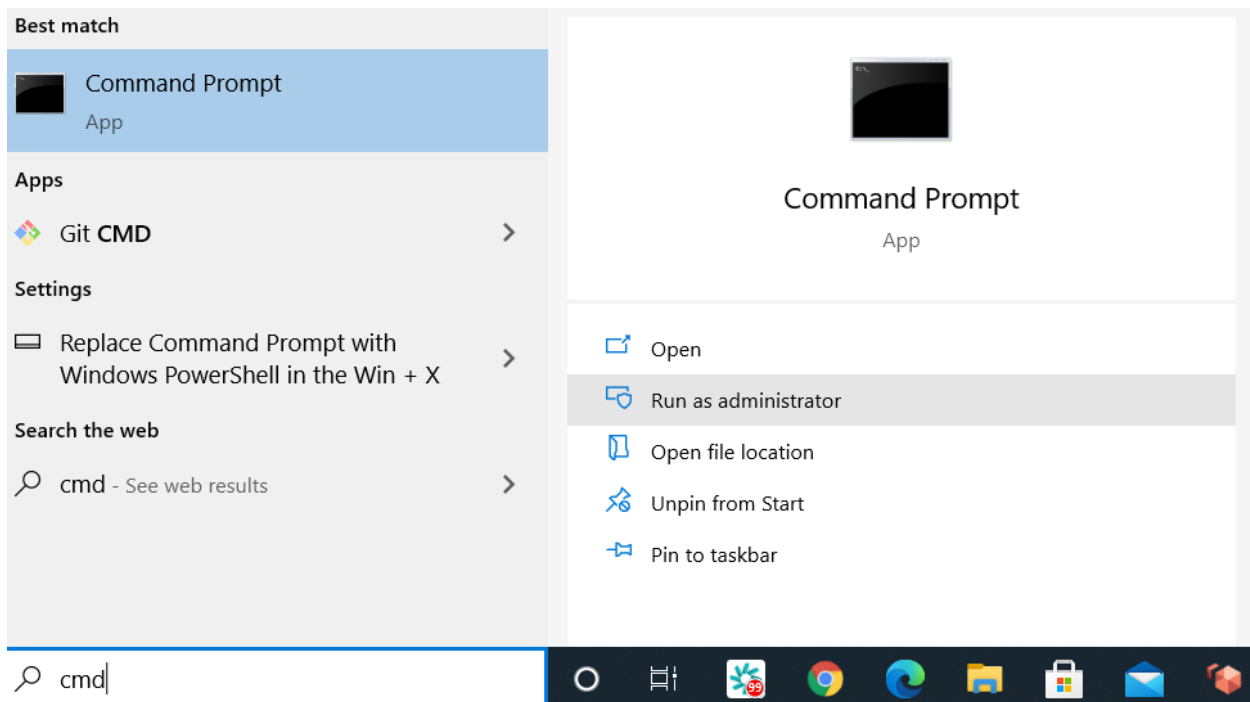


Experiment 02: Working With Kind

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.

In Windows:

Open a Windows Command Prompt with "run as administrator"



```
C:\> mkdir c:\projects
```

```
C:\> mkdir c:\projects\kind
```

```
C:\> cd c:\projects\kind
```

```
C:\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.

C:\projects\kind> **kind get clusters**

Kind

Check the kubectl client version

C:\>**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

C:\> **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 automatically 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.

