Распределенный запуск E2E-тестов в CI

Roman Zolotarev

YCLIENTS

HEISENBUG

Роман Золотарев

- В QA с 2016 года
- SDET Teamlead @ YCLIENTS

@rzltrv

SaaS b2b-решение для онлайн-записи и автоматизации бизнеса в сфере услуг

- ERP-платформа
- REST API
- Мобильное приложение
- Виджеты онлайн-записи

- 35 000 активных b2b-клиентов
- 12 000 RPS трафика
- Монолит на PHP
- ~ 350 сотрудников

План

- Почему мы разработали собственный инструмент?
- Что должен уметь инструмент?
- Реализация MVP
- Эволюция MVP

Почему мы разработали собственный инструмент?

Базовые ожидания от автотестов

- Много тестов
- Быстрые
- Надежные

Тесты как сервис:

- разработка новых тестов
- интеграцию тестов в pipeline
- поддержка актуальности
- разбор сложных падений

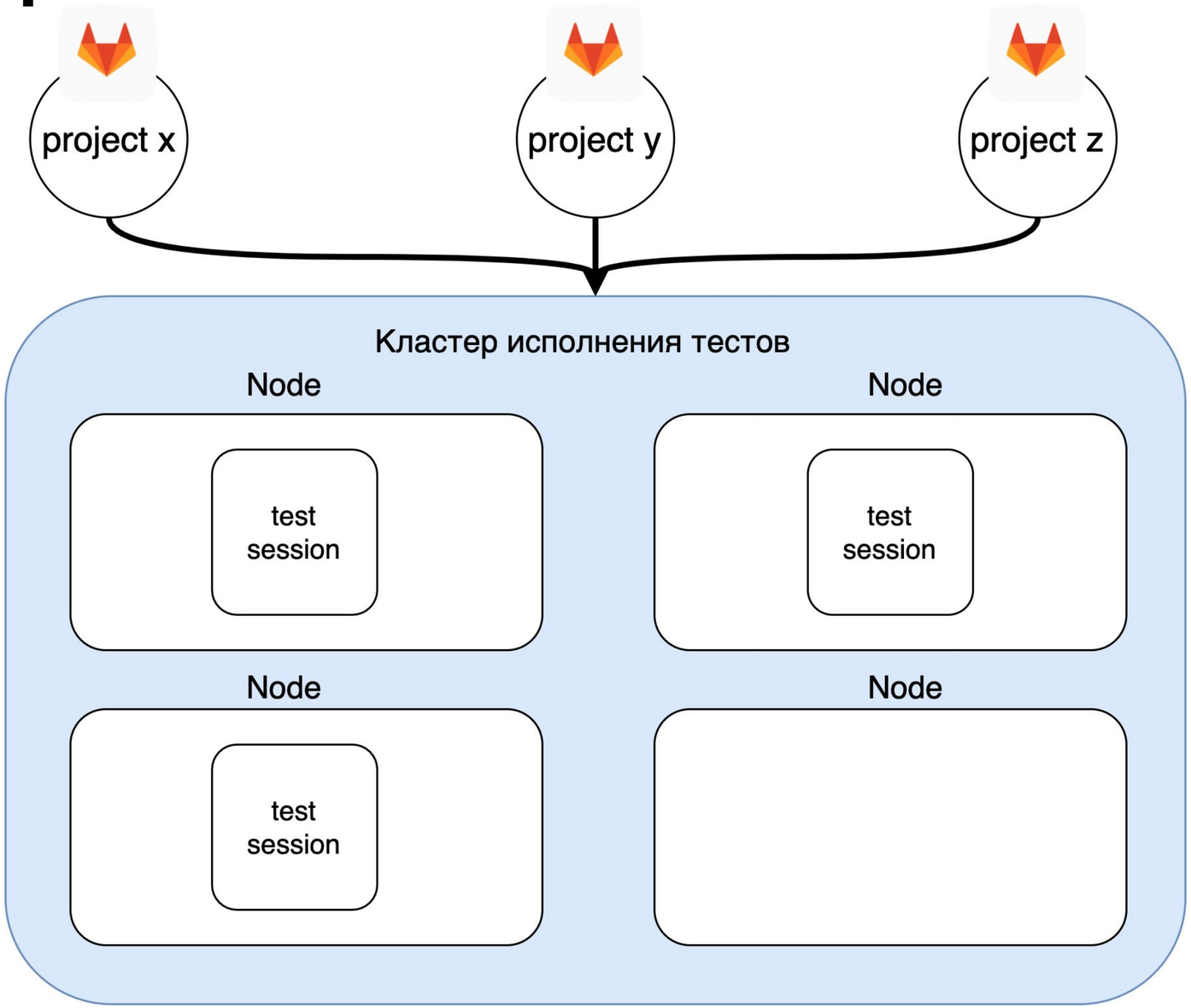
Обзор проекта с тестами

- Python
- pytest
- Selenium
- Gitlab CI
- Allure TestOps

Обзор проекта с тестами

- Около 1000 тестов
- Проходят примерно за 20 минут

Обзор проекта с тестами



yclients

Проблемы схемы

- Отсутствует горизонтальное масштабирование

Проблемы схемы

- Отсутствует горизонтальное масштабирование
- Неконтролируемый профиль нагрузки на ноды исполнения тестов

Проблемы схемы

- Отсутствует горизонтальное масштабирование
- Неконтролируемый профиль нагрузки на ноды исполнения тестов
- Нет максимальной утилизации доступных ресурсов

Проблемы схемы

- Отсутствует горизонтальное масштабирование
- Неконтролируемый профиль нагрузки на ноды исполнения тестов
- Нет максимальной утилизации доступных ресурсов
- Ограничения на количество одновременно запущенных тестовых наборов

Боли

- Разработчикам приходилось следить за количеством запущенных пайплайнов

Боли

- Разработчикам приходилось следить за количеством запущенных пайплайнов
- Нестабильность прохождения тестовых наборов

Боли

- Разработчикам приходилось следить за количеством запущенных пайплайнов
- Нестабильность прохождения тестовых наборов
- Тратим время на разбор случайных и непонятных падений тестов

Боли

- Разработчикам приходилось следить за количеством запущенных пайплайнов
- Нестабильность прохождения тестовых наборов
- Тратим время на разбор случайных и непонятных падений тестов
- Теряем в скорости разработки тестов

Боли

- Разработчикам приходилось следить за количеством запущенных пайплайнов
- Нестабильность прохождения тестовых наборов
- Тратим время на разбор случайных и непонятных падений тестов
- Теряем в скорости разработки тестов
- Теряем в Time to market

Что должен уметь инструмент?

Сбор требований

- Сократить длительность прохождения тестовых наборов

Сбор требований

- Сократить длительность прохождения тестовых наборов
- Горизонтального масштабирования тестовой инфраструктуры

Сбор требований

- Сократить длительность прохождения тестовых наборов
- Горизонтального масштабирования тестовой инфраструктуры
- Максимальная утилизация доступных ресурсов

Сбор требований

- Сократить длительность прохождения тестовых наборов
- Горизонтального масштабирования тестовой инфраструктуры
- Максимальная утилизация доступных ресурсов
- Управление приоритетами запускаемых тестов

Сбор требований

- Сократить длительность прохождения тестовых наборов
- Горизонтального масштабирования тестовой инфраструктуры
- Максимальная утилизация доступных ресурсов
- Управление приоритетами запускаемых тестов
- Снять ограничение количества одновременно запущенных тестовых наборов

Сбор требований

- Сократить длительность прохождения тестовых наборов
- Горизонтального масштабирования тестовой инфраструктуры
- Максимальная утилизация доступных ресурсов
- Управление приоритетами запускаемых тестов
- Снять ограничение количества одновременно запущенных тестовых наборов
- Динамическое масштабирование тестовой инфраструктуры*

Реконфигурация Gitlab runners

Плюсы

- Можно сделать быстро

Минусы

- Только частично закрывает требования

pytest-xdist

Плюсы

- Довольно быстрая интеграция в проект, готовый инструмент, используемый в сообществе

Минусы

- Ненадежно. Deprecated фича.
- Только частично закрывает требования

Playwright sharding

Плюсы

- Набирающий популярность инструмент для e2e браузерных тестов

Минусы

- Долго
- Необходимо переписать проект на TS
- Научить команду писать тесты на TS
- Удовлетворяет требованиям только частично

Свой инструмент для запуска тестов

Плюсы

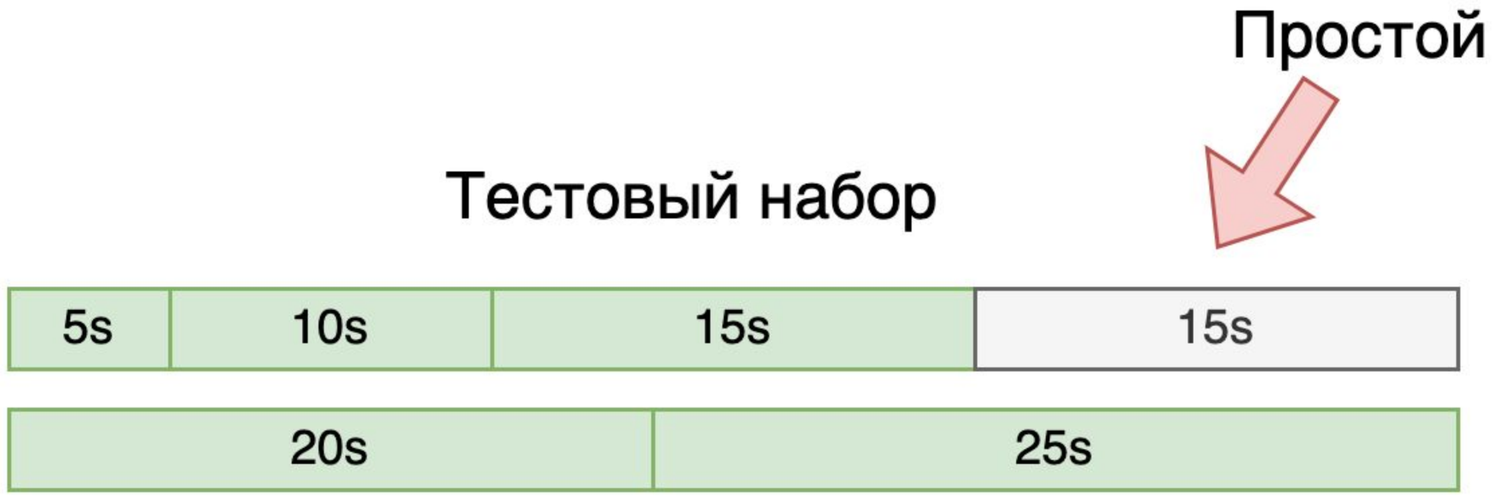
- Полностью удовлетворит требования

Минусы

- Долго?

