

# Карты конфигураций



Сергей  
Андрюнин



**Сергей Андрюнин**

DevOps-инженер

RTLabs




Сергей Андрюнин



# План занятия

1. [Общие сведения о картах конфигураций](#)
2. [Использование](#)
3. [Итоги](#)
4. [Домашнее задание](#)



# Общие сведения о картах конфигураций

---

## Общие сведения

- предназначены для хранения параметров;
- неопределенный размер;
- неопределенный состав;
- карты конфигураций в kubernetes создаются в виде объектов типа `configMap`;
- карты конфигураций очень похожи на секреты, но при этом для работы с манифестами YAML и JSON для секретов нужна конвертация в `base64`.

---

# Что? Как? Где?

- простые данные могут быть переданы упрощёнными методами, например, через командную строку или через переменные окружения;
- сложные данные могут быть переданы через файл или сеть;
- динамические настройки зависят от множества переменных. Применение шаблонизатора;
- избыточность параметров;

---

# Что? Как? Где?

- настройки должны содержать только описание общего поведения сервиса. Например, куда и как подключаться к СУБД;
- настройки нельзя жёстко «зашивать» в сервис, контейнер или приложение;
- версионирование настроек.

---

# Шаблонизатор и шаблонизация

- шаблонизация используется не только для веб-разработки. Нельзя недооценивать силу и мощь шаблонизации для создания сложных конфигурационных файлов;
- один из шаблонизаторов **Jinja** написан на языке Python и используется в ansible и др. для создания готовых файлов из шаблона с подстановкой переменных;
- **Jinja** можно использовать для работы с картами конфигураций.



# Пример шаблонизации настроек (шаблон)

```
server {
    listen      80;
    server_name  {{ domain }} www.{{ domain }};
    access_log   /var/log/nginx/domains/{{ domain
}}-access.log  main;
    error_log    /var/log/nginx/domains/{{ domain
}}-error.log info;
    location / {
        include proxy_params;
        proxy_pass      http://{{ ip
}}:8080/;
    }
}
```

# Пример шаблонизации настроек (использование)

```
from jinja2 import Environment, FileSystemLoader

env = Environment(
    loader=FileSystemLoader('templates')
)
template = env.get_template('nginx.vhosts.jinja')

domains = [{'domain': 'netology.ru',
            'ip': '10.10.10.10'}]

for item in domains:
    config=template.render(
        domain=item['domain'], ip=item['ip']
    )
```

---

# Предназначены для хранения параметров

- именованное значение по ключу (ассоциативный массив);
- хранение за пределами контейнера;
- как переменные среды;
- как аргументы командной строки;
- как файлы в примонтированном томе.



# Неопределенный размер

От одного символа до большого файла.

---

## Неопределенный состав:

- число;
- строка;
- список;
- словарь;
- непечатаемая последовательность символов, закодированная например в base64;
- файл.



# Использование

---

# Особенности

- конфигурации должны быть созданы до того, как они будут использованы в модулях. Ссылки на несуществующие объекты предотвратят запуск Pod;
- можно использовать как файлы в смонтированном томе;
- можно использовать как переменную среды;
- смонтированные конфигурации обновляются автоматически, но при этом не все приложения умеют перечитывать настройки, а также в случае переменных среды требуется перезапуск модуля.

---

# Создание

- из литералов;
- из файлов или каталога;
- гибрид.



## Пример создания

```
kubectl create configmap params --from-literal=var=val  
kubectl create configmap configs  
--from-file=./config.yml
```

---

# Хранение

- размер не более 1MB. Если в base64, то не более 750kB;
- создание множества мелких конфигураций может истощить память;
- карты конфигураций находятся в пространстве имён, что означает доступ модулей только из того же пространства имён.

# Просмотр

```
$ kubectl get configmap params -o yaml
apiVersion: v1
data:
  var: val
kind: ConfigMap
metadata:
  creationTimestamp: "2021-06-15T14:00:16Z"
  name: params
  namespace: default
  resourceVersion: "72549"
  selfLink: /api/v1/namespaces/default/configmaps/params
  uid: b605f107-a595-4596-945f-c94918bab288
```

```
$ kubectl describe configmap params
```

```
Name:          params
Namespace:     default
Labels:        <none>
Annotations:   <none>
```

```
Data
```

```
====
```

```
var:
```

```
----
```

```
val
```

```
Events:  <none>
```

# Пример модуля

```
---
apiVersion: v1
kind: Pod
metadata:
  name: netology-14.3
spec:
  containers:
    - name: myapp
      image: fedora:latest
      command: ['/bin/bash', '-c']
      args: ["env; ls -la /etc/nginx/conf.d"]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: nginx-config
              key: nginx.conf
      envFrom:
        - configMapRef:
            name: nginx-config
      volumeMounts:
        - name: config
          mountPath: /etc/nginx/conf.d
          readOnly: true
      volumes:
        - name: config
          configMap:
            name: nginx-config
```

# Данные в модуле

```
$ kubectl logs netology-14.3
nginx.conf=server {
    listen 80;
    server_name  netology.ru www.netology.ru;
    access_log  /var/log/nginx/domains/netology.ru-access.log  main;
    error_log   /var/log/nginx/domains/netology.ru-error.log  info;
    location / {
        include proxy_params;
        proxy_pass http://10.10.10.10:8080/;
    }
}

SPECIAL_LEVEL_KEY=server {
    listen 80;
    server_name  netology.ru www.netology.ru;
    access_log  /var/log/nginx/domains/netology.ru-access.log  main;
    error_log   /var/log/nginx/domains/netology.ru-error.log  info;
    location / {
        include proxy_params;
        proxy_pass http://10.10.10.10:8080/;
    }
}

total 0
drwxrwxrwx 3 root root 77 Jun 23 23:09 .
drwxr-xr-x 3 root root 20 Jun 23 23:09 ..
drwxr-xr-x 2 root root 24 Jun 23 23:09 ..2021_06_23_23_09_19.997484711
lrwxrwxrwx 1 root root 31 Jun 23 23:09 ..data ->
..2021_06_23_23_09_19.997484711
lrwxrwxrwx 1 root root 17 Jun 23 23:09 nginx.conf -> ..data/nginx.conf
```

---

# Итоги

Сегодня мы изучили:

- что такое карты конфигураций;
- как их создавать и использовать в kubernetes.

---

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Сергей Андрюнин**