Эволюция review-стендов:
Как мы ускоряли тестирование новых фич

Самутичев Константин

TechLead DevOps

@backk

Обо мне

- K8s
- Grafana / Mimir / Loki / Tempo
- Kube-Prometheus-Stack
- Gitlab CI
- Golang

HR TECH®
РОСАТОМ

42 кластера k8s

16 разрабатываемых продуктов

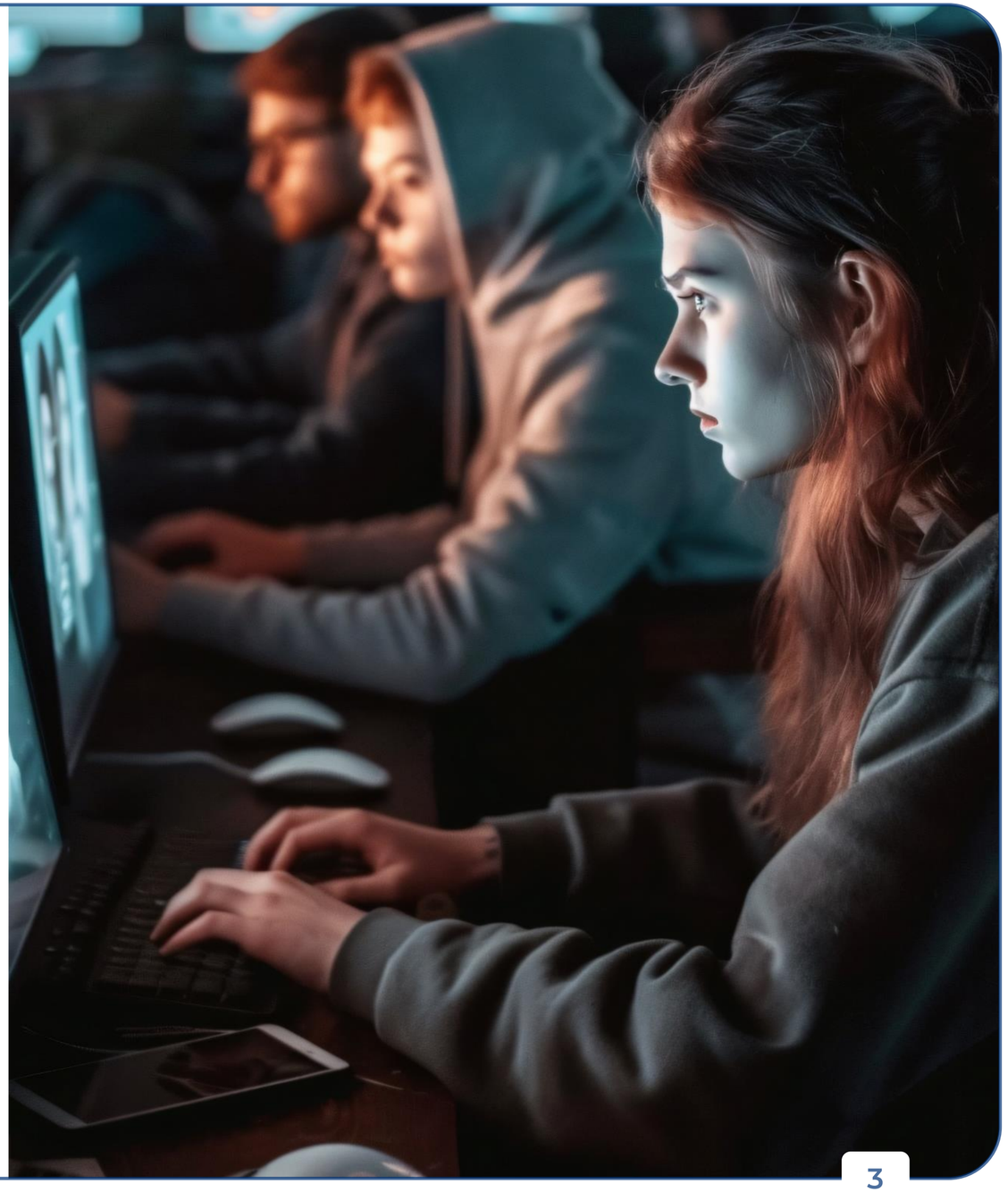
4 года

Гринатом

Центр HR Tech

NodeJS / Elixir / Moleculer / Java / ...

