BONUS EXERCISE:

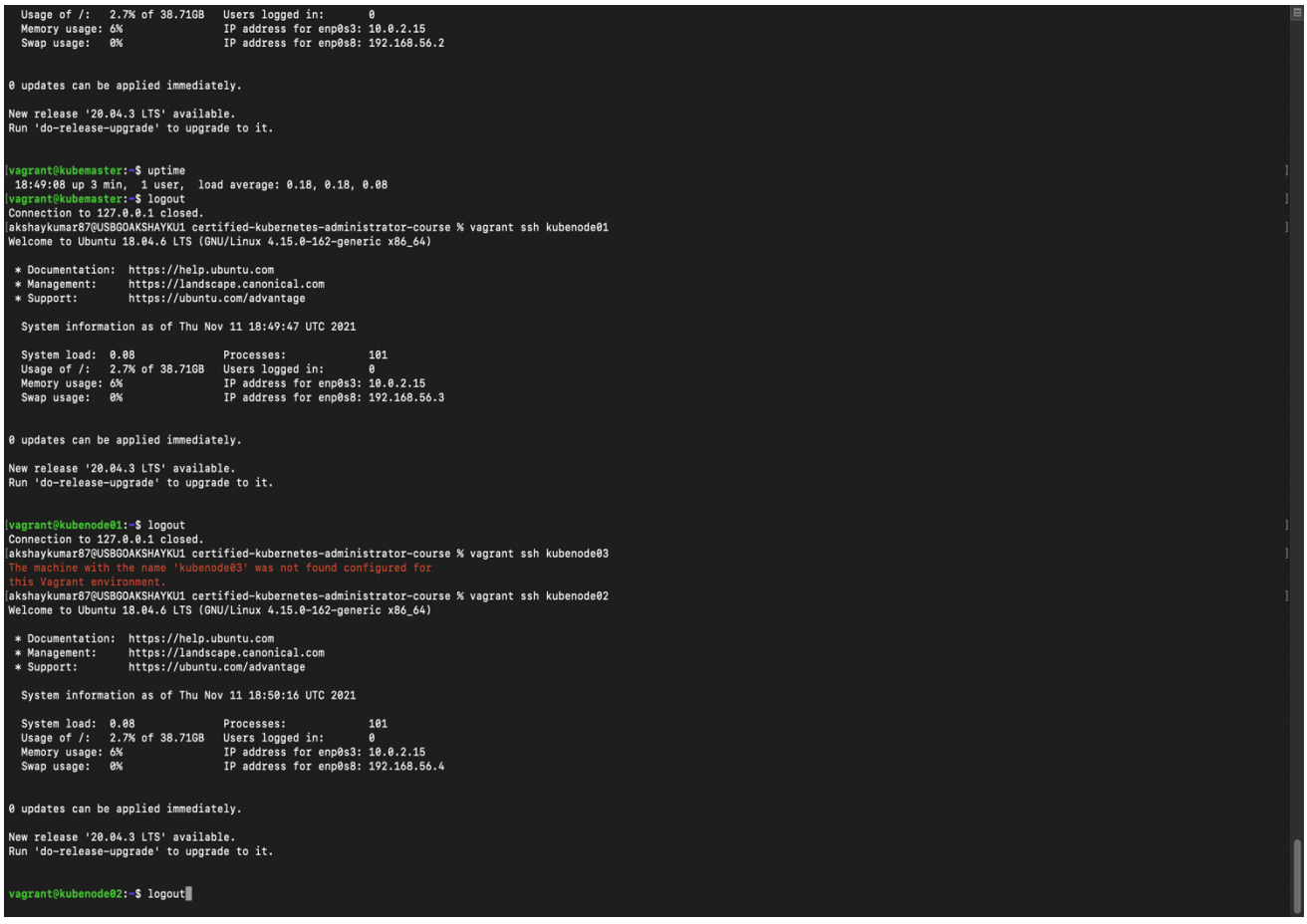
Configure 3 nodes Kubernetes cluster with a master and 2 nodes(node 1,node2) using kubeadm.

