**Exercise 14a**

*Create a Kubernetes Cluster in DigitalOcean and Deploy an app*

**Prior Knowledge**

Unix Command Line Shell

YAML

**Learning Objectives**

Introduction to Kubernetes

**Software Requirements**

Browser

kubectl

**Overview**

In this exercise we are going to sign up to Digital Ocean to get some free credit, then instantiate a Kubernetes cluster in DO, then install an app onto the kubernetes cluster. Finally we will do some monitoring.   
  
There is a follow up lab that then installs cassandra onto the cluster.

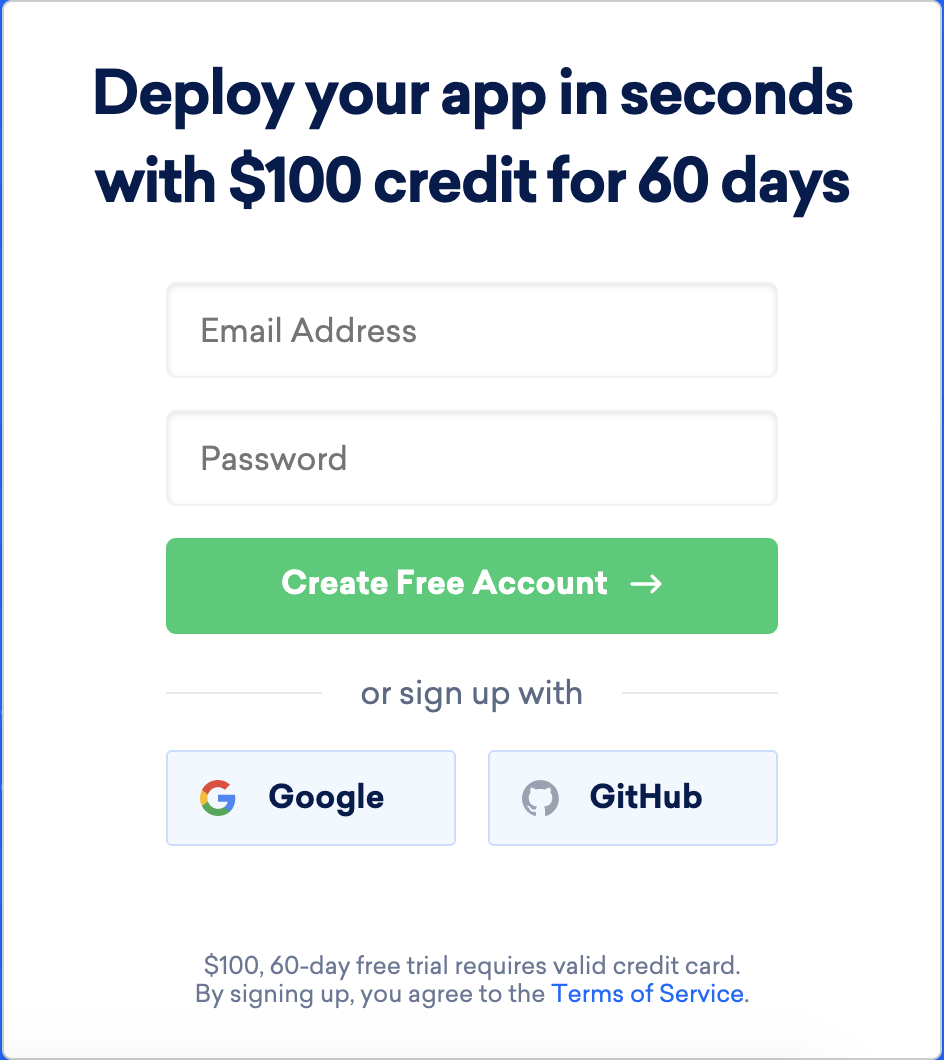
**PART A: SIGN UP WITH DIGITAL OCEAN AND START A K8S CLUSTER**

1. Sign up and get a free Digital Ocean account. At the time of writing, Digital Ocean is offering $100 free credit (you will need to provide a credit card). However, even if you already have an account, this exercise should cost less than $1 assuming you kill off the kubernetes cluster when you are done.

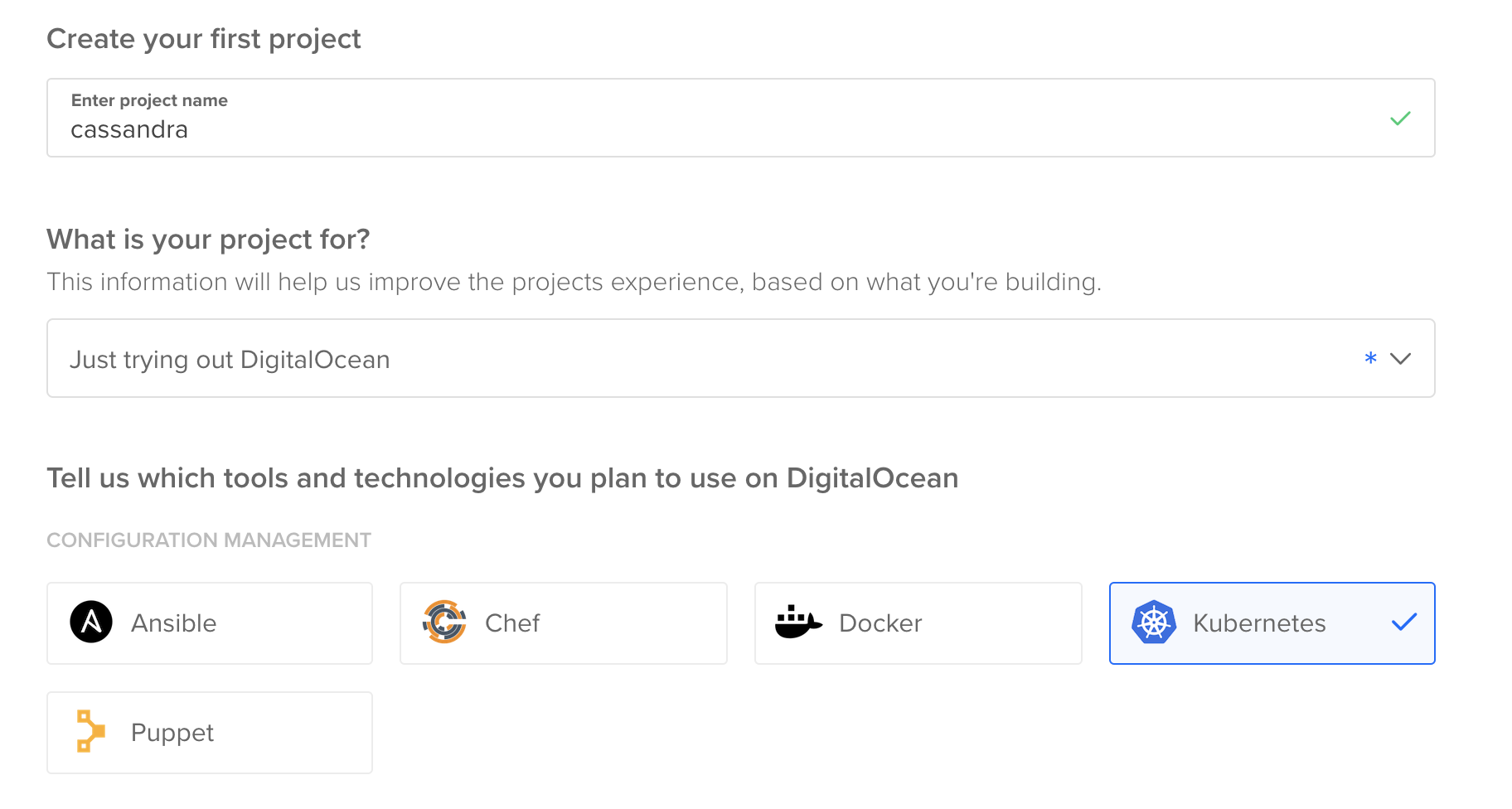
If you already have a DO account and credit, skip this step.

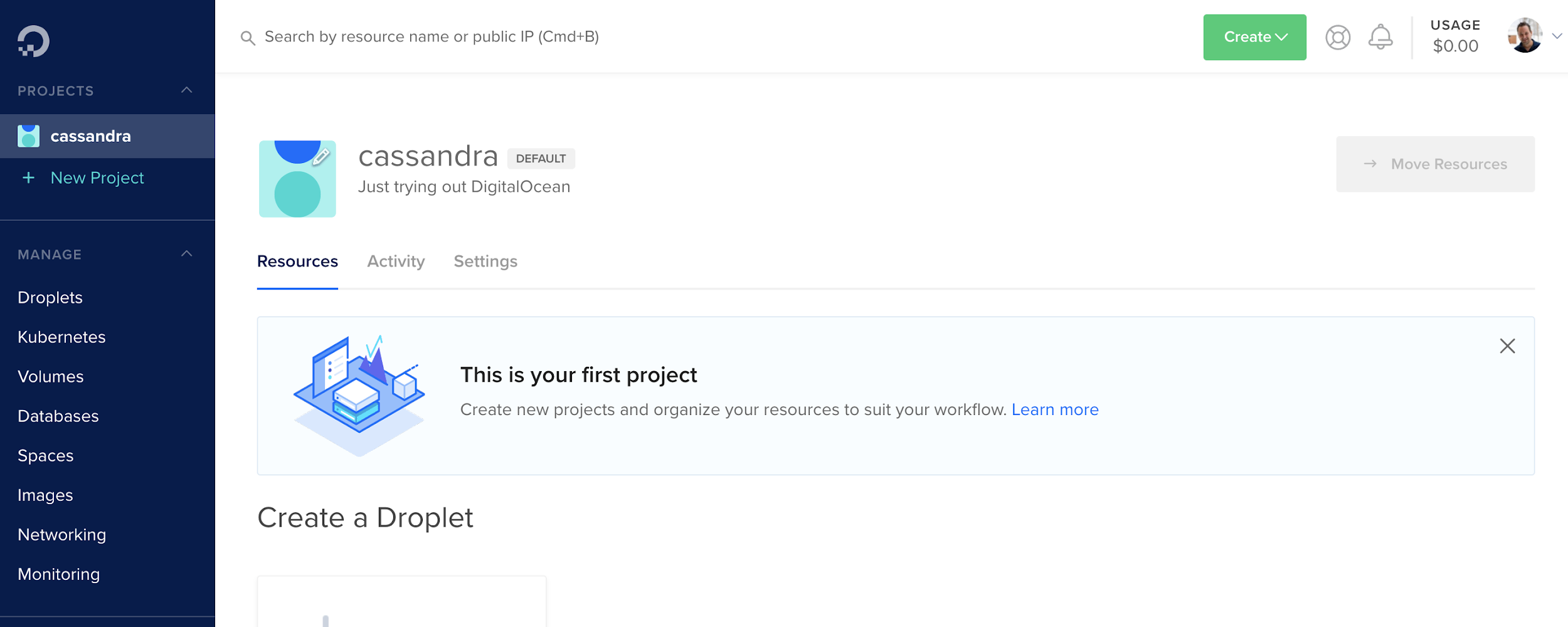
* 1. go to: <https://try.digitalocean.com/developerBrand/>

