Cluster Kubernetes automatisé avec Vagrant + Ansible sur Debian 13 (Mac & Windows)

Déploiement complet d'un cluster Kubernetes 1 master + 2 workers depuis zéro avec Vagrant (VMware Fusion/Workstation) et Ansible (Debian 13, K8s v1.36, Flannel). Procédure Mac + Windows/WSL, tous les fichiers de config, pièges classiques et solutions.

45 min de lecture **Niveau Avancé**

Document généré le 11/07/2026 à 20h35 · nouvy.fr/wiki/cluster-kubernetes-vagrant-ansible-debian-13

Sommaire

56 section(s) · 45 min de lecture

Ce wiki décrit la mise en place complète d'un **cluster Kubernetes v1.36** avec **1 master + 2 workers** sur des VMs Debian 13, orchestré par **Vagrant** (provisioning VM) et **Ansible** (configuration + bootstrap K8s). Compatible **macOS** et **Windows/WSL**.

---

1. Architecture cible

```
Ton hôte (Mac ou Windows)
├─ VMware Fusion (Mac) / VMware Workstation (Windows)
├─ Vagrant                      → orchestre les VMs
├─ Ansible (Mac natif / WSL)    → configure les VMs
│
└─ 3 VMs Debian 13 (192.168.194.0/24)
   ├─ master1   192.168.194.10   4 Go RAM, 2 vCPU  (control plane)
   ├─ worker1   192.168.194.11   2 Go RAM, 2 vCPU
   └─ worker2   192.168.194.12   2 Go RAM, 2 vCPU
```

Copier

Rôle IaC :

- **Vagrant** = lifecycle VM *(create/destroy/halt)* — équivalent Terraform pour local
- **Ansible** = configuration OS + bootstrap Kubernetes *(containerd, kubeadm, Flannel, join)*

---

2. Installation sur macOS

2.1 VMware Fusion (gratuit)

Depuis mai 2024, **Fusion Pro est gratuit** pour usage personnel :

```
brew install --cask vmware-fusion
```

Copier

Au 1er lancement : accepter la **Personal Use License**.

2.2 Vagrant

```
brew install --cask vagrant
vagrant --version
```

Copier

2.3 Plugin VMware pour Vagrant

Le plugin est **payant en usage pro** *(~80 $)*, période d'essai gratuite.

```
brew install --cask vagrant-vmware-utility
vagrant plugin install vagrant-vmware-desktop
vagrant plugin list
```

Copier

2.4 Ansible

```
brew install ansible
ansible --version
```

Copier

---

3. Installation sur Windows

3.1 VMware Workstation Pro (gratuit)

Gratuit depuis mai 2024 :

```
winget install VMware.WorkstationPro
```

Copier

⚠ **Désactiver Hyper-V** *(sinon VMware plante)* :

```
Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-All
bcdedit /set hypervisorlaunchtype off
# reboot
```

Copier

3.2 Vagrant

```
winget install Hashicorp.Vagrant
```

Copier

3.3 Plugin VMware Vagrant

- Télécharger `vagrant-vmware-utility-windows-*.msi` : https://developer.hashicorp.com/vagrant/install/vmware
- Installer, puis :

```
vagrant plugin install vagrant-vmware-desktop
```

Copier

3.4 Ansible via WSL2 (obligatoire — Ansible ne tourne pas sur Windows natif)

```
wsl --install
# reboot
```

Copier

Au reboot, Ubuntu s'ouvre. Créer utilisateur/mot de passe, puis dans Ubuntu WSL :

```
sudo apt update && sudo apt install -y ansible
ansible --version
```

Copier

Activer le **networking mirrored** pour que WSL voie les VMs VMware — créer `%USERPROFILE%\.wslconfig` :

```
[wsl2]
networkingMode=mirrored
```

