Terraform: ресурсы, модули и работа в команде

План

- Декларативное vs процедурное описание
- Взаимозависимости ресурсов
- Data sources
- Remote state
- Modules

Процедурное описание

- Описываем действия по достижению желаемого состояния
- Желаемое состояние достижимо, но сильно зависит от метода использования инструмента
- Отвечает на вопрос "как и что сделать?"

Создание ресурсов

```
$ gcloud compute instances create example-instance-1 example-instance-2 --
machine-type f1-micro --zone europe-west1-b --image-family ubuntu-1604-lts --
image-project ubuntu-os-cloud

NAME              ZONE            MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP
EXTERNAL_IP    STATUS
example-instance-1  europe-west1-b  f1-micro                   10.132.0.3
35.195.239.50  RUNNING
example-instance-2  europe-west1-b  f1-micro                   10.132.0.2
35.187.163.47  RUNNING
```

Повторное применение команды

```
$ gcloud compute instances create example-instance-1 example-instance-2 --
machine-type f1-micro --zone europe-west1-b --image-family ubuntu-1604-lts --
image-project ubuntu-os-cloud

ERROR: (gcloud.compute.instances.create) Could not fetch resource:
 - The resource 'projects/infra-179014/zones/europe-west1-b/instances/example-
instance-1' already exists
 - The resource 'projects/infra-179014/zones/europe-west1-b/instances/example-
instance-2' already exists
```

Удаление

```
$ gcloud compute instances delete example-instance-2

Do you want to continue (Y/n)?  y
```

Декларативное описание

- Описываем желаемое состояние
- Инструмент отвечает за приведение инфраструктуры в соответствие с описанием
- Отвечает на вопрос "что должно быть?", а не "как сделать?"

Создание ресурсов

main.tf

```
resource "google_compute_instance" "app" {
  name         = "example-instance-${count.index + 1}"
  machine_type = "f1-micro"
  zone         = "europe-west1-b"
  count = 2
  boot_disk {
    initialize_params {
      image = "ubuntu-1604-lts"
    }
  }
  network_interface {
    network       = "default"
    access_config = {}
  }
}
```

Создание ресурсов

```
$ terraform apply
google_compute_instance.app.0: Creation complete after 16s (ID: example-instance-1)
google_compute_instance.app.1: Creation complete after 16s (ID: example-instance-2)

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Повторное применение команды:

```
$ terraform apply
google_compute_instance.app.1: Refreshing state... (ID: example-instance-2)
google_compute_instance.app.0: Refreshing state... (ID: example-instance-1)

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

Удаление ресурса

```
resource "google_compute_instance" "app" {
  name         = "example-instance-${count.index + 1}"
  machine_type = "f1-micro"
  zone         = "europe-west1-b"
  count = 1
...
```

Говорим инструменту привести в желаемое состояние:

```
$ terraform apply

google_compute_instance.app.1: Destruction complete after 46s

Apply complete! Resources: 0 added, 0 changed, 1 destroyed.
```

