



DevOps

РТЛабс

2026

Проверки Helm-чартов в CI/CD:

где заканчивается линтинг и начинается
эксплуатационная валидация

Харчук Михаил Викторович

Руководитель направления DevOps,
АО «РТ ЛАБС»





Михаил Харчук

Руководитель направления DevOps

Развиваю CI/CD и автоматизирую
поставку приложений в Kubernetes

РТЛабс

Цифровые государственные сервисы:
Госуслуги и системы электронного правительства

План доклада

- 1 Проверка на уровне YAML**
- 2 Проверка изменения конфигурации**
- 3 Сетевые зависимости приложения**
- 4 Готовые инструменты и результаты**

Что лежит внутри конфигурации

Содержимое файла

Проверка на уровне YAML

Проверка изменения конфигурации

Сетевые зависимости

После релиза приложение не запускается

Фрагмент лога приложения

```
2026-09-15T10:42:18.351Z  INFO  loading configuration
2026-09-15T10:42:18.358Z  INFO  source=/etc/demo/rules.json
2026-09-15T10:42:18.359Z  ERROR  configuration parsing failed
```

JSONDecodeError: Expecting ',' delimiter

line 3, column 5

```
    "rps": 100
```

```
    "rpsPerIp": 10
```

```
    ^
```

YAML же валидный!



а JSON ?



Ошибка JSON переехала в CI

Раньше

После deploy

`container log`

Приложение не стартует



Теперь

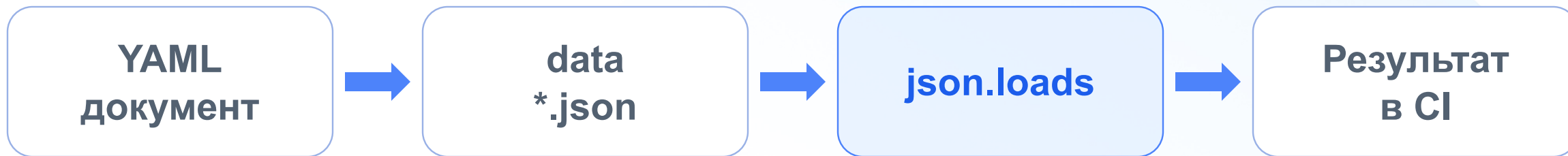
До deploy

`tpl/demo-config.yml`

`data.rules.json` · line 3,
column 5

Та же ошибка обнаруживается до установки приложения

Как находим JSON



Разбор вложенного JSON

Фрагмент проверки

```
for doc in yaml.safe_load_all(stream):  
    if not isinstance(doc, dict): continue  
    for key, value in doc.get('data', {}).items():  
        if key.endswith('.json'):  
            json.loads(value.strip())
```

В отчёте нужен путь до ошибки

GitLab / demo-service / Jobs / json-check

Пример

Failed json-check

```
$ pytest /test -n 7 --tb=no --junitxml=unittests_result.xml
[gw1] FAILED test_json.py::test_config_json[rules.json]
source: tpl/demo-config.yaml key: data["rules.json"]
JSONDecodeError: Expecting ',' delimiter: line 3 column 5
JSON fragment:
  2 |      "rps": 100
  3 |      "rpsPerIp": 10
    |      ^
FAIL: invalid JSON in ConfigMap demo-config
===== 1 failed =====
Uploading artifacts for failed job
unittests_result.xml: found 1 matching artifact file
Uploading artifacts as "junit"... 201 Created
ERROR: Job failed: exit code 1
```

Что гарантирует JSON-проверка

Проверяет сейчас

Запятые, кавычки и скобки

Escape-последовательности

Документ разбирается целиком

Не следует из успешного parse

Обязательные поля

Типы и диапазоны значений

Бизнес-правила приложения

Что меняется в конфигурации

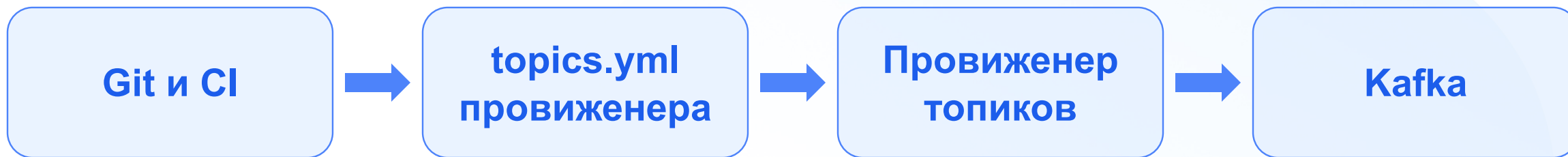
Ветка и diff

Проверка на уровне YAML

Проверка изменения конфигурации

Сетевые зависимости

Конфигурация топиков в Git



Изменение файла проходит review и CI

Что хранится в topics.yml

Обезличенный фрагмент конфигурации провиженера
topics:

- **name: svc-feature.redis-sync.pipe.priv.v1**

partitions:

prod: 4

config:

segment.bytes: 524288000

retention.ms: 300000

description:

producer: [svc-feature]

consumer: [svc-feature]

owner: <owner-group>

Проверка имени топика

Соглашение об именовании

service.feature.event.priv.v1

service

feature

event

priv

v1

Сервис

Назначение

Тип сообщения

Видимость

Версия

CI проверяет имена в изменённых topics.yml

Обычное изменение параметров

Фрагмент diff в release/*

partitions:

