Documentation Technique Complète - Infrastructure MLOps Ansible

Table des Matières

Structure du projet

Architecture complète du dossier ansible

```
/home/wassim/pfe/mlops/ansible/
  ansible.cfg                       # Configuration principale Ansible
  inventories/
     hosts.ini                      # Inventaire des 7 serveurs
  group_vars/
     all.yml                        # Variables pour tous les groupes
     haproxy.yml                    # Variables spécifiques HAProxy
     k8s-cluster.yml                # Variables cluster Kubernetes
  playbooks/
     site.yml                       # Playbook principal orchestrateur
     install-metallb.yml            # Installation MetalLB
     install-prometheus.yml         # Installation stack monitoring
  roles/
      common/
         tasks/
             main.yml               # Tâches communes tous serveurs
      haproxy/
         handlers/
            main.yml                # Handlers restart/reload HAProxy
         tasks/
            main.yml                # Installation et configuration HAProxy
         templates/
             haproxy.cfg.j2         # Template configuration HAProxy
      kubernetes-master/
         tasks/
             main.yml               # Configuration masters Kubernetes
      kubernetes-worker/
```

```
        tasks/
            main.yml                # Configuration workers Kubernetes
    metallb/
        tasks/
            main.yml                # Installation MetalLB
```

Fichiers de configuration

ansible.cfg

```
[defaults]
# Configuration Ansible principale
inventory = inventories/hosts.ini
remote_user = user
private_key_file = ~/.ssh/id_rsa
host_key_checking = False
gather_facts = True
timeout = 30
retry_files_enabled = False

# Optimisation performances
forks = 10
gathering = smart
fact_caching = memory
fact_caching_timeout = 86400

# Logging
log_path = /tmp/ansible.log

[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=no
pipelining = True
control_path = /tmp/ansible-ssh-%%h-%%p-%%r
```

Explications :

- `inventory` : Pointe vers le fichier d'inventaire des serveurs
- `remote_user` : Utilisateur pour connexions SSH (user)
- `host_key_checking = False` : Désactive vérification clés SSH
- `forks = 10` : Exécution parallèle sur 10 hôtes maximum
- `gathering = smart` : Collecte intelligente des facts
- `pipelining = True` : Optimisation SSH

Variables globales

group_vars/all.yml

```
---
# Variables globales pour tous les groupes d'hôtes

# Configuration utilisateur système
ansible_user: user
ansible_ssh_common_args: '-o StrictHostKeyChecking=no'

# Configuration réseau cluster
pod_network_cidr: "10.244.0.0/16"          # Réseau pods Flannel
service_cidr: "10.96.0.0/12"               # Réseau services Kubernetes
cluster_dns: "10.96.0.10"                  # IP DNS cluster

# Versions logicielles
kubernetes_version: "1.31.4"
containerd_version: "1.7.27"
flannel_version: "latest"

# Configuration monitoring
monitoring_namespace: "monitoring"
prometheus_nodeport: 30900
grafana_nodeport: 30300
alertmanager_nodeport: 30903

# Configuration ingress
ingress_namespace: "ingress-nginx"
ingress_http_nodeport: 32624
ingress_https_nodeport: 31316

# Paramètres système
disable_swap: true
enable_ipv4_forward: true
bridge_nf_call_iptables: true
```

Explications :

- `pod_network_cidr` : Plage IP pour les pods (compatible Flannel)
- `service_cidr` : Plage IP pour les services Kubernetes
- `kubernetes_version` : Version stable LTS utilisée
- `*_nodeport` : Ports d'accès externe aux services

