Create a Kubernetes cluster using minikube

# Install Minikube on Mac CLI

To install minikube on x86–64 mac OS, run the following two commands:

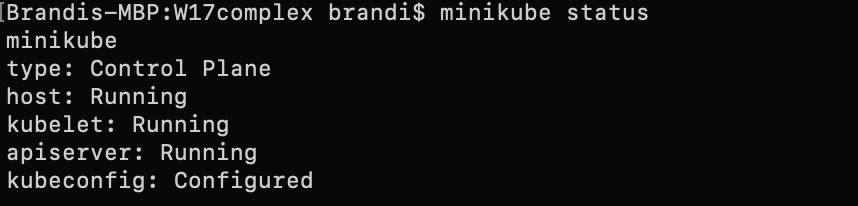
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-darwin-amd64  
sudo install minikube-darwin-amd64 /usr/local/bin/minikube

Use the following command to start a cluster:

minikube start

Use the following command to confirm minikube is running:

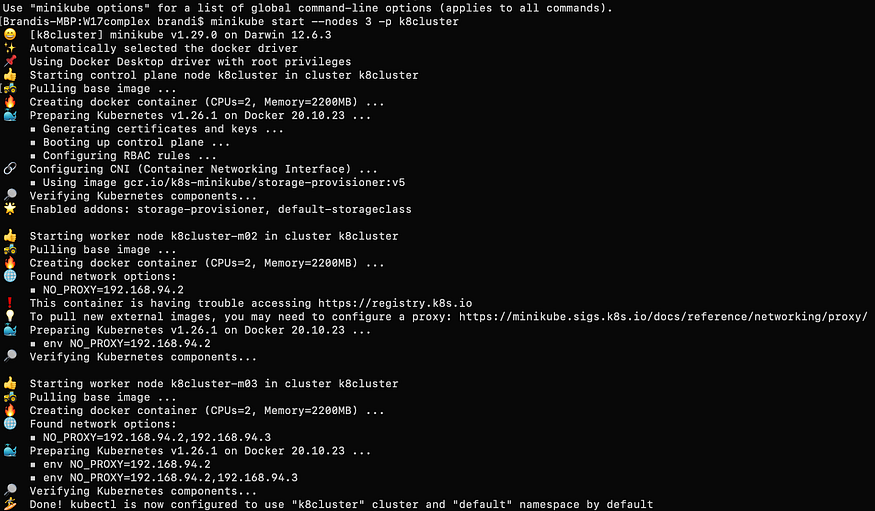
minikube status



# Create a 3 Node Cluster

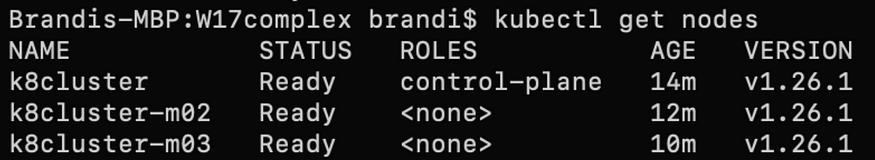
We want to create a 3 node cluster that will consist of one master node (for the rest of the tutorial we will call this node “control-plane”) and two worker nodes. We can create a 3 node cluster by using the following command:

minikube start --nodes # -p <cluster\_name>



The cluster may take several minutes to create, but minikube will output updates as things are happening. Once it says “Done!” use the following command to see the three nodes you just created.

kubectl get nodes

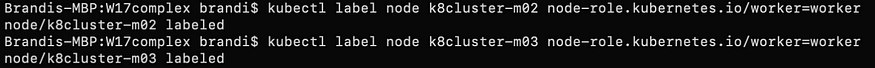


3 node cluster

# Label Nodes

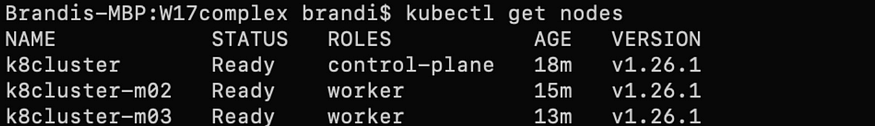
When we deploy our Redis and Apache pods, we do not want them to deploy to our control-plane, so we need to label our second and third nodes as “worker”. Use the following command to apply a worker label to the k8cluster-m02 and k8cluster-m03 nodes. You will have to run the command twice, once for each of the above nodes.