Copier

Puis redémarrer WSL :

```
wsl --shutdown
wsl
```

Copier

---

4. Structure du projet

Arborescence finale à créer :

```
vagrant-cluster/
├─ Vagrantfile
├─ .vagrant/                    (auto-créé par Vagrant)
└─ ansible/
   ├─ ansible.cfg
   ├─ inventory.ini
   ├─ site.yml
   ├─ group_vars/
   │  └─ all.yml
   └─ roles/
      ├─ common/
      │  ├─ tasks/main.yml
      │  └─ handlers/main.yml
      ├─ master/
      │  └─ tasks/main.yml
      └─ worker/
         └─ tasks/main.yml
```

Copier

Créer l'arborescence

Mac :

```
mkdir -p ~/Downloads/vagrant-cluster/ansible/group_vars
mkdir -p ~/Downloads/vagrant-cluster/ansible/roles/common/tasks
mkdir -p ~/Downloads/vagrant-cluster/ansible/roles/common/handlers
mkdir -p ~/Downloads/vagrant-cluster/ansible/roles/master/tasks
mkdir -p ~/Downloads/vagrant-cluster/ansible/roles/worker/tasks
cd ~/Downloads/vagrant-cluster
```

Copier

Windows WSL :

```
mkdir -p /mnt/c/Users/$USER/Downloads/vagrant-
cluster/ansible/{group_vars,roles/{common/{tasks,handlers},master/tasks,worker/tasks}}
cd /mnt/c/Users/$USER/Downloads/vagrant-cluster
```

Copier

---

5. Fichier `Vagrantfile`

⚠ IMPORTANT — Adapter le subnet à ta config VMware

Le subnet `192.168.194.0/24` utilisé dans ce wiki est **celui de vmnet8 (NAT) sur l'installation VMware de l'auteur**. Chaque installation VMware attribue un subnet **différent et aléatoire** à vmnet8 à l'installation. **Il faut absolument le remplacer par le tien**.

Comment trouver ton subnet NAT VMware

Sur Mac (Fusion) :

1. Ouvrir **VMware Fusion**
2. Menu **VMware Fusion → Réglages...** *(Cmd + ,)*
3. Onglet **Réseau**
4. Cliquer sur **"Partager...on Mac"** *(vmnet8 — c'est la NAT par défaut, celle avec l'icône partage internet)*
5. Champ **IP de sous-réseau** : par exemple `192.168.194.0`, `192.168.85.0`, `192.168.132.0`... **c'est cette valeur qu'il faut noter**

Ou en ligne de commande :

```
grep -A 3 "VNET_8" "/Library/Preferences/VMware Fusion/networking" | grep SUBNET
# → answer VNET_8_HOSTONLY_SUBNET 192.168.194.0
```

Copier

Sur Windows (Workstation) :

1. Ouvrir **VMware Workstation Pro**

2. Menu **Edit → Virtual Network Editor**
3. Sélectionner **VMnet8 (NAT)**
4. Champ **Subnet IP** : par exemple `192.168.85.0`, `192.168.244.0`...

Vérifier depuis une VM déjà créée

Si tu as déjà lancé une VM sans IP fixe, tu peux voir l'IP DHCP obtenue :

```
vagrant ssh master1 -c "ip -4 addr show | grep 192.168"
# → inet 192.168.194.197/24 brd 192.168.194.255 scope global dynamic ...
```

Copier

L'IP commence par `192.168.194` (dans cet exemple) → **ton subnet est `192.168.194.0/24`**.

Adapter les IPs

Une fois le subnet identifié, remplace **partout** dans les fichiers `Vagrantfile`, `inventory.ini` et `group_vars/all.yml` les 3 IPs par des IPs **hors de la plage DHCP** VMware *(qui commence généralement à `.128`)*. La convention est de prendre **`.10`, `.11`, `.12`** :

Ton subnet	master1	worker1	worker2
192.168.194.0/24	192.168.194.10	192.168.194.11	192.168.194.12
192.168.85.0/24	192.168.85.10	192.168.85.11	192.168.85.12
192.168.132.0/24	192.168.132.10	192.168.132.11	192.168.132.12
192.168.244.0/24	192.168.244.10	192.168.244.11	192.168.244.12

Astuce recherche/remplacement dans les fichiers :

```
# Mac / Linux / WSL — remplace 194 par ton subnet (ex. 85)
sed -i "" "s/192\.168\.194\./192.168.85./g" Vagrantfile ansible/inventory.ini
ansible/group_vars/all.yml
```

Copier

(Sous Linux/WSL, `sed -i` sans les guillemets vides.)

