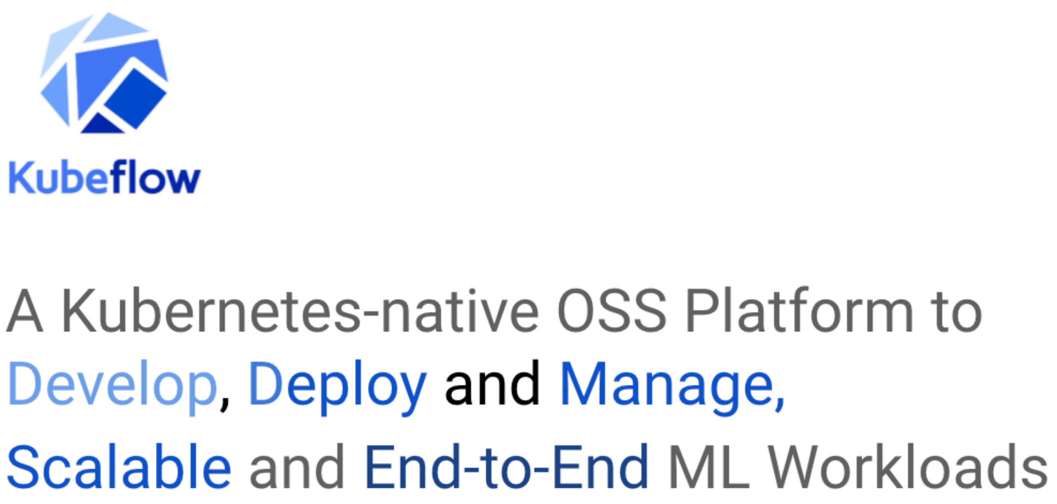
Kubeflow Installation on Kapsule

Disclaimer: This document is my own installation guide to setup Kubeflow on Scaleway Kapsule. An official technical blog post by Scaleway should be published in July. This document is not intended to deploy a production cluster, and is provided as this.

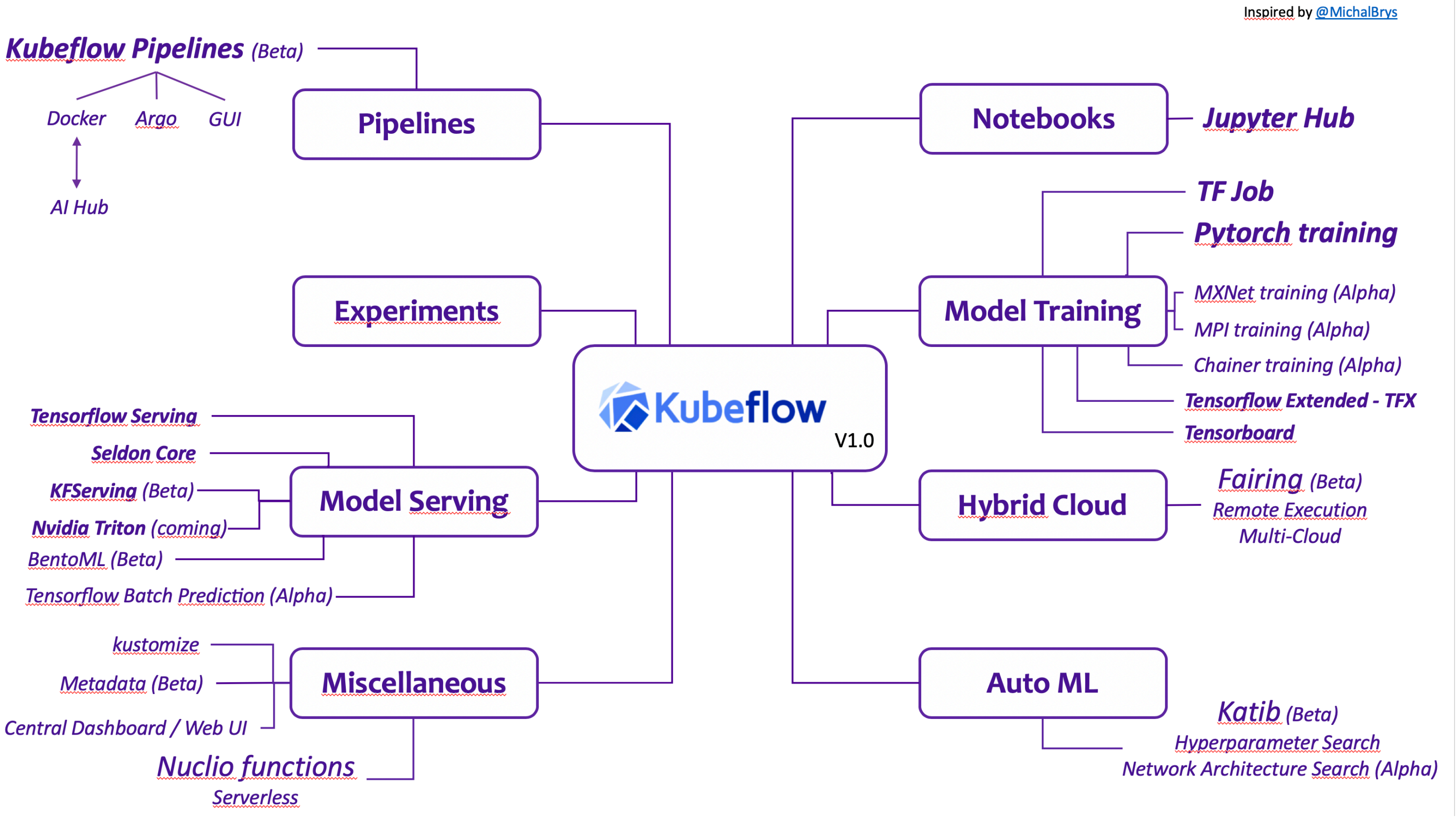
What is Kubeflow ?



Kubeflow

A Kubernetes-native OSS Platform to Develop, Deploy and Manage, Scalable and End-to-End ML Workloads

Kubeflow provides several modules to handle the Machine Learning / Deep Learning workloads:

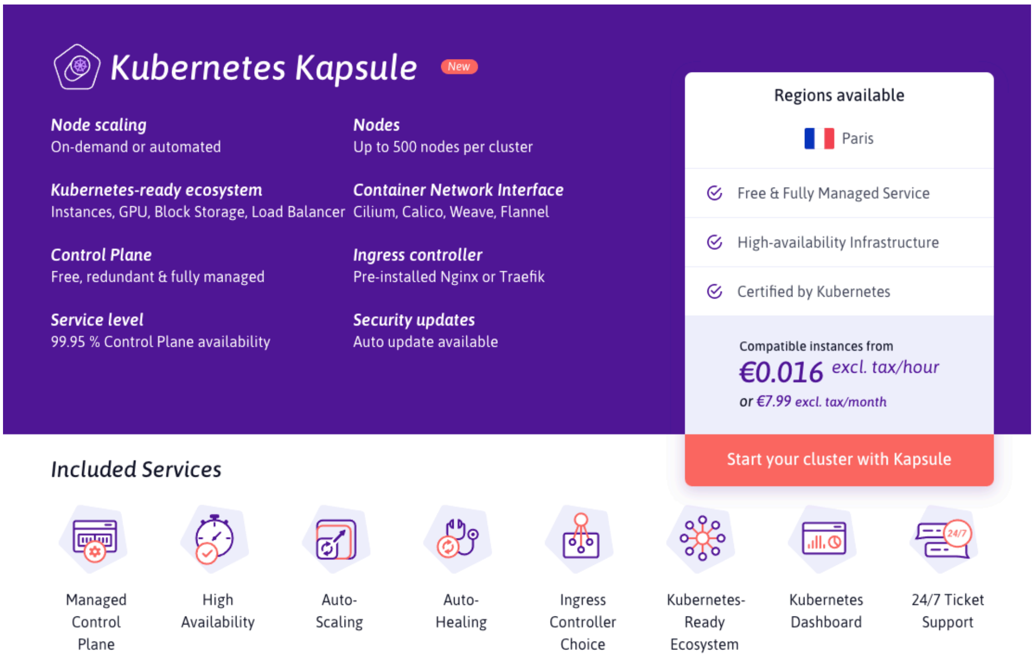


Why you should use Kubeflow ?

- Open Source solution running on Kubernetes
    - Kubernetes Production Ready
    - Multi-Cloud (No vendor locking)
- End-to-End Machine Learning solution, from model creation to production
- Easy, repeatable, portable deployments on a diverse infrastructure
- Makes the work easier and more efficient. Data Scientists can iterate faster
- Data Scientists don't have to care about setting up the environment where the code will be executed
- Hybrid cloud is possible (On-prem + Cloud/Autoscaling) = Better ROI & Time-to-Market
- Use the GPUs ressources you need, when you need them.
- “Pay-as-you-go”
- Balance iteration speed / costs per project

What is Kapsule ?

Scaleway Kubernetes Kapsule is a service that allows you to run containerized applications in a managed Kubernetes environment.