8 DevOps-инженеров



Этапы эволюции продукта

Wave 3

Full-scale product

Wave 2

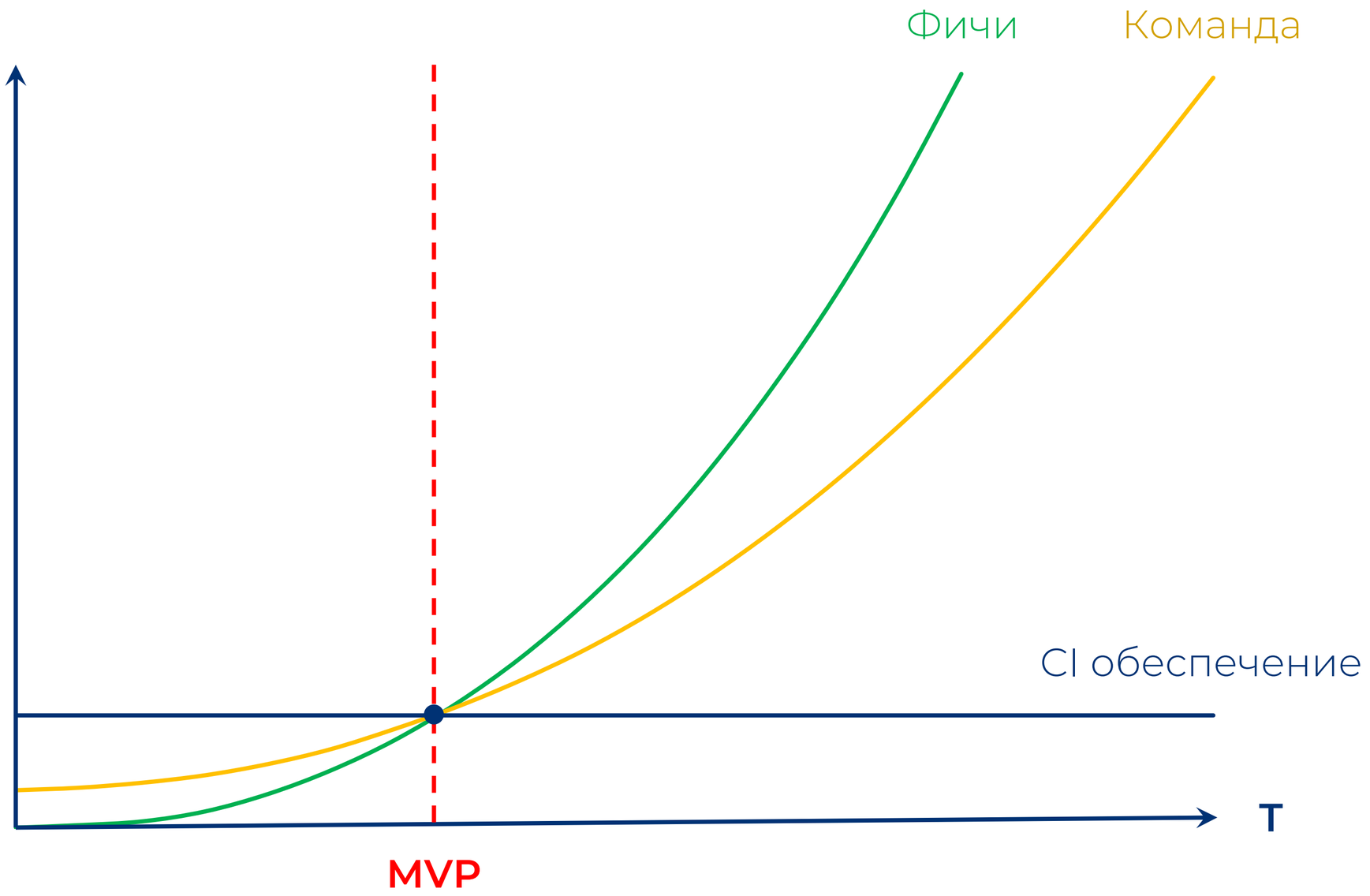
Core product

Wave 1

MVP

dev

Эволюция CI



Этап 0

«До начала времён»

Есть стенд – dev

Этап 0

«До начала времён»

Есть стенд – dev

Разработчики и тестировщики работают на одном стенде

Этап 0

«До начала времён»

Есть стенд – dev

- В dev сливалось «готовое»

Разработчики и тестировщики работают на одном стенде

Этап 0

«До начала времён»

Есть стенд – dev

- В dev сливалось «готовое»
- Кейс тестировщика аффектит кейс другого тестировщика

Разработчики и тестировщики работают на одном стенде

Этап 0

«До начала времён»

Есть стенд – dev

- В dev сливалось «готовое»
- Кейс тестировщика аффектит кейс другого тестировщика
- Попытка уйти на локальные окружения

Разработчики и тестировщики работают на одном стенде

Этап 0

«До начала времён»

Есть стенд – dev

- В dev сливалось «готовое»
- Кейс тестировщика аффектит кейс другого тестировщика
- Попытка уйти на локальные окружения
- ...

