Интеграция Kubernetes в GitlabCI

Не забудь включить запись!

План

- Helm
- CI/CD в GitLab
- Стратегии деплоя в Kubernetes

Helm

Проблемы?

- Устали писать манифесты вручную (они большие и их много)
- Версии манифестов не привязаны к версиям приложений
- Нет шаблонов манифестов (не понимают переменных окружений и т.д.)
- Что если надо сделать Rollback нескольких манифестов
- Как поделиться пакетом манифестов?

Helm

Helm - пакетный менеджер для Kubernetes, разработанный ребятами из Deis

Сейчас это также часть проекта Kubernetes. Что умеет Helm?

- упаковывать несколько ресурсов k8s в один пакет - Chart
- шаблонизировать установку
- устанавливать Chart'ы - делать Release
- делать Upgrade (обновления) и Rollback (откаты) выкатываемых приложений
- управлять зависимостями между пакетами
- хранить пакеты в удаленных репозиториях

Charts

Chart - пакет в Helm. По сути, chart - коллекция файлов и папок.

Структура Chart'а приложения ui

```
ui/
  Chart.yaml
  README.md
  requirements.yaml
  values.yaml
  charts/
  templates/
  templates/NOTES.txt
```

Chart.yml

Обязательный файл, содержащий информацию о Chart’е:

- имя
- описание
- версию чарта
- версию приложения
- различная метаинформация (разработчик, ссылки и т.д)

Chart.yml

```
name: ui
version: 0.1.1
description: OTUS reddit application
maintainers:
  - name: Someone
    email: my@mail.com
appVersion: 0.7.0
...
```

! Версия приложения и версия чарта могут различаться

Templates

Шаблоны манифестов Chart'а, содержатся в директории templates/

```
ui/
  Chart.yaml
  templates/
    ui-deployment.yml
    ui-ingress.yml
    ui-service.yml
```

Templates

ui-service.yml

```
---
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
spec:
  type: NodePort
  ports:
  - port: 9292
    protocol: TCP
    targetPort: 9292
  selector:
    app: reddit
    component: ui
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-ui
  labels:
    app: reddit
    component: ui
spec:
  type: {{ .Values.service.type }}
  ports:
  - port: {{ .Values.service.port }}
    protocol: TCP
    targetPort: {{ .Values.pod.port }}
  selector:
    app: {{ .Values.metadata.app }}
    component: {{ .Values.metadata.component }}
```

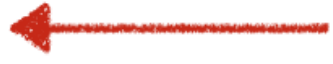
+ values.yml

Templates

ui-service.yml

```
---
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-ui
  labels:
    app: reddit
    component: ui
spec:
  type: {{ .Values.service.type }}
  ports:
  - port: {{ .Values.service.port }}
    protocol: TCP
    targetPort: 9292
  selector:
    app: {{ .Values.metadata.app }}
    component: {{ .Values.metadata.component }}
```

Предопределенные переменные Helm



Кастомные переменные, передаваемые через values.yml

Values

values.yml

```
service:
  type: NodePort
  port: 9292

metadata:
  app: reddit
  component: ui
```

- Указывает, что должно подставлено в template
- У каждого chart-а есть свой default-ный файл values.yml
- Значения могут быть перезаписаны для конкретного release

Charts

Можно инициализировать шаблон для создания чарта

```
helm create ui
```

Получаем готовую структуру

```
ui/
  Chart.yaml
  values.yaml
  charts/
  templates/
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    service.yaml
```

Go Templating

В основе Helm лежит шаблонизатор Go с 50+ встроенными функциями.

```
{{- if .Values.server.persistentVolume.enabled }}
          persistentVolumeClaim:
…
{{- else }}
```

Условия

```
{{- range $key, $value := .Values.server.annotations }}
   {{ $key }}: {{ $value }}
{{- end }}
```

Циклы

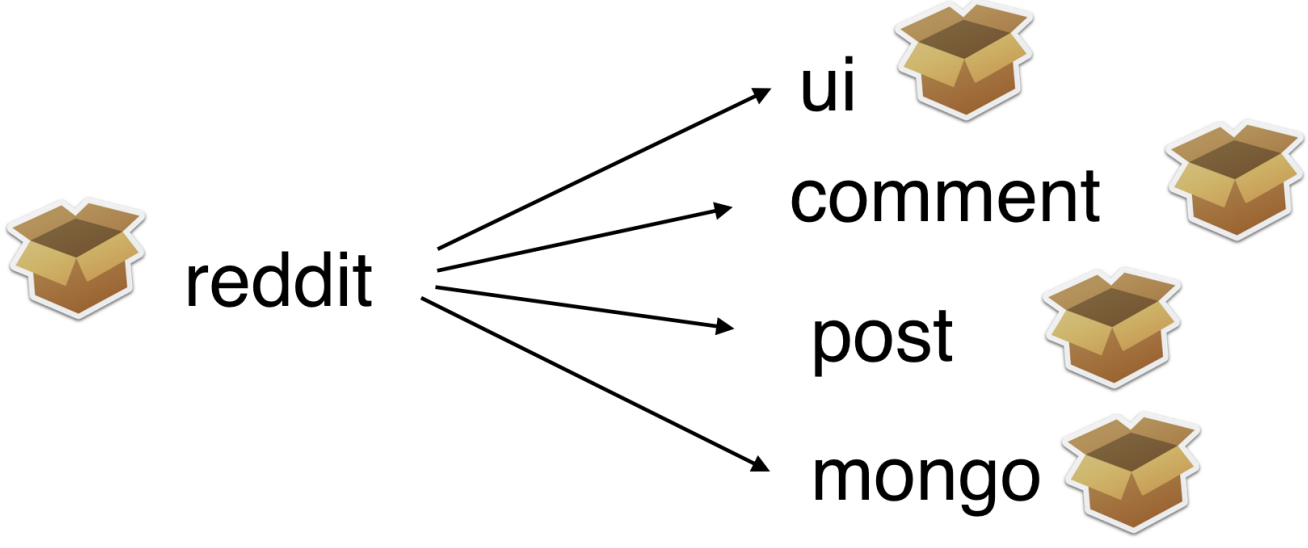
```
value: {{required ".Values.who required!" .Values.who }}
```

Функции

[Ссылка на документацию шаблонизатора]

Управление зависимостями

Структура приложения reddit



Управление зависимостями

Зависимости описываются в requirements.yml

```
reddit/
    requirements.yml
    charts/
        ui/
            Chart.yaml
            templates/
```

Управление зависимостями

requirements.yml

```
dependencies:
  - name: ui
    version: 1.0.0
    repository: "file://../ui"
  - name: post
    version: 2.0.1
    repository: “file://../post"
  - name: comment
    version: 1.0.2
    repository: "file://../comment"
  - name: mongo
    version: 3.2
    repository: "http://example.com/charts"
```

Управление зависимостями

А храним в charts/

reddit/
  requirements.yml
  charts/

ui/
  Chart.yaml
  # ...

