

# CI/CD в Kubernetes

# 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** - пакетный менеджер для 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:

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

# Chart.yaml

Обязательный файл, содержащий информацию о Chart'е:

- ИМЯ
- описание
- версию чарта
- версию приложения
- различная метайнформация (разработчик, ссылки и т.д)

# Chart.yaml

```
name: application
version: 0.1.1
description: My app description
application maintainers:
  - name: Your Name
    email: test@mail.ru
appVersion: 0.7.0
...
```

Версии приложения и  
версия Chart'а могут  
различаться

The diagram consists of two black arrows. The first arrow originates from the text 'Версии приложения и версия Chart'а могут различаться' and points to the 'version: 0.1.1' line in the code block. The second arrow also originates from the same text and points to the 'appVersion: 0.7.0' line in the code block.



# Templates

Шаблоны манифестов Chart'a, содержатся в директории **templates/**

```
app/  
  Chart.yaml  
  templates/  
    deployment.yml  
    ingress.yml  
    service.yml
```

# Templates

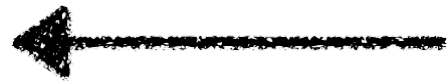
## service.yml

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

# Templates

## service.yml

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



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

**+ values.yml**

# Templates

## service.yml

---

apiVersion: v1

kind: Service

metadata:

name: {{ .Release.Name }}

labels:

app: {{ .Release.Name }}

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.yaml

```
---
service:
  type: NodePort
  port: 9292

metadata:
  app: my-app
```

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

# Go Templating

В основе Helm лежит [шаблонизатор Go](#) с 50+ встроенными функциями.

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

...

```
{{- else }}
```

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

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

```
food: {{ .food | upper | quote }}
```

Условия

Циклы

Функции

Пайплайны

# Управление зависимостями

Зависимости описываются в **requirements.yml**



```
my-app/  
  requirements.yml  
  charts/  
    app/  
      Chart.yaml  
      templates/
```

# Управление зависимостями

## requirements.yml

---

dependencies:

- name: ui  
version: 1.0.0  
repository: "file://../ui"
- name: back  
version: 2.0.1  
repository: "file://../back"
- name: worker  
version: 1.0.2  
repository: "file://../worker"
- name: mongo  
version: 3.2  
repository: "<http://example.com/charts>"



# 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 ([подробнее](#))

# Helm-client

**helm** - консольное приложение

- Используем при локальной работе с Chart'ами
- Управление репозиториями
- Отправка информации об установке Chart'ов (Releas'ax)
- Опрос информации о релизах
- Запрос на удаление или обновление релиза

