Containers

For this assignment I will be installing Docker and Kubernetes on both servers. Docker is a tool which allows the deployment of applications inside Linux containers. Kubernetes is similar in how it is also a tool for managing containerized workloads. After installing both Docker and Kubernetes, I will also be installing Minikube.

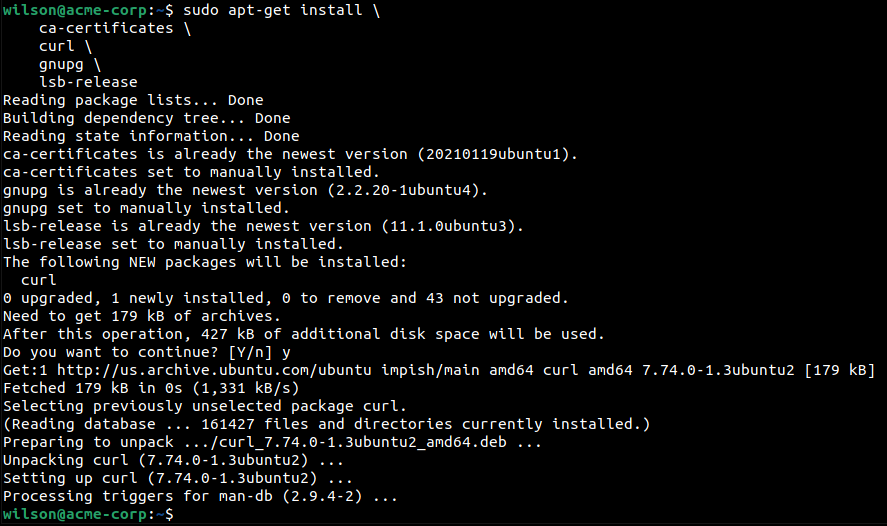
Docker on Ubuntu

The first thing we need to do is make sure our list of packages from our repositories and PPA’s are up to date. To perform this update, run the following command.

Text

Description automatically generatedsudo apt-get update

Next, we also need to make sure certs, curl and all other related packages are up to date and installed. This will be required to use Docker. To perform this, use the follow command. After running the command, type y and press enter to continue.

sudo apt-get install \ ca-certificates \ curl \ gnupg \ lsb-release

Now we need to add dockers official GPG key. GPG is a system similar to PGP, used to verify the integrity of a software package before installing it. This is required to download the official docker packages. The curl command will allow us to obtain the GPG key from the server. To perform this, run the following command.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg



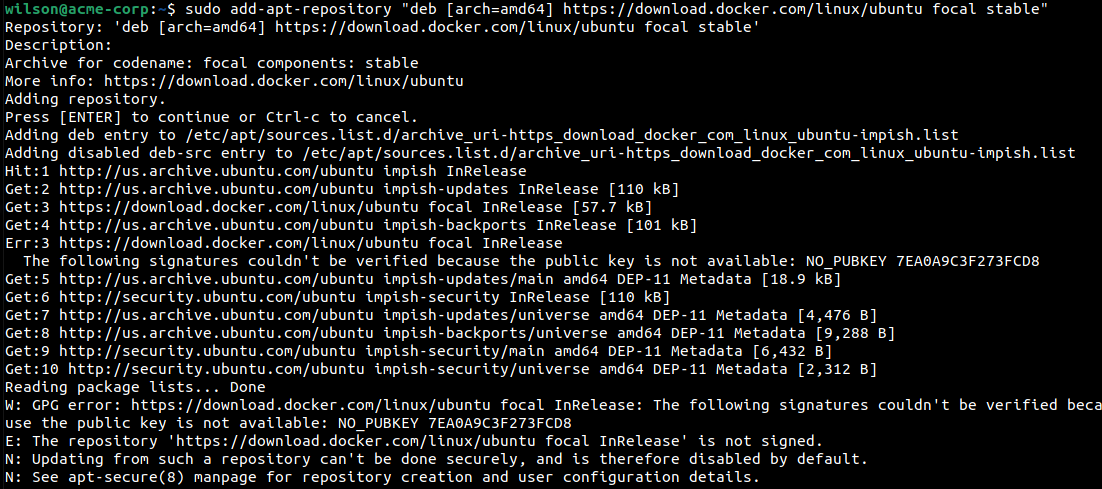
Moving on, we now need to add the Docker repository to our APT sources. Performing this command will also update our package database with the latest Docker packages. To add the repository, run the following command.

