Kubernetes

- K8s Top Commands
- Установка K8s
- Схемы k8s

K8s Top Commands

?????? nodes

```
kubectl get nodes
# kubectl get nodes -o wide
```

??????? namespace

```
kubectl create ns my-ns
```

????????? ????????

```
kubectl get ns my-ns -o yaml > ns.yaml
```

????????? ????????

```
kubectl apply -f ns.yaml
```

????????? info

```
kubectl describe ns my-ns
```

???????? label

```
kubectl label ns my-ns kubernetes=rulezz
```

???????? ?????????

```
kubectl annotate ns my-ns annotations=likelabels
```

????????? K8s

???? ????????????? k8s ????????

Terraform ???????? VMs

> Ресурсы облака **Yandex.Cloud** на **Ubuntu 24.04 server** (2-CPU/4gb-RAM/30gb-Disk)

main.tf

```
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
    local = {
      source = "hashicorp/local"
    }
  }
}

provider "yandex" {
  service_account_key_file = "${file("~/.ssh/yc.service.key")}"
  folder_id                = "b1gtv3uhmd1feqdajugp"
  zone                     = "ru-central1-a"
}

# Актуальный образ Ubuntu 24.04 LTS
data "yandex_compute_image" "last_ubuntu" {
  family = "ubuntu-24-04-lts"
}

# Существующая подсеть, в которую разместятся все ВМ
```

```
data "yandex_vpc_subnet" "default_a" {
  name = "default-ru-central1-a"
}

locals {
  task_name    = "kube"
  worker_names = ["node01", "node02", "node03"]
  ssh_pub_key  = file("~/.ssh/yc.pub")
  ssh_priv_key = "~/.ssh/yc"
}

# Control-plane нода (inventory-имя — kube-master)
resource "yandex_compute_instance" "cplane01" {
  name        = "${local.task_name}-cplane01"
  platform_id = "standard-v3"

  resources {
    cores  = 2
    memory = 4
  }

  boot_disk {
    initialize_params {
      image_id = data.yandex_compute_image.last_ubuntu.id
      size     = 30
    }
  }

  labels = {
    task_name  = local.task_name
    user_email = "xeonmp22_at_gmail.com"
    role       = "control-plane"
  }

  network_interface {
    subnet_id = data.yandex_vpc_subnet.default_a.subnet_id
    nat       = true
  }
```

```
  metadata = {
    ssh-keys = "ubuntu:${local.ssh_pub_key}"
  }
}

# 3 worker-ноды (node01..node03), создаются циклом по local.worker_names
resource "yandex_compute_instance" "workers" {
  for_each    = toset(local.worker_names)
  name        = "${local.task_name}-${each.key}"
  platform_id = "standard-v3"

  resources {
    cores  = 2
    memory = 4
  }

  boot_disk {
    initialize_params {
      image_id = data.yandex_compute_image.last_ubuntu.id
      size     = 30
    }
  }

  labels = {
    task_name  = local.task_name
    user_email = "xeonmp22_at_gmail.com"
    role       = "worker"
  }

  network_interface {
    subnet_id = data.yandex_vpc_subnet.default_a.subnet_id
    nat       = true
  }

  metadata = {
    ssh-keys = "ubuntu:${local.ssh_pub_key}"
  }
}
```

```
# Генерирует inventory.ini с реальными IP всех 4 ВМ (шаблон — inventory.tpl)
resource "local_file" "ansible_inventory" {
  filename = "${path.module}/inventory.ini"
  content = templatefile("${path.module}/inventory.tpl", {
    master_ip          =
yandex_compute_instance.cplane01.network_interface.0.nat_ip_address
    master_internal_ip = yandex_compute_instance.cplane01.network_interface.0.ip_address
    workers            = yandex_compute_instance.workers
    ssh_priv_key       = local.ssh_priv_key
  })
}

# Ждёт готовности SSH на control-plane перед запуском Ansible
resource "null_resource" "wait_ssh_master" {
  depends_on = [yandex_compute_instance.cplane01]

  connection {
    type        = "ssh"
    user        = "ubuntu"
    host        = yandex_compute_instance.cplane01.network_interface.0.nat_ip_address
    private_key = file(local.ssh_priv_key)
    timeout     = "7m"
  }

  provisioner "remote-exec" {
    inline = ["echo check connection"]
  }
}

# Ждёт готовности SSH на каждой worker-ноде перед запуском Ansible
resource "null_resource" "wait_ssh_workers" {
  for_each   = yandex_compute_instance.workers
  depends_on = [yandex_compute_instance.workers]

  connection {
    type        = "ssh"
    user        = "ubuntu"
    host        = each.value.network_interface.0.nat_ip_address
    private_key = file(local.ssh_priv_key)
```

```
    timeout     = "7m"
  }

  provisioner "remote-exec" {
    inline = ["echo check connection"]
  }
}

# Устанавливает Ansible-коллекции
resource "null_resource" "ansible_collections" {
  provisioner "local-exec" {
    command = "ansible-galaxy collection install community.general ansible.posix"
  }
}

# Единый запуск playbook.yml сразу на все 4 ноды (master + workers)
resource "null_resource" "provision_cluster" {
  depends_on = [
    null_resource.ansible_collections,
    local_file.ansible_inventory,
    null_resource.wait_ssh_master,
    null_resource.wait_ssh_workers,
  ]

  provisioner "local-exec" {
    command = "ansible-playbook -i ${local_file.ansible_inventory.filename} playbook.yml"
  }
}

output "cplane01_internal_ip" {
  value = yandex_compute_instance.cplane01.network_interface.0.ip_address
}

output "cplane01_external_ip" {
  value = yandex_compute_instance.cplane01.network_interface.0.nat_ip_address
}

output "workers_ips" {
  value = {
```

```
    for k, v in yandex_compute_instance.workers : k => {
      internal = v.network_interface.0.ip_address
      external = v.network_interface.0.nat_ip_address
    }
  }
}
```