Плагин для pytest — cli утилита для запуска тестов параллельно

Неоптимальное случайное распределение тестов



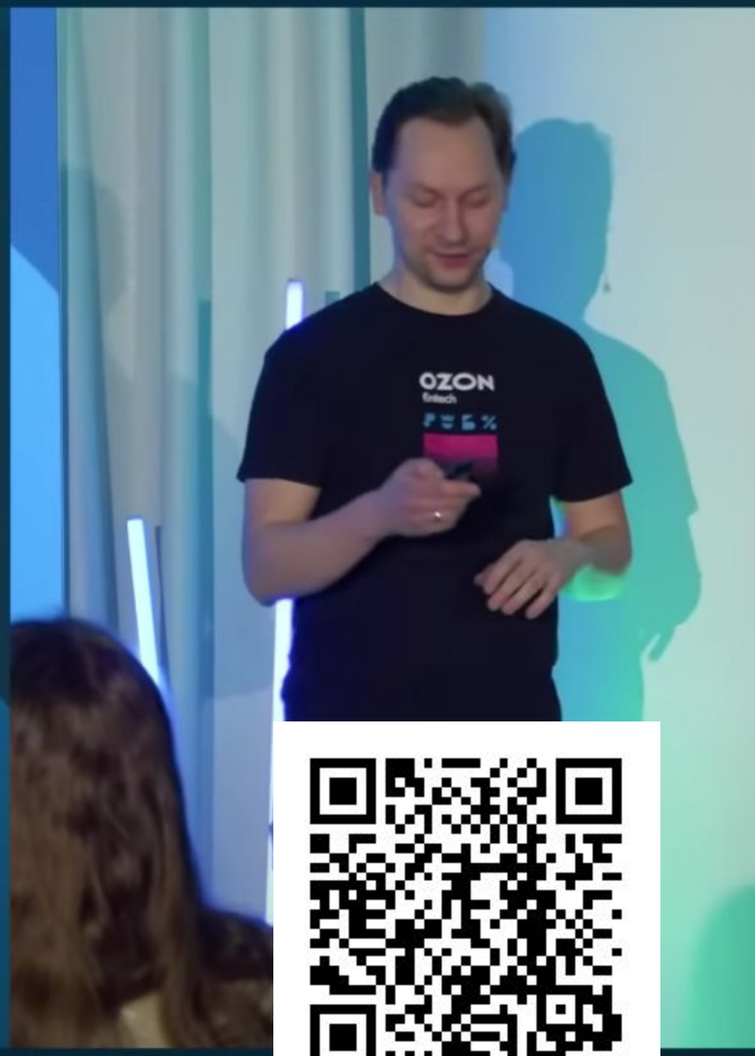
Игорь Балагуров, руководитель группы тестирования
Инфраструктура тестирования для API-тестов на Python

OZON tech

Инфраструктура API-тестов на Python

Игорь Балагуров, руководитель тестирования Ozon Fintech

ibalagurov@ozon.ru, t.me/ibalagurov



@ozon_tech

@ozontech

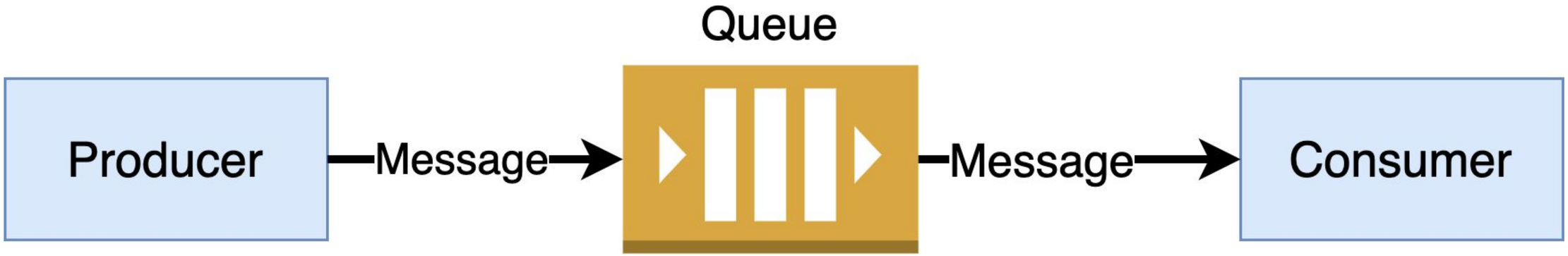
OZON tech

Реализация MVP

Простая идея — внедряем очередь тестов.

Очередь — это временное хранилище сообщений, ожидающих дальнейшей обработки или доставки в пункт назначения

Простая схема работы с очередью



Алгоритм работы

1. Взять тест
2. Положить тест в очередь
3. Забрать из очереди
4. Исполнить

Алгоритм работы

1. Как взять тест?
2. Как положить тест в очередь?
3. Как забрать?
4. Как исполнить?

Как взять тест?

Разделить запуск и исполнение тестов.

Пример запуска тестов: **`pytest -m suite_smoke`**

Как взять тест?

- хук **`pytest_itemcollected`** - исполняется, когда происходит событие “сбор теста”

Как взять тест?

```
def pytest_itemcollected(item):
    """we just collected a test item."""
    message = {
            "node_id": item.nodeid,
            "env": {
                **os.environ.copy(),
            },
        }
    Reactor().collect(message)
```

Как взять тест?

- хук **`pytest_itemcollected`** - исполняется, когда происходит событие “сбор теста”
- хук **`pytest_runtestloop`** - исполнение протокола запуска теста

Как взять тест?

```
def pytest_runtestloop(session):
    if Reactor().items["ready"]:
        Reactor().run()
    return False
def run():
    for item in self.items["ready"]:
        self.channel.default_exchange.publish(
            Message(
                body=json.dumps(item).encode(),
                content_type="application/json",
            ),
            routing_key="queue_name",
        )
```

Как запустить отдельный тест?

nodeid — Уникальный идентификатор теста в pytest.

Запуск теста через nodeid - `pytest test_login.py::test_login`

Сообщение для отправки в очередь

```
{
    "node_id": "test_login.py::test_login",
    "env": {
        "AUTOTESTS_HUB_URL": "https://grid.some.domain",
        "YCL_SANDBOX_FQDN": "https://test-01.sandbox.some.domain"
    }
}
```

Что нужно для запуска теста?

- Окружение - Python и установленный пакет с нашими тестами
- Собрать команду для запуска теста

Шаблон команды:

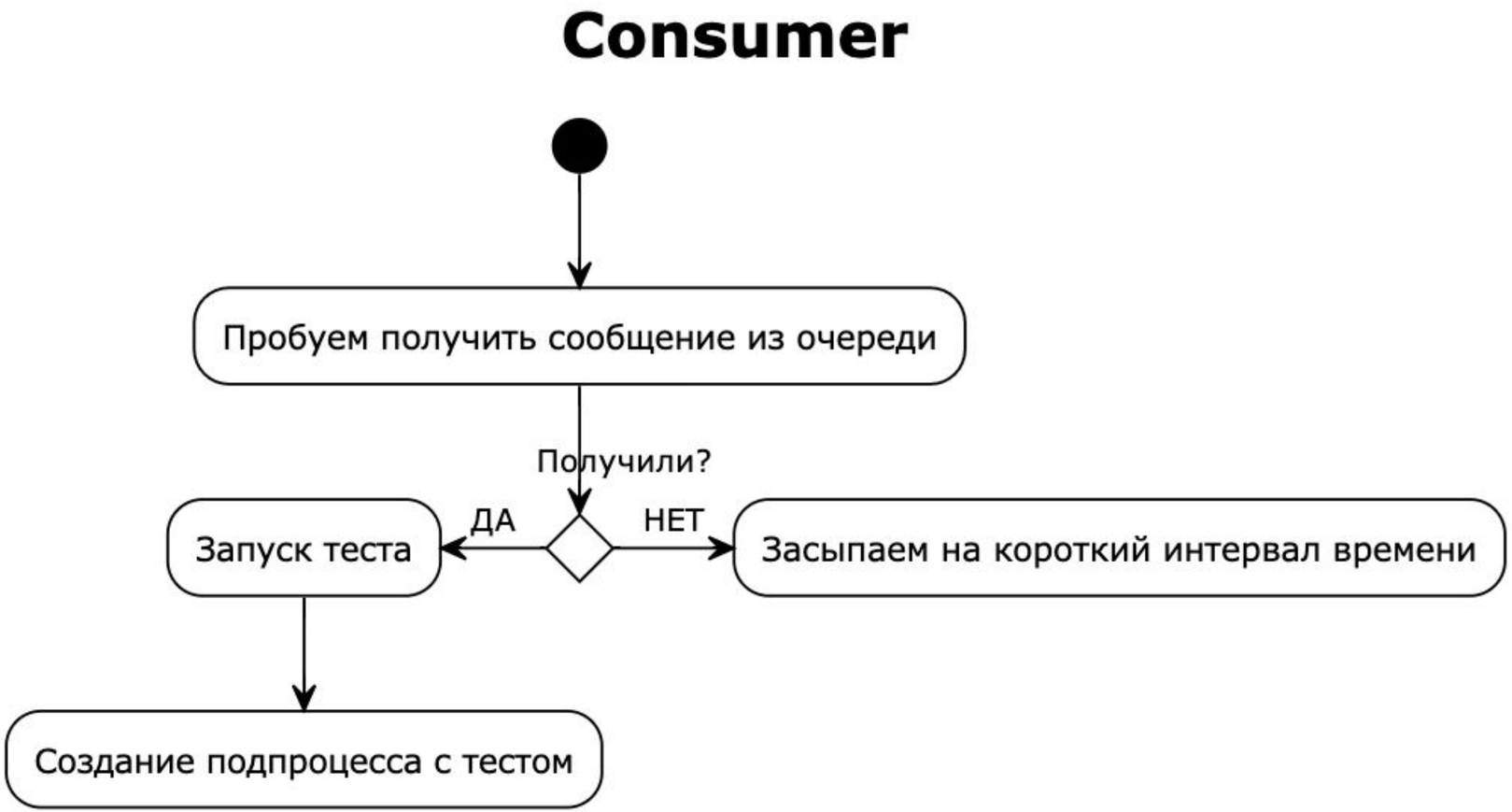
```
{путь к pytest} {путь к пакету с тестами} {nodeid теста}
```

```
pyenv/versions/3.7.9/envs/autotests-1/bin/pytest
pyenv/versions/3.7.9/envs/autotests-1/lib/python3.7/site-packages/autotests/test_login.py:
:test_login
```

