

CI/CD

Адилет Асанкожоев
Python-разработчик в makers.kg



Адилет Асанкожоев

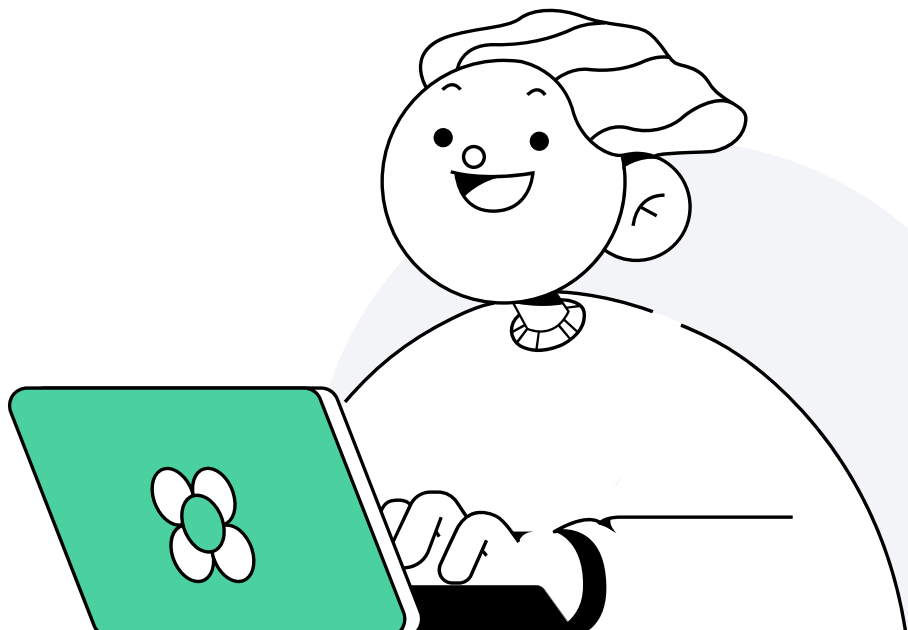
Python-разработчик в makers.kg



План занятия

- 1 Процессы
- 2 Сервисы
- 3 Настройка сервера
- 4 GitHub Actions
- 5 Переменные окружения
- 6 Развёртывание на сервере
- 7 Итоги
- 8 Домашнее задание

*Нажми на нужный раздел для перехода



Процессы



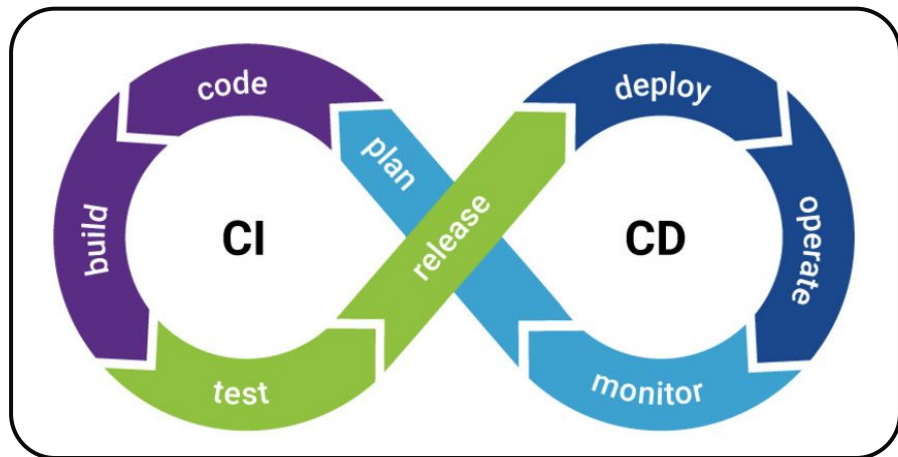
1

Определение

CI/CD (*continuous Integration/continuous delivery*) — комплекс мероприятий для непрерывной интеграции и доставки программного обеспечения.

Одна из задач DevOps — настроить архитектуру для всех CI/CD-процессов.

