

The background features a complex network of blue and orange nodes connected by thin lines, creating a web-like structure. The nodes vary in size, and the lines are thin and light blue. The overall color scheme is dark blue with accents of orange and light blue.

СЛЁРМ

Helm

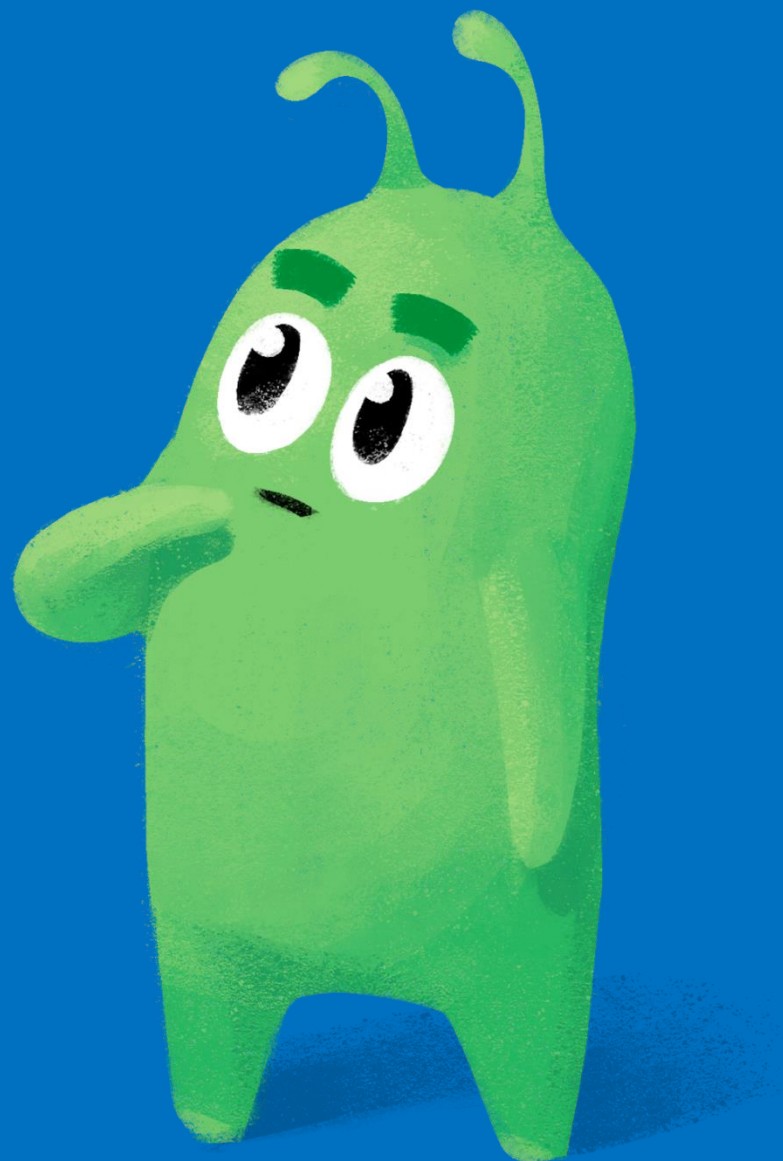
Марсель Ибраев
СТО Слёрм

План

- Почему Helm?
- Основы работы с Helm
- Создаем свой чарт в Helm
- Advanced things