Consumer

```
def consume(self):
    while True:
        message = self.queue.get(fail=False, timeout=1)
        if message:
            self.process_message(message)
        else:
            sleep(1)

def process_message(self, message):
    cmd = [f'{pytest}{site_packs}/{message["node_id"]}']

    sp = SubProcess(
        cmd=cmd,
        env=message["env"],
    )
    sp.run()
```

Схема consumer



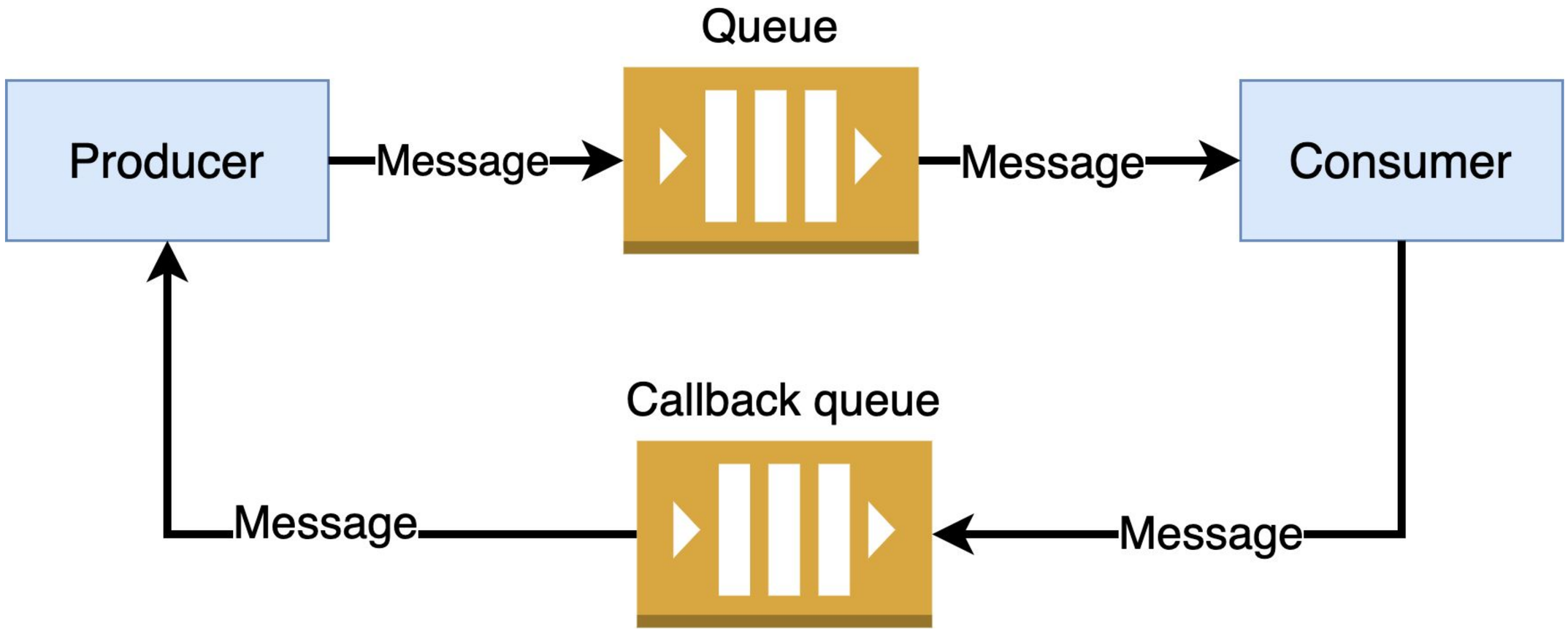
Запустили тест, но есть вопросы

- Был ли исполнен тест?
- Какой результат теста?

RPC - remote procedure call

RPC — это метод коммуникации между разными хостами, который позволяет вызывать функции на удаленном хосте и получать результаты их выполнения.

Реализация rpc через очередь



Сообщение для отправки в очередь

```
{
    "node_id": "test_login.py::test_login",
    "env": {
        "AUTOTESTS_HUB_URL": "https://grid.some.domain",
        "YCL_SANDBOX_FQDN": "https://test-01.sandbox.some.domain"
    },
    "correlation_id": "a398e982-5de7-4cdf-859d-28fd445366f7",
    "reply_to": "callback_queue_name"
}
```

Запуск тестов параллельно

Нужно добавить

- Ограничение количества одновременно запущенных тестов - threads
- Асинхронный запуск процесса с тестом

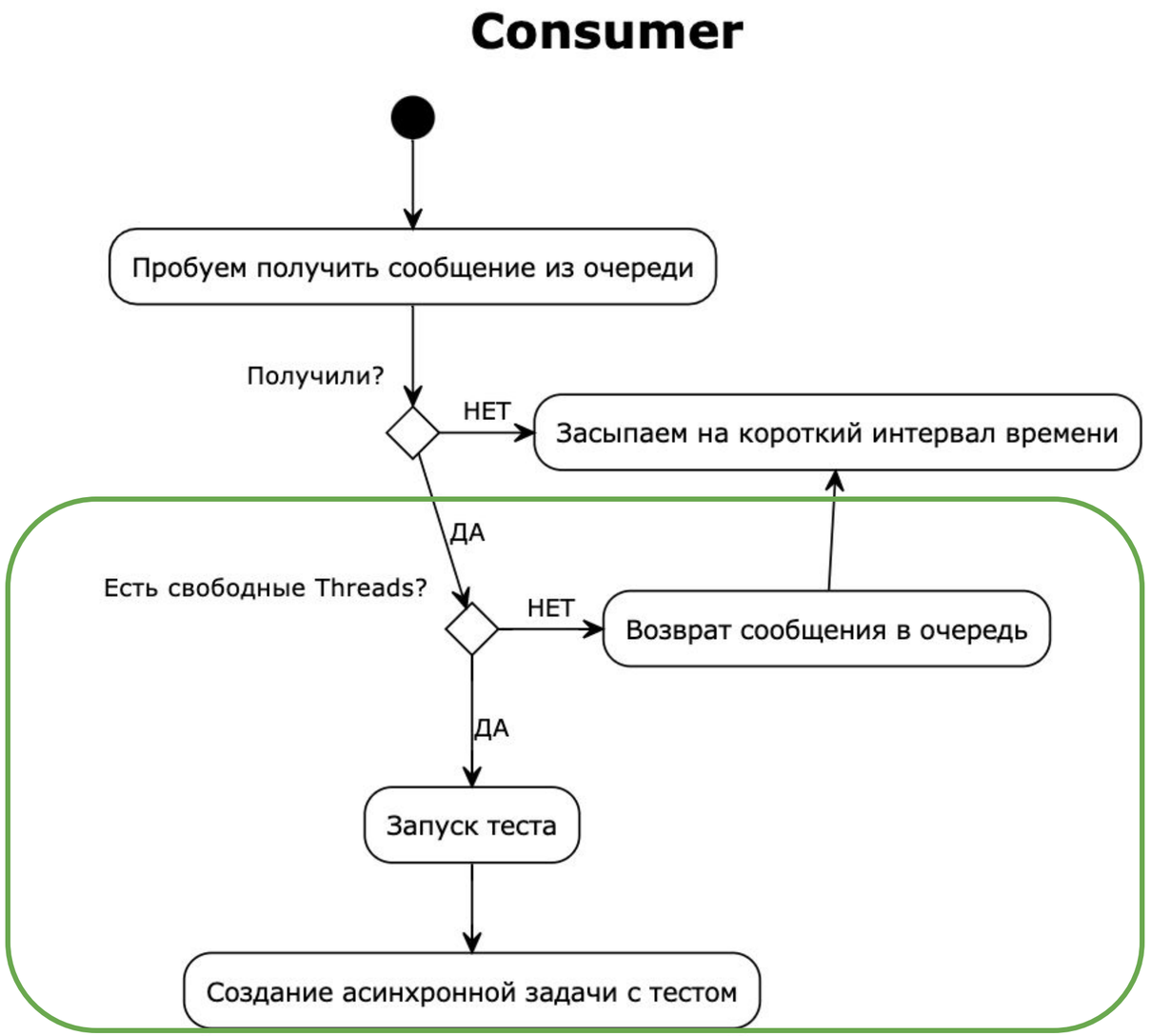
Запуск тестов параллельно

```
def consume(self):
    while True:
        if self.threads["busy"] < self.threads["max"]:
            message = self.queue.get(fail=False, timeout=1)
            if message:
                self.threads["busy"] += 1
                asyncio.create_task(self.run_test(message))

            else:
                sleep(1)
```

Запуск тестов параллельно

```
async def run_test(self, message):
    cmd = [f'{pytest}{site_packs}/{message["node_id"]}']

    sp = SubProcess(
        cmd=cmd,
        env=message["env"],
    )
    task = asyncio.create_task(sp.run())
    await task
    await asyncio.create_task(self.send_reply_msg(task))
    self.threads["busy"] -= 1
```

Запуск тестов параллельно



Запуск тестов параллельно

- Сервис может запустить больше параллельных тестов, чем может обработать тестовое окружение?
- Как сформировать отчет о прохождении тестов?
- Как быть с версионированием тестов?

