ШПАРГАЛКА · HELM

# Helm: чарт, релиз и где он живёт

Как Helm добавляет сущность приложения в Kubernetes

12 мин чтения · обновлено 13 сентября 2026 г. · к ролику «Helm: зачем он нужен и что происходит внутри кластера»

ПОДЫЧ

Выжимка по Helm: чарт и релиз, где лежит состояние, что на самом деле делают upgrade и rollback, какие команды нужны в работе и на чём спотыкаются все.

В ШПАРГАЛКЕ



**prostodevops.ru/cheatsheets/helm**

Практика в настоящем кластере, бесплатно: prostodevops.ru/cheatsheets/helm/practice

ОТКУДА БОЛЬ

## В Kubernetes нет сущности приложения

**Самый скучный сервис в кластере — это Deployment, Service, Ingress, ConfigMap, Secret, HPA и ServiceAccount. Семь объектов, семь файлов. Связь между ними существует только в вашей голове и в имени папки.**

Кластер видит семь независимых записей. Что это одно приложение, знаете только вы.

Отсюда копипаста между окружениями. Дев, стейдж и прод — это те же семь файлов в трёх экземплярах, а отличаются в них строк пятнадцать: тег образа, реплики, домен, лимиты.

Отсюда же мусор в неймспейсах. Команды «обновить приложение» нет, есть только apply вот этих файлов. Снесли Deployment, про ConfigMap забыли — он висит год, пока на него случайно не наткнутся.

И чужой софт: ингресс-контроллер или оператор базы ставится копированием двух десятков манифестов, которые вы видите впервые.

ЧАРТ

**Форма**

Шаблоны манифестов плюс значения по умолчанию. Один комплект на все окружения.

VALUES

**Заполнение**

Чем именно заполнена форма в этот раз: реплики, образ, домен.

РЕЛИЗ

**Подписанный экземпляр**

Чарт, установленный в кластер под именем. Имя — ключ ко всему.

РЕВИЗИЯ

**Номер в журнале**

Каждый install, upgrade и rollback — новая запись. Только вперёд.

ПОДЫЧ

Держите эту четвёрку перед глазами: форма, заполнение, подписанный экземпляр, номер в журнале. Дальше всё выводится из неё.

ЧТО ОН ДЕЛАЕТ

## Три работы Helm

Helm почти всегда называют шаблонизатором YAML. Это одна треть правды, и из-за неё половина людей не понимает, зачем он нужен.

1

### Шаблонизатор

Один комплект файлов на все окружения. Отличия — в values, а не в копиях.

2

### Пакетный менеджер

Чужой софт ставится одной командой из реестра, как apt install.

3

### Менеджер релизов

У установленного приложения появляются имя, номер ревизии и история.

Ключевое — третье. Будь Helm только шаблонизатором, он проиграл бы kustomize. Выиграл он тем, что принёс в кластер понятие «приложение» и ведёт его учёт.

УСТРОЙСТВО

## Как устроен чарт и как значения доезжают до манифеста

| СЛОВО | ЧТО ЭТО | ГДЕ ЛЕЖИТ |
|---|---|---|
| **Чарт** | пакет: шаблоны, значения по умолчанию и паспорт с именем и версией | в гите или в OCI-реестре |
| **values** | файл ответов: сколько реплик, какой тег, какой домен | рядом с чартом или у вас в репозитории стенда |
| **Релиз** | чарт, установленный в кластер под своим именем | в кластере: объекты плюс секрет со слепком |
| **Ревизия** | номер версии релиза, растёт при каждом upgrade и rollback | в имени секрета: sh.helm.release.v1.web.v3 |

```
myapp/
  Chart.yaml  — паспорт: имя, version чарта, appVersion приложения
  values.yaml  — значения по умолчанию
  charts/  — зависимости — чужие чарты подчартами
  templates/  — шаблоны манифестов
    deployment.yaml
    service.yaml
    _helpers.tpl  — именованные куски, чтобы не копировать метки
```

```
# templates/deployment.yaml
replicas: {{ .Values.replicaCount }}
image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
```

1 **values.yaml внутри чарта**
умолчания автора

2 **ваш файл через -f**
values-prod.yaml, values-dev.yaml

3 **--set в командной строке**
перебивает всех

Практика: один чарт, рядом `values-dev.yaml` и `values-prod.yaml`. Ради этого копипаста и умирает. Обратная сторона зависимостей: чужие values протекают в ваши — подчарт настраивается через ключ с его именем.

ЯДРО

## Что происходит при helm install и где живёт состояние

1 **Собрал чарт**
подтянул зависимости

2 **Подставил значения**
чарт → -f → --set

3 **Один большой YAML**
всё приложение целиком

4 **Отправил в кластер**
от вашего имени, через API

5 **Записал слепок**
секретом рядом с подами

**Helm не хранит ничего у себя. Состояние лежит в самом кластере обычным секретом в том же неймспейсе: тип `helm.sh/release.v1`, имя `sh.helm.release.v1.<релиз>.v<ревизия>`. Внутри — чарт, values и отрендеренный манифест, сжатые gzip и закодированные base64.**

СЛЕДСТВИЕ 1

### Helm без кластера не знает ничего

Переставили систему — ничего не потеряли. Дали доступ коллеге — он видит то же самое.

