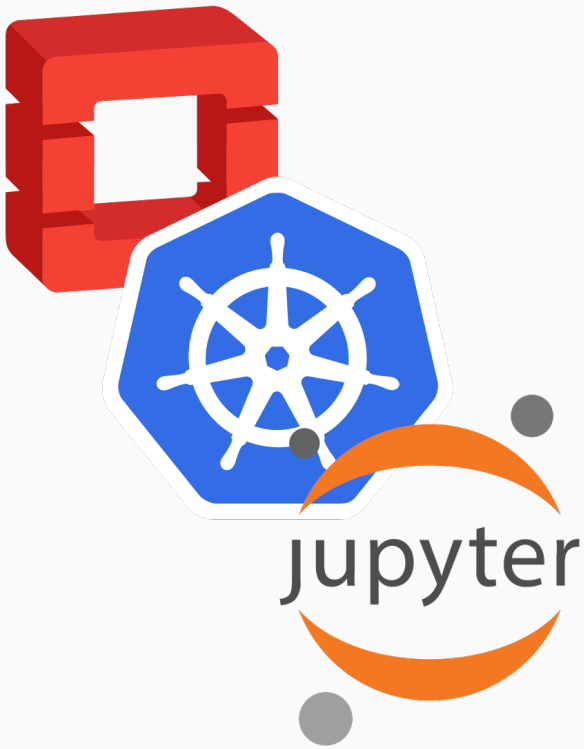
Déploiement de JupyterHub/GPU avec Kubernetes

Méthodes cloud appliquées à la fourniture de ressources de calcul

Rémi Cailletaud

21 janvier 2021



Objectifs:

- Environnement de développement/recherche à la demande: *notebooks*
- Déploiements rapides, reproductibles
- Contrôle des ressources

Architecture:

- OpenStack
- Kubernetes
- JupyterHub

- Fournisseur de ressources : CPU, RAM, disque, GPU ...
- API ouverte
- Infrastructure programmable

