Southbridge

СЛЁРМ

Infrastructure as Code

Степаненко Алексей
stepanenko@selectel.ru
f1ex@inbox.ru

slurm.io

План

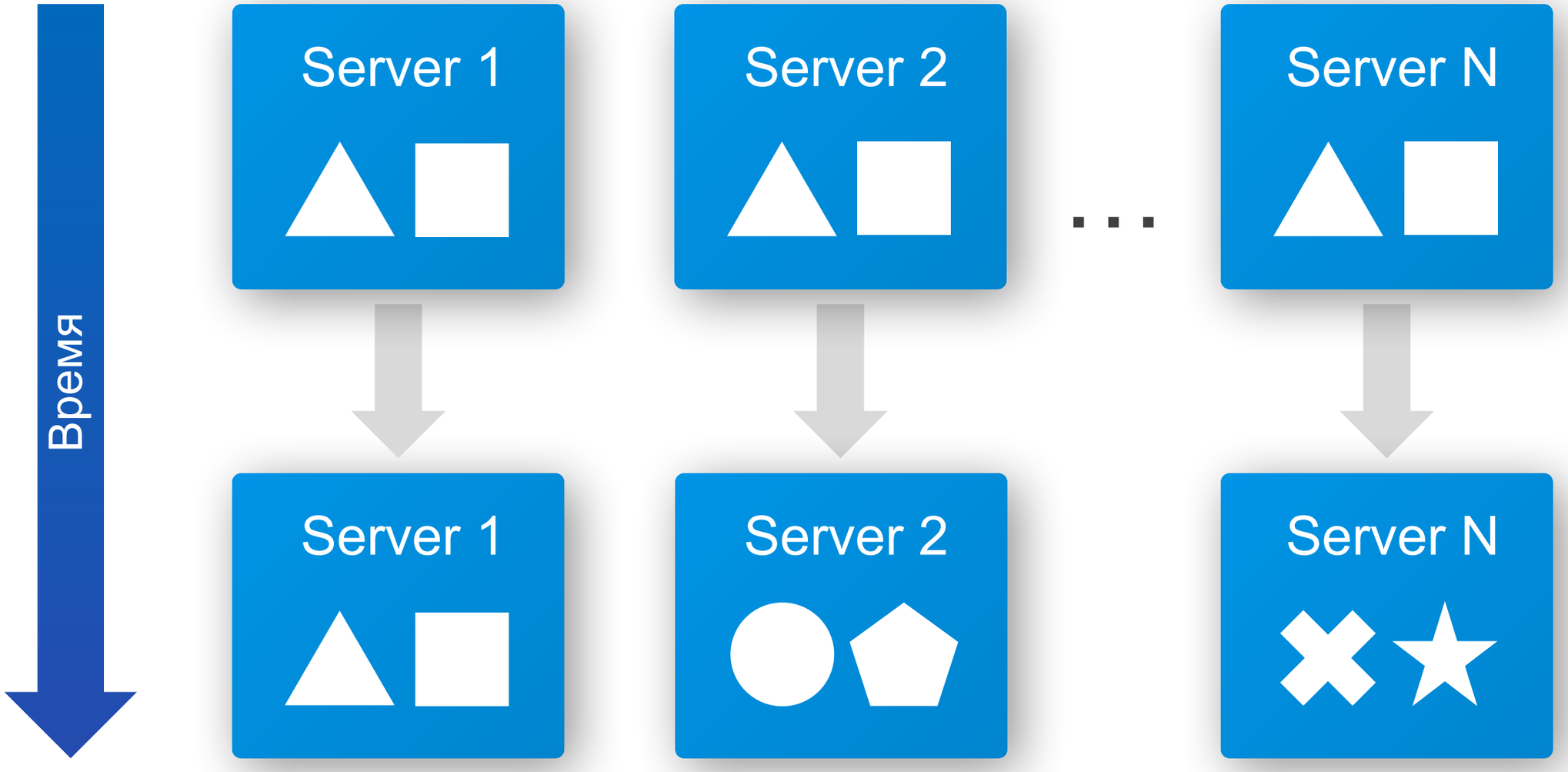
- Модели и методы управления инфраструктурой
- Работа с инфраструктурой как с кодом на примере Terraform
- Работа с образами на примере Packer
- Работа с Ansible

СЛЁРМ

Ручное конфигурирование

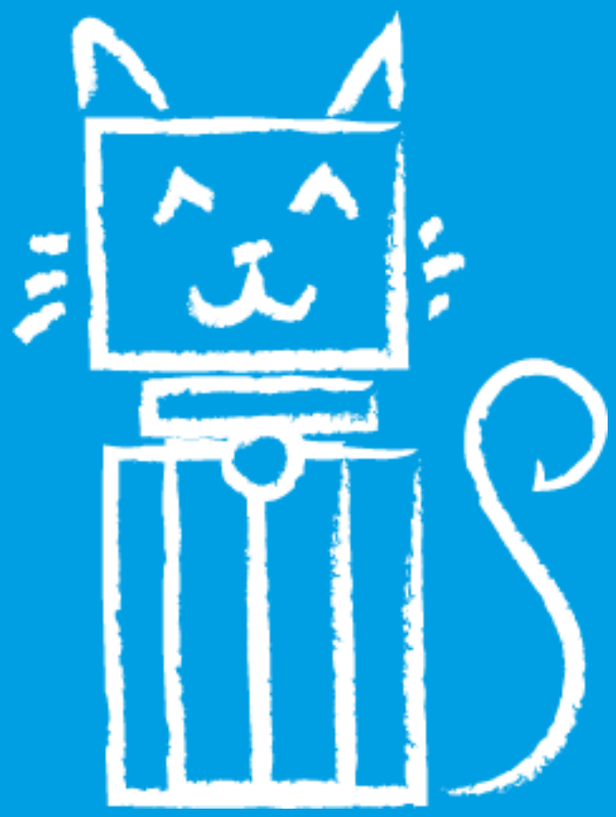
- Внесение ручных изменений
- Конфигурации серверов отличаются
- Environment может отличаться между dev, stage и prod
- Отсутствие информации о текущем состояние сервера и системы
- Тяжело работать в команде
- Нельзя повторить хост