Kubernetes Kapsule New

Node scaling
On-demand or automated

Nodes
Up to 500 nodes per cluster

Kubernetes-ready ecosystem
Instances, GPU, Block Storage, Load Balancer

Container Network Interface
Cilium, Calico, Weave, Flannel

Control Plane
Free, redundant & fully managed

Ingress controller
Pre-installed Nginx or Traefik

Service level
99.95 % Control Plane availability

Security updates
Auto update available

Regions available

Paris

✓ Free & Fully Managed Service

✓ High-availability Infrastructure

✓ Certified by Kubernetes

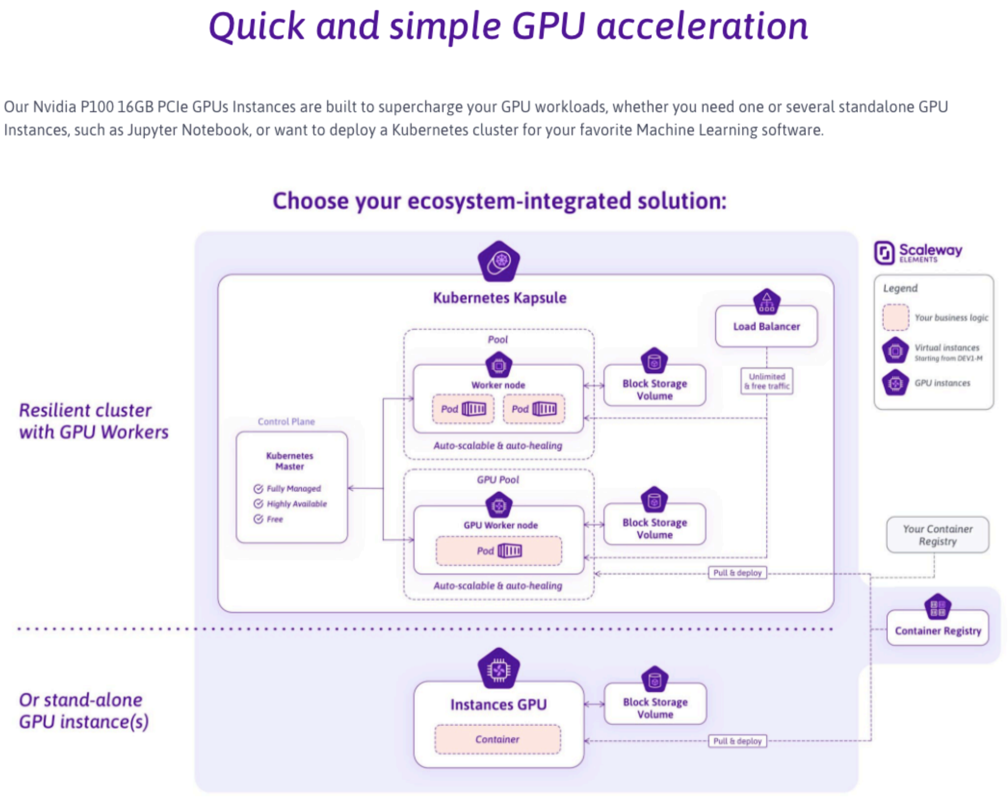
Compatible instances from
€0.016 excl. tax/hour
or €7.99 excl. tax/month

Start your cluster with Kapsule

Included Services

- Managed Control Plane
- High Availability
- Auto-Scaling
- Auto-Healing
- Ingress Controller Choice
- Kubernetes-Ready Ecosystem
- Kubernetes Dashboard
- 24/7 Ticket Support

You can add one or more GPU Instance workers (with auto scaling support) in a Kapsule GPU node Pool:



Quick and simple GPU acceleration

Our Nvidia P100 16GB PCIe GPUs Instances are built to supercharge your GPU workloads, whether you need one or several standalone GPU Instances, such as Jupyter Notebook, or want to deploy a Kubernetes cluster for your favorite Machine Learning software.

Choose your ecosystem-integrated solution:

Kapsule also provides a dynamic volume provisioner on Scaleway Block Storage.

Documentation & useful links

- Kubeflow home page : https://kubeflow.org
- Kubeflow examples : https://github.com/kubeflow/examples
- Scaleway Kapsule Product page : https://www.scaleway.com/en/kubernetes-kapsule/
- Kapsule documentation : https://www.scaleway.com/en/docs/get-started-with-scaleway-kubernetes-kapsule/
- Kubernetes home page : https://kubernetes.io/
- Scaleway GPU Instances Product page: https://www.scaleway.com/en/gpu-instances/
- Scaleway Container Registry Product Page : https://www.scaleway.com/en/container-registry/
- Scaleway Container Registry documentation : https://www.scaleway.com/en/docs/scaleway-container-registry/
- Scaleway Object Storage Product page : https://www.scaleway.com/en/object-storage/
- Scaleway Object Storage documentation : https://www.scaleway.com/en/docs/object-storage-feature/
- Scaleway Block Storage Product page : https://www.scaleway.com/en/block-storage/
- Scaleway Block Storage documentation : https://www.scaleway.com/en/docs/block-storage-overview/

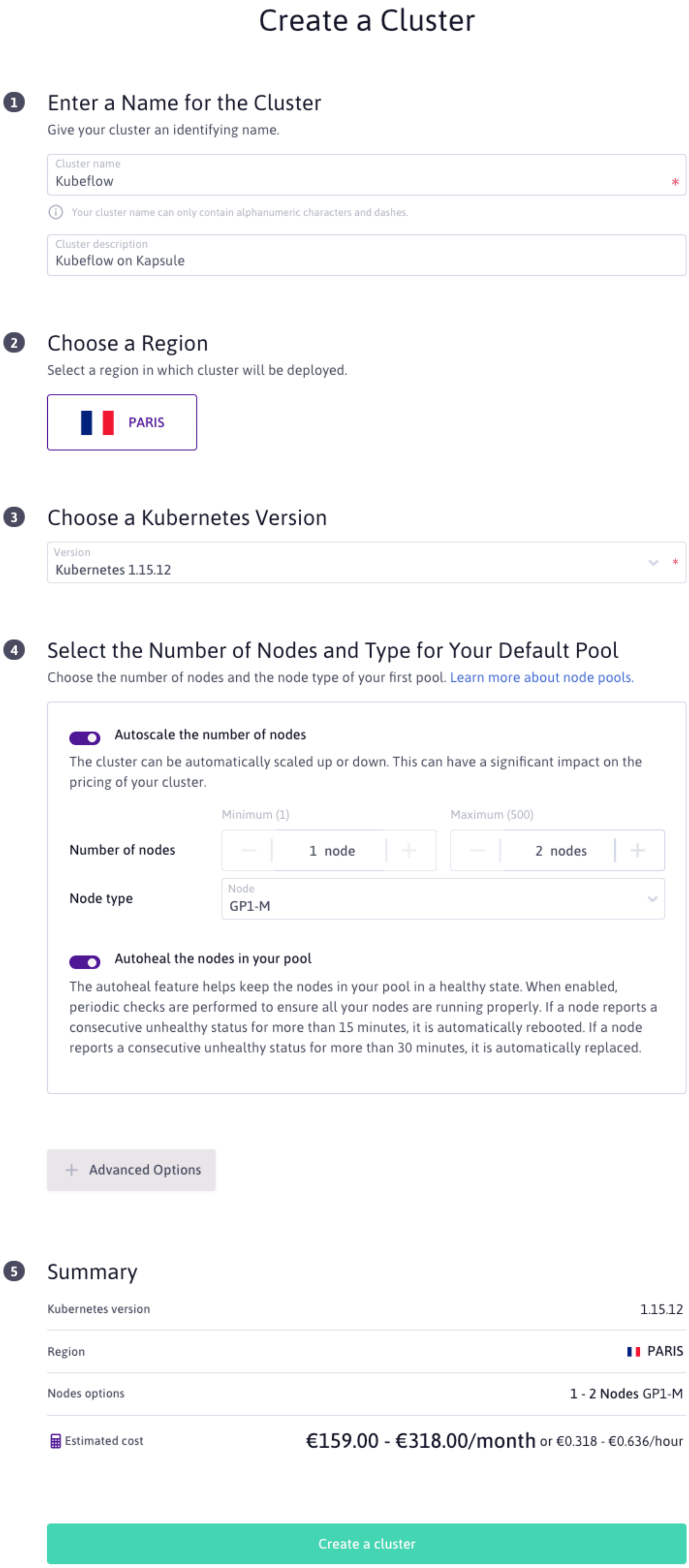
Prerequisites

- If needed, create a Scaleway account
  - https://console.scaleway.com/register
- If you haven't done it before, set an existing public SSH Key to your account to connect to your instance.
  - https://www.scaleway.com/en/docs/configure-new-ssh-key/

Create a Kapsule Cluster

If needed, have a look at Kapsule documentation : https://www.scaleway.com/en/docs/get-started-with-scaleway-kubernetes-kapsule/

- Verify the current compatible versions of Kubernetes and Kubeflow
  - https://www.kubeflow.org/docs/started/k8s/overview/
    - For example Kubeflow 1.0 on Kubernetes 1.15
- Open the Kapsule Dashboard in the Scaleway Console
  - https://console.scaleway.com/kapsule/clusters
- Click on the Create Cluster button
- Enter a Name for the Cluster
  - For example Kubeflow
  - Choose a Kubernetes Version
    - We choose version 1.15 which is compatible with Kubeflow
  - Select the Number of Nodes and Type for Your Default Pool
    - Activate “Autoscale the number of nodes”
    - Enter the number of nodes values: We choose Min 1 and Max 2
    - Select the Node Type: We choose GP1-M instances (16 cores 64GB RAM)
      - Note: If you choose a too small configuration for your base cluster, and if you later add GPU nodes pool with autoscaling, the automatic scaling down of the GPU nodes might be prevented by Kubeflow system pods being deployed on the GPU nodes (due to the high cpu/memory pressure on the CPU nodes)
- In the Advanced settings, keep the default values
  - Cilium for Interface Network Provider
  - **No Ingress controller deployment**



Create a Cluster

1 Enter a Name for the Cluster

Give your cluster an identifying name.

Cluster name: Kubeflow *

Your cluster name can only contain alphanumeric characters and dashes.

Cluster description: Kubeflow on Kapsule

2 Choose a Region

Select a region in which cluster will be deployed.

PARIS

3 Choose a Kubernetes Version

Version: Kubernetes 1.15.12 *

4 Select the Number of Nodes and Type for Your Default Pool

Choose the number of nodes and the node type of your first pool. Learn more about node pools.

Autoscale the number of nodes

The cluster can be automatically scaled up or down. This can have a significant impact on the pricing of your cluster.

	Minimum (1)	Maximum (500)
Number of nodes	− 1 node +	− 2 nodes +
Node type	Node: GP1-M	

Autoheal the nodes in your pool

The autoheal feature helps keep the nodes in your pool in a healthy state. When enabled, periodic checks are performed to ensure all your nodes are running properly. If a node reports a consecutive unhealthy status for more than 15 minutes, it is automatically rebooted. If a node reports a consecutive unhealthy status for more than 30 minutes, it is automatically replaced.