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

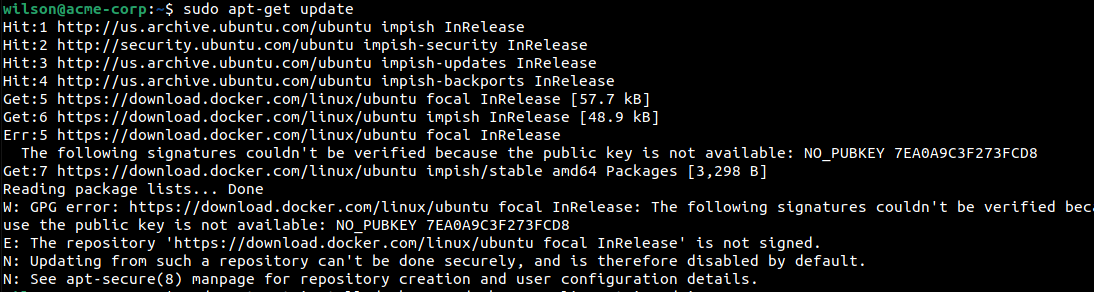
$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

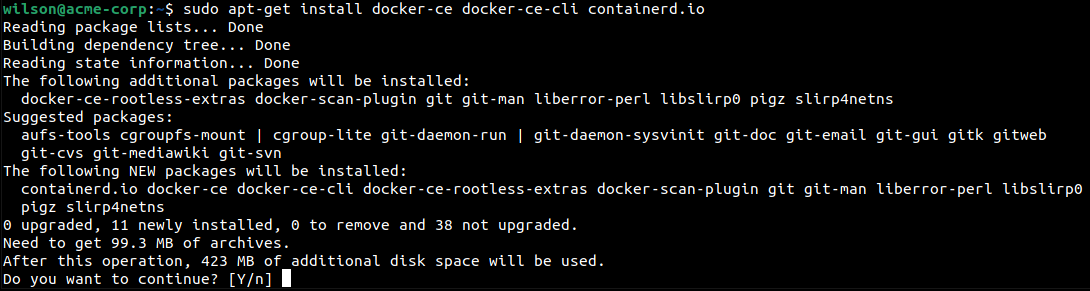
A message should then appear prompting if you would like to continue. Press enter to proceed.



Now that we have the repositories set up, we can install the latest Docker engine. Update the apt package with the first command then you can now run the second command which will install the Docker engine.

sudo apt-get update

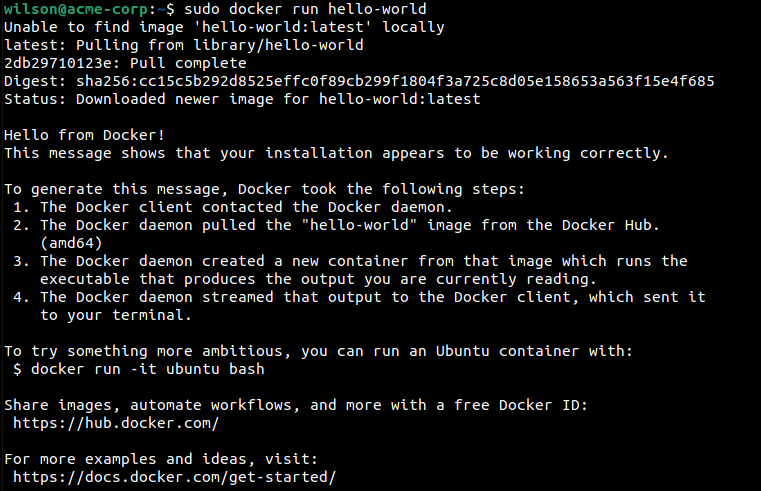


sudo apt-get install docker-ce docker-ce-cli containerd.io

Now that docker is installed, we can verify if it is working correctly by using the hello-world image. The hello world image simply runs in a container and prints a message, then exits. This can verify if Docker is working correctly.

sudo docker run hello-world

After running the hello-world docker image, you should see this, indicating that Docker is working correctly.