group_vars/haproxy.yml

```
---
# Variables spécifiques au load balancer HAProxy
```

```yaml
# Configuration HAProxy
haproxy_stats_enabled: true
haproxy_stats_port: 8404
haproxy_stats_uri: "/stats"
haproxy_stats_user: "admin"
haproxy_stats_password: "admin123"

# Backends Kubernetes API
k8s_api_backend_port: 6443
k8s_api_backends:
  - name: "master-1"
    address: "10.110.190.22"
    port: 6443
    check: "check"
  - name: "master-2"
    address: "10.110.190.23"
    port: 6443
    check: "check"
  - name: "master-3"
    address: "10.110.190.24"
    port: 6443
    check: "check"

# Backends Ingress HTTP
ingress_http_backends:
  - name: "worker-1"
    address: "10.110.190.25"
    port: 32624
    check: "check"
  - name: "worker-2"
    address: "10.110.190.26"
    port: 32624
    check: "check"
  - name: "worker-3"
    address: "10.110.190.27"
    port: 32624
    check: "check"

# Backends Ingress HTTPS
ingress_https_backends:
  - name: "worker-1"
    address: "10.110.190.25"
    port: 31316
    check: "check"
  - name: "worker-2"
```

```yaml
    address: "10.110.190.26"
    port: 31316
    check: "check"
  - name: "worker-3"
    address: "10.110.190.27"
    port: 31316
    check: "check"

# Timeouts HAProxy
haproxy_connect_timeout: "4000ms"
haproxy_client_timeout: "50000ms"
haproxy_server_timeout: "50000ms"
```

Explications :

- `haproxy_stats_*` : Configuration interface de statistiques
- `k8s_api_backends` : Liste des masters pour équilibrage API
- `ingress_*_backends` : Liste des workers pour trafic Ingress
- Timeouts optimisés pour environnement production

group__vars/k8s-cluster.yml

```yaml
---
# Variables cluster Kubernetes

# Identification cluster
k8s_cluster_name: "mlops-cluster"
k8s_version: "1.31.4"
k8s_api_server_port: 6443
k8s_node_port_range: "30000-32767"

# Point d'entrée control plane (HAProxy)
k8s_control_plane_endpoint: "10.110.190.21:6443"

# Runtime conteneurs
container_runtime: containerd
container_runtime_version: "1.7.27"

# Plugin réseau CNI
cni_plugin: flannel
flannel_version: "latest"

# Contrôleur Ingress
ingress_controller: nginx
ingress_http_nodeport: 32624
ingress_https_nodeport: 31316
```

```yaml
# Packages Kubernetes avec versions pinned
kubernetes_packages:
  - kubelet=1.31.4-1.1
  - kubeadm=1.31.4-1.1
  - kubectl=1.31.4-1.1

# Configuration cluster kubeadm
cluster_configuration:
  apiVersion: kubeadm.k8s.io/v1beta3
  kind: ClusterConfiguration
  kubernetesVersion: "1.31.4"
  controlPlaneEndpoint: "10.110.190.21:6443"
  networking:
    podSubnet: "10.244.0.0/16"
    serviceSubnet: "10.96.0.0/12"

# Spécifications nœuds
master_node_count: 3
worker_node_count: 3

# Configuration services
kube_dns_ip: "10.96.0.10"
cluster_dns_domain: "cluster.local"

# Configuration système
os_distribution: "Ubuntu"
os_version: "24.04.2"
kernel_version: "6.8.0-53-generic"
architecture: "x86_64"

# État cluster actuel (documenté)
cluster_status: "running"
cluster_nodes_total: 6
master_nodes_count: 3
worker_nodes_count: 3
monitoring_coverage: "100%"
monitoring_targets_total: 75
monitoring_targets_up: 75

# Configuration containerd
containerd_config_file: "/etc/containerd/config.toml"
containerd_systemd_cgroup: true
containerd_runtime_type: "io.containerd.runc.v2"

# Configuration Flannel CNI
flannel_namespace: "kube-flannel"
```

```yaml
flannel_configmap: "kube-flannel-cfg"
flannel_backend_type: "vxlan"
flannel_enable_nftables: false
flannel_mtu: 1450

# Configuration Ingress NGINX
ingress_namespace: "ingress-nginx"
ingress_controller_name: "ingress-nginx-controller"
ingress_service_type: "NodePort"
ingress_class_name: "nginx"

# Gestion certificats
cert_manager_namespace: "cert-manager"
enable_cert_manager: false

# Monitoring et logging
enable_metrics_server: true
enable_kubernetes_dashboard: false

# Politiques réseau
enable_network_policies: false
default_deny_all: false

# Configuration stockage
default_storage_class: "local-path"
enable_persistent_volumes: true

# Configuration sécurité
enable_pod_security_policy: false
enable_rbac: true

# Configuration API Server
api_server_port: 6443
api_server_secure_port: 6443
api_server_insecure_port: 0

# Scheduler et Controller Manager
scheduler_bind_address: "0.0.0.0"
controller_manager_bind_address: "0.0.0.0"

# Configuration etcd
etcd_data_dir: "/var/lib/etcd"
etcd_listen_client_urls: "https://127.0.0.1:2379"
etcd_listen_peer_urls: "https://127.0.0.1:2380"
```