Kubernetes: avertissement ⚠

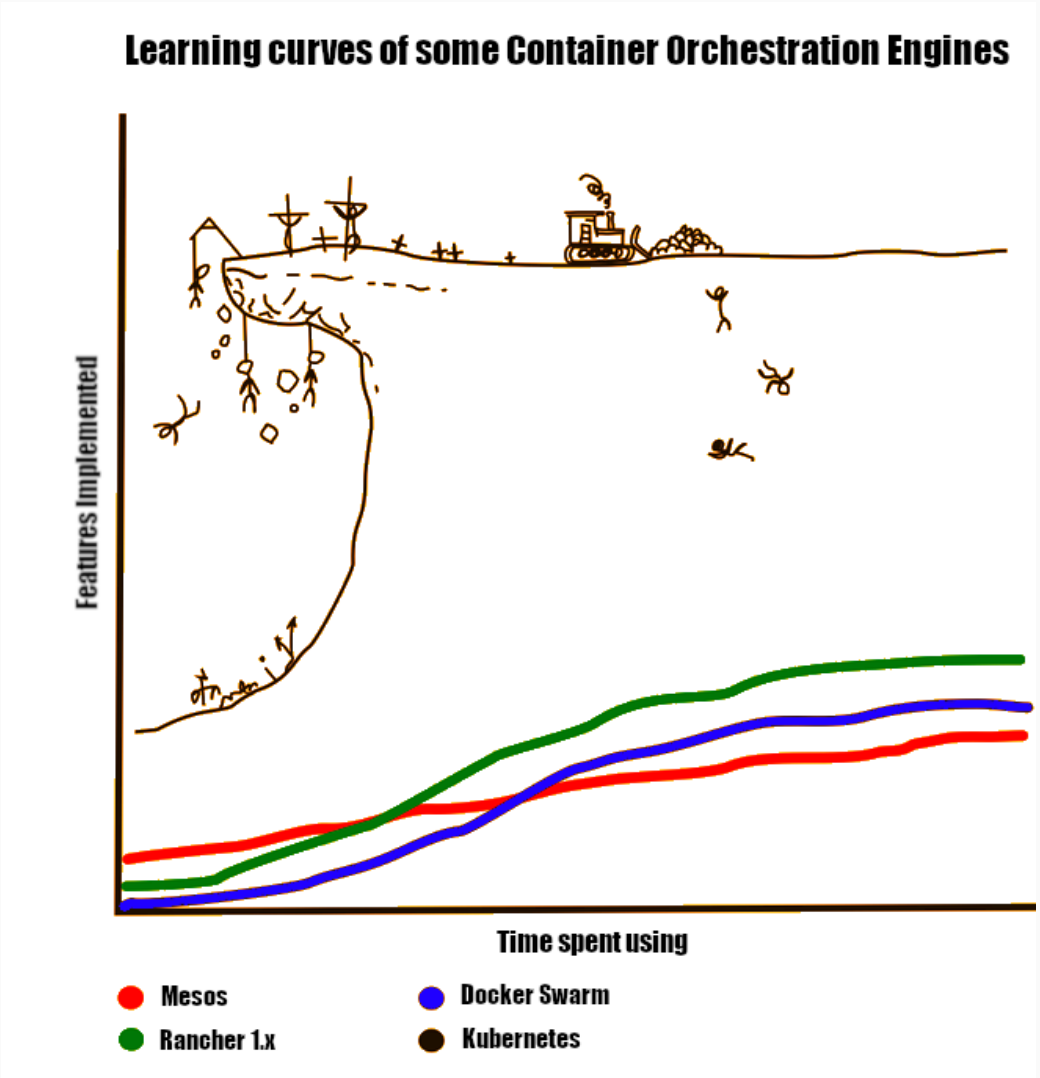


Figure 1: Une réputation méritée ?

Kubernetes: historique

- Deux projets Google dont Borg, en Java
- Réécriture en Go, version 1.0 en 2015
- Pilotage par Google, puis par la Cloud Native Computing Foundation (Linux Foundation) depuis août 2018

Kubernetes

« We must treat the datacenter itself as one massive warehouse-scale computer [1]*. »*

[1] in *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, André B.L., Clidaras J., Hölzle U

Kubernetes: objectifs

- L'abstraction des couches matérielles et système
- Le couplage faible des composants
- Un surcoût minimal
- Fonctionnement indifférent sur machines physiques et sur machine virtuelles

☸ Kubernetes: concepts

- Une API déclarative: on définit des contrats pour nos applications
- Des capacités d'autoréparation
- Une infrastructure immutable

Kubernetes: architecture

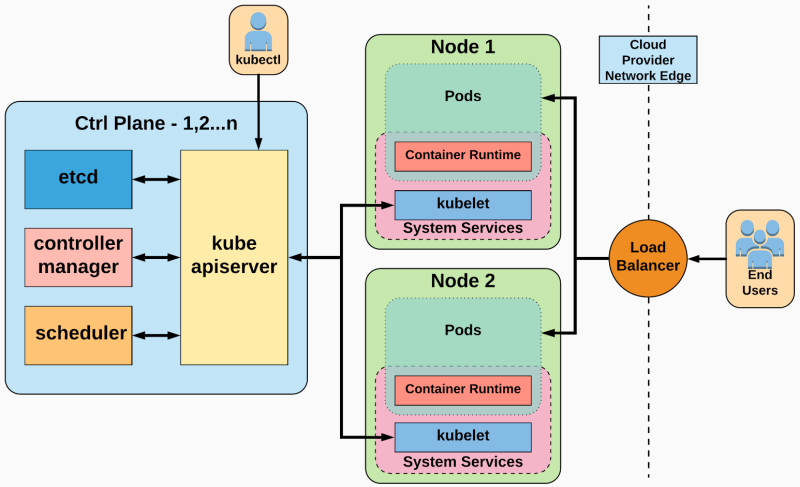


Figure 2: Architecture

- Une séparation nette du plan de travail et du plan de contrôle
- Configuration par un point unique, via l'API
- Composants faiblement couplés : communication uniquement avec l'API