- prod: 4

+ **prod: 6**

config:

- retention.ms: 300000

+ **retention.ms: 600000**

Diff показывает направление изменения

Среди параметров появился `deleted: true`

Другой фрагмент того же `release/*`

`topics:`

- `name: svc-feature.redis-sync.pipe.priv.v1`

`partitions:`

`prod: 4`

+ `deleted: true`

Для системы управления топиками это команда удаления

Откат приложения не восстановил данные

Версия приложения: предыдущая

Топик и сообщения: удалены

Корректное значение может нарушать правило релиза

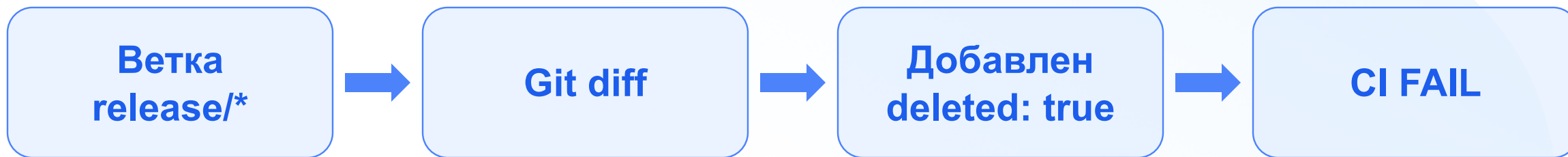
deleted: true — корректное значение YAML

В release/* удаление запрещено

Для решения нужны ветка и diff

Правило относится к процессу выпуска

Проверяем ветку и diff



CI останавливает изменение топика

GitLab / demo-service / Jobs / topics-check

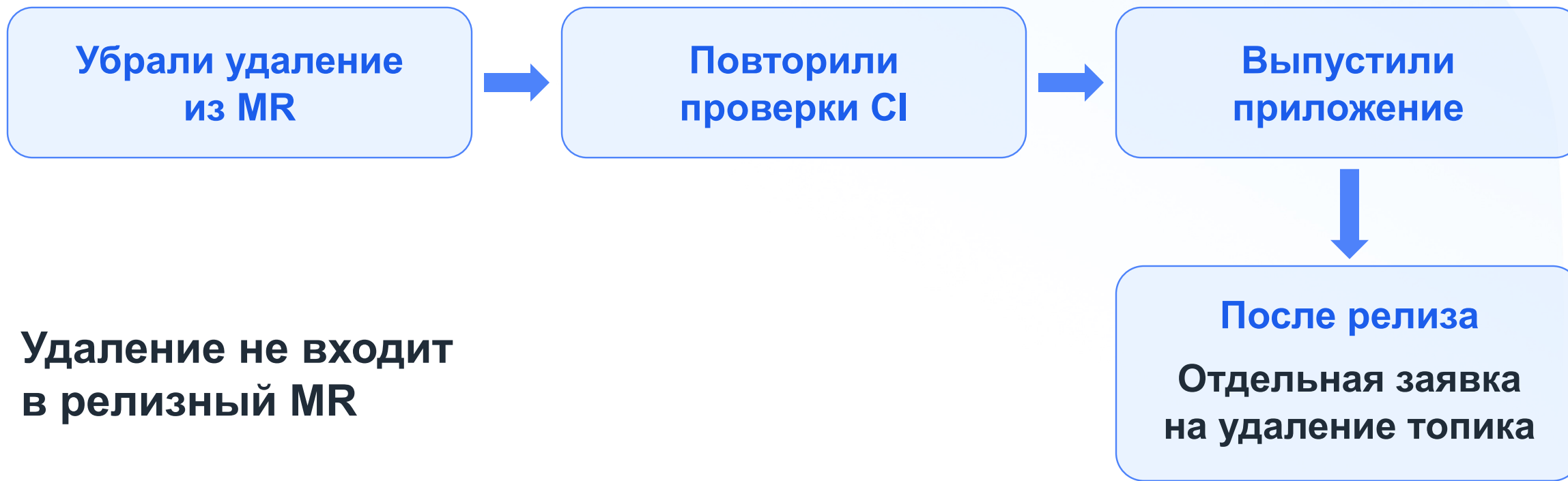
Пример

Failed topics-check

```
$ pytest /test -n 7 --tb=no --junitxml=unittests_result.xml
[gw0] FAILED test_topics.py::test_release_topic_changes
branch: release/2026.09
source: config/topics.yml   topic: service.feature.event.priv.v1
diff:
- deleted: false
+ deleted: true
FAIL: topic deletion is not allowed in a release branch
Action: remove deletion from this MR; use a separate change.
===== 1 failed =====
Uploading artifacts for failed job
unittests_result.xml: found 1 matching artifact file
Uploading artifacts as "junit"... 201 Created
ERROR: Job failed: exit code 1
```

