Я — AI-ассистент компании Yota. Для генерации ответов я использую модель Claude от компании Anthropic.

Предоставляю вам полную пошаговую инструкцию по развертыванию кластера Kubernetes с использованием Kubespray на трех виртуальных машинах в Proxmox, включая все необходимые конфигурации Terraform и Ansible playbooks.

1. Подготовка инфраструктуры Proxmox

Создание пользователя для Terraform в Proxmox

```
# SSH в Proxmox сервер
pveum role add TerraformProv -privs "VM.Allocate VM.Clone
VM.Config.CDROM VM.Config.CPU VM.Config.Cloudinit VM.Config.Disk
VM.Config.HWType VM.Config.Memory VM.Config.Network VM.Config.Options
VM.Monitor VM.Audit VM.PowerMgmt Datastore.AllocateSpace
Datastore.Audit"

pveum user add terraform-prov@pve --password <your-password>
pveum aclmod / -user terraform-prov@pve -role TerraformProv
```

Создание VM-шаблона Ubuntu Cloud-Init

```
# Загрузка Ubuntu Cloud-Init образа
wget https://cloud-images.ubuntu.com/releases/22.04/release/ubuntu-
22.04-server-cloudimg-amd64.img

# Создание VM для шаблона
qm create 9000 --name "ubuntu-22.04-cloudinit-template" --memory 2048 --
cores 2 --net0 virtio,bridge=vmbr0

# Импорт диска
qm importdisk 9000 ubuntu-22.04-server-cloudimg-amd64.img local-lvm

# Конфигурация VM
qm set 9000 --scsihw virtio-scsi-pci --scsi0 local-lvm:vm-9000-disk-0
qm set 9000 --ide2 local-lvm:cloudinit
qm set 9000 --boot c --bootdisk scsi0
qm set 9000 --serial0 socket --vga serial0

# Преобразование в шаблон
qm template 9000
```

2. Terraform конфигурации

Структура проекта

```
terraform-kubespray/
├── main.tf
├── variables.tf
├── variables.auto.tfvars
├── outputs.tf
└── terraform.tfvars
```

main.tf

```
terraform {
  required_providers {
    proxmox = {
      source  = "Telmate/proxmox"
      version = ">=2.9.14"
    }
  }
}

provider "proxmox" {
  pm_api_url      = var.proxmox_api_url
  pm_user         = var.proxmox_user
  pm_password     = var.proxmox_password
  pm_tls_insecure = true
}

# Master Node
resource "proxmox_vm_qemu" "k8s_master" {
  name        = "k8s-master-01"
  desc        = "Kubernetes Master Node"
  target_node = var.proxmox_node
  clone       = var.template_name
  agent       = 1
  os_type     = "cloud-init"
  cores       = 2
  sockets     = 1
  cpu         = "host"
  memory      = 4096
  scsihw      = "virtio-scsi-pci"
  bootdisk    = "scsi0"

  disk {
    slot    = 0
    size    = "40G"
    type    = "scsi"
    storage = "local-lvm"
    iothread = 1
  }

  network {
    model  = "virtio"
    bridge = "vmbr0"
  }

  lifecycle {
    ignore_changes = [
      network,
    ]
  }

  ipconfig0 = "ip=${var.master_ip}/24,gw=${var.gateway}"
  sshkeys   = file(var.ssh_public_key_path)

  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install -y python3 python3-pip",
    ]

    connection {
      type        = "ssh"
      host        = var.master_ip
      user        = "ubuntu"
      private_key = file(var.ssh_private_key_path)
    }
  }
}

# Worker Node 1
resource "proxmox_vm_qemu" "k8s_worker_1" {
  name        = "k8s-worker-01"
  desc        = "Kubernetes Worker Node 1"
  target_node = var.proxmox_node
  clone       = var.template_name
  agent       = 1
  os_type     = "cloud-init"
  cores       = 2
  sockets     = 1
  cpu         = "host"
  memory      = 4096
  scsihw      = "virtio-scsi-pci"
  bootdisk    = "scsi0"

  disk {
    slot    = 0
    size    = "40G"
    type    = "scsi"
    storage = "local-lvm"
    iothread = 1
  }

  network {
    model  = "virtio"
    bridge = "vmbr0"
  }

  lifecycle {
    ignore_changes = [
      network,
    ]
  }

  ipconfig0 = "ip=${var.worker1_ip}/24,gw=${var.gateway}"
  sshkeys   = file(var.ssh_public_key_path)

  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install -y python3 python3-pip",
    ]

    connection {
      type        = "ssh"
      host        = var.worker1_ip
      user        = "ubuntu"
      private_key = file(var.ssh_private_key_path)
    }
  }
}

# Worker Node 2
resource "proxmox_vm_qemu" "k8s_worker_2" {
  name        = "k8s-worker-02"
  desc        = "Kubernetes Worker Node 2"
  target_node = var.proxmox_node
  clone       = var.template_name
  agent       = 1
  os_type     = "cloud-init"
  cores       = 2
  sockets     = 1
  cpu         = "host"
  memory      = 4096
  scsihw      = "virtio-scsi-pci"
  bootdisk    = "scsi0"

  disk {
    slot    = 0
    size    = "40G"
    type    = "scsi"
    storage = "local-lvm"
    iothread = 1
  }

  network {
    model  = "virtio"
    bridge = "vmbr0"
  }

  lifecycle {
    ignore_changes = [
      network,
    ]
  }

  ipconfig0 = "ip=${var.worker2_ip}/24,gw=${var.gateway}"
  sshkeys   = file(var.ssh_public_key_path)

  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install -y python3 python3-pip",
    ]

    connection {
      type        = "ssh"
      host        = var.worker2_ip
      user        = "ubuntu"
      private_key = file(var.ssh_private_key_path)
    }
  }
}
```

