Как плагины помогли нам переписать все тесты. И написать еще

Андрей Донцов

Обо мне

- Занимаюсь тестированием системного софта

- Автоматизировал энтерпрайз сервисы и интеграции в Dell
- Автоматизировал тестирование в Окко

Для кого этот доклад?

- Для тех, кто собирается сделать новый или отрефакторить старый тестовый фреймворк
- Для тех, кто слышал про плагины, но боялся спросить
- Для тех, кто не слышал про плагины

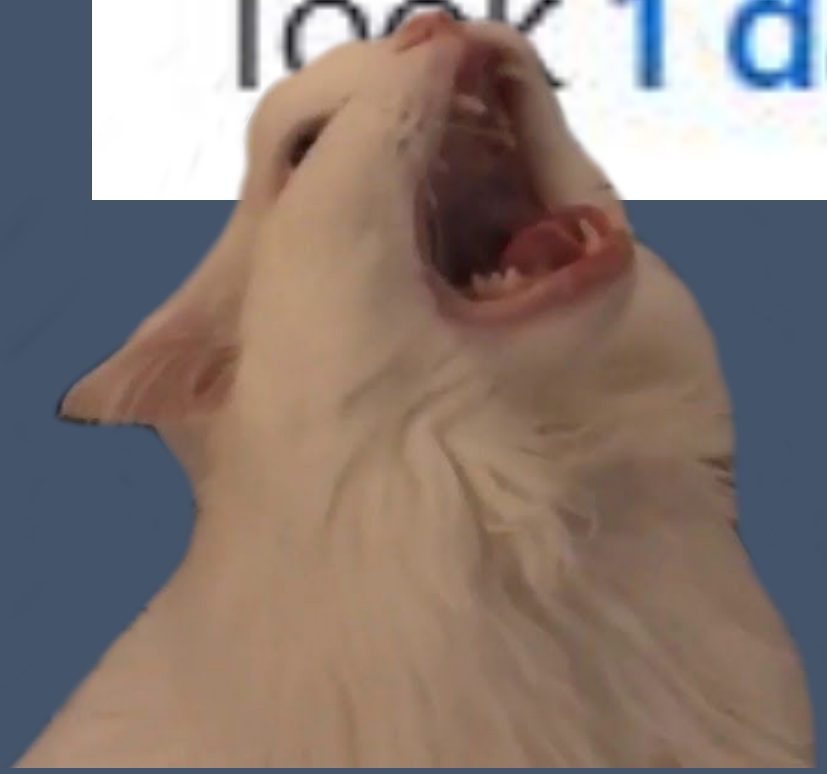
Откуда взялся доклад?

- Хочется больше поговорить о плагинах и возможностях, которые они дают
- Документация по плагинам размазана по разным разделам и не всегда достаточно понятная и явная

Какие тесты – плохие?

- Медленные
- Их трудно поддерживать
- Сложно реализованные сценарии
- Проблемы с версионированием
- Неинформативные репорты
- Bash скрипты вместо интеграции в ci