Взаимосвязи ресурсов

Ресурсы независимы

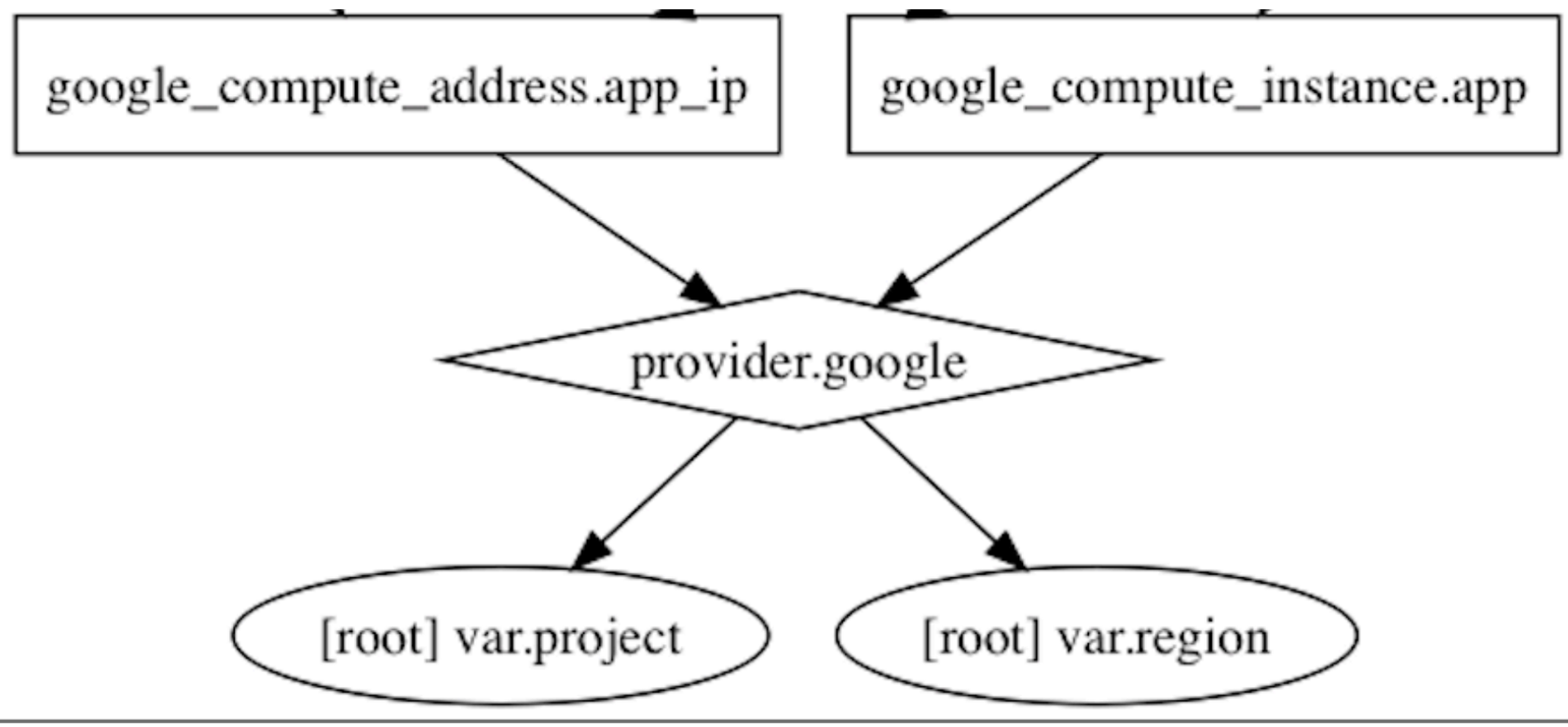
main.tf

```
resource "google_compute_instance" "app" {
  name         = "instance-1"
  machine_type = "f1-micro"
  zone         = "europe-west1-b"
  boot_disk {
    initialize_params {
      image = "ubuntu-1604-lts"
    }
  }
  network_interface {
    network       = "default"
    access_config = {

     }
  }
}
resource "google_compute_address" "app_ip" {
    name = "example-ip"
}
```

Ресурсы независимы

```
$ terraform graph | dot -Tpng > graph.png
```



Независимые ресурсы создаются одновременно

```
$ terraform apply
google_compute_address.app_ip: Creating...
  address:   "" => "<computed>"
  name:      "" => "example-ip"
  self_link: "" => "<computed>"
google_compute_instance.app: Creating...
  boot_disk.#:

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Неявная зависимость

main.tf