5 Summary

Kubernetes version	1.15.12
Region	PARIS
Nodes options	1 - 2 Nodes GP1-M
Estimated cost	€159.00 - €318.00/month or €0.318 - €0.636/hour

Install and Setup kubectl on your local computer

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

Follows the instruction on the Kubernetes website to install kubectl for your system

https://kubernetes.io/docs/tasks/tools/install-kubectl/

1) Install Kubectl (Mac OS example)

```
% curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://
storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/darwin/amd64/kubectl"
```

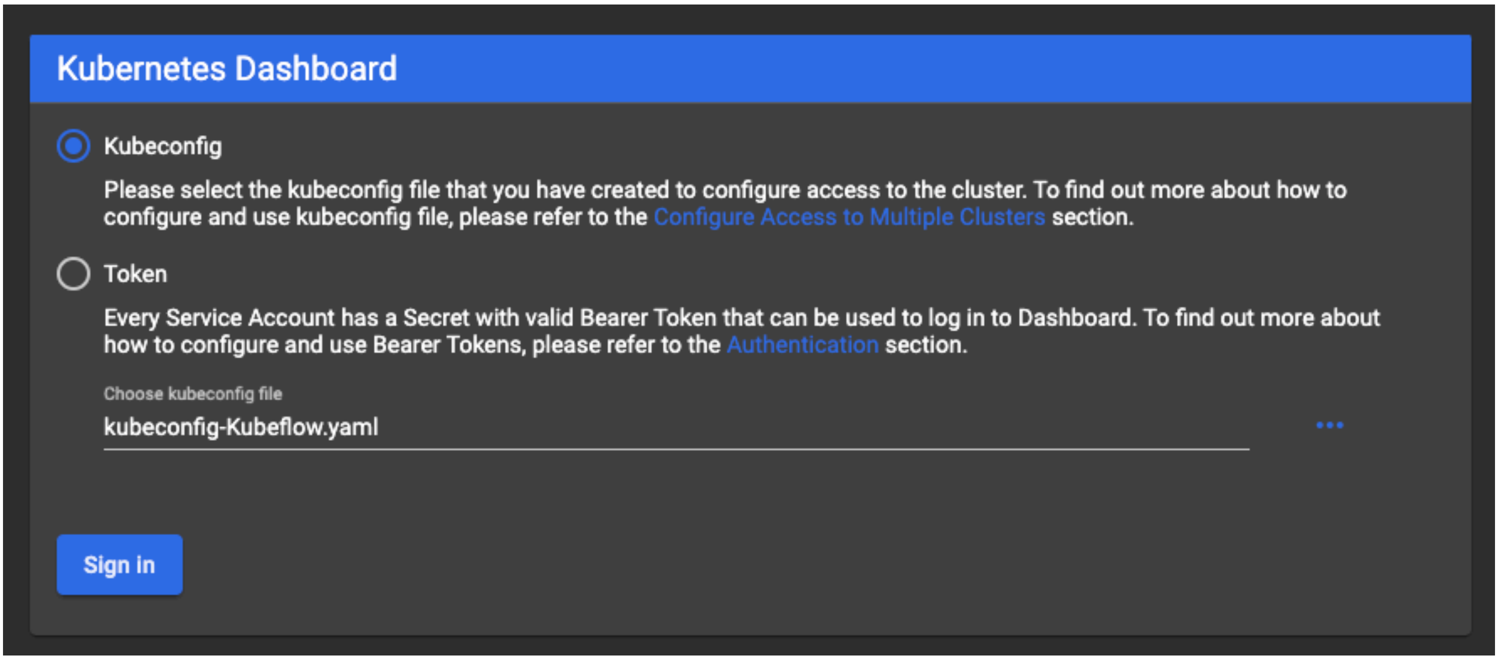
2) Download your Kapsule's kubeconfig in `~/.kube` , as described in https://www.scaleway.com/en/docs/get-started-with-scaleway-kubernetes-kapsule/#-Connecting-to-a-Kubernetes-Cluster-via-kubectl

3) Accessing the Kubernetes Dashboard using the kubectl (Command line is for Kubernetes < 1.16)

- Launch a kubectl proxy

```
% kubectl proxy
```

- Open a browser and paste the URL http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/login and select your ~/.kube/kubeconfig file for signing in.



Kubernetes Dashboard

● Kubeconfig

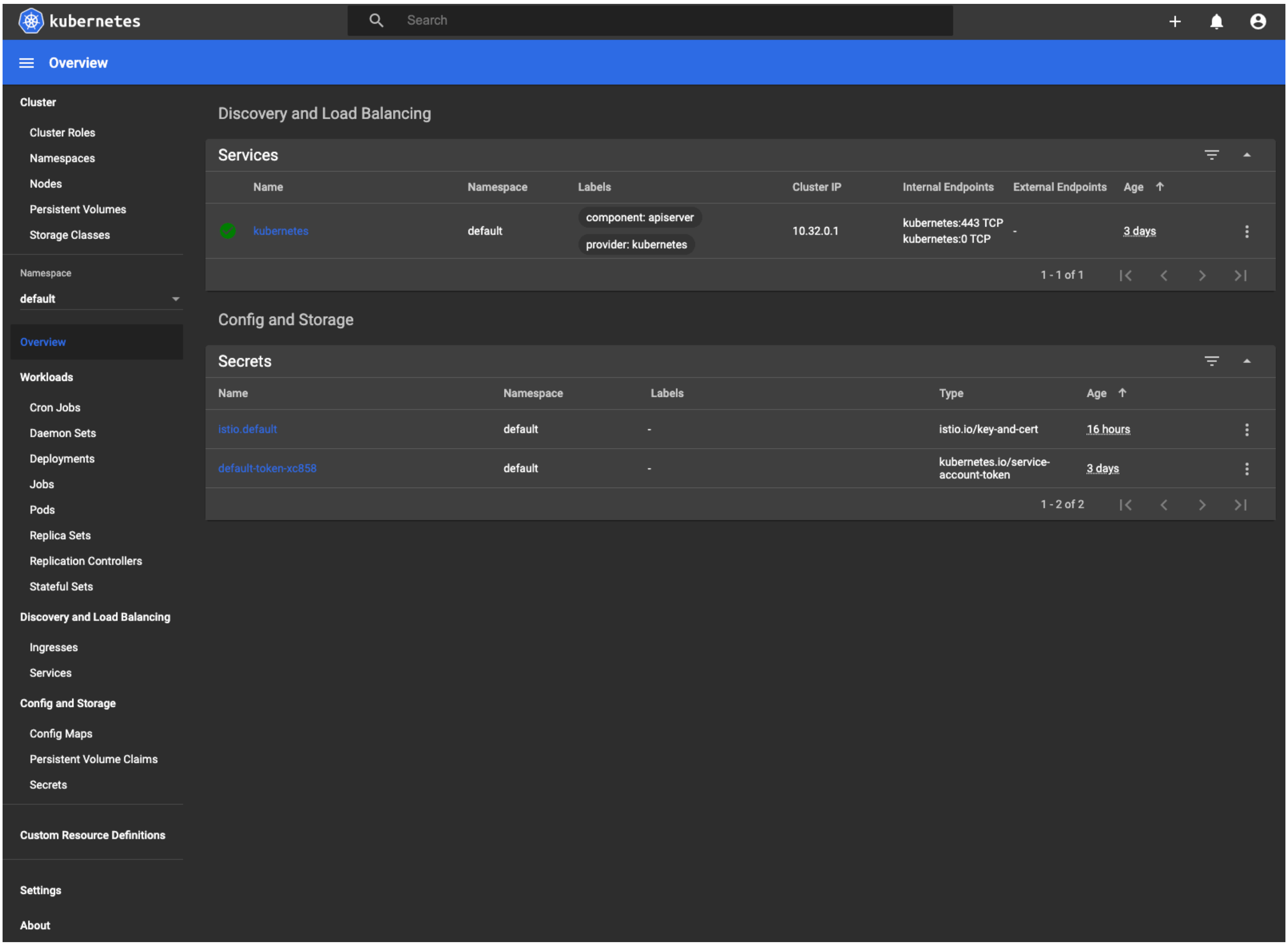
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

○ Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Choose kubeconfig file

kubeconfig-Kubeflow.yaml



Discovery and Load Balancing

Services

	Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Age ↑
✓	kubernetes	default	component: apiserver provider: kubernetes	10.32.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	3 days

1 - 1 of 1

Config and Storage

Secrets

Name	Namespace	Labels	Type	Age ↑
istio.default	default	-	istio.io/key-and-cert	16 hours
default-token-xc858	default	-	kubernetes.io/service-account-token	3 days

1 - 2 of 2

Deploy Kubeflow

Kubeflow Deployment with kfctl_k8s_istio

https://www.kubeflow.org/docs/started/k8s/kfctl-k8s-istio/

- Download the latest kfctl release from the Kubeflow releases page (v1.0.2 in our example), and install the kfct binary

```
% mkdir -p ~/bin
% export PATH=$PATH:~/bin
% mkdir -p ~/kubeflow
% cd  ~/kubeflow

% curl -L -o ~/bin/kfctl_v1.0.2-0-ga476281_darwin.tar.gz https://github.com/kubeflow/kfctl/
releases/download/v1.0.2/kfctl_v1.0.2-0-ga476281_darwin.tar.gz

% tar xvf ~/bin/kfctl_v1.0.2-0-ga476281_darwin.tar.gz -C ~/bin/
% chmod u+x ~/bin/kfctl
```

- Verify that the kfctl installation is ready

```
% kfctl
A client CLI to create kubeflow applications for specific platforms or 'on-prem'
to an existing k8s cluster.

Usage:
  kfctl [command]

Available Commands:
  alpha       Alpha kfctl features.
  apply       deploys a kubeflow application.
  build       Builds a KF App from a config file
  completion  Generate shell completions
  delete      Delete a kubeflow application.
  generate    'kfctl generate' has been replaced by 'kfctl build'
Please switch to new semantics.
To build a KFAPP run -> kfctl build -f ${CONFIG}
Then to install -> kfctl apply
For more information, run 'kfctl build -h' or read the docs at www.kubeflow.org.
  help        Help about any command
  init        'kfctl init' has been removed.
Please switch to new semantics.
To install run -> kfctl apply -f ${CONFIG}
For more information, run 'kfctl apply -h' or read the docs at www.kubeflow.org.
  version     Print the version of kfctl.

Flags:
  -h, --help   help for kfctl

Use "kfctl [command] --help" for more information about a command.
```

