Описание инфраструктуры с Terraform на будущее

Антон Бабенко — @antonbabenko

Октябрь 2018

Антон Бабенко

Terraform AWS фанатик с 2015 года

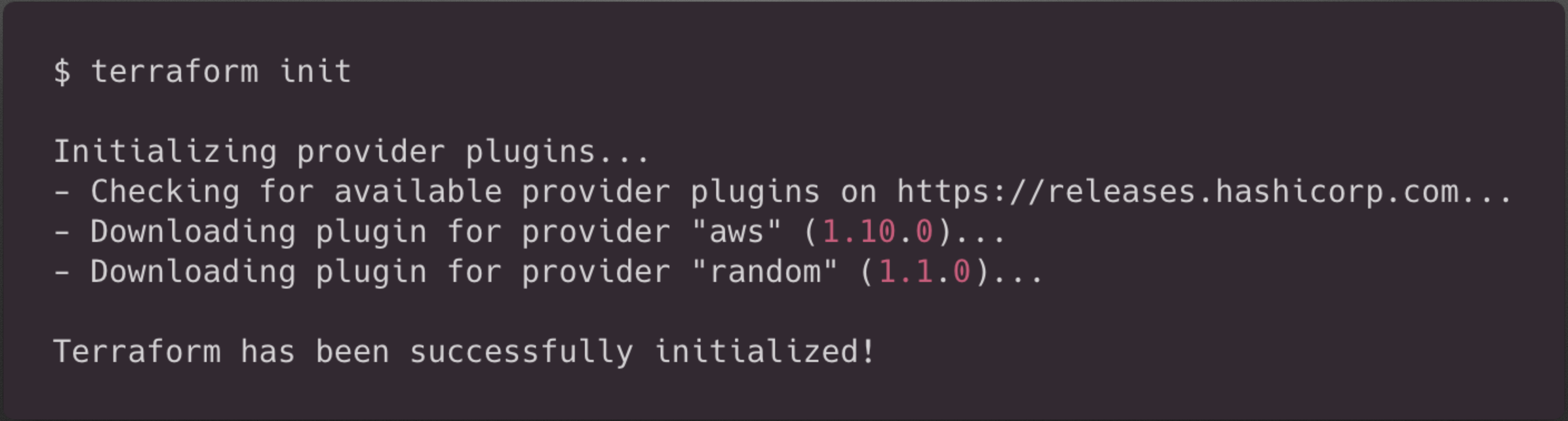
Я 💚 open-source:

- terraform-community-modules + terraform-aws-modules
- antonbabenko/pre-commit-terraform — автоформатирование кода и документации
- antonbabenko/terrapin — генератор Terraform модулей (WIP)
- antonbabenko/modules.tf-lambda — генератор Terraform кода из визуальных диаграм
- www.terraform-best-practices.com
- medium.com/@anton.babenko

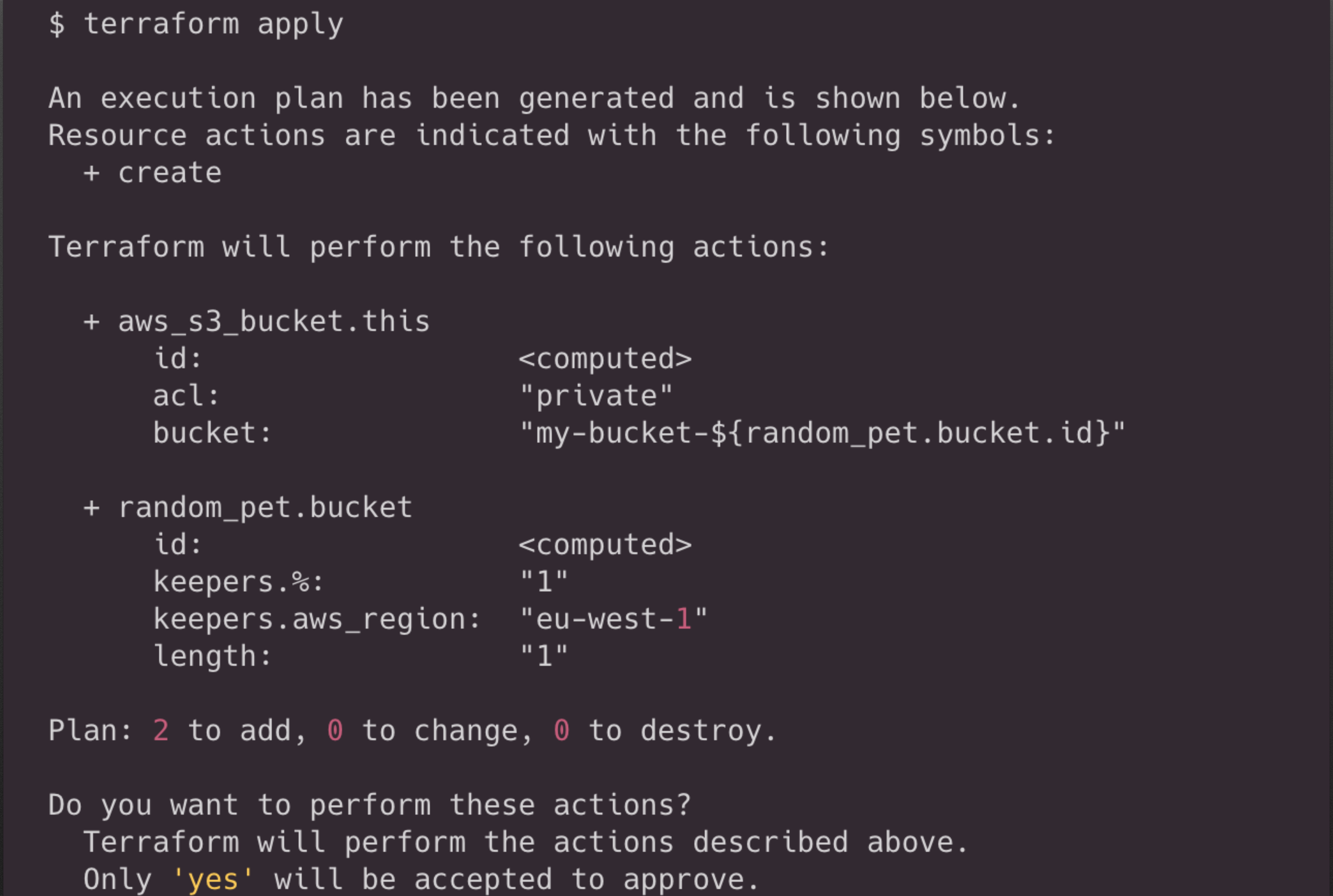
Пиши, планируй, и управляй инфраструктурой как код

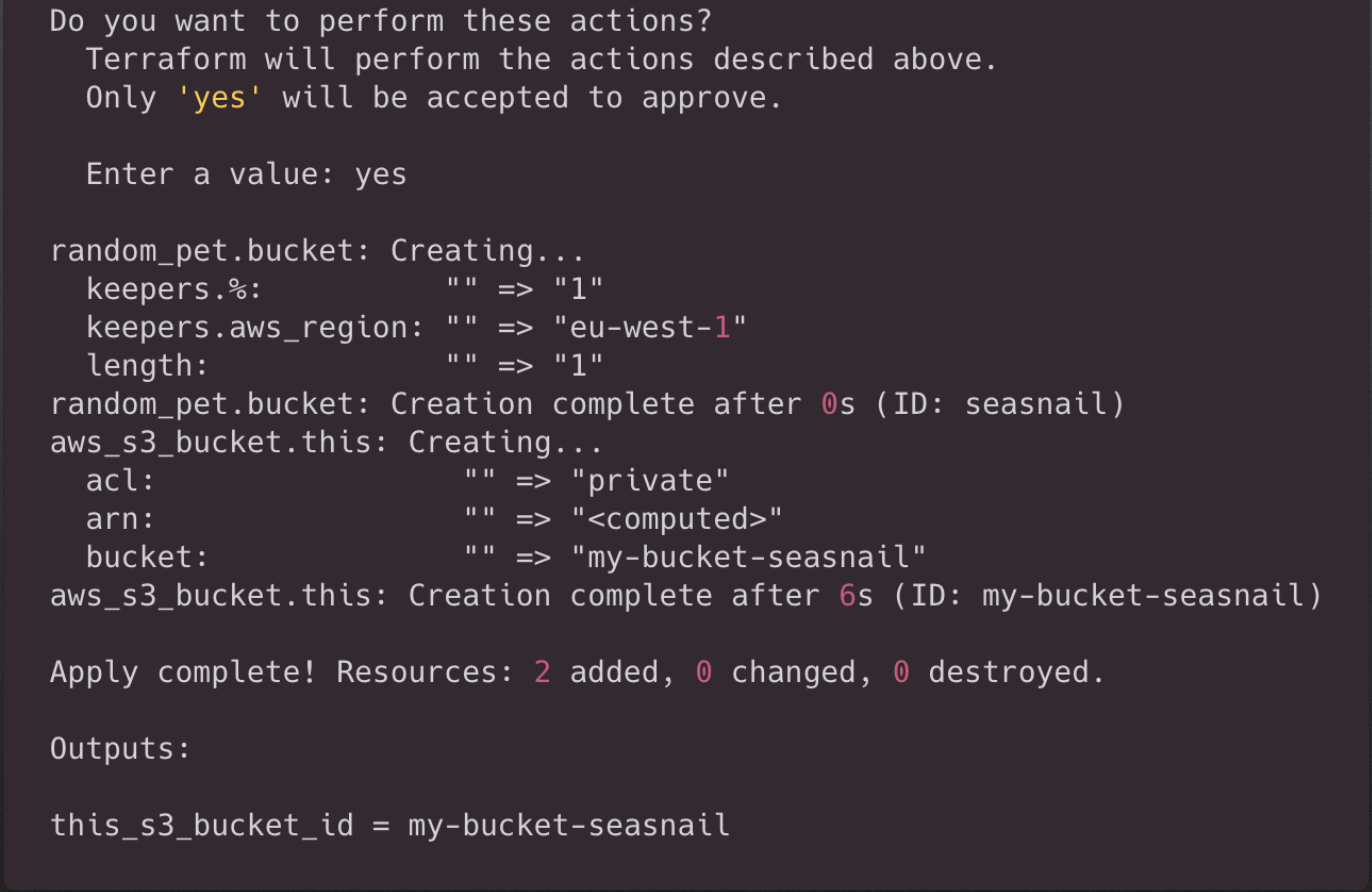
www.terraform.io

```hcl
variable "aws_region" {
  description = "Region where resources should be created"
  default     = "eu-west-1"
}

provider "aws" {
  region = "${var.aws_region}"
}

resource "aws_s3_bucket" "this" {
  bucket = "my-bucket-${random_pet.bucket.id}"
}

resource "random_pet" "bucket" {
  keepers = {
    aws_region = "${var.aws_region}"
  }

  length = 1
}

output "this_s3_bucket_id" {
  description = "ID of S3 bucket"
  value       = "${aws_s3_bucket.this.id}"
}
```

