

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
k9s

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.

Although I've given instructions for using the Ubuntu VM, you can install kubectl/k9s on your own machines and do this from there as well.

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.

- a. go to: <https://try.digitalocean.com/developerBrand/>

You should see:

- b. Sign up or use an existing Google or Github account to login.
- c. Enter your billing details
- d. You should see something like:

- e. Give your project the name “cassandra”:

Create your first project

Enter project name
cassandra ✓

What is your project for?

This information will help us improve the projects experience, based on what you're building.

Just trying out DigitalOcean * ▾

Tell us which tools and technologies you plan to use on DigitalOcean

CONFIGURATION MANAGEMENT

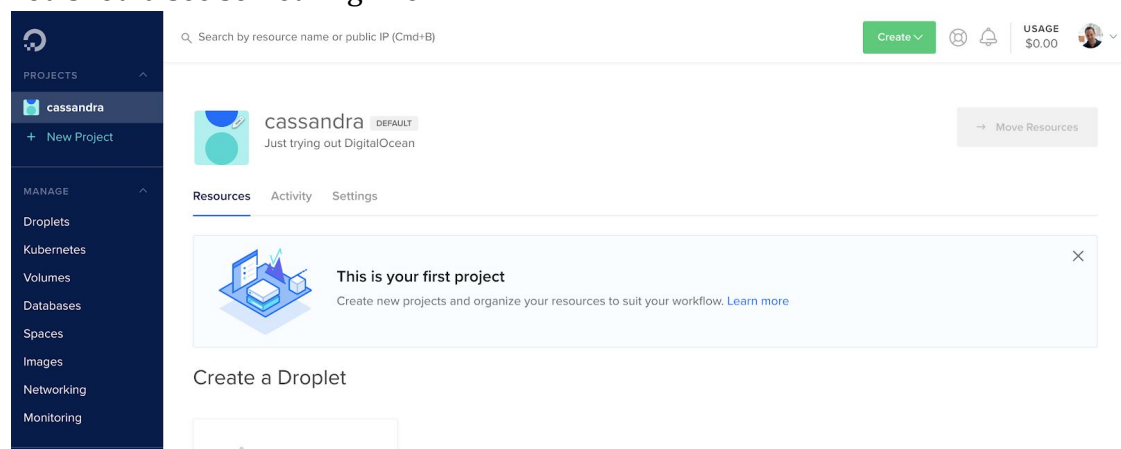
☐ Ansible ☐ Chef ☐ Docker ☒ Kubernetes ✓

☐ Puppet

f. Select “Just trying out DigitalOcean” and tick Kubernetes

g. Click “Start” at the bottom of the page

h. You should see something like:



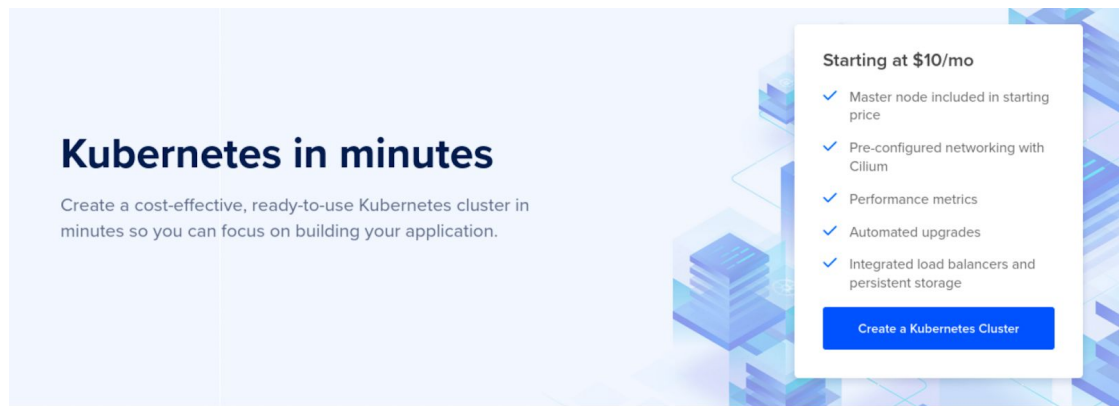
2. Before we create the Kubernetes cluster, we'd like to update the kubernetes CLI tool. By default the one in the Ubuntu package repo is out of date. We can fix that by doing the following commands (taken from <https://kubernetes.io/docs/tasks/tools/install-kubect/>)

```
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

3. Go back to your Firefox / DigitalOcean window.

4. Click on Kubernetes in the left hand side. You should see:



Now click **Create a Kubernetes Cluster**

Choose Kubernetes version 1.18.x

Create a cluster

Select a Kubernetes version

Select the Kubernetes version. The newest available version is selected by default.

1.18.3-do.0 (latest) ▼

! Tip: We generally recommend the latest version unless your team has a specific need. [See the DigitalOcean Kubernetes release notes.](#)

(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).

5. Choose your nearest datacentre (e.g. London)

6. Choose the following:

3 nodes

Standard Nodes

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

Choose cluster capacity ?

Increasing the number of nodes in a pool lets you run more instances of the scheduled services. Adding more node pools allows you to schedule pods to different node pools so each pod has the RAM, CPU, and storage it requires. You can add and remove nodes and node pools at any time.

NODE POOL NAME	MACHINE TYPE (DROPLET)	NODE PLAN ?	NUMBER NODES
<input type="text" value="Enter pool name"/> k8s-pool-cass ✓	Standard nodes Variable ratio of memory per shared CPU ▼	\$20/Month per node (\$0.030/hr) 2.5 GB RAM usable (4 GB Total) / 2 vCPUs ▼	3 ^ ▼
<input type="button" value="Add Additional Node Pool"/>			

7. Change the name to k8s-cass

Choose a name

You can edit the default name to something meaningful to you.

k8s-cass

✓

8. Click **Create Cluster**

9. You should see:

k8s-cass
LON1 - 1.18.3-do.0

Overview Nodes Insights Settings

Getting Started

- ✓ Create a Kubernetes cluster
- 2 Install management tools
- 3 Download the config file
- 4 Install 1-Click Apps
- 5 Next steps

[Skip Getting Started](#)
[Don't Show Again](#)

Thank you for using managed Kubernetes. Your Kubernetes cluster is being provisioned.

Provisioning is usually complete within 4 minutes. You can configure the cluster while you wait.

[Get Started](#)

TOTAL CLUSTER CAPACITY

6vCPUs CPU	12 GB Memory	240 GB Disk
---------------	-----------------	----------------

ACCESS CLUSTER CONFIG FILE

[Download Config File](#)

Certificates expire after 7 days. [How do I update my certificate?](#)

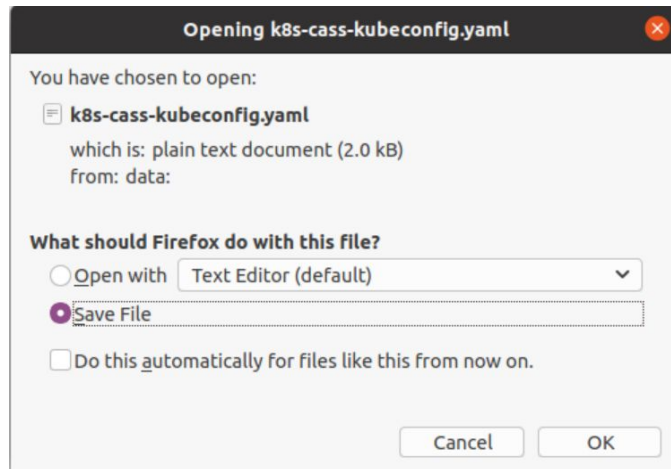
10. There is a nice “checklist” of actions you can do with your cluster. Click on #2. We already have the management tools downloaded (at least kubectl), so we can **Continue**

11. 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:

Quick connect with manual certificate management

If you're just trying out Kubernetes with a single cluster, you can [download the cluster configuration file](#) to your `~/.kube` directory now. The authentication certificate is good for one week, after which you will need to re-download it.

Click on “download the cluster configuration file”
Save File



12. 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
```

Copy

```
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

13. 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)

14. Check it works:

```
kubectl get all
```

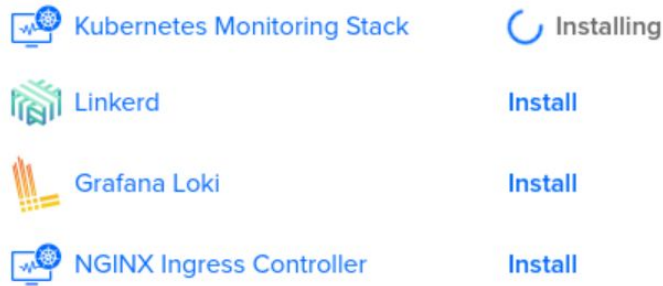
You should see something like:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.245.0.1	<none>	443/TCP	45m

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

Marketplace 1-Click Apps

Click 'Install' on any 1-Click App to deploy it to your Kubernetes cluster.



Or browse [all of our 1-Click Apps for Kubernetes](#). 

PART B: INSTALL AN APP INTO K8S

16. 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
```

17. Now let's apply (install) this app into kubernetes:

```
kubectl apply -f yaml/hello-kubernetes.yaml
```

19. 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     5m2s
service/kubernetes                   ClusterIP                 10.245.0.1      <none>           443/TCP          125m

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
```

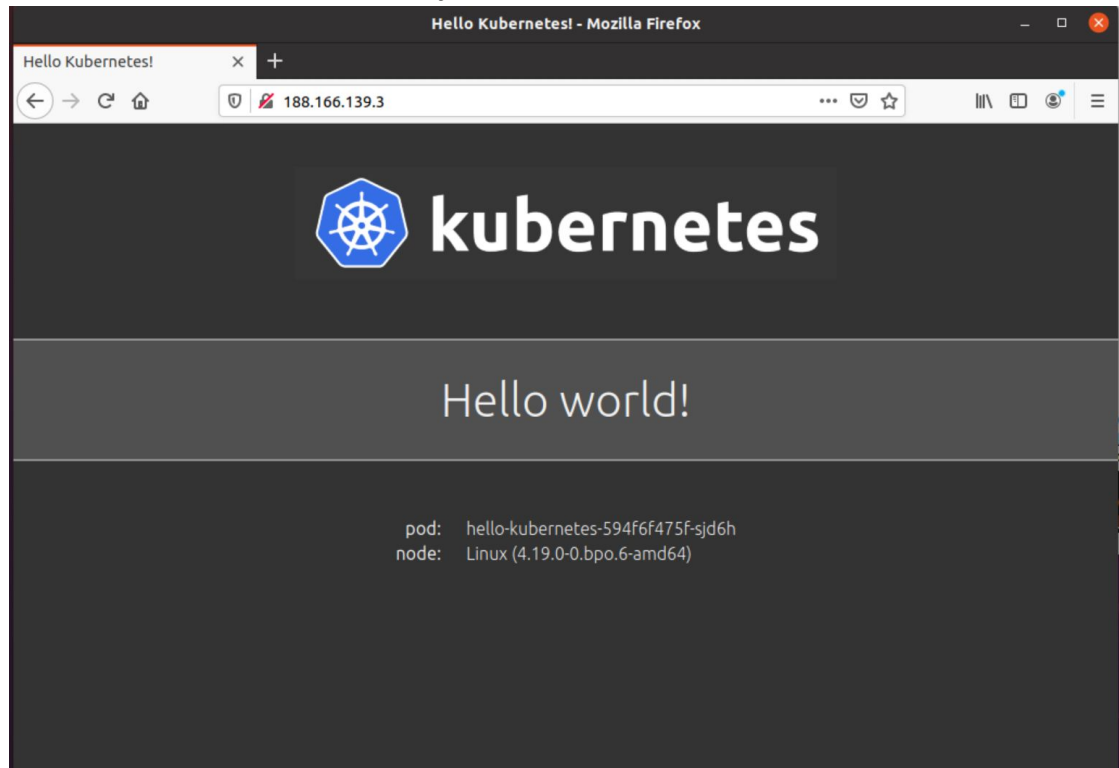
This basically defines a pod with 3 replicas containing a single container instance. There is then a load-balancer that balances load across the three replicas.

20. 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

21. Go to the external IP address in your browser:

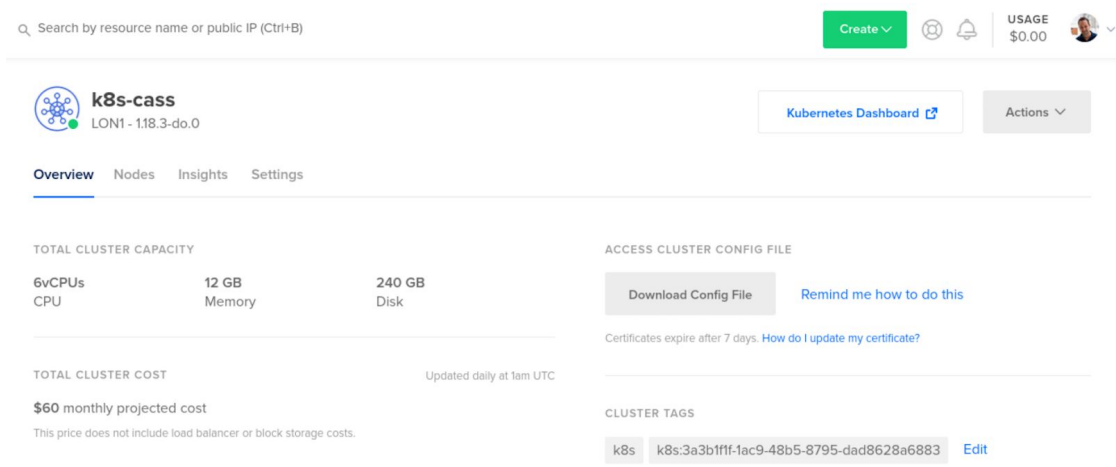


22. Keep reloading and you should see the pod details change.

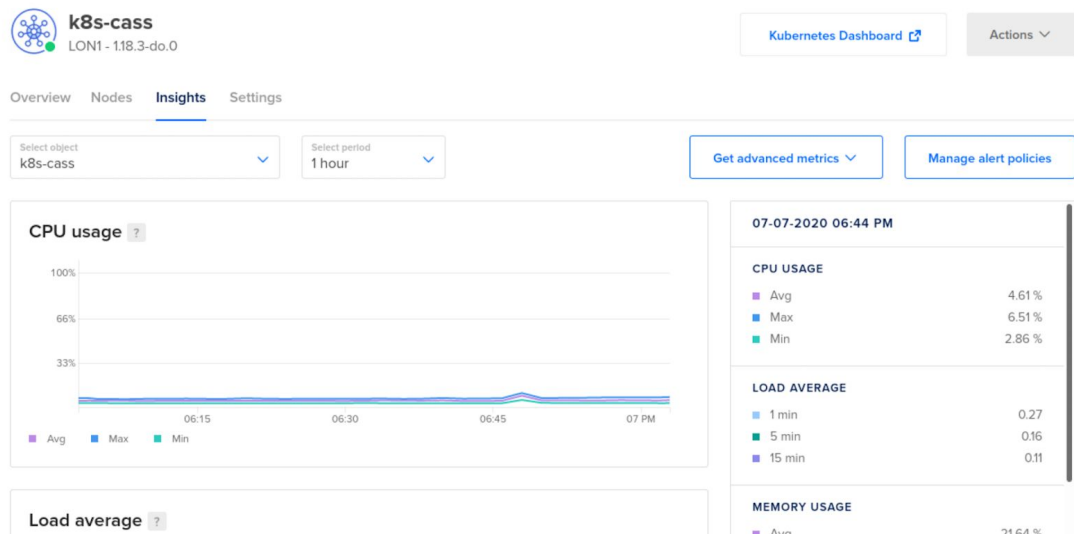
23. Congrats - you've deployed a k8s app.

PART C: MONITORING

24. We can go and monitor the system from the DigitalOcean web ui. Navigate to the cluster info page:



25. Click on **Insights**



26. You can see the system monitoring.

[Kubernetes Dashboard](#)

27. Click on

28. Navigate to look at Services

The image shows the 'Services' page of the Kubernetes Dashboard. The left sidebar contains a navigation menu with categories: Nodes, Persistent Volumes, Service Accounts, Storage Classes, Workloads, Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Service, Ingresses, and Config and Storage. The 'Services' page displays a table of services. The table has columns: Name, Namespace, Labels, Cluster IP, Internal Endpoints, External Endpoints, and Created. The services listed are 'hello-kubernetes' (default namespace, Cluster IP: 10.245.249.202), 'prometheus-operated' (prometheus-operator namespace, Cluster IP: None), and 'prometheus-operator-kubelet' (kube-system