Разработчику полезно владеть базовыми DevOps-навыками, чтобы запускать свои проекты для демонстрации



Источник

Workflow

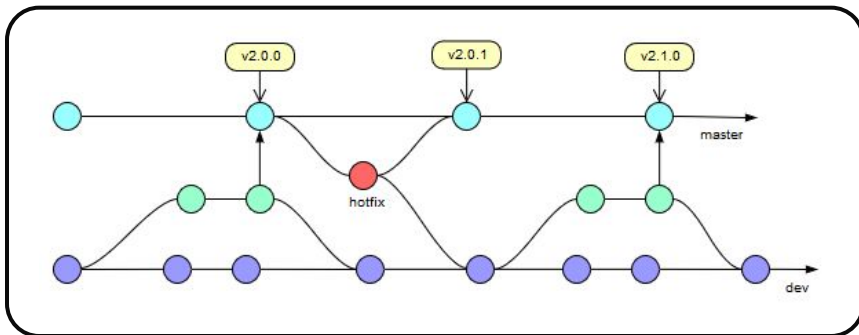
Есть различные схемы взаимодействия процессов разработки и поставки:
Centralized, Forking, Gitflow, GitHub, GitLab и др. workflow

Схему выбирают в зависимости от целей команды.

Например, по триггеру приложение разворачивается в разных окружениях:

- production — финальная версия из основной ветки
- stage — промежуточная версия для QA и бизнес-тестов
- develop — сырая версия из ветки с разработкой

Gitflow:



Сервисы



2

Сервисы интеграции и развёртки

Есть различные инструменты интеграции. Все они основаны на общем принципе конфигурации. По факту целевого события скрипт интеграции запускает команду для запуска тестов или сборки.

Инструменты интеграции, встроенные в git-хостинги:

- [GitHub](#) Actions
- [GitLab](#) Pipelines
- [Bitbucket](#) Pipelines

Специализированные инструменты интеграции, подключаемые к хостингу:

- [CircleCI](#)
- [Semaphore](#) CI
- [Jenkins](#)

Типичный процесс

На примере Django-проекта после коммита изменений на внешнем репозитории (GitHub) можно автоматически запускать тесты и при успешных проверках разворачивать изменения на сервере.

- 1 Создать репозиторий в GitHub
- 2 Отправить в репозиторий исходный код (с валидными тестами и линтерами)
- 3 Настроить CI на GitHub для запуска проверок, чтобы убедиться, что тесты проходят и линтер не ругается
- 4 Создать Docker-образ для публикации приложения
- 5 Развернуть Docker-образ на сервере

Альтернатива

Для некоторых связок есть упрощённый вариант. У GitHub+VPS+Django удобный механизм интеграции:

- 1 Создать репозиторий в GitHub
- 2 Отправить в репозиторий исходный код (с валидными тестами и линтерами)
- 3 Настроить CI на GitHub для запуска проверок, чтобы убедиться, что тесты проходят и линтер не ругается
- 4 Развернуть проект на сервере

Настройка сервера



3

Этапы настройки сервера

- 1 Заказать сервер на Ubuntu и подключиться к нему по ssh
- 2 Создать нового пользователя и назначить его суперпользователем (опционально)
- 3 Установить необходимые пакеты:
`venv, pip, postgresql, nginx`
- 4 Создать базу данных
- 5 Скачать проект из github `git clone ...`
- 6 Развернуть виртуальное окружение `python3 -m venv название_окружения`
- 7 Установить пакеты `pip install -r requirements.txt`
- 8 Настроить переменные окружения в файле `.env`
- 9 Проверить работоспособность проекта
`python manage.py runserver`
- 10 Настроить файл сервиса Gunicorn
`/etc/systemd/system/gunicorn.service`
- 11 Запустить сервис gunicorn
`sudo systemctl start gunicorn`
`sudo systemctl enable gunicorn`
- 12 Настроить конфигурационный файл nginx
`/etc/nginx/sites-available/название_проекта`
- 13 Запустить сайт `sudo ln -s /etc/nginx/sites-available/название_проекта /etc/nginx/sites-enabled`

GitHub Actions



4

GitHub Actions

GitHub Actions сканирует каталог `.github/workflows` на наличие скриптов в **yml** формате, соответствующий [шаблону](#).