Explications :

- `k8s_control_plane_endpoint` : Point d'entrée via HAProxy
- `kubernetes_packages` : Versions pinned pour stabilité
- `cluster_configuration` : Configuration kubeadm complète
- Paramètres réseau optimisés pour production

Inventaire des serveurs

inventories/hosts.ini

```ini
# Inventaire infrastructure MLOps
# Réseau: 10.110.190.0/24

[haproxy]
haproxy-1 ansible_host=10.110.190.21 ansible_user=user hostname=haproxy

[k8s-masters]
master-1 ansible_host=10.110.190.22 ansible_user=user hostname=k8s-master-1
master-2 ansible_host=10.110.190.23 ansible_user=user hostname=k8s-master-2
master-3 ansible_host=10.110.190.24 ansible_user=user hostname=k8s-master-3

[k8s-workers]
worker-1 ansible_host=10.110.190.25 ansible_user=user hostname=k8s-worker-1
worker-2 ansible_host=10.110.190.26 ansible_user=user hostname=k8s-worker-2
worker-3 ansible_host=10.110.190.27 ansible_user=user hostname=k8s-worker-3

[k8s-cluster:children]
k8s-masters
k8s-workers

[all:vars]
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Structure :

- **1 HAProxy** : Load balancer externe (10.110.190.21)
- **3 Masters** : Control plane haute disponibilité (10.110.190.22-24)
- **3 Workers** : Nœuds de traitement (10.110.190.25-27)
- **Groupe k8s-cluster** : Union masters + workers

Playbooks principaux

playbooks/site.yml

```yaml
---
# Playbook principal orchestrateur infrastructure MLOps
```

```yaml
- name: Configure HAProxy Load Balancer
  hosts: haproxy
  become: true
  roles:
    - haproxy
  tags: haproxy

- name: Configure Kubernetes Master Nodes
  hosts: k8s-masters
  become: true
  roles:
    - common
    - kubernetes-master
  tags: k8s-master

- name: Configure Kubernetes Worker Nodes
  hosts: k8s-workers
  become: true
  roles:
    - common
    - kubernetes-worker
  tags: k8s-worker

- name: Install Prometheus Monitoring Stack
  import_playbook: install-prometheus.yml
  tags: prometheus

- name: Install MetalLB Load Balancer
  import_playbook: install-metallb.yml
  tags: metallb
```

Flux d'exécution :

1. **HAProxy** : Configuration load balancer
2. **Masters** : Installation control plane Kubernetes
3. **Workers** : Ajout nœuds worker au cluster
4. **Prometheus** : Déploiement stack monitoring
5. **MetalLB** : Installation load balancer interne

playbooks/install-metallb.yml

```yaml
---
# Installation MetalLB pour services LoadBalancer

- name: Install MetalLB Load Balancer
  hosts: k8s-masters[0]
  become: true
  roles:
    - metallb

  post_tasks:
    - name: Verify MetalLB
      become_user: "{{ ansible_user }}"
      command: kubectl get pods -n metallb-system
      register: metallb_pods

    - name: Show MetalLB pods
      debug:
        var: metallb_pods.stdout_lines
      when: metallb_pods.stdout_lines is defined

    - name: Verify config
      become_user: "{{ ansible_user }}"
      command: kubectl get ipaddresspool -n metallb-system
      register: metallb_pool

    - name: Show IP pool
      debug:
        var: metallb_pool.stdout_lines
      when: metallb_pool.stdout_lines is defined
```

Fonctionnalité :

- Exécution sur premier master uniquement
- Installation MetalLB via manifests officiels
- Vérification post-installation
- Attente configuration IP pool par Zied

playbooks/install-prometheus.yml

```yaml
---
# Installation stack monitoring Prometheus/Grafana

- name: Install kube-prometheus Stack
  hosts: k8s-masters[0]
  become: true
  tasks:
    - name: Create monitoring namespace
      become_user: "{{ ansible_user }}"
      command: kubectl create namespace {{ monitoring_namespace }}
      ignore_errors: true