СЛЕДСТВИЕ 2

### Откат — не пересборка

Предыдущий слепок физически лежит рядом. Rollback достаёт старый экземпляр из архива.

СЛЕДСТВИЕ 3

### Лимит около мегабайта

Секрет ограничен, и жирный чарт с кучей CRD туда не влезает. Знаменитая боль.

```
Error: INSTALLATION FAILED: cannot re-use a name that is still in use
```

Релиз с таким именем уже стоит в этом неймспейсе — install ставит только новое.

**Что делать.** `helm upgrade --install` вместо `install`: команда одинаково работает и в первый раз, и в сотый.

```
Error: rendered manifests contain a resource that already exists
```

Объект с таким именем уже есть в кластере, но Helm его не ставил — например, его завели руками.

**Что делать.** Удалить чужой объект или передать релизу владение метками app.kubernetes.io/managed-by и аннотациями meta.helm.sh.

ПОДЫЧ

В Helm 2 состояние держал Tiller — серверный компонент с правами почти на всё. Кто дотянулся до Tiller, тот управлял кластером. В Helm 3 его выкинули целиком, клиент ходит в API от вашего имени. Слой убрали, потому что он был дырой.

РАЗОБЛАЧЕНИЕ

## upgrade и rollback честно

`helm upgrade` не накатывает новые файлы поверх. Он сравнивает три вещи — и поэтому замечает правку через `kubectl edit`.

**Прошлый слепок**
что Helm ставил в прошлый раз

**Новый манифест**
что получилось из чарта сейчас

**Живой кластер**
что реально лежит там сейчас

**Three-way merge**
в Helm 4 — через server-side apply, решает сам API-сервер

```
Error: UPGRADE FAILED: another operation (install/upgrade/rollback) is in progress
```

Прошлую операцию оборвали на середине: релиз завис в статусе pending-upgrade, и новый upgrade к нему не подступится.

**Что делать.** Посмотреть `helm history РЕЛИЗ` и откатиться на последнюю живую ревизию: `helm rollback РЕЛИЗ N`.

```
Error: UPGRADE FAILED: timed out waiting for the condition
```

Был флаг ожидания, и поды не пришли в готовность за отведённое время. Сам релиз при этом уже записан.

**Что делать.** Смотреть не на Helm, а на поды: `kubectl get pods`, `kubectl describe pod`. Helm здесь только гонец.

МИФ

### helm rollback — это машина времени

Нет. Откат возвращает манифест, а не состояние мира: он не вернёт мигрировавшие данные, не отменит применённую миграцию базы, не поднимет удалённый PVC. И сам откат создаёт новую ревизию, а не отматывает счётчик назад.

РАБОЧИЙ НАБОР

## Команды, которые реально нужны

| КОМАНДА | ЧТО ДЕЛАЕТ | КОГДА |
|---|---|---|
| `helm repo add ИМЯ URL && helm repo update` | подключить каталог чартов | один раз перед чужим софтом |
| `helm search repo ИМЯ` | найти чарт в подключённых репозиториях | выбор версии |
| `helm show values ЧАРТ` | посмотреть все настройки чарта | перед первой установкой |
| `helm template РЕЛИЗ ./чарт -f values.yaml` | отрендерить YAML локально, не касаясь кластера | отладка шаблонов |
| `helm install РЕЛИЗ ./чарт -n NS --create-namespace` | поставить первый релиз | неймспейс сам не появится |
| `helm upgrade --install РЕЛИЗ ./чарт -f values.yaml` | обновить или поставить, если ещё нет | CI: команда идемпотентна |
| `helm diff upgrade РЕЛИЗ ./чарт` | показать, что изменится в кластере (плагин) | перед любым upgrade на проде |
| `helm list -A` | перечислить релизы во всех неймспейсах | посмотреть, что вообще стоит |
| `helm status РЕЛИЗ` | показать состояние релиза и заметки | после установки |
| `helm history РЕЛИЗ` | лента ревизий | перед откатом |
| `helm rollback РЕЛИЗ N` | вернуть манифест ревизии N новой ревизией | сломали upgrade |
| `helm get values РЕЛИЗ` | с какими значениями стоит | посмотреть, какие значения заданы |
| `helm uninstall РЕЛИЗ` | удалить приложение целиком | снести приложение целиком |

Где лежит слепок: `kubectl get secret -l owner=helm -A`. Посмотреть внутрь — `kubectl get secret sh.helm.release.v1.web.v1 -o jsonpath='{.data.release}' | base64 -d | base64 -d | gunzip`.

ГДЕ СПОТЫКАЮТСЯ ВСЕ

# Грабли

### 1 Нажал upgrade на проде, не посмотрев diff

**Почему.** Upgrade сравнивает три источника, и результат не всегда совпадает с ожиданием — особенно после правок руками.