Configuration Drift



PETS VS CATTLE

СНЕЖИНКИ И ФЕНИКС СЕРВЕРЫ



vs

pets

cattle

Серверы-питомцы

- У них есть имя
- Мы их холим и лелеем
- Лечим
- Уникальны
- Неповторимы

СЛЁРМ

Стадо серверов

- Нет специальных имен, есть порядковый номер или идентификатор
- Мы не лечим, а убиваем и заменяем другим
- Повторяют друг друга

VS

СЛЁРМ

Феникс-серверы

- Серверы, как Феникс, всегда восстают из пепла,

  т.е. всегда получаем исходную конфигурацию

- Нет configuration drift

- Окружение надежно, в отличие от использования

  уникальных серверов-снежинок

Модели инфраструктуры

СЛЁРМ

Подходы управления инфраструктурой

- Ручное
- Скрипты автоматизации (bash, perl, python etc)
- Практика Infrastructure as Code (IaC)
- Immutable Infrastructure
- Immutable Delivery

СЛЁРМ

Тестирование инфраструктуры

Код ревью, шаринг знаний

Infrastructure as Code (IaC)

Версионирование изменений

Применяем в администрировании практики из программирования

Масштабирование

Immutable Infrastructure

- Образ VM - артефакт для деплоя
- Изменения вносятся через билд нового образа и пересоздание ВМ
- Изменения в ВМ не вносятся

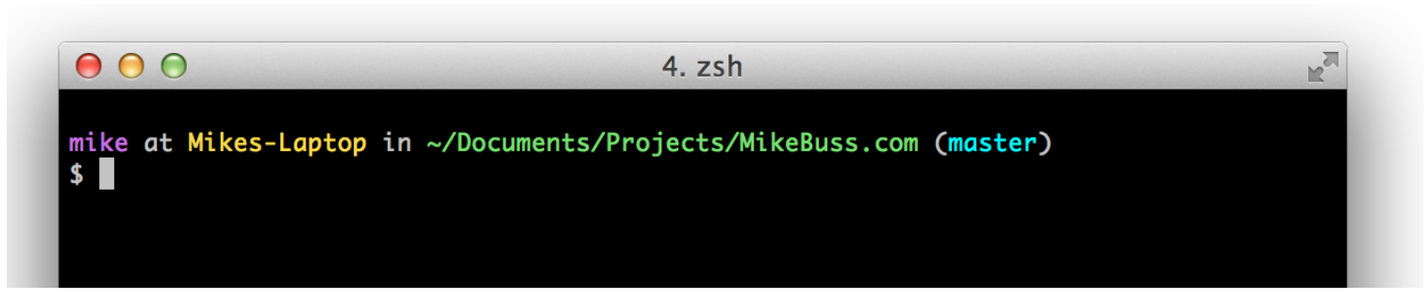
СЛЁРМ

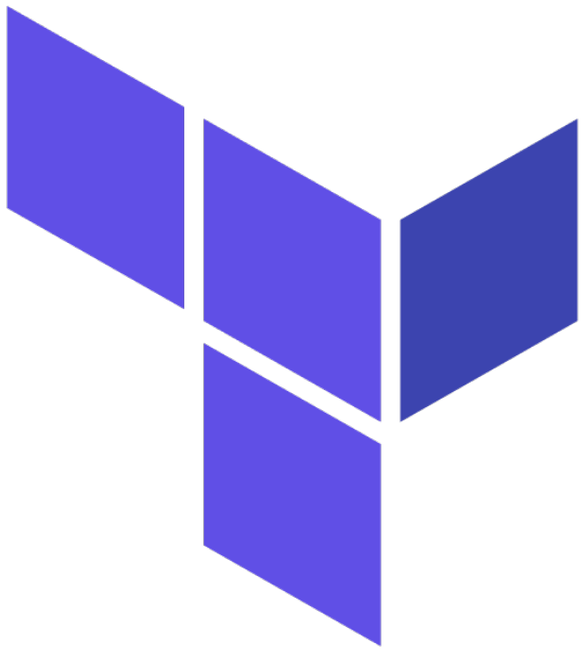
Immutable Delivery

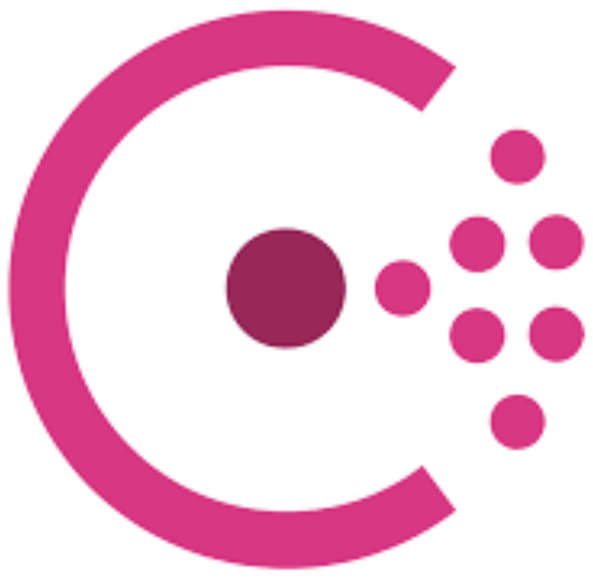
- Артефактом является приложение и ифраструктура
- Деплой приложения и инфраструктуры происходит одновременно

