

# Описание инфраструктуры в виде кода (IaC). Знакомство с Terraform.

# План

- Инфраструктурная модель BSA: Base слой
- Практика Infrastructure as a Code
- Принципы правильного управления инфраструктурой
- Инфраструктурный репозиторий
- Основы работы Terraform

# Base-service-app модель

Layer

Responsibility

Application

Dev  
Ops\*

Service (IIS, MS SQL, MongoDB)

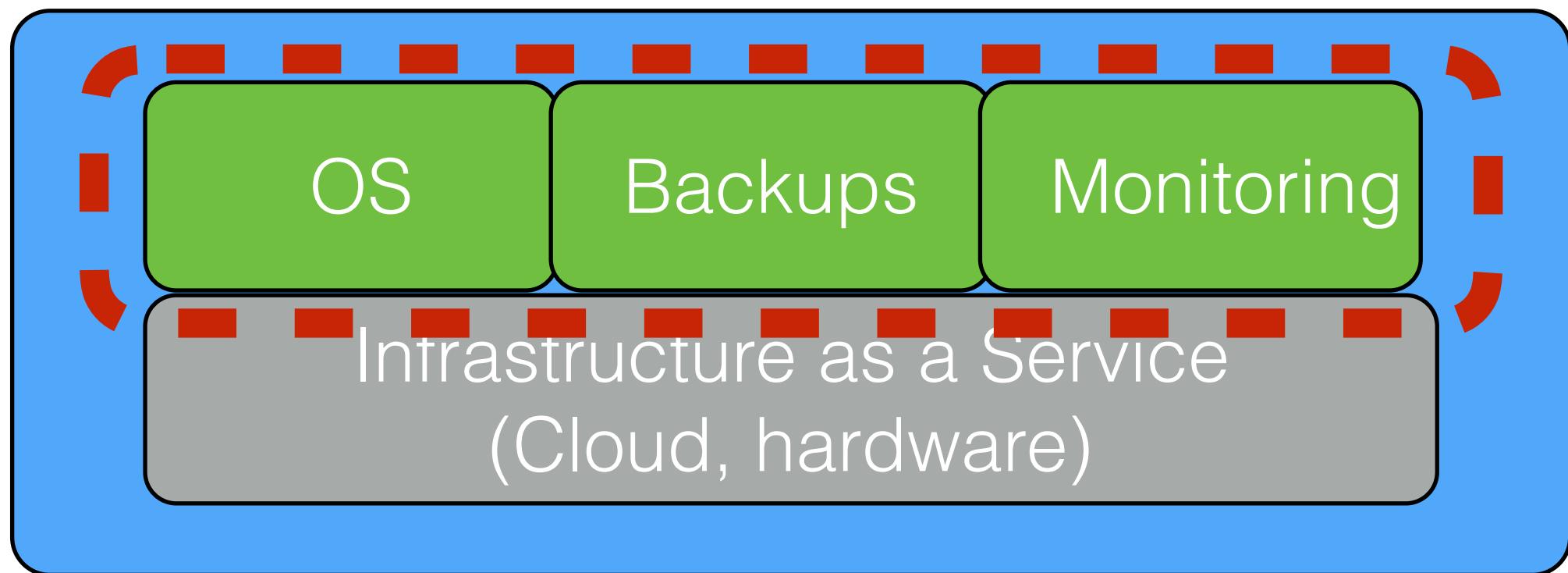
Ops

Base (Windows, Linux)