зависимость в виде Chart'а

post/
  Chart.yaml
  # …

comment/
  Chart.yaml
  # …

mongo-3.2.tgz

зависимость в виде архива Chart'а

Repository

Где хранить Chart'ы ?

1. Рядом с кодом приложений
2. Chart-репозиторий

- простой HTTP-сервер
  - файл индекса (index.yml) со список данных о чартах в репозитории
  - архивы чартов в .tgz-формате
  - .prov-файлы для проверки целостности

Helm-client

helm - консольное приложение

- Используем при локальной работе с Chart'ами
- Управление репозиториями
- Взаимодействие с сервером Tiller

Проще говоря, Helm-клиент отвечает за работу с Chart'ами

Tiller

Tiller- это серверная часть HELM, расположенная в кластере (это тоже POD) Сервер общается с Kubernetes API и отвечает за:

- Ожидание входящих запросов от Helm-клиента
- Сборку конфигурации чарта в Release
- Установку Chart'ов в Kubernetes и отслеживание соответствующего Releas'а
- Обновление и удаление Releas'ов
- Хранит информацию о релизах в своем неймспейсе в configMaps

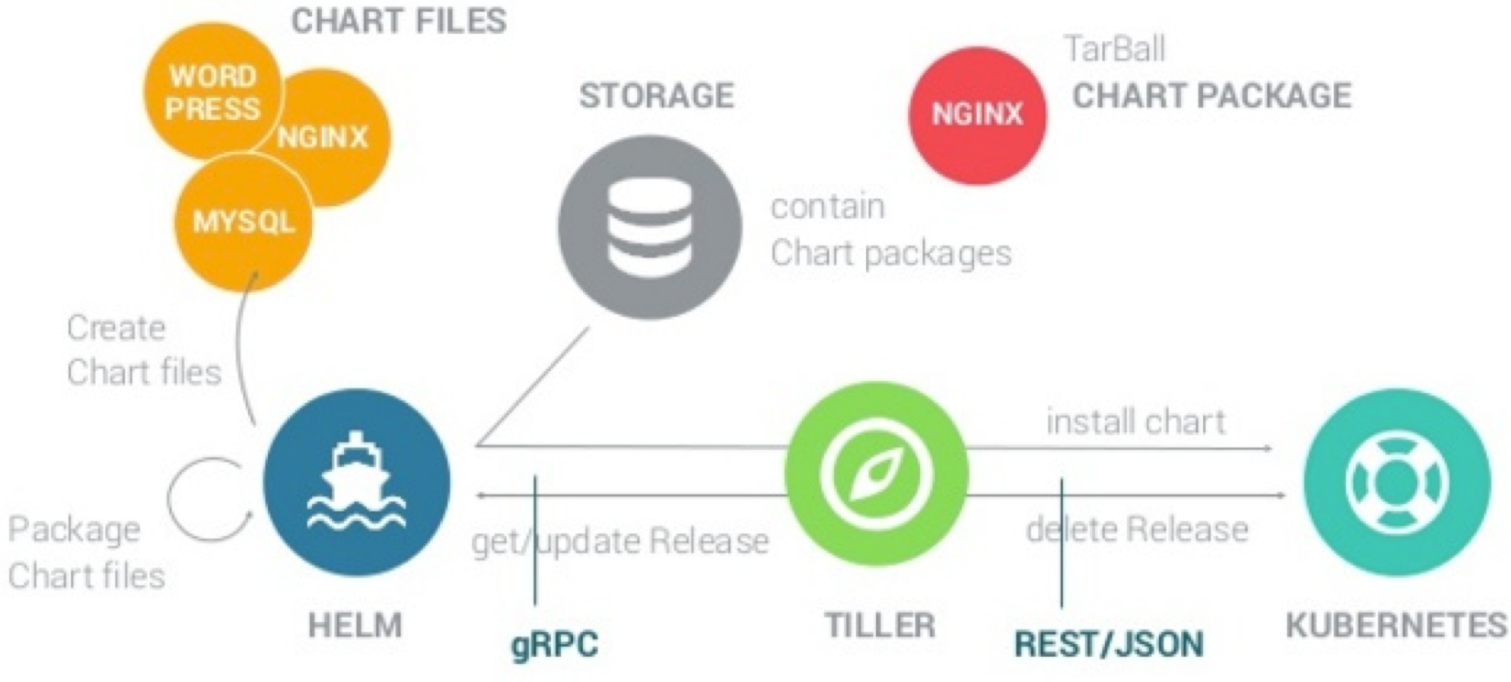
Проще говоря Tiller-сервер отвечает за работу с Releas'ами

Что нового в helm:3.0.0-alpha.1

- Убрали Tiller
- Переименованы некоторые команды клиента
  - helm inspect -> helm show
  - helm fetch -> helm pull
- Name (или --generate-name) теперь обязателен при `helm install`
- Release'ы теперь хранятся в namespace'ах
- Helm заявил о поддержке всего функционала кубернетеса, касающегося авторизации, идентификации и безопасности

Подробнее тут

Архитектура Helm




```
$ helm init
```

Helm install

```
$ helm install reddit  --values dev.yml --name reddit --namespace default
```

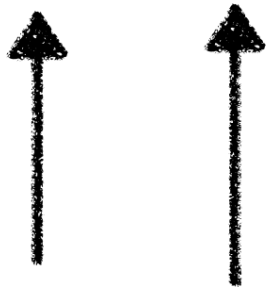
```
NAME:   reddit
LAST DEPLOYED: Thu Nov 30 15:31:01 2017
NAMESPACE: dev
STATUS: DEPLOYED

RESOURCES:
==> v1/Service
NAME         TYPE       CLUSTER-IP     EXTERNAL-IP  PORT(S)          AGE
comment      ClusterIP  10.11.243.233  <none>       9292/TCP         2s
comment-db   ClusterIP  10.11.241.40   <none>       27017/TCP        2s
post-db      ClusterIP  10.11.251.233  <none>       27017/TCP        2s
post         ClusterIP  10.11.242.113  <none>       5000/TCP         2s
ui           NodePort   10.11.245.194  <none>       9292:32395/TCP   2s

==> v1beta1/Deployment
NAME     DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
comment  1        1        1           0          2s
mongo    1        1        1           0          2s
ui       3        3        3           0          2s

==> v1beta2/Deployment
post  1  1  1  0  2s

==> v1beta1/Ingress
NAME  HOSTS  ADDRESS  PORTS  AGE
ui    *      80       2s
```