```
C:\> cd %USERPROFILE%
C:\Users\wcsadmin> mkdir .kube
C:\Users\wcsadmin> cd .kube
C:\Users\wcsadmin\.kube> notepad 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:

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURStLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURB
TkJna3Foa2lHOXcwQkFRc0ZBREFTWTVJN0dVRWURWUVFERXdwcmRXSmwKY201bGRHV
npNQjRYRFRJd01Ea3dPREF3TWpJME0xb1hEVE13TURrd05qQXdNakkwTTFvd0ZURVRNQk
VHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFB
RGdnRVBIBRENDQVFvQ2dnRUJBTBJJck9oYXpaZjJNamx1WFh5aTQ0cFRBbkxPZEZ5QlJV
b0FIRmZHUlRzd3UkZWZlZ0U5DUdRCmINvUTZUb20zejMkK2pydUZsUWhZK0grQ1A3U
jByRGMwUm0zbXdm0dPckJYcnRBNzB6N2MrTUdMMm5QaEt0OTZ4Q0lNdVlvZ1NUbGpIMV
FuY0pkM3I1T3RGd1I2czdGbWxyb1RRNkFoVE5GZktBMTV2UHZ4Mmw2bEV3a0JGRIMrNjZ3
dERPT2dJZy9CCklwQklzcWZ3bzYwbFRFUGorOEczbnBEYtPNkVTNlXNmZFIWJGJGtndIT2
1DL0cvdnN2Yks4MxV0anNNM1oyWmgKNUExNkxnlpNOXk2Rzl2bluYjNlekM2UWNMVG9Y
Vm94YU95d0l0MFJlaXBIVkN3U2hKRg9qcjU0ZTBWlncQo5MIFpeE1GWephUndVc1M0eV
U4Q0F3RUFBU1qTUNFd0RnWURWUjBQVFI0BUURBZ0trTUE4R0ExVWRFd0VCCi93U
UZnQU1CQWY4d0RRWUplb1pJaHZjTkFRRUxCUUFEZ2dFQkFMVUxZMkNXTVINSiUub0dQ
ZGVWSmQ3MmZoL24KbIFGWHRYT3dQbldmMEISR0grYTVwOVcyZytKRzQ4QjUwSUtZWFEr
VEZOTVZ0QUYrMDFWTFJRbzFQUEhudTM5NQppYlphSnNybTBhWGvVwOHFISWF2LzI0MUx
HcWk5dThkdFV6cjYvVEkwK2dUK2IEZIZ3bHRYSlZiVXN4Q2U1VE5UCmdQW9GWjBOdDVC
THUzRkdrV0x0OXRNslkyU0JNckNZZ0ZhZHg5bkVSMjd4L0NsdjEvR3picEhWRmJmYVlubmQ
KUFNZbVRIWncxYlgyZzJBdC9pQ1BSbDJGbgXYzbXBtdENjaEppVDNkK0U5aEtpV2p4T0kvVD
FsN2E5UWxJTXdpdApldjZ0MIJkUTQvZVdkeTFxY2hDYkZ6N0UyZEtvSXRONFFseFhQcVIMQ
1VnYzB4TEFGK2kxdkV1bDM3Zz0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=
```

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:

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURStLS0tCk1JSUM4akNDQWRxZ0F3SUJBZ0lJU3VyaX
```