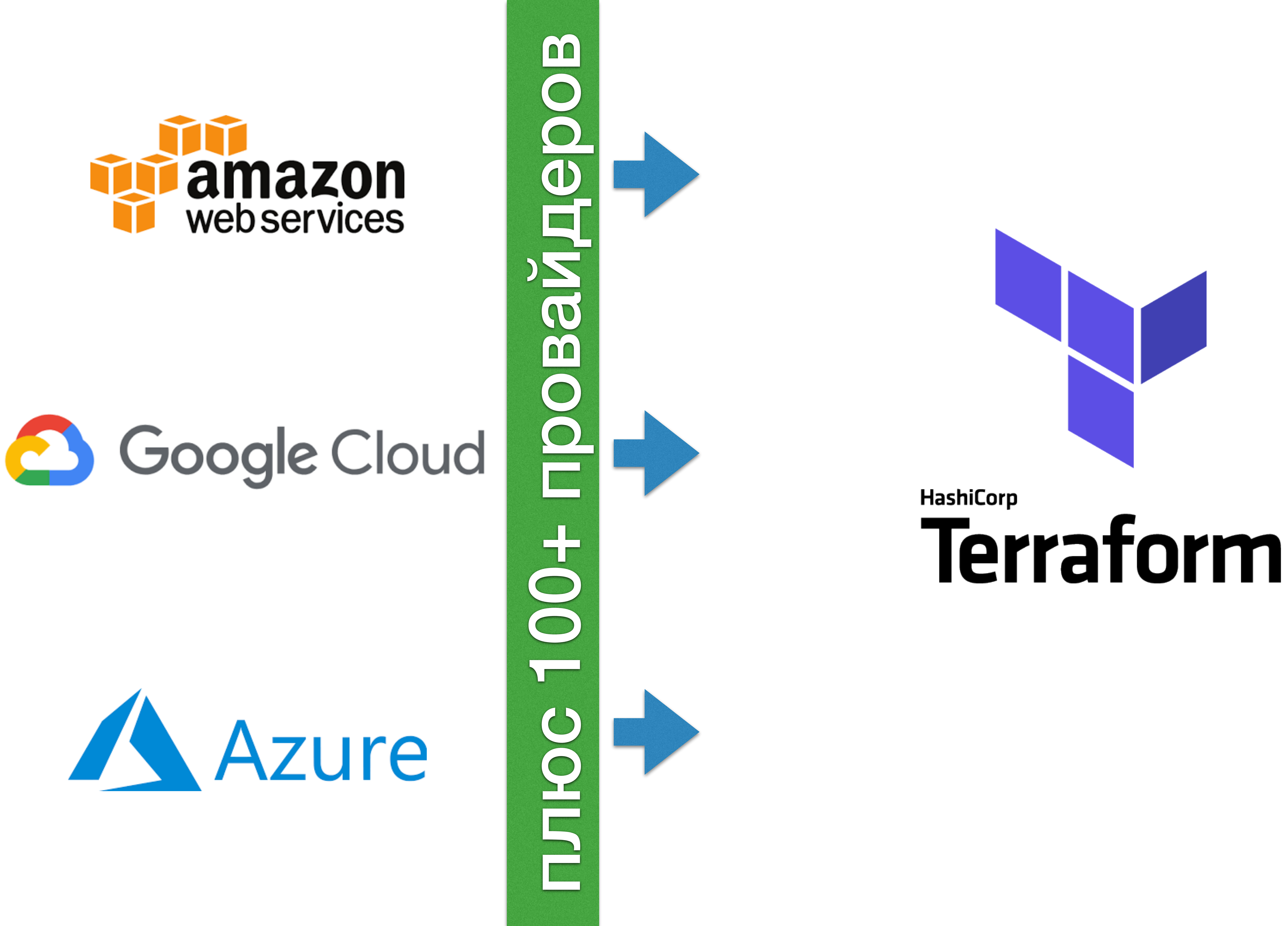


```
$ terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "aws" (1.10.0)...
- Downloading plugin for provider "random" (1.1.0)...

Terraform has been successfully initialized!
```


```
$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_s3_bucket.this
      id:                    <computed>
      acl:                   "private"
      bucket:                "my-bucket-${random_pet.bucket.id}"

  + random_pet.bucket
      id:                    <computed>
      keepers.%:             "1"
      keepers.aws_region:    "eu-west-1"
      length:                "1"

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
```



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_pet.bucket: Creating...
  keepers.%:          "" => "1"
  keepers.aws_region: "" => "eu-west-1"
  length:             "" => "1"
random_pet.bucket: Creation complete after 0s (ID: seasnail)
aws_s3_bucket.this: Creating...
  acl:                 "" => "private"
  arn:                 "" => "<computed>"
  bucket:              "" => "my-bucket-seasnail"
aws_s3_bucket.this: Creation complete after 6s (ID: my-bucket-seasnail)

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

Outputs:

this_s3_bucket_id = my-bucket-seasnail
```


amazon web services
Google Cloud
Azure
плюс 100+ провайдеров
HashiCorp Terraform

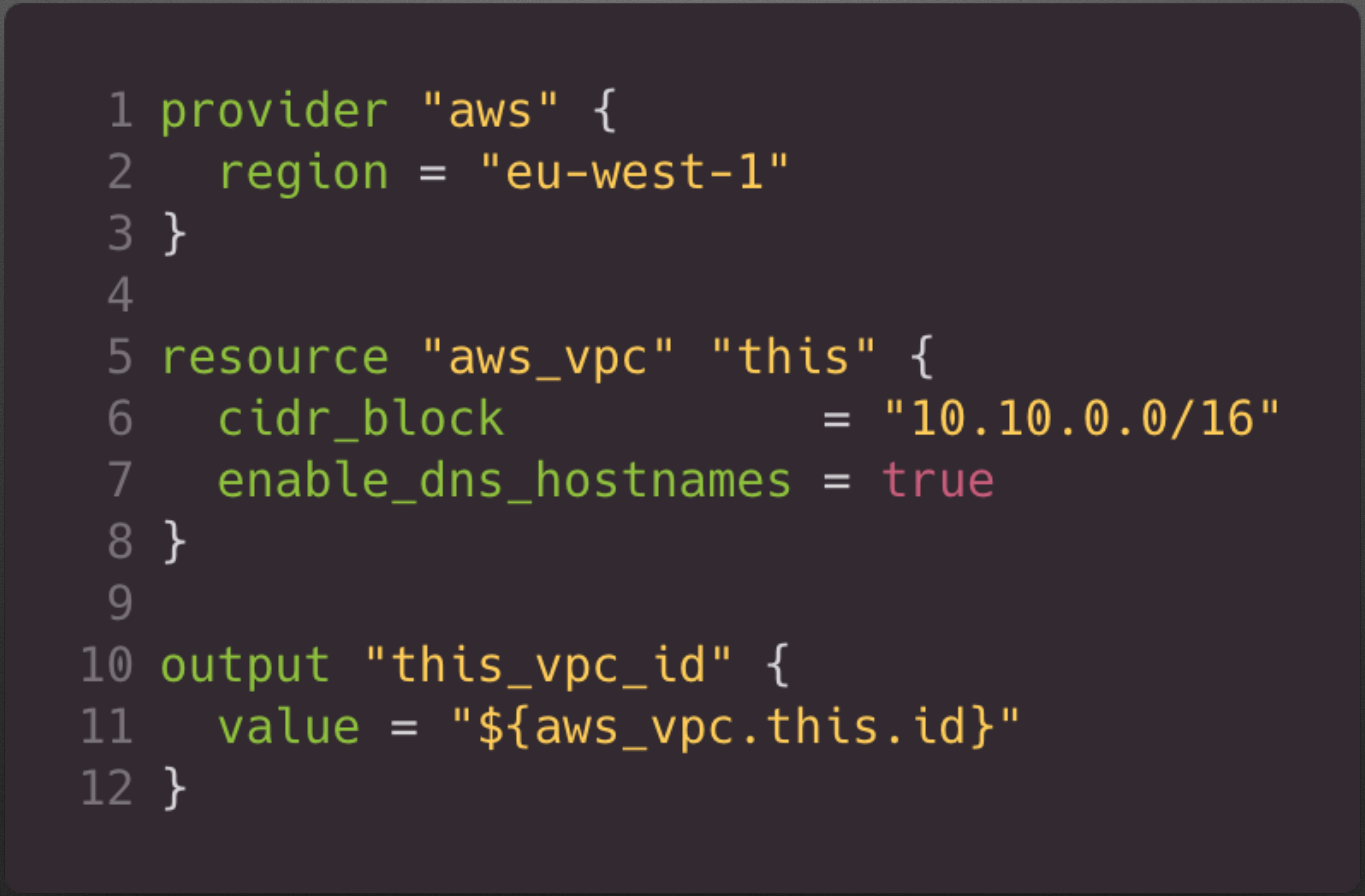
Цели Terraform

- Единый вид ресурсов используя инфраструктуру как код
- Поддержка современных платформ
- Разрешить пользователям и командам изменять инфраструктуру (безопасно и предсказуемо)

Terraform — универсальная утилита для всего, что имеет API

- GSuite
- Dropbox файлы и доступы
- New Relic метрики
- Datadog пользователи, метрики
- Баги в Jira
- Все Terraform провайдеры

Начало — всё влезло в main.tf

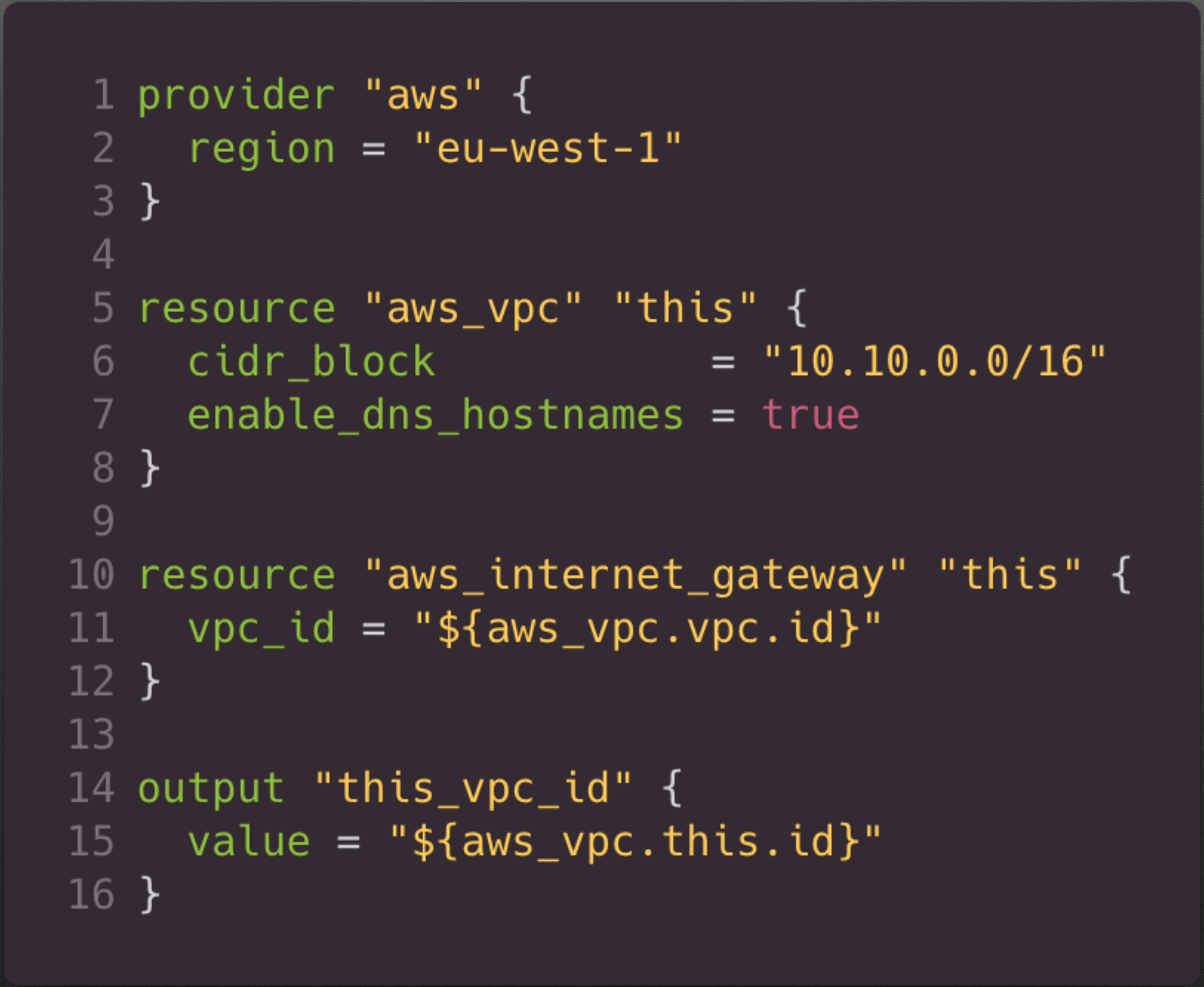


```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_vpc" "this" {
  cidr_block           = "10.10.0.0/16"
  enable_dns_hostnames = true
}

output "this_vpc_id" {
  value = "${aws_vpc.this.id}"
}
```

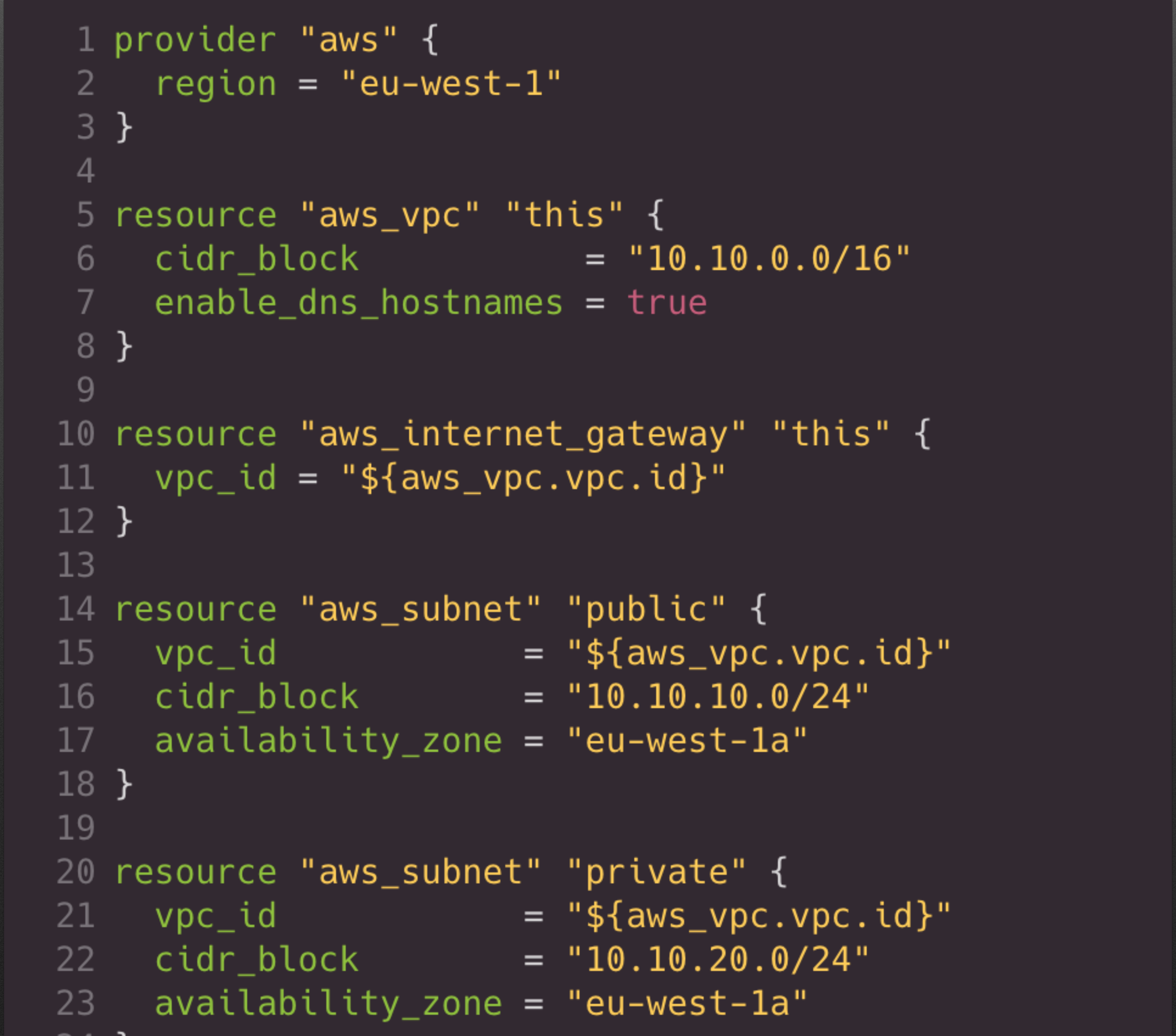
Проект растёт — main.tf: 20+ ресурсов и источников данных

Почему?

Ресурсы, регионы, провайдеры, …



