

Homework 4 : Docker & Kubernetes

1. Docker

Section 2: Getting Started Walk-through for Developers

STAGE 1 - The Basics

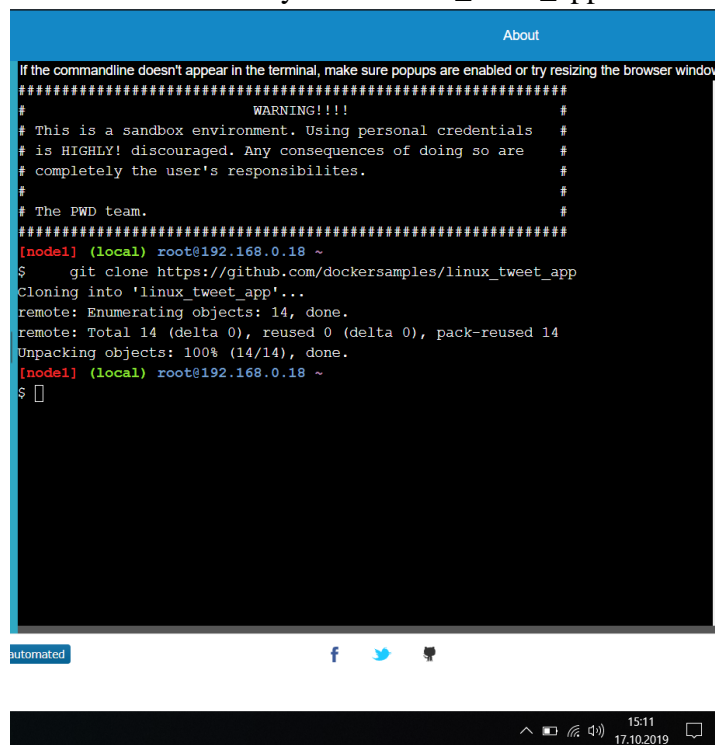
Sub-Module 1 - Docker for Beginners - Linux

<https://training.play-with-docker.com/beginner-linux/>

- Follow the on-screen instructions and complete all the four tasks mentioned below –

- **Task 0: Prerequisites**

First, I cloned the lab's repo from GitHub. This created a copy of the lab's repo in a new sub-directory called `linux_tweet_app`.



The screenshot shows a terminal window with a blue header bar containing the word "About". The terminal output is as follows:

```
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
#####
#                               #
#   WARNING!!!!                 #
# This is a sandbox environment. Using personal credentials   #
# is HIGHLY! discouraged. Any consequences of doing so are   #
# completely the user's responsibilities.                       #
#                                                             #
# The FWD team.                                               #
#####
[node1] (local) root@192.168.0.18 ~
$ git clone https://github.com/dockerexamples/linux_tweet_app
Cloning into 'linux_tweet_app'...
remote: Enumerating objects: 14, done.
remote: Total 14 (delta 0), reused 0 (delta 0), pack-reused 14
Unpacking objects: 100% (14/14), done.
[node1] (local) root@192.168.0.18 ~
$
```

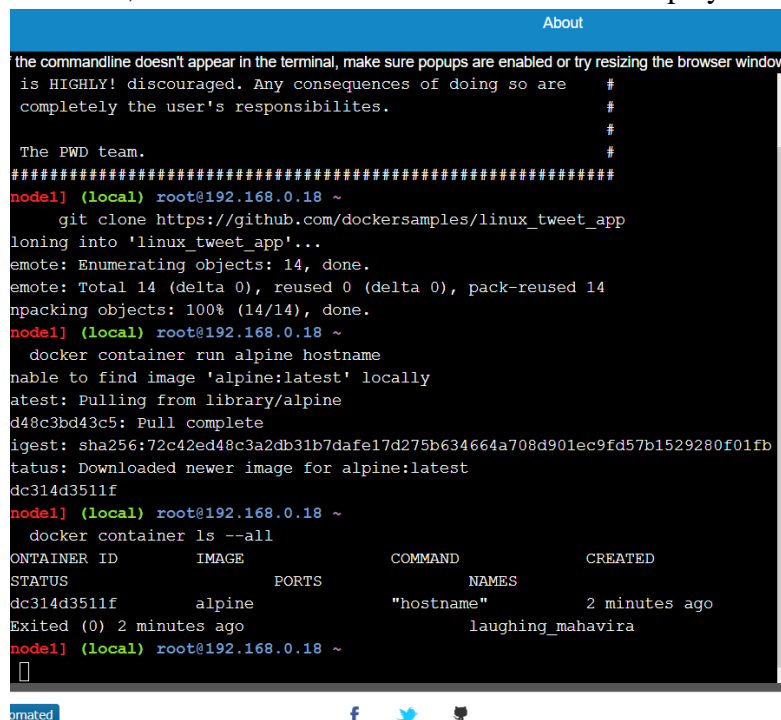
At the bottom of the terminal window, there is a status bar with the word "automated" on the left, social media icons (Facebook, Twitter, GitHub) in the center, and system icons (network, battery, volume) on the right, along with the time "15:11" and date "17.10.2019".

- **Task 1: Run some simple Docker containers**

Run a single task in an Alpine Linux container

Next, I started a new container and told the contained run the `hostname` command. It is looking on my local machine whether an image is available and if not, an image is automatically pulled from docker hub. After it got pulled, Docker is able to display the hostname. However, Docker only keeps the

container running as long as a process is running. When the hostname is displayed, the process ended and docker closes the container automatically. However, the resource is not deleted and can be displayed

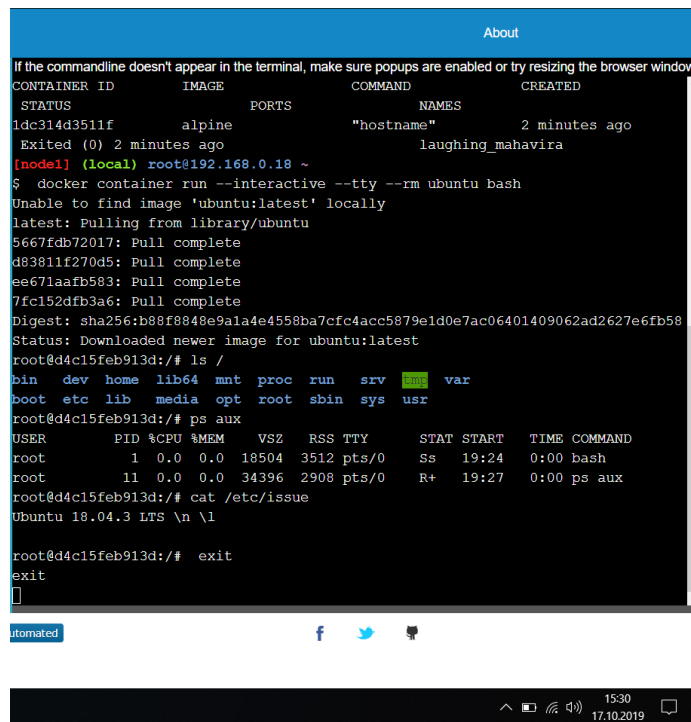


```
the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
is HIGHLY! discouraged. Any consequences of doing so are #
completely the user's responsibilities. #
#
The FWD team. #
#####
node1] (local) root@192.168.0.18 ~
    git clone https://github.com/dockerexamples/linux_tweet_app
Cloning into 'linux_tweet_app'...
remote: Enumerating objects: 14, done.
remote: Total 14 (delta 0), reused 0 (delta 0), pack-reused 14
Unpacking objects: 100% (14/14), done.
node1] (local) root@192.168.0.18 ~
    docker container run alpine hostname
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
d48c3bd43c5: Pull complete
Digest: sha256:72c42ed48c3a2db31b7d275b634664a708d901ec9fd57b1529280f01fb
Status: Downloaded newer image for alpine:latest
dc314d3511f
node1] (local) root@192.168.0.18 ~
    docker container ls --all
CONTAINER ID        IMAGE               PORTS              COMMAND              NAMES              CREATED
STATUS
dc314d3511f        alpine              "hostname"         laughing_mahavira    2 minutes ago
Exited (0) 2 minutes ago
```

Sometimes, I only want to execute a script. In this case, docker can keep the container open as long as the script is running. When I share this script and it will be executed on another machine, the code can run in the same container environment and will not complain about a missing packages and/or libraries.

