

# Deploying a Spring Boot Open API on Azure Red Hat OpenShift

Robert Rozas Navarro  
Senior Customer Engineer  
Azure-App Dev Domain



# Agenda

1. Introduction
2. Spring Boot on Visual Studio Code
3. Testing on Docker
4. K8s Manifest
5. DEMO- Azure OpenShift Deployment
6. Q&A



# \$ whoami

Robert Rozas Navarro

[robert.rozas@microsoft.com](mailto:robert.rozas@microsoft.com)

Senior Customer Engineer at Microsoft 

Open-Source Advocate 

Microsoft SME 

Github: [@ashwilliams](#)

StackOverflow: [@hackerman](#)



# Spring Boot on Visual Studio Code



Visual Studio Code

# Spring Initializr <https://start.spring.io>



## Project

☒ Maven Project ☐ Gradle Project

## Language

☒ Java ☐ Kotlin ☐ Groovy

## Spring Boot

☐ 2.5.1 (SNAPSHOT) ☒ 2.5.0 ☐ 2.4.7 (SNAPSHOT) ☐ 2.4.6  
☐ 2.3.12 (SNAPSHOT) ☐ 2.3.11

## Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 16 ☒ 11 ☐ 8

## Dependencies

[ADD DEPENDENCIES...](#) CTRL + B

