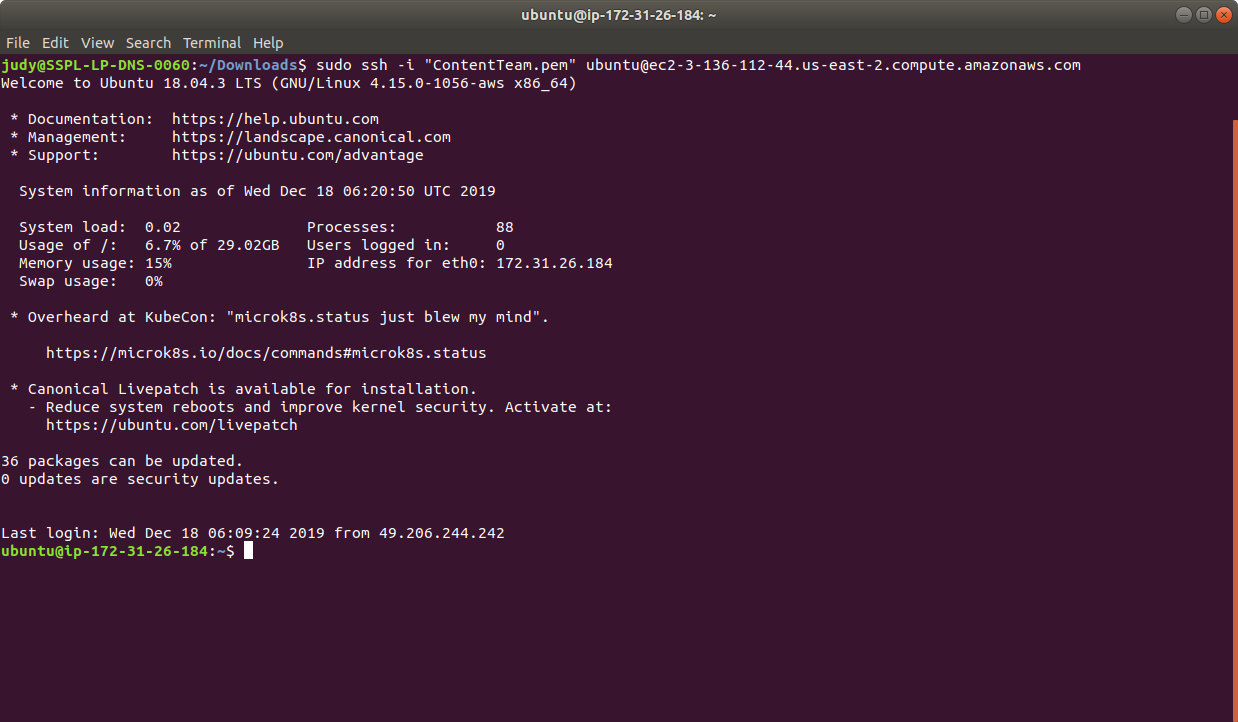
2.2 Labels and Selectors in ReplicaSet 

In this demo, we will show you how to define labels and selectors in a ReplicaSet.

* 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 a yaml file named replica.yaml file with the command **cat > replica.yaml**. We will create a simple frontend nginx app with 3 replicas.
* Enter the following to the file and save it:

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: frontend

labels:

app: guestbook

tier: frontend

spec:

# modify replicas according to your case

replicas: 3

selector:

matchLabels:

tier: frontend

template:

metadata:

labels:

tier: frontend

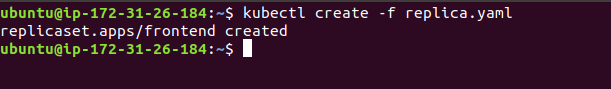
spec:

containers:

- name: php-redis

image: gcr.io/google\_samples/gb-frontend:v3

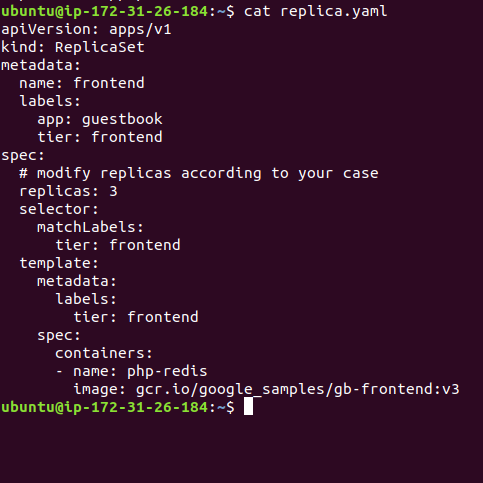
* Create the ReplicaSet using the command **kubectl create -f replica.yaml**



* Let’s now check the labels of the created ReplicaSet.   
  Labels are key/value pairs that are attached to an object, such as pods. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. An example of this is the yaml labels shown below:

labels:

tier: frontend



* Via a label selector, the client/user can identify a set of objects. The label selector is the core grouping primitive in Kubernetes. One of the use-case examples is the use of pods to specify node selection criteria. For example: the sample pod can select nodes with the label accelerator=nvidia-tesla-p100.
* This is how we define labels and selectors for a ReplicaSet in the Kubernetes Engine.