV  
UjBSTXdrYy9MVXNIVjJ6S2xUczZmanZFCi8vU0VTRm9FS2ZmWmkzbjAzUm81R3gvd  
WZxK1YwUnl6L0pNMzIKedI5N0o5NHVHdE1FSEVCY1VDZ1IFQTJqVXoKK1VzSnRB  
Q2NwNkKJ3TDNOV1h3U29aMk9nbzZnYjBOclpTUW9YbnNwdGt0UmF0R3NER3JTWG  
1CeGJNRUloVklERQpKbDBNTGc3QnFEaDcwVklBbmNJWmQzQkhNd0NYMmgzNDN  
6NmJZSTZ1RFBSU1k0amd4WHB0djBaVDdIV1ZDVFGwCk9TWnArYkpuMFpJb3gwSz  
BHc0FIcDk2WThlZnRReUoyU2FOendVRUNnWUVBbytzWfVONEluMk01RGdVWnlXe  
XEKQ1FJU2pkYII0NzVjZk42VjJGMkx5Smkzc0pmdnVZbFV5emo1NEE5cVh0RW1IUTI  
oVWpJOGNwczVTK0YrYUEzeQpNSUdWZm9DQmM0QTBNTI4bHpkaGhtVFE0Nkp  
MRjc0VE1ZZ1VLVGhpaXZIZTcyeXY2c2NGM1FvZERpWU5OUDhTCjVJc1I0Y3dWaDF  
