CI для CD

Антон Палий

- → Разработчик в Т-Банк
- → На самом деле Тимлид
- → Занимался SRE
- → 6 лет в Т-Банке

orangeatom

Что будет

- Из чего состоит наша инфраструктура
- Трагические истории
- Изменения, которые мы внесли
- Как вы можете применить у себя этот подход

Часть нулевая

Что имеем?

Что имеем?

- k8s as a service
- PCI DSS
- restrictions
- support

Еще есть

- postgres
- kafka
- redis
- vault
- elasticsearch(7.10.1)
- s3
- много http
- немного grpc
- и даже SOAP

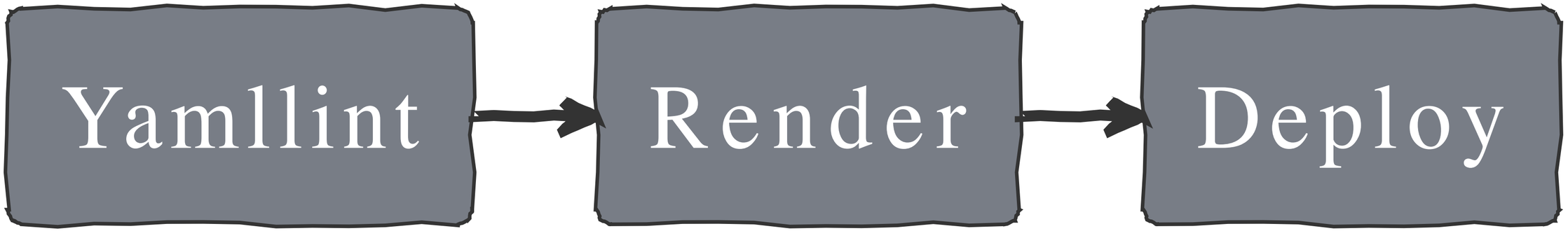
И сами деплоим

- deployment
- ingress
- service
- ~~operators~~

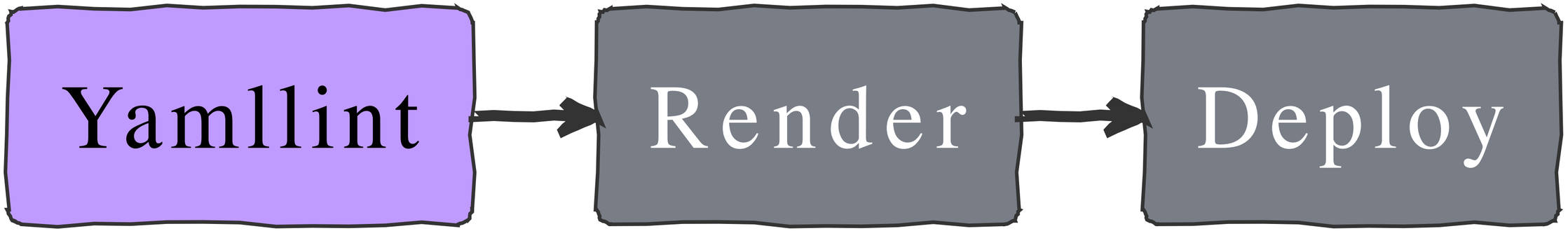
Деплой только из CD

helm update ...

Пайплайн

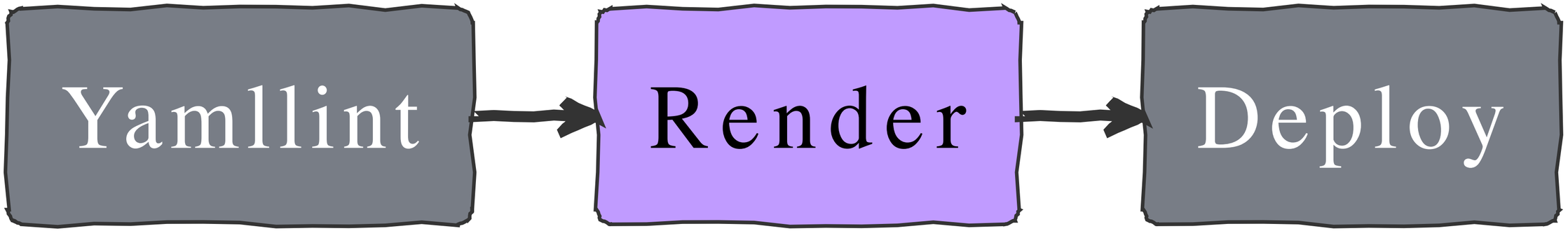


yamllint



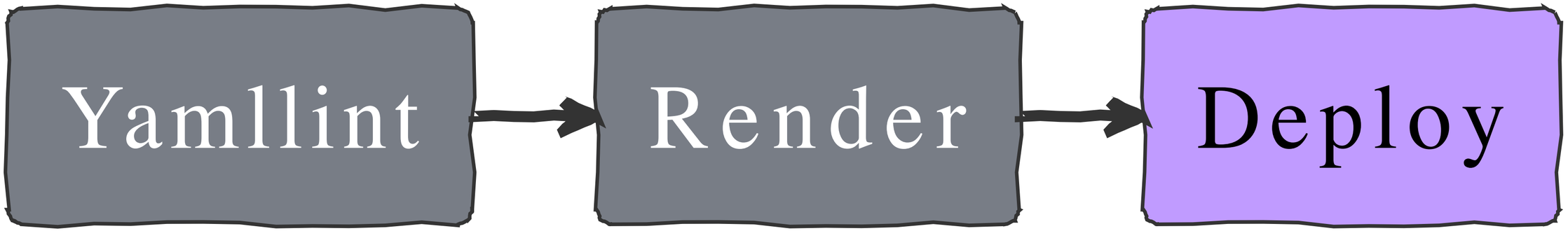
- Валидный yaml
- Простые ошибки форматирования

