

День 2: Изучение Terraform на практике

Спикер:
Павел Замошин





План



здать
вопрос

- Terraform модули — их виды, создание, использование
- Удаленное хранение и использование стейта
- Работа со стейтом в ручном режиме, восстановление стейта
- Обновление между версиями Terraform





В случае проблем



задать
вопрос

Если вдруг вы застряли, у вас ошибка и что-то не получается:

- Попробуйте внимательно прочитать ошибку. Разработчики Terraform постарались сделать вывод ошибок удобным и они вполне читаемые;
- Внимательно ознакомьтесь с предоставленными по теме ссылками, возможно там будет ответ на проблему;
- Посмотрите, как это было решено в коде следующего задания;
- И конечно же, задавайте вопросы в чате.

Если вдруг вы что-то не успели или пропустили — можно взять исходный код следующего задания и продолжить выполнение.





задать
вопрос

Задание 1: Подготовка

Изучим проект, с которым сегодня будем работать. Помимо уже известных **locals.tf**, **outputs.tf**, **providers.tf**, **variables.tf**, **versions.tf** в нем присутствуют следующие файлы:

cluster.tf — создание кластера ECS для публикации контейнеров

iam.tf — настройки доступов в AWS

network.tf — организация сети для инфраструктуры

service.tf — сервис, который будет деплоиться

sg.tf — создание security group (открытие портов)

Давайте инициализируем репозиторий и применим инфраструктуру. После используем **curl** с полученным dns-адресом, чтобы убедиться, что сервис опубликован и работает (это может занять минуту-две после деплоя).

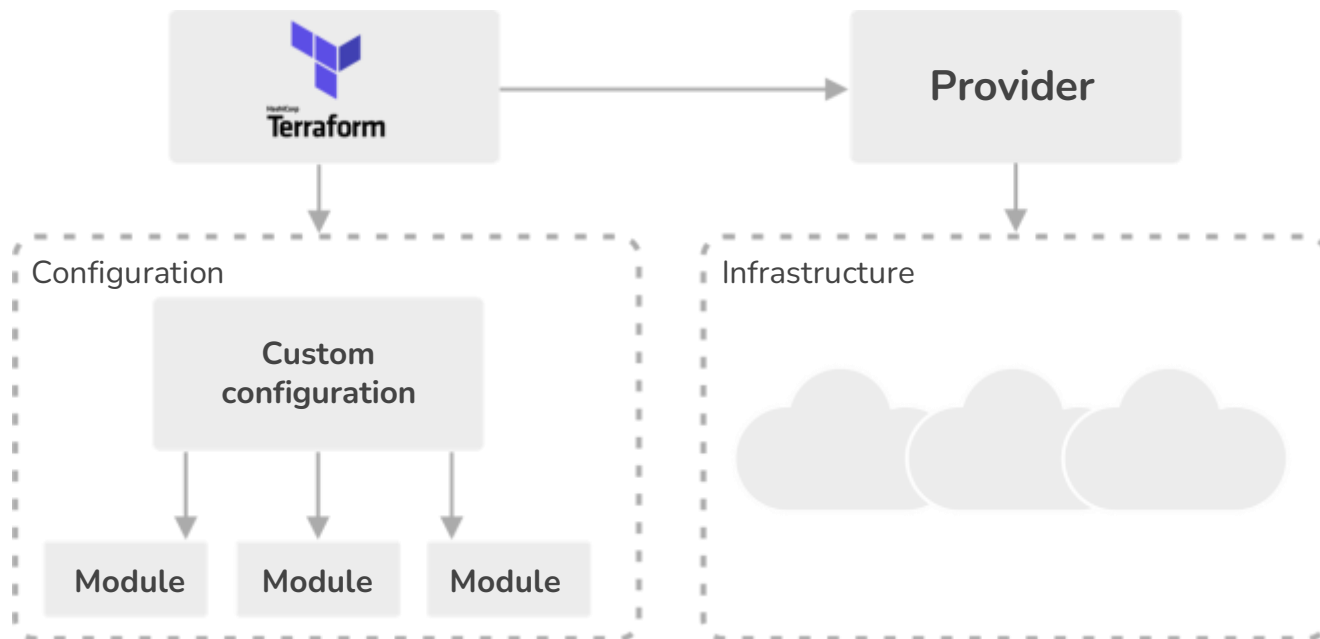


Terraform модули



здать
вопрос

Модуль — абстракция над набором ресурсов, позволяющая использовать их как единое целое. Каждый репозиторий кода Terraform — это модуль.



