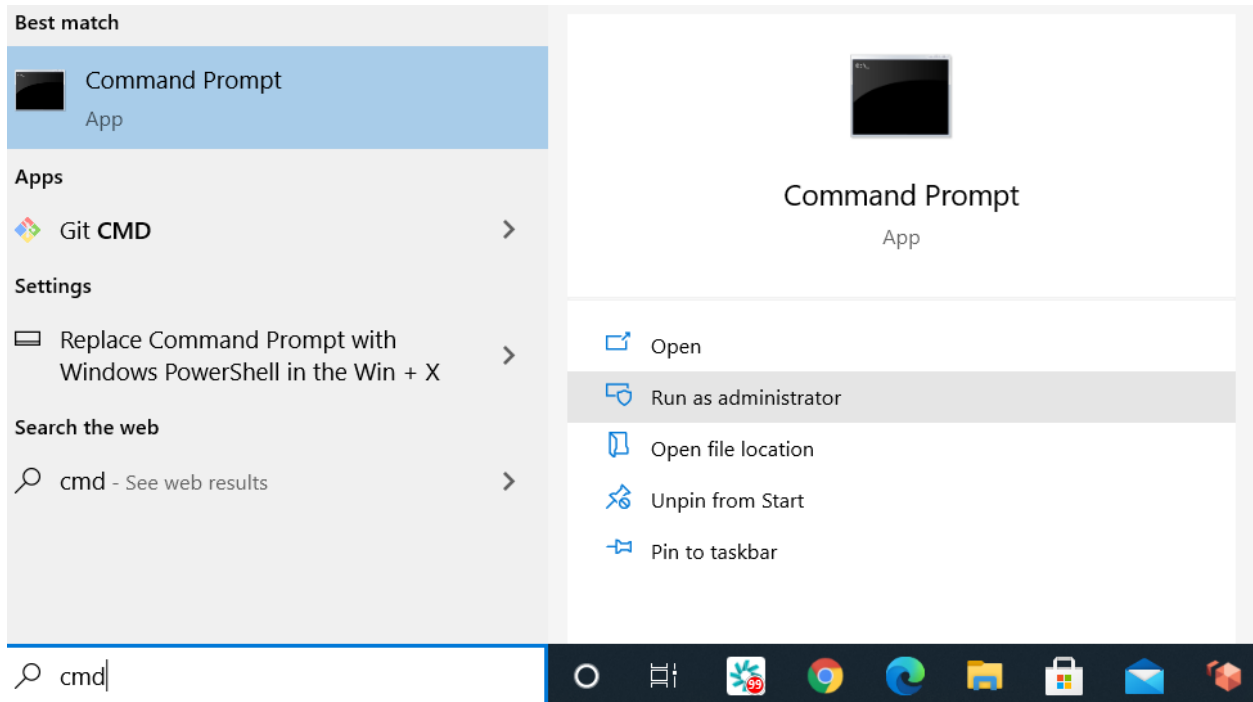


## Experiment 02: Working With Kind

In the Foundation experiment/lab we installed Docker and KIND, validate our Docker installation, and now we'll use KIND.

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
```

```
C:\Windows\system32>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

This command is converted behind the scenes to

```
docker ps -a --filter label=io.x-k8s.kind.cluster --format '{{.Label "io.x-k8s.kind.cluster"}}'
```

Check the kubectrl client version

C:\>**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

C:\> **kubectrl 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:

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURB
TkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUWFERXdwcmRXSmwKY201bGRHV
npNQjRYRFRJd01Ea3dPREF3TWpJME0xb1hEVE13TURd05qQXdNakkwTTFvd0ZURVRNQk
VHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFB
RGdnRVBIBRENDQVFvQ2dnRUJBTBJCk9oYXpaZjJNamx1WFh5aTQ0cFRBbXkPZEZ5QlJV
b0FIRmZHUlRzZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZl
Z3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3Uk
jByRGMwUm0zbXZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3UkZWZlZ3Uk
FuY0pkM3I1T3RGd1I2czdGbWxyb1RRNkFoVE5GZktBMTV2UHZ4Mmw2bEV3a0JGRIMrNjZ3
dERPT2dJZy9CCklwQklzcWZ3bzYwbFRFUGorOEczbnBEYtYDpNkVtNXNmZFIWZlZ3UkZWZl
1DL0cvdnN2Yks4MxV0anNNM1oyWmgKNUExNkxNlUlpNOXk2RzI2blluYjNlekM2UWNMVG9Y
Vm94YU95d0l0MFJlXBIvkn3U2hKRg9qcjU0ZTBWlncQo5MIFpeE1GWEphUndVc1M0eV
U4Q0F3RUFBYU1qTUNFd0RnWURWUjBQZVFILOjBUURBZ0trTUE4R0ExVWRFd0VCCi93U
UZnQU1CQWY4d0RRWUplb1pJaHJzTkFRRUxCUUFEZ2dFQkFMVUxZMkNXtVINSIJUbdQ
ZGVWSmQ3MmZoL24KbIFGWHRYT3dQblbmMEISR0grYTVwOVcyZytKRzQ4QjUwSUtZWFEr
VEZOTVZ0QUYrMDFWTFJRbzFQUEhudTM5NQppYlphSnNyBTBhWGVwOHFISWF2Lzl0MUx
HcWk5dThkdFV6cjYvVEkwK2dUK2IEZIZ3bHRYSlZiVXN4Q2U1VE5UCmdQW9GWjBOdDVC
THUzRkdrV0x0X0RNSlkyU0JNckNZZ0ZhZHg5bkVSMjd4L0NsdjEvR3picEhWRmJmYVlubmQ
KUFNZbVRiWncxYlgyZzJBdC9pQ1BSbDJGbzXYzbXBTdENjaEppVDNkK0U5aEtpV2p4T0kvVD
FsN2E5UWxJTXdpdApldjZoMlJkUTQvZVdkeTFxY2hDYkZ6N0UyZEtvSXRONFFseFhQcVIMQ
1VnYzB4TEFGK2kxdkV1bDM3Zz0KLS0tLS1FTkQgQ0VSVElGSUNBVEU0tLS0tLQo=
```

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:

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURStLS0tCk1JSUM4akNDQWRxZ0F3SUJBZ0lJU3VyaXQxdGJKNnN3RFFZSkvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TURBNU1EZ3dNREI5TkROYUZ3MHINVEE1TURnd01ESXIORFJhTURReApGekFWQmdOVkBB1REbk41YzNSbGJUcHRZWE4wWlhKek1Sa3dGd1IEVIFRREV4QnJkV0psY201bGRHVnpMV0ZrCmJXbHVNSUICSWpBTkNa3Foa2IHOXcwQkFRRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXo1K1d0L242RGtEbm5QblcKMFVmb3kxM1ArSldjQi9BMXdBTU3L2RSREpIZ1g4N1lqOFRibkRDdUNhcElxVFY0ZjBqdW9JdDRDZmNKQUxMVgpXTWxhYVhncmljSXgxexjBdzIEUDVNRHM2VkdJcWRKaGc1Z0ROQWVYdTIzZUNuaUgxVDBnUFJxNjdKWHY4YW9VCi9YbiY3UFdXNDVGbDlvd0NwSEc5Vlp4MUNlaVVOd1RVb09pd29oYzhpWmh6QzNHdS9CZ3FRbDR5K3hMVU0wTDdkKWxzEY0NBU2dYdEhCQkU3TWN2SWxlQWFFdmdKVV