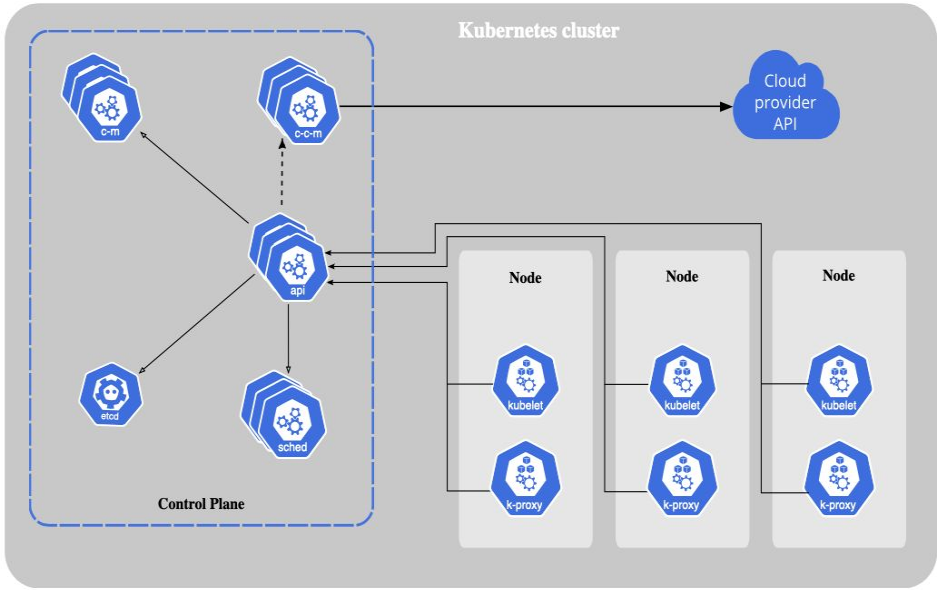
Cloud Computing Project : Kubernetes

Group 3 : Shuang Chen, Samuel D'Aprea, Lukas Atkinson, Louis-César Pagès.

Architecture and Concepts of Kubernetes



- Deployment
- StatefulSet
- Service
- Ingress

https://kubernetes.io/docs/concepts/overview/components/

Kubernetes setup

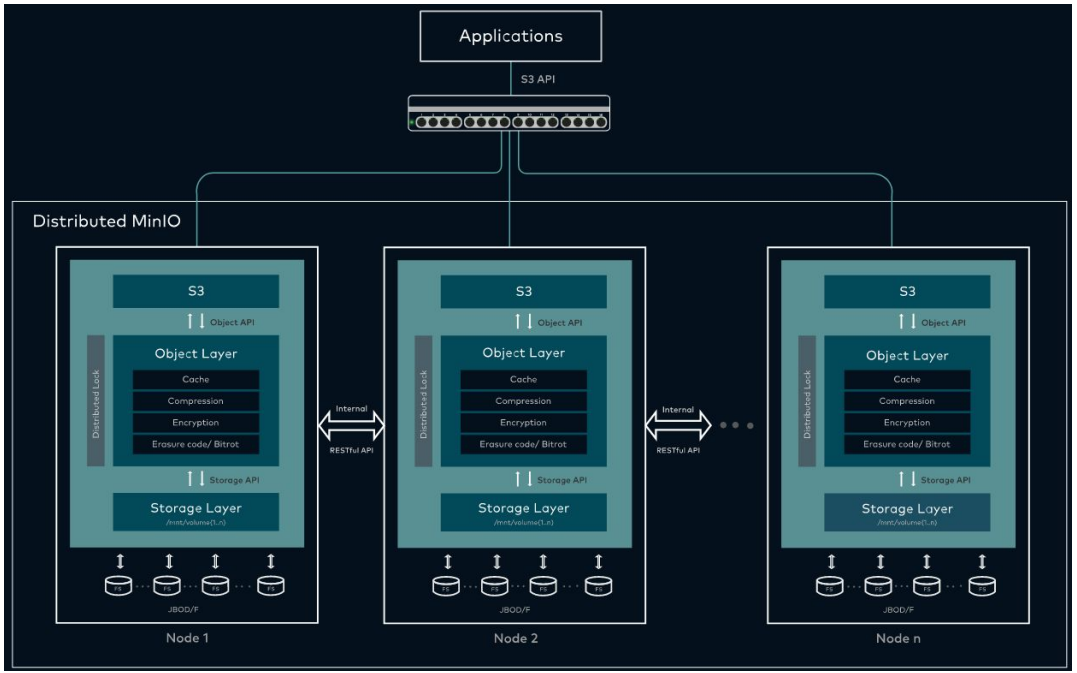
- Minikube
- Kubectl
- Interacting with the cluster

```
kubectl get pods -A
minikube dashboard

kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4
kubectl expose deployment hello-minikube --type=NodePort --port=8080

kubectl get services hello-minikube
minikube service hello-minikube
kubectl port-forward service/hello-minikube 7080:8080
```

