

GitLabизируемся

Nikita Grishko

Чего НЕ будет в этом докладе

- GitLab vs GitHub vs Bitbucket vs ...
- GitLab CI vs Jenkins vs Travis vs ...
- <https://about.gitlab.com/comparison/>

Что будет в этом докладе

- Как в жизнь PandaDoc пришел GitLab
- Как он навел у нас порядок
- Как он ускорил цикл разработки

Немного истории

- ~25 человек в R&D
- ~15 сервисов
- GitHub
- Jenkins
- Разрозненный деплой
- Задержки релизов

Пару слов про Jenkins

- Более менее настроен на одном проекте
- Линтеры запускались последовательно
- Тесты на SQLite
- Билды выполнялись последовательно

Компания растёт

- Увеличение R&D до 75 человек
- Уменьшить time-to-market
- Время пилить монолит на кусочки
- Менять тестирование: зелёный мастер, feature-ветки, меньше тестировать руками
- Деплоймент и Docker

Куда двигаться?



- Дешевле чем GitHub
- GitLab CI и его возможности

Python Package

- Пишем код
- Запускаем линтеры: isort, flake8 и mypy
- Тестируем: pytest, pytest-cov
- Пишем документацию: mkdocs
- Собираем: sdist, bdist_wheel
- Заливаем: twine

tox

- Создание окружений
- Установка зависимостей
- Запуск команд
- <http://tox.readthedocs.io>

```
[tox]
```

```
envlist =
```

```
    py36-flake8
```

```
    py36-isort
```

```
    py36-mkdocs
```

```
    py36-mypy
```

```
    py36-tests
```

```
[testenv]
```

```
commands =
```

```
    flake8: flake8 pdschema tests
```

```
    isort: isort --check-only --diff --recursive .
```

```
    mkdocs: mkdocs build
```

```
    mypy: mypy --ignore-missing-imports pdschema
```

```
    tests: py.test --cov-report term --cov-report html --cov=pdschema -vv {posargs:tests/}
```

```
deps =
```

```
    flake8: flake8
```

```
    isort: isort
```

```
    mkdocs: mkdocs-material
```

```
    mypy: mypy
```

```
    tests: pytest
```

```
    tests: pytest-cov
```

GitLab CI

- Положить в репозиторий `.gitlab-ci.yml` и начать его редактировать

GitLab CI

```
image: python:3.6.4
```

```
stages:
```

- lint
- test
- build
- publish
- pages

GitLab CI

```
lint-isort-py36:
  stage: lint
  script: tox -e py36-isort
  only:
    - branches
    - tags

lint-flake8-py36:
  stage: lint
  script: tox -e py36-flake8
  only:
    - branches
    - tags

lint-mypy-py36:
  stage: lint
  script: tox -e py36-mypy
  only:
    - branches
    - tags
```

GitLab CI

```
test-py36:
  stage: test
  script: tox -e py36-tests
  coverage: '/\d+\%\s*$/ '
  artifacts:
    paths:
      - htmlcov/
  only:
    - branches
    - tags

test-py37:
  stage: test
  image: python:3.7.0b2-stretch
  script: tox -e py36-tests
  only:
    - branches
    - tags
```

GitLab CI

test-py36[Retry](#)

Duration: 19 seconds
Runner: #73
Coverage: 98%
Tags: [docker](#)

Job artifacts

The artifacts will be removed in 30 days


[Keep](#)[Download](#)[Browse](#)

GitLab CI

Artifacts / htmlcov


Name




 coverage_html.js

 index.html

 jquery.ba-throttle-debounce.min.js

 jquery.hotkeys.js

 jquery.isonscreen.js

 jquery.min.js

 jquery.tablesorter.min.js

GitLab CI

```
build-package:
  stage: build
  script:
    - pip install wheel
    - python setup.py sdist bdist_wheel
  artifacts:
    paths:
      - dist
  only:
    - branches
    - tags
```

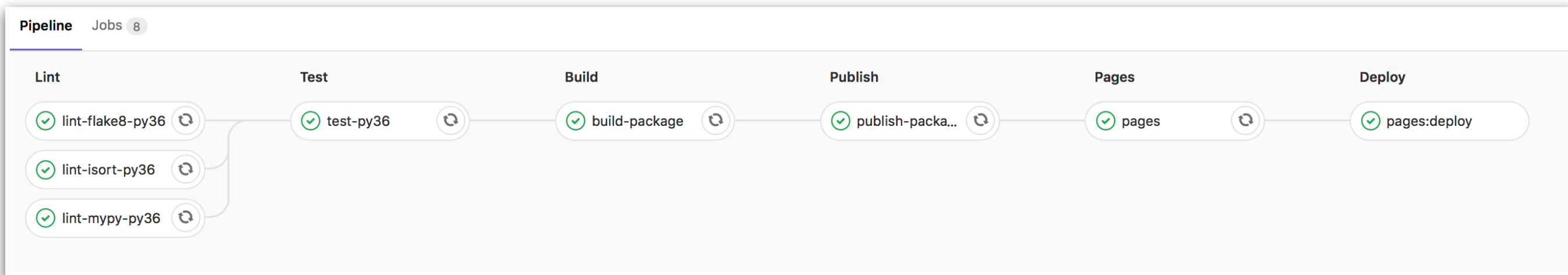
GitLab CI

```
publish-package:
  stage: publish
  script:
    - twine upload
      --repository-url=$PYPI_REPO
      --username=$PYPI_USERNAME
      --password=$PYPI_PASSWORD
      dist/*
  dependencies:
    - build-package
  only:
    - master
    - tags
```

GitLab CI

```
pages:
  stage: pages
  script:
    - tox -e py36-mkdocs
    - mv site public
  artifacts:
    paths:
      - public
  only:
    - master
```

GitLab CI Pipeline



Badges

pdschema pipeline passed coverage 98.00%

Schedules

Description

Run tests and build new development package

Interval Pattern

☒ Custom ([Cron syntax](#)) ☐ Every day (at 4:00am) ☐ Every week (Sundays at 4:00am) ☐ Every month (on the 1st at 4:00am)

0 */4 * * *

Cron Timezone

UTC

Target Branch

master

Variables

Input variable key

Input variable value

Activated

☒ Active

Integrations

- Slack
- HipChat
- Jira
- ...

Django Project

- Сборка Docker образа
- Запуск тестов

GitLab CI

```
build-docker-image:
  stage: build
  before_script:
    - docker login --username=$USERNAME --password=$PASSWORD $DOCKER_REGISTRY
  script:
    - docker build
      --build-arg USERNAME=$USERNAME
      --build-arg PASSWORD=$PASSWORD
      --build-arg BUILD_COMMIT_SHA=$CI_COMMIT_SHA
      --build-arg BUILD_RELEASE=$CI_COMMIT_REF_NAME
      --build-arg BUILD_PIPELINE_ID=$CI_PIPELINE_ID
      -t $CI_PROJECT_NAME:$CI_COMMIT_REF_NAME
      -f docker/Dockerfile .
    - docker tag $CI_PROJECT_NAME:$CI_COMMIT_REF_NAME $DOCKER_REGISTRY/$CI_PROJECT_PATH:$CI_COMMIT_REF_NAME
    - docker push $DOCKER_REGISTRY/$CI_PROJECT_PATH:$CI_COMMIT_REF_NAME
  after_script:
    - docker rmi $CI_PROJECT_NAME:$CI_COMMIT_REF_NAME
  only:
    - branches
    - tags
```

GitLab CI

```
test-integration:
  image: $ARTIFACTORY_DOCKER_REGISTRY/$CI_PROJECT_PATH:$CI_COMMIT_REF_NAME
  stage: test
  services:
    - postgres:9.6
  variables:
    POSTGRES_USER: test
    POSTGRES_PASSWORD: test
    POSTGRES_DB: test
  script:
    - env $(cat .env-base | xargs) DB_HOST=postgres tox --sitepackages -e py36-tests
  only:
    - branches
    - tags
```

GitLab CI Pipeline

Pipeline Jobs 10

Lint

- ✓ check-packag... ↻
- ✓ lint-docker ↻
- ✓ lint-flake8 ↻
- ✓ lint-isort ↻

Build

- ✓ build-docker-i... ↻

Docs

- ✓ build-docs ↻

Test

- ✓ test-django-ch... ↻
- ✓ test-integration ↻
- ✓ test-migrations ↻
- ✓ test-translations ↻