Release

Release - установленный Chart

$ helm ls

```
NAME        REVISION    UPDATED                     STATUS    CHART                 NAMESPACE
gitlab      14          Thu Nov 30 10:43:07 2017    DEPLOYED  gitlab-omnibus-0.1.36 default
reddit      2           Wed Nov 29 18:17:57 2017    DEPLOYED  reddit-0.1.1          default
```



имя релиза

ревизия (версия релиза)

Версия Chart'a

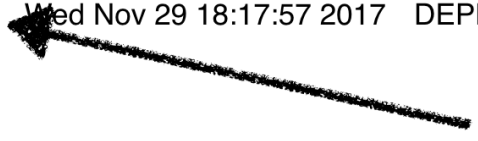
Helm rollback

$ helm rollback reddit 1

Номер ревизии

$ helm ls

```
NAME        REVISION    UPDATED                     STATUS      CHART                   NAMESPACE
gitlab      14          Thu Nov 30 10:43:07 2017    DEPLOYED    gitlab-omnibus-0.1.36   default
reddit      3           Wed Nov 29 18:17:57 2017    DEPLOYED    reddit-0.1.0            default
```



Номер новой (старой) ревизии

Hooks

Hooks - определенные действия, выполняемые в различные моменты жизненного цикла поставки.Hook, как правило, запускает Job.

Виды hook’ов:

- pre/post-install
- pre/post-delete
- pre/post-upgrade
- pre/post-rollback

Tests

- Когда хотим проверить, что запущенный Chart работает.
- Тест - описание Pod’а для проверки
- Лежат в templates/tests/
- Тест возвращает либо `success` (exit code 0) , либо `failure`

Недостатки Helm

- Kubernetes ничего не знает о Chart’ах
- Нет информативных логов о проблемах
- Слабый Lint’ер (пропускает много ошибок)
- При ошибках установки изменения прийдется удалять вручную
- Нет работы с Docker образами

Почему Helm, а не ...?

- Единый стандарт упаковки
- Дорабатывается сообществом Kubernetes
- Пакеты просто устанавливать и управлять ими
- Поддержание консистентного состояния
- Работающие механизмы обновлений и откатов
- Предложены механизмы тестирования

Что есть ещё?

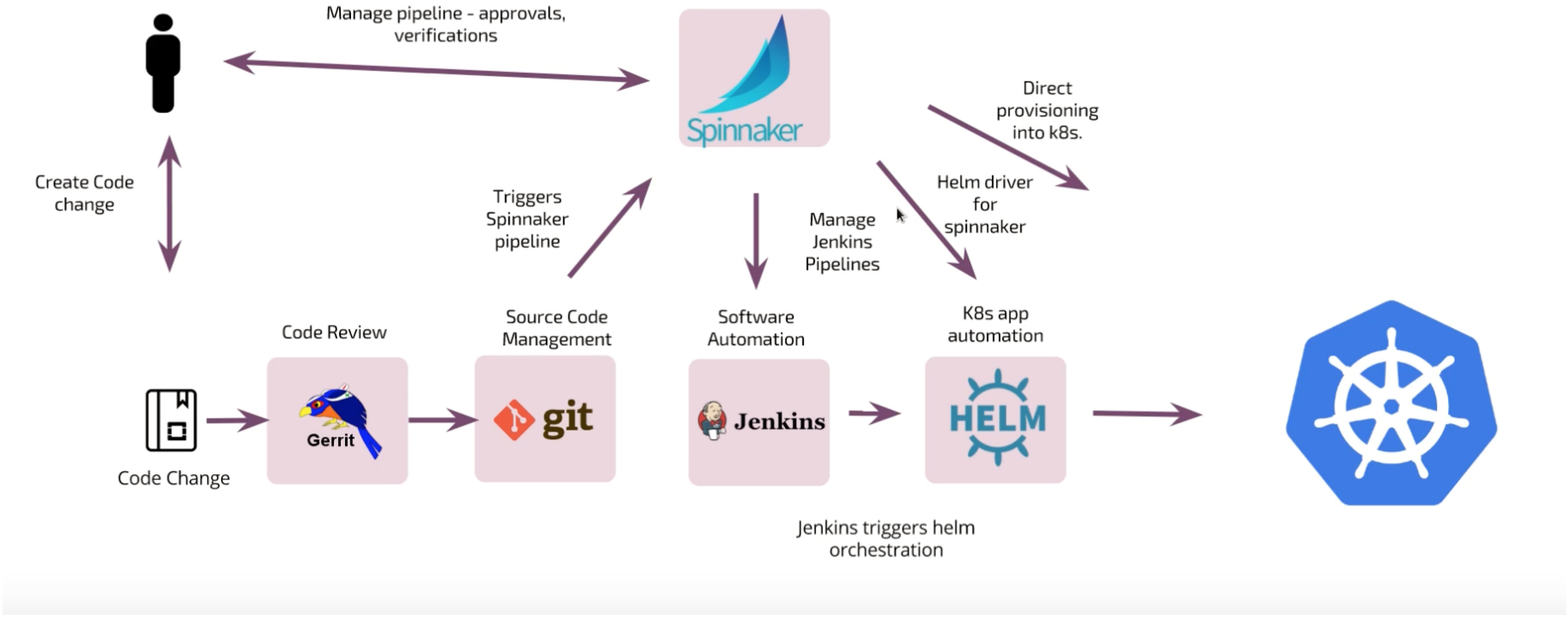
- Ksonnet
- Draft
- GitKube
- Skaffold
- Forge
- kubepack
- Spinnaker

Spinnaker

Continuous Delivery(CD) платформа для автоматизации доставки релизов.

- со встроенными best practices развертывания приложений из коробки
- аж 2 провайдера для kubernetes direct provisionning

Spinnaker flow examples



Spinnaker providers - legacy kubernetes

Мапинг сущностей Spinnaker на абстракции кубера:

- Instance -> Pod
- Server Group -> Replica Set
- Cluster -> Deployment (опционально)
- Load Balancer -> Service

