Goal 1: Build a small Kubernetes cluster in the cloud.

I spent several weeks on this particular goal as it required as much learning as doing. I went through several hours of video instruction that goes over

1. The fundamentals of Kubernetes, including the installation and configuration (https://www.pluralsight.com/courses/kubernetes-installation-configuration-fundamentals)
2. How to interact with a Kubernetes cluster by deploying containers/pods and exposing services (https://www.pluralsight.com/courses/managing-kubernetes-api-server-pods).
3. Because my application relies on a mysql database that requires persistent storage, I watched a portion of this course (https://www.pluralsight.com/courses/configuring-managing-kubernetes-storage-scheduling).

For the above videos, I watched, took notes, and performed all of the demonstrations.

I chose to build my Kubernetes cluster in Azure, partly because that is the public cloud I have the most experience with, but also because I took CNIT-420 (Azure Fundamentals) this semester, so I wanted to take advantage of course overlap.

It should be noted that Azure provides AKS (Azure Kubernetes Service), which is a service that builds Kubernetes cluster for you. However, I did not take this route. I hand built built my cluster using IaaS, eventually deciding on Ubuntu (18.04 LTS) and Kubernetes v1.20.4.

(Note: I'm not sure how familiar you are with Kubernetes and Azure, so I have included many links throughout this section)

Getting the cluster up and running took several iterations. The 1st Pluralsight course that detailed installation was originally from 2018 (it has since been updated in Feb '21—after I started :/) and it stood the cluster up on using a previous version Ubuntu and Kubernetes version (Ubuntu 16.04/Kubernetes 1.09). Being unfamiliar to Kubernetes, I wanted to follow the course material exactly, but at the end of the day that version of Kubernetes did not play nice with a newer Kubernetes module called a Kubernetes CNI (https://kubernetes.io/docs/concepts/cluster-administration/networking/). In a nutshell, a Kubernetes CNI is a separate "sidecar" container that provides a SDN (software defined network) that gives the illusion (using packet encapsulation) that every container (or pod) in the cluster is in a flat network space. One of the key requirements of building a Kubernetes cluster is that every container (or pod) can talk to any other container without NAT (network address translation). For this I unsuccessfully tried calico (https://docs.projectcalico.org/about/about-calico), and if you're curious, you can read my request for help on the calico msg board here (https://discuss.projectcalico.org/t/newb-q-cant-ping-pod-network-from-master-iaas-k8s-1-20-4-in-azure-on-ubuntu-18-04-w-calico/218). I finally went with the less popular weave CNI (https://www.weave.works/blog/weave-net-kubernetes-integration/) that uses an encapsulation type (VXLAN encapsulation) that is compatible with Azure.

Kubernetes is an ever evolving beast, so online documentation is often outdated. While the online Pluralsight videos (from 2018) were very helpful, when I decided to go with a more modern version of Ubuntu/Kubernetes, the "Kubernetes The Hard Way" tutorial here was also a great resource: https://github.com/ivanfioravanti/kubernetes-the-hard-way-on-azure