Версионирование

- Публикация пакета с тестами
- Установка актуальной стабильной версии перед запуском тестов в pipeline

Поддержка версионирования

- Создание виртуальных окружений
- Установка пакета с тестами

Сообщение для отправки в очередь

```
{
    "node_id": "test_login.py::test_login",
    "env": {
        "AUTOTESTS_HUB_URL": "https://grid.some.domain",
        "YCL_SANDBOX_FQDN": "https://test-01.sandbox.some.domain"
    },
    "correlation_id": "a398e982-5de7-4cdf-859d-28fd445366f7",
    "reply_to": "callback_queue_name",
    "package_version": "8.0.412"
}
```

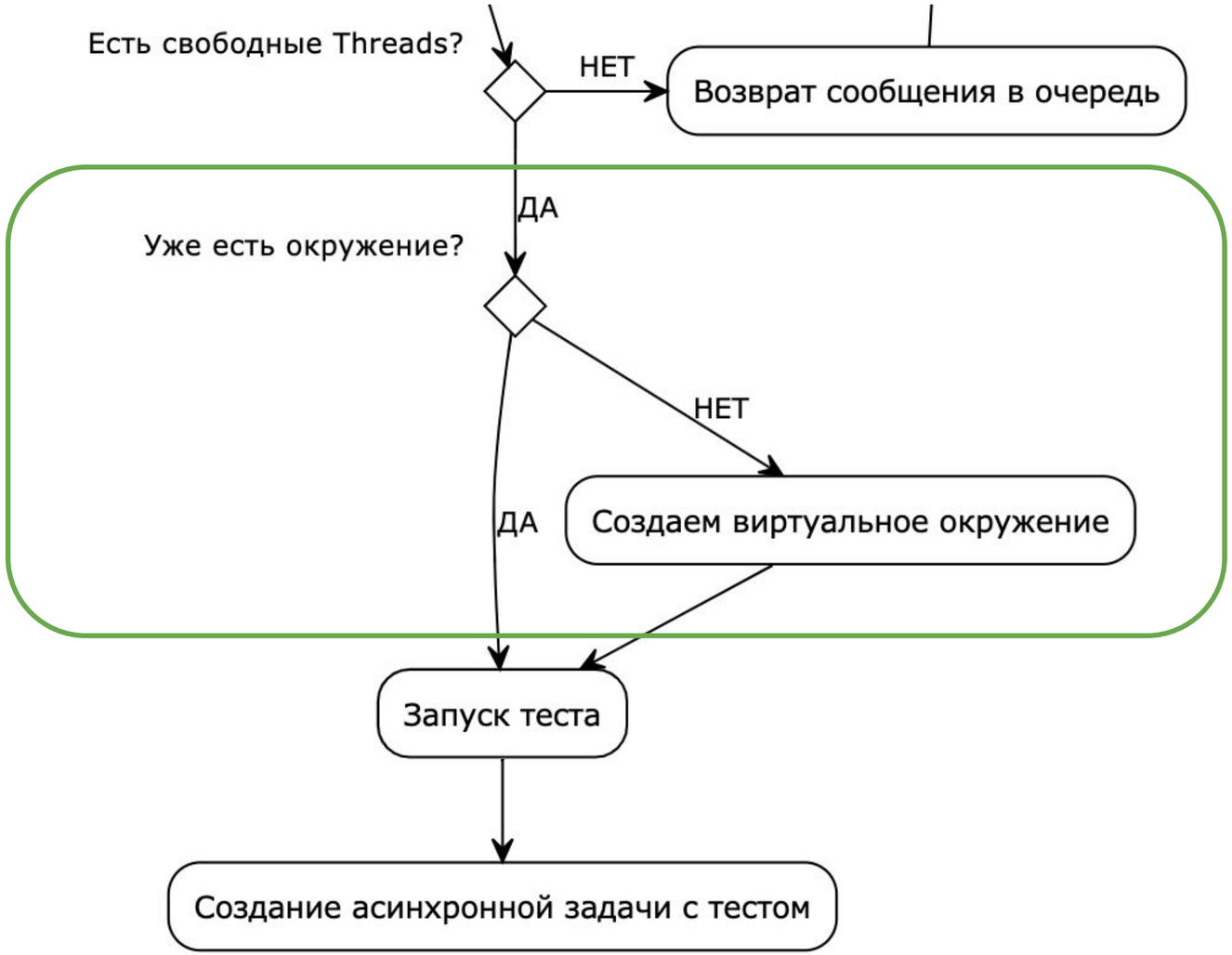
Поддержка версионирования

```
def create_venv(self, package_version):
    venv_dir = os.path.join(self.base_venvs_dir, f"{package_version}")
    if not os.path.isdir(venv_dir):
        cli_run([f"{venv_dir}"])
    venv_ = Venv(venv_dir=venv_dir)
    return venv_

def install_package(self, package_version):
    venv_: Venv = self.get_venv(package_version)
    cmd = [
        f"{venv_.python}",
        "-m",
        'pip',
        'install',
        f'yclients-autotests-blackbox=={package_version}'
    ]
    execute_sync(cmd)
```

Поддержка версионирования

```
def consume(self):
    while True:
        if self.threads["busy"] < self.threads["max"]:
            message = self.queue.get(fail=False, timeout=1)
            if message:
                package_version = message['package_version']
                if not self.venv_manager.venv_exist(package_version):
                    self.venv_manager.create_venv(package_version)
                    self.venv_manager.install_package(package_version)
                self.threads["busy"] += 1
                asyncio.create_task(self.run_test(message))

            else:
                sleep(1)
```

Поддержка версионирования



Формирование отчетов

- Отчеты в Allure TestOps
- **Launch** - Набор результатов тестов в рамках одного или нескольких прогонов
- **Job run** - прогон набора тестов

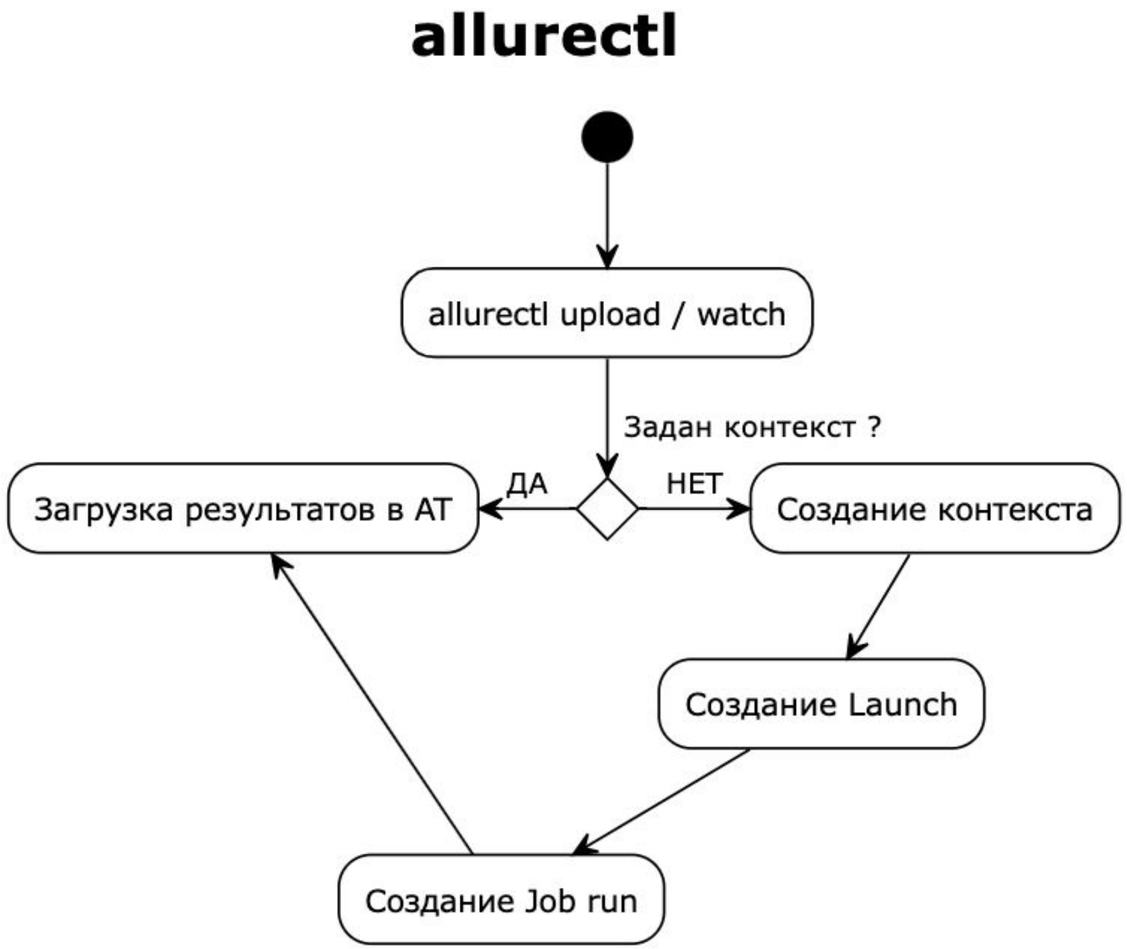
Загрузка отчета в Allure TestOps - allurectl

- **`allurectl upload`** - загрузка отчета после выполнения тестов
- **`allurectl watch`** - загрузка в реальном времени

Подходы к решению

- Агрегировать артефакты и загрузить их через `allurectl upload`
- Загружать результаты отдельных тестов по мере их прохождения