Удаление выполняем отдельно, после релиза

CI остановил MR: добавлено `deleted: true`



Удаление не входит в релизный MR

Граница ответственности правила

Проверка получает	Проверка решает	Остаётся процедуре
Ветка и diff	Блокировать удаление в release/*	Согласовать операцию
Новое значение	Объяснить причину отказа	Выбрать окно работ
Контекст релиза	Направить в отдельный процесс	Определить восстановление

Где работает конфигурация

Конфигурация и среда

Проверка на уровне YAML

Проверка изменения конфигурации

Сетевые зависимости

Адреса в конфигурации приложения

Фрагмент ConfigMap

data:

API_URL: <https://api.example.net/v2>

LEGACY_URL: <http://legacy.example.net/status>

STORAGE_ADDR: 192.0.2.20:9042

CACHE_ADDR: cache.example.net:6379

OTEL_ENDPOINT: <http://otel-collector:4317>

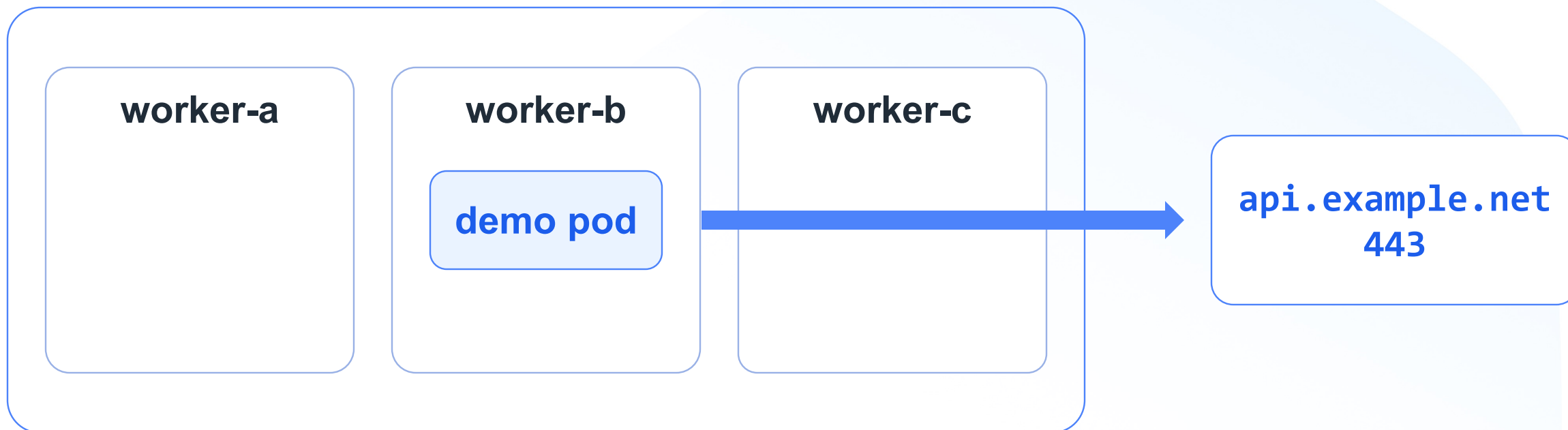
CONNECT_TIMEOUT: 7s

RETRIES: "3"

FEATURE_FLAG: "true"

Сетевой путь приложения

Kubernetes-кластер



Scheduler может разместить pod на любой подходящей worker-ноде

Приложение ждёт зависимость

Фрагмент лога

2026-09-15T10:42:18.351Z ERROR dependency check failed

service=demo

target=api.example.net:443

attempt=3

timeout_ms=7000

error="dial tcp 192.0.2.10:443: i/o timeout"