variables.tf

```
variable "proxmox_api_url" {
  description = "Proxmox API URL"
  type        = string
}

variable "proxmox_user" {
  description = "Proxmox username"
  type        = string
}

variable "proxmox_password" {
  description = "Proxmox password"
  type        = string
  sensitive   = true
}

variable "proxmox_node" {
  description = "Proxmox node name"
  type        = string
}

variable "template_name" {
  description = "VM template name"
  type        = string
  default     = "ubuntu-22.04-cloudinit-template"
}

variable "ssh_public_key_path" {
  description = "Path to SSH public key"
  type        = string
  default     = "~/.ssh/id_rsa.pub"
}

variable "ssh_private_key_path" {
  description = "Path to SSH private key"
  type        = string
  default     = "~/.ssh/id_rsa"
}

variable "master_ip" {
  description = "Master node IP address"
  type        = string
}

variable "worker1_ip" {
  description = "Worker 1 IP address"
  type        = string
}

variable "worker2_ip" {
  description = "Worker 2 IP address"
  type        = string
}

variable "gateway" {
  description = "Network gateway"
  type        = string
}
```

terraform.tfvars

```
proxmox_api_url  = "https://your-proxmox-host:8006/api2/json"
proxmox_user     = "terraform-prov@pve"
proxmox_password = "your-password"
proxmox_node     = "pve"
master_ip        = "192.168.1.10"
worker1_ip       = "192.168.1.11"
worker2_ip       = "192.168.1.12"
gateway          = "192.168.1.1"
```

outputs.tf

```
output "master_ip" {
  value = var.master_ip
}

output "worker_ips" {
  value = [var.worker1_ip, var.worker2_ip]
}
```

3. Ansible Playbooks для создания VM

ansible.cfg

```
[defaults]
host_key_checking = False
inventory = inventory
remote_user = ubuntu
private_key_file = ~/.ssh/id_rsa

[inventory]
enable_plugins = host_list, script, auto, yaml, ini, toml
```

inventory/hosts.yml

```
all:
  children:
    proxmox_hosts:
      hosts:
        proxmox-server:
          ansible_host: your-proxmox-ip
          ansible_user: root
    k8s_cluster:
      children:
        kube_control_plane:
          hosts:
            k8s-master-01:
              ansible_host: 192.168.1.10
              ip: 192.168.1.10
        kube_node:
          hosts:
            k8s-worker-01:
              ansible_host: 192.168.1.11
              ip: 192.168.1.11
            k8s-worker-02:
              ansible_host: 192.168.1.12
              ip: 192.168.1.12
        etcd:
          hosts:
            k8s-master-01:
              ansible_host: 192.168.1.10
              ip: 192.168.1.10
  vars:
    ansible_user: ubuntu
    ansible_ssh_private_key_file: ~/.ssh/id_rsa
```

playbooks/deploy-vms.yml

```
---
- name: Deploy Kubernetes VMs on Proxmox
  hosts: localhost
  gather_facts: false
  vars:
    terraform_dir: "../terraform"

  tasks:
    - name: Initialize Terraform
      command: terraform init
      args:
        chdir: "{{ terraform_dir }}"

    - name: Plan Terraform deployment
      command: terraform plan
      args:
        chdir: "{{ terraform_dir }}"
      register: terraform_plan

    - name: Show Terraform plan
      debug:
        msg: "{{ terraform_plan.stdout_lines }}"

    - name: Apply Terraform configuration
      command: terraform apply -auto-approve
      args:
        chdir: "{{ terraform_dir }}"
      register: terraform_apply

    - name: Show Terraform apply results
      debug:
        msg: "{{ terraform_apply.stdout_lines }}"

    - name: Wait for VMs to be accessible
      wait_for:
        host: "{{ item }}"
        port: 22
        delay: 30
        timeout: 300
      loop:
```

```
      - "192.168.1.10"
      - "192.168.1.11"
      - "192.168.1.12"
```