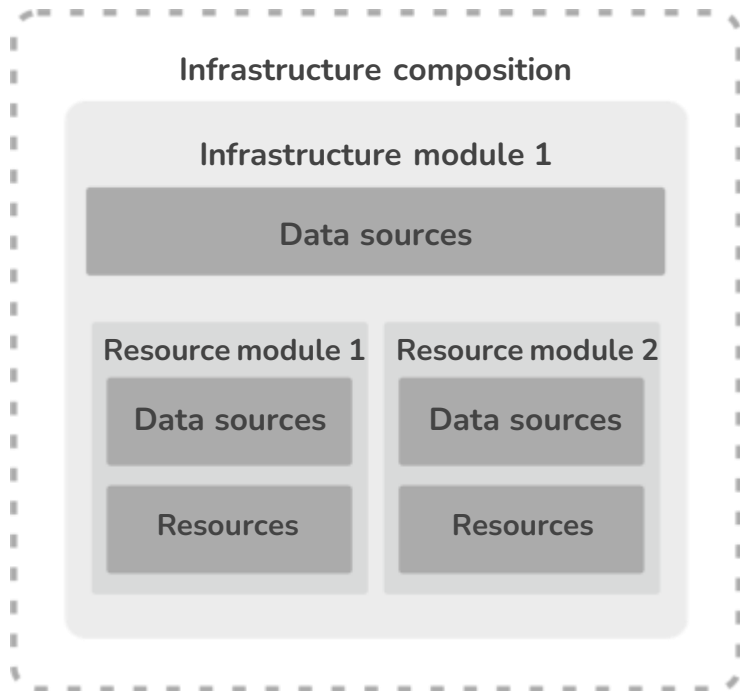


Виды модулей



здать
вопрос

В физическом смысле модули ничем не отличаются друг от друга.
Но в логическом они могут различаться на **инфраструктурные** и **ресурсные**.





Использование модулей



здать
вопрос

Модуль объявляется с помощью блока **module**, в который передается мета-аргумент **source** и принимаемые аргументы.

Пример:

```
module "servers" {  
  source = "./app-cluster"  
  
  servers = 5  
}
```



При использовании Terraform Registry можно также использовать переменную **version** для указания версии модуля.



Terraform Registry



задать
вопрос

Модулей в Terraform Registry очень много. Возможно, для вашей цели уже что-то есть.

The screenshot shows the Terraform Registry interface. On the left, there's a sidebar with 'Providers' and 'Modules' tabs. The 'Modules' tab is active, showing a list of modules. The main content area is titled 'Modules' and includes a description: 'Modules are self-contained packages of Terraform configurations that are managed as a group.' Below this, there are four module entries, each with a Terraform logo, a title, a description, a version number, a download count, and a provider name.

Module Name	Description	Version	Downloads	Provider
terraform-aws-modules / vpc	Terraform module which creates VPC resources on AWS	0.6 days ago	0.5M	aws
terraform-aws-modules / security-group	Terraform module which creates EC2-VPC security groups on AWS	0.2 months ago	0.1M	aws
terraform-aws-modules / eks	Terraform module to create an Elastic Kubernetes (EKS) cluster and associated worker instances on AWS	0.2 months ago	0.0M	aws
terraform-aws-modules / rds	Terraform module which creates RDS resources on AWS	0.24 days ago	0.1M	aws



здать
вопрос

Другие источники модулей

Помимо локальных директорий и Terraform Registry в качестве **source** можно использовать:

GitHub: `source = "github.com/hashicorp/example"`

Bitbucket: `source = "bitbucket.org/hashicorp/terraform-consul-aws"`

Git: `source = "git::<https://example.com/vpc.git?ref=v1.2.0>"`

Mercurial: `source = "hg::<http://example.com/vpc.hg>"`

HTTP URLs: `source = "<https://example.com/vpc.zip>"` (zip, tar.bz2, tar.gz, tar.xz)

S3 buckets: `source = "s3::<https://s3-eu-west-1.amazonaws.com/mods/vpc.zip>"`

GCS buckets: `source = "gcs::<https://www.googleapis.com/storage/v1/vpc.zip>"`



Задание 2.1: Создание модуля



здать
вопрос

1. Рядом с директорией проекта создайте директорию `module_service`. Переместите в нее файлы `service.tf` и `outputs.tf` — они касаются только сервиса
2. Из файла `locals.tf` переместите переменные, касающиеся сервиса, в такой же файл в директории модуля. Не забудьте обернуть их в блок `locals{ }`
3. Из файла `variables.tf` переместите объявление переменных `service_*` в такой же файл в директории модуля. Переменную `aws_region` скопируйте — она должна быть и там, и там
4. Создайте для модуля переменные `ecs_cluster_id`, `sg_lb`, `sg_outbound`, `vpc_id` типа `string` и переменную `subnets_ids` типа `list(string)`
5. Теперь необходимо в файле `service.tf` заменить обращение к ресурсам корневого модуля на созданные переменные. Список ресурсов для замены соответственно:
`aws_ecs_cluster.main.id`, `aws_security_group.lb.id`,
`aws_security_group.outbound.id`, `aws_vpc.main.id`, `aws_subnet.public.*.id`



задать
вопрос

Задание 2.2: Использование модуля

Теперь, в корневом модуле, можно создать новый файл **services.tf**, в котором объявить сервис из модуля:

Пример:

```
module "service" {  
  source = "../module_service"  
  aws_region = var.aws_region  
  ...  
}
```

Помимо них передайте переменные **service_name = "nginx"** и **service_image = "nginx:1.19"**

Также сделайте новый файл **outputs.tf** и сделайте в нем **output** с выходным значением из модуля сервиса — **service_dns_name**.

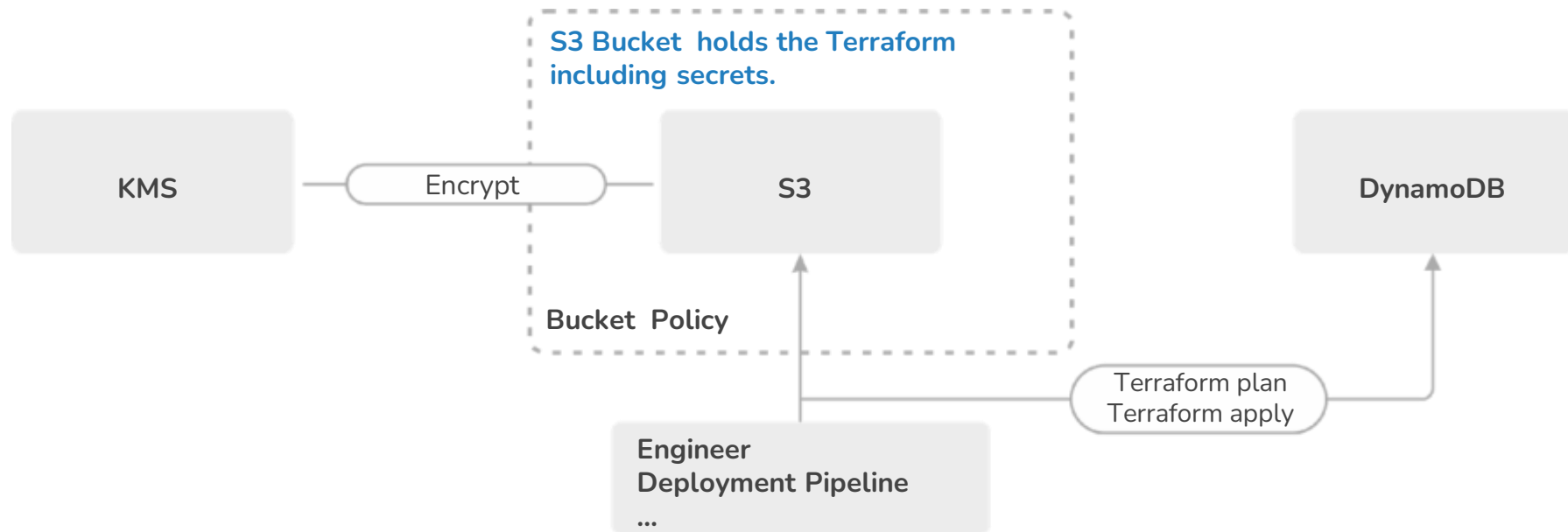


Удаленное хранение стейта



здать
вопрос

Для совместной работы инженеров стейт обычно хранят удаленно. Он может быть в зашифрованном виде, может поддерживать блокировки — все это зависит от провайдера. Для работы со стейтом используется команда **terraform state**





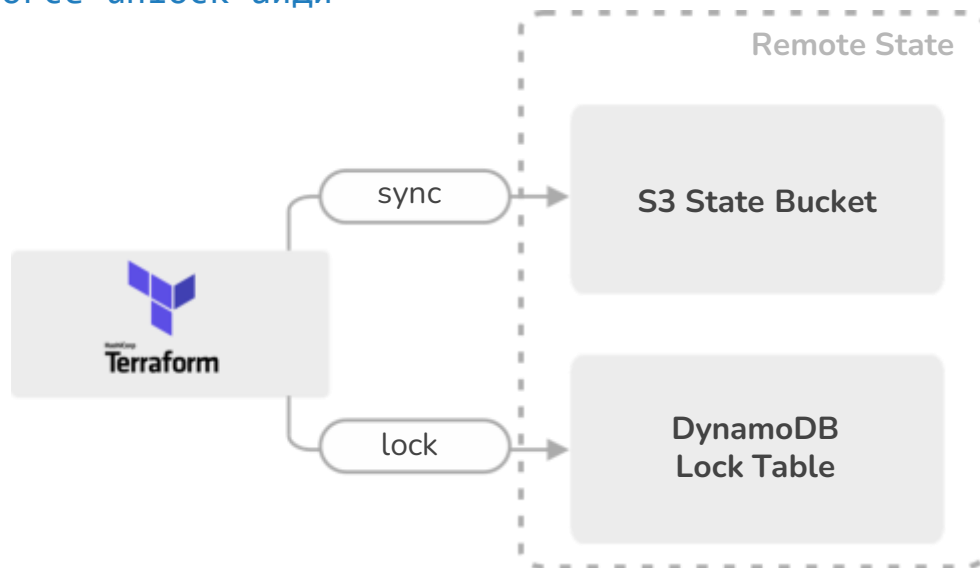
Блокировки стейта



здать
вопрос

Чтобы несколько инженеров не могли одновременно изменять одни и те же ресурсы существует блокировка стейта. Работает по принципу семафора — при использовании стейт блокируется, при завершении использования разблокируется.

Для принудительной разблокировки используется команда
`terraform force-unlock` айд





задать
вопрос

Задание 3.1: Подготовка хранения стейта

Для хранения стейта в S3 потребуется бакет. Его можно создать вручную, а можно с помощью Terraform.

1. Рядом с вашей директорией корневого модуля создайте директорию `state_bucket` и скопируйте в нее [versions.tf](#), [providers.tf](#). Также из [variables.tf](#) перенесите переменную `aws_region`.
2. Создайте файл [main.tf](#), в который запишите следующее:

```
resource "aws_s3_bucket" "terraform_state" {
  bucket = "super-unique-tfstate-bucket"
  versioning {
    enabled = true
  }
}

resource "aws_dynamodb_table" "terraform_state_lock" {
  name             = "super-unique-tfstate-lock"
  read_capacity    = 1
  write_capacity   = 1
  hash_key         = "LockID"
  attribute {
    name = "LockID"
    type = "S"
  }
}
```

3. Придумайте уникальные значения для `bucket` в `aws_s3_bucket` и `name` в `aws_dynamodb_table` (таковы особенности S3 в AWS)
4. Примените инфраструктуру



здать
вопрос

Задание 3.2: Удаленное хранение стейта

Теперь изменим хранение стейта в корневом модуле

1. Добавьте в блок `terraform` блок `backend`, заменив `bucket` и `dynamodb_table` на ваши переменные.

```
terraform {  
  ...  
  backend "s3" {  
    bucket          = "super-unique-tfstate-bucket"  
    dynamodb_table = "super-unique-tfstate-lock"  
    key             = "infra.tfstate"  
    region          = "us-east-1"  
    encrypt         = "true"  
  }  
}
```

2. Инициализируйте репозиторий. Если вы не удаляли предыдущую инфраструктуру — вам будет предложено перенести стейт в новый бэкэнд.
3. Примените инфраструктуру и убедитесь, что изменений нет.



здать
вопрос

Remote state

Для получения информации из другого стейта может использоваться дата-ресурс `terraform_remote_state`. Он позволяет получить данные из output. Для работы этого дата-ресурса не требуется никаких провайдеров.

Пример:

```
data "terraform_remote_state" "infra" {
  backend = "s3"

  config = {
    bucket = "super-unique-tfstate-bucket"
    key    = "infra.tfstate"
    region = "us-east-1"
  }
}

resource "aws_instance" "foo" {
  # ...
  subnet_id = data.terraform_remote_state.infra.outputs.subnet_id
}
```




здать
вопрос

Задание 4: Использование remote state

Давайте выделим создание сервиса в отдельный стейт. С помощью `remote_state` мы возьмем некоторые необходимые данные из корневого стейта.

1. Выделите создание сервиса в отдельный стейт, переместив файлы `services.tf` и `outputs.tf` в новую директорию — **services**, рядом с директориями **infra** и **module_service**. Скопируйте переменную **aws_region** из `variables.tf`. Также скопируйте в нее файл **providers.tf** и **versions.tf**, заменив в последнем переменную **key** стейта.
2. Создайте файл `data.tf` в директории `services` и объявите дата-ресурс **terraform_remote_state**, указав в качестве backend "**s3**", а внутри config указав **bucket = имя созданного бакета**, **key = ключ корневого стейта** (`infra.tfstate`), **region = регион**, который используется (`us-east-1`).
3. Для использования переменных их требуется передать в **output** корневого стейта. Объявите output переменные.
4. В объявлении сервиса замените ссылки на ресурсы на **data.terraform_remote_state.infra.outputs.переменная**
5. Примените изменения сначала на инфраструктурном стейте, а после на созданном, чтобы сначала создать переменные, а потом их взять.



Работа со стейтом



здать
вопрос

Для работы со стейтом используется команда `terraform state` с одной из следующих подкоманд:

`list`

— показать список ресурсов в стейте

`mv`

— переместить/переименовать ресурс внутри стейта

`pull`

— загрузить и вывести стейт в stdout

`push`

— обновить стейт с помощью файла

`replace-provider`

— заменить провайдер в стейте

`rm`

— удалить объект (ресурс/модуль) из стейта

`show`

— показать ресурс из стейта





здать
вопрос

Задание 5: Работа со стейтом

В качестве примера работы со стейтом давайте удалим ресурс из стейта, а после импортируем его обратно.

1. Зайдите в стейт с описанием сервисов. Введите **terraform state list** для получения списка ресурсов в стейте
2. Возьмите адрес ресурса **aws_ecs_service** (пример: **module.service.aws_ecs_service.main**) и подставьте его в команду **terraform state show** для отображения информации о ресурсе
3. Теперь удалим ресурс из стейта: **terraform state rm** адрес. Это удалит ресурс только из стейта, не из самого облачного провайдера — в нем он продолжает существовать
4. Импортируем ресурс обратно в стейт. Для этого введите **terraform import адрес кластер/сервис**. В нашем случае кластер называется main, а сервис nginx
5. Введите **terraform state list** и убедитесь, что ресурс снова в стейте
6. Удалите всю инфраструктуру из директорий services и infra (необходимо для следующего задания).

Важно: удалять нужно сначала services



Восстановление стейта



здать
вопрос

Если стейт был утерян — дальнейшая работа с ресурсами не может быть продолжена. Стейт требуется восстановить: либо из резервной копии, либо вручную.

Для ручного восстановления потребуются идентификаторы каждого ресурса.





Обновление между версиями Terraform



здать
вопрос

Так как версии Терраформ выходят часто, код приходится обновлять. Например, последние 3 мажорных версии вышли за последние полгода:

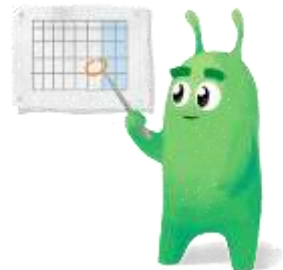
0.15.0 — 14 апреля 2021

0.14.0 — 2 декабря 2020

0.13.0 — 10 августа 2020

Для обновления между версиями в Terraform поставляют команды. Например, **terraform 0.13upgrade** обновит исходный код версии 0.12 для работы с версией 0.13. Стейт же будет обновлен при внесении изменений в инфраструктуру.

Важно: обновления всегда происходят только на одну версию.





tfswitch



здать
вопрос

Незаменимым помощником при обновлении и работе с разными версиями Terraform является сторонняя утилита **tfswitch**. Она позволяет переключаться между разными версиями Terraform.

Работать с ней просто — достаточно перед работой ввести **tfswitch** и либо версия Terraform установится на указанную в **required_version**, либо будет предложено выбрать необходимую версию.

```
$ tfswitch
Reading required version from terraform file, constraint: ~> 0.13.0
Matched version: 0.13.6
Downloading https://releases.hashicorp.com/terraform/0.13.6/terraform
Downloading...
35671246 bytes downloaded.
Switched terraform to version "0.13.6"
```



Задание 6: Обновление версии Terraform



здать
вопрос

Обновления производятся сначала над второстепенными модулями, а после над корневыми. Каждый воркспейс обновляется отдельно.

1. Прежде всего, потребуется обновить модуль создания сервиса.
В `module_service/versions.tf` поменяйте **required_version** с `~>` на `>=`
2. Перейдем в директорию `services`. В файле `versions.tf` измените параметр **required_version** на `~>0.14.0`
3. Выполните команду **tfswitch**
4. Инициализируйте репозиторий и примените инфраструктуру.
Убедитесь, что нет изменений. Проверьте в стеите, что версия Terraform обновилась;
5. Повторите действия для директории **infra**





Задание 7: Удаление ресурсов



задать
вопрос

Последнее задание — удалить все созданные ресурсы из всех стейтов:

terraform destroy

При удалении обратите внимание на зависимости между стейтами (services зависит от infra, а значит должен быть удален первым).



Завершение дня 2



задать
вопрос

Поздравляю! Это завершение дня 2 и нашего интенсива.

Сегодня мы рассмотрели:

- Terraform модули — для чего они нужны и как с ними работать
- Удаленное хранение стейта и работу с ним
- Обновление между версиями Terraform

Надеемся, что в процессе выполнения вам было интересно и вы получили новые знания, которые сможете применять на практике :)

Спасибо!

Спикер:
Павел Замошин