state=waiting_for_dependency

Ошибка относится к соединению, не к YAML

Недоступная зависимость меняет состояние pod

`api.example.net:443`

`dial tcp: i/o timeout`

Во время запуска

startupProbe FAIL

Порог ошибок превышен

Контейнер перезапускается

После запуска

readinessProbe FAIL

Pod получает Ready=false

Service не отправляет трафик

Одна сетевая проблема даёт разные симптомы

Ручной curl не покрывает конфигурацию целиком

Ноутбук
через VPN
→
api.example.net:443
200 OK

CI runner
маршрут CI
→
api.example.net:443
timeout

Стенд
одна точка
→
api.example.net:443
200 OK

Даже на стенде можно
проверить не всё

✓ **API:443**

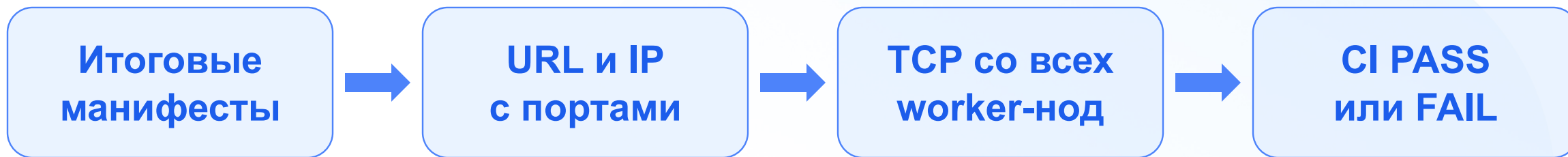
? Cache:6379

? Storage:9042

? Новый адрес

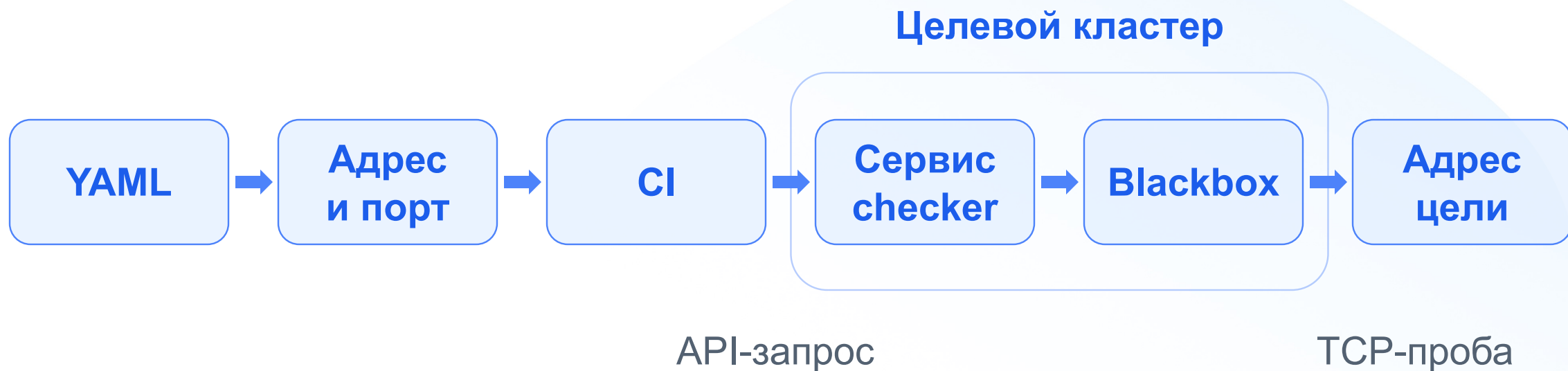
Ручная проверка подтверждает выбранный маршрут и выбранную цель

Проверяем маршрут до deploy



Одна недоступная нода блокирует CI

Маршрут сетевой проверки



Какие адреса находит проверка

Запись в конфигурации

`https://api.example.net/v2`

`http://legacy.example.net/status`

`https://api.example.net:8443/v1`

`cache.example.net:6379`

`192.0.2.20:9042`

`kafka-a:9092,kafka-b:9092`

`jdbc:postgresql://db.example.net:5432/app`

`redis://cache.example.net:6379/0`

Цель TCP-проверки

`api.example.net:443`

`legacy.example.net:80`

`api.example.net:8443`

`cache.example.net:6379`

`192.0.2.20:9042`

две цели :9092

`db.example.net:5432`

`cache.example.net:6379`

Источник: итоговые манифесты после helm template

Нормализация цели для TSP

Вход	Хост	Порт	Путь
https://api.example.net/v2	api.example.net	443	не участвует
http://legacy.example.net	legacy.example.net	80	нет
https://api.example.net:8443/v1	api.example.net	8443	не участвует
192.0.2.20:9042	192.0.2.20	9042	—

Запрос к checker из CI

Пример запроса

POST /v1/checktcp

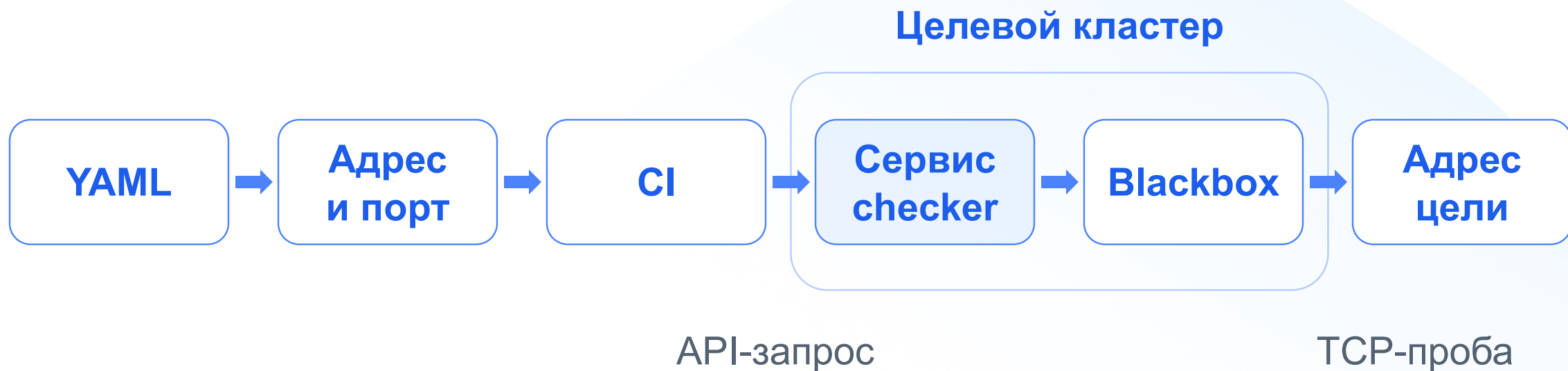
Authorization: Basic <CI credentials>

X-API-Key: <CI key>

```
{  
  "address": "api.example.net",  
  "port": 8443  
}
```