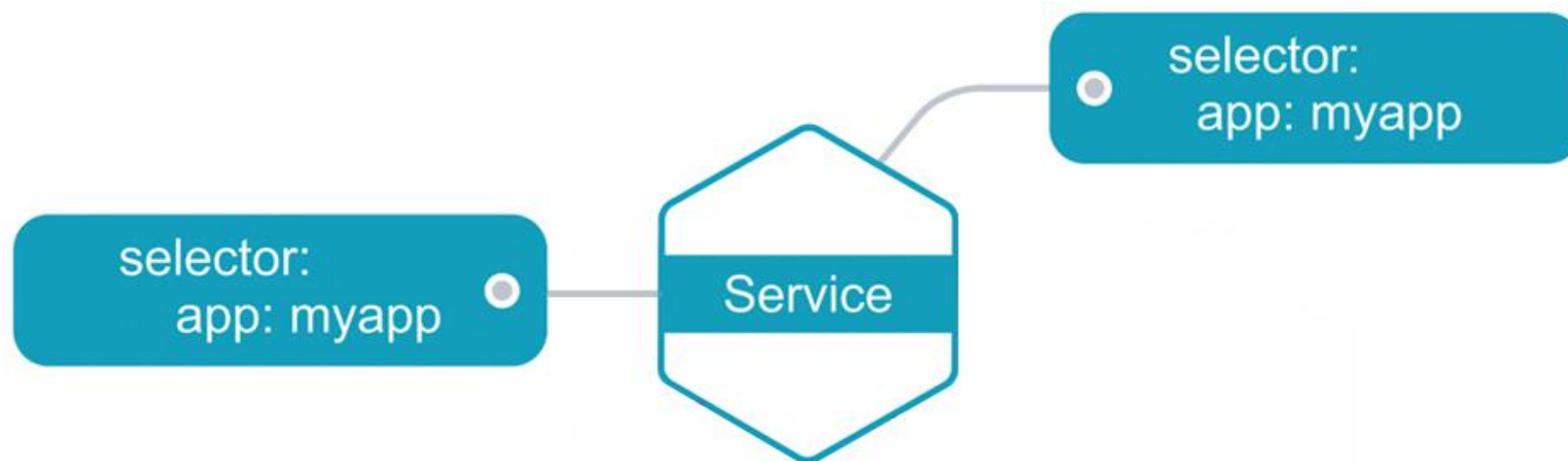


Как деплоить приложение в кластер?