Run an interactive Ubuntu container

It is also possible to run a container based on a different version of Linux than is running on your Docker host. Next, I will run an Ubuntu Linux container on top of an Alpine Linux Docker host. So, I tell Docker to start a container in an interactive session and remove the container when it is stopped. Moreover, I include a bash statement to be able to run commands in the container: First, I look into the contents of the root directory of the container, next, I list all running processes and then request which on which linux distro, the container is running – here: ubuntu. With the command ‘exit’, I leave the shell session of the docker.



```

About
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
1dc314d3511f       alpine             "hostname"         2 minutes ago
Exited (0) 2 minutes ago    laughing_mahavira

[node1] (local) root@192.168.0.18 ~
$ docker container run --interactive --tty --rm ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
5667fdb72017: Pull complete
d83811f270d5: Pull complete
ee671aafb583: Pull complete
7fc152dfb3a6: Pull complete
Digest: sha256:b88f8848e9a1a4e4558ba7cfc4acc5879e1d0e7ac06401409062ad2627e6fb58
Status: Downloaded newer image for ubuntu:latest
root@d4c15feb913d:/# ls /
bin    dev    home  lib64  mnt    proc   run    srv    tmp    var
boot  etc    lib   media  opt    root   sbin   sys    usr
root@d4c15feb913d:/# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  18504  3512 pts/0    Ss   19:24   0:00 bash
root        11  0.0  0.0   34396  2908 pts/0    R+   19:27   0:00 ps aux
root@d4c15feb913d:/# cat /etc/issue
Ubuntu 18.04.3 LTS \n \l

root@d4c15feb913d:/# exit
exit

```

With 'cat /etc/issue' I can check the version of the VM. It becomes clear that the host VM is running Alpine Linux and I was able to tun a Ubuntu container. It shows that the distribution of linux in VM and Docker do not need to match. However, it is important that both run on the same linux kernel. So, linux containers cannot run on windows hosts and windows containers cannot run on linux hosts.

Run a background MySQL container

I can background containers to run an application in the container. So I create a MySQL container with the name mydb. Again, as I do not have any associated image stored, it pulls one from Docker Hub again. This MySQL process is running on the background, so as long as the process is running, the container will keep being open. With 'docker container ls', I can list all running containers.

```

About

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
[node1] (local) root@192.168.0.18 ~
$ docker container run \
> --detach \
> --name mydb \
> -e MYSQL_ROOT_PASSWORD=my-secret-pw \
> mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
80369df48736: Pull complete
e8f52315cb10: Pull complete
cf2189b391fc: Pull complete
cc98f645c682: Pull complete
27a27ac83f74: Pull complete
falf04453414: Pull complete
d45bf7d22d33: Pull complete
3dbac26e409c: Pull complete
9017140fb8c1: Pull complete
b76dda2673ae: Pull complete
bea9eb46d12a: Pull complete
elf050a38d0f: Pull complete
Digest: sha256:7345ce4ce6f0c1771d01fa333b8edb2c606ca59d385f69575f8e3e2ec6695eee
Status: Downloaded newer image for mysql:latest
b56b1d24a8e253c8d9226676568e5fa65bf243689aa984dde491302d47233dcb
[node1] (local) root@192.168.0.18 ~
$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES

```

I can check the log of the running container, check the processes in the container and use 'docker container exec' to run commands within the container. This way, we can check the version of our application for instance. Moreover, I can connect to a new shell process within the container.

```

About

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
pem is self signed.
2019-10-17T19:41:24.849329Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2019-10-17T19:41:24.874774Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.18' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
2019-10-17T19:41:24.951986Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: '/var/run/mysqld/mysqldx.sock' bind-address: '::' port: 3306
[node1] (local) root@192.168.0.18 ~
$ docker container top mydb
PID             USER           TIME           COMMAND
15064           999            0:05           mysqld
[node1] (local) root@192.168.0.18 ~
$ docker exec -it mydb \
> mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql Ver 8.0.18 for Linux on x86_64 (MySQL Community Server - GPL)
[node1] (local) root@192.168.0.18 ~
$ docker exec -it mydb sh
# mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql Ver 8.0.18 for Linux on x86_64 (MySQL Community Server - GPL)
#

```

- **Task 2: Package and run a custom app using Docker**

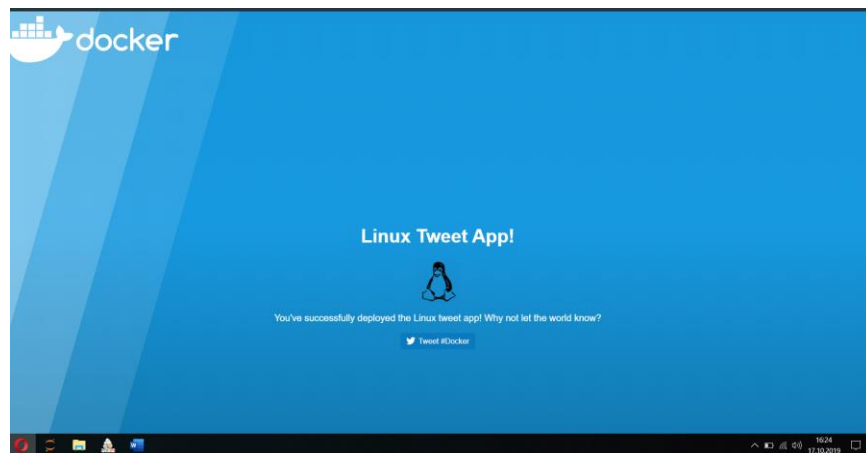
I can also run a custom app using Docker. Therefore, I need to package my application and then run it in a docker container: I will use an existing docker_file “linux_tweet_app”. So, I build docker image with the file and then I use the created image to create the tweet app container. As the container will be running a web server, I also need to specify port where I want to publish and allow traffic.

```

About
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
b0bbd1a78ca: Pull complete
Digest: sha256:77ebc94e0cec30b20f9056bac1066b09fbd049401b71850922c63fc0cc1762e
Status: Downloaded newer image for nginx:latest
--> 5a9061639d0a
Step 2/5 : COPY index.html /usr/share/nginx/html
--> 92484d43a4a3
Step 3/5 : COPY linux.png /usr/share/nginx/html
--> b86105cbd585
Step 4/5 : EXPOSE 80 443
--> Running in eb65c0944d6a
Removing intermediate container eb65c0944d6a
--> 3e96b4fc735e
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
--> Running in 09461d76d8e7
Removing intermediate container 09461d76d8e7
--> d27b8ae0c51c
Successfully built d27b8ae0c51c
Successfully tagged sknodler/linux_tweet_app:1.0
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
cf6f009197e4fa2981c319f6da807c9f616906d6df35e9567d575ac63433217d
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
```

automated



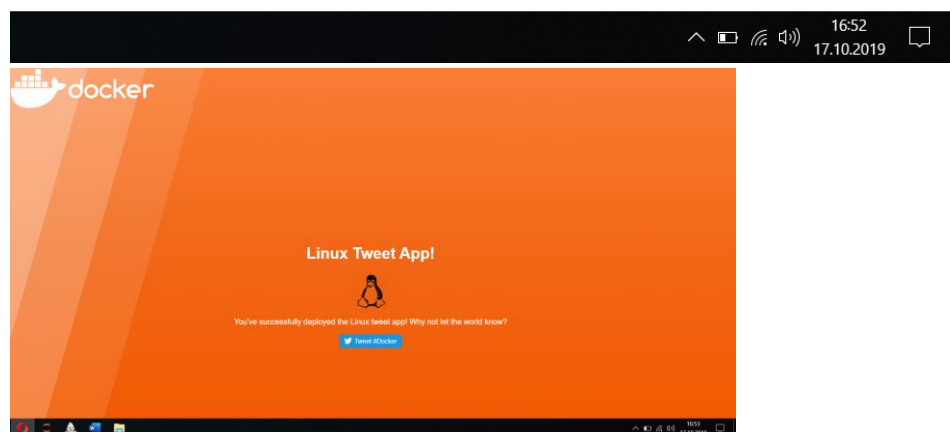


- **Task 3: Modify a Running Website**

When I work on the code of an application, it is very inconvenient to stop the container, rebuild a new image and run a new version. To avoid this, I can mount the source code directory on the local machine into the running container. Any changes I will make to the source code, can then be directly reflected in the container.