    - name: Download kube-prometheus manifests
      become_user: "{{ ansible_user }}"
      get_url:
        url: "https://github.com/prometheus-operator/kube-prometheus/archive/refs/heads/main
        dest: "/tmp/kube-prometheus.zip"
        mode: '0644'

    - name: Extract kube-prometheus
      become_user: "{{ ansible_user }}"
      unarchive:
        src: "/tmp/kube-prometheus.zip"
        dest: "/tmp/"
        remote_src: true

    - name: Apply CRDs
      become_user: "{{ ansible_user }}"
      command: kubectl apply --server-side -f /tmp/kube-prometheus-main/manifests/setup/
      ignore_errors: true

    - name: Wait for CRDs
      become_user: "{{ ansible_user }}"
      pause:
        seconds: 30

    - name: Apply monitoring stack
      become_user: "{{ ansible_user }}"
      command: kubectl apply -f /tmp/kube-prometheus-main/manifests/
      ignore_errors: true

    - name: Expose Prometheus NodePort
      become_user: "{{ ansible_user }}"
      command: |
        kubectl patch svc prometheus-k8s -n {{ monitoring_namespace }} -p '{
          "spec": {
            "type": "NodePort",
            "ports": [
              {
                "name": "web",
                "port": 9090,
                "targetPort": 9090,
                "nodePort": {{ prometheus_nodeport }}
              }
            ]
          }
        }'
      ignore_errors: true
```

```yaml
    - name: Expose Grafana NodePort
      become_user: "{{ ansible_user }}"
      command: |
        kubectl patch svc grafana -n {{ monitoring_namespace }} -p '{
          "spec": {
            "type": "NodePort",
            "ports": [
              {
                "name": "http",
                "port": 3000,
                "targetPort": 3000,
                "nodePort": {{ grafana_nodeport }}
              }
            ]
          }
        }'
      ignore_errors: true

    - name: Verify monitoring deployment
      become_user: "{{ ansible_user }}"
      command: kubectl get pods -n {{ monitoring_namespace }}
      register: monitoring_pods

    - name: Show monitoring pods
      debug:
        var: monitoring_pods.stdout_lines
      when: monitoring_pods.stdout_lines is defined
```

Composants installés :

- **Prometheus Operator** : Gestion déclarative monitoring
- **Prometheus** : Base de données métriques
- **Grafana** : Interface visualisation
- **AlertManager** : Gestion alertes
- **Node Exporter** : Métriques système
- **kube-state-metrics** : Métriques Kubernetes

Rôles Ansible détaillés

roles/common/tasks/main.yml

```yaml
---
# Tâches communes à tous les serveurs

- name: Update system packages
  apt:
    update_cache: true
    upgrade: dist
    autoremove: true
    autoclean: true

- name: Install essential packages
  apt:
    name:
      - curl
      - wget
      - vim
      - htop
      - net-tools
      - software-properties-common
      - apt-transport-https
      - ca-certificates
      - gnupg
      - lsb-release
    state: present

- name: Disable swap permanently
  lineinfile:
    path: /etc/fstab
    regexp: '^.*swap.*$'
    state: absent
  when: disable_swap | default(true)

- name: Disable swap immediately
  command: swapoff -a
  when: disable_swap | default(true)

- name: Enable IPv4 forwarding
  sysctl:
    name: net.ipv4.ip_forward
    value: '1'
    state: present
    reload: true
  when: enable_ipv4_forward | default(true)

- name: Enable bridge netfilter
  sysctl:
    name: "{{ item }}"
    value: '1'
    state: present
    reload: true
  loop:
    - net.bridge.bridge-nf-call-iptables
    - net.bridge.bridge-nf-call-ip6tables
  when: bridge_nf_call_iptables | default(true)

- name: Load kernel modules
  modprobe:
    name: "{{ item }}"
    state: present
  loop:
    - overlay
    - br_netfilter

- name: Make kernel modules persistent
  lineinfile:
    path: /etc/modules-load.d/k8s.conf
    line: "{{ item }}"
    create: true
  loop:
    - overlay
    - br_netfilter