Le Vagrantfile

Vagrant décrit les 3 VMs Debian 13. Le script shell inline **assigne une IP fixe** sur `eth0` (interface NAT vmnet8) pour que le Mac/Windows puisse joindre les VMs sur des IPs stables **sans reconfiguration Fusion** :

```
Vagrant.configure("2") do |config|
    config.vm.box = "bento/debian-13"
    nodes = [
        { name: "master1", ip: "192.168.194.10" },
        { name: "worker1", ip: "192.168.194.11" },
        { name: "worker2", ip: "192.168.194.12" }
    ]

    nodes.each do |node|
        config.vm.define node[:name] do |node_config|
            node_config.vm.hostname = node[:name]

            node_config.vm.provider "vmware_desktop" do |v|
                v.memory = node[:name] == "master1" ? 4096 : 2048
                v.cpus = 2
            end

            node_config.vm.provision "shell", inline: <<-SHELL
                IFACE=$(ip -o link show | awk -F": " "$2 ~ /^e(th|ns|np)/ && $2 !~ /lo/
{print $2; exit}")
                ip addr add #{node[:ip]}/24 dev $IFACE 2>/dev/null || true

                cat > /etc/systemd/system/static-alias.service <<EOF
[Unit]
Description=Assign static IP alias
After=network-online.target
Wants=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/sbin/ip addr add #{node[:ip]}/24 dev $IFACE
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
EOF
                systemctl daemon-reload
                systemctl enable static-alias.service
            SHELL
        end
    end
end
```

Copier

Points clés

- `bento/debian-13` — box Debian 13 officielle (Chef), publie pour VMware **et** VirtualBox
- `v.memory = node[:name] == "master1" ? 4096 : 2048` — **master = 4 Go** (obligatoire pour le control plane K8s 1.36), workers = 2 Go
- Pas de `private_network` — évite l'erreur "*Networks with custom subnet/mask values are not supported*" sur Fusion Apple Silicon
- Le script shell ajoute un **alias IP fixe** sur eth0 (vmnet8) + systemd service pour la persistance au reboot

---

6. Créer les 3 VMs

```
cd ~/Downloads/vagrant-cluster   # Mac
# ou :
cd /mnt/c/Users/$USER/Downloads/vagrant-cluster   # WSL

vagrant up --provider=vmware_desktop
```

Copier

Durée : ~5-8 min. Vagrant télécharge la box (une fois, ~500 Mo) puis crée les 3 VMs en parallèle.

Vérifier

```
vagrant status
# master1                   running (vmware_desktop)
# worker1                   running (vmware_desktop)
# worker2                   running (vmware_desktop)

ping 192.168.194.10
ping 192.168.194.11
ping 192.168.194.12
```

Copier

---

7. Fichiers Ansible

7.1 `ansible/ansible.cfg`

```
[defaults]
inventory = inventory.ini
host_key_checking = False
remote_user = vagrant
retry_files_enabled = False
stdout_callback = default
result_format = yaml

[ssh_connection]
pipelining = True
```

Copier

⚠ Depuis Ansible-core 2.13, `stdout_callback = yaml` a été **remplacé** par `stdout_callback = default` + `result_format = yaml`.