- Create environment variables to make the deployment easier

```
# Set KF_NAME to the name of your Kubeflow deployment. You also use this
# value as directory name when creating your configuration directory.
# For example, your deployment name can be 'my-kubeflow' or 'kf-test'.
% export KF_NAME=kubeflow

# Set the path to the base directory where you want to store one or more
# Kubeflow deployments. For example, /opt/.
# Then set the Kubeflow application directory for this deployment.
% export BASE_DIR=${HOME}/kubeflow-cluster
% export KF_DIR=${BASE_DIR}/${KF_NAME}

# Set the configuration file to use when deploying Kubeflow.
# The following configuration installs Istio by default.
% export CONFIG_URI="https://raw.githubusercontent.com/kubeflow/manifests/v1.0-branch/kfdef/
kfctl_k8s_istio.v1.0.2.yaml"
```

- Deploy Kubeflow

```
% mkdir -p ${KF_DIR}
% cd ${KF_DIR}
% kfctl apply -V -f ${CONFIG_URI}
```

- You can monitor the Kubeflow deployment in the Kubernetes dashboard (in the Kubeflow namespace) or via the following command

```
% kubectl get pods -n kubeflow
```

- Other useful Kubernetes commands

```
# Pods cleaning: Delete completed succeeded pods
% kubectl delete pod --field-selector=status.phase==Succeeded
```

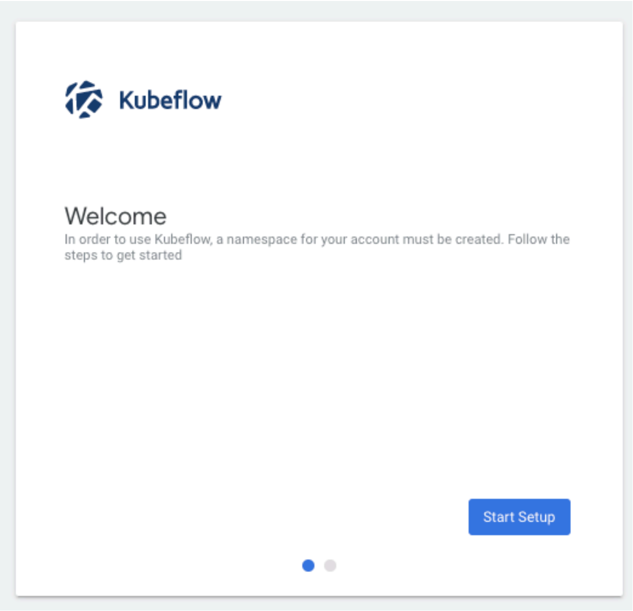
Access to Kubeflow Dashboard

https://www.kubeflow.org/docs/components/central-dash/overview/

- Use the following command to set up port forwarding to the Istio gateway.

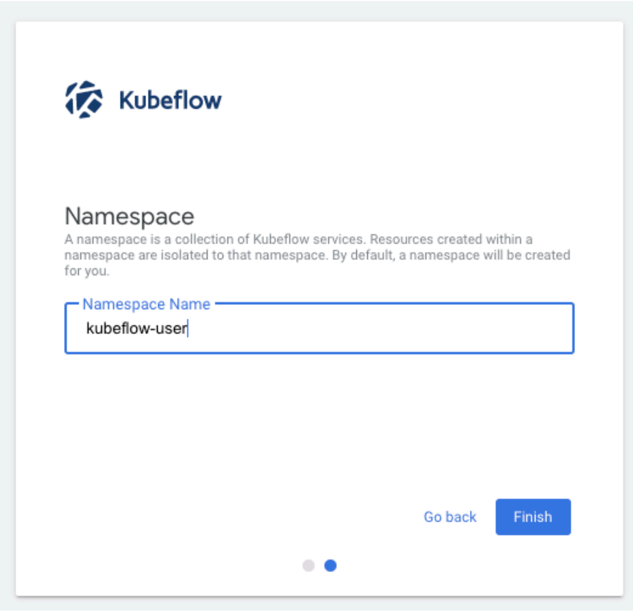
```
% kubectl port-forward -n istio-system svc/istio-ingressgateway 8080:80
```

- Open the following URL to access the Kubeflow central Dashboard: http://localhost:8080/
- In order to use Kubeflow, a namespace for your account must be created the first time you access the Dashboard. Namespace. A namespace is a collection of Kubeflow services. Resources created within a namespace are isolated to that namespace. By default, a namespace will be created for you.



Welcome

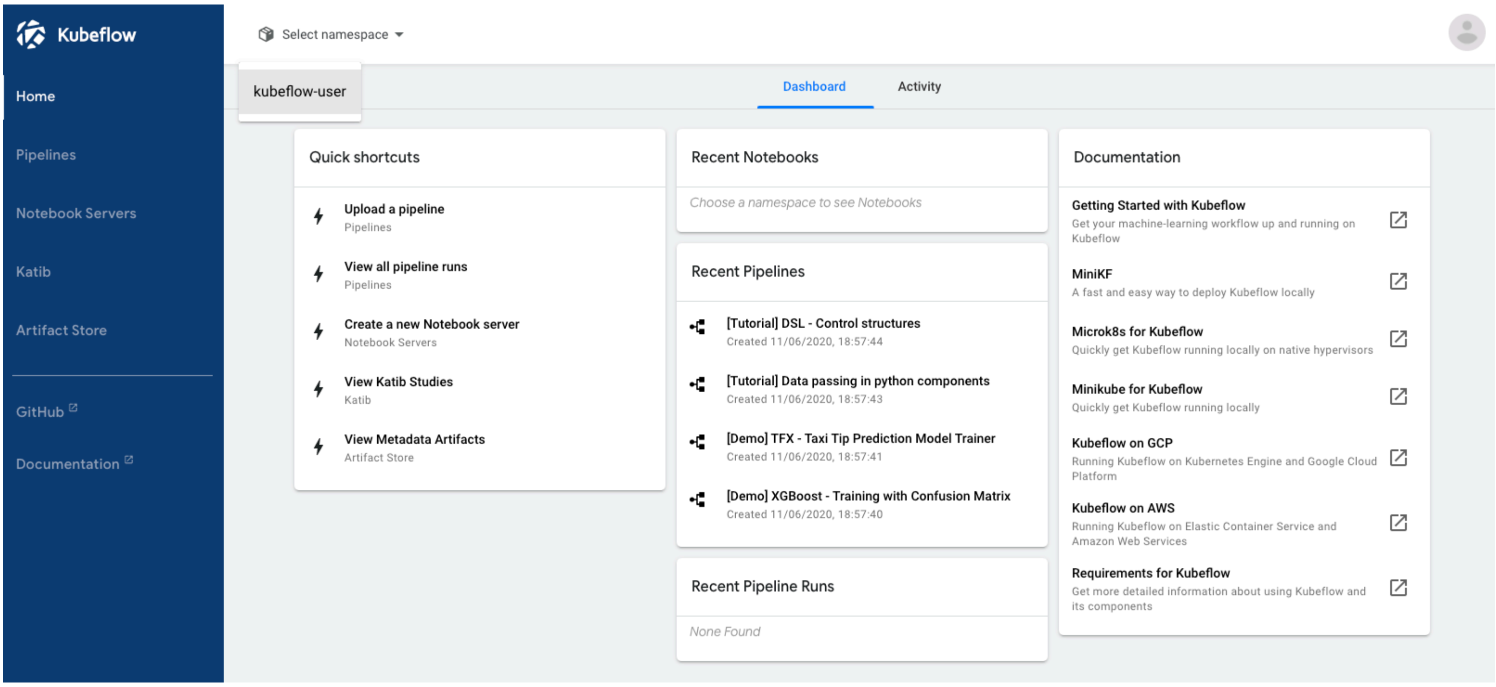
In order to use Kubeflow, a namespace for your account must be created. Follow the steps to get started



Namespace

A namespace is a collection of Kubeflow services. Resources created within a namespace are isolated to that namespace. By default, a namespace will be created for you.

Namespace Name: kubeflow-user



Quick shortcuts

- Upload a pipeline — Pipelines
- View all pipeline runs — Pipelines
- Create a new Notebook server — Notebook Servers
- View Katib Studies — Katib
- View Metadata Artifacts — Artifact Store

Recent Notebooks

Choose a namespace to see Notebooks

Recent Pipelines

- [Tutorial] DSL - Control structures — Created 11/06/2020, 18:57:44
- [Tutorial] Data passing in python components — Created 11/06/2020, 18:57:43
- [Demo] TFX - Taxi Tip Prediction Model Trainer — Created 11/06/2020, 18:57:41
- [Demo] XGBoost - Training with Confusion Matrix — Created 11/06/2020, 18:57:40

Recent Pipeline Runs

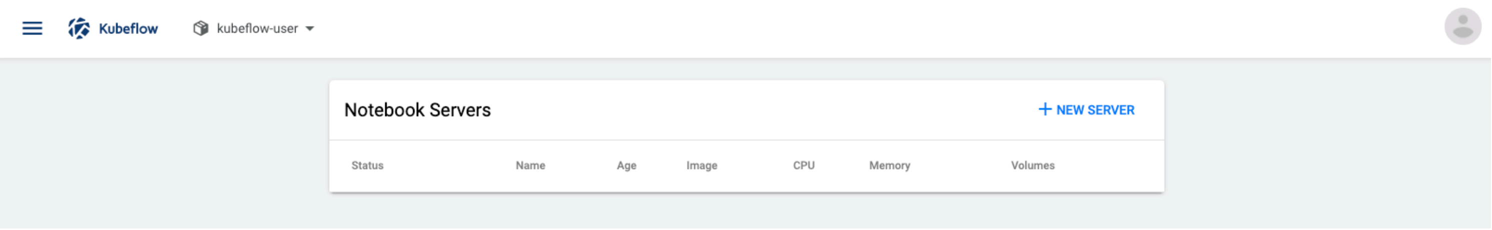
None Found

Documentation

- Getting Started with Kubeflow — Get your machine-learning workflow up and running on Kubeflow
- MiniKF — A fast and easy way to deploy Kubeflow locally
- Microk8s for Kubeflow — Quickly get Kubeflow running locally on native hypervisors
- Minikube for Kubeflow — Quickly get Kubeflow running locally
- Kubeflow on GCP — Running Kubeflow on Kubernetes Engine and Google Cloud Platform
- Kubeflow on AWS — Running Kubeflow on Elastic Container Service and Amazon Web Services
- Requirements for Kubeflow — Get more detailed information about using Kubeflow and its components

