# Déploiement de Kubernetes dans cloud.ca

Clément Contini

20 mars 2018

# Objectif

**Objectif** : Présenter comment utiliser un outil d'*infrastructure en tant que code* pour déployer un cluster Kubernetes dans un système infonuagique public.

Cette présentation va aborder les points suivants :

- Explication des technologies impliquées
- Détails de la configuration Terraform utilisée
- Présentation du résultat dans une démo

# À propos de nous

*Clément Contini*

- Diplôme d'ingénierie logicielle de l'ÉTS de Montréal
- Je travaille sur l'infrastructure de notre plateforme infonuagique à CloudOps et cloud.ca

*CloudOps*

- Experts en infrastructure infonuagique depuis 2005
- Conçoit, construit et s'occupe de solutions infonuagiques publiques, privées ou hybrides

*cloud.ca*

- Plateforme d'infrastructure en tant que service, basée à Montréal
- Existe depuis août 2014

# Terraform

Outil d'infrastructure en tant que code :

- Compatible avec plusieurs plateformes
- Les configurations sont versionnables et réutilisables
- Peut générer des plans d'exécution
- Nous avons créé un connecteur pour cloud.ca

Outils similaires :

- OpenStack Heat
- AWS CloudFormation


**Figure 1 –** logo de Terraform

# Kubernetes

- C'est un système d'orchestration de conteneurs
- Provient à l'origine de Borg de Google

Outils similaires :

- Docker Swarm
- $\sim$ Apache Mesos / Marathon
- $\sim$ HashiCorp Nomad

**Figure 2 –** logo de Kubernetes

# Démonstration

Cette démo va créer un cluster Kubernetes en utilisant *kubeadm*.

Nous allons déployer :

- un environnement dans cloud.ca avec un VPC, des réseaux et toutes les machines virtuelles dont nous avons besoin
- Un maître Kubernetes et plusieurs travailleurs sur ces machines virtuelles
- Démarrer un déploiement de Vault en créant 3 répliques de chaque conteneur

Nous allons lancer la démo et expliquer la configuration pendant que le cluster se fait créer.