namespace, Cluster IP: None). The 'hello-kubernetes' service has internal endpoints 'hello-kubernetes:80 TCP' and 'hello-kubernetes:30816 TCP', and an external endpoint '188.166.139.3:80'. The 'prometheus-operated' service has internal endpoints 'prometheus-operated.prometheus-operator:9090 TCP' and 'prometheus-operated.prometheus-operator:0 TCP'. The 'prometheus-operator-kubelet' service has internal endpoints 'prometheus-operator-kubelet.kube-system:10250 TCP' and 'prometheus-operator-kubelet.kube-system:10255 TCP'.

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Created
hello-kubernetes	default	-	10.245.249.202	hello-kubernetes:80 TCP hello-kubernetes:30816 TCP	188.166.139.3:80	19 minutes ago
prometheus-operated	prometheus-operator	operated-prometheus: true	None	prometheus-operated.prometheus-operator:9090 TCP prometheus-operated.prometheus-operator:0 TCP	-	an hour ago
prometheus-operator-kubelet	kube-system	k8s-app: kubelet	None	prometheus-operator-kubelet.kube-system:10250 TCP prometheus-operator-kubelet.kube-system:10255 TCP	-	an hour ago

29. You can see a nice link to the external webpage of your app.

30. Browse pods and go look at a pod:

The screenshot shows the Kubernetes dashboard interface. The left sidebar contains a navigation menu with categories like Nodes, Persistent Volumes, Service Accounts, Storage Classes, Workloads, Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Service, Ingresses, Services, Config and Storage, Config Maps, Persistent Volume Claims, Secrets, and Custom Resource Definitions. The 'Pods' section is selected. The main panel displays details for a specific pod: 'hello-kubernetes-594f6f475f-4rksn' in the 'default' namespace. It was created on 'Jul 7, 2020' and is '20 minutes ago' old. The pod is running on node 'k8s-pool-cass-3o8i7' with IP '10.244.0.111' and has 0 restarts. The 'Conditions' table shows the following:

Type	Status	Last probe time	Last transition time	Reason	Message
Initialized	True	-	20 minutes ago	-	-
Ready	True	-	20 minutes ago	-	-
ContainersReady	True	-	20 minutes ago	-	-
PodScheduled	True	-	20 minutes ago	-	-

31. 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.

32. For some obscure reason we need to create a directory for the .k9s config file:

```
mkdir ~/.k9s
```

33. Now start k9s:

```
k9s
```

You should see:

The screenshot shows the k9s terminal interface. At the top, it displays context information: Context: do-lon1-k8s-cass, Cluster: do-lon1-k8s-cass, User: do-lon1-k8s-cass-admin, K9s Rev: 0.7.12, K8s Rev: v1.18.3, CPU: 4%, and MEM: 30%. A list of keyboard shortcuts is shown on the right, including <ctrl-d> for Delete, <d> for Describe, <e> for Edit, <l> for Logs, <shift-l> for Logs Pre..., <ctrl-s> for Save, and <s> for Shell. Below this, a table titled 'Pods(default)[3]' lists three running pods with their names, ready status, status, restarts, CPU, memory, and IP addresses. The first pod is 'hello-kubernetes-594f6f475f-4rksn' on node 'k8s-p...'. The second is 'hello-kubernetes-594f6f475f-h25gz' on node 'k8s-p...'. The third is 'hello-kubernetes-594f6f475f-sjd6h' on node 'k8s-p...'. At the bottom, there is a prompt '<po>'.

34. This is an awesome tool. Hit enter twice to see the pod logs. Have a look at the docs here: <https://k9scli.io/>
35. You can “drill” into pods and containers just by hitting Enter. You get back to the main screen with Esc. You can leave k9s by using Ctrl-C.
36. Do you remember that we installed the Kubernetes 1-click monitoring. Let’s take a look at that.
37. 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-xgk1c      2/2      Running    0      113m
```

Copy that name into this:

```
kubectl port-forward prometheus-operator-grafana-cf6954699-xgk1c \
-n prometheus-operator 8080:3000
```

Changing the name to match yours.