VWHdFSE1zYWZnU1YwQ2dZRUFpUmY2RW5zeG1uVHo5MkVXZXNsMUQzZW1zbV  
ptcTh2WHhnMXAKakxrWmcvdGNnbTZhBw1uTlpuN2w2M3IOVWpHS0xVUKZISERuV

VJ5V1VURkRvWXhxdExNWk92QkEyMnRZL0lldQpOWnhwRkc0d0xoVm1tZ0NLyJZmd  
Xdjald1MjVZZDVQOVg5YWhxRzZOU1o4Um5iZHlzbzZkZ0x1YXpTSWI0QjBMCndsSSSt  
UTUVDZ1ICSTFrek5EeHhWUFVucERleXNnWmxIS3QwcDltTzhKWjIGcU5UM044eXY1  
OEZnbitkSFU2WmYKWXZUNDdkK3FxdHRYREJEa1ZIWkJ4KzVSdW5kWlgxM3NKUk  
g2NUhVeE5QczBkdGVLa3RObnh5Zm9ubEFZTU13QgorWXBIMFVUT2xaK3FOT3d1U  
zFkZHM2MU14SVZWRIJxTDVzVGIVZGdkU3VLenU3ZW41R2hiY3c9PQotLS0tLUVOR  
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

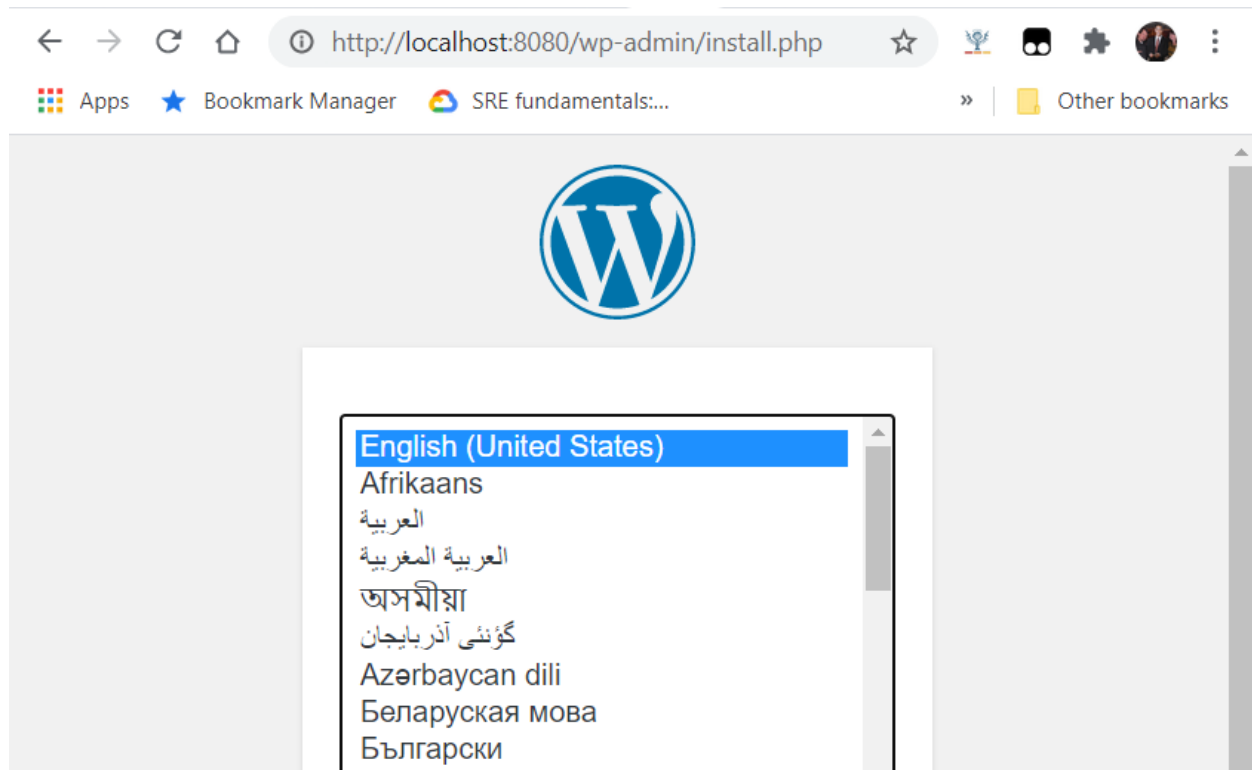
Create a port forward to allow us to access our Wordpress environment

```
~/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/>

### MySQL Shell 8.0.21

Select Operating System:

Microsoft Windows

Recommended Download:

**MySQL Installer**  
for Windows

**All MySQL Products. For All Windows Platforms.  
In One Package.**

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

**Windows (x86, 32 & 64-bit), MySQL Installer MSI**

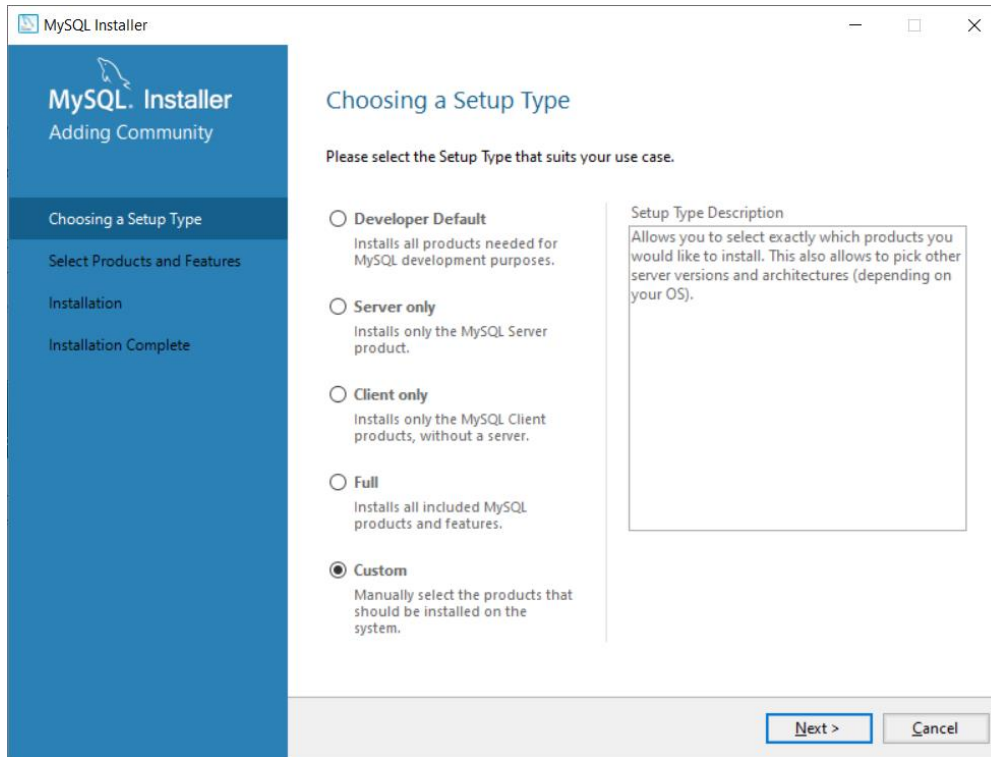


