

Helm

- [Helm](#)
- [Go templates](#)
- [Пример проекта](#)

Helm

Один из самых популярных пакетных менеджеров для Kubernetes.

Управление helm

Установка helm:

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg]
https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
sudo apt-get update
sudo apt-get install helm
```

Репозиторий

Команда	Доп. пар.	Описание
helm repo	list	Список репозиториев
	add repo_name repo_url	<div>Добавить репозиторий repo_name с адресом repo_url</div> <div>helm repo add stable https://kubernetes-charts.storage.googleapis.com/</div> <div>Часто используют bitnami, но в России он сейчас закрыт. Есть зеркало:</div> <div>helm repo add bitnami https://raw.githubusercontent.com/bitnami/charts/archive-full-index/bitnami</div>
	update	обновить репозиторий
helm search	repo keyword	Поиск чартов по репозиториям ключевого слова keyword
	hub keyword	В официальном репозитории

Команда	Доп. пар.	Описание
	--max-col-width=0	+ hub/репо полный вывод текста
	--output yaml	+hub/репо вывод в yaml
	--versions	Отсортировать по версиям чарта

Плагины

Он сам по себе мощный, но [ссылка на плагины](#)

Команда	Доп. пар.	Описание
helm plugin install url		Установка плагина
helm plugin list		Список плагинов
helm plugin update pl_name		Обновление плагина
helm plugin uninstall pl_name		Удаление плагина

Переменные окружения

Зависит от переменных окружения. Основные переменные:

Переменная	Описание
XDG_CACHE_HOME	Размещение кешированных данных. По умолчанию ~/.cache/helm
XDG_CONFIG_HOME	Размещение конфигурационного файла По умолчанию ~/.config/helm
XDG_DATA_HOME	Размещение плагинов helm По умолчанию ~/.local/share/helm
HELM_DRIVER	Драйвер для хранения данных. Secret - хранение авторизационных данных в файле, может быть configmap и memory
HELM_NO_PLUGINS	Отключить плагины
KUBECONFIG	Размещение конфигурационного файла kubectl

Charts

Команда	Доп. пар.	Описание
helm install	name_chart repo	Установить из репозитория repo чарт name_chart

Команда	Доп. пар.	Описание
	--...	Переменные внутри чарта <div> helm install kubeapps -- namespace kubeapps bitnami/kubeapps </div>
	--debug --dry-run pr_name path_to_ch	протестировать без установки чарта
helm inspect values	name_chart > ...	Сохранение чарта в файл <div> helm inspect values stable/kube-ops-view > kube-ops-view.yaml </div>
helm lint --strict path-to-chart		Проверить соответствие values схеме
helm fetch name_chart		Скачать чарт в tar
	--untar	И распаковать <div> helm fetch bitnami/wordpress -- untar </div>
helm ls	--namespace namespace	список установленных чартов
helm upgrade	...	Обновление
helm rollback ch_name count		Откатить чарт ch_name на count назад <div> helm rollback redis 1 -- namespace=redis </div>
helm uninstall name_ch		<div> helm uninstall kubeapps -- namespace kubeapps </div>

Общая структура чарта

Helm автоматически определяет последовательность применения шаблонов в чарте.

Директория/файл	Описание	Обяз.
-----------------	----------	-------

Chart.yaml	Метаданные чарта	+
templates/	Ресурсы кубера в формате yaml helm (yaml с переменными) Но файлы начинающиеся с _ не обрабатываются, *.tpl обрабатываются как helper файлы.	+ если не составной
templates/NOTES.txt	Инструкции по использованию	-
values.yaml	Переменные по умолчанию	-
.helmignore	Файлы исключения при упаковке чарта	-
charts/	Зависимости (другие чарты)	-, при отсутствии helm их сгенерирует в соответствии с Chart.yaml
Chart.lock	Первично примененные зависимости.	-, будет создан автоматически
crds/	Зависимости, которые должны быть собраны до основного чарта	-
README.md	Описание	-
LICENSE	Лицензия	-
value.schema.json	Шаблон в json формате	-
files/	Дополнительные файлы	

templates/

Измененные yaml. Добавлены переменные в формате Go шаблонизации. Переменные берутся из файла values.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}
data:
  configuration.txt: |-
    {{ .Values.configurationData }}
```