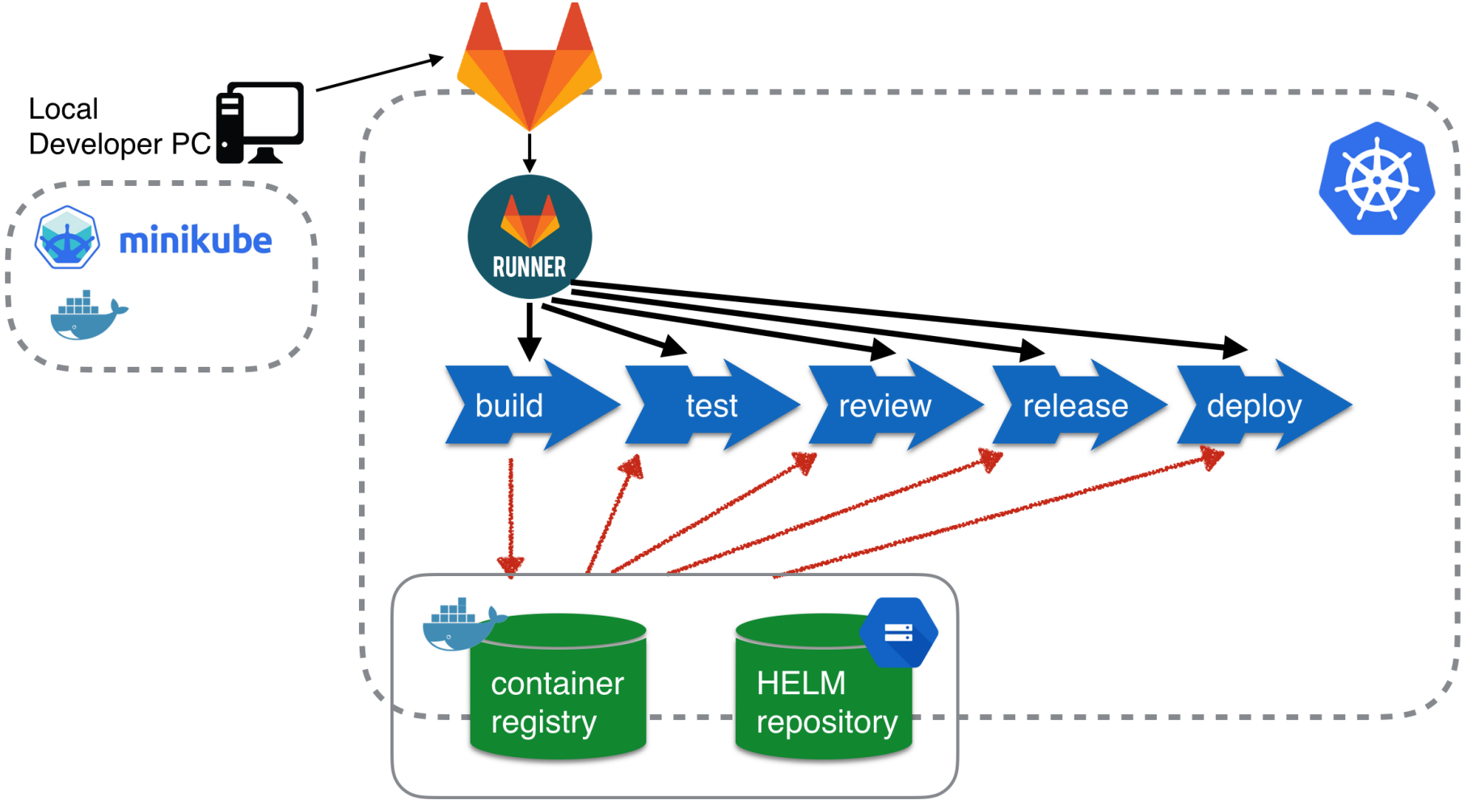
Spinnaker providers - modern kubernetes

- *manifest stability* концепция развертывания
  - у каждого ресурса свои индикаторы достижения стабильности
- поддержка удаленно хранящихся манифестов
- куча удобных надстроек:
  - versioned
  - max-version-history
  - use-source-capacity

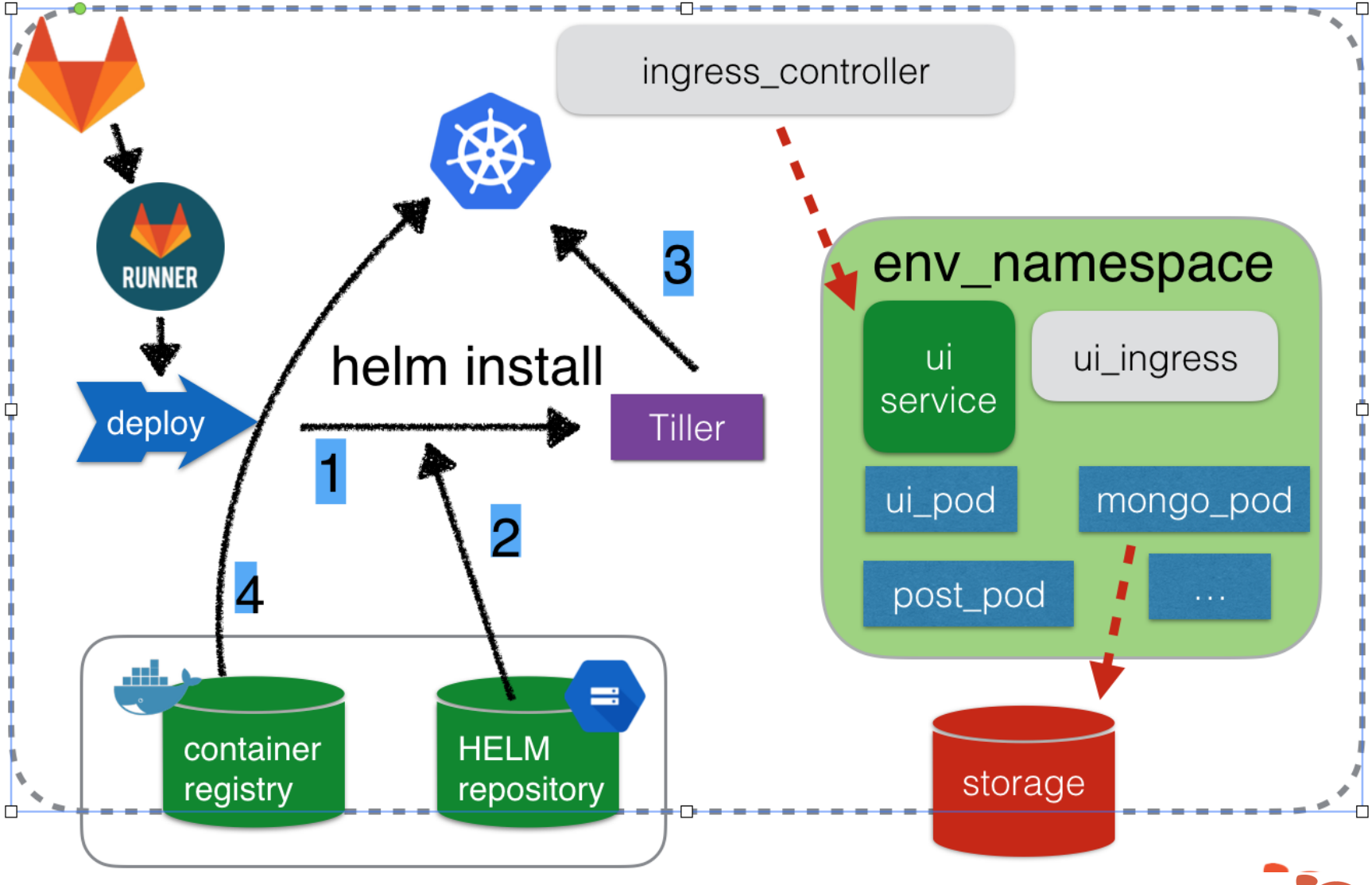
GitLab + Kubernetes

CI/CD инструментарий

Pipeline


Local
Developer PC
minikube
RUNNER
build
test
review
release
deploy
container
registry
HELM
repository