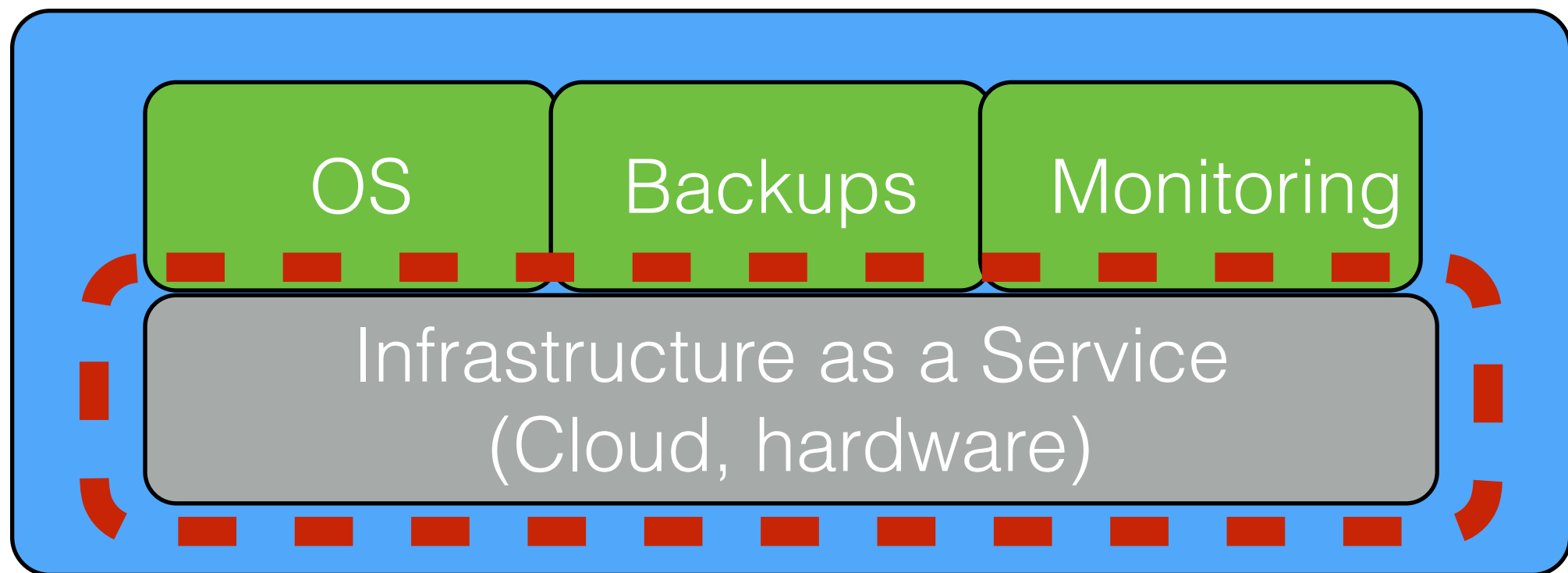
Ops

# Base слой в разрезе

# Системный уровень



# Инфраструктурный уровень



# Какие проблемы у нас все еще на базовом уровне?

- ...
- ...
- ...



# Методы управления

- Manual
- Scripts
- **Infrastructure as a Code (IaC)**
- Immutable Infrastructure
- Immutable Delivery





# Infrastructure as a Code

- Описание желаемого состояния инфраструктуры в виде кода
- Приведение состояние инфраструктуры в соответствие с описанием
- Готовые прототипы конфигурации
- Применимы практики из разработки: версионирование, ревью изменений, документирование

## Пример конфигурационного файла Terraform main.tf

```
provider "google" {  
  project = "infra-179014"  
  region  = "europe-west1"  
}  
  
resource "google_compute_instance" "app" {  
  name          = "reddit-app"  
  machine_type  = "g1-small"  
  zone          = "europe-west1-b"  
  boot_disk {  
    initialize_params {  
      image = "reddit-base-1504639663"  
    }  
  }  
  network_interface {  
    network = "default"  
    access_config {}  
  }  
}
```

# Что дает?

- Не нужно придумывать свои велосипеды в виде скриптов
- Возможность долгосрочного управления инфраструктурой через код
- Версионирование и контроль изменений
- Самодокументация
- Автоматизация
- Воспроизводимость текущей конфигурации



# Принципы управления инфраструктурой

- Описывается в виде кода (IaC)
- Любое изменение через код
- Инфра код хранится в VCS
- Повторяемость
- Документация (самодокументация)
- Командная работа



# Infra репозиторий

- Содержит описание всей инфраструктуры проекта в виде кода
- Версионирование конфигурации инфраструктуры
- Инфраструктурные изменения производятся через PR (достаточно получить один апрув для мерджа)
- Документация команд и структуры репозитория (README.md)
- Повторяемость и переиспользуемость



# Основы работы с Terraform

# Terraform

- Инструмент для декларативного описания инфраструктуры
- Описание инфраструктурных компонент на языке HCL
- Описание инфраструктуры хранится в конфигурационных файлах (имеют расширение **.tf**)