I can start a web app and mount the current directory in the container. I mount the directory on the host inside the container.

```
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> --mount type=bind,source="$(pwd)",target=/usr/share/nginx/html \
> $DOCKERID/linux_tweet_app:1.0
55d33c57e48bec5551b28ef2ca58bb45e1244ecd9101fe0ab01afe266463bb69
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$
```



So, I have just replaced the html file with another html file but I did not need to stop, rebuild, and rerun a container.

```

About
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
$ docker container rm --force linux_tweet_app
linux_tweet_app
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> --mount type=bind,source="$(pwd)",target=/usr/share/nginx/html \
> $DOCKERID/linux_tweet_app:1.0
55d33c57e48bec5551b28ef2ca58bb45e1244ecd9101fe0ab01afe266463bb69
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ cp index-new.html index.html
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker rm --force linux_tweet_app
linux_tweet_app
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
8ebcfcfcfeee3900f7425b1978387dc4bb6be68b91af1250a95d8872903c9d9b4
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker rm --force linux_tweet_app
linux_tweet_app
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$

```

Automated

16:55
17.10.2019

Now, I need to build a new version of the image to be able to use the image later again to have the latest version of the application. So, I create another image:

```

About
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
----> Running in e5e0439716c3
Removing intermediate container e5e0439716c3
----> 09befc58a5
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
----> Running in cd4e2c60adb1
Removing intermediate container cd4e2c60adb1
----> 1e1359d3dc10
Successfully built 1e1359d3dc10
Successfully tagged sknodler/linux_tweet_app:2.0
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker image ls
REPOSITORY              TAG                IMAGE ID           CREATED
sknodler/linux_tweet_app 2.0                1e1359d3dc10      4 seconds ago
o                        126MB
sknodler/linux_tweet_app 1.0                fa3d58e25043      6 minutes ago
o                        126MB
mysql                    latest             c8ee894bd2bd      16 hours ago
                        456MB
nginx                     latest             5a9061639d0a      16 hours ago
                        126MB
ubuntu                    latest             2ca708c1c9cc      4 weeks ago
                        64.2MB
alpine                    latest             961769676411      8 weeks ago
                        5.58MB
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$

```

Automated

16:58
17.10.2019

I am now able to run two web app containers side by side on two different ports.

```
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 8080:80 \
> --name old_linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
ce9693256afd91f1354d63c4b91f0a2300820ee3a924975c81f3163ddd259b05
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:2.0
```

automated

f t

17:00
17.10.2019

I can list the containers that are associated to my docker id and stored in my docker hosts local repository. So, these images are not yet available to any other users. However, I can push them to a private or public repository, such as the default public repository Docker Hub.

```
ABOUT
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker image push $DOCKERID/linux_tweet_app:1.0
The push refers to repository [docker.io/sknodler/linux_tweet_app]
64a3b09851aa: Pushed
72198baa5e84: Pushed
cf2436e84ea8: Mounted from library/nginx
ed4a4820ee08: Mounted from library/nginx
b67d19e65ef6: Mounted from library/nginx
1.0: digest: sha256:2bf676ed3c31d0681733f41ea76c4a0dcacf59a7c4f96c30db2f55f971d7
d04fd size: 1363
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$ docker image push $DOCKERID/linux_tweet_app:2.0
The push refers to repository [docker.io/sknodler/linux_tweet_app]
eedf8a53bc88: Pushed
fc80a04baf04: Pushed
cf2436e84ea8: Layer already exists
ed4a4820ee08: Layer already exists
b67d19e65ef6: Layer already exists
2.0: digest: sha256:250f5a4612f32965c94b6d335fadf6ee4539641a983bac123d3830b45e0
90795 size: 1363
[node1] (local) root@192.168.0.53 ~/linux_tweet_app
$
```

automated

f t

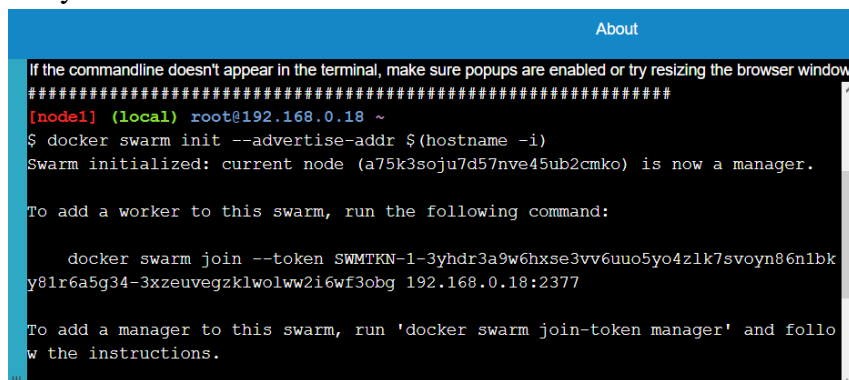
17:06
17.10.2019

Sub-Module 2 - Swarm Stack Introduction

<https://training.play-with-docker.com/swarm-stack-intro/#>

In this part of the lab, I will deploy a stack (multi services application) against a Swarm using a docker compose file. To do so, I will work with two terminals, namely the swarm manager node and the swarm worker.

- Init your swarm



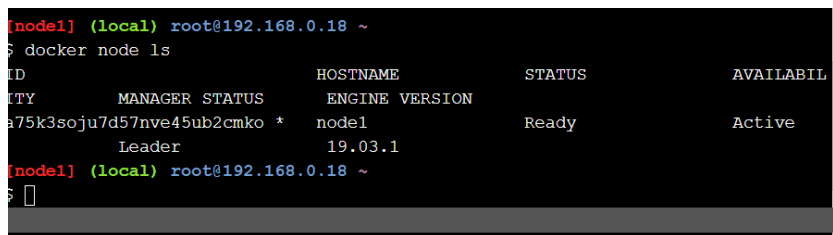
```
#####
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
#####
[node1] (local) root@192.168.0.18 ~
$ docker swarm init --advertise-addr $(hostname -i)
Swarm initialized: current node (a75k3soju7d57nve45ub2cmko) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3yhdr3a9w6hxse3vv6uuo5yo4zlk7svoy86n1bk
y8lr6a5g34-3xzeuvegzklw0lww2i6wf3obg 192.168.0.18:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

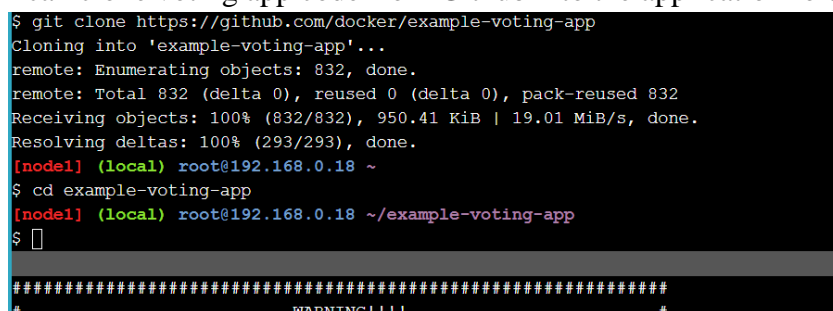
- Show members of swarm



```
[node1] (local) root@192.168.0.18 ~
$ docker node ls
ID                                HOSTNAME          STATUS          AVAILABILITY
a75k3soju7d57nve45ub2cmko * node1             Ready           Active
Leader
19.03.1
```

- Clone the voting-app

I can clone voting app code from Github into the application folder



```
$ git clone https://github.com/docker/example-voting-app
Cloning into 'example-voting-app'...
remote: Enumerating objects: 832, done.
remote: Total 832 (delta 0), reused 0 (delta 0), pack-reused 832
Receiving objects: 100% (832/832), 950.41 KiB | 19.01 MiB/s, done.
Resolving deltas: 100% (293/293), done.
[node1] (local) root@192.168.0.18 ~
$ cd example-voting-app
[node1] (local) root@192.168.0.18 ~/example-voting-app
$
#####
# WARNING!!! #
```

- Deploy a stack

A stack is a group of services that I can deploy together. I used the docker-stack.yml file to deploy the voting app as a stack. Next, I listed the deployed services as well as the tasks of the vote service.

