Getting started with Kubernetes

In this lab you are going to learn the basics of Kubernetes. The lab uses kubectl together with a minikube cluster and is very easy to go through as it uses an already existing docker image.

If you want to do a more extensive getting started you can follow this tutorial.

Prerequisites

Before you can execute the commands you need to have setup your environment on either native windows 10 or using a vagrant VM with Ubuntu.

Getting started

First thing you need to do is start your minikube cluster if it is not yet running. Open `boxstarter shell` or `Powershell` on Windows 10 or `bash shell` on Ubuntu. To start your minikube cluster issue the command `minikube start`. You should see a similar result as below:

```
Starting local Kubernetes v1.12.4 cluster...
Starting VM...

Everything looks great. Please enjoy minikube!
```

Testing if cluster is live

Test if you can connect to the `minikube` cluster from your command-line using `kubectl`. It is your local CLI command center. You can issue commands to the cluster either by `kubectl` or directly calling REST APIs exposed by the master. Issue the following command to get some info about your cluster.

```
kubectl cluster-info
```

The result of the command should be similar as below:

```
Kubernetes master is running at https://10.1.127.124:8443
KubeDNS is running at https://10.1.127.124:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

How many nodes are part of your cluster?

If you want to know how many nodes (bare metal machines / VMs) are part of your cluster you can run the command `kubectl get nodes`. As we are using `minikube` this should have to following result.

```
$> kubectl get nodes

NAME       STATUS    ROLES     AGE       VERSION
minikube   Ready     master    3d        v1.13.2
```

What pods are available by default?

The first concept of Kubernetes are **pods**. A pod is *a group of one or more containers*. Pods are **the smallest deployable units** that can be created and managed in Kubernetes. Most of Kubernetes components also runs on pods. Issue the command `kubectl get pods` to get all available pods or search on a namespace.

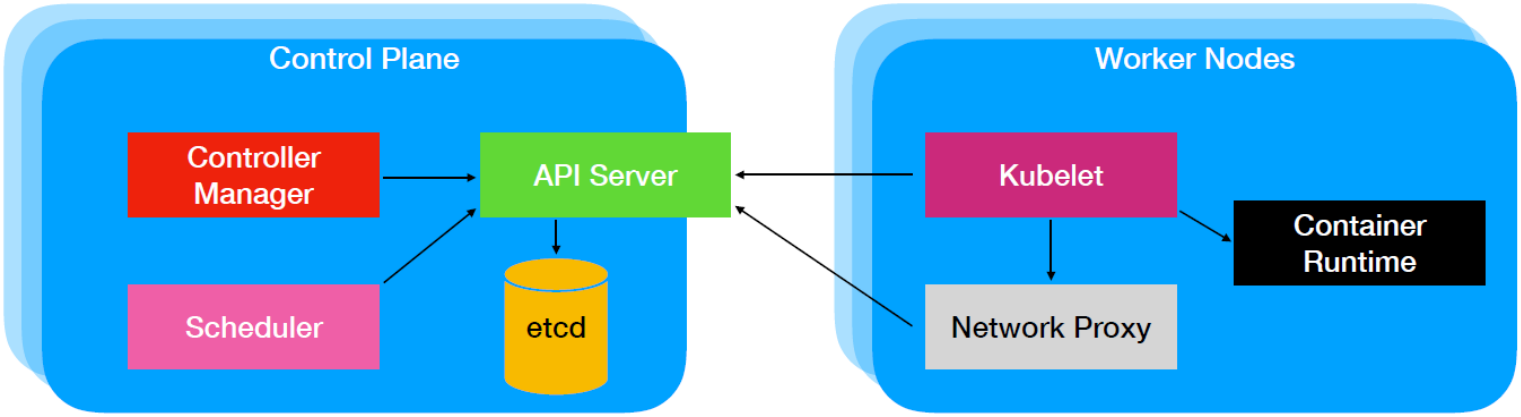
The second concept, **namespaces**, are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you should not need to create or think about namespaces at all.

All pods by *kubernetes* and *minikube* are part of the `kube-system` namespace. You can see them by issuing the same command but with the addition of `-n kube-system`.

```
$> kubectl get pods -n kube-system

NAME                                    READY     STATUS    RESTARTS   AGE
coredns-576cbf47c7-qzmz4                1/1       Running   1          25m
coredns-576cbf47c7-swsrf                1/1       Running   1          25m
etcd-minikube                           1/1       Running   0          3m
kube-addon-manager-minikube             1/1       Running   1          24m
kube-apiserver-minikube                 1/1       Running   0          3m
kube-controller-manager-minikube        1/1       Running   0          3m
kube-proxy-kpbs9                        1/1       Running   0          3m
kube-scheduler-minikube                 1/1       Running   1          24m
kubernetes-dashboard-5bff5f8fb8-rvwbh   1/1       Running   1          24m
storage-provisioner                     1/1       Running   1          24m
```

Can you spot he above pods in the below Kubernetes architecture?



Master node(s) host the Kubernetes control plane that control and manage the cluster

Worker nodes run the actual applications

Contol Plane

Multiple components that can run on a single master node or be split across multiple (master) nodes and replicated to ensure high availability:

- **API Server:** communication center for developers, sysadmin and other Kubernetes components
- **Scheduler:** assigns a worker node to each deployable component
- **Controller Manager**: performs cluster-level functions (replication, keeping track of worker nodes, handling nodes failures...)
- **etcd**: reliable distributed data store where the cluster configuration is persisted

Worker Node

Machines that run containerized applications. It runs, monitors and provides services to applications via components:

- **Docker, rkt, or another container runtime:** runs the containers
- **Kubelet:** talks to API server and manages containers on its node
- **Network Proxy:** load balance network traffic between application components

Connect minikube cluster to local docker environment

Before you can deploy docker images on the Kubernetes container runtime you need to configure minikube so that it uses your local docker repository. This can done be either issuing the command

- Powershell: `minikube docker-env | Invoke-Expression`
- Bash: `eval $(minikube docker-env)`

If the command is successful it returns an empty result.

Create a deployment on Kubernetes cluster

A *Deployment* controller provides declarative updates for Pods and ReplicaSets (a replica set (3rd concept of Kubernetes) ensures how many replica of pod should be running). You describe a *desired state* in a Deployment object, and the Deployment controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

In this lab we are going to install Nginx (a reverse proxy) as a deployment. Issue the following command in `powershell` or `bash`.

```
kubectl create deployment hello-nginx --image=nginx
```

This command looks in your local docker repository if the image exists, if not it is downloaded using docker, and deploys it as a pod on your cluster. You will get similar (final) output is shown below.

```
deployment.extensions "hello-nginx" created
```

To view your new deployment issue the command `kubectl get deployments`.

```
$> kubectl get deployments

NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
hello-nginx   1         1         1            1           1m
```

View the pods the deployment has started by issuing the command `kubectl get pods`.

```
$> kubectl get pods

NAME                           READY     STATUS    RESTARTS   AGE
hello-nginx-6476b4b4-dnsrp     1/1       Running   0          21m
```

To understand what is happening in the background there is a handy command for this. Issue the command `kubectl get events` to get a similar result as below.

```
LAST SEEN   FIRST SEEN   COUNT     NAME                                                KIND         SUBOBJECT
                 TYPE      REASON              SOURCE                    MESSAGE
2m          2m           1         hello-nginx-84d8d5d8f5-b8wkh.15779d3700ff4a2c   Pod
                  Normal    Scheduled           default-scheduler       Successfully assigned
default/hello-nginx-84d8d5d8f5-b8wkh to minikube
2m          2m           1         hello-nginx-84d8d5d8f5-b8wkh.15779d372bdd96e0   Pod
spec.containers{nginx}        Normal    Pulling             kubelet, minikube       pulling image "nginx"
1m          1m           1         hello-nginx-84d8d5d8f5-b8wkh.15779d4909d2e2b4   Pod
spec.containers{nginx}        Normal    Pulled              kubelet, minikube       Successfully pulled
image "nginx"
1m          1m           1         hello-nginx-84d8d5d8f5-b8wkh.15779d49109067e8   Pod
spec.containers{nginx}        Normal    Created             kubelet, minikube       Created container
1m          1m           1         hello-nginx-84d8d5d8f5-b8wkh.15779d4918c345ac   Pod
spec.containers{nginx}        Normal    Started             kubelet, minikube       Started container
2m          2m           1         hello-nginx-84d8d5d8f5.15779d36fd7a9c58         ReplicaSet
                  Normal    SuccessfulCreate    replicaset-controller   Created pod: hello-nginx-
84d8d5d8f5-b8wkh
2m          2m           1         hello-nginx.15779d36fa8c2c3c                    Deployment
```

Create a service for the deployment

Last step for getting the docker image started is by exposing the deployment as a service. A **service** is an abstraction which defines a logical set of `Pods` and a policy by which to access them - sometimes called a micro-service. The set of `Pods` targeted by a `Service` is (usually) determined by a `Label Selector`.

1. Expose the Pod to the public internet using the `kubectl expose` command:

   ```
   kubectl expose deployment hello-nginx --type=LoadBalancer --port=80
   ```

   The `--type=LoadBalancer` flag indicates that you want to expose your Service outside of the cluster.

2. View the Service you just created:

   ```
   kubectl get services
   ```

Output:

```
NAME          TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
hello-nginx   LoadBalancer   10.98.44.97   <pending>     80:30529/TCP   22s
kubernetes    ClusterIP      10.96.0.1     <none>        443/TCP        3d
```

On cloud providers that support load balancers, an external IP address would be provisioned to access the Service. On Minikube, the `LoadBalancer` type makes the Service accessible through the `minikube service` command.

3. Finally, retrieve the URL of the deployed Nginx service by issuing the following command.

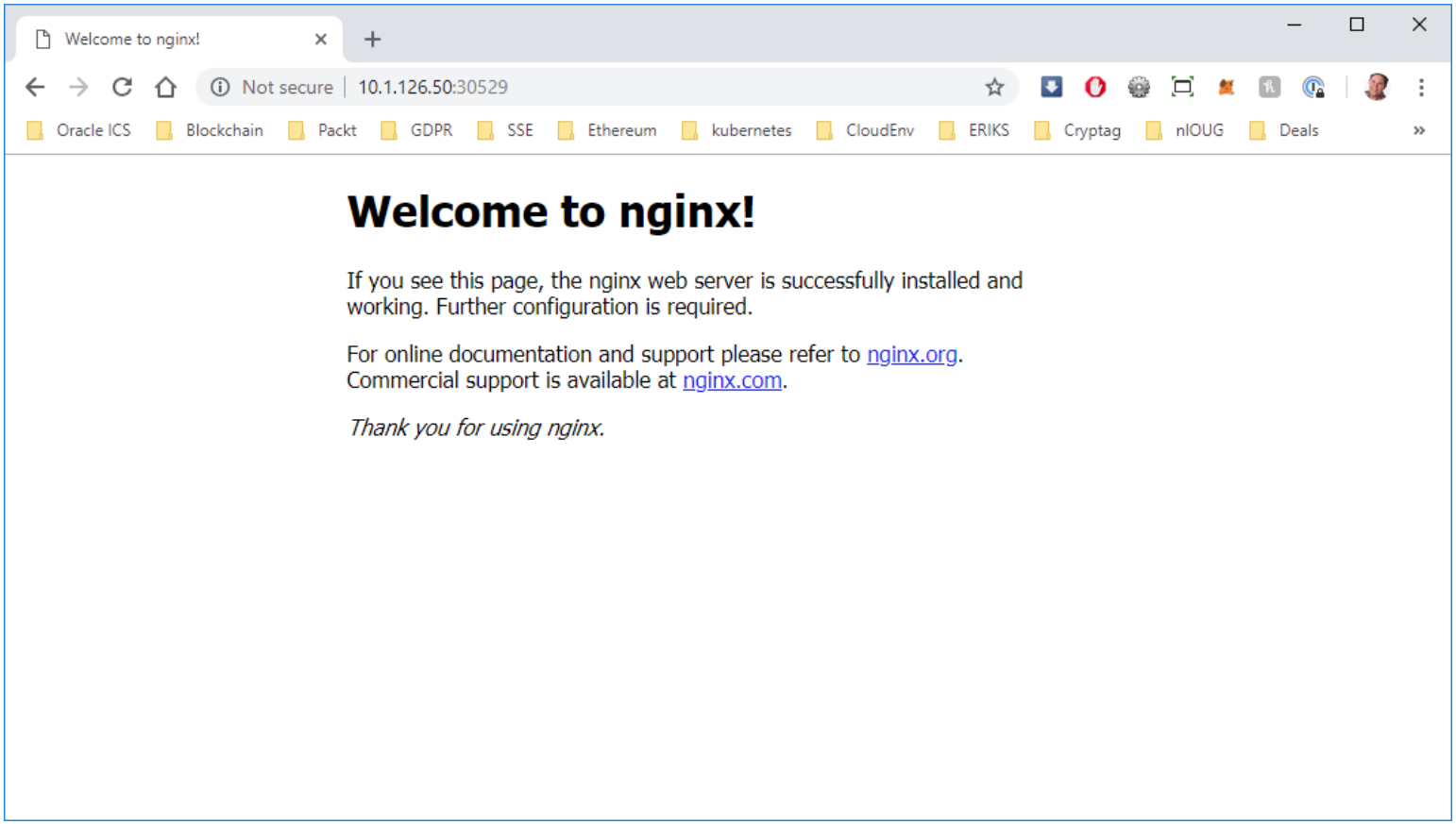
```
minikube service --url=true hello-nginx
```

This command asks `minikube` to return the URL of the `hello-nginx` deployment.

Output

```
http://10.1.126.50:30529
```

4. Go to the url given to see if it is running:



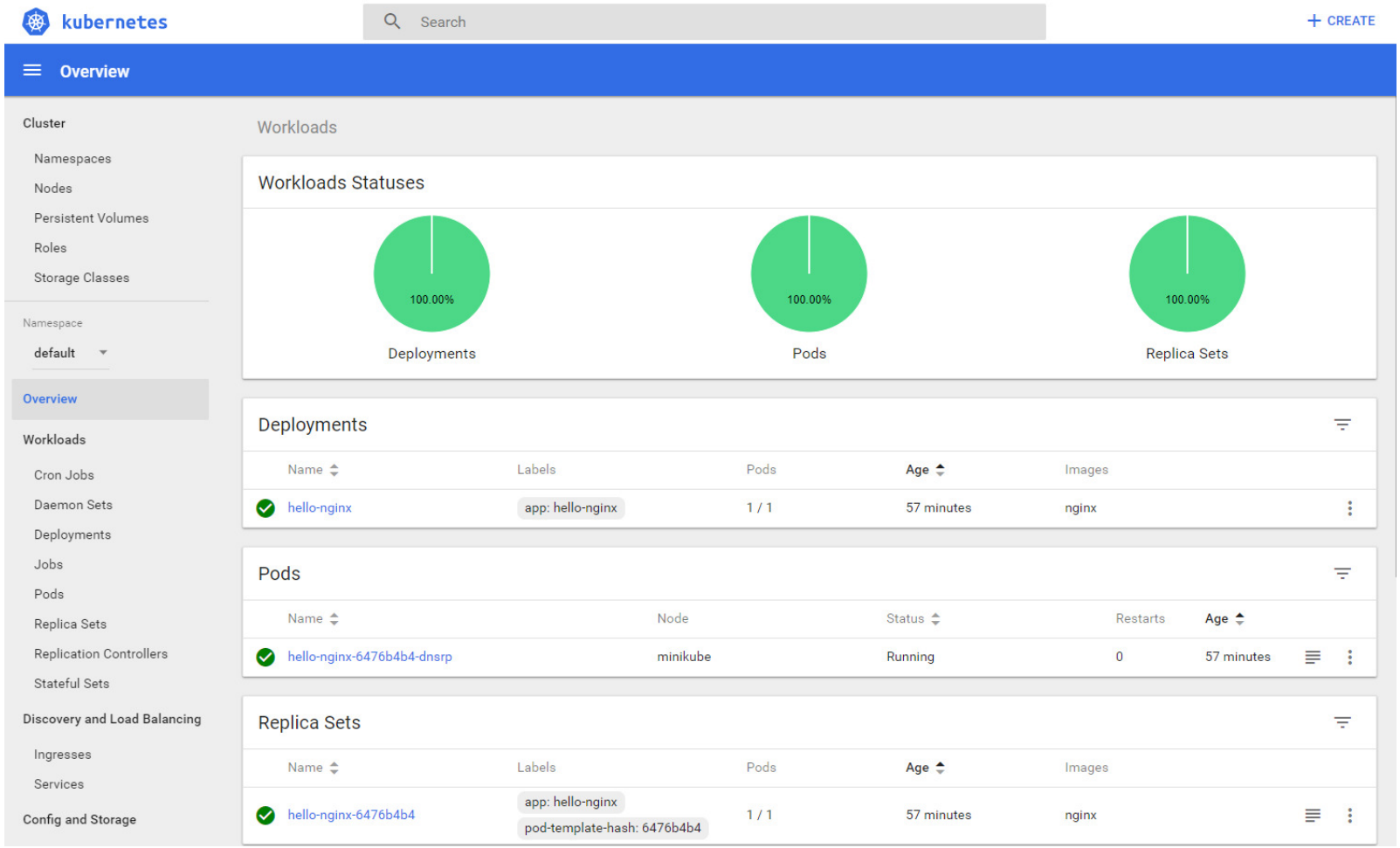
Starting the kubernetes dashboard

Kubernetes has a standard dashboard we can deploy to see what is running on the cluster. When using minikube this dashboard is already available out-of-the-box. Start the dashboard by issuing the following command `minikube dashboard`.

```
$> minikube dashboard

Enabling dashboard ...
Verifying dashboard health ...
Launching proxy ...
Verifying proxy health ...
Opening http://127.0.0.1:56817/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/ in
your default browser...
```

Which if everything goes in order a browser opens the dashboard page, as shown as the image below.



Go and take a look around to see if you can find more details about the hello-nginx deployment/service.

Adding more dashboard functionality

Minikube has a set of built in addons that can be used enabled, disabled, and opened inside of the local k8s environment. Minikube must be running for these commands to take effect.

1. See all addons for minikube using command `minikube addons list`:

```
$> minikube addons list

- registry: disabled
- registry-creds: disabled
- freshpod: disabled
- addon-manager: enabled
- dashboard: enabled
- heapster: disabled
- efk: disabled
- ingress: disabled
- default-storageclass: enabled
- storage-provisioner: enabled
- storage-provisioner-gluster: disabled
- nvidia-driver-installer: disabled
- nvidia-gpu-device-plugin: disabled
```

One interesting addon is heapster. After enabling heapster you can inspect the health of your environment in much more detail (CPU, Memory, Network usage).

2. Enable heapster addon using command `minikube addons enable heapster`:

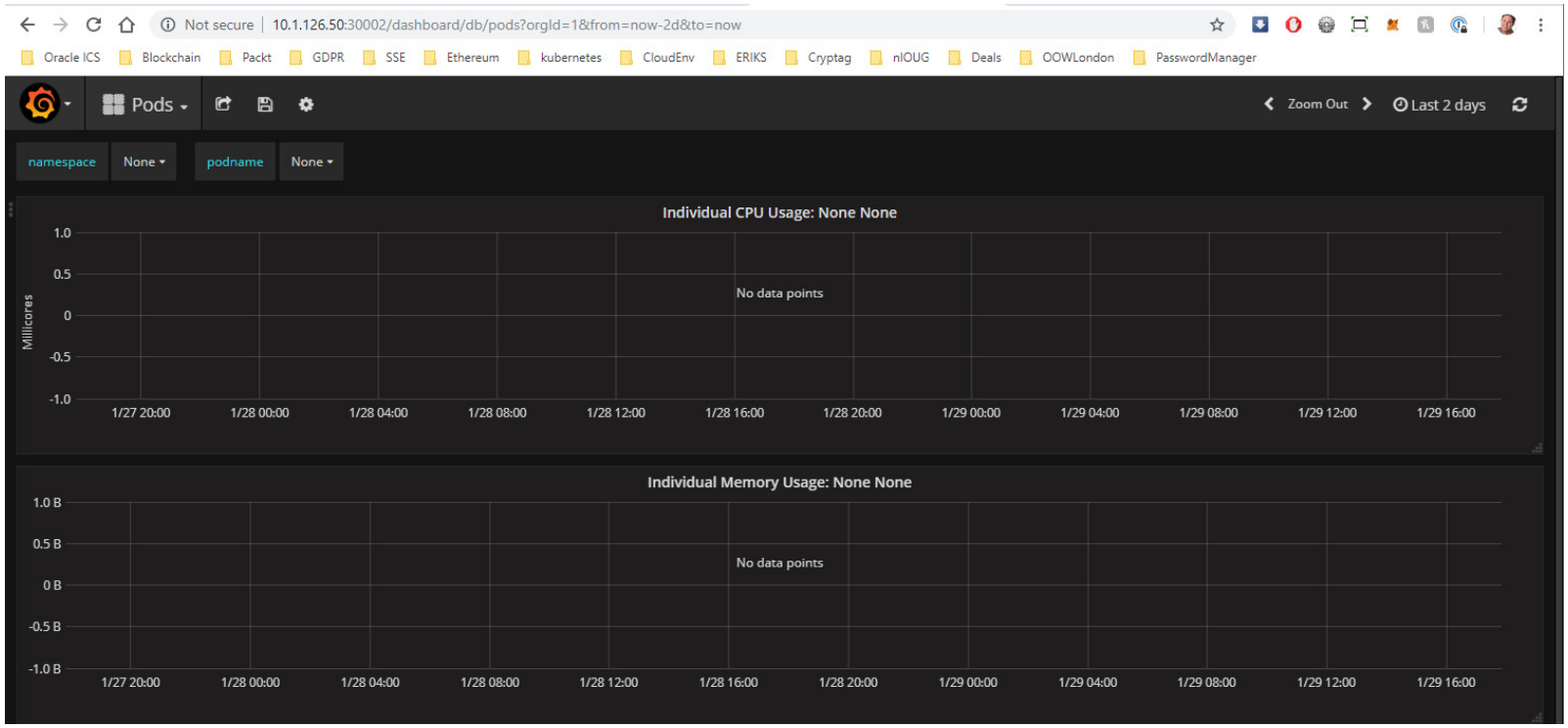
```
$> minikube addons enable heapster

heapster was successfully enabled
```

3. Open heapster in browser using command `minikube addons open heapster`:

```
# This will open grafana (interacting w/ heapster) in the browser
```

The following image shows graphics collected from pods:



Conclusion

In this lab you got started around with Kubernetes. If you want to play some more you can look at our Kubernetes Cheat Sheet.