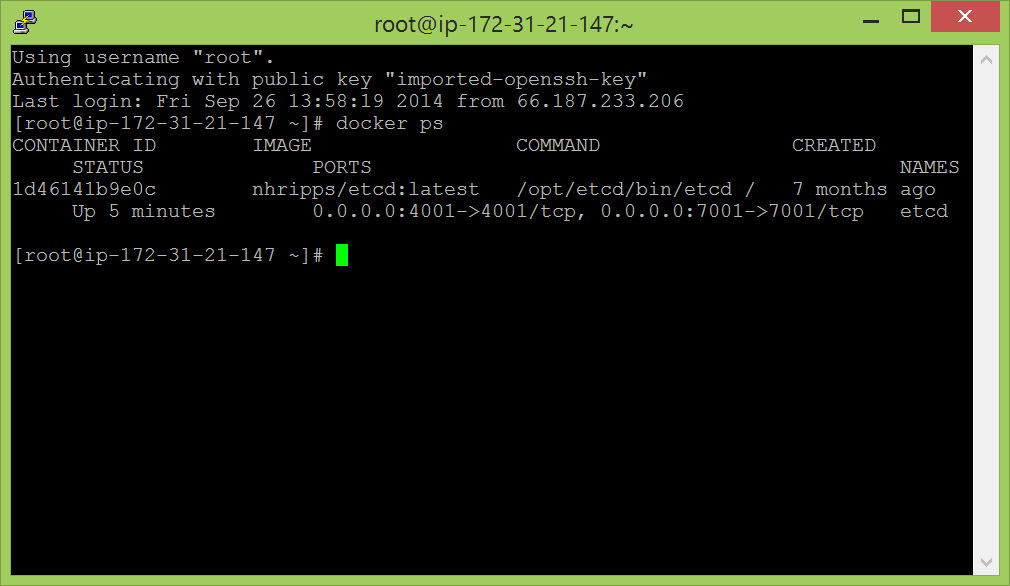
Lab #2: Kubernetes

* Run an instance of the publicly available images provided for the lab.
* Connect to the amazon web services using the putty connection like in the previous lab.
* To check the containers running in the local instance:

docker ps



* The etcd shown here means that it is a data store where the data for the instance is going to be.
* There is no guarantee that a restful api needs a key value pair. There can be many other ways to get it running. But, we are using the ey value pair. To create a key value pair, we use:

curl -L -X PUT http://127.0.0.1:4001/v2/keys/message -d value="Hello"



* To get the key value pair which has been created, we use:

curl -L -X GET <http://127.0.0.1:4001/v2/keys/message>



* As shown in the screenshot above, I was just trying to see if the value could be changed for the same key and if it is being updated or not. When not needed anymore, the key value pair can be deleted using:

curl -L -X DELETE http://127.0.0.1:4001/v2/keys/message

I used get to verify if the key value pair was deleted.