```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_vpc" "this" {
  cidr_block           = "10.10.0.0/16"
  enable_dns_hostnames = true
}

resource "aws_internet_gateway" "this" {
  vpc_id = "${aws_vpc.vpc.id}"
}

output "this_vpc_id" {
  value = "${aws_vpc.this.id}"
}
```



```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_vpc" "this" {
  cidr_block           = "10.10.0.0/16"
  enable_dns_hostnames = true
}

resource "aws_internet_gateway" "this" {
  vpc_id = "${aws_vpc.vpc.id}"
}

resource "aws_subnet" "public" {
  vpc_id            = "${aws_vpc.vpc.id}"
  cidr_block        = "10.10.10.0/24"
  availability_zone = "eu-west-1a"
}

resource "aws_subnet" "private" {
  vpc_id            = "${aws_vpc.vpc.id}"
  cidr_block        = "10.10.20.0/24"
  availability_zone = "eu-west-1a"
```

```

resource "aws_subnet" "private" {
  vpc_id            = "${aws_vpc.vpc.id}"
  cidr_block        = "10.10.20.0/24"
  availability_zone = "eu-west-1a"
}

output "this_vpc_id" {
  value = "${aws_vpc.this.id}"
}

output "private_subnet_ids" {
  value = ["${aws_subnet.private.*.id}"]
}

output "public_subnet_ids" {
  value = ["${aws_subnet.public.*.id}"]
}

# @todo: Add NAT gateways, routes, routing tables
```

Terraform модули

Типы Terraform модулей

- Ресурсные модули (terraform-aws-modules, например)
- Инфраструктурные модули

Ресурсные модули

- Очень гибкие
- Open-source

Ресурсные модули

```
module "atlantis_alb_sg" {
  source  = "terraform-aws-modules/security-group/aws//modules/https-443"
  version = "v2.0.0"

  name        = "atlantis-alb"
  vpc_id      = "vpc-12345678"
  description = "Security group with HTTPS ports open for everybody (IPv4 CIDR)"

  ingress_cidr_blocks = ["0.0.0.0/0"]
}
```

Q: Зачем использовать ресурсные модули вместо ресурсов?
A: Ресурсы не могут быть версиоными, а модули могут.

```
locals {
  this_sg_id = "${element(concat(coalescelist(aws_security_group.this.*.id, aws_security_group.this_name_prefix.*.id), list("")), 0)}"
}

##########################
# Security group with name
##########################
resource "aws_security_group" "this" {
  count = "${var.create && ! var.use_name_prefix ? 1 : 0}"

  name        = "${var.name}"
  description = "${var.description}"
  vpc_id      = "${var.vpc_id}"

  tags = "${merge(var.tags, map("Name", format("%s", var.name)))}"
}

#################################
# Security group with name_prefix
#################################
resource "aws_security_group" "this_name_prefix" {
  count = "${var.create && var.use_name_prefix ? 1 : 0}"

  name_prefix = "${var.name}-"
  description = "${var.description}"
  vpc_id      = "${var.vpc_id}"

  tags = "${merge(var.tags, map("Name", format("%s", var.name)))}"
```

Инфраструктурные модули

- Состоят из ресурсных модулей
- Теги, стандарты компании
- Препроцессоры, jsonnet, cookiecutter

Инфраструктурные модули

```
module "atlantis" {
  source = "terraform-aws-modules/atlantis/aws"

  name = "atlantis"

  # VPC
  cidr            = "10.20.0.0/20"
  azs             = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]
  private_subnets = ["10.20.1.0/24", "10.20.2.0/24", "10.20.3.0/24"]
  public_subnets  = ["10.20.101.0/24", "10.20.102.0/24", "10.20.103.0/24"]

  # DNS
  route53_zone_name = "terraform-aws-modules.modules.tf"

  # Atlantis app
  atlantis_github_user       = "atlantis-bot"
  atlantis_github_user_token = "examplegithubtoken"
}
```

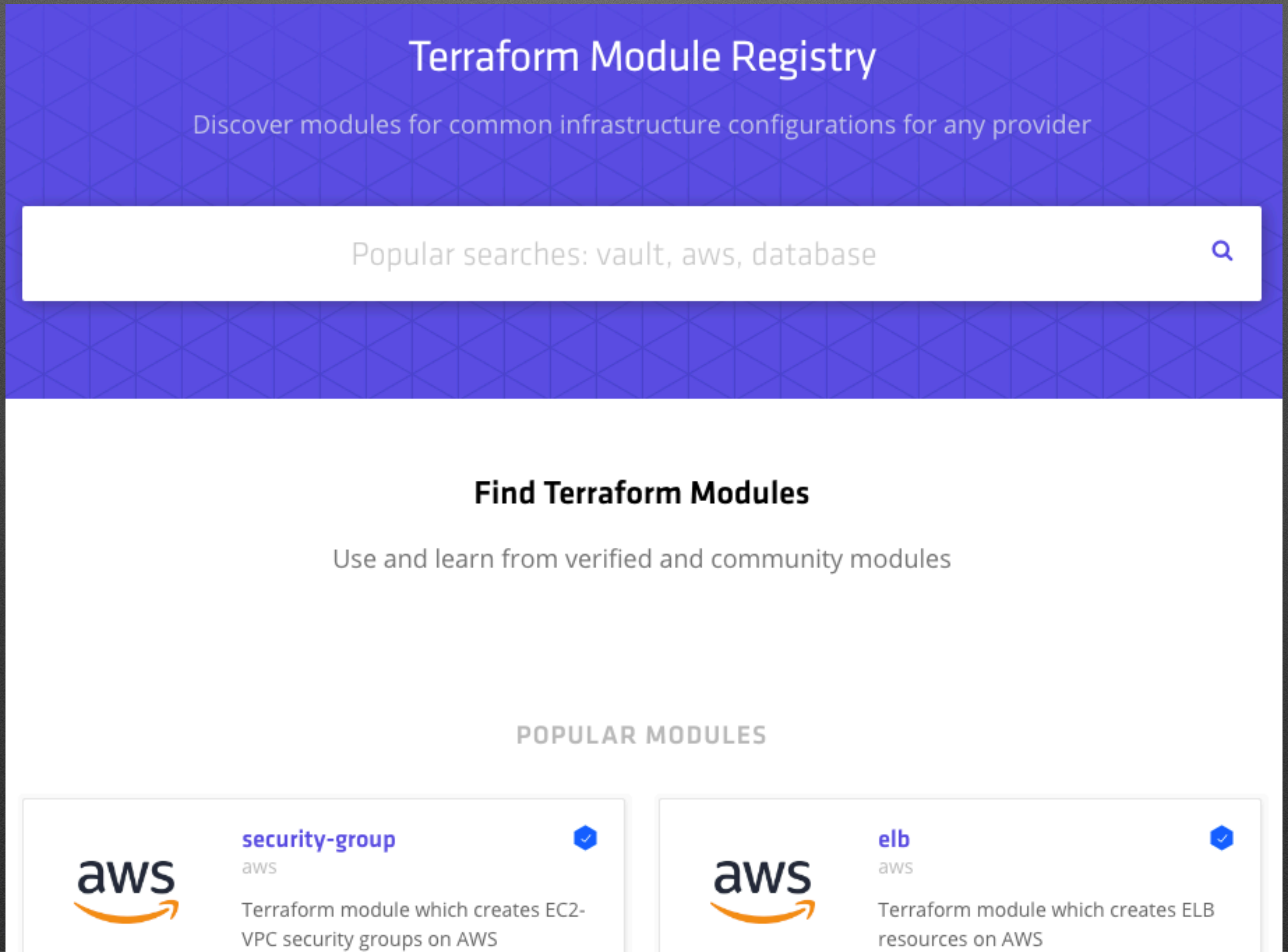
```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "v1.32.0"
  # ...
}

module "alb" {
  source  = "terraform-aws-modules/alb/aws"
  version = "v3.4.0"

  # ...
}

module "alb_https_sg" {
  source  = "terraform-aws-modules/security-group/aws//modules/https-443"
  version = "v2.0.0"

  # ...
}

module "ecs" {
  source  = "terraform-aws-modules/ecs/aws"
  version = "v1.0.0"

  # ...
}
```

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "v1.32.0"

  name = "${var.name}"

  cidr            = "${var.cidr}"
  azs             = "${var.azs}"
  private_subnets = "${var.private_subnets}"
  public_subnets  = "${var.public_subnets}"

  enable_nat_gateway = true
  single_nat_gateway = true

  tags = "${local.tags}"
}

module "alb" {
  source  = "terraform-aws-modules/alb/aws"
  version = "v3.4.0"

  load_balancer_name = "${var.name}"

  vpc_id          = "${local.vpc_id}"
  subnets         = ["${local.public_subnet_ids}"]
  security_groups = ["${module.alb_https_sg.this_security_group_id}"]
  logging_enabled = false

  https_listeners = [{
```

```
- [ ] Как писать модули
- [ ] Как вызывать модули
- [ ] Как работать с кодом
```

Совет №0

Проверь Terraform Registry перед тем, как начать писать ресурсный модуль



Terraform Module Registry

Discover modules for common infrastructure configurations for any provider

Popular searches: vault, aws, database

Find Terraform Modules

Use and learn from verified and community modules

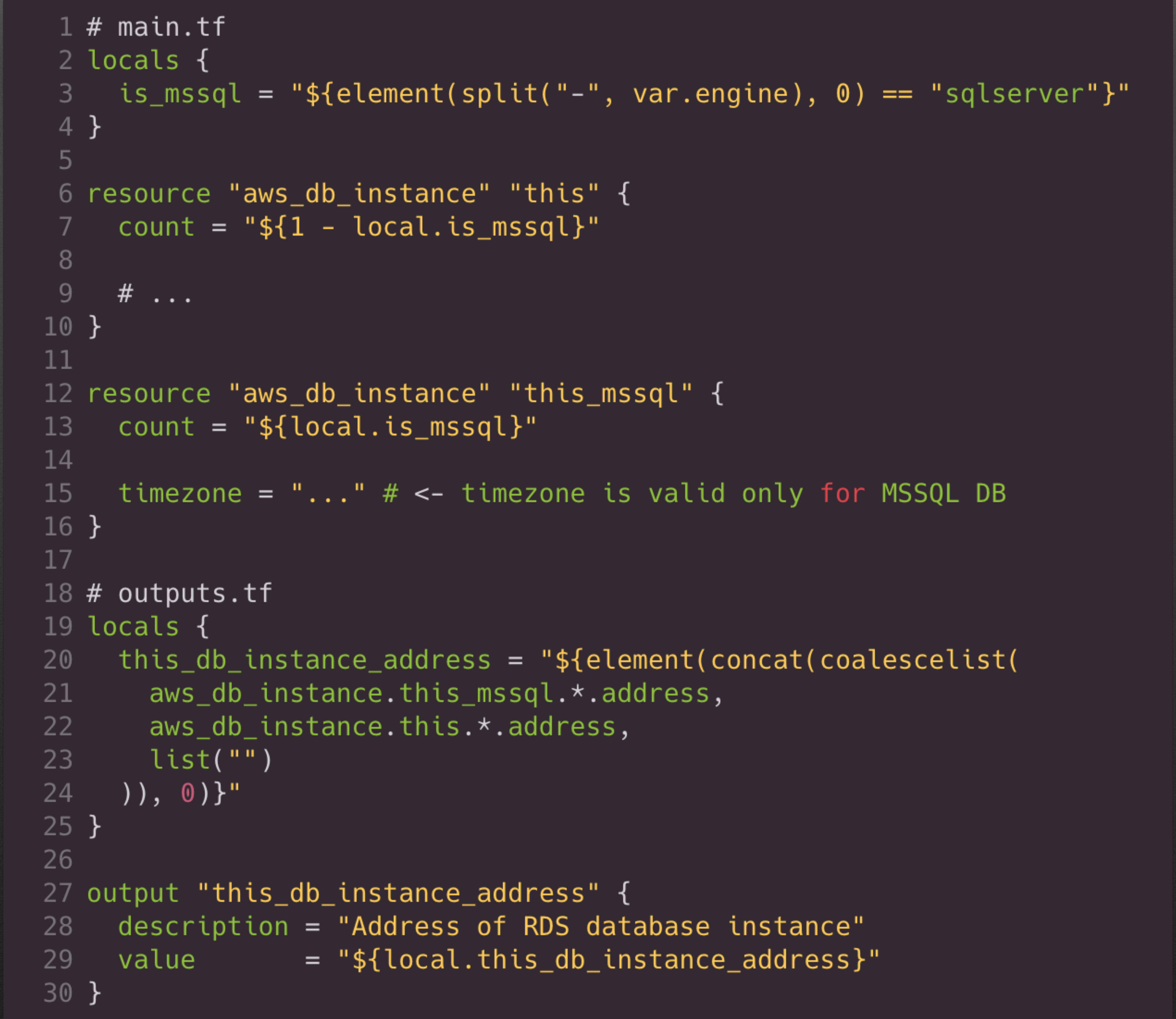
POPULAR MODULES

security-group
aws
Terraform module which creates EC2-VPC security groups on AWS

elb
aws
Terraform module which creates ELB resources on AWS

Прячь специфику

```
module "db_instance" {
  source   = "terraform-aws-modules/rds/aws"

  engine   = "sqlserver-ex"
  timezone = "Central Standard Time" # <- supported only in MSSQL
}

module "db_instance_mysql" {
  source = "terraform-aws-modules/rds/aws"

  engine = "mysql"
}
```



```
# main.tf
locals {
  is_mssql = "${element(split("-", var.engine), 0) == "sqlserver"}"
}

resource "aws_db_instance" "this" {
  count = "${1 - local.is_mssql}"

  # ...
}

resource "aws_db_instance" "this_mssql" {
  count = "${local.is_mssql}"

  timezone = "..." # <- timezone is valid only for MSSQL DB
}

# outputs.tf
locals {
  this_db_instance_address = "${element(concat(coalescelist(
    aws_db_instance.this_mssql.*.address,
    aws_db_instance.this.*.address,
    list("")
  )), 0)}"
}

output "this_db_instance_address" {
  description = "Address of RDS database instance"
  value       = "${local.this_db_instance_address}"
}
```

Размер

```
module "slow_iam_user_1" {
  source  = "github.com/company/terraform-modules.git//iam-user"
  version = "1.0.0"
}

module "slow_iam_user_2" {
  source  = "terraform-aws-modules/iam/aws//modules/iam-user"
  version = "1.0.0"
}
```

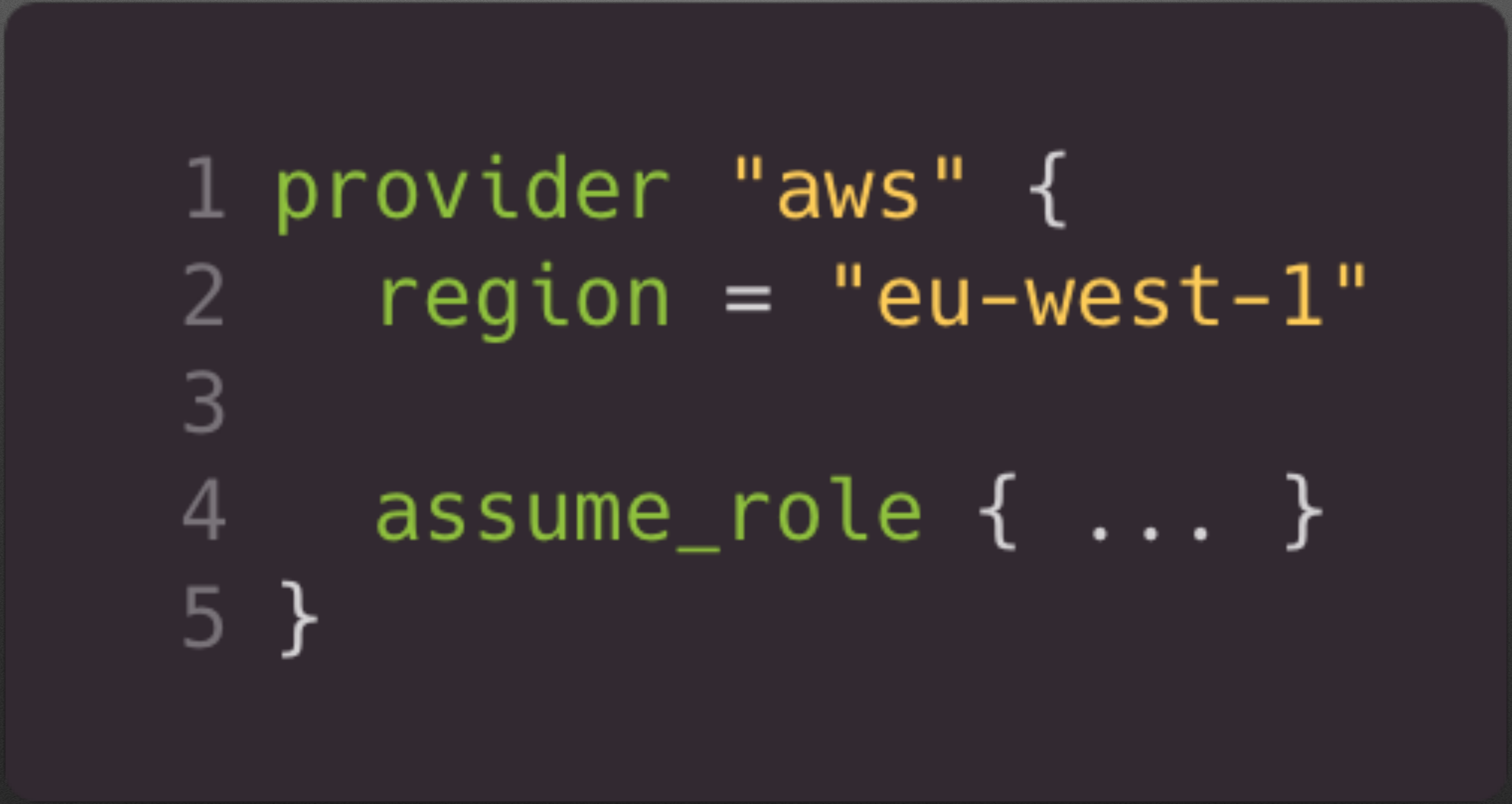
Размер

```
# Run "git clone git@github.com:company/terraform-modules.git" before usage
module "faster_iam_user_1" {
  source = "./terraform-modules/iam-user" # versioning is not supported
}

# Smaller = faster. Build by CI server.
module "faster_iam_user_2" {
  source = "http://artefacts.company.com/terraform-modules/iam-user-1.0.0.zip"
}
```

https://github.com/mbtproject/mbt

Избегай в модулях

Провайдеры в модулях — зло

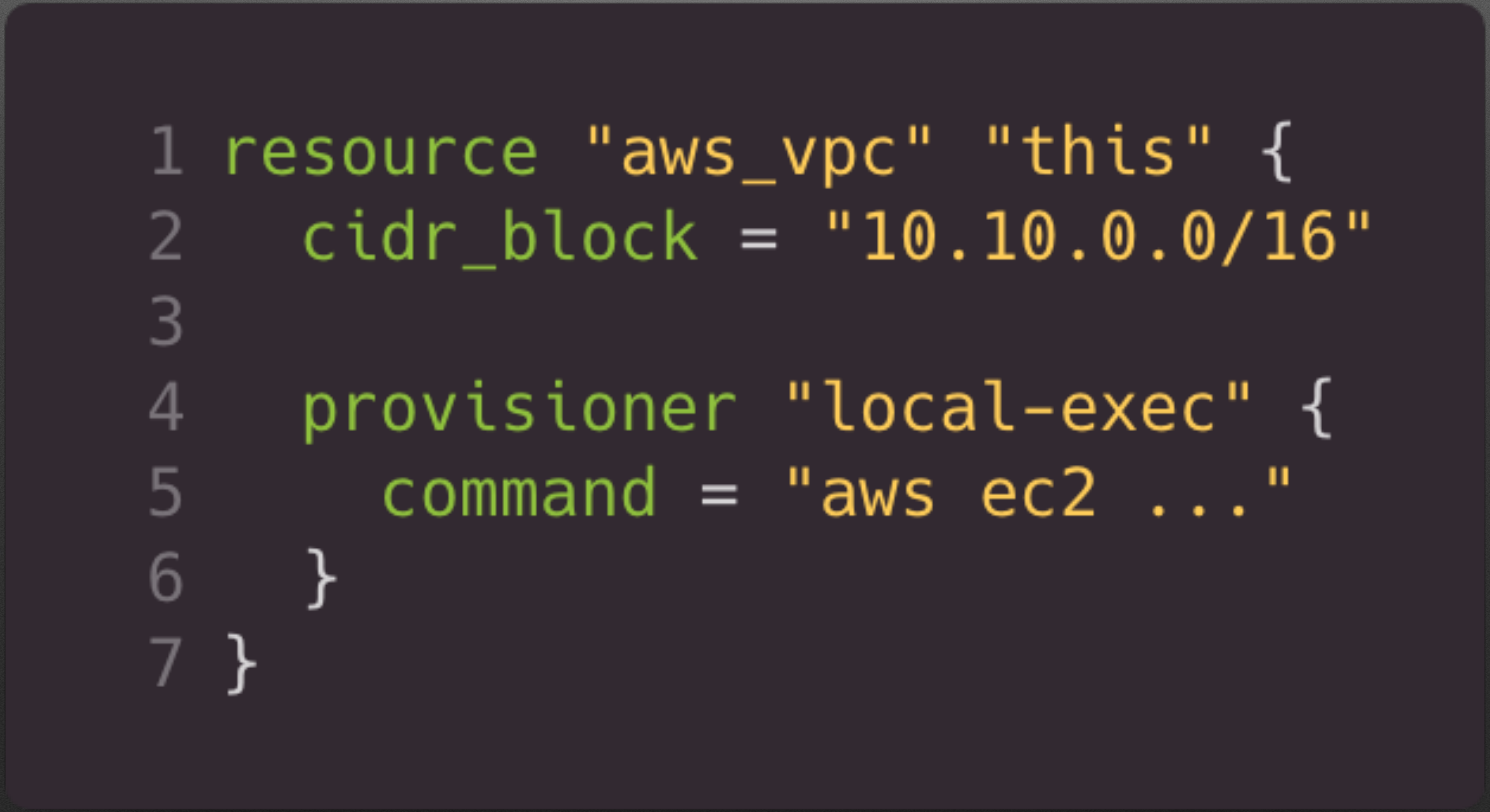


```
provider "aws" {
  region = "eu-west-1"

  assume_role { ... }
}
```

Исключение: логические провайдеры (template, random, local, http, external)

```hcl
# Vasya likes defaults
provider "aws" {
  region = "eu-west-1"
}

# Petya likes shared credentials file
provider "aws" {
  region = "eu-west-1"

  shared_credentials_file = "~/.aws/my_secrets"
}

# They both lose!
```

Provisioner — зло


```
resource "aws_vpc" "this" {
  cidr_block = "10.10.0.0/16"

  provisioner "local-exec" {
    command = "aws ec2 ..."
  }
}
```

Избегайте provisioner во всех ресурсах

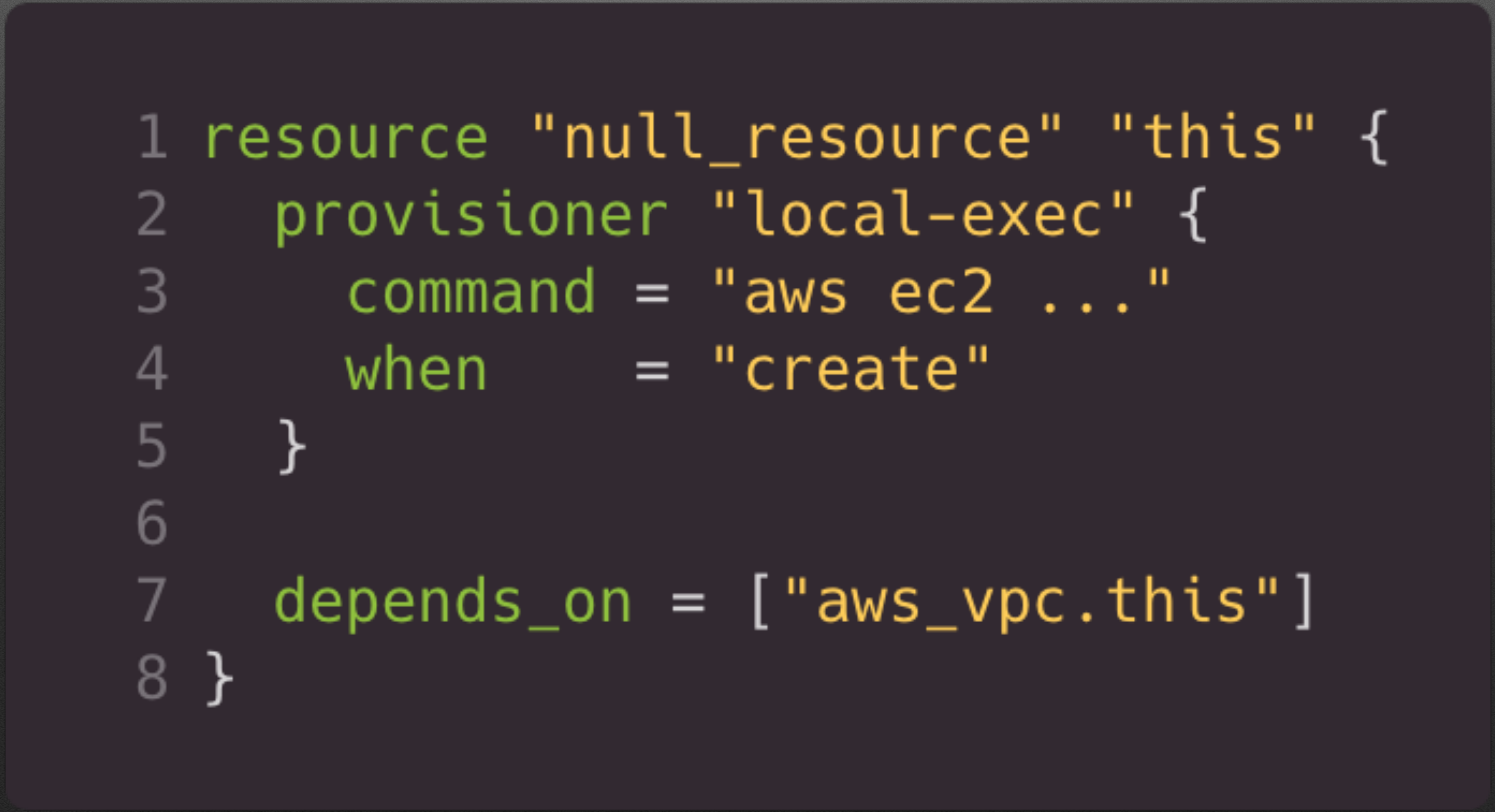
Provisioner — зло

```
resource "aws_instance" "this" {
  ami           = "ami-12345678"
  instance_type = "t3.large"

  provisioner "local-exec" {
    command = "ansible-playbook ..."
  }
}
```

Избегайте provisioner даже в EC2 ресурсах

```
# Solution 1
resource "aws_instance" "this" {
  ami           = "ami-12345678"
  instance_type = "t3.large"

  user_data     = "aws s3 cp ... & ansible-playbook ..."
}

# Solution 2 - Autoscaling group
resource "aws_launch_configuration" "this" {
  image_id      = "ami-12345678"
  instance_type = "t3.large"

  user_data     = "aws s3 cp ... & ansible-playbook ..."
}
```

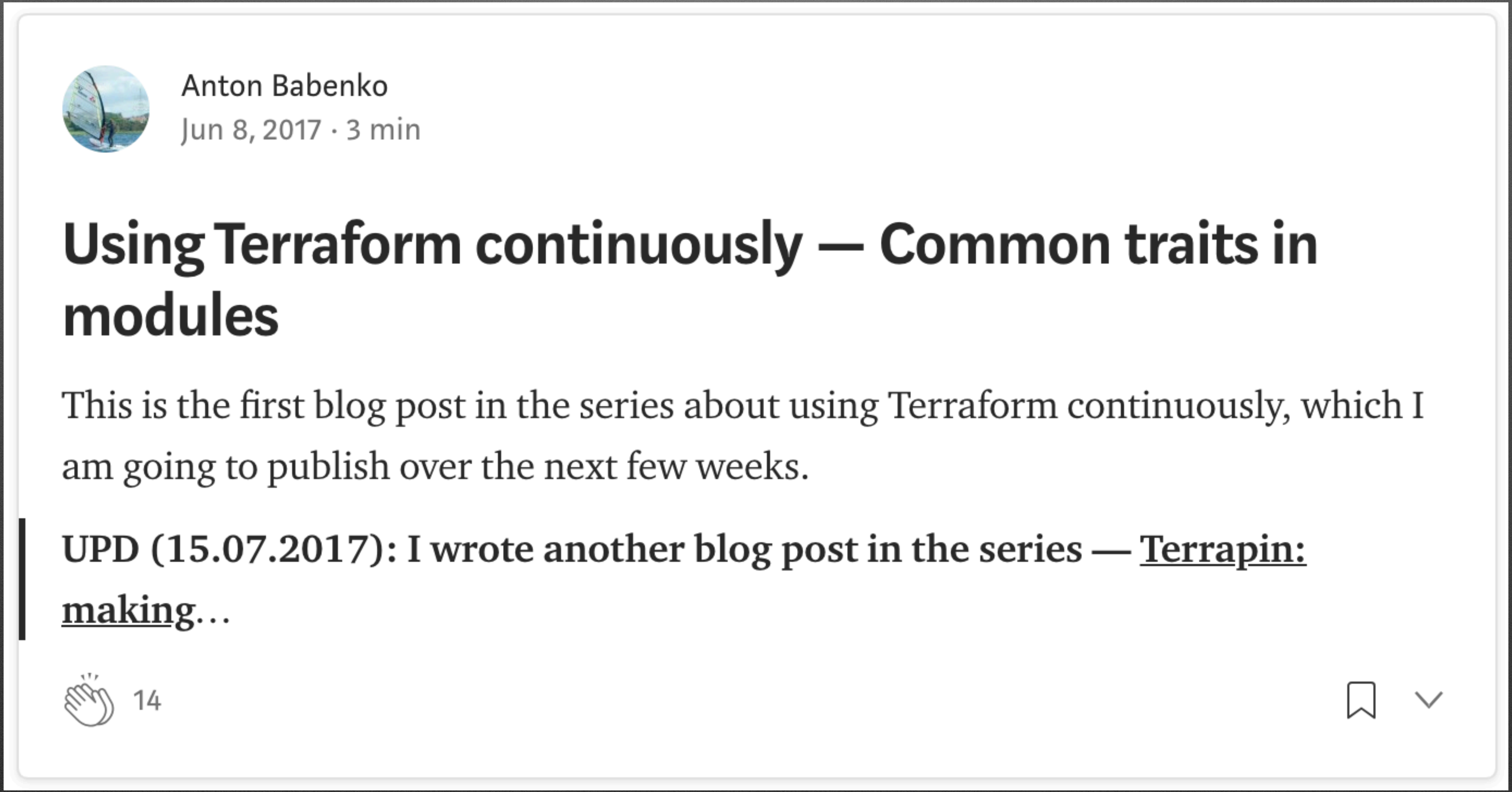
null_resource provisioner — добро



```
resource "null_resource" "this" {
  provisioner "local-exec" {
    command = "aws ec2 ..."
    when    = "create"
  }

  depends_on = ["aws_vpc.this"]
}
```

Признаки хороших Terraform модулей

- Документация и примеры
- Полный функционал
- Разумные значения по-умолчанию
- Чистый код
- Тесты



Anton Babenko
Jun 8, 2017 · 3 min

Using Terraform continuously — Common traits in modules

This is the first blog post in the series about using Terraform continuously, which I am going to publish over the next few weeks.

> **UPD (15.07.2017): I wrote another blog post in the series — Terrapin: making…**

14

https://medium.com/@anton.babenko/using-terraform-continuously-common-traits-in-modules-8036b71764db

- [x] Как писать модули
- [ ] Как вызывать модули
- [ ] Как работать с кодом

Всё в одном

Хорошо:

1. Описывать переменные и значения в меньшем количестве мест

Плохо:

1. Большая область действия

2. Блокируется всё сразу

3. Нет возможности указать зависимости между модулями (depends_on)

```
# terraform.tfvars
region = "eu-west-1"

# main.tf
provider "aws" {
  region = "${var.region}"
}

module "vpc" {}
module "alb" {}
module "application" {}
module "security_group_alb" {}
module "security_group_app" {}
module "security_group_microservice" {}
module "microservice_1" {}
module "microservice_2" {}
module "microservice_X" {}
```

1-in-1

Хорошо:

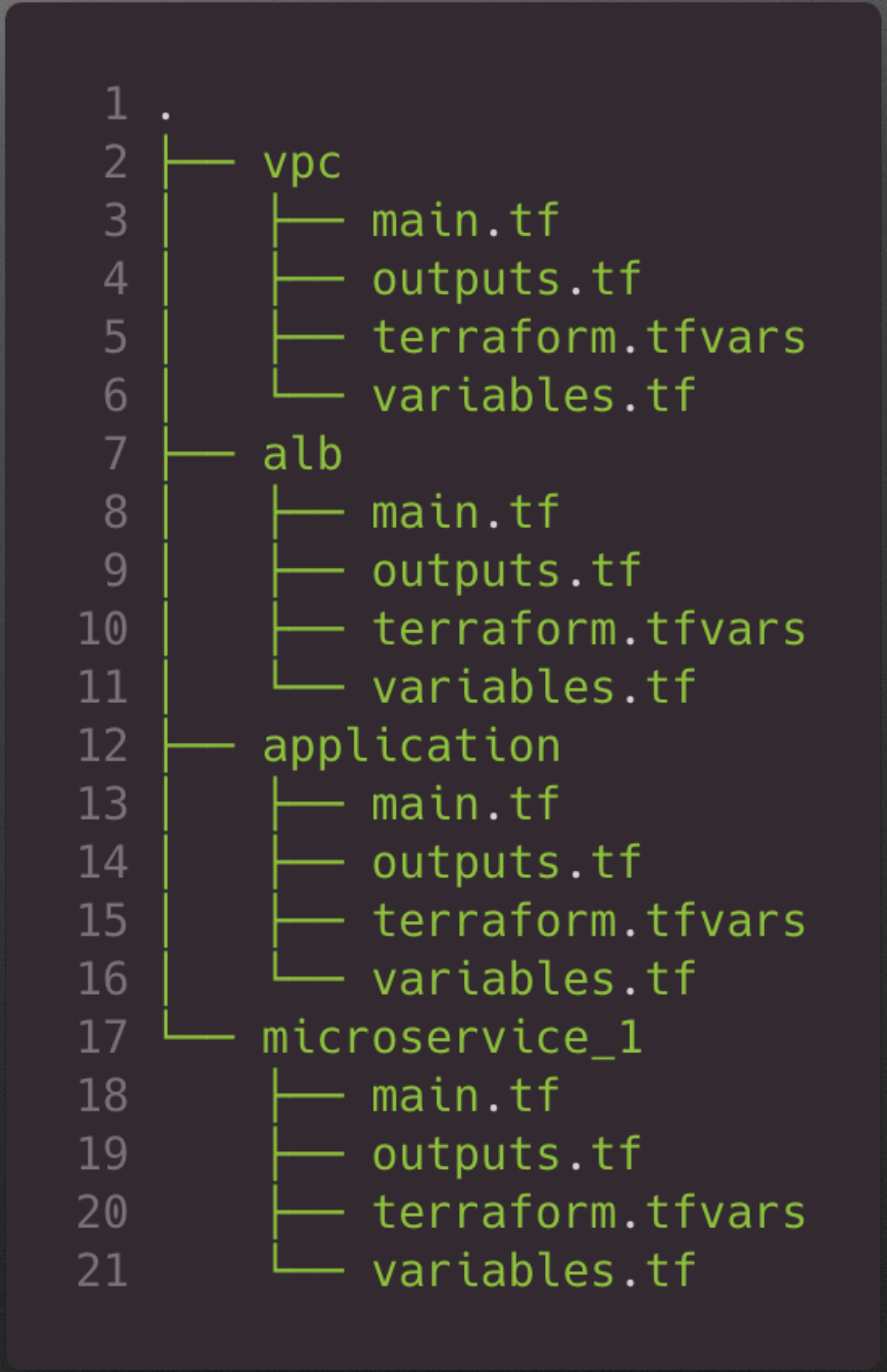
1. Меньше область действия

2. Возможно связать вызовы

3. Легче и быстрее работается

Плохо:

1. Описывать переменный в нескольких местах



```
.
├── vpc
│   ├── main.tf
│   ├── outputs.tf
│   ├── terraform.tfvars
│   └── variables.tf
├── alb
│   ├── main.tf
│   ├── outputs.tf
│   ├── terraform.tfvars
│   └── variables.tf
├── application
│   ├── main.tf
│   ├── outputs.tf
│   ├── terraform.tfvars
│   └── variables.tf
└── microservice_1
    ├── main.tf
    ├── outputs.tf
    ├── terraform.tfvars
    └── variables.tf
```

А как у вас?

"Всё в одном" или 1-in-1 ?

Правильный ответ:
где-то посередине

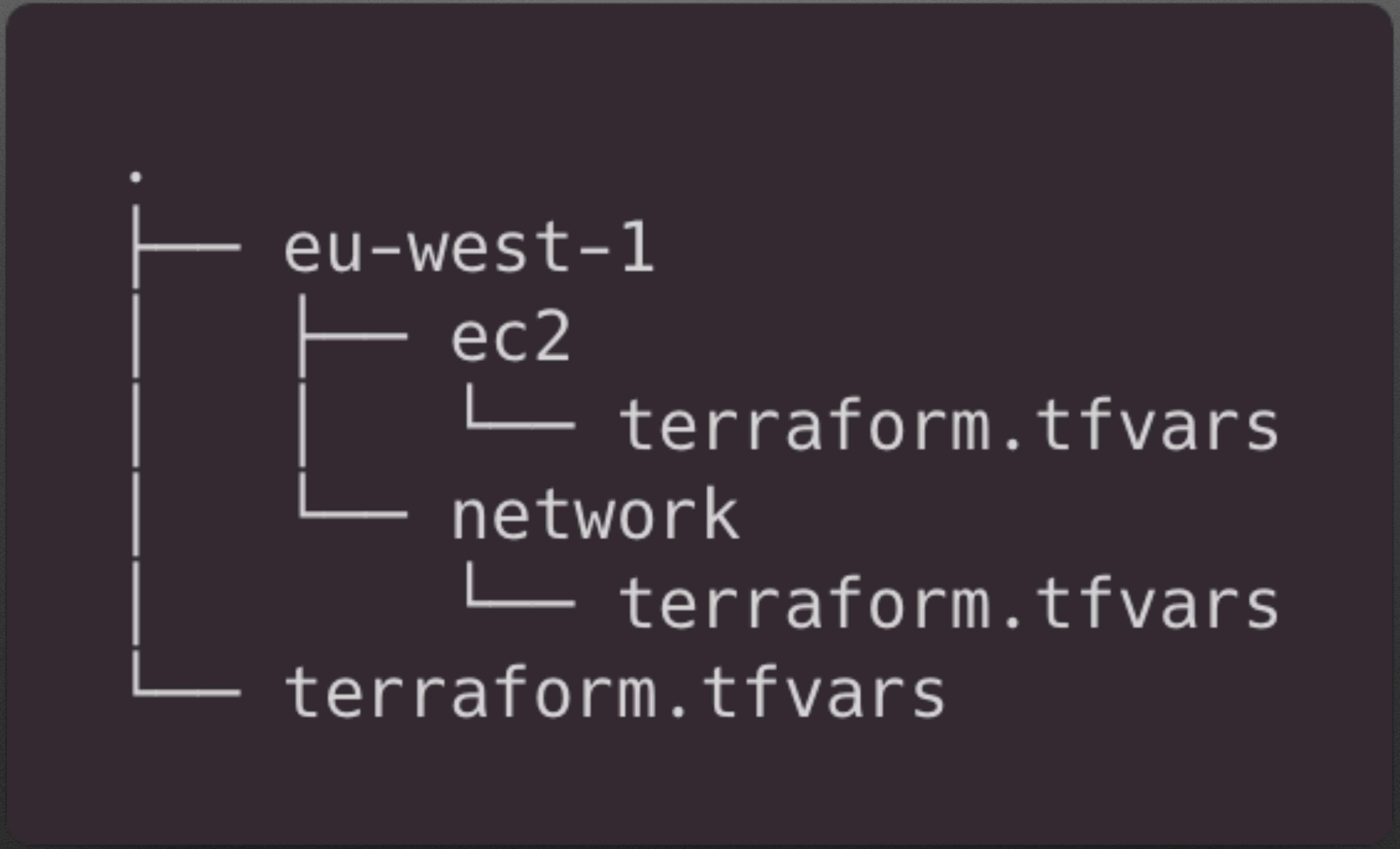
Оркестрация в Terraform

```
resource "null_resource" "this" {
  provisioner "local-exec" {
    command = "terraform apply -target=aws_iam_user.this_user"
  }
}
```

Не повторяйте это дома!

```
# Do not try this at home!
resource "null_resource" "this" {
  provisioner "local-exec" {
    command = "terraform apply -target=aws_iam_user.this_user"
  }
}
```

Оркестрация = Terragrunt



```
.
├── eu-west-1
│   ├── ec2
│   │   └── terraform.tfvars
│   └── network
│       └── terraform.tfvars
└── terraform.tfvars
```

https://github.com/gruntwork-io/terragrunt/

Оркестрация = Terragrunt

```
# eu-west-1/ec2/terraform.tfvars
terragrunt = {
  terraform {
    source = "git::git@github.com:terraform-aws-modules/terraform-aws-ec2-instance.git?ref=v1.0.0"
  }

  dependencies {
    paths = ["../network"]
  }
}
```

https://github.com/gruntwork-io/terragrunt/

- [x] Как писать модули
- [x] Как вызывать модули
- [ ] Как работать с кодом

Новые фичи

Обычно, это легко…

Новый или существующий

```
data "aws_vpc" "selected" {
  count = "${var.vpc_id != "" ? 1 : 0}"

  id = "${var.vpc_id}"
}

resource "aws_vpc" "this" {
  count = "${var.vpc_id == "" ? 1 : 0}"

  cidr_block = "${var.cidr}"
}

output "vpc_id" {
  value = "${element(coalescelist(data.aws_vpc.selected.*.id, aws_vpc.this.*.id), 0)}"
}
```

Работа со списками

```
# terraform.tfvars
ssh_public_keys_names = [
  "user1",
  "user2",
  "user3",
  "user4",
]

# main.tf
resource "aws_s3_bucket_object" "ssh_public_keys" {
  count = "${length(var.ssh_public_keys_names)}"

  bucket  = "my-bucket-for-ssh-public-keys"
  key     = "${element(var.ssh_public_keys_names, count.index)}.pub"
  content = "${file("${element(var.ssh_public_keys_names, count.index)}.pub")}"
}
```

Работа со списками (stateful)

```
# data.json
[{"username": "vasya_pupkin",
  "keybase": "vasyapupkin"
}, {
  "username": "petya_pyatochkin",
  "keybase": "petyapyatochkin"
}]
```

https://jsonnet.org/

Работа со списками (stateful)

```
# users.tf.jsonnet
local users = import "data.json";

{ module: {
    [user.username]: {
      source: "terraform-aws-modules/iam/aws//modules/iam-user",
      name: user.username,
      path: "/" + user.username + "/",
      pgp_key: "keybase:" + user.keybase,
    } for user in users
  },
  output: {
    [user.username]: {
      value: "${module." + user.username + ".instructions}",
    } for user in users
  }
}
```

Работа со списками (stateful)

```
$ jsonnet -o users.tf.json users.tf.jsonnet
$ cat users.tf.json
{
   "module": {
      "user1": {
         "name": "vasya_pupkin",
         "path": "/vasya_pupkin/",
         "pgp_key": "keybase:vasyapupkin",
         "source": "terraform-aws-modules/iam/aws//modules/iam-user"
      },
      ...
}
$ terraform init
$ terraform apply

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

Импорт

```
terraform import aws_iam_account_alias.this alias
```

https://github.com/dtan4/terraforming

Переименование

```
# Rename created resource
$ terraform state mv aws_eip.my_website aws_eip.do_not_release

# Move created ECS cluster "atlantis" inside ECS module
$ terraform state mv aws_ecs_cluster.atlantis module.ecs.aws_ecs_cluster.this
```

Интеграция

```
output "this_waf_acl_command" {
  value = "aws waf-regional associate-web-acl --web-acl-id ${aws_waf_web_acl.this.id} --resource-arn arn:aws:elasticloadbalancing:eu-west-1:123456789012:abc"
}

# Execute:
$(terraform output this_waf_acl_command)
```

Интеграция

```
resource "null_resource" "auto_instructions" {
  triggers = {
    waf_acl_id = "${aws_waf_web_acl.this.id}"
  }

  provisioner "local-exec" {
    command = "aws waf-regional associate-web-acl --web-acl-id ${aws_waf_web_acl.this.id} --resource-arn ${data.terraform_remote_state.alb_public.this_alb_arn}"
  }
}
```

Тестирование

- Ресурсные модули — сложно, не нужно
- Инфраструктурные модули — сложно, но можно

Тестирование

- Обязательно — pre-commit (fmt, validate)
- Terraform plan:
  - Локально
  - Через pull-requests — Atlantis (runatlantis.io)
- Инфраструктурные тесты — terratest, awspec
- Интеграционные тесты — сложно (мониторинг проще)

Гладко было на бумаге (edge cases)

- Разные AWS регионы (S3 подпись, EC2 ClassicLink, IPv6)
- Дата открытия AWS аккаунта
- Лимиты на ресурсы в AWS

Избегай в Terraform

- Несекретных аргументов в коммандой строке => tfvars
  - -target
  - -parallelism
- "Terraform workspaces" зло => отдельная директория
- Ад зависимостей в модулях (Dependency Hell)

Итоги

Пиши меньше и проще

Используй готовые модули, код и утилиты

Спасибо!

github.com/antonbabenko

twitter.com/antonbabenko