# Пример конфигурационного файла *main.tf*

```
provider "google" {
  credentials = "${file("account.json")}"
  project     = "infra-14367"
  region      = "europe-west1"
}

resource "google_compute_instance" "app" {
  name          = "reddit-app"
  machine_type  = "g1-small"
  zone          = "europe-west1-b"
  # определение загрузочного диска
  boot_disk {
    initialize_params {
      image = "reddit-base-1504639663"
    }
  }
  network_interface {
    network = "default"
    access_config {}
  }
}
```



# Providers

- Содержат настройки аутентификации и подключения к платформе или сервису
- Предоставляют набор ресурсов для управления
- Являются загружаемыми модулями с версии 0.10.0
- Поддержка большого количества провайдеров: AWS, Google Cloud, GitHub, PostgreSQL, etc.



*./provider.tf*

```
provider "google" {  
    credentials = "${file("account.json")}"  
    project     = "infra-14367"  
    region      = "europe-west1"  
}
```


Установка провайдера производится командой  
terraform init

# Resources

- Определяются типом провайдера
- Позволяют управлять компонентами платформы или сервиса
- Могут иметь обязательные и необязательные аргументы
- Могут ссылаться на другие ресурсы

Тип ресурса

Имя ресурса



```
resource "google_compute_instance" "app" {  
  name          = "reddit-app"  
  machine_type  = "g1-small"  
  zone          = "europe-west1-b"  
  # определение загрузочного диска  
  boot_disk {  
    initialize_params {  
      image = "reddit-base-1504639663"  
    }  
  }  
}
```

Комбинация тип ресурса + имя уникально идентифицирует ресурс в рамках данной конфигурации

# Применение описанной конфигурации

Для приведение системы в описанное состояние используется команда:

```
$ terraform apply
```

# Что будет если применить команду дважды?

# Что будет если применить команду дважды?

```
$ terraform apply
```

```
google_compute_firewall.firewall_puma: Refreshing state... (ID: allow-puma-default)  
google_compute_instance.app: Refreshing state... (ID: reddit-app)  
google_compute_project_metadata_item.appuser_ssh_key: Refreshing state...  
(ID: sshKeys)
```

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Второго инстанса не будет создано.

# State файлы

- Terraform хранит информацию об управляемых ресурсах в terraform.tfstate
- terraform.tfstate обновляется при каждом apply или refresh
- terraform.tfstate.backup используется для бекапа предыдущего terraform.state
- По умолчанию сохраняются локально в папке с конфигурацией



# ./terraform.tfstate

```
"resources": {  
  "google_compute_firewall.firewall_puma": {  
    "type": "google_compute_firewall",  
    "depends_on": [],  
    "primary": {  
      "id": "allow-puma-default",  
      "attributes": {  
        "allow.#": "1",  
        "allow.931060522.ports.#": "1",  
        "allow.931060522.ports.0": "9292",  
        "allow.931060522.protocol": "tcp",  
        "deny.#": "0",  
        "description": "",  
        "destination_ranges.#": "0",  
        "id": "allow-puma-default",  
        "name": "allow-puma-default",  
        "network": "default",  
        "project": "infra-179014",  
        "self_link": "https://www.googleapis.com/compute/v1/projects/infra-179014/global/firewalls/allow-puma-default",  
        "source_ranges.#": "1",  
        "source_ranges.1080289494": "0.0.0.0/0",  
        "source_tags.#": "0",  
        "target_tags.#": "1",  
        "target_tags.1799682348": "reddit-app"
```

# ./terraform.tfstate

```
"google_compute_instance.app": {  
  "type": "google_compute_instance",  
  "depends_on": [],  
  "primary": {  
    "id": "reddit-app",  
    "attributes": {  
      "boot_disk.0.initialize_params.#": "1",  
      "boot_disk.0.initialize_params.0.image": "reddit-base",  
      "boot_disk.0.initialize_params.0.size": "0",  
      "boot_disk.0.initialize_params.0.type": "",  
      "boot_disk.0.source": "reddit-app",  
      "can_ip_forward": "false",  
      "create_timeout": "4",  
      "disk.#": "0",  
      "id": "reddit-app",  
      "label_fingerprint": "42WmSpB8rSM=",  
      "machine_type": "g1-small",  
      "metadata.%": "0",  
      "metadata_fingerprint": "l2jrYHRVThk=",  
      "name": "reddit-app",  
      "network.#": "0",  
      "network_interface.#": "1",  
      "network_interface.0.access_config.#": "1",  
      "network_interface.0.access_config.0.assigned_nat_ip": "104.199.5.22",  
      "network_interface.0.access_config.0.nat_ip": "",  
      "network_interface.0.address": "10.132.0.2",
```