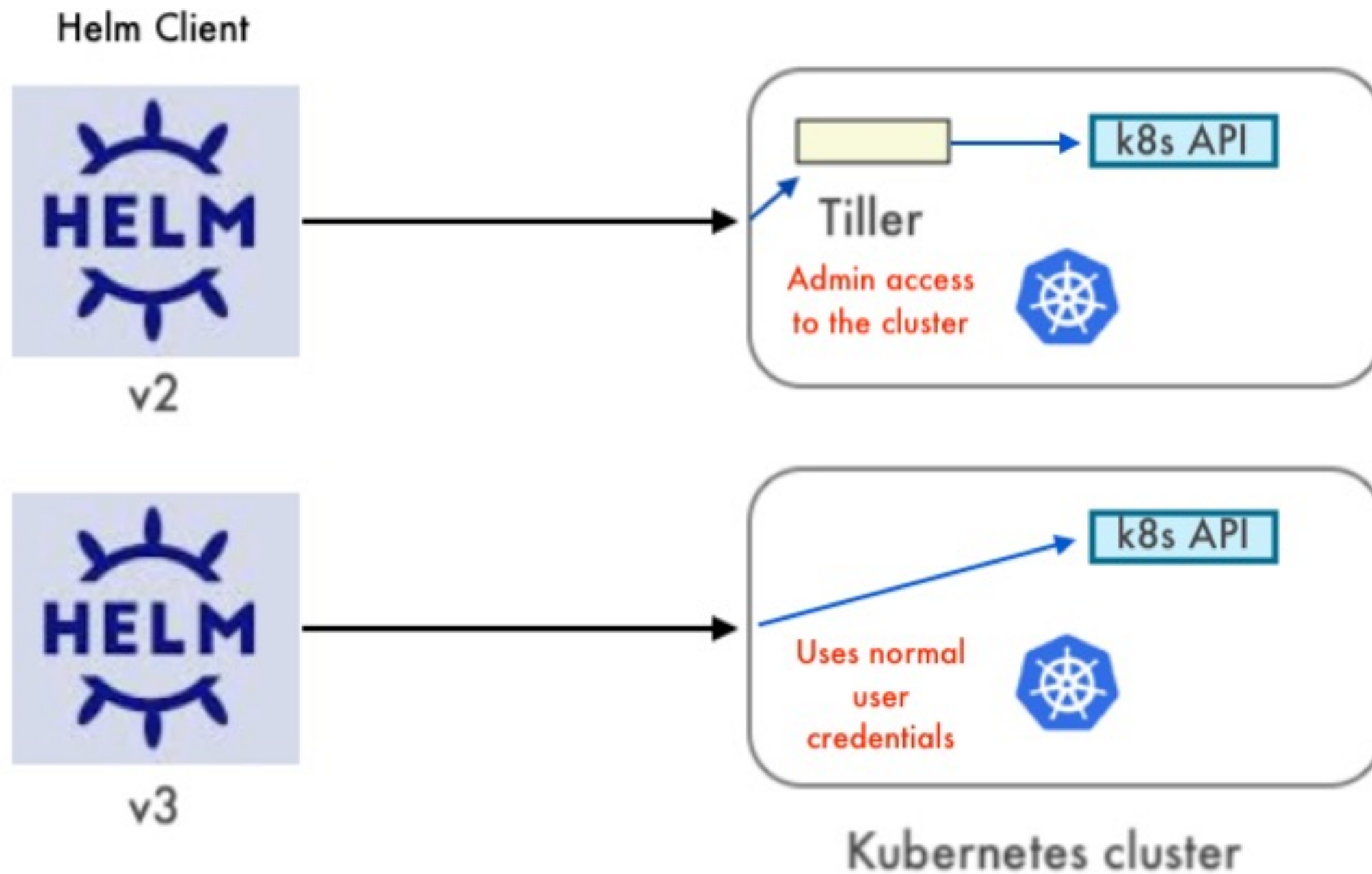
# Tiller (актуален для Helm v2)

**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 Release'a
- Обновление и удаление Release'ов

# Архитектура Helm



# Helm команды

# установка или обновление helm-чарта

```
helm upgrade --install -n dev -f clickhouse-helm/values.yaml dev-clickhouse  
./clickhouse-helm/ --debug -wait -history-max 2 --atomic
```

# установка helm-чарта

```
helm install --namespace=filebeat --values=filebeat-helm-values.yml filebeat  
elastic/filebeat
```

# добавление helm-репозитория

```
helm repo add prometheus-community https://prometheus-  
community.github.io/helm-charts
```

# удаление релиза

```
helm uninstall --namespace=filebeat filebeat
```

# список релизов

```
helm list
```

# запуск установки helm-чарта в режиме проверки

```
helm -n monitoring upgrade --install --dry-run -f values-main.yaml  
prometheus prometheus-community/kube-prometheus-stack
```

# скачать helm-чарт

```
helm pull bitnami/redis
```

# вывести сформированные манифесты

```
helm template ... --set imagePrefix=tgbot
```

# 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
my-app	2	Wed Nov 29 18:17:57 2017	DEPLOYED	my-app-0.1.1	default

имя  
релиза

ревизия  
(версия релиза)

Версия Chart'a

# Helm rollback

```
$ helm rollback my-app 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
my-app	3	Wed Nov 29 18:17:57 2017	DEPLOYED	my-app-0.1.0	default

# 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
- test

Политики удаления ресурсов (hook delete policy):

- before-hook-creation (удалить старые ресурсы хуков перед запуском нового hook, by default)
- hook-succeeded (удалить ресурсы хуков после успешного выполнения job)
- hook-failed (удалить ресурсы даже если job не сработал во время выполнения)

Подробнее: [https://helm.sh/docs/topics/charts\\_hooks/](https://helm.sh/docs/topics/charts_hooks/)



# Hooks

Пример использования Hook:  
если тесты прошли неуспешно, то откат релиза  
произойдет автоматически  
(при использовании параметра `--atomic`)!

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myapp
  annotations:
    "helm.sh/hook": post-install,post-upgrade
    "helm.sh/hook-delete-policy": before-hook-creation,hook-succeeded
spec:
  template:
    metadata:
      name: myapp-smoke-test
    spec:
      restartPolicy: Never
      containers:
      - name: tests
        image: test-image
        command: ['/bin/sh', '-c', '/test/run-test.sh']
```

