

Использование Terraform в команде

Елисей Ильин
DevOps-инженер в Itransition



Елисей Ильин

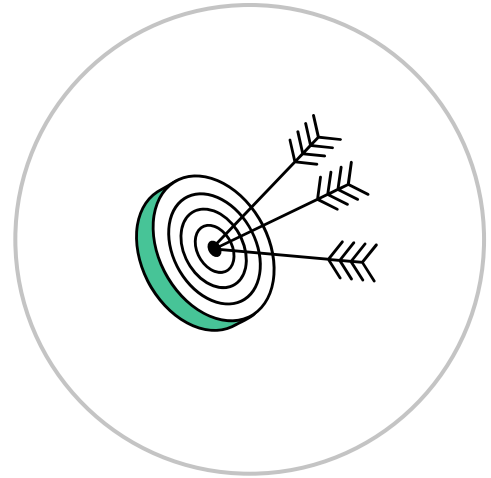
О спикере:

- DevOps-инженер
- Опыт работы в IT 6 лет



Цели занятия

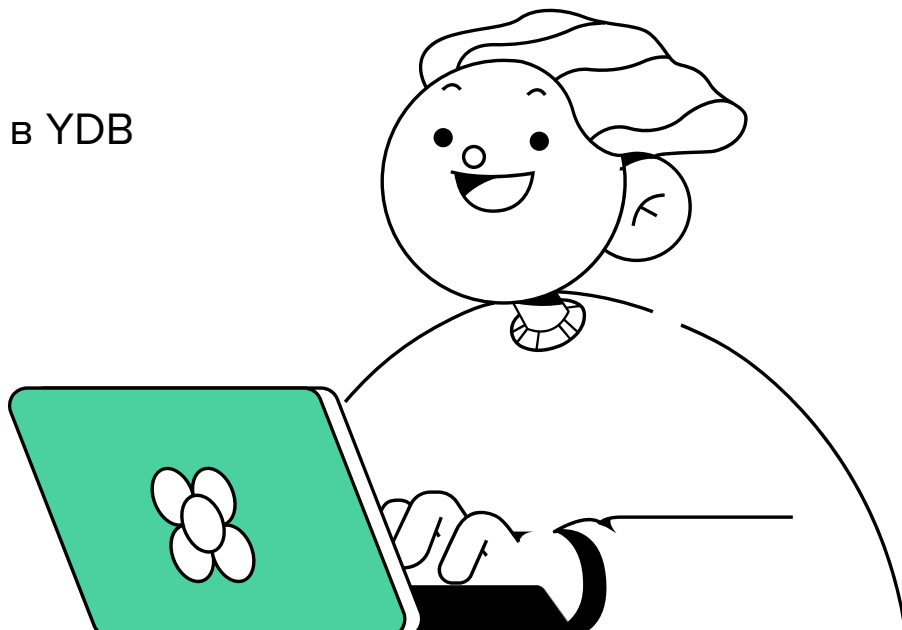
- Узнать, чем отличаются одиночная и командная разработка кода Terraform
- Научиться безопасно хранить и использовать remote State



План занятия

- 1 Работа в команде
- 2 Name & code conventions
- 3 Terraform backend
- 4 Настройка backend S3 с блокировкой в YDB
- 5 Security scan
- 6 Code & Plan review
- 7 Полезные практики

Нажмите на раздел для перехода

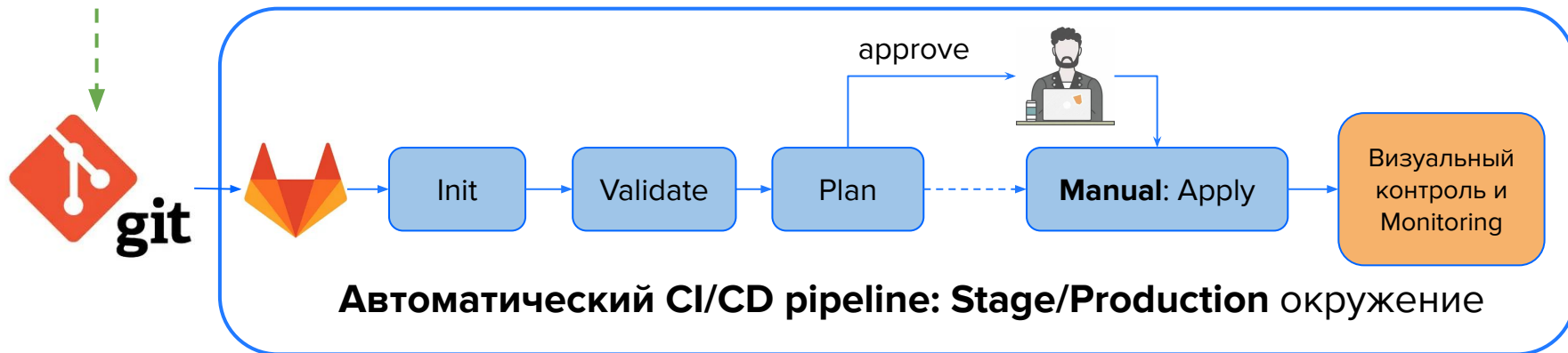
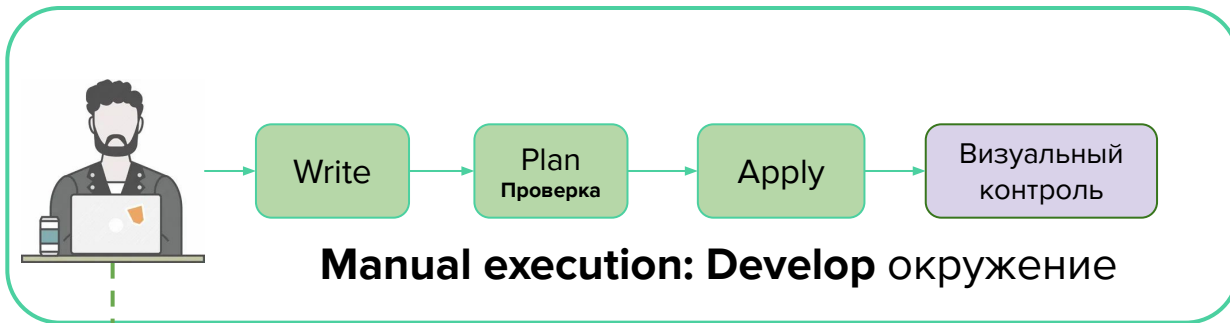


Работа в команде



1

Одиночная разработка в Terraform



Подводные камни Terraform при работе в команде

Проблема	Способы решения
Документация модулей и их переменных, окружений, костылей	Валидация переменных в Модулях, Terraform-docs, контроль через PR
Name & Code conventions	terraform-style-guide ; TFLINT
Версионирование terraform, плагинов, модулей	Привязка версий к CI/CD pipelines
Совместный доступ к State	Remote state
Повреждение State при одновременной записи	State locking
Code & Plan review; CI/CD	GitLab; Github + any Ci/CD
Security	Static code analysis Checkov , шифрование remote state

Name & code conventions



2

TFLINT

Помогает найти в коде неиспользуемые переменные, незафиксированные версии, неопределенный тип переменных и прочее по [списку правил](#).

Также можно «прикрутить» дополнительные плагины по name convention.

[Релизы проекта](#)

```
Warning: variable "vm_db_name" is  
declared but not used  
(terraform_unused_declarations)  
on variables.tf line 51:  
51: variable "tflint_example"
```

Вспомните, как во втором домашнем задании вы вручную искали неиспользуемые более переменные :)

TFLINT настройка

Конфигурация правил **tflint** задается в файле `~/.tflint.hcl` или с помощью ключа `--config=<path>`.

Поддерживается подключение плагинов

```
config {  
  format = "compact"  
  plugin_dir = "~/.tflint.d/plugins"  
  module = true  
}  
plugin "terraform" {  
  enabled = true  
  preset  = "recommended"  
}
```

TFLINT запуск

Проще всего запустить через docker из каталога с кодом terraform

```
docker run --rm -v "$(pwd):/tflint" ghcr.io/terraform-linters/tflint /tflint
```

Блок валидации переменных

Если вы используете модули, написанные разнородной командой, важно не только предоставлять пример вызова модуля в документации к нему, но и валидировать переменные как по типу, так и по содержимому.

```
variable "platform" {  
  type          = string  
  default       = "standard-v1"  
  description   = "Example to validate VM platform."  
  validation {  
    condition    = contains(["standard-v1", "standard-v2", "standard-v3"], var.platform)  
    error_message = "Invalid platform provided."  
  }  
}  
  
variable "api_token" {  
  type          = string  
  description   = "API token"  
  validation {  
    condition    = length(var.api_token) >= 32  
    error_message = "Must be at least 32 character long API token."  
  }  
}
```

Блок валидации переменных

```
module "test-vm" {  
  ...  
  platform = "standard-v4"  
  ...  
}
```

Error: Invalid value for variable

on main.tf line 42, in module "test-vm":

```
42:   platform = "standard-v4"  
    |_____|  
    |  
    | var.platform is "standard-v4"
```

Invalid platform provided.

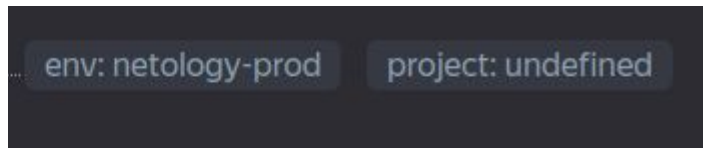
This was checked by the validation rule at
.terraform/modules/test-vm/variables.tf:27,4-14.

Лейблы ресурсов

Лейблы помогают выгружать финансовую статистику по проектам из облака.

Пример labels по-умолчанию в модуле VM, если они не были переданы при вызове модуля.

```
locals {  
  labels = length(keys(var.labels)) > 0 ?  
  var.labels: {  
    "env"      = var.env_name  
    "project" = "undefined"  
  }  
}
```



Описание ресурсов

Описание может пригодиться другим сотрудникам — экономистам, бухгалтерам, сотрудникам тех.поддержки.

Дефолтное описание, можно использовать для свежесозданных ресурсов на этапе внедрения.

Сразу видно, что его нужно доработать.

```
variable "description" {  
  type = string  
  default = "TODO: description;  
{{terraform managed}}"  
}
```

Labels

Description

TODO: description; {{terraform managed}}

Terraform backend



3



Terraform backend — это метод хранения и доступа к State. По типам делится на **local** и **remote**.
Некоторые виды **backend** поддерживают **State locking**



State locking — отметка о монопольном доступе к State. Запрещает операции записи в State для иных источников. Исключает повреждение State.

Сравнение популярных видов backend

Вид	Название backend	Поддержка блокировки
local	backend "local"	Нет
Generic S3	backend "s3"	Нет
Amazon S3 bucket	backend "s3"	Да, хранение в DynamoDB
Yandex Object Storage	backend "s3"	Да, хранение в YDB (аналог DynamoDB)
Google Cloud Storage	backend "gcs"	Да, хранение в файле .tflock прямо в S3
Hashicorp Consul	backend "consul"	Да, интегрирована
Postgres DB	backend "pg"	Да, на уровне блокировок БД
HTTP (например Gitlab)	backend "http"	Да, 200: OK , 423: Locked or 409: Conflict

Local backend

- Backend по умолчанию, настройка опциональна
- State сохраняется в Root Module
- Переопределить путь можно с помощью блока **backend "local" {..}**

```
terraform {  
  backend "local" {  
    path = "<относительный путь>/terraform.tfstate"  
  }  
}
```

Remote s3 backend

Блок **backend "s3" {..}**

```
terraform {  
  backend "s3" {  
    bucket = "bucket_name"  
    key     = "path/terraform.tfstate"  
    region = "us-east-1"  
    access_key = "..." #Только для примера! Не хардкодим секретные данные!  
    secret_key = "..." #Только для примера! Не хардкодим секретные данные!  
    dynamodb_table = "tfstate-lock" #таблица блокировок  
    encrypt = true #шифрование State сервером terraform  
  }  
}
```

Backend Security

Использовать переменные в **backend** нельзя, но и хардкодить секреты недопустимо

#Недопустимый пример

```
backend "s3" {  
  ...  
  access_key = var.s3_access_key  
  secret_key = var.s3_secret_key  
  ...  
}
```

Error: Variables not allowed

on providers.tf lines 13,14 , in terraform:
13: access_key = var.s3_access_key
14: secret_key = var.s3_secret_key

Variables may not be used here.

Backend Security

Вместо этого нужно инициализировать Terraform командой:

```
terraform init -backend-config="access_key=<s3_access_key>" -backend-config="secret_key=<s3_secret_key>"
```

Ключи доступа к backend будут сохранены в открытом виде в **локальном** файле **.terraform/terraform.tfstate**

Миграция local backend -> backend S3

```
terraform init -backend-config="access_key=<s3_access_key>" -backend-config="secret_key=<s3_secret_key>"
```

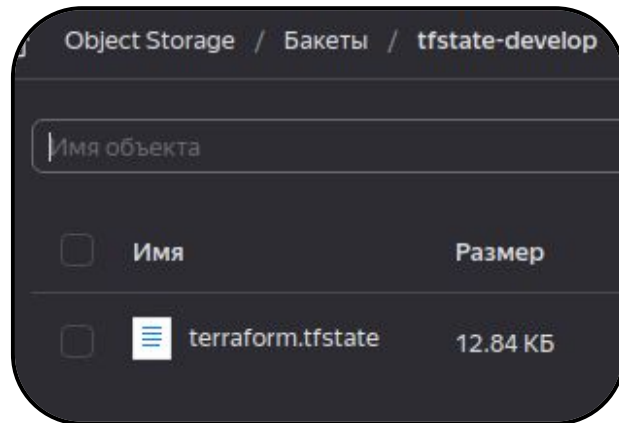
Initializing the backend...

Do you want to copy existing state to the new backend?

Pre-existing state was found while migrating the previous "local" backend to the newly configured "s3" backend. No existing state was found in the newly configured "s3" backend. Do you want to copy this state to the new "s3" backend? Enter "yes" to copy and "no" to start with an empty state.

Enter a value: yes

Terraform has been successfully initialized



Полезные команды при работе с remote s3 State

<code>terraform state pull > example.tfstate</code>	Скачивает State из backend в файл (backup)
<code>terraform state push example.tfstate -force</code>	Принудительная загрузка State в backend (restore backup)
<code>terraform plan -out=TFPLAN_FILENAME</code>	Сохраняет план изменения в файл-артефакт. Необходимо для CI/CD pipelines и Code&Plan review
<code>terraform apply TFPLAN_FILENAME</code>	Исполняет ранее сохраненный план изменений
<code>terraform force-unlock <LOCK_ID></code>	Ручная разблокировка State в случае проблем. Помогает в случае прерывания процесса, например Ctrl-c.

Если ваш terraform console блокирует remote State для всей команды, то решение следующее:

```
terraform state pull > ~/test.tfstate && terraform console -state=~/test.tfstate
```

Настройка backend S3 с блокировкой в YDB



4

free tier S3 в YC

Yandex Object Storage

Каждый месяц **не тарифицируются** ресурсы стандартного хранилища:

- первый 1 ГБ в месяц хранения
- первые 10 000 операций PUT, POST
- первые 100 000 операций GET, HEAD

default Object Storage / Бакеты / Новый бакет

Новый бакет

Имя* ? tfstate-develop

Макс. размер 1 ГБ ☐ Без ограничения

Доступ на чтение объектов ? Ограниченный Публичный

Доступ к списку объектов ? Ограниченный Публичный

Доступ на чтение настроек ? Ограниченный Публичный

Класс хранилища ? Стандартное Холодное Ледяное

Создать бакет

Имя	Занято	Кол-во объектов	
tfstate-develop	0 Б / 1 ГБ	0 объектов	...

free tier Managed YDB в YC

Serverless Managed Service for YDB

Каждый месяц **не тарифицируются** первые:

- 1 000 000 операций (в единицах Request Unit)
- 1 ГБ/месяц хранения данных

Инструкция YDB

The screenshot shows the 'Новая база данных' (New Database) form in the YDB Managed Service console. The form includes the following fields and options:

- Имя*** (Name): tfstate-lock
- Метки** (Tags): A table with columns 'Ключ' (Key) and 'Значение' (Value). A 'Добавить' (Add) button is below it.
- Тип базы данных** (Database Type): Radio buttons for 'Serverless' (selected) and 'Dedicated'.
- Ограничения** (Limits):
 - Пропускная способность, RU/c** (Throughput, RU/c): Input field with '10' and a checkbox for 'Без ограничения' (No limit).
 - Объем данных, ГБ** (Data Volume, GB): Input field with '1'.
- Тарификация** (Billing):
 - Выделенная пропускная способность, RU/c** (Dedicated throughput, RU/c): Input field with '0'. Below it, a note states: 'Почасовой тариф на выделенную пропускную способность не применяется. Все запросы оцениваются по тарифу «за фактическое потребление»' (Hourly rate for dedicated throughput is not applied. All requests are evaluated by the 'pay-as-you-go' rate).

At the bottom, there are two buttons: 'Создать базу данных' (Create Database) and 'Отменить' (Cancel).

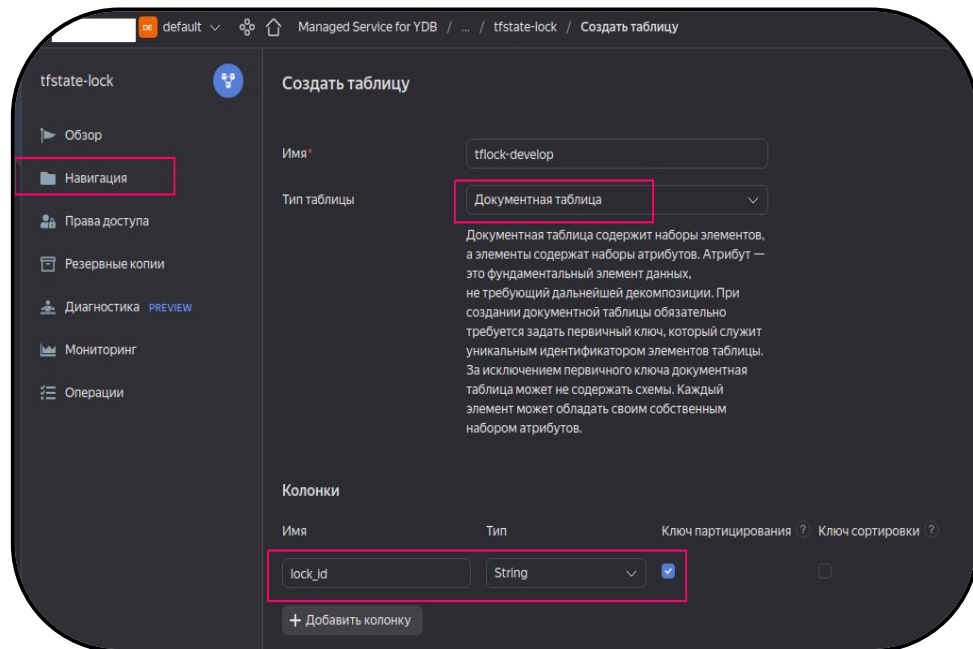
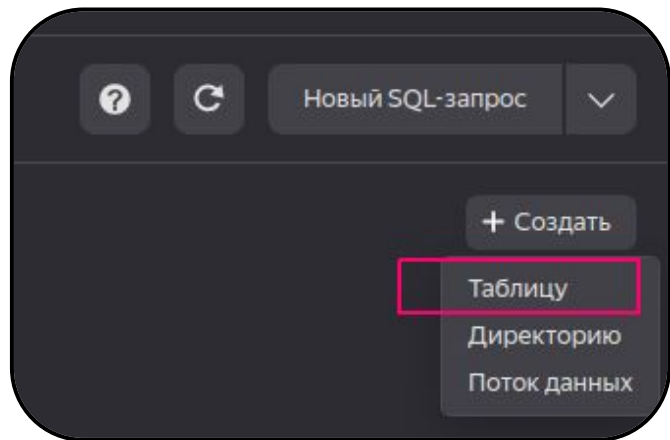
YDB api endpoint

Во вкладке «Обзор» найдите Document API эндпоинт, он нужен для подключения к YDB.

Document API эндпоинт

Эндпоинт <https://docapi.serverless.yandexcloud.net/ru-central1/b1gn3ndpua1j6jaabf79/etnij6ph9brodq9ohs8d>

Создание таблицы блокировок в YDB



имя колонки: LockID
тип: string

Создание сервисного аккаунта для S3 и YDB в YC

[Статья](#) о сервисных аккаунтах

Создание сервисного аккаунта

Имя ?

tfstate

Описание ?

handmade account for terraform tfstate

Роли в каталоге

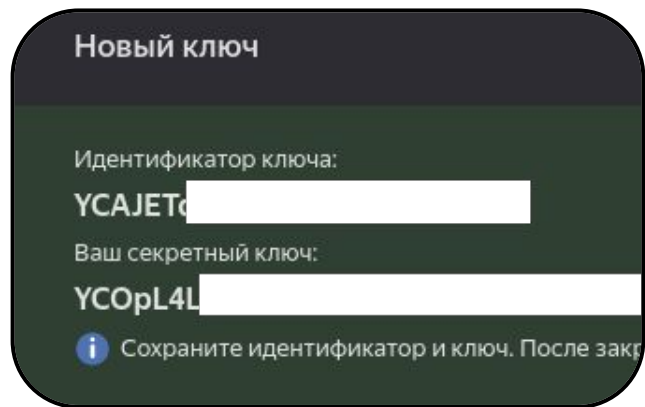
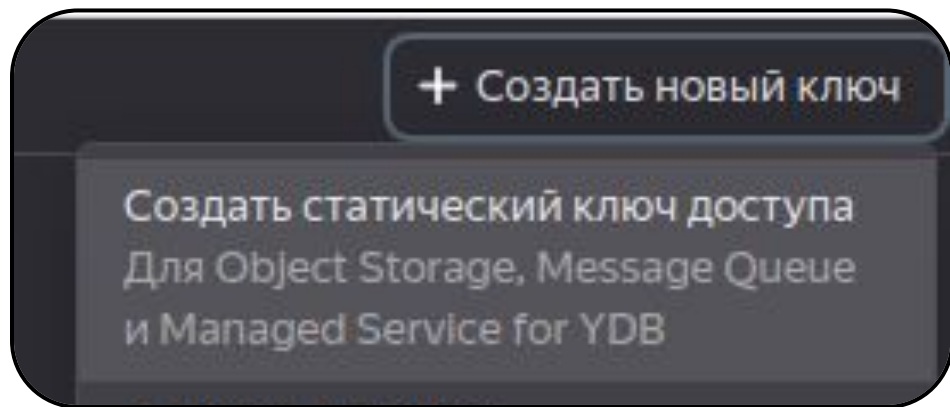
+ Добавить роль

Отменить

Создать


Создание ключа доступа для сервисного аккаунта в YC

[Статья](#) о создании статических ключей доступа



Выдача доступов в S3 для сервисного аккаунта в YC

Редактирование ACL


 tfstate-develop 0 Б, 0 объектов

Выберите пользователя

READ

Добавить

Пользователь

 tfstate
Сервисный аккаунт

Разрешения

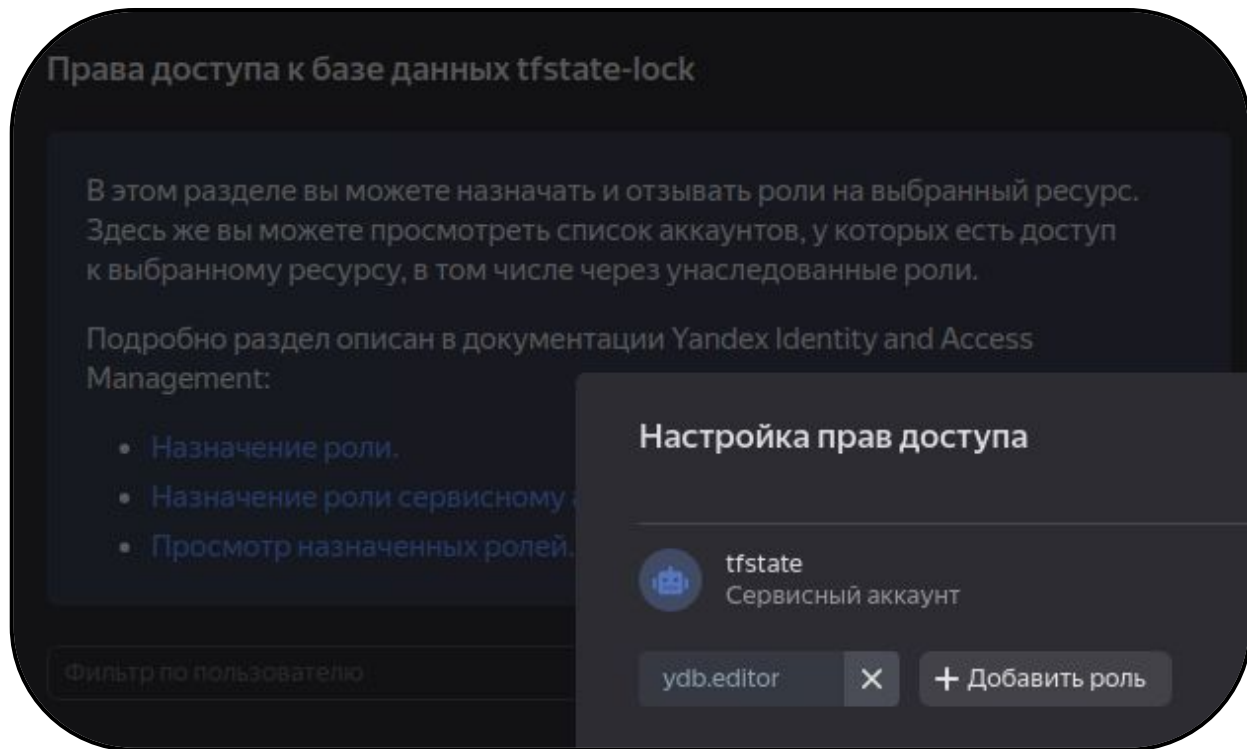
READ и WRITE

Отменить

Отменить

Сохранить

Выдача доступов YDB для сервисного аккаунта в YC

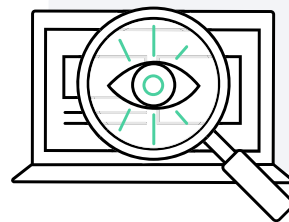


Пример настройки backend s3 для YC storage с блокировками State

```
backend "s3" {  
    endpoint      = "storage.yandexcloud.net"  
    bucket        = "tfstate-develop"  
    region        = "ru-central1"  
    key           = "terraform.tfstate"  
  
    skip_region_validation      = true  
    skip_credentials_validation = true  
  
    dynamodb_endpoint = "https://docapi.serverless.yandexcloud.net/ru-central1/xxx/yyy"  
    dynamodb_table    = "tfstate-lock-develop"  
}
```

Демонстрация работы

- Настройка remote state в Yandex object storage с блокировкой в YDB
- Миграция local State в S3 на примере демонстрации в лекции «Продвинутые методы работы с Terraform»
-



Security scan



5

Checkov

[Инструкция](#) по установке и настройке

Поддерживается множество IaC-tools:

- Terraform
- K8s
- Helm
- Kustomize
- Ansible
- Dockerfile
- Gitlab и другие

Yandex Provider поддерживается checkov

В примере Checkov проверил настройки security groups и обнаружил ключ от s3 bucket в открытом виде

```
Check: CKV_SECRET_6: "Base64 High Entropy String"
```

```
    FAILED for resource:
```

```
    9ac6c31563686ca9a2347974391f8555b3cef93c
```

```
    Severity: LOW
```

```
    File: /providers.tf:14-15
```

```
    Guide:
```

```
    https://docs.bridgecrew.io/docs/git_secrets_6
```

```
14 |         # secret_key =
```

```
    "YC0pL4*****"
```

Запуск Chekov

Проще всего использовать docker образ для проверки текущего каталога:

```
docker pull bridgecrew/checkov
```

```
docker run --rm --tty --volume $(pwd):/tf --workdir /tf bridgecrew/checkov \  
--download-external-modules true --directory /tf
```

Code & Plan review



6



В CI/CD всегда используйте конструкцию вида:

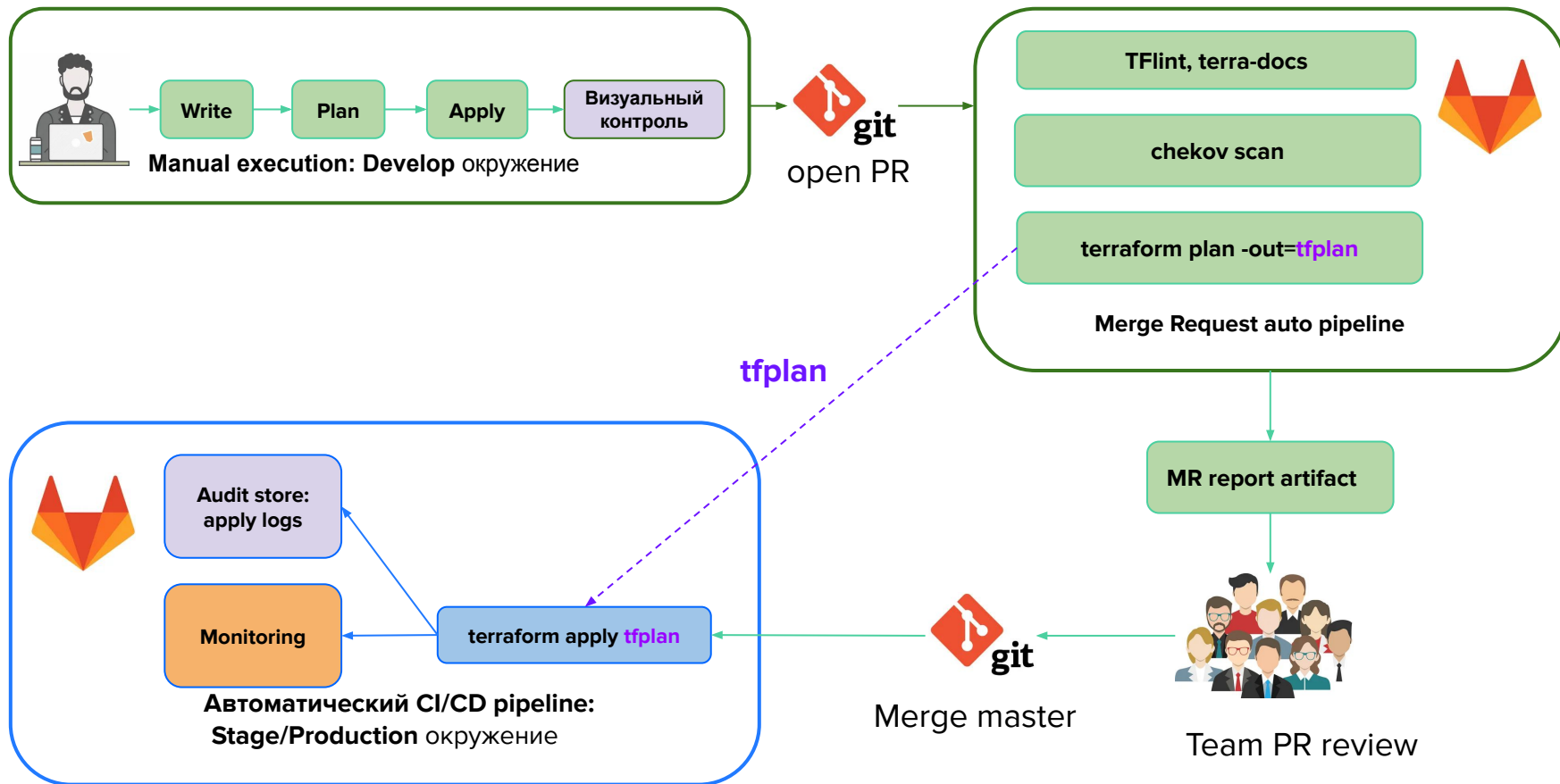
1. terraform plan -out=**tfplan**
2. plan ревью
3. terraform apply **tfplan**

Это не критично при ручном запуске Terraform.

В CI/CD пайплайне между **terraform plan** и **terraform apply** в репозитории могут появиться новые коммиты, и новый plan будет отличаться от того, что был проверен в ходе ревью кода.

Важно! Не забудьте добавить **tfplan** в **.gitignore** .

CI/CD на примере Gitlab



Скрипт пайплайна на примере Gitlab

```
workflow:
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event" #Запускаем pipeline если это Merge request
      when: always
  stages:
    - lint_plan
    - apply
  before_script: #Секретные переменные сохранены в Gitlab Vars. before_script выполняется в каждом этапе
    - yc config set service-account-key "${CI_CLOUD_ROBOT}" #Авторизуемся в УС через сервисную УС
    - export cloud_access_token=$(yc iam create-token)
    - terraform init -backend-config="secret_key=${CI_S3_SECRET_KEY_NETOTFSTATE}" #Авторизуемся в backend
  lint_plan:
    script:
      - tflint>tflint; checkov>checkov #сохраняем результаты сканирования в артефакт, игнорируем не 0 exit code
      - terraform plan -var="cloud_access_token=${cloud_access_token}" -out=outfile #Сохраняем plan в артефакт
    allow_failure:
      exit_codes: [ 2, 3 ] #разрешаем выполнение пайплайна в случае предупреждений от линтеров
  artifacts:
    when: always
    paths: [ "outfile", "tflint", "checkov" ] #Сохраняем ci-артефакты для code review и последующего применения
  apply:
    stage: apply
    script:
      - terraform apply "outfile" #Применяем ранее сохраненную конфигурацию
  when: manual #с ручным подтверждением. Или автоматически после подтверждения MR несколькими коллегами
  dependencies: [ "lint_plan" ] #этап apply выполняется только после этапа lint_plan
```

[Ссылка на GitHub](#)

Полезные практики



7

Полезные практики

- Не храните секреты в Git, используйте Hashicorp vault
- Фиксируйте версии зависимостей
- Пользуйтесь линтерами и анализаторами кода
- Разделяйте dev/stage/prod окружения
- Разделяйте компоненты окружения на логические элементы (сеть, права и доступы, БД, VM)
- Используйте модули и валидацию переменных. Полезно приложить пример вызова модуля
- Используйте remote state + locking+versioning
- Вносите изменения в Git master-ветку только через Pull/Merge request

Полезные практики

- Вносите изменения в Git только через Pull/Merge request
- Используйте `sensitive=true`, чтобы не светить важные переменные в плане
- Тегируйте все ресурсы, хотя бы тегами по-умолчанию (окружение + «terraform managed»)
- Вносите изменения в production среду только через CI/CD
- Изучите go lang ([ссылка](#)) и освоите тестирование Terraform кода
- Пишите сопровождающую документацию к своему проекту
- Изучайте release notes, когда хотите обновить версию Terraform
- Прочитайте всю доступную документацию Hashicorp по Terraform

Итоги занятия

Сегодня мы:

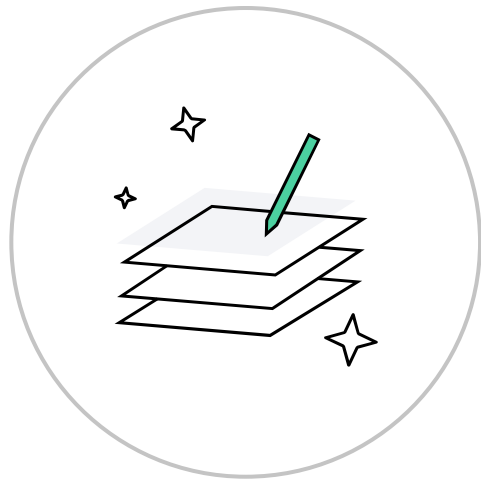
- 1 Изучили, какие проблемы могут возникнуть при работе с Terraform в команде
- 2 Научились успешно решать эти проблемы



Домашнее задание

Давайте посмотрим ваше домашнее задание.

- 1 Вопросы по домашней работе задавайте в чате группы
- 2 Задачи можно сдавать по частям
- 3 Зачёт по домашней работе ставят после того, как приняты все задачи



Дополнительные материалы

- [Naming conventions](#)
- [backends](#)
- [Yandex Object Storage](#)
- [Таблицы DynamoDB в YDB](#)



Подготовьте вопросы к разборному вебинару

Елисей Ильин
DevOps-инженер в Itransition