Features and Architecture of MinIO

- **Open-source**
- High-performance **object storage**
- Max. supported object size of **5TB**
- **S3**-compatible API
- **Scalable** and **lightweight**
- Compression
- Server-side **encryption** (e.g. AES-256-GCM)
- **Bitrot** protection / **Erasure coding**
- **Client SDK**: **Java** and **Python**

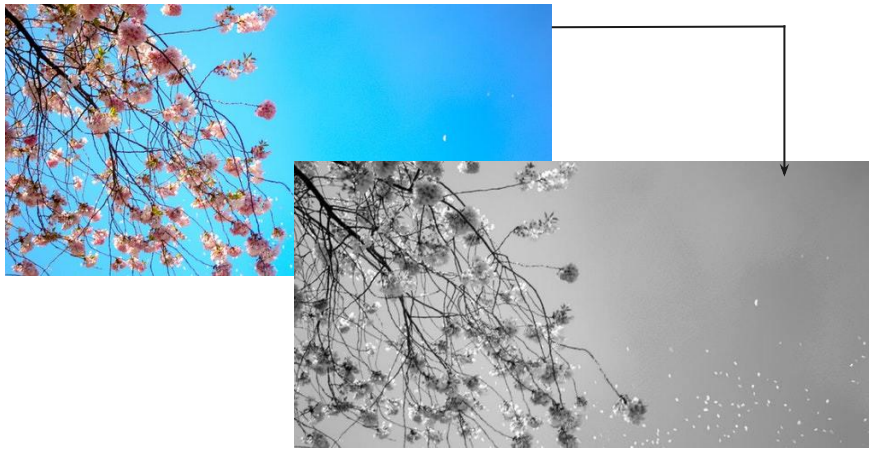
→ **Example** demo for using **MinIO** with **Python Client API**



https://min.io/product/overview#architecture

Which application is containerized ?

- REST API
  - GET /
  - POST /incoming
  - POST /processed
  - GET /processed/<name>
- Image processing App
- Python + Flask web framework
  + Minio SDK for Python
- JSON input



https://images.pexels.com/photos/72161/pexels-photo-72161.jpeg?dl&fit=crop&w=640&h=318

How is the App set up in Kubernetes Cluster ?