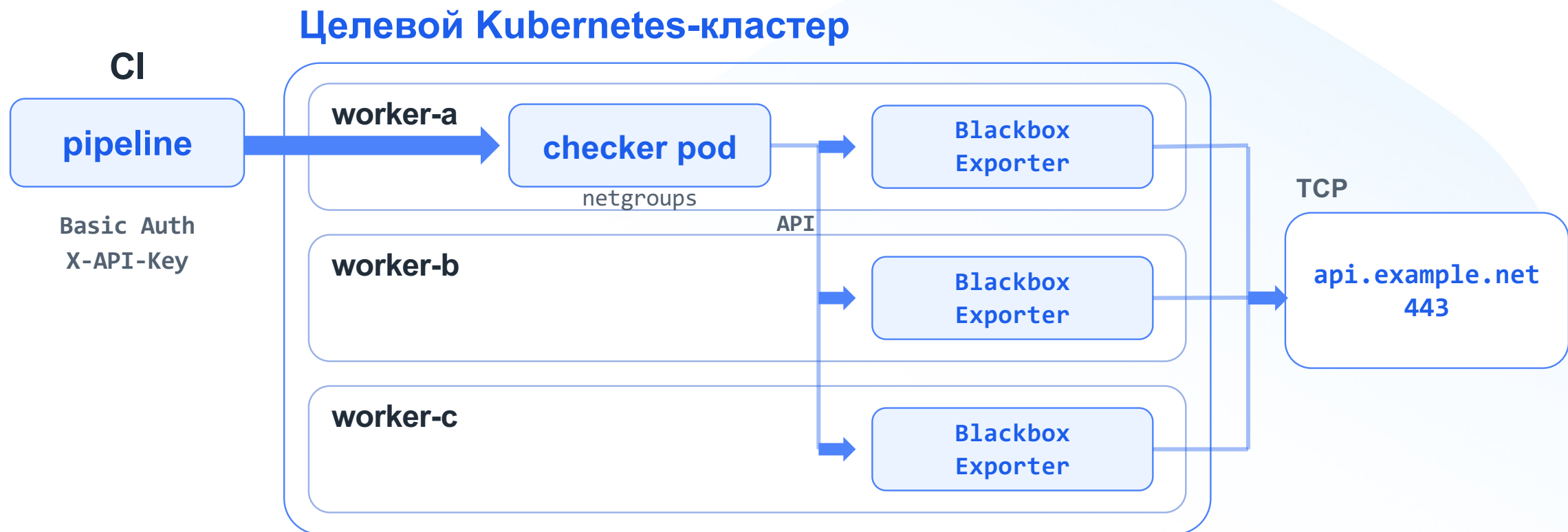
Basic Auth и API key, доступ из CI, базовый rate limit

Выбор worker-нод



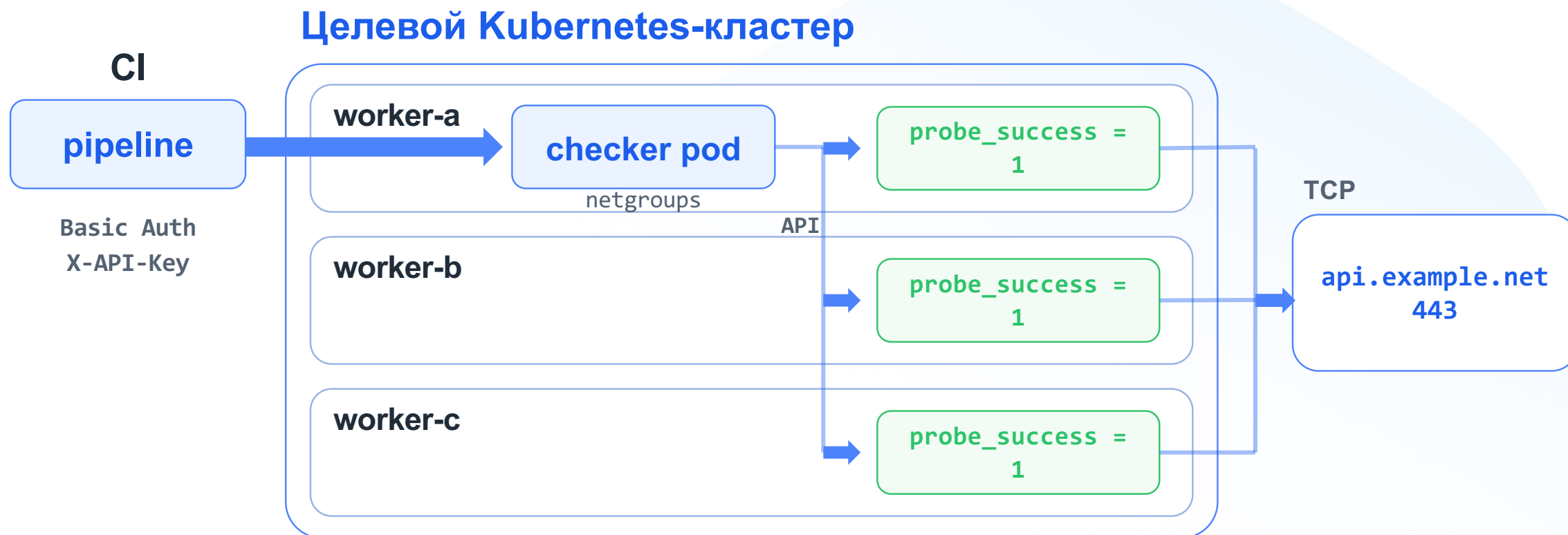
netgroups задаёт экспортеры выбранного стенда