[Go to Download Page >](#)

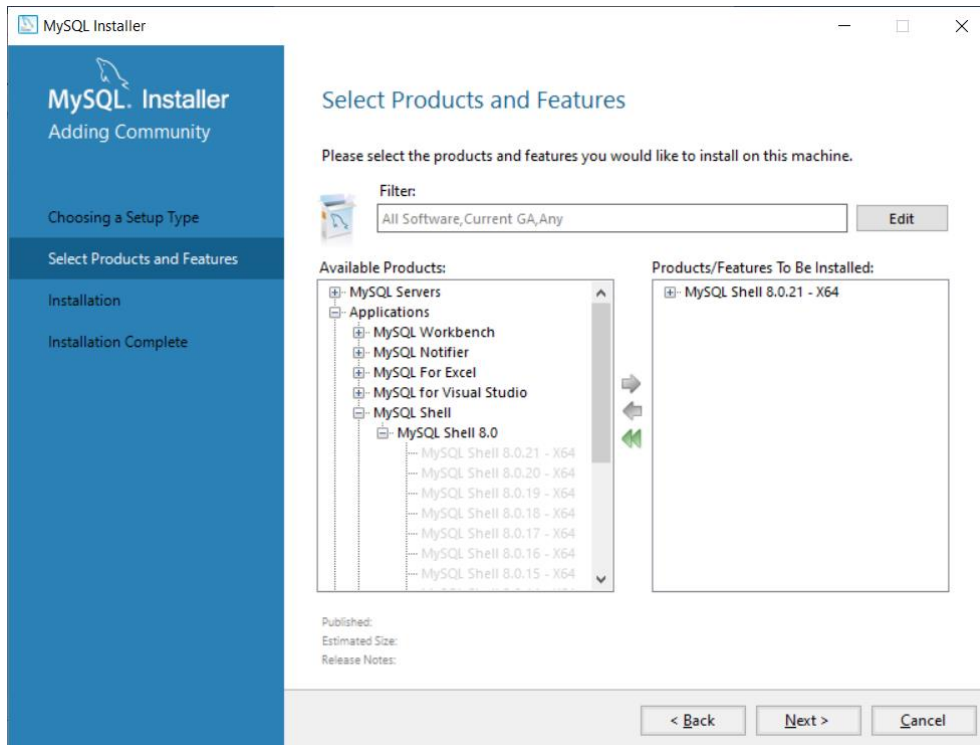
If you have an Oracle account, you can login or you can select “No thanks, just download”

Install the MySQL Shell, by running the downloaded installer

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

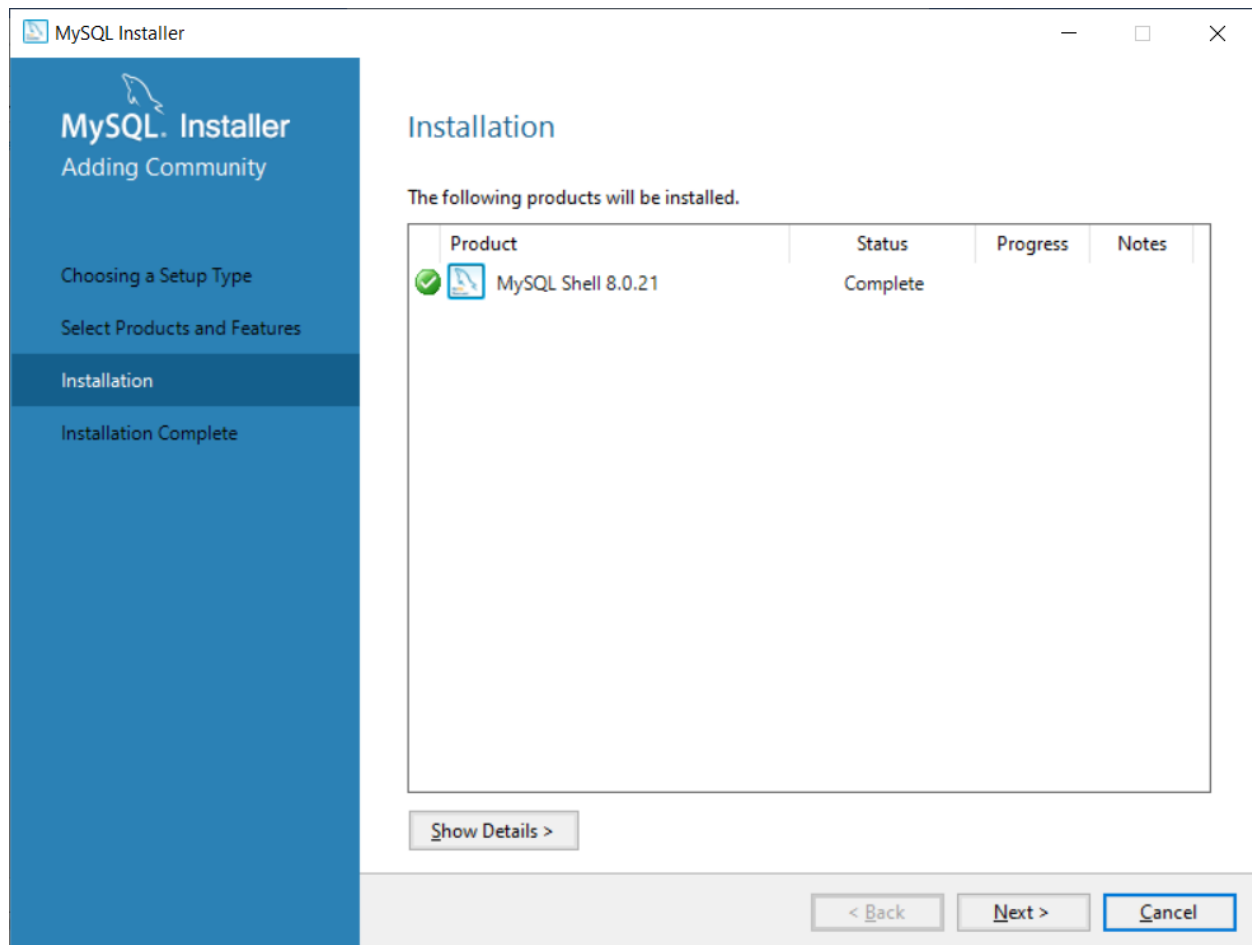


Select “Next”

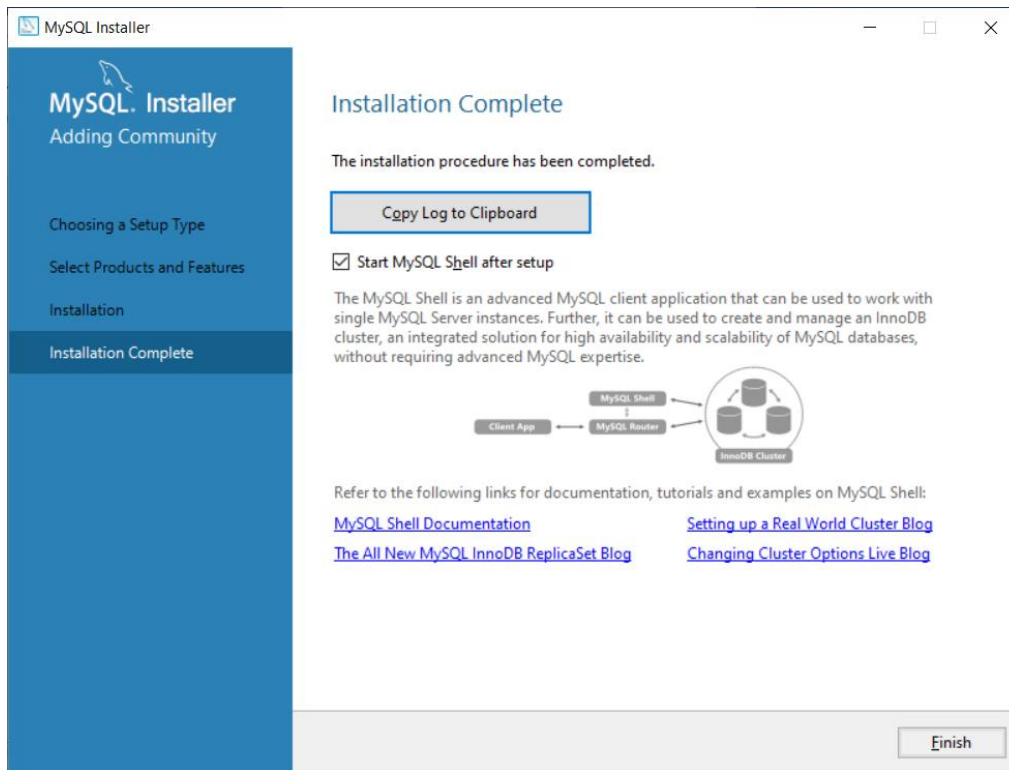


Expand the options under Applications -> MySQL Shell -> MySQL Shell 8.0 and then select MySQL Shell 8.0.21, selecting the top arrow to add that to the Products/Features To Be Installed list.

