

Apprendre Kubernetes et créer son propre labo expérimental



Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Introduction.

Kubernetes est incontestablement devenu le standard en matière de déploiement d'applications Cloud-natives. Un composant clé que doivent maîtriser tous les opérationnels et les développeurs. Inscrire **Kubernetes** parmi ses compétences créera sur un CV un vrai distinguo. Reste donc à apprendre. Et sur ce point, la courbe d'apprentissage de cet orchestrateur de containers est bien abrupte.

Ce guide vous accompagnera dans la mise en place de votre propre labo Kubernetes afin que vous puissiez tester vos connaissances et accélérer votre maîtrise de cette nouvelle technologie

MiniKube ou Vagrant ? Le choix dépendra des connaissances que vous souhaitez acquérir. Pour un labo Kubernetes facile à démarrer et qui s'intègre bien avec divers OS, l'option Minikube (dans le premier article de ce guide) permet au débutant de se concentrer sur l'utilisation d'un cluster Kubernetes, plutôt que sur la manière de le déployer sur une infrastructure.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Pour une expérimentation qui explore avant tout les caractéristiques de déploiement de Kubernetes, le labo qui utilise Vagrant (second article de ce guide) héberge un [Docker](#) de base et un cluster Kubernetes sur trois VM Linux.

Maîtriser les rudiments de Kubernetes est devenu un pré-requis. Ce guide vous accompagne dans la mise en place d'un centre Kubernetes pour tester cette technologie.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Comment créer son cluster Kubernetes en local avec Minikube

Alastair Cooke, Consultant

Ce tutoriel vous guide dans l'installation de l'outil Minikube et dans la mise en place de son propre environnement de test sur Kubernetes. De quoi apprendre et tester les fonctions de l'orchestrateur de containers n°1.

Kubernetes est devenu la brique logicielle standard si l'on souhaite entrer dans le monde des containers et des architectures de [microservices](#). Le hic : sa courbe d'apprentissage peut souvent rebuter les moins aguerris. Ces professionnels ont dès lors besoin de se familiariser avec la belle mécanique de [Kubernetes](#) et de développer leurs compétences à leur rythme. Une des solutions : monter son propre Kubernetes et apprendre.

Ce tutoriel utilise MiniKube, un outil développé pour installer l'orchestrateur de containers en local et commencer à tester les fonctions. Cela ne nécessite pas l'installation d'équipement et de logiciels supplémentaires, ni même d'un investissement en temps important pour l'installer. Ce laboratoire domestique permet donc d'isoler l'essentiel de la nouvelle technologie, hors des

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

environnements critiques de la production. En fait, [Minikube](#) exécute un cluster Kubernetes à un nœud dans une VM.

Cet article vous guide pas à pas dans l'installation de votre labo, vous aide à exécuter les commandes kubectl, et vous transporte au plus près des workloads dans K8.

Un labo Kubernetes : quelques exigences

Un cluster Minikube Kubernetes est installé pré-construit et fonctionne dans une seule machine virtuelle sur l'ordinateur de l'utilisateur. Minikube fonctionne sous Linux, Windows et MacOS et peut utiliser une variété d'hyperviseurs pour sa VM.

Les lignes de commande Minikube kubectl s'exécutent directement sur l'ordinateur d'origine, et les applications motorisées par Kubernetes y sont également accessibles.

Ce tutoriel ne fait tourner qu'une simple application hébergée par Kubernetes sous macOS avec Oracle VM VirtualBox. Les processus d'installation pour Linux et Windows sont un peu différents. Il existe également d'autres hyperviseurs. Le site web de Kubernetes [détaille le déploiement de Minikube sur n'importe quel OS](#).

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Minikube : installation

Pour démarrer ce tutoriel, installez la dernière version de VirtualBox, puis installez la ligne de commande kubectl [pour gérer Kubernetes](#), par téléchargement direct. Minikube est également installé par téléchargement direct.

```
AlastaisMacBook:~ alastaircooke$ curl -LO https://storage.googleapis.com/kuberne
tes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/r
elease/stable.txt`/bin/darwin/amd64/kubectl
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 63.7M 100 63.7M 0 0 6290k 0 0:00:10 0:00:10 --:--:-- 7703k
AlastaisMacBook:~ alastaircooke$ chmod +x ./kubectl
AlastaisMacBook:~ alastaircooke$ sudo mv ./kubectl /usr/local/bin/kubectl
AlastaisMacBook:~ alastaircooke$
AlastaisMacBook:~ alastaircooke$ curl -Lo minikube https://storage.googleapis.co
m/minikube/releases/v0.25.0/minikube-darwin-amd64
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 41.2M 100 41.2M 0 0 5816k 0 0:00:07 0:00:07 --:--:-- 7475k
AlastaisMacBook:~ alastaircooke$ chmod +x minikube
AlastaisMacBook:~ alastaircooke$ sudo mv minikube /usr/local/bin/
AlastaisMacBook:~ alastaircooke$
```

Figure 1. Installez kubectl et Minikube sur votre machine.

Rendez-vous [au dépôt GitHub](#) dédié aux labos Kubernetes, puis trouvez Minikube.txt. Ces commandes peuvent être clonées et collées dans une fenêtre de terminal. Une fois l'installation terminée, Minikube démarrera et téléchargera la Minikube VM, puis le cluster Kubernetes.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Une fois téléchargé, Minikube n'a pas besoin d'un accès Internet - à condition que toutes les images Docker nécessaires aient déjà été téléchargées.

```
AlastaisMacBook:~ alastaircooke$ minikube start
Starting local Kubernetes v1.9.0 cluster...
Starting VM...
Downloading Minikube ISO
 142.22 MB / 142.22 MB [=====] 100.00% 0s
Getting VM IP address...
Moving files into cluster...
Downloading localkubernetes binary
 162.41 MB / 162.41 MB [=====] 100.00% 0s
 65 B / 65 B [=====] 100.00% 0s
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
Loading cached images from config file.
AlastaisMacBook:~ alastaircooke$
```

Figure 2. Minikube start configure votre labo Kubernetes en local.

Minikube et Kubernetes : utilisation

A ce stade du tutoriel, vous avez tout ce dont vous avez besoin pour commencer à apprendre Kubernetes. Démarrez deux instances du serveur web Nginx, puis vérifiez que les deux instances sont disponibles.