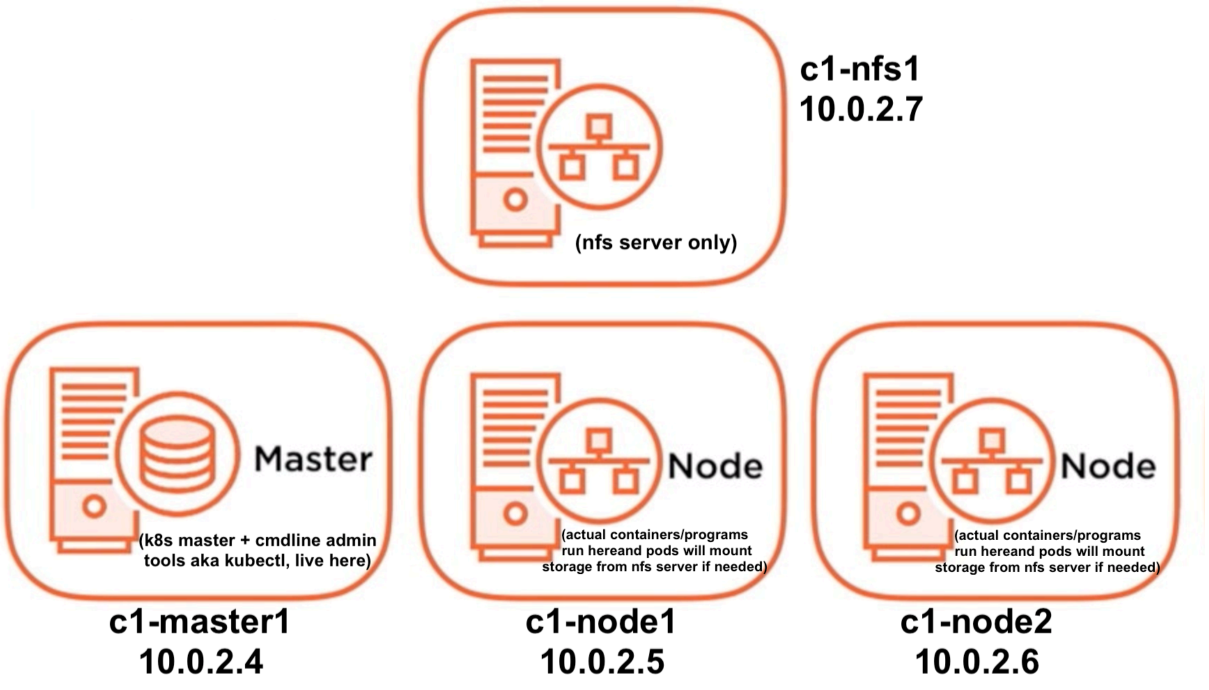
As you might imagine, there was a lot of "bumper bowling" my way through the cluster build and I was oftentimes unsure of where I was potentially going wrong. With so much trial and error, I scripted the build and tear down of the all the Ubuntu VM's in the cluster using the azure cli (https://docs.microsoft.com/en-us/cli/azure/) using ARM templates (https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview).

Here are my scripts used for automating the IaaS cluster build:
https://github.com/olivelawn/cs199-indep-k8s/tree/main/iaas-cluster-build/arm-templates/create-vm-masterB2s-nodeB1s

Once you have the hosts up, here are the final steps I took to build the Kubernetes master and nodes, respectively.
https://github.com/olivelawn/cs199-indep-k8s/blob/main/k8s-cluster-build/master.txt
https://github.com/olivelawn/cs199-indep-k8s/blob/main/k8s-cluster-build/node.txt

I also found that running this cluster in Azure was costing me a small fortune, so I tried to reduce the size of the VM's to Azure's smallest size (B1s), but the Kubernetes master node fails to initialize with 1vCPU/1GB RAM, so I ended up going with a slightly beefier Kubernetes master, but went with the smallest size for the worker nodes (and separate NFS server—more on that later).

Here is the high level architecture of my Kubernetes cluster:


c1-nfs1
10.0.2.7
(nfs server only)
Master
(k8s master + cmdline admin tools aka kubectl, live here)
Node
(actual containers/programs run hereand pods will mount storage from nfs server if needed)
Node
(actual containers/programs run hereand pods will mount storage from nfs server if needed)
c1-master1
10.0.2.4
c1-node1
10.0.2.5
c1-node2
10.0.2.6

Below shows the VM instances as views from Azure:

```
[parker@parker k8s4_rg]$ az vm list -g K8S4_RG -d -o table
Name        ResourceGroup    PowerState    PrivateIps     Fqdns    Location    Zones
----------  ---------------  ------------  -------------  -------  ----------  -------
c1-master1  K8S4_RG          VM running    10.0.2.4                westus2
c1-nfs1     K8S4_RG          VM running    10.0.2.7                westus2
c1-node1    K8S4_RG          VM running    10.0.2.5                westus2
c1-node2    K8S4_RG          VM running    10.0.2.6                westus2
```