```
If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.
[node1] (local) root@192.168.0.18 ~/example-voting-app
$ docker service ps voting_stack_vote
```

ID	NODE	NAME	DESIRED STATE	CURRENT STATE	ERROR
PORTS					
ftxmtlppxa3		voting_stack_vote.1	dockersamples/examplevotingapp_vote:b		
efore node1		Running	Running 4 seconds ago		
kbvq8lhn5ctu		voting_stack_vote.2	dockersamples/examplevotingapp_vote:b		
efore node1		Running	Running 4 seconds ago		

STAGE 2 - Digging Deeper

<https://training.play-with-docker.com/dev-stage2/>

Sub-Module 1 - Windows Container Setup

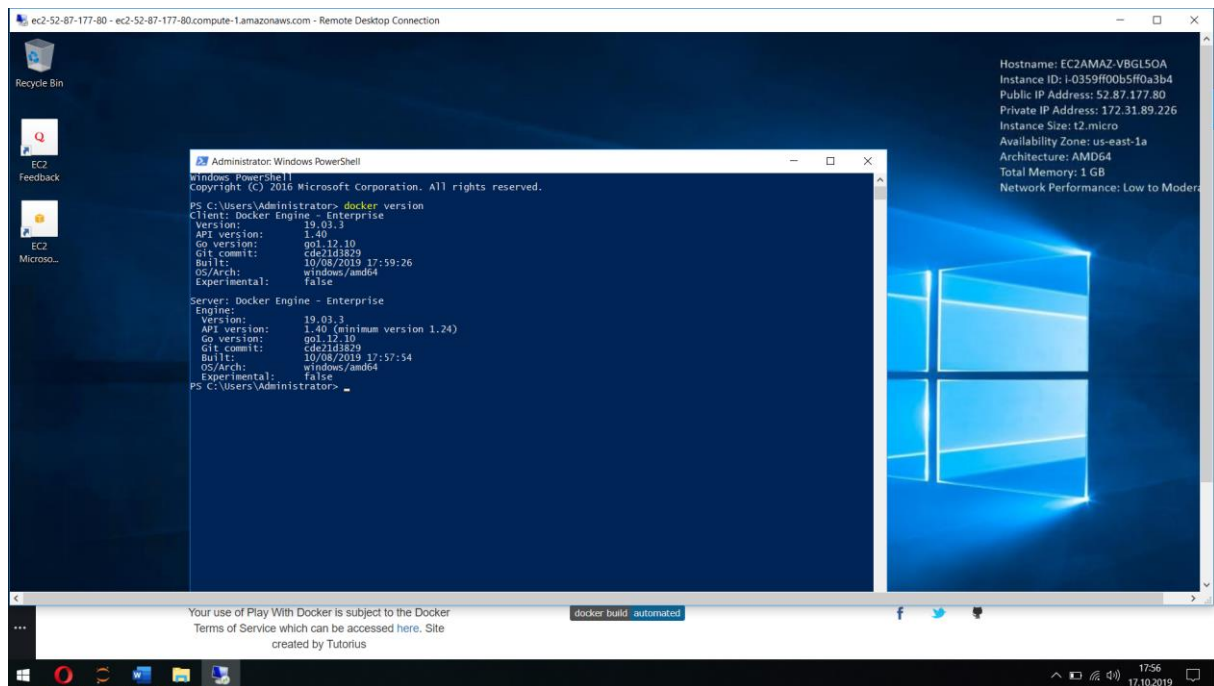
<https://training.play-with-docker.com/windows-containers-setup/>

- Out of the four setup environments, select the following one to set-up Windows Container
 - Windows Server 2016 on AWS - <https://training.play-with-docker.com/windows-containers-setup/#aws>

I created a VM on AWS, based on a Microsoft Windows Server 2016 Base with Containers AMI and accessed the VM via remote desktop. Next, I opened powershell with admin rights and ran:

```
Install-Module -Name DockerMsftProvider -Force
Install-Package -Name docker -ProviderName DockerMsftProvider -Force
Restart-Computer -Force
```

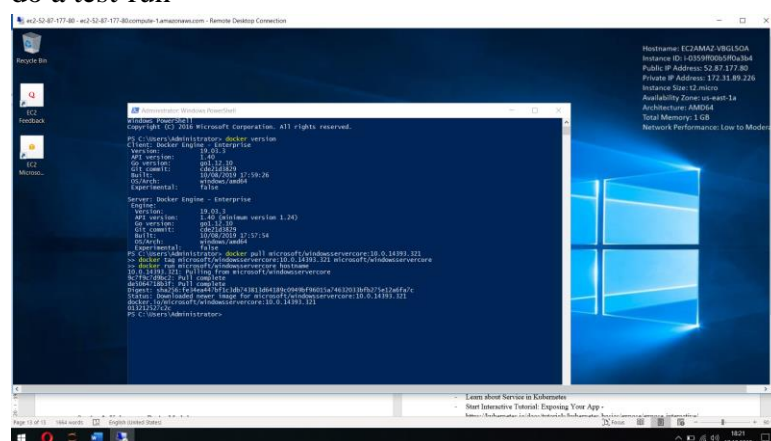
To install docker and restart the computer.



Sub-Module 2 - Windows Container Basics

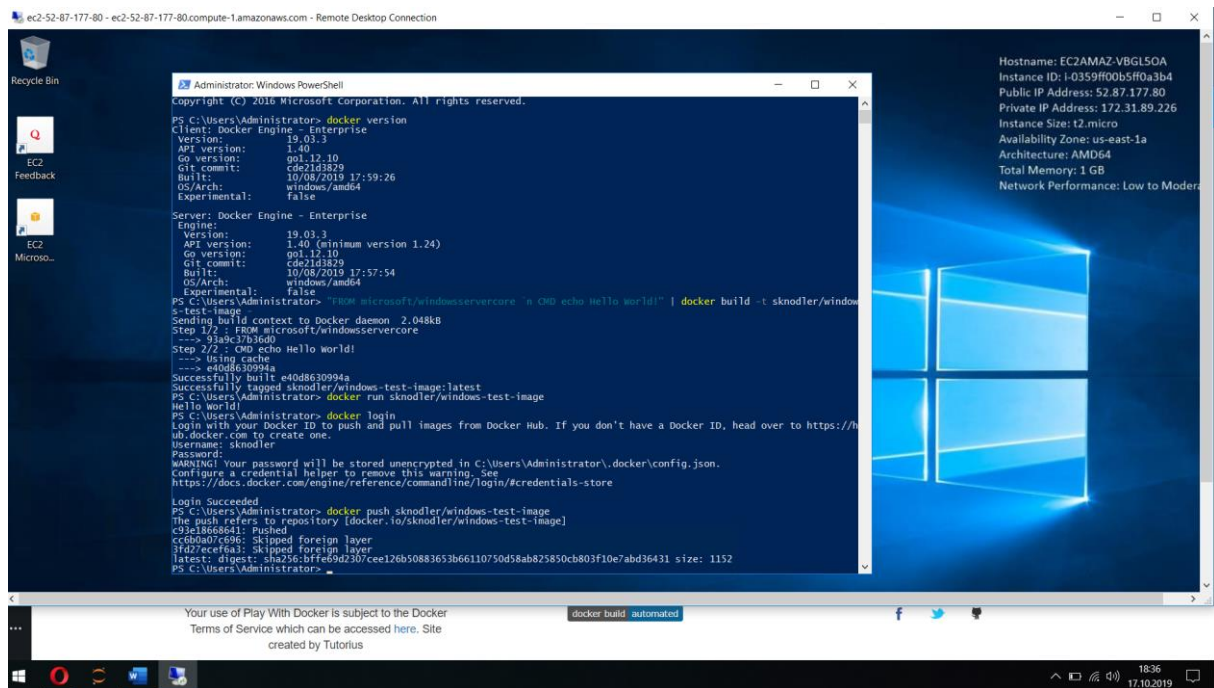
<https://training.play-with-docker.com/windows-containers-basics/>

- Follow the on-screen instructions and complete all the two parts with the respective steps mentioned below
 - Getting Started with Windows Containers
 - make sure the Docker installation is working
 - pull a base image that's compatible with the evaluation build
 - re-tag the base image
 - do a test-run



- Building and pushing Windows container images
 - create an image on the fly in PowerShell
 - test the image
 - push the image

I created the image, logged in to my docker account and pushed the image to store it online.



The screenshot shows a remote desktop connection to an Amazon EC2 instance. In the center is a Windows PowerShell terminal window with Administrator privileges. The terminal output shows the following commands and results:

```
PS C:\Users\Administrator> docker version
Client: Docker Engine - Enterprise
Version: 19.03.3
API version: 1.40
Go version: go1.12.10
Git commit: c9e21d3829
Built: 10/08/2019 17:59:26
OS/Arch: windows/amd64
Experimental: false

Server: Docker Engine - Enterprise
Engine:
Version: 19.03.3
API version: 1.40 (minimum version 1.24)
Go version: go1.12.10
Git commit: c9e21d3829
Built: 10/08/2019 17:57:54
OS/Arch: windows/amd64
Experimental: false

PS C:\Users\Administrator> "from microsoft/windowsservercore 'n CMD echo Hello world!' | docker build -t sknodler/windows-test-image
Sending build context to Docker daemon 2.048kb
Step 1/2 : FROM microsoft/windowsservercore
--> 93a9c3b36d0
Step 2/2 : CMD echo Hello world!
--> Using cache
--> e40d8630994a
Successfully built e40d8630994a
Successfully tagged sknodler/windows-test-image:latest
PS C:\Users\Administrator> docker run sknodler/windows-test-image
Hello world!

PS C:\Users\Administrator> docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: sknodler
Password:
WARNING! Your password will be stored unencrypted in C:\Users\Administrator\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
PS C:\Users\Administrator> docker push sknodler/windows-test-image
The push refers to repository [docker.io/sknodler/windows-test-image]
c93e186e8641: Pushed
cc60a07c906: Skipped foreign layer
3fd27ecf6a3: Skipped foreign layer
latest digest: sha256:bff6e9d2307cee126b50883653b66110750d58ab825850cb803f10e7abd36431 size: 1152
PS C:\Users\Administrator>
```

On the right side of the terminal window, a system information panel is visible, showing details such as Hostname (EC2AMAZ-VBGL50A), Instance ID (i-0359f00b5f0a3b4), Public IP Address (52.87.177.80), Private IP Address (172.31.89.226), Instance Size (t2.micro), Availability Zone (us-east-1a), Architecture (AMD64), Total Memory (1 GB), and Network Performance (Low to Moderate).

At the bottom of the screen, a taskbar shows the Windows Start button and several application icons. A notification area at the bottom right displays the time as 18:36 on 17.10.2019.

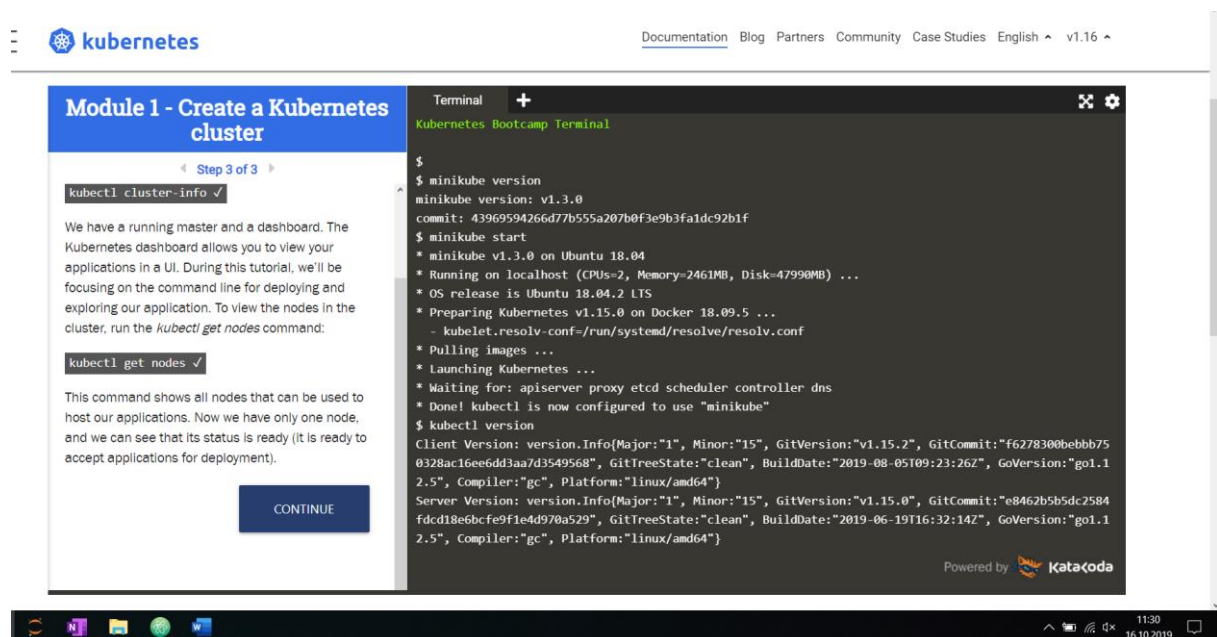
1. Kubernetes

Section 1: Getting Started

Go to <https://kubernetes.io/docs/tutorials/kubernetes-basics/#kubernetes-basics>

Complete all the six Kubernetes Basics Modules

Kubernetes supports to manage containerized applications. It makes sure that the applications run where and when they need to. I can deploy containerized applications to a cluster without attaching them individually to individual machines. Kubernetes automates the distribution and scheduling of the used containers within my cluster. Therefore, the master coordinates the cluster and the nodes are the workers that run applications. That means that the master is a coordinator and the node is either a VM or a physical computer. Each node runs a Kubelet which is an agent that manages the node and communicates to the Kubernetes master. When I deploy an application on Kubernetes, I advise the master to start an application container. So, the master will schedule the container on a certain node

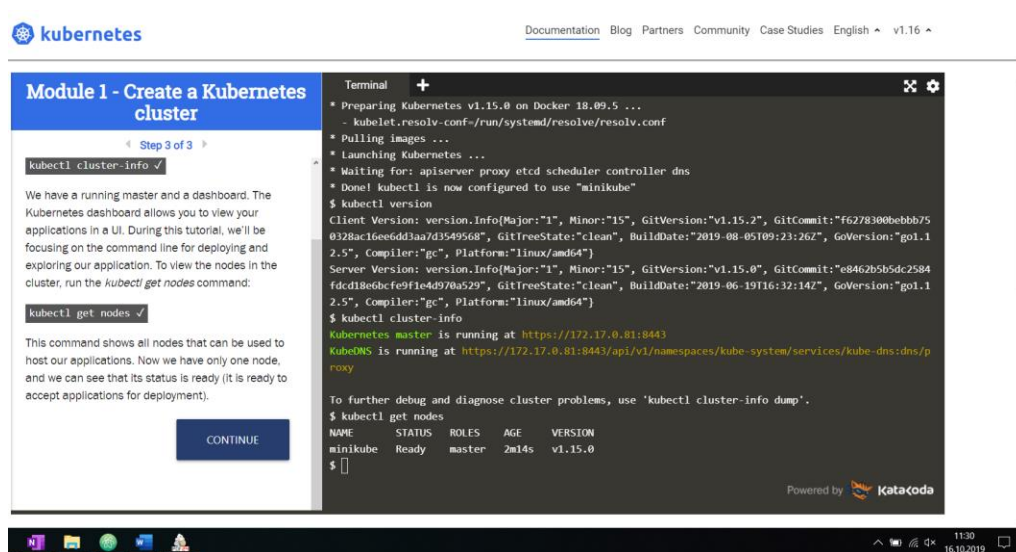


First, I checked that minikube is properly installed, by running the *minikube version* command. Minikube is a lightweight Kubernetes implementation which creates a VM on my local machine and deploys a simple cluster containing only one node.

a. Create a Kubernetes Cluster - Using Minikube to Create a Cluster Interactive Tutorial

I created the cluster by running *minikube start*. So, I managed to run a Kubernetes cluster. Minikube started a virtual machine with the specified that are presented in the online terminal. The Kubernetes cluster is now running in that VM. To deploy an app,

I first need to identify available nodes that I can use to host an application. I run the `kubectl get nodes` command in order to view the nodes in the cluster. It shows only one node, and we can see that its status is ready to accept applications for deployment.




b. Deploy an App - Using kubectl to Create a Deployment Interactive Tutorial

Once the Kubernetes cluster is running, I can deploy my containerized applications on top of it. Therefore, I need to create a Kubernetes Deployment configuration that instructs Kubernetes how to create and update application instances. The Kubernetes command line interface Kubectl allows me to create and manage a Deployment.

Documentation Blog Partners Community Case Studies English ^ v1.16 ^