Select "Execute" to download the MySQL Shell



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
```

```
Administrator: Command Prompt - kubectl port-forward wordpress-mysql-65b8f6b6bd-bmf81 3306:3306
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
wordpress-74984574d4-rw525          1/1     Running   0           36m
wordpress-mysql-65b8f6b6bd-bmf81    1/1     Running   0           36m

C:\Windows\system32>kubectl port-forward wordpress-mysql-65b8f6b6bd-bmf81 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]: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`

```
C:\Program Files\MySQL\MySQL Shell 8.0\bin\mysqlsh.exe
MySQL Shell 8.0.21
Copyright (c) 2016, 2020, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '? ' for help; '\quit' to exit.
MySQL JS > \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Please provide the password for 'root@localhost:3306': *****
Save password for 'root@localhost:3306'? [Y]es/[N]o/[e]ver (default No): Y
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 5
Server version: 5.6.49 MySQL Community Server (GPL)
No default schema selected; type \use <schema> to set one.
```

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**

```
C:\Program Files\MySQL\MySQL Shell 8.0\bin\mysqlsh.exe
MySQL localhost:3306 JS > \sql show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wordpress |
+-----+
4 rows in set (0.0040 sec)
MySQL localhost:3306 JS > \sql use wordpress;
Query OK, 0 rows affected (0.0020 sec)
MySQL localhost:3306 wordpress JS > \sql show tables;
Empty set (0.0023 sec)
```

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 WordPress instance, as yet.

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

Please provide the following information. Don't worry, you can always change these settings later.

Site Title	<input type="text" value="KIND-WP"/>
Username	<input type="text" value="georgeniece"/> <small>Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.</small>
Password	<div><input type="password" value="VerySecure2020"/><div>Hide</div></div> <div>Medium</div> <p><b>Important:</b> You will need this password to log in. Please store it in a secure location.</p>
Your Email	<input type="text" value="george.niece@digitaltransform"/> <small>Double-check your email address before continuing.</small>
Search Engine Visibility	<input type="checkbox"/> Discourage search engines from indexing this site <small>It is up to search engines to honor this request.</small>
<input type="button" value="Install WordPress"/>	

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

```
C:\Program Files\MySQL\MySQL Shell 8.0\bin\mysqlsh.exe
MySQL localhost:3306 JS > \sql show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wordpress |
+-----+
4 rows in set (0.0040 sec)
MySQL localhost:3306 JS > \sql use wordpress;
Query OK, 0 rows affected (0.0020 sec)
MySQL localhost:3306 wordpress JS > \sql show tables;
Empty set (0.0023 sec)
MySQL localhost:3306 wordpress JS > \sql show tables;
+-----+
| Tables_in_wordpress |
+-----+
| wp_commentmeta |
| wp_comments |
| wp_links |
| wp_options |
| wp_postmeta |
| wp_posts |
| wp_term_relationships |
| wp_term_taxonomy |
+-----+
```

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 ./
```

```
Handling connection for 8080
Handling connection for 8080

C:\projects\kind\k8s-wp>kubectl delete -k ./
secret "mysql-pass-d2dmhcd6k7" deleted
service "wordpress-mysql" deleted
service "wordpress" deleted
deployment.apps "wordpress-mysql" deleted
deployment.apps "wordpress" deleted
persistentvolumeclaim "mysql-pv-claim" deleted
persistentvolumeclaim "wp-pv-claim" deleted

C:\projects\kind\k8s-wp>
```



```
~/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

	grogu-external-load-balancer	kindest/haproxy:v20200708-548e36db
	RUNNING	PORT: 58324
	grogu-control-plane	kindest/node:v1.21.1
	RUNNING	PORT: 58325
	grogu-worker3	kindest/node:v1.21.1
	RUNNING	
	grogu-control-plane2	kindest/node:v1.21.1
	RUNNING	PORT: 58326
	grogu-worker2	kindest/node:v1.21.1
	RUNNING	
	grogu-control-plane3	kindest/node:v1.21.1
	RUNNING	PORT: 58323
	grogu-worker	kindest/node:v1.21.1
	RUNNING	

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

	kind-control-plane	kindest/node:v1.21.1
	RUNNING	PORT: 52052

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

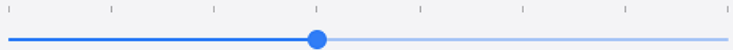
Docker Engine

Experimental Features

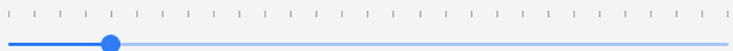
Kubernetes

## Resources Advanced

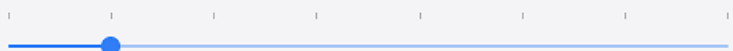
CPUs: 4



Memory: 2.00 GB



Swap: 1 GB



Disk image size: 59.6 GB (11.7 GB used)