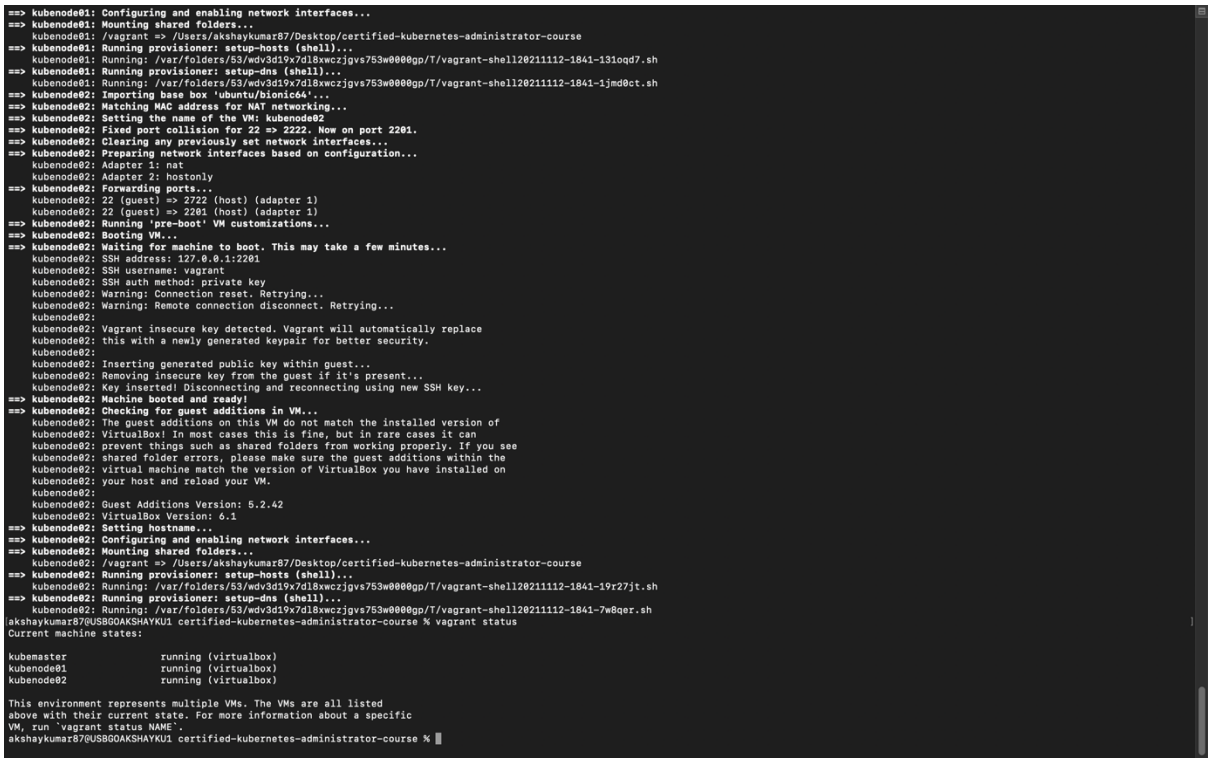
Answer: For this there are some steps that we need follow. They are:

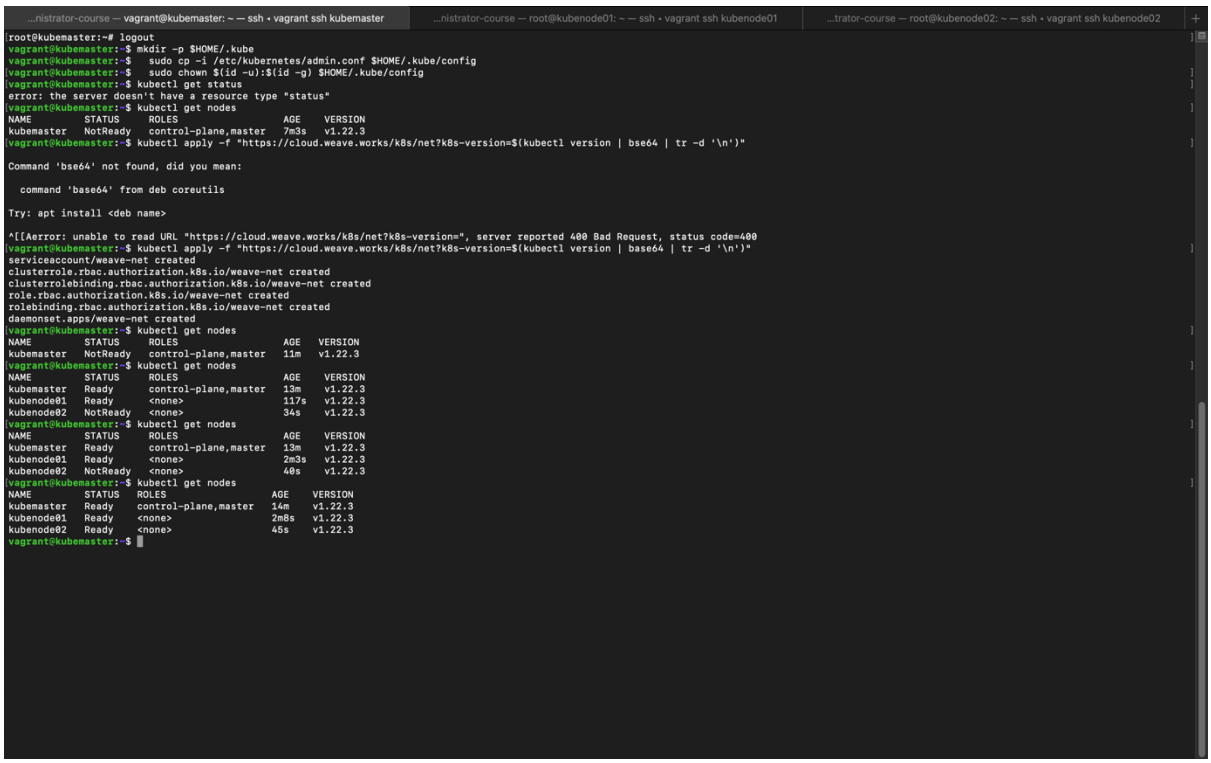
a) Need to configure Virtual Machine
b) Install container runtime on every node like docker
c) Install kubeadm in every node
d) Need to initialize master server
e) Set-up the pod network
f) Join worker node to the master node

After going through all the steps as described in the document of Kubernetes, I did the above configuration.

Showing screenshots in the support of this.







Last screenshot showing the all three nodes are in the ready status.

1. **Deploy a nginx deployment with name web1 and custom index.html. Express it to nodePort of 30080 in web namespace for this scenario.**

Answer:

For accomplishing this need to use **congiMap.** This is used to add custom content to the nginx deployement.

Here, made 3 yaml files for this. Showing below one by one.

Conf.yaml file:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: index-config
 namespace: default
data:
 index.html: |
  <html>
  <h1>Welcome Linker</h1>
  </br>
  <h1>This is the custom index file.</h1>
  </html
```

Created **myapp.yaml** file for the deployment of nginx.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 namespace: default
spec:
 selector:
  matchLabels:
   app: nginx
```

```
 replicas: 2
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
    volumeMounts:
      - name: nginx-index-file
        mountPath: /usr/share/nginx/html/
   volumes:
   - name: nginx-index-file
    configMap:
     name: index-config
```

Lastly created service file as servi.yaml:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
 namespace: default
spec:
 selector:
  app: nginx
 type: NodePort
 ports:
  - port: 80
   nodePort: 30080
   targetPort: 80
```

2.. Create a wordpress deployment with mysql as a database and expose it to nodeport 30081. Use "volume" namespace for this scenario and login to wordpress with predefined credentials.

Answer: For accomplishing the above we need to do some important steps. Where we need PV, PVC and secrets for storing the credentials.

Creating a kustomization.yaml file where it contains a Secret generator, Mysql resource config, wordpress resource config.

Following is the command for adding Secret generator:

```
Cat <<EOF >./kustomization.yaml
secretGenerator:
        - name: mysql-pass
          literals:
         - password=akshay01
EOF
```

After this need to add the resource config for mysql and wordpress.

Uploaded the same in my git hub account.

Now downloading the mysql deployment configuration file
And wordpress config file.

After this need to add these in kustomization.yaml file.

And then can apply it using the command:

We need to then apply it. So for this we can use command:

Kubectl apply -k ./

And then running the command

minikube service wordpress –url can get the url and we can check this as well.

In support of this showing the screenshot as well:

```
Last login: Wed Nov 10 21:12:33 on ttys000
akshaykumar87@USBGOAKSHAYKU1 ~ % kubectl apply -k ./
error: unable to find one of 'kustomization.yaml', 'kustomization.yml' or 'Kustomization' in directory '/Users/akshaykumar87'
akshaykumar87@USBGOAKSHAYKU1 ~ % ls
Desktop                 Downloads               Movies                  Parallels               Public                  kubectl.sha256          minikube-darwin-arm64
Documents               Library                 Music                   Pictures                Vagrantfile             kuber
akshaykumar87@USBGOAKSHAYKU1 ~ % cd kuber
akshaykumar87@USBGOAKSHAYKU1 kuber % kubectl apply -k ./
secret/mysql-pass-6999bkmk7d created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
akshaykumar87@USBGOAKSHAYKU1 kuber % kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-6lgjj   kubernetes.io/service-account-token   3      21h
mysql-pass-6999bkmk7d   Opaque                              1      14s
akshaykumar87@USBGOAKSHAYKU1 kuber % kubectl get pvc
NAME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-pv-claim   Bound    pvc-6983ae57-a79a-4239-9b2d-2dc82b44935d   20Gi       RWO            standard       32s
wp-pv-claim      Bound    pvc-7a4c8d7d-3b95-4d48-8008-f3b972311af9   20Gi       RWO            standard       32s
akshaykumar87@USBGOAKSHAYKU1 kuber % kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
my-deployment-8b45d68b4-gmbdx      1/1     Running             0          110m
my-deployment-8b45d68b4-mfdwf      1/1     Running             0          110m
my-deployment-8b45d68b4-q4lw7      1/1     Running             0          110m
nginx-deployment-c7c8cb6d4-5nhp5   1/1     Running             0          98m
nginx-deployment-c7c8cb6d4-qrs2c   1/1     Running             0          98m
replica-set-2bhjx                  1/1     Running             0          110m
replica-set-9htwq                  1/1     Running             0          110m
replica-set-g82n7                  1/1     Running             0          110m
wordpress-644958c448-cjf8x         0/1     ContainerCreating   0          50s
wordpress-mysql-67456c9c78-xws2w   1/1     Running             0          50s
akshaykumar87@USBGOAKSHAYKU1 kuber % kubectl get services wordpress
NAME        TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
wordpress   LoadBalancer   10.99.171.76   <pending>     80:30231/TCP   62s
akshaykumar87@USBGOAKSHAYKU1 kuber % minikube service wordpress --url
🏃  Starting tunnel for service wordpress.
|-----------|-----------|-------------|------------------------|
| NAMESPACE |   NAME    | TARGET PORT |          URL           |
|-----------|-----------|-------------|------------------------|
| default   | wordpress |             | http://127.0.0.1:64140 |
|-----------|-----------|-------------|------------------------|
http://127.0.0.1:64140
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

Showing the creation of all above files and the url of wordpress.