Разработчики и тестировщики работают на одном стенде

Статические стенды

Этап 0

Решение проблемы

Этап 0

Решение проблемы

Статические стенды

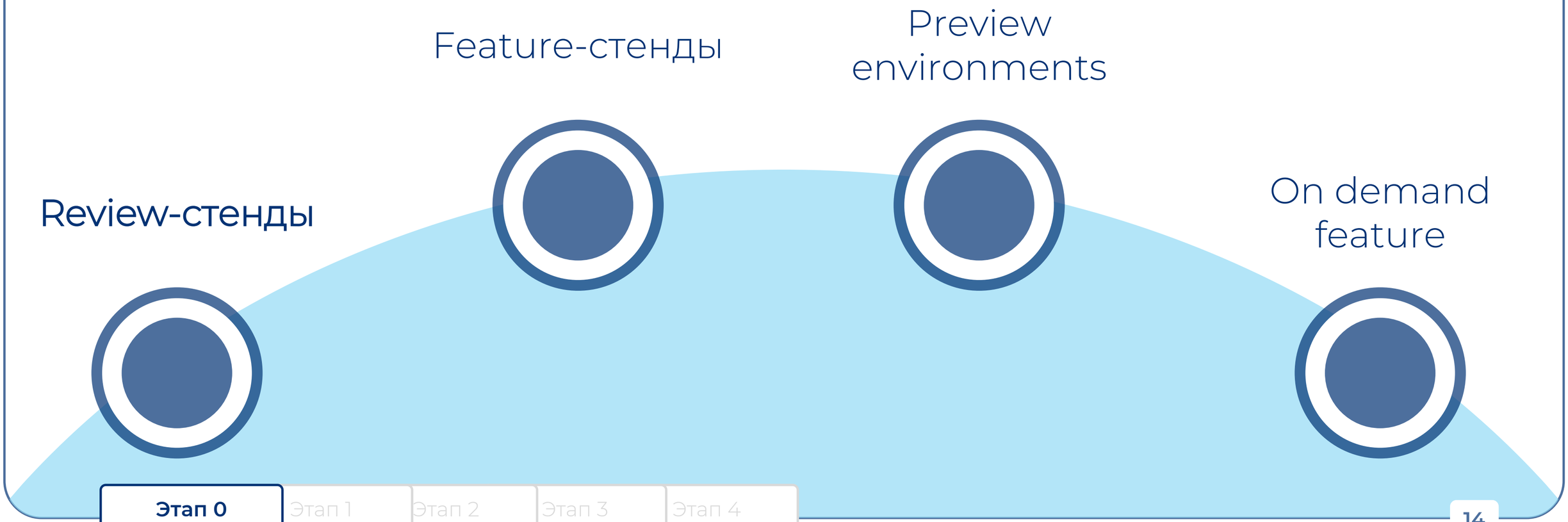
Динамические стенды

Этап 0

Решение проблемы

Статические стенды

Динамические стенды



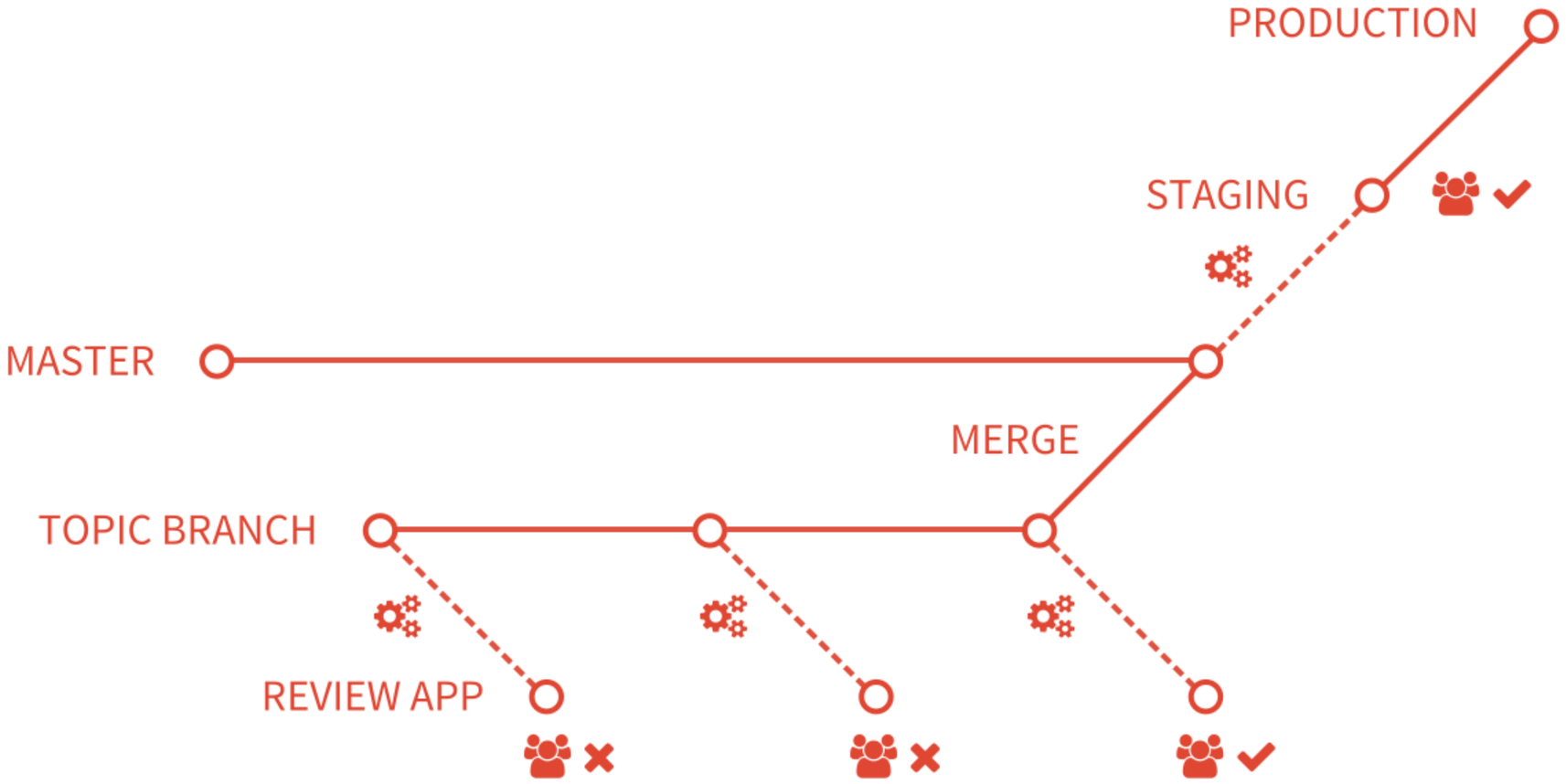
Что такое review-стенд

Ревью-стенд – это инструмент для совместной работы, который предоставляет среду для демонстрации изменений в продукте