Медленные



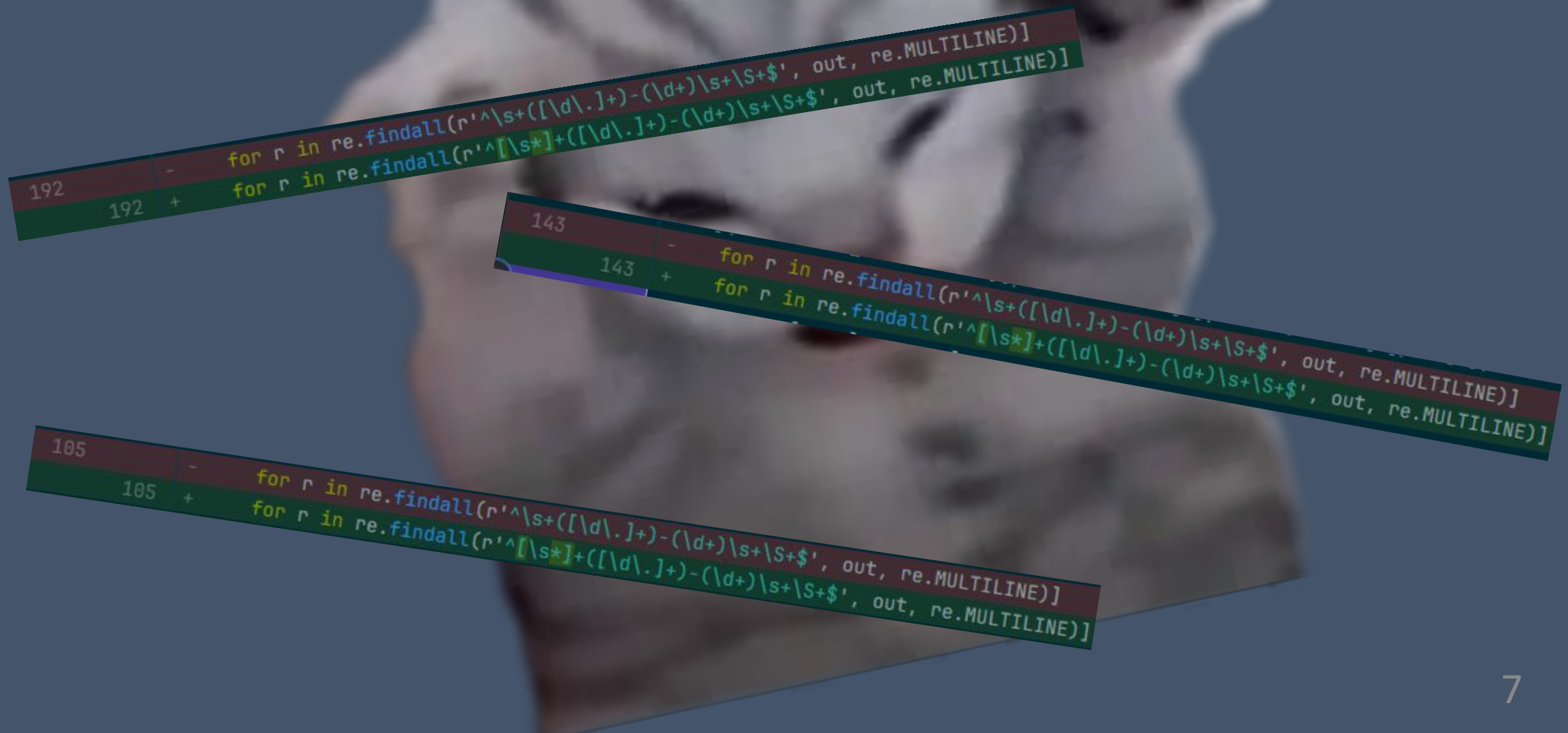
Took 1 day 17 hr

3 days 0 hr

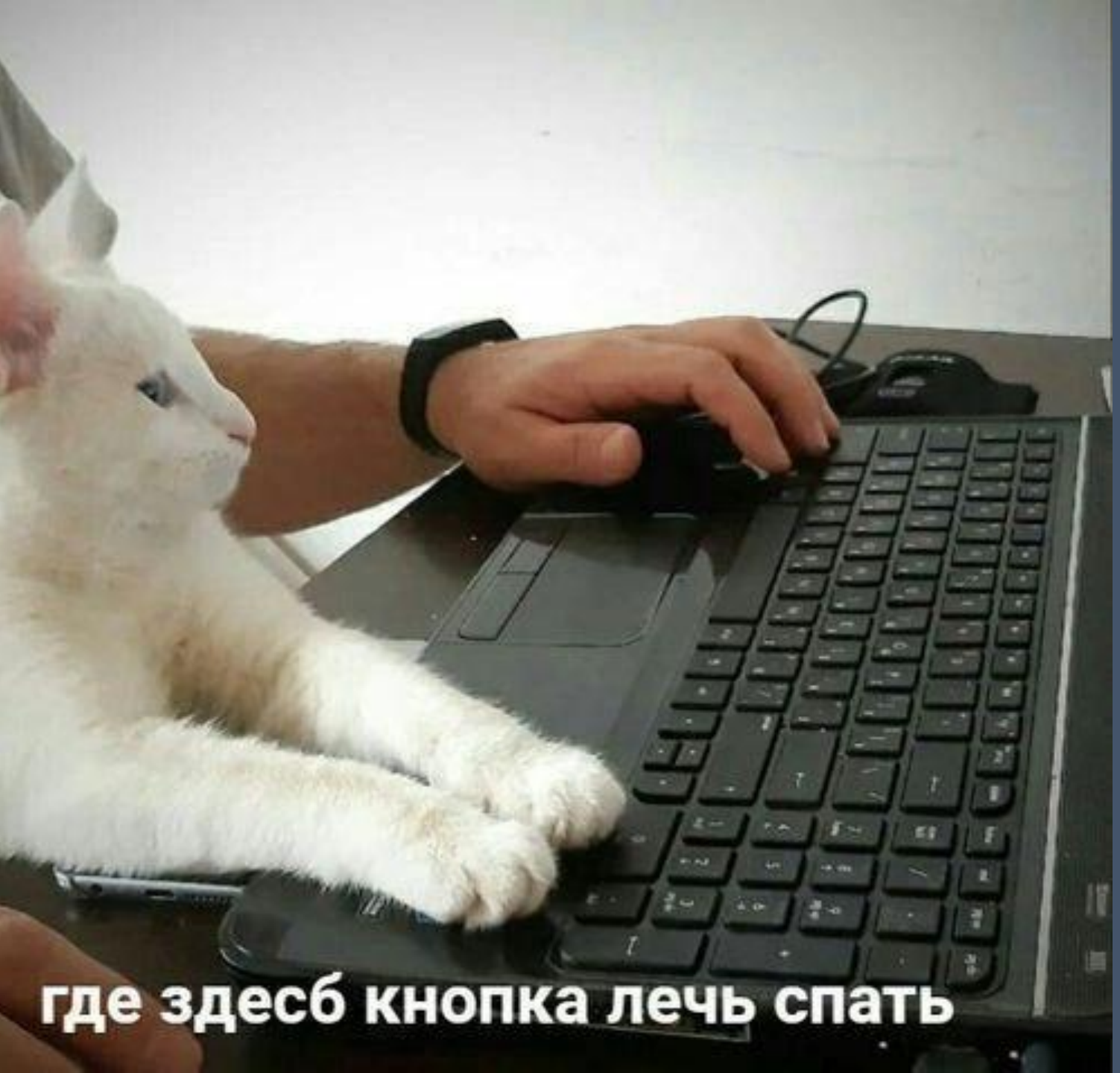
2 days 6 hr

Took 17 hr

Их трудно поддерживать



(правда) Переусложненные сценарии



Ветки в гите – плохой способ версионирования

Неинформативные репорты

Console Output (parsed)

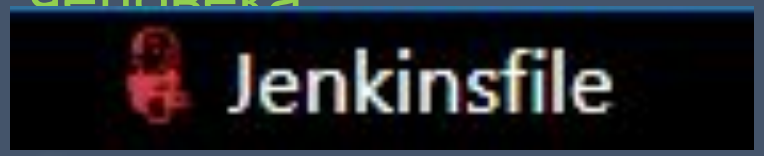
Show only this

- Error (8)
- Warning (0)
- Info (0)
- Debug (0)

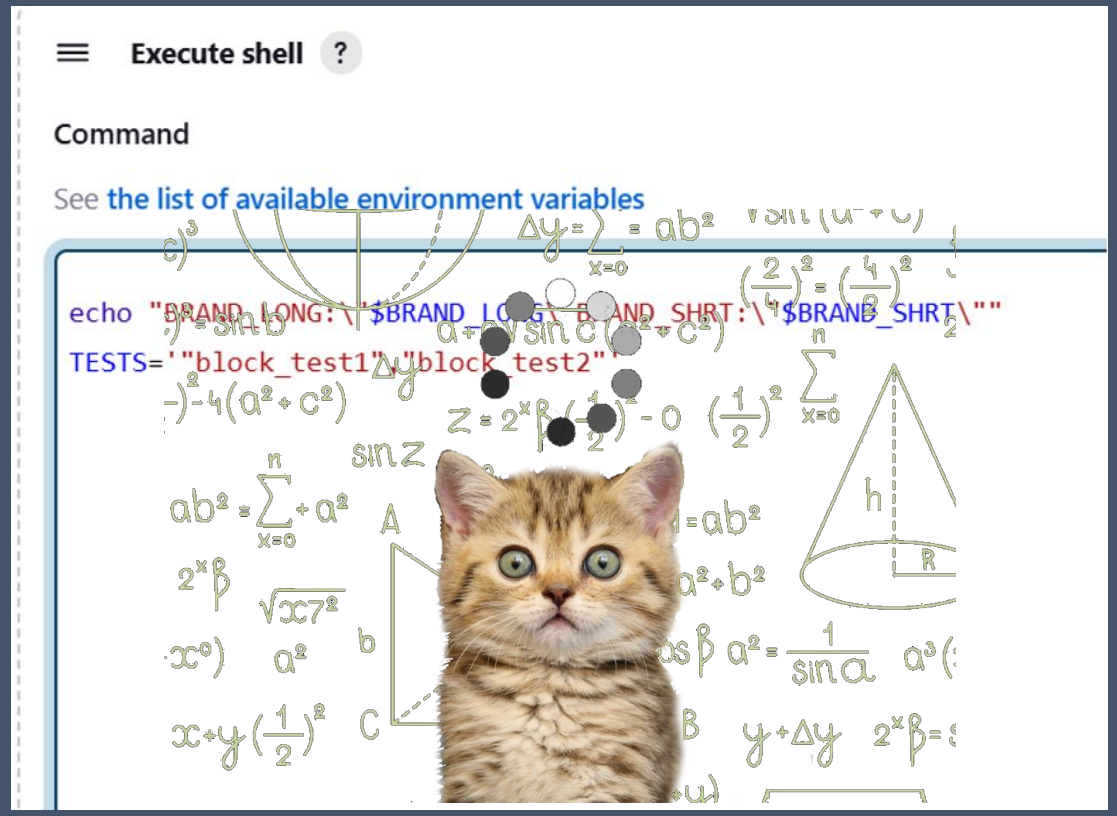
testrun sample 501.00 MB view

Bash скрипты вместо интеграций в ci

Jenkinsfile здорового человека

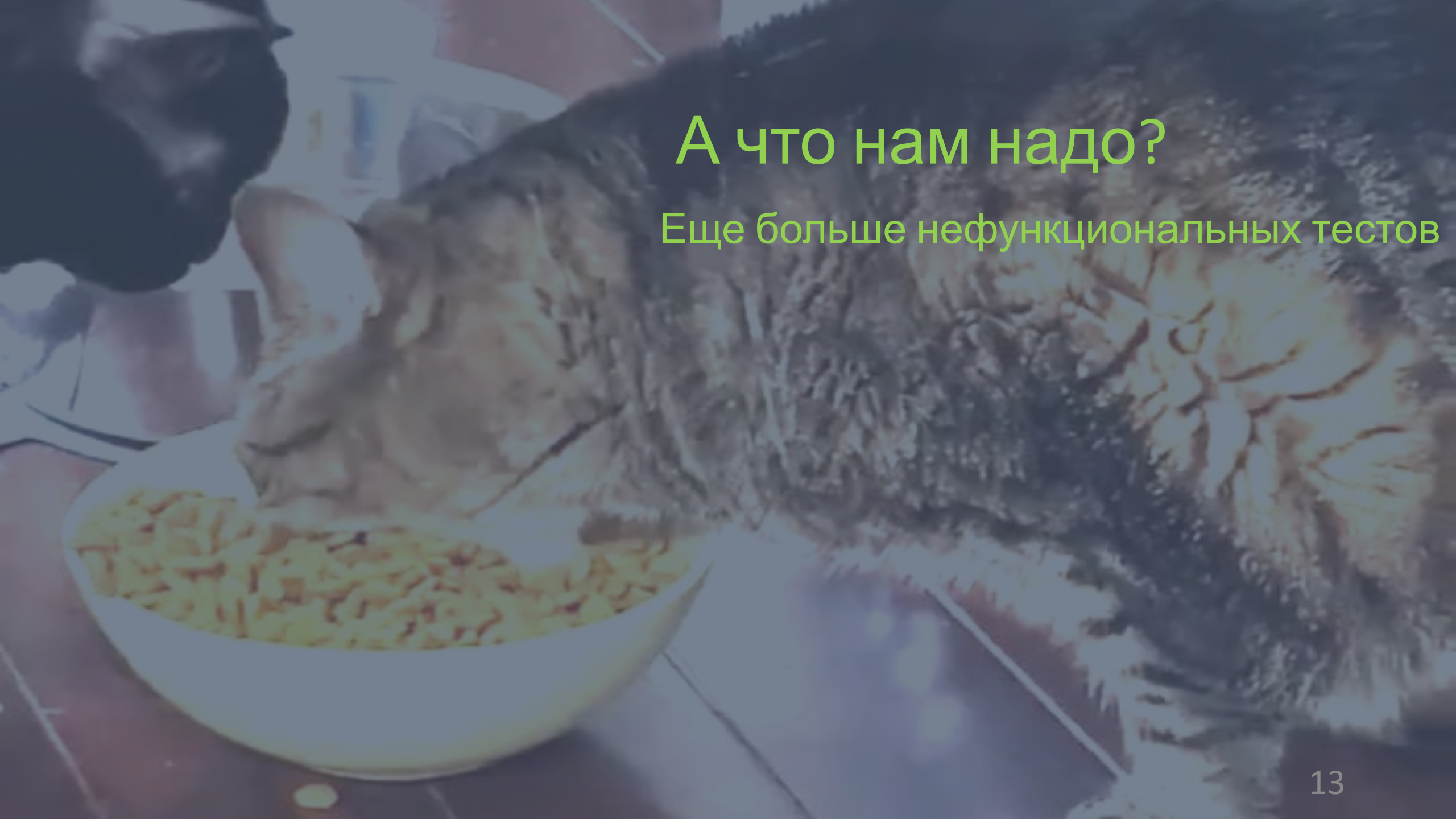


Jenkinsfile курильщика



А что нам надо?

Много функциональных тестов (как и всем)



А что нам надо?

Еще больше нефункциональных тестов

А что нам надо?

Параметризуемые и
совместимые тесты

Гибкие тесты – настройка контекста

1. Фикстуры
2. Управление pytest

И тут - плагины!

Кто такие ваши плагины?

Что нам говорит документация?
«A plugin contains one or multiple hook functions»…

pytest: writing plugins

Что такое хуки?