Container Registry `image:pkg.example.com/kube-image-processing`

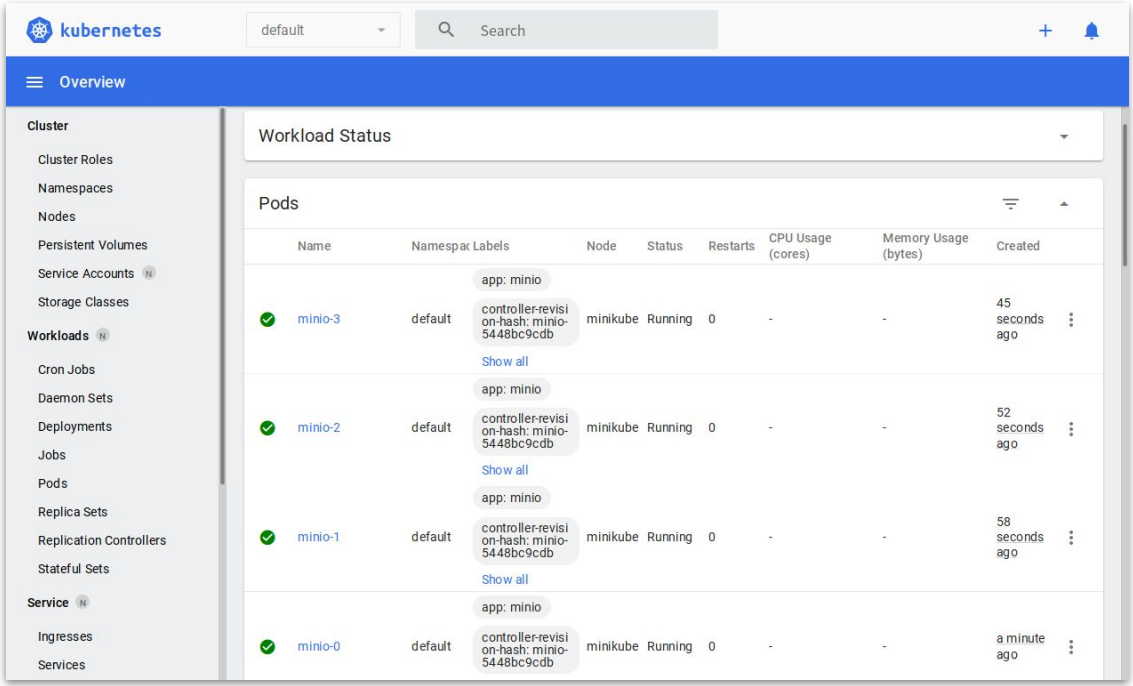
CI/CD

.gitlab-ci.yml 315 Bytes — Edit | Web IDE | Replace | Delete

```
stages:
    - build

docker-build:
  image: docker:latest
  stage: build
  services:
    - docker:dind
  before_script:
  - echo $CI_BUILD_TOKEN | docker login -u "$CI_REGISTRY_USER" --password-stdin  $CI_REGISTRY
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE" .
    - docker push "$CI_REGISTRY_IMAGE"
```

Demo!