СЛЁРМ

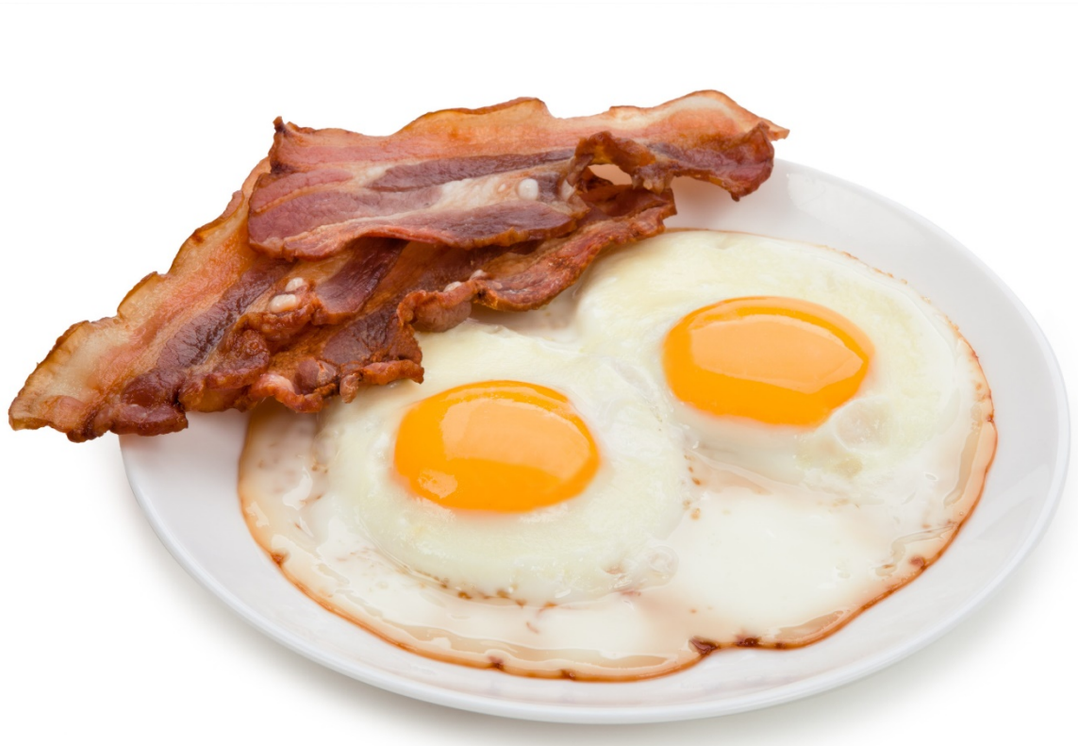
Инструменты







Baked vs Fried



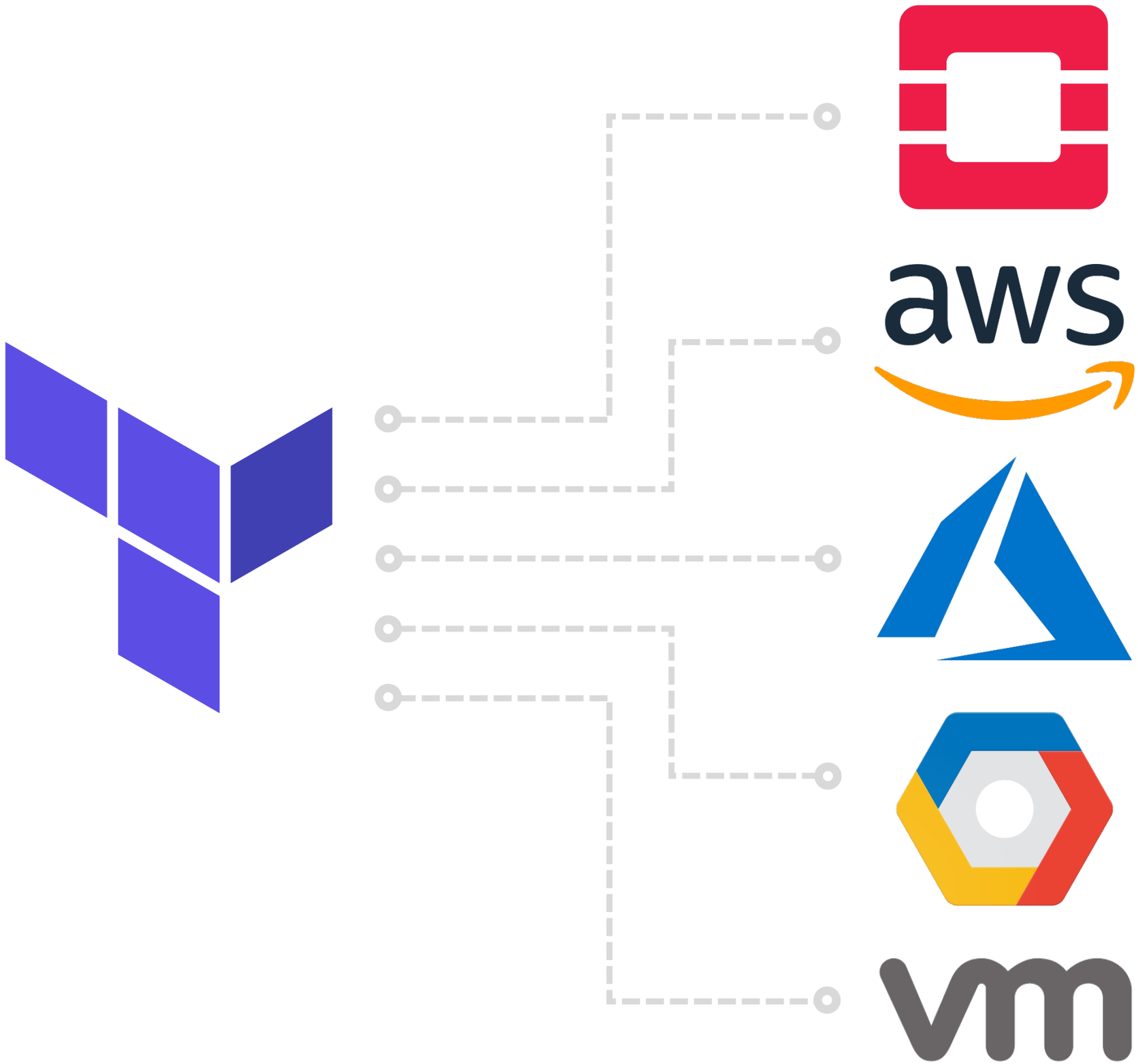
СЛЁРМ

СЛЁРМ

Практика

HashiCorp

Terraform



СЛЁРМ

	Source	Cloud	Type
Chef	Open	All	Config Mgmt
Puppet	Open	All	Config Mgmt
Ansible	Open	All	Config Mgmt
SaltStack	Open	All	Config Mgmt
CloudFormation	Closed	AWS	Orchestration
Heat	Open	All	Orchestration
Terraform	Open	All	Orchestration

```
#######################
# Provider
#######################
provider "google" {
  project = "{{YOUR GCP PROJECT}}"
  region  = "us-central1"
  zone    = "us-central1-c"
}

#######################
# Instance
#######################
resource "google_compute_instance" "vm_instance" {
  name         = "terraform-instance"
  machine_type = "f1-micro"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
    }
  }

  network_interface {
    # A default network is created for all GCP projects
    network       = "${google_compute_network.vpc_network.self_link}"
    access_config {
    }
  }
}
```

```
#####################################
# Configure the OpenStack Provider
#####################################
provider "openstack" {
  domain_name = "${var.domain_name}"
  tenant_id   = "${var.project_id}"
  user_name   = "${var.user_name}"
  password    = "${var.user_password}"
  auth_url    = "https://api.selvpc.ru/identity/v3"
  region      = "${var.region}"
}
#####################################
# Create Instance
#####################################
resource "openstack_compute_instance_v2" "instance_2" {
  name              = "www-node"
  flavor_id         = "${data.openstack_compute_flavor_v2.flavor_1.id}"
  key_pair          = "${openstack_compute_keypair_v2.terraform_key.id}"