kubectl label node <node\_name> node-role.kubernetes.io/worker=worker



Use the following command to view the newly labeled nodes.

kubectl get nodes



The YAML files that we will deploy will search for these worker nodes based on a key:value label pair. Use the following command to apply a key:value label to the worker nodes:

kubectl label nodes <node\_name> role=worker

https://miro.medium.com/v2/resize:fit:875/1*2mpilhSbSxLy1Q_CKAtUQQ.png

Run the command again for the second worker node (k8cluster-m03).

https://miro.medium.com/v2/resize:fit:875/1*sKyHAYBX8QDmbuh0u3uaWA.png

# Redis Deployment YAML File

In my last tutorial [Create a Kubernetes Cluster Using Docker Desktop](https://medium.com/@bm54cloud/create-a-kubernetes-cluster-using-docker-desktop-72b493f3faa8), I went into detail on how to create a Kubernetes deployment YAML file, If you need to create a YAML file for the Redis and Apache deployments, please visit that tutorial now and come back when your files are created. Alternatively, you can just copy the code below for the Redis deployment with four replicas and save it an a .yml file.

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: redis-deploy  
 labels:  
 app: redis  
spec:   
 replicas: 4  
 selector:  
 matchLabels:  
 app: redis  
 template:   
 metadata:  
 labels:  
 app: redis  
 spec:  
 containers:  
 - name: redis  
 image: redis:7.0.9  
 ports:  
 - containerPort: 6379   
 nodeSelector:  
 role: worker

Notice in the file we’ve added **nodeSelector** on the container being created. With nodeSelector, we can choose which node our containers will be deployed on. Earlier we assigned our second and third nodes to be “worker” nodes and indicated that with the key:value pair role:worker. When we deploy this YAML file, the engine will look for any nodes that have the role:worker label and will deploy replicas on those nodes only. By using these labels, we are avoiding deploying replicas on the control-panel.

# Apache Deployment YAML File

Here is the file contents for the Apache deployment with ten replicas.

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: httpd-deploy  
 labels:  
 app: httpd  
spec:  
 replicas: 10  
 selector:  
 matchLabels:  
 app: httpd  
 template:  
 metadata:  
 labels:  
 app: httpd  
 spec:  
 containers:  
 - name: httpd  
 image: httpd:2.4.56  
 ports:  
 - containerPort: 80  
 nodeSelector:  
 role: worker

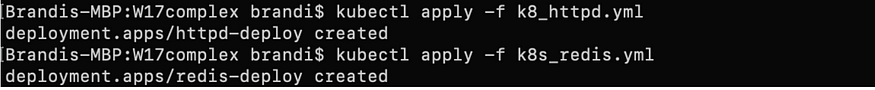
Note that these two files can actually be combined into one file by using --- to separate each manifest. To see what asingle file would look like, visit my GitHub [here](https://github.com/bm54cloud/Kubernetes/blob/main/deployment.yml).

# Deploy Redis and Apache Pods

Using the CLI, navigate to the directory that contains your Redis and Apache YAML files. Use the following command to deploy the pods:

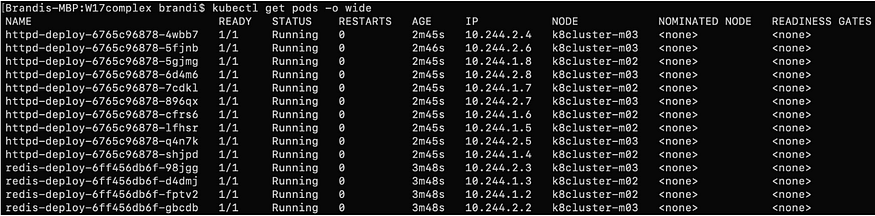
kubectl apply -f <yaml\_file.yml>

You will need to run this command twice. Once for the Redis YAML file and once for the Apache YAML file.



