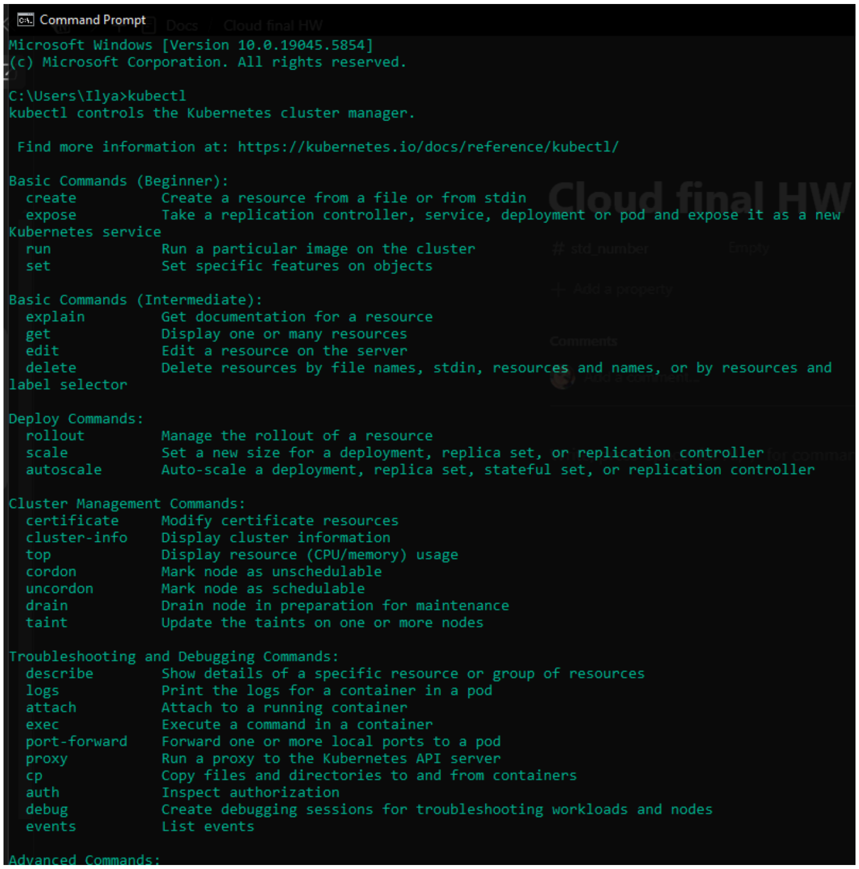
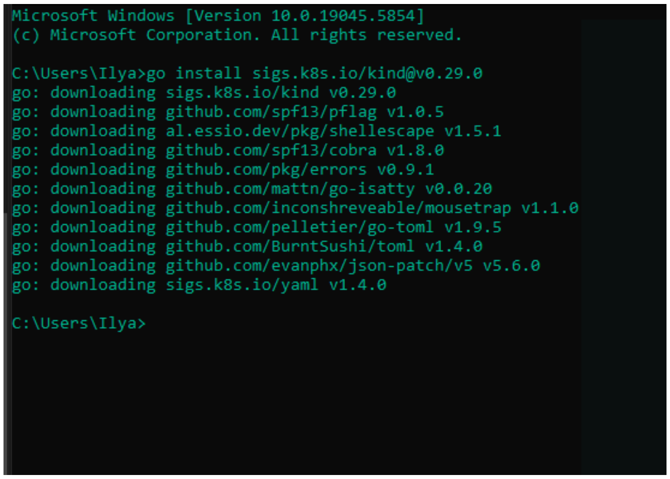
Cloud final HW

Parts 1 & 2

kind installation:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ilya>kubectl
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create          Create a resource from a file or from stdin
  expose          Take a replication controller, service, deployment or pod and expose it as a new
Kubernetes service
  run             Run a particular image on the cluster
  set             Set specific features on objects

Basic Commands (Intermediate):
  explain         Get documentation for a resource
  get             Display one or many resources
  edit            Edit a resource on the server
  delete          Delete resources by file names, stdin, resources and names, or by resources and
label selector

Deploy Commands:
  rollout         Manage the rollout of a resource
  scale           Set a new size for a deployment, replica set, or replication controller
  autoscale       Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate     Modify certificate resources
  cluster-info    Display cluster information
  top             Display resource (CPU/memory) usage
  cordon          Mark node as unschedulable
  uncordon        Mark node as schedulable
  drain           Drain node in preparation for maintenance
  taint           Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe        Show details of a specific resource or group of resources
  logs            Print the logs for a container in a pod
  attach          Attach to a running container
  exec            Execute a command in a container
  port-forward    Forward one or more local ports to a pod
  proxy           Run a proxy to the Kubernetes API server
  cp              Copy files and directories to and from containers
  auth            Inspect authorization
  debug           Create debugging sessions for troubleshooting workloads and nodes
  events          List events

Advanced Commands:
```



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ilya>go install sigs.k8s.io/kind@v0.29.0
go: downloading sigs.k8s.io/kind v0.29.0
go: downloading github.com/spf13/pflag v1.0.5
go: downloading al.essio.dev/pkg/shellescape v1.5.1
go: downloading github.com/spf13/cobra v1.8.0
go: downloading github.com/pkg/errors v0.9.1
go: downloading github.com/mattn/go-isatty v0.0.20
go: downloading github.com/inconshreveable/mousetrap v1.1.0
go: downloading github.com/pelletier/go-toml v1.9.5
go: downloading github.com/BurntSushi/toml v1.4.0
go: downloading github.com/evanphx/json-patch/v5 v5.6.0
go: downloading sigs.k8s.io/yaml v1.4.0

C:\Users\Ilya>
```

setting up a kind cluster and deploying nginx.

we use the kubectl command-line tool and below are the .yml/.yaml files

kind-cluster.yaml

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 80
    hostPort: 8080
    protocol: TCP
  - containerPort: 30080      # ← add this block
    hostPort: 30080
    protocol: TCP
```

nginx-deploy.yaml

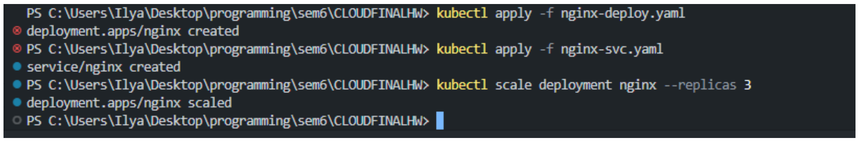
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

nginx-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30080
```

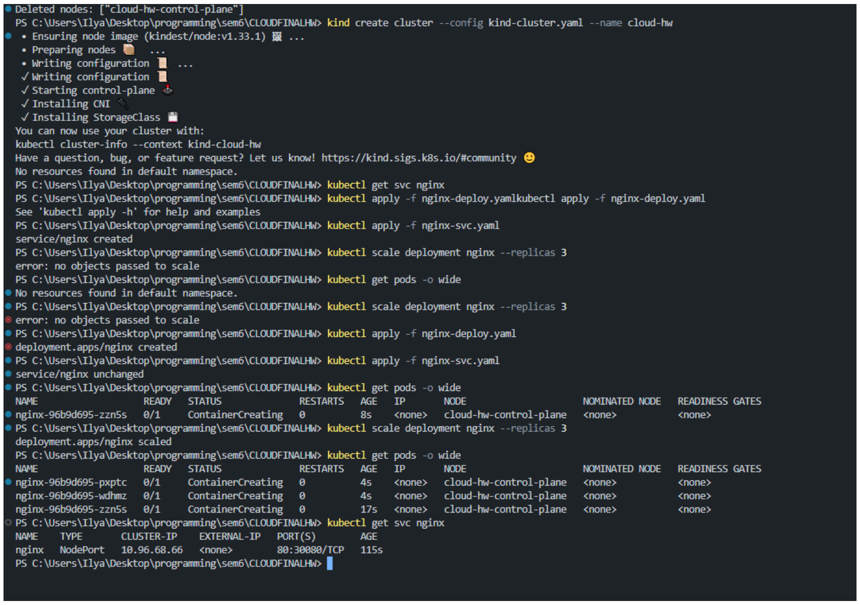
To expose the application I used the `nodeport` method because it seemed to be more straight forward.

After deploying nginx i scaled the deployment to three replicas.

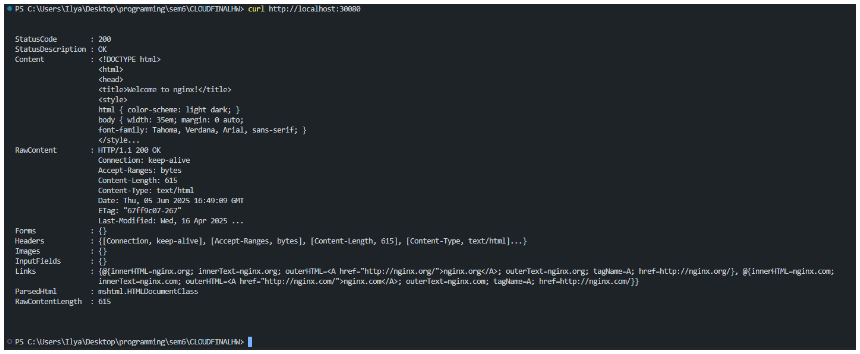


```
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-deploy.yaml
⊗ deployment.apps/nginx created
⊗ PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-svc.yaml
● service/nginx created
● PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl scale deployment nginx --replicas 3
● deployment.apps/nginx scaled
○ PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

There were some problems so I had to tear everything down and start again. Below is an image of all the commands I used to create a kind cluster, deploy and scale nginx.

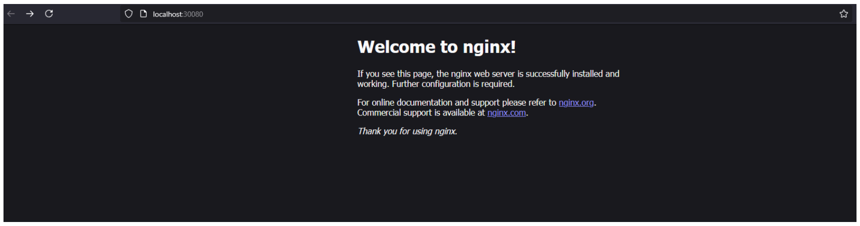