k0c1MvMUhZMytZQmZVcCtzUzZjchBKWFpveXhaTGdaMm51YgpTYmVhQ0J0Wk9wbGR3YTgvV29SbGtrRkFpK0RURkt3cEZmTC9DYkxqOHd3QTVXdnNtWVdyWkFRZGE2aTBKQ2k5CmdaMzhCUUIEQVFBQm95Y3dKVEFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1IEVllwbEJBd3dDZ1lJS3dZQkJRvUgKQXdJd0RRWUpLb1pJaHJZjTkFRRUxCUUFEZ2dFQkFCOS9FL3o1cnNBeDNoaDVxVlo3ZUswTy95bUtES05SOFVDTApwc21kaVl2dm1qZEVkc0I2WFhWM2dMeWQ1TmF0SSt2WU5rbS9YODlqNTIURXNVEGhtS2V2SEorQmRZWXBpCkEQ5CmY3Y3dBNnNIS2IEcGJUMVVuajRoeHJoZyt6VEFXN3hleHZEam8vdG5xazg1dDR1bzVLdFVrRjRfNEN1QXQ3TFUKUjJaTIBta2UwdTRsUWgzaFU5Slg1bzY5N3FHb21EMkZTeDdvb212eXdDMS8rbFBIVkNURk95U2pPbXEya001VApiZCtRbmhoeVVRXJheC9sbnV5SDhkUIJYZkZWelk1WEILNS81RnBBMFFUUKltRXJUVExSem5mckNtNXVas3VTCjVFL0d4Z083MU1vZGN3ajM2NWZKWE9JT1c0UIZ4TXJa bEpnQVZla0lGUitQSUV0bIERRT0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=

client-key-data:

LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUIfcEFJQkFBS0NBUEVBejUrV3QvbjZEa0RublBuVzBVZm95MTNQK0pXY0lvQTF3QU01Ny9kUkRKSGdYODdZCmo4VGJuREN1Q2FwSXFUVjRmMGp1b0l0NENmY0pBTEwVW01sYWYFYZ3JpY0l4MXo3QXc5RFA1TURzNIZHSXFkSmgKZzVnRE5BZVh1MjNIQ25pSDFUMGdQUe2N0pYdjhbb1UvWG5WN1BXVzQ1Rmw5b3dDcEhHOVZaeDFDSGIVTndUVQpvT2l3b2hjOGLaaHpDM0d1L0JncVFfSNHkreExVTTBMOVI2RGNDQVNnWHRIQkJFN01jdklsSEFhRXZnSIVZNHNTCi8xSFkzK1ICZlVwK3NTNmNwcEpYWm95eFpMZ1oybnViU2JiYUNCdFpPcGxkd2E4L1dvUmhra0ZBaStEVEZLd3AKRmZML0NiTGo4d3dBNVd2c21ZV3JaQVVFkYTZpMEpDaTlnWjM4QlFJREFRQUJBb0lCQUVzYmxFNWhvOC9jTXUxYQpoQmVaUitHcHdqNVBBTzd1T3NPSFowSWoyYkIPWTNqRIB4cGpRSdYwTFIGWmxJZUJ6R0ZmWk5PM0IHbWFiQ3RNCmhsbGtIY3pocC81aHZkMzcyWWY4MWZnT3dxVjAxVmQ1djhUM0ROR1puWTQwSklydEoxWkFrcFVJUW02cm51MXgKZGI1c2dMTUQ5TjNHRDNpdEZAZWZmYnFtcXlreEYvN2tTQ0ZKSEpyR0JoMk9GT3JSTXdxUXREaFBjWS9DTzN6VQoybVBOTXN2Rk5JQjBoK2IKVXEwWEpwOHAvd3hmYkYva3FuRXBEYTZhaTFXT21NdzhzUjFtRFZqSjR3TWtZdEkwCjZrTWFTdHYvVVBBeVlVZzFvV0ZTeGFGcGs4OEh6UUp1ZnpXamZ6c2Z4bDN4ZnNwY1FKNzNZUIZzb29kRFdWTEUKRkwxVFRBRUNnWUVBODVWwklMcHp3M0lSa21rQkJOT0VLk5wWWNOZEYyOUNDOGNuYWJRMFdmNVlqUWEzdjBYMgo5YXdiVIIQVXBCck1wZUY0VINPalhSTWplazkwZDlzM1dwRlkdjJlUjBjSTXdrYy9MVXNlVjJ6S2xUczZmanZFCi8vU0VTRm9FS2ZmWmkzbjAzUm81R3gvdWZxK1YwUnl6L0pNMzIKedl5N0o5NHVHdE1FSEVCY1VDZ1IFQTJqVXoKK1VzSnRBQ2NwNk1J3TDNOV1h3U29aMk9nbzZnYjBOclpTUW9YbnNwdGt0UmF0R3NER3JTWG1CeGJNRUloVklERQpKbDBNTGc3QnFEaDcwVklBbmNjWmQzQkhNd0NYMmgzNDN6NmJZSTZ1RFBSU1k0amd4WHB0djBaVDdIV1ZDVfgwCk9TWnArYkpuMFpJb3gwSzbHc0FicDk2WThlZnRReUoyU2FOendVRUNnWUVBbytzWVFONEluMk01RGdVWwNlXeXEkQ1FJU2pkYlI0NzVjZk42VjJGMkx5Smkzc0pmdnVZbFV5emo1NEE5cVh0RW1IUTlOVWpJOGNwcZVTK0YrYUEzeQpNSUdWZm9DQmM0QTBNTI4bHpkaGhtVFE0NkpMRjc0VE1ZZ1VLVGhpaXZlZTcyYXZc2NGM1FvZERpWU5OUdHtCjVJc1I0Y3dWADFVWHdFSE1zYWZnU1YwQ2dZRUFPuYmY2Rw5zeG1uVHo5MkVXZXNsMUQzZW1zbVptcTh2WHhnMXAKakxrWmcvdGNnbTZHbW1uTlpuN2w2M3IOVWpHS0xvUkZISERuVVJ5V1VURkRvWXhxdExNWk92QkEyMn