GitLab Templates

- `.gitlab/issue_templates/*.md`
- `.gitlab/merge_request_templates/*.md`

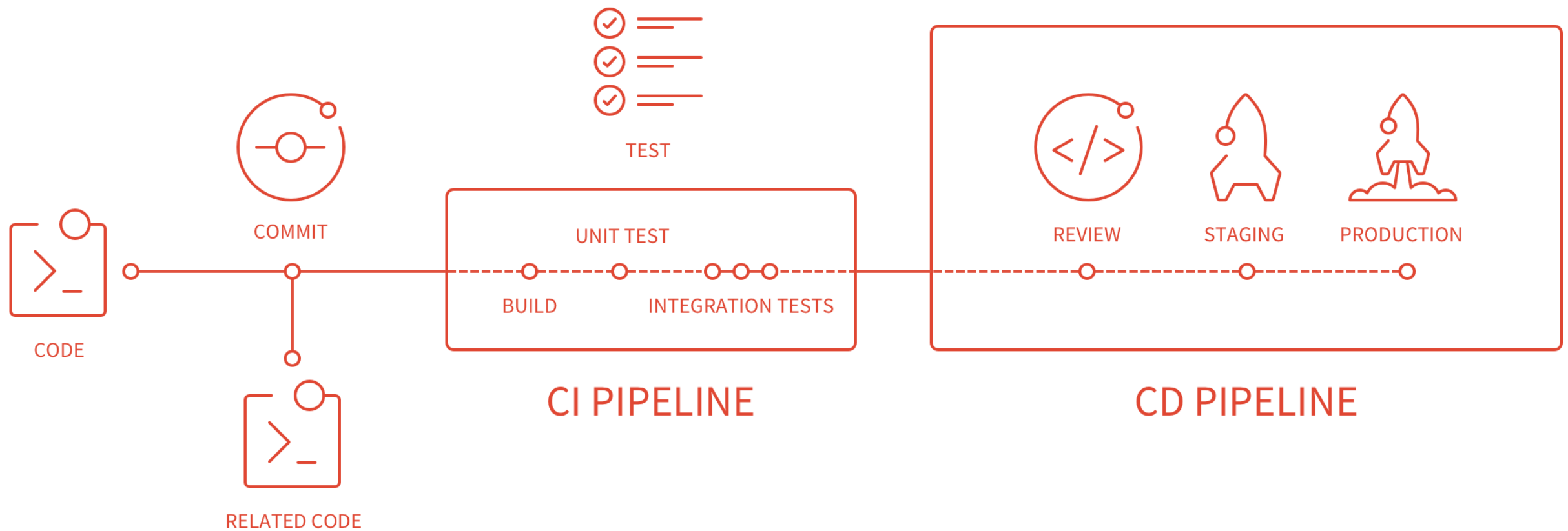
Все ли так хорошо?

```
test-integration:
  image: docker-compose:0.3
  stage: test
  variables:
    # docker-compose related variables
    COMPOSE_PROJECT_NAME: ${CI_PROJECT_NAME}_${CI_COMMIT_REF_NAME}_${CI_JOB_ID}
    COMPOSE_FILE: docker/docker-compose-ci.yml
    # test related variables
    APP_DIR: /var/app
  before_script:
    - docker login --username=$USERNAME --password=$PASSWORD $DOCKER_REGISTRY
    - export $(cat .env-ci | xargs)
    - docker-compose up -d kafka && sleep 10
    - docker-compose exec
      -T kafka /opt/kafka_2.11-0.10.1.0/bin/kafka-topics.sh
      --create --zookeeper 127.0.0.1:2181 --replication-factor 1 --partitions 1 --topic user_events
    - docker-compose up -d microservice
    - docker-compose ps
  script:
    - docker-compose run --rm --volume=$CI_PROJECT_DIR:$APP_DIR --workdir=$APP_DIR
      testrunner tox -e py36-integration-tests
  after_script:
    - export $(cat .env-ci | xargs)
    - docker-compose logs
    - docker-compose down
    - docker-compose ps
```

Что у нас получилось

- С декабря 2017-ого вся разработка ведется на GitLab
- 154 проекта с настроенными пайплайнами
- Более 30.000 выполненных задач
- Код тестируется
- Артефакты собираются

Что дальше?



Thank you!



565 Commercial street, 2nd Floor
San Francisco, CA 94111
pandadoc.com



Yorick Peterse

@yorickpeterse · Member since August 4, 2015

yorickpeterse@gmail.com ·  · yorickpeterse.com ·  The Netherlands ·  GitLab

Database (removal) Specialist at GitLab