QxdGJKNN3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmla
WEp1WlhSbGN6QWVGdzB5TURBNU1EZ3dNREI5TkROYUZ3MHINVEE1TURnd01ESXIORFJ
hTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4wWlhKek1Sa3dGd1IEVIFRREV
4QnJKV0psY201bGRHVnpMV0ZrCmJXbHVNSUICSWpBTkJna3Foa2IHOXcwQkFRRUZBQU9
DQVE4QU1JSUJDZ0tDQVFFQXo1K1d0L242RGtEbm5QblcKMFVmb3kxM1ArSldjQi9BMXdBT
TU3L2RSREplZ1g4N1lqOFRibkRDdUNhcElxVFY0ZjBqdW9JdDRDZmNKQUxMVgpXTWxhYV
hncmljSXgxejdBdzIEUDVNRHM2VkdJcWRKaGc1Z0ROQWVYdTlZUNuaUgxVDBnUFJxNjdK
WHY4YW9VCi9YbIY3UFdXNDVGbDlvd0NwSEc5Vlp4MUNlaVVOd1RVb09pd29oYzhpWmh6Q
zNHdS9CZ3FRbDR5K3hMVU0wTDkKWZXEY0NBU2dYdEhCQkU3TWN2SWxlQWFFdmdKVV
k0c1MvMUhZMytZQmZVcCtzUzZjcHBKWFpveXhaTGdaMm51YgpTYmVhQ0J0Wk9wbGR3YT
gvV29SbGtrRkFpK0RURkt3cEZmTC9DYkxqOHd3QTVXdnNtWVdyWkFRZGE2aTBKQ2k5Cmd
aMzhCUUIEQVFBQm95Y3dKVEFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1IEVllwbEJBd3dD
Z1JS3dZQkJRUVUgKQXdJd0RRWUpLb1pJaHJzTkFRRUxCUUFEZ2dFQkFCOS9FL3o1cnNBe
DNoaDVxVlo3ZUswTy95bUtES05SOFVDTApwc21kaVI2dm1qZEVCk0I2WFhWM2dMeWQ1Tm
F0SSt2WU5rbS9YODlqNTIURXNVEGhtS2V2SEorQmRZWXBpCEQ5CmY3Y3dBNNNIS2IEcGJ
UMVVuajRoeHJoZyt6VEFXN3hleHZEam8vdG5xazg1dDR1bzVLdFVrRjRFNEN1QXQ3TFUKUj
JaTiBta2UwdTRsUWgzaFU5Slg1bzY5N3FHb21EMkZTeDdvb212eXdDMS8rbFBIVkNURk95U2
pPbXEya001VApiZCtRbmhoeVVTRXJheC9sbnV5SDhkUIJYkZWelk1WEILNS81RnBBMFFUU
klRXJUVExSem5mcktnQXVaS3VTCjVFL0d4Z083MU1vZGN3ajM2NWZKWE9JT1c0UIZ4TXJa
bEpnQVZla0IGUiQSUUV0bIERRT0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=

client-key-data:

LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUJFfEFJQkFBS0NBUEVBejUrV3Q
vbJZEa0RublBuVzBVZm95MTNQK0pXY0lvQTF3QU01Ny9kUkRKSGdYODdZCmo4VGJuREN
1Q2FwSXFUVjRmMGp1b0l0NENmY0pBTEwVW01sYWYFYZ3JpY0I4MXo3QXc5RFA1TURzNIZ
HSXFKsmgKZzVnRE5BZVh1MjNIQ25pSDFUMGdQUeE2N0pYdjhhb1UvWG5WN1BXVzQ1Rm
w5b3dDcEhHOVZaeDFDSGIVTndUVQpvT2l3b2hjOGlaaHpDM0d1L0JncVFsnHkreExVTTBMO
VI2RGNDQVNnWHRIQkJFN01jdklsSEFhRXZnSIVZNHNTCi8xSFkzK1ICZlVwK3NTNmNwcEp
YWm95eFpMZ1oybnViU2JiYUNCdFpPcGxkd2E4L1dvUmxra0ZBaStEVEZLd3AKRmZML0NiT
Go4d3dBNVd2c21ZV3JaQVFkYTZpMEpDaTlnWjM4QlFJREFRQUJBb0lCQUVzYmxFNWWhvO
C9jTXUxYQpoQmVaUitHcHdqNVBBTzd1T3NPSFowSWoyYkIPWTNqRIB4cGpRSDYwTFIGW
mxJZUJ6R0ZmWk5PM0IHbWfjQ3RNCmhsbGtIY3pocC81aHZkMzcyWWY4MWZnT3dxVjAxV
mQ1djhUM0ROR1puWTQwSklydEoxWkFrcFVJUW02cm51MXgKZGI1c2dMTUQ5TjNHRDNpd
EZAZWZmYnFtcXIreEYvN2tTQ0ZKSEpyR0JoMk9GT3JSTXdxUXREaFBjWS9DTzN6VQoybV
BOTXN2Rk5JQjBoK2IKVXEWEpwoHAvd3hmYkYva3FuRXBEYTZhaTFXT21NdzhUqJfFRFZq
SjR3TWtZdEkWcjZrTWFTdHYvVVBBeVlvZzFvV0ZTeGFGcGs4OEh6UUp1ZnpXamZ6c2Z4bD
N4ZnNwY1FKNzNZUIZzb29kRFdWTEUKRkwXVFRBRUNnWUVBODVWwKIMcHp3M0ISa21rQ
kJOT0VLyK5wWWNOZEYyOUNDOGNuYWJRMFdmNVlqUWEzdzBYMgo5YXdiVIIQVXBCck1w
ZUY0VINPalhSTWplazkwZDlvM1dwRlkxdjJVUjBSTXdrYy9MVXNIVjJ6S2xUczZmanZFCi8vU0
VTRm9FS2ZmWmkzbjAzUm81R3gvdWZxK1YwUnl6L0pNMzIKedI5N0o5NHVHdE1FSEVCY1
VDZ1IFQTJqVXoKK1VzSnRBQ2NwNkJ3TDNOV1h3U29aMk9nbzZnYjBOclpTUW9YbnNwdGt0
UmF0R3NER3JTWG1CeGJNRUIoVklERQpKbDBNTGc3QnFEaDcwVklBbmNJWmQzQkhNd0
NYMmgzNDN6NmJZSTZ1RFBSU1k0amd4WHB0djBaVDdIV1ZDVFGwCk9TWnArYkpuMFpJb3
gwSzBHc0FicDk2WThlZnRReUoyU2FOendVRUNnWUVBbytzWfVONEluMk01RGdVWnlXeX
EKQ1FJU2pkYII0NzVjZk42VjJGMkx5Smkzc0pmdnVZbFV5emo1NEE5cVh0RW1IUTloVWpJO
GNwczVTK0YrYUEzeQpNSUdWZm9DQmM0QTBNTI4bHpkaghtVFE0NkpMRjc0VE1ZZ1VL
VGhpaXZlZTcyYXY2c2NGM1FvZERpWU5OUDhTCjVJc1I0Y3dWADFVWHdFSE1zYWZnU1Yw
Q2dZRUFPuM2RW5zeG1uVHo5MkVXZXNsMUQzZW1zbVptcTh2WHhnMXAKakxrWmcvdG
NnbTZHbW1uTlpuN2w2M3IOVWpHS0xVUkZISERuVVJ5V1VURkRvWXhxdExNWk92QkEyMn
RZL0lIdQpOWnhwRkc0d0xoVm1tZ0NLyJZmdXdjald1MjVZZDVQOVg5YWhxRzZOU1o4Um5iZ
HlzbZkZ0x1YXpTSWI0QjBMCndsSSStUTUVDZ1ICSTFfek5EeHhWUfVucERleXNnWmxIS3Qw
cDltTzhKWjIGcU5UM044eXY1OEZnbitkSFU2WmYKWXZUNDdkK3FxdHRYREJEa1ZIWKJ4Kz

```
VSdW5kWlgxM3NKUkg2NUhVeE5QczBkdGVLa3RObnh5Zm9ubEFZTU13QgorWXBIMFVUT2
xaK3FOT3d1UzFkZHM2MU14SVZWRIJxTDVzVGIVZGdkU3VLenU3ZW41R2hY3c9PQotLS0t
LUVORCBSU0EgUFJJVkJFURSBLRVktLS0tLQo=
```

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

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

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

Create kustomization.yaml with the following content

```
C:\projects\kind\kind-wp> notepad kustomization.yaml
```

In MacOS:

```
~: 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
```

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.