Launch a Jupiter Notebook server

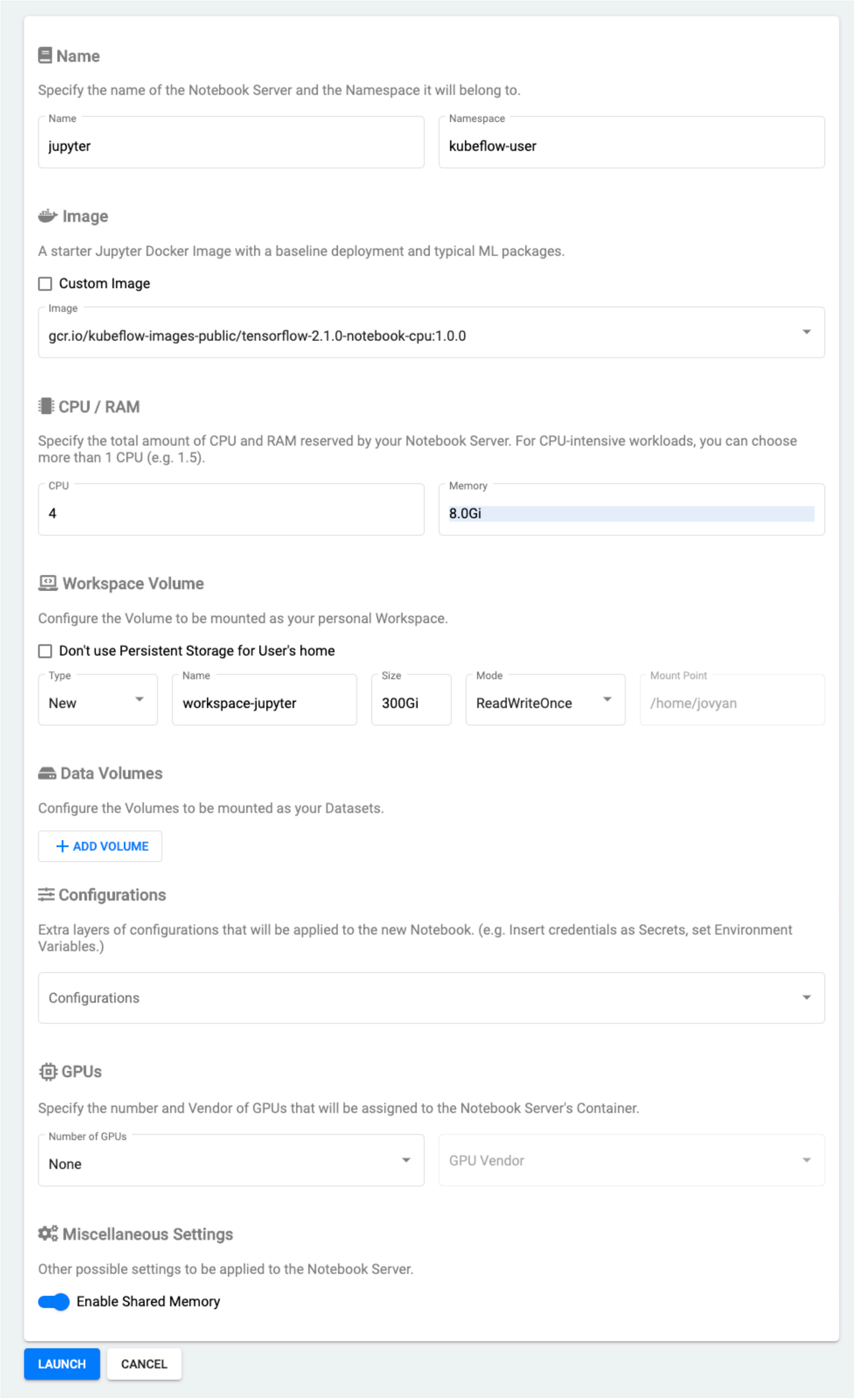
- Click on “Notebook Servers” in the Kubeflow menu



Notebook Servers

Status	Name	Age	Image	CPU	Memory	Volumes

- Click on “+New Server”
  - Select a CPU Jupyter Docker Image with a baseline deployment and typical ML packages Tensorflow, Pytorch)
- Configure CPU, RAM and Data Volume
- Note: With Kubeflow Pipelines, the Jupiter Server does not need to run on a GPU node (but pipeline’s tasks might be executed on GPU nodes) However if you want to add GPU, don’t forget to a GPU node pool to your cluster (See next section)
- Click win the “Launch Button”



Name

Specify the name of the Notebook Server and the Namespace it will belong to.

Name: jupyter
Namespace: kubeflow-user

Image

A starter Jupyter Docker Image with a baseline deployment and typical ML packages.

□ Custom Image

Image: gcr.io/kubeflow-images-public/tensorflow-2.1.0-notebook-cpu:1.0.0

CPU / RAM

Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

CPU: 4
Memory: 8.0Gi

Workspace Volume

Configure the Volume to be mounted as your personal Workspace.

□ Don't use Persistent Storage for User's home

Type	Name	Size	Mode	Mount Point
New	workspace-jupyter	300Gi	ReadWriteOnce	/home/jovyan

Data Volumes

Configure the Volumes to be mounted as your Datasets.

Configurations

Extra layers of configurations that will be applied to the new Notebook. (e.g. Insert credentials as Secrets, set Environment Variables.)

Configurations

GPUs

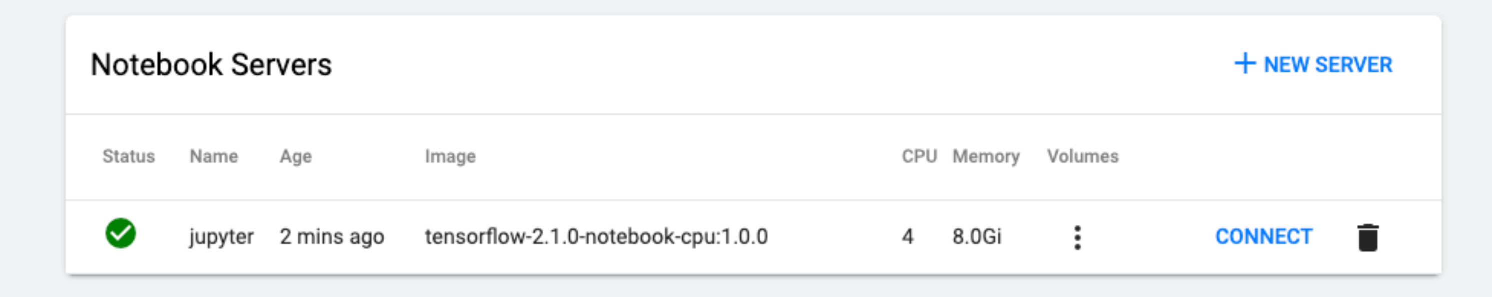
Specify the number and Vendor of GPUs that will be assigned to the Notebook Server's Container.

Number of GPUs: None
GPU Vendor

Miscellaneous Settings

Other possible settings to be applied to the Notebook Server.

● Enable Shared Memory

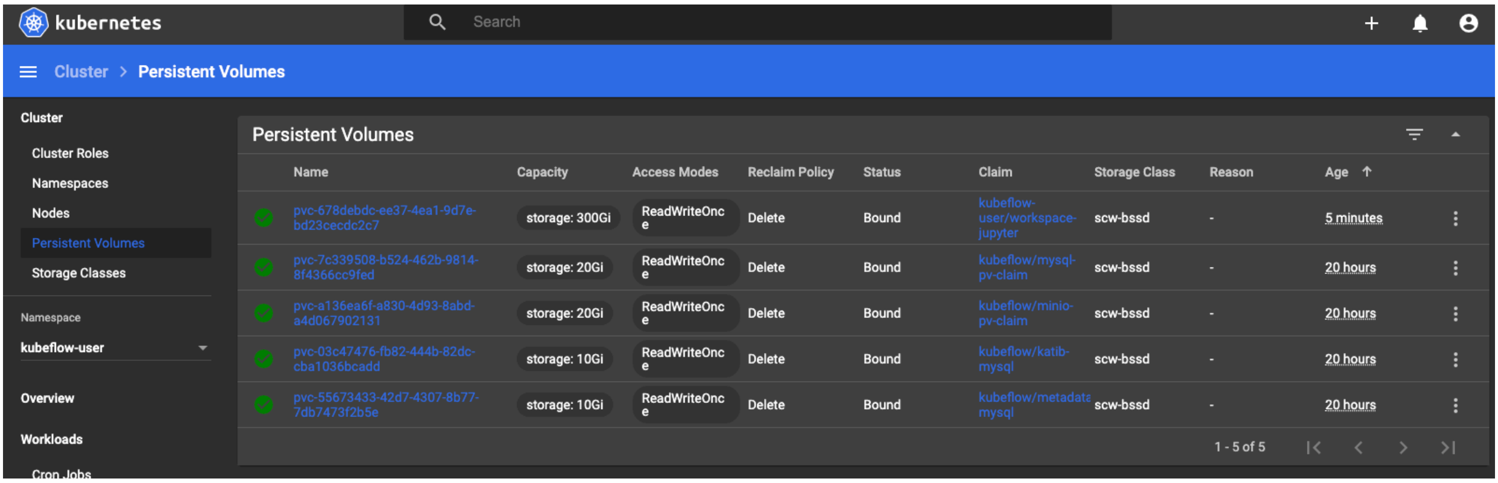


Notebook Servers

Status	Name	Age	Image	CPU	Memory	Volumes		
✓	jupyter	2 mins ago	tensorflow-2.1.0-notebook-cpu:1.0.0	4	8.0Gi	⋮	CONNECT	🗑

- click on connect to open Jupiter Notebook in a separate browser tab

- We can check in Kubernetes Dashboard that a 300Gi Block Storage volumes has been created