helm render



- Проверяет yaml на совместимость с k8s
- Валидирует наличие обязательных параметров
- Избавляет разработчика от понимания всей машинерии k8s

helm deploy

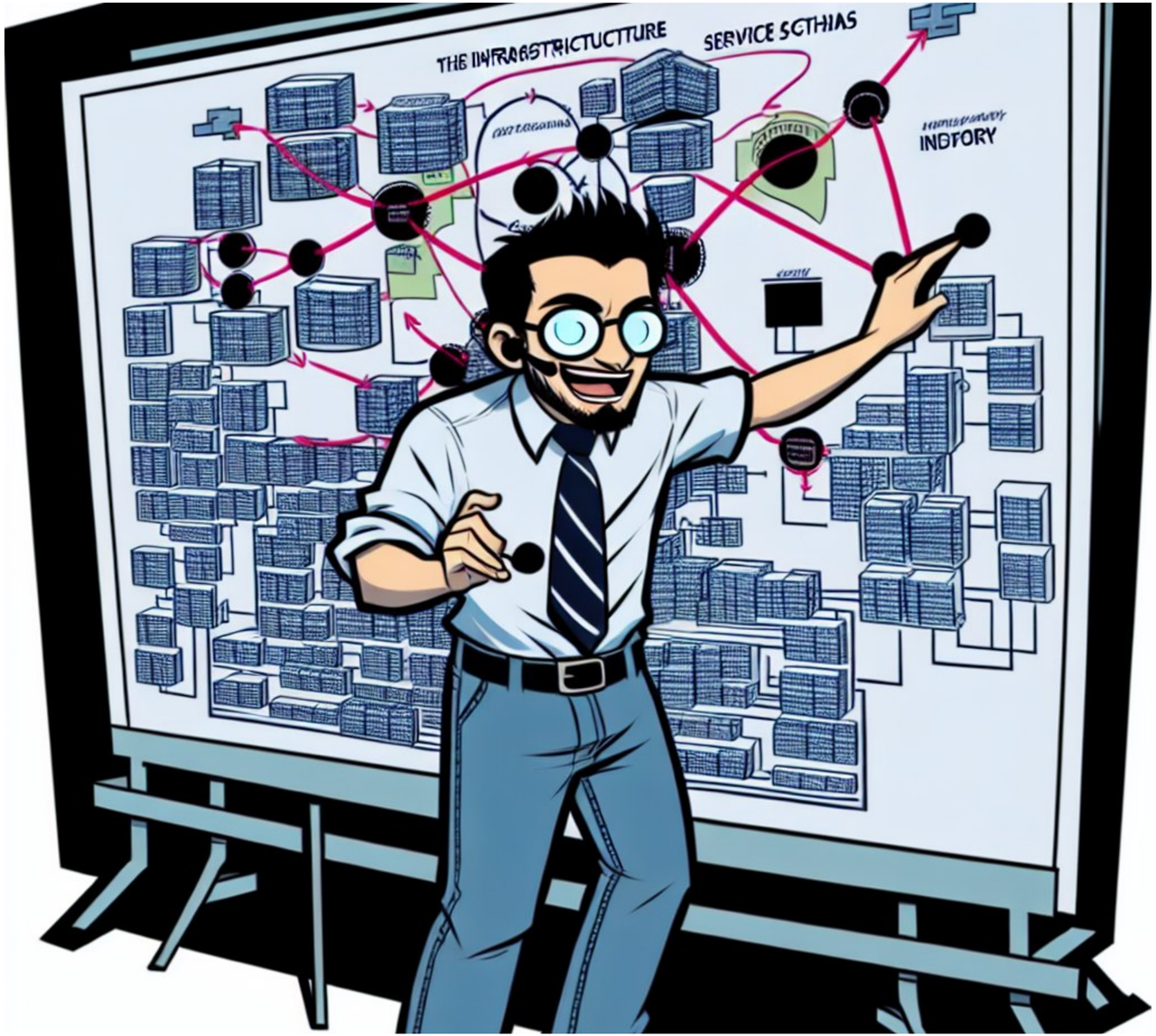


- Применяет yaml в k8s

Большой монорепозиторий для манифестов k8s

- 20+ неймспейсов
- 100+ деплойментов

Нюансов много, за всем уследить достаточно сложно



А за SLA следить нужно нам!

SLA - service level agreement

Иными словами, ответственность за прод - на команде разработки

цифры мерцают,
сложности взаимосвязей,
IT лабиринт

Часть первая
Вымышленный эпик фейл

Конечно же, такой ситуации никогда не было

Представьте, что у нас пожар!

Нужно обновить приложение, СРОЧНО!

Как это обычно происходит?

- Собираем через CI образ
- Тесты - обязательно
- Просим QA по-быстрому протестить, что все ок
- Берем тег
- Копируем, вставляем, коммитим, пушим
- Просим галки
- Накатываем (релиз конечно же)
- Вы великолепны!