inventory.tpl

```
[kube-master]
kube-master ansible_host=${master_ip} internal_ip=${master_internal_ip}

[kube-node]
%{ for name, inst in workers ~}
${name} ansible_host=${inst.network_interface.0.nat_ip_address}
%{ endfor ~}

[k8s_cluster:children]
kube-master
kube-node

[k8s_cluster:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=${ssh_priv_key}
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Ansible ????-????????? VMs

playbook.yml

```
---

- name: Common OS / kubeadm / kubelet / kubectl prerequisites (все ноды)
  hosts: k8s_cluster
  become: true
  vars:
    k8s_minor_version: "1.37"
```

```
  tasks:
    - name: Common prerequisites
      ansible.builtin.import_tasks: common-prereqs.yml
  handlers:
    - name: Restart containerd
      ansible.builtin.systemd:
        name: containerd
        state: restarted

- name: Initialize control-plane and get join command
  hosts: kube-master
  become: true
  vars:
    pod_network_cidr: "10.244.0.0/16"
  tasks:
    - name: Check whether the cluster is already initialized
      ansible.builtin.stat:
        path: /etc/kubernetes/admin.conf
      register: admin_conf

    - name: kubeadm init
      ansible.builtin.command: >
        kubeadm init
        --apiserver-advertise-address={{ internal_ip }}
        --pod-network-cidr={{ pod_network_cidr }}
      when: not admin_conf.stat.exists

    - name: Create .kube directory for ubuntu user
      ansible.builtin.file:
        path: /home/ubuntu/.kube
        state: directory
        owner: ubuntu
        group: ubuntu
        mode: "0700"

    - name: Copy admin.conf into ubuntu kubeconfig
      ansible.builtin.copy:
        src: /etc/kubernetes/admin.conf
        dest: /home/ubuntu/.kube/config
```

```
        remote_src: true
        owner: ubuntu
        group: ubuntu
        mode: "0600"

    - name: Apply Flannel CNI manifest
      become_user: ubuntu
      environment:
        KUBECONFIG: /home/ubuntu/.kube/config
      ansible.builtin.command: >
        kubectl apply -f https://github.com/flannel-
io/flannel/releases/latest/download/kube-flannel.yml

    - name: Wait for control-plane node to become Ready
      become_user: ubuntu
      environment:
        KUBECONFIG: /home/ubuntu/.kube/config
      ansible.builtin.shell: >
        kubectl get nodes --no-headers | awk '{print $2}' | grep -q '^Ready$'
      register: node_ready
      retries: 20
      delay: 10
      until: node_ready.rc == 0
      changed_when: false

    - name: Getting join command
      ansible.builtin.shell: kubeadm token create --print-join-command
      register: k8s_token

- name: Setting kubernetes worker nodes
  hosts: kube-node
  become: true
  gather_facts: false
  tasks:
    - name: Check whether this node already joined the cluster
      ansible.builtin.stat:
        path: /etc/kubernetes/kubelet.conf
      register: kubelet_conf
```

```
    - name: Joining cluster
      ansible.builtin.shell: "{{ hostvars['kube-master']['k8s_token'].stdout }}"
      when: not kubelet_conf.stat.exists
```

common-prereqs.yml

```
- name: Disable swap
  ansible.builtin.command: swapoff -a
  changed_when: false

- name: Remove swap entry from /etc/fstab
  ansible.builtin.replace:
    path: /etc/fstab
    regexp: '^([^#].*\sswap\s.*)$'
    replace: '# \1'