Download the following two resource files to our (Windows) **c:\projects\kind\kind-wp** or (MacOS) **~/projects/kind/kind-wp** folders

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.

```
$ 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

C:\projects\kind\kind-wp> **kubectl apply -k ./**

Run the following commands to inspect the deployment

C:\projects\kind\kind-wp> **kubectl get secrets**

C:\projects\kind\kind-wp> **kubectl get pvc**

C:\projects\kind\kind-wp> **kubectl get pods**

Once the "get pods" shows status as Running, rather than ContainerCreating we'll check the services for wordpress

C:\projects\kind\kind-wp> **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

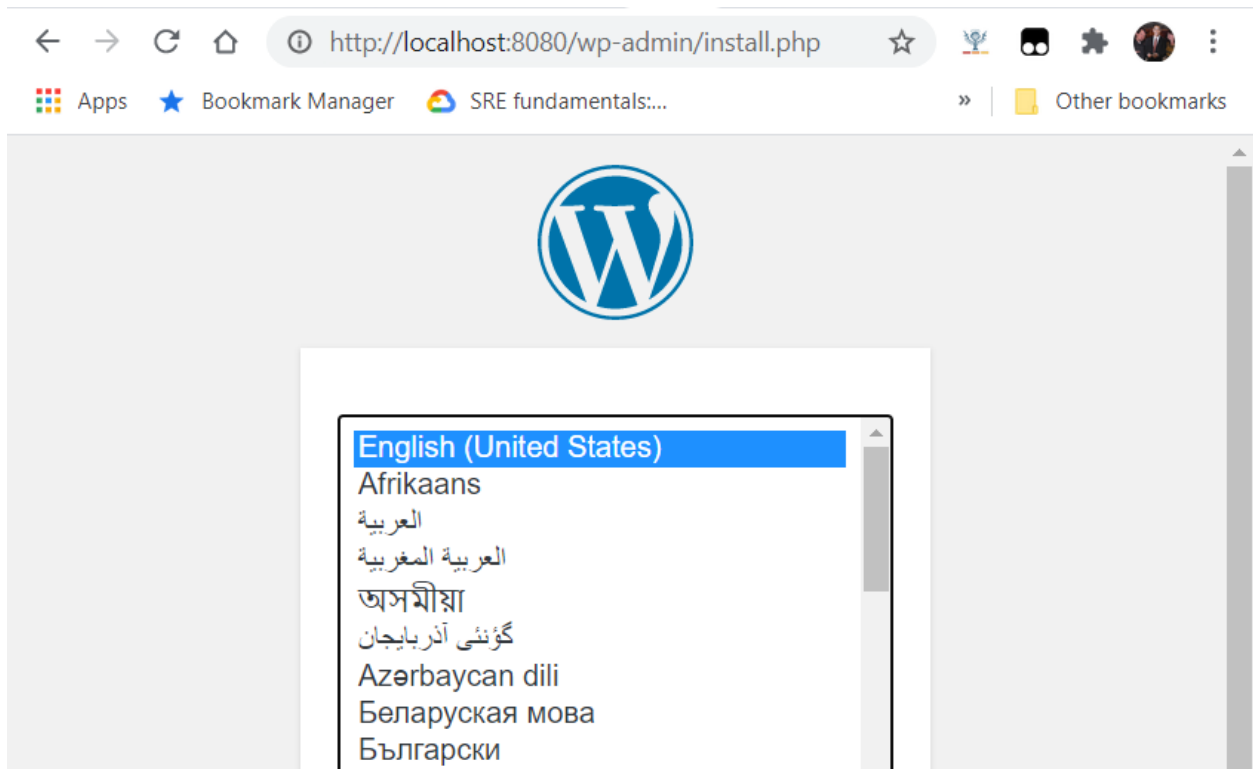
Note: We see that this is refused but recall that we see our service, but have not enabled a port forward to get to that WordPress instance

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.

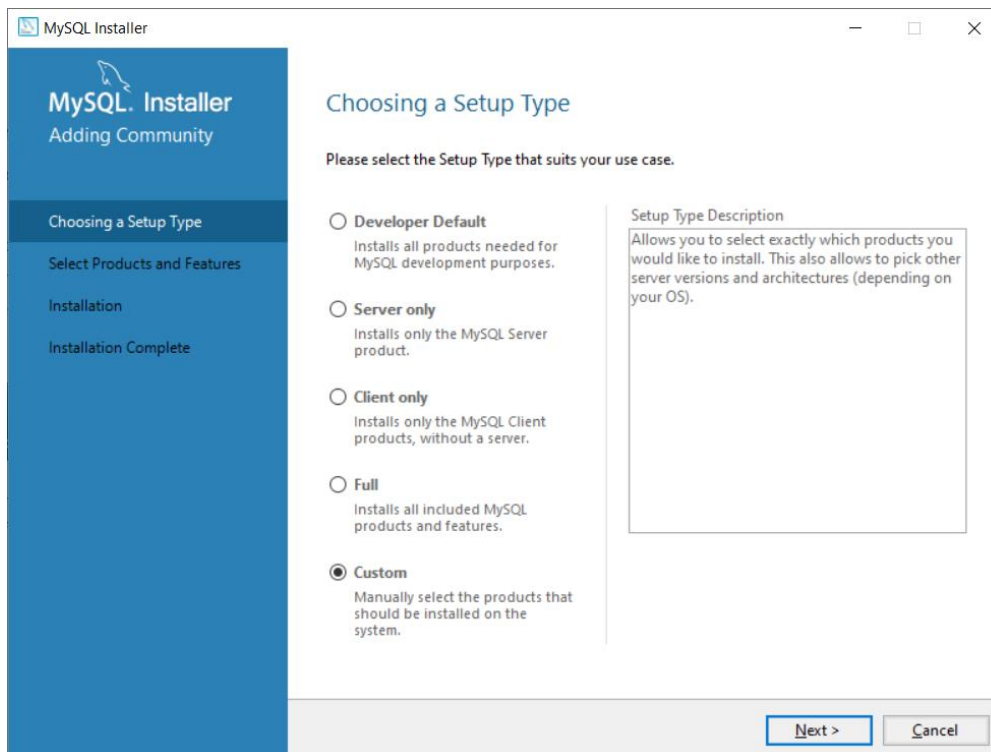
Windows (x86, 32 & 64-bit), MySQL Installer MSI

[Go to Download Page >](#)

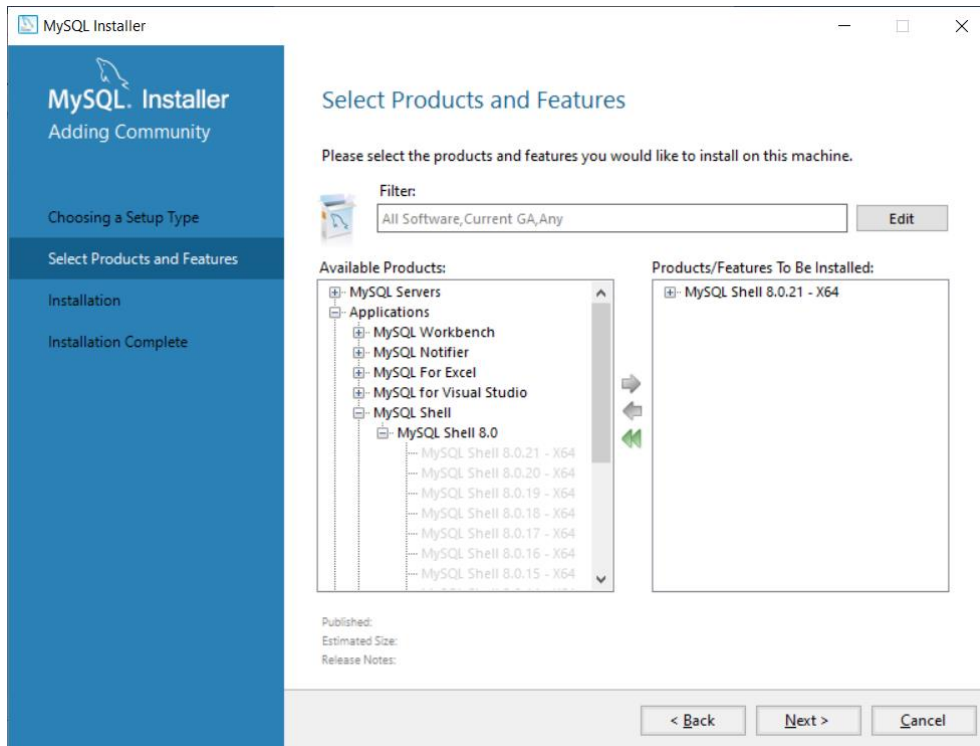
If you have an Oracle account, you can login or you can select “No thanks, just download”

Install the MySQL Shell, by running the downloaded installer