Как помогают review-стенды

Индивидуальное независимое окружение

Review apps в Gitlab



Этап 1. После MVP

Требования к review-стендам

Этап 1. После MVP

Требования к review-стендам

Короткоживущие стенды для оперативного тестирования изменений

Этап 1. После MVP

Требования к review-стендам

Короткоживущие стенды для оперативного тестирования изменений

- Полная копия эталонной среды разработки

Состав и версии микросервисов

БД

3rd party решения

Этап 1. После MVP

Требования к review-стендам

Короткоживущие стенды для оперативного тестирования изменений

- Полная копия эталонной среды разработки

Состав и версии микросервисов

БД

3rd party решения

- Предопределённые переменные

Этап 1. После MVP

Требования к review-стендам

Короткоживущие стенды для оперативного тестирования изменений

- Полная копия эталонной среды разработки

Состав и версии микросервисов

БД

3rd party решения

- Предопределённые переменные
- Скорость развёртывания

Этап 1. После MVP

Наш стек для dev

- K8s – v1.23
- CSI – Longhorn
- PostgreSQL, Kafka, Redis в K8s
- Gitlab

Этап 1. После MVP

Golden-стенд (dev)

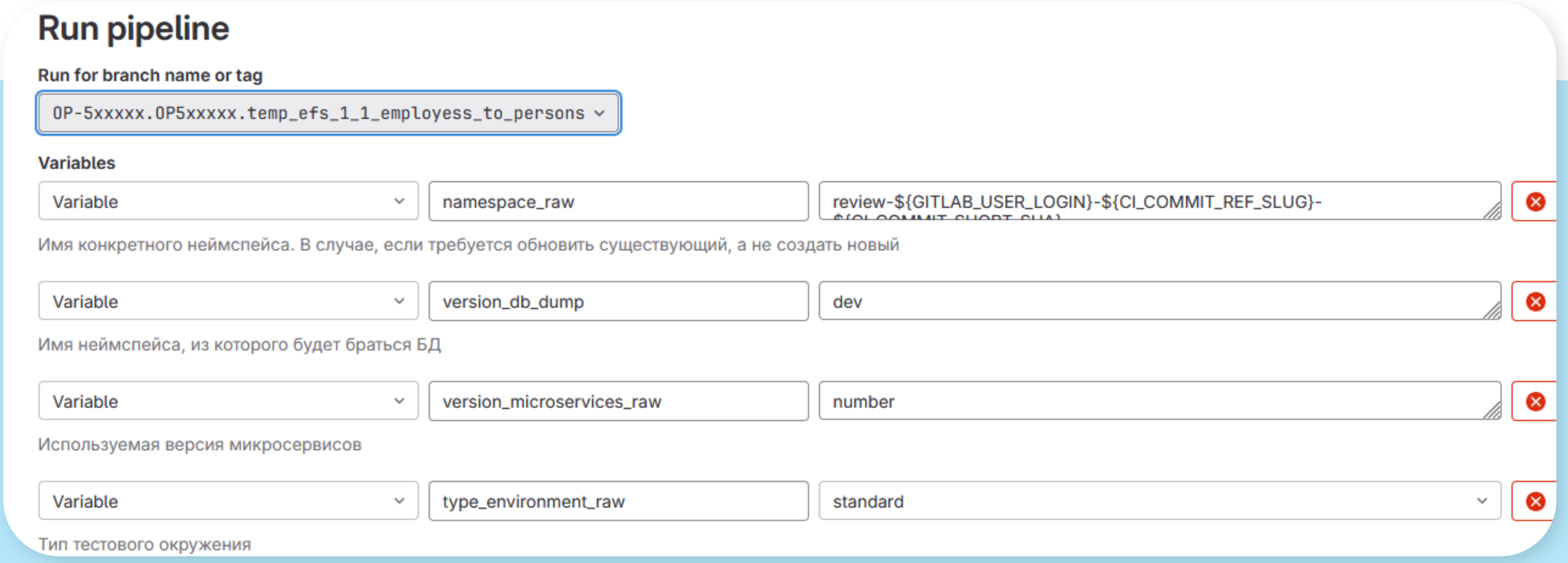
- СУБД PostgreSQL - ~ 3Gb
- 3rd Party – Kafka, Redis, PGBouncer
- 20 подов

Этап 1. После MVP

Что у нас получилось

```
rules:
  - if: $CI_COMMIT_TAG
    when: never
  - if: '$CI_PIPELINE_SOURCE == "web" && $CI_COMMIT_BRANCH != "dev"'
```