Kubernetes: le plan de contrôle

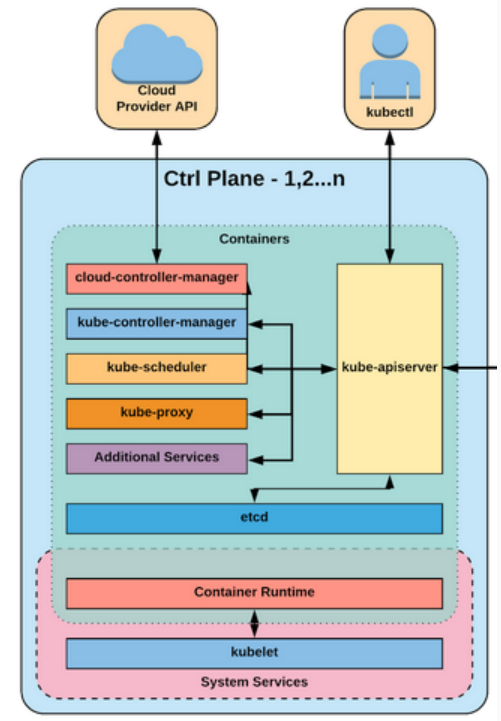


Figure 3: Plan de contrôle

- L'API au centre du système
- etcd pour le stockage clé/valeur
- Des boucles de contrôle: *kube-controller-manager*
- Un scheduler: *kube-scheduler*
- L'intégration à l'infrastructure sous-jacente: *cloud-controller-manager*.

Kubernetes: le plan de travail

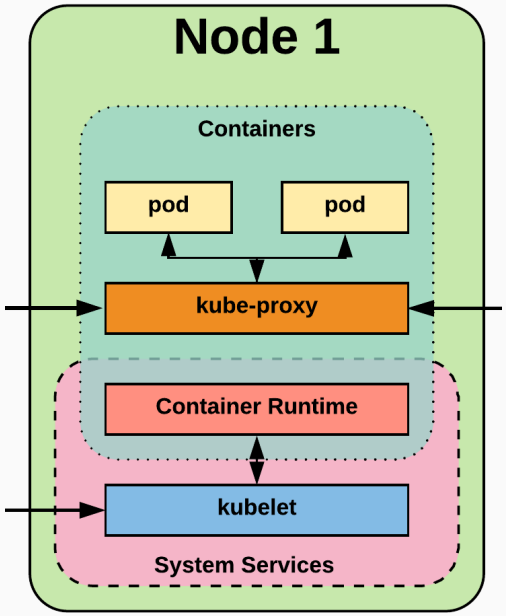


Figure 4: Plan de travail

- Gestion des charges de travail (*pods*) : *kubelet*
- Gestion des règles de *forwarding* et de l'équilibrage de charge : *kube-proxy*
- Un *runtime* compatible *Container Runtime Interface* : Docker, containerd, rkt, CRI-O, Kata, Singularity-CRI..

☸ Kubernetes: les objets de base de l'API

- **Namespace**: conteneur logique
- **Pod**: unité de base
- **Deployment**, **StatefulSet**, **DaemonSet**: ordonnancement
- **Service**: réseau et répartition de charge
- **Job**, **CronJob**: tâches
- **PersistentVolume**, **PersistentVolumeClaim**: volumes persistants
- **ConfigMap**, **Secret**: configuration et secret
- **Ingress**: règles de *reverse proxy*