Kubernetes on Ubuntu

Since Docker is up and running, I can now install Kubernetes. First we need to add a signing key to download the package. Use the following command.

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

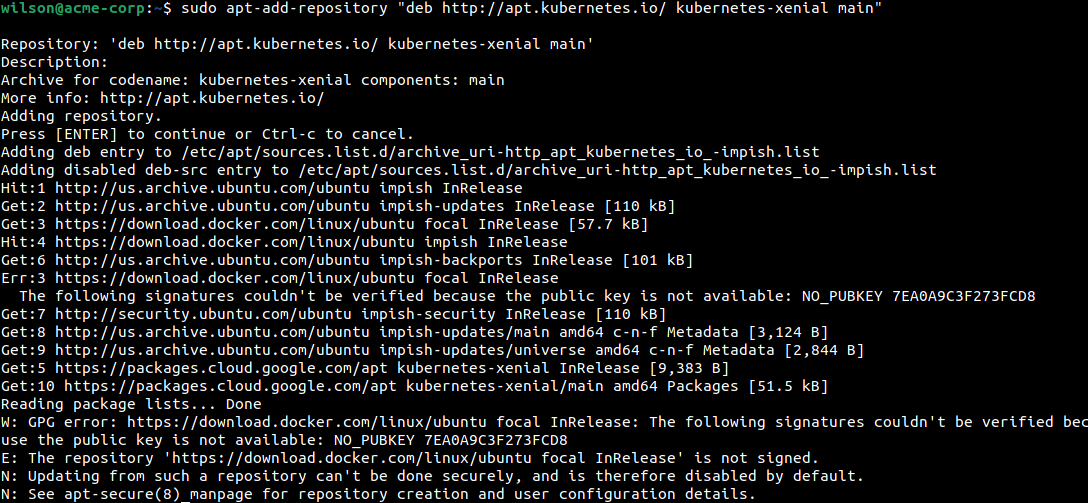
Note: From installing curl earlier there should not be any errors. However if there is an error, use the following to install curl.

sudo apt-get install curl

Since Kubernetes is not included within the default repos, we need to manually add it by using the following command.

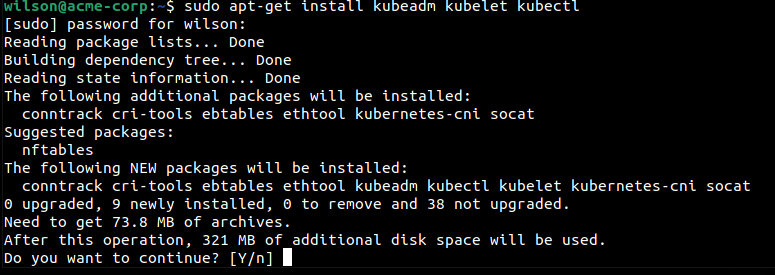
sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

A message should then appear prompting if you would like to continue. Press enter to proceed.

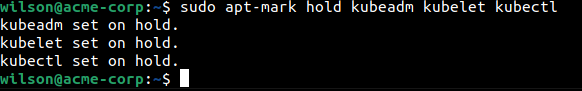


Now we can install the Kubernetes packages kubeadm, kubelet, and kubect1. Kubelet is the command to bootstrap the cluster, kublet can start pods and containers, and kubect1 is a command line utility to communicate with your cluster. Install these packages with the following commands

sudo apt-get install kubeadm kubelet kubectl



sudo apt-mark hold kubeadm kubelet kubectl



Now Kubernetes should be successfully installed on both the master node and worker node.

Minikube

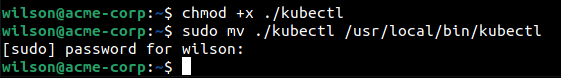
Minikube allows us to run Kubernetes in a single node cluster within a virtual machine. This will let us demo Kubernetes on a small scale without a full configuration.

The first thing we should do is make the binary kubectl from earlier executable using the following command.

chmod +x ./kubectl

Now lets move it to the bin folder in an executable pathway.

