Debugging Java Applications Running in Containers



Esteban Herrera Author | Developer | Consultant

@eh3rrera eherrera.net

Overview



Remote debugging

Debugging features IntelliJ (Docker plugin)

Debugging features Visual Studio Code (Java and Docker plugins)

Course summary

Remote Debugging Concepts

Remote Debugging



Java Debug Wire Protocol (JDWP)

 Part of the Java Platform Debugging Architecture (JPDA)

Usually activated with an agent

- An agent is an external library that can be injected into the JVM at runtime
- Pass to the JVM a startup argument with the format: -agentlib:libname[=options]

JDWP Agent

```
-agentlib:jdwp=transport=dt_socket, server=y, suspend=n, address=*:5005
```

Dockerfile

```
FROM openjdk
...
ENTRYPOINT [ # Or CMD
"java",
"-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5005",
"-jar", "app.jar"
]
```

Docker Run Command

my-app-image java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5005

docker run -p 8080:8080

-jar app.jar

Docker Compose

```
version: '3.8'
services:
  web-app:
    ...
    entrypoint: ["java", # Or command: [...]
        "-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5005",
        "-jar", "app.jar"
]
```

For WAR Applications



We don't directly specify these parameters

A script starts the web server

Depending on the server, JDWP is configured in different ways

Generally, environment variables are used

Remote Debugging Environment Variables for Tomcat

Dockerfile

```
FROM openjdk
...
ENTRYPOINT ["catalina.sh", "jpda", "run"]
# CMD ["catalina.sh", "jpda", "run"]
```

Docker Run Command

```
docker run -p 8080:8080
   --entrypoint catalina.sh jpda run
   my-app-image

docker run -p 8080:8080
   my-app-image catalina.sh jpda run
```

Docker Compose

```
version: '3.8'
services:
  web-app:
    ...
  entrypoint: ["catalina.sh", "jpda", "run"]
  # command: ["catalina.sh", "jpda", "run"]
```

Configuring Remote Debugging for Containers in IntelliJ

Configuring Remote Debugging for Containers in Visual Studio Code



Containers and images

Approaches for building applications

- Dockerfiles
- Maven and Gradle Docker images
- Multi-stage builds
- Fabric8's Docker Maven plugin
- Palantir's Docker Gradle plugin
- Spring Boot and Google Jib plugins

Memory and CPU options for containers

Base images in addition to OpenJDK



Docker Compose

- To manage more than one container for the same application
- Uses a declarative style

Configuring applications

- Environment variables
- Java system properties and commandline options
- Properties files external to the application
- Overriding docker-compose files



Docker logging model

- Log everything to the standard output and error streams
- Logging drivers (JSON File is the default one)
- Multiline problem
 - Logging everything in one entry
 - Replacing the new line character
 - Using JSON
 - Sending the logs to a logging aggregator that can parse them
 - Fluentd with concat plugin



Docker plugins

- IntelliJ
- Visual Studio Code

To debug applications in containers, enable remote debugging (JDWP)

- IntelliJ's Docker plugin allows you to override a running configuration
- Visual Studio Code uses the Java debugger plugin and optionally tasks

Thanks for watching