```
terminal
Kubernetes Bootcamp Terminal

$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.....
Kubernetes Started
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.2", GitCommit:"f6278300bebbb75
0328ac16ee6dd3aa7d3549568", GitTreeState:"clean", BuildDate:"2019-08-05T09:23:26Z", GoVersion:"go1.1
2.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.0", GitCommit:"e8462b5b5dc2584
fdcd18e6bcfe9f1e4d970a529", GitTreeState:"clean", BuildDate:"2019-06-19T16:32:14Z", GoVersion:"go1.1
2.5", Compiler:"gc", Platform:"linux/amd64"}
$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready     master   68s   v1.15.0
$
```

Powered by  Katacoda


19:00
17.10.2019

I listed my node to see that I have one running. Next, I deployed my first app and listed my deployments

Documentation Blog Partners Community Case Studies English ^ v1.16 ^

```
terminal
Kubernetes Bootcamp Terminal

$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.....
Kubernetes Started
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.2", GitCommit:"f6278300bebbb75
0328ac16ee6dd3aa7d3549568", GitTreeState:"clean", BuildDate:"2019-08-05T09:23:26Z", GoVersion:"go1.1
2.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.0", GitCommit:"e8462b5b5dc2584
fdcd18e6bcfe9f1e4d970a529", GitTreeState:"clean", BuildDate:"2019-06-19T16:32:14Z", GoVersion:"go1.1
2.5", Compiler:"gc", Platform:"linux/amd64"}
$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
minikube      Ready     master   68s   v1.15.0
$ kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1
deployment.apps/kubernetes-bootcamp created
$ kubectl get deployments
NAME          READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp  1/1      1             1            7s
$
```

Powered by  Katacoda

19:00
17.10.2019

Now, the instance is running inside the docker container on my node.

10/19/2019

But it is not visible from outside yet. Consequently, I create a proxy that can forward the communication and make it accessible via port 8001

The first terminal screenshot shows the following commands and output:

```
minikube create deployment kubernetescamp --image=gcr.io/google-samples/kubernetescamp:v1
Error from server (AlreadyExists): deployments.apps "kubernetescamp" already exists
$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
kubernetescamp 1/1     1            1           3m34s
$ curl http://localhost:8001/version
{
  "major": "1",
  "minor": "15",
  "gitVersion": "v1.15.0",
  "gitCommit": "e8462b5b5dc2584fcd18e6bce9f1e4d970a529",
  "gitTreeState": "clean",
  "buildDate": "2019-06-19T16:32:14Z",
  "goVersion": "go1.12.5",
  "compiler": "gc",
  "platform": "linux/amd64"
}
$ export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetescamp-75bccb7d87-wsbzn
```

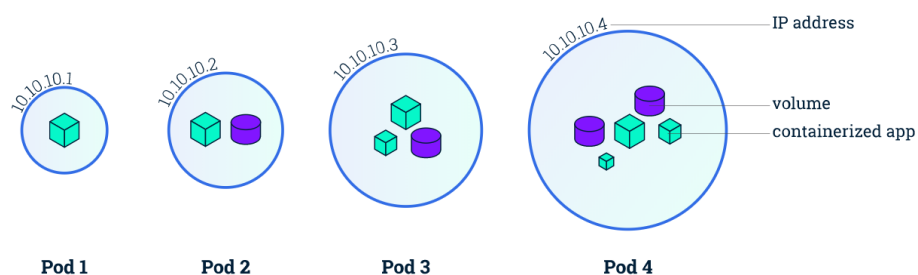
The second terminal screenshot shows the following commands and output:

```
$ echo -e "\n\n\nStarting Proxy. After starting it will not output a response. Please click the first Terminal Tab\n\n"

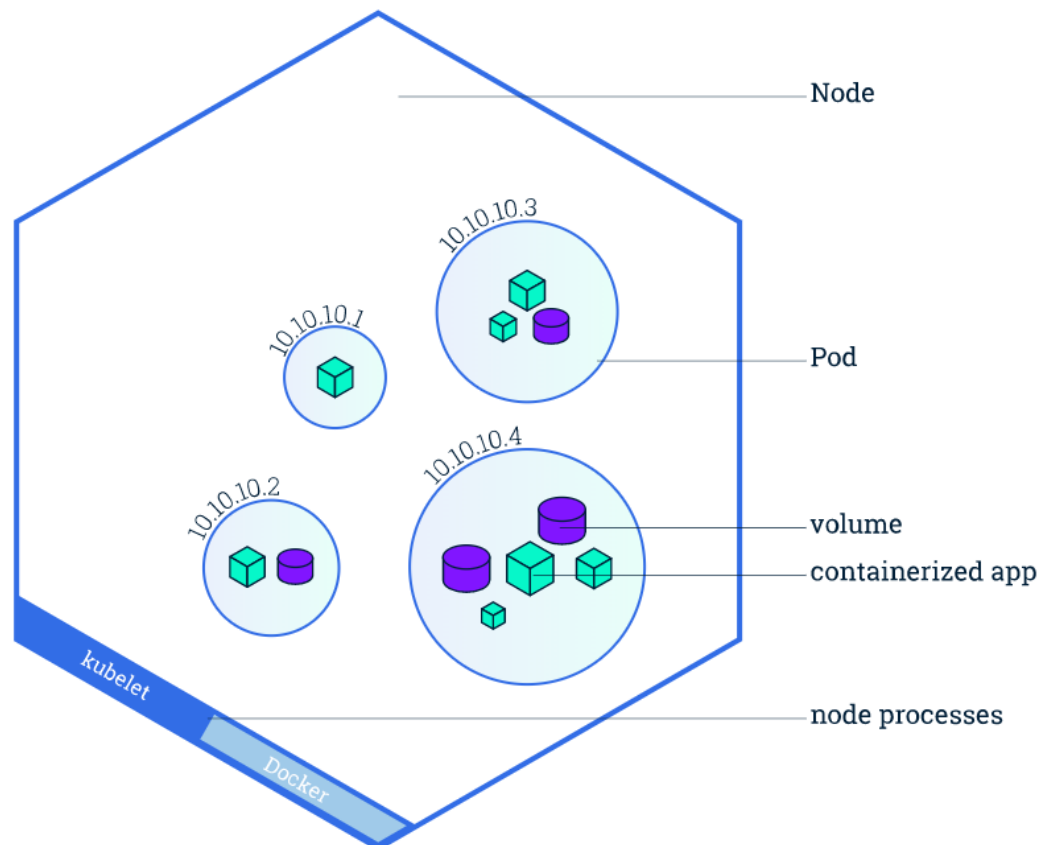
Starting Proxy. After starting it will not output a response. Please click the first Terminal Tab