  availability_zone = "${var.az_zone}"

  network {
    port = "${openstack_networking_port_v2.port_2.id}"
  }

  block_device {
    uuid             = "${openstack_blockstorage_volume_v3.volume_2.id}"
    source_type      = "volume"
    destination_type = "volume"
    boot_index       = 0
  }
}
```

Data

```
#####################################
# Flavor
#####################################
data "openstack_compute_flavor_v2" "flavor_1" {
  name = "SL1.1-1024"
}

#####################################
# Get image ID
#####################################
data "openstack_images_image_v2" "image_bastion" {
  most_recent = true
  visibility  = "private"
  tag         = "bastion-server"
}
```

Module

```
module "nodes" {
  source = "./modules/create_server"

  server_count = "${var.server_count}"

  network_id = "${openstack_networking_network_v2.network_1.id}"
  subnet_id  = "${openstack_networking_subnet_v2.subnet_1.id}"

  image_id    = "${data.openstack_images_image_v2.image_node.id}"
  region      = "${var.region}"
  az_zone     = "${var.az_zone}"
  volume_type = "${var.volume_type}"

  key_pair_id = "${openstack_compute_keypair_v2.terraform_key.id}"
  flavor_id   = "${data.openstack_compute_flavor_v2.flavor_1.id}"
}
```

```
modules
└── create_server
    ├── main.tf
    ├── output.tf
    └── vars.tf
```

СЛЁРМ

Output.tf

```
output "bastion_floatingip_address" {
  value = "${openstack_networking_floatingip_v2.floatingip_bastion.address}"
}
output "prometheus_dashboard" {
  value = "http://${openstack_networking_floatingip_v2.floatingip_bastion.address}
  :9090/targets"
}
```

Vars.tf

```
variable "user_name" {}

variable "user_password" {}

variable "server_count" {
  default = 2
}
```