# Планирование изменений

```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
```

```
...
~ google_compute_instance.app[0]
  tags.#: "1" => "0"
  tags.1799682348: "reddit-app" => ""

+ google_compute_instance.app[1]
  boot_disk.#: "1"
  boot_disk.0.auto_delete: "true"

...
- google_compute_project_metadata_item.appuser_ssh_key
```

**Plan:** 1 to add, 1 to change, 1 to destroy



# Input переменные

- Позволяют параметризовать конфигурационные файлы
- Три типа: `string`, `map`, `list`
- Можно передать из файла, из командной строки или интерактивно



# Объявление input переменных

*variables.tf*

```
variable project {  
    description = "Project ID"  
}  
variable region {  
    description = "Region"  
    default = "europe-west1"  
}
```



# Использование в конфигах

*main.tf*

```
provider "google" {  
  project = "${var.project}"  
  region  = "${var.region}"  
}
```

# Задание переменных через файл

*my-vars.tfvars*

```
project = "infra-179015"  
public_key_path = "~/.ssh/appuser.pub"
```

Указываем путь до файла при запуске команд terraform-a:

```
$ terraform apply -var-file=my-vars.tfvars
```

Если файл называется terraform.tfvars, то переменные загружаются автоматически.

# Получение информации об инфраструктуре



# terraform.tfstate

```
"google_compute_instance.app": {
  "type": "google_compute_instance",
  "depends_on": [],
  "primary": {
    "id": "reddit-app",
  "attributes": {
    "boot_disk.0.initialize_params.#": "1",
    "boot_disk.0.initialize_params.0.image": "reddit-base",
    "boot_disk.0.initialize_params.0.size": "0",
    "boot_disk.0.initialize_params.0.type": "",
    "boot_disk.0.source": "reddit-app",
    "can_ip_forward": "false",
    "create_timeout": "4",
    "disk.#": "0",
    "id": "reddit-app",
    "label_fingerprint": "42WmSpB8rSM=",
    "machine_type": "g1-small",
    "metadata.%": "0",
    "metadata_fingerprint": "l2jrYHRVThk=",
    "name": "reddit-app",
    "network.#": "0",
    "network_interface.#": "1",
    "network_interface.0.access_config.#": "1",
  "network_interface.0.access_config.0.assigned_nat_ip": "104.199.5.22",
    "network_interface.0.access_config.0.nat_ip": "",
    "network_interface.0.address": "10.132.0.2",
```

# terraform show

Искать нужные атрибуты ресурсов по state файлу не очень удобно, поэтому terraform предоставляет команду show для чтения стейт файла:

```
$ terraform show
```

```
google_compute_instance.app:
```

```
...
```

```
  name = reddit-app
```

```
  network.# = 0
```

```
  network_interface.# = 1
```

```
  network_interface.0.access_config.# = 1
```

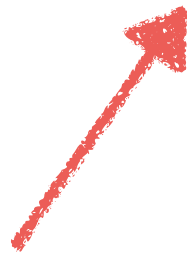
```
  network_interface.0.access_config.0.assigned_nat_ip =  
104.199.54.241
```

# Output переменные

- Позволяют сохранить выходные значения после создания ресурсов.
- Облегчают процедуру поиска нужных данных
- Используются в модулях как входные переменные для других модулей

*outputs.tf*

```
output "app_external_ip" {  
  value = "${google_compute_instance.app.network_interface.0.access_config.0.assigned_nat_ip}"  
}
```



Идентифицируем  
ресурс, указывая его  
тип и имя



Указываем нужные  
атрибуты ресурса

# Посмотреть значение output переменной

```
$ terraform output  
app_external_ip = 104.199.54.241
```



# Примеры встроенных функций

- `file(path)`
- `basename(path)`
- `join(delimiter, list)`
- `length(list)`
- ВЕСЬ СПИСОК

# Базовая организация конфигурационных файлов

- `main.tf` - основная конфигурация
- `variables.tf` - объявление input переменных
- `outputs.tf` - определение выходных переменных
- `terraform.tfvars.example`

# To be continued ...