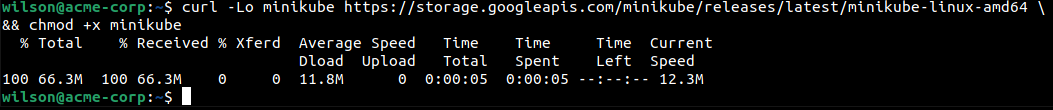
sudo mv ./kubectl /usr/local/bin/kubectl



Now to install Minikube using the latest release. We can do this by using the following command.

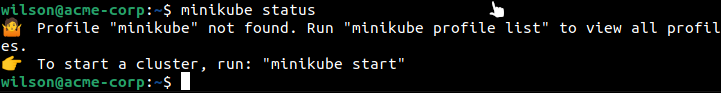
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \

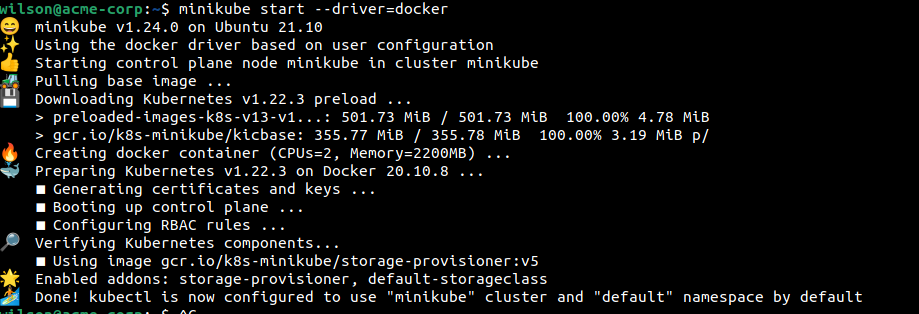
&& chmod +x minikube

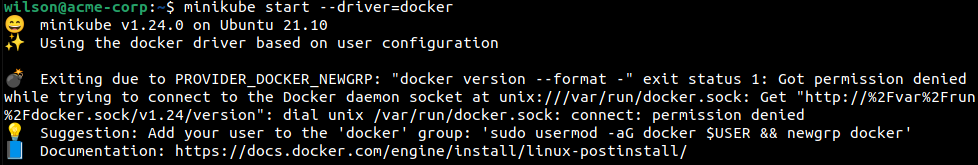


Lets check the status of our minikube with the following command. We can now see that minikube is installed but not started yet. Use the second command provided to start minikube using the Docker driver.