СЛЁРМ

Secret.tfvars

```
user_name = "USERNAME"
user_password = "PASSWORD"
domain_name = "DOMAIN"
project_id = "PROJECT_ID"
public_key = "SSH PUBLIC KEY"
region = "РЕГИОН"
az_zone = "ЗОНА"
volume_type = "fast.ЗОНА"
```

СЛЁРМ

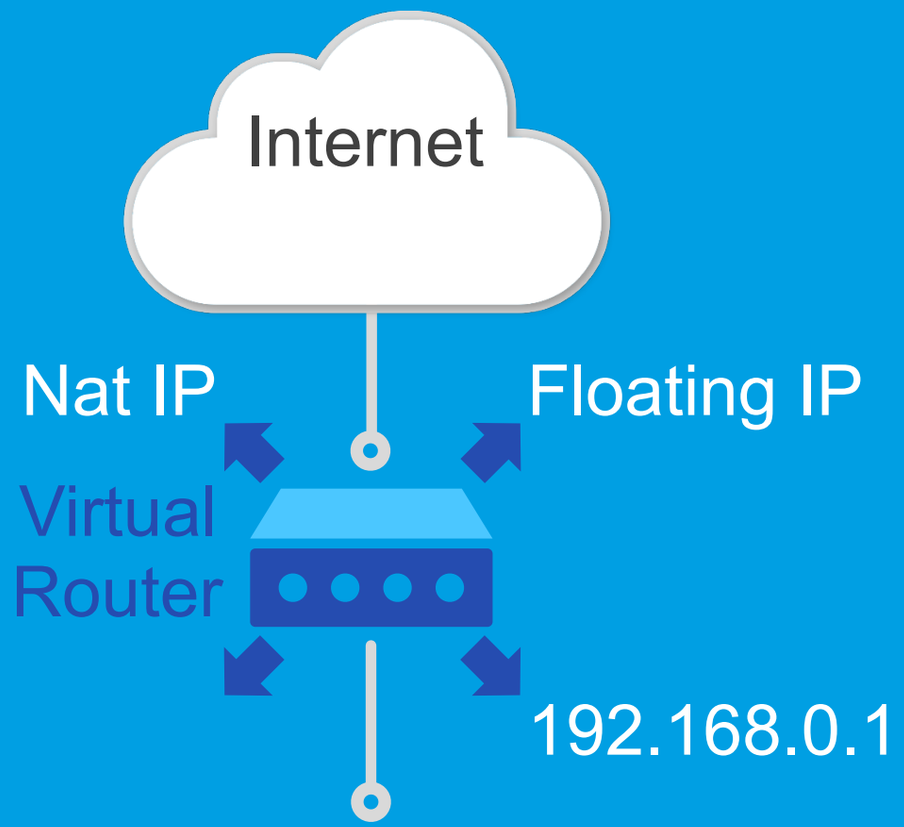
Count

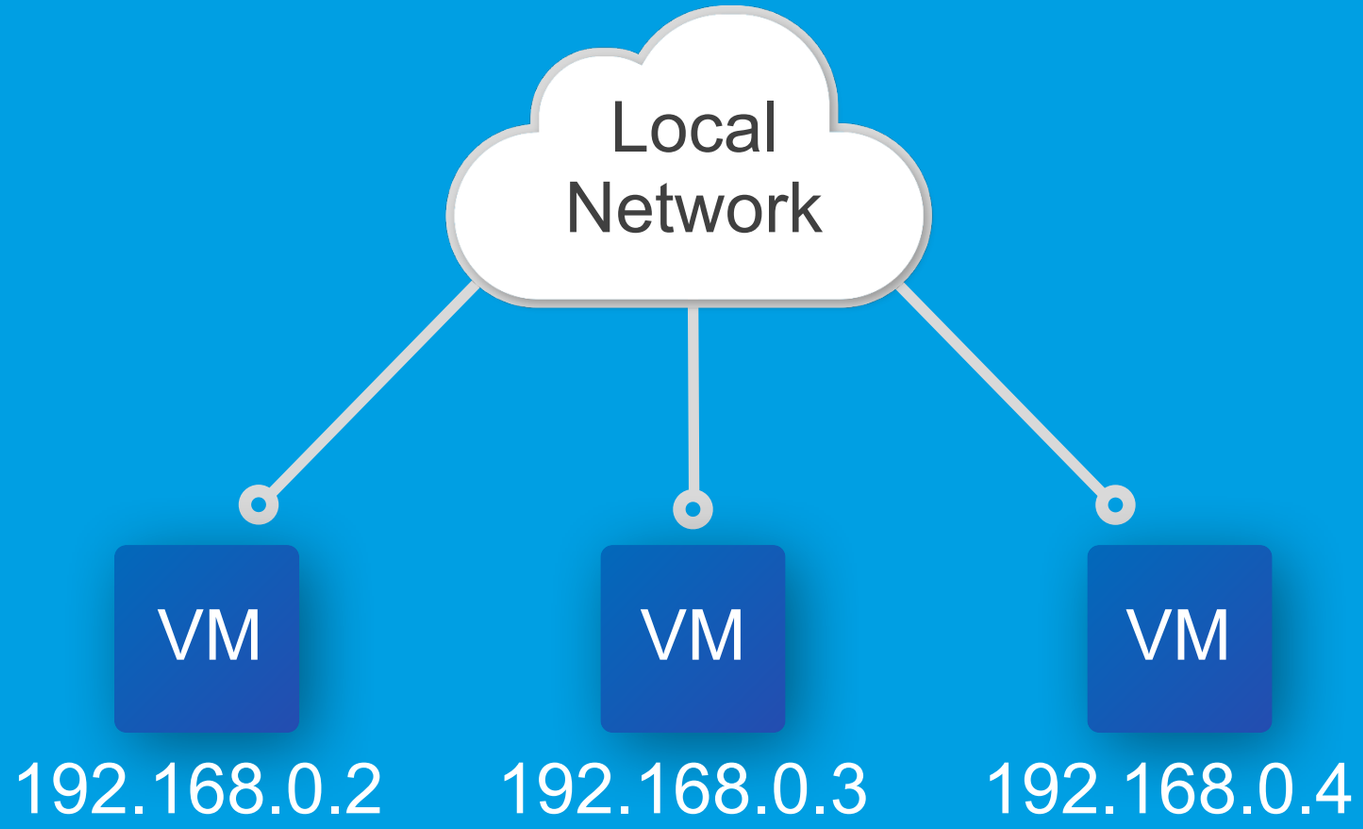
```
resource "openstack_networking_port_v2" "port" {
  count      = "${var.server_count}"
  name       = "node-${count.index}-eth0"
...
}

resource "openstack_blockstorage_volume_v3" "volume" {
  count              = "${var.server_count}"
  name               = "volume-for-node-${count.index}"
...
}

resource "openstack_compute_instance_v2" "node" {
  count              = "${var.server_count}"
  name               = "node-${count.index}"
...
  network {
    port = "${openstack_networking_port_v2.port[count.index].id}"
  }
  block_device {
    uuid             = "${openstack_blockstorage_volume_v3.volume[count.index].id}"
    ...
  }
...
}
```