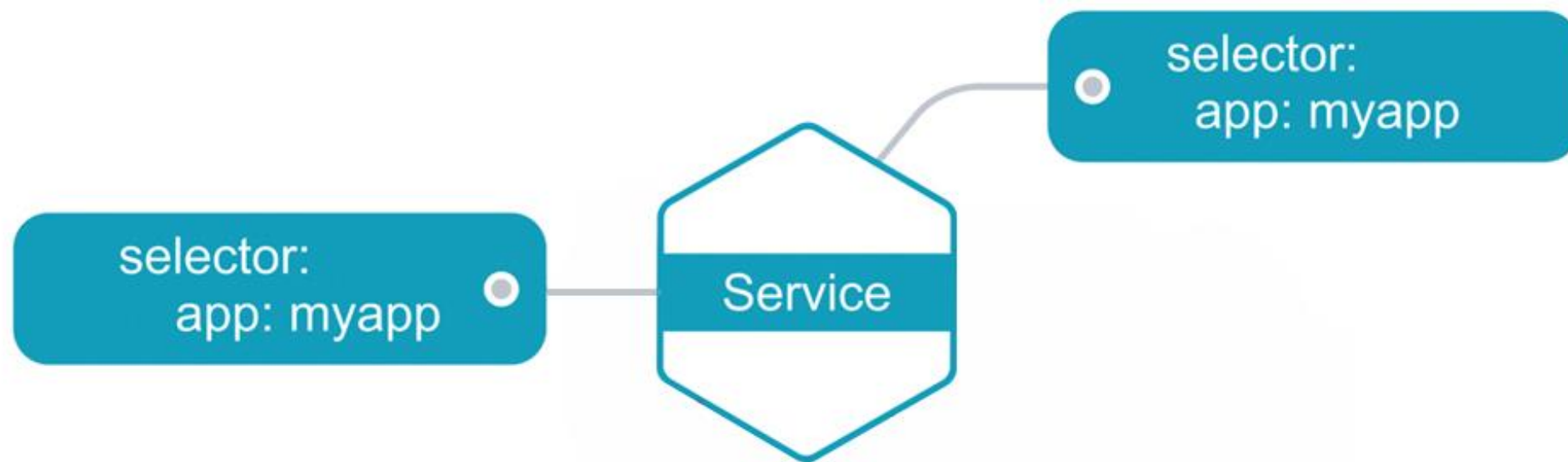
■ Темплэйтируем наше приложение

- Sed / Envsubst



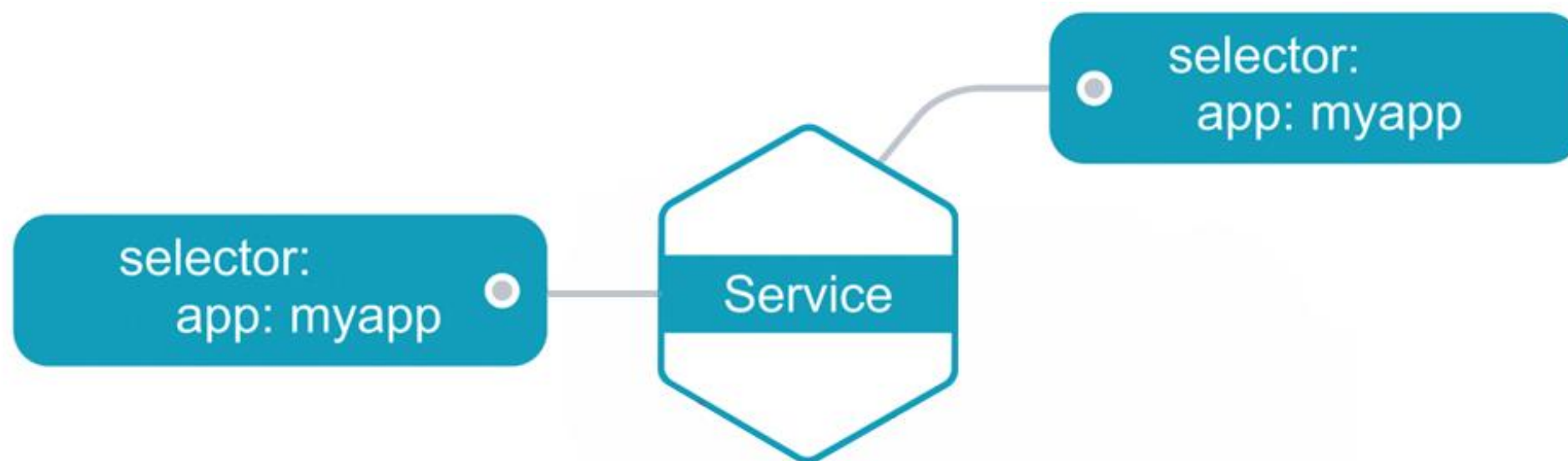
■ Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible



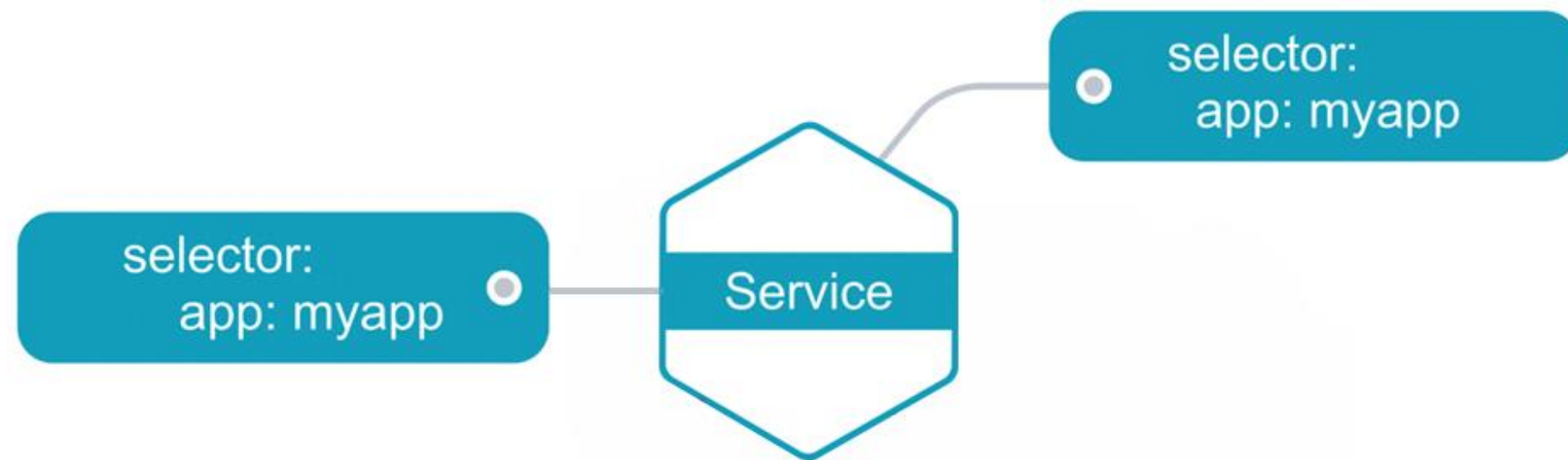
■ Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible
- Kubectl based



■ Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible
- Kubectl based
- Helm



Почему Helm?

- «Пакетный менеджер»



Почему Helm?

- «Пакетный менеджер»
- CNCF



Почему Helm?

- «Пакетный менеджер»
- CNCF
- Декларативный



Почему Helm?

- «Пакетный менеджер»
- CNCF
- Декларативный
- Есть важные фичи для построения CD
 - Watch
 - Rollback



Почему Helm?

- «Пакетный менеджер»
- CNCF
- Декларативный
- Есть важные фичи для построения CD
 - Watch
 - Rollback
- Система плагинов



Пакет

- Набор темплейтированных манифестов
- Файл со значениями переменных
- Мета



.tgz

=

Чарт

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
  labels:
    app: {{ .Chart.Name }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: {{ .Release.Name }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Chart.Name }}
      release: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app: {{ .Chart.Name }}
        chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
        release: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Release.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          ports:
            - containerPort: {{ .Values.service.port }}
```

values.yaml:

```
image:  
  repository: nginx  
  tag: stable
```

```
replicas: 2
```

```
service:  
  port: 80
```

Основы работы с Helm

- *helm search* – поиск чарта
- *helm install* – установка чарта
- *helm upgrade* – обновление чарта
- *helm get* – скачать чарт
- *helm show* – показать инфу о чарте
- *helm list* – список установленных чартов
- *helm uninstall* – удалить чарт



Деплой приложения

- `helm search hub kube-ops`
- `helm show values stable/kube-ops-view > values.yaml`
- `helm install ops-view stable/kube-ops-view \`
`--namespace=kube-system -f values.yaml`
- `helm ls --namespace kube-system`

■ Что внутри

- `helm pull stable/kube-ops-view`
- `tar -zxvf kube-ops-view-XX.YY.tgz`
- `cd kube-ops-view/`

■ Пишем свой чарт

1. Добавляем темплейты в labels

■ Пишем свой чарт

1. Добавляем темплейты в labels
2. Находим https://helm.sh/docs/topics/chart_best_practices/labels/

■ Пишем свой чарт

1. Добавляем темплэйты в labels
2. Находим https://helm.sh/docs/topics/chart_best_practices/labels/
3. Добавляем темплэйты в image

■ Пишем свой чарт

1. Добавляем темплэйты в labels
2. Находим https://helm.sh/docs/topics/chart_best_practices/labels/
3. Добавляем темплэйты в image
4. Добавляем темплэйты в реплики

■ Пишем свой чарт

1. Добавляем темплэйты в labels
2. Находим https://helm.sh/docs/topics/chart_best_practices/labels/
3. Добавляем темплэйты в image
4. Добавляем темплэйты в реплики
5. Добавляем темплэйты в ресурсы

■ Пишем свой чарт

1. Добавляем темплейты в labels
2. Находим https://helm.sh/docs/topics/chart_best_practices/labels/
3. Добавляем темплейты в image
4. Добавляем темплейты в реплики
5. Добавляем темплейты в ресурсы
6. Добавляем темплейты в env

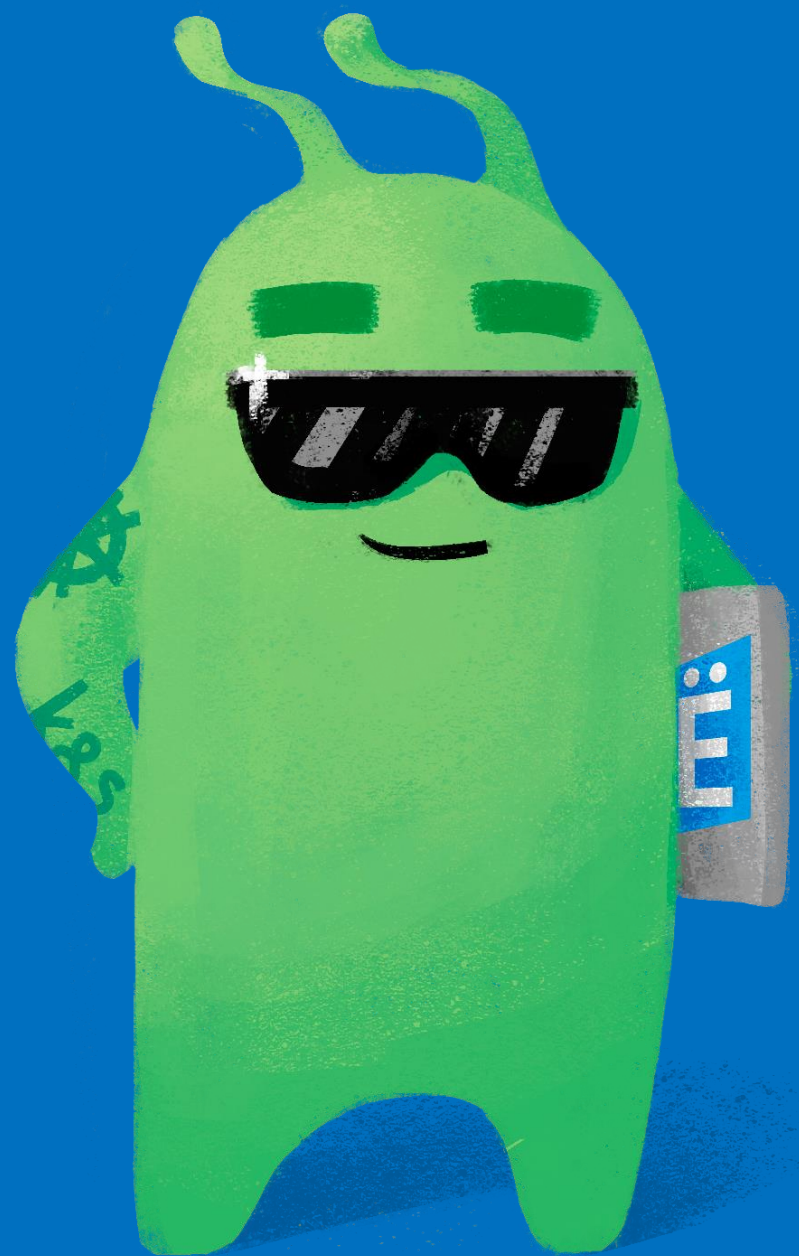
■ Пишем свой 100ый чарт

- Узнаем про команду `helm create chart_name`

■ Пишем свой 100ый чарт

- Узнаем про команду `helm create chart_name`
- Узнаем что можно создавать свои стартеры

Advanced level



■ Тестирование релиза

1. Создаем папку *templates/tests/*
2. Кладем туда манифесты объектов k8s которые будут тестить релиз
3. Манифесты должны содержать аннотацию *helm.sh/hook: test*
4. Запускаем в CI *helm test <release name>*