Demo: Dashboard with MinIO



Demo: Config: Pods

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deployment
  labels:
    app: web.app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web.app
  template:
    metadata:
      labels:
        app: web.app
    spec:
      containers:
        - name: web-container
          image: example.com/kube-image-processing
          ports:
            - containerPort: 5000
          env:
            - name: minio_hostname
              value: minio:9000
            - name: minio_access_key
              value: minio
            - name: minio_secret_key
              value: minio123
      imagePullSecrets:
        - name: kube-image-processing-regcred
```

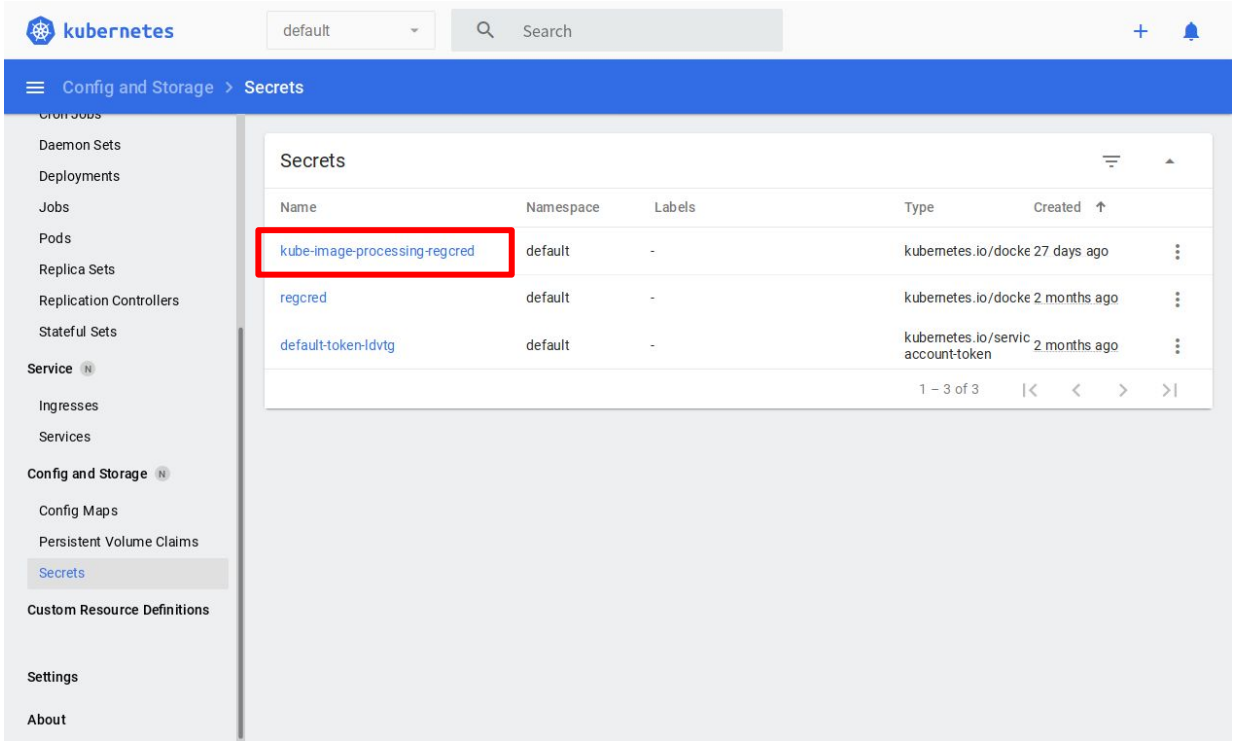
Demo: Config: Service

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web.app
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
```

Demo: Config: Ingress

```yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web-service
                port:
                  number: 80
```

