Управление зависимостями в Helm

Что разберём

- Что такое **values.yaml**
- Как работают **templates**
- Зачем нужен **requirements.yaml / Chart.yaml (dependencies)**
- Как использовать общие шаблоны и переменные
- Установка внешнего чарта (пример: Prometheus)

Часть 1. Values

Что такое `values.yaml`

Основано на знаниях Helm и общих принципах DevOps.

Файл с конфигурацией по умолчанию.

Пример:

```yaml
replicaCount: 3

image:
  repository: nginx
  tag: "1.21"
```

Используется шаблонами:

```
replicas: {{ .Values.replicaCount }}
```

Как передавать values

Три варианта:

1. Файл:

```
helm install app ./chart -f custom.yaml
```

2. Несколько файлов:

```
helm install app ./chart -f base.yaml -f prod.yaml
```

3. Параметр:

```
helm install app ./chart --set replicaCount=5
```

Часть 2. Templates

Что такое `templates/`

Файлы с Go-шаблонами → генерируют YAML.

Пример Deployment:

```
spec:
  replicas: {{ .Values.replicaCount }}
  template:
    spec:
      containers:
      - name: app
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
```

Важные функции шаблонов

Основано на документации Helm.

- `{{ include "name" . }}` — вставляет шаблон
- `{{ template "name" . }}` — старый вариант
- `{{ required "msg" .Values.key }}` — проверка
- `{{ toYaml .Values.env | indent 8 }}` — преобразование в YAML

Часть 3. Общие шаблоны

Файл `_helpers.tpl`

Используется для:

- общих имён ресурсов
- аннотаций
- повторяющихся блоков

Пример:

```
{{- define "app.fullname" -}}
{{ .Chart.Name }}-{{ .Release.Name }}
{{- end }}
```

Использование:

```
metadata:
  name: {{ include "app.fullname" . }}
```

Часть 4. Зависимости Helm

Зависимости: `Chart.yaml`

Современный формат (Helm 3).

В `Chart.yaml`:

```yaml
dependencies:
  - name: prometheus
    version: "15.0.0"
    repository: "https://prometheus-community.github.io/helm-charts"
```

Команды:

```
helm dependency update
helm dependency build
```

Старый формат `requirements.yaml`

Используется в Helm 2.
В Helm 3 заменён на `dependencies` в `Chart.yaml`.

(Если работаешь с современными чартами — requirements.yaml не нужен.)

Практика

Практика 1: Использовать общие шаблоны

Цель

Создать `_helpers.tpl` и вынести в него общую логику.

Пример:

`templates/_helpers.tpl`:

```
{{- define "app.labels" -}}
app.kubernetes.io/name: "{{ .Chart.Name }}"
app.kubernetes.io/version: "{{ .Chart.AppVersion }}"
{{- end }}
```

Использование в Deployment:

```
metadata:
  labels:
    {{- include "app.labels" . | nindent 4 }}
```

Практика 2: Подключить внешний чарт Prometheus

Шаги

1. Добавляем репозиторий:

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

2. Обновляем:

```
helm repo update
```

3. Подключаем как dependency в `Chart.yaml`:

```yaml
dependencies:
  - name: prometheus
    version: "15.0.0"
    repository: "https://prometheus-community.github.io/helm-charts"
```