Accessing Secrets Securely on Kubernetes

Mark Heath
Software Architect

@mark_heath www.markheath.net

Managing Secrets

Avoid storing secrets in source control

Use secret stores

- e.g. Kubernetes secrets, Azure Key Vault

Why use Dapr secret management?

- Simplified, consistent API
- Reference secrets in YAML configuration
- Swap between secret stores for local and production

Module Overview

Configuring secret stores

- Local development
- Kubernetes

Fetching a secret directly

Referencing secrets in component

Running GloboTicket in Kubernetes

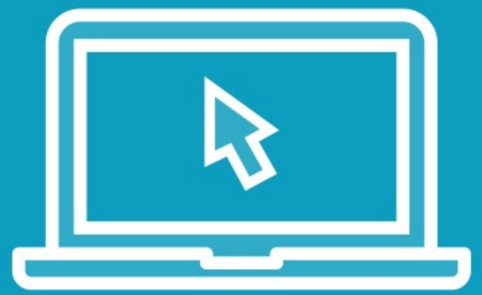
Dapr Secrets Management

Access secrets via an API

Several supported backing stores

http://localhost:3500/v1.0/secrets/vault/mysecret

Demo



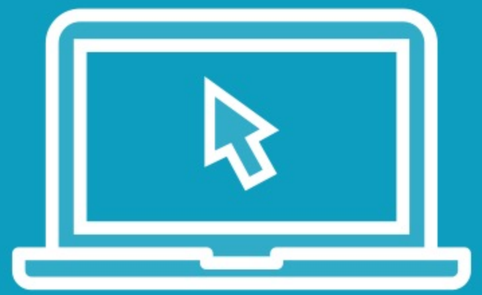
Requesting a secret programmatically

A Dapr secret can contain multiple key-value pairs

To access a secret value you need:

1. secret store name
2. secret name
3. secret key

Demo

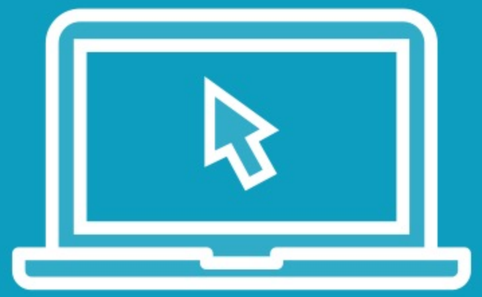


Using secrets with Kubernetes

Creating Azure resources

- AKS cluster
- Azure Storage Account
- Azure Service Bus

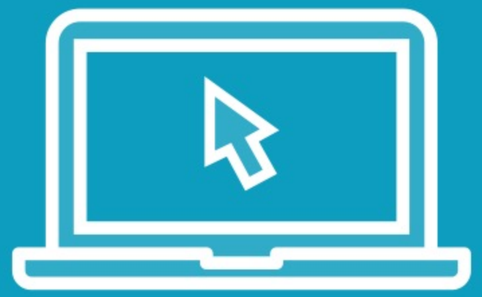
Demo



Installing GloboTicket onto Kubernetes

- dapr init –k
- Build container images
- Kubernetes deployment YAML

Demo



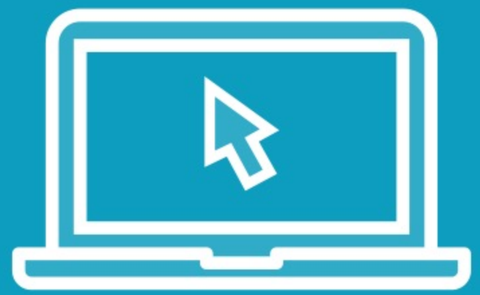
Creating Kubernetes secrets

Accessing secrets in component configuration files

Deploying microservices and Dapr components to Kubernetes

- kubectl apply

Demo



Testing secret access on Kubernetes

- State store and pub sub components can connect to Azure
- Catalog service can retrieve connection string

Secret Scoping

Many secrets are needed only by a single microservice

Principle of least privilege

Summary

Dapr secrets management building block

Many supported secret stores

- Local file secret store
- Kubernetes secret store

Access secrets in component definition files

- secretKeyRef

Access secrets programmatically

- Simplified by the Dapr SDK

Running on Kubernetes

- kubectl apply

Up Next:

Integrating with External Services using Bindings