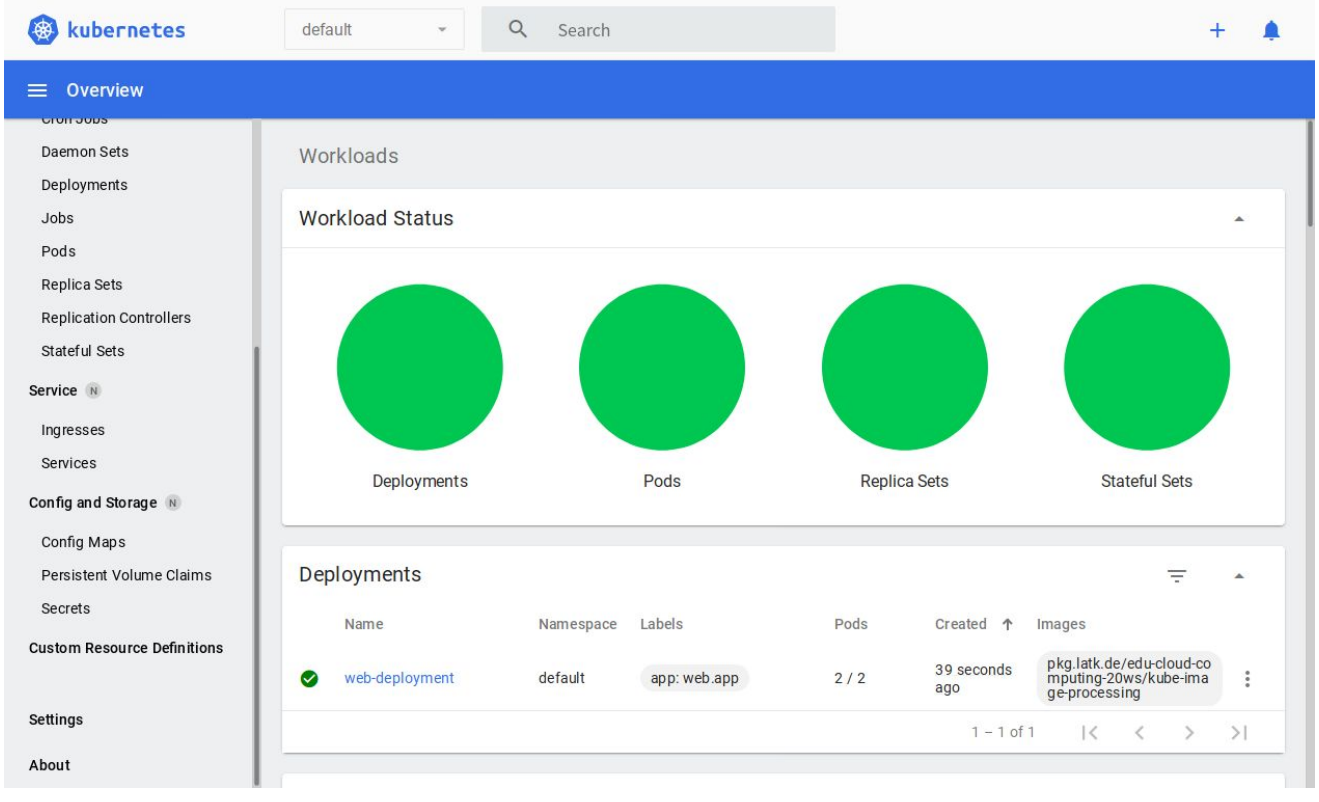
Demo: Secrets



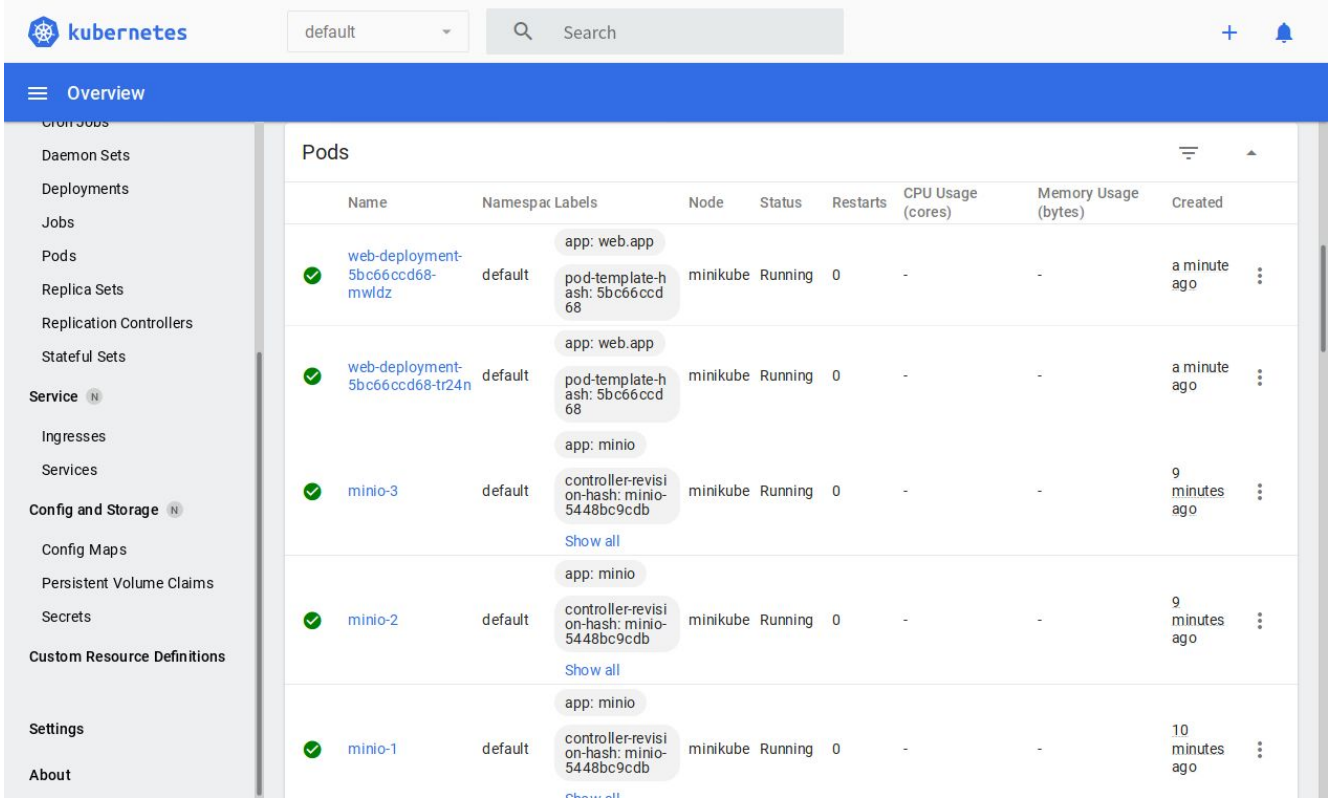
Demo: Apply Configuration