Хуки – специальные функции, которые позволяют изменять **поведение pytest** в определенных местах тестовой сессии.

pytest: writing hook functions

Но есть нюанс!

- In case **you want to use fixtures** from a project that does not use entry points, you can **define pytest_plugins** in your top conftest.py file to register that module as a plugin.

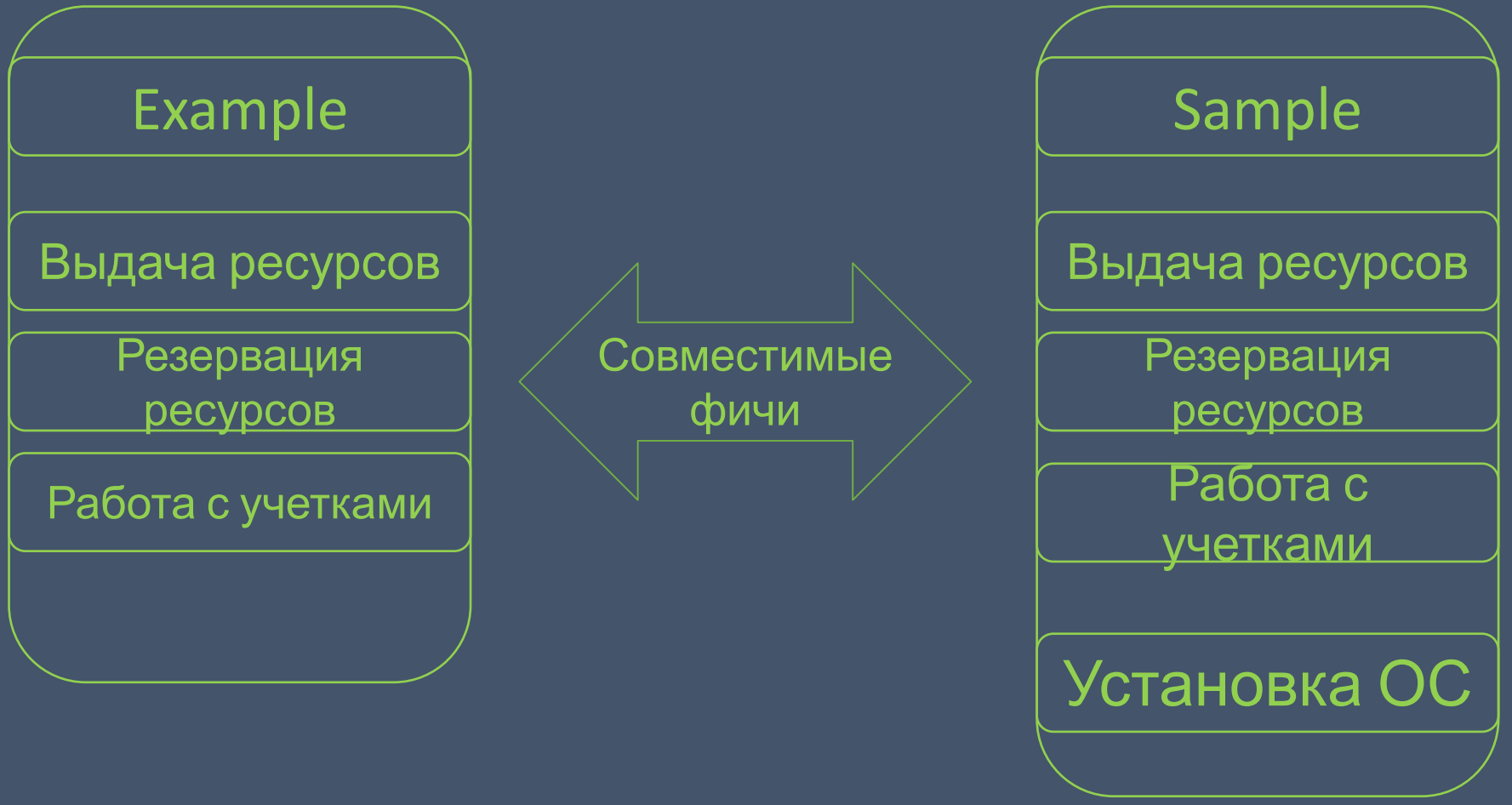
pytest: how to write fixtures

Тренируемся на кошках

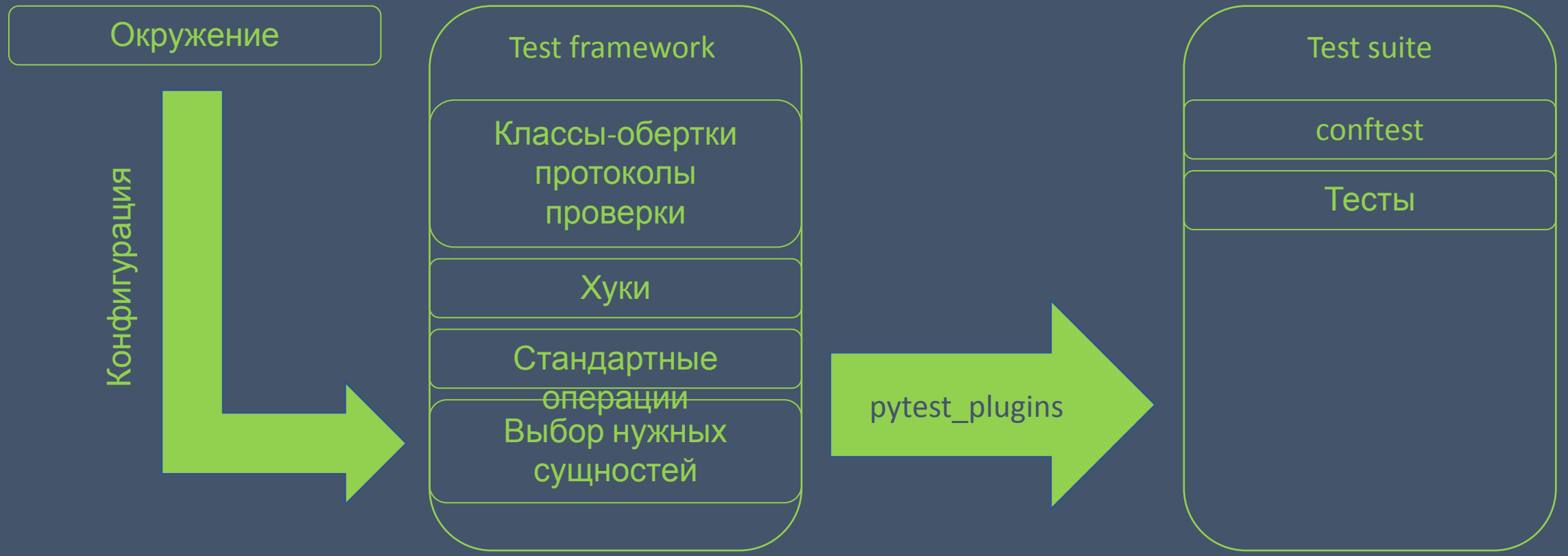
Два продукта – один тест

Посмотреть исходники

Тренируемся на кошках

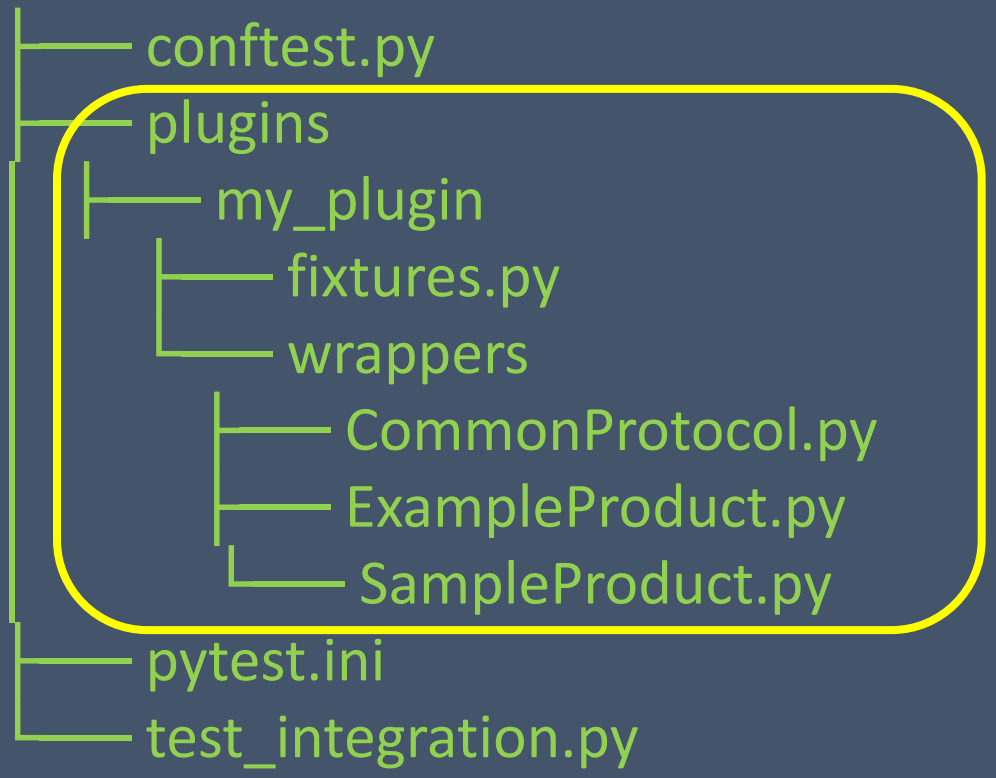


Строим фреймворк на основе плагинов



Общие сценарии для всех продуктов

Структура проекта



Реализуем плагин: фикстуры

```
@pytest.fixture(scope="session")
def product():
    if os.getenv("TARGET_PRODUCT") == "example":
        return ExampleProduct("https://www.example.com")
    elif os.getenv("TARGET_PRODUCT") == "sample":
        return SampleProduct("https://www.sample.com")
    else:
        raise ValueError(f"Unexpected product configured
{os.getenv('TARGET_PRODUCT')}")
```