Kubernetes: *manifest*

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello
spec:
  template:
    # This is the pod template
    spec:
      containers:
      - name: hello
        image: busybox
        command: ['sh', '-c', 'echo "Hello, Kubernetes!"']
      restartPolicy: OnFailure
    # The pod template ends here
```

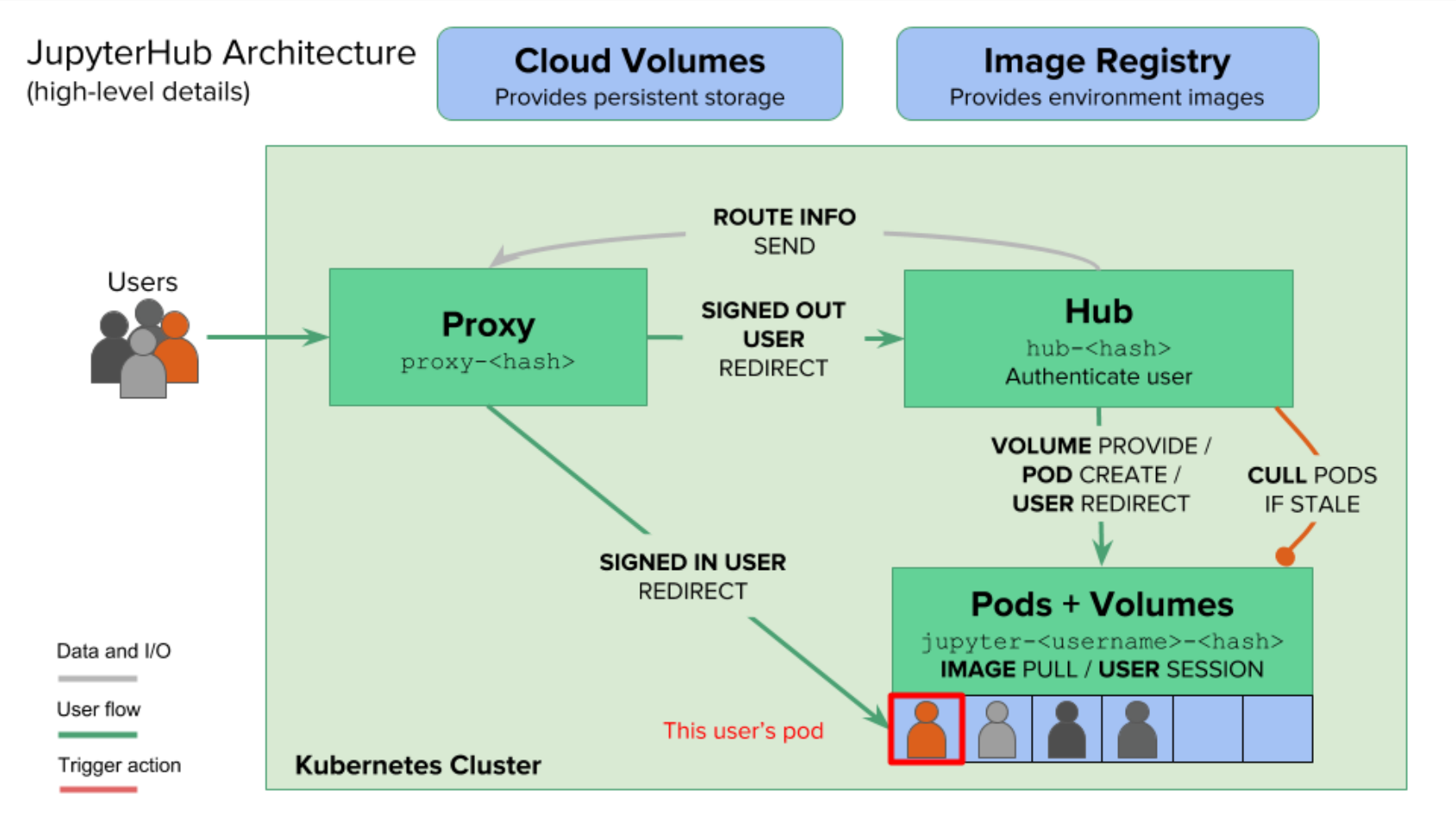
JupyterHub

- Serveur de *notebooks* (singleuser)
- Personnalisable: langage, interface, ...
- Flexible, scalable, portable
- Plusieurs *spawners* disponibles: *SystemdSpawner*, *SudoSpawner*, *DockerSpawner*, **KubeSpawner**...

L'utilisation d'images Docker facilite la création et la distribution d'environnements (transfert vers l'univers du HPC par ex.)

JupyterHub: KubeSpawner

Avantages: isolation, contrôle des ressources, passage à l'échelle



Les outils: Packer

- Hashicorp, Open Source (MPL2)
- Construction d'image de VM
- Multi plateforme (VirtualBox, vSphere, Openstack, différents *cloud providers*)
- Utilisation possible d'un *SCM* (puppet, ansible, salt, chef)

Les outils: Packer