# Тесты

- 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

Подробнее: [https://helm.sh/docs/topics/chart\\_tests/](https://helm.sh/docs/topics/chart_tests/)

# Недостатки HELM

- Kubernetes ничего не знает о **Chart**'ах
- Нет встроенной поддержки окружений
- Нет информативных логов о проблемах старта приложений
- Слабый Linter (пропускает много ошибок)
- При ошибках установки зачастую требуется вмешательство пользователя

# 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

# GitLab + Kubernetes

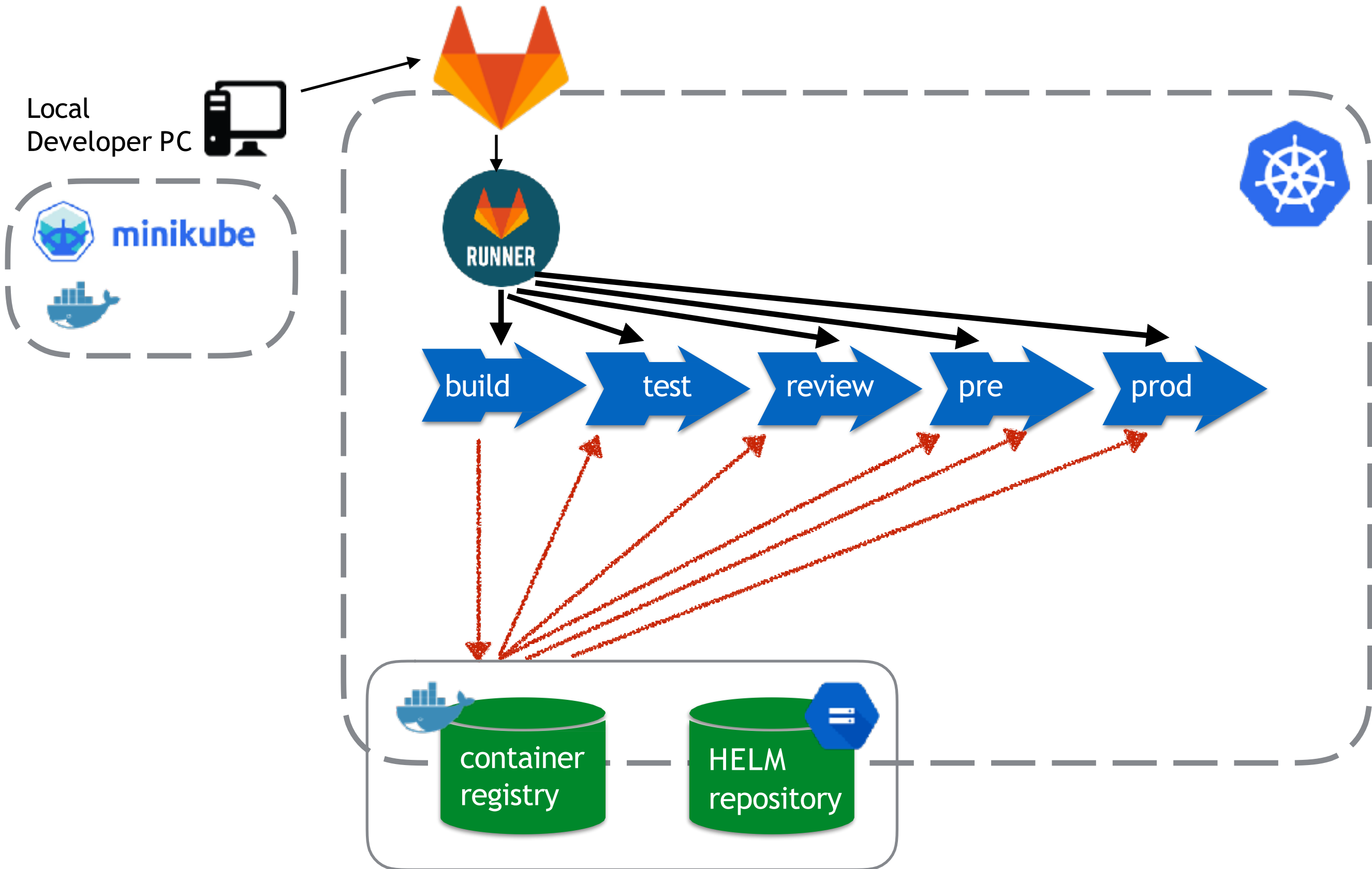
# CI/CD Инструментарий



# Environments

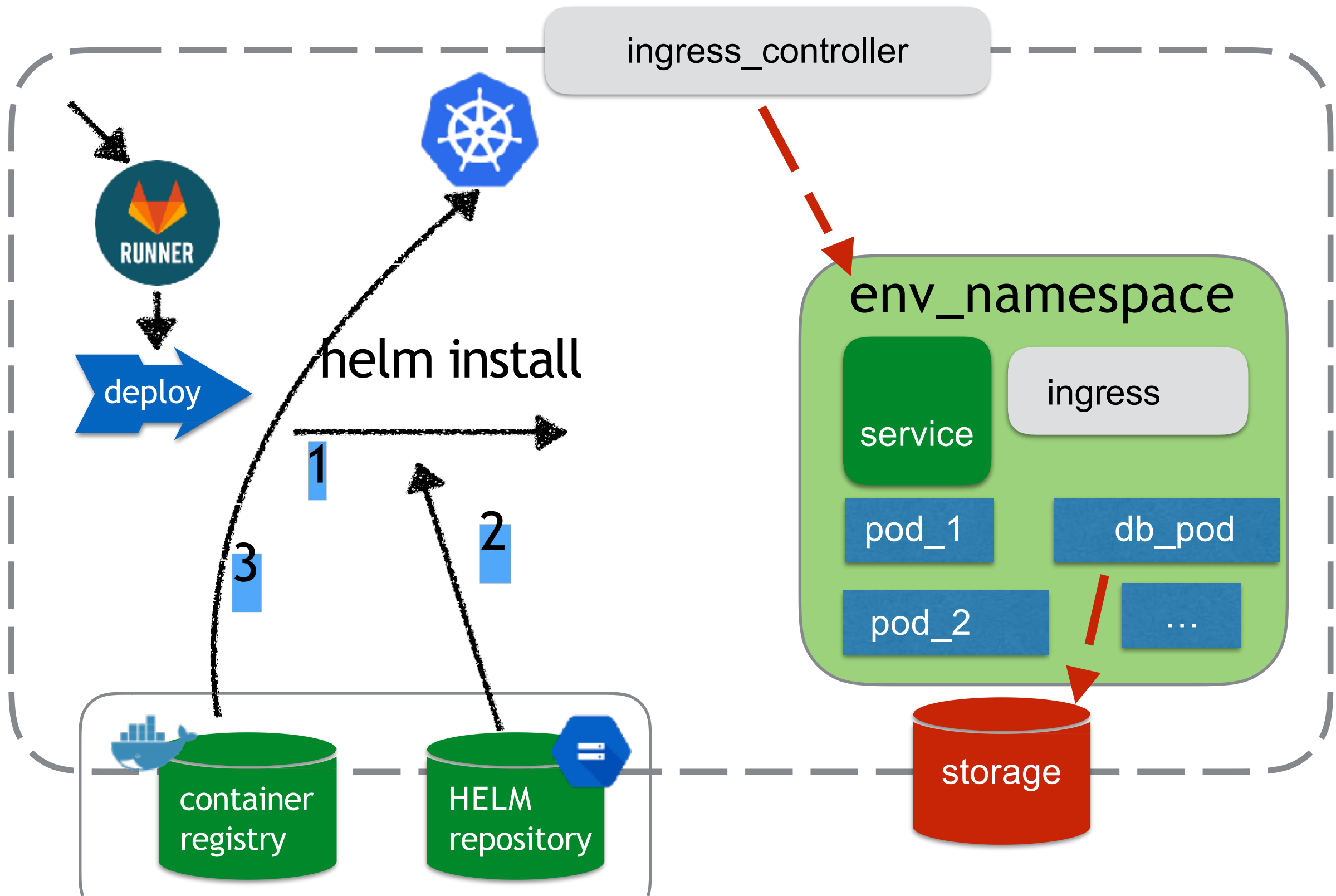
- Можно создать несколько виртуальных кластеров в рамках одного физического кластера.
- **Namespace** - один виртуальный кластер