Dans ce guide

- Comment créer son cluster
Kubernetes en local avec
Minikube

- Comment bien se former à
Kubernetes

- Comment utiliser Github dans
son labo DevOps

- Git : 5 commandes basiques
que l'on doit maîtriser

- Démarrer avec Ansible pour
configurer son labo DevOps

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  config.vm.define "master", primary: true do |master|
    master.vm.hostname = 'master'
    master.vm.box = "bento/fedora-25"
    master.vm.network "private_network", ip: "172.16.33.10"
    master.vm.provision :shell, path: "master.sh"
    master.vm.provider "virtualbox" do |v|
      v.memory = 2048
      v.gui = true
    end
  end
  config.vm.define "node1" do |node1|
    node1.vm.box = "bento/fedora-25"
    node1.vm.hostname = 'node1'
    node1.vm.network "private_network", ip: "172.16.33.11"
    node1.vm.provision :shell, path: "worker.sh"
    node1.vm.provider "virtualbox" do |v|
      v.memory = 1024
      v.cpus = 1
    end
  end
  config.vm.define "node2" do |node2|
    node2.vm.box = "bento/fedora-25"
    node2.vm.hostname = 'node2'
    node2.vm.network "private_network", ip: "172.16.33.12"
    node2.vm.provision :shell, path: "worker.sh"
    node2.vm.provider "virtualbox" do |v|
      v.memory = 1024
      v.cpus = 1
    end
  end
end
```

Figure 3. Deux replicas de Nginx tournent sur Minikube.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Nginx s'exécute dans deux conteneurs Docker mais n'est pas accessible hors du cluster Kubernetes. La façon la plus simple d'exposer le déploiement et de le rendre accessible de l'extérieur est d'utiliser l'adresse IP standard 192.168.99.100 (de la Minikube VM) et d'exposer un NodePort. Le système convertit l'adresse pour le port TCP de cette adresse IP vers l'adresse IP interne du conteneur.

```
AlastaisMacBook:~ alastaircooke$ sudo kubectl expose deployment my-nginx --port=80 --type=NodePort
service "my-nginx" exposed
AlastaisMacBook:~ alastaircooke$ sudo kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	8m
my-nginx	NodePort	10.101.62.218	<none>	80:30709/TCP	2s

```
AlastaisMacBook:~ alastaircooke$
```

Figure 4. Expose un NodePort pour permettre à Nginx d'accéder depuis l'extérieur au cluster Kubernetes.

Pour s'assurer que Nginx est actif, ouvrez un navigateur Web, et entrez `http://192.168.99.100:<port>`, où le port est celui indiqué dans la commande `kubectl get services`, comme le montre la Figure 4 ci-dessus.

Un serveur web est maintenant déployé sur Kubernetes.

Jusqu'à présent, nous avons utilisé des outils en ligne de commande pour gérer Kubernetes, ce qui est une excellente pratique pour l'automatisation.

Cependant, une interface graphique facilite grandement l'exploration de ce qui

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

se passe dans ce labo Kubernetes. Utilisez la commande minikube dashboard pour lancer un navigateur Web et ouvrir le tableau de bord.

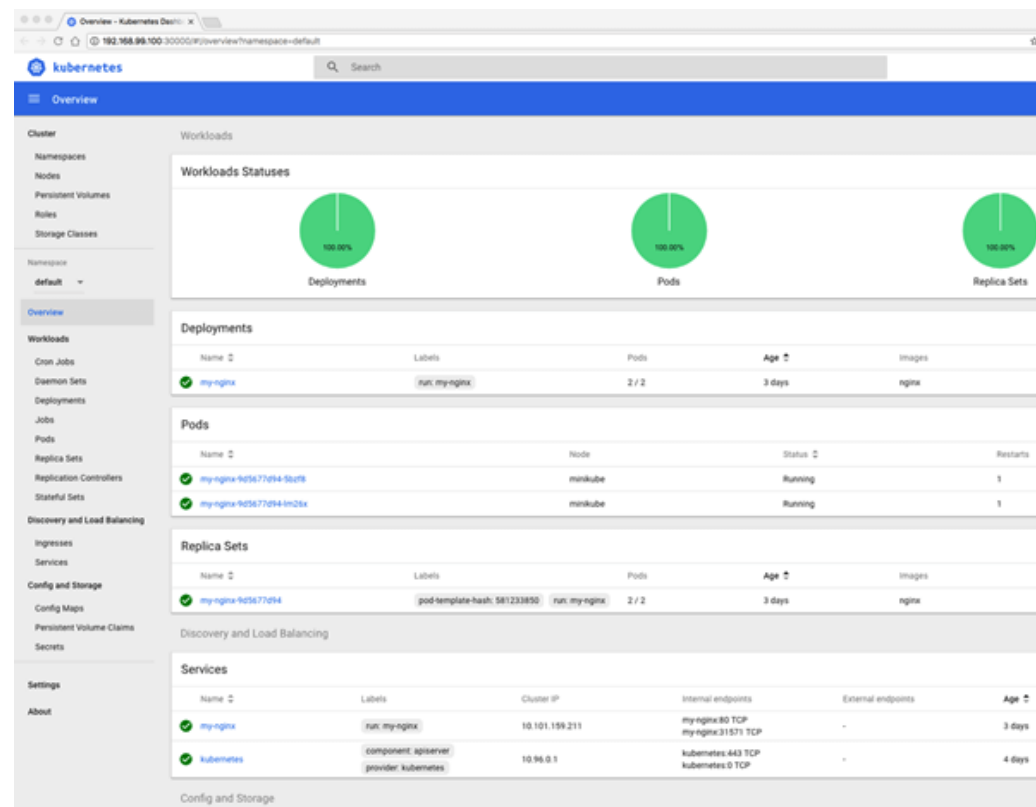


Figure 5. Une interface graphique (GUI) permet aux utilisateurs d'explorer leur propre laboratoire Kubernetes en dehors de la ligne de commande.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Le tableau de bord affiche des informations sur les déploiements actifs, l'exécution de Pods (là où se trouvent les containers) et plusieurs autres éléments qui dépassent le cadre de ce tutoriel.

Pour fermer Minikube, exécutez la commande `minikube stop` pour arrêter la VM. Minikube conservera les déploiements ainsi que les services définis au démarrage. Pour empêcher le déploiement de reprendre, supprimez l'installation `kubectl delete po,svc,deploy my-nginx` avant de fermer Minikube.

Un cluster Kubernetes en production aura certes plus d'un hôte ; ce laboratoire n'est utile que pour des tests à petite échelle. Cependant, ce tutoriel donne les premières réponses que l'on doit se poser pour comprendre Kubernetes et prendre en main cette technologie désormais indissociable des architectures modernes.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Comment bien se former à Kubernetes

Alastair Cooke, Consultant

Les serveurs en production ne sont assurément pas les meilleurs environnements pour se tester à essayer un nouvel outil. Cet article vous aide à configurer un cluster Kubernetes sur votre machine pour effectuer vos expérimentations.

S'ils ne sont pas déjà en production dans les entreprises, les containers sont et seront à coup sûr la prochaine étape de l'évolution des SI. Mettre en place son propre labo Kubernetes est un bon moyen de comprendre l'orchestration et la gestion de ces containers.

Ce didacticiel porte avant tout sur l'installation et le support du cluster, matériel compris. Cet environnement, qui servira avant tout à la formation, doit être une réplique d'une installation sur site de Kubernetes. Son déploiement s'appuie sur HashiCorp Vagrant, un outil en ligne de commande qui peut déployer des environnements virtuels. Il inclut trois machines virtuelles Linux et des versions spécifiques de Docker et de Kubernetes.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Vagrant et Kubernetes : les pré-requis

Pour commencer, installez Vagrant. Rendez-vous sur vagrantup.com, et téléchargez le programme d'installation pour votre OS. Vagrant fonctionne avec de nombreuses plateformes de virtualisation, mais ce didacticiel utilise Oracle VM VirtualBox (gratuite et open source) car elle est facile à utiliser. Installez donc la dernière version de VirtualBox.