Как работает allurectl



Загрузка отчета в Allure TestOps

- Создать launch и job run

Создание Launch и Job Run

```
def create_launch(self):
    cmd = [
        f"{self.cli.allure}",
        f"launch create --endpoint {URL} --token {API_TOKEN}",
        f"--project-id {ALLURE_PROJECT_ID} --launch-tags {ALLURE_LAUNCH_TAGS
--launch-name {ALLURE_LAUNCH_NAME}",
        "--output json",
    ]
    output = execute_sync(cmd, env=os.environ.copy()).stdout
    result = str(json.loads(output)[0]["id"])
    self.launch_id = result
```

Создание Launch и Job Run

```
def start_job_run(self):
    cmd = [
        f"{self.cli.allure}",
        "job-run",
        "start",
        "--launch-id",
        f"{self.launch_id}",
    ]
    output = str(execute_sync(cmd, env=os.environ.copy()).stdout)
    self.job_run_id = re.findall("(?<=Job run \[)[^]]+(?=\])", output)[
        0
    ]
```

Загрузка отчета в Allure TestOps

- Создать launch и job run
- Добавить в сообщение контекст для загрузки

Сообщение для отправки в очередь

```
{
    "node_id": "test_login.py::test_login",
    "env": {
        "AUTOTESTS_HUB_URL": "https://grid.some.domain",
        "YCL_SANDBOX_FQDN": "https://test-01.sandbox.some.domain",
        "ALLURE_LAUNCH_ID": "9232312",
        "ALLURE_JOB_RUN_ID": "34534234"
    },
    "correlation_id": "a398e982-5de7-4cdf-859d-28fd445366f7",
    "reply_to": "callback_queue_name",
    "package_version": "8.0.412"
}
```

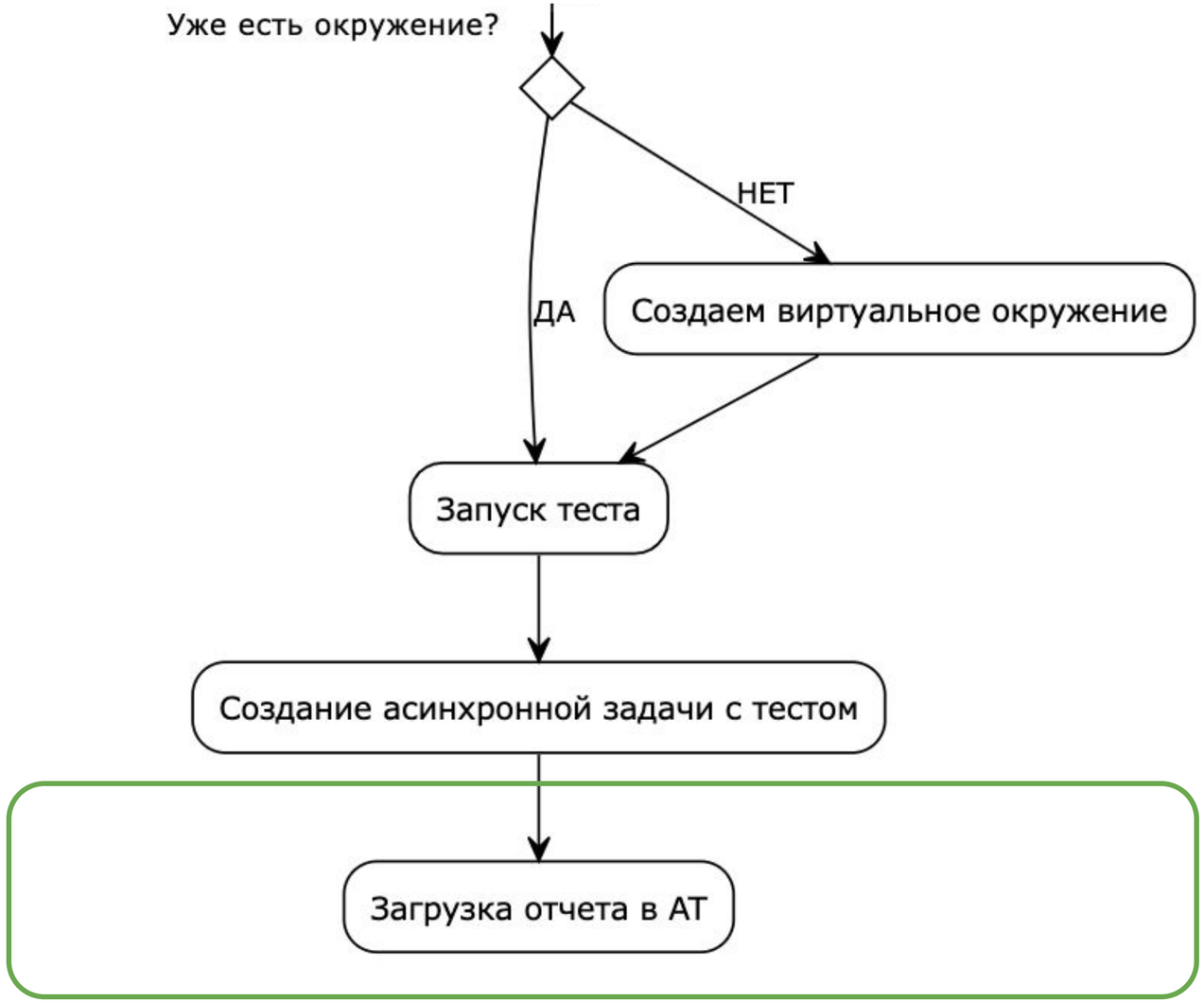
Загрузка отчета в Allure TestOps

- Создать launch и job run
- Добавить в сообщение контекст для загрузки
- Генерацию артефактов в формате allure
- Новый шаг - загрузка отчета

Загрузка отчета в Allure TestOps

```
async def upload_result(self, test_report):
    cmd = [f"{self.allurectl}",
           f"upload --endpoint {self.allure_endpoint} --token
{self.allure_token} --project-id",
           f"{test_report.allure_project_id} --launch-id
{test_report.allure_launch_id} --job-run-id",
           f"{test_report.job_run_id} {test_report.test_results_path}"
           ]
    env = {
           **os.environ,
           **test_report.item_env
        }
    sp = SubProcess(
        cmd=cmd,
        env=env,
    )
    task = asyncio.create_task(sp.run())
```

Загрузка отчета в Allure TestOps



yclients

Ограничения тестовых окружений

Сервис может запустить больше параллельных тестов, чем может обработать тестовое окружение, что делать?

Ввести дополнительное ограничение для запуска тестов - емкость тестового окружения.

Примитивы синхронизации

- **Lock —** механизм, позволяющий обеспечить эксклюзивный доступ к ресурсу одному потоку или процессу в определенный момент времени.

- **Semaphore —** счетчик, который контролирует доступ к ресурсу для ограниченного количества потоков или процессов.

Реализация semaphore

- **Redis —** база данных в оперативной памяти, которая использует модель ключ-значение для хранения и обработки данных

- Библиотека redis-semaphore

Реализация semaphore

- **Название** - имя тестового окружения или что-то уникальное в рамках тестового окружения
- **Размер** - сколько одновременных потоков разрешено

Сообщение для отправки в очередь

```
{
    "node_id": "test_login.py::test_login",
    "env": {
        "AUTOTESTS_HUB_URL": "https://grid.some.domain",
        "YCL_SANDBOX_FQDN": "https://test-01.sandbox.some.domain",
        "ALLURE_LAUNCH_ID": "9232312",
        "ALLURE_JOB_RUN_ID": "34534234"
    },
    "correlation_id": "a398e982-5de7-4cdf-859d-28fd445366f7",
    "reply_to": "callback_queue_name",
    "package_version": "8.0.412",
    "capacity_namespace": "test_environment_name",
    "capacity_slots_count": "100"
}
```

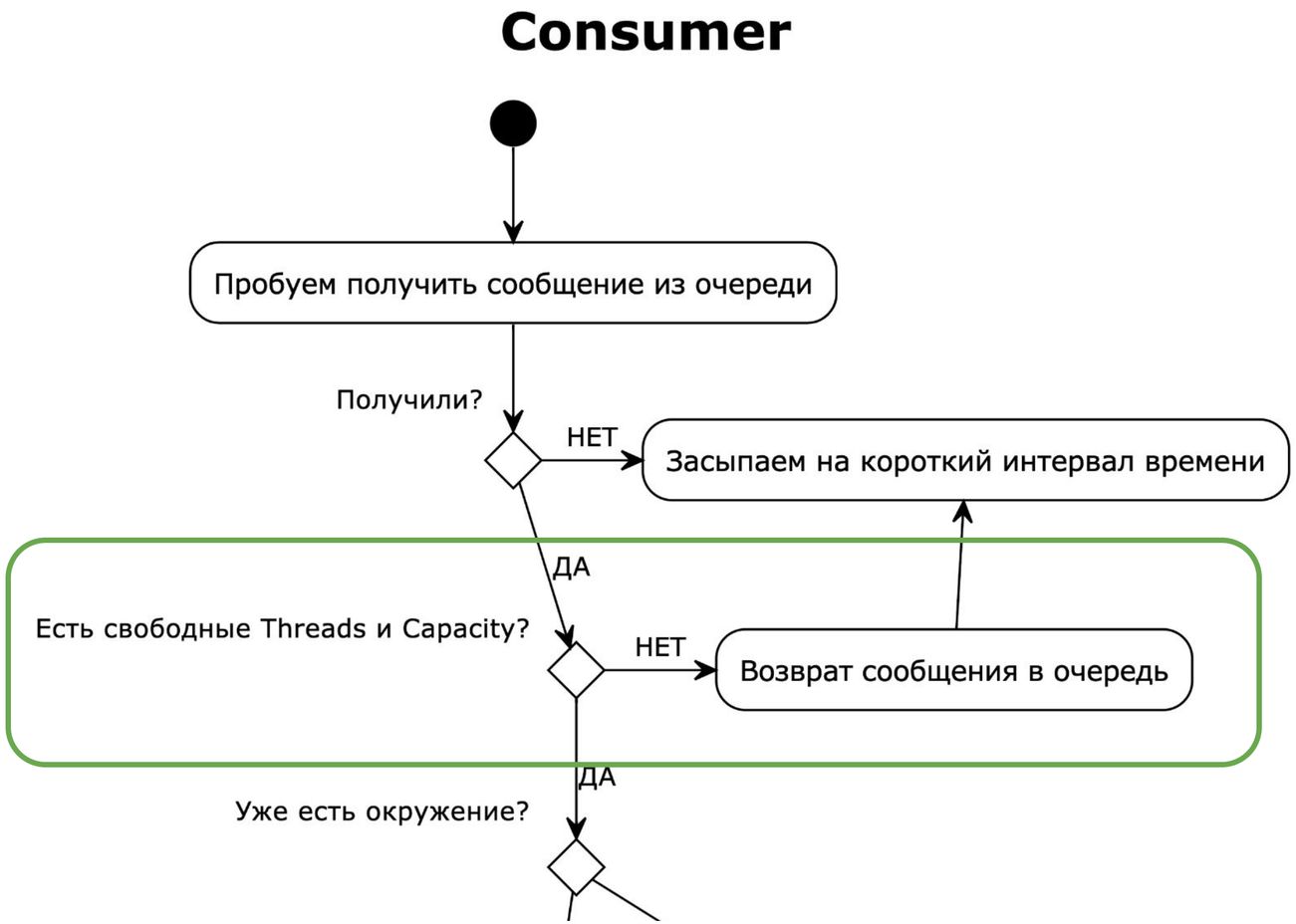
Реализация semaphore

```
def consume(self):
    while True:
        if self.threads["busy"] < self.threads["max"]:
            message = self.queue.get(fail=False, timeout=1)
            capacity_slot = self.capacity_slots_manager.acquire_slot(message)
            if message and capacity_slot:
                package_version = message['package_version']
                if not self.venv_manager.venv_exist(package_version):
                    self.venv_manager.create_venv(package_version)
                    self.venv_manager.install_package(package_version)
                self.threads["busy"] += 1
                asyncio.create_task(self.run_test(message))

            else:
                sleep(1)
```