Your pods containing your replicas should have successfully deployed. To confirm this, use the following command:

kubectl get pods -o wide



You can see that all fourteen replicas are running and they are only running on the worker nodes (k8cluster-m02 and k8cluster-m03).

## Get Brandi McCall’s stories in your inbox

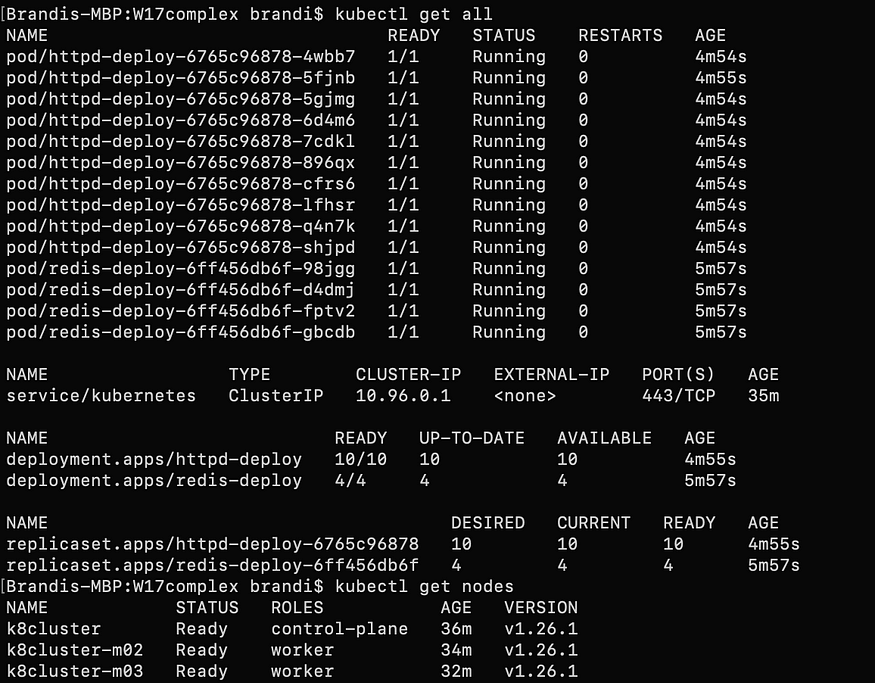
Join Medium for free to get updates from this writer.



Subscribe

To get more information about what we just created, use the following command:

kubectl get all



With this command you can view all your replicas, your cluster IP that was automatically created, your two deployments, your replicaSets that were created from your YAML files, and your three nodes. You have now successfully created a Kubernetes cluster with one control-plane node and two worker nodes that are running fourteen pods.

# Clean Up

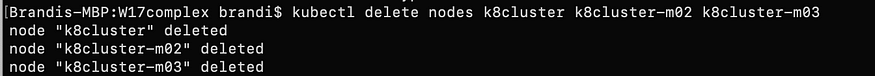
Clean up is optional but is a good practice to save space on your local computer. To clean up what you just created, we need to delete the deployments and the nodes. You can delete both deployments in a single command:

kubectl delete deployments <httpd\_deployment\_name> <redis\_deployment\_name>

https://miro.medium.com/v2/resize:fit:875/1*8mu6uiQEjDVyqQ3Ql8m7Mw.png

To delete your cluster nodes, use the following command:

kubectl delete nodes <node\_name> <node\_name> <node\_name>



To stop minikube, run the following command:

minikube stop

To delete minikube, run the following command:

minikube delete