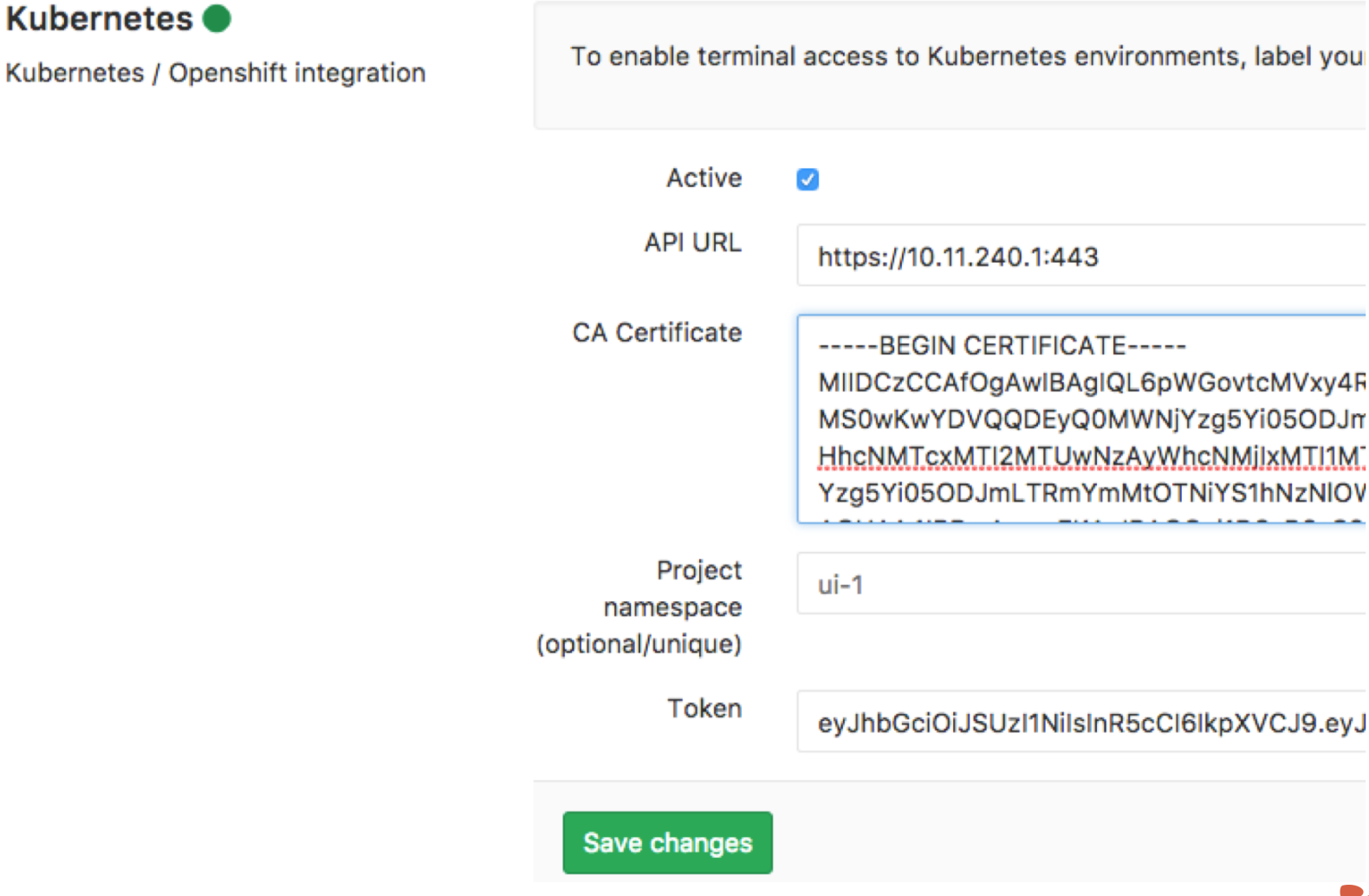
Pipeline



Что в ДЗ

- Разворачивание динамического окружения для feature ветки с удалением по кнопке
- CI/CD для Kubernetes от commit в репозиторий до релиза на статичные окружения

GitLab + Kubernetes



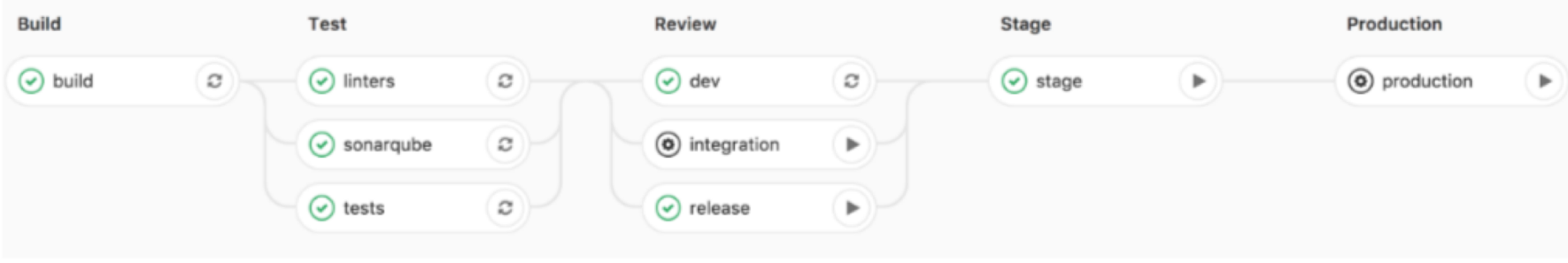
GitLab AutoDevOps

- Это набор шаблонов для настройки CI/CD стандартных веб-приложений
- Есть возможность не писать свои (генерирует собственные)
  - Dockerfile
  - gitlab-ci.yml
  - HELM Charts