You should see:



* 1. Sign up or use an existing Google or Github account to login.
  2. Enter your billing details
  3. You should see something like:  
     
  4. Give your project the name “cassandra”:



* 1. Select “Just trying out DigitalOcean” and tick Kubernetes
  2. Click “Start” at the bottom of the page
  3. You should see something like:  
     

1. Before we create the Kubernetes cluster, we’d like to update the kubernetes CLI tool. By default the one in the Ubuntu package repo is out of date. We can fix that by doing the following commands (taken from <https://kubernetes.io/docs/tasks/tools/install-kubectl/>)  
     
   sudo apt-get update

sudo apt-get install -y apt-transport-https gnupg2

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl

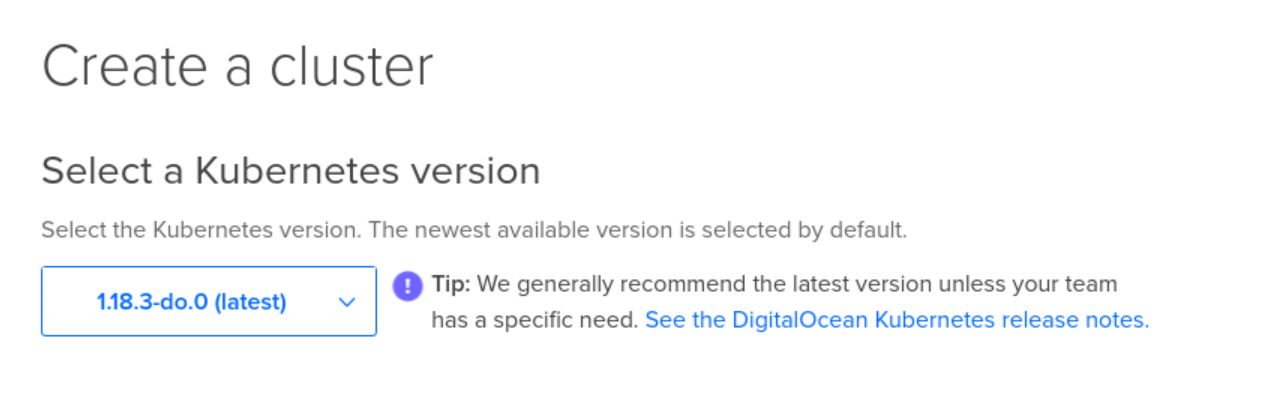
Your version should now be 1.18.x

1. Go back to your Firefox / DigitalOcean window.
2. Click on Kubernetes in the left hand side. You should see:



Now click **Create a Kubernetes Cluster**

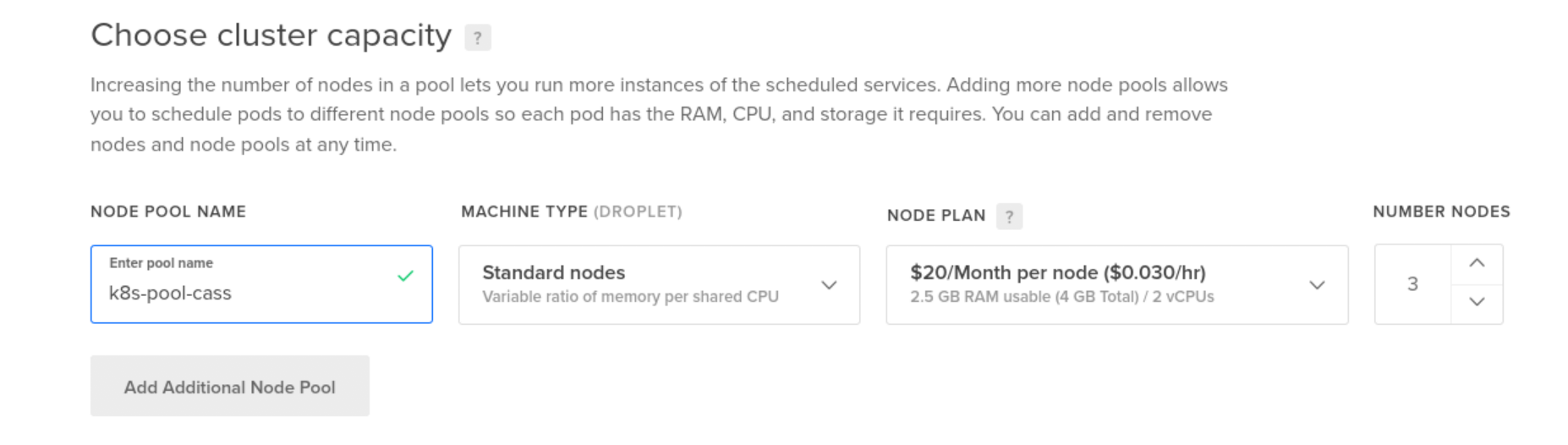
Choose Kubernetes version 1.18.x

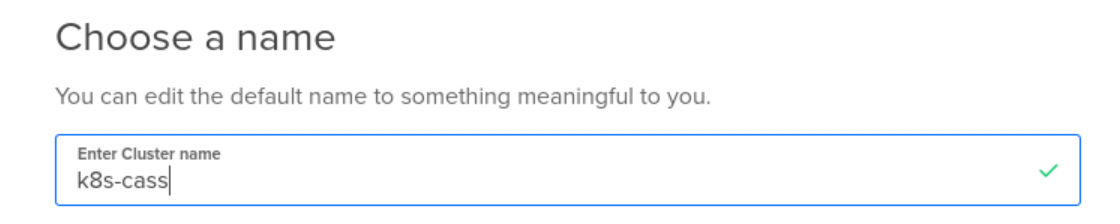


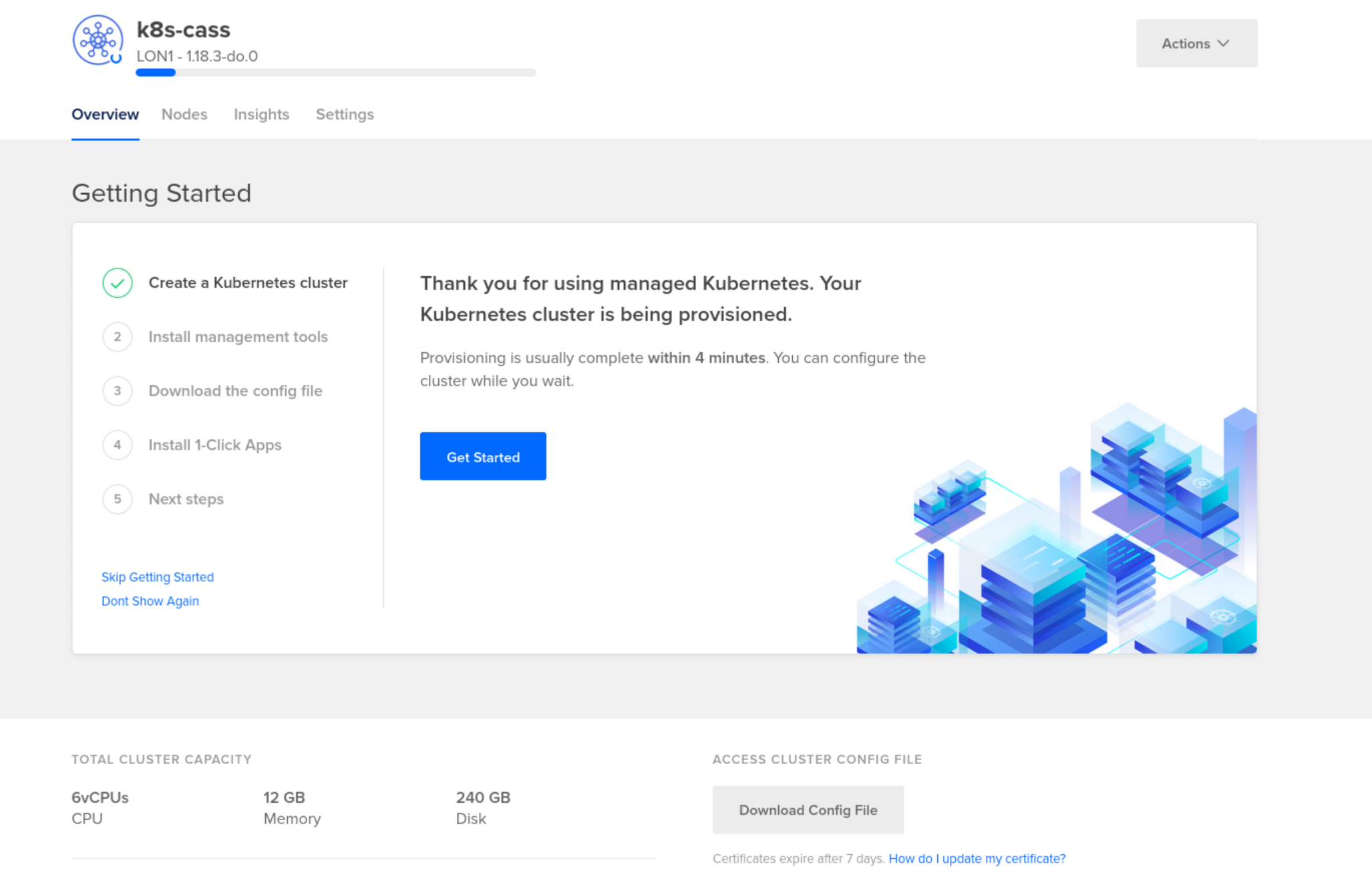
(The kubectl client version and server version should be within one major revision of each other. e.g. 1.17 and 1.18 are compatible but 1.16 and 1.18 might not be).