$ kubectl proxy
Starting to serve on 127.0.0.1:8001
^C
$ ^C
$
```

To host my application instance, Kubernetes created a Pod. A Pod is a Kubernetes abstraction that stands for group of one or more application containers as well as some shared resources for those containers.



Node overview




c. Explore Your App - Viewing Pods and Nodes Interactive Tutorial

I started by showing the app in the terminal, viewed container logs and executed command on one container. Therefore, I started a bash session in the Pod's container and ran a NodeJS application.

DocumentationBlogPartnersCommunityCase StudiesEnglish ^ v1.16 ^


```
$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdc-bd-5fkw1  1/1      Running   0           17s

$ kubectl describe pods
Name:          kubernetes-bootcamp-5b48cfdc-bd-5fkw1
Namespace:    default
Priority:      0
Node:         minikube/172.17.0.12
Start Time:   Thu, 17 Oct 2019 23:18:34 +0000
Labels:       pod-template-hash=5b48cfdc-bd
              run=kubernetes-bootcamp
Annotations:  <none>
Status:       Running
IP:           172.18.0.2
Controlled By: ReplicaSet/kubernetes-bootcamp-5b48cfdc-bd
Containers:
  kubernetes-bootcamp:
    Container ID:  docker://5c73237354369ae7548bf4f8dc35b33236cab4a41117357a5b1c50010a162396
    Image:         gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:      docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:         8080/TCP
```

Powered by  **Katacoda**

DocumentationBlogPartnersCommunityCase StudiesEnglish ^ v1.16 ^

```
var podName= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v-1\n");
  console.log("Running On: ",host, " | Total Requests: ",++requests," | App Uptime: ", (new Date() - startime)/1000 , "seconds", " | Log Time: ",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
  startTime = new Date();
  host = process.env.HOSTNAME;
  console.log ("Kubernetes Bootcamp App Started At:",startTime, " | Running On: ",host, "\n");
});
root@kubernetes-bootcamp-5b48cfdc-bd-5fkw1:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdc-bd-5fkw1 | v-1
root@kubernetes-bootcamp-5b48cfdc-bd-5fkw1:~#
```

Powered by  **Katacoda**

d. Expose Your App Publicly - Using a Service to Expose Your App Interactive Tutorial


To expose my app publicly, I first looked for existing pods and services. Only the default service Kubernetes is running. I am creating a new service and expose it to external traffic

[Documentation](#) [Blog](#) [Partners](#) [Community](#) [Case Studies](#) [English](#) ^ v1.16 ^

```
$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.kubectl get pods
.....
Kubernetes Started
$ kubectl get pods
No resources found.
$ kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
kubernetes-bootcamp-5b48cfdbc-d-5q9s6  0/1     ContainerCreating   0          8s
$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP   18s
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service/kubernetes-bootcamp exposed
$ kubectl get services
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP   10.96.0.1    <none>        443/TCP          90s
kubernetes-bootcamp  NodePort    10.107.95.235 <none>        8080:31008/TCP   4s
$
```

Powered by  Katacoda

```
Documentation Blog Partners Community Case Studies English ^ v1.16 ^  
$ kubectl describe services/kubernetes-bootcamp  
Name: kubernetes-bootcamp  
Namespace: default  
Labels: run=kubernetes-bootcamp  
Annotations: <none>  
Selector: run=kubernetes-bootcamp  
Type: NodePort  
IP: 10.107.95.235  
Port: <unset> 8080/TCP  
TargetPort: 8080/TCP  
NodePort: <unset> 31008/TCP  
Endpoints: 172.18.0.3:8080  
Session Affinity: None  
External Traffic Policy: Cluster  
Events: <none>  
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')  
$ echo NODE_PORT=$NODE_PORT  
NODE_PORT=31008  
$ curl $(minikube ip):$NODE_PORT  
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcdb-5q9s6 | v=1  
$
```

Powered by  Katacoda



As you can see, I created an environment variable that has the value of the Node port assigned and made sure that the app is exposed to the outside. The Deployment created automatically a label for the pod. I got the name of the Pod and stored it in the pod environment variable. Next, I applied a new label and made sure the label is attached to the pod.

```

volumes:
  default-token-vdjhh:
    Type: Secret (a volume populated by a Secret)
    SecretName: default-token-vdjhh
    Optional: false
QoS Class:           BestEffort
Node-Selectors:      <none>
Tolerations:         node.kubernetes.io/not-ready:NoExecute for 300s
                     node.kubernetes.io/unreachable:NoExecute for 300s


Events:
  Type            Reason      Age   From          Message
  ----            -
  Normal          Scheduled   5m20s default-scheduler Successfully assigned default/kubernetes-bootcamp-5b48cfdbcdbd-5q9s6 to minikube
  Normal          Pulled      5m11s kubelet, minikube Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal          Created     5m11s kubelet, minikube Created container kubernetes-bootcamp
  Normal          Started     5m11s kubelet, minikube Started container kubernetes-bootcamp

$ kubectl get pods -l app=v1
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdbcdbd-5q9s6  1/1     Running   0           5m24s
$

```

Afterwards, I deleted the service again and made sure it's not accessible anymore

```
Events:
  Type    Reason      Age   From              Message
  ----    -
  Normal  Scheduled   5m20s default-scheduler Successfully assigned default/kubernetes-bootcamp-5b48cfdc
8cfdcdbd-5q9s6 to minikube
  Normal  Pulled      5m11s kubelet, minikube Container image "gcr.io/google-samples/kubernetes-boo
tcamp:v1" already present on machine
  Normal  Created     5m11s kubelet, minikube Created container kubernetes-bootcamp
  Normal  Started     5m11s kubelet, minikube Started container kubernetes-bootcamp
$ kubectl get pods -l app=v1
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdcdbd-5q9s6 1/1     Running   0          5m24s
$ kubectl delete service -l run=kubernetes-bootcamp
service "kubernetes-bootcamp" deleted
$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes ClusterIP  10.96.0.1     <none>        443/TCP      8m14s
$ curl $(minikube ip):$NODE_PORT
curl: (7) Failed to connect to 172.17.0.55 port 31008: Connection refused
$ kubectl exec -ti $POD_NAME curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcdbd-5q9s6 | v=1
$
```

Powered by  Katacoda

e. Scale Your App - Running Multiple Instances of Your App Interactive Tutorial

In case of increased traffic, we need to be able to scale the App. Scaling is achieved by increasing and decreasing the number of replicas in a deployment.

First, I list the deployments and then increase the number of replicas to 4.

10/19/2019

```
$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.kubectl get deployments
.....
Kubernetes Started
$ kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp 0/1      0             0            3s
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.extensions/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
kubernetes-bootcamp 4/4      4             4            21s
$
```

I can also display the current number of pods and list the deployment logs. Here, I can see the change I have just triggered.

```

Documentation Blog Partners Community Case Studies English ^ v1.16 ^

$ kubectl get deployments
NAME                READY  UP-TO-DATE  AVAILABLE  AGE
kubernetes-bootcamp 4/4    4           4          21s


$ kubectl get pods -o wide
NAME                READY  STATUS  RESTARTS  AGE  IP            NODE             NO
MINATED NODE  READINESS GATES
kubernetes-bootcamp-5b48cfdcdb-fpt2p 1/1    Running  0         40s  172.18.0.4    minikube        <n
one>          <none>
kubernetes-bootcamp-5b48cfdcdb-hdf1b 1/1    Running  0         30s  172.18.0.7    minikube        <n
one>          <none>
kubernetes-bootcamp-5b48cfdcdb-mtx9g 1/1    Running  0         30s  172.18.0.8    minikube        <n
one>          <none>
kubernetes-bootcamp-5b48cfdcdb-qtns9 1/1    Running  0         30s  172.18.0.6    minikube        <n
one>          <none>

$ kubectl describe deployments/kubernetes-bootcamp
Name:                kubernetes-bootcamp
Namespace:           default
CreationTimestamp:    Sat, 19 Oct 2019 14:18:12 +0000
Labels:               run=kubernetes-bootcamp
Annotations:          deployment.kubernetes.io/revision: 1
Selector:             run=kubernetes-bootcamp

```

10/19/2019