- name: Load kernel modules (overlay, br_netfilter)
  community.general.modprobe:
    name: "{{ item }}"
    state: present
  loop:
    - overlay
    - br_netfilter

- name: Persist kernel modules across reboots
  ansible.builtin.copy:
    dest: /etc/modules-load.d/k8s.conf
    content: |
      overlay
      br_netfilter

- name: Set sysctl params required by Kubernetes networking
  ansible.posix.sysctl:
    name: "{{ item.name }}"
    value: "{{ item.value }}"
    sysctl_file: /etc/sysctl.d/k8s.conf
    reload: true
```

```
  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: Install containerd
  ansible.builtin.apt:
    name: containerd
    update_cache: true
    state: present

- name: Ensure /etc/containerd exists
  ansible.builtin.file:
    path: /etc/containerd
    state: directory

- name: Generate default containerd config (if not present)
  ansible.builtin.shell: containerd config default > /etc/containerd/config.toml
  args:
    creates: /etc/containerd/config.toml

- name: Enable SystemdCgroup in containerd config
  ansible.builtin.replace:
    path: /etc/containerd/config.toml
    regexp: "SystemdCgroup = false"
    replace: "SystemdCgroup = true"
  notify: Restart containerd

- name: Ensure containerd is started and enabled
  ansible.builtin.systemd:
    name: containerd
    state: started
    enabled: true

- name: Install packages required to add the Kubernetes apt repo
  ansible.builtin.apt:
    name:
      - apt-transport-https
```

```
      - ca-certificates
      - curl
      - gpg
    state: present

- name: Create /etc/apt/keyrings directory
  ansible.builtin.file:
    path: /etc/apt/keyrings
    state: directory
    mode: "0755"

- name: Download Kubernetes apt signing key
  ansible.builtin.get_url:
    url: "https://pkgs.k8s.io/core:/stable:/v{{ k8s_minor_version }}/deb/Release.key"
    dest: /tmp/k8s-release.key
    mode: "0644"

- name: Dearmor key into apt keyring
  ansible.builtin.shell: gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
/tmp/k8s-release.key
  args:
    creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg

- name: Add Kubernetes apt repository
  ansible.builtin.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_minor_version }}/deb/ /\n"

- name: Install kubelet, kubeadm, kubectl
  ansible.builtin.apt:
    name:
      - kubelet
      - kubeadm
      - kubectl
    update_cache: true
    state: present
```

```
- name: Hold kubelet/kubeadm/kubectl at installed version
  ansible.builtin.dpkg_selections:
    name: "{{ item }}"
    selection: hold
  loop:
    - kubelet
    - kubeadm
    - kubectl
```

?????? ????????? k8s ????????

- **Install kubeadm #1**
- **Install kubeadm #2**
- **Install kubectl**

Hypervisor	VM/Node	Role	IP
srv-01	vm-01	control-plane	192.168.1.10
srv-01	vm-02	worker	192.168.1.11
srv-02	vm-03	worker	192.168.1.12
srv-02	vm-04	worker	192.168.1.13

> Требуется полная сетевая связность между хостами-гипервизорами.

VM-01/02/03/04

????????? ??

Уникальные hostname / MAC / product_uuid

```
sudo hostnamectl set-hostname vm-01      # своё имя на каждой ноде
ip link                                  # проверка MAC
sudo cat /sys/class/dmi/id/product_uuid  # проверка uuid
```

Прописать `/etc/hosts` со всеми 4 IP+именами (или настроить DNS).

Отключение swap

```
sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

Включение Модулей ядра и sysctl

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
```

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
```

```
sudo sysctl --system
```

> **Диапазоны открытых портов:Control-plane:** 6443, 2379-2380, 10250, 10259, 10257
> **Workers:** 10250, 30000-32767

????????? containerd

```
sudo apt-get update
sudo apt-get install -y containerd

sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl enable containerd
```

????????? kubeadm, kubelet, kubectl

```
sudo apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.34/deb/Release.key | \
```

```
  sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.34/deb/ /' | \
  sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

> Замените `v1.34` на актуальную минорную версию (**Kubernetes Releases**)

VM-01

????????????? control-plane

```
sudo kubeadm init \
  --apiserver-advertise-address=192.168.1.10 \
  --pod-network-cidr=10.244.0.0/16
```

> Сохраните `token` из команды `kubeadm join ...`
> Повторная генерация: `kubeadm token create --print-join-command`

????????? kubectl

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

??????? ?????? CNI

```
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-
flannel.yml
```

`kubectl get nodes` — control-plane должна перейти в `Ready`

VM-02/03/04

??????????? workers