Ensuite, installez le dépôt Git. Ce labo Kubernetes est disponible dans les fichiers Git de l'auteur de cet article.

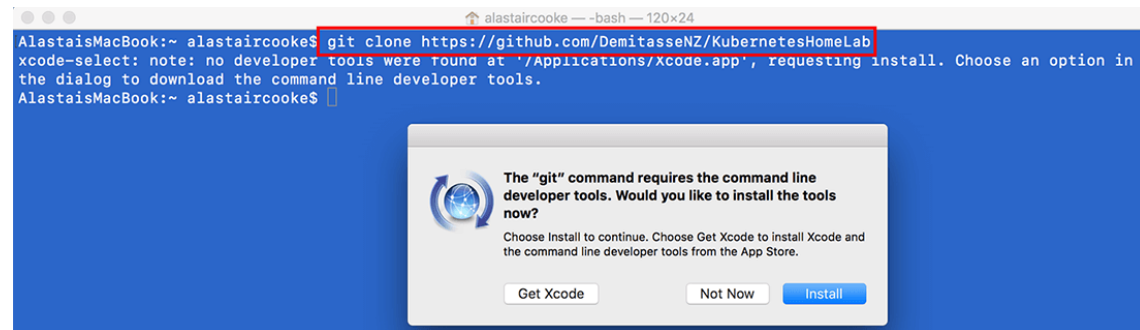


Figure 1

Les outils Git s'installent automatiquement lorsque l'on exécute pour la première fois une commande Git (voir Figure 1). Il existe une variété d'outils Git, comme l'application officielle GitHub Desktop (pour Windows par exemple).

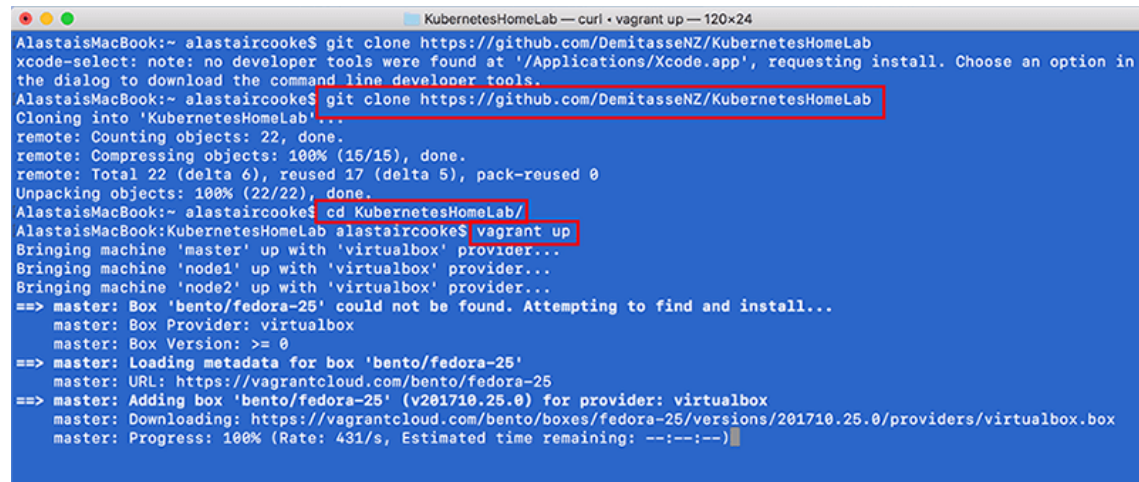
Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Ce laboratoire a été créé sur Red Hat Fedora 25. Cet OS offre la possibilité d'utiliser la bonne version de Docker et permet un déploiement d'infrastructure Kubernetes assez simple. Kubernetes a la réputation d'être un projet complexe. Vous pouvez alors partager du code sur un dépôt GitHub avec d'autres membres de la communauté opérationnelle de Kubernetes.

Cloner ce labo Kubernetes

Une fois Git installé, clonez le dépôt depuis github.com/DemitasseNZ/KubernetesHomeLab. Allez dans le répertoire et tapez Vagrant up.



```
KubernetesHomeLab — curl • vagrant up — 120x24
AlastaisMacBook:~ alastaircooke$ git clone https://github.com/DemitasseNZ/KubernetesHomeLab
xcode-select: note: no developer tools were found at '/Applications/Xcode.app', requesting install. Choose an option in
the dialog to download the command line developer tools.
AlastaisMacBook:~ alastaircooke$ git clone https://github.com/DemitasseNZ/KubernetesHomeLab
Cloning into 'KubernetesHomeLab'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 22 (delta 6), reused 17 (delta 5), pack-reused 0
Unpacking objects: 100% (22/22), done.
AlastaisMacBook:~ alastaircooke$ cd KubernetesHomeLab/
AlastaisMacBook:KubernetesHomeLab alastaircooke$ vagrant up
Bringing machine 'master' up with 'virtualbox' provider...
Bringing machine 'node1' up with 'virtualbox' provider...
Bringing machine 'node2' up with 'virtualbox' provider...
==> master: Box 'bento/fedora-25' could not be found. Attempting to find and install...
master: Box Provider: virtualbox
master: Box Version: >= 0
==> master: Loading metadata for box 'bento/fedora-25'
master: URL: https://vagrantcloud.com/bento/fedora-25
==> master: Adding box 'bento/fedora-25' (v201710.25.0) for provider: virtualbox
master: Downloading: https://vagrantcloud.com/bento/boxes/fedora-25/versions/201710.25.0/providers/virtualbox.box
master: Progress: 100% (Rate: 431/s, Estimated time remaining: --:--:--)
```

Figure 2.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Le clonage prendra un certain temps, selon la connexion Internet. Le système télécharge une image VirtualBox de Fedora 25 et l'utilise pour créer trois VM : un master Kubernetes et deux slaves. Docker et Kubernetes se déploient ensuite sur les trois VM, toujours par téléchargement.

Une fois le master finalisé, ce dernier [initialise un cluster Kubernetes](#) - ce qui prend quelques minutes de plus pour le mettre en place.

Comme c'est toujours le cas avec un labo local, les spécifications de l'ordinateur portable déterminent la réactivité de votre espace d'expérimentation ; beaucoup de RAM, de disques durs SSD et un processeur rapide aideront en cela.

Utilisez le temps de téléchargement et d'installation pour en savoir plus sur Vagrant et sur comment bâtir l'infrastructure Kubernetes.

Le fichier Vagrantfile permet de créer les trois machines virtuelles Fedora 25, chacune avec une adresse IP sur le réseau privé (voir Figure 3). La machine virtuelle maître tourne en premier et utilise le script master.sh pour terminer sa configuration. Les deux nœuds worker utilisent le script worker.sh.

