



# Урок 18. Темплейтирование приложений

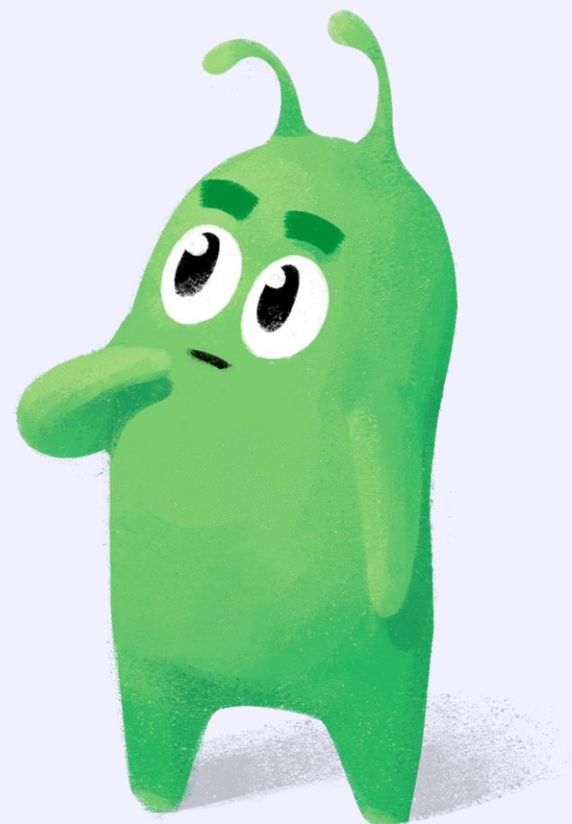
Марсель Ибраев  
СТО Слёрм

# План

- 1 Какие есть варианты?
- 2 Почему Helm?
- 3 Основы работы с Helm
- 4 Создаем свой чарт в Helm
- 5 Advanced things

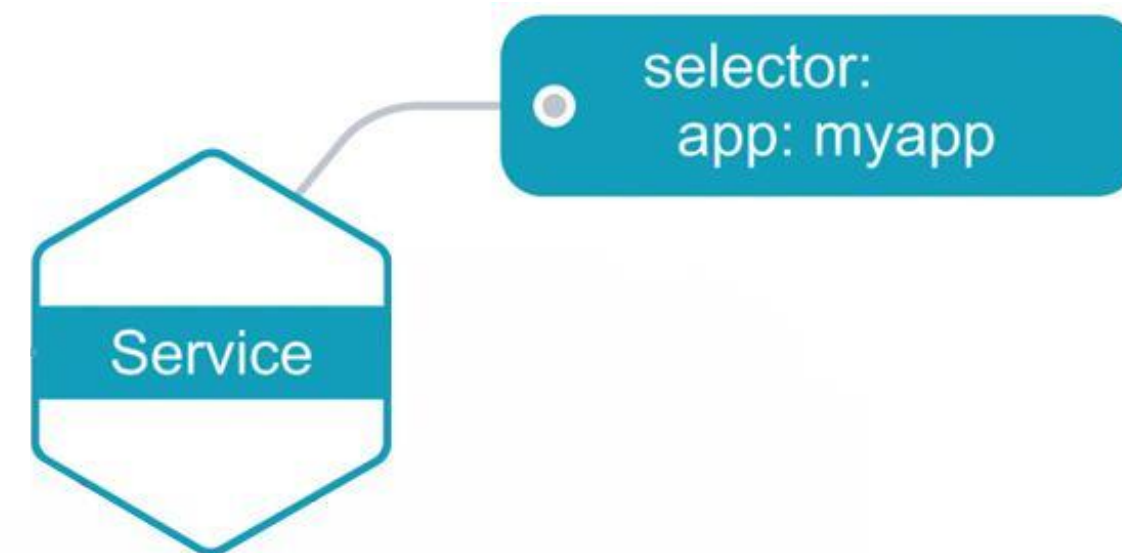


# Как деплоить приложение в кластер?



# Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible / Kustomize / Jsonnet



# Темплэтируем наше приложение

- Есть набор манифестов приложения
- Есть конфигурационный файл
- Есть способ деплоя в кластер

# Kustomize

- Есть набор манифестов приложения

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: the-deployment
  labels:
    deployment: demo
spec:
  replicas: 1
  template:
    metadata:
      labels:
        deployment: demo
    spec:
      containers:
        - name: the-container
          image: annabaker/kustomize-demo-app:latest
          ports:
            - containerPort: 8080
          env:
            - name: MY_MYSQL_DB
              valueFrom:
                configMapKeyRef:
                  name: the-map
                  key: mysqlDB
```

```
apiVersion: v1
kind: Service
metadata:
  name: demo
  labels:
    app: demo
spec:
  ports:
    - port: 8080
  selector:
    app: demo
  type: LoadBalancer
```

# Kustomize

- Есть набор манифестов приложения
- Есть конфигурационный файл