# Démonstration

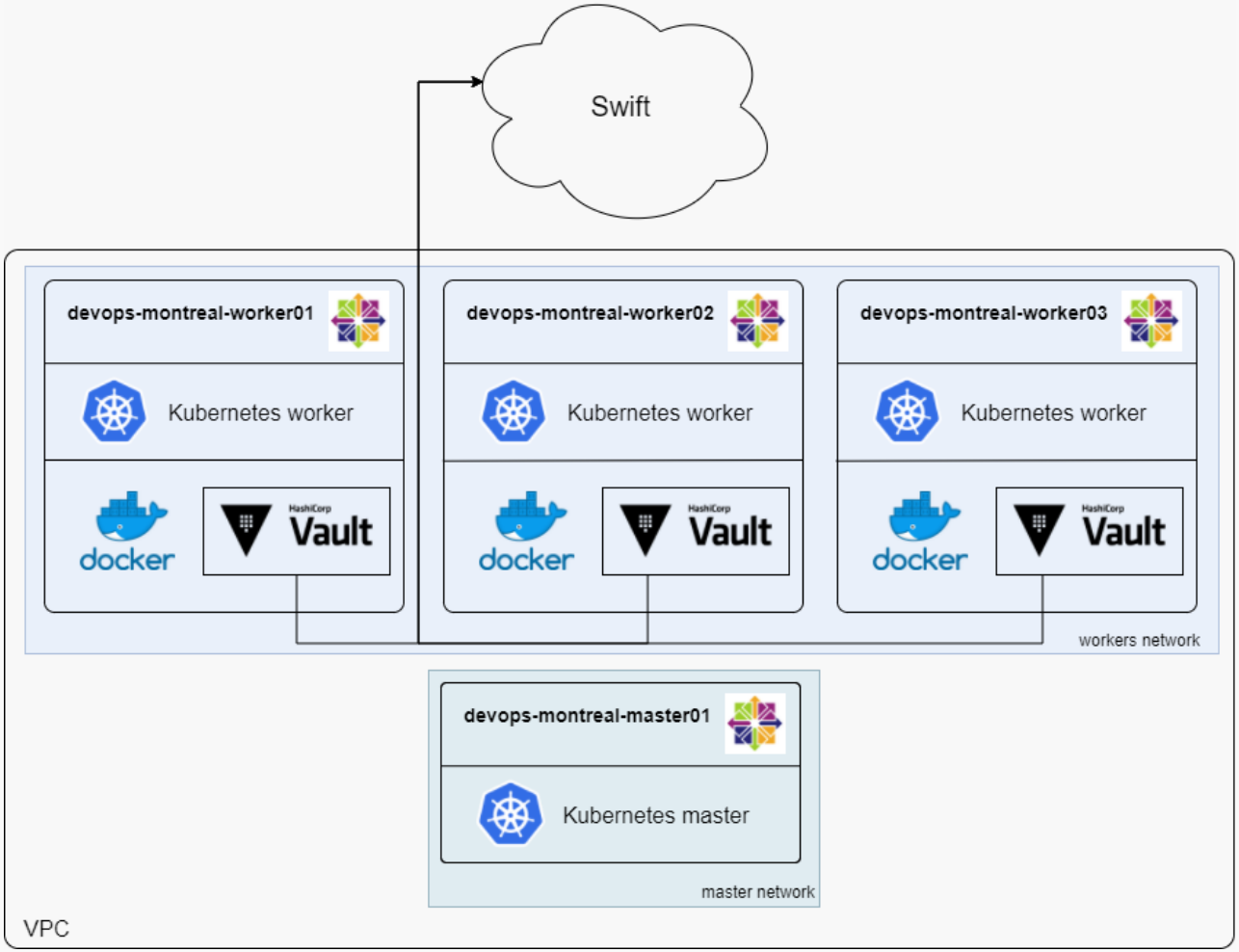


**Figure 3 –** Déploiement du cluster Kubernetes

# Environment

Nous allons commencer par créer un environnement dans cloud.ca qui va accueillir notre déploiement.

Voici les ressources que nous allons créer :

- un cloudca_environment
- un cloudca_vpc
- deux cloudca_public_ip

```
1 resource "cloudca_environment" "kubernetes" {
2   service_code      = "${var.service_code}"
3   organization_code = "${var.organization_code}"
4   name              = "${format("%s-env", var.prefix)}"
5   description       = "Environment for a Kubernetes cluster"
6   admin_role        = ["${var.admin}"]
7   read_only_role    = ["${var.read_only}"]
8 }
```

**Listing 1** – définition de l'environnement

# Variables

Nous allons déclarer les variables qui vont nous aider à configurer l'environnement, dans notre cas :

- service_code
- organization_code
- prefix
- admin
- read_only

```
1 variable "service_code" { default = "compute-on" }
2 variable "organization_code" {}
3 variable "admin" { type = "list" }
4 variable "read_only" { type = "list" }
5 variable "prefix" {}
```

**Listing 2 –** Déclaration des variables

# Variables

Certaines variables ont besoin d'être spécifiées (par exemple la clé d'API) et d'autres peuvent simplement utiliser les valeurs par défaut. Pour spécifier une valeur, vous devez l'écrire dans le fichier *terraform.tfvars*.

```
1 service_code      = "compute-qc"
2 organization_code = "system"
3 admin             = [ "ccontini" ]
4 read_only         = [ ]
5 prefix            = "devops-montreal"
```

**Listing 3 –** Assignation des variables

# Maître

Nous pouvons maintenant définir toutes les ressources nécessaires pour démarrer le maître :

- Un réseau et des ACLs qui lui seront appliquées
- Une instance
- Une règle de redirection de port

```
1 resource "cloudca_network" "master" {
2   environment_id   = "${cloudca_environment.kubernetes.id}"
3   name             = "${format("%s-network-master", var.
      ↪ prefix)}"
4   description      = "Network for the master nodes"
5   vpc_id           = "${cloudca_vpc.kubernetes.id}"
6   network_offering = "Standard Tier"
7   network_acl_id   = "${cloudca_network_acl.master.id}"
8 }
```