You should see:

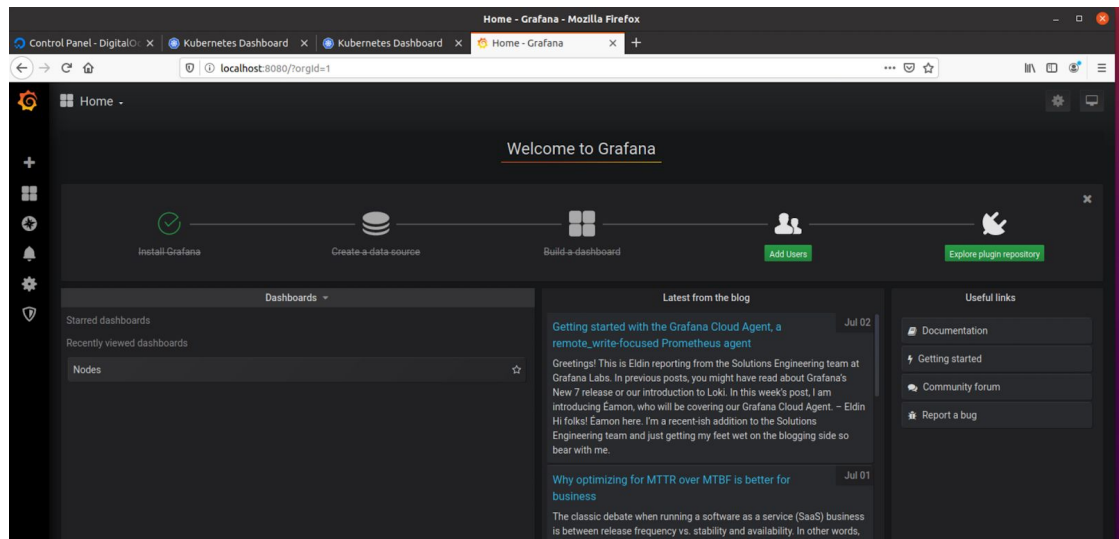
```
Forwarding from 127.0.0.1:8080 -> 3000
```