Dans ce guide

- Comment créer son cluster
Kubernetes en local avec
Minikube

- Comment bien se former à
Kubernetes

- Comment utiliser Github dans
son labo DevOps

- Git : 5 commandes basiques
que l'on doit maîtriser

- Démarrer avec Ansible pour
configurer son labo DevOps

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  config.vm.define "master", primary: true do |master|
    master.vm.hostname = 'master'
    master.vm.box = "bento/fedora-25"
    master.vm.network "private_network", ip: "172.16.33.10"
    master.vm.provision :shell, path: "master.sh"
    master.vm.provider "virtualbox" do |v|
      v.memory = 2048
      v.gui = true
    end
  end
  config.vm.define "node1" do |node1|
    node1.vm.box = "bento/fedora-25"
    node1.vm.hostname = 'node1'
    node1.vm.network "private_network", ip: "172.16.33.11"
    node1.vm.provision :shell, path: "worker.sh"
    node1.vm.provider "virtualbox" do |v|
      v.memory = 1024
      v.cpus = 1
    end
  end
  config.vm.define "node2" do |node2|
    node2.vm.box = "bento/fedora-25"
    node2.vm.hostname = 'node2'
    node2.vm.network "private_network", ip: "172.16.33.12"
    node2.vm.provision :shell, path: "worker.sh"
    node2.vm.provider "virtualbox" do |v|
      v.memory = 1024
      v.cpus = 1
    end
  end
end
```

Figure 3.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Le script master (voir Figure 4) est plus complexe que celui du worker, car il doit configurer le cluster Kubernetes et la machine virtuelle maître que l'administrateur utilisera pour gérer le cluster. Le premier bloc de master.sh (en rouge) concerne l'installation de Docker, en particulier Docker version 17.03 dans cet article. La ligne qui démarre sed -i change une ligne dans un fichier de configuration Docker afin que celui-ci utilise les pilotes que Kubernetes préfère. Le deuxième bloc (en rose) installe Kubernetes et modifie à nouveau un fichier de configuration par souci de cohérence. Le troisième bloc (en bleu) crée le cluster Kubernetes et le plug-in de mise en réseau qui permet aux connexions réseau de fonctionner sur les nœuds. Les deux lignes du script de formation de cluster Kubernetes redirigent leur sortie dans un log. Il se situe dans /vagrant, qui est un dossier à l'intérieur de la machine virtuelle. Il correspond au dossier où se trouve le Vagrantfile.

Figure 4.

```
#!/usr/bin/env bash

swapoff -a
dnf -y install dnf-plugins-core
dnf config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo
dnf install -y docker-ce-17.03.0.ce-1.fc25
sed -i "s|ExecStart=/usr/bin/dockerd|ExecStart=/usr/bin/dockerd --exec-opt native.cgroupdriver=cgroupfs|" /lib/systemd/system/docker.service
systemctl daemon-reload
systemctl enable docker.service
systemctl start docker
```

1. Install Docker

```
cat /vagrant/hosts | tee -a /etc/hosts
setenforce 0
cp /vagrant/kubernetes.repo /etc/yum.repos.d/
yum install -y kubelet kubeadm kubectl
sed -i 's|driver=systemd|driver=cgroupfs|g' /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
systemctl daemon-reload
systemctl enable kubelet
systemctl start kubelet
```

2. Install Kubernetes

```
kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address 172.16.33.10 > /vagrant/kubeinit.log
mkdir ~/.kube
cp -i /etc/kubernetes/admin.conf ~/.kube/config
export KUBECONFIG=~/.kube/config
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml >> /home/vagrant/kubeinit.log
```

3. Initialise Kubernetes

```
mkdir -p /home/vagrant/.kube
cp -i /etc/kubernetes/admin.conf /home/vagrant/.kube/config
chown vagrant:vagrant /home/vagrant/.kube
chown vagrant:vagrant /home/vagrant/.kube/config
echo "export KUBECONFIG=/home/vagrant/.kube/config" | tee -a /home/vagrant/.bashrc
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Bien que l'installation nécessite un certain temps (voir Figure 5), il a fallu une demi-heure pour construire les VM et le cluster Kubernetes. Par exemple, le téléchargement de l'image de Fedora 25 a pris beaucoup de temps, mais vous ne répétez pas cette étape si vous recréez un labo identique sur ce même ordinateur.

```
node2:
node2: Installing : kubeadm-1.9.2-0.x86_64 6/6
node2:
node2: Verifying : kubelet-1.9.2-0.x86_64 1/6
node2:
node2: Verifying : kubernetes-cni-0.6.0-0.x86_64 2/6
node2:
node2: Verifying : kubeadm-1.9.2-0.x86_64 3/6
node2:
node2: Verifying : kubectl-1.9.2-0.x86_64 4/6
node2:
node2: Verifying : ethtool-2:4.13-1.fc25.x86_64 5/6
node2:
node2: Verifying : socat-1.7.3.2-1.fc25.x86_64 6/6
node2:
node2: Installed:
node2: ethtool.x86_64 2:4.13-1.fc25 kubeadm.x86_64 1.9.2-0
node2: kubectl.x86_64 1.9.2-0 kubelet.x86_64 1.9.2-0
node2: kubernetes-cni.x86_64 0.6.0-0 socat.x86_64 1.7.3.2-1.fc25
node2: Complete!
node2: Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /etc/systemd/system/kubelet.service.
AlastairMacBook:KubernetesHomeLab alastaircooke$
```

Figure 5.

Le fichier log, Kubeinit.log (voir Figure 6), commence par les actions prises par kubeadm, une boîte à outils de bootstrapping pour clusters Kubernetes. Les logs montrent alors que le master est initialisé. Ensuite, le log donne la ligne de commande à utiliser pour relier les nœuds worker au cluster. Le token est un mot de passe pour que les nœuds rejoignent le cluster, tandis que le certificat d'autorité à la fin (ca-cert) assure qu'il a rejoint le bon.

Dans ce guide

- Comment créer son cluster
Kubernetes en local avec
Minikube
- Comment bien se former à
Kubernetes
- Comment utiliser Github dans
son labo DevOps
- Git : 5 commandes basiques
que l'on doit maîtriser
- Démarrer avec Ansible pour
configurer son labo DevOps

```
[init] Using Kubernetes version: v1.9.3
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks.
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [master kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "scheduler.conf"
[controlplane] Wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"
[controlplane] Wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] Wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"
[init] Waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests".
[init] This might take a minute or longer if the control plane images have to be pulled.
[apiclient] All control plane components are healthy after 73.088007 seconds
[uploadconfig] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[markmaster] Will mark node master as master by adding a label and a taint
[markmaster] Master node tainted and labelled with key/value: node-role.kubernetes.io/master=""
[bootstraptoken] Using token: d6744b.17b50e0a5fe9bdf1
[bootstraptoken] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstraptoken] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstraptoken] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: kube-dns
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

kubeadm join --token d6744b.17b50e0a5fe9bdf1 172.16.33.10:6443 --discovery-token-ca-cert-hash sha256:250e7742e9785b9a340703b6bf32898b3efed6db41da1f03dc
```

Figure 6.