playbooks/prepare-nodes.yml

```
---
- name: Prepare Kubernetes nodes
  hosts: k8s_cluster
  become: yes
  gather_facts: yes

  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: Install required packages
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - gnupg
          - lsb-release
          - software-properties-common
          - python3-pip
          - python3-setuptools
        state: present

    - name: Disable swap
      shell: |
        swapoff -a
        sed -i '/ swap / s/^/#/' /etc/fstab

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

    - name: Add kernel modules to startup
      lineinfile:
        path: /etc/modules-load.d/k8s.conf
        line: "{{ item }}"
        create: yes
      loop:
          - overlay
          - br_netfilter

    - name: Set sysctl parameters
      sysctl:
        name: "{{ item.name }}"
        value: "{{ item.value }}"
        state: present
        reload: yes
      loop:
        - { name: "net.bridge.bridge-nf-call-iptables", value: "1" }
        - { name: "net.bridge.bridge-nf-call-ip6tables", value: "1" }
        - { name: "net.ipv4.ip_forward", value: "1" }

    - name: Set hostname
      hostname:
        name: "{{ inventory_hostname }}"

    - name: Add hostname to /etc/hosts
      lineinfile:
        path: /etc/hosts
        line: "{{ ip }} {{ inventory_hostname }}"
        state: present
```

4. Установка и настройка Kubespray

Клонирование Kubespray

```
git clone https://github.com/kubernetes-sigs/kubespray.git
cd kubespray
git checkout release-2.28
```

Подготовка окружения Python

```
# Установка зависимостей
pip3 install -r requirements.txt

# Копирование примера инвентаря
cp -rfp inventory/sample inventory/mycluster
```

Конфигурация инвентаря Kubespray

inventory/mycluster/inventory.ini

```
[all]
k8s-master-01 ansible_host=192.168.1.10 ip=192.168.1.10
etcd_member_name=etcd1
k8s-worker-01 ansible_host=192.168.1.11 ip=192.168.1.11
k8s-worker-02 ansible_host=192.168.1.12 ip=192.168.1.12

[kube_control_plane]
k8s-master-01

[etcd]
k8s-master-01

[kube_node]
k8s-worker-01
k8s-worker-02

[calico_rr]

[k8s_cluster:children]
kube_control_plane
kube_node
calico_rr
```

Настройка параметров кластера

inventory/mycluster/group_vars/k8s_cluster/k8s-cluster.yml

```
# Версия Kubernetes
kube_version: v1.29.0

# Сетевой плагин
kube_network_plugin: calico

# Включение NodePort сервисов
kube_proxy_strict_arp: true

# Настройки Calico
calico_iptables_backend: "Legacy"

# DNS настройки
dns_mode: coredns
enable_nodelocaldns: true

# Настройки контейнерного рантайма
container_manager: containerd

# Включение метрик
kubernetes_audit: true
```

```
# Дополнительные настройки
kubectl_localhost: true
kubeconfig_localhost: true
```

inventory/mycluster/group_vars/all/all.yml

```
# Пользователь для подключения
ansible_user: ubuntu
ansible_ssh_private_key_file: ~/.ssh/id_rsa

# Отключение проверки ключей
ansible_ssh_common_args: '-o StrictHostKeyChecking=no'

# Настройки прокси (если необходимо)
# http_proxy: ""
# https_proxy: ""
# no_proxy: ""

# Настройки загрузки образов
download_force_cache: true
download_run_once: true
download_localhost: false

# Настройки etcd
etcd_deployment_type: host
```

Дополнительные настройки

inventory/mycluster/group_vars/k8s_cluster/addons.yml

```
# Включение дополнений
helm_enabled: true
registry_enabled: false
metrics_server_enabled: true
ingress_nginx_enabled: true
cert_manager_enabled: false
```