```

apiVersion: batch/v1
kind: Job
metadata:
  name: "{{ .Release.Name }}-credentials-test"
  annotations:
    "helm.sh/hook": test
spec:
  template:
    spec:
      containers:
        - name: main
          image: {{ .Values.image }}
          env:
            - name: MARIADB_HOST
              value: {{ template "mariadb.fullname" . }}
            - name: MARIADB_PORT
              value: "3306"
            - name: WORDPRESS_DATABASE_NAME
              value: {{ default "" .Values.mariadb.mariadbDatabase | quote }}
            - name: WORDPRESS_DATABASE_USER
              value: {{ default "" .Values.mariadb.mariadbUser | quote }}
            - name: WORDPRESS_DATABASE_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: {{ template "mariadb.fullname" . }}
                  key: mariadb-password
          command: ["sh", "-c", "mysql --host=$MARIADB_HOST --port=$MARIADB_PORT --user=$WORDPRESS_DATABASE_USER
--password=$WORDPRESS_DATABASE_PASSWORD"--execute="SELECT 1"]
      restartPolicy: Never

```

Хуки в Helm

1. *pre-install, post-install, pre-delete, post-delete, pre-upgrade, post-upgrade, pre-rollback, post-rollback*
2. Это те же манифесты k8s
3. Одинаковые хуки сортируются по весу и имени объекта
4. Сперва отрабатывают объекты с меньшим весом (от - к +)
5. Хуки не входят в релиз (*helm.sh/hook-delete-policy*)

```

apiVersion: batch/v1
kind: Job
metadata:
  name: "{{ .Release.Name }}"
  labels:
    app.kubernetes.io/managed-by: {{ .Release.Service | quote }}
    app.kubernetes.io/instance: {{ .Release.Name | quote }}
    app.kubernetes.io/version: {{ .Chart.AppVersion }}
    helm.sh/chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
  annotations:
    # This is what defines this resource as a hook. Without this line, the
    # job is considered part of the release.
    "helm.sh/hook": post-install
    "helm.sh/hook-weight": "-5"
    "helm.sh/hook-delete-policy": hook-succeeded
spec:
  template:
    metadata:
      name: "{{ .Release.Name }}"
      labels:
        app.kubernetes.io/managed-by: {{ .Release.Service | quote }}
        app.kubernetes.io/instance: {{ .Release.Name | quote }}
        helm.sh/chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    spec:
      restartPolicy: Never
      containers:
        - name: post-install-job
          image: "alpine:3.3"
          command: ["/bin/sleep", "{{ default \"10\" .Values.sleepyTime }}"]

```

■ Своё репо Helm

1. Можно хранить и в гите
2. Хранилищем может быть любой HTTP-сервер, который может обслуживать файлы YAML и tar и отвечать на запросы GET
3. На сервере должен быть *index.yaml* с индексом всех чартов

<https://example.com/charts>

charts/

- |
- | - index.yaml
- |
- | - alpine-0.1.1.tgz
- |
- | - alpine-0.1.1.tgz.prov

apiVersion: v1

entries:

alpine:

- created: 2016-10-06T16:23:20.499814565-06:00

description: Deploy a basic Alpine Linux pod

digest: 99c76e403d752c84ead610644d4b1c2f2b453a74b921f422b9dcb8a7c8b559cd

home: <https://helm.sh/helm>

name: alpine

sources:

- <https://github.com/helm/helm>

urls:

- <https://technosophos.github.io/tscharts/alpine-0.2.0.tgz>

version: 0.2.0

- created: 2016-10-06T16:23:20.499543808-06:00

description: Deploy a basic Alpine Linux pod

digest: 515c58e5f79d8b2913a10cb400ebb6fa9c77fe813287afbacf1a0b897cd78727

home: <https://helm.sh/helm>

name: alpine

sources:

- <https://github.com/helm/helm>

urls:

- <https://technosophos.github.io/tscharts/alpine-0.1.0.tgz>

version: 0.1.0

generated: 2016-10-06T16:23:20.499029981-06:00

Спасибо!

СЛЁРМ



slurm.io