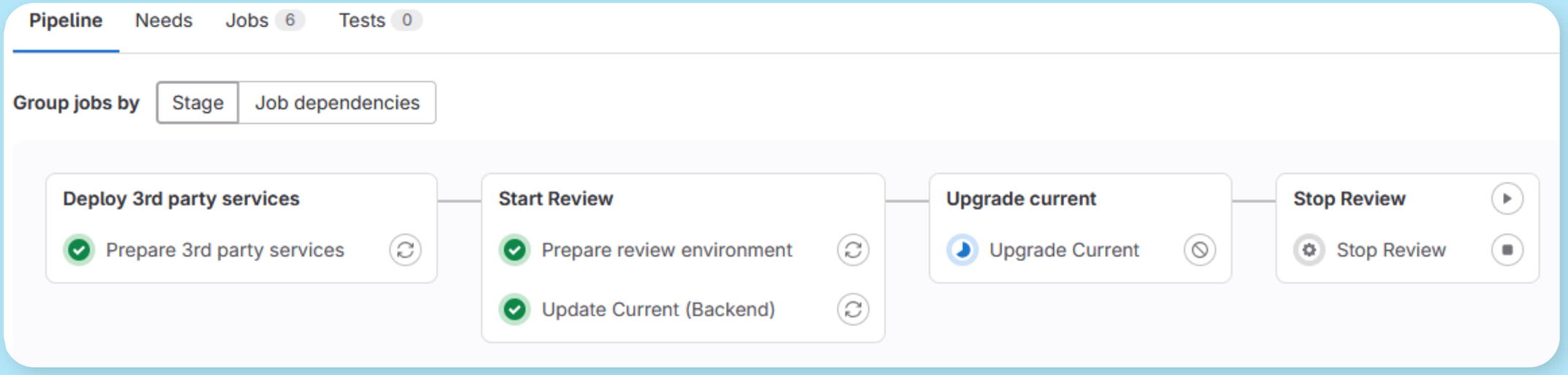
Предопределённые переменные



Этап 1. После MVP

Что у нас получилось

Пайплайн стенда



- Деплой сторонних компонентов
- Заливка БД
- Список версий микросервисов

- Раскатка микросервисов ИС из чартов
- Билд feature-образа

- Раскатка feature-образа в стенд

- Ручная или автоматическая остановка стенда

Этап 1. После MVP

Результат

Время развёртывания – 7 минут

После MVP

Проблемы

Команда продукта

Количество данных

Время раскатки стенда

После MVP

Проблемы

Команда продукта

Количество данных

~ 10Gb

Время раскатки стенда

15 минут

Этап 2. CSI

Что такое Longhorn

Лёгкая, надёжная и мощная распределённая **блочная** система хранения данных для Kubernetes

- Отсутствие единой точки отказа
- Инкрементальные снапшоты
- Бэкап изменений на NFS или S3 хранилище
- Бесшовное обновление

Мощь CSI

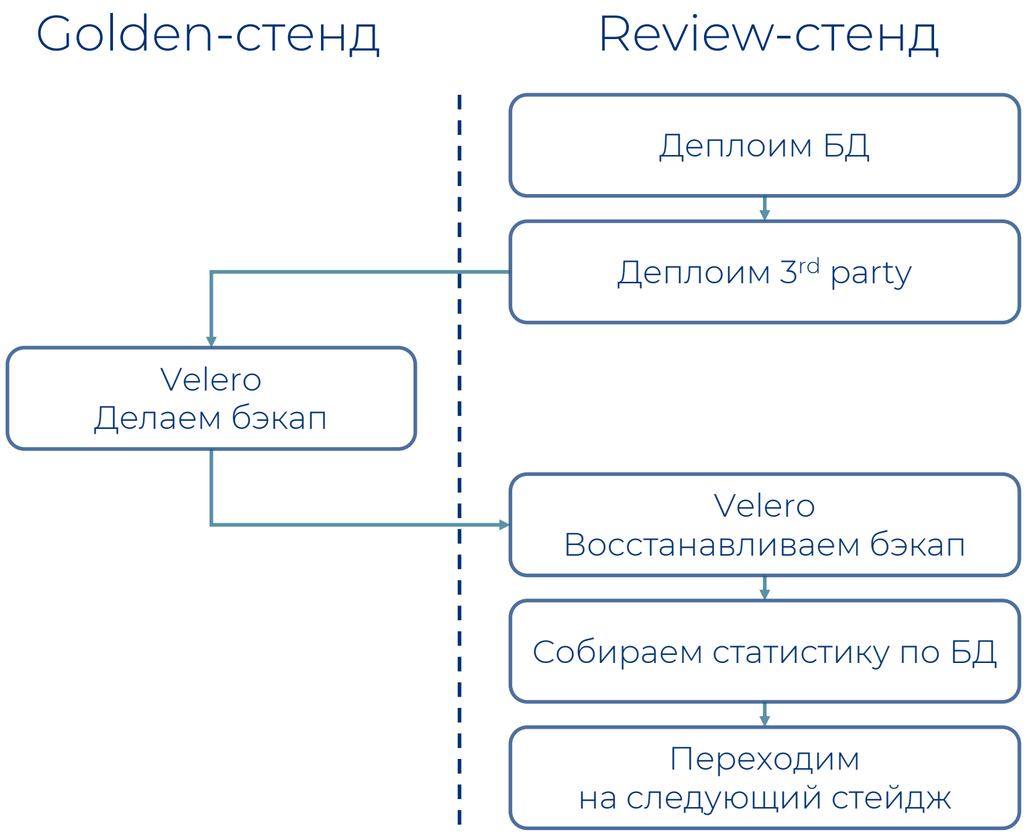
LONGHORN

Этап 2. CSI


Golden-стенд
Review-стенд
Деплоим БД
Деплоим 3rd party
Запускаем переливку данных по пайпу
Собираем статистику по БД
Переходим
на следующий стейдж