```
Deleted nodes: ["cloud-hw-control-plane"]
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kind create cluster --config kind-cluster.yaml --name cloud-hw
 • Ensuring node image (kindest/node:v1.33.1) 🖼 ...
 • Preparing nodes 📦  ...
 • Writing configuration 📜 ...
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
You can now use your cluster with:
kubectl cluster-info --context kind-cloud-hw
Have a question, bug, or feature request? Let us know! https://kind.sigs.k8s.io/#community 🙂
No resources found in default namespace.
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl get svc nginx
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-deploy.yamlkubectl apply -f nginx-deploy.yaml
See 'kubectl apply -h' for help and examples
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-svc.yaml
service/nginx created
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl scale deployment nginx --replicas 3
error: no objects passed to scale
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl get pods -o wide
No resources found in default namespace.
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl scale deployment nginx --replicas 3
error: no objects passed to scale
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-deploy.yaml
deployment.apps/nginx created
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl apply -f nginx-svc.yaml
service/nginx unchanged
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl get pods -o wide
NAME                    READY   STATUS              RESTARTS   AGE   IP       NODE                     NOMINATED NODE   READINESS GATES
nginx-96b9d695-zzn5s    0/1     ContainerCreating   0          8s    <none>   cloud-hw-control-plane   <none>           <none>
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl scale deployment nginx --replicas 3
deployment.apps/nginx scaled
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl get pods -o wide
NAME                    READY   STATUS              RESTARTS   AGE   IP       NODE                     NOMINATED NODE   READINESS GATES
nginx-96b9d695-pxptc    0/1     ContainerCreating   0          4s    <none>   cloud-hw-control-plane   <none>           <none>
nginx-96b9d695-wdhmz    0/1     ContainerCreating   0          4s    <none>   cloud-hw-control-plane   <none>           <none>
nginx-96b9d695-zzn5s    0/1     ContainerCreating   0          17s   <none>   cloud-hw-control-plane   <none>           <none>
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl get svc nginx
NAME    TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
nginx   NodePort   10.96.68.66   <none>        80:30080/TCP   115s
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```



```
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> curl http://localhost:30080


StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html>
                    <html>
                    <head>
                    <title>Welcome to nginx!</title>
                    <style>
                    html { color-scheme: light dark; }
                    body { width: 35em; margin: 0 auto;
                    font-family: Tahoma, Verdana, Arial, sans-serif; }
                    </style...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Accept-Ranges: bytes
                    Content-Length: 615
                    Content-Type: text/html
                    Date: Thu, 05 Jun 2025 16:49:09 GMT
                    ETag: "67ff9c07-267"
                    Last-Modified: Wed, 16 Apr 2025 ...
Forms             : {}
Headers           : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 615], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {@{innerHTML=nginx.org; innerText=nginx.org; outerHTML=<A href="http://nginx.org/">nginx.org</A>; outerText=nginx.org; tagName=A; href=http://nginx.org/}, @{innerHTML=nginx.com;
                    innerText=nginx.com; outerHTML=<A href="http://nginx.com/">nginx.com</A>; outerText=nginx.com; tagName=A; href=http://nginx.com/}}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 615


PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

To make sure everything is working alright we just go to the exposed port specified in the kind-cluster.yaml files to check if nginx is running. I do this using both a curl command opening localhost:30080 in my browser.



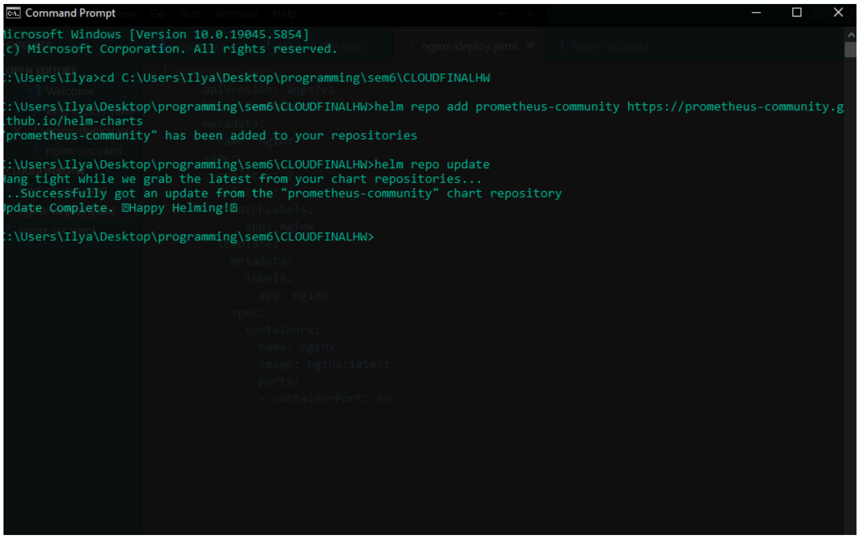
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

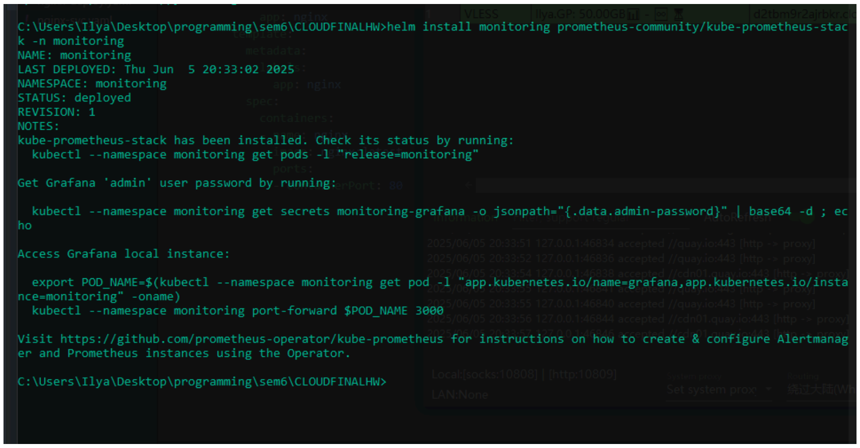
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

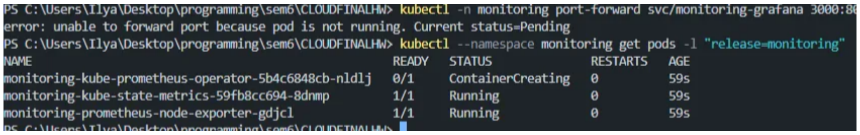
Up next we use helm to install applications for monitoring the cluster.



```
icrosoft Windows [Version 10.0.19045.5854]
c) Microsoft Corporation. All rights reserved.

:\Users\Ilya>cd C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW

:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo add prometheus-community https://prometheus-community.g
thub.io/helm-charts
prometheus-community" has been added to your repositories

:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo update
ang tight while we grab the latest from your chart repositories...
..Successfully got an update from the "prometheus-community" chart repository
pdate Complete. □Happy Helming!□

:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

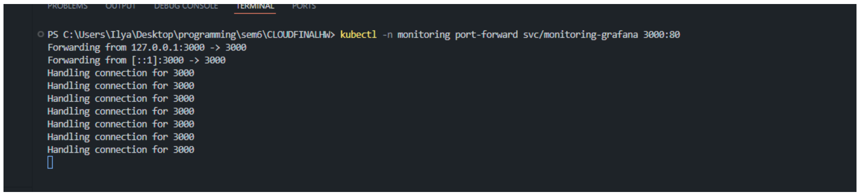


```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm install monitoring prometheus-community/kube-prometheus-stac
k -n monitoring
NAME: monitoring
LAST DEPLOYED: Thu Jun  5 20:33:02 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=monitoring"

Get Grafana 'admin' user password by running:

  kubectl --namespace monitoring get secrets monitoring-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; ec
ho

Access Grafana local instance:

  export POD_NAME=$(kubectl --namespace monitoring get pod -l "app.kubernetes.io/name=grafana,app.kubernetes.io/insta
nce=monitoring" -oname)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanag
er and Prometheus instances using the Operator.

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

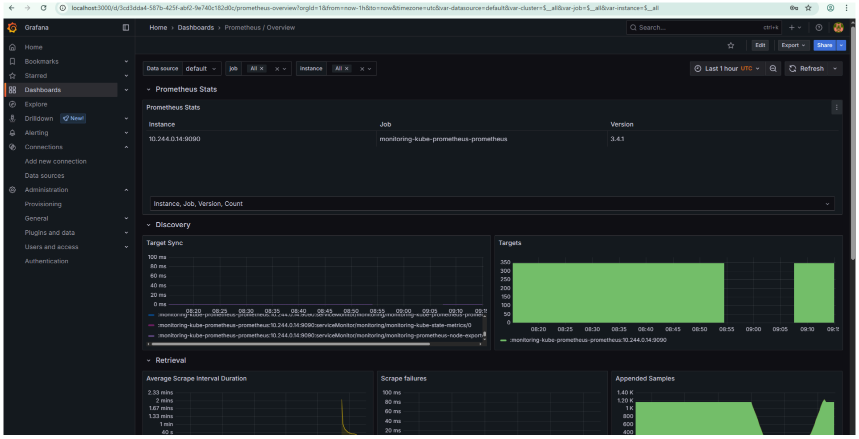


```
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl -n monitoring port-forward svc/monitoring-grafana 3000:8
error: unable to forward port because pod is not running. Current status=Pending
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl --namespace monitoring get pods -l "release=monitoring"
NAME                                                  READY   STATUS              RESTARTS   AGE
monitoring-kube-prometheus-operator-5b4c6848cb-nldlj   0/1     ContainerCreating   0          59s
monitoring-kube-state-metrics-59fb8cc694-8dnmp         1/1     Running             0          59s
monitoring-prometheus-node-exporter-gdjcl              1/1     Running             0          59s
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

After installing them using helm, we forward the ports and then go to port 3000 for graphana and port 9000 for prometheus



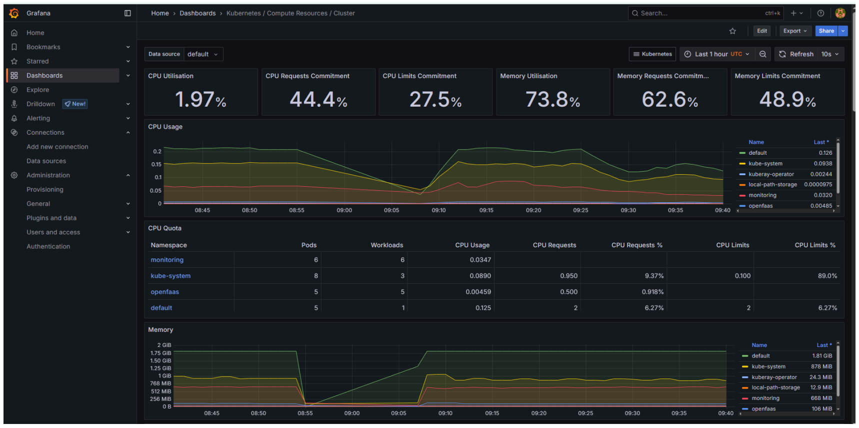
```
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl -n monitoring port-forward svc/monitoring-grafana 3000:80
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
```



Prometheus Stats

Instance	Job	Version
10.244.0.14:9090	monitoring-kube-prometheus-prometheus	3.4.1

Instance, Job, Version, Count



CPU Utilisation	CPU Requests Commitment	CPU Limits Commitment	Memory Utilisation	Memory Requests Commitm...	Memory Limits Commitment
1.97%	44.4%	27.5%	73.8%	62.6%	48.9%

CPU Usage

Name	Last *
default	0.126
kube-system	0.0938
kuberay-operator	0.00244
local-path-storage	0.0000975
monitoring	0.0320
openfaas	0.00485

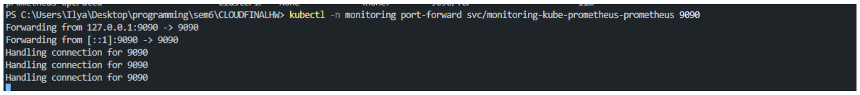
CPU Quota

Namespace	Pods	Workloads	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
monitoring	6	6	0.0347				
kube-system	8	3	0.0890	0.950	9.37%	0.100	89.0%
openfaas	5	5	0.00459	0.500	0.918%		
default	5	1	0.125	2	6.27%	2	6.27%

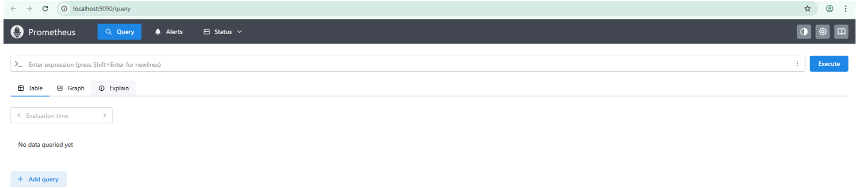
Memory

Name	Last *
default	1.81 GiB
kube-system	878 MiB
kuberay-operator	24.3 MiB
local-path-storage	12.9 MiB
monitoring	668 MiB
openfaas	106 MiB

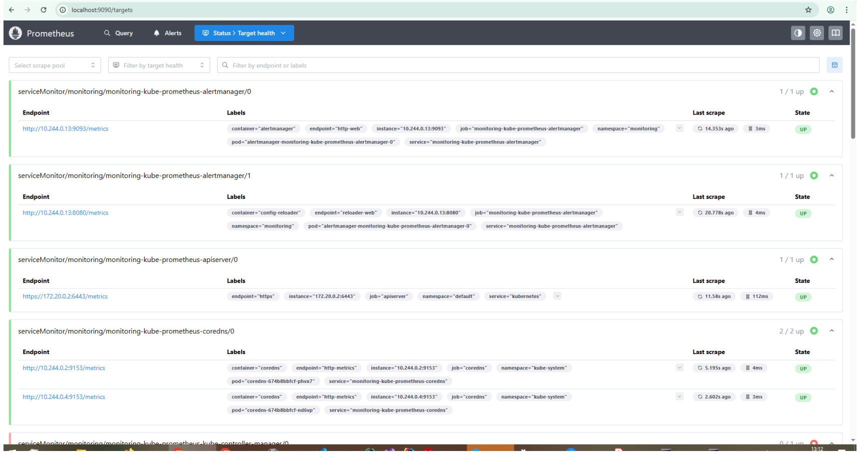
And then here is the prometheus dashboard (I took the screenshots after finishing part 4 of the homework)



```
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl -n monitoring port-forward svc/monitoring-kube-prometheus-prometheus 9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
```