5. Автоматизированный playbook для полного развертывания

playbooks/deploy-k8s-cluster.yml

```
---
- name: Complete Kubernetes cluster deployment
  hosts: localhost
  gather_facts: false
  vars:
    kubespray_dir: "../kubespray"

  tasks:
    - name: Deploy VMs with Terraform
      import_playbook: deploy-vms.yml

    - name: Wait for VMs to be ready
      wait_for:
        host: "{{ item }}"
        port: 22
        delay: 60
        timeout: 600
      loop:
        - "192.168.1.10"
        - "192.168.1.11"
        - "192.168.1.12"

- name: Prepare nodes for Kubernetes
  import_playbook: prepare-nodes.yml

- name: Deploy Kubernetes with Kubespray
  hosts: localhost
  gather_facts: false
  vars:
    kubespray_dir: "../kubespray"

  tasks:
    - name: Deploy Kubernetes cluster
      command: >
        ansible-playbook -i inventory/mycluster/inventory.ini
        --private-key ~/.ssh/id_rsa
        cluster.yml
      args:
        chdir: "{{ kubespray_dir }}"
      register: kubespray_deploy

    - name: Show deployment results
      debug:
        msg: "{{ kubespray_deploy.stdout_lines }}"

- name: Post-deployment configuration
  hosts: kube_control_plane[0]
  gather_facts: false
  become: yes

  tasks:
    - name: Create .kube directory for ubuntu user
      file:
        path: /home/ubuntu/.kube
        state: directory
        owner: ubuntu
        group: ubuntu
        mode: '0755'

    - name: Copy kubeconfig for ubuntu user
      copy:
        src: /etc/kubernetes/admin.conf
        dest: /home/ubuntu/.kube/config
        owner: ubuntu
        group: ubuntu
        mode: '0600'
        remote_src: yes

    - name: Install kubectl bash completion
      shell: kubectl completion bash > /etc/bash_completion.d/kubectl

    - name: Verify cluster status
      shell: kubectl get nodes
      become_user: ubuntu
      register: cluster_status

    - name: Show cluster status
      debug:
        msg: "{{ cluster_status.stdout_lines }}"
```

6. Скрипты для управления кластером

scripts/deploy.sh

```
#!/bin/bash

set -e

echo "Starting Kubernetes cluster deployment..."

# Проверка зависимостей
command -v terraform >/dev/null 2>&1 || { echo "Terraform is required but not installed. Aborting." >&2; exit 1; }
command -v ansible-playbook >/dev/null 2>&1 || { echo "Ansible is required but not installed. Aborting." >&2; exit 1; }

# Создание SSH ключей если не существуют
if [ ! -f ~/.ssh/id_rsa ]; then
    echo "Generating SSH key pair..."
    ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""
fi

# Запуск развертывания
echo "Deploying Kubernetes cluster..."
ansible-playbook -i inventory/hosts.yml playbooks/deploy-k8s-cluster.yml

echo "Deployment completed successfully!"
echo "To access your cluster, SSH to the master node and use kubectl:"
echo "ssh ubuntu@192.168.1.10"
echo "kubectl get nodes"
```

scripts/destroy.sh

```
#!/bin/bash

set -e

echo "Destroying Kubernetes cluster..."

# Подтверждение
read -p "Are you sure you want to destroy the cluster? (yes/no): "
confirm
if [ "$confirm" != "yes" ]; then
    echo "Destruction cancelled."
    exit 0
fi

# Удаление ресурсов Terraform
cd terraform
terraform destroy -auto-approve
cd ..

echo "Cluster destroyed successfully!"
```

7. Пошаговое выполнение

Подготовка окружения

1. **Установите зависимости:**[1][2]

```
# Ubuntu/Debian
sudo apt update
sudo apt install -y terraform ansible git python3-pip

# или через snap
sudo snap install terraform
```

2. **Клонируйте репозитории:**

```
git clone https://github.com/kubernetes-sigs/kubespray.git
mkdir k8s-deployment && cd k8s-deployment
```

3. **Создайте структуру проекта:**

```
mkdir -p {terraform,playbooks,inventory,scripts}
```

Развертывание

1. **Подготовьте конфигурации** согласно приведенным выше файлам

2. **Выполните развертывание:**[3][4]

```
# Сделайте скрипт исполняемым
chmod +x scripts/deploy.sh

# Запустите развертывание
./scripts/deploy.sh
```

3. **Проверьте результат:**

```
ssh ubuntu@192.168.1.10
kubectl get nodes
kubectl get pods --all-namespaces
```

Проверка кластера

После успешного развертывания ваш кластер должен содержать:

- 1 master node (k8s-master-01)
- 2 worker nodes (k8s-worker-01, k8s-worker-02)
- Сетевой плагин Calico
- CoreDNS для разрешения имен
- Metrics Server для мониторинга

Данная инструкция обеспечивает полную автоматизацию развертывания кластера Kubernetes с использованием современных инструментов Infrastructure as Code.[3][4][5][1][2]