K8S 57: Kubernetes. Helm. Tips and Tricks

Описание:

Helm является очень вариативным инструментом с огромным множеством возможностей. В данном обзоре будет приведено лишь небольшое количество хитростей, которые можно использовать в работе с данным инструментом, да и список, приведенный в официальной документации, не полон. Однако, если вы хотите лучше понять этот инструмент и делать множество вещей наиболее элегантным, с технической точки зрения, образом, то данная информация послужит хорошим фундаментом в процессе освоения инструмента.
hash templating
Очень часто передаваемые изменения в чарте при обновлении затрагивают такие элементы, как ConfigMap/Secrets. При этом сами Deployments, которые используют конфигурации из ConfigMap/Secret, остаются неизменными. Это может привести к тому, что при обновлении ресурсов поды продолжат работать на старых конфигах, так как они не были перезапущены.
Чтобы избежать данной проблемы, вы можете использовать sha256sum для определения изменения ресурсов и при их изменении обновлять связанные ресурсы. Подробнее можно почитать тут
Golang magic
При использовании шаблонизатора Golang вы часто будете сталкиваться с необходимостью более глубокого взаимодействия с инструментом, чем простое подставление данных из values. Например, можно использовать дополнительную логику с проверкой наличия значения:

```
value: {{ required "A valid .Values.who entry required!" .Values.who
}}
```

Данный подход позволяет еще на этапе рендеринга конфигурационных файлов выявить проблему в чарте.
Indent-функция определяет количество пробелов от начала вставки, данная опция позволяет легко соблюдать структуру yaml-формата:

```
{{ include "toYaml" $value | indent 2 }}
```

Генерация таких элементов, как ImagePullSecret, может быть произведена несколькими путями. Вы можете просто взять и загнать base64 от конфига с авторизационными данными в соответствующий секрет, а можете сгенерировать тоже самое, используя tml:

```
Values.yml: |
  imageCredentials:
    registry: quay.io
```

```
    username: someone
    password: sillyness

tpl: |
  {{- define "imagePullSecret" }}
  {{- printf "{\"auths\": {\"%s\": {\"auth\": \"%s\"}}}"
.Values.imageCredentials.registry (printf "%s:%s"
.Values.imageCredentials.username .Values.imageCredentials.password |
b64enc) | b64enc }}
  {{- end }}

secret.yaml: |
  data:
    .dockerconfigjson: {{ template "imagePullSecret" . }}
```

И множество иных хитростей...
Helm install/upgrade
В процессе организации CI/CD вам необходимо иметь универсальную команду, которая будет устанавливать чарт, если его нет, обновлять, если он уже существует, еще и делать это максимально надежно. Такой командой является

```
helm upgrade chart_name --install --force ./helm_chart/
```

Debug
В процессе составления чарта вам может потребоваться наличие дебаг инструмента. Ну, разумеется, есть Helm lint, но что же еще?
Командой Helm template можно отрендерить ваши конфигурационные файлы, имитируя установку. В выводе, соответственно, будут все сгенерированные манифесты. Для дебага yaml-формата можно использовать утилиту yq ., работая с выводом команды helm template.

Полезные ссылки:

- Helm Docs: Tips and Tricks

Задание:

Составьте helm_chart со следующими условиями:

1. Должно генерироваться 3-4 разных configMap (из одного шаблона), содержащих идентичные nginx.conf, с одним различающимся параметром.

2. Для каждого отдельного configMap должен генерироваться собственный деплоймент (из одного шаблона), содержащий под nginx и использующий свой configMap как монтируемый конфиг.
3. Загрузите полученный чарт в сетевой git-репозиторий, ссылку на него предоставьте в качестве ответа на задание.

]