На каком шаге ошибка получается дороже всего?

- Собираем через CI образ
- Тесты - обязательно
- Просим QA по-быстрому протестить, что все ок
- Берем тег
- Копируем, вставляем, коммитим, пушим
- Просим галки
- Накатываем (релиз конечно же)
- Вы великолепны!

На каком шаге ошибка получается дороже всего?

Конечно же, когда накатываем

А теперь проследуем за историей

Собрали образ

Прогнались тесты

Запушили в репозиторий

Параллельно убедили QA, что можно проверить фичу

МУУУУУУУЧИТЕЛЬНО дождались тестов

- от QA
- от CI

Параллельно с этим получили галки

Берем тег, копируем его

```
my-company-registry/awesome-team-awesome-service:tag-kek-image-{current_date}-ac0698b
```

Готовимся накатить

Добавляем в манифест

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
      - name: example-container
        image: my-company-registry/awesome-team-awesome-service:tag-kek-image-{current_date}-ac0698
        ports:
        - containerPort: 8080
...
```

Коммит в репозиторий с манифестами

```
git add . && git commit -m "HOT FEATURE, APPROVE NOW PLEASE"
```

Кидает другому разрабу, просим еще галку

Происходит CI/CD

KO/Fatality

SLA - вышел из чата

Откатываем

А что же не так?

Ошибка загрузки образа

Пристально посмотрим на yaml, который попал в прод

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
      - name: example-container
        image: my-company-registry/awesome-team-awesome-service:tag-kek-image-{current_date}-ac0698
        ports:
        - containerPort: 8080
...
```

А изначальный образ

```
my-company-registry/awesome-team-awesome-service:tag-kek-image-<current_date>-ac0698b
```

Один символ!

Сквозь пайплайн проплыл коммит
“Simple fix” Кукуева.
Весь продакшн поломала
Одна строчка *****.

Какие уроки мы можем извлечь?

- Автор реквеста оказался невнимательным
- Ревьюер(ы) не сходил(и) в реджистри
- CD сделал свою работу хорошо

Ошибка свершена,
Разработчик грустит лишь,
Постмортем пишет.

Постмортем

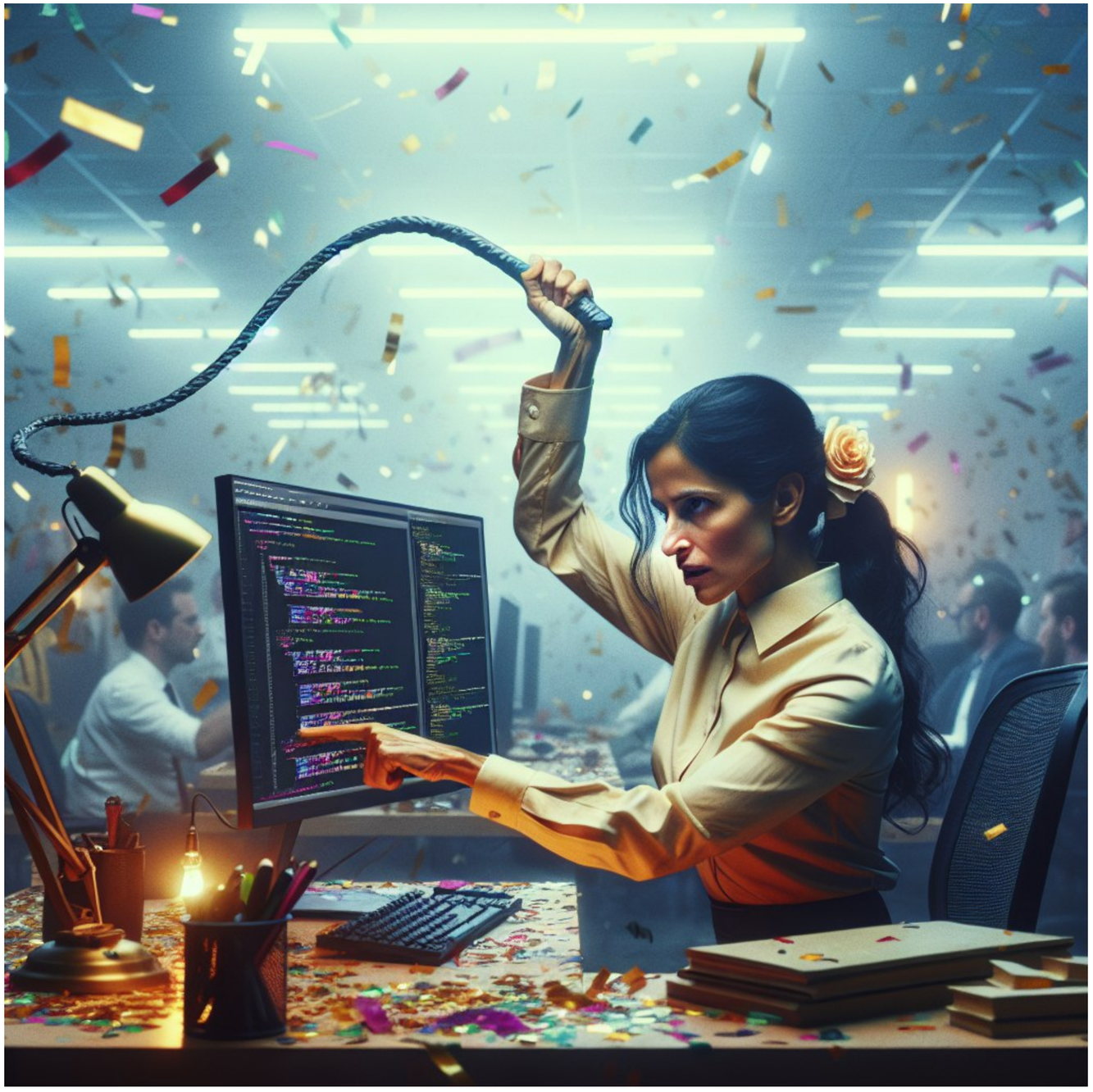
Детальный разбор сбоя, с описанием шагов, по избежанию подобных ситуаций