Реализация semaphore

```
def acquire_slot(self, message):
    slot_namespace = message.headers.get("capacity_namespace")
    capacity_slot_count = message.headers.get("capacity_slot_count")
    capacity_semaphore = self._get(slot_namespace, capacity_slot_count)

    try:
        capacity_slot = capacity_semaphore.acquire()
    except NotAvailable:
        capacity_slot = None

    return capacity_slot
```

Реализация semaphore



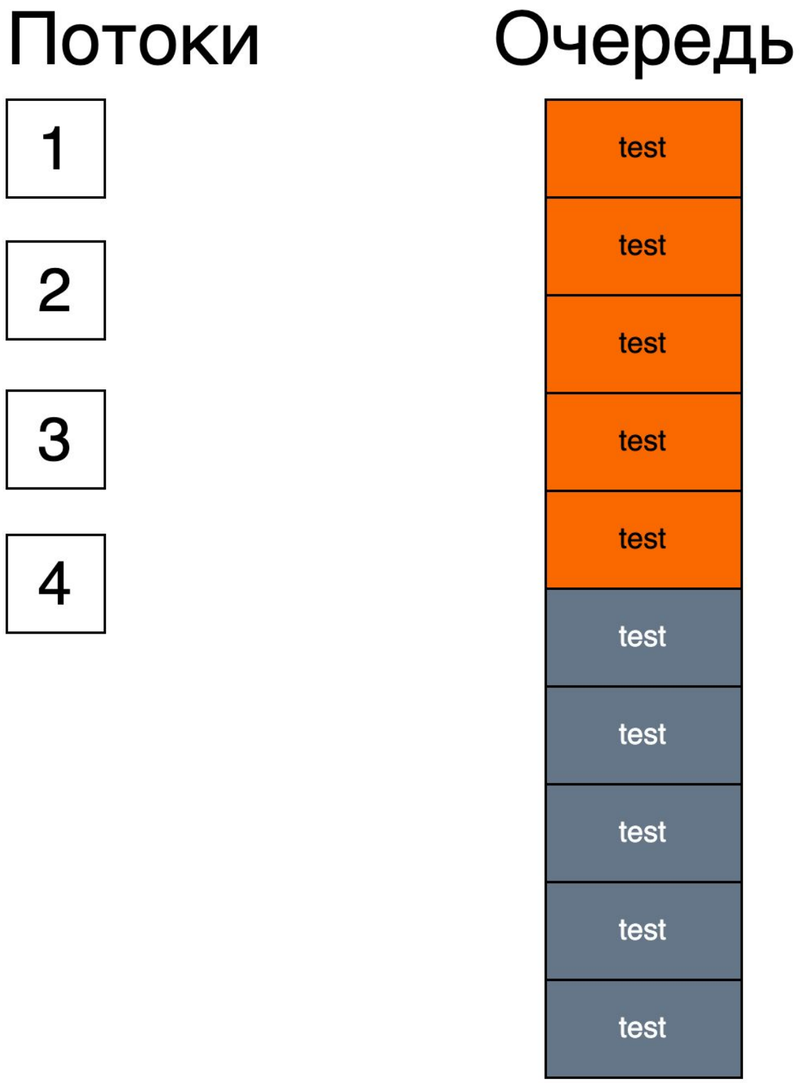
Несколько наборов тестов

- Как обрабатывать несколько одновременно запущенных тестовых наборов?
- Как запускать тесты для релизных веток в приоритете
- Как утилизировать все доступные ресурсы

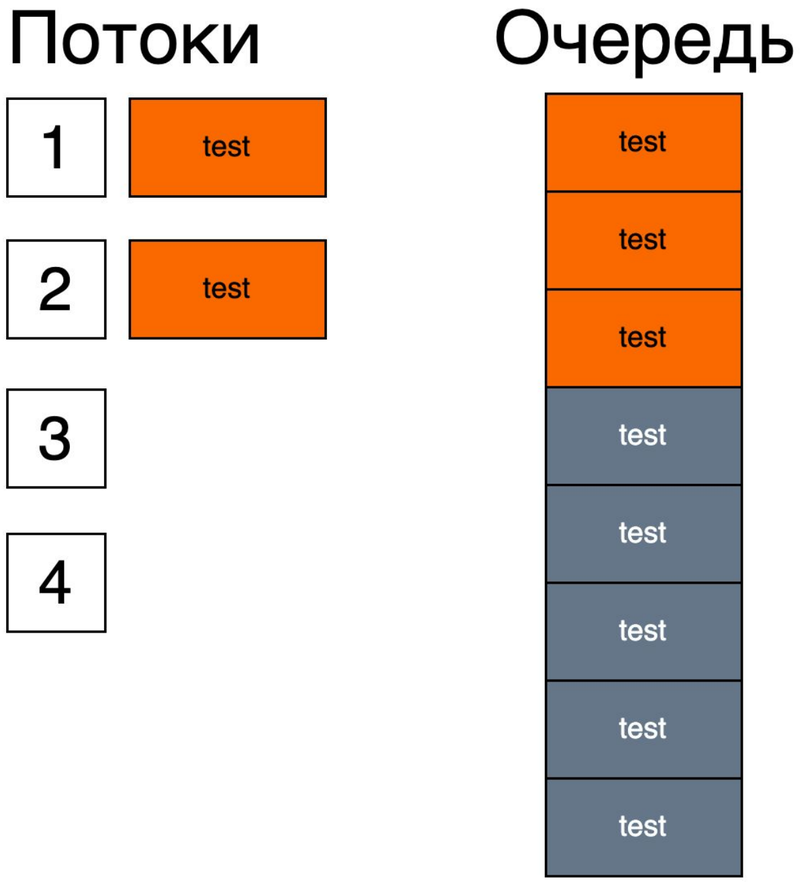
Несколько наборов тестов

- 2 набора тестов
- 5 тестов в наборе
- Ограничение сервиса - 4 потока
- Емкость тестовых окружений - 2 одновременных теста