```
{
    "builders": [
        {
            "type": "openstack",
            "image_name": "ubuntu-18.04-docker-x86_64",
            "source_image_name": "ubuntu-18.04-bionic-x86_64",
            "ssh_username": "ubuntu",
            "networks": [
                "59981445-9379-44f2-ad36-a4e91e174f96"
            ],
            "floating_ip_network": "b7658133-6810-4804-aa76-a0ab096092ee",
            "floating_ip": "7ac29e38-3332-4a21-b568-ab2d8fac6600",
            "security_groups": "default"
        }
    ],
    "provisioners": [
      {
        "type": "salt-masterless",
        "bootstrap_args": "-X -d -D stable 2019.2.0",
        "local_state_tree": "./salt",
        "custom_state": "docker",
        "salt_call_args": "pillar='{\"docker_version\": \"5:18.09.1~3-0~ubuntu-bionic\"}'"
      },
      {
        "type": "shell",
        "inline": ["sudo adduser ubuntu docker"]
      }
    ]
}
```

Les outils: Terraform

HashiCorp
Terraform

- Hashicorp, Open Source (MPL2)
- *Infrastructure as Code*
- Multi plateforme (plusieurs centaines de fournisseurs)
- Gestion des dépendances
- Gestion et partage de l'état de l'infrastructure

Les outils: Terraform