```
RZL0lIdQpOWnhwRkc0d0xoVm1tZ0NLyJZmdXdjald1MjVZZDVQOVg5YWhxRzZOU1o4Um5iZ
HlzbzZkZ0x1YXpTSWI0QjBMCndsSStUTUVDZ1ICSTFrek5EeHhWUFVucERleXNnWmxIS3Qw
cDltTzhKWJlGcU5UM044eXY1OEZnbtkSFU2WmYKWXZUNDdkK3FxdHRYREJEa1ZIWkJ4Kz
VSdW5kWIgxM3NKUkg2NUhVeE5QczBkdGVLa3RObnh5Zm9ubEFZTU13QgorWXBIMFVUT2
xaK3FOT3d1UzFkZHM2MU14SVZWRIJxTDVzVGIVZGdkU3VLenU3ZW41R2hY3c9PQotLS0t
LUVORCBSU0EgUFJJVkfURSBLRVktLS0tLQo=
```

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
```

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

Paste the following content into our file and save.

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

Alternatively you can clone the git repo for the class which has this file in the labs subfolder

```
$> git clone https://github.com/GeorgeNiece/DevOpsForMicroservicesWithKubernetes-3day.git
```

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.

Download the following two resource files to our c:\projects\kind\kind-wp or ~/projects/kind/kind-wp folders

```
$ 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 c:\projects\kind\kind-wp folder, if it wasn't save 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, this will result in a 404, and we’ll correct that gap in the next update

open <http://localhost:8080/>

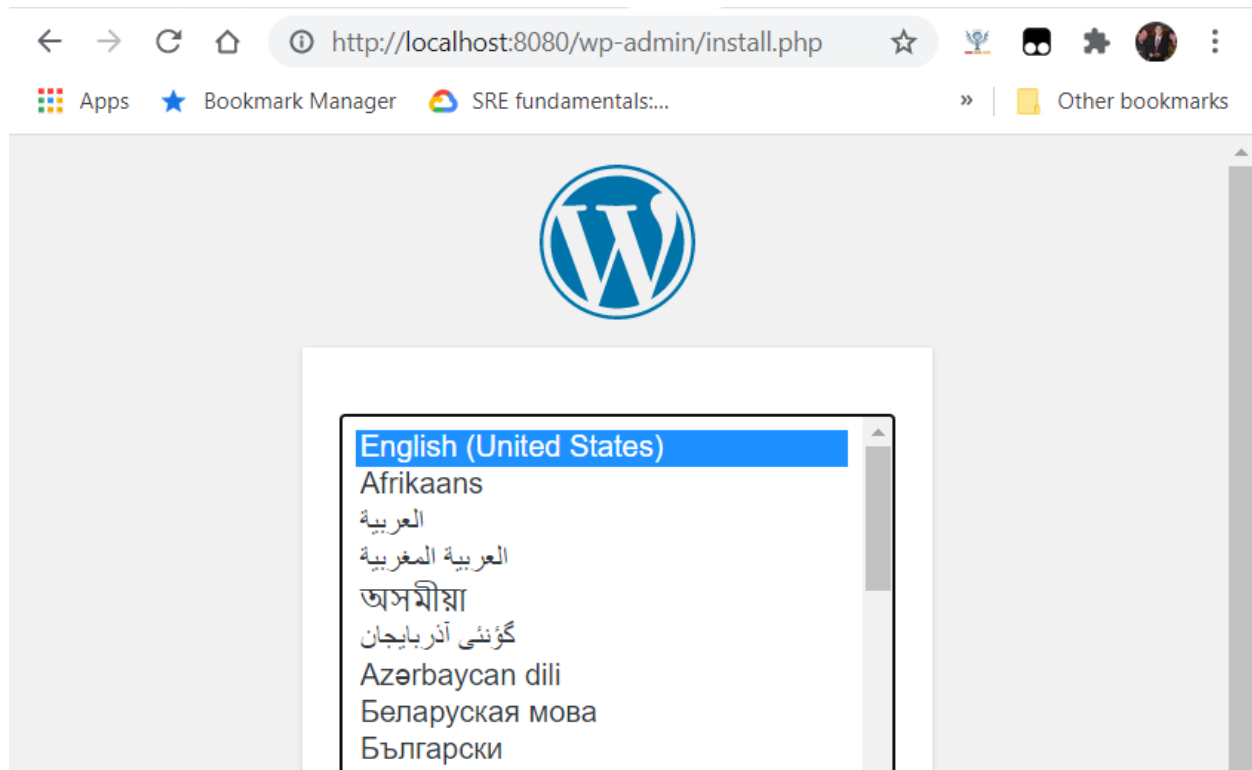
Note: We see that this is refused but recall that we see our service, but have not enabled a port forward to get to that WordPress instance