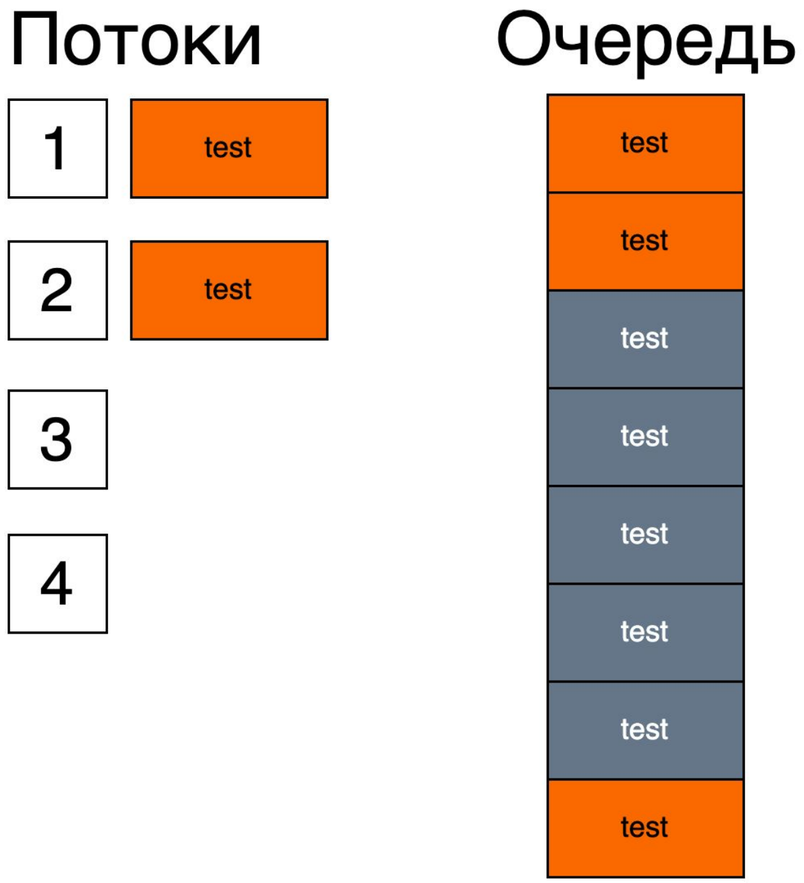
Несколько наборов тестов



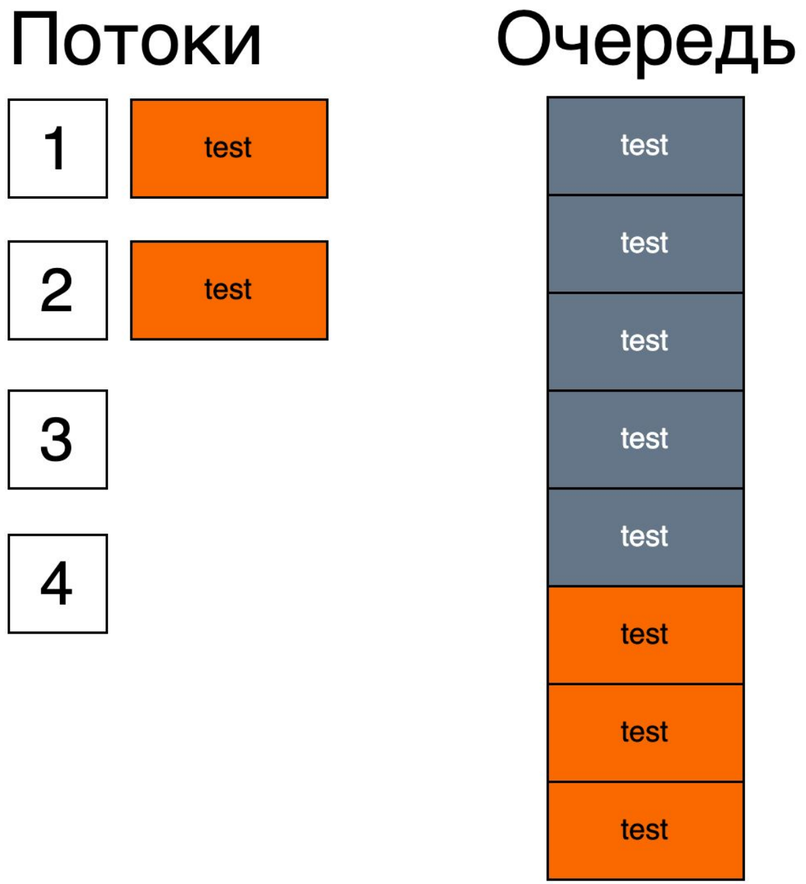
Несколько наборов тестов



Несколько наборов тестов

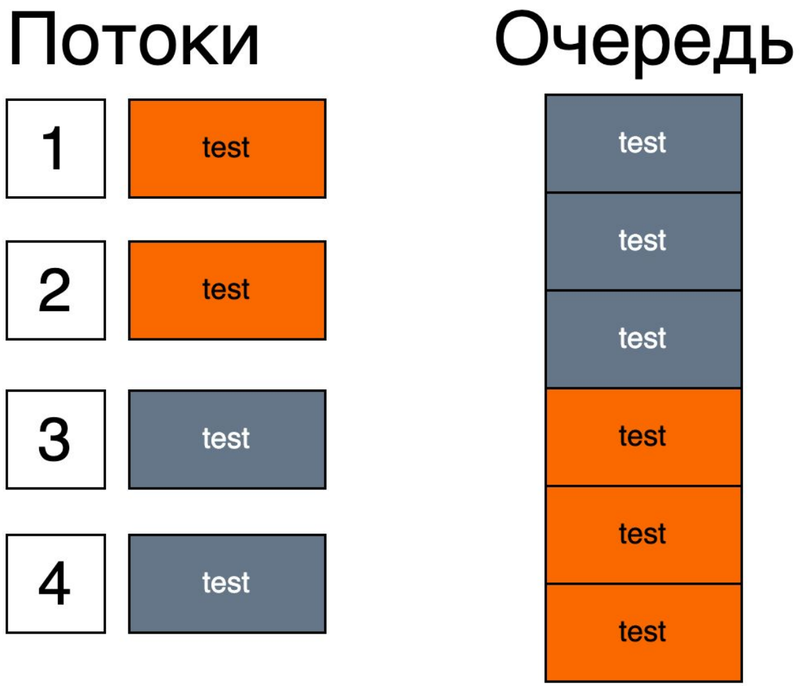


Несколько наборов тестов

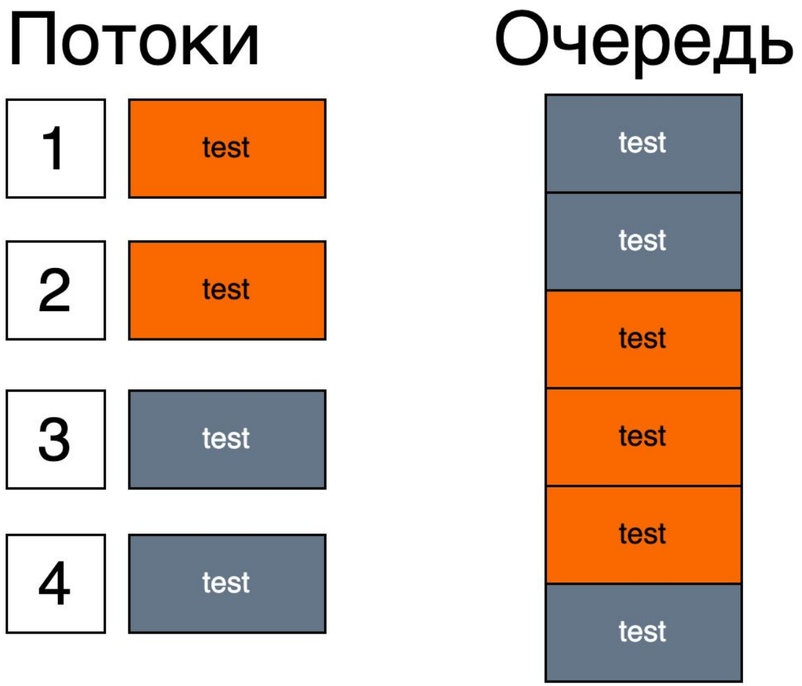


yclients

Несколько наборов тестов



Несколько наборов тестов



Несколько наборов тестов

- Зависимые между собой тестовые наборы
- Длительность прохождения каждого отдельного набора зависит от общего количества
- Добавить приоритет сообщениям?

Несколько наборов тестов

Решение - разделить приоритетные и неприоритетные тесты на отдельные очереди.