minikube status

minikube start --driver=docker

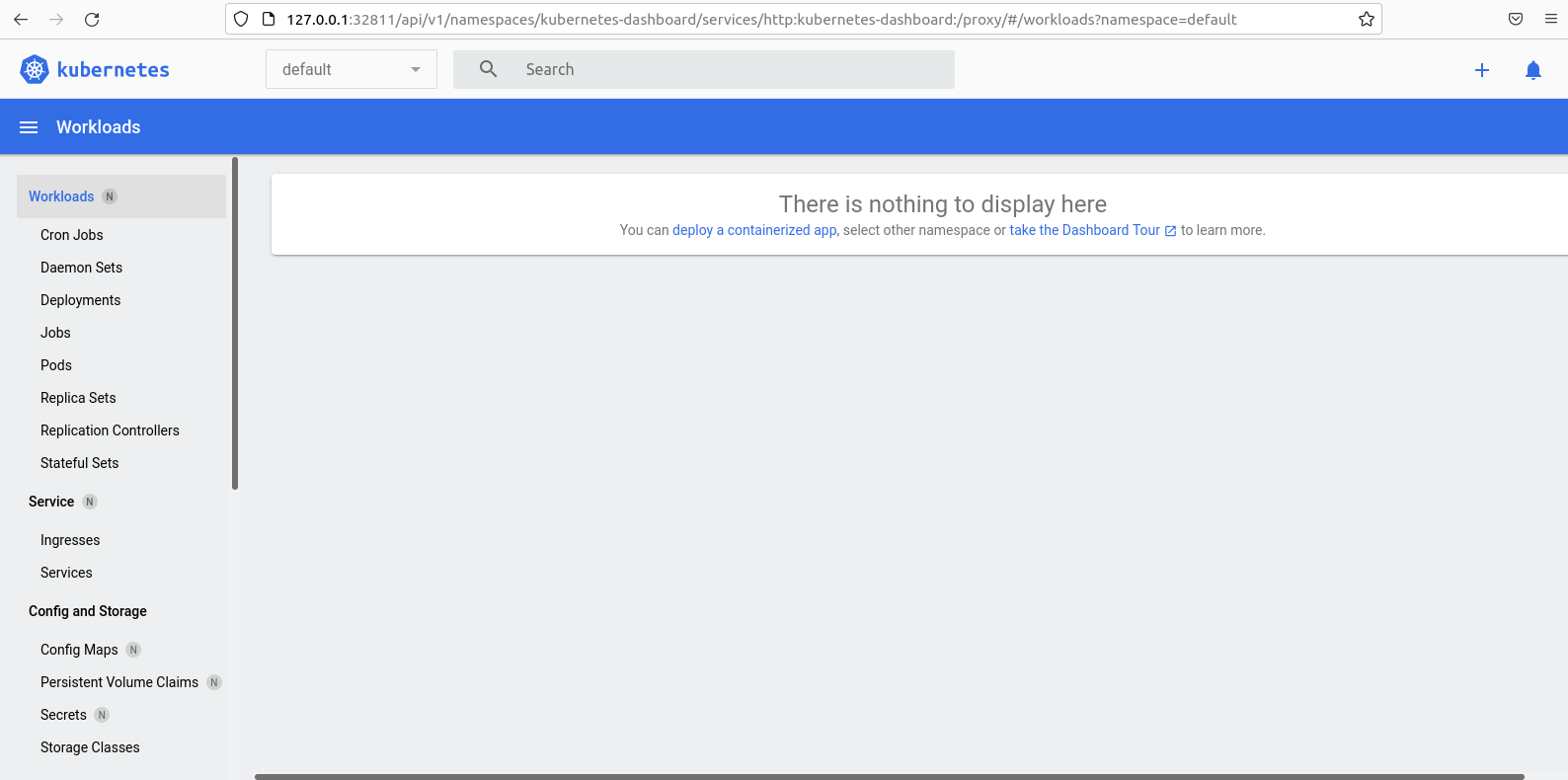


Note: You may get an error with permission denied when running it with docker. It will look like this. If this happens, run the following command. Replace $USER with your user name in Linux. This will give your user permissions to user Docker with minikube

sudo usermod -aG docker $USER && newgrp docker

Now to open the dashboard, use the following command.

minikube dashboard



In your default browser, you should now see a the Kubernetes dashboard window

Now that we know the dashboard is working, lets create a deployment to manage a pod. The pod runs a container based upon our docker image. Running the following kubectl command to create the deployment. Then use the second command to view the deployment and verify it is working.

kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4

kubectl get deployments



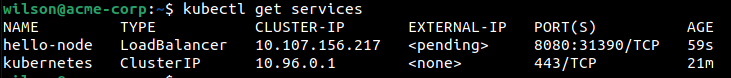
Now lets create a service to open our hello-node to outside the Kubernetes network. We can do this by exposing the pod as a Kubernetes service. Use the following command to expose the pod on port 8080.

kubectl expose deployment hello-node --type=LoadBalancer --port=8080

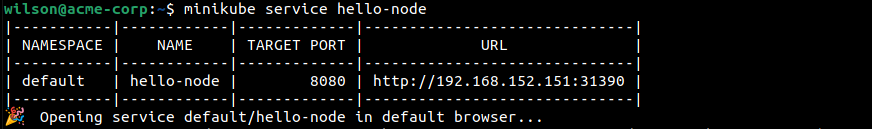


Now lets check our services with the following command. You should now see the hello-node within the services.

kubectl get services



Now we can run the hello node service. Perform this with the following command. You should now see a window pop up with information from the hello-node.

minikube service hello-node

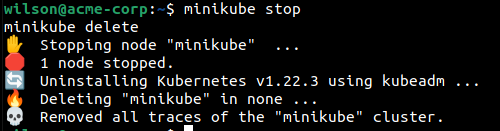
Since we have successfully got the service running, lets clean up our cluster and finish by removing the resources. Run the following commands to delete the kubectl service and our deployment of hello-node.

kubectl delete service hello-node

kubectl delete deployment hello-node

Finally stop and delete minikube with the following commands.

minikube stop

minikube delete