Родительские пространства имен у переменных:

Пространство имен	Описание
-------------------	----------

.Release	Переменные, связанные с релизом в устанавливаемой системе. .Release.Name - имя релиза .Release.Namespace - пространство имен релиза .Release.Revision - номер версии
.Values	Переменные, размещенные в файле values.yaml
.Chart	Переменные, получаемые из файла Chart.yaml Например, .Chart.Name, .Chart.Version .Chart.AppVersion
.Files	Работа с файлами в директории из директории files. Если файл не существует - вернется ошибка. .Files.Get - Извлекает содержимое файлов .Files.AsSecrets - Возвращает Base-64 закодированную строку для создания secret .Files.AsConfig - Возвращает данные для использования в виде ConfigMap
.Subcharts	Пространство имен дочерних чартов. Например .Subchart.MyChart.firstvalue

values.yaml

В виде обычного key: value yaml

Chart.yaml

Чарты бывают application и library. Application используются для деплоя приложений, library - для предоставления именованных шаблонов, используемых в других чартах. В library чартах не может быть ни одного шаблона, только helper файлы.

Обязательные поля:

Поле	Описание
apiVersion	Версия. В helm 3 формате используется v2
name	Имя чарта. Должно совпадать с именем директории чарта. В именах стоит использовать только -, например first-chart
version	Версия. Формат X.Y.Z

Пример файла

Команда	Доп. пар.	Описание
helm create chart_name		Создание шаблона чарта

Команда	Доп. пар.	Описание
helm install proj_name path		Создание проекта с названием proj_name используя чарт по пути path
	-f par_file	ссылка на другой файл параметров
	--set foo=bar	Ручная установка параметров
helm get manifest proj_name		Получить манифест проекта proj_name

Зависимости чарта

Заполняются в разделе dependencies файла Chart.yaml

Команда	Доп. пар.	Описание
helm dependency build	путь до Chart.yaml	Перестроить зависимые чарты, базируясь на файле Chart.lock. Если этого файла нет - то же что и update
helm dependency list		Список зависимостей
helm dependency update		Обновление чартов и генерация Chart.lock

В зависимостях могут быть условия:

```
dependencies:
- name: dependency1
  repository: https://example.com
  version: 1.x.x
  condition: dependency1.enabled
  tags:
    - monitoring: true
- name: dependency2
  repository: https://example.com
  version: 2.x.x
  condition: dependency2.enabled
  tags:
    - monitoring: true
```

В данном случае переменные dependency1.enabled и dependency2.enabled должны быть установлены в values.yaml файле. Можно несколько переменных через запятую, но лучше завести одну общую переменную и в values ее заполнять. Раздел tags разделяет по группам: если в родительском чарте переменная monitoring не выставлена - данные зависимости

установлены не будут. Может быть несколько тэгов, но если хотя бы один подходит - будет добавлено.

Дочерние параметры чарта могут быть переопределены.

Можно импортировать параметры из дочернего чарта (если его параметры отмечены как экспортируемые).

Также есть хуки, позволяющие выполнять что-то при достижении определенной стадии.

Go templates

Элементы шаблонизации заключены в двойные фигурные скобки, остальные - статический текст. Элементы шаблонизации могут включать переменные, условия, циклы, функции.