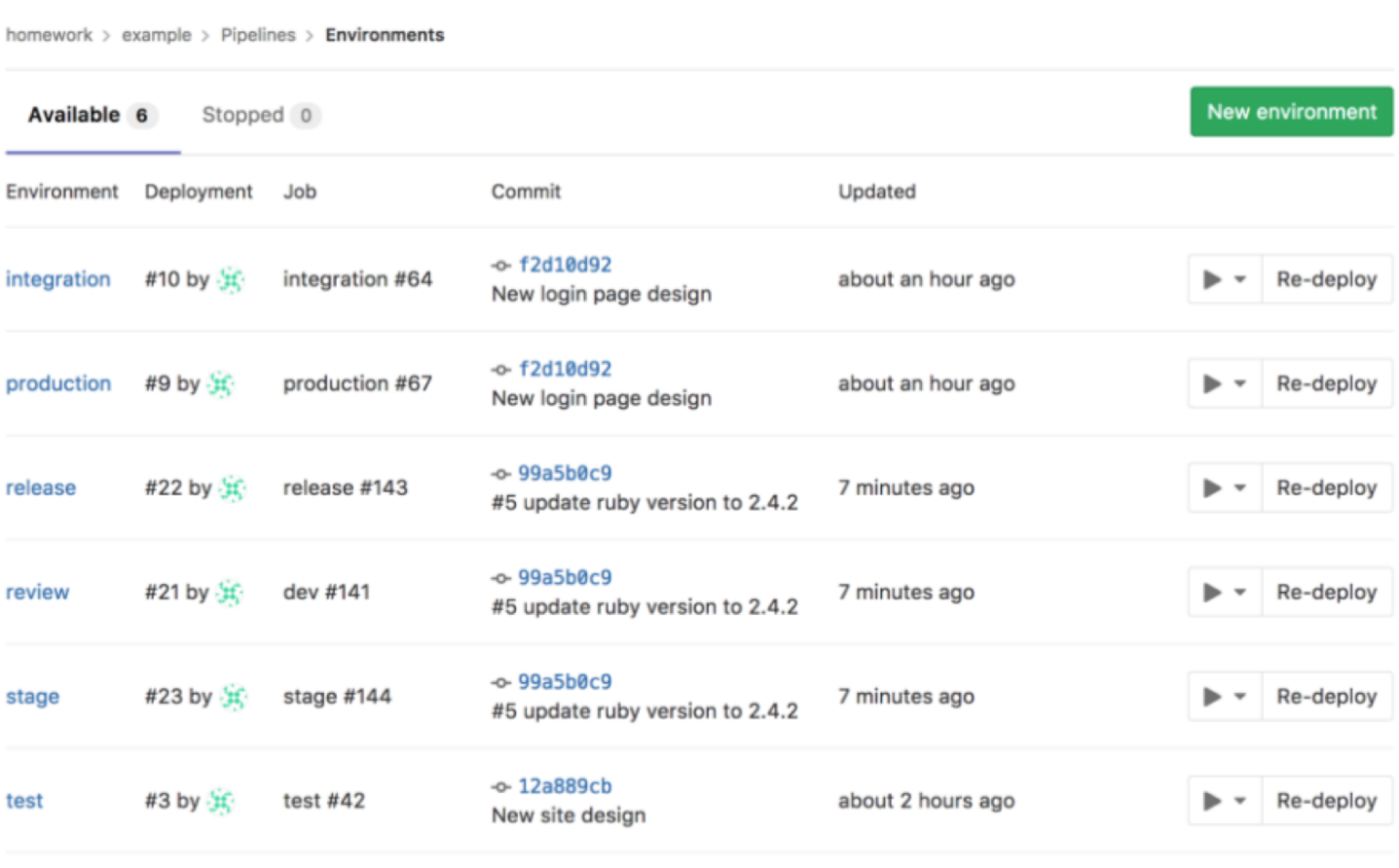
GitLab AutoDevOps

1. Auto Build
2. Auto Test
3. Auto Code Quality
4. Auto Review Apps
5. Auto Deploy
6. Auto Monitoring

Pipelines

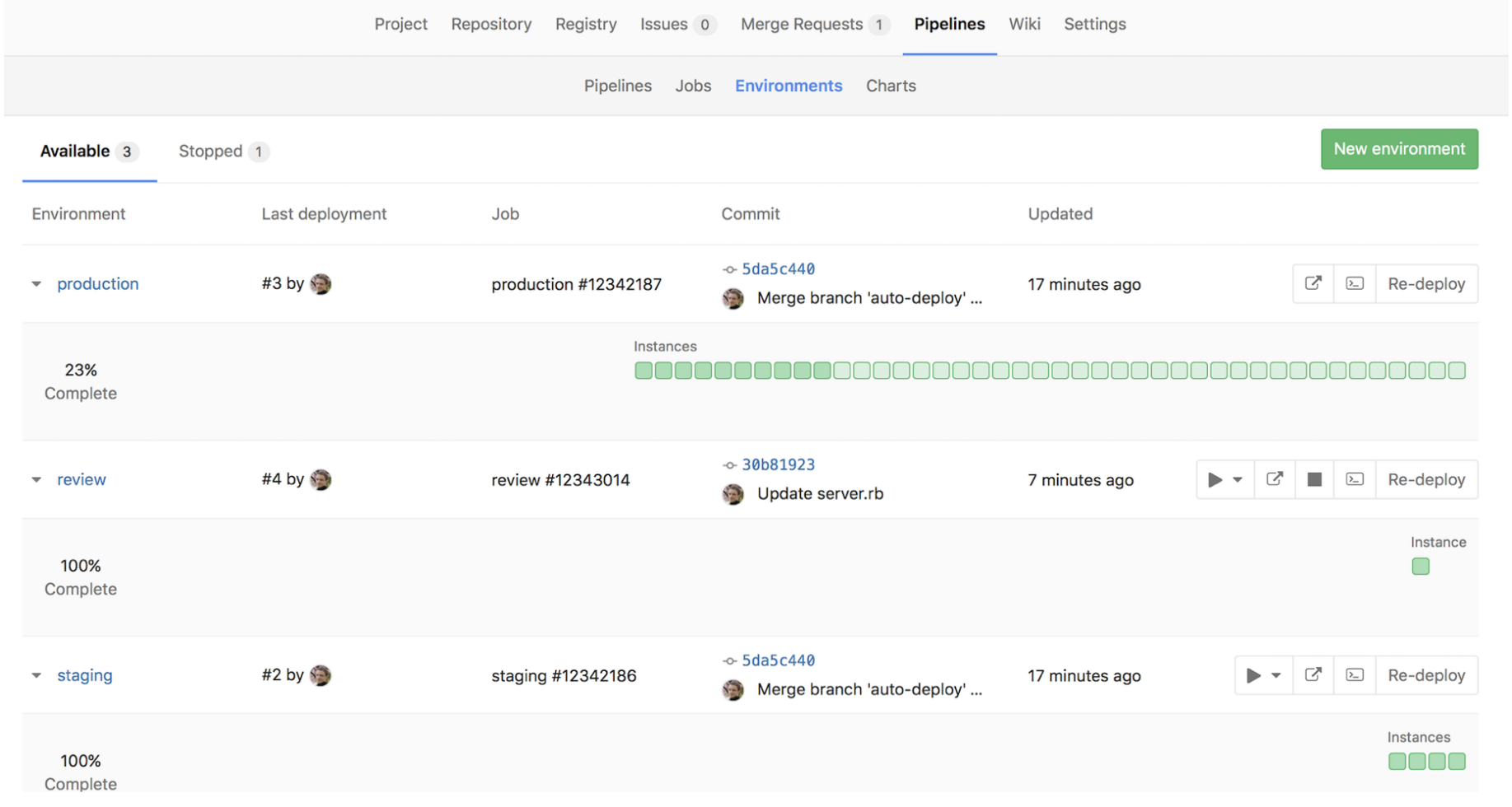

Build
Test
Review
Stage
Production
build
linters
sonarqube
tests
dev
integration
release
stage
production

Environments



Deploy Board

! Только в платных версиях



Деплоймент в Kubernetes

Стратегии Деплоя

- recreate: выключить все старые версии и запустить новые
- rolling release: выкатить новую версию, постепенно выключая старую
- blue/green: запустить новую версию параллельно старой
