# **Rapport de Projet 3 : Script Simple de Déploiement de Machines Virtuelles Informations Générales**

Noms:Safae Mamouni Alaoui,Sara Zahidi

**Objectifs du Projet**

## Ce projet avait pour but de :

* Automatiser le déploiement basique de machines virtuelles (VMs) via script.
* Se familiariser avec les outils VMware en ligne de commande (PowerCLI).
* Créer un script modulaire, réutilisable, clair et gérant les erreurs.
* Apprendre les bases d’une infrastructure automatisée.

## **️ Technologies et Outils Utilisés**

|  |  |
| --- | --- |
| **Élément** | **Description** |
| **VMware vCenter** | Plateforme de gestion centralisée des hôtes et VMs |
| **VMware ESXi** | Hyperviseur sur lequel les VMs sont déployées |
| **PowerShell** | Shell d’automatisation Windows |
| **PowerCLI** | Module PowerShell pour VMware |
| **Fichier de configuration** | Format JSON ou dictionnaire PowerShell pour les paramètres |

## **🧠 Architecture du Script**

Le script est divisé en **trois fonctions principales** et un **bloc principal d’exécution**.

### **1. Connect-ToVCenter**

Cette fonction établit une connexion sécurisée avec le serveur vCenter :

powershell

CopyEdit

Connect-VIServer -Server $vcenter -User $username -Password $password

Gestion des erreurs intégrée pour éviter les plantages du script en cas d’échec.

### **2. Get-TemplateVM**

Permet de vérifier l’existence d’un template de VM et de le récupérer :

powershell

CopyEdit

$template = Get-Template -Name $templateName -ErrorAction SilentlyContinue

Affiche une erreur et stoppe le script si le template est introuvable.

### **3. Clone-VM**

Effectue le déploiement complet :

* Sélection d’un hôte ESXi aléatoire
* Récupération d’un pool de ressources
* Clonage de la VM depuis le template
* Attribution des ressources (CPU, RAM)
* Démarrage de la VM

powershell

CopyEdit

New-VM -Name $config.vm\_name -Template $template ...  
Set-VM -VM $vm -MemoryMB $config.memory -NumCpu $config.cpu ...  
Start-VM -VM $vm

## **⚙️ Paramètres de Déploiement Utilisés**

Le script utilise un objet $config contenant les informations nécessaires à la configuration de la VM :

powershell

CopyEdit

$config = @{  
 vm\_name = "VM-Demo01"  
 template = "TemplateUbuntu"  
 cluster = "Cluster-Prod"  
 datastore = "Datastore1"  
 network = "VM Network"  
 cpu = 2  
 memory = 4096  
}

## **🔐 Connexion à vCenter**

Le script utilise les identifiants suivants (à sécuriser dans une version future) :

powershell

CopyEdit

$vcenter = "vcenter.local"  
$username = "[administrator@vsphere.local](mailto:administrator@vsphere.local)"  
$password = "password"

# **P**

## **1. 🎯 Objectifs pédagogiques**

Ce projet a pour objectif de :

* Comprendre les composants de base d’un cluster Kubernetes
* Automatiser l’installation d’un cluster simple à l’aide de kubeadm
* Apprendre les concepts fondamentaux : nodes, pods, services, CNI, dashboard
* Déployer une application de test sur le cluster

## **2. 🗂️ Structure du projet**

arduino

CopyEdit

k8s-simple-install/  
├── scripts/  
│ ├── install-master.sh  
│ ├── install-worker.sh  
│ ├── common-setup.sh  
│ └── deploy-test-app.sh  
├── config/  
│ ├── hosts.txt  
│ └── cluster-config.yaml  
├── manifests/  
│ ├── dashboard.yaml  
│ └── test-app.yaml  
└── docs/  
 └── installation-guide.md

## **3 📜 Scripts principaux**

### **common-setup.sh (exécuté sur toutes les machines)**

bash

CopyEdit

sudo apt-get update  
sudo apt-get install -y apt-transport-https curl gnupg lsb-release  
curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg  
echo "deb [arch=amd64] <https://download.docker.com/linux/ubuntu> focal stable" | sudo tee /etc/apt/sources.list.d/docker.list  
sudo apt-get update  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io  
sudo systemctl enable docker && sudo systemctl start docker  
sudo apt-get install -y kubelet kubeadm kubectl  
sudo apt-mark hold kubelet kubeadm kubectl

### **install-master.sh**

bash

CopyEdit

bash ./common-setup.sh  
sudo kubeadm init --pod-network-cidr=10.244.0.0/16  
  
# Configuration kubeconfig  
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config  
  
# Installation de Flannel  
kubectl apply -f <https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml># Génération du token de jointure  
kubeadm token create --print-join-command > join.sh

### **install-worker.sh**

bash

CopyEdit

bash ./common-setup.sh  
bash ./join.sh # récupéré depuis le master

### **deploy-test-app.sh**

bash

CopyEdit

kubectl apply -f ../manifests/test-app.yaml  
kubectl apply -f ../manifests/dashboard.yaml

## **4 🚀 Déploiement de l'application test**

* **Application utilisée** : Serveur web Nginx simple
* **Type d’exposition** : Service de type ClusterIP ou NodePort
* **Dashboard** : Accessible via proxy avec kubectl proxy

## **5. ✅ Commandes de validation**

bash

CopyEdit

kubectl get nodes # Vérifier que les nœuds sont bien en Ready  
kubectl get pods --all-namespaces # Vérifier les pods système  
kubectl get svc # Vérifier le service de l’application  
kubectl proxy # Accès au dashboard via localhost

## **6. 📌 Résultats obtenus**

* Cluster Kubernetes simple opérationnel sur 3 machines
* Communication réseau fonctionnelle avec Flannel
* Application Nginx déployée avec succès
* Dashboard accessible pour visualisation