1. Choose your nearest datacentre (e.g. London)
2. Choose the following:  
   3 nodes  
   Standard Nodes

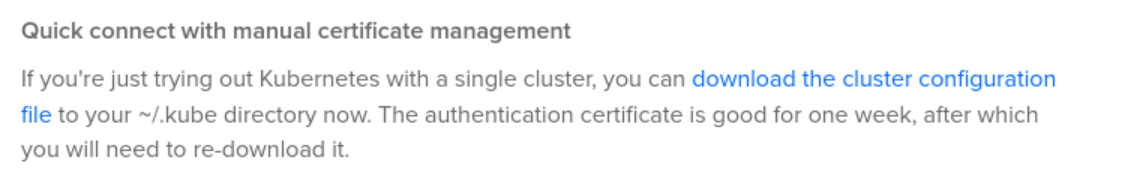
$20/month per node (2.5Gb RAM / 2 vcpus)



1. Change the name to k8s-cass  
   
2. Click **Create Cluster**
3. You should see:

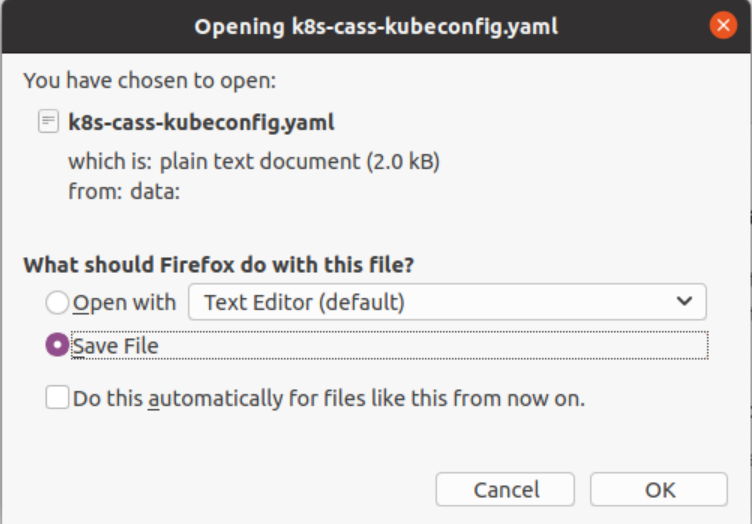


1. There is a nice “checklist” of actions you can do with your cluster.   
   Click on #2. We already have the management tools downloaded (at least kubectl), so we can **Continue**
2. If you are going to use DO Kubernetes a lot, I suggest you read the section on using their doctl CLI tool. However, since I am more interested in you learning about kubernetes right now, I’d like you to follow the “manual” approach:



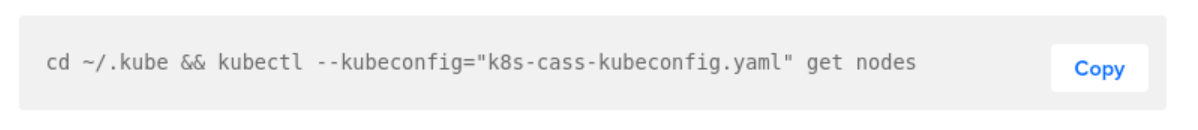
Click on “download the cluster configuration file”

**Save File**



1. Open a terminal window and type:

mkdir ~/.kube

mv ~/Downloads/k8s-cass-kubeconfig.yaml ~/.kube/  
  
Then you will see the command shown in the Web UI and execute that:  
  


cd ~/.kube && kubectl --kubeconfig="k8s-cass-kubeconfig.yaml" get nodes

You should see something like:

NAME STATUS ROLES AGE VERSION

k8s-pool-cass-3o8i7 Ready <none> 35m v1.18.3

k8s-pool-cass-3o8ic Ready <none> 34m v1.18.3

k8s-pool-cass-3o8iu Ready <none> 34m v1.18.3

1. We want to use this config file all the time (without needing to do --kubeconfig="k8s-cass-kubeconfig.yaml" on every command):  
     
   export KUBECONFIG=~/.kube/k8s-cass-kubeconfig.yaml

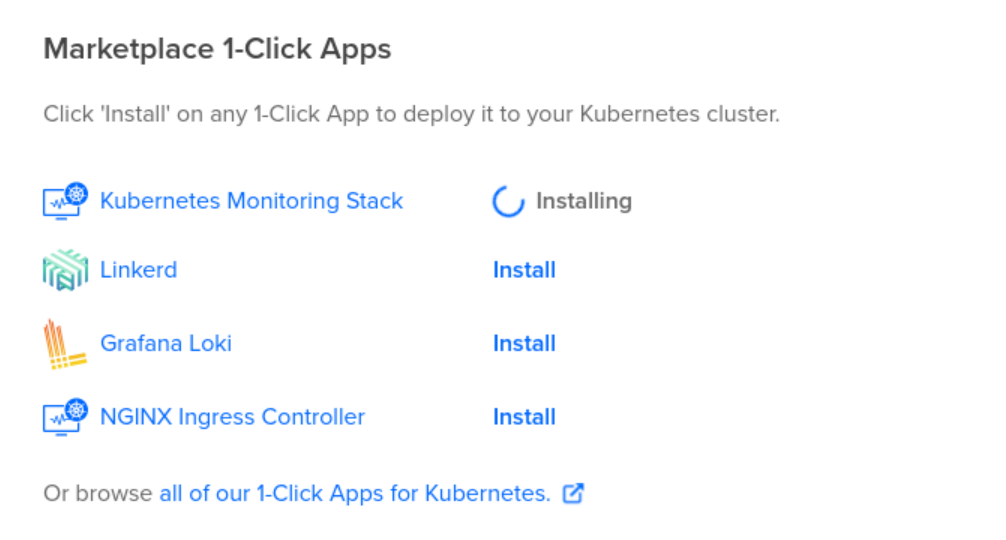
(There are also other things we can do, but this works fine)

1. Check it works:

kubectl get all

You should see something like:  
  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/kubernetes ClusterIP 10.245.0.1 <none> 443/TCP 45m

1. Back in the Web UI, go to part 3 of the Getting Started, and install the **Kubernetes Monitoring Stack**  
   

**PART B: INSTALL AN APP INTO K8S**

1. Let’s deploy a sample app:  
     
   This app <https://github.com/paulbouwer/hello-kubernetes> is a great starting place to check out Kubernetes:

cd ~

git clone https://github.com/paulbouwer/hello-kubernetes.git

cd ~/hello-kubernetes

1. Now let’s apply (install) this app into kubernetes:

kubectl apply -f yaml/hello-kubernetes.yaml

1. The install will be quick, but it might take a while to allocate an external address:

kubectl get all

NAME READY STATUS RESTARTS AGE

pod/hello-kubernetes-594f6f475f-4rksn 0/1 ContainerCreating 0 5s

pod/hello-kubernetes-594f6f475f-h25gz 0/1 ContainerCreating 0 5s

pod/hello-kubernetes-594f6f475f-sjd6h 0/1 ContainerCreating 0 5s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/hello-kubernetes LoadBalancer 10.245.249.202 <pending> 80:30816/TCP 5s

service/kubernetes ClusterIP 10.245.0.1 <none> 443/TCP 125m

NAME READY UP-TO-DATE AVAILABLE AGE

deployment.apps/hello-kubernetes 0/3 3 0 5s

NAME DESIRED CURRENT READY AGE

replicaset.apps/hello-kubernetes-594f6f475f 3 3 0 5s

We are going to wait until everything is running (maybe a few minutes). When it’s ready it should look like this:  
  
NAME READY STATUS RESTARTS AGE

pod/hello-kubernetes-594f6f475f-4rksn 1/1 Running 0 5m2s

pod/hello-kubernetes-594f6f475f-h25gz 1/1 Running 0 5m2s

pod/hello-kubernetes-594f6f475f-sjd6h 1/1 Running 0 5m2s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/hello-kubernetes LoadBalancer 10.245.249.202 188.166.139.3 80:30816/TCP

service/kubernetes ClusterIP 10.245.0.1 <none> 443/TCP

NAME READY UP-TO-DATE AVAILABLE AGE

deployment.apps/hello-kubernetes 3/3 3 3 5m2s

NAME DESIRED CURRENT READY AGE

replicaset.apps/hello-kubernetes-594f6f475f 3 3 3 5m2s

While you are waiting, you can look at the YAML:

apiVersion: v1

kind: Service

metadata:

name: hello-kubernetes

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 8080

selector:

app: hello-kubernetes

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: hello-kubernetes

spec:

replicas: 3

selector:

matchLabels:

app: hello-kubernetes

template:

metadata:

labels:

app: hello-kubernetes

spec:

containers:

- name: hello-kubernetes

image: paulbouwer/hello-kubernetes:1.8

ports:

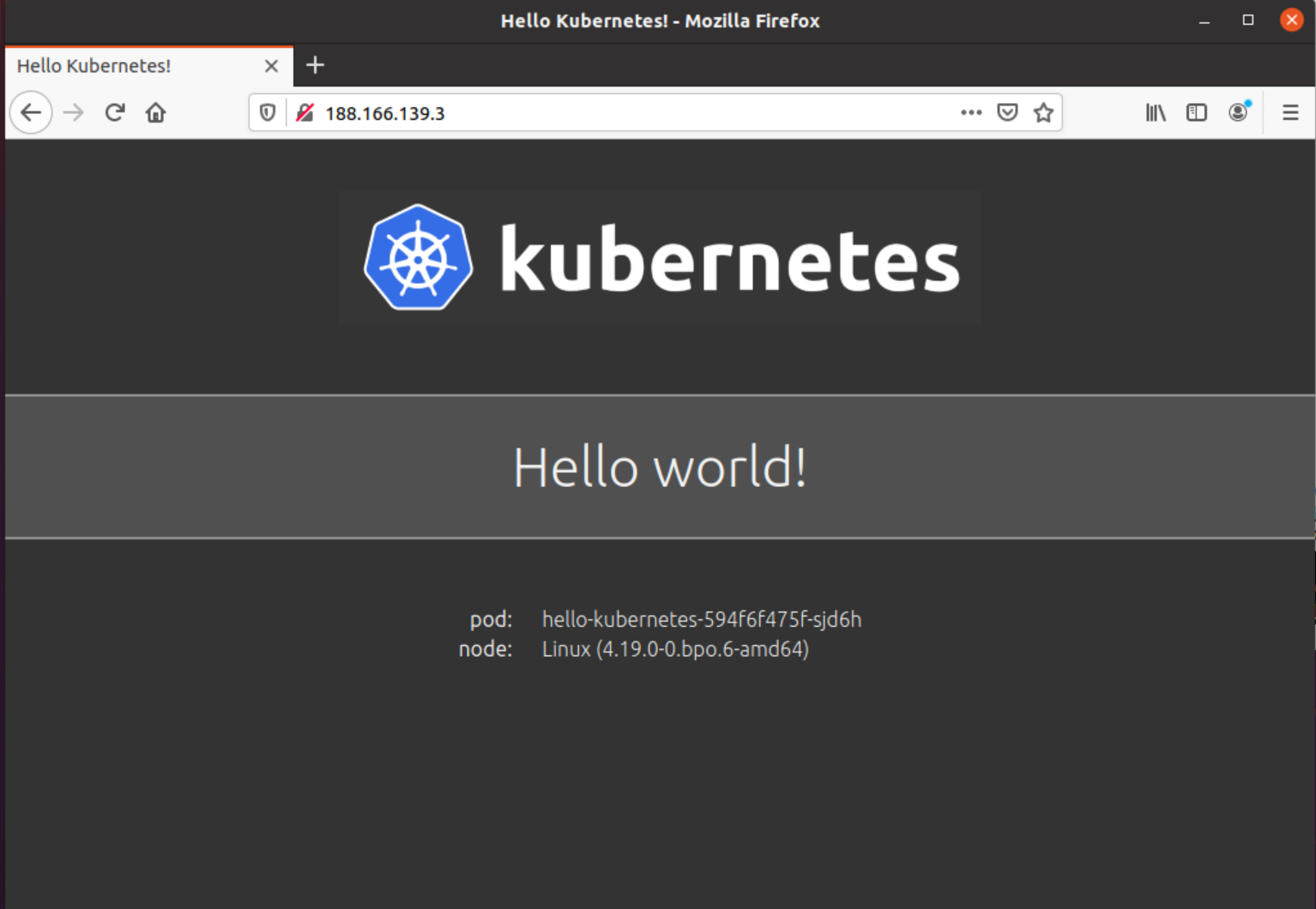
- containerPort: 8080

This basically defines a pod with 3 replicas containing a single container instance. There is then a load-balancer that balances load across the three replicas.