Checker и Blackbox внутри кластера



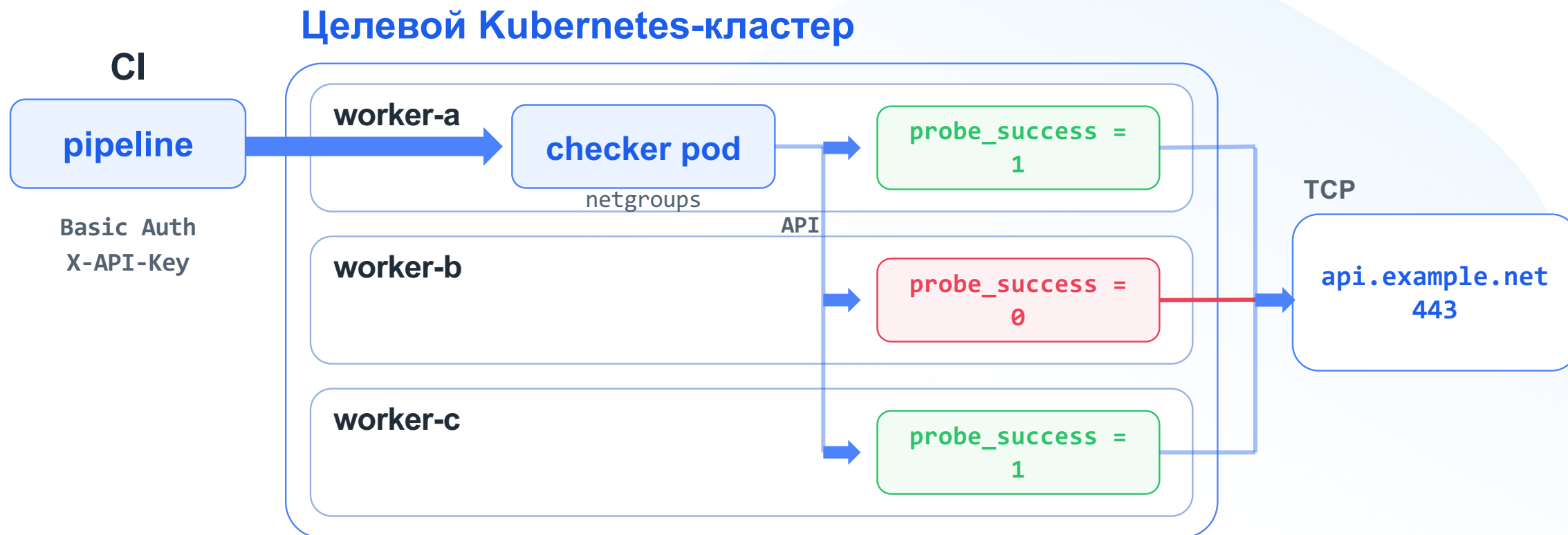
netgroups задаёт список exporter выбранного стенда

Успех нужен со всех worker-нод



CI: PASS только при успехе на каждой ноде

Одна недоступная нода блокирует CI



CI: FAIL. Pod может оказаться на worker-b

Ошибка CI ведёт к месту в конфигурации

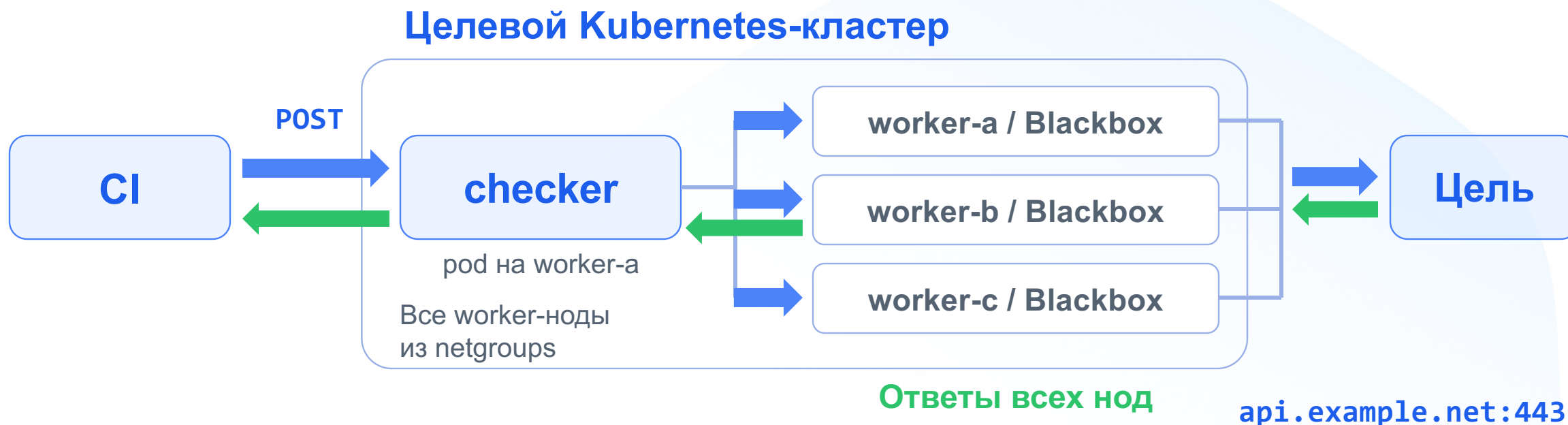
GitLab / demo-service / Jobs / connect-check

