Master ConfigMaps & Secrets in Kubernetes

The complete guide every DevOps engineer needs

- 🔧 Decouple configuration from code
- 🛡️ Secure sensitive data properly
- ⚡ CKAD exam-ready techniques
- 🎯 Production best practices

Swipe for 8 essential concepts →

#kubernetes #devops #cloudnative

✕ Without them:

- Configuration hardcoded in images
- Rebuilding images for config changes
- Sensitive data exposed in code
- Same app, different environments = chaos

With ConfigMaps & Secrets:

- Generic, reusable container images
- Runtime configuration injection
- Secure handling of sensitive data
- Environment-agnostic deployments

Configuration management = The foundation of scalable K8s apps

ConfigMaps: Your Non-Sensitive Config Store

Store configuration data in key-value pairs

What they can store:

- Environment variables
- Configuration files
- JSON/YAML data
- Command-line arguments

Key Benefits:

- Decouple configuration from container images
- Keep configuration in a central place
- Easily updated without redeploying pods
- Ideal for non-sensitive configuration data

Size limit: 1MB per ConfigMAp

3 Essential Creation Methods

From Literals

Quick & Dirty

```
kubectl create
configmap
example-config
--from-literal=
key1=value1
--from-literal=
key2=value2
```

From Files

Real-world

txt

```
key1=value1
key2=value2
```

Declarative YAML

Production

```
aplVersion: v1
Kind: ConfigMap
metadata:
  data:
  key1: value1
  key2: value2
```

Pro tip: Use YAML for production, literals for quick testing

Secrets:
HANDLE WITH CARE

Common Secret Types

- generic
- docker-registry
- tis
- service-account-token

Key Differences

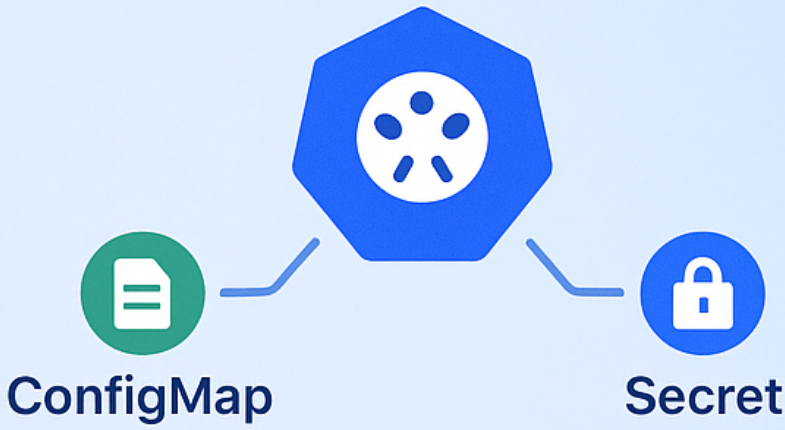
Encoded

Can be Base64 decoded

Araropting

Encrypted in transit

Remember: Secrets are encoded, not encrypted by default



3 Ways to Consume Configuration

Environment Variables

```
container:
 my-VNOlume=
_env:
   'MY_VAR '=
    my-config
```

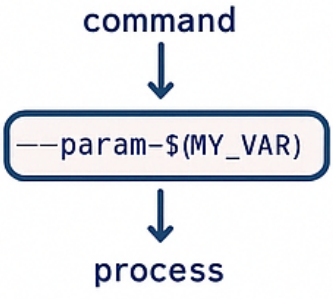
```
MY_VAR=value
```

Volume Mounts

```
roles:
 my--volume
 at:/etcconfig
volumes:
 /etc/cofig
```

Command Arguments

```
mame:
 my-container
arguments
 --param=(MY_PORT)
```



Volume mounts = File-based config

Env vars = Simple values

Pod

Production Best Practices

ConfigMaps & Secrets in Production

Organization

- Standardize structure
- Use appropriate names

Security

- Avoid storing sensitive data
- Control access with RBAC

Updates & Operations

- Implement rollouts
- Automate changes

CKAD Tips

- Know exam objectives
- Master imperative commands

LEVEL UP YOUR CONFIG GAME

ADVANCED TECHNIQUES

- LEVEL 1 — Projected Volumes
- LEVEL 2 — Immutable ConfigMaps
- LEVEL 3 — Downward API
- LEVEL 4 — Init Containers

ENTERPRISE PATTERNS

Enterprise Patterns

CONTINUE LEARNING

CONTINUE LEARNING

💪 Take Action Today!

⭐

1 Star

Give the repository a star

🤝

2 Practice

Try hands-on exercises

📖

3 Study

Review the materials

4 Share

Spread the knowledge

💬 What's your biggest ConfigMap/Secret challenge?

GitHub Follow for more →

#Docker #Kubernetes #DevOps #Cloud