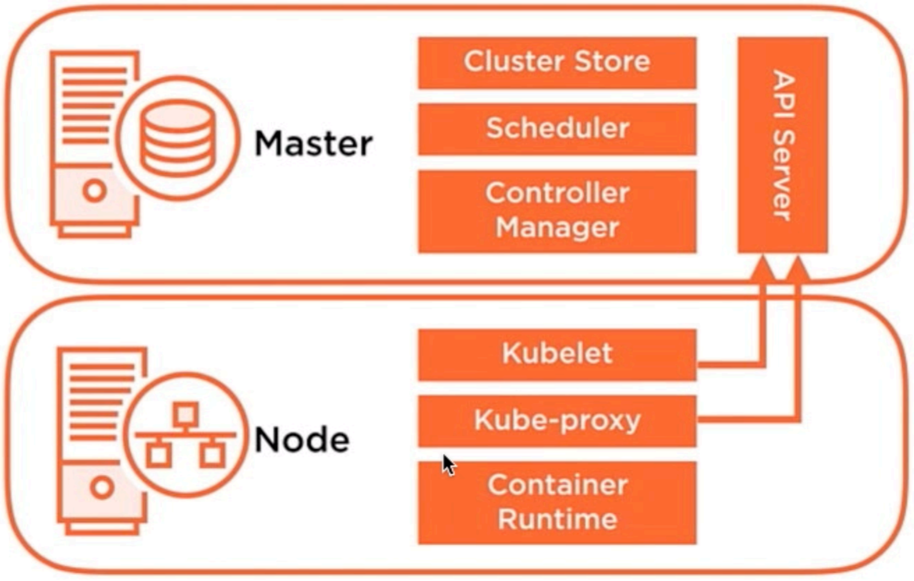
Below shows all the members, or "nodes" of the cluster as viewed from the Kubernets master:

```
parker@c1-master1:~$ kubectl get nodes
NAME         STATUS   ROLES                  AGE    VERSION
c1-master1   Ready    control-plane,master   71d    v1.20.4
c1-node1     Ready    <none>                 71d    v1.20.4
c1-node2     Ready    <none>                 112s   v1.21.1
```

Finally, this is what a Kubernetes cluster looks like with no user-defined containers or "pods" running. This is essentially an "empty" cluster. The important thing to understand here is that even the components that make up kubernetes itself are run as containers or "pods."

```
parker@c1-master1:~$ kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                 READY   STATUS            RESTARTS   AGE     IP          NODE
kube-system   coredns-74ff55c5b-25zh6              1/1     Running           14         71d     10.32.0.3   c1-master1
kube-system   coredns-74ff55c5b-2prdj              1/1     Running           14         71d     10.32.0.2   c1-master1
kube-system   etcd-c1-master1                      1/1     Running           14         71d     10.0.2.4    c1-master1
kube-system   kube-apiserver-c1-master1            1/1     Running           15         71d     10.0.2.4    c1-master1
kube-system   kube-controller-manager-c1-master1   1/1     Running           14         71d     10.0.2.4    c1-master1
kube-system   kube-proxy-l2kp7                     1/1     Running           0          9m50s   10.0.2.6    c1-node2
kube-system   kube-proxy-mwlgs                     1/1     Running           15         71d     10.0.2.5    c1-node1
kube-system   kube-proxy-wz4j5                     1/1     Running           14         71d     10.0.2.4    c1-master1
kube-system   kube-scheduler-c1-master1            1/1     Running           14         71d     10.0.2.4    c1-master1
kube-system   weave-net-gdrhp                      2/2     Running           30         71d     10.0.2.4    c1-master1
kube-system   weave-net-q8s6x                      2/2     Running           1          9m50s   10.0.2.6    c1-node2
kube-system   weave-net-vw96w                      2/2     Running           31         71d     10.0.2.5    c1-node1
```

The below has less to do with my goal of building a cluster, but if you're curious, here are the various components that make up a Kubernetes cluster, which coincide with many of the pods listed above.


Master
Cluster Store
Scheduler
Controller Manager
API Server
Node
Kubelet
Kube-proxy
Container Runtime