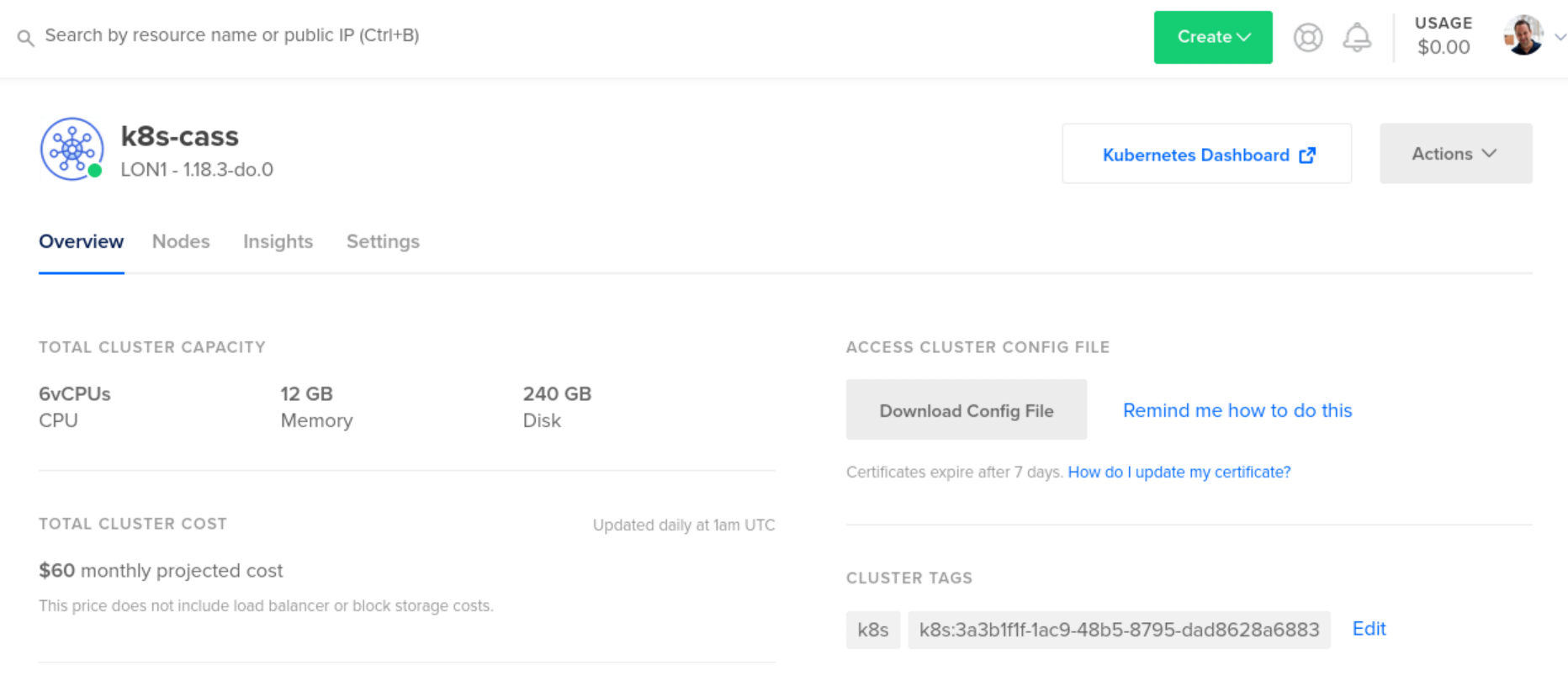
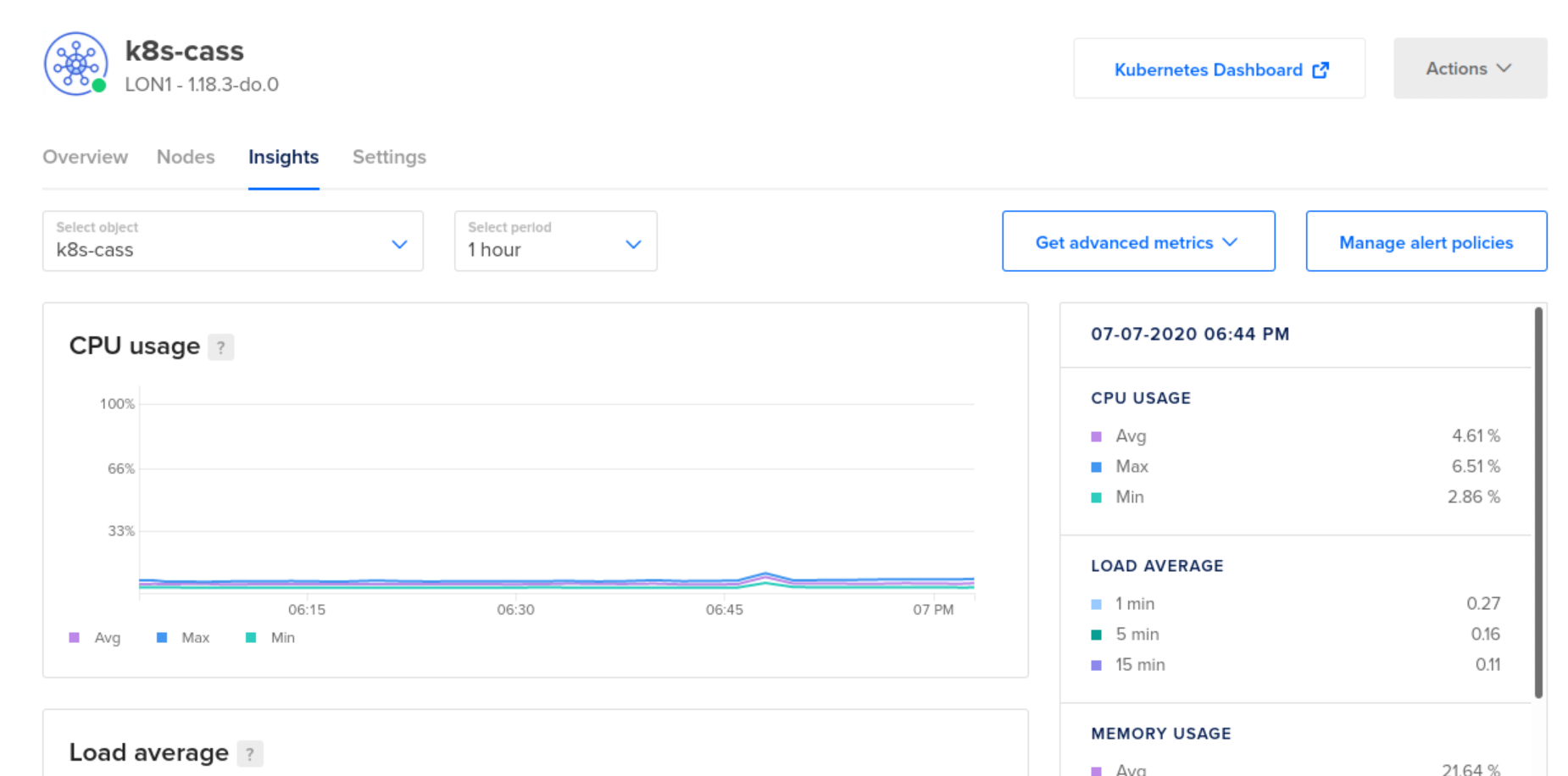
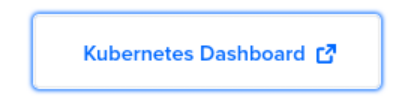
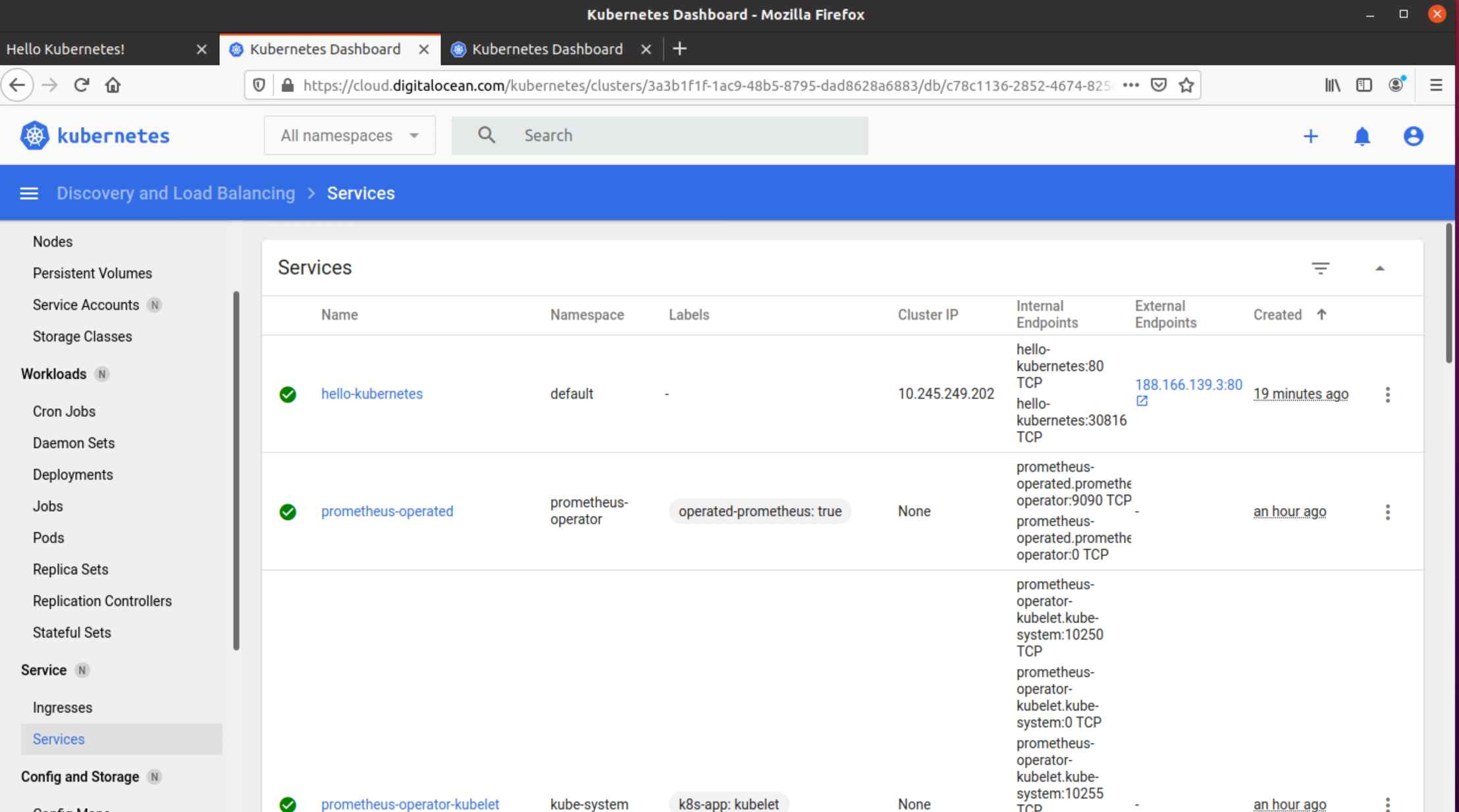
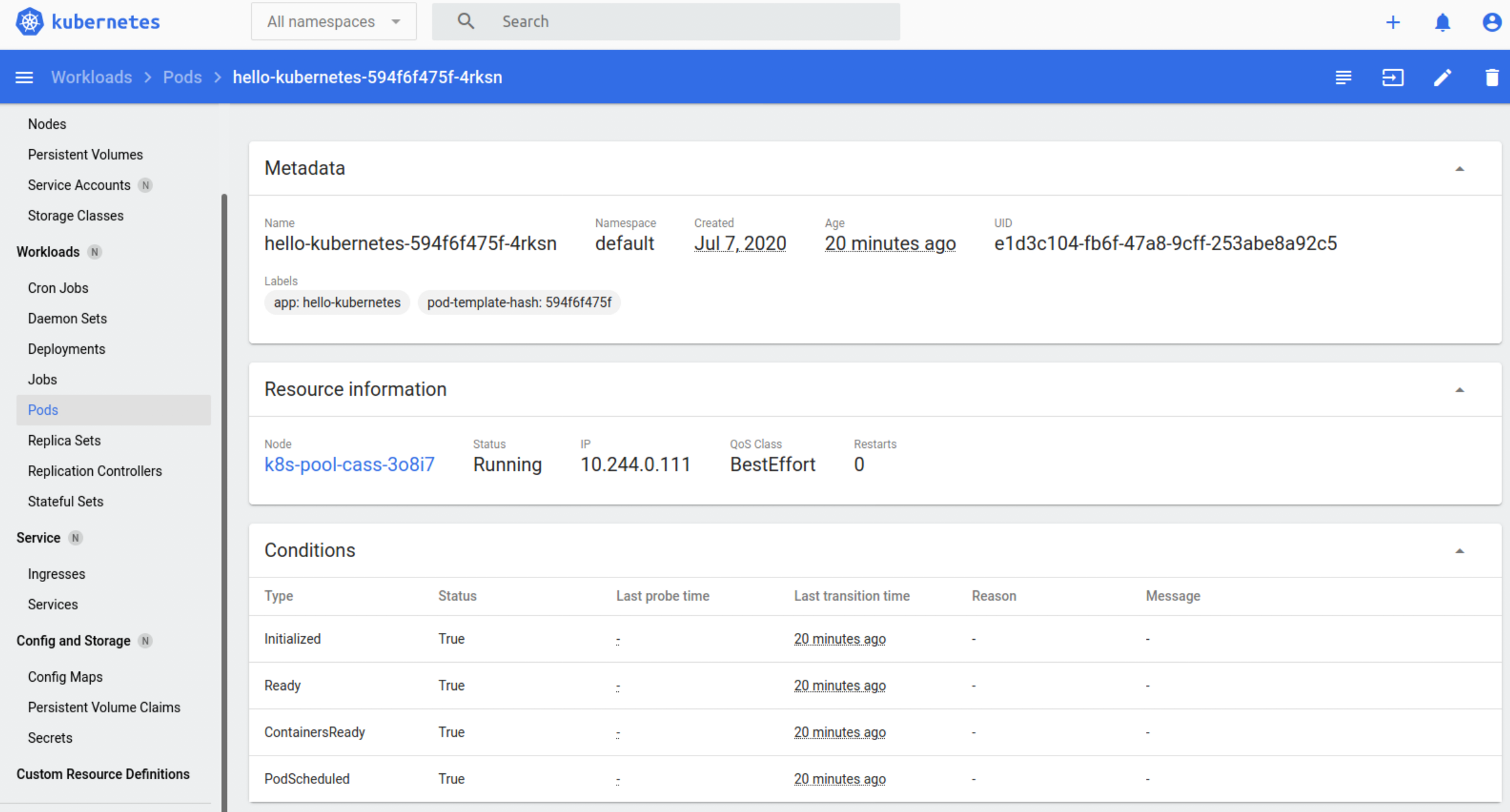
1. Now it should be running, get the external IP address:  
     