# Pipeline





# Pipeline



























# Environments

homework > example > Pipelines > Environments

Available 6

Stopped 0

New environment

Environment	Deployment	Job	Commit	Updated	
integration	#10 by 	integration #64	 <a href="#">f2d10d92</a> New login page design	about an hour ago	  Re-deploy
production	#9 by 	production #67	 <a href="#">f2d10d92</a> New login page design	about an hour ago	  Re-deploy
release	#22 by 	release #143	 <a href="#">99a5b8c9</a> #5 update ruby version to 2.4.2	7 minutes ago	  Re-deploy
review	#21 by 	dev #141	 <a href="#">99a5b8c9</a> #5 update ruby version to 2.4.2	7 minutes ago	  Re-deploy
stage	#23 by 	stage #144	 <a href="#">99a5b8c9</a> #5 update ruby version to 2.4.2	7 minutes ago	  Re-deploy
test	#3 by 	test #42	 <a href="#">12a889cb</a> New site design	about 2 hours ago	  Re-deploy

# Pipelines



# Gitlab AutoDevops

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

# 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

# Деплой в Kubernetes

# Стратегии Деплоя

- **recreate**: выключить все старые версии и запустить новые
- **rolling update**: 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/v1
kind: Deployment
metadata:
  name: ui
  labels:
    app: my-app
    component: ui
spec:
  replicas: 3
  strategy:
    type: Recreate
```

← Меняем стратегию



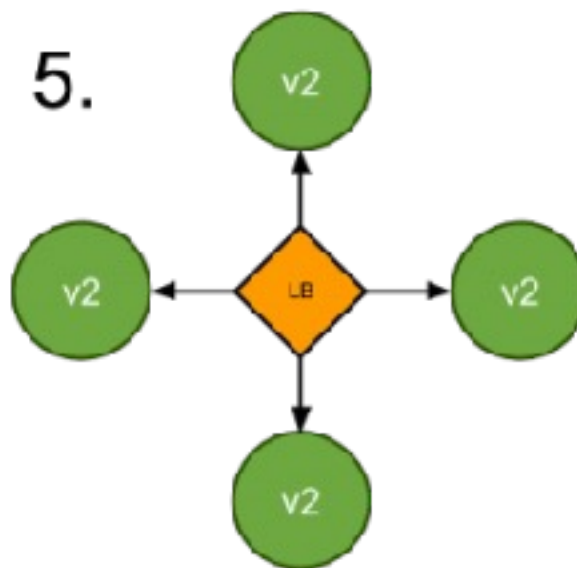
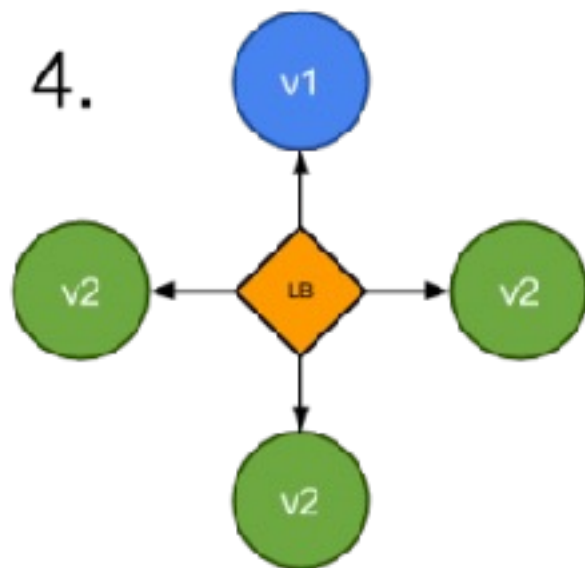
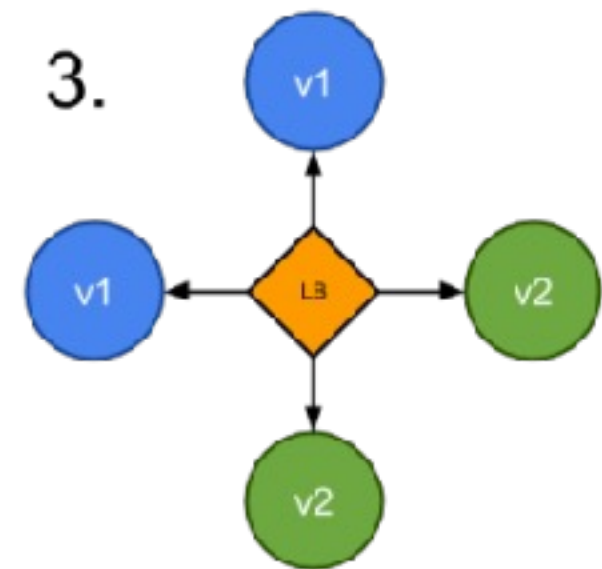
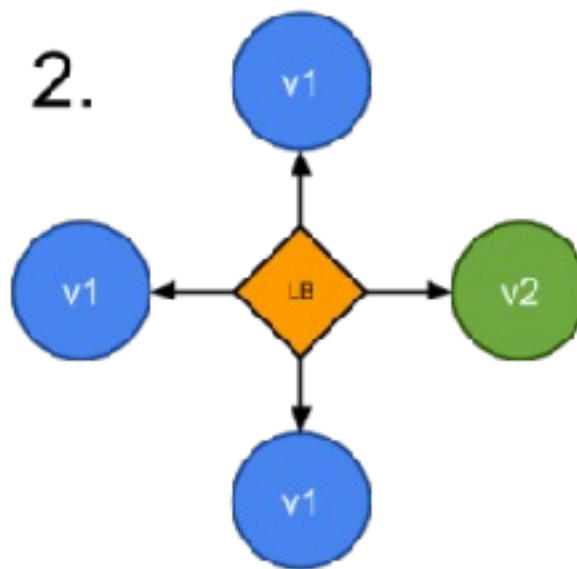
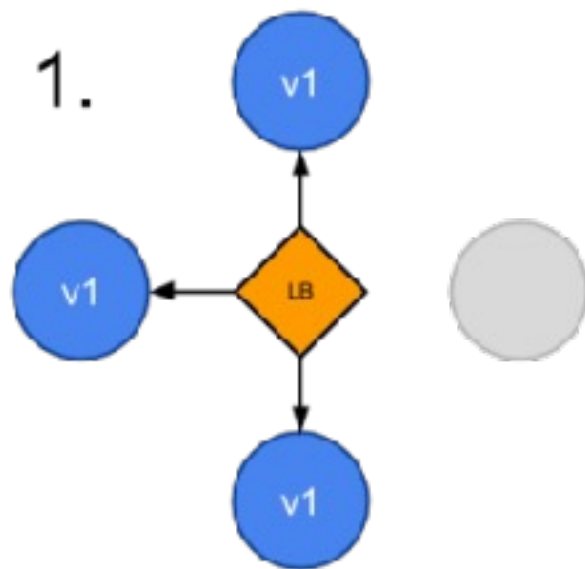
# Rolling Release

**rolling release (default):** выкатить новую версию, постепенно  
выключая старую

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ui
  labels:
    app: my-app
    component: ui
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
```