На вкладке **actions** проекта GitHub доступны стандартные шаблоны для типовых проектов, например, Python application.

jobs — перечень задач, сборка (build), тестирование (tests)

branches — перечень веток, по действию в которых следует запускать интеграцию

- Например, версия разработки будет собираться на тестовый сервер только из ветки develop, а финальная версия всегда соответствовать состоянию ветки main

steps — последовательность действий для интеграции

- Командой **uses** можно использовать уже заготовленные скрипты
- Командой **name** и **run** можно создавать новые скрипты, задав имя и указав, что нужно выполнить

Подробнее о синтаксисе скрипта [в документации](#)

Переменные окружения



5

Переменные окружения

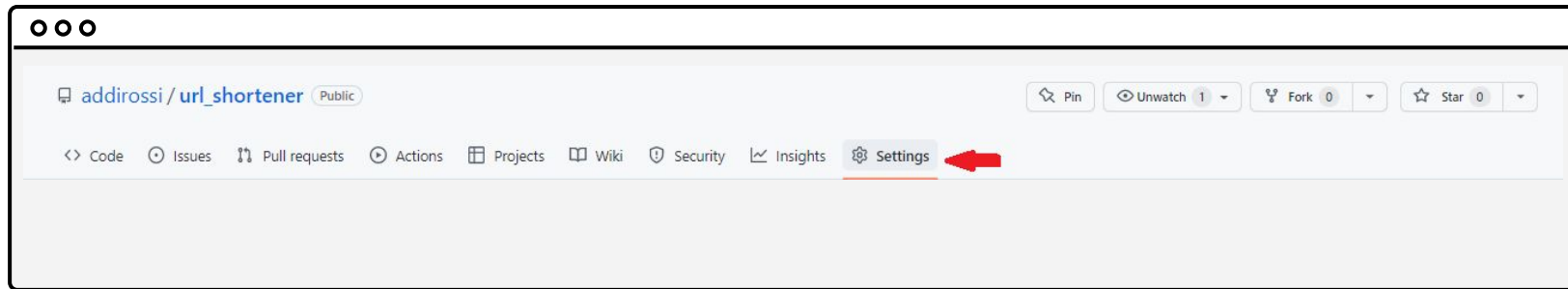
Во время тестов или при работе с определёнными ресурсами внутри процессов Actions могут потребоваться переменные окружения.

Чтобы задать их, можно использовать **GitHub Actions Secrets**



Как задать переменные

Шаг 1. Открыть настройки репозитория:



Как задать переменные

Шаг 2. Выбрать пункт [Secrets -> Actions](#):



Чтобы добавить новую переменную окружения, нажать кнопку [New repository secret](#):



Как задать переменные

Шаг 3. В открывшейся форме указать название переменной и её значение:



The image shows a web browser window with a form for adding a secret. The form has two main sections: 'Name *' and 'Secret *'. The 'Name *' section contains a text input field with the value 'DB_HOST'. The 'Secret *' section contains a larger text area with the value 'localhost'. At the bottom left of the form is a green button labeled 'Add secret'.

ooo

Name *

DB_HOST

Secret *

localhost

Add secret

Пример .yml-файла

ooo

```
name: Django Testing and Deploy
```

```
#Указываем при каких условиях срабатывает workflow
```

```
on:
```

```
  push:
```

```
    branches: [master]
```

```
jobs:
```

```
  tests:
```

```
    runs-on: ubuntu-20.04
```

```
    env:
```

```
      POSTGRES_USER: ${ secrets.POSTGRES_USER }
```

```
      POSTGRES_PASS: ${ secrets.POSTGRES_PASS }
```

```
      POSTGRES_HOST: ${ secrets.POSTGRES_HOST }
```

```
      POSTGRES_PORT: ${ secrets.POSTGRES_PORT }
```

```
      POSTGRES_DB: ${ secrets.POSTGRES_DB }
```

```
      SECRET_KEY: ${ secrets.SECRET_KEY }
```

```
      ALLOWED_HOSTS: ${ secrets.ALLOWED_HOSTS }
```

```
  services:
```

```
    postgres_main:
```

```
      image: postgres:12
```

```
      env:
```

```
        POSTGRES_USER: ${ env.POSTGRES_USER }
```

Развёртывание на сервере



6

Развёртывание на сервере

Теперь нужно сделать так, чтобы изменения автоматически переносились из репозитория на сервер. Для этого воспользуемся уже написанным действием, которое будет подключаться к нашему серверу и выполнять заданные нами команды.