СЛЁРМ

Основные команды

- terraform init
- terraform validate
- terraform fmt
- terraform plan
- terraform apply (-auto-approve)
- terraform destroy (-auto-approve)


Internet
Nat IP
Floating IP
Virtual
Router
192.168.0.1

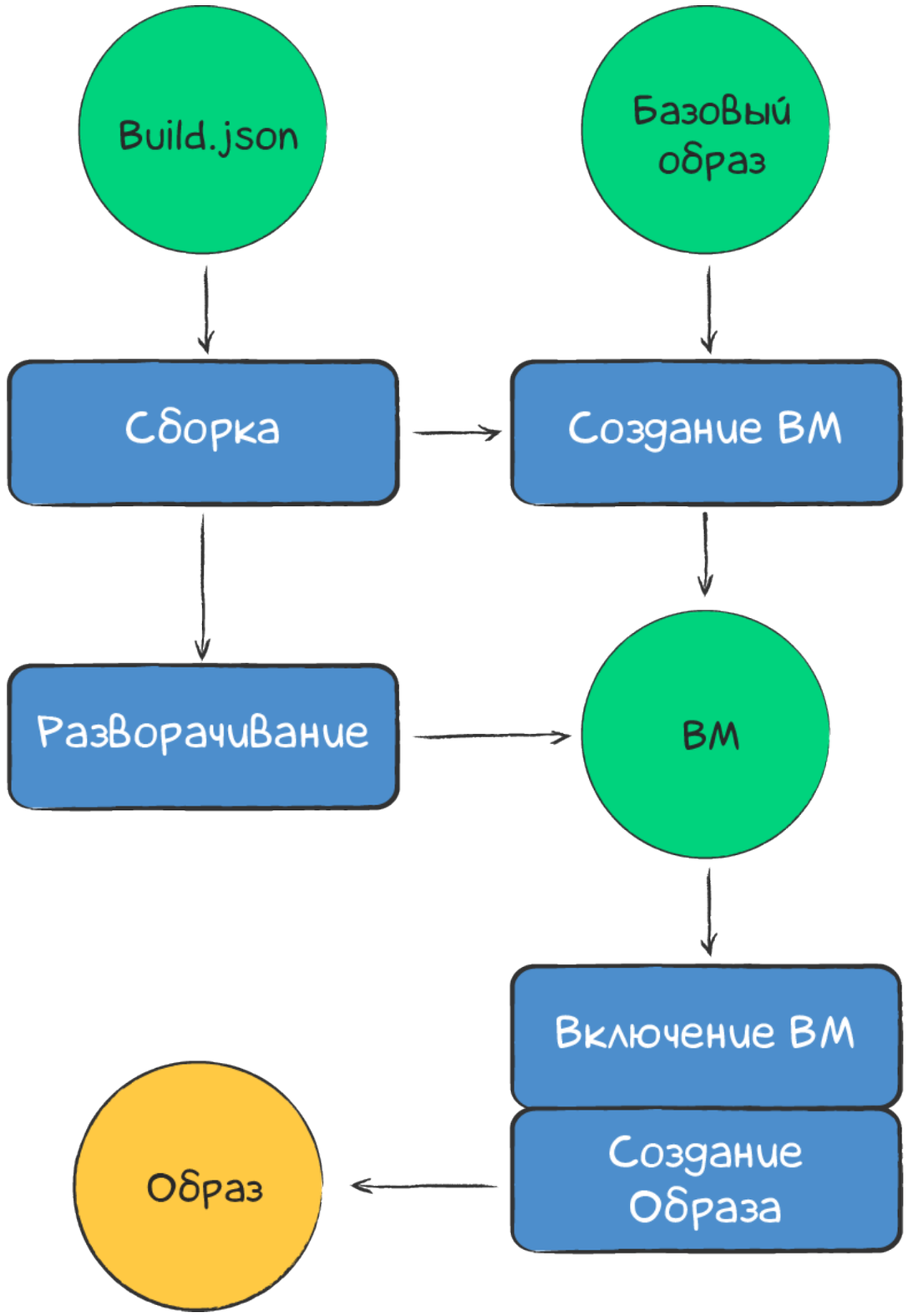

Local
Network
VM
VM
VM
192.168.0.2
192.168.0.3
192.168.0.4

СЛЁРМ

Практика

HashiCorp

Packer


Build.json
Базовый образ
Сборка
Создание ВМ
Развораччивание
ВМ
Включение ВМ
Создание Образа
Образ