- Избегаем осуждения и делаем разбор конструктивно
- Указываем хронологию событий
- Считаем ущерб

Какие варианты у нас есть?

Пальчиком атата? Попросить быть внимательнее?

- Сказать, что прод ложить нельзя!
- Внимательнее ревью
- Внимательнее копировать тег
- Вообще работай внимательно
- Сказать, что накажешь, если будет ломать прод еще раз



Это не будет работать

Никакого осуждения!

Человек склонен ошибаться, и эти ошибки ведут к новым знаниям

The cost of failure is education. © - Devin Carraway

Какие знания мы можем получить тут?

helm, yamllint, webhooks - решают только часть проблем

А мы создаем больше!

Мы же разработчики/QA/SRE

Покрываем код тестами

Много тестов

unit/integration/e2e/etc

Почему бы и CD не покрыть тестами?

Взяли python

- У нас весь код на python, зачем придумывать что то новое?

Взяли pytest

- стандарт для тестирования в python
- мы очень любим тесты

Взяли либу pdk8s

- python библиотека для работы с k8s
- но мы взяли только модельки для сущностей k8s

Написали тест

Алгоритм этого теста

- собираем все деплойменты
- из этих деплойментов достаем имя образа
- ходим в реджистри со словами “а этот образ есть?”
- если образа нету, тест падает, мердж в мастер или релиз не разраешается
- если все образа есть, вы великолепны и можно делать дела

Компилируем манифесты

```
@pytest.fixture
def render_templates(template_directory: Path, manifests_dir: Path) -> list[str]:
    for cluster, namespace, chart in iter_charts(manifests_dir):
        output_dir = template_directory / cluster / namespace.name / chart.name
        output_dir.mkdir(exist_ok=True, parents=True)
        render_chart(cluster, chart, namespace, output_dir)
```

Компилируем манифесты

```
1 @pytest.fixture
2 def render_templates(template_directory: Path, manifests_dir: Path) -> list[str]:
3     for cluster, namespace, chart in iter_charts(manifests_dir):
4         output_dir = template_directory / cluster / namespace.name / chart.name
5         output_dir.mkdir(exist_ok=True, parents=True)
6         render_chart(cluster, chart, namespace, output_dir)
```

Компилируем манифесты

```python
@pytest.fixture
def render_templates(template_directory: Path, manifests_dir: Path) -> list[str]:
    for cluster, namespace, chart in iter_charts(manifests_dir):
        output_dir = template_directory / cluster / namespace.name / chart.name
        output_dir.mkdir(exist_ok=True, parents=True)
        render_chart(cluster, chart, namespace, output_dir)
```

Вытаскиваем деплойменты

```python
@pytest.fixture
def deployments() -> Iterator[K8sObject[Deployment]]:
    template_directory = Path(__file__).parent / 'render_result'
    generator = _charts_generator(template_directory)
    k8s_objects = k8s_object_generator(generator, template_directory)

    return filter_by_type(k8s_objects, Deployment)
```

Вытаскиваем деплойменты

```
1  @pytest.fixture
2  def deployments() -> Iterator[K8sObject[Deployment]]:
3      template_directory = Path(__file__).parent / 'render_result'
4      generator = _charts_generator(template_directory)
5      k8s_objects = k8s_object_generator(generator, template_directory)
6
7      return filter_by_type(k8s_objects, Deployment)
```

Вытаскиваем деплойменты

```
1  @pytest.fixture
2  def deployments() -> Iterator[K8sObject[Deployment]]:
3      template_directory = Path(__file__).parent / 'render_result'
4      generator = _charts_generator(template_directory)
5      k8s_objects = k8s_object_generator(generator, template_directory)
6
7      return filter_by_type(k8s_objects, Deployment)
```

Вытаскиваем деплойменты

```python
@pytest.fixture
def deployments() -> Iterator[K8sObject[Deployment]]:
    template_directory = Path(__file__).parent / 'render_result'
    generator = _charts_generator(template_directory)
    k8s_objects = k8s_object_generator(generator, template_directory)

    return filter_by_type(k8s_objects, Deployment)
```

Ходим реджистри и собираем образа

```python
def test_availability_images(deployments):
    checked = {}
    for deployment in deployments:
        image_name: str = deployment.spec.template.spec.containers[0].image
        res: bool = check_image_available(image_name)
        checked{image_name} = res
    assert False in checked.values(), get_unavaliable_images(checked)
```