Comment utiliser ce labo

Vagrant partage le dossier /home/vagrant entre les nœuds. Le script worker.sh est identique au script master, jusqu'à la fin du processus d'installation de Kubernetes, où la commande kubeadm join ... est extraite du log et exécutée.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Vagrant ouvre une console pour le maître ; les deux nœuds restent headless, ce qui est la valeur par défaut de Vagrant. Utilisez ssh master vagrant pour ouvrir une session sur le master, ou utilisez la fenêtre de la console. Pour observer le cluster, connectez-vous à la console de la machine virtuelle maître comme utilisateur vagrant avec vagrant pour mot de passe. Ensuite, lancez `sudo kubectl get nodes` : vous avez trois nœuds, un maître et deux avec <none> comme rôle.

```
[vagrant@master ~]$ sudo kubectl get nodes
NAME        STATUS    ROLES    AGE     VERSION
master      Ready     master   11m     v1.9.3
[vagrant@master ~]$ sudo kubectl get nodes
NAME        STATUS    ROLES    AGE     VERSION
master      Ready     master   17m     v1.9.3
node1       NotReady  <none>    24s     v1.9.3
node2       NotReady  <none>    10s     v1.9.3
[vagrant@master ~]$ _
```

Figure 7.

Ce didacticiel crée un cluster Kubernetes de base que l'on peut tester sans se soucier de problème de production. Pour arrêter le laboratoire et libérer ses 4 Go de RAM, utilisez `vagrant halt`. Vous pouvez utiliser `vagrant destroy` pour supprimer toutes les VM, puis `vagrant up` pour ré-initialiser. Pour que ces commandes fonctionnent, elles doivent être saisies dans le répertoire où réside le Vagrantfile.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Comment utiliser Github dans son labo DevOps

Alastair Cooke, Consultant

Cet article vous aide à faire vos premiers pas avec Git et GitHub pour l'intégrer à votre environnement d'expérimentation DevOps.

Un déploiement [DevOps](#) nécessite une infrastructure qui peut être rapidement provisionnée pour s'adapter aux besoins des applications. Lorsque l'on crée [son environnement d'expérimentation DevOps](#) avec Vagrant VM, ce dernier produit des fichiers contenant du code qui crée et décrit l'infrastructure du labo. Le logiciel de contrôle de version Git, ou toute autre outil - il est préférable d'en choisir un adapté -, protège ce code et archive les anciennes versions lorsque des modifications sont réalisées.

Un commit Git sauvegarde un état cohérent - une copie séparée - du projet sur la machine du labo DevOps. Il permet à l'utilisateur de revenir à un point fixe en cas de modification du code. [GitHub](#) est d'une grande aide pour cela : il s'agit d'une plateforme de contrôle de version distribuée, qui se comporte comme [un référentiel central pour le code](#).

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Git : le commencement

Installez Git sur la machine de votre labo DevOps. L'installation est automatique lorsque git s'exécute pour la première fois.

Ouvrez le terminal dans le dossier où se trouve le fichier Vagrant.

Configurez Git pour lui faire savoir qui vous êtes, en entrant votre nom et votre adresse email :

```
amc$ git config --global user.name "Alastair Cooke"  
amc$ git config --global user.email "alastair@example.com"
```


Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Puis initialisez Git dans ce dossier. Ajoutez le fichier VagrantFile et validez le fichier avec les commandes suivantes :

```
amc$ git init
Initialized empty Git repository in /Users/amc/Dropbox
(vBrownBagCrew)/Demitasse/Writing/DevOpsHomeLab/.git/
amc$ git add Vagrantfile
amc$ git commit -m 'Initial Vagrantfile with three VMs'
[master (root-commit) 0d3d4cc] Initial Vagrantfile with three VMs
1 file changed, 40 insertions(+)
create mode 100644 Vagrantfile
```

©2017 TECHTARGET ALL RIGHTS RESERVED 

Des modifications peuvent être effectuées librement car vous pouvez être sûr que le déploiement peut revenir à cet état à tout moment. Utilisez GitHub à chaque fois que vous effectuez un changement significatif.

La commande `git add` permet d'ajouter des fichiers. Testez également certaines fonctions comme `git add *.sh` pour ajouter tous les scripts shell.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Les fichiers des expériences de cet article peuvent être clonés [depuis le dépôt GitHub](#) :

```
amc$ git clone https://github.com/DemitasseNZ/DevOpsHomeLab
Cloning into 'DevOpsHomeLab'...
remote: Counting objects: 15, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15 (delta 2), reused 15 (delta 2), pack-reused 0
Unpacking objects: 100% (15/15), done.
amc$ ls -l
total 0
drwxr-xr-x 14 amc staff 476 2 Mar 14:00 DevOpsHomeLab
amc$ cd DevOpsHomeLab/
amc$ ls -l
total 72
-rw-r--r-- 1 amc staff 55 2 Mar 14:00 AnsibleHosts
-rw-r--r-- 1 amc staff 2336 2 Mar 14:00 DeployDocker.yml
-rw-r--r-- 1 amc staff 392 2 Mar 14:00 ReadMe.md
-rw-r--r-- 1 amc staff 1247 2 Mar 14:00 Vagrantfile
-rw-r--r-- 1 amc staff 70 2 Mar 14:00 ansible.sh
-rw-r--r-- 1 amc staff 1691 2 Mar 14:00 docker-stack.yml
-rw-r--r-- 1 amc staff 440 2 Mar 14:00 master.sh
-rw-r--r-- 1 amc staff 41 2 Mar 14:00 ping.yml
-rw-r--r-- 1 amc staff 61 2 Mar 14:00 python.sh
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Clonez les fichiers puis copiez-les dans le dossier de votre projet. Enfin, rendez tous les scripts shell exécutables :

```
amc$ sudo chmod a+x *.sh
amc$ ls -l
total 72
-rw-r--r--  1 amc  staff    55  2 Mar 14:00 AnsibleHosts
-rw-r--r--  1 amc  staff  2336  2 Mar 14:00 DeployDocker.yml
-rw-r--r--  1 amc  staff   392  2 Mar 14:00 ReadMe.md
-rw-r--r--  1 amc  staff  1247  2 Mar 14:00 Vagrantfile
-rwxr-xr-x  1 amc  staff    70  2 Mar 14:00 ansible.sh
-rw-r--r--  1 amc  staff  1691  2 Mar 14:00 docker-stack.yml
-rwxr-xr-x  1 amc  staff   440  2 Mar 14:00 master.sh
-rw-r--r--  1 amc  staff    41  2 Mar 14:00 ping.yml
-rwxr-xr-x  1 amc  staff    61  2 Mar 14:00 python.sh
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Git : 5 commandes basiques que l'on doit maîtriser

Cameron McKenzie, TechTarget

Cet article vous livre les 5 commandes Git que l'on doit absolument connaître pour utiliser correctement le système de contrôle de versions de code.

Git représente aujourd'hui un outil de base que l'on doit maîtriser si l'on souhaite créer des lignes de code en mode projet pour la création d'un logiciel. Il s'agit d'une des clés du développement et du déploiement d'applications et un des maillons du processus [DevOps](#).

Une fois Git installé, la prochaine étape est de maîtriser ces cinq commandes de base de Git : init, config, add, status et commit. Ce tutoriel Git pour débutants vous guidera pas à pas à travers chacune de ces commandes.