Variables

```
"variables": {
    "region": null,
    "tenant_id": null,
    "domain_name": null,
    "username": null,
    "password": null,
    "networks": null,
    "availability_zone": null,
    "volume_type": null,
    "volume_size": "5"
},
"sensitive-variables": [
    "password"
],
```

СЛЁРМ

Builders

```
"builders": [
  {
    "name": "bastion-host",
    "type": "openstack",
    "identity_endpoint": "https://api.selvpc.ru/identity/v3",
    "region": "{{user `region`}}",
    "tenant_id": "{{user `tenant_id`}}",
    "domain_name": "{{user `domain_name`}}",
    "username": "{{user `username`}}",
    "password": "{{user `password`}}",
    "networks": "{{user `networks`}}",
    "availability_zone": "{{user `availability_zone`}}",
    "volume_type": "{{user `volume_type`}}",
    "volume_size": "{{user `volume_size`}}",
    "floating_ip_network": "external-network",
    "reuse_ips": true,
    "flavor": "BL1.1-1024",
    "ssh_username": "root",
    "image_name": "centos-bastion-packer-{{timestamp}}",
    "source_image_name": "CentOS 7 Minimal 64-bit",
    "use_floating_ip": true,
    "use_blockstorage_volume": true,
    "image_visibility": "private",
    "image_tags": ["bastion-server"]
  },
```

СЛЁРМ

Provisioners

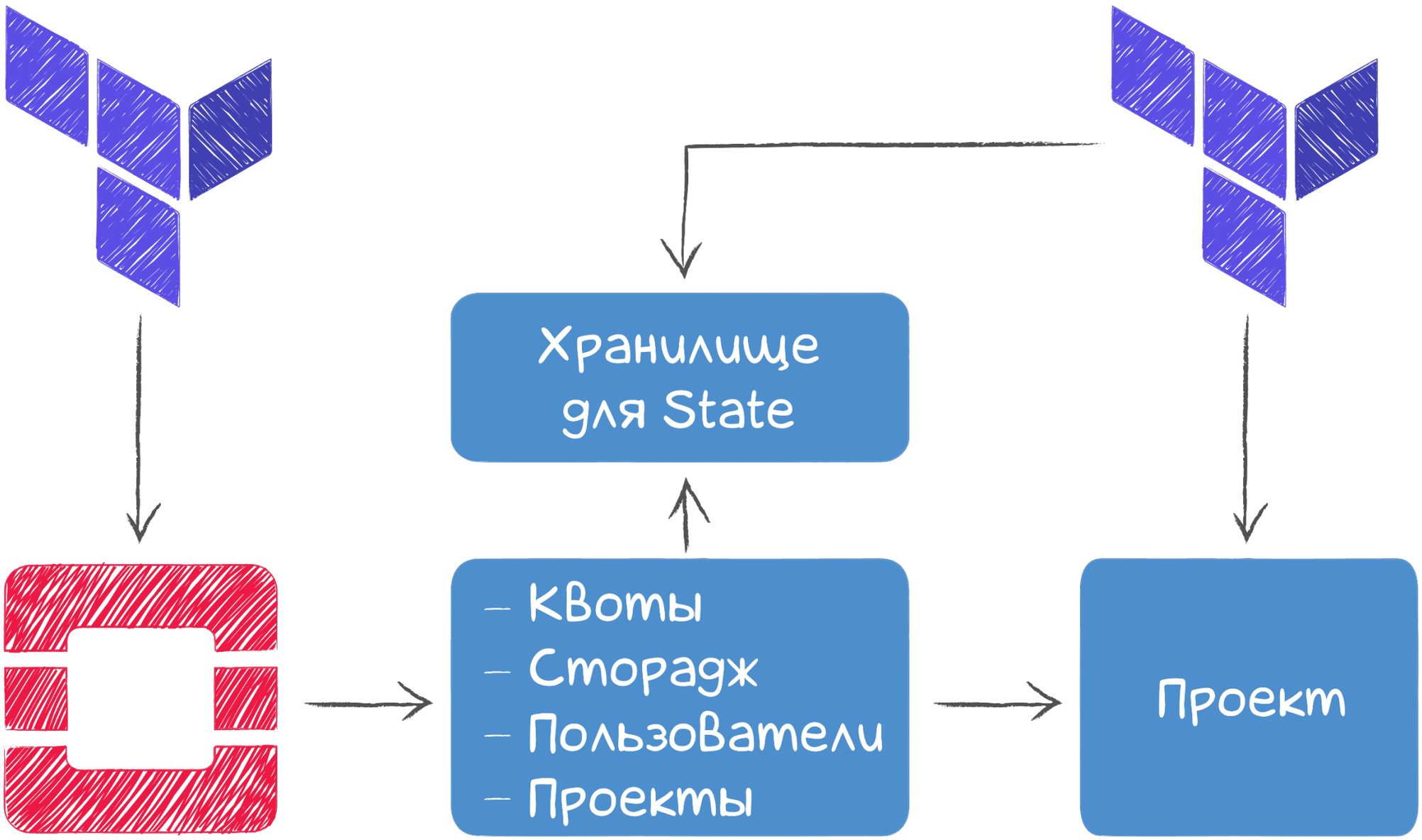
```
"provisioners": [
  {
    "type": "shell",
    "script": "setup_prometheus_server.sh",
    "only": [ "bastion-host" ]
  },
  {
    "type": "file",
    "source": "prometheus.yaml",
    "destination": "/etc/prometheus/prometheus.yaml",
    "only": [ "bastion-host" ]
  },
  {
    "type": "shell",
    "script": "setup_node_exporter.sh"
  },
```

