Secret Config

Securely manage configuration and credential information in AWS

Introduction

- Configuration
- Settings
- Secrets
- Credentials

Example Configuration

- Database
- Username
- Password
- Host
- Port
- Connection Pool Size

Traditional approach

- Hard code config for each "environment"

Lots of different configuration files

- database.yml
- mongodb.yml
- redis.yml
- supplier.yml
- symmetric-encryption.yml
- .....

Credentials

- Scattered throughout
- Hard coded passwords
  - Encrypted for every environment

New environment requires

- New hard coded config
- Code Change
- New deploy
- Ongoing maintenance
- Takes time
- Hard to change
- Encrypted passwords
- Example: UAT

Common externalized approach

- Single configuration file
- application.yml
- test and development
- Shared across engines / gems

Common externalized approach

- Environment Variables
  - MYSQL_HOST
  - MYSQL_PASSWORD
- Insecure
  - Not encrypted
  - Hacker can read process env vars

Common externalized approach

- AWS Secrets Manager
  - 0.40 per secret per month
  - Use JSON BLOB to reduce number of secrets

AWS SSM Parameter Store

- The hidden gem AWS does not want you to know about!

AWS SSM Parameter Store

- Easy to manage
- Web UI
- Single value
  - /release/connect/web/mysql/host
- Terraform Integration

AWS SSM Parameter Store

- Highly available
  - Within Region
- Keys shared across entire account

AWS SSM Parameter Store

- Secure Encryption
- AWS KMS
- Custom encryption key
- FIPS Compliant Hardware
- Impossible to read master encryption key

AWS SSM Parameter Store

- Secure Access
  - Access to AWS SSM Parameter Store API
    - Read: GetParametersByPath
    - Write: PutParameter
    - Delete: DeleteParameter
    - Path: Resource
  - Access to AWS KMS encryption key
    - Key per "environment" / account.

AWS SSM Parameter Store

- Very Expensive Enterprise Pricing
- FREE
- $1 per month per custom KMS Encryption key

Secret Config

- Supports
  - Single Shared Configuration file
    - application.yml
  - Override using env vars
    - export LOGGER_LEVEL=info
  - AWS SSM Parameter Store
    - Encrypt Everything

application.yml

```
development:
  mysql:
    database:   secret_config_development
    username:   secret_config
    password:   secret_configrules
    host:       127.0.0.1

test:
  mysql:
    database:   secret_config_test
    username:   secret_config
    password:   secret_configrules
    host:       127.0.0.1
```

Environment Variables

- No secrets in config files

File: application.yml

```
production:
  mysql:
    database: my_app_production
    username:
    password:
    host:
```

Environment Variables

```
export MYSQL_USERNAME=jackofthebeanstalk
export MYSQL_PASSWORD=KeepThisSafe
export MYSQL_HOST=really_fast.mysql.net
```

Environment Variables

- Not Secure
- Hackers...

AWS SSM Parameter store

```
export SECRET_CONFIG_PROVIDER=ssm
```

- No config file

AWS SSM Parameter store

- All Entries
  - In AWS SSM Parameter Store
  - Secured
  - Encrypted with custom AWS KMS managed Encryption key
  - Centrally managed
    - AWS Console
    - AWS CLI
    - AWS API
  - Highly Available

AWS Configuration

```
export AWS_REGION=us-east-1
export AWS_ACCESS_KEY_ID=.....
export AWS_SECRET_ACCESS_KEY=.....
export SECRET_CONFIG_KEY_ID=b75e28f0-f062-4a9e-a611-a83ff8028208
```

Documentation

github.com/rocketjob/secret_config

Q&A

- Questions?
- Feature Requests?

Agenda

- Introduction
- Overview
- API
- Command Line Interface
- Development Configuration
- Docker Configuration
- Centralized Configuration

Target Audience: Anyone involved or interested in creating, deploying, managing, or supporting docker containers in AWS.