No data queried yet



serviceMonitor/monitoring/monitoring-kube-prometheus-alertmanager/0 — 1 / 1 up

Endpoint	Labels	Last scrape	State
http://10.244.0.13:9093/metrics	container="alertmanager" endpoint="http-web" instance="10.244.0.13:9093" job="monitoring-kube-prometheus-alertmanager" namespace="monitoring" pod="alertmanager-monitoring-kube-prometheus-alertmanager-0" service="monitoring-kube-prometheus-alertmanager"	14.353s ago 3ms	UP

serviceMonitor/monitoring/monitoring-kube-prometheus-alertmanager/1 — 1 / 1 up

Endpoint	Labels	Last scrape	State
http://10.244.0.13:8080/metrics	container="config-reloader" endpoint="reloader-web" instance="10.244.0.13:8080" job="monitoring-kube-prometheus-alertmanager" namespace="monitoring" pod="alertmanager-monitoring-kube-prometheus-alertmanager-0" service="monitoring-kube-prometheus-alertmanager"	20.778s ago 4ms	UP

serviceMonitor/monitoring/monitoring-kube-prometheus-apiserver/0 — 1 / 1 up

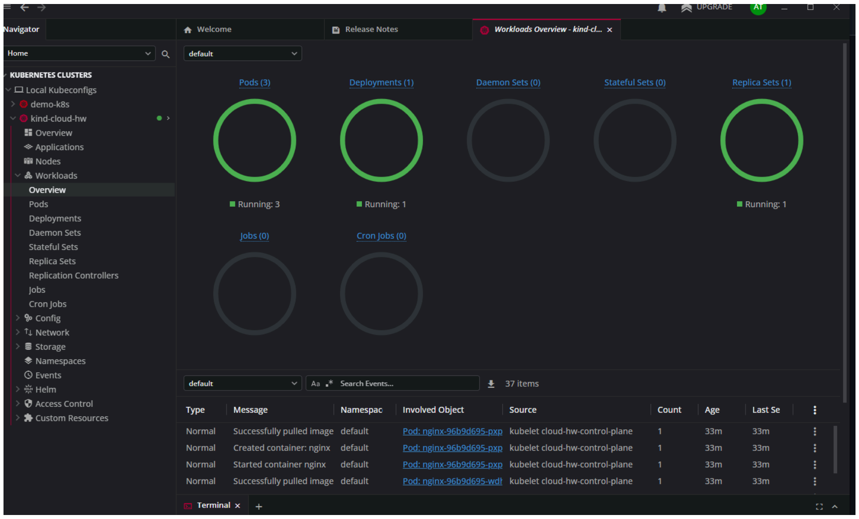
Endpoint	Labels	Last scrape	State
https://172.20.0.2:6443/metrics	endpoint="https" instance="172.20.0.2:6443" job="apiserver" namespace="default" service="kubernetes"	11.58s ago 112ms	UP

serviceMonitor/monitoring/monitoring-kube-prometheus-coredns/0 — 2 / 2 up

Endpoint	Labels	Last scrape	State
http://10.244.0.2:9153/metrics	container="coredns" endpoint="http-metrics" instance="10.244.0.2:9153" job="coredns" namespace="kube-system" pod="coredns-674b8bbfcf-phvx7" service="monitoring-kube-prometheus-coredns"	5.195s ago 4ms	UP
http://10.244.0.4:9153/metrics	container="coredns" endpoint="http-metrics" instance="10.244.0.4:9153" job="coredns" namespace="kube-system" pod="coredns-674b8bbfcf-nd6vp" service="monitoring-kube-prometheus-coredns"	2.602s ago 3ms	UP

serviceMonitor/monitoring/monitoring-kube-prometheus-kube-controller-manager/0 — 0 / 1 up

And finally here is the lens dashboard in the overview tab. The cluster name was `kind-cloud-hw` :



Type	Message	Namespac	Involved Object	Source	Count	Age	Last Se
Normal	Successfully pulled image	default	Pod: nginx-96b9d695-pxp	kubelet cloud-hw-control-plane	1	33m	33m
Normal	Created container: nginx	default	Pod: nginx-96b9d695-pxp	kubelet cloud-hw-control-plane	1	33m	33m
Normal	Started container nginx	default	Pod: nginx-96b9d695-pxp	kubelet cloud-hw-control-plane	1	33m	33m
Normal	Successfully pulled image	default	Pod: nginx-96b9d695-wdł	kubelet cloud-hw-control-plane	1	33m	33m

In this homework kind seems to act as a Kubernetes-construction kit that uses **one Docker container per Kubernetes node**.

Inside each of those node-containers lives a second container runtime, and *that* runtime pulls and runs the workload images (Nginx, Prometheus, etc.).

```
┌──────────────────────────────────────────────────────────────────────┐
│ Windows host (WSL-2 kernel)                          │
│   └── Docker Desktop daemon                           │
│       └── Docker container ➔ kind node    (kindest/node:v1.33.1 image) │
│           • runs kube-apiserver, controller-manager, scheduler, etcd       │
│           • runs containerd                          │
│               └── Kubernetes Pods ▶ cgroups/namespaces in the *same kernel* │
│                   • nginx:latest                     │
│                   • grafana/grafana:10.x                  │
│                   • quay.io/prometheus/prometheus:2.x             │
└──────────────────────────────────────────────────────────────────────┘
```

Layer-by-layer

Layer	Artifact	Purpose

Outer container	`kindest/node` **image** (~1 GB)	Provides a minimal Linux distro plus all Kubernetes control-plane binaries and containerd.
Inner containers (Pods)	e.g. `nginx:latest`, `grafana/grafana`	These are the workloads you deploy with `kubectl apply`.
Images "inside images"	Pulled into `/var/lib/containerd` **inside** the node-container	They're invisible to the outer Docker daemon; that's why `docker images` on the host doesn't show them.

Bottom line:

kind arranges **containers inside a privileged container** to simulate real Kubernetes nodes, letting you run a full cluster with zero hypervisors or cloud VMs—perfect for local experiments like the homework you just finished.

For monitoring the cluster, we used Helm and Graphana which we installed using helm. Later on, we also used Lens.

Helm

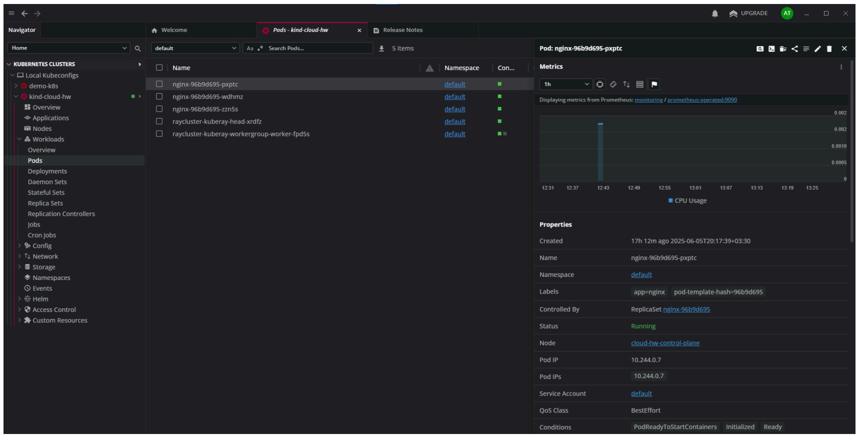
Helm is the package manager fgor Kuberneetes.

- `helm repo add prometheus-community … && helm repo update` – adds chart index.
- `kubectl create namespace monitoring` – isolates the monitoring stack.
- `helm install monitoring prometheus-community/kube-prometheus-stack -n monitoring` – renders ~200 manifests (CRDs, ServiceMonitors, StatefulSets, etc.) and applies them as a **Helm release** called *monitoring*.

Why Helm is handy: upgrading or deleting the stack later is a single `helm upgrade` / `helm uninstall` command instead of manually editing hundreds of YAML objects.

Lens

This seems like a decent tool to monitor the kubernetes cluster. It gives stats and pretty much unifies everything one needs to know about the cluster into an easy to use UI. So instead of just spamming `kubectl` I can just go to the ui and view everything there. like i can enter a `kubectl` command or just go to the workloads>Pods tab and view the available pods and their stats there:



Name	Namespace	Con...
nginx-96b9d695-pxptc	default	🟩
nginx-96b9d695-wdhmz	default	🟩
nginx-96b9d695-zzn5s	default	🟩
raycluster-kuberay-head-xrdfz	default	🟩
raycluster-kuberay-workergroup-worker-fpd5s	default	🟩 ▪

Pod: nginx-96b9d695-pxptc

Metrics

Displaying metrics from Prometheus: monitoring / prometheus-operated:9090

Properties

Created	17h 12m ago 2025-06-05T20:17:39+03:30
Name	nginx-96b9d695-pxptc
Namespace	default

Labels	app=nginx
	pod-template-hash=96b9d695

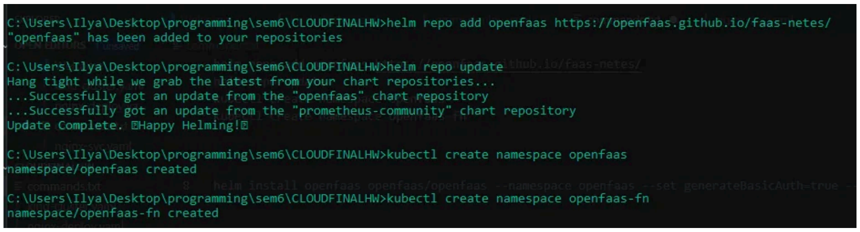
Controlled By	ReplicaSet nginx-96b9d695
Status	Running
Node	cloud-hw-control-plane
Pod IP	10.244.0.7
Pod IPs	10.244.0.7
Service Account	default
QoS Class	BestEffort
Conditions	PodReadyToStartContainers Initialized Ready

Alright then, that is it for part 1 & 2 of the homework. Moving on…

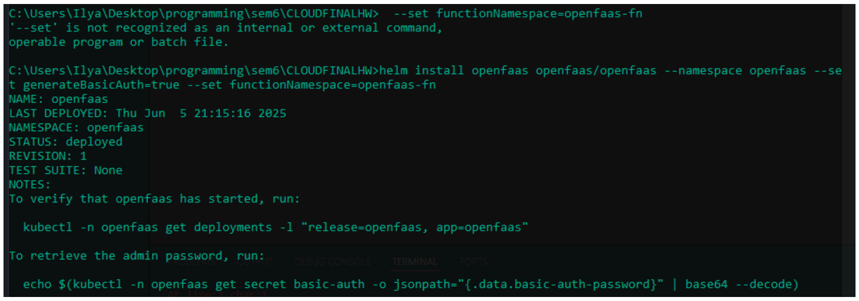
Part 3

I used OpenFaaS for the severless platform.