kubectl apply -f imageProcessingDeployment.yaml

Demo: Dashboard with Deployment



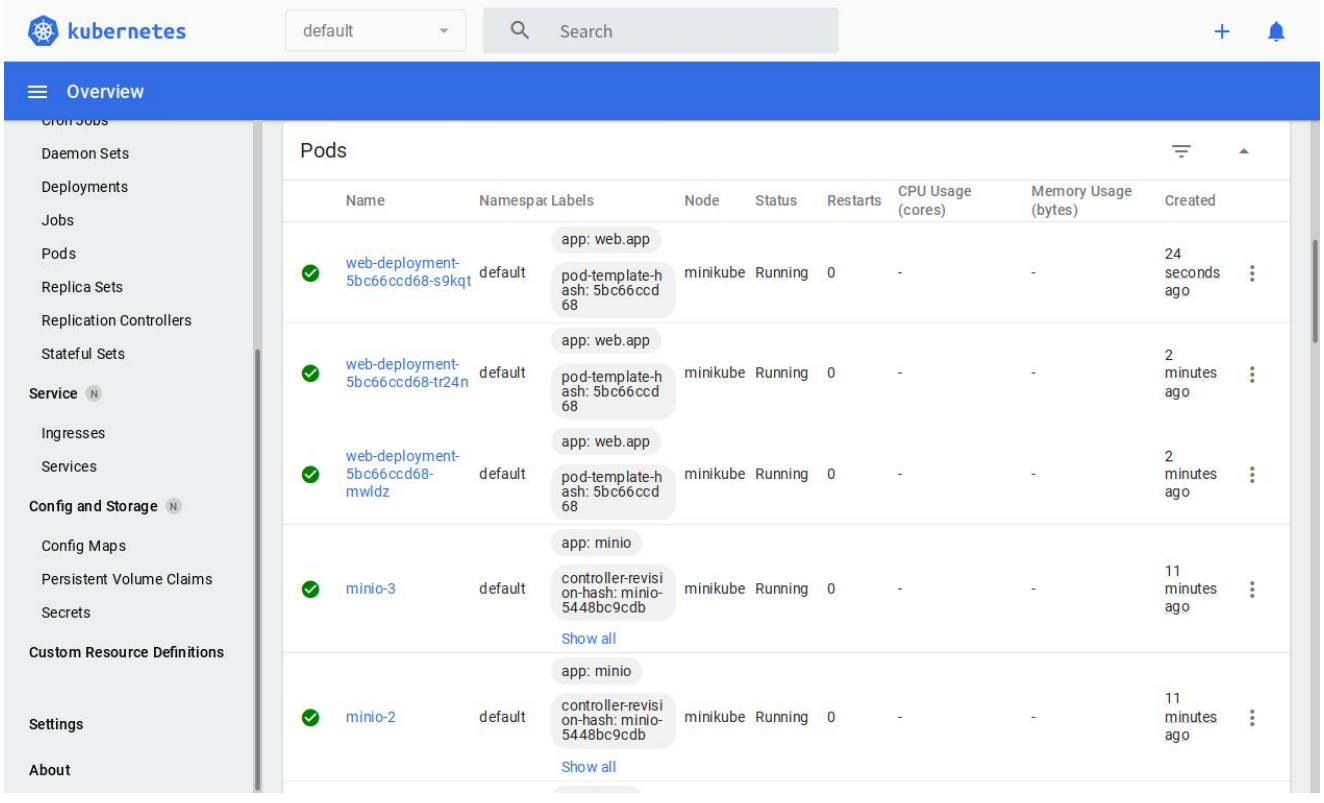
Demo: Dashboard with Deployment