No dependency selected



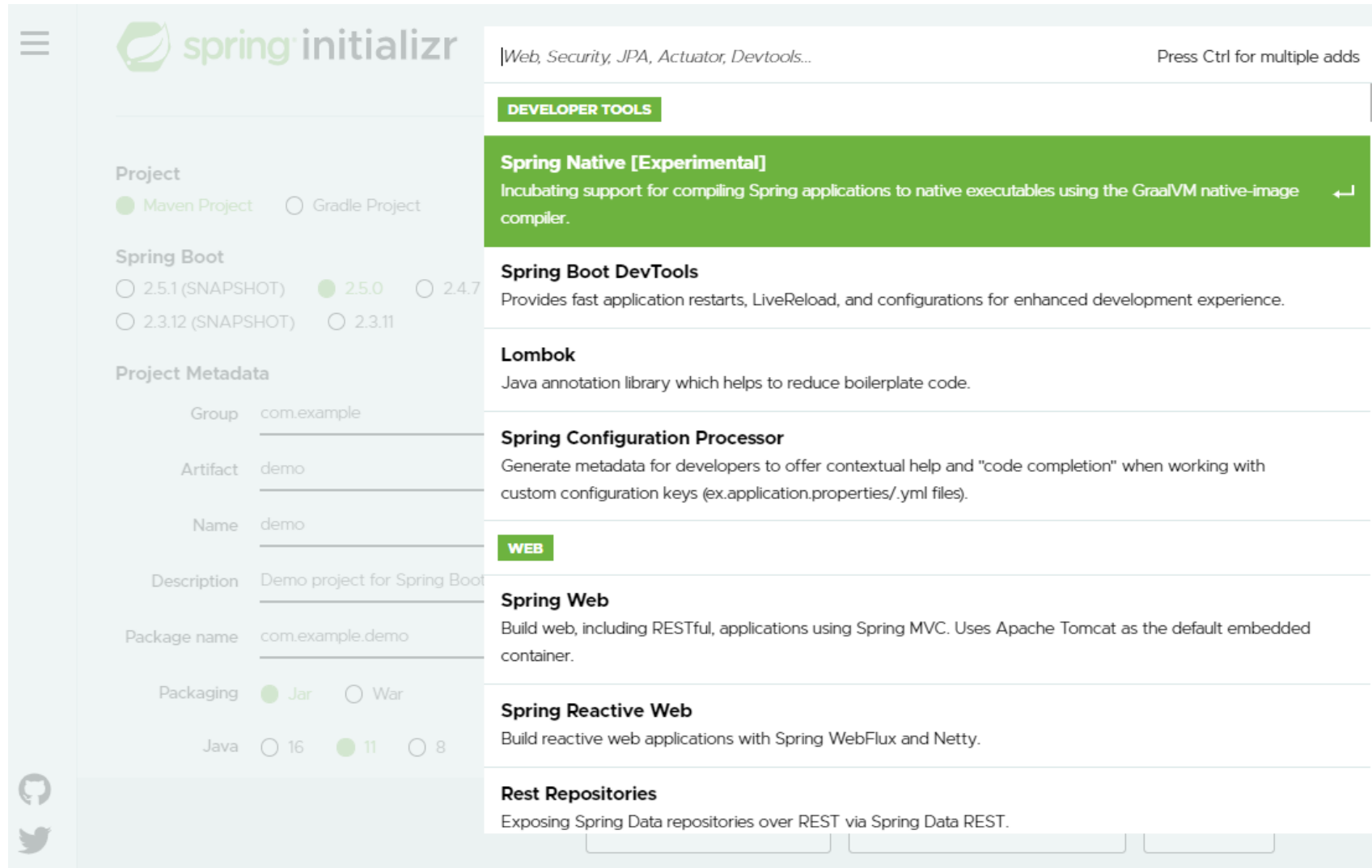
**GENERATE** CTRL + G

**EXPLORE** CTRL + SPACE

**SHARE...**



# Spring Initializr <https://start.spring.io>



The screenshot shows the Spring Initializr web application interface. On the left, there is a sidebar with a menu icon and the Spring Initializr logo. The main content area is divided into two columns. The left column contains form fields for project configuration, and the right column displays a list of recommended dependencies.

**Project Configuration:**

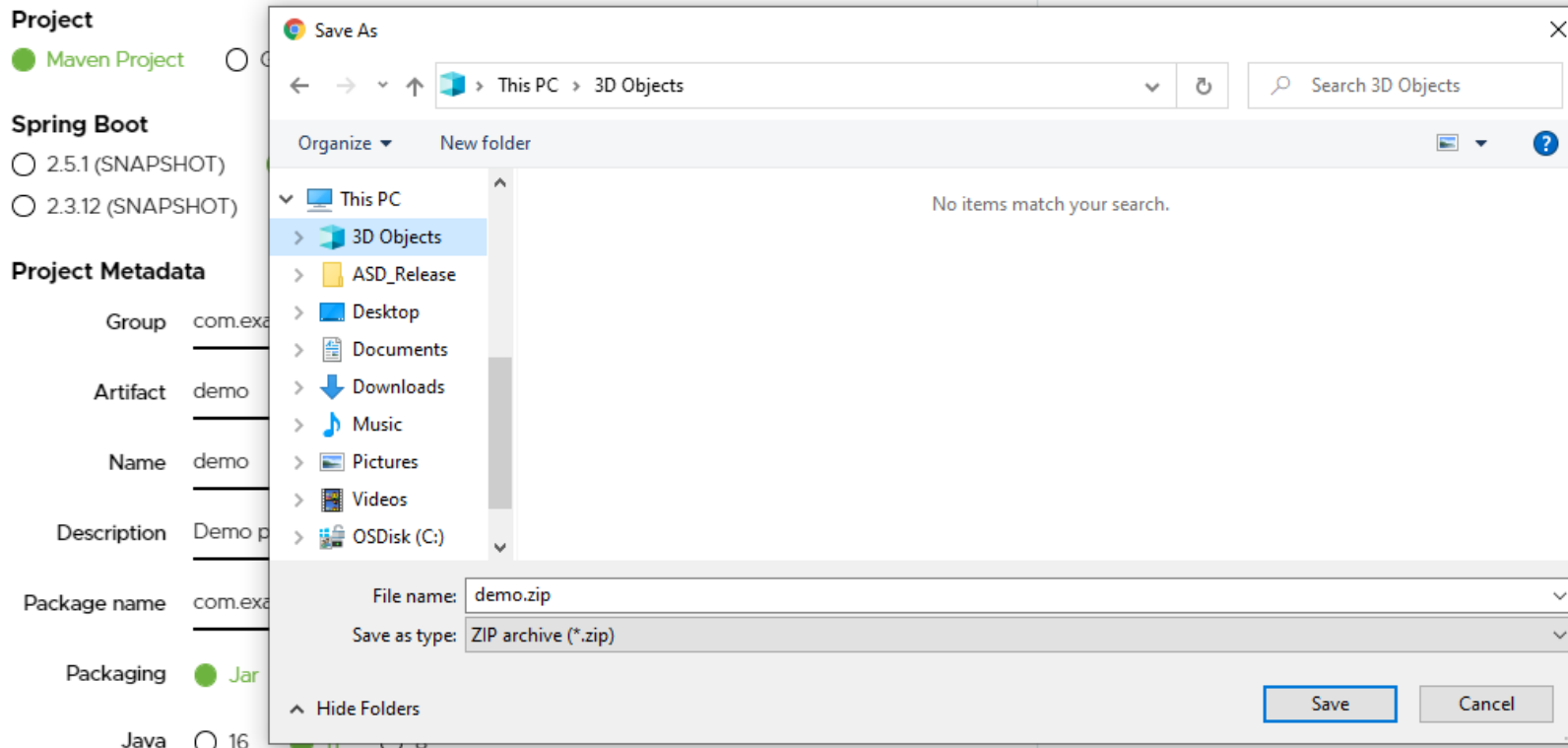
- Project:** ☒ Maven Project ☐ Gradle Project
- Spring Boot:** ☐ 2.5.1 (SNAPSHOT) ☒ 2.5.0 ☐ 2.4.7  
☐ 2.3.12 (SNAPSHOT) ☐ 2.3.11
- Project Metadata:**
  - Group: com.example
  - Artifact: demo
  - Name: demo
  - Description: Demo project for Spring Boot
  - Package name: com.example.demo
  - Packaging: ☒ Jar ☐ War
  - Java: ☐ 16 ☒ 11 ☐ 8