7.2 `ansible/inventory.ini`

```
[masters]
master1 ansible_host=192.168.194.10
ansible_ssh_private_key_file=../.vagrant/machines/master1/vmware_desktop/private_key

[workers]
worker1 ansible_host=192.168.194.11
ansible_ssh_private_key_file=../.vagrant/machines/worker1/vmware_desktop/private_key
worker2 ansible_host=192.168.194.12
ansible_ssh_private_key_file=../.vagrant/machines/worker2/vmware_desktop/private_key

[cluster:children]
masters
workers
```

Copier

Le méta-groupe `[cluster:children]` permet de cibler master + workers d'un coup dans le playbook.

7.3 `ansible/group_vars/all.yml`

⚠ **Attention aux fautes de frappe** : dossier `group_vars` *(pas `groupe_vars`)*, variable `k8s_version` *(sans "s")*.

```
k8s_version: "1.36"
master_ip: "192.168.194.10"
pod_network_cidr: "10.244.0.0/16"
```

Copier

7.4 `ansible/site.yml`

```
- name: Préparer toutes les VMs
  hosts: cluster
  become: yes
  roles:
    - common

- name: Configurer le MASTER
  hosts: masters
  become: yes
  roles:
    - master

- name: Joindre les WORKERS au cluster
  hosts: workers
  become: yes
  roles:
    - worker
```

Copier

7.5 `ansible/roles/common/tasks/main.yml`

Ce rôle est exécuté sur les 3 VMs. Il installe containerd, désactive swap, configure les modules kernel, installe kubelet/kubeadm/kubectl, et applique les 2 correctifs critiques :

sandbox_image pause:3.10.2 et **symlink CNI**.

```
- name: Forcer un DNS public (8.8.8.8 + 1.1.1.1)
  copy:
    dest: /etc/resolv.conf
    content: |
      nameserver 8.8.8.8
      nameserver 1.1.1.1
    force: yes

- name: Installer outils de base
  apt:
    name:
      - curl
      - gnupg2
      - apt-transport-https
      - ca-certificates
    update_cache: yes
    state: present

- name: Installer containerd
  apt:
    name: containerd
    state: present

- name: Créer /etc/containerd
  file:
    path: /etc/containerd
    state: directory

- name: Générer la config par défaut de containerd
  shell: containerd config default > /etc/containerd/config.toml
  args:
    creates: /etc/containerd/config.toml

- name: Activer SystemdCgroup dans containerd
  replace:
    path: /etc/containerd/config.toml
    regexp: 'SystemdCgroup = false'
    replace: 'SystemdCgroup = true'
  notify: restart containerd

- name: Fixer le sandbox_image (pause 3.10.2 compatible K8s 1.36)
  lineinfile:
    path: /etc/containerd/config.toml
    insertafter: '\[plugins\."io\.containerd\.grpc\.v1\.cri"\]'
    line: '    sandbox_image = "registry.k8s.io/pause:3.10.2"'
    state: present
  notify: restart containerd

- name: Démarrer et activer containerd
  service:
    name: containerd
    state: started
    enabled: yes

- name: Créer le symlink CNI (fix Flannel/CNI)
  file:
    src: /opt/cni/bin
    dest: /usr/lib/cni
    state: link
    force: yes

- name: Désactiver le swap immédiatement
  shell: swapoff -a
```

```
  when: ansible_swaptotal_mb > 0

- name: Désactiver le swap dans /etc/fstab
  replace:
    path: /etc/fstab
    regexp: '^([^#].*\sswap\s.*)$'
    replace: '# \1'

- name: Charger overlay + br_netfilter au boot
  copy:
    dest: /etc/modules-load.d/k8s.conf
    content: |
      overlay
      br_netfilter

- name: Charger les modules maintenant
  shell: "modprobe {{ item }}"
  loop:
    - overlay
    - br_netfilter

- name: Paramètres sysctl Kubernetes
  copy:
    dest: /etc/sysctl.d/k8s.conf
    content: |
      net.bridge.bridge-nf-call-iptables  = 1
      net.bridge.bridge-nf-call-ip6tables = 1
      net.ipv4.ip_forward                 = 1

- name: Appliquer les sysctl
  shell: sysctl --system

- name: Créer /etc/apt/keyrings
  file:
    path: /etc/apt/keyrings
    state: directory
    mode: '0755'

- name: Télécharger la clé GPG Kubernetes v{{ k8s_version }}
  shell: |
    curl -fsSL https://pkgs.k8s.io/core:/stable:/v{{ k8s_version }}/deb/Release.key
      | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
  args:
    creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg

- name: Ajouter le dépôt Kubernetes v{{ k8s_version }}
  copy:
    dest: /etc/apt/sources.list.d/kubernetes.list
    content: |
      deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v{{ k8s_version }}/deb/ /

- name: Installer kubelet, kubeadm, kubectl
  apt:
    name:
      - kubelet
      - kubeadm
      - kubectl
    update_cache: yes
    state: present

- name: Hold sur les paquets Kubernetes
  shell: apt-mark hold kubelet kubeadm kubectl
```