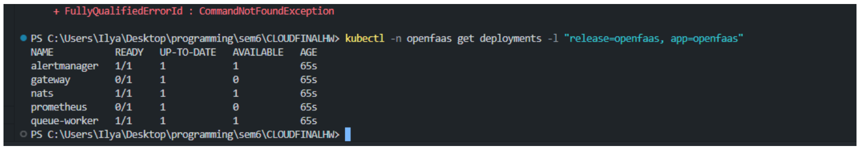
We used the helm package manager for kubernetes to install OpenFaaS.



```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo add openfaas https://openfaas.github.io/faas-netes/
"openfaas" has been added to your repositories

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "openfaas" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. □Happy Helming!□

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl create namespace openfaas
namespace/openfaas created

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl create namespace openfaas-fn
namespace/openfaas-fn created
```

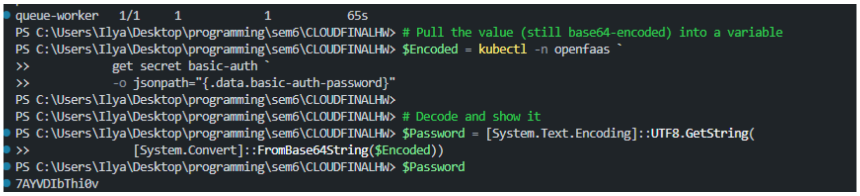


```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>  --set functionNamespace=openfaas-fn
'--set' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm install openfaas openfaas/openfaas --namespace openfaas --se
t generateBasicAuth=true --set functionNamespace=openfaas-fn
NAME: openfaas
LAST DEPLOYED: Thu Jun  5 21:15:16 2025
NAMESPACE: openfaas
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
To verify that openfaas has started, run:

  kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"

To retrieve the admin password, run:

  echo $(kubectl -n openfaas get secret basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode)
```



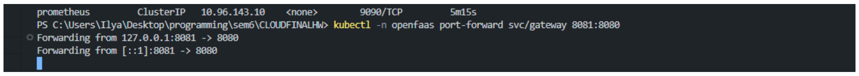
```
     + FullyQualifiedErrorId : CommandNotFoundException

● PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"
  NAME           READY   UP-TO-DATE   AVAILABLE   AGE
  alertmanager   1/1     1            1           65s
  gateway        0/1     1            0           65s
  nats           1/1     1            1           65s
  prometheus     0/1     1            0           65s
  queue-worker   1/1     1            1           65s
○ PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

As we can see in the above image, after installing OpenFaaS, the platform tells us that we can get the password using the command `echo $(kubectl -n openfaas get secret basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode)` however, that command doesn't properly work for windows so i got another command that does the job shown in the below image.



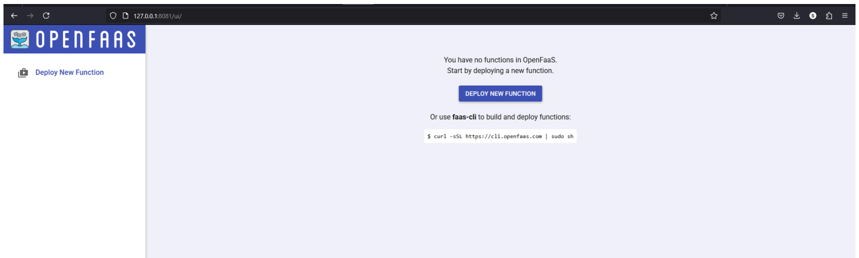
```
● queue-worker   1/1     1            1           65s
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> # Pull the value (still base64-encoded) into a variable
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> $Encoded = kubectl -n openfaas `
  >>             get secret basic-auth `
  >>             -o jsonpath="{.data.basic-auth-password}"
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> # Decode and show it
● PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> $Password = [System.Text.Encoding]::UTF8.GetString(
● >>               [System.Convert]::FromBase64String($Encoded))
● PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> $Password
● 7AYVDIbThi0v
```

Okay so we now have a password that we can use for our admin serverless platform later on.



```
  prometheus        ClusterIP   10.96.143.10    <none>        9090/TCP        5m15s
  PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl -n openfaas port-forward svc/gateway 8081:8080
○ Forwarding from 127.0.0.1:8081 -> 8080
  Forwarding from [::1]:8081 -> 8080
```

Finally, we forward the port 8080 from the contrainer to 8081 so that we can access the openfaas UI via our browser.



You have no functions in OpenFaaS.
Start by deploying a new function.

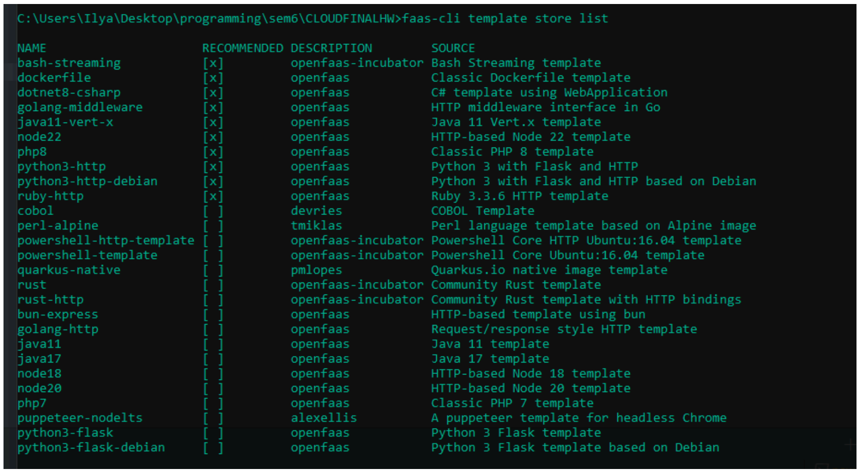
Or use **faas-cli** to build and deploy functions:

```
$ curl -sSL https://cli.openfaas.com | sudo sh
```

using the username `admin` adn the password given to us we login to the UI. We can see there are currently no functions available. So now we have to write a simple function and deploy it on the serverless platform.

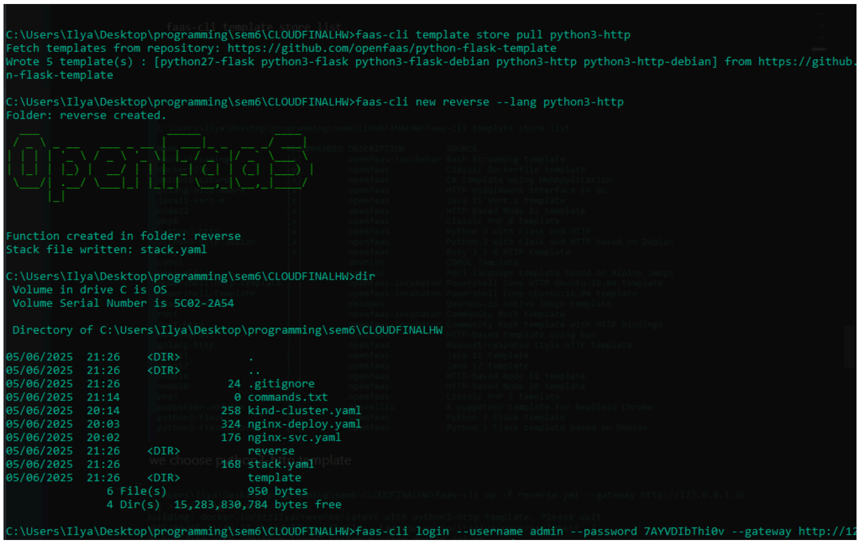
we go to the store list to see a bunch of available templates to create a function:

```
faas-cli template store list
```



```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>faas-cli template store list

NAME                     RECOMMENDED DESCRIPTION        SOURCE
bash-streaming           [x]         openfaas-incubator Bash Streaming template
dockerfile               [x]         openfaas           Classic Dockerfile template
dotnet8-csharp           [x]         openfaas           C# template using WebApplication
golang-middleware        [x]         openfaas           HTTP middleware interface in Go
java11-vert-x            [x]         openfaas           Java 11 Vert.x template
node22                   [x]         openfaas           HTTP-based Node 22 template
php8                     [x]         openfaas           Classic PHP 8 template
python3-http             [x]         openfaas           Python 3 with Flask and HTTP
python3-http-debian      [x]         openfaas           Python 3 with Flask and HTTP based on Debian
ruby-http                [x]         openfaas           Ruby 3.3.6 HTTP template
cobol                    [ ]         devries            COBOL Template
perl-alpine              [ ]         tmiklas            Perl language template based on Alpine image
powershell-http-template [ ]         openfaas-incubator Powershell Core HTTP Ubuntu:16.04 template
powershell-template      [ ]         openfaas-incubator Powershell Core Ubuntu:16.04 template
quarkus-native           [ ]         pmlopes            Quarkus.io native image template
rust                     [ ]         openfaas-incubator Community Rust template
rust-http                [ ]         openfaas-incubator Community Rust template with HTTP bindings
bun-express              [ ]         openfaas           HTTP-based template using bun
golang-http              [ ]         openfaas           Request/response style HTTP template
java11                   [ ]         openfaas           Java 11 template
java17                   [ ]         openfaas           Java 17 template
node18                   [ ]         openfaas           HTTP-based Node 18 template
node20                   [ ]         openfaas           HTTP-based Node 20 template
php7                     [ ]         openfaas           Classic PHP 7 template
puppeteer-nodelts        [ ]         alexellis          A puppeteer template for headless Chrome
python3-flask            [ ]         openfaas           Python 3 Flask template
python3-flask-debian     [ ]         openfaas           Python 3 Flask template based on Debian
```

we choose python3-http template



```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>faas-cli template store pull python3-http
Fetch templates from repository: https://github.com/openfaas/python-flask-template
Wrote 5 template(s) : [python27-flask python3-flask python3-flask-debian python3-http python3-http-debian] from https://github.
n-flask-template

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>faas-cli new reverse --lang python3-http
Folder: reverse created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: reverse
Stack file written: stack.yaml

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>dir
 Volume in drive C is OS
 Volume Serial Number is 5C02-2A54

 Directory of C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW

05/06/2025  21:26    <DIR>          .
05/06/2025  21:26    <DIR>          ..
05/06/2025  21:26                24 .gitignore
05/06/2025  21:14                 0 commands.txt
05/06/2025  20:14               258 kind-cluster.yaml
05/06/2025  20:03               324 nginx-deploy.yaml
05/06/2025  20:02               176 nginx-svc.yaml
05/06/2025  21:26    <DIR>          reverse
05/06/2025  21:26               168 stack.yaml
05/06/2025  21:26    <DIR>          template
               6 File(s)            950 bytes
               4 Dir(s)  15,283,830,784 bytes free

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>faas-cli login --username admin --password 7AYVDIbThi0v --gateway http://12
```

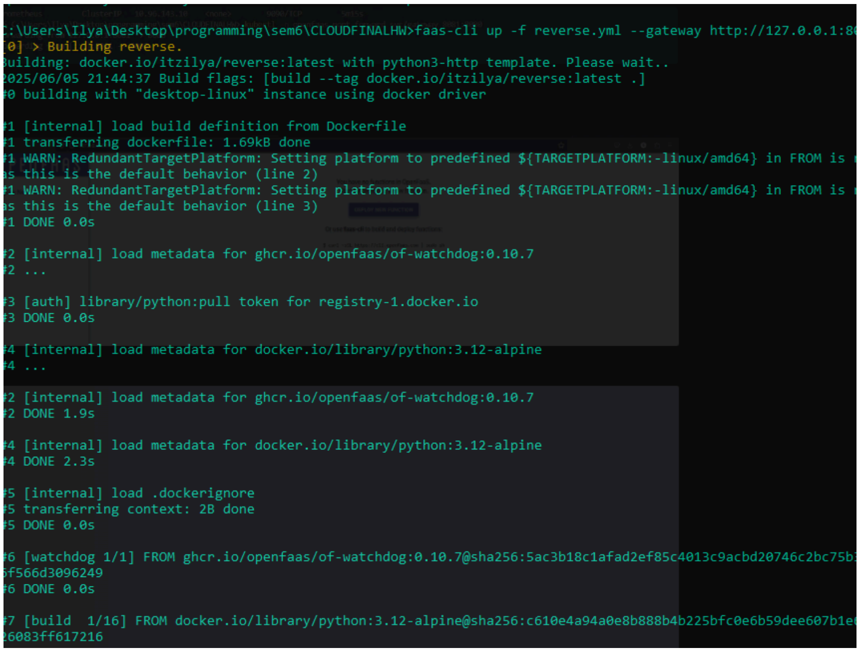
okay the template creates a directory named reverse which has a sample handler inside and also creates a **stack.yaml file which i renamed to reverse.yml.**

reverse.yml (stack.yaml)

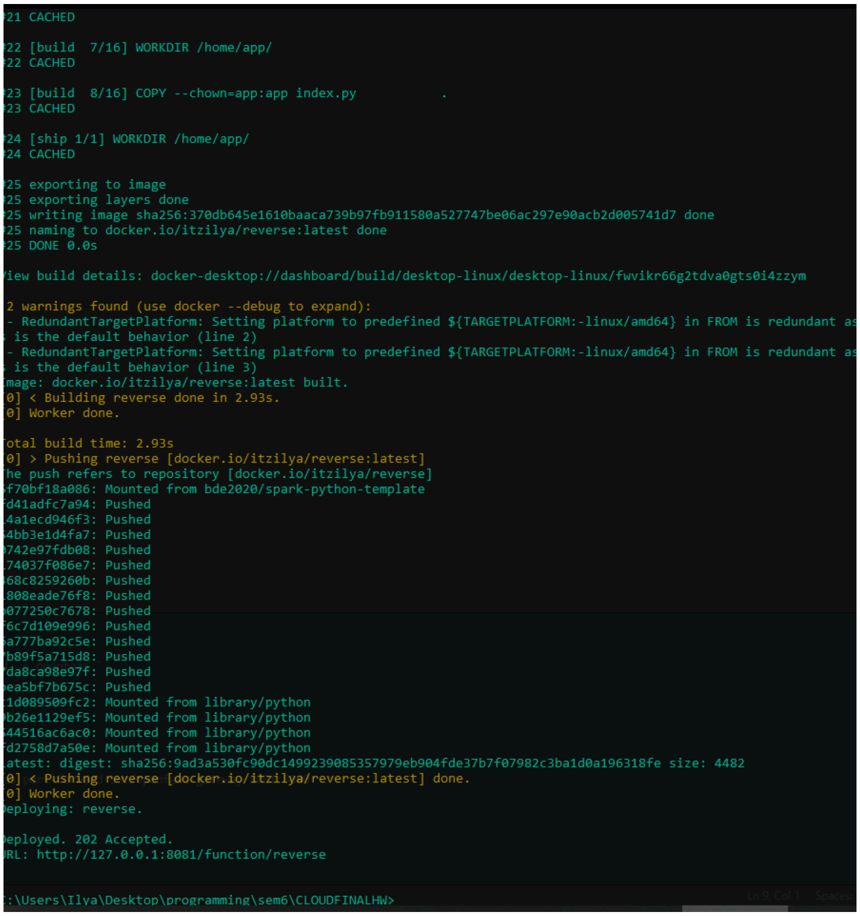
```
version: 1.0
provider:
  name: openfaas
functions:
  reverse:
    lang: python3-http
    handler: ./reverse
    image: docker.io/itzilya/reverse:latest
```

here is a simple handler function that just reverse the string in the http request it receives and returns it.

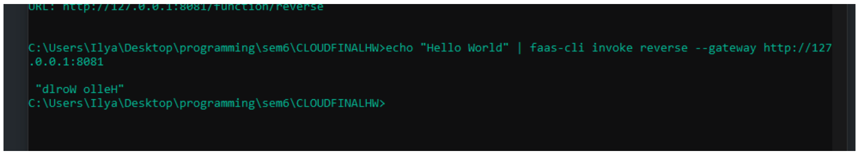
```
def handle(event, context):
    """
    Receives a string in the HTTP body and returns it reversed.
    """
    return event.body[::-1]
```



```
:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>faas-cli up -f reverse.yml --gateway http://127.0.0.1:8
0] > Building reverse.
uilding: docker.io/itzilya/reverse:latest with python3-http template. Please wait..
025/06/05 21:44:37 Build flags: [build --tag docker.io/itzilya/reverse:latest .]
0 building with "desktop-linux" instance using docker driver

1 [internal] load build definition from Dockerfile
1 transferring dockerfile: 1.69kB done
1 WARN: RedundantTargetPlatform: Setting platform to predefined ${TARGETPLATFORM:-linux/amd64} in FROM is
s this is the default behavior (line 2)
1 WARN: RedundantTargetPlatform: Setting platform to predefined ${TARGETPLATFORM:-linux/amd64} in FROM is
s this is the default behavior (line 3)
1 DONE 0.0s

2 [internal] load metadata for ghcr.io/openfaas/of-watchdog:0.10.7
2 ...

3 [auth] library/python:pull token for registry-1.docker.io
3 DONE 0.0s

4 [internal] load metadata for docker.io/library/python:3.12-alpine
4 ...

2 [internal] load metadata for ghcr.io/openfaas/of-watchdog:0.10.7
2 DONE 1.9s

4 [internal] load metadata for docker.io/library/python:3.12-alpine
4 DONE 2.3s

5 [internal] load .dockerignore
5 transferring context: 2B done
5 DONE 0.0s

6 [watchdog 1/1] FROM ghcr.io/openfaas/of-watchdog:0.10.7@sha256:5ac3b18c1afad2ef85c4013c9acbd20746c2bc75b
f566d3096249
6 DONE 0.0s

7 [build  1/16] FROM docker.io/library/python:3.12-alpine@sha256:c610e4a94a0e8b888b4b225bfc0e6b59dee607b1e
6083ff617216
```



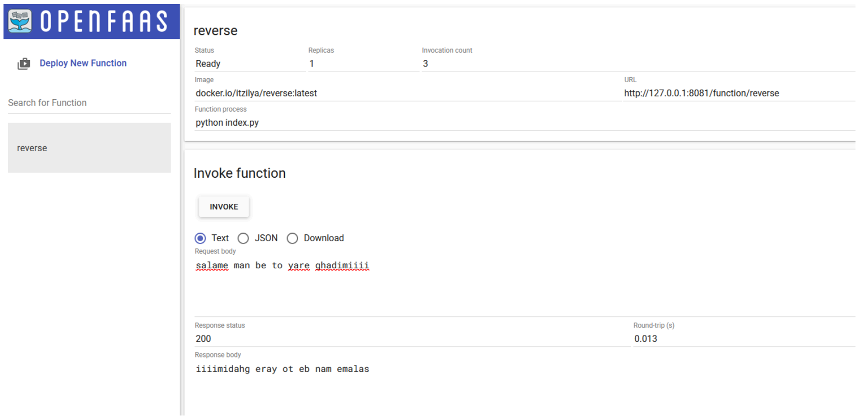
```
21 CACHED

22 [build  7/16] WORKDIR /home/app/
22 CACHED

23 [build  8/16] COPY --chown=app:app index.py           .
23 CACHED

24 [ship 1/1] WORKDIR /home/app/
24 CACHED

25 exporting to image
25 exporting layers done
25 writing image sha256:370db645e1610baaca739b97fb911580a527747be06ac297e90acb2d005741d7 done
25 naming to docker.io/itzilya/reverse:latest done
25 DONE 0.0s

iew build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/fwvikr66g2tdva0gts0i4zzym

 2 warnings found (use docker --debug to expand):
 - RedundantTargetPlatform: Setting platform to predefined ${TARGETPLATFORM:-linux/amd64} in FROM is redundant as
  is the default behavior (line 2)
 - RedundantTargetPlatform: Setting platform to predefined ${TARGETPLATFORM:-linux/amd64} in FROM is redundant as
  is the default behavior (line 3)
mage: docker.io/itzilya/reverse:latest built.
0] < Building reverse done in 2.93s.
0] Worker done.

otal build time: 2.93s
0] > Pushing reverse [docker.io/itzilya/reverse:latest]
he push refers to repository [docker.io/itzilya/reverse]
f70bf18a086: Mounted from bde2020/spark-python-template
d41adfc7a94: Pushed
4a1ecd946f3: Pushed
4bb3e1d4fa7: Pushed
742e97fdb08: Pushed
74037f086e7: Pushed
68c8259260b: Pushed
808eade76f8: Pushed
077250c7678: Pushed
6c7d109e996: Pushed
a777ba92c5e: Pushed
b89f5a715d8: Pushed
da8ca98e97f: Pushed
ea5bf7b675c: Pushed
1d089509fc2: Mounted from library/python
b26e1129ef5: Mounted from library/python
44516ac6ac0: Mounted from library/python
d2758d7a50e: Mounted from library/python
atest: digest: sha256:9ad3a530fc90dc1499239085357979eb904fde37b7f07982c3ba1d0a196318fe size: 4482
0] < Pushing reverse [docker.io/itzilya/reverse:latest] done.
0] Worker done.
eploying: reverse.

eployed. 202 Accepted.
RL: http://127.0.0.1:8081/function/reverse


:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```



```
URL: http://127.0.0.1:8081/function/reverse

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>echo "Hello World" | faas-cli invoke reverse --gateway http://127
.0.0.1:8081

 "dlroW olleH"
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

the above images show that we deployed the handler.

We then tested it in the final image by sending a string (“Hello World”) and say that it returned the reversed string.



reverse

Status: Ready
Replicas: 1
Invocation count: 3
Image: docker.io/itzilya/reverse:latest
URL: http://127.0.0.1:8081/function/reverse
Function process: python index.py

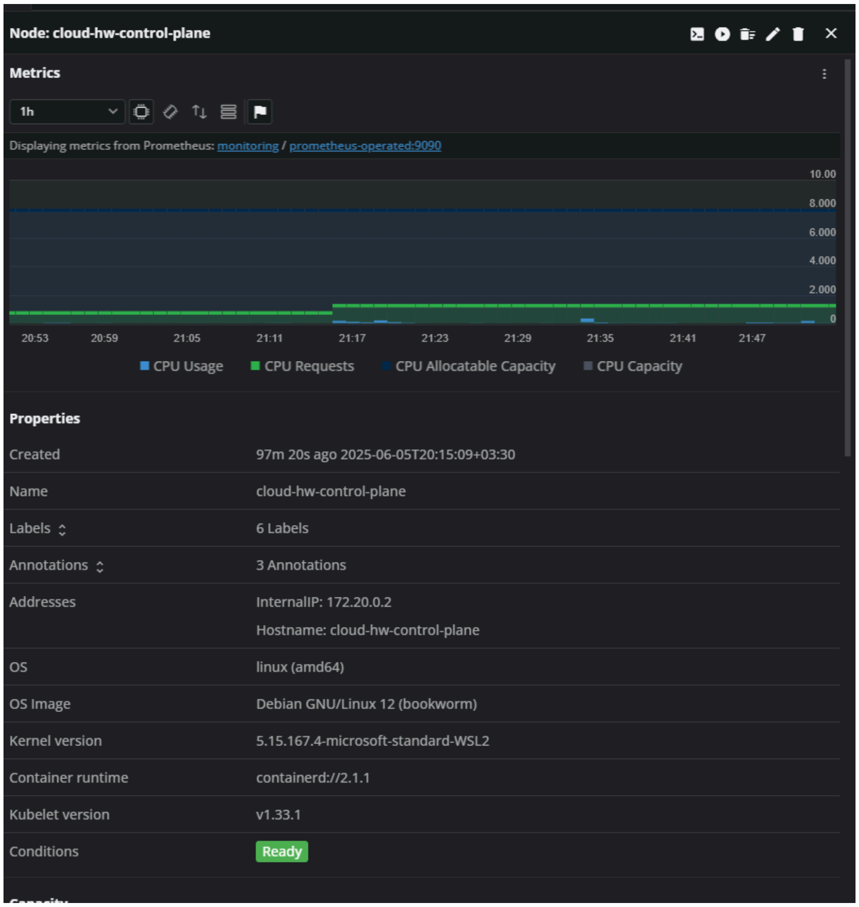
Invoke function

● Text ○ JSON ○ Download

Request body: salame man be to yare ghadimiiii

Response status: 200
Round-trip (s): 0.013
Response body: iiiimidahg eray ot eb nam emalas

We also tested the function using the OpenFaaS UI and saw that it worked.



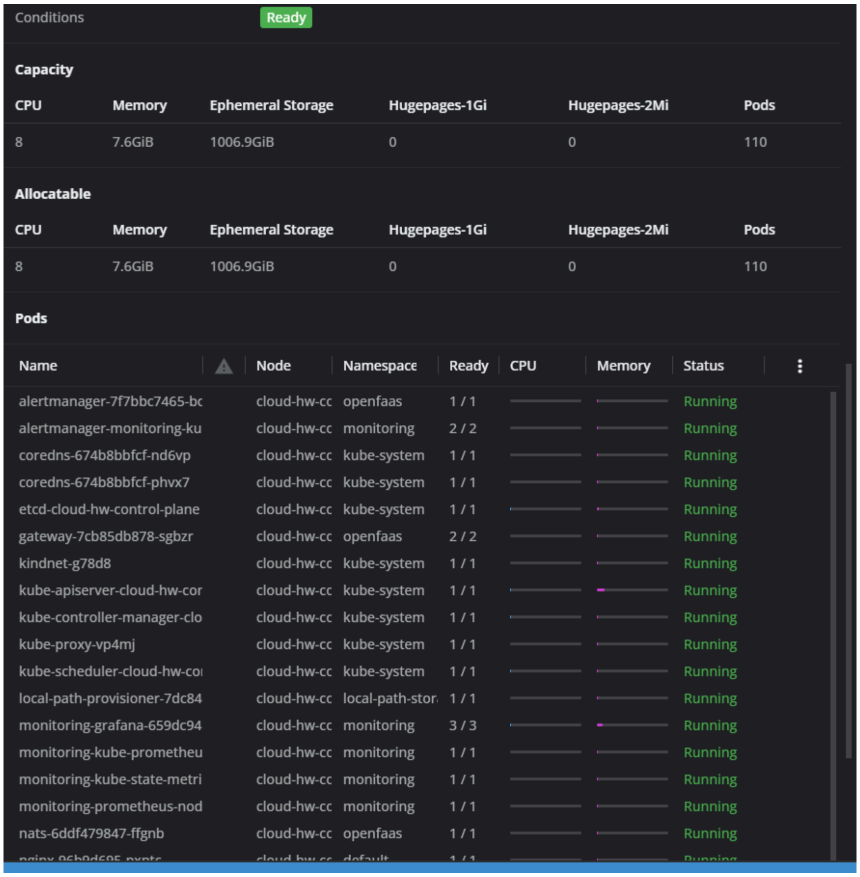
Node: cloud-hw-control-plane

Metrics

Properties

Created	97m 20s ago 2025-06-05T20:15:09+03:30
Name	cloud-hw-control-plane
Labels	6 Labels
Annotations	3 Annotations
Addresses	InternalIP: 172.20.0.2 Hostname: cloud-hw-control-plane
OS	linux (amd64)
OS Image	Debian GNU/Linux 12 (bookworm)
Kernel version	5.15.167.4-microsoft-standard-WSL2
Container runtime	containerd://2.1.1
Kubelet version	v1.33.1
Conditions	Ready

Capacity



Conditions Ready

Capacity

CPU	Memory	Ephemeral Storage	Hugepages-1Gi	Hugepages-2Mi	Pods
8	7.6GiB	1006.9GiB	0	0	110

Allocatable

CPU	Memory	Ephemeral Storage	Hugepages-1Gi	Hugepages-2Mi	Pods
8	7.6GiB	1006.9GiB	0	0	110

Pods

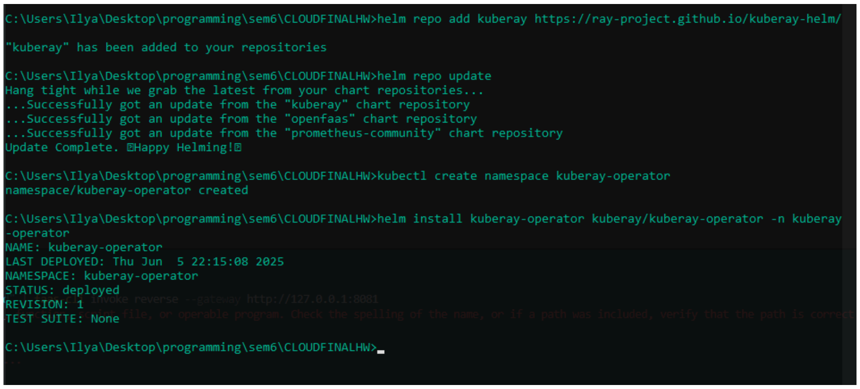
Name	⚠	Node	Namespace	Ready	CPU	Memory	Status
alertmanager-7f7bbc7465-bc		cloud-hw-cc	openfaas	1 / 1			Running
alertmanager-monitoring-ku		cloud-hw-cc	monitoring	2 / 2			Running
coredns-674b8bbfcf-nd6vp		cloud-hw-cc	kube-system	1 / 1			Running
coredns-674b8bbfcf-phvx7		cloud-hw-cc	kube-system	1 / 1			Running
etcd-cloud-hw-control-plane		cloud-hw-cc	kube-system	1 / 1			Running
gateway-7cb85db878-sgbzr		cloud-hw-cc	openfaas	2 / 2			Running
kindnet-g78d8		cloud-hw-cc	kube-system	1 / 1			Running
kube-apiserver-cloud-hw-cor		cloud-hw-cc	kube-system	1 / 1			Running
kube-controller-manager-clo		cloud-hw-cc	kube-system	1 / 1			Running
kube-proxy-vp4mj		cloud-hw-cc	kube-system	1 / 1			Running
kube-scheduler-cloud-hw-co		cloud-hw-cc	kube-system	1 / 1			Running
local-path-provisioner-7dc84		cloud-hw-cc	local-path-stor	1 / 1			Running
monitoring-grafana-659dc94		cloud-hw-cc	monitoring	3 / 3			Running
monitoring-kube-prometheu		cloud-hw-cc	monitoring	1 / 1			Running
monitoring-kube-state-metri		cloud-hw-cc	monitoring	1 / 1			Running
monitoring-prometheus-nod		cloud-hw-cc	monitoring	1 / 1			Running
nats-6ddf479847-ffgnb		cloud-hw-cc	openfaas	1 / 1			Running
nginx-96b9d695-nxptc		cloud-hw-cc	default	1 / 1			Running

The above shows the metrics for the control plane inside lens. (I didn't know what you really wanted to see when you said "check the cluster status in lens" since there a lot of tabs in lens but I just put the images anyway...)

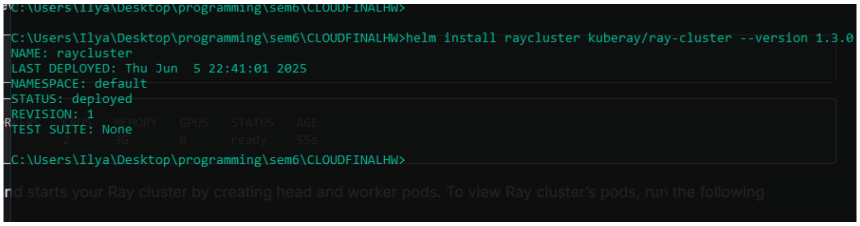
And that is it for part 3. We deployed openfaas on our cluster, uploaded a simple function that reverses a string sent to it via http, and then tested to see if it works

Part 4

To install ray on the cluster, we use helm:

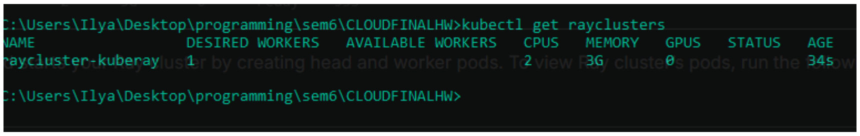


```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo add kuberay https://ray-project.github.io/kuberay-helm/

"kuberay" has been added to your repositories

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "kuberay" chart repository
...Successfully got an update from the "openfaas" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⎈Happy Helming!⎈

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl create namespace kuberay-operator
namespace/kuberay-operator created

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm install kuberay-operator kuberay/kuberay-operator -n kuberay
-operator
NAME: kuberay-operator
LAST DEPLOYED: Thu Jun  5 22:15:08 2025
NAMESPACE: kuberay-operator
STATUS: deployed
REVISION: 1
TEST SUITE: None

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```



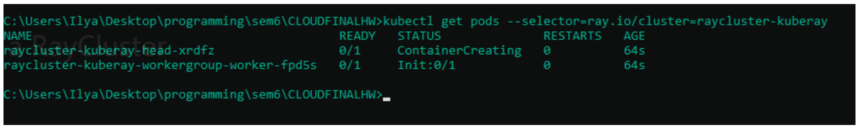
```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>helm install raycluster kuberay/ray-cluster --version 1.3.0
NAME: raycluster
LAST DEPLOYED: Thu Jun  5 22:41:01 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

Okay now we just check to see if we have a ray cluster:



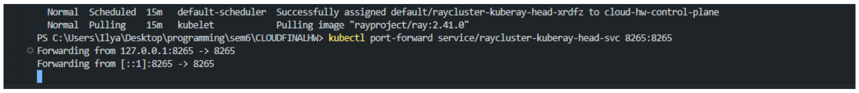
```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl get rayclusters
NAME                 DESIRED WORKERS   AVAILABLE WORKERS   CPUS   MEMORY   GPUS   STATUS   AGE
raycluster-kuberay   1                                     2      3G       0               34s

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

Everything seems to be okay so we checkout the pods:



```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl get pods --selector=ray.io/cluster=raycluster-kuberay
NAME                                       READY   STATUS              RESTARTS   AGE
raycluster-kuberay-head-xrdfz              0/1     ContainerCreating   0          64s
raycluster-kuberay-workergroup-worker-fpd5s   0/1     Init:0/1            0          64s

C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

At the time i took this image, the pods were not ready yet (I think they were still being pulled) so I waited like 15 minutes and then once they were ready, allowed access to port ***8265*** **(it was in the ray docks)**



```
  Normal  Scheduled  15m   default-scheduler  Successfully assigned default/raycluster-kuberay-head-xrdfz to cloud-hw-control-plane
  Normal  Pulling    15m   kubelet            Pulling image "rayproject/ray:2.41.0"
PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl port-forward service/raycluster-kuberay-head-svc 8265:8265
Forwarding from 127.0.0.1:8265 -> 8265
Forwarding from [::1]:8265 -> 8265
```

Now the following screenshots show the deployment of the ray job. TO my understanding, there are two ray pods working here. One of them is head node and the other one is the worker node. We can see both of them in the above image. I followed the official documentation of ray from here: https://docs.ray.io/en/latest/cluster/kubernetes/getting-started/raycluster-quick-start.html#kuberay-raycluster-quickstart which in step 4 in that page, it shows how to run an application on a ray cluster. I followed method 2 (https://docs.ray.io/en/latest/cluster/kubernetes/getting-started/raycluster-quick-start.html#method-2-submit-a-ray-job-to-the-raycluster-using-ray-job-submission-sdk**)** which allows us to "Submit a Ray job to the RayCluster using ray job submission SDK".

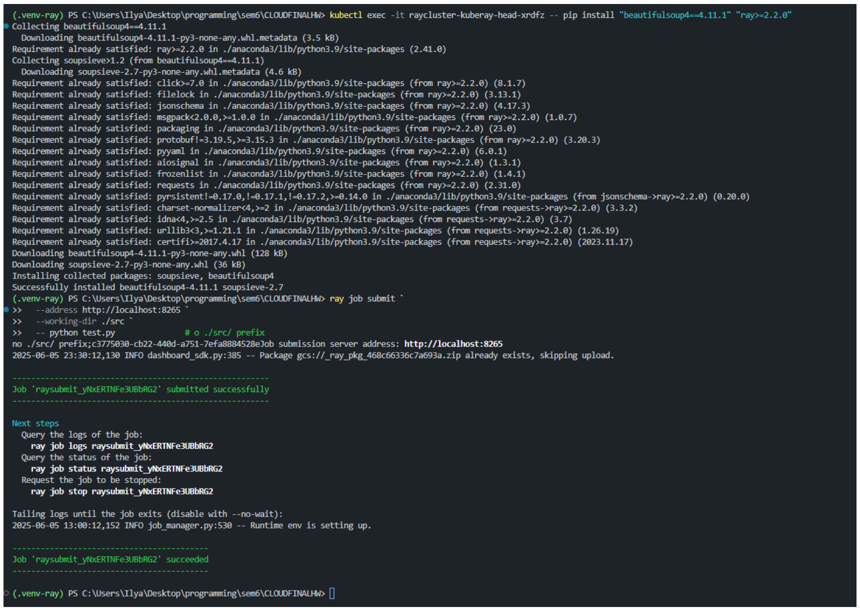
However, I changed the job to what you said in the homework description to follow an intermediate or advanced example within the ray docs.

below is the example I chose which is a webcrawler made to run on a ray cluster. I changed the code a bit from the documentation. (https://docs.ray.io/en/latest/ray-core/examples/web-crawler.html)

/src/test.py

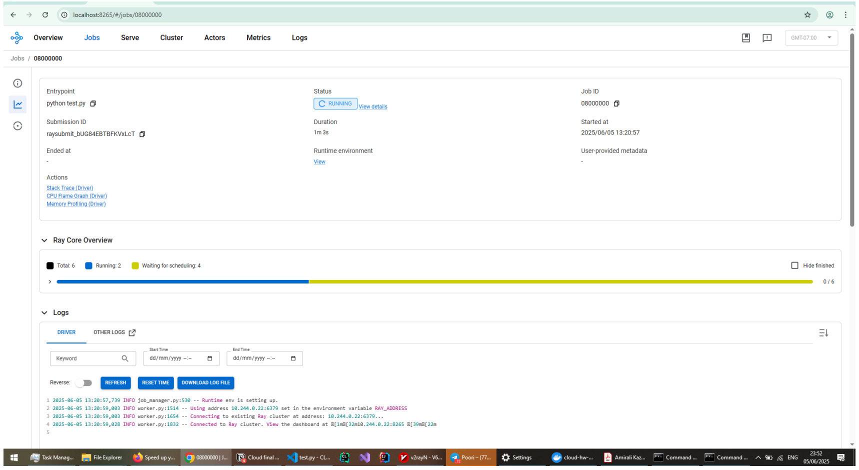
```
import sys, ray, requests
from bs4 import BeautifulSoup

def extract_links(elements, base_url, max_results=100):
    links = []
    for e in elements:
        url = e["href"]
        if "https://" not in url:
            url = base_url + url
        if base_url in url:
            links.append(url)
    return set(links[:max_results])

def find_links(start_url, base_url, depth=2):
    """Depth-first crawl (sequential)."""
    if depth == 0:
        return set()
    page = requests.get(start_url, timeout=10)
    soup = BeautifulSoup(page.content, "html.parser")
    links = extract_links(soup.find_all("a", href=True), base_url)
    for url in links.copy():
        links |= find_links(url, base_url, depth - 1)
    return links

# ---------------- RAY PART ----------------
@ray.remote
def find_links_task(start_url, base_url, depth=2):
    return find_links(start_url, base_url, depth)

if __name__ == "__main__":
    ray.init(address="auto")

    base = sys.argv[1] if len(sys.argv) > 1 else "https://docs.ray.io/en/latest/"
    # launch 6 crawlers in parallel
    tasks = [find_links_task.remote(f"{base}{suffix}", base)
             for suffix in ["", "", "rllib/index.html", "tune/index.html", "serve/index.html", "data/index.html"]]

    results = ray.get(tasks)
    for idx, links in enumerate(results, 1):
        print(f"Crawler {idx}: {len(links)} links")
```

Now the crawler uses the beautiful soup and the requests library so i had to make sure the dependencies were already installed on the ray pods which are the head node and the worker node (I'm not sure whether it was necessary to install the dependencies on the head node but I did it anyway)



```
(.venv-ray) PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl exec -it raycluster-kuberay-head-xrdfz -- pip install "beautifulsoup4==4.11.1" "ray>=2.2.0"
Collecting beautifulsoup4==4.11.1
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: ray>=2.2.0 in ./anaconda3/lib/python3.9/site-packages (2.41.0)
Collecting soupsieve>1.2 (from beautifulsoup4==4.11.1)
  Downloading soupsieve-2.7-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: click>=7.0 in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (8.1.7)
Requirement already satisfied: filelock in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (3.13.1)
Requirement already satisfied: jsonschema in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (4.17.3)
Requirement already satisfied: msgpack<2.0.0,>=1.0.0 in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (1.0.7)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (23.0)
Requirement already satisfied: protobuf!=3.19.5,>=3.15.3 in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (3.20.3)
Requirement already satisfied: pyyaml in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (6.0.1)
Requirement already satisfied: aiosignal in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (1.3.1)
Requirement already satisfied: frozenlist in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (1.4.1)
Requirement already satisfied: requests in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (2.31.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in ./anaconda3/lib/python3.9/site-packages (from jsonschema->ray>=2.2.0) (0.20.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./anaconda3/lib/python3.9/site-packages (from requests->ray>=2.2.0) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.9/site-packages (from requests->ray>=2.2.0) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./anaconda3/lib/python3.9/site-packages (from requests->ray>=2.2.0) (1.26.19)
Requirement already satisfied: certifi>=2017.4.17 in ./anaconda3/lib/python3.9/site-packages (from requests->ray>=2.2.0) (2023.11.17)
Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
Downloading soupsieve-2.7-py3-none-any.whl (36 kB)
Installing collected packages: soupsieve, beautifulsoup4
Successfully installed beautifulsoup4-4.11.1 soupsieve-2.7
(.venv-ray) PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> ray job submit `
>>   --address http://localhost:8265 `
>>   --working-dir ./src `
>>   -- python test.py                # o ./src/ prefix
no ./src/ prefix;c3775030-cb22-440d-a751-7efa8884528eJob submission server address: http://localhost:8265
2025-06-05 23:30:12,130 INFO dashboard_sdk.py:385 -- Package gcs://_ray_pkg_468c66336c7a693a.zip already exists, skipping upload.

-------------------------------------------------------
Job 'raysubmit_yNxERTNFe3UBbRG2' submitted successfully
-------------------------------------------------------

Next steps
  Query the logs of the job:
    ray job logs raysubmit_yNxERTNFe3UBbRG2
  Query the status of the job:
    ray job status raysubmit_yNxERTNFe3UBbRG2
  Request the job to be stopped:
    ray job stop raysubmit_yNxERTNFe3UBbRG2

Tailing logs until the job exits (disable with --no-wait):
2025-06-05 13:00:12,152 INFO job_manager.py:530 -- Runtime env is setting up.

------------------------------------------
Job 'raysubmit_yNxERTNFe3UBbRG2' succeeded
------------------------------------------

(.venv-ray) PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>
```

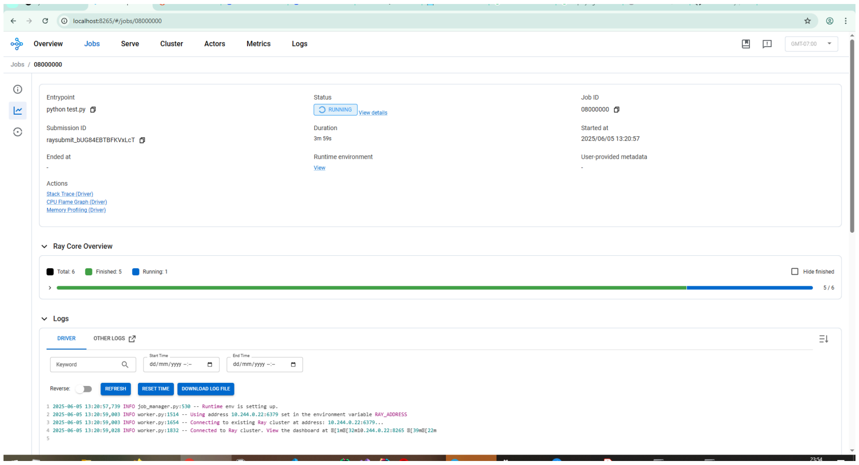
After a whole bunch of bugs and whole bunch of tries I finally got the job running which I tracked the progress through the ray UI as shown below. The job took about 4 minutes and the crawler went to almost a thousand links during that 5 minute period.



Entrypoint: python test.py
Status: RUNNING View details
Job ID: 08000000
Submission ID: raysubmit_bUG84EBTBFKVxLcT
Duration: 1m 3s
Started at: 2025/06/05 13:20:57
Ended at: -
Runtime environment: View
User-provided metadata: -

■ Total: 6 ■ Running: 2 ■ Waiting for scheduling: 4 □ Hide finished 0 / 6

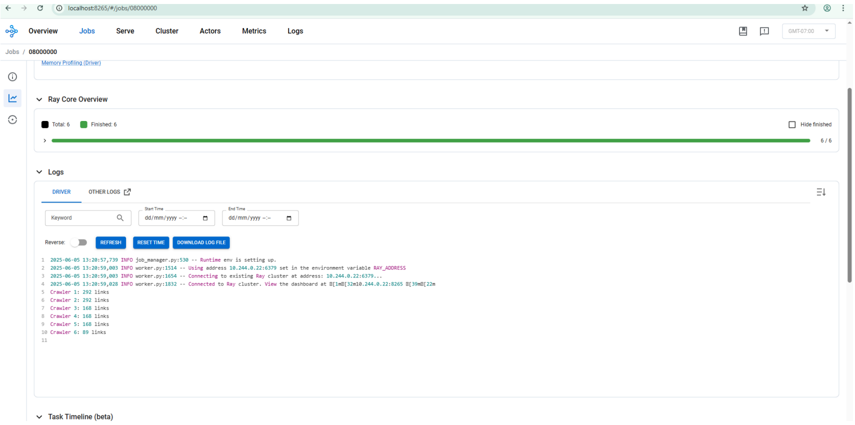
```
2025-06-05 13:20:57,739 INFO job_manager.py:530 -- Runtime env is setting up.
2025-06-05 13:20:59,003 INFO worker.py:1514 -- Using address 10.244.0.22:6379 set in the environment variable RAY_ADDRESS
2025-06-05 13:20:59,003 INFO worker.py:1654 -- Connecting to existing Ray cluster at address: 10.244.0.22:6379...
2025-06-05 13:20:59,028 INFO worker.py:1832 -- Connected to Ray cluster. View the dashboard at [1m[32m10.244.0.22:8265 [39m[22m

```



Entrypoint: python test.py
Status: RUNNING View details
Job ID: 08000000
Submission ID: raysubmit_bUG84EBTBFKVxLcT
Duration: 3m 59s
Started at: 2025/06/05 13:20:57
Ended at: -
Runtime environment: View
User-provided metadata: -

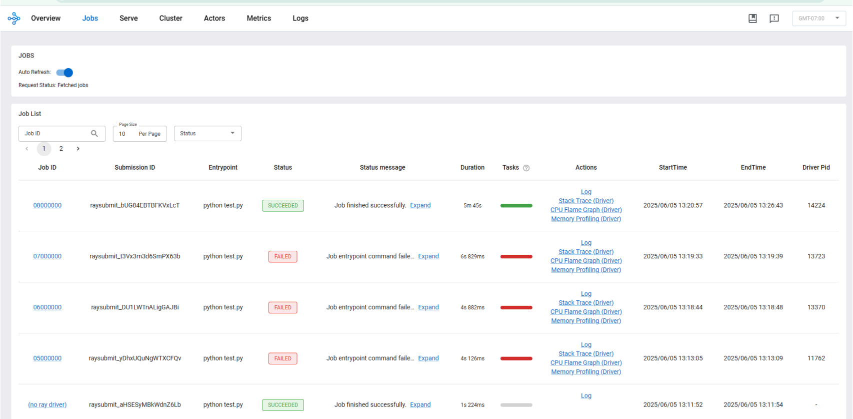
■ Total: 6 ■ Finished: 5 ■ Running: 1 □ Hide finished 5 / 6

```
2025-06-05 13:20:57,739 INFO job_manager.py:530 -- Runtime env is setting up.
2025-06-05 13:20:59,003 INFO worker.py:1514 -- Using address 10.244.0.22:6379 set in the environment variable RAY_ADDRESS
2025-06-05 13:20:59,003 INFO worker.py:1654 -- Connecting to existing Ray cluster at address: 10.244.0.22:6379...
2025-06-05 13:20:59,028 INFO worker.py:1832 -- Connected to Ray cluster. View the dashboard at [1m[32m10.244.0.22:8265 [39m[22m

```



■ Total: 6 ■ Finished: 6 □ Hide finished 6 / 6