```
kustomization.yaml    commonLabels:
                        app: demo

                        resources:
                        - deployment.yaml
                        - service.yaml
                        - configMap.yaml
```

# Kustomize

- Есть набор манифестов приложения
- Есть конфигурационный файл
- Есть способ деплоя в кластер:

To deploy locally (with kubectl 1.14+), run the following:

```
kubectl apply -k overlays/staging
```

```
kubectl apply -k overlays/production
```

If using an older version of kubectl, use:

```
kustomize build overlays/staging | kubectl apply -f
```

```
kustomize build overlays/production | kubectl apply -f
```



# Kustomize

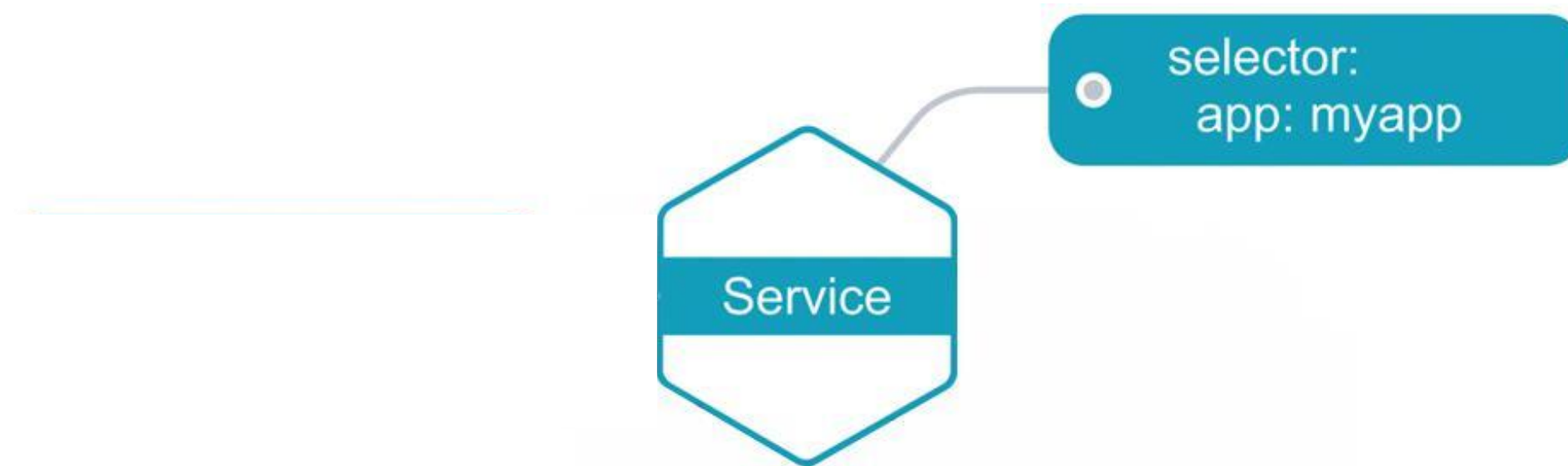
```
someapp/
├─ base/
│   ├── kustomization.yaml
│   ├── deployment.yaml
│   ├── configMap.yaml
│   └── service.yaml
└─ overlays/
    ├── production/
    │   ├── kustomization.yaml
    │   ├── replica_count.yaml
    └── staging/
        ├── kustomization.yaml
        └── cpu_count.yaml
```

<https://speakerdeck.com/spesnova/introduction-to-kustomize?slide=103>

<https://github.com/codefresh-contrib/kustomize-sample-app>

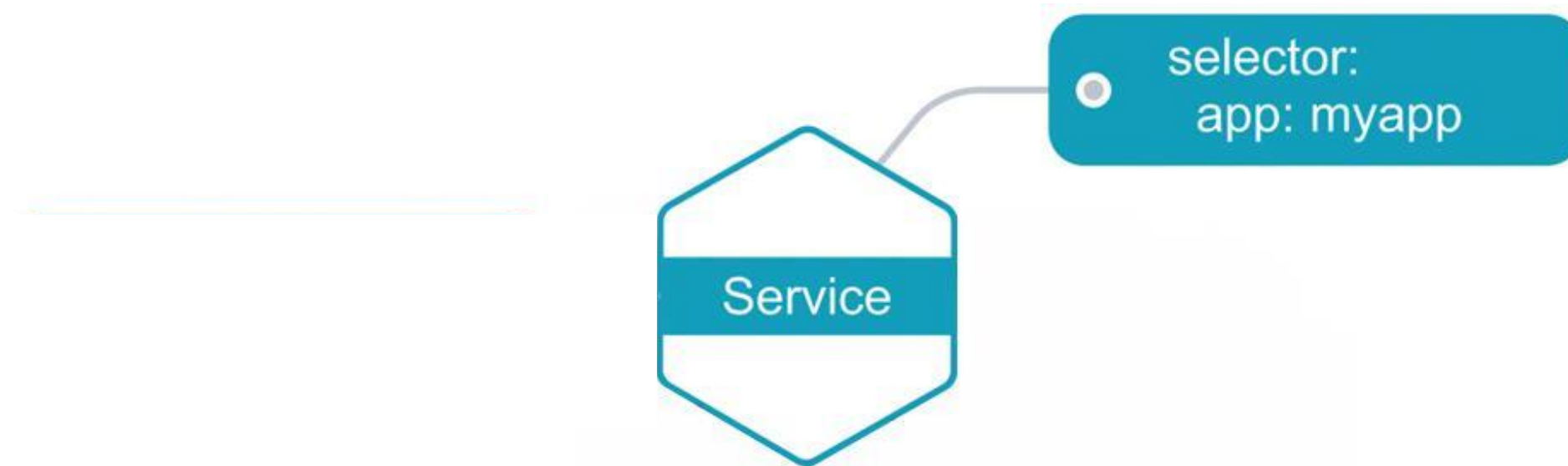
# Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible / Kustomize / Jsonnet
- Kubectl based



# Темплэйтируем наше приложение

- Sed / Envsubst
- Ansible / Kustomize / Jsonnet
- 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
  - Hooks



# Почему Helm?

- «Пакетный менеджер»
- CNCF
- Декларативный
- Есть важные фичи для построения CD
  - Watch
  - Rollback
  - Hooks
- 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: app
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          ports:
            - containerPort: {{ .Values.service.port }}
```