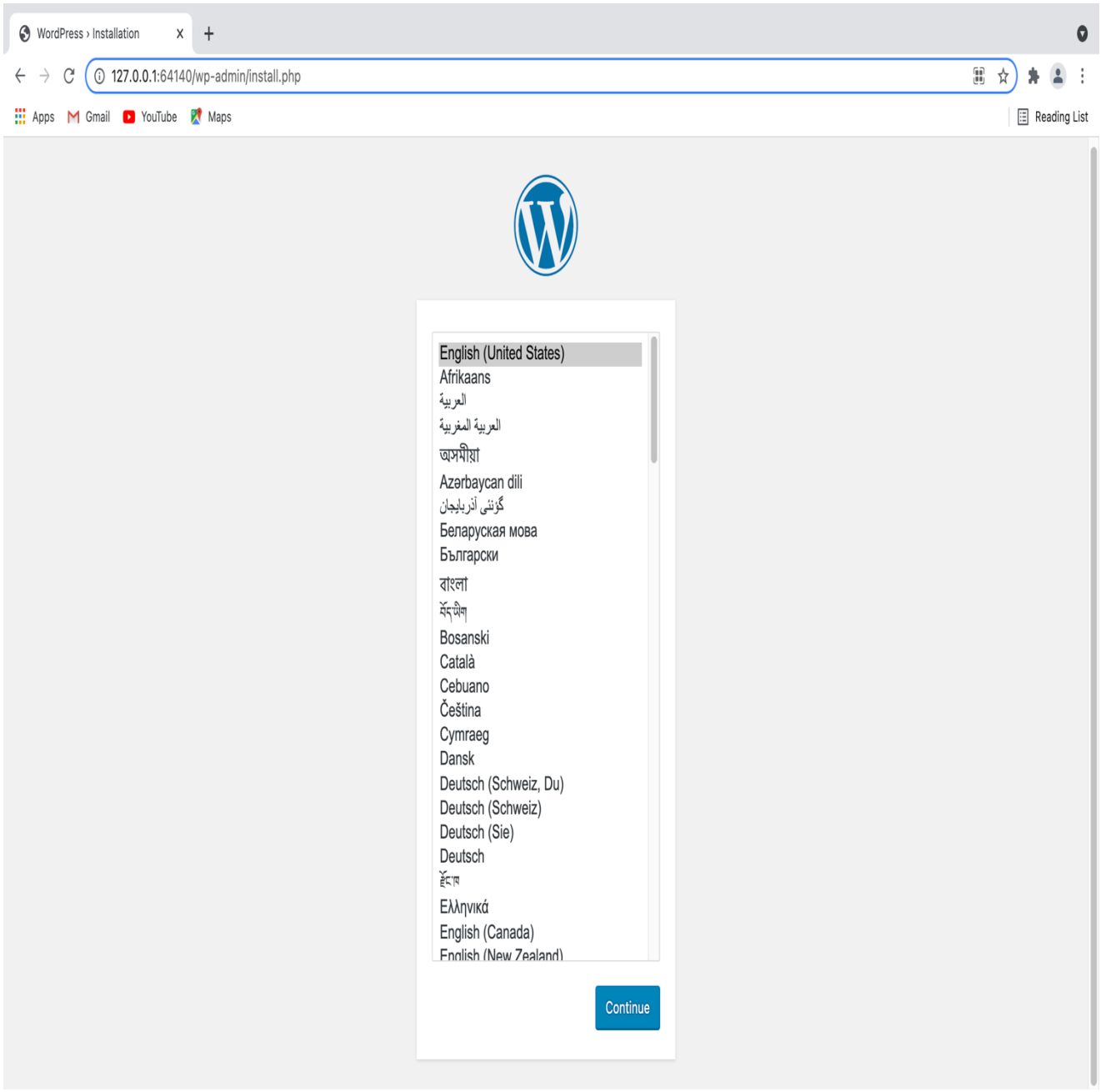


Figure showing the accessing part of wordpress.

3. **Create a pod named sidecar with 2 containers named main and sidecar respectively. Fetch the logs from app container to sidecar container. Use nginx for main and busybox:1.28 image for sidecar container. Use namespace sidecar for this scenario.**

Answer:

For accomplish this it can be done in many ways. Here, I;m doing the communication using sharing the same volume.

For this made a myapp.yaml file where the sidecar container add the current date to the volume and when the user request it from the main then it read and return the same from the volume where it is mounted.

Showing the myapp.yaml file for the same.

```
apiVersion: v1
kind: Pod
metadata:
  name: sidecar
spec:
  volumes:
  - name: html
    emptyDir: {}
  containers:
  - name: main
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: sidecar
    image: busybox:1.28
    volumeMounts:
    - name: html
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
          - while true; do
        date >> /html/index.html;
        sleep 1;
      done
```

Here showing the output: