

Средства командной работы



Андрей
Борю



Андрей Борю

Principal DevOps Engineer, Snapcart



План занятия

1. [Средства командной работы](#)
2. [Terraform cloud](#)
3. [Атлантис](#)
4. [Remote state](#)
5. [Модули](#)
6. [Домашнее задание](#)



Средства командной работы

Основные инструменты

- **Terraform Cloud / Terraform Enterprise**
 - Есть бесплатные тарифы, но полный функционал платный.
 - Хостится на сервера hashicorp.
- **Atlantis**
 - Решение с открытым исходным кодом.
 - Запускается на ваших серверах.

Зачем использовать эти инструменты?

- Прозрачность процессов
 - Если каждый запускает локально – не понятно что сейчас задеплоено.
 - Кто-то мог забыть закоммитить изменения.
 - Есть логи всех **plan** и **apply**.

Каждый может сделать PR в инфраструктуру

- Каждый может сделать **PR** и посмотреть **plan** своих изменений.
 - Это существенно ускоряет процессы: вместо того чтобы ставить задачу на простое изменение, разработчик просто делает **PR**.
- Все секретные данные будут скрыты в пределах этой системы.

Более качественный процесс ревью PR

- Можно проверить код.
- Сразу виден результат работы **terraform plan**.

Стандартизация рабочих процессов

- Система блокирует рабочее пространство (**workspace**) до применения изменений или ручного снятия блокировки.
- Помимо стандартных команд **plan** и **apply** можно добавить дополнительные инструкции.



Terraform Cloud

Terraform Cloud

Поддерживает несколько типов:

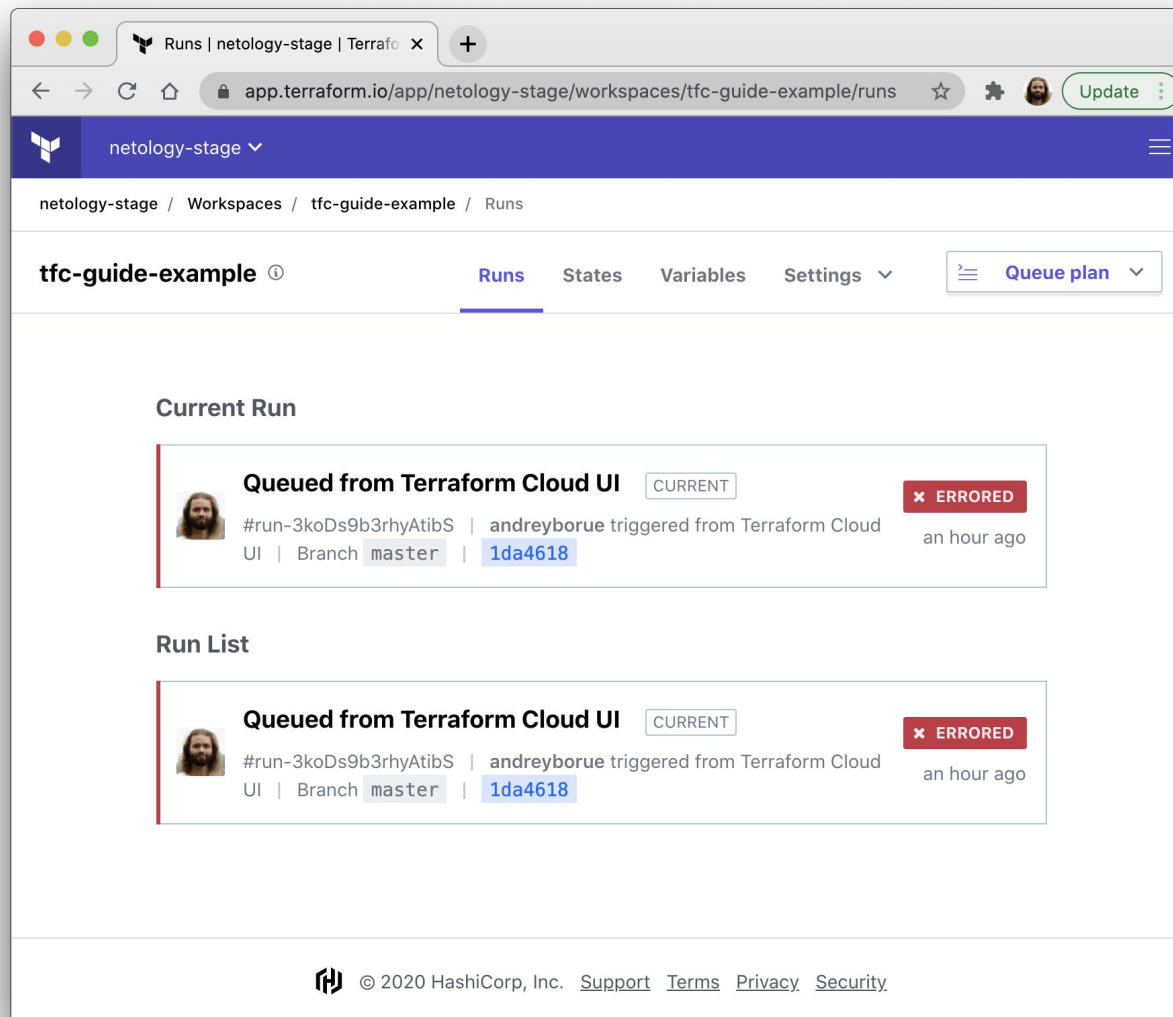
- На основе системы контроля версий,
- Просто remote backend запускаемый вручную,
- На основе API (много возможностей, но сложно использовать).

Terraform Cloud

Особенности:

- Поддерживаем remote state.
- Необходимо обеспечить доступ к api своих сервисов.
- Поддерживается командой терраформа.
- Есть бесплатный тарифный план.
- Хороший UI с возможностью ревью изменений.
- Интеграция с slack.
- Можно задавать переменные для разных окружений.

Интерфейс



Интерфейс

The screenshot displays the Terraform Cloud web interface. At the top, a browser window shows the URL `app.terraform.io/app/netology-stage/workspaces/tfc-guide-example/run-3koDs9b3rhyAtibS`. The page header includes the 'netology-stage' workspace name and a navigation menu with 'Runs', 'States', 'Variables', and 'Settings'. The 'Runs' tab is active, showing a list of runs for the 'tfc-guide-example' workspace. The first run, 'run-3koDs9b3rhyAtibS', is in an 'ERRORED' state, indicated by a red 'x' icon. The run was triggered by 'andreyborue' from the Terraform Cloud UI an hour ago. The status 'Plan errored' is shown with a red 'x' icon. Below this, a message states 'Apply will not run'. At the bottom, there is a comment section with the text 'Comment: Leave feedback or record a decision.' and an 'Add Comment' button. The footer of the page contains the HashiCorp logo, copyright information for 2020, and links to 'Support', 'Terms', 'Privacy', and 'Security'.

run-3koDs9b3rhyAtibS | Runs

app.terraform.io/app/netology-stage/workspaces/tfc-guide-example/run-3koDs9b3rhyAtibS

netology-stage

netology-stage / Workspaces / tfc-guide-example / Runs / run-3koDs9b3rhyAtibS

tfc-guide-example

Runs States Variables Settings

Queue plan

ERRORED Queued from Terraform Cloud UI

CURRENT

andreyborue triggered a run from Terraform Cloud UI an hour ago Run Details

Plan errored an hour ago

— Apply will not run

Comment: Leave feedback or record a decision.

Add Comment

© 2020 HashiCorp, Inc. [Support](#) [Terms](#) [Privacy](#) [Security](#)



Атлантис

Автоматизация pull request

Сайт: <https://www.runatlantis.io/>

Совместимо с:

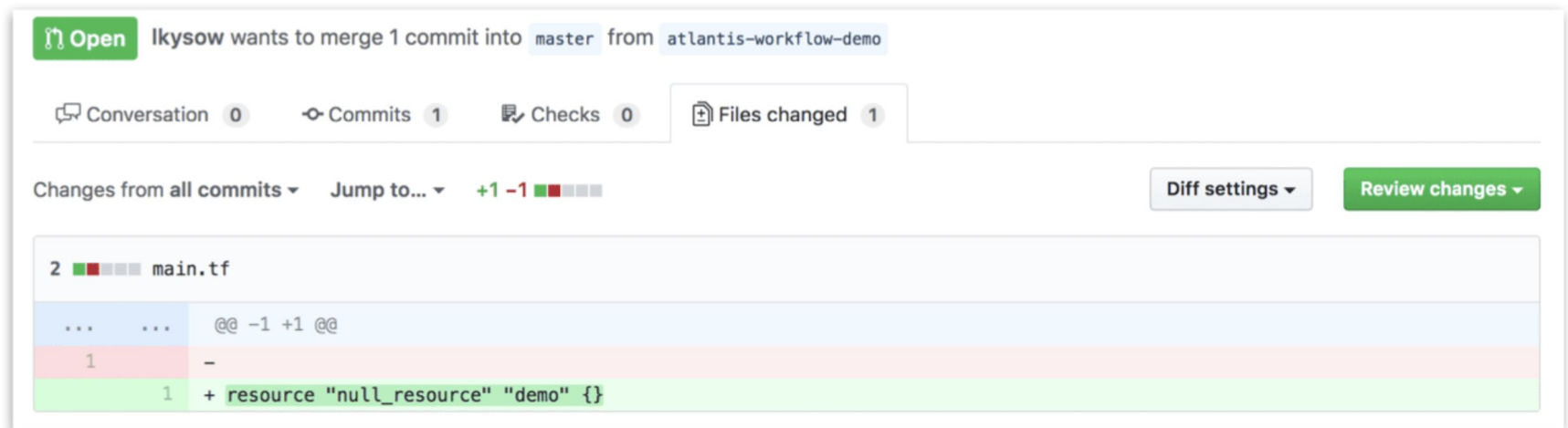
- GitHub
- GitLab
- Bitbucket
- Azure DevOps

Особенности

- Бесплатное решение.
- Поддерживает множество популярных хранилищ репозитория.
- Достаточно где угодно запустить докер контейнер с набором переменных окружения.

Шаг 1

Открывает pull request в репозитории с кодом terraform.



Шаг 2



atlantisbot commented





Ran Plan in dir: `.` workspace: `default`

Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

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:

+ `null_resource.demo`
id: `<computed>`
Plan: 1 to add, 0 to change, 0 to destroy.

-  To **apply** this plan, comment:
 - `atlantis apply -d .`
 -  To **delete** this plan click [here](#)
 -  To **plan** this project again, comment:
 - `atlantis plan -d .`
-
-  To **apply** all unapplied plans, comment:
 - `atlantis apply`

Атлантис

автоматически

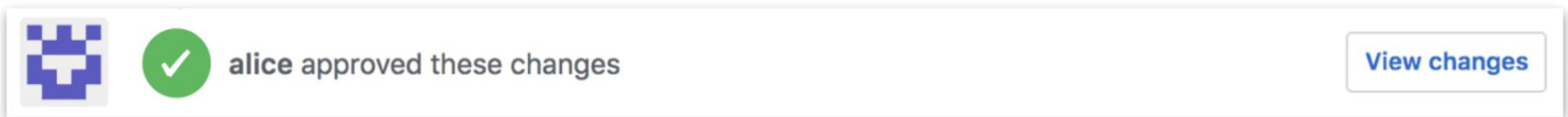
запускает **terraform**

plan и оставляет

комментарий к PR.

Шаг 3

Кто-то из вашей команды просматривает план и одобряет PR



Шаг 4

Вы оставляете комментарий **atlantis apply**



lkysow commented

atlantis apply

Шаг 5

Атлантис исполняет terraform apply и оставляет комментарий об этом



atlantisbot commented

Ran Apply in dir: `.` workspace: `default`

```
null_resource.demo: Creating...  
null_resource.demo: Creation complete after 0s (ID: 4542221565395344699)  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Шаг 6

Атлантис автоматически мержит PR



Pull request successfully merged and closed

You're all set—the `atlantis-workflow-...` branch can be safely deleted.

Delete branch

Настройка сервера

Давайте посмотрим пример конфига:

<https://www.runatlantis.io/docs/server-side-repo-config.html>

- Проверка апрува PR.
- Проверка "Mergeable" статуса.
- Разрешения изменять настройки внутри репозитория.
- Изменения стандартного воркфлоу атлантиса.

Настройка клиента

Давайте посмотрим пример конфига:

<https://www.runatlantis.io/docs/repo-level-atlantis-yaml.html>

- Настройка авто планирования.
- Триггеры авто планирования.
- Установка конкретной версии терраформа.
- Разные воркспейсы.
- Апрув для конкретных воркспейсов.



Remote state



Чтение удаленного состояния проекта

Получает данные об удаленном состоянии проекта Terraform.

Это позволяет использовать outputs параметры одного проекта в качестве входных данных для других проектов.

Пример remote_state

```
data "terraform_remote_state" "vpc" {
  backend = "remote"

  config = {
    organization = "hashicorp"
    workspaces = {
      name = "vpc-prod"
    }
  }
}

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



Модули

Рекомендации

- Старайтесь держать модули простыми.
- Иногда есть смысл использовать отдельный проект с `remote_state`, а не отдельный модуль.
- Используйте сторонние модули предварительно изучив их.

Каталог модулей

- Основные провайдеры имеют наборы готовых модулей.
- Основной каталог модулей:

<https://registry.terraform.io/browse/modules>.

Итого

Что мы разобрали:

- Средства для организации командной работы.
- Как организовывать свои модули и связи между проектами.
- Как пользоваться существующими модулями.



Домашнее задание



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Андрей Борю



[andreyborue](https://t.me/andreyborue)



[andreyborue](https://netology.ru/andreyborue)



[andreyborue](https://t.me/andreyborue)