

## 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 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

```
$ 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

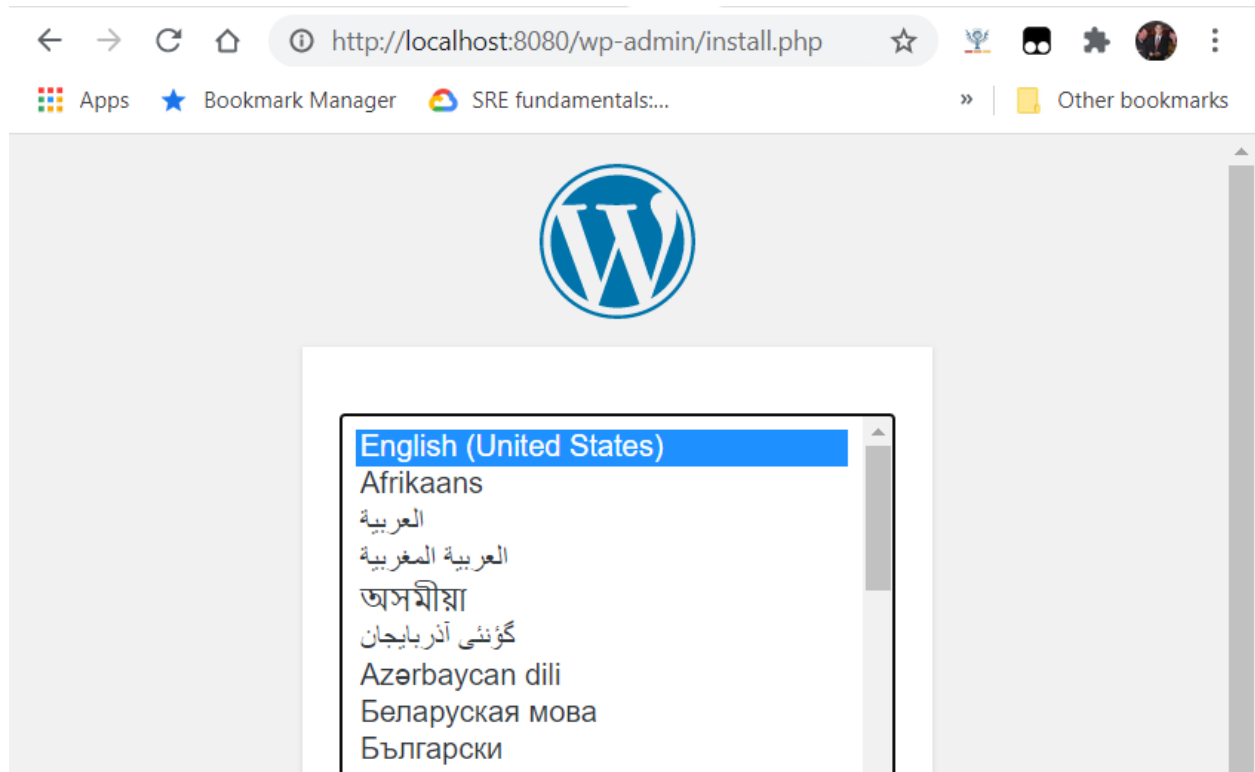
Note: We see that this is refused but recall that we see our service, but have not enabled a port forward to get to that WordPress instance

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.