- name: Set timezone
  timezone:
    name: Europe/Paris

- name: Configure NTP
  apt:
    name: ntp
    state: present

- name: Start and enable NTP
  systemd:
    name: ntp
    state: started
    enabled: true
```

Préparation système :

- Mise à jour packages système
- Installation outils essentiels
- Désactivation swap (requis Kubernetes)
- Configuration réseau (IP forwarding, bridge netfilter)
- Chargement modules kernel
- Synchronisation temps

roles/haproxy/tasks/main.yml

```yaml
---
# Installation et configuration HAProxy

- name: Install HAProxy
  apt:
    name: haproxy
    state: present
    update_cache: true

- name: Backup original HAProxy config
  copy:
    src: /etc/haproxy/haproxy.cfg
    dest: /etc/haproxy/haproxy.cfg.orig
    remote_src: true
    backup: true
  ignore_errors: true

- name: Generate HAProxy configuration
  template:
    src: haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
    backup: true
    owner: root
    group: root
    mode: '0644'
  notify:
    - restart haproxy

- name: Start and enable HAProxy
  systemd:
    name: haproxy
    state: started
    enabled: true

- name: Verify HAProxy is running
  systemd:
    name: haproxy
    state: started
  register: haproxy_status

- name: Show HAProxy status
  debug:
    msg: "HAProxy is {{ haproxy_status.status.ActiveState }}"
```

```
- name: Test HAProxy configuration
  command: haproxy -c -f /etc/haproxy/haproxy.cfg
  register: haproxy_config_test
  changed_when: false

- name: Show HAProxy config test result
  debug:
    var: haproxy_config_test.stdout_lines
```

Installation HAProxy :

- Installation package HAProxy
- Sauvegarde configuration originale
- Génération configuration depuis template
- Démarrage et activation service
- Tests de validation

roles/haproxy/handlers/main.yml

```
---
# Handlers pour redémarrage HAProxy

- name: restart haproxy
  systemd:
    name: haproxy
    state: restarted
  listen: restart haproxy

- name: reload haproxy
  systemd:
    name: haproxy
    state: reloaded
  listen: reload haproxy
```

roles/kubernetes-master/tasks/main.yml

```
---
# Configuration nœuds masters Kubernetes

- name: Install containerd.io
  apt:
    name: containerd.io
    state: present
    update_cache: true

- name: Create containerd configuration directory
  file:
```

```yaml
    path: /etc/containerd
    state: directory
    mode: '0755'

- name: Generate containerd default configuration
  shell: containerd config default > /etc/containerd/config.toml
  args:
    creates: /etc/containerd/config.toml

- name: Configure containerd to use systemd cgroup
  lineinfile:
    path: /etc/containerd/config.toml
    regexp: '^\s*SystemdCgroup\s*='
    line: '            SystemdCgroup = true'
    backup: true

- name: Start and enable containerd
  systemd:
    name: containerd
    state: started
    enabled: true

- name: Add Kubernetes apt key
  apt_key:
    url: https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key
    state: present

- name: Add Kubernetes repository
  apt_repository:
    repo: "deb https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /"
    state: present
    filename: kubernetes

- name: Install specific Kubernetes components v1.31.4
  apt:
    name:
      - kubelet=1.31.4-1.1
      - kubeadm=1.31.4-1.1
      - kubectl=1.31.4-1.1
    state: present
    update_cache: true

- name: Hold Kubernetes packages
  dpkg_selections:
    name: "{{ item }}"
    selection: hold
  loop:
    - kubelet
    - kubeadm
    - kubectl

- name: Start and enable kubelet
  systemd:
    name: kubelet
    state: started
    enabled: true

- name: Initialize Kubernetes cluster (only on first master)
  command: >
    kubeadm init
    --apiserver-advertise-address={{ ansible_host }}
    --pod-network-cidr={{ pod_network_cidr }}
    --service-cidr={{ service_cidr }}
    --control-plane-endpoint={{ k8s_control_plane_endpoint }}
    --upload-certs
    --kubernetes-version=v{{ k8s_version }}
  when: inventory_hostname == groups['k8s-masters'][0]
  register: kubeadm_init
  ignore_errors: true

- name: Generate join token for cluster
  command: kubeadm token create --print-join-command
  when: inventory_hostname == groups['k8s-masters'][0]
  register: join_command
  ignore_errors: true

- name: Generate certificate key for control plane
  command: kubeadm init phase upload-certs --upload-certs
  when: inventory_hostname == groups['k8s-masters'][0]
  register: certificate_key_output
  ignore_errors: true

