Patching Kubernetes Objects

Nigel Brown

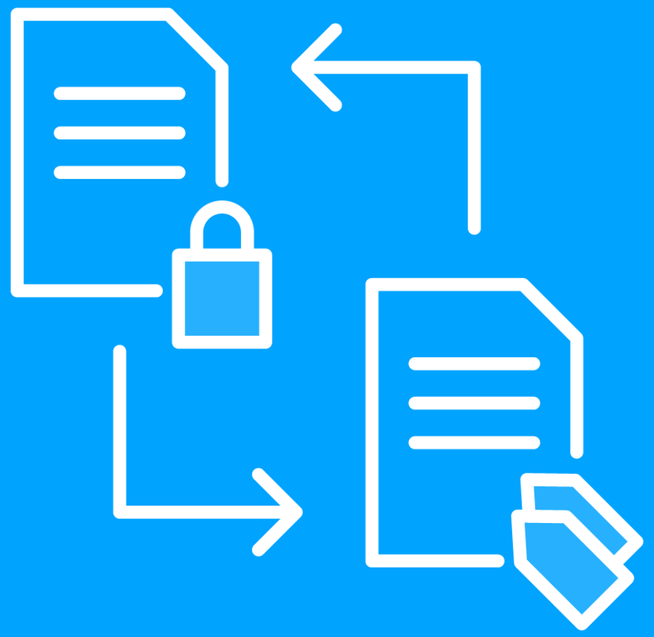
Freelance Technical Author

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

Module Outline

Coming up:

- Patching workload image definitions
- Introduce two patching techniques
- Explore JSON and Strategic Merge patches



Workload Images

Container images will reflect different application versions according to the stage of the software delivery process.

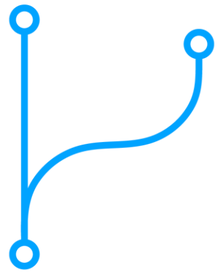
Container Image Tags

```
[host[:port]/][namespace/]repo[:tag]
```

A fully qualified image name contains some optional elements, including a tag. The tag serves to differentiate different variants of an application image.

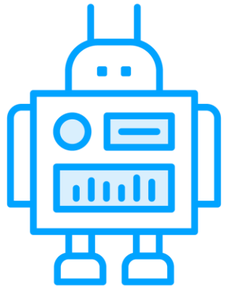
Tag Conventions

Container image tag conventions are personal or organizational choice and can take on many different forms. Here are some examples.



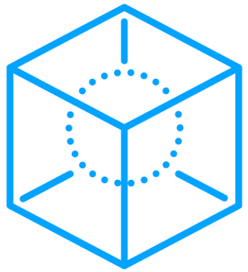
Git Commit SHA

`app:f8a2ab4`



CI Build ID

`app:ci-782.3`



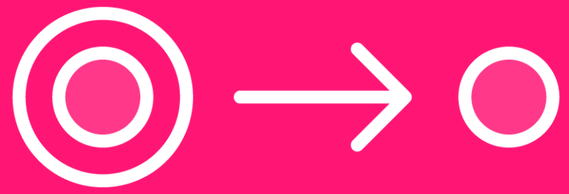
Date/Time

`app:20230502115851`

Setting Image Parameters

kustomization.yaml

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

transformers:
  - |-
    apiVersion: builtin
    kind: ImageTagTransformer
    metadata:
      name: image-tag-transformer
    imageTag:
      name: app
      newTag: ci-782.3
```



Matching an Image

Kustomize looks for a matching image in workload definitions, on which to perform the transformation.

Setting Image Parameters

kustomization.yaml

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

transformers:
  - |-
    apiVersion: builtin
    kind: ImageTagTransformer
    metadata:
      name: image-tag-transformer
    imageTag:
      name: app
      newTag: ci-782.3
```

Image Transformations

Tag

An image tag can be transformed for different scenarios

Name

The name of a container image can be replaced

Digest

Value of image field can be altered to reference a digest

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

images:
  - name: app
    newTag: ci-782.3
  - name: app
    newName: ghcr.io/nbrownuk/app
  - name: ghcr.io/nbrownuk/app
    digest: sha256:0d86234445da6ed8c37aa6b975e73dece58655197982ef91507a7f1b446aeb52
```

Using the Images Field

Demo

Redefining the Container Image in an Overlay

- Build and save output of existing Kustomization
- Add image transformation to change tag
- Build and save output of amended Kustomization
- Compare build outputs side by side

Patching

Patching Kubernetes object definitions plays a big part in producing configuration variants for different environments.

Patching Strategies

Strategic Merge Patch

Contains complete or partial object definition to alter existing object definition

https://bit.ly/3nomq2K

JSON Patch

Specifies patching operations in JSON format that are applied to object definitions

https://bit.ly/3LpCTM6

Some History on Patching

A transformer is provided for each strategy

Kustomization fields exist for both strategies

Both strategies subsumed into generic approach

Patch Transformer

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

transformers:
  - |-
    apiVersion: builtin
    kind: PatchTransformer
    metadata:
      name: patch-transformer
```