Terraform

- Хранит состояние в state
- Внешние объекты нужно экспортировать или работать через data
- Модули для повторяемых операций
- Внешний сторадж для работы с командой (S3, PG, Consul)
- Output-переменные
- Различные Provisioners
- Источник данных об инфраструктуре для систем управления конфигурациями

HashiCorp

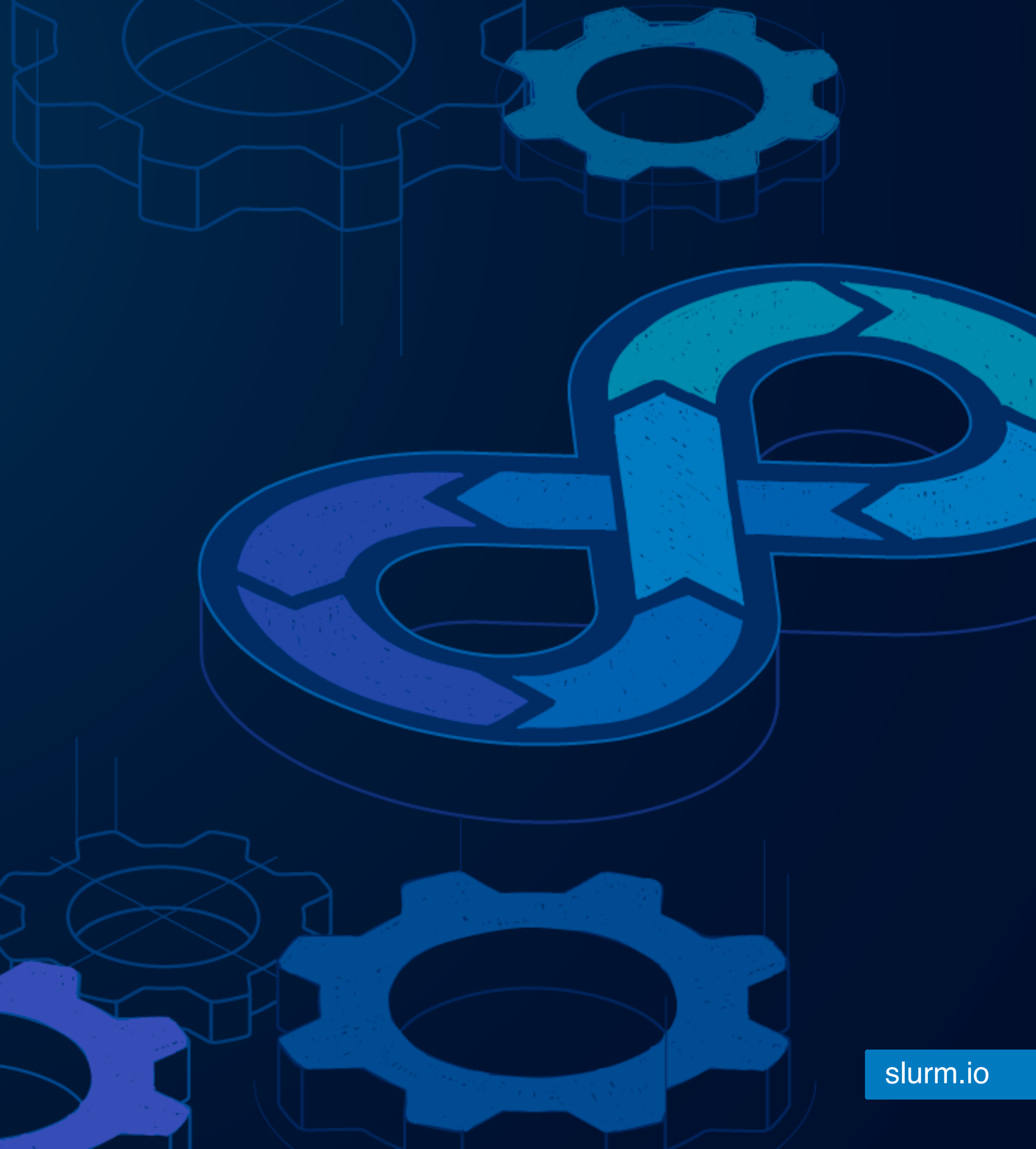
Terraform


Хранилище
для State
– Квоты
– Сторадж
– Пользователи
– Проекты
Проект

СЛЁРМ

Southbridge

СЛЕРМ



slurm.io