   kubectl get service hello-kubernetes

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

hello-kubernetes LoadBalancer 10.245.249.202 188.166.139.3 80:30816/TCP 11m

1. Go to the external IP address in your browser:  
   
2. Keep reloading and you should see the pod details change.
3. Congrats - you’ve deployed a k8s app.

**PART C: MONITORING**

1. We can go and monitor the system from the DigitalOcean web ui. Navigate to the cluster info page:  
     
   
2. Click on **Insights**   
   
3. You can see the system monitoring.
4. Click on 
5. Navigate to look at Services  
   
6. You can see a nice link to the external webpage of your app.
7. Browse pods and go look at a pod:  
   
8. If you are a command-line person instead, let’s try a more CLI-ish approach:

sudo snap install k9s

Warning: /snap/bin was not found in your $PATH. If you've not restarted your session since you

installed snapd, try doing that. Please see https://forum.snapcraft.io/t/9469 for

more details.

k9s 0.7.12 from Fernand Galiana (derailed) installed

Ignore the warning.

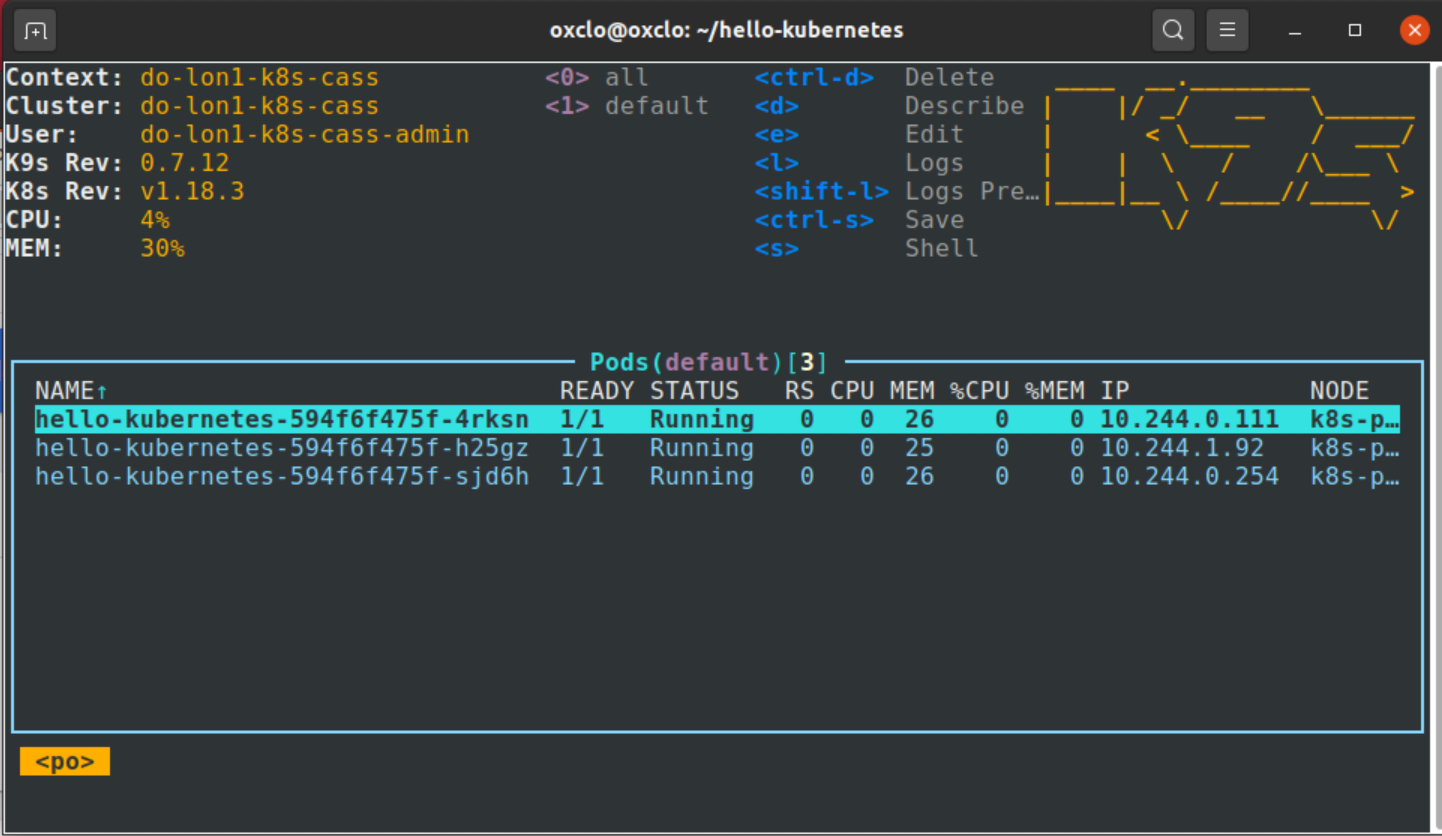
1. For some obscure reason we need to create a directory for the .

k9s config file:

mkdir ~/.k9s

1. Now start k9s:

k9s

You should see:  


1. This is an awesome tool. Hit enter twice to see the pod logs. Have a look at the docs here: <https://k9scli.io/>
2. Do you remember that we installed the Kubernetes 1-click monitoring. Let’s take a look at that.
3. We need to be able to access the pod containing Grafana:

Find the pod name with:

kubectl -n prometheus-operator get pods | grep \ prometheus-operator-grafana

You should see something like:  
prometheus-operator-grafana-cf6954699-xgklc 2/2 Running 0 113m

Copy that name into this:

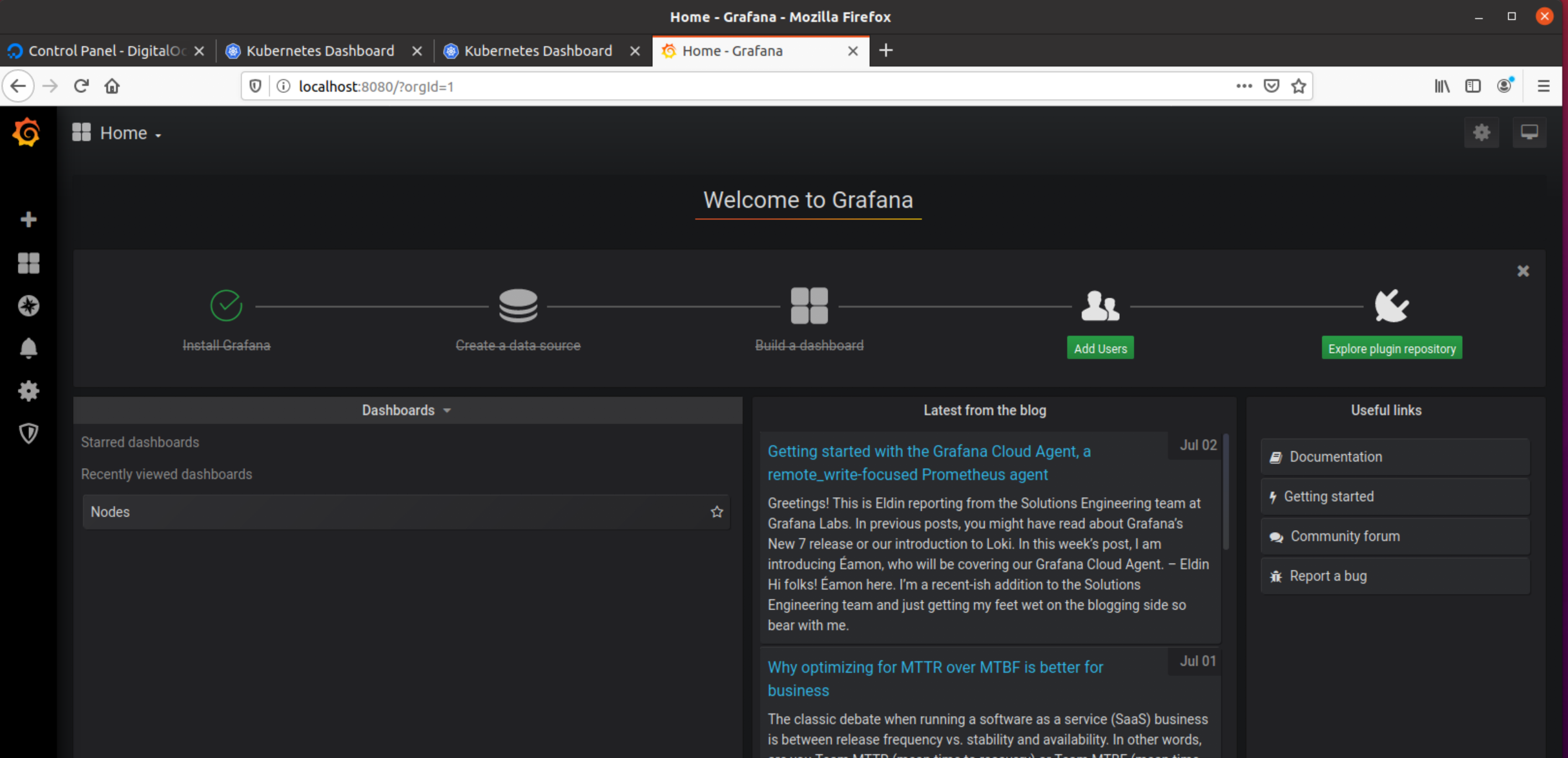
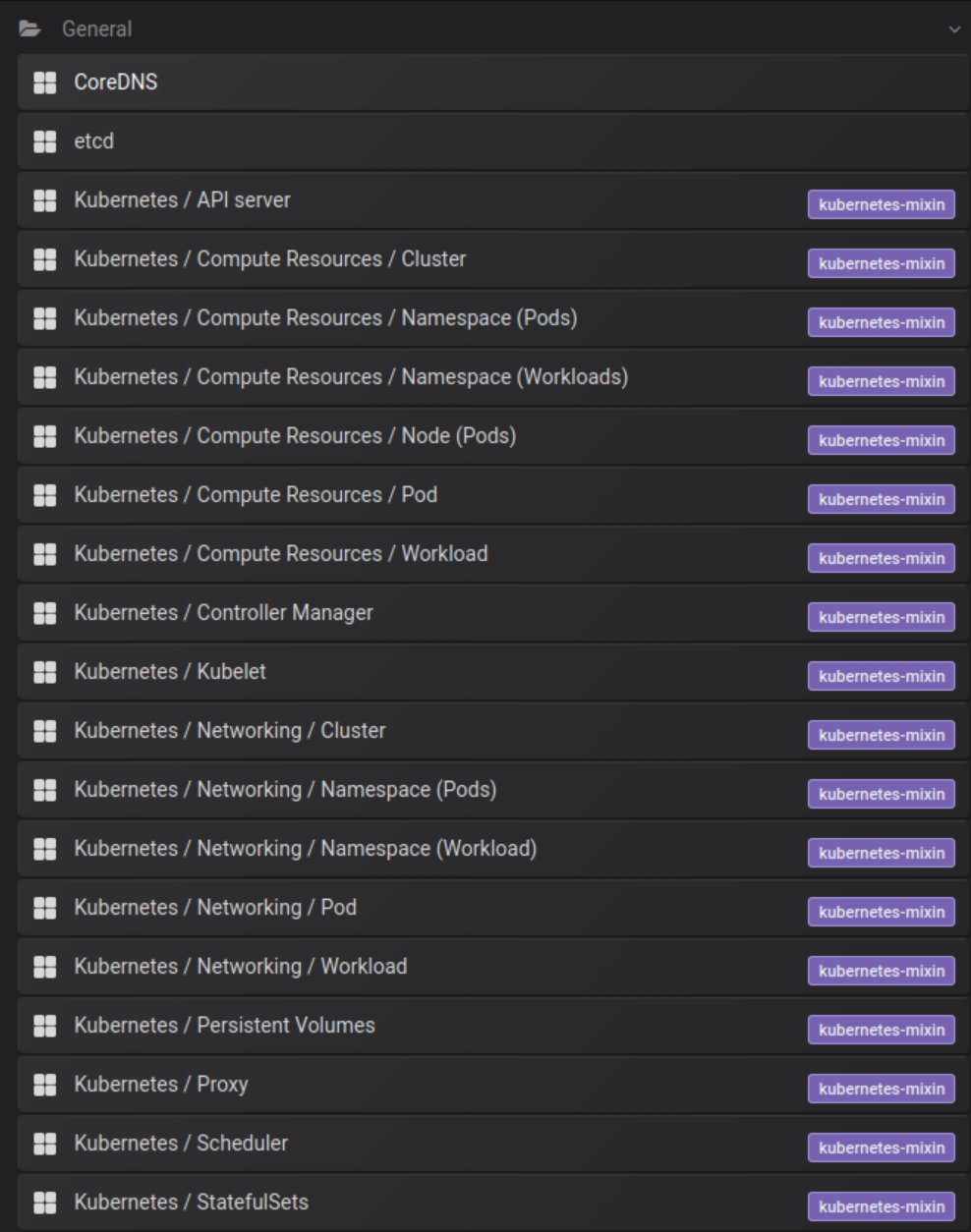
kubectl port-forward **prometheus-operator-grafana-cf6954699-xgklc** \

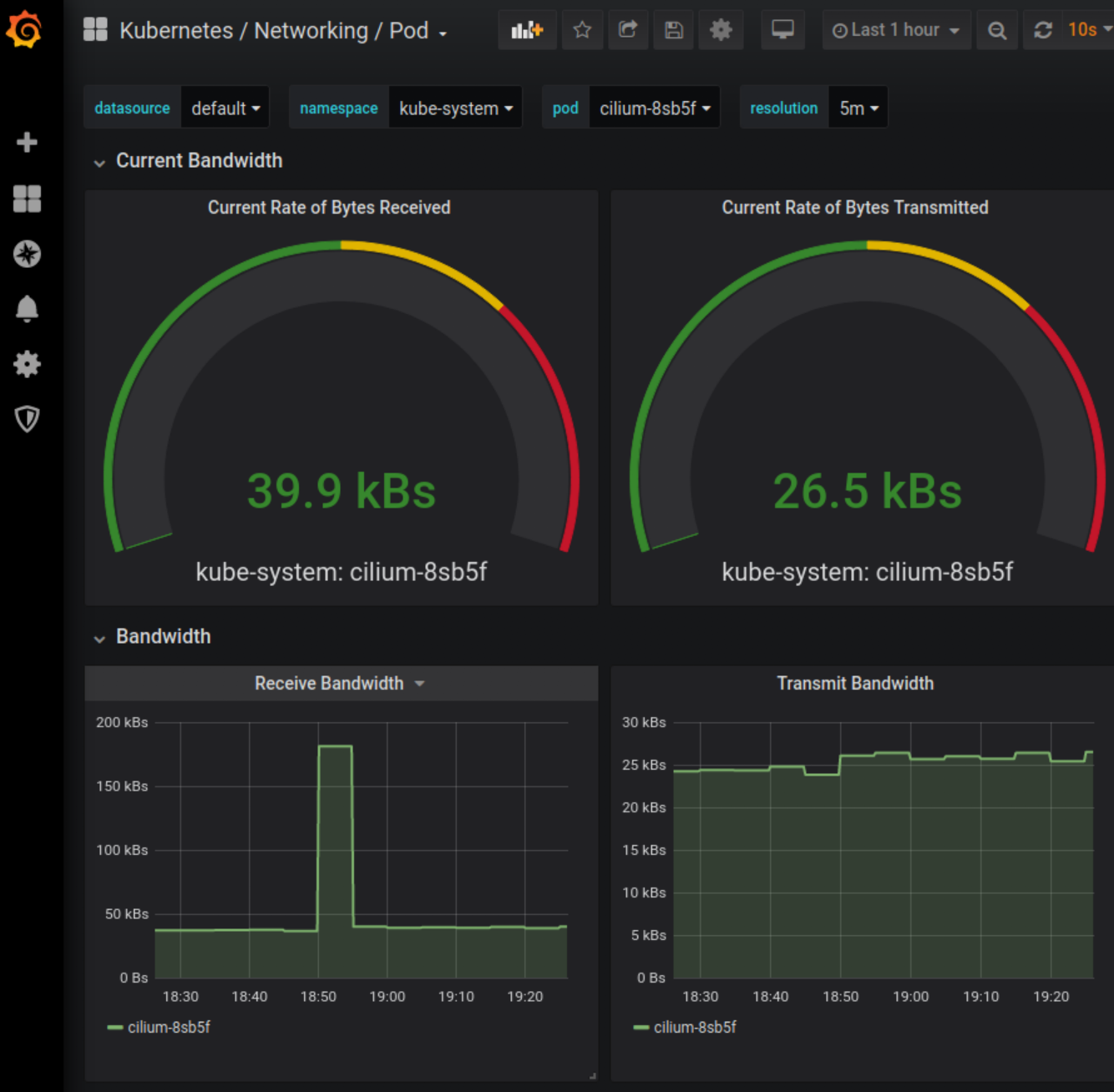
-n prometheus-operator 8080:3000

Changing the name to match yours.