La commande git init

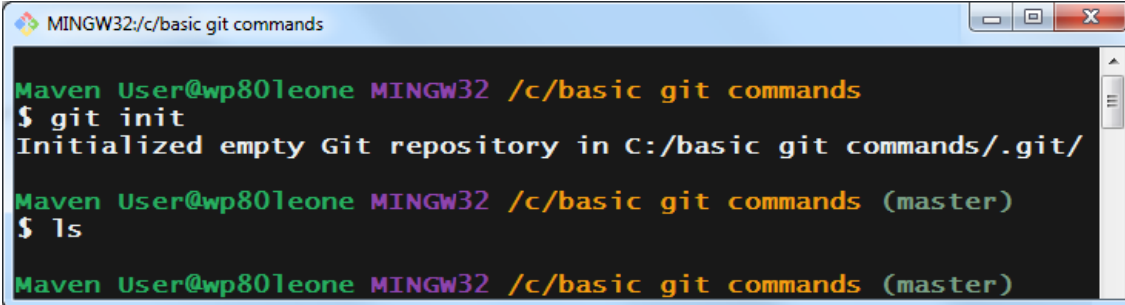
L'essor La première des cinq commandes de base est init. Cette commande de base initialise un dépôt Git.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Si vous explorez le dossier dans lequel cette commande s'exécute, vous noterez un dossier caché nommé `.git`, avec plusieurs sous-dossiers - hooks, info, objets et refs. Git y va gérer et maintenir l'historique complet des sources de votre dépôt.

Vous n'avez besoin d'installer Git qu'une seule fois, après quoi vous pouvez créer un nombre illimité de dépôts Git, tant que ces dépôts ne sont pas imbriqués. Chaque référentiel est autonome et ne connaît aucun autre référentiel Git.



```

MINGW32:/c/basic git commands
Maven User@wp801eone MINGW32 /c/basic git commands
$ git init
Initialized empty Git repository in C:/basic git commands/.git/
Maven User@wp801eone MINGW32 /c/basic git commands (master)
$ ls
Maven User@wp801eone MINGW32 /c/basic git commands (master)

```

La première commande de base de Git que les débutants doivent apprendre : `git init`.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

La commande git config

C'est probablement la moins intéressante des cinq commandes basiques de Git. Si vous venez d'installer Git, vous ne pouvez pas lancer une commande `commit` sans d'abord configurer votre nom et votre adresse email. Cela se fait avec la commande `git config`.

Après avoir installé Git mais avant de [pouvoir valider un code](#), Git a besoin de connaître le nom et l'adresse email du committer. Cela signifie qu'il faut exécuter deux fois la commande `git config`. Pour la première exécution, utilisez l'attribut `user.name`, et la deuxième fois, l'attribut `user.email`.

```
/c/ basic git commands (tutorial)
$ git config --global user.name "Learn Git"
```

```
/c/ basic git commands (tutorial)
$ git config --global user.email "basic@commands.com"
```

Il y a trois types de configuration Git : *local*, *global* et *system*. Le global l'emporte sur le local, et le system sur le global. La meilleure pratique est de définir les propriétés `user.name` et `user.email` au niveau global et de ne plus jamais s'en soucier.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Où Git stocke-t-il la configuration « global » ?

Si vous regardez dans le répertoire `.git` de n'importe quel référentiel, vous trouverez un fichier sans extension nommé `config`.

Toute la configuration en local de Git y est stockée.

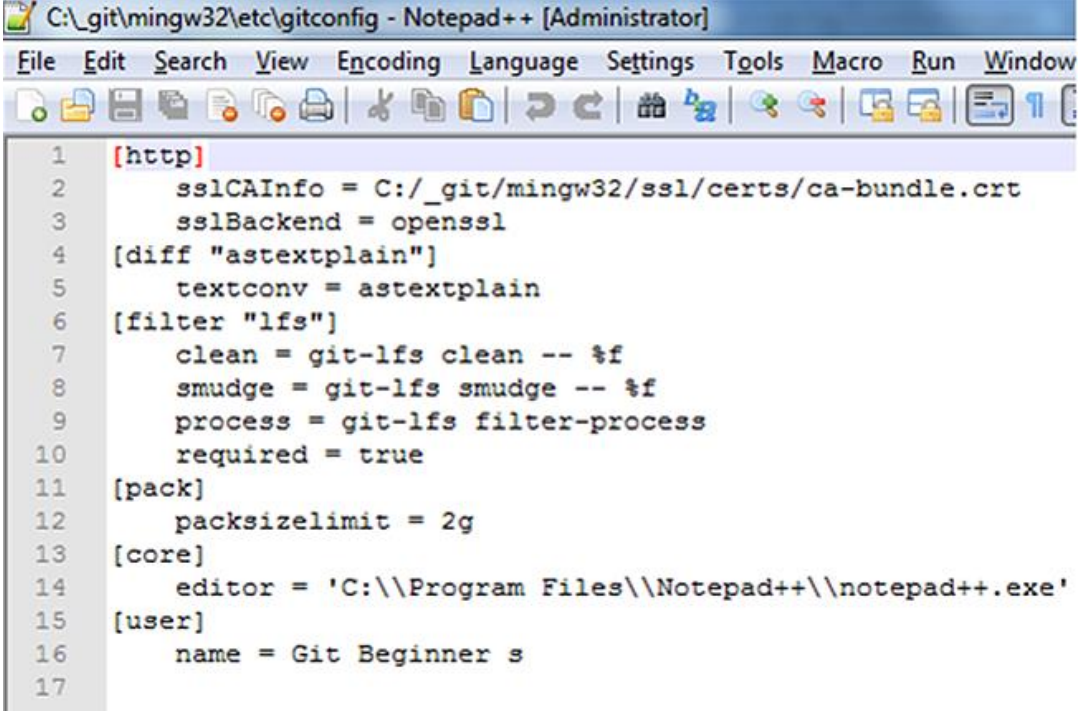
Dans le dossier du profil de l'utilisateur, on retrouve un fichier nommé `.gitconfig`.

C'est ici que toute la configuration « global » de Git est stockée.

Enfin, la [configuration du système Git](#) est sauvegardée dans un fichier nommé `gitconfig`, qui se trouve dans le sous-dossier `ming\etc` de votre installation Git.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps



```

1  [http]
2      sslCAInfo = C:/_git/mingw32/ssl/certs/ca-bundle.crt
3      sslBackend = openssl
4  [diff "astextplain"]
5      textconv = astextplain
6  [filter "lfs"]
7      clean = git-lfs clean -- %f
8      smudge = git-lfs smudge -- %f
9      process = git-lfs filter-process
10     required = true
11 [pack]
12     packsizelimit = 2g
13 [core]
14     editor = 'C:\\Program Files\\Notepad++\\notepad++.exe'
15 [user]
16     name = Git Beginner s
17

```

Les données de configuration du système Git sont sauvegardées dans le sous-dossier etc de l'installation.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

La commande git add

Vous n'avez pas besoin de committer tous les fichiers au contrôle du code source. Avant qu'un développeur ne fasse un commit, il doit explicitement identifier les fichiers à regrouper dans le cadre du commit. Ceci s'effectue en lançant une commande git add avec le nom des fichiers à inclure dans le prochain commit.

