

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Практическая работа №6

Выполнил:  
Борсов Никита

Группа К33402

Проверил:  
Добряков Д. И.

Санкт-Петербург

2023 г.

## Задача

Необходимо настроить автодеплой (с триггером на обновление кода в вашем репозитории, на определённой ветке) для вашего приложения на удалённый сервер с использованием Github Actions или Gitlab CI (любая другая CI-система также может быть использована).

## Ход работы

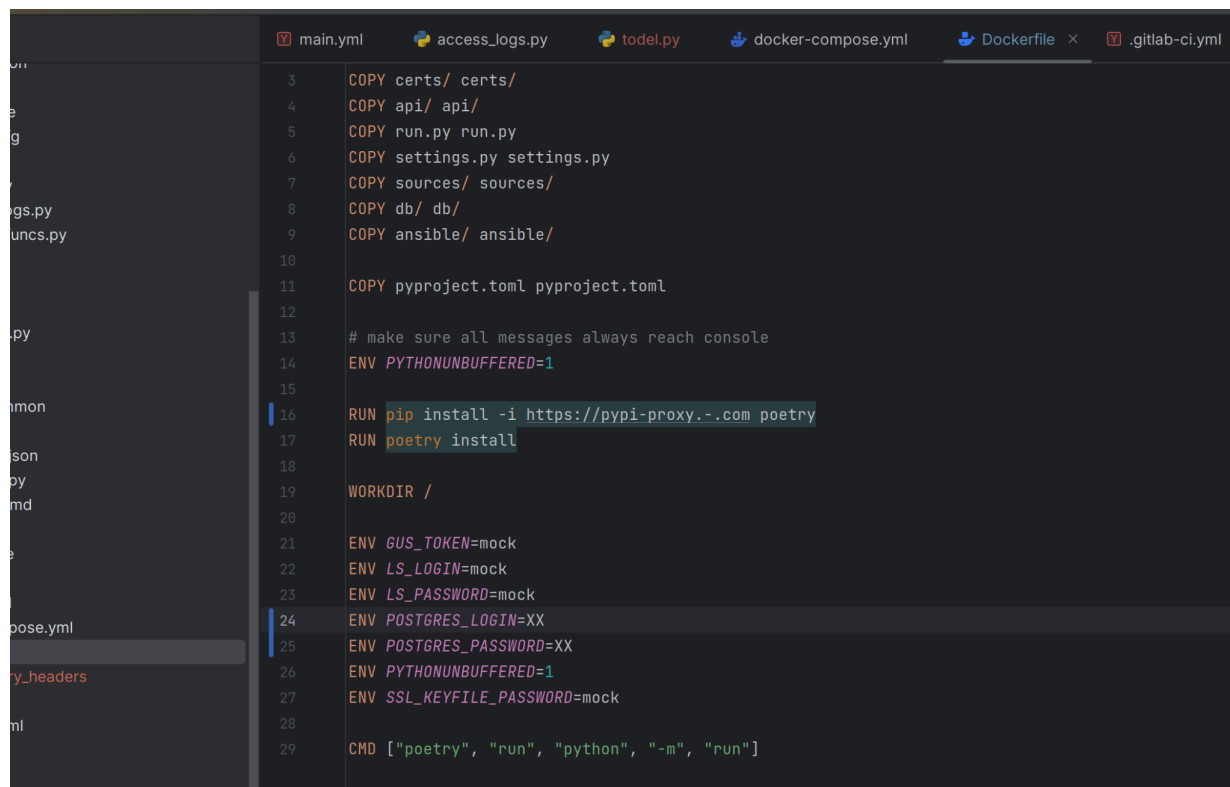
В рамках практики было решено спроектировать и настроить pipeline внутри частного контура с приватным рурі и приватным артефакториумом для Docker образов.

С помощью GitLab CI/CD инструментов были настроены 5 этапов для реализации деплоя на выделенном стенде. Этапы включают в себя:

- version- bump
- Тестирование кода юнит-тестами
- Билд образа и загрузка его на артефакторий
- Деплой образа на один из стендов dev, test, prod в соответствии с текущей веткой
- Сохранение артефактов в артефакторий

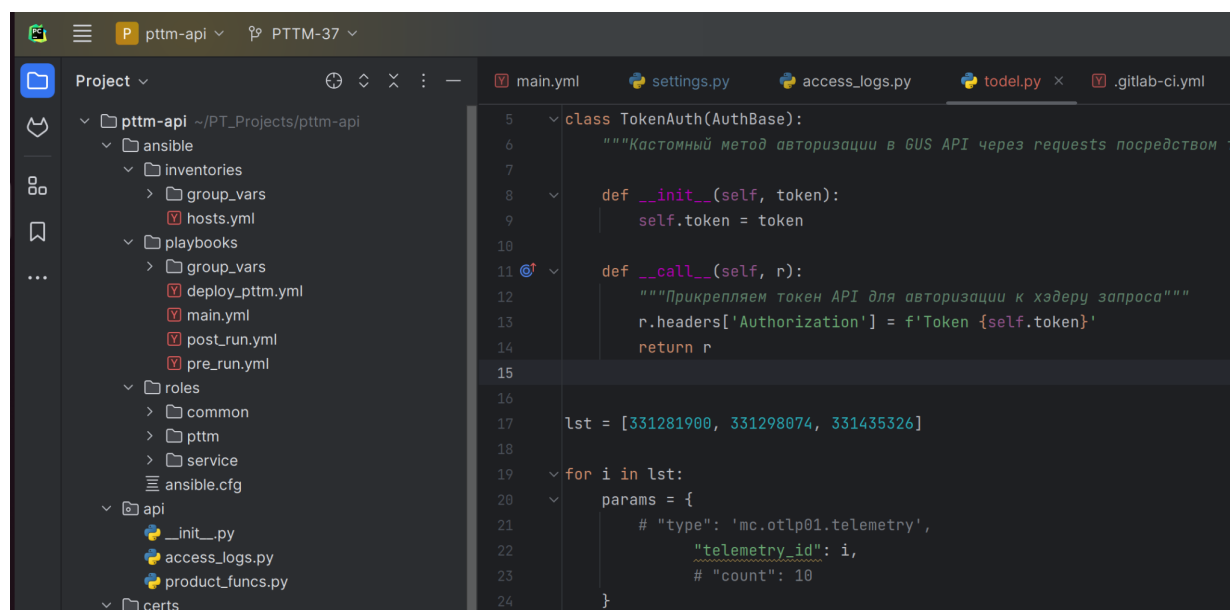
The screenshot displays a GitLab CI/CD pipeline interface. At the top, a green status bar indicates the pipeline is 'passed'. Below this, the pipeline is identified as 'Pipeline #2679252' triggered 2 weeks ago by 'Nikita Borsov'. The main heading is 'PTTM-37: Добавление нового эндпоинта /client\_access/by\_product\_and\_version/'. A summary box shows '5 jobs for PTTM-37' completed in '3 minutes and 7 seconds' using '0.0 compute credits'. The 'latest' commit is '77399d14'. A related merge request is listed: '1 related merge request: 12 PTTM-37: Реализация первых пяти ручек ptm-api'. The pipeline progress bar shows 'Pipeline' as the active stage, with 'Needs', 'Jobs' (5), and 'Tests' (0) as sub-steps. The 'Group jobs by' dropdown is set to 'Stage'. The pipeline consists of five stages: 'prepare' (job: version - bump), 'test' (job: test - python), 'publish' (job: build - publish - image), 'deploy' (job: deploy - service), and 'external' (job: repo: webapp/#!/artifacts/browse/tree/General...). Each job is marked with a green checkmark and a refresh icon.

Ниже можно увидеть пример DockerFile который содержит в себе инструкции на yaml, развертывание python приложения производится с помощью poetry, вместе стандартного pip:



```
3 COPY certs/ certs/
4 COPY api/ api/
5 COPY run.py run.py
6 COPY settings.py settings.py
7 COPY sources/ sources/
8 COPY db/ db/
9 COPY ansible/ ansible/
10
11 COPY pyproject.toml pyproject.toml
12
13 # make sure all messages always reach console
14 ENV PYTHONUNBUFFERED=1
15
16 RUN pip install -i https://pypi-proxy.-.com poetry
17 RUN poetry install
18
19 WORKDIR /
20
21 ENV GUS_TOKEN=mock
22 ENV LS_LOGIN=mock
23 ENV LS_PASSWORD=mock
24 ENV POSTGRES_LOGIN=XX
25 ENV POSTGRES_PASSWORD=XX
26 ENV PYTHONUNBUFFERED=1
27 ENV SSL_KEYFILE_PASSWORD=mock
28
29 CMD ["poetry", "run", "python", "-m", "run"]
```

В самом процессе деплоя ключевую роль играет ansible, благодаря которому происходит менеджмент процессами стенда через systemctl:



```
5 class TokenAuth(AuthBase):
6     """Кастомный метод авторизации в GUS API через requests посредством то
7
8     def __init__(self, token):
9         self.token = token
10
11     def __call__(self, r):
12         """Прикрепляем токен API для авторизации к хэдеру запроса"""
13         r.headers['Authorization'] = f'Token {self.token}'
14         return r
15
16
17 lst = [331281900, 331298074, 331435326]
18
19 for i in lst:
20     params = {
21         # "type": 'mc.otlp01.telemetry',
22         "telemetry_id": i,
23         # "count": 10
24     }
```

## **Вывод**

В результате практической работы была реализована комплексная система end-to-end пайплайна для деплоя в закрытом контуре используя продвинутое инструменты в том числе - локальный артефакторий и ruir.