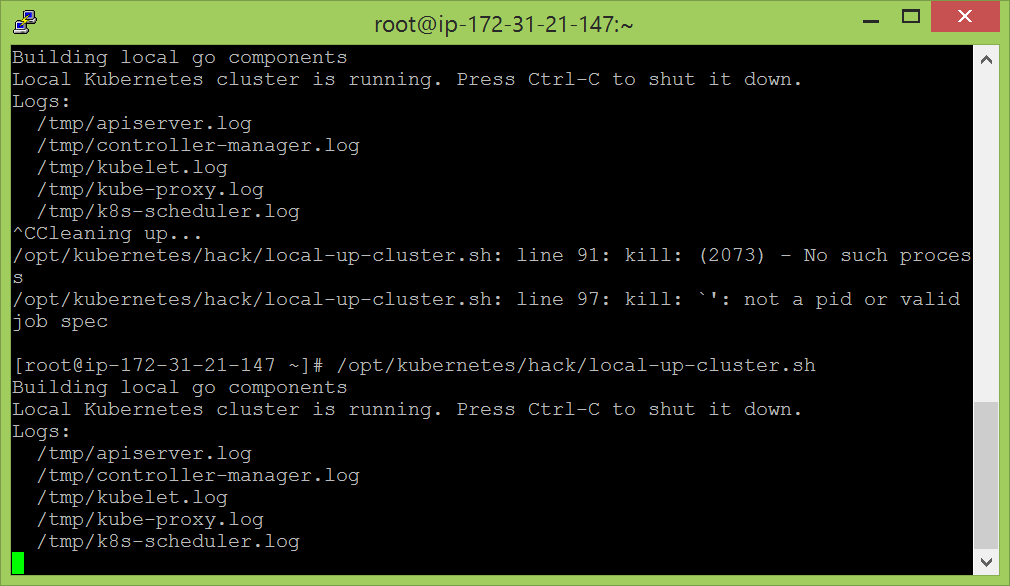


* The utility service: etcdctl command line tool.



* To start the kubernetes service:

/opt/kubernetes/hack/local-up-cluster.sh



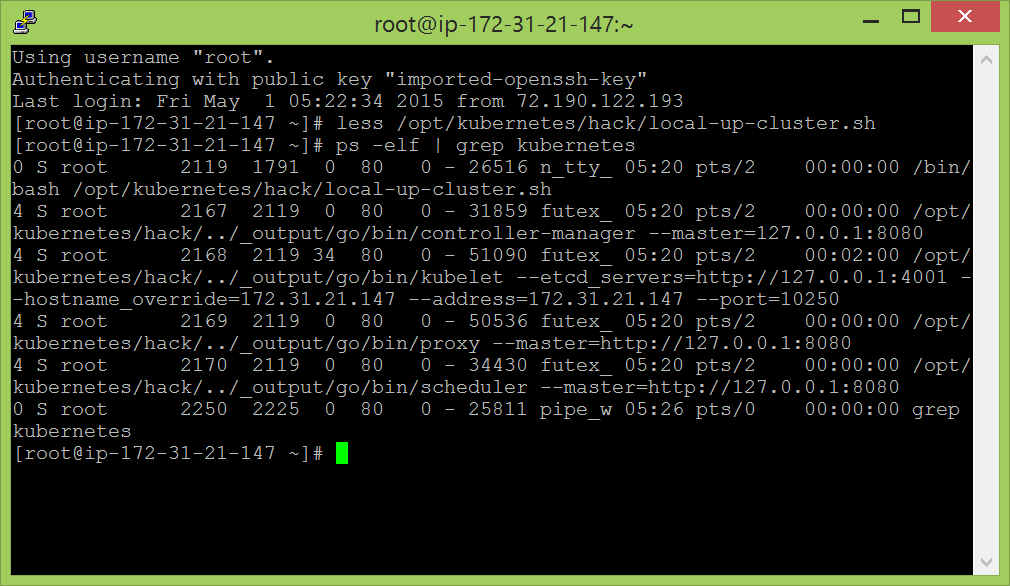
* Open a duplicate putty terminal and run the following command:

less /opt/kubernetes/hack/local-up-cluster.sh

and then :wq. The first command shows a script which has the functions to start minions, controllers, etc. the second exits from the script.

* To verify if all the processes are running, use:

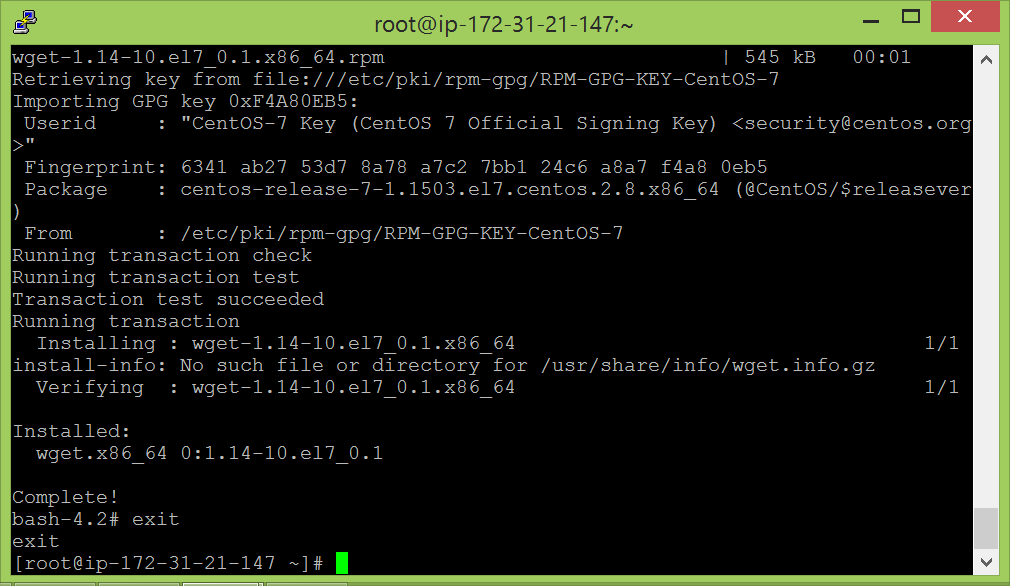
ps -elf | grep kubernetes



* To check another script to run a pod:

vi /root/guestbook/redis-master.json





* To create new pods:

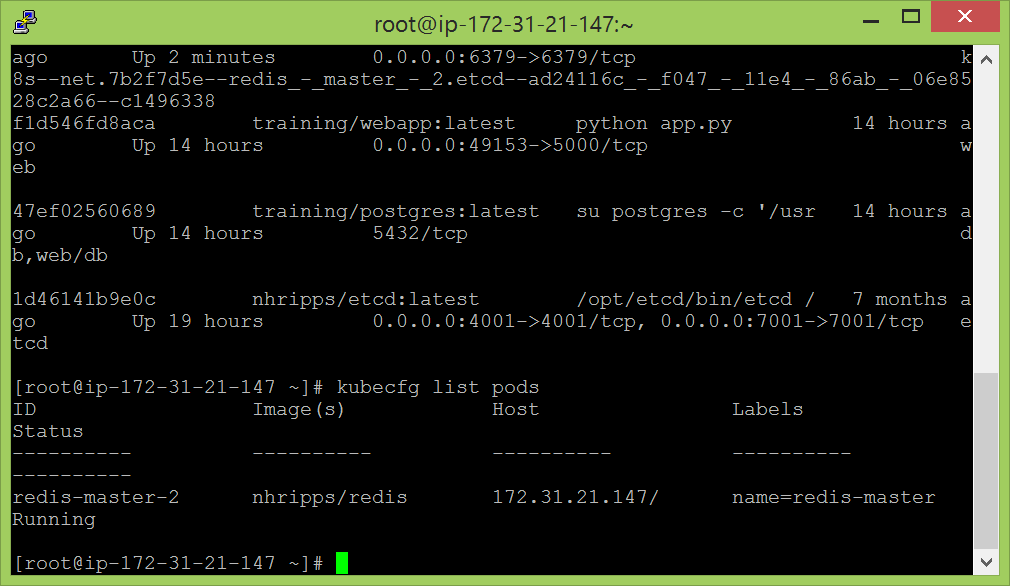
kubecfg –c /root/guestbook/redis-master.json create pods