Ходим реджистри и собираем образа

```
1 def test_availability_images(deployments):
2     checked = {}
3     for deployment in deployments:
4         image_name: str = deployment.spec.template.spec.containers[0].image
5         res: bool = check_image_available(image_name)
6         checked{image_name} = res
7     assert False in checked.values(), get_unavaliable_images(checked)
```

Ходим реджистри и собираем образа

```python
def test_availability_images(deployments):
    checked = {}
    for deployment in deployments:
        image_name: str = deployment.spec.template.spec.containers[0].image
        res: bool = check_image_available(image_name)
        checked{image_name} = res
    assert False in checked.values(), get_unavaliable_images(checked)
```

Ходим реджистри и собираем образа

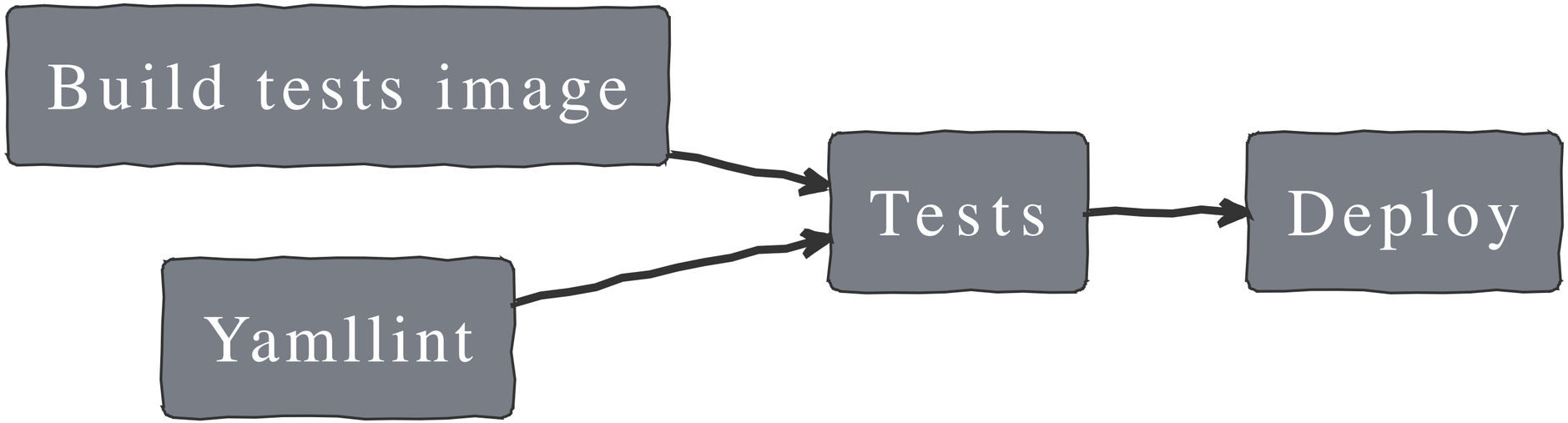
```
1 def test_availability_images(deployments):
2     checked = {}
3     for deployment in deployments:
4         image_name: str = deployment.spec.template.spec.containers[0].image
5         res: bool = check_image_available(image_name)
6         checked{image_name} = res
7     assert False in checked.values(), get_unavaliable_images(checked)
```

Ходим реджистри и собираем образа

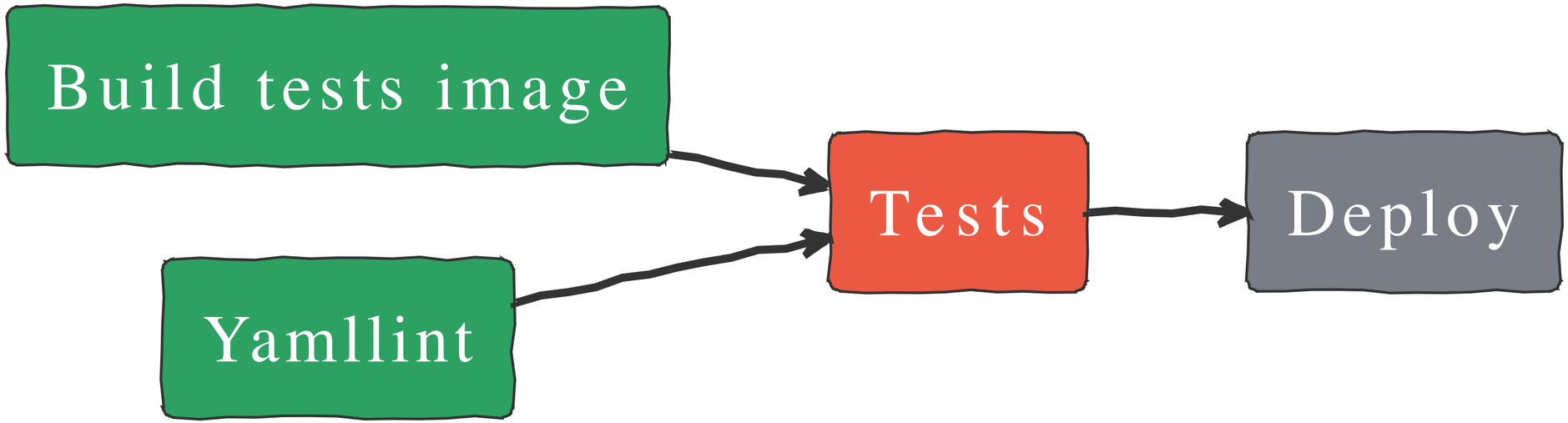
```python
def test_availability_images(deployments):
    checked = {}
    for deployment in deployments:
        image_name: str = deployment.spec.template.spec.containers[0].image
        res: bool = check_image_available(image_name)
        checked{image_name} = res
    assert False in checked.values(), get_unavaliable_images(checked)
```

Добавляем это дело в пайплайн

Новый пайплайн



Тесты падают, если образ не существует



И мы можем добавлять нужные нам дополнительные проверки

А зачем так сложно?