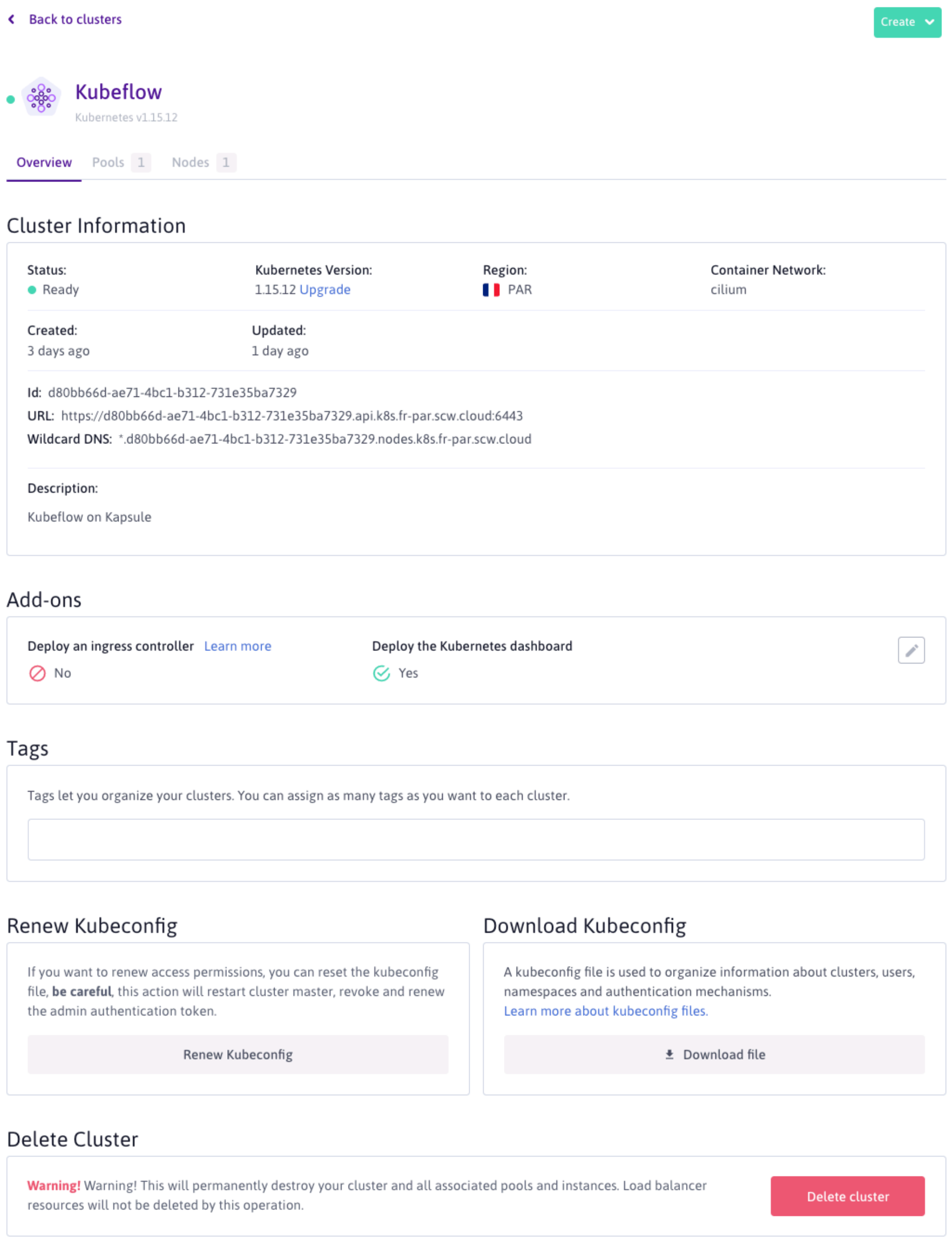
Persistent Volumes

	Name	Capacity	Access Modes	Reclaim Policy	Status	Claim	Storage Class	Reason	Age ↑	
✓	pvc-678debdc-ee37-4ea1-9d7e-bd23cecdc2c7	storage: 300Gi	ReadWriteOnce	Delete	Bound	kubeflow-user/workspace-jupyter	scw-bssd	-	5 minutes	⋮
✓	pvc-7c339508-b524-462b-9814-8f4366cc9fed	storage: 20Gi	ReadWriteOnce	Delete	Bound	kubeflow/mysql-pv-claim	scw-bssd	-	20 hours	⋮
✓	pvc-a136ea6f-a830-4d93-8abd-a4d067902131	storage: 20Gi	ReadWriteOnce	Delete	Bound	kubeflow/minio-pv-claim	scw-bssd	-	20 hours	⋮
✓	pvc-03c47476-fb82-444b-82dc-cba1036bcadd	storage: 10Gi	ReadWriteOnce	Delete	Bound	kubeflow/katib-mysql	scw-bssd	-	20 hours	⋮
✓	pvc-55673433-42d7-4307-8b77-7db7473f2b5e	storage: 10Gi	ReadWriteOnce	Delete	Bound	kubeflow/metadata mysql	scw-bssd	-	20 hours	⋮

1 - 5 of 5

Add a GPU node Pool

- In the Scaleway console, display the details of your Kapsule Kubeflow cluster



Cluster Information

Status: Ready

Kubernetes Version: 1.15.12 Upgrade

Region: PAR

Container Network: cilium

Created: 3 days ago

Updated: 1 day ago

Id: d80bb66d-ae71-4bc1-b312-731e35ba7329

URL: https://d80bb66d-ae71-4bc1-b312-731e35ba7329.api.k8s.fr-par.scw.cloud:6443

Wildcard DNS: *.d80bb66d-ae71-4bc1-b312-731e35ba7329.nodes.k8s.fr-par.scw.cloud

Description:

Kubeflow on Kapsule

Add-ons

Deploy an ingress controller Learn more: No

Deploy the Kubernetes dashboard: Yes

Tags

Tags let you organize your clusters. You can assign as many tags as you want to each cluster.

Renew Kubeconfig

If you want to renew access permissions, you can reset the kubeconfig file, **be careful**, this action will restart cluster master, revoke and renew the admin authentication token.

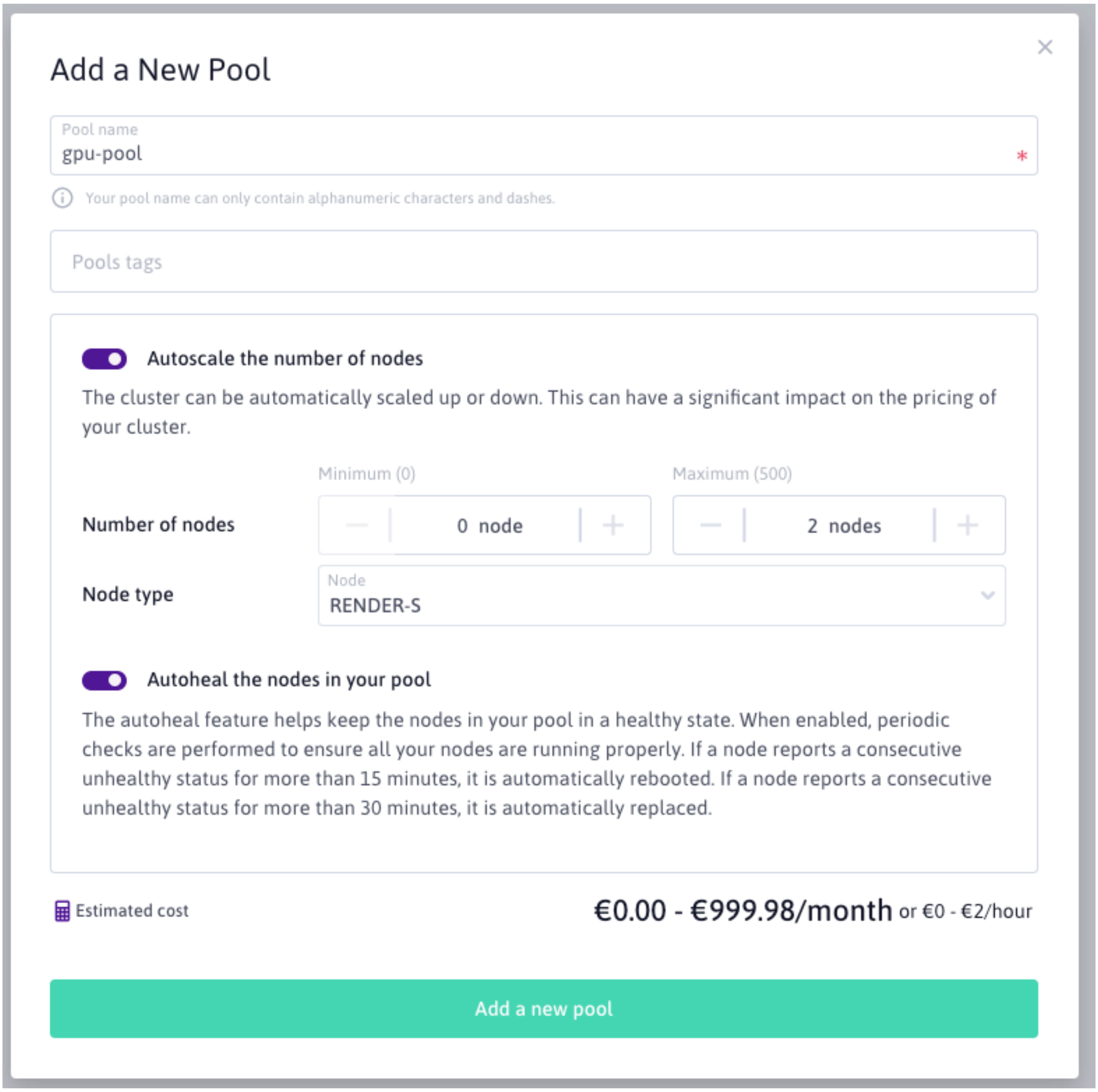
Download Kubeconfig

A kubeconfig file is used to organize information about clusters, users, namespaces and authentication mechanisms.
Learn more about kubeconfig files.

Delete Cluster

Warning! Warning! This will permanently destroy your cluster and all associated pools and instances. Load balancer resources will not be deleted by this operation.

- Click on Pools and then on the “+” button to Add a new pool
  - Enter a pool name
  - Choose a GPU Node type like the RENDER-S, Activate Autoscaling and choose the min and max number of nodes (With autoscaling and min=0, a GPU node will be added only when needed. Note that it takes a few minutes to spawn the instance. When a GPU node is not used for a little time (a few minutes), the node instance is removed. Note that, in that case, you will no more be able to access the pods log of a past Kubeflow pipeline task execution that has been executed on the deleted node)



Add a New Pool

Pool name: gpu-pool *

Your pool name can only contain alphanumeric characters and dashes.

