Generating ConfigMaps and Secrets

Nigel Brown

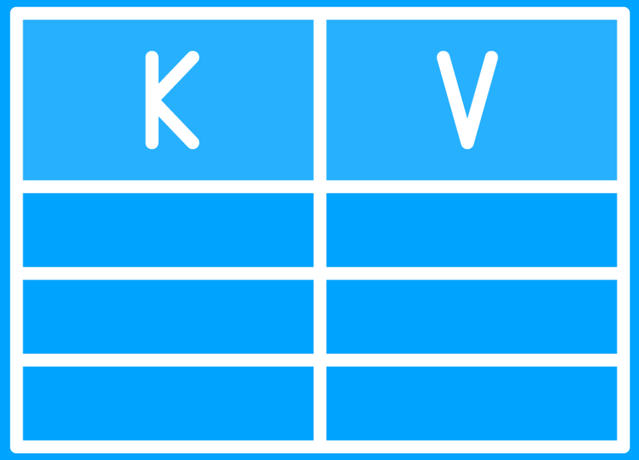
Freelance Technical Author

@n_brownuk | @nigelb@fosstodon.org | windsock.io

Module Outline

Coming up:

- Using Kustomize to create ConfigMaps
- Data updates and workload currency
- Generic generator options and rendering behavior
- Handling Kubernetes secrets with Kustomize



Configuration Data for Applications

Kubernetes provides configuration to
applications using the ConfigMap object.

```
$ kubectl create configmap mysql-user-file \
    --from-literal MYSQL_USER_FILE=/etc/mysql/MYSQL_USER
```

Imperative Generation of ConfigMaps

A declarative approach fits better with the expression and management of desired state

ConfigMap Generator

kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

generators:
  - |-
    apiVersion: builtin
    kind: ConfigMapGenerator
    metadata:
      name: mysql-user-file
    literals:
      - MYSQL_USER_FILE=/etc/mysql/MYSQL_USER
```

Using the configMapGenerator Field

kustomization.yaml

```
<snip>

configMapGenerator:
  - name: mysql-user-file
    literals:
      - MYSQL_USER_FILE=/etc/mysql/MYSQL_USER

<snip>
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-user-file-chbf79kkm5
data:
  MYSQL_USER_FILE: /etc/mysql/MYSQL_USER
```

Configuration Source Options

Literals

Key/value pairs defined within the Kustomization

Envs

Key/value pairs defined within an ‘env’ file

Files

Key is the filename and value is the content of the file

Source Examples

Envs

```
<snip>

configMapGenerator:
  - name: properties
    envs:
      - properties.env

<snip>
```

Files

```
<snip>

configMapGenerator:
  - name: nginx
    files:
      - nginx.conf

<snip>
```

Canonical Sources

The sources in the Kustomization reflect those available sources in the imperative ‘kubectl create’ command.

ConfigMap Name Suffix

Kustomize appends a suffix to the declared name of the ConfigMap object. The suffix is a hash of the content.

Reflection of ConfigMap Name

Suffixed name is reflected in objects that refer to it

Data changes bring about a ConfigMap with a new name

Referent objects will accordingly receive updated definitions

Workload Update

When rendered output is applied to a cluster:

- Workload definition has changed
- Pods are replaced according to set strategy
- Significant benefit of ConfigMapGenerator

Name and Content Relationship

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
  - deployment.yaml
configMapGenerator:
  - name: db
    literals:
      - SQLITE_DB_LOCATION=/tmp/sqlite.db
```

Name Reflection

ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-t5f8b26m9k
data:
  SQLITE_DB_LOCATION: /tmp/sqlite.db
```

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
spec:
  <snip>
  template:
    <snip>
    spec:
      containers:
        - envFrom:
            - configMapRef:
                name: db-t5f8b26m9k
```

Name and Content Relationship

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
  - deployment.yaml
configMapGenerator:
  - name: db
    literals:
      - SQLITE_DB_LOCATION=/tmp/db/sqlite.db
```

Name Reflection

Workload updated in cluster, which now references the new ConfigMap.

ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-g257f577kf
data:
  SQLITE_DB_LOCATION: /tmp/db/sqlite.db
```

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
spec:
  <snip>
  template:
    <snip>
    spec:
      containers:
        - envFrom:
            - configMapRef:
                name: db-g257f577kf
```

Generator Options

Name Suffix

- On by default, but can be disabled

Metadata

- Labels and annotations can be added to all generated objects

Immutability

- Used to prohibit changes to generated objects

Scope of Options

The ‘generatorOptions’ apply to all generated objects.

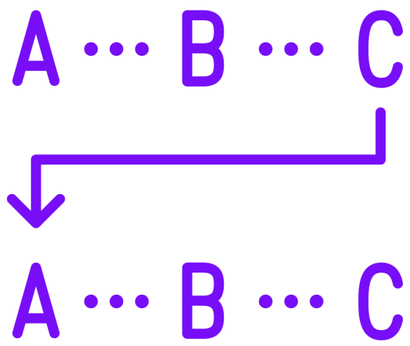
All Objects

```
<snip>

generatorOptions:
  immutable: true
  disableNameSuffixHash: true

<snip>
```

Specific Objects

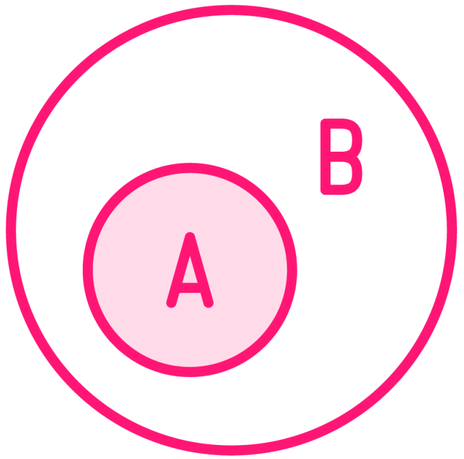
```
<snip>

configMapGenerator:
  - name: log-level
    options:
      immutable: true
      disableNameSuffixHash: true
    literals:
      - error-log-level=debug

<snip>
```

Top-level Generator Options



Precedence

Take precedence over options defined in specific generator definitions



Scope

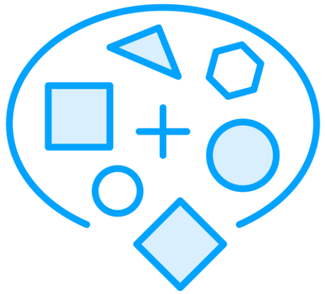
Applies to other generated object types in the Kustomization, such as secrets

Rendering Behavior

Values in ConfigMaps from base kustomizations can be affected by generators defined in overlay kustomizations.

Create

Generates a new ConfigMap



Merge

Adds or updates values in ConfigMap

Replace

Replaces an entire ConfigMap

Defining Behavior

kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

configMapGenerator:
  - name: log-level
    literals:
      - error-log-level=debug
    behavior: replace
```

Demo

Generating App Configuration Data

- Define variable for the app using the ConfigMapGenerator
- Deployment references the ConfigMap
- Perform a build and deploy to the cluster
- Make some checks to verify the outcome

Context

Changes are being made in the base Kustomization instead of the overlay. A patch is required, which we'll cover later.

Demo

Workload Updates on Configuration Change

- Alter value of environment variable
- Generate new ConfigMap with new name
- Apply changes to the cluster
- Check the app consumes new value of environment variable

Secrets

Kustomize can handle Kubernetes Secrets in much the same way as it does ConfigMaps.

Secret Generator

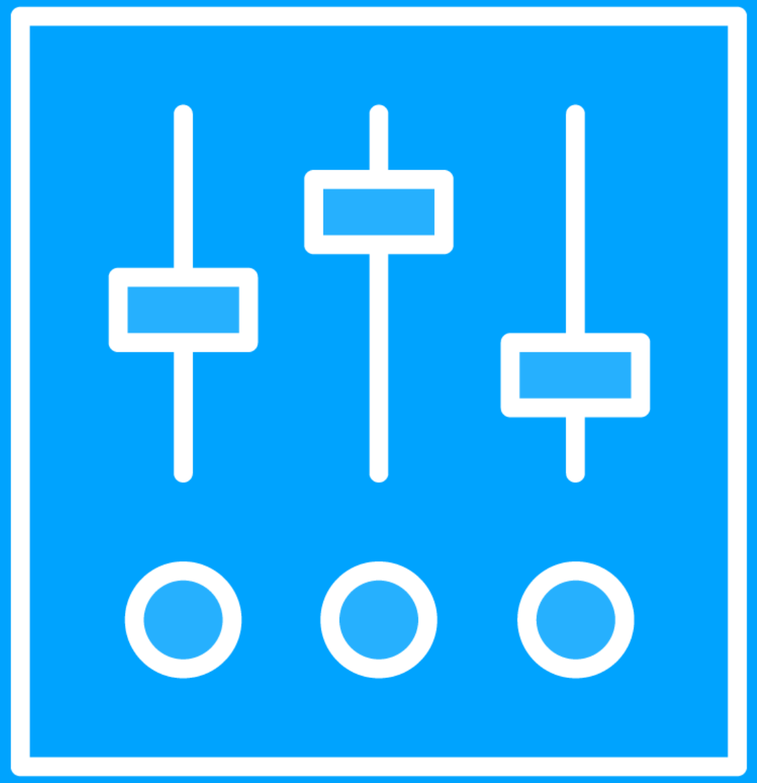
kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

generators:
  - |-
    apiVersion: builtin
    kind: SecretGenerator
    metadata:
      name: registry-credentials
    files:
      - config.json
    type: kubernetes.io/dockerconfigjson
```

Secret Generator Field

kustomization.yaml

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

secretGenerator:
  - name: registry-credentials
    files:
      - config.json
    type: "kubernetes.io/dockerconfigjson"
```



Optional

We don’t have to use Kustomize to manage Kubernetes secrets, but it’s one of the options available to us.

Up Next:

Patching Kubernetes Objects

Module Summary

What we covered:

- Generating ConfigMaps with Kustomize
- Use different canonical source types
- Name suffixes aid workload currency
- Generation behavior set with options
- Secrets also handled with a generator