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**

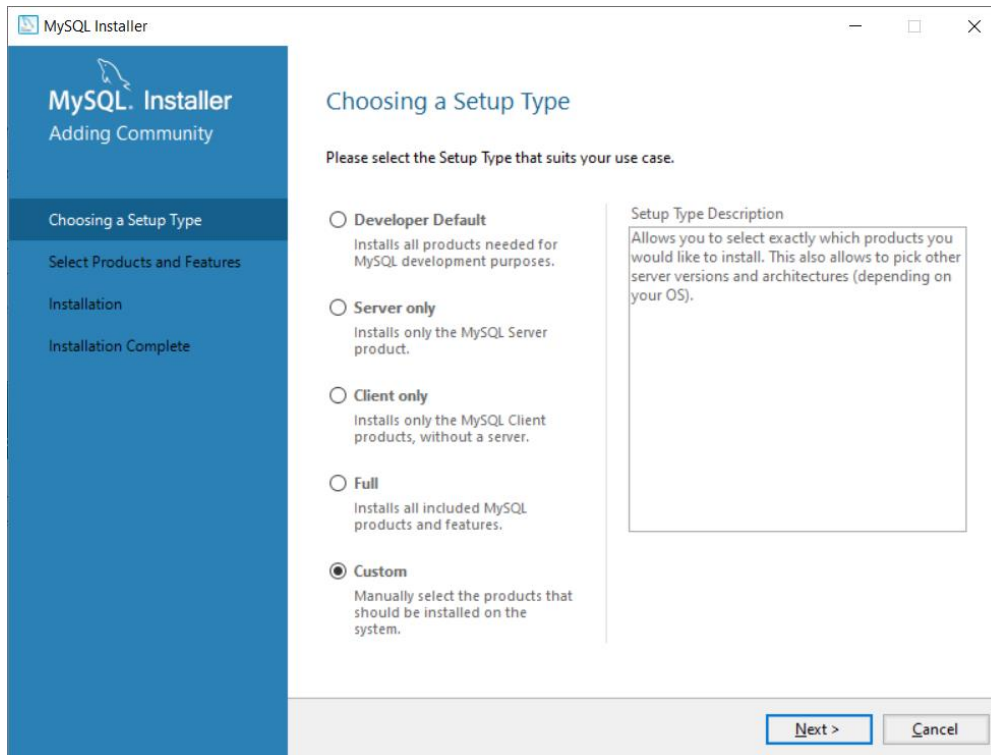


[Go to Download Page >](#)

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

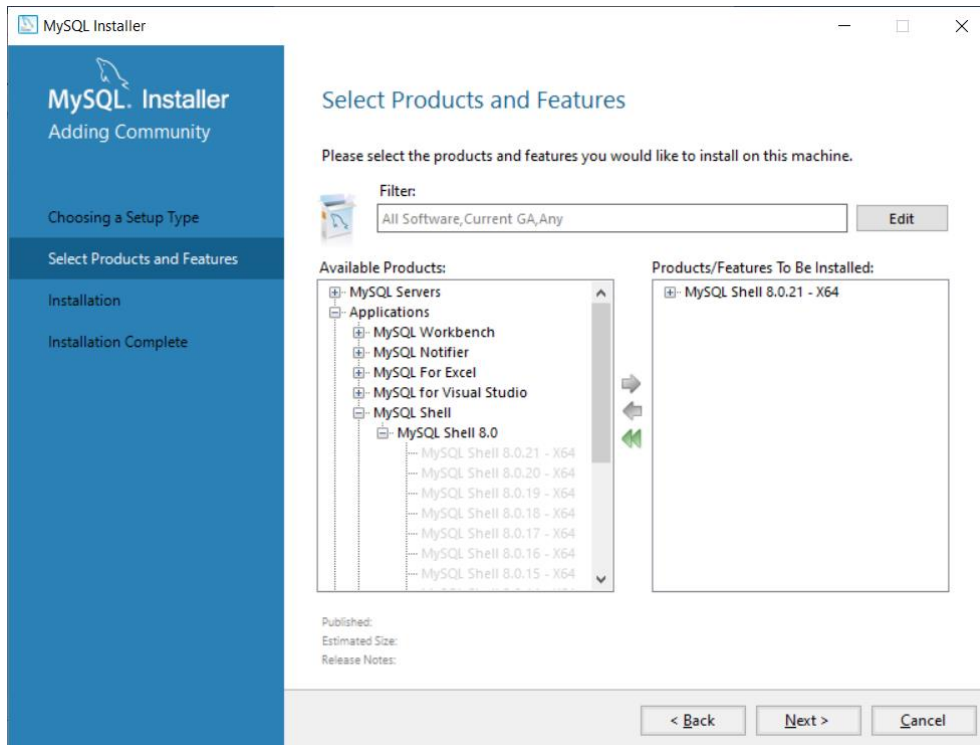
Install the MySQL Shell, by running the downloaded installer

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



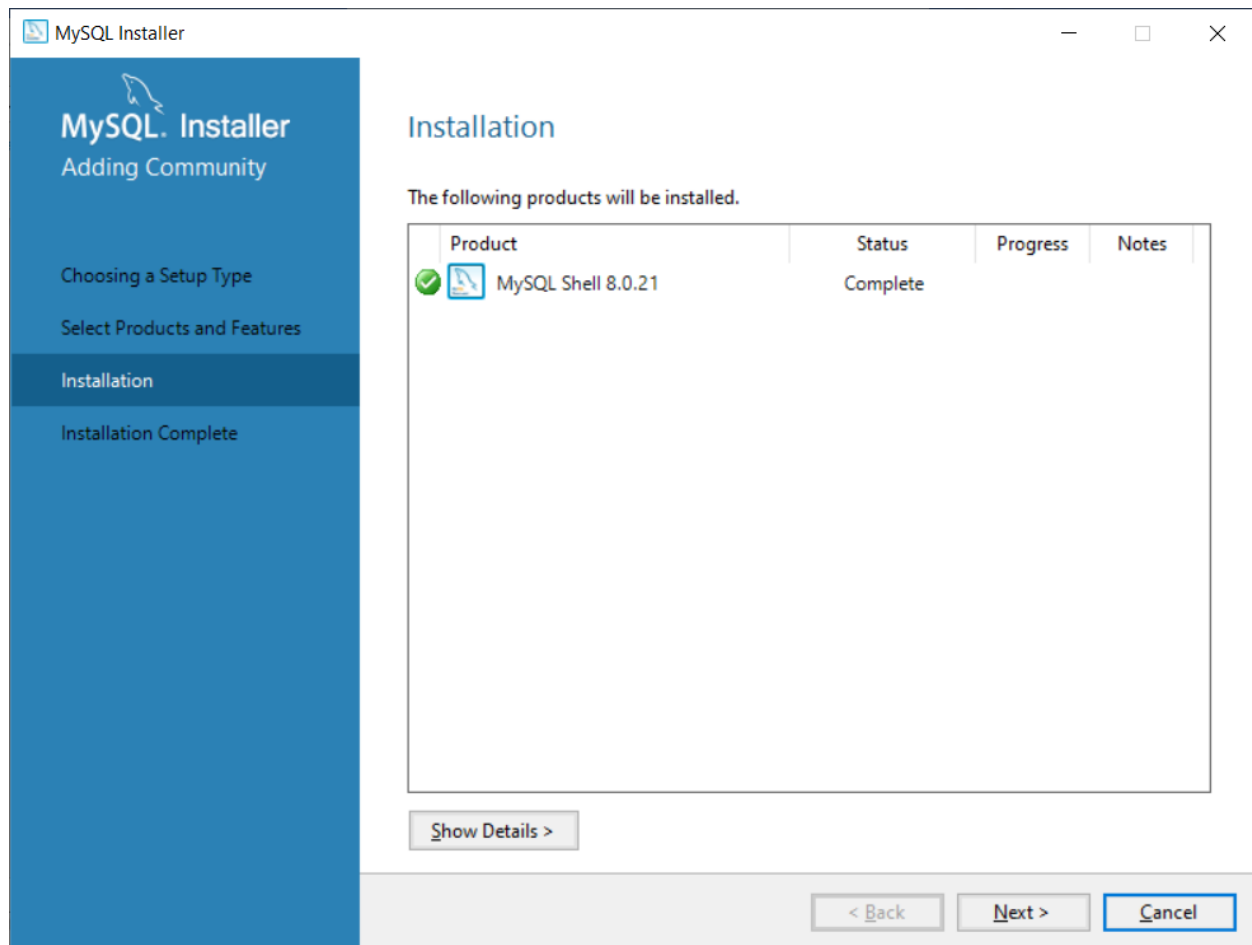
Select “Next”



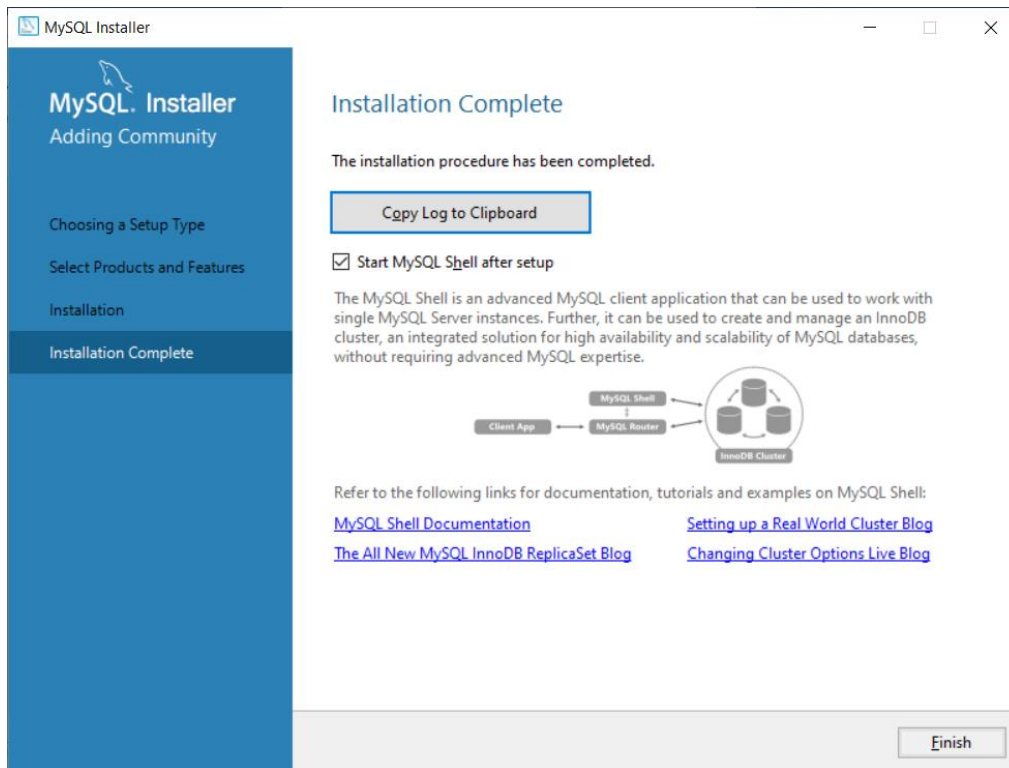


Expand the options under Applications -> MySQL Shell -> MySQL Shell 8.0 and then select MySQL Shell 8.0.21, selecting the top arrow to add that to the Products/Features To Be Installed list.

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

Enter `\connect root@localhost:3306`

Or from MacOS you can

```
$> mysql -h 127.0.0.1 -u root -P 3306 -D wordpress -p
```

MacOS MySQL client install note:

If the brew install fails you can download the DMG from <https://dev.mysql.com/downloads/shell/>

Then run from Security and Privacy

```
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

```
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" ...

If you were to see an error relating to “docker ps” after the deletion, that would indicate your

Docker Desktop is not running.

```
docker ps -a --filter label=io.x-k8s.kind.cluster=kind --format '{{.Names}}'
```