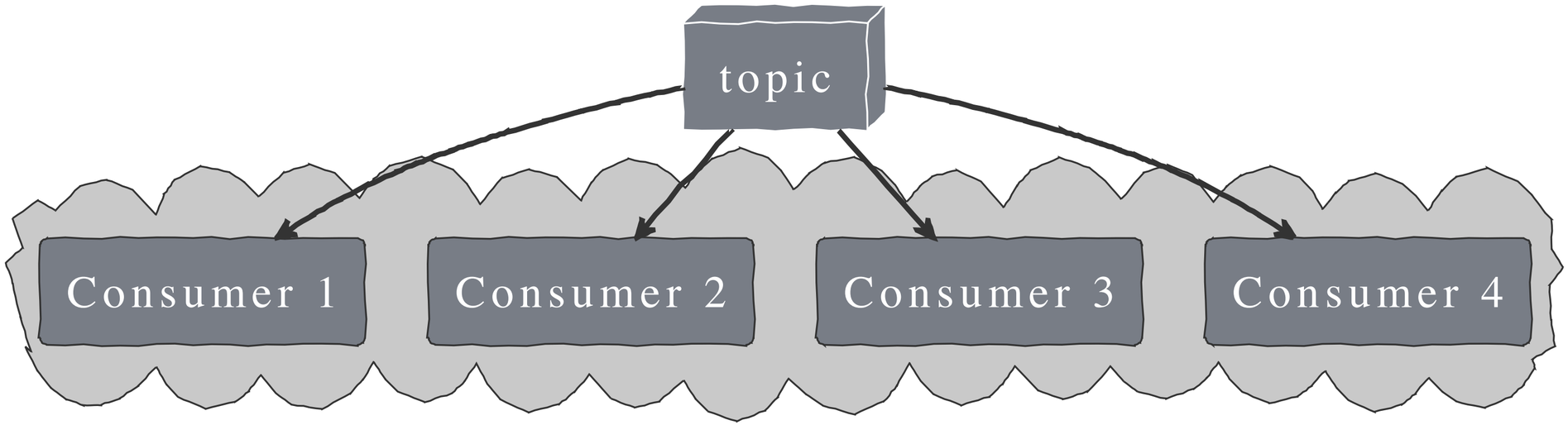
Для этого случая может и есть альтернативы