```
resource "google_compute_instance" "app" {
  name         = "instance-1"
  machine_type = "f1-micro"
  zone         = "europe-west1-b"
  boot_disk {
    initialize_params {
      image = "ubuntu-1604-lts"
    }
  }
  network_interface {
    network       = "default"
    access_config = {
      nat_ip = "${google_compute_address.app_ip.address}"
    }
  }
}
resource "google_compute_address" "app_ip" {
    name = "example-ip"
}
```

Ссылка на атрибут другого ресурса

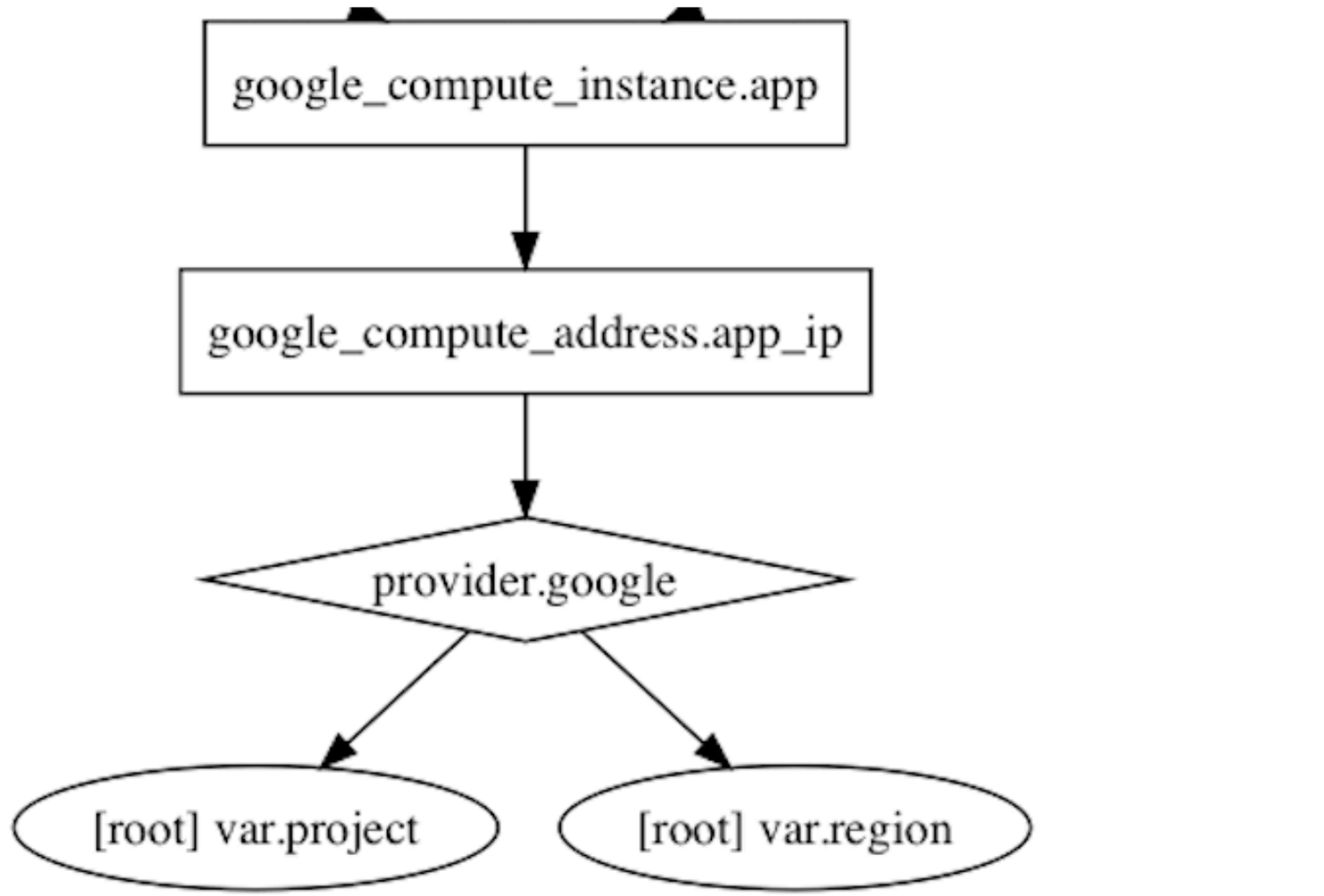
Ссылка на атрибут ресурса

```
"${google_compute_address.app_ip.address}"
```

Тип ресурса — Имя — Атрибут

Неявная зависимость