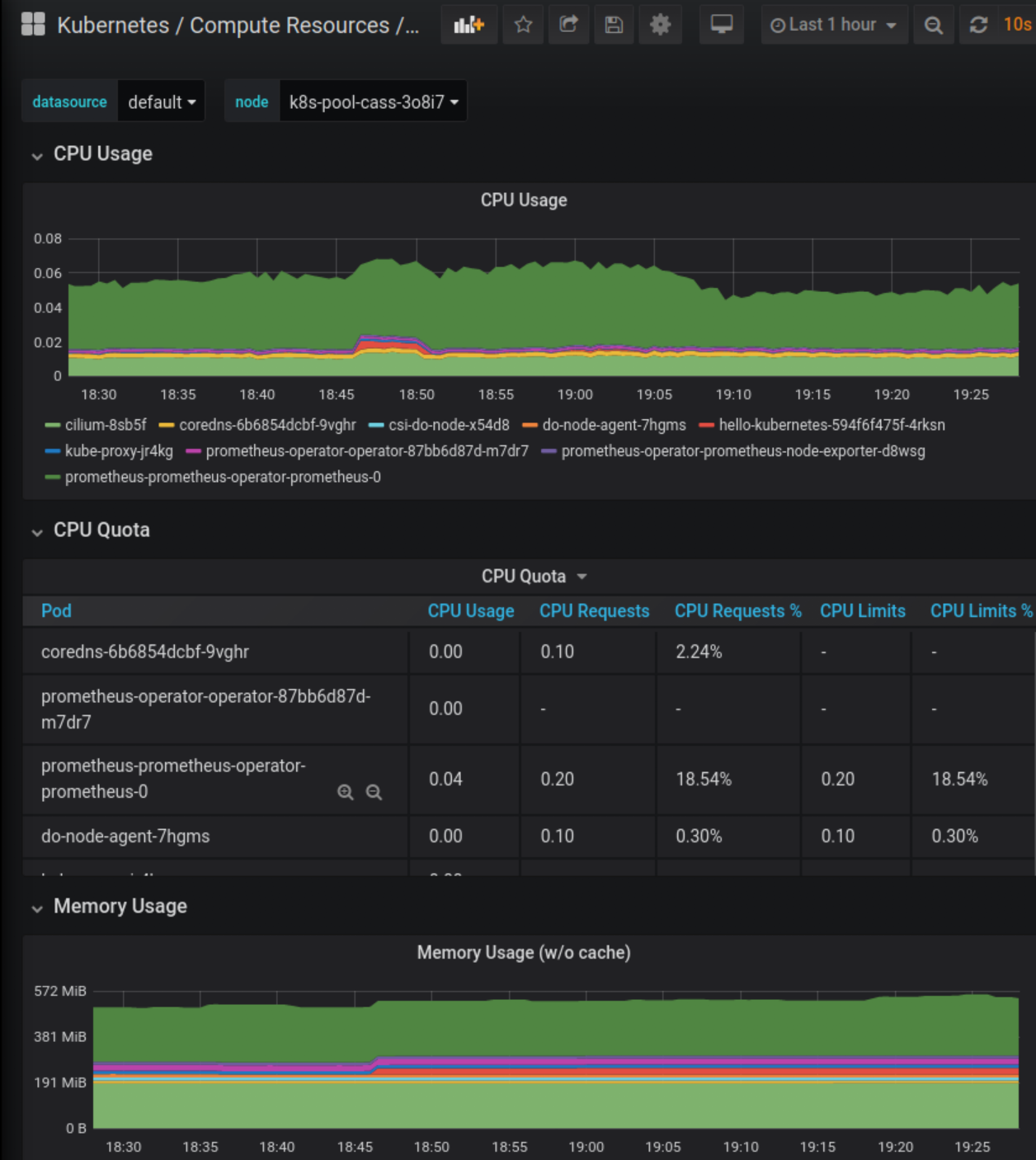
You should see:  
Forwarding from 127.0.0.1:8080 -> 3000