**Recommended Dependencies:**

- DEVELOPER TOOLS**
  - Spring Native [Experimental]**  
Incubating support for compiling Spring applications to native executables using the GraalVM native-image compiler.
  - Spring Boot DevTools**  
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
  - Lombok**  
Java annotation library which helps to reduce boilerplate code.
  - Spring Configuration Processor**  
Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yml files).
- WEB**
  - Spring Web**  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
  - Spring Reactive Web**  
Build reactive web applications with Spring WebFlux and Netty.
  - Rest Repositories**  
Exposing Spring Data repositories over REST via Spring Data REST.



# Spring Initializr <https://start.spring.io>



GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...



# https://github.com/spring-io/start.spring.io

github.com/spring-io/start.spring.io

☰ README.adoc

## start.spring.io chat on gitter

This repository configures a [Spring Initializr](#) instance with a custom UI running at <https://start.spring.io>. The following modules are available:

- `start-client` : client-side assets
- `start-site` : server infrastructure and metadata configuration
- `start-site-verification` : tests to verify the validity of the metadata

## Using

There is a [dedicated page](#) that describes how you can interact with the service.

## Building from Source

You need Java 11 and a bash-like shell.

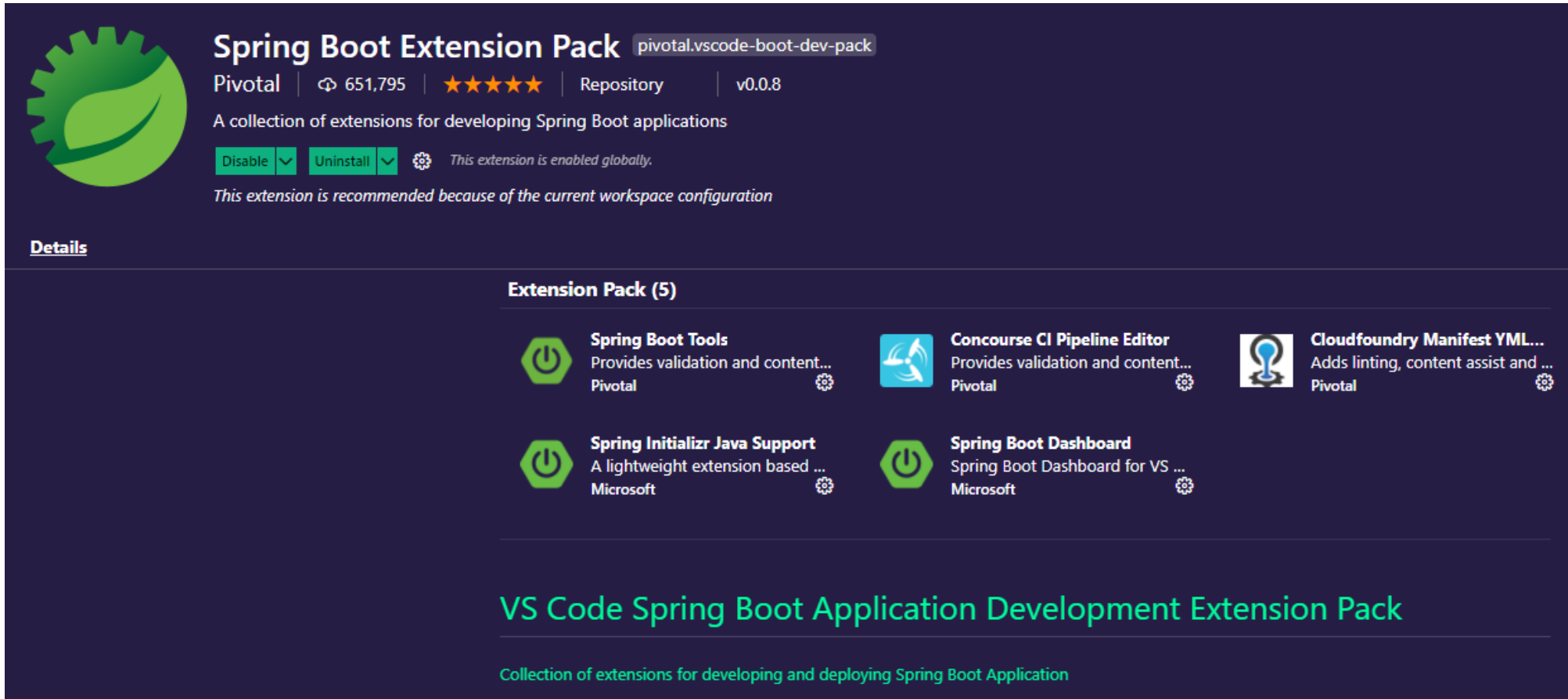