Посмотрим на более сложный случай

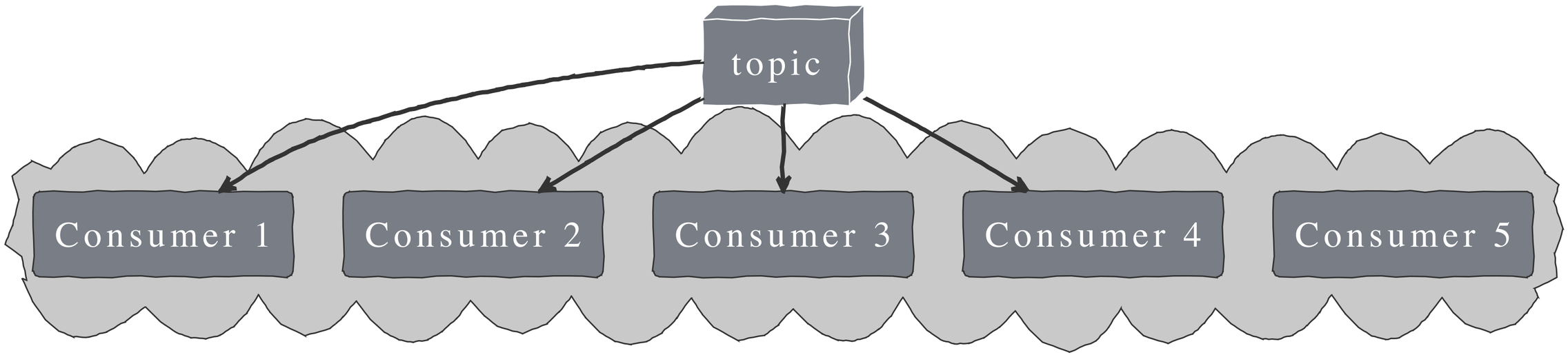
kafka

- 1 топик с 4 партициями
- чарт который слушает этот топик

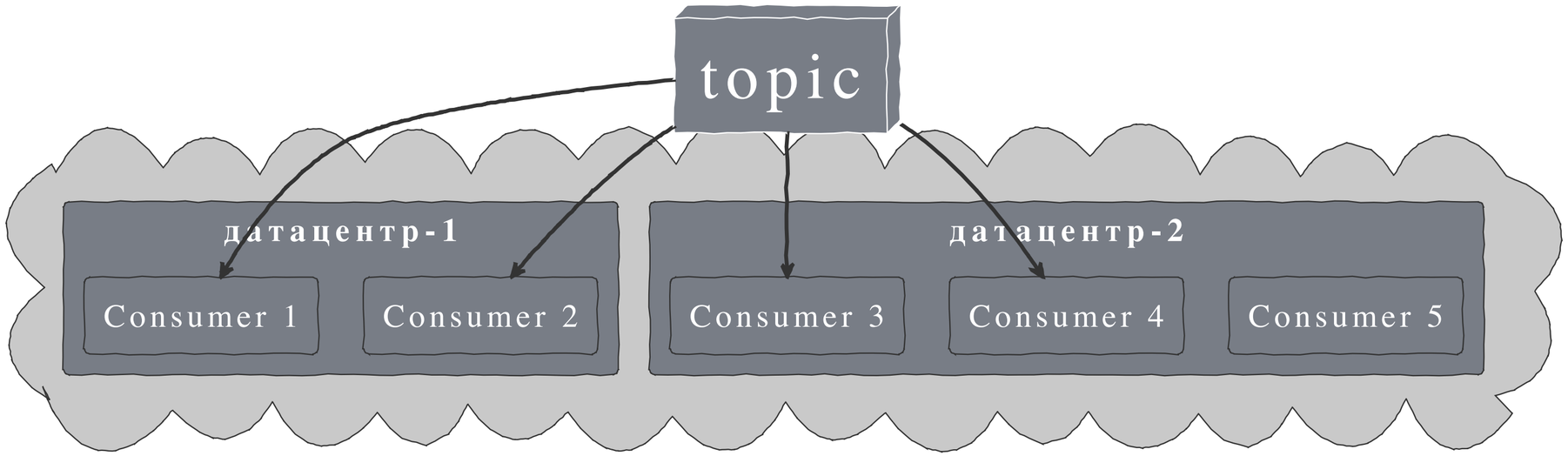
kafka



kafka


topic
Consumer 1
Consumer 2
Consumer 3
Consumer 4
Consumer 5

kafka


topic
датацентр-1
Consumer 1
Consumer 2
датацентр-2
Consumer 3
Consumer 4
Consumer 5

Нет необходимости поднимать консьюмер, если он ничего не будет слушать

Алгоритм проверки

- Собираем количество партиций в топиках
- Найти все консьюмеры
- Собираем их консьюмер группы
- Падаем в тесте, если количество консьюмеров больше чем партиций в топике

Собираем количество партиций в топиках

```python
class TopicInfo(NamedTuple):
    cluster: str
    name: str

@pytest.fixture
def topics_info(deployments):
    topics_info: dict[TopicInfo, int] = defaultdict(int)
    for deployment in deployments:
        topics_env = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')
        if topics_env is None:
            continue
        if TopicInfo(TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")) in topics_info:
            continue
        topics_info[TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")] = get_partition_count(
                cluster="kafka_kaas_tmsg_cluster", topic="topic_name"
            )
    return topics_info
```

Собираем количество партиций в топиках

```python
class TopicInfo(NamedTuple):
    cluster: str
    name: str

@pytest.fixture
def topics_info(deployments):
    topics_info: dict[TopicInfo, int] = defaultdict(int)
    for deployment in deployments:
        topics_env = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')
        if topics_env is None:
            continue
        if TopicInfo(TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")) in topics_info:
            continue
        topics_info[TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")] = get_partition_count(
                cluster="kafka_kaas_tmsg_cluster", topic="topic_name"
            )
    return topics_info
```

Собираем количество партиций в топиках

```python
class TopicInfo(NamedTuple):
    cluster: str
    name: str

@pytest.fixture
def topics_info(deployments):
    topics_info: dict[TopicInfo, int] = defaultdict(int)
    for deployment in deployments:
        topics_env = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')
        if topics_env is None:
            continue
        if TopicInfo(TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")) in topics_info:
            continue
        topics_info[TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")] = get_partition_count(
                cluster="kafka_kaas_tmsg_cluster", topic="topic_name"
            )
    return topics_info
```

Собираем количество партиций в топиках

```python
class TopicInfo(NamedTuple):
    cluster: str
    name: str

@pytest.fixture
def topics_info(deployments):
    topics_info: dict[TopicInfo, int] = defaultdict(int)
    for deployment in deployments:
        topics_env = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')
        if topics_env is None:
            continue
        if TopicInfo(TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")) in topics_info:
            continue
        topics_info[TopicInfo(cluster="кластер из деплоймента", name="топик из деплоймента")] = get_partition_count(
                cluster="kafka_kaas_tmsg_cluster", topic="topic_name"
            )
    return topics_info
```

Сейчас будет немного больно

Ищем все консьюмеры

```python
def test_partition_is_divided_by_consumers(
        self, deployments: Iterator[Deployment], topics_info: dict[TopicInfo, int]
    ):
    consumers = defaultdict(lambda: defaultdict(int))
    for *_, chart, deployment in deployments:
        envs = deployment.spec.template.spec.containers[0].env
        topics = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')

        if topics is None:
            continue

        consumer_group_id = get_env_value_or_default(
            get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_GROUP_ID')
        )

        bootstrap_servers_env = get_env_by_name_suffix_or_none(envs, '_BOOTSTRAP_SERVERS')
        cluster_name = get_cluster_name_from_broker(
            json.loads(get_env_value_or_default(bootstrap_servers_env, '[]'))
        )

        assert cluster_name is not None

        topic_info = TopicInfo(cluster=cluster_name, name=topic)
        consumers[topic_info][consumer_group_id] += get_consumers_count(deployment)
```