Copier

7.6 `ansible/roles/common/handlers/main.yml`

```
- name: restart containerd
  service:
    name: containerd
    state: restarted
```

Copier

7.7 `ansible/roles/master/tasks/main.yml`

```
- name: Initialiser le cluster (kubeadm init)
  shell: |
    kubeadm init
      --apiserver-advertise-address={{ master_ip }}
      --pod-network-cidr={{ pod_network_cidr }}
  args:
    creates: /etc/kubernetes/admin.conf

- name: Créer ~/.kube pour vagrant
  file:
    path: /home/vagrant/.kube
    state: directory
    owner: vagrant
    group: vagrant

- name: Copier admin.conf pour l'utilisateur vagrant
  copy:
    src: /etc/kubernetes/admin.conf
    dest: /home/vagrant/.kube/config
    owner: vagrant
    group: vagrant
    remote_src: yes

- name: Installer Flannel (CNI) v0.28.5 (URL release, pas master)
  become_user: vagrant
  shell: |
    kubectl apply -f
https://github.com/flannel-io/flannel/releases/download/v0.28.5/kube-flannel.yml
  args:
    creates: /home/vagrant/.flannel-installed

- name: Marquer Flannel installé
  file:
    path: /home/vagrant/.flannel-installed
    state: touch

- name: Générer la commande de join pour les workers
  shell: kubeadm token create --print-join-command
  register: join_command
  changed_when: false

- name: Publier la commande de join en mémoire Ansible
  add_host:
    name: JOIN_HOST
    join_cmd: "{{ join_command.stdout }}"
```

Copier

⚠ **Ne jamais utiliser** `.../flannel/master/Documentation/kube-flannel.yml` — la branche `master` est en dev et incompatible avec les images publiées.

7.8 `ansible/roles/worker/tasks/main.yml`

```
- name: Rejoindre le cluster
  shell: "{{ hostvars['JOIN_HOST']['join_cmd'] }}"
  args:
    creates: /etc/kubernetes/kubelet.conf
```

Copier

---

8. Déploiement du cluster

Tester la connexion Ansible

```
cd ansible
ansible cluster -m ping
```

Copier

Attendu :

```
master1 | SUCCESS => { "ping": "pong" }
worker1 | SUCCESS => { "ping": "pong" }
worker2 | SUCCESS => { "ping": "pong" }
```

Copier

Vérifier que la variable est chargée

```
ansible cluster -m debug -a "var=k8s_version"
```

Copier

Attendu :

```
master1 | SUCCESS => { "k8s_version": "1.36" }
```

Copier

Lancer le playbook complet

```
ansible-playbook site.yml
```

Copier

Durée : 10-15 min. Ansible :

1. Prépare les 3 VMs *(containerd, kubelet, kubeadm)*

2. Fait `kubeadm init` sur master1
3. Installe Flannel
4. Génère un token de join
5. Fait rejoindre worker1 et worker2

---

9. Vérification

```
cd ..
vagrant ssh master1 -c "kubectl get nodes -o wide"
```

Copier

Attendu :

```
NAME      STATUS   ROLES           AGE     VERSION   INTERNAL-IP
master1   Ready    control-plane   3m      v1.36.2   192.168.194.10
worker1   Ready    <none>          2m      v1.36.2   192.168.194.11
worker2   Ready    <none>          2m      v1.36.2   192.168.194.12
```