Возьмём действие [appleboy/ssh-action](#), укажем требуемые параметры:

- **host** — IP-адрес или доменное имя сервера
- **port** — порт, по которому подключается ssh (по умолчанию 22)
- **username** — имя пользователя на сервере
- **password** — пароль пользователя
- **script** — команды, которые нужно выполнить после подключения

Action для ssh-подключения к серверу

The screenshot shows the GitHub Actions marketplace page for the 'SSH Remote Commands' action. The page is titled 'SSH Remote Commands' and is version v0.1.5. It features a 'Use latest version' button. The action is described as 'SSH for GitHub Actions' and 'GitHub Action for executing remote ssh commands.' A preview image shows the action's interface with fields for 'uses', 'args', and 'secrets'. The 'args' field is set to ['--script', 'whoami']. The 'secrets' field is set to HOST, USERNAME, and PASSWORD. The page also displays the number of stars (2.7k) and contributors. A warning message at the bottom states: 'Important: Only support Linux docker container.' The 'Input variables' section is also visible.

Marketplace / Actions / SSH Remote Commands

GitHub Action

SSH Remote Commands

v0.1.5 Latest version

Use latest version

SSH for GitHub Actions

GitHub Action for executing remote ssh commands.

Executing remote ssh ...

SSH Commands

uses appleboy/ssh-action@mas...

args ["--script", "whoami"]

secrets HOST USERNAME PASSWORD

Edit

remote ssh command failing

Important: Only support Linux docker container.

Input variables

See action.yml for more detailed information.

Stars

Star 2.7k

Contributors

Categories

Continuous integration Deployment

Links

appleboy/ssh-action

Open issues 54

Pull requests 1

Report abuse

SSH Remote Commands is not certified by GitHub. It is provided by a third-party and is governed by separate terms of service, privacy policy, and support documentation.

Активация Windows

Чтобы активировать Windows, перейдите в раздел "Параметры".

Пример использования

ooo

```
- name: Деплой
  uses: appleboy/ssh-action@master
  with:
    host: ${ secrets.HOST }
    username: ${ secrets.USERNAME }
    password: ${ secrets.PASSWORD }
    script: expect /home/adilet/url_shortener/pull.exp
```


Пример проекта

- [Docker](#)
- [Без Docker](#)



Итоги

- 1 Познакомились с автоматическим запуском тестов при публикации
- 2 Настроили публикацию актуальной версии приложения из основной ветки

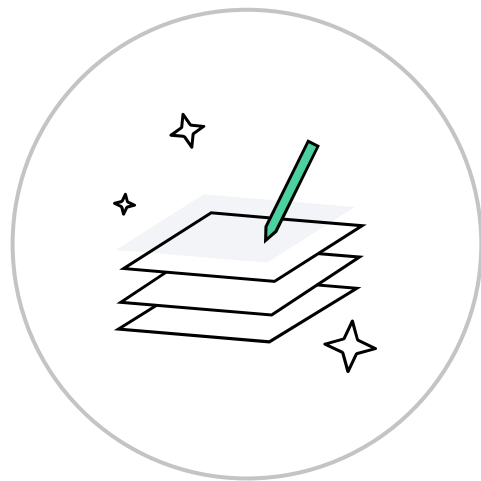


Домашнее задание

Обязательного домашнего задания по лекции нет

Есть необязательное [домашнее задание](#)

Вопросы по теме задавайте в чате группы



Задавайте вопросы и пишите отзыв о лекции

Адилет Асанкожоев
Python-разработчик в makers.kg