```
$ terraform graph | dot -Tpng > graph.png
```



Зависимые ресурсы создаются в соответствии с зависимостями:

```
$ terraform apply
```

```
google_compute_address.app_ip: Creating...
  address:    "" => "<computed>"
  name:       "" => "example-ip"
  self_link:  "" => "<computed>"
google_compute_address.app_ip: Creation complete after 13s (ID: example-ip)
google_compute_instance.app: Creating...
  boot_disk.#:
```

Зависимые ресурсы удаляются в соответствии с зависимостями:

```
$ terraform destroy

google_compute_instance.app: Destroying... (ID: instance-1)
google_compute_instance.app: Still destroying... (ID: instance-1, 10s elapse
google_compute_instance.app: Still destroying... (ID: instance-1, 20s elapse
google_compute_instance.app: Still destroying... (ID: instance-1, 30s elapse
google_compute_instance.app: Destruction complete after 36s
google_compute_address.app_ip: Destroying... (ID: example-ip)
```

Data source

```
data "aws_ami" "image" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64*"]
  }
}

module "web" {
  ami             = "${data.aws_ami.image.id}"
  source          = "modules/web"
  pub_key_path    = "${var.pub_key_path}"
```

Управление стейтом и работа в команде

State хранит информацию о соответсвии конфигурации Terraform реальным инфраструктурным единицам

Имя тераформ ресурса

Информация о реальном инстансе VM


```
"google_compute_instance.app": {
                    "type": "google_compute_instance",
                    "depends_on": [
                        "google_compute_address.app_ip"
                    ],
                    "primary": {
                        "id": "instance-1",
                        "attributes": {
                            "boot_disk.0.initialize_params.0.image": "ubuntu-1604-lts",
                            "disk.#": "0",
                            "id": "instance-1",
                            "machine_type": "f1-micro",
                            "name": "instance-1",
                            "network_interface.0.access_config.0.assigned_nat_ip": "35.195.194.15",
                            "network_interface.0.access_config.0.nat_ip": "35.195.194.15",
                            "network_interface.0.address": "10.132.0.2",
```

Проблемы управления стейтом

- Доступность всем членам команды
- Блокировка стейт файла
- Разделение по окружениям

Проблемы хранения в гите

- Небезопасно
- Забываем обновлять

Remote backends

- Централизованное хранение стейта
- Автоматическое обновление при каждом изменении
- Поддержка блокирования стейта

Пример

backend.tf

```
terraform {
  backend "s3" {
    bucket = "mk-aws"
    key = "artemkin/remote_state"
    region = "eu-central-1"
    dynamodb_table = "terraform-state-lock"
  }
}
```

Инициализация нового бекенда прозводится при помощи terrafrom init

Плюсы remote backends

- Централизованное хранение стейта
- Автоматическое обновление при каждом изменении
- Поддержка блокирования стейта

Модули

Модули

- Набор конфигурационных файлов для управления частью инфраструктуры
- Позволяют переиспользовать уже написанный код
- Способствуют однородности окружений
- Важна параметризация

Пример

modules/app/main.tf

```
resource "google_compute_instance" "app" {
  name         = "reddit-app-${count.index + 1}"
  machine_type = "g1-small"
  zone         = "europe-west1-b"
  tags         = ["reddit-app"]
  count        = "${var.count}"

  boot_disk {
    initialize_params {
      image = "${var.app_disk_image}"
    }
  }

  network_interface {
    network       = "default"
    access_config = {
    }
  }
}
```

prod/main.tf

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

module "app" {
  source           = "modules/app"
  count            = "6"
  public_key_path = "${var.public_key_path}"
  app_disk_image  = "reddit-app-base-20170910-033251"
}
```

Используем более новую версию образа и меньшее количество машин.

stage/main.tf

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

module "app" {
  source           = "modules/app"
  count            = "3"
  public_key_path  = "${var.public_key_path}"
  app_disk_image   = "reddit-app-base-20170911-033251"
}
```