docker ps

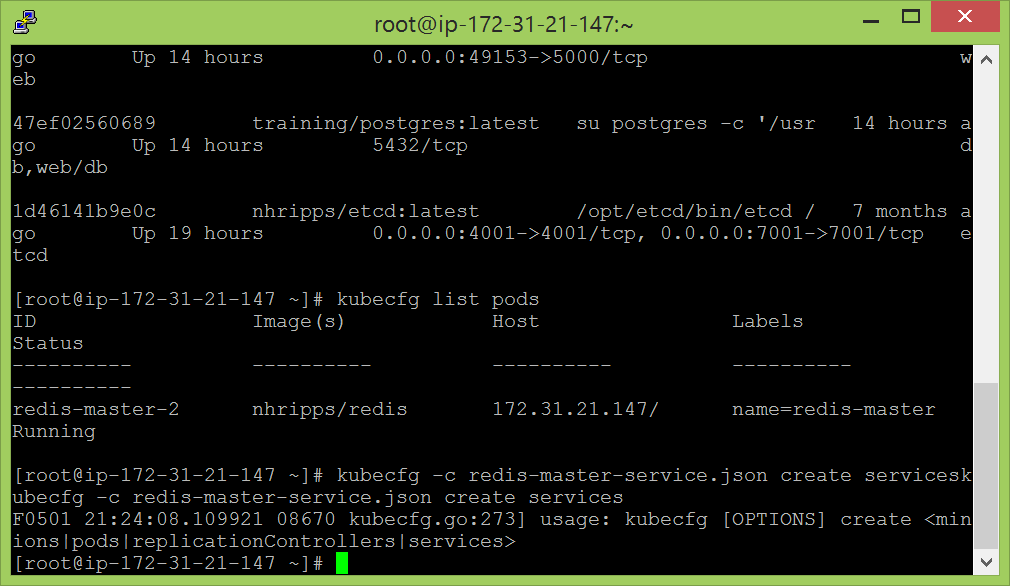
* To list all the pods that are in the local instance:

kubecfg list pods



* After creation of pods, the services have to be created. To create services:

kubecfg -c /root/guestbook/redis-master-service.json create services



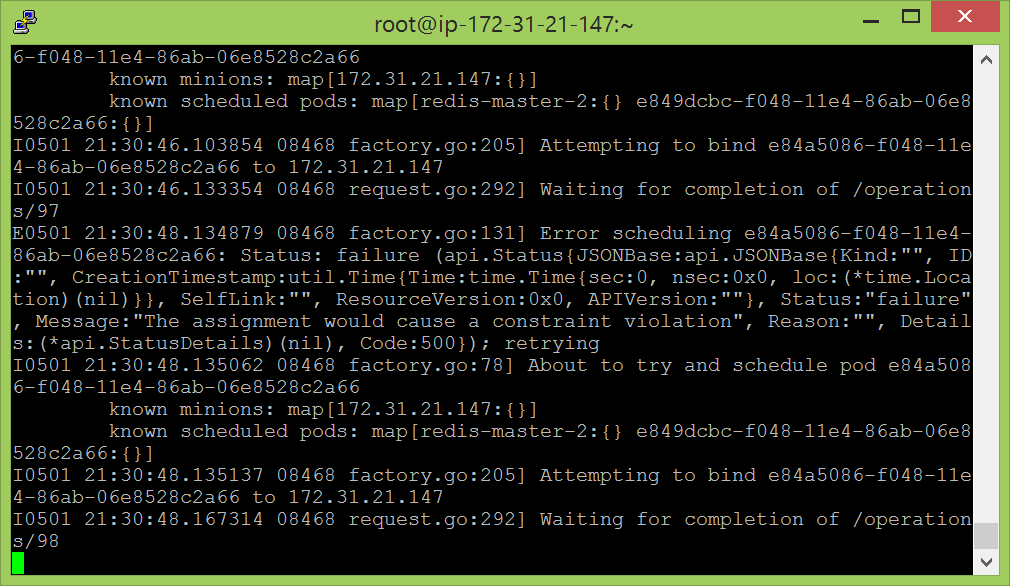


* After the services being created, the controllers are also to be created.

kubecfg -c /root/guestbook/redis-slave-controller.json create replicationControllers