# values.yaml:

```
image:  
  repository: nginx  
  tag: stable  
  
replicas: 2  
  
service:  
  port: 80
```

# Go-template:

- Библиотека для темплейтов в Helm:

<https://masterminds.github.io/sprig/>

- Документация по темплейтам:

<https://pkg.go.dev/text/template>

# 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 repo add southbridge <https://charts.southbridge.ru/>
- helm search hub kube-ops
- helm show values southbridge/kube-ops-view > values.yaml
- helm install ops-view southbridge/kube-ops-view -f values.yaml
- helm ls

# Что внутри

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



# Пишем свой чарт

1. Добавляем темплеты в labels

# Пишем свой чарт

1. Добавляем темплейты в labels
2. Находим [https://helm.sh/docs/topics/chart\\_best\\_practices/labels/](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/](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/](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/](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 темплэйты в labels
2. Находим [https://helm.sh/docs/topics/chart\\_best\\_practices/labels/](https://helm.sh/docs/topics/chart_best_practices/labels/)
3. Добавляем темплэйты в image
4. Добавляем темплэйты в реплики
5. Добавляем темплэйты в ресурсы
6. Добавляем темплэйты в env

# Пишем свой 100ый чарт

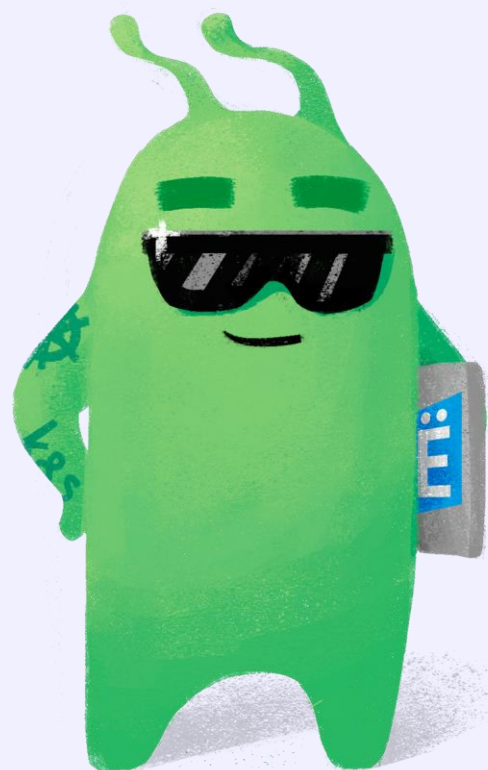
- Узнаем про команду `helm create chart_name`

# 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

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 }}"]
```

# Где хранить чарты Helm?

1. Сделать свой репо на базе веб-сервера
2. Хранить чарты вместе с исходным кодом в отдельной папке

# Library Charts

1. Библиотечные чарты позволяют сделать ещё более универсальные шаблоны
2. Добавлять их в основной чарт нужно как зависимости
3. Сами библиотечный чарты установить нельзя, они лишь основа генерации шаблона



# Southbridge

Проводим аудит, настраиваем  
и сопровождаем кластера Kubernetes.  
Строим и поддерживаем  
DevOps-решения

[Подробнее](#)