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

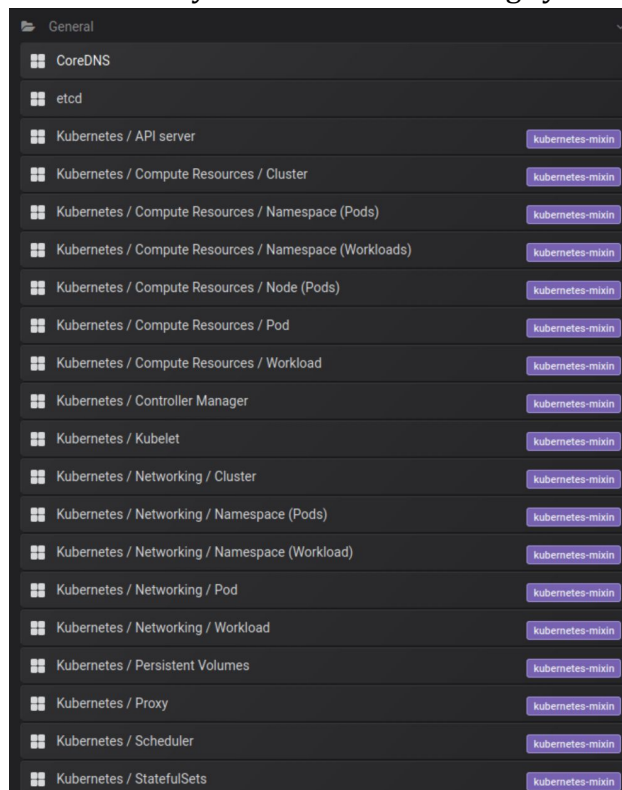
39. Now browse <http://localhost:8080>

The username you need is **admin**
And the password is **prom-operator**

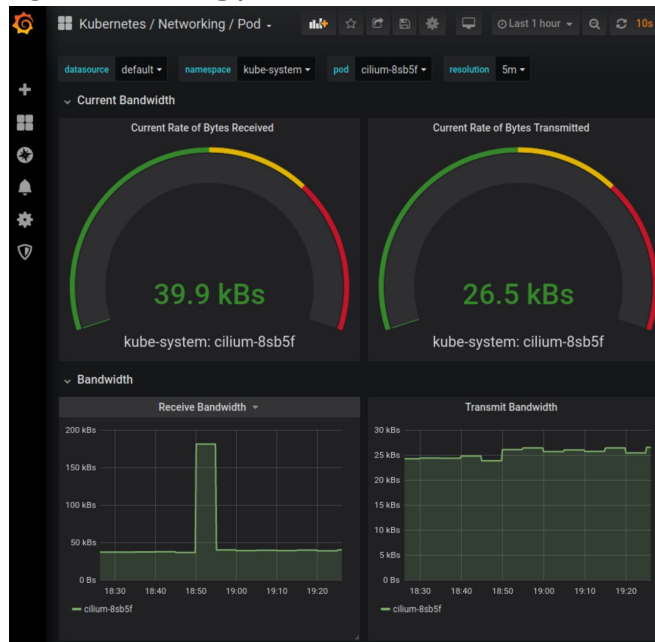
Obviously in a prod system you'd need to change these!



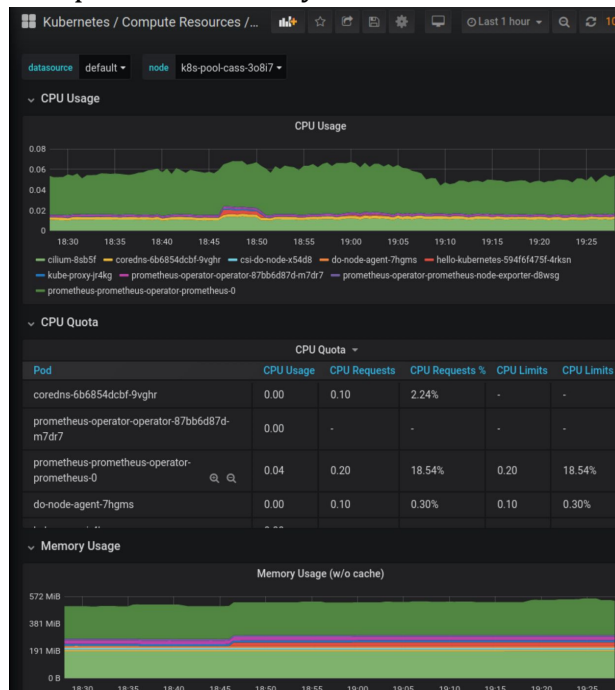
40. Under **Home** you will see lots of things you can look at:



41. e.g. Networking / Pod



42. Compute resources by Node:



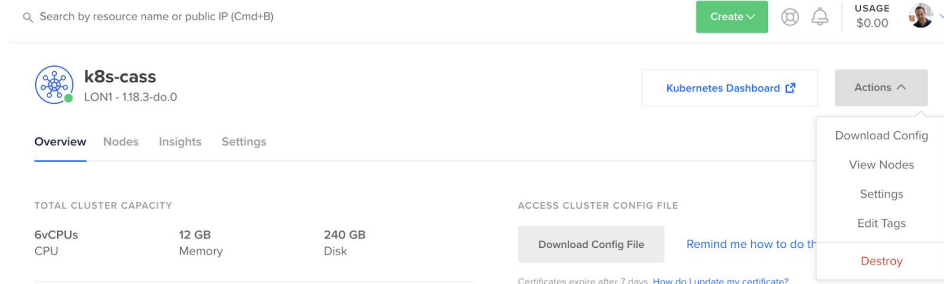
43. And lots more - have a good look.

44. 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.

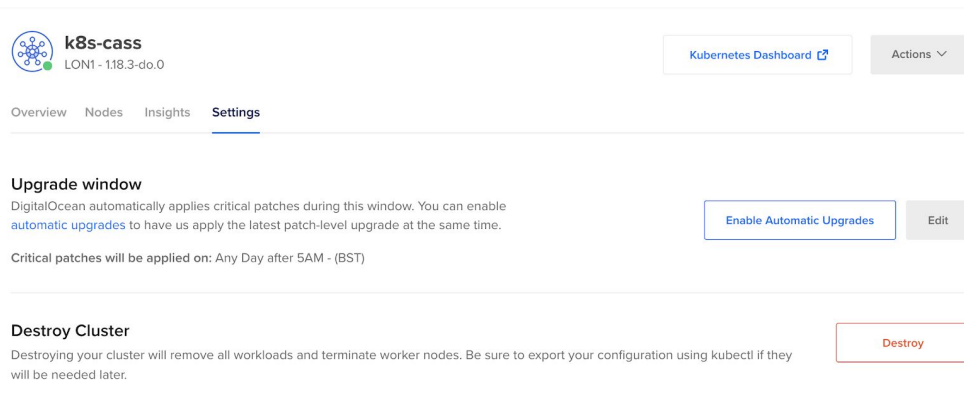
45. If you want to install Cassandra, go to **Exercise 14b**

DESTROY THE CLUSTER

46. Go to the kubernetes cluster page and find **Actions -> Destroy**

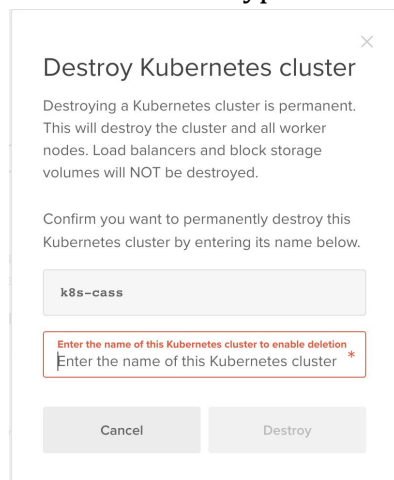


47. You will see:



Click on **Destroy**

48. You will need to type the name of the cluster: **k8s-cass**



49. Then click **Destroy**

50. DigitalOcean will also have created a load-balancer to handle the incoming traffic for your service. Go to **Networking -> Load Balancers**

PROJECTS

cassandra

+ New Project

MANAGE

Droplets

Kubernetes

Volumes

Databases

Spaces

Images

Networking

Monitoring

Q Search by resource name or public IP (Cmd+B)

Create

USAGE

\$0.00

Networking

Domains Floating IPs **Load Balancers** VPC Firewalls PTR records

Create Load Balancer

Name	Status	IP Address	Healthy	Reqs/s	Created
a653833c77a014ae2bc105e719cc... LON1 / 0 Droplets	No droplets	188.166.139.3	0/0	0 reqs/s	13 hours ago More

Load balancing basics

[Load Balancer overview](#)
Learn about DigitalOcean Load Balancers, or follow our [step-by-step guide to creating one](#).

[API docs](#)
Use the DigitalOcean API to create and manage Load Balancers programmatically.

[Tell us what you think](#)
Submit your feedback on Load Balancers.

Create Load Balancer

hy	Reqs/s	Created
	0 reqs/s	13 hours ago More

[Tell us what you think](#)
Submit your feedback on Load

View Droplets

View graphs

Edit settings

Move to...

Destroy

51. Click on **Destroy** and once again enter the name (copy and paste!)

×

Destroy load balancer

a653833c77a014ae2bc105e719cc3a2a will be permanently destroyed. Any associated Droplets will be disconnected and will stop receiving distributed traffic. Droplets will **not** be destroyed.

You will lose the provisioned IP address, which might impact any DNS records pointing to it. This will not affect any associated Droplets.

Confirm you want to permanently destroy this load balancer by entering its name below.

`a653833c77a014ae2bc105e719cc3a2a`

Enter the name of this load balancer

a653833c77a014ae2bc105e719cc3a2a ✓

Cancel

Destroy

52. This lab is done! Congratulations.