← Меняем стратегию

# Rolling Release

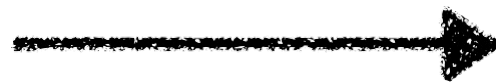


# Blue/Green

Запустить новую версию параллельно старой

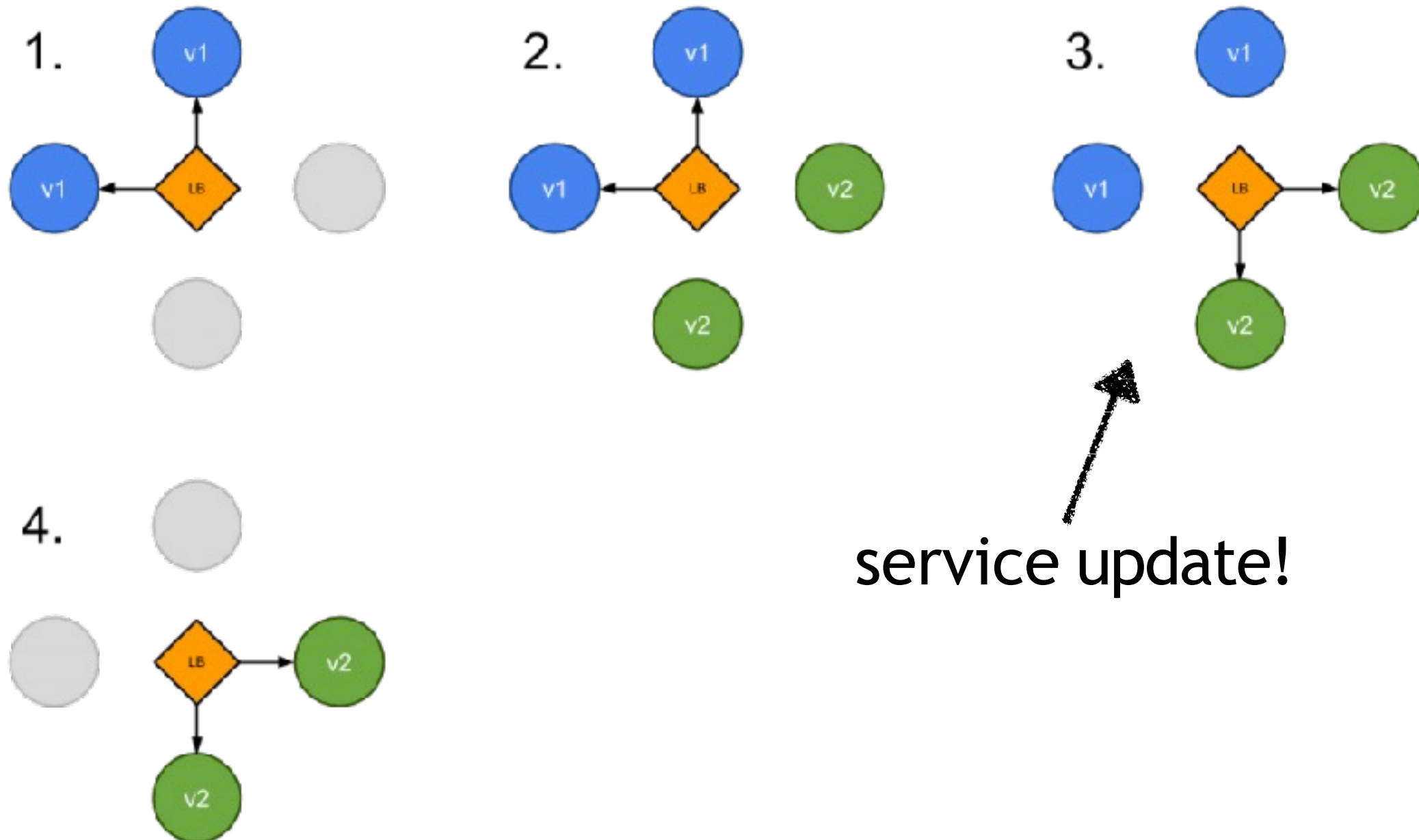
Управляется конфигурацией балансировщика или сервиса

```
---
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: my-app
    component: ui
    version: 1.0
spec:
...
```



```
---
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: my-app
    component: ui
    version: 2.0
spec:
...
```

# Blue/Green





## Выпустить новую версию на ограниченное число пользователей

```
---
apiVersion: v1
kind: Service
metadata:
  name: ui
  labels:
    app: my-app
    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/v1
kind: Deployment
metadata:
  name: ui-ver-1
  labels:
    app: my-app
    component: ui
spec:
  replicas: 3
  template:
    metadata:
      name: ui-pod
      labels:
        app: my-app
        component: ui
```

# Canary

## Выпустить новую версию на ограниченное число пользователей

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ui-ver-1
  labels:
    app: my-app
    component: ui
spec:
  replicas: 3
  template:
    metadata:
      name: ui-pod
      labels:
        app: my-app
        component: ui
```



```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ui-ver-2
  labels:
    app: my-app
    component: ui
spec:
  replicas: 1
  template:
    metadata:
      name: ui-pod
      labels:
        app: my-app
        component: ui
```

# Canary

Выпустить новую версию на ограниченное число пользователей