Реализуем плагин: протокол

```
from abc import ABC, abstractmethod


class CommonProtocol(ABC):
    @abstractmethod
    def create_new_admin_user_session(self, *args, **kwargs):
        pass

    @abstractmethod
    def get_resources(self, *args, **kwargs):
        pass

    @abstractmethod
    def reserve_resource(self, resource_config, user_data, *args, **kwargs):
        pass
```

Реализуем плагин: клиенты

```
class ExampleProduct(CommonProtocol):
    ...

class SampleProduct(CommonProtocol):
    ...
```

Реализуем плагин: фикстуры

```
@pytest.fixture(scope="session")
def admin_user(product):
    user_data =
product.create_new_admin_user_session()["response"]
    if os.getenv("TARGET_PRODUCT") == "example":
        registration_response = product.register_admin_token(
            user_data["token"], user_data["user_id"]
        ).json()
        if not
registration_response["response"]["registered"]:
            pytest.fail(f"Cant register user
{user_data['user_id']=} as admin")
    return user_data
```

Реализуем плагин: фикстуры

```
@pytest.fixture(scope="session")
def admin_user(product):
    user_data =
product.create_new_admin_user_session()["response"]
    if os.getenv("TARGET_PRODUCT") == "example":
        registration_response = product.register_admin_token(
            user_data["token"], user_data["user_id"]
        ).json()
        if not
registration_response["response"]["registered"]:
            pytest.fail(f"Cant register user
{user_data['user_id']=} as admin")
    return user_data
```

Реализуем плагин: фикстуры

```
@pytest.fixture()
def create_reservation_config(product):
    def factory(vendor, drives_type):
        available_resources = product.get_resources()
        target_server = next(
            server
            for server in available_resources["servers"]
            if server["manufacturer"] == vendor
        )
        required_drives = target_server["disk_count"]
        prepared_drives = random.choice(
                [
                    drive_type
                    for drive_type in available_resources["drives"]
                    if (drive_type["available"] >= required_drives)
                    and (drive_type["type"] == drives_type)
                ]
            )
        return {"server": target_server, "drives": prepared_drives}
```

Реализуем плагин: фикстуры

```
@pytest.fixture()
def create_reservation(product, admin_user, release_resources):
    def factory(reservation_config):
        reservation_data = product.reserve_resource(
            resource_config=reservation_config,
            user_data=admin_user,
        ).json()
        product.wait_deploy(
            reservation_data["reservation_id"], reservation_data["reserver_id"]
        )
        if os.getenv("TARGET_PRODUCT") == "sample":
            product.install_os(
                reservation_config,
                reservation_data["reservation_id"],
                product.get_available_os(filter={"name": "CentOS"}),
            )
        return reservation_data
    return factory
```

Тестовый сценарий

1. Подготовить конфигурацию
2. Забронировать ресурс
3. Проверить, что резервация создана
4. Проверить, что состояние ресурса изменилось

Подключаем плагины

```
# conftest.py

pytest_plugins = ("plugins.my_plugin.fixtures",)
```

Реализация теста

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Реализация теста

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Шаг 1

Реализация теста

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Шаг 2

Реализация теста

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Проверки

Реализация теста

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Промежуточные итоги: гибкие тесты

- Сам тест – декларативный набор инструкций

Промежуточные итоги: совместимые тесты

- Сам тест – декларативный набор инструкций
- Контролируем совместимость клиентов

Промежуточные итоги: фикстуры

- Сам тест – декларативный набор инструкций
- Контролируем совместимость
- Комплексные операции сокрыты в фикстурах

Промежуточные итоги: фикстуры

- Сам тест – декларативный набор инструкций
- Контролируем совместимость
- Комплексные операции сокрыты в фикстурах
- Реализованные однажды фикстуры легко распространять

Совместимые ассерты

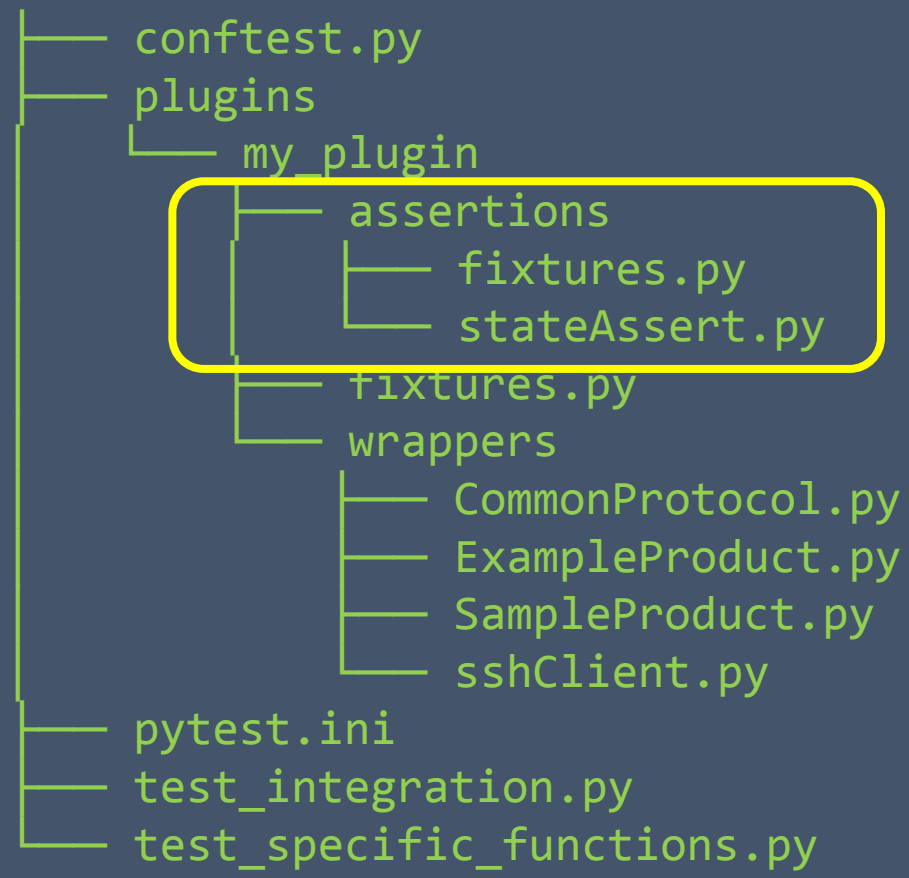
```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create(
    product, target_vendor, create_reservation_config, create_reservation,
drives_type
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert product.get_resource_status(reservation_config["server"]["model"])[
        "is_reserved"
    ]
```

Совместимые ассерты: данные

```
{
  "resource": resource_model,
  "is_reserved": True
}
```

```
{
  "resource": resource_model,
  "is_reserved": True,
  "os_installation": "complete",
}
```

Совместимые ассерты



Совместимые ассерты: фикстура

```
@pytest.fixture()
def assert_resource_reserved(product):
    def factory(resource_data):
        reservation_data = product.get_resource_status(resource_data)
        if os.getenv("TARGET_PRODUCT") == "example":
            assert reservation_data["is_reserved"]
        elif os.getenv("TARGET_PRODUCT") == "sample":
            assert reservation_data["is_reserved"]
            assert reservation_data["os_installation"] == "complete"

    return factory
```

Совместимые ассерты: фикстура

```
@pytest.fixture()
def assert_resource_reserved(product):
    def factory(resource_data):
        reservation_data = product.get_resource_status(resource_data)
        if os.getenv("TARGET_PRODUCT") == "example":
            assert reservation_data["is_reserved"]
        elif os.getenv("TARGET_PRODUCT") == "sample":
            assert reservation_data["is_reserved"]
            assert reservation_data["os_installation"] == "complete"

    return factory
```

Совместимые ассерты: модуль

```
class StateAssertions:
    PRODUCTS_ASSERTIONS_MAPPING = {
        "resource_state": {
            "sample": "sample_resource_reserved",
            "example": "example_resource_reserved",
        }
    }

    @classmethod
    def assert_resource_reserved(cls, reservation_data):
        target_assertion = getattr(
            cls,
            cls.PRODUCTS_ASSERTIONS_MAPPING["resource_state"].get(
                os.getenv("TARGET_PRODUCT")
            ),
        )
        return target_assertion(reservation_data)
```

Совместимые ассерты: модуль

```
class StateAssertions:
    PRODUCTS_ASSERTIONS_MAPPING = {
        "resource_state": {
            "sample": "sample_resource_reserved",
            "example": "exmaple_resource_reserved",
        }
    }

    @classmethod
    def assert_resource_reserved(cls, reservation_data):
        target_assertion = getattr(
            cls,
            cls.PRODUCTS_ASSERTIONS_MAPPING["resource_state"].get(
                os.getenv("TARGET_PRODUCT")
            ),
        )
        return target_assertion(reservation_data)
```