```
2025-06-05 13:20:57,739 INFO job_manager.py:530 -- Runtime env is setting up.
2025-06-05 13:20:59,003 INFO worker.py:1514 -- Using address 10.244.0.22:6379 set in the environment variable RAY_ADDRESS
2025-06-05 13:20:59,003 INFO worker.py:1654 -- Connecting to existing Ray cluster at address: 10.244.0.22:6379...
2025-06-05 13:20:59,028 INFO worker.py:1832 -- Connected to Ray cluster. View the dashboard at [1m[32m10.244.0.22:8265 [39m[22m
Crawler 1: 292 links
Crawler 2: 292 links
Crawler 3: 168 links
Crawler 4: 168 links
Crawler 5: 168 links
Crawler 6: 89 links

```



JOBS

Auto Refresh:

Request Status: Fetched jobs

Job List

Job ID | Page Size 10 Per Page | Status

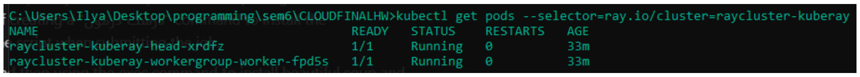
‹ 1 2 ›

Job ID	Submission ID	Entrypoint	Status	Status message	Duration	Tasks	Actions	StartTime	EndTime	Driver Pid
08000000	raysubmit_bUG84EBTBFKVxLcT	python test.py	SUCCEEDED	Job finished successfully. Expand	5m 45s	🟩	Log Stack Trace (Driver) CPU Flame Graph (Driver) Memory Profiling (Driver)	2025/06/05 13:20:57	2025/06/05 13:26:43	14224
07000000	raysubmit_t3Vx3m3d6SmPX63b	python test.py	FAILED	Job entrypoint command faile... Expand	6s 829ms	🟥	Log Stack Trace (Driver) CPU Flame Graph (Driver) Memory Profiling (Driver)	2025/06/05 13:19:33	2025/06/05 13:19:39	13723
06000000	raysubmit_DU1LWTnALigGAJBi	python test.py	FAILED	Job entrypoint command faile... Expand	4s 882ms	🟥	Log Stack Trace (Driver) CPU Flame Graph (Driver) Memory Profiling (Driver)	2025/06/05 13:18:44	2025/06/05 13:18:48	13370
05000000	raysubmit_yDhxUQuNgWTXCFQv	python test.py	FAILED	Job entrypoint command faile... Expand	4s 126ms	🟥	Log Stack Trace (Driver) CPU Flame Graph (Driver) Memory Profiling (Driver)	2025/06/05 13:13:05	2025/06/05 13:13:09	11762
(no ray driver)	raysubmit_aHSESyMBkWdnZ6Lb	python test.py	SUCCEEDED	Job finished successfully. Expand	1s 224ms	▨	Log	2025/06/05 13:11:52	2025/06/05 13:11:54	-

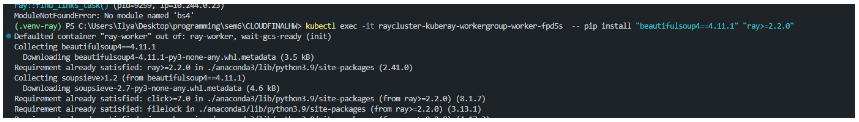
```
2025-06-05 13:20:57,739 INFO job_manager.py:530 -- Runtime env is setting up.
2025-06-05 13:20:59,003 INFO worker.py:1514 -- Using address 10.244.0.22:6379 set in the environment variable RAY_ADDRESS
2025-06-05 13:20:59,003 INFO worker.py:1654 -- Connecting to existing Ray cluster at address: 10.244.0.22:6379...
2025-06-05 13:20:59,028 INFO worker.py:1832 -- Connected to Ray cluster. View the dashboard at [1m[32m10.244.0.22:8265 [39m[22
Crawler 1: 292 links
Crawler 2: 292 links
Crawler 3: 168 links
Crawler 4: 168 links
Crawler 5: 168 links
Crawler 6: 89 links
```

not gonna lie I kinda played it the lazy way and just manually installed the packages in both the head and worker containers instead of creating a “درست درمون” command to install the requirements.txt file before running the script when submitting the job.

I did so by checking the pod names and then using the exec command to install beautiful soup and ray



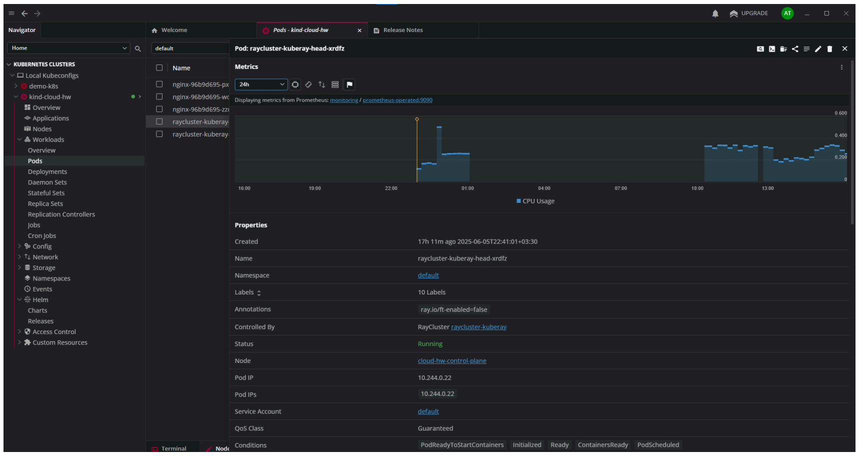
```
C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW>kubectl get pods --selector=ray.io/cluster=raycluster-kuberay
NAME                                         READY   STATUS    RESTARTS   AGE
raycluster-kuberay-head-xrdfz                1/1     Running   0          33m
raycluster-kuberay-workergroup-worker-fpd5s  1/1     Running   0          33m
```



```
ray::find_links_task() (pid=9259, ip=10.244.0.23)
ModuleNotFoundError: No module named 'bs4'
(.venv-ray) PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl exec -it raycluster-kuberay-workergroup-worker-fpd5s  -- pip install "beautifulsoup4==4.11.1" "ray>=2.2.0"
Defaulted container "ray-worker" out of: ray-worker, wait-gcs-ready (init)
Collecting beautifulsoup4==4.11.1
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: ray>=2.2.0 in ./anaconda3/lib/python3.9/site-packages (2.41.0)
Collecting soupsieve>1.2 (from beautifulsoup4==4.11.1)
  Downloading soupsieve-2.7-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: click>=7.0 in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (8.1.7)
Requirement already satisfied: filelock in ./anaconda3/lib/python3.9/site-packages (from ray>=2.2.0) (3.13.1)
```

```
<SNIP>

(.venv-ray) PS C:\Users\Ilya\Desktop\programming\sem6\CLOUDFINALHW> kubectl exec -it raycluster-kuberay-workergroup-worker-fp
Defaulted container "ray-worker" out of: ray-worker, wait-gcs-ready (init)
Collecting beautifulsoup4==4.11.1
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: ray>=2.2.0 in ./anaconda3/lib/python3.9/site-packages (2.41.0)
Collecting soupsieve>1.2 (from beautifulsoup4==4.11.1)
  Downloading soupsieve-2.7-py3-none-any.whl.metadata (4.6 kB)
<SNIP>
```

below is the status for the head of the ray cluster for the past 24 hours



Pod: raycluster-kuberay-head-xrdfz

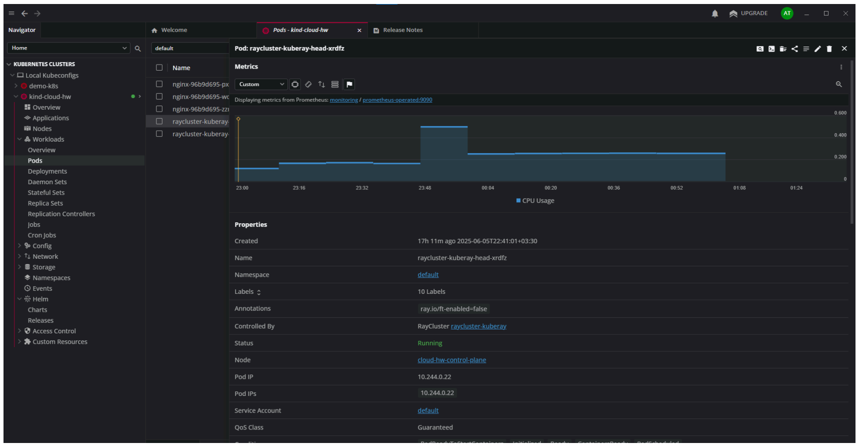
Metrics

Displaying metrics from Prometheus: monitoring / prometheus-operated:9090

Properties

Created	17h 11m ago 2025-06-05T22:41:01+03:30
Name	raycluster-kuberay-head-xrdfz
Namespace	default
Labels	10 Labels
Annotations	ray.io/ft-enabled=false
Controlled By	RayCluster raycluster-kuberay
Status	Running
Node	cloud-hw-control-plane
Pod IP	10.244.0.22
Pod IPs	10.244.0.22
Service Account	default
QoS Class	Guaranteed
Conditions	PodReadyToStartContainers Initialized Ready ContainersReady PodScheduled

and here is when i was running the job (around midnight)



Pod: raycluster-kuberay-head-xrdfz

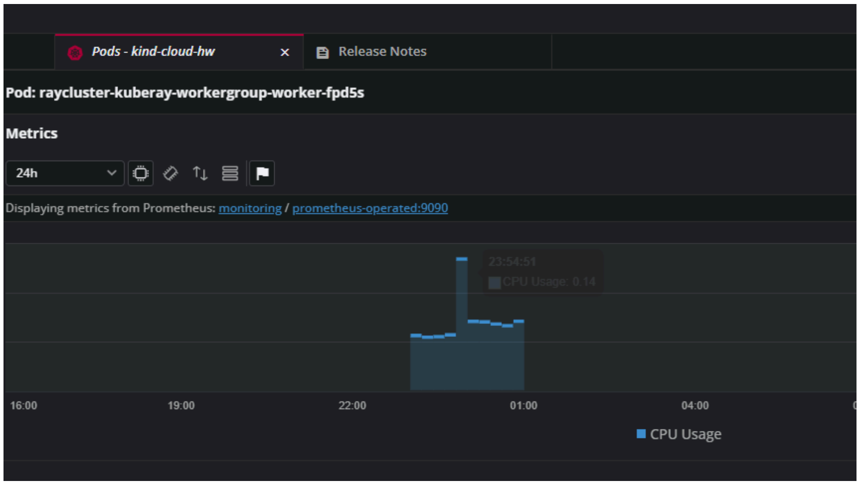
Metrics

Displaying metrics from Prometheus: monitoring / prometheus-operated:9090

Properties

Created	17h 11m ago 2025-06-05T22:41:01+03:30
Name	raycluster-kuberay-head-xrdfz
Namespace	default
Labels	10 Labels
Annotations	ray.io/ft-enabled=false
Controlled By	RayCluster raycluster-kuberay
Status	Running
Node	cloud-hw-control-plane
Pod IP	10.244.0.22
Pod IPs	10.244.0.22
Service Account	default
QoS Class	Guaranteed

and then the worker:



Pod: raycluster-kuberay-workergroup-worker-fpd5s

Metrics

Displaying metrics from Prometheus: monitoring / prometheus-operated:9090

and that is it... moving on to compare ray and serverless

Goal

ray is for stateful yet parallel applications that need constant memory access whereas serverless plafrorms such as OpenFaaS are for stateless event driven functions that need to scale up and down fast (in other words they are ephemeral)

Architecture

Ray is a cluster with 1 head pod and many worker pods.

the control plane uses gRPC and has a shared object store.

Has its own scheduler.

Serverless platforms usually have short-lived pods where most of the time each pod only runs a single function.

Pods become up and running when a function is called or when an event happens.

Execution Model

The entire application may be stateful that is why ray has both stateless tasks and stateful actors which can be called via an API.

in ray, you essentially call the decorated function (ray.remote) and that fucntion is then distributed and then executed.

if you need a value to proceed, you use ray.get()

In OpenFaaS, you make an HTTP request like we did with curl or opened it in our browser and then that request is handled by a pod.

Cold starts

Ray is always up, no cold starts and aslo features autoscaling.

openfaas has cold starts which can happen when an image is being pulled.

Support

ray is a bit more limited in terms of supporting different languages compared to OpenFaaS which we saw had a whole bunch of templates available for a whole bunch of different languages.

When to pick
Ray

Choose Ray if …	Example
You need **in-memory, iterative computation** over large data	Real-time advert bidding model that updates every few seconds using streaming features.
The job is **long-running** and latency between subtasks matters	Reinforcement-learning training loop that runs for hours, sharing a policy actor.
You want a **Pythonic parallel API** rather than HTTP endpoints	`ray.get([f.remote(x) for x in data])` feels like local `multiprocessing`, scales to a cluster.

when to pick **OpenFaaS**

Choose OpenFaaS if …	Example
Workloads are **bursty, request-/event-driven**	GitHub webhook triggers a lint-and-build function a few times per hour.
You need many **tiny, language-agnostic glue pieces**	Data-engineering cron jobs that call different cloud APIs, each under 30 s.
Cold-start penalty is acceptable and zero cost when idle is attractive	Image-thumbnailer that runs only when a user uploads a picture.