**Что делать.** Правило: `helm template` и `helm diff` перед апгрейдом. Без diff на прод не жмём.

### 2 Релиз залип в pending-upgrade

**Почему.** Команду прервали (Ctrl+C, упал CI), и в слепке остался промежуточный статус. Следующий upgrade отказывается работать.

**Что делать.** Посмотреть `helm history`, откатиться на последнюю успешную ревизию или снять залипший секрет ревизии.

### 3 --set портит числа и ключи с точками

**Почему.** Число 3 превращается в строку, а точка в имени ключа разбирается как вложенность.

**Что делать.** В CI — значения файлами через `-f`. `--set` оставить для разовых экспериментов.

### 4 Неймспейс сам не создался

**Почему.** Helm не заводит неймспейс без флага. И одинаковое имя релиза в разных неймспейсах — это два разных релиза.

**Что делать.** `--create-namespace` при install и всегда явный `-n`.

### 5 CRD не обновились при upgrade

**Почему.** Helm ставит CRD один раз и не трогает при апгрейде. Это осознанное решение.

**Что делать.** Обновлять CRD отдельным шагом до upgrade, как рекомендует автор чарта.

### 6 Секреты в values

**Почему.** values целиком лежат в том самом секрете со слепком, доступном всем, кто читает секреты неймспейса.

**Что делать.** Секреты — отдельным объектом или внешним менеджером, в values — только ссылка на него.

АКТУАЛЬНОЕ

## Helm 4, судьба Helm 3 и когда Helm не нужен

**ноябрь 2025**
Вышел Helm 4

**сентябрь 2026**
Последний функциональный релиз Helm 3

**февраль 2027**
Конец поддержки Helm 3 — дальше только заплатки по безопасности до этой даты

### Что поменялось в Helm 4

Применение через server-side apply. Плагины стали обычными расширениями и умеют собираться в WebAssembly. Умное ожидание готовности вместо таймера. Часть флагов переименовали: --atomic теперь --rollback-on-failure, старые имена работают с предупреждением — на них и ломаются пайплайны.

### Что не поменялось

Сами чарты переписывать не нужно: что работало на третьей версии, работает и на четвёртой. Отдельные репозитории чартов с индексом уходят, основной способ раздачи — OCI-реестры образов.

ПОДЫЧ

Когда Helm не нужен: три манифеста на пет-проекте — хватит kubectl. Нужна только подстановка значений без пакетов и истории — kustomize проще и встроен в kubectl. Живёте в GitOps с ArgoCD — там Helm часто только рендерер, а состоянием владеет Argo.

СОБЕС

# Как это спрашивают на собеседовании

**? Чем чарт отличается от релиза?**

Чарт — форма: шаблоны плюс значения по умолчанию. Релиз — подписанный экземпляр: чарт, установленный в кластер под именем, с историей ревизий. Из одного чарта можно поставить сколько угодно релизов.

**? Где Helm хранит состояние? Что было в Helm 2?**

В секрете в неймспейсе релиза, тип helm.sh/release.v1, по одному на ревизию. В Helm 2 состояние держал Tiller — компонент в кластере с широкими правами; в Helm 3 его убрали, клиент ходит в API напрямую.

**? Почему helm rollback не вернёт данные?**

Откат возвращает манифест из старого слепка, а не состояние мира: миграции базы, данные в PVC, внешние сервисы он не трогает. И создаёт новую ревизию, а не отматывает историю.

**? Что такое three-way merge в upgrade?**

Сравнение прошлого слепка, нового манифеста и живого состояния кластера. Поэтому правка через kubectl edit замечается. В Helm 4 слияние делает API-сервер через server-side apply.

**? Как обновить CRD у чарта?**

Helm ставит CRD только при install и не трогает при upgrade. Обновлять их отдельным шагом до апгрейда — обычно kubectl apply на файлы из папки crds чарта.

ЧЕТЫРЕ ВОПРОСА

# Проверь себя

**1. Где Helm хранит информацию о том, что он установил?**

**Ответ:** В секрете в том же неймспейсе, что и релиз

Секрет типа helm.sh/release.v1, имя sh.helm.release.v1.<релиз>.v<ревизия>. Tiller был в Helm 2 и выкинут в Helm 3.

**2. После helm rollback web 1 что покажет helm history web, если до отката было две ревизии?**

**Ответ:** Три ревизии

Откат создаёт новую ревизию с содержимым старой, а не отматывает счётчик. История только растёт.

**3. Кто побеждает, если один ключ задан и в values.yaml чарта, и в файле через -f, и через --set?**

**Ответ:** --set

Приоритет растёт слева направо: чарт → файл → командная строка. --set перебивает всех.

**4. Что Helm делает с CRD при helm upgrade?**

**Ответ:** Не трогает: ставит один раз при install

CRD ставятся один раз и при апгрейде не обновляются — это осознанное решение и вечный источник сюрпризов.