1. Git clone my repo (and cd into it)
2. Provision VMs (Need virtualBox and vagrant)
    a. Vagrant status (shows not created)
    b. Vagrant up (Spins up 3 vms, 1 master, 2 worker)
    c. Vagrant status should show running

3. Go here and set up a container runtime (i.e. containerd) for all 3 nodes
    a. Enable IPv4 packet forwarding
        i. NOTE: Lab was different than docs. This worked:

```
sudo modprobe br_netfilter
echo 'br_netfilter' | sudo tee /etc/modules-load.d/br_netfilter.conf

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sudo vim /etc/sysctl.conf
# Add these lines:
#net.bridge.bridge-nf-call-iptables = 1
#net.bridge.bridge-nf-call-ip6tables = 1
sudo sysctl -p
```

    b. Follow the setup for the container runtime and install containerd
    c. i.e. in my case for ubuntu I just had to run a whole bunch of commands and then ‘sudo apt-get install containerd.io’
        i. They updated their download. Just run:

```
Sudo apt update
sudo apt install -y containerd
```

4. Set up cgroup driver (systemd is not default, so we have to set it)
    a. Run this on all three nodes:

```
containerd config default | sed 's/SystemdCgroup = false/SystemdCgroup =
true/' | sudo tee /etc/containerd/config.tomlsudo systemctl restart
containerd
```

    b. OR Go to /etc/containerd/config.toml for all 3 nodes and replace with:

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
    SystemdCgroup = true
```

c. Then run sudo systemctl restart containerd

5. Installing kudeadm, kubelet, and kubectl
    a. kubeadm: the command to bootstrap the cluster.
    b. kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
    c. kubectl: the command line util to talk to your cluster.
    d. Debian based instructions are for Kubernetes v1.31.

NOTE this worked on lab and on my machine:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
# To see the new version labels
sudo apt-cache madison kubeadm
sudo apt-get install -y kubelet=1.31.0-1.1 kubeadm=1.31.0-1.1
kubectl=1.31.0-1.1
sudo apt-mark hold kubelet kubeadm kubectl
```

NOTE this did not work on lab or my machine, but was on K8s docs:

1. Update the apt package index and install packages needed to use the Kubernetes apt repository:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

2. Download the public signing key for the Kubernetes package repositories. The same signing key is used for all repositories so you can disregard the version in the URL:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

3. Add the appropriate Kubernetes apt repository. Please note that this repository have packages only for Kubernetes 1.31; for other Kubernetes minor versions, you need to change the Kubernetes minor version in the URL to match your desired minor version (you should also check that you are reading the documentation for the version of Kubernetes that you plan to install).

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

4. Update the apt package index, install kubelet, kubeadm and kubectl, and pin their version:

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

5. (Optional) Enable the kubelet service before running kubeadm:

```
sudo systemctl enable --now kubelet
```

6. Create the cluster on the master node
    a. Look for enp0s8 under 'ip add' to get the ip address of your master node
        i. Might also need --apiserver-cert-extra-sans=controlplane

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--apiserver-advertise-address=192.168.1.89 --upload-certs
```

    b. You should get a command along the lines of "sudo kubeadm join …" SAVE THIS
    c. Run:

```
mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

    d. kubectl apply -f https://reweave.azurewebsites.net/k8s/v1.29/net.yaml. OR Alt CNIs
        i. I used flannel:

```
kubectl apply -f
https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
```

    e. kubectl edit ds <CNI_ds_name> -n <CNI_namespace>
        i. If using weave, add under weave pod's env:

```
- name: IPALLOC_RANGE
   value: 10.244.0.0/16
```

7. If for any reason something was irreversibly messed up you can list the vms, turn them off, and remove them from your local machine with:

```
VBoxManage list runningvms
VBoxManage controlvm <vmName> poweroff
VBoxManage unregistervm <vmName> --delete
```

8. Should have something like this when done:

```
vagrant@controlplane:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   24m   v1.31.0
node01         Ready    <none>          36s   v1.31.0
node02         Ready    <none>          24s   v1.31.0
vagrant@controlplane:~$ kubectl get pods -A
NAMESPACE      NAME                                   READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-66rjd                  1/1     Running   0          30s
kube-flannel   kube-flannel-ds-ll99l                  1/1     Running   0          42s
kube-flannel   kube-flannel-ds-qrwsg                  1/1     Running   0          7m58s
kube-system    coredns-6f6b679f8f-hjmz8               1/1     Running   0          24m
kube-system    coredns-6f6b679f8f-knvgc               1/1     Running   0          24m
kube-system    etcd-controlplane                      1/1     Running   0          24m
kube-system    kube-apiserver-controlplane            1/1     Running   0          24m
kube-system    kube-controller-manager-controlplane   1/1     Running   0          24m
kube-system    kube-proxy-rzbct                       1/1     Running   0          30s
kube-system    kube-proxy-sk9jl                       1/1     Running   0          24m
kube-system    kube-proxy-tjpxt                       1/1     Running   0          42s
kube-system    kube-scheduler-controlplane            1/1     Running   0          24m
```

```
vagrant@controlplane:~$ kubectl get all -A
NAMESPACE      NAME                                   READY   STATUS    RESTARTS   AGE
kube-flannel   pod/kube-flannel-ds-66rjd              1/1     Running   0          48s
kube-flannel   pod/kube-flannel-ds-ll99l              1/1     Running   0          60s
kube-flannel   pod/kube-flannel-ds-qrwsg              1/1     Running   0          8m16s
kube-system    pod/coredns-6f6b679f8f-hjmz8           1/1     Running   0          24m
kube-system    pod/coredns-6f6b679f8f-knvgc           1/1     Running   0          24m
kube-system    pod/etcd-controlplane                  1/1     Running   0          24m
kube-system    pod/kube-apiserver-controlplane        1/1     Running   0          24m
kube-system    pod/kube-controller-manager-controlplane   1/1     Running   0          24m
kube-system    pod/kube-proxy-rzbct                   1/1     Running   0          48s
kube-system    pod/kube-proxy-sk9jl                   1/1     Running   0          24m
kube-system    pod/kube-proxy-tjpxt                   1/1     Running   0          60s
kube-system    pod/kube-scheduler-controlplane        1/1     Running   0          24m

NAMESPACE     NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
default       service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP                  24m
kube-system   service/kube-dns     ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   24m

NAMESPACE      NAME                             DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
  AGE
kube-flannel   daemonset.apps/kube-flannel-ds   3         3         3       3            3           <none>
  12m
kube-system    daemonset.apps/kube-proxy        3         3         3       3            3           kubernetes.io/os=linux
  24m

NAMESPACE     NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/coredns   2/2     2            2           24m

NAMESPACE     NAME                                 DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/coredns-6f6b679f8f   2         2         2       24m
```

The network is now set up. Now you could (or make your developers) start building the network. I.e.:

kubectl run reverse-proxy --image=nginx