Ищем все консьюмеры

```python
def test_partition_is_divided_by_consumers(
        self, deployments: Iterator[Deployment], topics_info: dict[TopicInfo, int]
    ):
    consumers = defaultdict(lambda: defaultdict(int))
    for *_, chart, deployment in deployments:
        envs = deployment.spec.template.spec.containers[0].env
        topics = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')

        if topics is None:
            continue

        consumer_group_id = get_env_value_or_default(
            get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_GROUP_ID')
        )

        bootstrap_servers_env = get_env_by_name_suffix_or_none(envs, '_BOOTSTRAP_SERVERS')
        cluster_name = get_cluster_name_from_broker(
            json.loads(get_env_value_or_default(bootstrap_servers_env, '[]'))
        )

        assert cluster_name is not None

        topic_info = TopicInfo(cluster=cluster_name, name=topic)
        consumers[topic_info][consumer_group_id] += get_consumers_count(deployment)
```

Ищем все консьюмеры

```python
def test_partition_is_divided_by_consumers(
        self, deployments: Iterator[Deployment], topics_info: dict[TopicInfo, int]
    ):
    consumers = defaultdict(lambda: defaultdict(int))
    for *_, chart, deployment in deployments:
        envs = deployment.spec.template.spec.containers[0].env
        topics = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')

        if topics is None:
            continue

        consumer_group_id = get_env_value_or_default(
            get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_GROUP_ID')
        )

        bootstrap_servers_env = get_env_by_name_suffix_or_none(envs, '_BOOTSTRAP_SERVERS')
        cluster_name = get_cluster_name_from_broker(
            json.loads(get_env_value_or_default(bootstrap_servers_env, '[]'))
        )

        assert cluster_name is not None

        topic_info = TopicInfo(cluster=cluster_name, name=topic)
        consumers[topic_info][consumer_group_id] += get_consumers_count(deployment)
```

Ищем все консьюмеры

```
def test_partition_is_divided_by_consumers(
        self, deployments: Iterator[Deployment], topics_info: dict[TopicInfo, int]
    ):
    consumers = defaultdict(lambda: defaultdict(int))
    for *_, chart, deployment in deployments:
        envs = deployment.spec.template.spec.containers[0].env
        topics = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')

        if topics is None:
            continue

        consumer_group_id = get_env_value_or_default(
            get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_GROUP_ID')
        )

        bootstrap_servers_env = get_env_by_name_suffix_or_none(envs, '_BOOTSTRAP_SERVERS')
        cluster_name = get_cluster_name_from_broker(
            json.loads(get_env_value_or_default(bootstrap_servers_env, '[]'))
        )

        assert cluster_name is not None

        topic_info = TopicInfo(cluster=cluster_name, name=topic)
        consumers[topic_info][consumer_group_id] += get_consumers_count(deployment)
```

Ищем все консьюмеры

```python
def test_partition_is_divided_by_consumers(
        self, deployments: Iterator[Deployment], topics_info: dict[TopicInfo, int]
    ):
    consumers = defaultdict(lambda: defaultdict(int))
    for *_, chart, deployment in deployments:
        envs = deployment.spec.template.spec.containers[0].env
        topics = get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_TOPICS')

        if topics is None:
            continue

        consumer_group_id = get_env_value_or_default(
            get_env_by_name_suffix_or_none(envs, 'KAFKA_CONSUMER_GROUP_ID')
        )

        bootstrap_servers_env = get_env_by_name_suffix_or_none(envs, '_BOOTSTRAP_SERVERS')
        cluster_name = get_cluster_name_from_broker(
            json.loads(get_env_value_or_default(bootstrap_servers_env, '[]'))
        )

        assert cluster_name is not None

        topic_info = TopicInfo(cluster=cluster_name, name=topic)
        consumers[topic_info][consumer_group_id] += get_consumers_count(deployment)
```

И проверяем, что нет плохих конфигов

```
...
bad_topics = []
for topic, partition_count in topics_info.items():
    for consumer_group_id, consumers_count in consumers[topic].items():
        if partition_count % consumers_count != 0:
            bad_topics.append((topic, consumer_group_id, partition_count, consumers_count))

assert not bad_topics
```

Теперь не будет бесполезных консьюмеров

Дополнительные проверки, которые можно добавить

- Все серверы кафки прописаны в конфиге
- У консьюмера всегда есть его топик
- Есть доступ до брокеров

У нас есть также тесты не только на это

- Валидация переменных окружения из vault
- Количество соединений с postgres
- Валидация, что для приложения проставлены все обязательные энвы
- Наличие всех сетевых доступов

Тесты пишем, что они не привязаны к конкретному приложению

Общие библиотеки создают неявное соглашение и именование переменных окружения имеет фиксированные суффиксы

Тесты пишут все

- QA
- SRE
- SDE

Такой подход очень редко где либо применяется

Как вы можете начать тестировать CD?

Как вы можете начать тестировать CD?

- Взять ваш фреймворк для тестирования
- Разобраться в том, что чаще всего болит
- Написать на это дело тест(ы)

Итоги

Итоги

Автоматизированные проверки помогают поддерживать стабильность приложений

Итоги

Выбираемые тесты граничат лишь с вашей фантазией

Итоги

Неочевидные тесты решают много проблем

Спасибо за внимание!

Антон Палий

orangeatom