Un exemple avec git add

Dans les exemples suivants de shell Bash, nous allons créer trois fichiers avec la commande echo, à savoir cinq.html, basic.html et commands.html.

```
/c/ basic git commands (tutorial)
$ echo "5" > five.html && echo "b" > basic.html && echo "c"
> commands.html
```

```
/c/ basic git commands (tutorial)
$ ls
basic.html  commands.html  five.html
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Cependant, malgré le fait que les trois fichiers résident dans le même répertoire - là où le dépôt Git a été initialisé -, seuls ceux associés à une commande git add feront partie du prochain commit. Donc, si nous voulons laisser le fichier command.html en dehors du prochain commit, nous ne l'identifierons tout simplement pas dans un appel git add.

```
/c/ basic git commands (tutorial)  
$ git add basic.html
```

```
/c/ basic git commands (tutorial)  
$ git add five.html
```

Il existe un terme technique pour identifier les fichiers qui feront partie de la prochaine livraison : le staging. Lorsque vous lancez une commande git add sur un fichier, vous avez placé le fichier dans l'index Git. L'index Git est ce qui garde la trace de tous les fichiers à inclure dans le prochain commit.

Si vous avez modifié un grand nombre de fichiers et que vous voulez les placer en staging, vous pouvez lancer la commande git add[space]. Ceci ajoutera chaque fichier modifié à l'index, à l'exception de tout ce qui est répertorié dans le fichier .gitignore.

```
/c/ basic git commands (tutorial)  
$ git add.
```

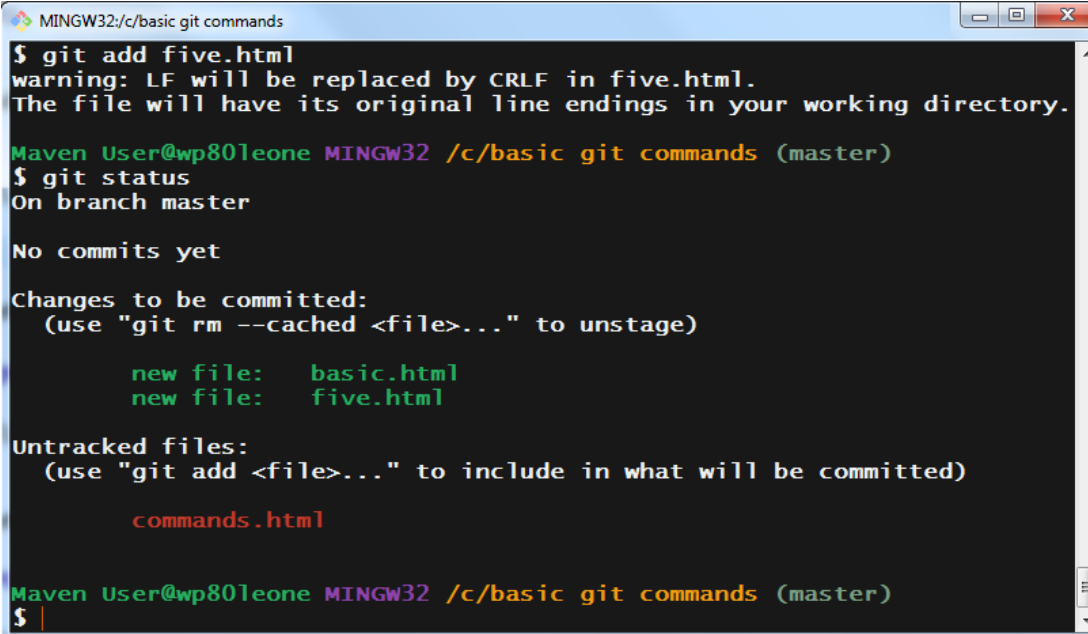
Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

La commande git status

Parfois, les développeurs perdent la trace de ce qu'ils ont ou n'ont pas ajouté à la zone de staging. C'est pourquoi nous incluons git status à ce tutoriel. Cette commande indiquera sur quelle branche vous êtes, quels fichiers sont en staging et quels fichiers restent non suivis et n'ont pas été ajoutés à l'index. La commande git status est un excellent outil pour comprendre rapidement l'état actuel de votre travail.

Apprenez la commande d'état de git pour pouvoir afficher l'état de l'arborescence de travail.



```

MINGW32/c/basic git commands
$ git add five.html
warning: LF will be replaced by CRLF in five.html.
The file will have its original line endings in your working directory.
Maven User@wp801eone MINGW32 /c/basic git commands (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   basic.html
        new file:   five.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        commands.html

Maven User@wp801eone MINGW32 /c/basic git commands (master)
$
  
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

La commande git commit

Il s'agit de la dernière commande de base à connaître.

Pour sauvegarder de façon permanente les modifications apportées au dépôt Git, vous devez réaliser un commit. Ce dernier doit être associé à un message qui le décrit. De plus, Git garde la trace du nom et de l'adresse email de l'utilisateur qui effectue ce commit. C'est pourquoi vous devez fournir ces informations via la commande git config.

Pour committer, utilisez l'option -m pour fournir un message texte décrivant le commit. Si vous n'utilisez pas cette commande, Git demandera une entrée dans l'interpréteur de commandes Bash, via l'éditeur Vi.

```
/c/ basic git commands (tutorial)
$ git commit -m "basic git commands tutorial completed!"
[tutorial (root-commit) d6a80cf] basic git commands
tutorial completed!
 2 files changed, 2 insertions(+)
 create mode 100644 basic.html
 create mode 100644 five.html
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Démarrer avec Ansible pour configurer son labo DevOps

Alastair Cooke, Consultant

Ansible est devenu l'outil de référence en matière de gestion de configuration. Il convient dès lors à en connaître le fonctionnement. Cet article vous en donne les premiers rudiments, associé à GitHub et Vagrant.

La seule installation de Linux ne suffit pas si l'on souhaite lancer ses propres expérimentations [DevOps](#) ; les machines virtuelles doivent être configurées pour s'adapter à l'application et à ses utilisateurs, et imiter au plus près l'environnement de production réel. Les outils de gestion de configuration sont d'une grande aide quand il s'agit de gérer ces nouveaux systèmes et leurs mises à jour, car ils éliminent le travail manuel.

Il existe de nombreux outils de gestion de configuration, mais cet article ne portera que sur Ansible, devenu une des références des entreprises. C'est un bon outil, sans agent, si on souhaite [apprendre les concepts d'automatisation de DevOps](#).

Ansible est un outil open source né chez Red Hat.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Installer et exécuter Ansible

Pour [apprendre Ansible](#), il convient d'installer certains prérequis : Ansible sur la VM maître et Python sur les nœuds. Pour les besoins de cet article, l'outil de gestion de configuration a été associé à Vagrant - l'installation fait partie du processus de provisioning de Vagrant :