Copier

Voir tous les pods :

```
vagrant ssh master1 -c "kubectl get pods -A"
```

Copier

Doit lister *(tous en `Running`)* :

- `coredns-*` × 2
- `etcd-master1`
- `kube-apiserver-master1`
- `kube-controller-manager-master1`
- `kube-scheduler-master1`
- `kube-proxy-*` × 3
- `kube-flannel-ds-*` × 3

Test end-to-end avec nginx :

```
vagrant ssh master1 -c "kubectl create deployment nginx --image=nginx && kubectl expose
deployment nginx --port=80 --type=NodePort && sleep 5 && kubectl get pods,svc"
```

Copier

---

10. Importer dans Lens

Récupérer le kubeconfig :

```
vagrant ssh master1 -c "sudo cat /etc/kubernetes/admin.conf"
```

Copier

Puis dans **Lens** :

1. **File → Add Cluster → From kubeconfig**
2. Coller le contenu
3. **Add cluster**

Le cluster apparaît dans la sidebar avec les 3 nodes.

---

11. Ajouter un worker (~8 minutes)

Étape 1 — Ajouter dans le `Vagrantfile`

```
nodes = [
    { name: "master1", ip: "192.168.194.10" },
    { name: "worker1", ip: "192.168.194.11" },
    { name: "worker2", ip: "192.168.194.12" },
    { name: "worker3", ip: "192.168.194.13" }   # nouveau
]
```

Copier

Étape 2 — Créer uniquement la nouvelle VM

```
vagrant up worker3
```

Copier

Étape 3 — Ajouter dans `inventory.ini`

```
[workers]
worker1 ansible_host=192.168.194.11
ansible_ssh_private_key_file=../.vagrant/machines/worker1/vmware_desktop/private_key
worker2 ansible_host=192.168.194.12
ansible_ssh_private_key_file=../.vagrant/machines/worker2/vmware_desktop/private_key
worker3 ansible_host=192.168.194.13
ansible_ssh_private_key_file=../.vagrant/machines/worker3/vmware_desktop/private_key
```

Copier

Étape 4 — Relancer

```
cd ansible
ansible-playbook site.yml
```

Copier

Ansible skippera tout ce qui est déjà fait sur les autres nodes *(idempotence via `creates:`)* et n'installera que sur worker3. Total ~5 min.

---

12. Pièges classiques et solutions

12.1 Erreur `Networks with custom subnet/mask values are not supported`

Sur Fusion Apple Silicon, Vagrant ne peut pas créer un `private_network` sur un subnet inconnu. **Solution** : ne pas utiliser `private_network`, mais un alias IP fixe posé par un shell provisioner sur eth0 (vmnet8), comme dans le Vagrantfile.

12.2 kubeadm init : `192.168.164.10:6443 connection refused`

L'IP du master dans `group_vars/all.yml` ne correspond pas à celle de la VM. **Vérifier** que `master_ip` = IP réelle assignée par le Vagrantfile.

12.3 Erreur DNS : `lookup dl.k8s.io on 192.168.194.2:53: no such host`

Le DNS NAT VMware ne relaie pas certaines requêtes. **Solution** : forcer 8.8.8.8 dans `/etc/resolv.conf` via la 1ère tâche du rôle common.

12.4 Flannel : `stat /opt/bin/install-conf: no such file or directory`

Utilisation de la branche `master` du manifest Flannel qui référence un binaire non présent dans l'image `v0.28.5`. **Solution** : utiliser l'URL de release taggée `https://github.com/flannel-io/flannel/releases/download/v0.28.5/kube-flannel.yml`.