```
resource "openstack_compute_instance_v2" "instance" {
  name            = "nvidialic"
  image_name      = "ubuntu-18.04.4-bionic-x86_64"
  flavor_name     = "m1.medium"
  key_pair        = "cailletr-rsa"
  network {
    uuid = openstack_networking_network_v2.nvidialic_net.id
    fixed_ip_v4 = "192.168.0.10"
  }
}
```

Les outils: Helm

- *Package manager* pour Kubernetes (charts)
- Système de template pour K8S
- Gestion des dépôts
- De très nombreux logiciels empaquetés

La pratique: vue d'ensemble

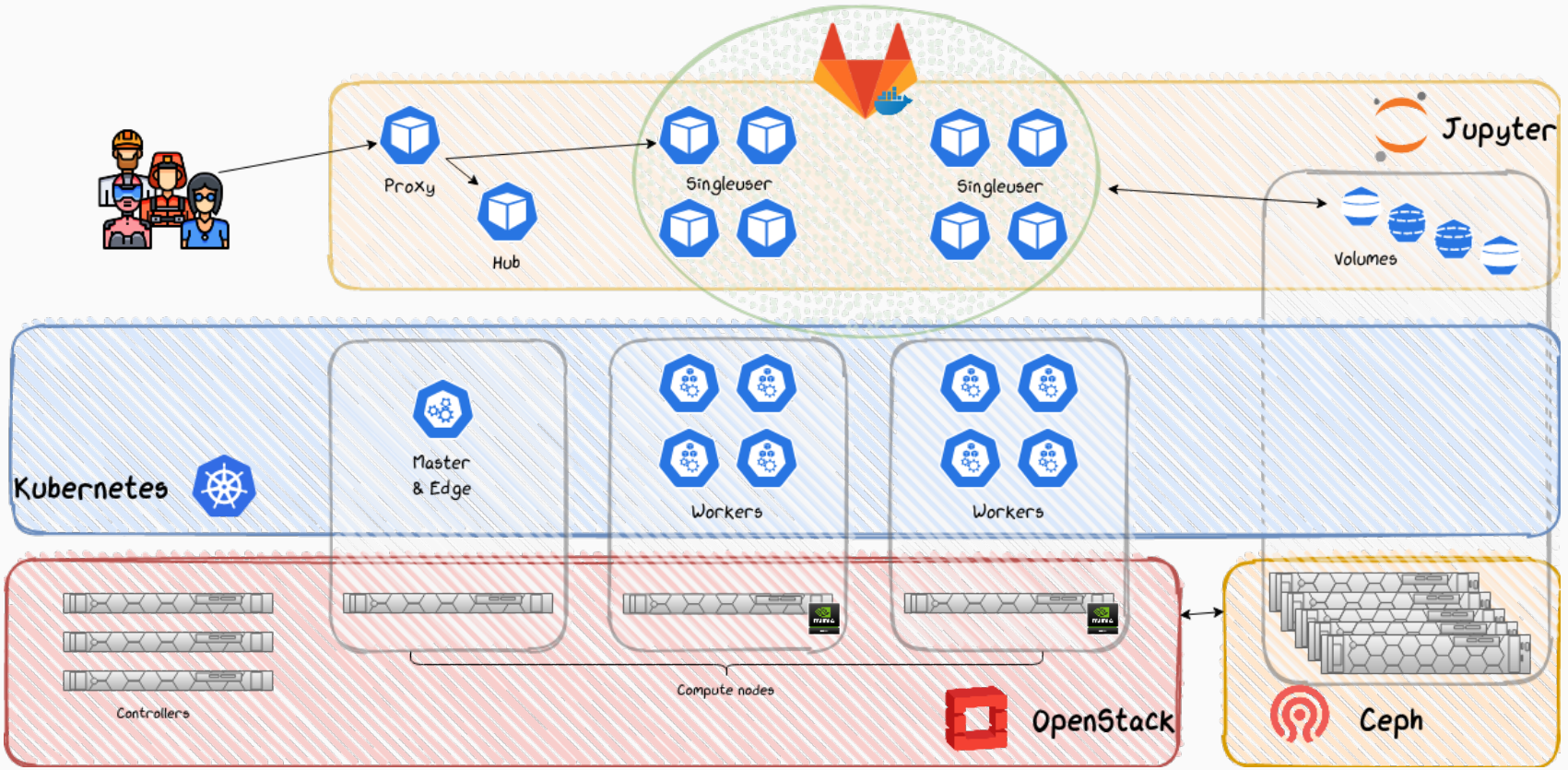


Figure 5: Architecture

terraform-openstack-rke

- *a Terraform module to deploy Kubernetes with RKE on OpenStack*
- Création des routeurs, réseaux, groupes de sécurité, machines virtuelles, IP flottantes et déploiement de Kubernetes
- utilisation de terraform-provider-openstack et terraform-provider-rke

Zero To JupyterHub with Kubernetes (z2jh)

- *a Helm chart for JupyterHub*
- Déploiement des différents composants de JupyterHub à l'aide d'un chart Helm

Merci de votre attention !

UST4HPC

- Présentation JupyterHub/GPU avec K8S
- TP gpu-hub

Autres resources

- JRES 2019: Article Kubernetes
- ANF ADA: Kubernetes TP

Made with Pandoc, Metropolis Beamer Theme and FontAwesome