```
Events
-----
Type      Reason      Age   From          Message
-----
Normal    ScalingReplicaSet  65s   deployment-controller  Scaled up replica set kubernet
5b48cfdcbb to 1
Normal    ScalingReplicaSet  54s   deployment-controller  Scaled up replica set kubernet
5b48cfdcbb to 4
$
```

Powered by  KataCoda




Next, I can make use of load balancing. Therefore, I identify the exposed IP and Port and create an environment variable. When communicating through the exposed ip and port, we will get an answers from different pods. This shows that load balancing is working

```
$ kubectl describe services/kubernetes-bootcamp
Name:      kubernetes-bootcamp
Namespace: default
Labels:    run=kubernetes-bootcamp
Annotations: <none>
Selector:  run=kubernetes-bootcamp
Type:      NodePort
IP:        10.107.215.157
Port:      <unset> 8080/TCP
TargetPort: 8080/TCP
NodePort:   <unset> 31407/TCP
Endpoints:  172.18.0.4:8080,172.18.0.6:8080,172.18.0.7:8080 + 1 more...
Session Affinity: None
External Traffic Policy: Cluster
Events:     <none>
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=31407
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbb-hdf1b | v=1
```

Powered by  KataCoda





```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdbc-dhflb | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdbc-qtns9 | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdbc-mtx9g | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdbc-fpt2p | v=1
$
```

Powered by  KataCoda

The same way we scaled up, we can also scale down:

```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.extensions/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2/2     2            2           8m58s
$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-5b48cfdbc-fpt2p 1/1     Running   0          8m52s  172.18.0.4    minikub
e <none>          <none>
kubernetes-bootcamp-5b48cfdbc-dhflb 1/1     Running   0          8m42s  172.18.0.7    minikub
e <none>          <none>
kubernetes-bootcamp-5b48cfdbc-mtx9g 1/1     Terminating 0      8m42s  172.18.0.8    minikub
e <none>          <none>
kubernetes-bootcamp-5b48cfdbc-qtns9 1/1     Terminating 0      8m42s  172.18.0.6    minikub
e <none>          <none>
$
```


Powered by  KataCoda

f. Update Your App - Performing a Rolling Update Interactive Tutorial

Users expect to be able to use the application all the time and developers want to update the app whenever needed. To avoid downtime, Kubernetes uses Rolling updates. It allows to update the application without downtime. Multiple pods can be updated incrementally and we make sure that always enough pods are available.


Therefore, I can list the running pods and the installed current images on each of them

```
$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.kubectl get deployments
.....
Kubernetes Started
$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 0/4     0            0           2s
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdbc-d8sddw 1/1     Running   0          21s
kubernetes-bootcamp-5b48cfdbc-q8xvz  1/1     Running   0          21s
kubernetes-bootcamp-5b48cfdbc-t992j  1/1     Running   0          21s
kubernetes-bootcamp-5b48cfdbc-v7jk7  1/1     Running   0          21s
$ kubectl describe pods
Name:          kubernetes-bootcamp-5b48cfdbc-d8sddw
Namespace:     default
Priority:       0
Node:          minikube/172.17.0.29
Start Time:    Sat, 19 Oct 2019 14:33:49 +0000
Labels:        pod-template-hash=5b48cfdbc
```


Powered by  KataCoda

I can initiate a rolling update by set the image to another version. I can also verify the update by running a rollout status command

```
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdbc-d8sddw 1/1     Terminating 0          2m
kubernetes-bootcamp-5b48cfdbc-q8xvz  1/1     Terminating 0          2m
kubernetes-bootcamp-5b48cfdbc-t992j  1/1     Terminating 0          2m
kubernetes-bootcamp-5b48cfdbc-v7jk7  1/1     Terminating 0          2m
kubernetes-bootcamp-cfc74666-9mn7v  1/1     Running       0          9s
kubernetes-bootcamp-cfc74666-b2jwd  1/1     Running       0          7s
kubernetes-bootcamp-cfc74666-dzkbq  1/1     Running       0          7s
kubernetes-bootcamp-cfc74666-trd7p  1/1     Running       0          9s
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-cfc74666-9mn7v  1/1     Running   0          2m10s
kubernetes-bootcamp-cfc74666-b2jwd  1/1     Running   0          2m8s
kubernetes-bootcamp-cfc74666-dzkbq  1/1     Running   0          2m8s
kubernetes-bootcamp-cfc74666-trd7p  1/1     Running   0          2m10s
$
```

Powered by  KataCoda

Documentation Blog Partners Community Case Studies English ^ v1.16 ^

```
Session Affinity:      None
External Traffic Policy: Cluster
Events:                <none>
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=32739
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-cfc74666-dzkbq | v=2
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
$ kubectl describe pods
Name:          kubernetes-bootcamp-cfc74666-9mn7v
Namespace:     default
Priority:       0
Powered by  Katacoda
```


10:39 19.10.2019 ENG

I can also rollback an update. To demonstrate this I update to v10 and rollback the update to the stable v2

```
Documentation Blog Partners Community Case Studies English ^ v1.16 ^

Type: Secret (a volume populated by a Secret)
SecretName: default-token-lxbr4
Optional: false
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
              node.kubernetes.io/unreachable:NoExecute for 300s


Events:
  Type     Reason      Age      From      Message
  ----     -
Normal    Scheduled   17s      default-scheduler Successfully assigned default/kubernetes
-kubernetes-bootcamp-547469f5dd-jqc52 to minikube
Normal    Pulling     16s      kubelet, minikube Pulling image "gcr.io/google-samples/kub
ernetes-bootcamp:v10"
Warning   Failed      15s      kubelet, minikube Failed to pull image "gcr.io/google-samp
les/kubernetes-bootcamp:v10": rpc error: code = Unknown desc = Error response from daemon: manifest
for gcr.io/google-samples/kubernetes-bootcamp:v10 not found
Warning   Failed      15s      kubelet, minikube Error: ErrImagePull
Normal    BackOff     14s (x2 over 15s) kubelet, minikube Back-off pulling image "gcr.io/google-sa
mples/kubernetes-bootcamp:v10"
Warning   Failed      14s (x2 over 15s) kubelet, minikube Error: ImagePullBackOff

Powered by  KataCoda
```

As you can see something went wrong when updating to v10. So, I can roll back the update:


```
Documentation Blog Partners Community Case Studies English ^ v1.16 ^

Normal Started 4m55s kubelet, minikube Started container kubernetes-bootcamp
$ kubectl rollout undo deployments/kubernetes-bootcamp
deployment.extensions/kubernetes-bootcamp rolled back
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-547469f5dd-jqc52 0/1     Terminating 0           24s
kubernetes-bootcamp-cfc74666-9f7f9    1/1     Running    0           5s
kubernetes-bootcamp-cfc74666-9mn7v    1/1     Running    0           5m3s
kubernetes-bootcamp-cfc74666-b2jwd    1/1     Terminating 0           5m1s
kubernetes-bootcamp-cfc74666-dzkbq    1/1     Running    0           5m1s
kubernetes-bootcamp-cfc74666-trd7p    1/1     Running    0           5m3s
$ kubectl describe pods
Name:          kubernetes-bootcamp-cfc74666-9f7f9
Namespace:    default
Priority:      0
Node:         minikube/172.17.0.29
Start Time:   Sat, 19 Oct 2019 14:40:38 +0000
Labels:       pod-template-hash=cfc74666
              run=kubernetes-bootcamp
Annotations:   <none>
Status:       Running

Powered by  KataCoda
```

10/19/2019

```
node.kubernetes.io/unreachable:NoExecute 10m 500s
Events:
  Type    Reason            Age   From                  Message
  ----    -
  Normal  Scheduled         5m10s default-scheduler     Successfully assigned default/kubernetes-bootcamp-cfc
74666-trd7p to minikube
  Normal  Pulled            5m9s  kubelet, minikube     Container image "jocatalin/kubernetes-bootcamp:v2" al
ready present on machine
  Normal  Created           5m9s  kubelet, minikube     Created container kubernetes-bootcamp
  Normal  Started           5m9s  kubelet, minikube     Started container kubernetes-bootcamp
$
```

Powered by  KataCoda