Forwarding from [::1]:8080 -> 3000

1. Now browse <http://localhost:8080> 
2. Under **Home** you will see lots of things you can look at:  
   
3. e.g. **Networking / Pod**

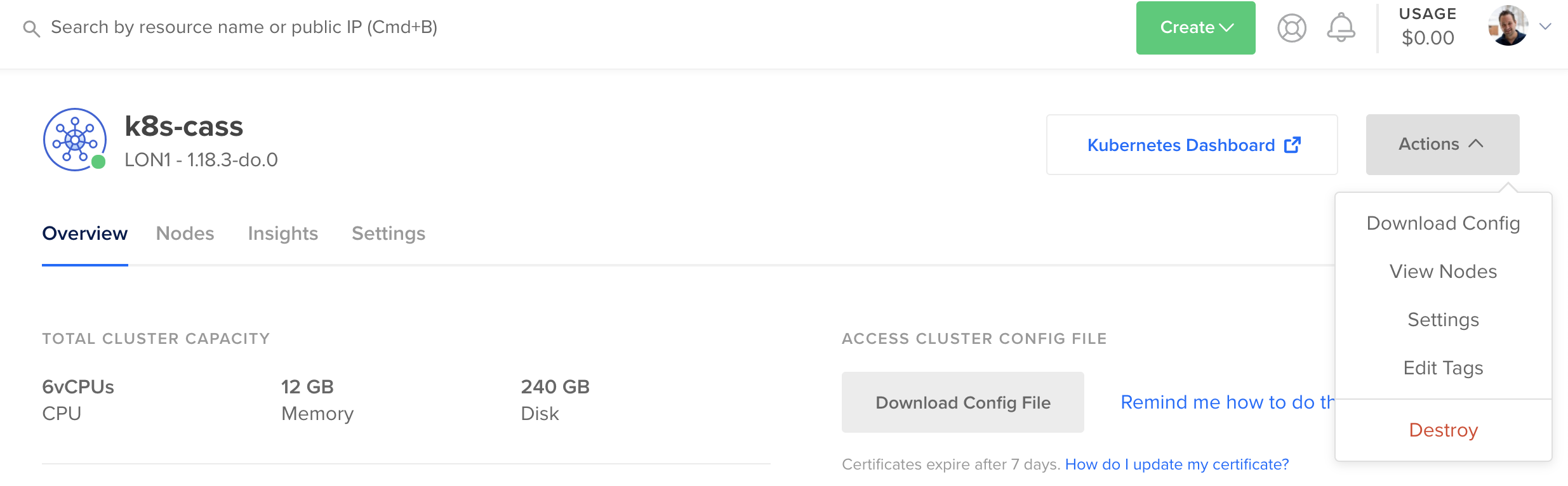
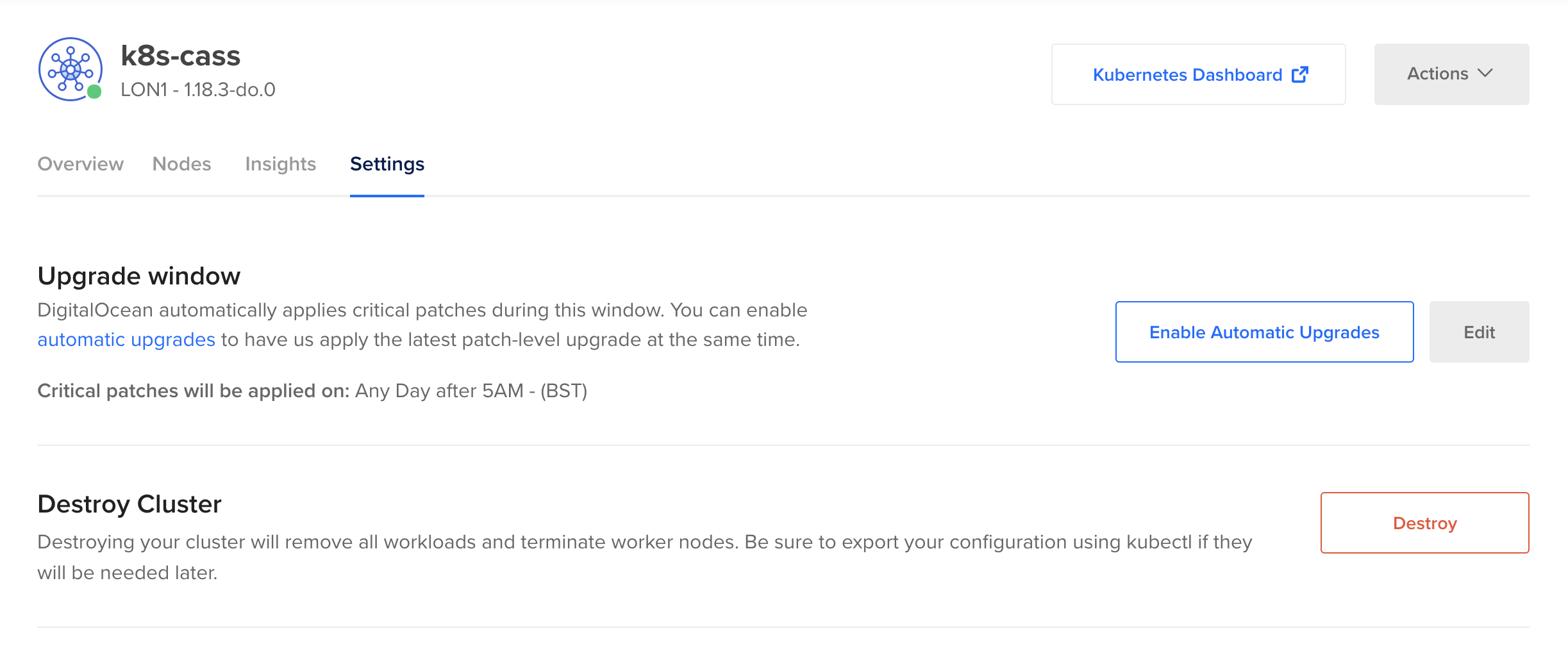
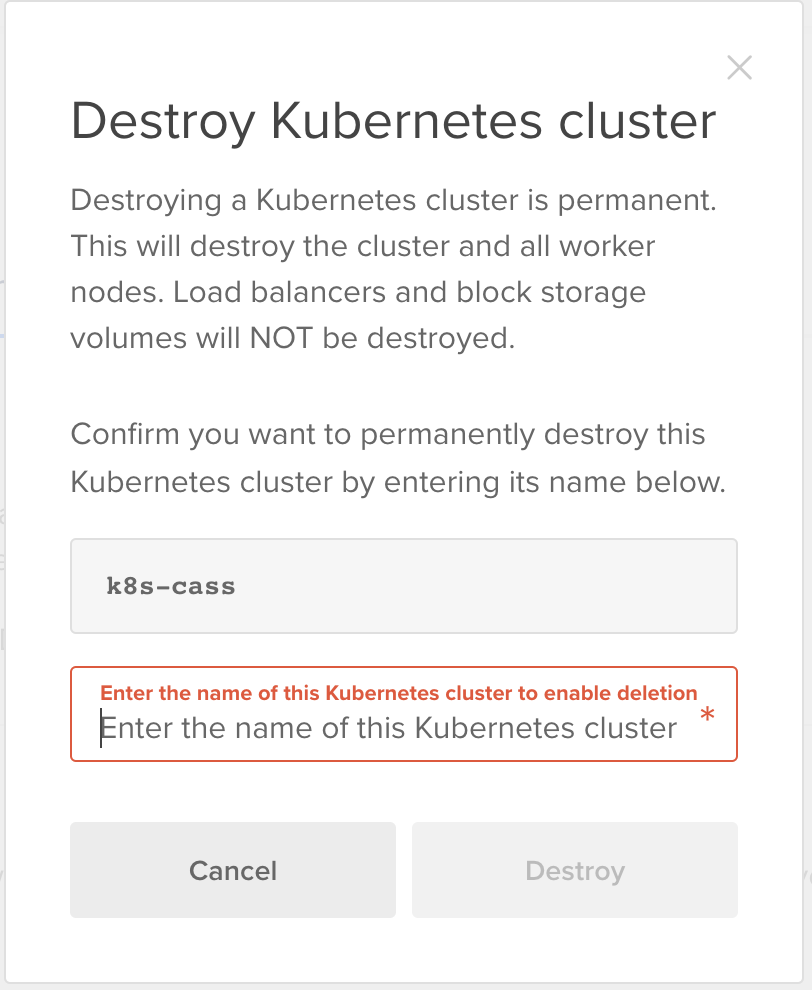
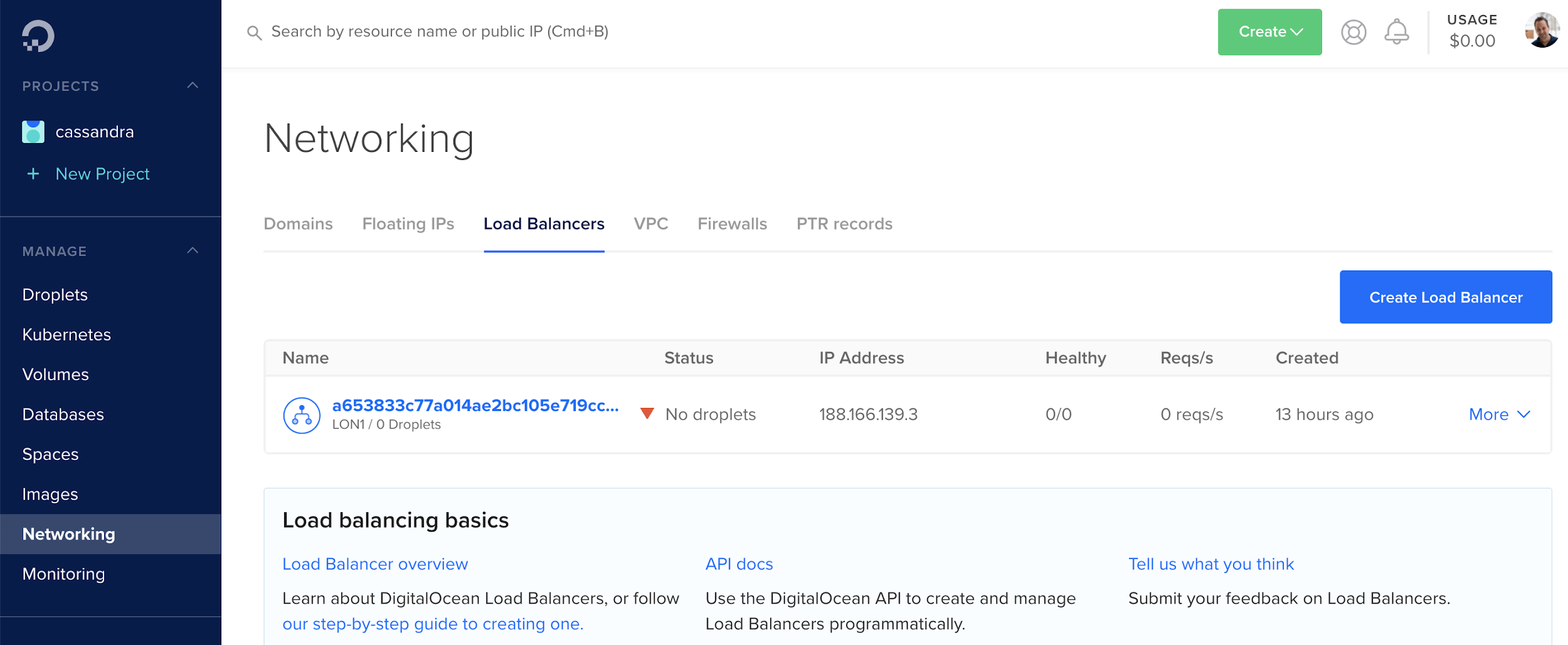
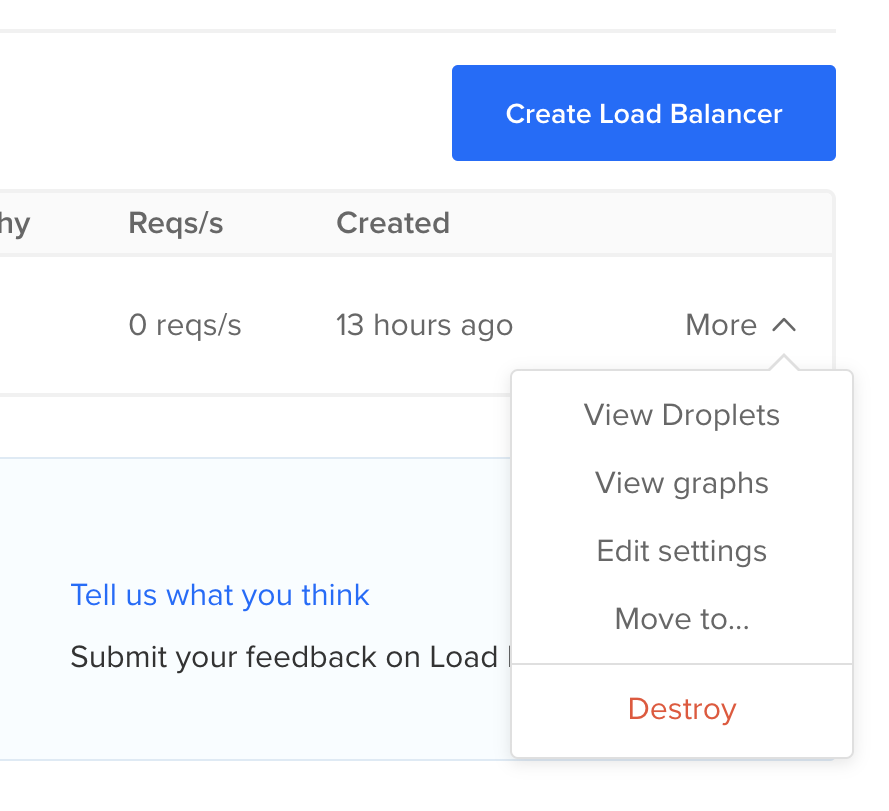
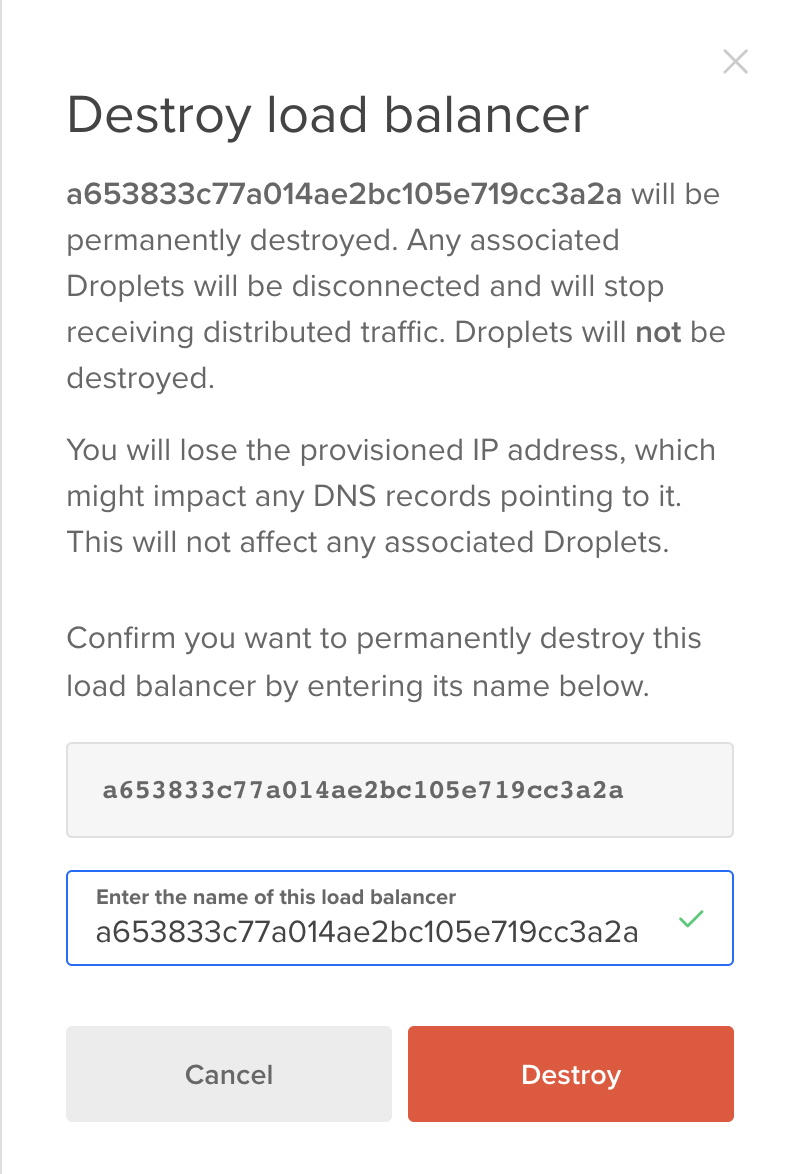
****

1. Compute resources by Node:



1. And lots more - have a good look.
2. That is the end of the lab. You have two choices now. Either you can delete the Kubernetes cluster (and stop spending that credit), or you can continue with the next exercise where we install cassandra into the cluster. If you want to delete the cluster, follow the next steps.
3. If you want to install Cassandra, go to **Exercise 14b**

**DESTROY THE CLUSTER**

1. Go to the kubernetes cluster page and find **Actions -> Destroy  
   **
2. You will see:  
   ****Click on **Destroy**
3. You will need to type the name of the cluster: **k8s-cass  
   **
4. Then click **Destroy**
5. DigitalOcean will also have created a load-balancer to handle the incoming traffic for your service. Go to **Networking -> Load Balancers  
   **
6. Click on **Destroy** and once again enter the name (copy and paste!)  
   
7. This lab is done! Congratulations.