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

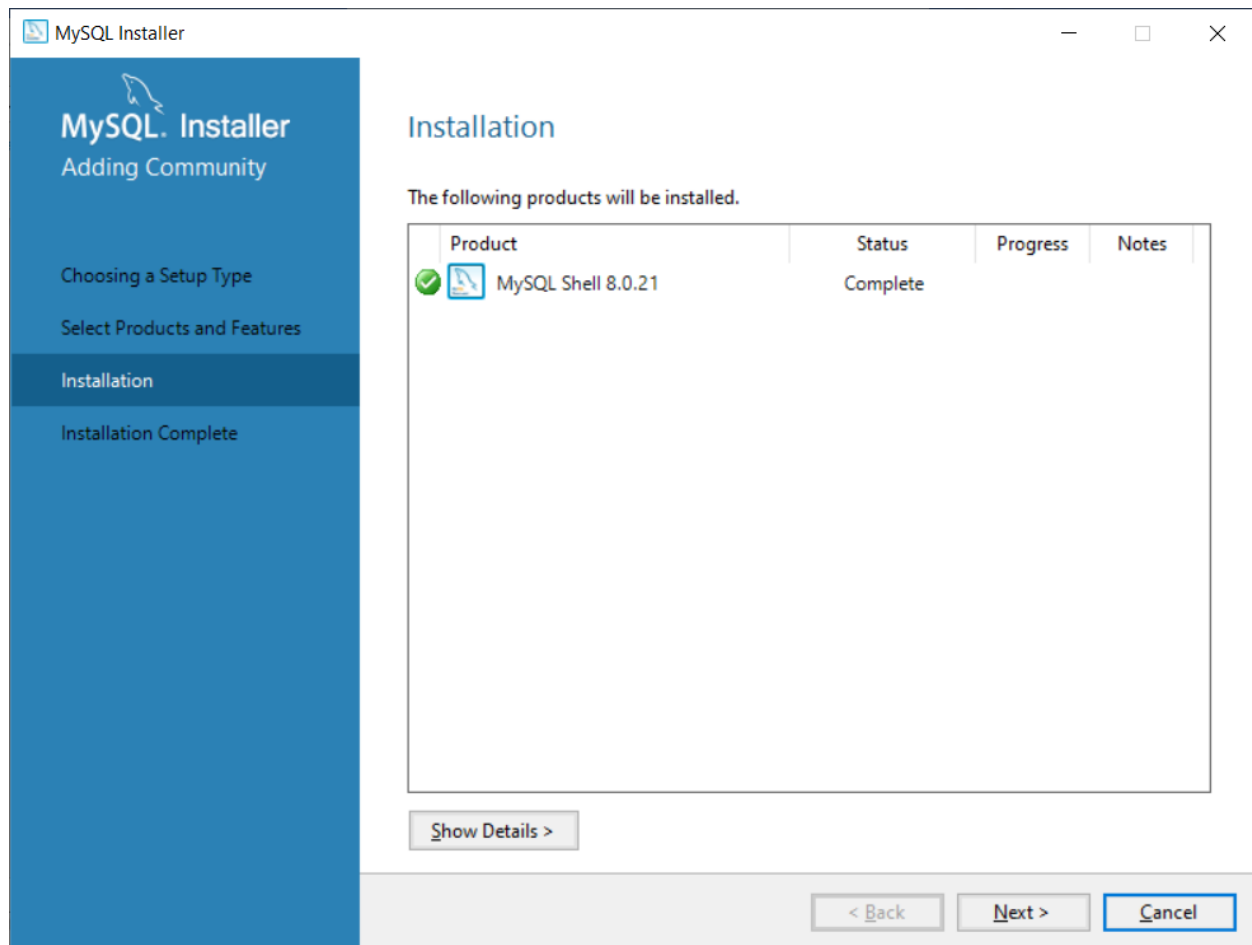


Select “Next”

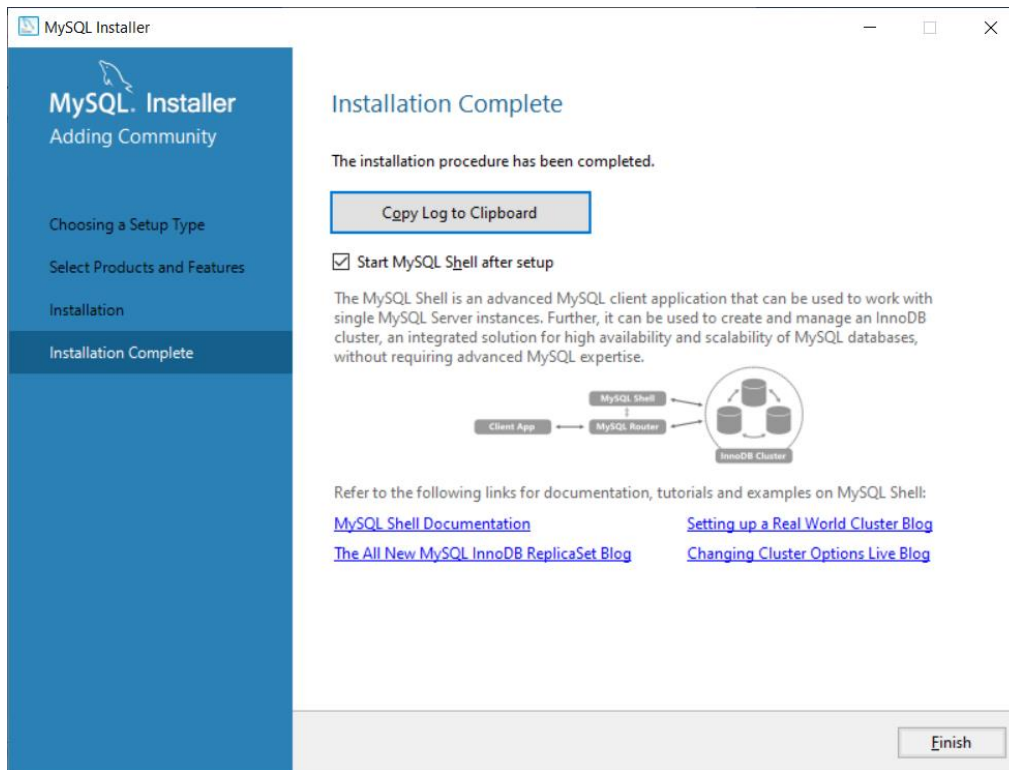


Expand the options under Applications -> MySQL Shell -> MySQL Shell 8.0 and then select MySQL Shell 8.0.21, selecting the top arrow to add that to the Products/Features To Be Installed list.

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
```

```
C:\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

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password Hide
Medium

Your Email
Double-check your email address before continuing.

Search Engine Visibility ☐ Discourage search engines from indexing this site
It is up to search engines to honor this request.

[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)

```
C:\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>
```

```
C:\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.

```
C:\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

And not surprisingly the same in Windows

```
D:\projects\kind>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! 🙌


```
D:\projects\Microservices_On_Kubernetes\labs>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

D:\projects\kind> **kind delete cluster --name grogu**

Deleting cluster "grogu" ...