Пример

Failed connect-check

```
$ pytest /test -n 7 --tb=no --junitxml=unittests_result.xml
[gw2] FAILED test_connect.py::test_target[api.example.net:443]
source: tpl/demo-config.yaml:2 key: data.API URL
url: https://api.example.net/v2
target: api.example.net:443 HTTP 400 after 4 attempts
worker-a connected
worker-b disconnected (TCP probe failed)
worker-c connected
FAIL: target must be reachable from every worker node
PASS cache.example.net:6379: worker-a=OK worker-b=OK worker-c=OK
===== 1 failed, 1 passed =====
Uploading artifacts for failed job
unittests_result.xml: found 1 matching artifact file
Uploading artifacts as "junit"... 201 Created
ERROR: Job failed: exit code 1
```

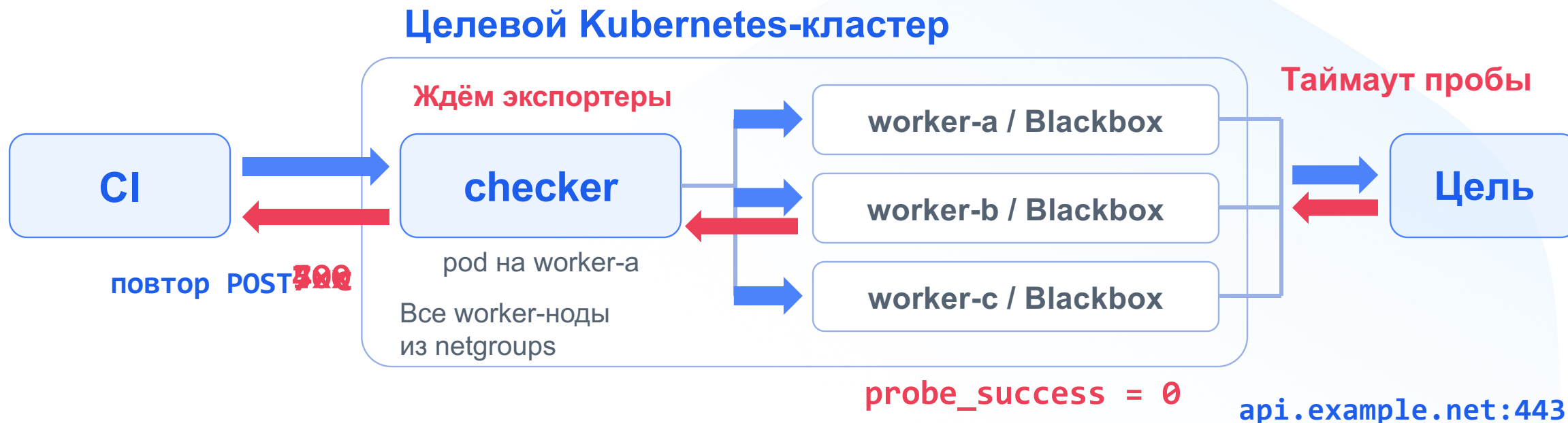
Как проходит одна проверка



2. Создание Blackbox-проверки и результат

Вот пример того, как выглядит результат проверки. HTTP 400.

Ошибки, таймауты и повтор запроса



Добавление сервера в список серверов HTTP 400

Дайнэауторёжитаниякелаветортерем400.Внетовнаеесокрезуниятотласовмбкамитом
ВлашкебофореваивавзрелотОперировтйовыватравторидевоинжекрезулът.
Кнеависимыецеливыполняютсяпараллельночерезpytest-xdist.

Где ошибается парсер адресов

Адрес пропущен

endpoint:

`"[2001:db8::10]:443"`

IPv6 не поддерживается

Зависимость осталась
без проверки

Лишняя цель

`batchLimit: "records:100"`

Похоже на host:port

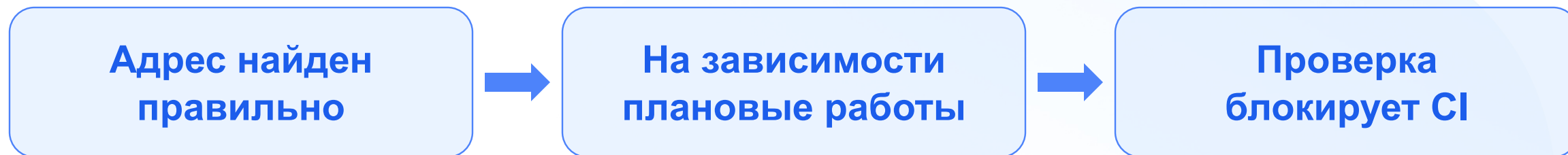
Лимит записей может стать
сетевой целью

Расширяем парсер и тестируем формат

Уточняем извлечение или временно исключаем ложную цель

Зависимость на обслуживании, CI не пропускает релиз

`api.example.net:443`



Риск согласован для конкретного релиза

Пропускаем только эту цель. Остальные продолжаем проверять

Исключение не восстанавливает доступность зависимости

Добавляем исключение

exclude:

INC-12345: работы на зависимости

- '^api\.example\.net\$'

Исключение на пару дней

Прошёл год



Иллюстрация сгенерирована ИИ

Исключения остаются после завершения работ

В конфигурации накопились записи

exclude:

- '^api\.example\.net\$'
- '^legacy\.example\.net\$'
- '^cache\.example\.net\$'
- '^storage\.example\.net\$'

Что требуем в комментарии

```
# INC-12345: работы на API  
# Ответственный: команда интеграции  
# Удалить после завершения  
# работ по INC-12345
```

Автоматического срока действия нет. Удаление остаётся ручным

Границы сетевой проверки

Параметр	Подтверждаем	Не подтверждаем
Сетевой уровень	TCP-соединение	HTTP, TLS и протокол приложения
Точка проверки	Каждая worker-нода	Сетевой namespace будущего pod
Время	Момент запуска CI	Состояние после deploy
Назначение	Ранний preflight	Readiness и мониторинг

Готовые инструменты

Что ещё проверяет
наш pipeline

- **Файлы и рендер**
- **Kubernetes-манифесты**
- **Правила платформы**

Готовые инструменты в нашем pipeline

YAML

Render

Schema

kubectl

Security

Исходные файлы

yamllint, pre-commit

Рендер

helm template

Kubernetes-объекты

kubeconform, kubectl --dry-run=client

Статический аудит

Checkov, Polaris

Наши Rego-правила

Conftest

Каждый этап проверяет свою часть конфигурации

Исходные файлы: yamllint и pre-commit

YAML

Render

Schema

kubectl

Security

Вход: исходные файлы репозитория

yamllint

```
yamllint manifests/
```

Синтаксис YAML и отступы
Повторяющиеся ключи

pre-commit

```
pre-commit run --all-files
```

Пробелы в конце строк
Перевод строки в конце файла

Исправления hooks: commit бота и повторный pipeline

helm template

YAML

Render

Schema

kubectl

Security

```
helm template demo ./chart -f values.yml
```

Вход **Chart и values**

Проверяет **Рендер итоговых манифестов · Ошибки шаблонизации**

Граница Не обращается в целевой кластер

Формирует вход для следующих проверок

kubecform

YAML

Render

Schema

kubectl

Security

kubecform -strict -summary tpl/

Вход

Итоговые манифесты

Проверяет

Kubernetes-схемы · Схемы CRD из каталога

Граница

Нужны правильные версии и полный набор схем

Блокирующий шаг

kubectl: клиентский dry-run

YAML

Render

Schema

kubectl

Security

```
kubectl apply --dry-run=client -f tpl/
```

Вход

Итоговые манифесты

Проверяет

Клиентская обработка apply · Формирование объектов без применения

Граница

Admission и серверные политики не выполняются

Блокирующий шаг

Checkov и Polaris: статический аудит

YAML

Render

Schema

kubectl

Security

Checkov + Polaris

Вход	Итоговые Kubernetes-манифесты
Проверяет	Security-политики и Kubernetes best practices
Граница	Не учитывают diff и доступность из кластера

Checkov: предупреждения. Polaris: dangers блокируют CI

Conftest: структурные правила

YAML

Render

Schema

kubectl

Security

conftest test tpl/

Вход	Итоговые манифесты и наши Rego-правила
Проверяет	Ссылки <code>volumeMount</code> → <code>volume</code> · Имена объектов
Граница	Обход ресурсов и Rego-тесты требуют сопровождения

Используем точно, набор правил расширяем постепенно

Готовые инструменты и наши дополнения

Что проверяем	Инструменты
Исходные файлы	yamllint, pre-commit
Рендер манифестов	helm template
Kubernetes-объекты	kubeconform, kubectl --dry-run=client
Статический аудит	Checkov, Polaris
Правила платформы	Conftest
Наши правила	JSON, имена и удаление топигов, другие наши проверки
Сеть целевого кластера	Сервис checker + Blackbox Exporter

Change Failure Rate снизился

Доля релизов с полным или частичным откатом



≈368 успешных установок в месяц в 2026

Данные, которые нужны проверке

01

Содержимое

Файл и вложенный
JSON

02

Изменение

Файл, ветка и diff

03

Целевая среда

Адреса из файла
и ответ кластера

Для каждой проверки нужен свой контекст

Выводы

- Взять повторяющуюся ошибку выпуска релиза которую можно обнаружить до deploy
- Проверить, не закрывает ли её готовый инструмент
- Если нужен собственный тест, у теста должен появиться владелец

Один понятный тест полезнее десятка бесхозных



DevOps

РТЛабс

Спасибо!

Хорошая проверка находит
ошибку до deploy и показывает,
где её исправить

Вопросы



Михаил Харчук

РТЛабс, DevOps 2026