Несколько наборов тестов

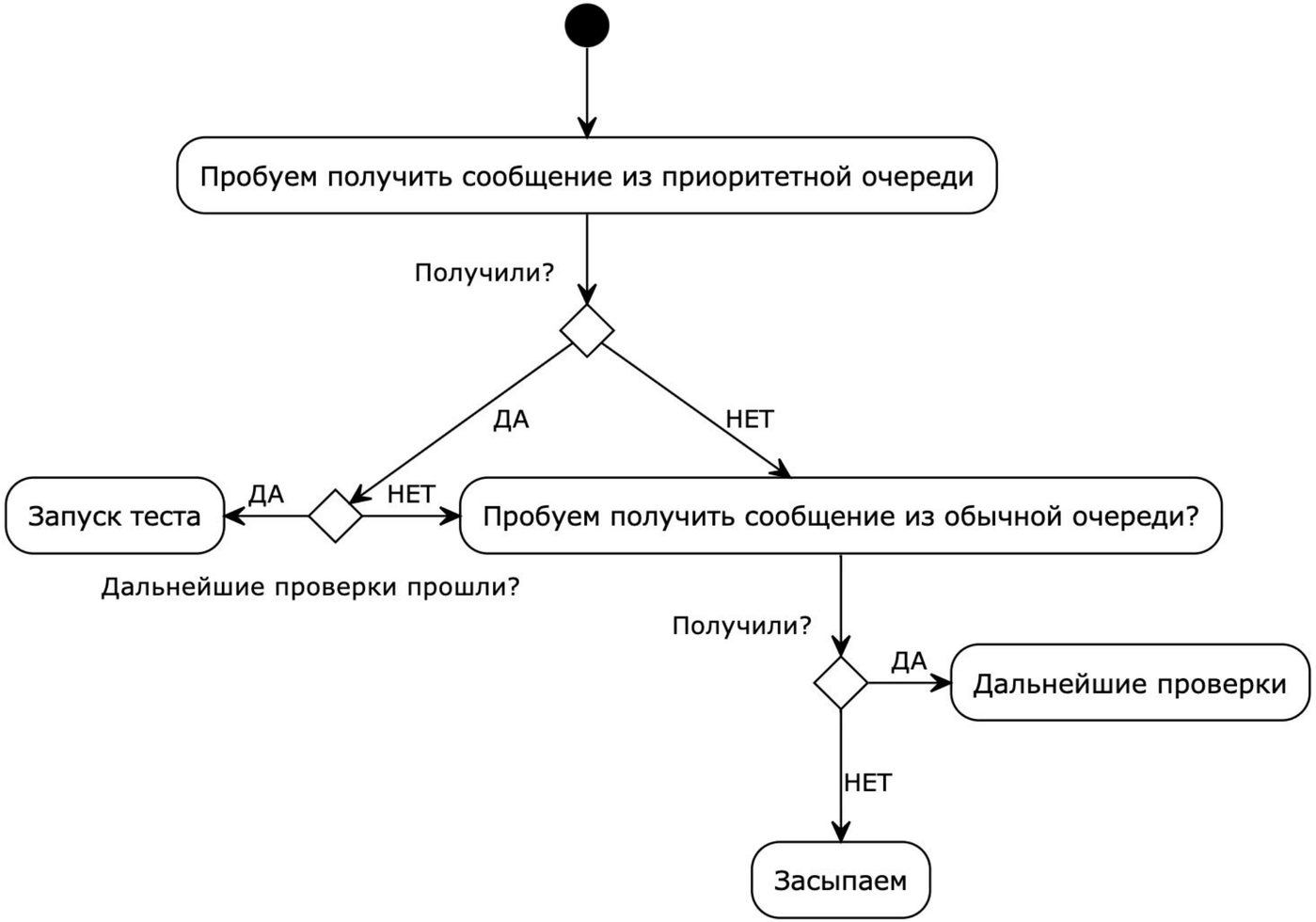
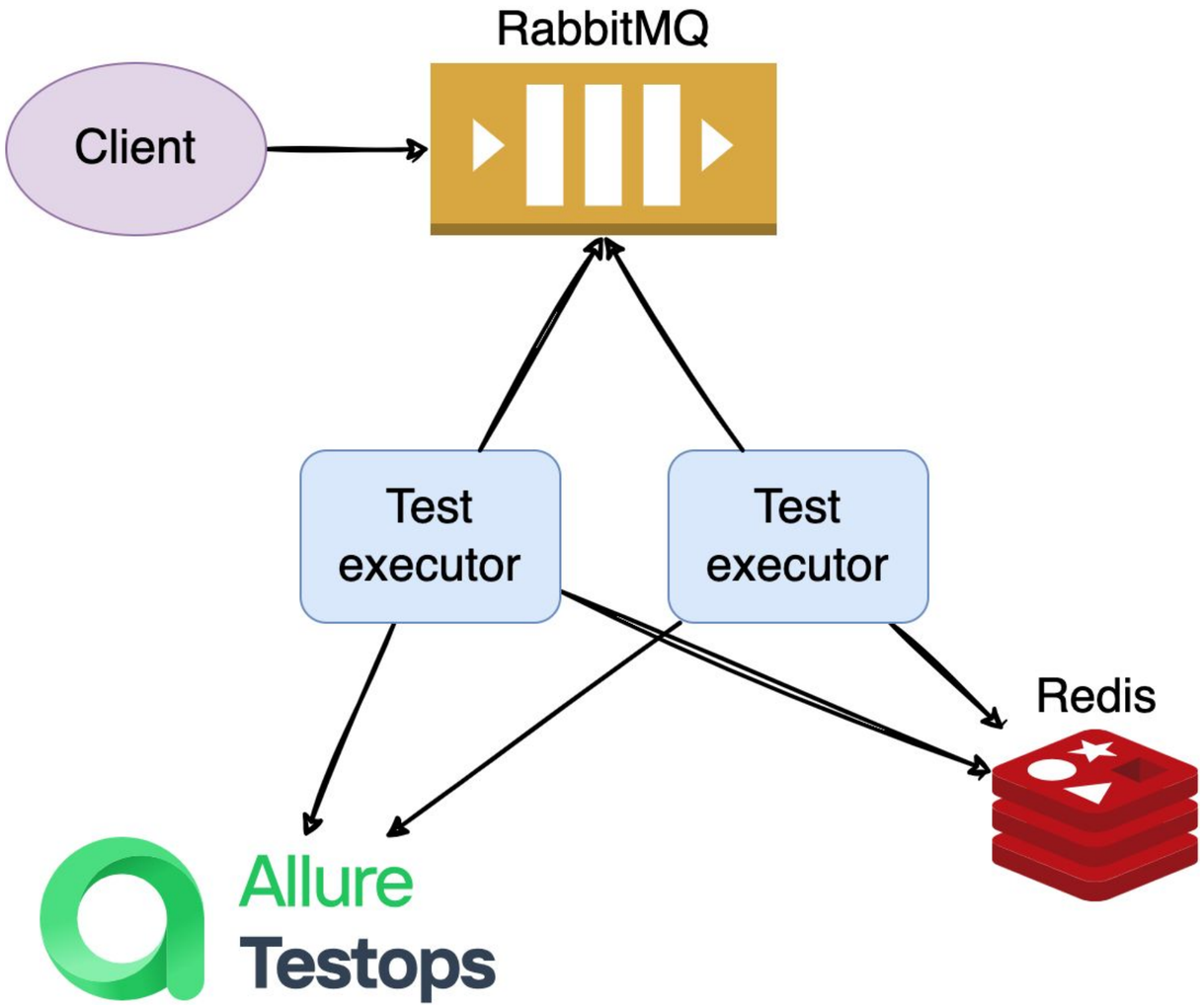


Схема развертывания сервиса

- **RabbitMQ** — брокер сообщений
- **Redis** — база данных в оперативной памяти
- Сервис исполняющий тесты
- **Allure TestOps**

Схема развертывания сервиса



Управление конфигурацией

- **IaC** подход
- **Ansible** роли
- Сервис - это **systemd unit**

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120


длительность сек.
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
03.03.23
05.03.23
07.03.23
09.03.23
11.03.23
13.03.23
15.03.23
17.03.23
19.03.23
21.03.23
23.03.23
25.03.23
27.03.23
29.03.23
31.03.23
02.04.23
100

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120
- Убрали ограничение на количество запущенных наборов

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120
- Убрали ограничение на количество запущенных наборов
- Лучше утилизируем доступные серверные мощности для запуска тестов

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120
- Убрали ограничение на количество запущенных наборов
- Лучше утилизируем доступные серверные мощности для запуска тестов
- Повысили стабильность прохождения тестов

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120
- Убрали ограничение на количество запущенных наборов
- Лучше утилизируем доступные серверные мощности для запуска тестов
- Повысили стабильность прохождения тестов
- Повысили скорость разработки тестов

Достигнутые результаты

- Сократили длительность прохождения тестовых наборов в 2 раза с 20 до 10 минут
- Увеличение потоков прохождения тестов более чем в 2 раза с 50 до 120
- Убрали ограничение на количество запущенных наборов
- Лучше утилизируем доступные серверные мощности для запуска тестов
- Повысили стабильность прохождения тестов
- Повысили скорость разработки тестов
- Релизы продуктовых проектов стали проходить значительно быстрее

Проблемы в продакш

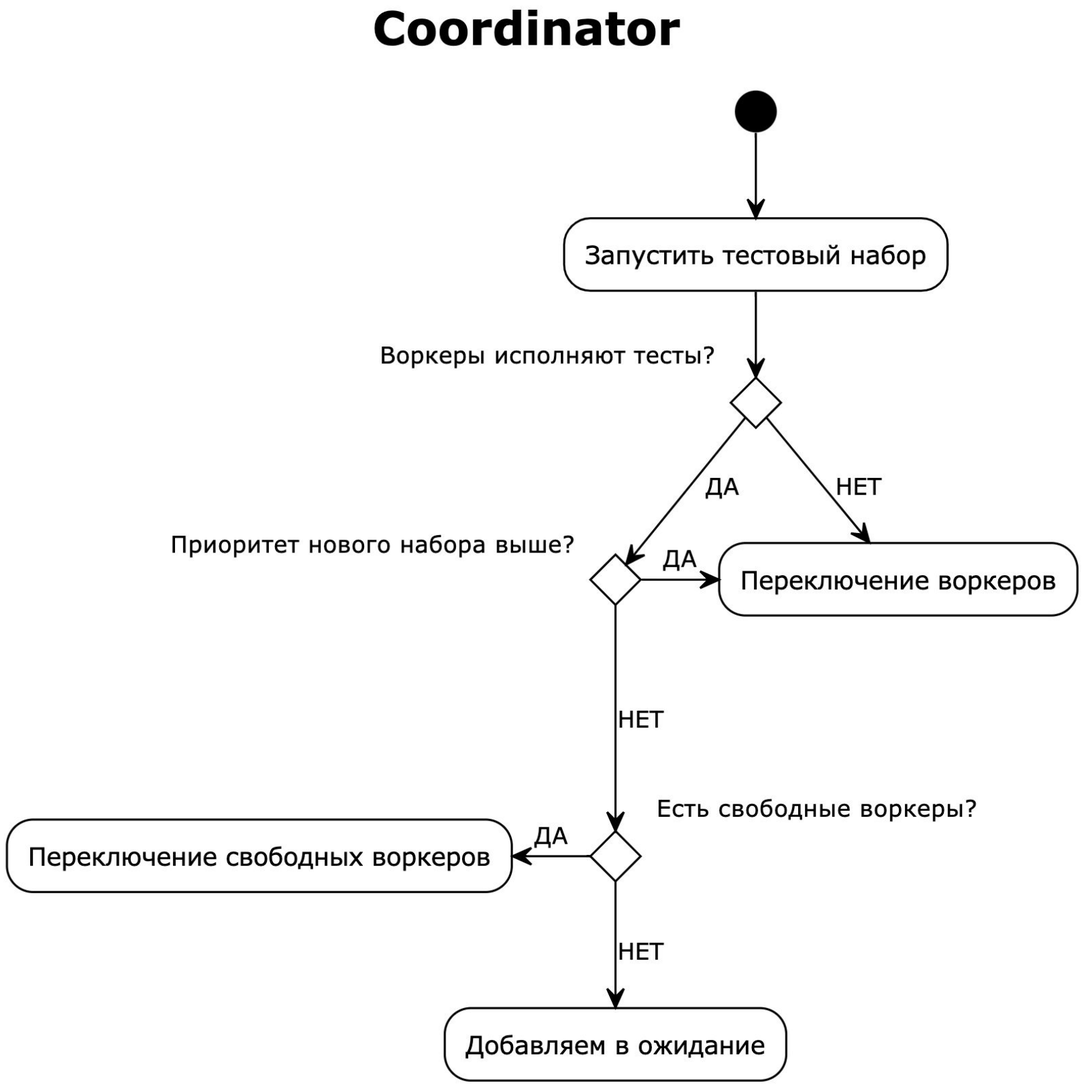
- Зависимость тестовых наборов с одинаковым приоритетом
- Сетевые флапы - кратковременные проблемы или неполадки в работе сетевого соединения.
- Недостаточная наблюдаемостью сервиса - метрики, алерты, логи
- Сложности с реализацией доработок и исправлений для сервиса

Эволюция MVP

Эволюция MVP

- **Celery —** распределенная платформа для асинхронной обработки задач в Python, использующая модель "брокер-воркер"
- Каждый тестовый набор будет отдельной очередью
- **Координатор —** управляющий сервис
- **Воркер —** исполняет тесты

Эволюция MVP



yclients

Эволюция MVP

Хотим в opensource

@RZLTRV

DI

@DISTRIBUTED_TESTS

yclients

Не бойтесь писать свои инструменты

Вопросы