Этап 2. CSI

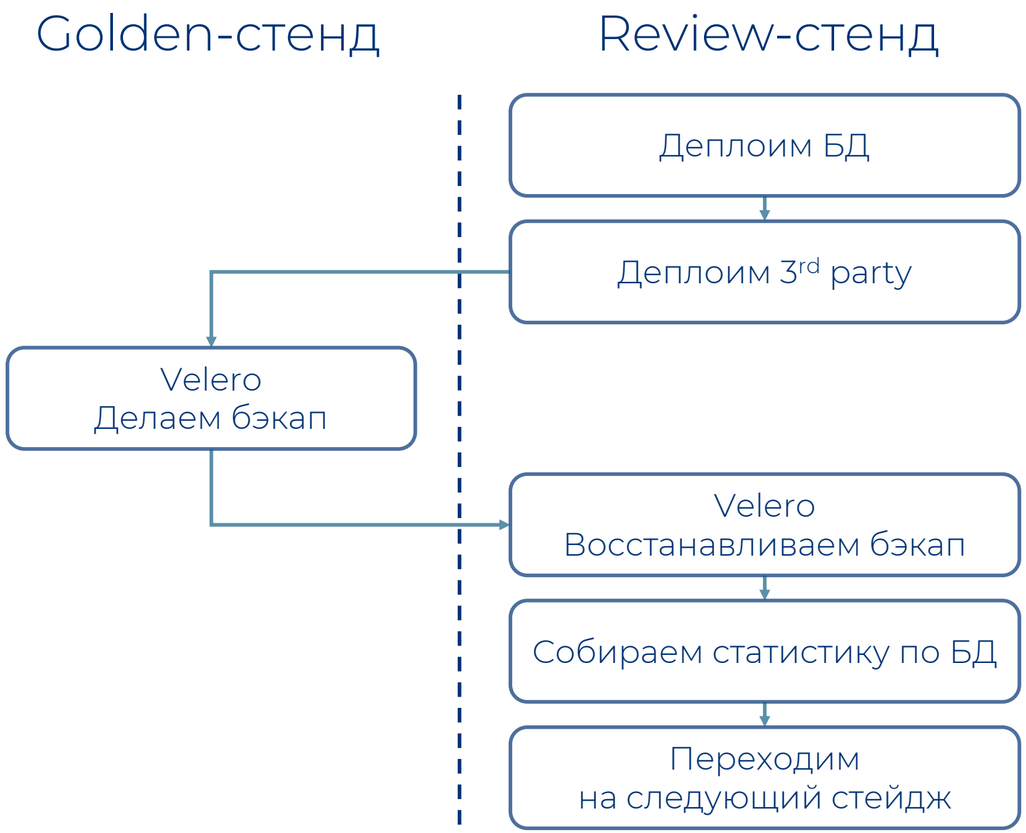
CSI + External-Snapshotter + Velero + Minio



Этап 2. CSI

CSI + External-Snapshotter + Velero + Minio

15 минут



~ 15 минут

Этап 2. CSI

Мощь CSI

CSI

Проблемы

Количество данных

~ 20Gb

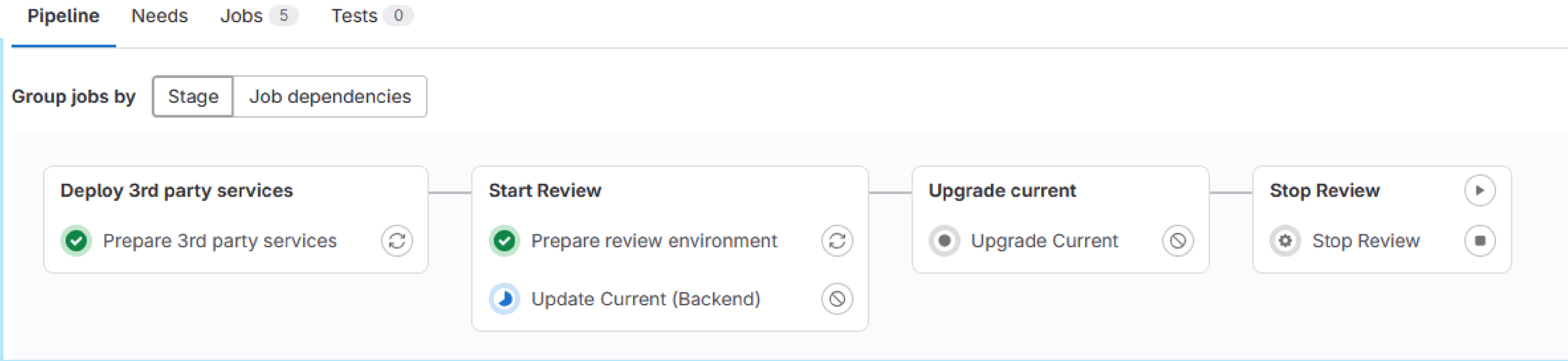
Время раскатки стенда

30-40 минут

Этап 3. Ускорение

Кэш раннеров

```yaml
cache:
  - key:
      files:
        - yarn.lock
    paths:
      - node_modules
    policy: pull
```