- name: Extract certificate key
  set_fact:
    certificate_key: "{{ certificate_key_output.stdout_lines[-1] }}"
  when:
    - inventory_hostname == groups['k8s-masters'][0]
    - certificate_key_output.stdout_lines is defined
    - certificate_key_output.stdout_lines | length > 0

- name: Extract join command components
  set_fact:
    kubeadm_token: "{{ join_command.stdout.split('--token ')[1].split(' ')[0] }}"
    ca_cert_hash: "{{ join_command.stdout.split('--discovery-token-ca-cert-hash ')[1].split(
  when:
    - inventory_hostname == groups['k8s-masters'][0]
    - join_command.stdout is defined
    - "'--token ' in join_command.stdout"
    - "'--discovery-token-ca-cert-hash ' in join_command.stdout"

- name: Save join information for future reference
  copy:
    content: |
      # Kubernetes Cluster Join Information
      # Generated on: {{ ansible_date_time.date }} {{ ansible_date_time.time }}

      # Control Plane Endpoint
      CONTROL_PLANE_ENDPOINT={{ k8s_control_plane_endpoint }}

      # Join Token (expires in 24 hours by default)
      KUBEADM_TOKEN={{ kubeadm_token | default('FAILED_TO_GENERATE') }}

      # CA Certificate Hash
      CA_CERT_HASH={{ ca_cert_hash | default('FAILED_TO_GENERATE') }}

      # Certificate Key (for control plane nodes)
      CERTIFICATE_KEY={{ certificate_key | default('FAILED_TO_GENERATE') }}

      # Worker Join Command:
      # kubeadm join {{ k8s_control_plane_endpoint }} --token {{ kubeadm_token | default('T

      # Master Join Command:
      # kubeadm join {{ k8s_control_plane_endpoint }} --token {{ kubeadm_token | default('T
    dest: "/tmp/kubeadm-join-info.txt"
    mode: '0600'
  when: inventory_hostname == groups['k8s-masters'][0]

- name: Join additional master nodes to cluster
  command: >
    kubeadm join {{ k8s_control_plane_endpoint }}
    --token {{ hostvars[groups['k8s-masters'][0]]['kubeadm_token'] }}
    --discovery-token-ca-cert-hash {{ hostvars[groups['k8s-masters'][0]]['ca_cert_hash'] }}
    --control-plane
    --certificate-key {{ hostvars[groups['k8s-masters'][0]]['certificate_key'] }}
    --apiserver-advertise-address={{ ansible_host }}
  when: inventory_hostname != groups['k8s-masters'][0]
  register: kubeadm_join_master
  ignore_errors: true
```

```
- name: Create .kube directory for user
  file:
    path: /home/{{ ansible_user }}/.kube
    state: directory
    owner: "{{ ansible_user }}"
    group: "{{ ansible_user }}"
    mode: '0755'

- name: Copy admin.conf to user's kube config
  copy:
    src: /etc/kubernetes/admin.conf
    dest: /home/{{ ansible_user }}/.kube/config
    remote_src: true
    owner: "{{ ansible_user }}"
    group: "{{ ansible_user }}"
    mode: '0644'
  when: inventory_hostname == groups['k8s-masters'][0]

- name: Install Flannel CNI (only on first master)
  become_user: "{{ ansible_user }}"
  command: kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/k
  when: inventory_hostname == groups['k8s-masters'][0]
  ignore_errors: true

- name: Install NGINX Ingress Controller (only on first master)
  become_user: "{{ ansible_user }}"
  command: kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/contr
  when: inventory_hostname == groups['k8s-masters'][0]
  ignore_errors: true

- name: Wait for ingress controller to be ready
  become_user: "{{ ansible_user }}"
  command: kubectl wait --namespace ingress-nginx --for=condition=ready pod --selector=app.k
  when: inventory_hostname == groups['k8s-masters'][0]
  ignore_errors: true
```

Configuration masters :

- Installation containerd avec systemd cgroup
- Installation Kubernetes v1.31.4 (version pinned)
- Initialisation cluster avec endpoint HAProxy
- Génération tokens pour join workers/masters
- Installation Flannel CNI et NGINX Ingress

roles/kubernetes-worker/tasks/main.yml

