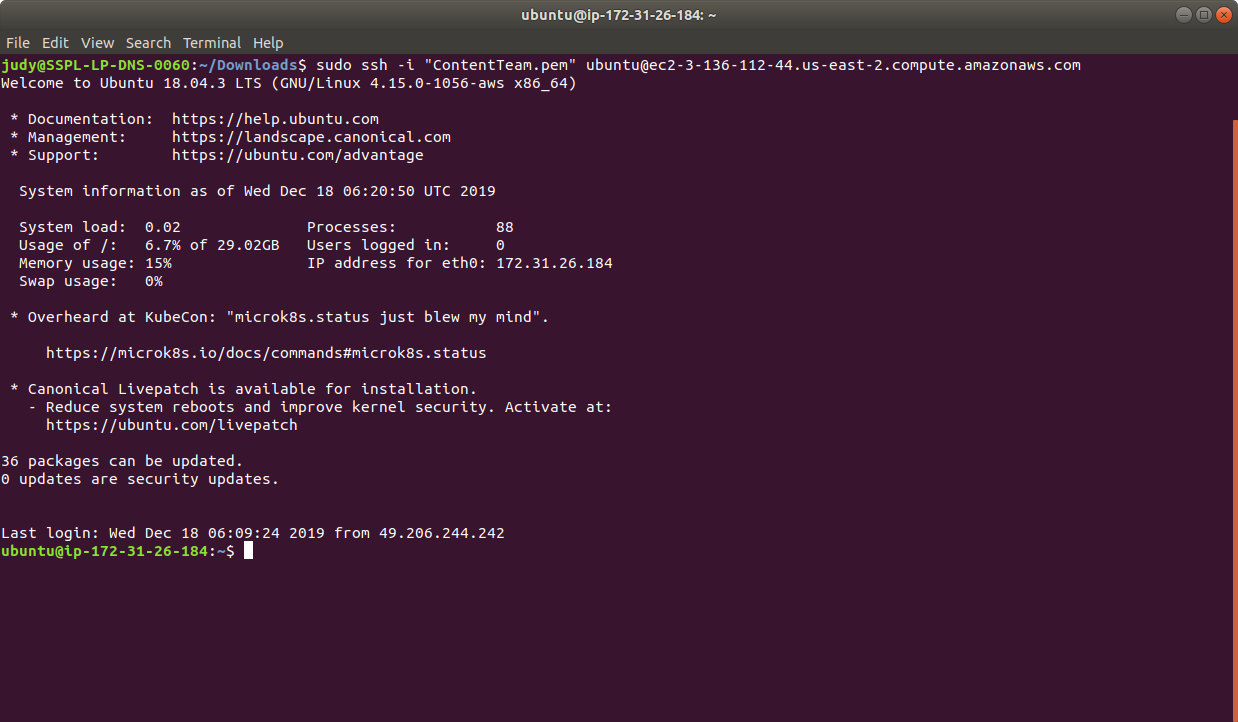
2.7 Configure a Pod with Custom Scheduler

In this demo, we will show you how to create a pod with certain resource requirements and limits.

* Login to your aws console
* Restart your ec2 instance and your EKS cluster nodes
* Open your terminal and SSH to the ec2 instance



* If you don’t have an existing EKS cluster, create one with the command, **eksctl create cluster --name=myeks-cluster --nodes=2 --region=us-east-2**
* Create the deployment specified in the config in the Kubernetes cluster. For that, create a yaml file as shown below:

**cat > my-scheduler.yaml**

* Let’s create the deployment config that manages a ReplicaSet which in turn manages the pods, thereby making the scheduler resilient to failures as shown in screen below:

apiVersion: v1

kind: ServiceAccount

metadata:

name: my-scheduler

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: my-scheduler-as-kube-scheduler

subjects:

- kind: ServiceAccount

name: my-scheduler

namespace: kube-system

roleRef:

kind: ClusterRole

name: system:kube-scheduler

apiGroup: rbac.authorization.k8s.io

---

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

component: scheduler

tier: control-plane

name: my-scheduler

namespace: kube-system

spec:

selector:

matchLabels:

component: scheduler

tier: control-plane

replicas: 1

template:

metadata:

labels:

component: scheduler

tier: control-plane

version: second

spec:

serviceAccountName: my-scheduler

containers:

- command:

- /usr/local/bin/kube-scheduler

- --address=0.0.0.0

- --leader-elect=false

- --scheduler-name=my-scheduler

image: gcr.io/my-gcp-project/my-kube-scheduler:1.0

livenessProbe:

httpGet:

path: /healthz

port: 10251

initialDelaySeconds: 15

name: kube-second-scheduler

readinessProbe:

httpGet:

path: /healthz

port: 10251

resources:

requests:

cpu: '0.1'

securityContext:

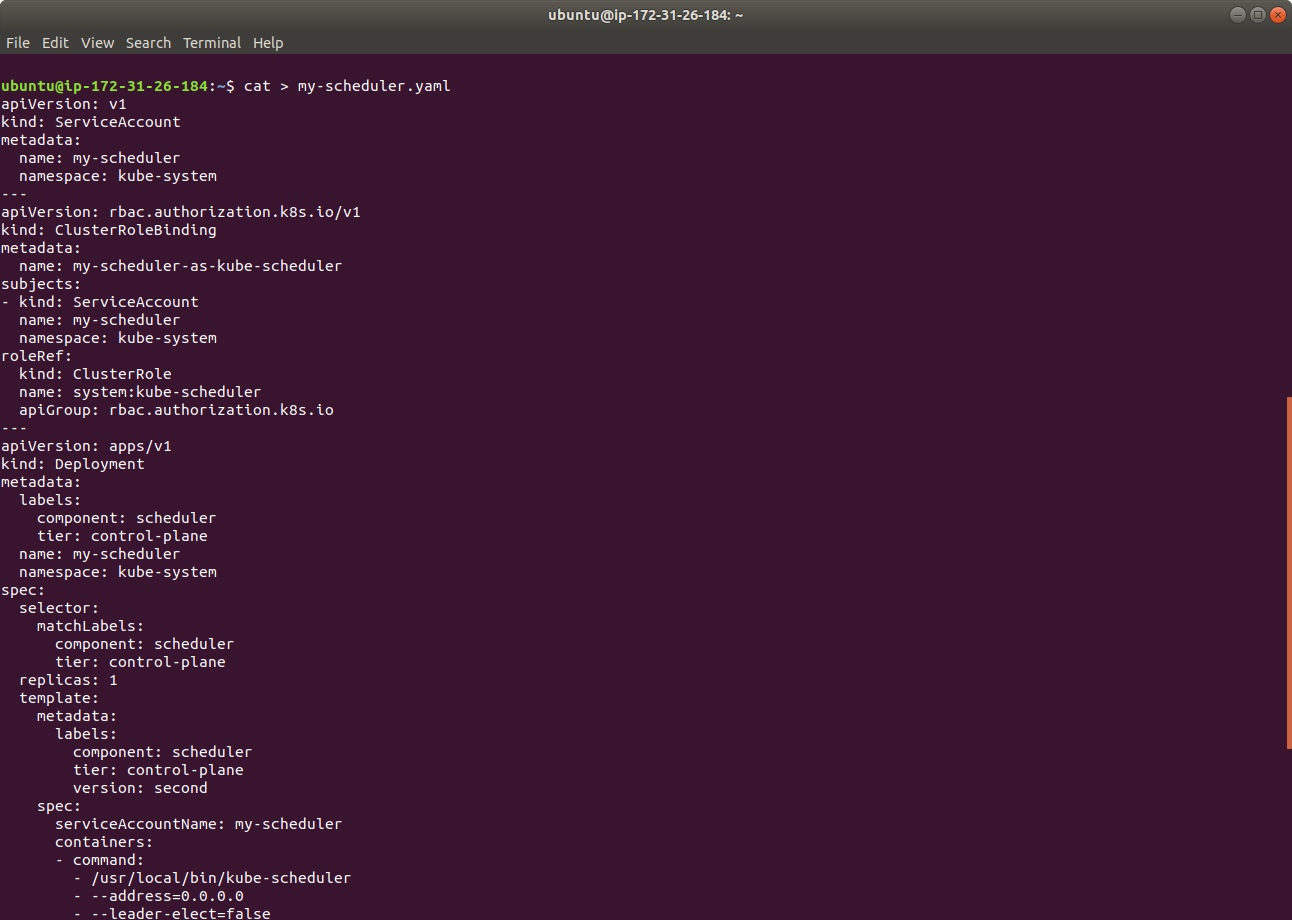
privileged: false

volumeMounts: []

hostNetwork: false

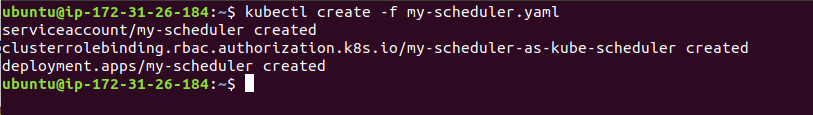
hostPID: false

volumes: []



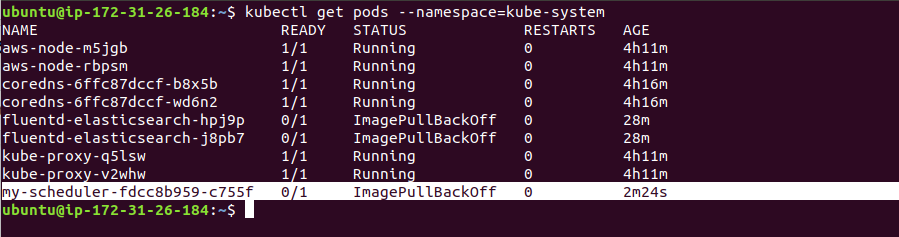
* Now that we have a config file that creates the deployment as a custom scheduler, use the kubectl command to run your scheduler in a Kubernetes cluster as shown below:

**kubectl create -f my-scheduler.yaml**



* As the next step, let’s verify that the scheduler pod is running as shown below:

**kubectl get pods --namespace-kube-system**



* This is how we can schedule a pod in the Kubernetes Engine.
* Run **eksctl delete cluster --name=myeks-cluster** to delete the cluster.