Совместимые ассерты: модуль

```
@classmethod
def sample_resource_reserved(cls, reservation_data):
  assert reservation_data["is_reserved"]
  assert reservation_data["os_installation"] == "complete"

@classmethod
def example_resource_reserved(cls, reservation_data):
  assert reservation_data["is_reserved"]
```

Совместимые ассерты: подключаем плагины

```
pytest_plugins = (
    "plugins.my_plugin.fixtures",
    "plugins.my_plugin.assertions.fixtures",
)
```

Совместимые ассерты: тест

```
@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create_compable_asserts(
    product,
    target_vendor,
    create_reservation_config,
    create_reservation,
    drives_type,
    assert_resource_reserved,
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    assert_resource_reserved(
        product.get_resource_status(reservation_config["server"]["model"])
    )
```

Совместимые ассерты: тест

```
from plugins.my_plugin.assertions import StateAssertions

@pytest.mark.universal
@pytest.mark.parametrize("target_vendor", ["Tyan", "Fujitsu", "Dell"])
@pytest.mark.parametrize("drives_type", ["HDD", "SSD"])
def test_reservation_create_compable_asserts(
    product,
    target_vendor,
    create_reservation_config,
    create_reservation,
    drives_type,
):
    reservation_config = create_reservation_config(target_vendor, drives_type)
    reservation_data = create_reservation(reservation_config)
    assert reservation_data["reservation_id"]
    StateAssertions.assert_resource_reserved(
        product.get_resource_status(reservation_config["server"]["model"])
    )
```

Промежуточные выводы (снова)

- Плагины можно использовать как обычные Python модули

Промежуточные выводы (снова)

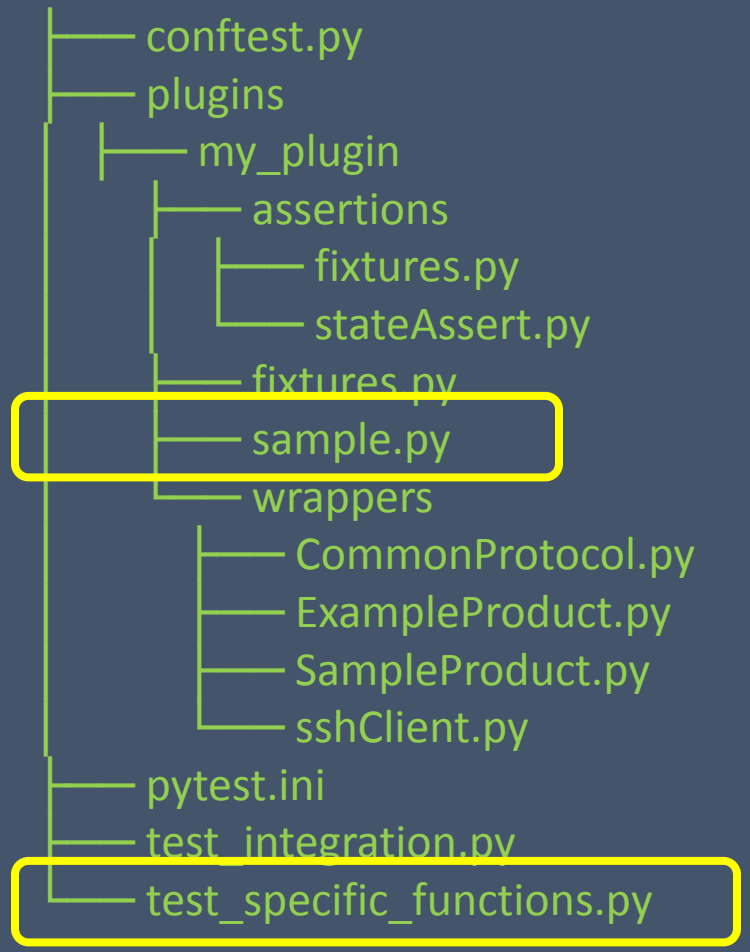
- Плагины можно использовать как обычные Python модули
- pytest_plugins позволяет управлять видимостью фикстур в тестах

А если нужны специфические тесты

Предусловие: тест работает только на SampleProduct

1. Забронировать ресурс
2. Установить на полученный сервер ОС
3. Дождаться завершения установки
4. Проверить, что установка завершилась корректно
5. Подключиться к серверу
6. Проверить, что установилась нужная ОС

А если нужны специфические тесты



А если нужны специфические тесты

```
@pytest.mark.sample
@pytest.mark.parametrize("vendor", ["Tyan", "Fujitsu", "Dell"])
def test_sample_os_installation(
    product, create_sample_reservation, vendor, create_ssh_connection
):
    available_resources = product.get_resources()
    # prepare reservation configuration
    reservation_data = create_sample_reservation(reservation_config)
    product.install_os(
        reservation_config,
        reservation_data["reservation_id"],
        product.get_available_os(filter={"name": "Ubuntu Server"}),
    )
    installation_data =
product.wait_os_installation(reservation_data["reservation_id"])
    assert installation_data["installation_status"] == "complete"
    deployed_resource_connection = create_ssh_connection(ip="111.111.11.11")
    assert deployed_resource_connection.get_os()
```

А если нужны специфические тесты

```
@pytest.mark.sample
@pytest.mark.parametrize("vendor", ["Tyan", "Fujitsu", "Dell"])
def test_sample_os_installation(
    product, create_sample_reservation, vendor, create_ssh_connection
):
    available_resources = product.get_resources()
    # prepare reservation configuration
    reservation_data = create_sample_reservation(reservation_config)
    product.install_os(
        reservation_config,
        reservation_data["reservation_id"],
        product.get_available_os(filter={"name": "Ubuntu Server"}),
    )
    installation_data =
product.wait_os_installation(reservation_data["reservation_id"])
    assert installation_data["installation_status"] == "complete"
    deployed_resource_connection = create_ssh_connection(ip="111.111.11.11")
    assert deployed_resource_connection.get_os()
```

А если нужны специфические тесты

```python
@pytest.fixture(scope="session")
def product():
    return SampleProduct("https://www.sample.com")


@pytest.fixture()
def create_sample_reservation(product):
    def factory(reservation_config):
        reservation_data = product.reserve_resource(
            resource_config=reservation_config,
            user_data=product.create_new_admin_user_session()["response"],
        ).json()
        product.wait_deploy(
            reservation_data["reservation_id"], reservation_data["reserver_id"]
        )
        return reservation_data

    return factory
```

А если нужны специфические тесты: подключаем

```
# Доступ только к специфическим фикстурам
pytest_plugins = ("plugins.my_plugin.sample",)
```

А если нужны специфические тесты: подключаем

```
# И так тоже можно
pytest_plugins = (
  "plugins.my_plugin.sample",
  "plugins.my_plugin.fixtures",
  "plugins.my_plugin.assertions.fixtures",
)
```

Выводы

- Плагины – способ распространить реализованные однажды фикстуры

Выводы

- Плагины – способ распространить реализованные однажды фикстуры
- Разные продукты – одинаковые сценарии

Выводы

- Плагины – способ распространить реализованные однажды фикстуры
- Разные продукты – одинаковые сценарии
- «Сильные» связи – это не страшно