Demo: Scaling

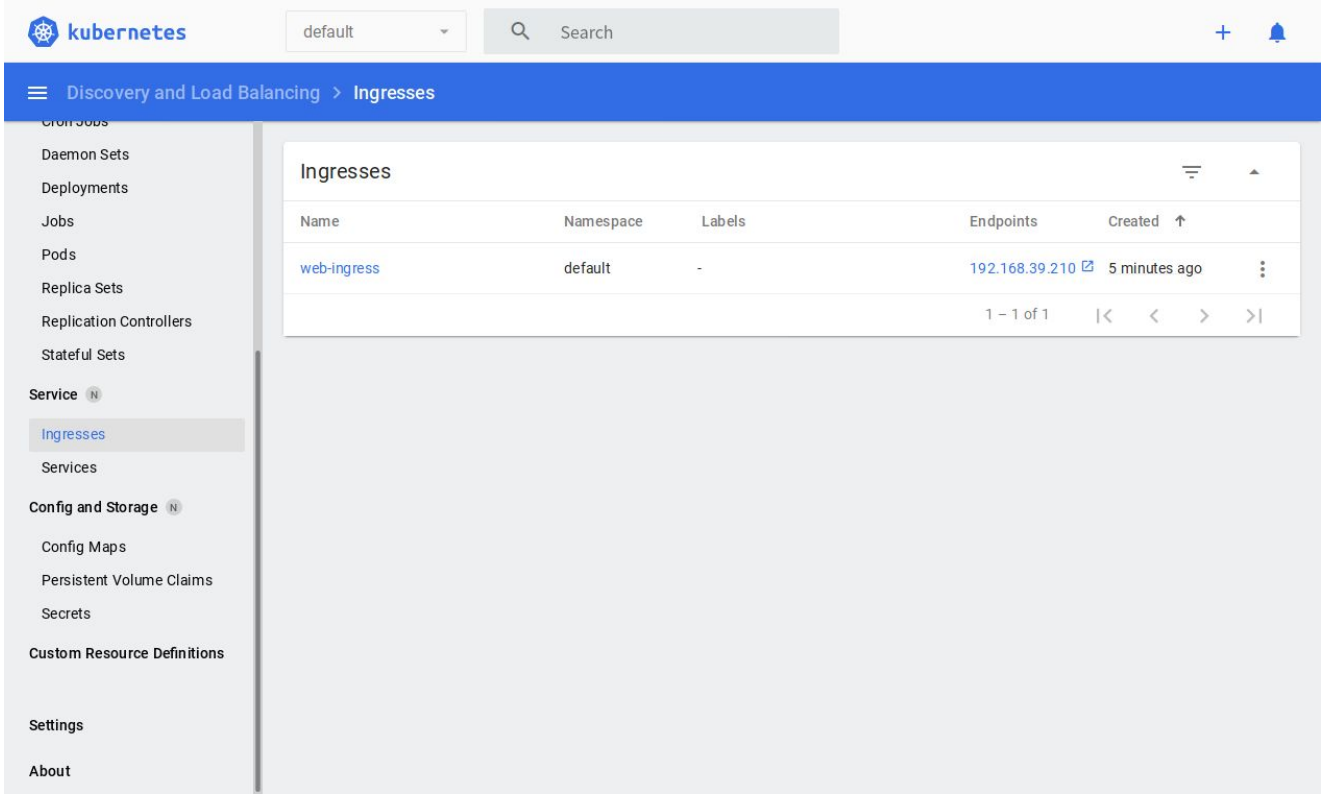
```
kubectl scale --replicas=3 deployment/web-deployment
```