```
sudo kubeadm join 192.168.1.10:6443 --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>
```

TEST k8s ????????

> На **VM-01** controller-manager

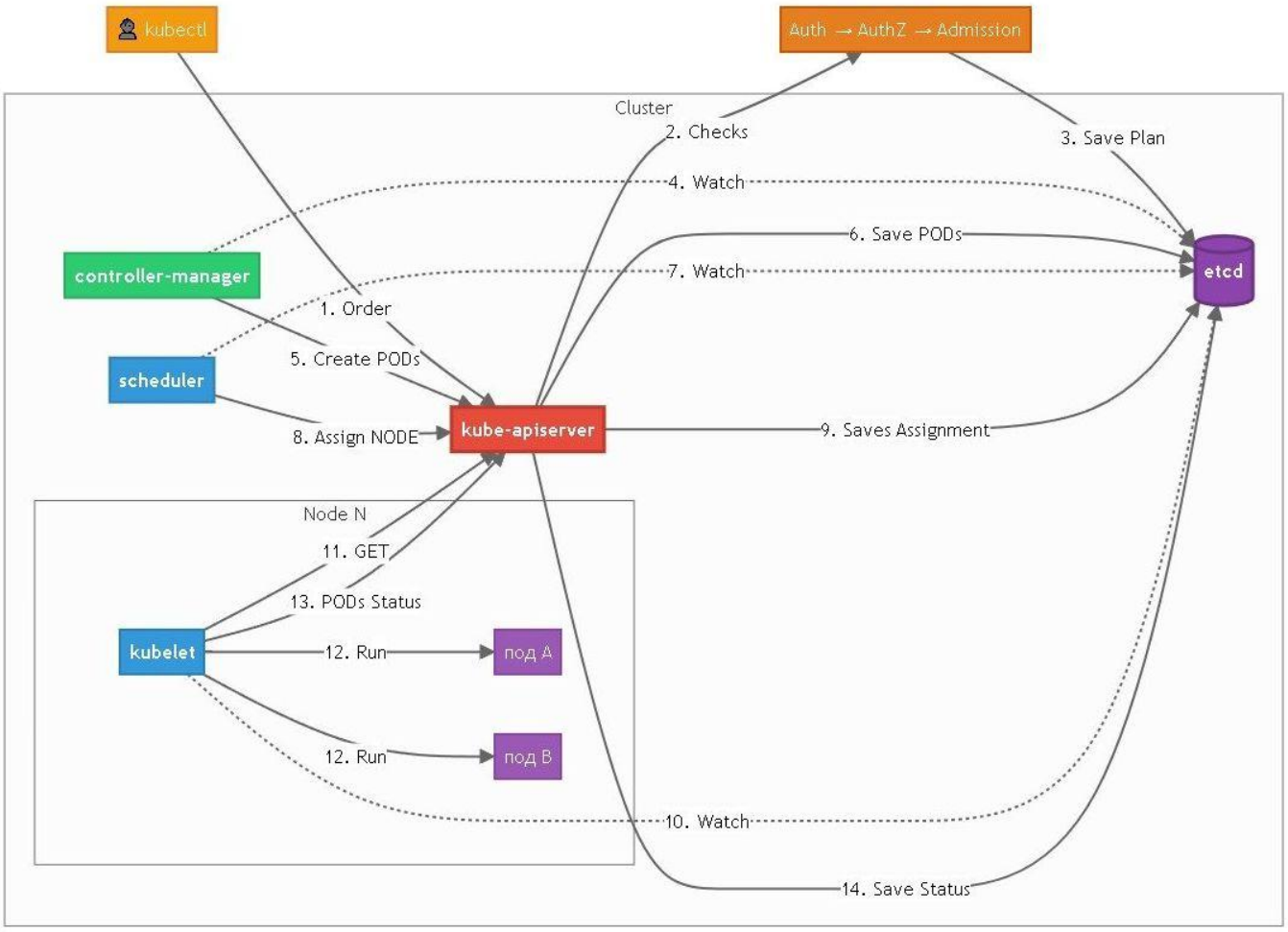
```
kubectl get nodes -o wide
kubectl get pods -A
```

> **Успешный результат**
> All NODEs — `Ready`
> System PODs (coredns, kube-proxy, CNI) — `Running`

Для управления с другой машины скопируйте `~/.kube/config` с `vm-01`

????? k8s

????? ??????? ?????? k8s


kubectl
Auth → AuthZ → Admission
Cluster
2. Checks
3. Save Plan
4. Watch
6. Save PODs
controller-manager
7. Watch
etcd
1. Order
5. Create PODs
scheduler
8. Assign NODE
kube-apiserver
9. Saves Assignment
Node N
11. GET
13. PODs Status
kubelet
12. Run
под A
12. Run
под B
10. Watch
14. Save Status