- canary: выпустить новую версию на ограниченное число пользователей

Recreate

recreate: выключить все старые версии и запустить новые. Всю работу за нас делает Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
spec:
  replicas: 3
  strategy:
    type: Recreate
```

Rolling Release

rolling release (default): выкатить новую версию, постепенно выключая старую. Среди надстроек:

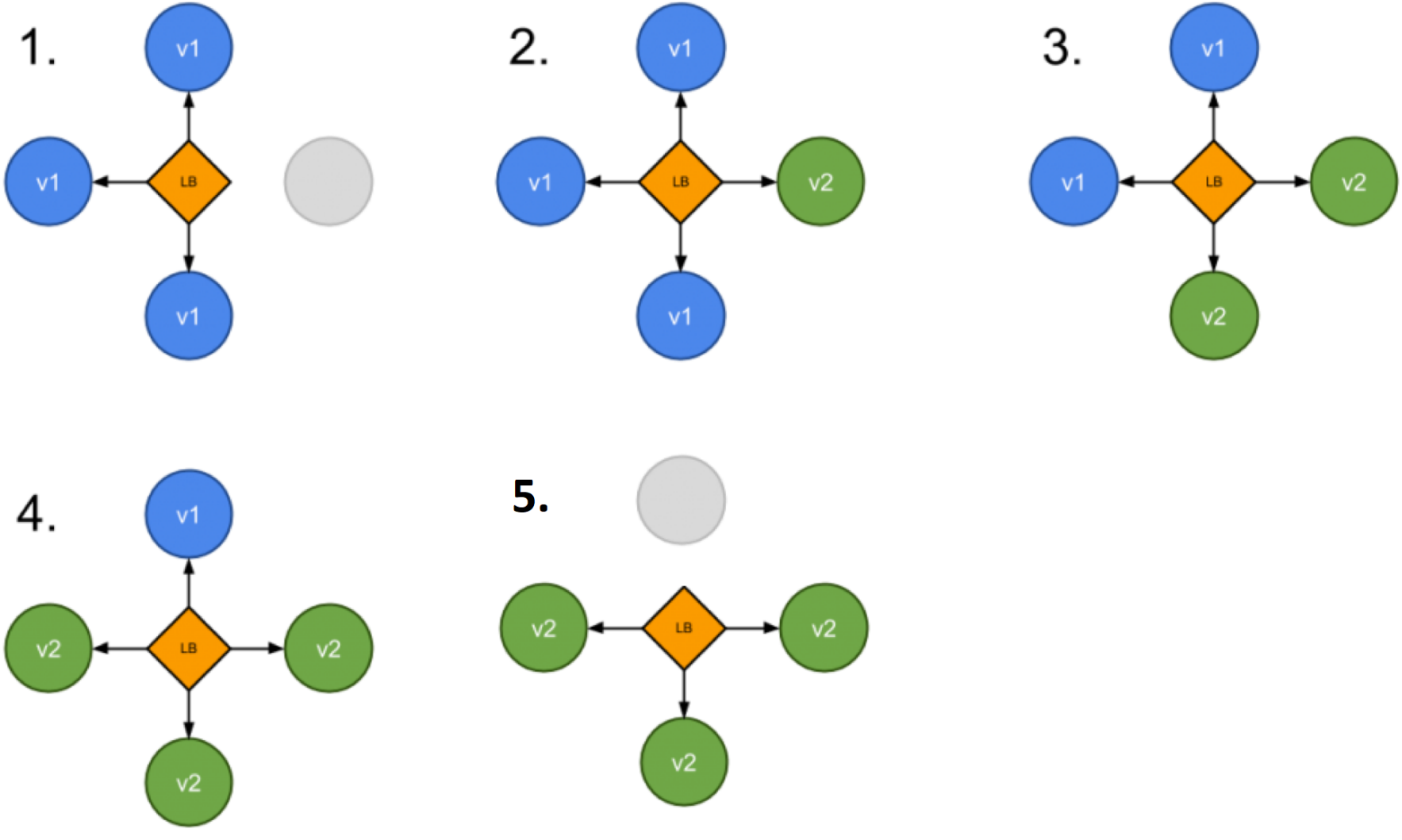
- maxSurge - на сколько подов моэно превысить указанное количество
- maxUnavailable - сколько подов могут быть недоступны

> ! Дефолтные значения для обоих - 25%

Rolling Release

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
```