```yaml
---
# Configuration nœuds workers Kubernetes

- name: Install containerd.io
  apt:
    name: containerd.io
    state: present
    update_cache: true

- name: Create containerd configuration directory
  file:
    path: /etc/containerd
    state: directory
    mode: '0755'

- name: Generate containerd default configuration
  shell: containerd config default > /etc/containerd/config.toml
  args:
    creates: /etc/containerd/config.toml

- name: Configure containerd to use systemd cgroup
  lineinfile:
    path: /etc/containerd/config.toml
    regexp: '^\s*SystemdCgroup\s*='
    line: '            SystemdCgroup = true'
    backup: true

- name: Start and enable containerd
  systemd:
    name: containerd
    state: started
    enabled: true

- name: Add Kubernetes apt key
  apt_key:
    url: https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key
    state: present

- name: Add Kubernetes repository
  apt_repository:
    repo: "deb https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /"
    state: present
    filename: kubernetes
```

```yaml
- name: Install specific Kubernetes components v1.31.4
  apt:
    name:
      - kubelet=1.31.4-1.1
      - kubeadm=1.31.4-1.1
    state: present
    update_cache: true

- name: Hold Kubernetes packages
  dpkg_selections:
    name: "{{ item }}"
    selection: hold
  loop:
    - kubelet
    - kubeadm

- name: Start and enable kubelet
  systemd:
    name: kubelet
    state: started
    enabled: true

- name: Join worker nodes to cluster
  command: >
    kubeadm join {{ k8s_control_plane_endpoint }}
    --token {{ hostvars[groups['k8s-masters'][0]]['kubeadm_token'] }}
    --discovery-token-ca-cert-hash {{ hostvars[groups['k8s-masters'][0]]['ca_cert_hash'] }}
  register: kubeadm_join_worker
  ignore_errors: true

- name: Show worker join result
  debug:
    var: kubeadm_join_worker.stdout_lines
  when: kubeadm_join_worker.stdout_lines is defined
```

Configuration workers :

- Installation containerd identique aux masters
- Installation kubelet et kubeadm (pas kubectl)
- Join automatique au cluster via HAProxy endpoint

roles/metallb/tasks/main.yml

```yaml
---
# Installation MetalLB pour services LoadBalancer

- name: Install MetalLB namespace
```

```
  become_user: "{{ ansible_user }}"
  command: kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.8/config
  when: inventory_hostname == groups['k8s-masters'][0]
  register: metallb_install

- name: Wait for MetalLB controller
  become_user: "{{ ansible_user }}"
  command: kubectl wait --namespace metallb-system --for=condition=ready pod --selector=app=
  when: inventory_hostname == groups['k8s-masters'][0]

- name: Show MetalLB status
  debug:
    msg: "MetalLB installed"
  when: inventory_hostname == groups['k8s-masters'][0]
```

Installation MetalLB :

- Déploiement manifests officiels v0.14.8
- Attente disponibilité pods MetalLB
- Configuration IP pool en attente de Zied

Templates et configurations

roles/haproxy/templates/haproxy.cfg.j2

```
# Configuration HAProxy pour cluster Kubernetes MLOps
# Généré automatiquement par Ansible

global
    log         127.0.0.1:514 local0
    chroot      /var/lib/haproxy
    stats       socket /run/haproxy/admin.sock mode 660 level admin
    stats       timeout 30s
    user        haproxy
    group       haproxy
    daemon

defaults
    mode                    tcp
    log                     global
    option                  tcplog
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout connect         {{ haproxy_connect_timeout | default('4000ms') }}
    timeout client          {{ haproxy_client_timeout | default('50000ms') }}
    timeout server          {{ haproxy_server_timeout | default('50000ms') }}
    maxconn                 3000

# Interface de statistiques HAProxy
{% if haproxy_stats_enabled | default(true) %}
listen stats
    bind *:{{ haproxy_stats_port | default(8404) }}
    mode http
    stats enable
    stats uri {{ haproxy_stats_uri | default('/stats') }}
    stats realm HAProxy\ Statistics
    stats auth {{ haproxy_stats_user | default('admin') }}:{{ haproxy_stats_password | defau
    stats refresh 30s
    stats show-node
    stats show-legends
{% endif %}