Конструкция `{{- удаляет строку в которой функция.`

Условия:

```
{{ if ConditionOne }}  
# Do something  
{{ else if ConditionTwo }}  
# Do something else  
{{ else }}  
# Default case  
{{ end }}
```

```
{{ if eq .Values.favorite.drink "coffee" }}mug: "true"{{ end }}
```

With

Работает так же как и в python

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: {{ .Release.Name }}-configmap  
data:  
  myvalue: "Hello World"  
  {{- with .Values.favorite }}  
  drink: {{ .drink | default "tea" | quote }}  
  food: {{ .food | upper | quote }}  
  {{- end }}
```

Циклы

values.yaml:

```
favorite:
  drink: coffee
  food: pizza
pizzaToppings:
  - mushrooms
  - cheese
  - peppers
  - onions
  - pineapple
```

cur.yaml:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  {{- end }}
  toppings: |-
    {{- range .Values.pizzaToppings }}
    - {{ . | title | quote }}
    {{- end }}
```

Функции:

Можно использовать pipeline: {{ .Values.favorite.drink | quote }}

Функция	Описание
quote	Добавляет кавычки <div>{{ quote .Values.favorite.food }}</div>
upper	В верхний регистр

Функция	Описание
repeat n	Повтор значения n раз <div>{{ .Values.favorite.drink repeat 5 quote }}</div>
default "some_hy"	Присвоить значение по умолчанию если отсутствует <div>drink: {{ .Values.favorite.drink default "tea" quote }}</div>
eq, ne, lt, gt, and, or	Логические функции
indent n	Поставить n пробелов перед конструкцией

[Список функций](#)

Именованные шаблоны

С шаблонами нужно запускать

```
helm install --dry-run --disable-openapi-validation
```

Шаблон внутри _helpers.tpl файла:

```
{{- define "first.labels" -}}
labels:
  'app.kubernetes.io/instacce': {{ .Release.Name }}
  'app.kubernetes.io/managed-by': {{ .Release.Service }}
{{- end }}

{{- define "first.nameofchart" -}}
{{- printf "mycurname" -}}
{{ - end }}
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "first.nameofchart" . }}
{{- include "first.labels" . | nindent 2 }}
data:
  myvalue: "Hello World"
  food: {{ .Values.favourite.food }}
```

```
drink: {{ .Values.favourite.drink }}
```

Точка означает передачу внутрь шаблона всех переменных.

Разница между `include` и `template`: проще использовать `include`, `template` не дает использовать дополнительные функции через `pipeline`

Пример проекта

Задача: helm чарт приложения guestbook с БД redis

Создаем namespace для теста

```
kubectl create namespace guestbook-learn
```

Создаем шаблон структуры папок

```
helm create guestbook
```

Добавляем зеркало проекта bitnami и находим последнюю версию чарта

```
helm repo add bitnami https://raw.githubusercontent.com/bitnami/charts/archive-full-index/bitnami
```

```
helm search repo redis --versions
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
bitnami/redis	20.11.4	7.4.2	Redis(R) is an open source, advanced key-value ...
bitnami/redis	20.11.3	7.4.2	Redis(R) is an open source, advanced key-value ...
bitnami/redis	20.11.2	7.4.2	Redis(R) is an open source, advanced key-value ...

В моем случае это 20.11.4. Добавляем зависимость в Chart.yaml

```
dependencies:
```

```
- name: redis
```

```
  version: 20.11.x
```

```
  repository: https://raw.githubusercontent.com/bitnami/charts/archive-full-index/bitnami
```

Попробуем загрузить зависимость

```
guestbook# helm dependency update .
```

```
Hang tight while we grab the latest from your chart repositories...
```

```
...Successfully got an update from the "bitnami" chart repository
```

```
Update Complete. 🎉Happy Helming!🎉
```

```
Saving 1 charts
```

```
Downloading redis from repo https://raw.githubusercontent.com/bitnami/charts/archive-full-index/bitnami
```

```
Pulled: registry-1.docker.io/bitnamicharts/redis:20.11.4
```

```
Digest: sha256:51ee4afc621d0e0b26109d41c32bf23f3db114f15dd816c1acf8d1ddbf8d57ed
```

```
Deleting outdated charts
```

Действительно, в папке charts появился архив

```
guestbook# ls charts/  
redis-20.11.4.tgz
```

Для завершения настройки redis посмотрим переменные, необходимые для запуска.

```
helm show values charts/redis-20.11.4.tgz
```

Это выдало портянку (если убрать комментарии) в 698 строк. Был задан вопрос ИИ. Ответ отличался от приведенного варианта в книге и прямое использование не поехало бы. В values.yaml было добавлено