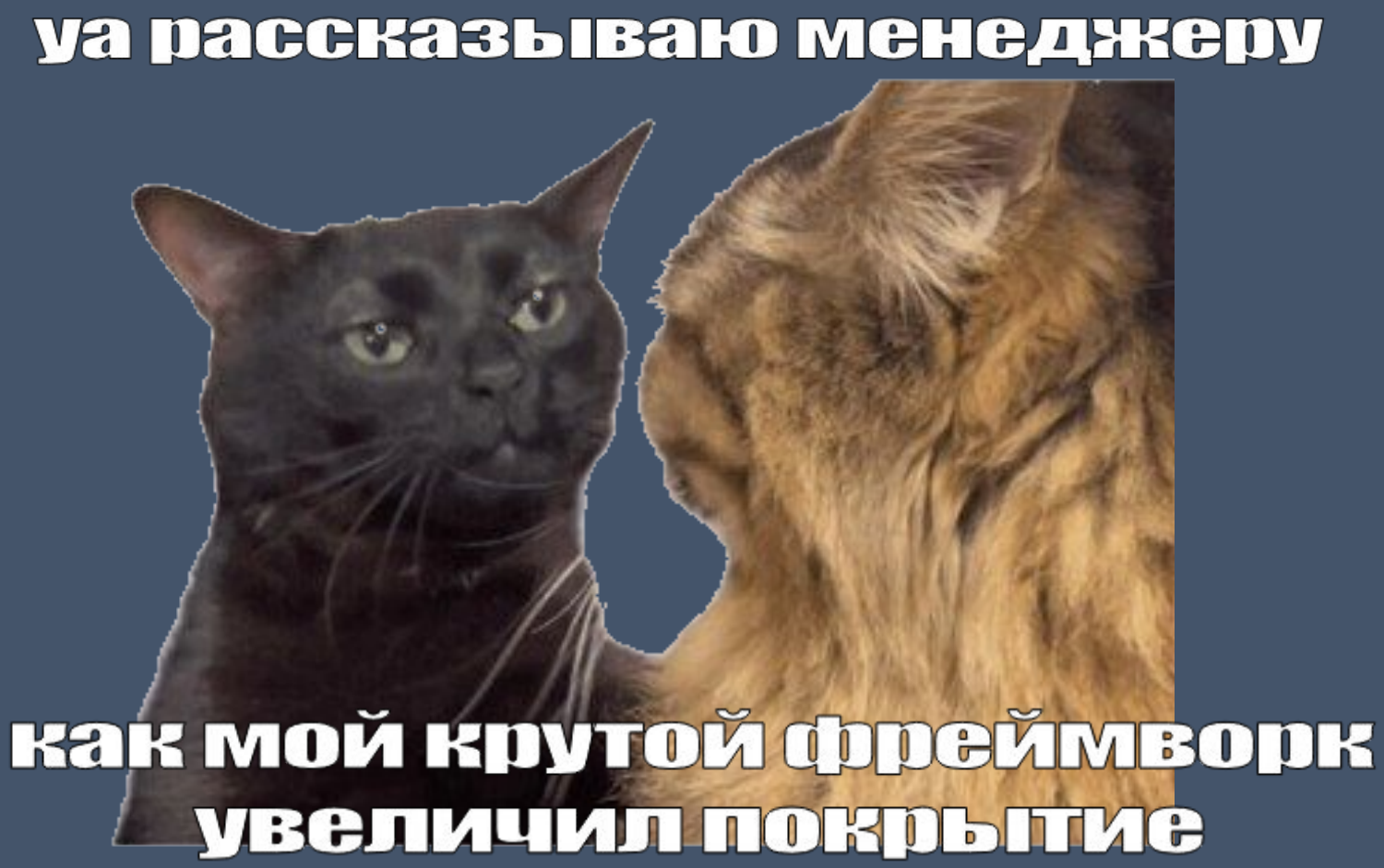
Выводы

- Плагины – способ распространить реализованные однажды фикстуры
- Разные продукты – одинаковые сценарии
- «Сильные» связи – это не страшно
- «Сильные» связи – большая ответственность

Где это использоват

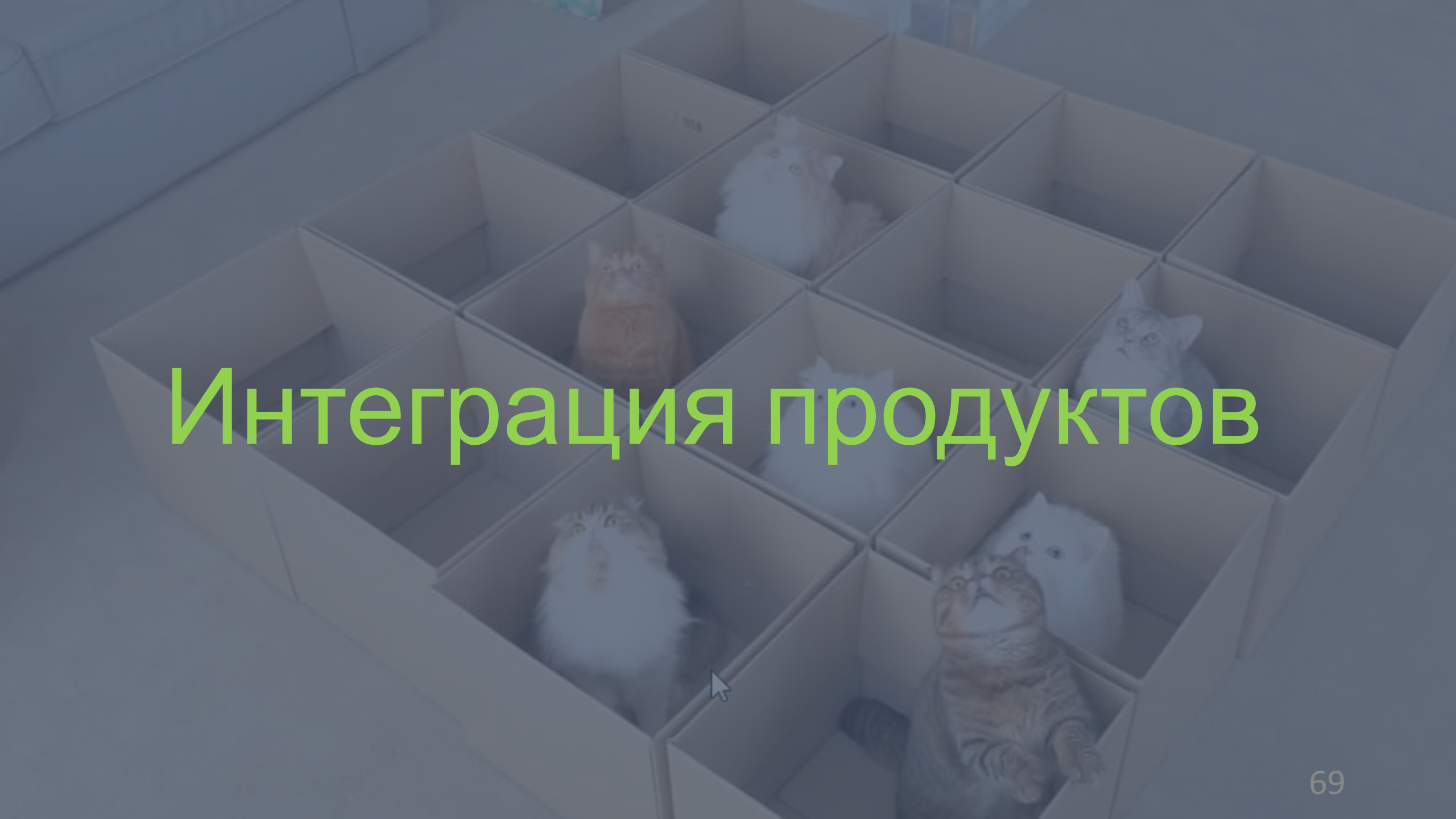
- Системный софт
- Работающий на нескольких платформах софт
- Сервисы, с разными клиентами или API
- Моки

А откуда больше тестов?



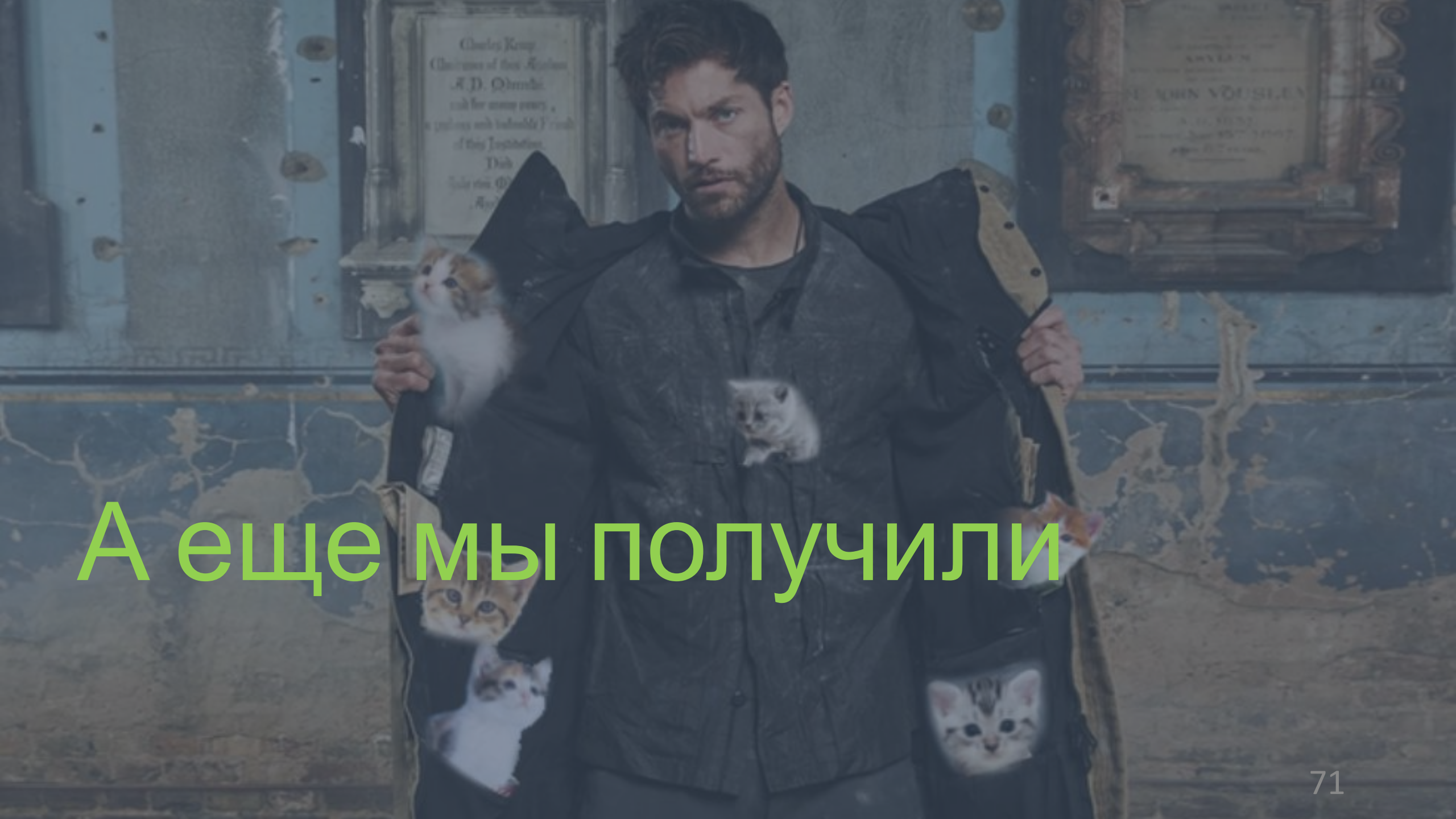
Проще сценарии

- Более декларативные сценарии
  - Такие сценарии проще поддерживат
  - Их проще параметризовать