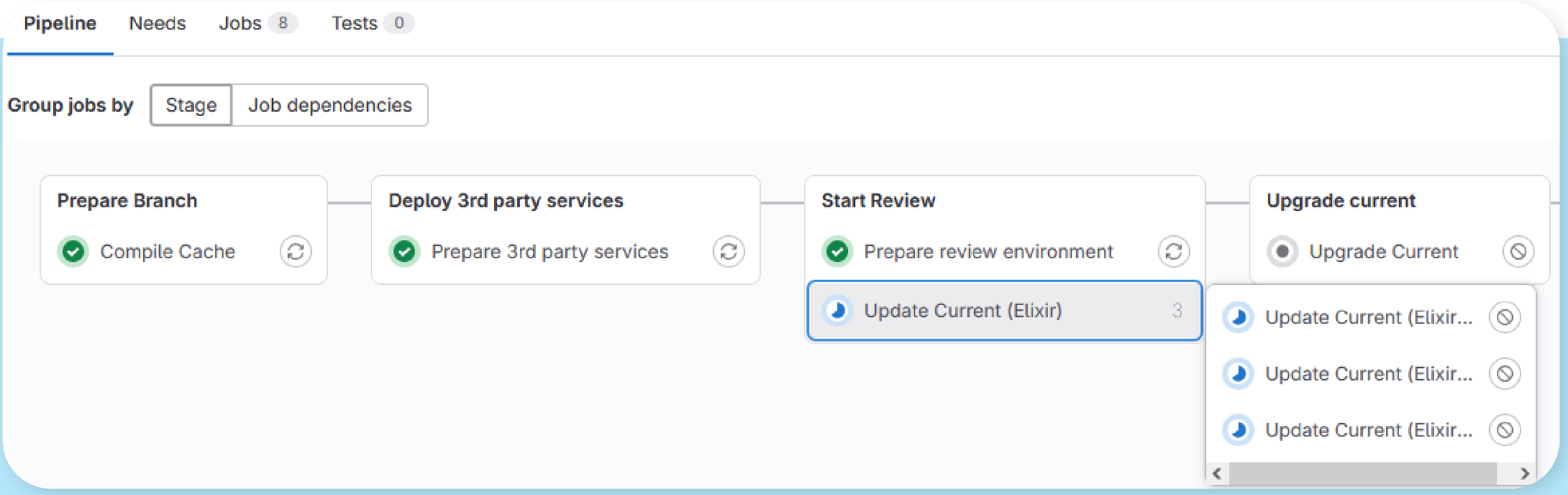
Этап 3. Ускорение

Параллельная сборка

```
Update Current (Elixir) 1/3:
  variables:
    RELEASE_NAME: "elixir-service-one"

Update Current (Elixir) 2/3:
  variables:
    RELEASE_NAME: "elixir-service-two"

Update Current (Elixir) 3/3:
  variables:
    RELEASE_NAME: "elixir-service-three"
```



Ситуация на текущий момент

- У нас есть CSI
- External-Snapshotter => VolumeSnapshot
- Backup / Restore

Этап 3. Ускорение

Ситуация на текущий момент

Ситуация на текущий момент

- У нас есть CSI
- External-Snapshotter => VolumeSnapshot
- Backup / Restore

```yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: test-csi-volume-snapshot-longhorn
  namespace: dev
spec:
  volumeSnapshotClassName: longhorn-snapshot-vsc
  source:
    persistentVolumeClaimName: data-postgresql-0
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-restore-pvc
spec:
  storageClassName: longhorn
  dataSource:
    name: test-csi-volume-snapshot-longhorn
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Этап 3. Ускорение

Ситуация на текущий момент

Ситуация на текущий момент

- У нас есть CSI
- External-Snapshotter => VolumeSnapshot
- Backup / Restore


```yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: test-csi-volume-snapshot-longhorn
  namespace: dev
spec:
  volumeSnapshotClassName: longhorn-snapshot-vsc
  source:
    persistentVolumeClaimName: data-postgresql-0
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-restore-pvc
spec:
  storageClassName: longhorn
  dataSource:
    name: test-csi-volume-snapshot-longhorn
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Новые фичи Kubernetes

Kubernetes v1.26: Alpha support for cross-namespace storage data sources

`CrossNamespaceVolumeDataSource`	`false`	Alpha	1.26	–

- `CrossNamespaceVolumeDataSource`: Enable the usage of cross namespace volume data source to allow you to specify a source namespace in the `dataSourceRef` field of a PersistentVolumeClaim.

Этап 3. Ускорение

Research and Development

1. Включаем фичу для kube-apiserver и kube-controller-manager

   - --feature-gates=CrossNamespaceVolumeDataSource=true

Этап 3. Ускорение

Research and Development

1. Включаем фичу для kube-apiserver и kube-controller-manager

   - --feature-gates=CrossNamespaceVolumeDataSource=true

2. Включаем фичу для csi-provisioner

   - --feature-gates=CrossNamespaceVolumeDataSource=true

Research and Development

3. В кластере должна поддерживаться абстракция ReferenceGrant

> **Note:**
> When you specify a namespace for a volume data source, Kubernetes checks for a ReferenceGrant in the other namespace before accepting the reference. ReferenceGrant is part of the `gateway.networking.k8s.io` extension APIs. See ReferenceGrant in the Gateway API documentation for details. This means that you must extend your Kubernetes cluster with at least ReferenceGrant from the Gateway API before you can use this mechanism.

git clone https://github.com/kubernetes-sigs/gateway-api.git

kubectl create -f config/crd/standard/gateway.networking.k8s.io_referencegrants.yaml

4. Выдадим права Longhorn на использование ReferenceGrant

- apiGroups: ["gateway.networking.k8s.io"]

  resources: ["referencegrants"]

  verbs: ["get", "list", "watch"]

kubectl patch clusterrole longhorn-role --type='json' -p='[{"op": "add", "path": "/rules/0", "value":{ "apiGroups": ["gateway.networking.k8s.io"], "resources": ["referencegrants"], "verbs": ["get", "list", "watch"]}}]'

Research and Development

5. Выдаём разрешение на клонирование

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: ReferenceGrant
metadata:
  name: allow-clone-prod-pvc
  namespace: dev
spec:
  from:
  - group: ""
    kind: PersistentVolumeClaim
    namespace: review-new
  to:
  - group: ""
    kind: PersistentVolumeClaim
    name: data-postgresql-0
```

Этап 3. Ускорение

Research and Development

5. Выдаём разрешение на клонирование

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: ReferenceGrant
metadata:
  name: allow-clone-prod-pvc
  namespace: dev
spec:
  from:
  - group: ""
    kind: PersistentVolumeClaim
    namespace: review-new
  to:
  - group: ""
    kind: PersistentVolumeClaim
    name: data-postgresql-0
```

6. Клонируем

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cloned-pvc
  namespace: review-new
spec:
  storageClassName: longhorn
  dataSourceRef:
    name: data-postgresql-0
    kind: PersistentVolumeClaim
    namespace: dev
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Этап 3. Ускорение