12.5 Pods système en `CrashLoopBackOff` (scheduler, controller-manager, kube-proxy)

Deux causes cumulées :

- **Sandbox image** `pause:3.8` de containerd 1.7.24 incompatible avec K8s 1.36 (attend `pause:3.10.2`). Fix : ajouter `sandbox_image = "registry.k8s.io/pause:3.10.2"` dans `/etc/containerd/config.toml`.
- **Master avec 2 Go RAM** → OOM. Fix : passer master à 4 Go dans le Vagrantfile.

12.6 Nodes en `NotReady` : `cni plugin not initialized`

containerd cherche les binaires CNI dans `/usr/lib/cni` mais ils sont dans `/opt/cni/bin`. **Solution** : `ln -s /opt/cni/bin /usr/lib/cni` (tâche du rôle common).

12.7 `k8s_version is undefined`

Deux erreurs classiques :

- Dossier nommé `groupe_vars` au lieu de `group_vars` (typo francophone).
- Variable écrite `k8s_versions` (avec S) au lieu de `k8s_version`.

12.8 `stdout_callback = yaml` : plugin removed

Le plugin `community.general.yaml` est supprimé depuis `community.general 12.0.0`. **Solution** : dans `ansible.cfg`, utiliser `stdout_callback = default` + `result_format = yaml`.

12.9 SSH : `Could not resolve hostname ansible_ssh_private_key_file=...`

Ligne d'hôte cassée sur plusieurs lignes dans `inventory.ini`. **Chaque hôte doit être sur UNE seule ligne**.

12.10 SSH port 22 : timeout depuis le Mac

Le subnet privé n'est pas exposé côté hôte. **Solution** : le Vagrantfile de ce wiki utilise vmnet8 (NAT défaut) sur lequel le Mac a déjà une interface (192.168.194.1) → accès direct sans reconfiguration Fusion.

---

13. Commandes utiles

Vagrant

```
vagrant up                              # démarre les 3 VMs
vagrant status                          # état
vagrant ssh master1                     # SSH direct
vagrant halt                            # éteint proprement
vagrant destroy -f                      # supprime complètement
vagrant reload master1                  # reboot d'une VM (après modif config)
```

Copier

Ansible

```
ansible cluster -m ping                 # test connexion
ansible cluster -m debug -a "var=k8s_version"    # debug variable
ansible-playbook site.yml --check       # dry-run
ansible-playbook site.yml               # exécution
ansible-playbook site.yml --limit masters       # cible masters uniquement
ansible-playbook site.yml -vvv          # mode très verbeux
```

Copier

Kubernetes (depuis master1)

```
kubectl get nodes                       # état des nodes
kubectl get pods -A                     # tous les pods
kubectl get svc -A                      # tous les services
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80 --type=NodePort
kubectl delete node worker3             # retirer un worker
```

Copier

Reset complet en cas de problème

```
# Reset K8s sur les 3 VMs
for vm in master1 worker1 worker2; do
  vagrant ssh $vm -c "sudo kubeadm reset -f && sudo rm -rf /etc/kubernetes /var/lib/etcd
/home/vagrant/.kube"
done

# Puis relancer le playbook
cd ansible
ansible-playbook site.yml
```

Copier

---

14. Récapitulatif

Composant	Version / Rôle
OS VM	Debian 13 (bento/debian-13)
Runtime container	containerd (Debian 13 default)
K8s	v1.36.2
CNI	Flannel v0.28.5
VM	1 master (4 Go) + 2 workers (2 Go)
Réseau	vmnet8 NAT + alias IP fixe (192.168.194.10-12)
Provisioning	Vagrant + VMware Desktop
Config	Ansible (rôles common, master, worker)

Résultat : un cluster Kubernetes 1 master + 2 workers reproductible en **1 commande** (`vagrant up && ansible-playbook site.yml`), de zéro à opérationnel en ~15 min, entièrement versionnable dans Git.

Pour l'ajouter, retirer un node ou reset : 1 modification de fichier + `ansible-playbook site.yml`. C'est le workflow IaC standard 2026.