```
node1.vm.provision :shell, path: "python.sh"
```

©2017 TECH-TARGET ALL RIGHTS RESERVED TechTarget

Vagrant copie le fichier depuis le même dossier que le fichier Vagrant et l'exécute à l'intérieur de la machine virtuelle node1 pendant le provisioning. Le fichier contient des commandes pour installer certains paquets :

```
#!/usr/bin/env bash
apt-get update
apt-get install -y python
```

©2017 TECH-TARGET ALL RIGHTS RESERVED TechTarget

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

node1 et node2 utilisent ce même script pour installer Python, qui fournit l'environnement d'exécution d'Ansible. Pour la VM maître, utilisez un script différent appelé `ansible.sh` pour installer Ansible ainsi que Python. Vous avez déjà créé les VMs, alors utilisez la commande `vagrant provision` pour que Vagrant exécute ces scripts. Utilisez `vagrant destroy`, puis `vagrant up` pour avoir un déploiement propre.

Une fois la configuration d'Ansible finalisée, committez à Git en ajoutant les deux nouveaux fichiers script shell (.sh) puis en validant les modifications :

```
amc$ ls -l
total 24
-rw-r--r--@ 1 amc  staff  1247  1 Mar 17:13 Vagrantfile
-rwxr-xr-x@ 1 amc  staff    70  1 Mar 16:04 ansible.sh
-rwxr-xr-x@ 1 amc  staff    61  1 Mar 15:28 python.sh
amc$ git add *.sh
amc$ git commit -m 'Added Ansible deployment to Vagrantfile'
[master 495de9f] Added Ansible deployment to Vagrantfile
2 files changed, 8 insertions(+)
create mode 100755 ansible.sh
create mode 100755 python.sh
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Ce laboratoire DevOps nécessite une authentification par clé RSA pour que l'utilisateur de la VM maître puisse se connecter (SSH) dans les nœuds sans mot de passe.

C'est ainsi qu'Ansible gère la configuration sans agent : Il se connecte via SSH et exécute des commandes.

Utilisez `ssh vagrant` pour vous connecter au maître, puis exécutez la commande `ssh-keygen` pour créer des clés **RSA** pour l'utilisateur Vagrant :

```
vagrant@master:~$ ssh-keygen -b 2048 -t rsa -f /home/vagrant/.ssh/id_rsa -q -N ""
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

La commande `ssh-copy-id` copie une clé dans node1. Il vous invite à accepter la clé **SSH** et à fournir le mot de passe de l'utilisateur Vagrant, qui est *vagrant*.

Exécutez à nouveau la commande pour copier la clé dans le nœud2 :

```
vagrant@master:~$ ssh-copy-id vagrant@192.168.33.11
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vagrant/.ssh/id_rsa.pub"
The authenticity of host '192.168.33.11 (192.168.33.11)' can't be established.
ECDSA key fingerprint is SHA256:RI5/3ep65qXeDkZSACi/rN0hBxiLrBxMvcky9CfLkyg.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
vagrant@192.168.33.11's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'vagrant@192.168.33.11'"
and check to make sure that only the key(s) you wanted were added.
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Le laboratoire est maintenant configuré, mais ce n'est que le début.

Regardez le fichier d'inventaire qui liste les machines gérées par Ansible.

La liste d'inventaire par défaut est conservée dans `/etc/ansible/hosts`, où sont listés le maître et les nœuds en deux groupes :

```
[master]
127.0.0.1
[nodes]
192.168.33.11
192.168.33.12
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Cette configuration n'a pas de système de résolution de noms, alors référez-vous aux nœuds par leurs adresses IP. Il est également important que vous puissiez parler aux deux nœuds en utilisant le module Ansible ping avec la commande `ansible nodes -m ping`:

```
vagrant@master:~$ ansible nodes -m ping
192.168.33.11 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
192.168.33.12 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Les deux nœuds doivent répondre par un pong, qui vérifie que la connexion SSH fonctionne et que l'inventaire contient les bonnes adresses IP.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

A ce stade, vous pouvez continuer à [exécuter des commandes individuelles](#), mais pour vraiment apprendre Ansible, placez les commandes dans un playbook qui fonctionne à partir d'une seule invite.

Les playbooks sont écrits en YAML.

Le playbook pour pinger les nœuds est simple :

```
---  
- hosts: nodes  
  tasks:  
    - ping:
```

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Pour exécuter le playbook, utilisez la commande `ansible-playbook` et passez-lui le nom du fichier YAML :

```
vagrant@master:~$ ansible-playbook /vagrant/ping.yml

PLAY
*****

TASK [setup]
*****
ok: [192.168.33.12]
ok: [192.168.33.11]

TASK [ping]
*****
ok: [192.168.33.12]
ok: [192.168.33.11]

PLAY RECAP
*****
192.168.33.11      : ok=2    changed=0    unreachable=0    failed=0
192.168.33.12      : ok=2    changed=0    unreachable=0    failed=0
```


Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

Dans cet exemple, deux tâches ont été effectuées pour chaque nœud : la tâche d'installation est un test – il vérifie que l'utilisateur peut bien parler au nœud –, puis la tâche ping s'exécute.

La tâche ping est redondante, car la tâche d'installation vérifie qu'Ansible fonctionne. Les playbooks peuvent contenir des centaines de lignes de tâches.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps

■ Accéder à plus de contenu exclusif PRO+

Vous avez accès à cet e-Handbook en tant que membre via notre offre PRO+ : une collection de publications gratuites et offres spéciales rassemblées pour vous par nos partenaires et sur tout notre réseau de sites internet.

L'offre PRO+ est gratuite et réservée aux membres du réseau de sites internet TechTarget.

Profitez de tous les avantages liés à votre abonnement sur: <http://www.lemagit.fr/eproducts>

Images : Fotolia

©2018 TechTarget. Tout ou partie de cette publication ne peut être transmise ou reproduite dans quelque forme ou de quelque manière que ce soit sans autorisation écrite de la part de l'éditeur.

Dans ce guide

- Comment créer son cluster Kubernetes en local avec Minikube
- Comment bien se former à Kubernetes
- Comment utiliser Github dans son labo DevOps
- Git : 5 commandes basiques que l'on doit maîtriser
- Démarrer avec Ansible pour configurer son labo DevOps



Le document consulté provient du site www.lemagit.fr

Cyrille Chausson | *Rédacteur en Chef*
TechTarget
22 rue Léon Jouhaux, 75010 Paris
www.techtarget.com

©2018 TechTarget Inc. Aucun des contenus ne peut être transmis ou reproduit quelle que soit la forme sans l'autorisation écrite de l'éditeur. Les réimpressions de TechTarget sont disponibles à travers The [YGS Group](#).

TechTarget édite des publications pour les professionnels de l'IT. Plus de 100 sites qui proposent un accès rapide à un stock important d'informations, de conseils, d'analyses concernant les technologies, les produits et les process déterminants dans vos fonctions. Nos événements réels et nos séminaires virtuels vous donnent accès à des commentaires et recommandations neutres par des experts sur les problèmes et défis que vous rencontrez quotidiennement. Notre communauté en ligne "IT Knowledge Exchange" (Echange de connaissances IT) vous permet de partager des questionnements et informations de tous les jours avec vos pairs et des experts du secteur.