Patch Transformer

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

transformers:
  - |-
    apiVersion: builtin
    kind: PatchTransformer
    metadata:
      name: patch-transformer
    patch: |-
      - op: remove
        path: /spec/template/spec/securityContext
```

Patch Transformer

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

transformers:
  - |-
    apiVersion: builtin
    kind: PatchTransformer
    metadata:
      name: patch-transformer
    patch: |-
      - op: remove
        path: /spec/template/spec/securityContext
    target:
      group: apps
      version: v1
      kind: Deployment
```

Patch Transformer

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

transformers:
  - |-
    apiVersion: builtin
    kind: PatchTransformer
    metadata:
      name: patch-transformer
    patch: |-
      - op: remove
        path: /spec/template/spec/securityContext
    target:
      group: apps
      version: v1
      kind: Deployment
      name: app
      namespace: default
```

Patch Transformer

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

transformers:
  - |-
    apiVersion: builtin
    kind: PatchTransformer
    metadata:
      name: patch-transformer
    patch: |-
      - op: remove
        path: /spec/template/spec/securityContext
    target:
      kind: Deployment
      labelSelector: app.k8s.io/name=frontend
```

Patches Field

kustomization.yaml

```
<snip>

patches:
  - path: deployment-patch.yaml
    target:
      group: apps
      version: v1
      kind: Deployment
      name: app

<snip>
```

```yaml
# deployment-patch.yaml
- op: replace
  path: /spec/template/spec/containers/0/imagePullPolicy
  value: Always
```

JSON Patch in YAML

```
# deployment-patch.json
[
    {
        "op": "replace",
        "path": "/spec/template/spec/containers/0/imagePullPolicy",
        "value": "Always"
    }
]
```

JSON Patch in JSON

JSON Patch Operations

https://www.ietf.org/rfc/rfc6902.html#section-4

- Add
- Remove
- Replace
- Move
- Copy
- Test

Patches Field

kustomization.yaml

```
<snip>

resources:
  - deployment.yaml

patches:
  - path: deployment-patch.yaml

<snip>
```

Strategic Merge Patch

deployment-patch.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
spec:
  template:
    spec:
      containers:
        - name: app
          imagePullPolicy: Always
```

Specifying a Target

kustomization.yaml

```
<snip>

resources:
  - deployment.yaml

patches:
  - path: deployment-patch.yaml



<snip>
```

deployment-patch.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
spec:
  template:
    spec:
      containers:
        - name: app
          imagePullPolicy: Always
```

Specifying a Target

kustomization.yaml

```
<snip>

resources:
  - deployment.yaml

patches:
  - path: deployment-patch.yaml
    target:
      kind: Deployment
      name: app

<snip>
```

deployment-patch.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: whatever
spec:
  template:
    spec:
      containers:
        - name: app
          imagePullPolicy: Always
```

Specifying a Target

kustomization.yaml

```
<snip>

resources:
  - deployment.yaml

patches:
  - path: deployment-patch.yaml
    target:
      kind: Deployment
      name: ^app-[1-9].*

<snip>
```

deployment-patch.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: whatever
spec:
  template:
    spec:
      containers:
        - name: app
          imagePullPolicy: Always
```

Target Summary

Without a target	VS	With a target
The object to be patched is selected according to the API group, version, kind, and object name. Details taken from the patch itself.		Object(s) for patching are selected according to the details provided in the target, including enumeration of any regex. Details taken from Kustomization.

Delete an Object

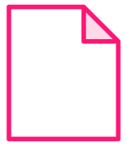
kustomization.yaml

```
<snip>

patches:
  - patch: |-
      $patch: delete
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: app

<snip>
```

Patching Fields

patchesJson6902 – used for specifying patches using the IETF JSON patch strategy

patchesStrategicMerge – used for specifying patches using the StrategicMergePatch strategy



patches – used for specifying patches of either strategy type, with Kustomize interpreting appropriately

Which Field?

Favor the ‘patches’ field ahead of the other two which are now deprecated.

Demo

Patching a Workload Using a Strategic Merge Patch

- Reset the configuration in the base
- Define a ConfigMap in the overlay Kustomization
- Add the patch definition and perform a build
- Repeat build and save generated output

Demo

Patching a Workload Using a JSON Patch

- Define patch to remove security context
- Perform build and save output
- Make comparison to verify outcome

Security Context

Image uses non-privileged user, but writes to privileged location

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: todo
  name: todo
spec:
  template:
    spec:

    <snip>

      securityContext:
        runAsUser: 0
        runAsGroup: 0
```

overlay/kustomization.yaml

```yaml
<snip>

configMapGenerator:
  - name: db
    literals:
      - SQLITE_DB_LOCATION=/tmp/db/todo.db

<snip>
```

Up Next:

Working with Modular Configuration

Module Summary

What we covered:

- Transforming image tags
- Different patching techniques
- JSON patch operations
- Importance of favoring the ‘patches’ field