Application Configuration

» ConfigMaps

Creating ConfigMaps

Create a ConfigMap from literal values.

```
kubectl create configmap my-config --from-literal foo=bar --from-literal hello=world
kubectl get configmaps my-config
kubectl get configmaps my-config -o yaml
```

Next, create a directory with some configuration files.

```
mkdir config
```

```
cat > config/game.properties <<EOF
enemies=aliens
lives=3
enemies.cheat=true
enemies.cheat.level=noGoodRotten
secret.code.passphrase=UUDDLRLRBABAS
secret.code.allowed=true
secret.code.lives=30
EOF
```

```
cat > config/ui.properties <<EOF
color.good=purple
color.bad=yellow
allow.textmode=true
how.nice.to.look=fairlyNice
EOF
```

```
ls -la config
```

Create a ConfigMap from the directory contents and view the detail.

```
kubectl create configmap dir-config --from-file config
kubectl describe configmaps dir-config
kubectl get configmaps dir-config -o yaml
```

Finally, create a ConfigMap from a single file and view the detail.

```
kubectl create configmap file-config --from-file config/ui.properties
kubectl describe configmaps file-config
kubectl get configmaps file-config -o yaml
```

Using ConfigMaps

Consume a ConfigMap from environment variables. Define a manifest to specify individual keys.

```
cat > configmap-env.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: configmap-env
spec:
  containers:
    - name: my-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: CONFIG_FOO
          valueFrom:
            configMapKeyRef:
              name: my-config
              key: foo
        - name: CONFIG_HELLO
          valueFrom:
            configMapKeyRef:
              name: my-config
              key: hello
  restartPolicy: Never
EOF
```

Create the Pod and view the result.

```
kubectl create -f configmap-env.yaml
kubectl get pods --show-all
kubectl logs configmap-env
```

Kubernetes 1.6 adds `envFrom` to specify an entire ConfigMap.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-env
spec:
  containers:
    - name: my-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
        - configMapRef:
            name: my-config
  restartPolicy: Never
```

Use a ConfigMap for command-line arguments. Define a manifest to specify the command-line arguments.

```
cat > configmap-cmd.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: configmap-cmd
spec:
  containers:
    - name: my-container
      image: busybox
      command: [ "/bin/sh", "-c", "echo \$(FOO) \$(HELLO)" ]
      env:
        - name: FOO
          valueFrom:
            configMapKeyRef:
              name: my-config
              key: foo
        - name: HELLO
          valueFrom:
            configMapKeyRef:
              name: my-config
              key: hello
  restartPolicy: Never
EOF
```

Create the Pod and view the result.

```
kubectl create -f configmap-cmd.yaml
kubectl get pods --show-all
kubectl logs configmap-cmd
```

Clean Up

```
kubectl delete pod configmap-cmd
kubectl delete pod configmap-env
```