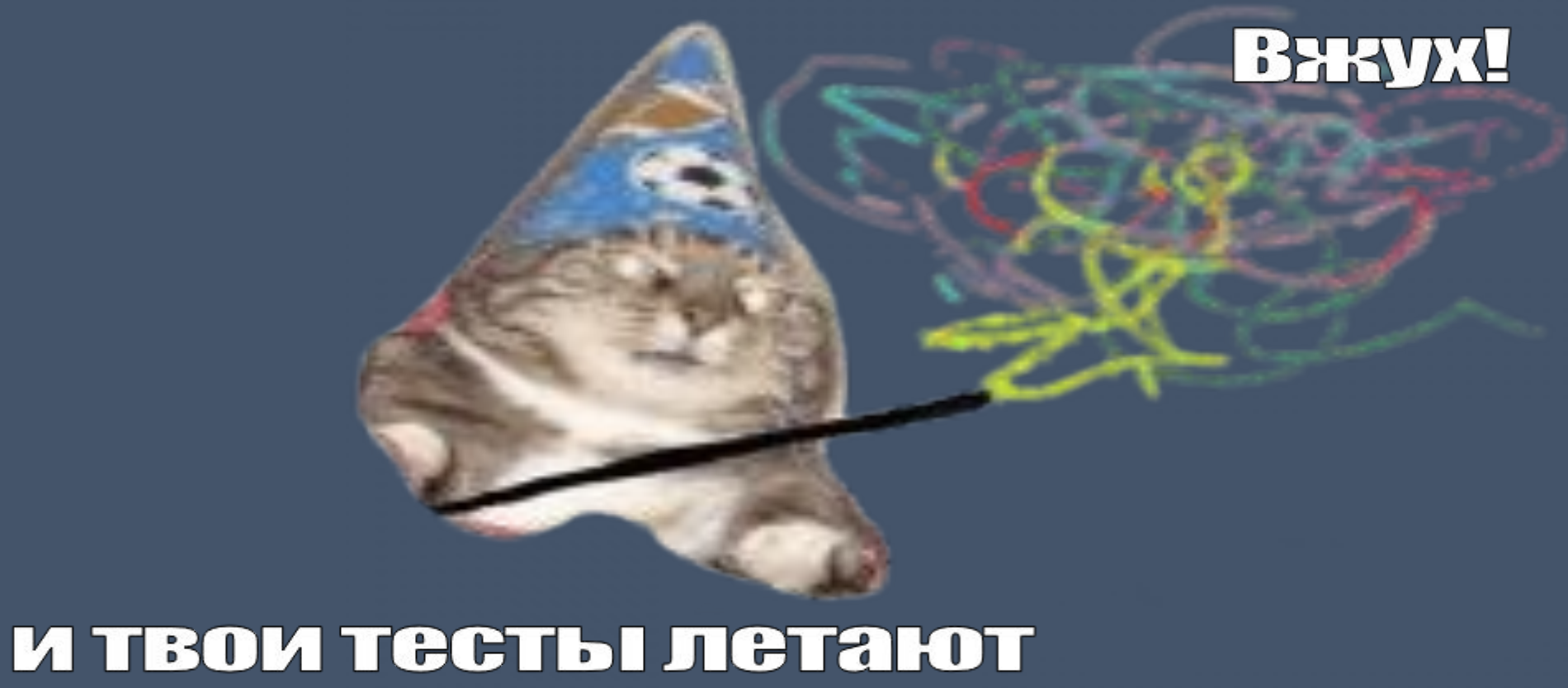
Интеграция продуктов

Исправили баги в старых тестах

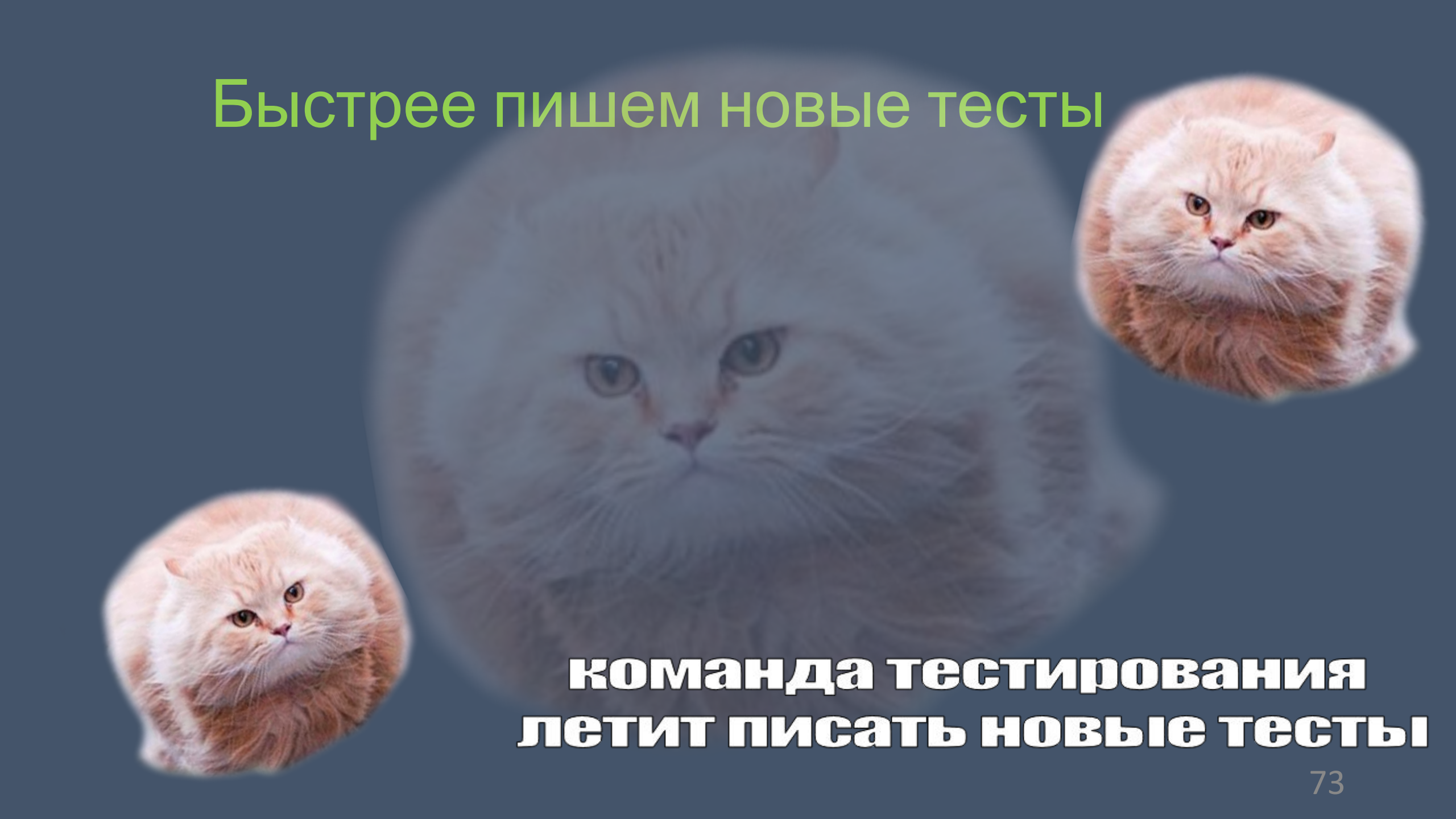


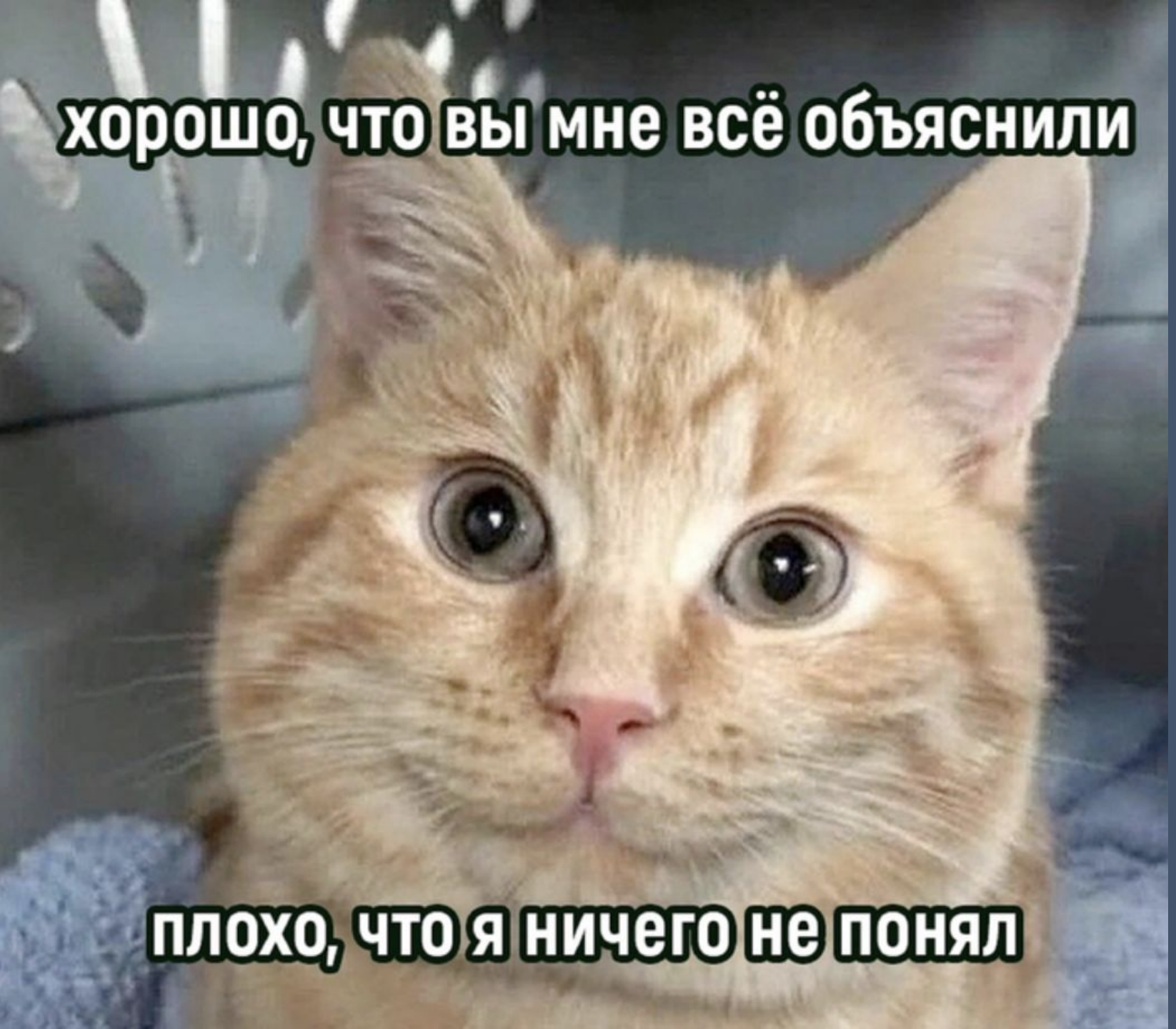
А еще мы получили

Тесты стали работать быстрее

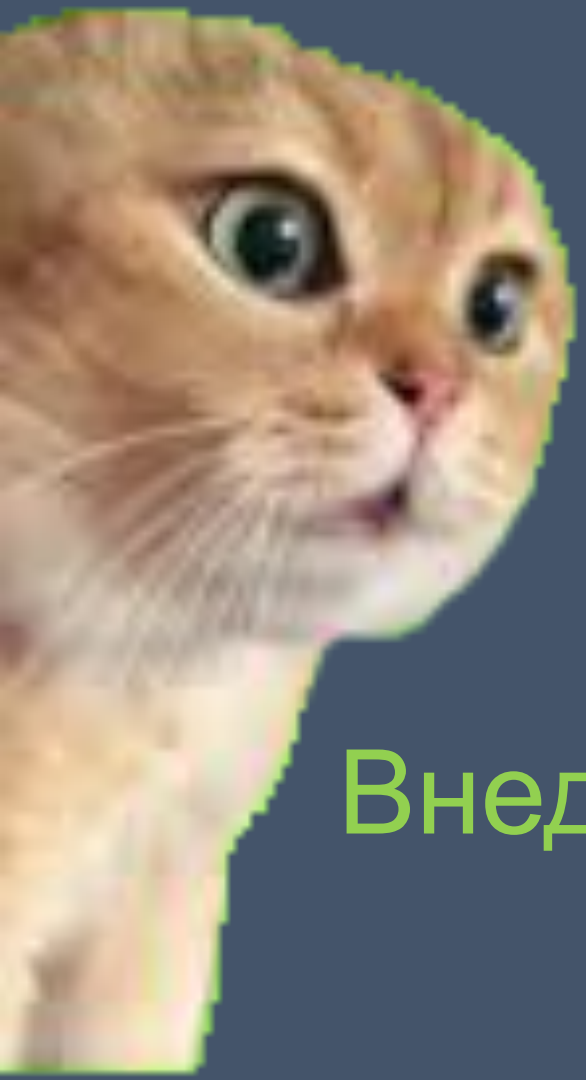


Быстрее пишем новые тесты

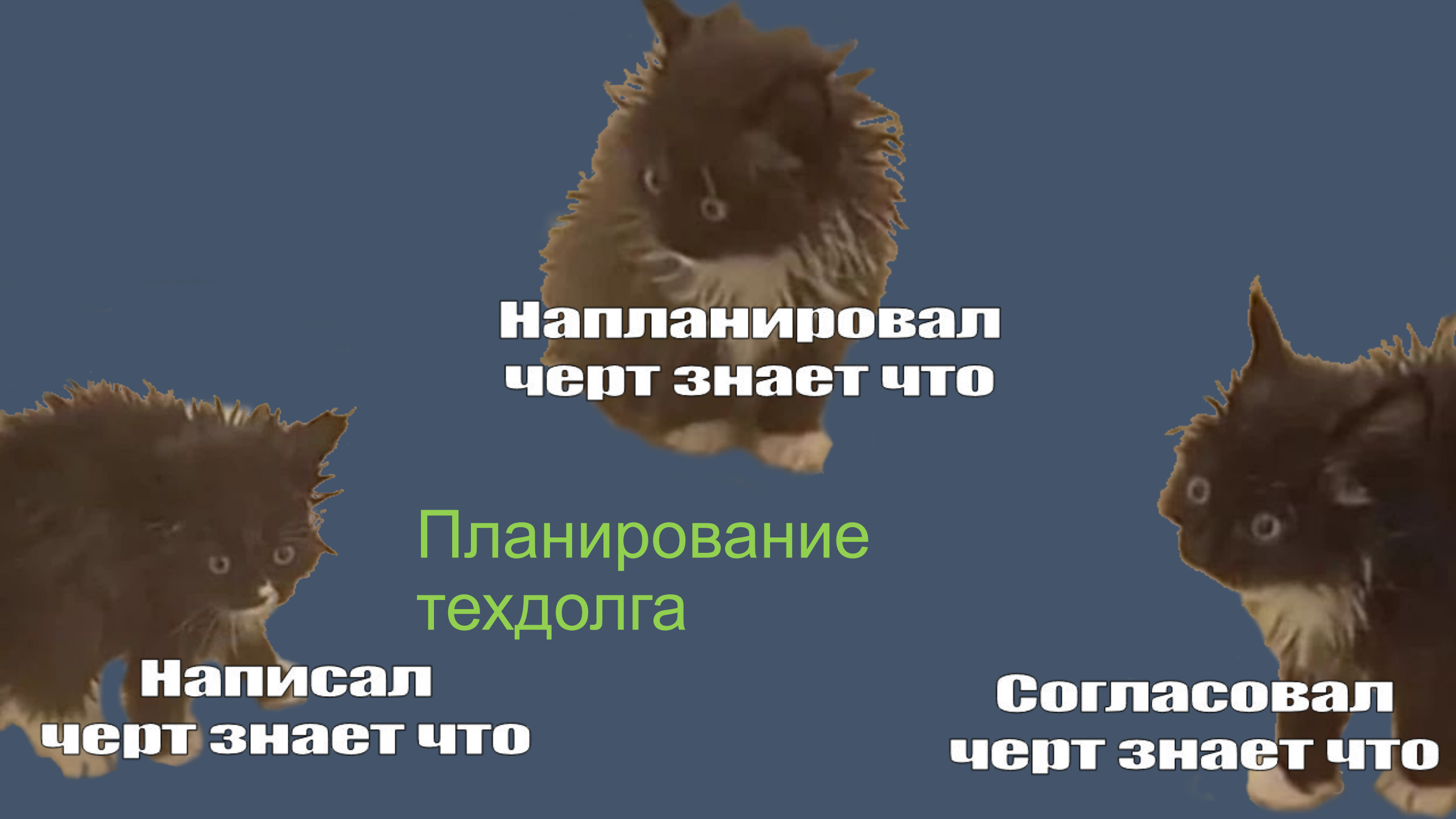




Проще работать
с результатами



Внедрили код-ревью в процесс



Планирование техдолга

Ссылки

- Слайд 16 – https://docs.pytest.org/en/7.1.x/how-to/writing_plugins.html
- Слайд 17 – https://pytest-with-eric.com/hooks/pytest-configure/#:~:text=At%20their%20core%2C%20Pytest%20hooks,the%20way%20you%20want%20it.
- Слайд 17 – https://docs.pytest.org/en/7.1.x/how-to/writing_hook_functions.html
- Слайд 18 – https://docs.pytest.org/en/7.1.x/how-to/fixtures.html

Еще ссылки

- Доклад про плагины с Heisenbug – https://www.youtube.com/watch?v=p3XUv8C8FKo
- Проект – https://github.com/AstralRomance/heisen_demo
- Некоторые смешные коты
  - https://t.me/arisetude
  - https://t.me/kisandkisa
- Я в ТГ: @astralromance

Спасибо за внимание