Configuring Java Applications in Containers



Esteban Herrera Author | Developer | Consultant

@eh3rrera eherrera.net

What Can We Use for Configuration?

Properties Files **Environment Variables**

System Properties

Sample Configuration Approach



Read configuration from a general properties file



Override configuration with a properties file for a particular environment



Override particular settings with environment variables/command line options

Overview



Using environment variables

Passing parameters with ENTRYPOINT and CMD

Mounting external properties files

Overriding Docker Compose configuration files

Spring Boot Configuration Priority

Config data files (e.g., application.properties) OS environment variables Java system properties Properties from SPRING APPLICATION JSON Command-line arguments More priority

Using Environment Variables

```
FROM openjdk
ENV VERSION=1
ENV FILE="my file.txt"
ENV LABEL=MY\ LABEL
ENV DIR=/config LEVEL=INFO
ENV DB mysql
LABEL $LABEL
WORKDIR ${DIR}
```

Dockerfile

ENV instruction

docker run -e FILE="my file.txt" my-image
docker run --env DB my-image

docker run Command

Setting environment variables with -e and --env

```
docker run --env-file env.dev my-image
env.dev
# Comment
VERSION=1
DIR
```

docker run Command

Setting environment variables from a file with --env-file

```
# List
environment:
  - VERSION=1
  - ENABLED= 'true'
  - DIR
# Dictionary
environment:
  VERSION: 1
  ENABLED: 'true'
  DIR:
```

Docker Compose

Setting environment variables with the environment section

env_file:

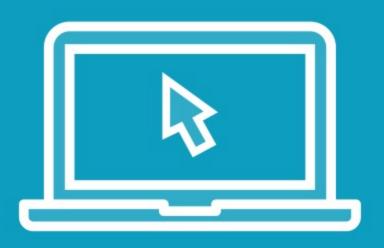
- ./env.dev
- ./web/env.web

Docker Compose

Setting environment variables from a file with the env_file section

Alternatively, place a .env file in the project directory (usually the current directory)

Demo



Setting a profile with an environment variable

Understanding ENTRYPOINT and CMD

```
# Exec form
ENTRYPOINT ["executable", "param1", "param2"]

# Shell form
ENTRYPOINT command param1 param2
```

ENTRYPOINT Instruction

```
# Exec form
CMD ["executable", "param1", "param2"]

# Shell form
CMD command param1 param2

# As parameters of ENTRYPOINT (both should be specified)
CMD ["param1", "param2"]
```

CMD Instruction

Combining ENTRYPOINT and CMD

```
ENTRYPOINT ["java", "-jar", "app.jar"]
CMD ["--server.port=8081"]
```

java -jar app.jar --server.port=8081

Combining ENTRYPOINT and CMD

```
ENTRYPOINT ["java", "-jar", "app.jar"]
CMD --server.port=8081
```

java -jar app.jar --server.port=8081

Combining ENTRYPOINT and CMD

```
ENTRYPOINT java -jar app.jar
CMD --server.port=8081
```

java -jar app.jar

docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]

docker run Command Syntax

Overriding CMD

```
ENTRYPOINT ["java", "-jar", "app.jar"]
CMD --server.port=8081

docker run my-image --server.port=8082

java -jar app.jar --server.port=8082
```

docker run --entrypoint /bin/bash my-image

Overriding ENTRYPOINT with the docker run Command

Use the option --entrypoint

ENTRYPOINT ["java", "-Dserver.port=8081","-jar","app.jar"]

Defining Java System Properties

ENTRYPOINT ["java", "\${JAVA_OPTS}","-jar","app.jar"] # Doesn't work

Defining Java System Properties

Variable substitution requires a shell

ENTRYPOINT java \${JAVA_OPTS} -jar app.jar # This works

Defining Java System Properties

Variable substitution requires a shell

ENTRYPOINT ["sh", "-c", "java \${JAVA_OPTS} -jar app.jar"]

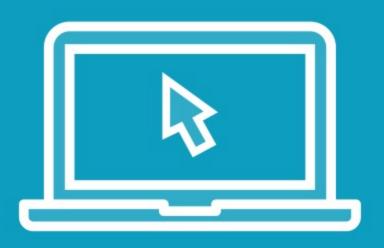
Defining Java System Properties

ENTRYPOINT ["sh", "-c", "java \${JAVA_OPTS} -jar app.jar \${0} \${@}"]

Defining Java System Properties

To also allow command-line property overriding

Demo



Using Java system properties and command-line options

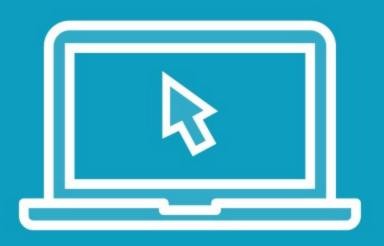
Mounting External Properties Files

```
#To replace default properties file
spring.config.location=/config/my.properties # File
spring.config.location=/config/ # Directory
spring.config.name=my, dev
# To add additional properties files
spring.config.additional-location=/config/dev.yaml # File
spring.config.additional-location=/config/ # Directory
# As environment variables
spring_config_location # Or SPRING_CONFIG_LOCATION
spring_config_name # Or SPRING_CONFIG_NAME
spring_config_additional-location # Or SPRING_CONFIG_ADDITIONAL-LOCATION
```

Spring Boot Options

For reading properties files from the filesystem

Demo



Bind mount an external properties file

Overriding Docker Compose Configuration Files

Overriding Docker Compose Files

Replaces or extends settings

docker-compose -f docker-compose.yml -f docker-compose-dev.yml up

Overriding Docker Compose Files

```
version: '3.8'
services:
    api:
    image: api-app
```

docker-compose.yml

```
services:
    api:
    environment:
```

- PROFILE=dev

docker-compose-dev.yml

Demo



Overriding Docker Compose files

Summary



Approaches to configure an application

- Environment variables
- Java system properties and commandline options
- External properties files
- Overriding Docker Compose files

Same application, different configurations

Up Next:

Managing Application Logs with Docker