Pools tags

Autoscale the number of nodes

The cluster can be automatically scaled up or down. This can have a significant impact on the pricing of your cluster.

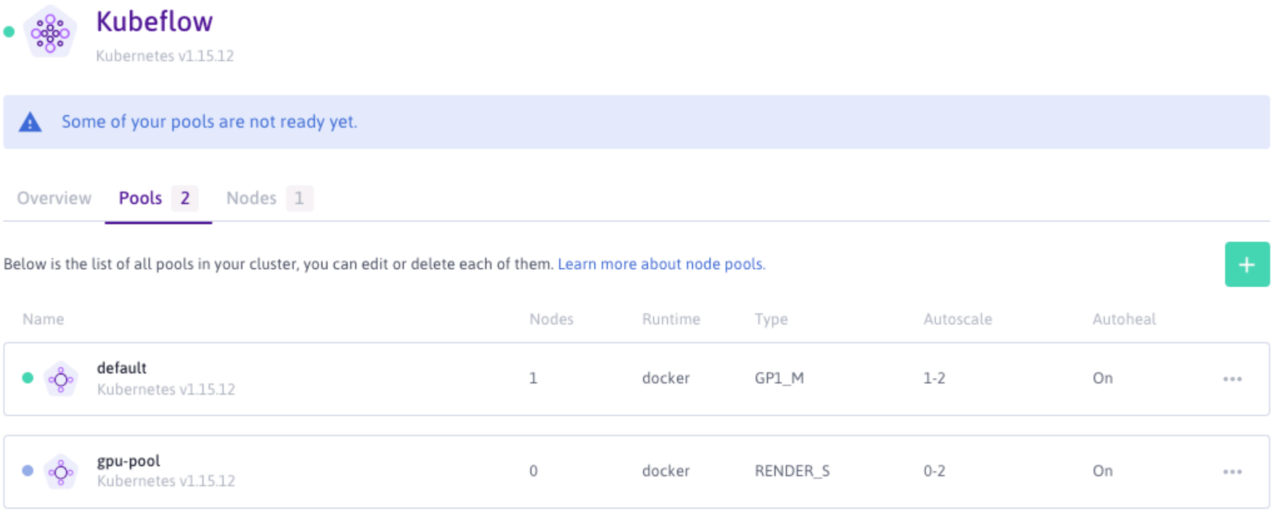
Number of nodes: Minimum (0) 0 node; Maximum (500) 2 nodes

Node type: Node RENDER-S

Autoheal the nodes in your pool

The autoheal feature helps keep the nodes in your pool in a healthy state. When enabled, periodic checks are performed to ensure all your nodes are running properly. If a node reports a consecutive unhealthy status for more than 15 minutes, it is automatically rebooted. If a node reports a consecutive unhealthy status for more than 30 minutes, it is automatically replaced.

Estimated cost €0.00 - €999.98/month or €0 - €2/hour


Kubeflow
Kubernetes v1.15.12

Some of your pools are not ready yet.

Below is the list of all pools in your cluster, you can edit or delete each of them. Learn more about node pools.

Name	Nodes	Runtime	Type	Autoscale	Autoheal
default Kubernetes v1.15.12	1	docker	GP1_M	1-2	On
gpu-pool Kubernetes v1.15.12	0	docker	RENDER_S	0-2	On

Add Block Storage to store datasets & models

Create NFS storage on a Block Storage Volume

The idea here is to add a NFS server that will use a Block Storage volume. The NFS server will be accessible from several pods (whereas a Block Storage volume can only be mounted by one server instance or pod). By doing so, Several Kubeflow pipeline tasks can share a common storage to exchange/reuse some data.

- Setup a nfs-pv PersistentVolumeClaim

```
cat > ./scw_pvc.yaml <<- "EOF"
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pv
spec:
  accessModes: [ "ReadWriteOnce" ]
  resources:
    requests:
      storage: 300Gi
EOF

kubectl create -f scw_pvc.yaml -n kubeflow
```

- Setup a nfs-server Replication Controller

```
cat > ./nfs-server-rc.yaml <<- "EOF"
apiVersion: v1
kind: ReplicationController
metadata:
  name: nfs-server
spec:
  replicas: 1
  selector:
    role: nfs-server
  template:
    metadata:
      labels:
        role: nfs-server
    spec:
      containers:
      - name: nfs-server
        image: k8s.gcr.io/volume-nfs:0.8
        ports:
          - name: nfs
            containerPort: 2049
          - name: mountd
            containerPort: 20048
          - name: rpcbind
            containerPort: 111
        securityContext:
          privileged: true
        volumeMounts:
          - mountPath: /exports
            name: mypvc
      volumes:
        - name: mypvc
          persistentVolumeClaim:
            claimName: nfs-pv
EOF

kubectl create -f nfs-server-rc.yaml -n kubeflow
```

- Setup a nfs-server service

```
cat > ./nfs-server-service.yaml <<- "EOF"
kind: Service
apiVersion: v1
metadata:
  name: nfs-server
spec:
  ports:
    - name: nfs
      port: 2049
    - name: mountd
      port: 20048
    - name: rpcbind
      port: 111
  selector:
    role: nfs-server
EOF

kubectl create -f nfs-server-service.yaml -n kubeflow
```

- Setup a nfs Persistent Volume

```
cat > ./nfs-pv.yaml.tmp <<- "EOF"
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs
spec:
  capacity:
    storage: 300Gi
  accessModes:
    - ReadWriteMany
  nfs:
    # replace the following ip with your NFS IP
    server:  REPLACE_IP
    path: "/"
EOF

export NFS_IP=$(kubectl get svc nfs-server -n kubeflow -o jsonpath='{.spec.clusterIP}')
sed "s/REPLACE_IP/$NFS_IP/" ./nfs-pv.yaml.tmp > ./nfs-pv.yaml

kubectl create -f nfs-pv.yaml -n kubeflow
```

- Setup a nfs Persistent Volume Claim

```
cat > ./nfs-pvc.yaml <<- "EOF"
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 300Gi
EOF

kubectl create -f nfs-pvc.yaml -n kubeflow
```

Now, if you want the Kubeflow Jupyter notebooks to be able to mount and write to this nfs volume

- Open a shell on the NFS server pod

```
NFSPOD=`kubectl -n kubeflow get pods --selector=role=nfs-server| tail -1 | awk '{print $1}'`
kubectl -n kubeflow exec -it $NFSPOD bash
```

- In the docker container/pod, creates a data directory that will be read-writable by the Jupyter notebooks when mounted.

```
cd exports/
mkdir data
chown -R 1000:100 data
exit
```

To delete the NFS server use the following commands (WARNING: This will delete the Data and the Block Storage Volume)

```
kubectl delete -f nfs-pvc.yaml -n kubeflow
kubectl delete -f nfs-pv.yaml -n kubeflow
kubectl delete -f nfs-server-service.yaml -n kubeflow
kubectl delete -f nfs-server-rc.yaml -n kubeflow
kubectl delete -f scw_pvc.yaml -n kubeflow
```

How to access a NFS storage with a shell ?

- Create a `nfs_access.yaml` file

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-access
spec:
  containers:
  - name: bash
    image: bash:latest
    command: ["/bin/sh", "-ec", "while :; do echo '.'; sleep 5 ; done"]
    volumeMounts:
    - mountPath: "/mnt/nfs"
      name: workdir
  volumes:
  - name: workdir
    persistentVolumeClaim:
      claimName: nfs
```

- Create a Pod from this specifications

```
kubectl apply -f nfs_access.yaml -n kubeflow
```

- Connect with a shell to this pods (note: there is no prompt on the command line:

```
kubectl exec -t -i -n kubeflow nfs-access -- /bin/sh
# you can explore the /mnt directory from here
alias ll=‘ls -la’
cd /mnt/nfs/data/
```

- In a similar manner, you can use the kubectl cp command to copy data from/to the PVC

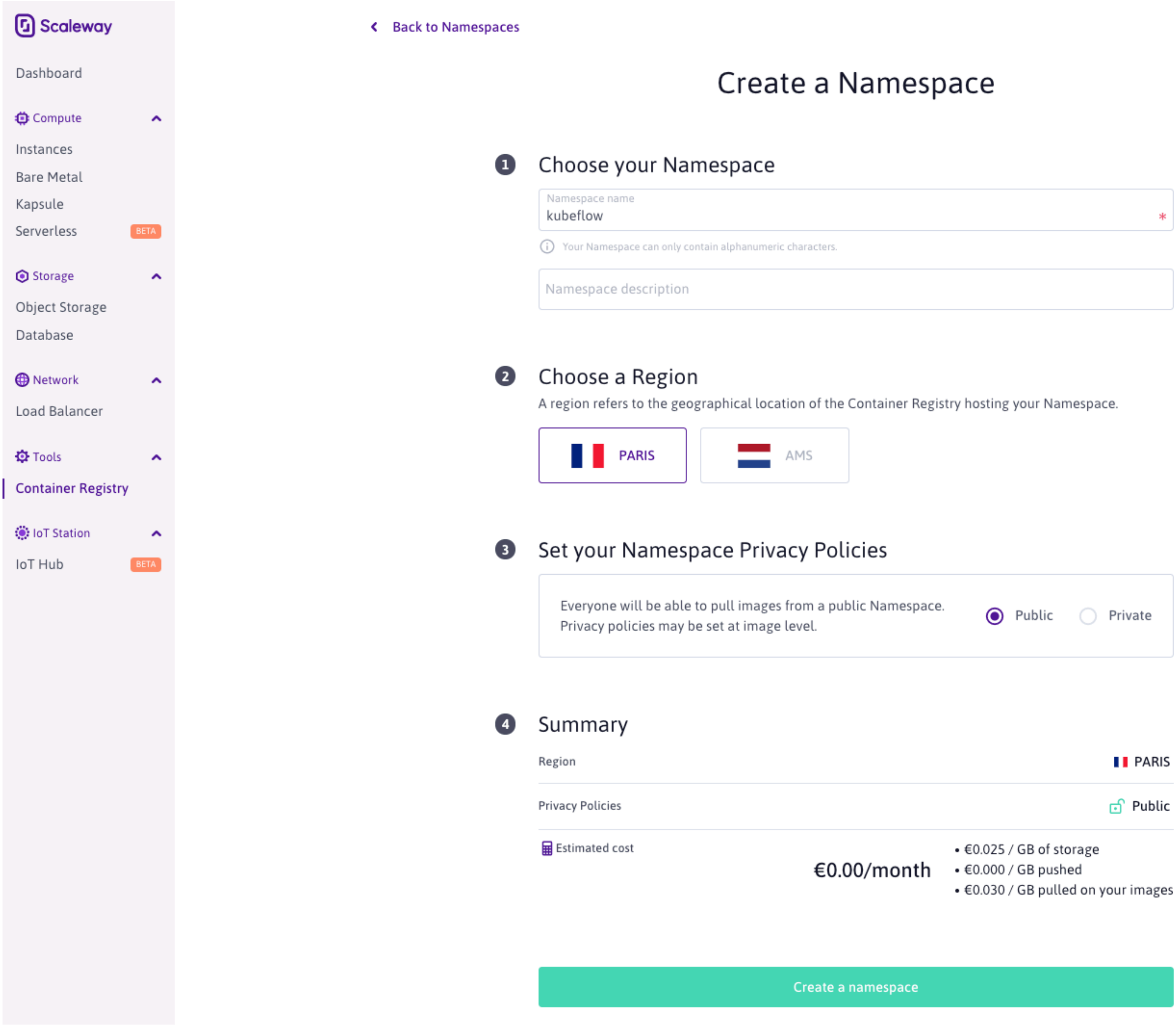
- Delete the data-access pods when you have finished