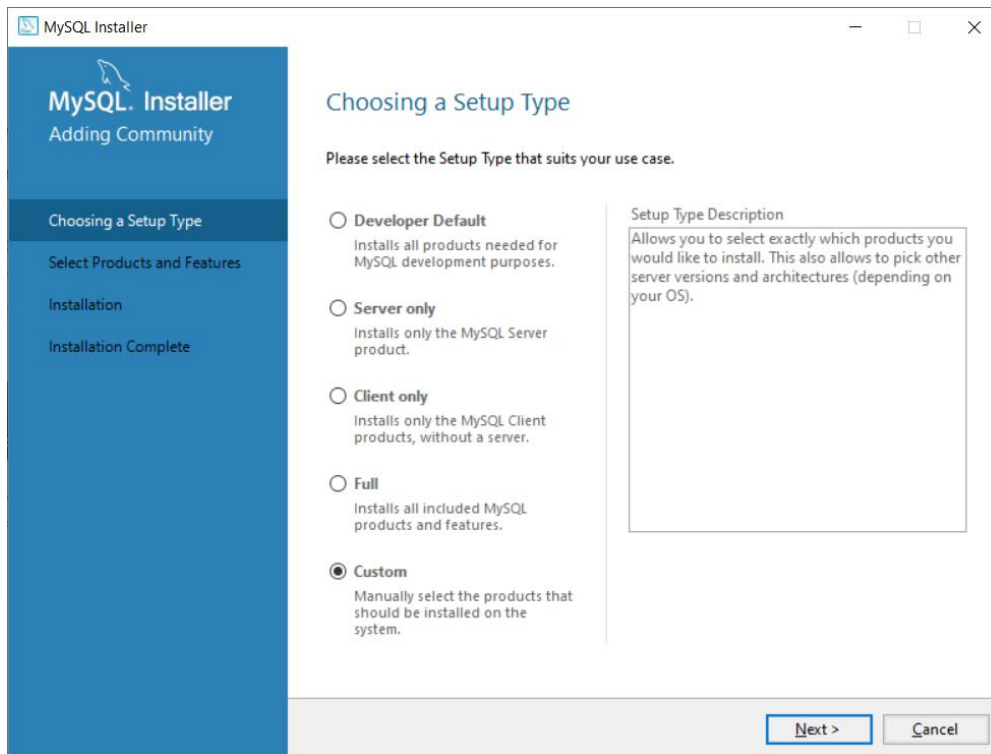
**Windows (x86, 32 & 64-bit), MySQL Installer MSI**

[Go to Download Page >](#)

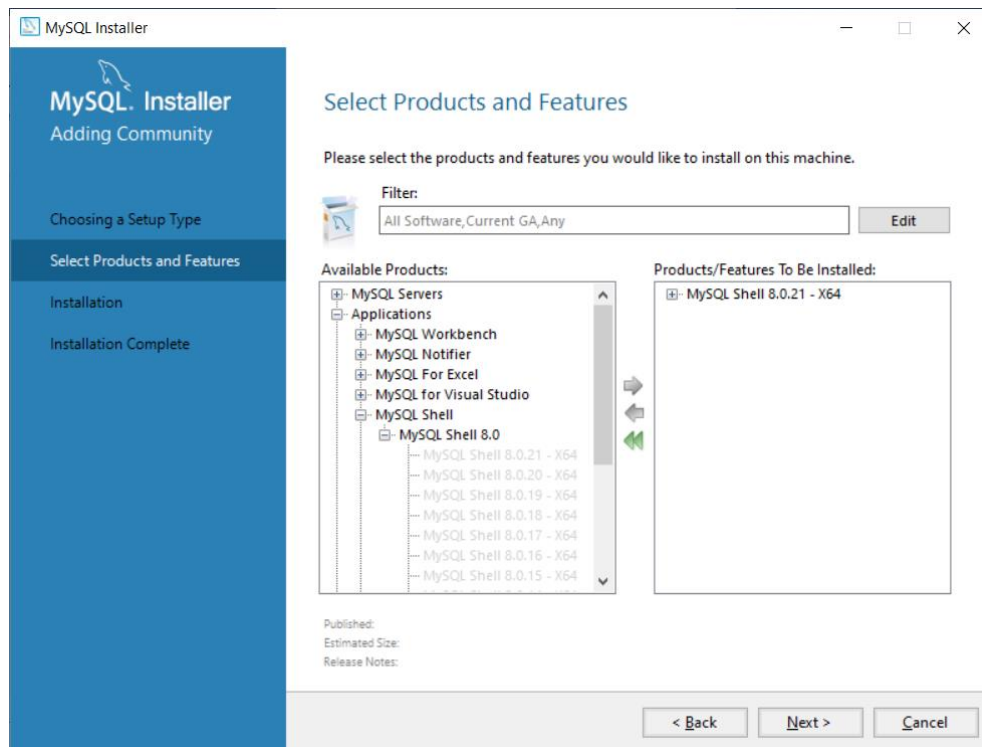
If you have an Oracle account, you can login or you can select “No thanks, just download”

Install the MySQL Shell, by running the downloaded installer

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

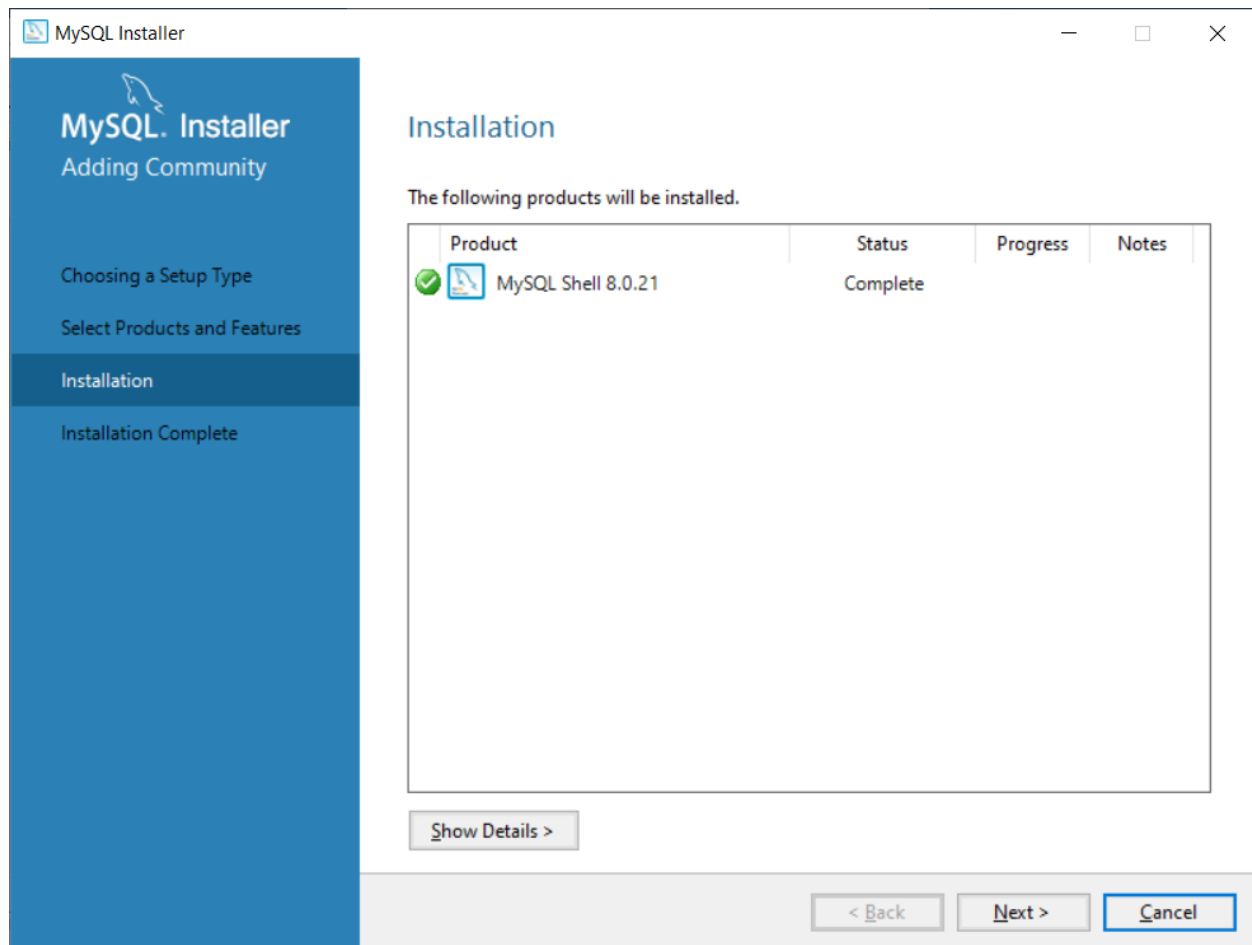


Select “Next”

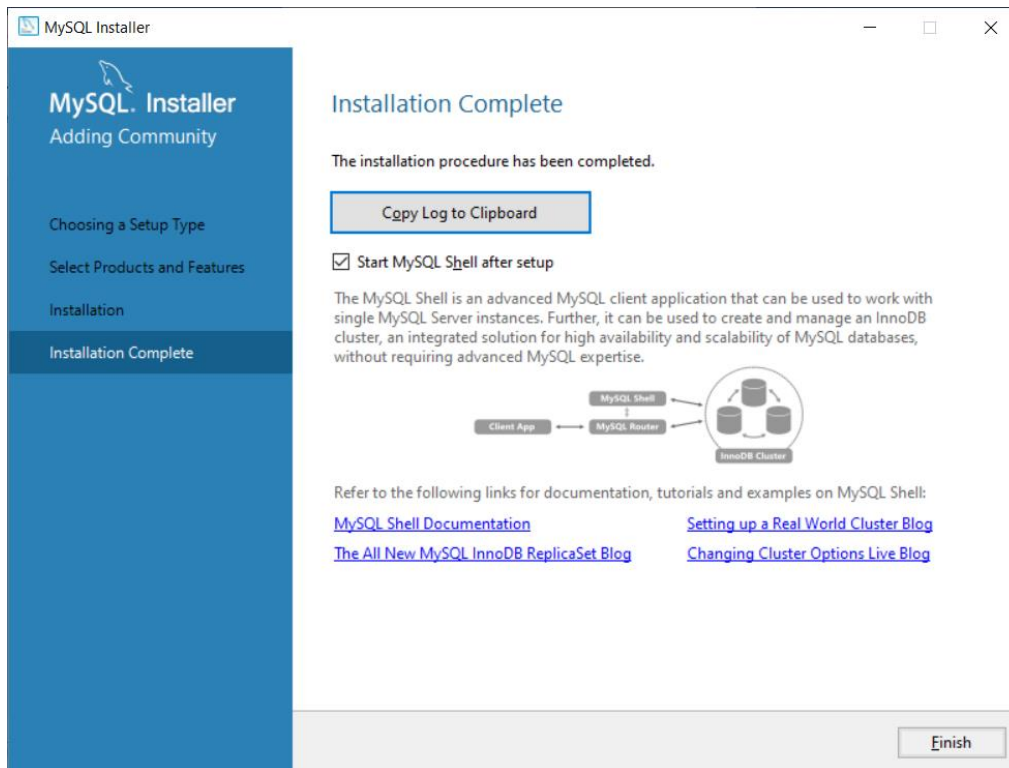


Expand the options under Applications -> MySQL Shell -> MySQL Shell 8.0 and then select MySQL Shell 8.0.21, selecting the top arrow to add that to the Products/Features To Be Installed list.

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
```

```
$> 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**    
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.

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)

```
$> 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>
```

```
$> 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.

```
$> 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
```

```
$> 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" ...

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

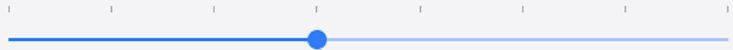
Docker Engine

Experimental Features

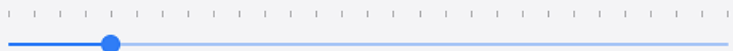
Kubernetes

## Resources Advanced

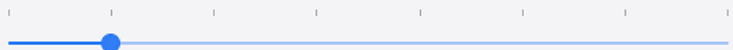
CPUs: 4



Memory: 2.00 GB



Swap: 1 GB



Disk image size: 59.6 GB (11.7 GB used)