Rolling Release



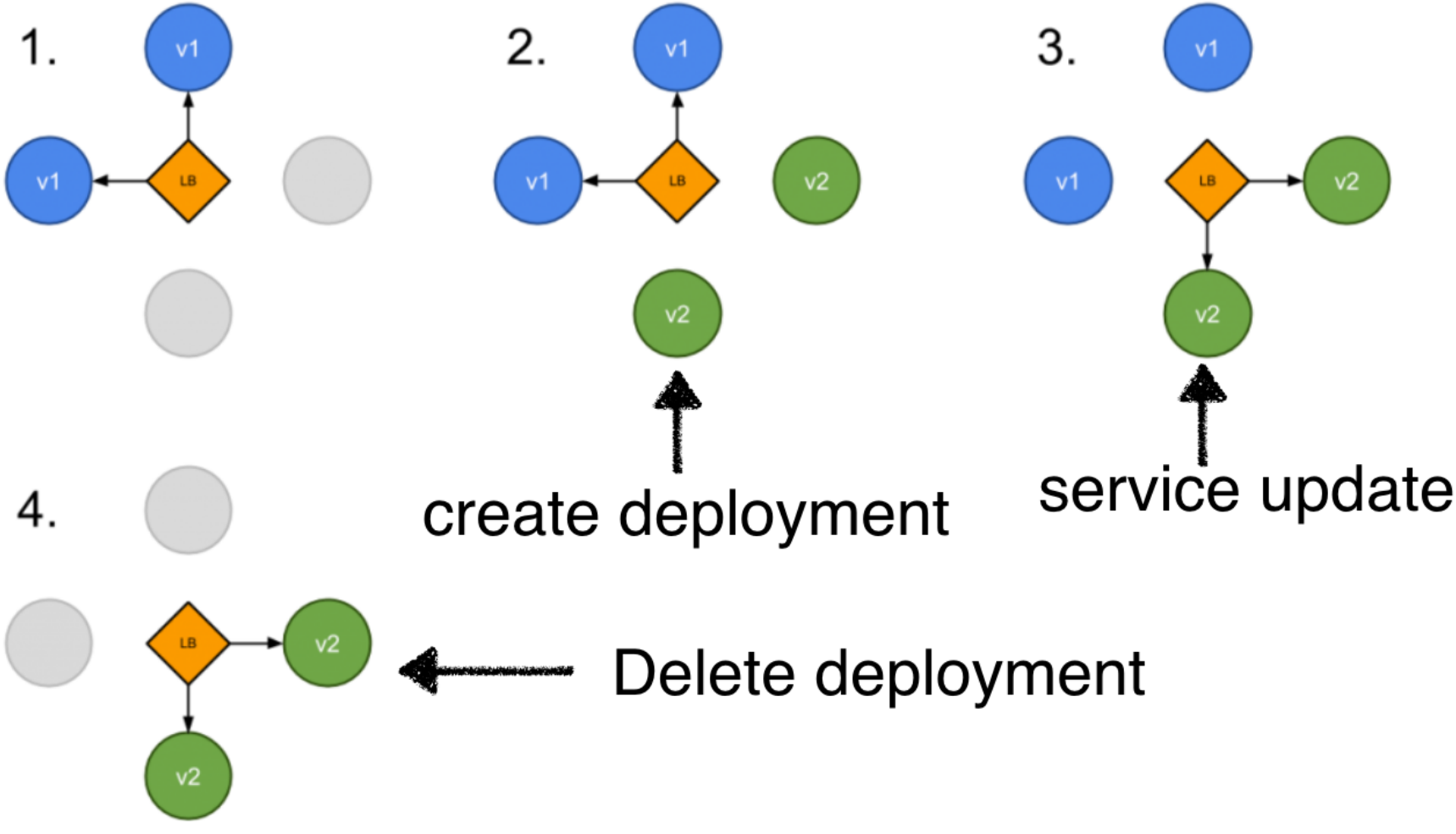
Blue/Green

Запустить новую версию параллельно старой. Управляется конфигурацией балансировщика.

```
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
    version: 1.0
spec:
…
```

```
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
    version: 2.0
spec:
…
```

Blue/Green


1.
v1
v1
LB
2.
v1
v1
LB
v2
v2
create deployment
3.
v1
v1
LB
v2
v2
service update
4.
LB
v2
v2
Delete deployment

Canary

Выпустить новую версию на ограниченное число пользователей

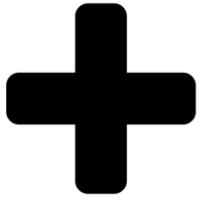
```
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: reddit
    component: ui
spec:
  type: NodePort
…
 selector:
    app: reddit
    component: ui
```

! Service перенаправляет трафик на все найденные POD-ы

Canary

Выпустить новую версию на ограниченное число пользователей

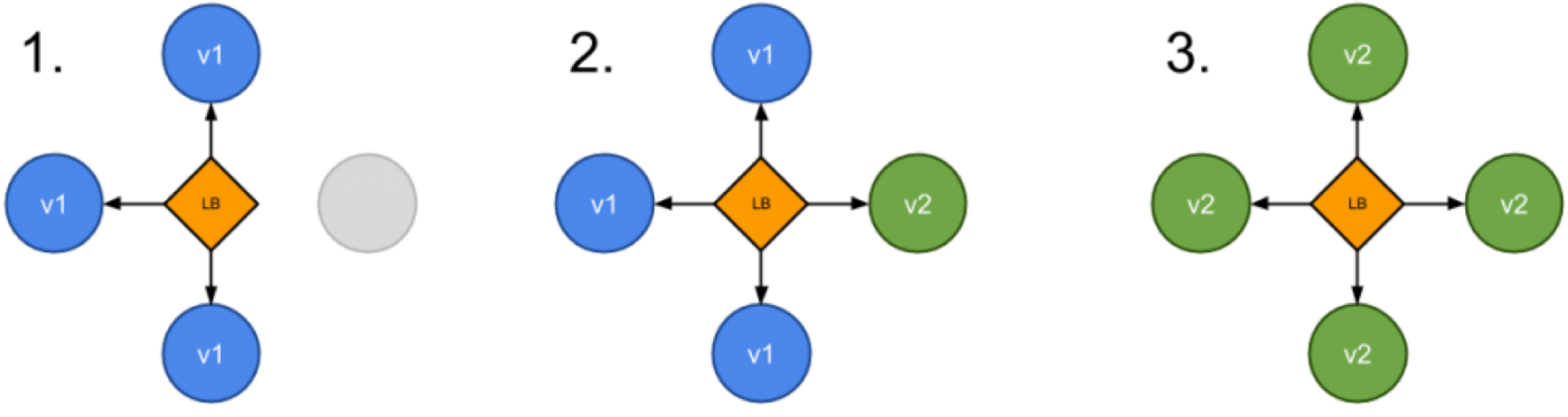
```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: ui-ver-1
  labels:
    app: reddit
    component: ui
spec:
  replicas: 3
  selector:
    matchLabels:
      app: reddit
      component: ui
      version: 1.0
```



```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: ui-ver-2
  labels:
    app: reddit
    component: ui
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reddit
      component: ui
      version: 2.0
```

Canary

Выпустить новую версию на ограниченное число пользователей



Ссылки по теме

- Сравнение Draft, Gitkube, Helm, Ksonnet, Metaparticle и Skaffold
- Helmsman
- Доклад Ивана Глушкова: Пакеты и пакетные менеджеры для k8s
- https://draft.sh
- GitLab AutoDevOps
- Spinnaker