Demo: Dashboard with scaled Deployment



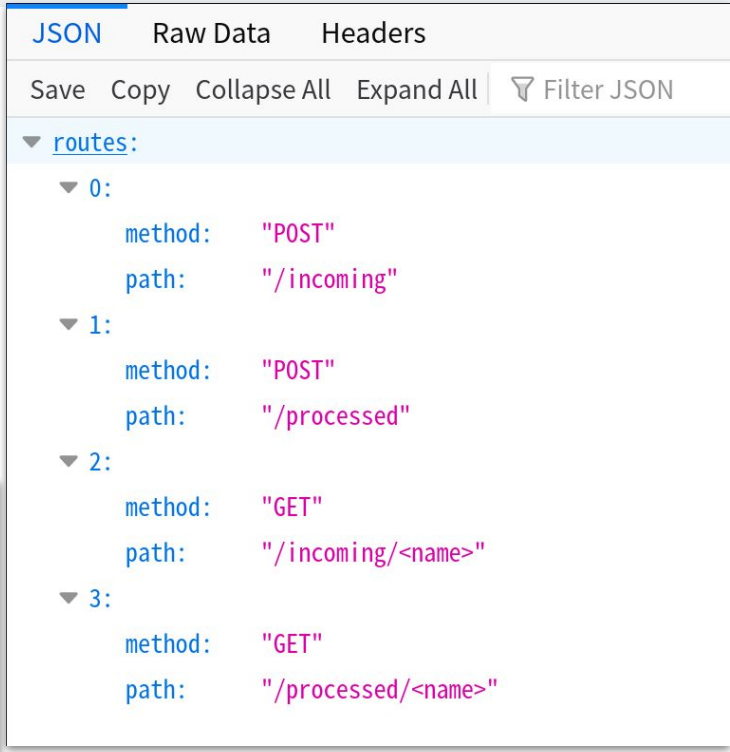
Demo: Dashboard with Ingress

Ingress at IP
192.168.39.210



Demo: REST API

open http://192.168.39.210



Demo: Upload data

```
curl http://192.168.39.210/incoming
  -H 'Content-Type: application/json'
  --data '{"url":"http://192.168.1.103:8123/norway.jpg"}'
```

Response:


```
{"path":"/incoming/56df5bf60b0500d5
e9a6e85193ca4b52b21024b8dc5ce6f8c48
82bfaacea24b1"}
```

Demo: Convert image

```
curl http://192.168.39.210/processed
  -H 'Content-Type: application/json'
  --data '{"path":"/incoming/56df5bf..."}'
```

Response:

```
{"path":"/processed/011e8ba0a6ace92
8861845837e84578efbf75382d8c1595bb1
aa66e33b621e4f"}
```