```
kubectl delete -n kubeflow -f nfs_access.yaml
```

How to push custom Docker images in the Scaleway Docker registry ?

If you want to build and push custom Docker Image into the Scaleway Container Registry for using in Kubeflow, here is the procedure to follow

- From the Scaleway Console's Settings/Credentials, generate a new API token
- From the Scaleway Console's Container Registry, create a namespace



Create a Namespace

1 Choose your Namespace

Namespace name: kubeflow *

Your Namespace can only contain alphanumeric characters.

Namespace description

2 Choose a Region

A region refers to the geographical location of the Container Registry hosting your Namespace.

PARIS | AMS

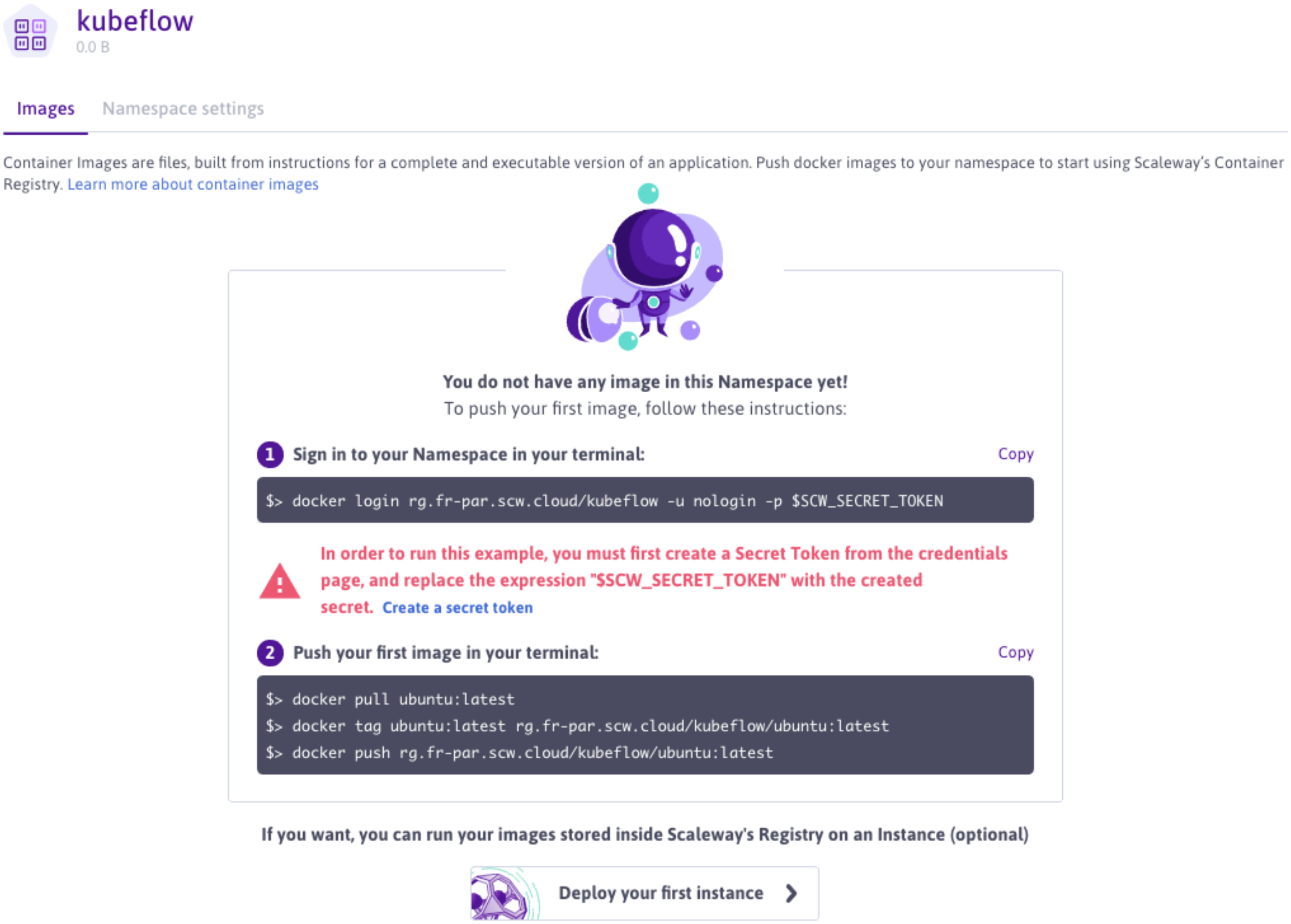
3 Set your Namespace Privacy Policies

Everyone will be able to pull images from a public Namespace. Privacy policies may be set at image level.

● Public ○ Private

4 Summary

Region	PARIS
Privacy Policies	Public
Estimated cost	€0.00/month • €0.025 / GB of storage • €0.000 / GB pushed • €0.030 / GB pulled on your images



kubeflow

0.0 B

Container Images are files, built from instructions for a complete and executable version of an application. Push docker images to your namespace to start using Scaleway's Container Registry. Learn more about container images

You do not have any image in this Namespace yet!

To push your first image, follow these instructions:

1 Sign in to your Namespace in your terminal:

```
$> docker login rg.fr-par.scw.cloud/kubeflow -u nologin -p $SCW_SECRET_TOKEN
```

In order to run this example, you must first create a Secret Token from the credentials page, and replace the expression "$SCW_SECRET_TOKEN" with the created secret. Create a secret token

2 Push your first image in your terminal:

```
$> docker pull ubuntu:latest
$> docker tag ubuntu:latest rg.fr-par.scw.cloud/kubeflow/ubuntu:latest
$> docker push rg.fr-par.scw.cloud/kubeflow/ubuntu:latest
```

If you want, you can run your images stored inside Scaleway's Registry on an Instance (optional)

- Sign in to your Namespace in your terminal:

In order to run this example, you must first create a Secret Token from the credentials page, and replace the expression "$SCW_SECRET_TOKEN" with the created secret.

```
docker login rg.fr-par.scw.cloud/kubeflow -u nologin -p $SCW_SECRET_TOKEN
```

- Push your first image in your terminal:

```
docker tag my_image:latest rg.fr-par.scw.cloud/kubeflow/my_image:latest
docker push rg.fr-par.scw.cloud/kubeflow/my_image:latest
```

- You can then use your custom docker image in Kubeflow

Annex : Kubeflow application matrix

Supported Kubernetes Versions

See: https://www.kubeflow.org/docs/started/k8s/overview/

Status indicators for Kubeflow:

- **Incompatible**: the combination does not work at all
- **Compatible**: all Kubeflow features have been tested and verified for the Kubernetes version
- **No known issues**: the combination has not been fully tested but there are no repoted issues

Kubernetes Versions	Kubeflow 1.0
1.15	**compatible**
1.16	**no known issues**
1.17	**no known issues**
1.18	**no known issues**

Application versioning and stable status

See: https://www.kubeflow.org/docs/reference/version-policy/

Application status indicators for Kubeflow:

- **Stable** means that the application complies with the criteria to reach application version 1.0, and that the Kubeflow community has deemed the application stable for this release of Kubeflow.
- **Beta** means that the application is working towards a version 1.0 release and its maintainers have communicated a timeline for satisfying the criteria for the stable status.
- **Alpha** means that the application is in the early phases of development and/or integration into Kubeflow.

Application	Status in Kubeflow v1.0.2	Application version in Kubeflow v1.0.2
Central dashboard: Kubeflow UI (GitHub)	Stable	1.0.0
Chainer operator (GitHub)	Alpha	

Hyperparameter tuning: Katib (GitHub)	Beta	v1alpha3
KFServing (GitHub)	Beta	v0.2.2
Metadata (GitHub)	Beta	0.2.1
MPI training: MPI operator (GitHub)	Alpha	
MXNet training: MXNet operator (GitHub)	Alpha	
Notebook web app (GitHub)	Stable	1.0.0
Notebook controller (GitHub)	Stable	1.0.0
Pipelines (GitHub)	Beta	0.2.0
Profile Controller for multi-user isolation (GitHub)	Stable	1.0.0
PyTorch training: PyTorch operator (GitHub)	Stable	1.0.0
Seldon Core Serving (GitHub)	Stable	1.0.1
TensorFlow training: TFJob operator (GitHub)	Stable	1.0.0
XGBoost training: XGBoost operator (GitHub)	Alpha	

SDK / CLI	Status with Kubeflow v1.0.2	SDK/CLI version
Fairing (GitHub)	Beta	0.7.1
kfctl (GitHub)	Stable	1.0.0
Kubeflow Pipelines SDK (GitHub)	Beta	0.2.0