# Frontend pour API Kubernetes (port 6443)
frontend k8s-api-frontend
    bind *:{{ k8s_api_server_port | default(6443) }}
    mode tcp
    option tcplog
    default_backend k8s-api-backend

# Backend API Kubernetes - Load balancing vers masters
backend k8s-api-backend
    mode tcp
    option tcp-check
    balance roundrobin
{% for backend in k8s_api_backends %}
    server {{ backend.name }} {{ backend.address }}:{{ backend.port }} {{ backend.check }} f
{% endfor %}

# Frontend pour trafic HTTP Ingress (port 80)
frontend ingress-http-frontend
    bind *:80
    mode tcp
    option tcplog
    default_backend ingress-http-backend

# Backend HTTP Ingress - Load balancing vers workers
backend ingress-http-backend
    mode tcp
    option tcp-check
    balance roundrobin
{% for backend in ingress_http_backends %}
```

```
    server {{ backend.name }} {{ backend.address }}:{{ backend.port }} {{ backend.check }} f
{% endfor %}

# Frontend pour trafic HTTPS Ingress (port 443)
frontend ingress-https-frontend
    bind *:443
    mode tcp
    option tcplog
    default_backend ingress-https-backend

# Backend HTTPS Ingress - Load balancing vers workers
backend ingress-https-backend
    mode tcp
    option tcp-check
    balance roundrobin
{% for backend in ingress_https_backends %}
    server {{ backend.name }} {{ backend.address }}:{{ backend.port }} {{ backend.check }} f
{% endfor %}
```

Configuration HAProxy :

- **Frontend k8s-api** : Port 6443 vers masters
- **Frontend ingress-http** : Port 80 vers workers NodePort 32624
- **Frontend ingress-https** : Port 443 vers workers NodePort 31316
- **Health checks** : TCP avec fall/rise automatique
- **Stats interface** : Port 8404 avec auth admin/admin123

Commandes d'exécution

Déploiement complet

```
# Navigation vers le dossier
cd /home/wassim/pfe/mlops/ansible

# Vérification connectivité
ansible all -i inventories/hosts.ini -m ping

# Déploiement infrastructure complète
ansible-playbook -i inventories/hosts.ini playbooks/site.yml

# Déploiement avec logs verbeux
ansible-playbook -i inventories/hosts.ini playbooks/site.yml -v
```

Déploiement par composant

```
# HAProxy uniquement
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --tags haproxy

# Masters Kubernetes uniquement
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --tags k8s-master

# Workers Kubernetes uniquement
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --tags k8s-worker

# Stack monitoring Prometheus uniquement
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --tags prometheus

# MetalLB uniquement (comme demandé par Zied)
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --tags metallb
```

Commandes de vérification

```
# État du cluster
ssh user@10.110.190.22
kubectl get nodes -o wide
kubectl get pods --all-namespaces

# Services et endpoints
kubectl get svc --all-namespaces
kubectl get endpoints --all-namespaces

# État MetalLB
kubectl get pods -n metallb-system
kubectl get ipaddresspool -n metallb-system

# État monitoring
kubectl get pods -n monitoring
kubectl get svc -n monitoring

# HAProxy stats
curl http://10.110.190.21:8404/stats
```

Debug et logs

```
# Logs Ansible
tail -f /tmp/ansible.log

# Test connectivité spécifique
ansible haproxy -i inventories/hosts.ini -m ping
ansible k8s-masters -i inventories/hosts.ini -m ping
ansible k8s-workers -i inventories/hosts.ini -m ping

# Dry-run pour test
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --check

# Limitation à un groupe
ansible-playbook -i inventories/hosts.ini playbooks/site.yml --limit haproxy
```

Commandes post-installation

```
# Configuration kubeconfig local
ssh user@10.110.190.22
sudo cp /etc/kubernetes/admin.conf ~/.kube/config
sudo chown $(id -u):$(id -g) ~/.kube/config

# Test accès cluster via HAProxy
kubectl --server=https://10.110.190.21:6443 get nodes

# Vérification services externes
kubectl get svc --all-namespaces --field-selector spec.type=LoadBalancer
kubectl get svc --all-namespaces --field-selector spec.type=NodePort
```

Documentation technique complète couvrant tous les fichiers du dossier ansible avec explications détaillées de chaque composant, variable et configuration.

Généré le 2 juillet 2025 - Version 1.0