**Listing 4 –** Définition du réseau du maître

# Maître

```
1  resource "cloudca_instance" "master_node" {
2    environment_id          = "${cloudca_environment.kubernetes
       ↪  .id}"
3    name                    = "${format("%s-master01", var.
       ↪  prefix)}"
4    network_id              = "${cloudca_network.master.id}"
5    template                = "${var.template_name}"
6    compute_offering        = "${var.default_offering}"
7    cpu_count               = "${var.master_vcpu}"
8    memory_in_mb            = "${var.master_ram}"
9    root_volume_size_in_gb = "${var.master_disk}"
10   user_data               = "${data.template_file.cloudinit.
       ↪  rendered}"
11 }
```

**Listing 5** – Définition de la machine maître

# Maître

Le maître est configuré en utilisant un script cloudinit. Celui-ci va :

- Créer un utilisateur sur la machine virtuelle
- Installer Docker et Kubernetes
- Partir un script d'initialisation pour démarrer le cluster

```
1 data "template_file" "cloudinit" {
2   template = "${file("templates/cloudinit.tpl")}"
3
4   vars {
5     public_key = "${replace(file("./id_rsa.pub"), "\n", "")}
        ↪ "
6     username   = "${var.username}"
7   }
8 }
```

**Listing 6 –** Interpolation du template Cloudinit

# Maître

```
1 resource "cloudca_port_forwarding_rule" "
    ↪ management_master_ssh" {
2   environment_id      = "${cloudca_environment.kubernetes.id}
      ↪ "
3   public_ip_id        = "${cloudca_public_ip.master_ip.id}"
4   public_port_start   = 2200
5   private_ip_id       = "${cloudca_instance.master_node.
      ↪ private_ip_id}"
6   private_port_start = 22
7   protocol            = "TCP"
```

**Listing 7 –** Redirection de port pour accéder au maître

Une fois que la redirection de port est créée, on l'utilise pour fournir à la machine le script d'initialisation et un manifeste qui sera appliqué quand le cluster va démarrer.

# Maître

```
1   connection {
2     type        = "ssh"
3     user        = "${var.username}"
4     private_key = "${file("./id_rsa")}"
5     host        = "${cloudca_public_ip.master_ip.ip_address}
          ↪ "
6     port        = 2200
7   }
8
9   provisioner "file" {
10    content     = "${data.template_file.bootstrap_master.
          ↪ rendered}"
11    destination = "/home/${var.username}/bootstrap.sh"
12  }
13
14  provisioner "file" {
15    source      = "manifests"
16    destination = "/home/${var.username}/manifests"
17  }
```

**Listing 8 –** Configuration du maître

# Travailleurs

On créé les travailleurs de la même façon, avec :

- Un réseau et des ACLs qui lui seront appliquées
- Plusieurs instances (nous allons utiliser le mot clé *count*)
- Une règle de redirection de port

```
1  resource "cloudca_instance" "worker_nodes" {
2    environment_id          = "${cloudca_environment.kubernetes
       ↪  .id}"
3    name                    = "${format("%s-worker%02d", var.
       ↪  prefix, count.index + 1)}"
4    network_id              = "${cloudca_network.worker.id}"
5    template                = "${var.template_name}"
6    compute_offering        = "${var.default_offering}"
7    cpu_count               = "${var.worker_vcpu}"
8    memory_in_mb            = "${var.worker_ram}"
9    root_volume_size_in_gb = "${var.worker_disk}"
10   count                   = "${var.worker_count}"
11   user_data               = "${data.template_file.cloudinit.
       ↪  rendered}"
12 }
```

**Listing 9 –** Définition des instances travailleurs

# Déploiement de Vault

Voyons ce qui a été déployé dans Kubernetes.

# Quelques alternatives

Plusieurs solutions peuvent être employées pour créer et gérer d'un cluster Kubernetes :

- Nous aurions pu utiliser le fournisseur de Terraform pour Kubernetes pour gérer les ressources dans le cluster Kubernetes.
- Nous aurions pu prendre une autre approche et utiliser Rancher, ce qui nous aurait permis de déployer le cluster avec une interface graphique.

*cloud.ca* envisage de s'intégrer de façon plus poussée avec Kubernetes pour permettre de créer des ressources consommables par Kubernetes à travers l'API de cloud.ca, par exemple :

- Des volumes
- Des règles de répartition de charge

# Questions

Des questions ?

Merci de votre présence à ce meetup !

Le code source est disponible ici :
`https://github.com/vilisseranen/terraform-kubeadm`.