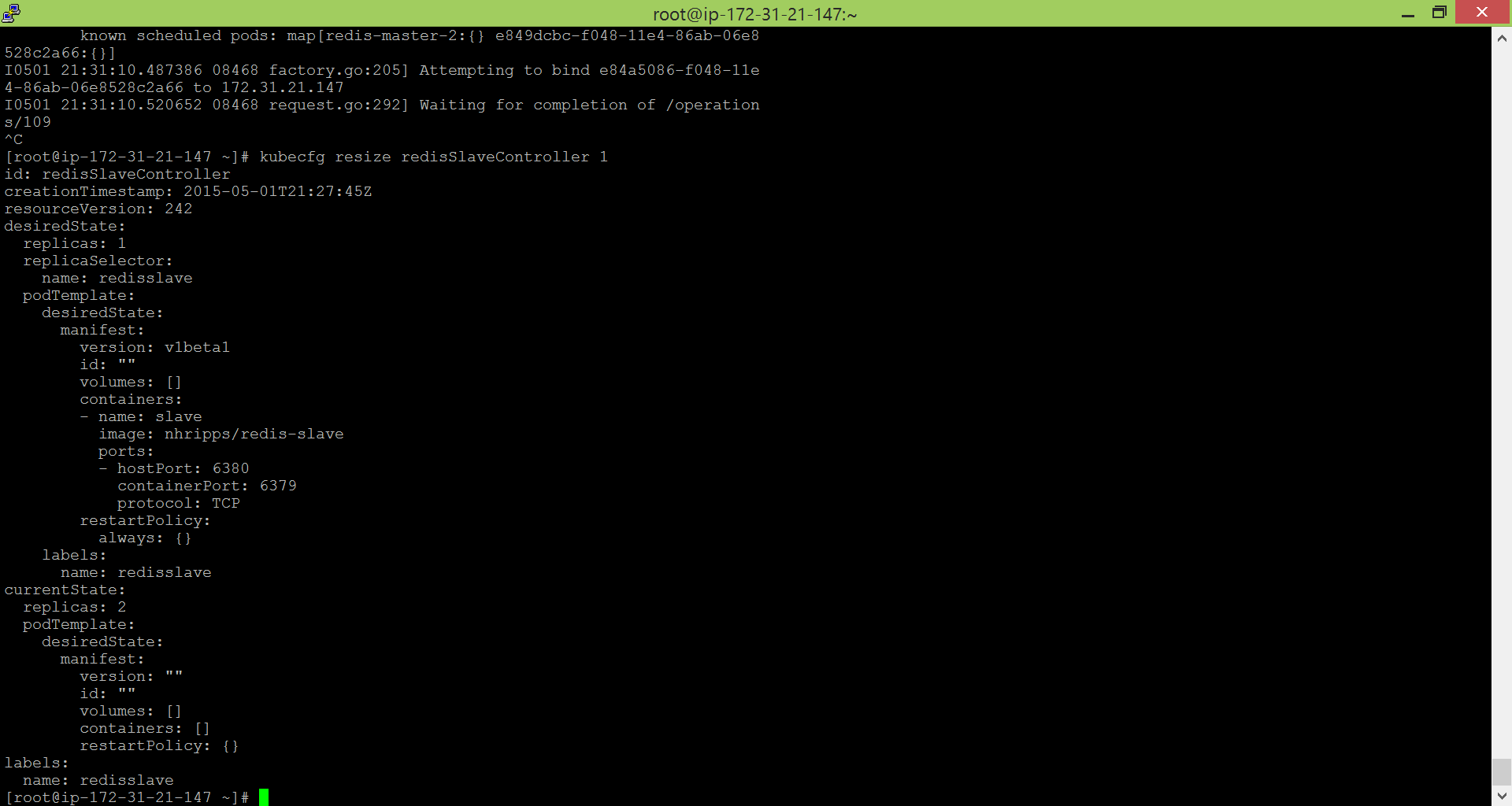
* To check the starting of the services. If the first service succeeds, then, the second one fails repeatedly.

tail -f /tmp/k8s-scheduler.log



* Press <CTRL>+C to exit the tail.
* The controllers which mage the services can be modified in size using:

kubecfg resize redisSlaveController 1



* To check what is happening with the containers:

docker ps

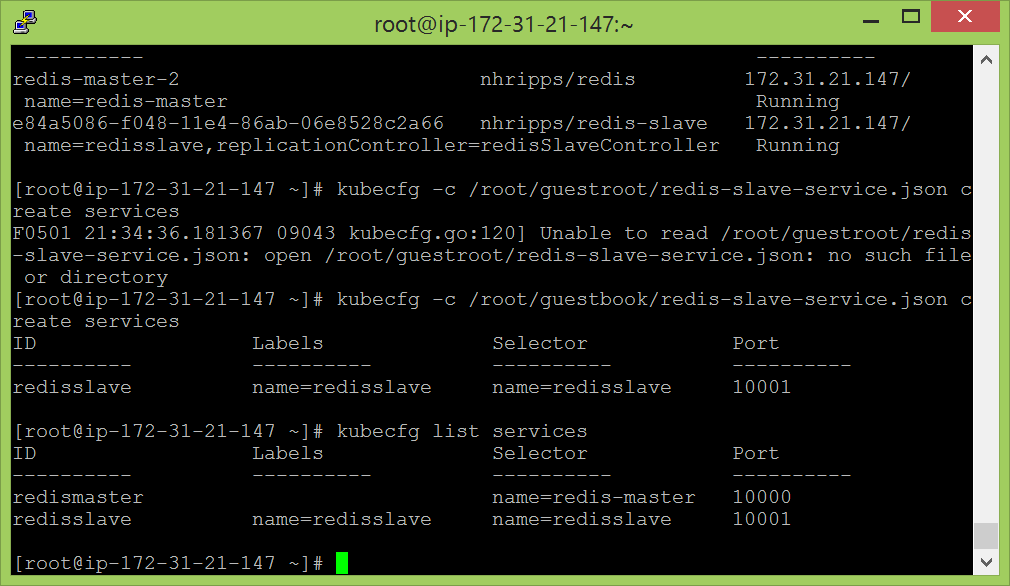


* Again listing all the pods which are present:

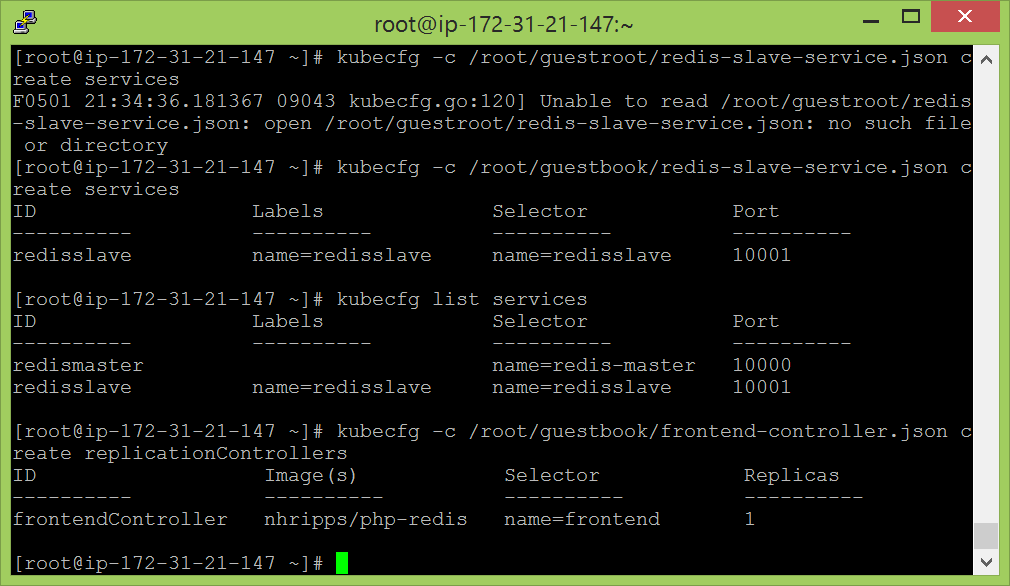
kubecfg list pods



* kubecfg -c redis-slave-service.json create services
* kubecfg list services



* kubecfg -c frontend-controller.json create replicationControllers



* A complete web application:

http:// 52.24.66.11:8000/

* Initially, the guestbook was not opening in the web browser. So, had to change the security groups in the Amazon Web services – Ec2. I created a new security group and added the inbound ports as all traffic. I bound this security group with my instance and ran the putty again. Now, open the web browser and used my public ip to open the link. It worked.