Мощь CSI

Golden-стенд

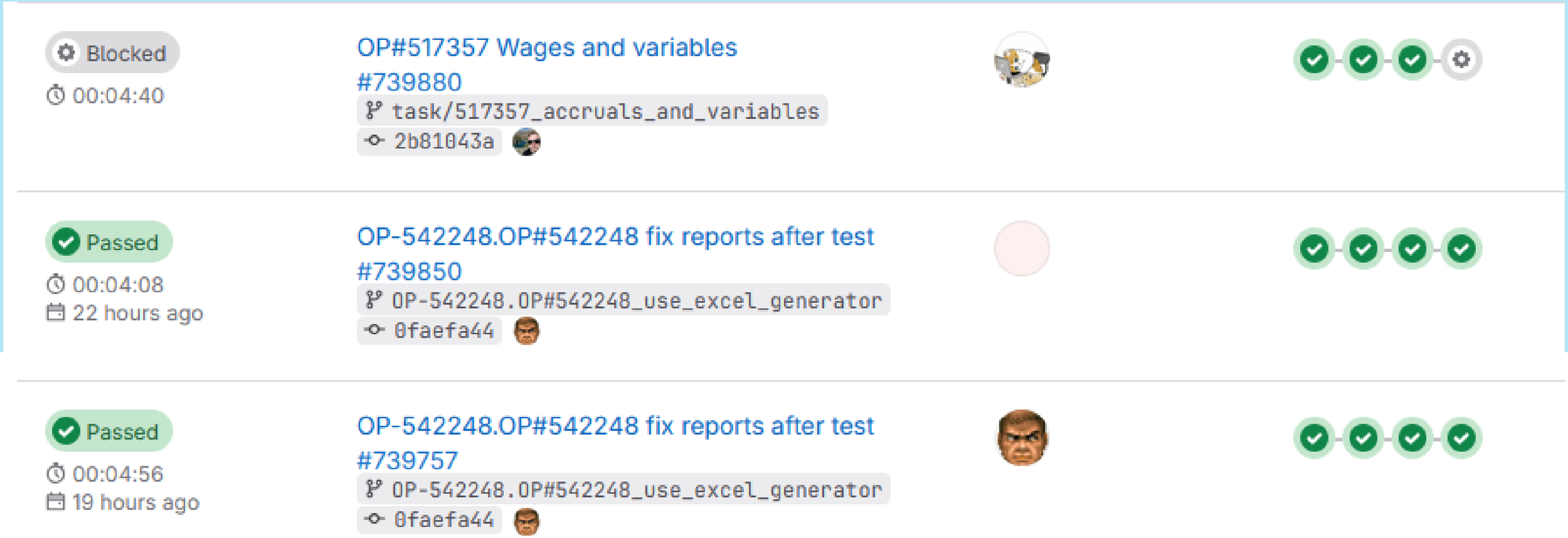
- Деплоим БД
- Деплоим 3rd party
- Запускаем клонирование в review

Review-стенд

- Деплоим 3rd party
- Деплоим СУБД с существующей pvc
- Переходим на следующий стейдж

Этап 3. Ускорение

Результаты



Ускорение

Проблемы

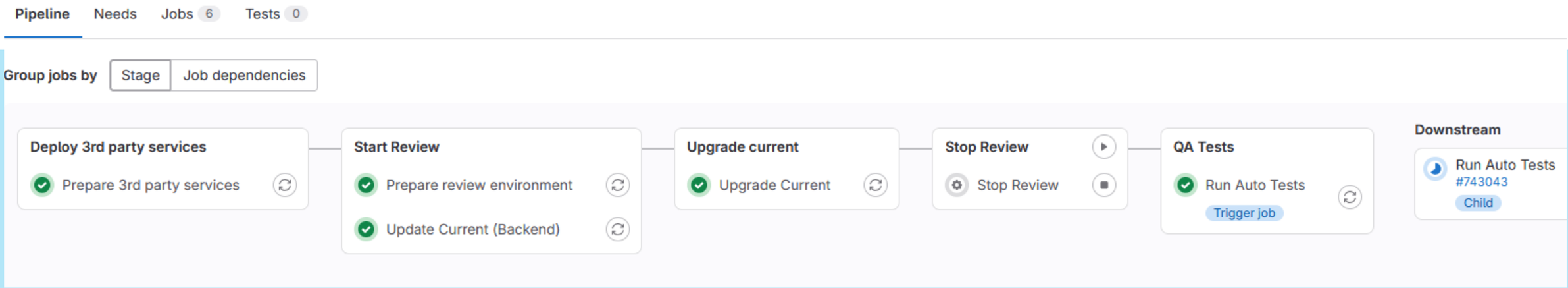
Количество MR

Количество оттестированных фич

Стабильность продукта

Этап 4. Настоящее время

Автотесты



Этап X. Будущее

Отдалённая перспектива

- БД из других тенантов
- Более плотная упаковка стендов
- Единый кластер для review-стендов

Каких вопросов не коснулись

- Раскатка стендов
- Внутренние проверки стендов
- Получение версий микросервисов

Выводы

- Review-стенды даже в минимальной конфигурации позволяют вывести качество тестирования на новый уровень
- Отрицательный результат даёт фундамент для будущих побед
- При обновлении CSI не забудьте повторно выдать права на review-стенды :)

Ссылки на материалы

Олег Вознесенский – Как правильно создавать feature-окружения на Gitlab, Helm и Kubernetes

Kubernetes Blog. CrossNamespaceVolumeDataSource

Kubernetes Documentation. CrossNamespaceVolumeDataSource