## Building

Invoke the build at the root of the project

```
$ ./mvnw clean install
```



# VSCode Spring Boot Extension Pack



The image shows the VS Code extension marketplace page for the 'Spring Boot Extension Pack' by Pivotal. The pack is identified by the ID 'pivotal.vscode-boot-dev-pack' and version 'v0.0.8'. It has 651,795 installations and a 5-star rating. The description states it is a collection of extensions for developing Spring Boot applications. Controls for 'Disable' and 'Uninstall' are visible, along with a note that the extension is enabled globally and recommended based on the current workspace configuration. A 'Details' link is present. Below, the 'Extension Pack (5)' section lists five included extensions: 'Spring Boot Tools' (Pivotal), 'Concourse CI Pipeline Editor' (Pivotal), 'Cloudfoundry Manifest YML...' (Pivotal), 'Spring Initializr Java Support' (Microsoft), and 'Spring Boot Dashboard' (Microsoft). At the bottom, the full name 'VS Code Spring Boot Application Development Extension Pack' is displayed in green, followed by a subtitle 'Collection of extensions for developing and deploying Spring Boot Application'.

**Spring Boot Extension Pack** `pivotal.vscode-boot-dev-pack`  
Pivotal | 651,795 | ★★★★★ | Repository | v0.0.8  
A collection of extensions for developing Spring Boot applications  
Disable Uninstall ⚙️ This extension is enabled globally.  
This extension is recommended because of the current workspace configuration

Details

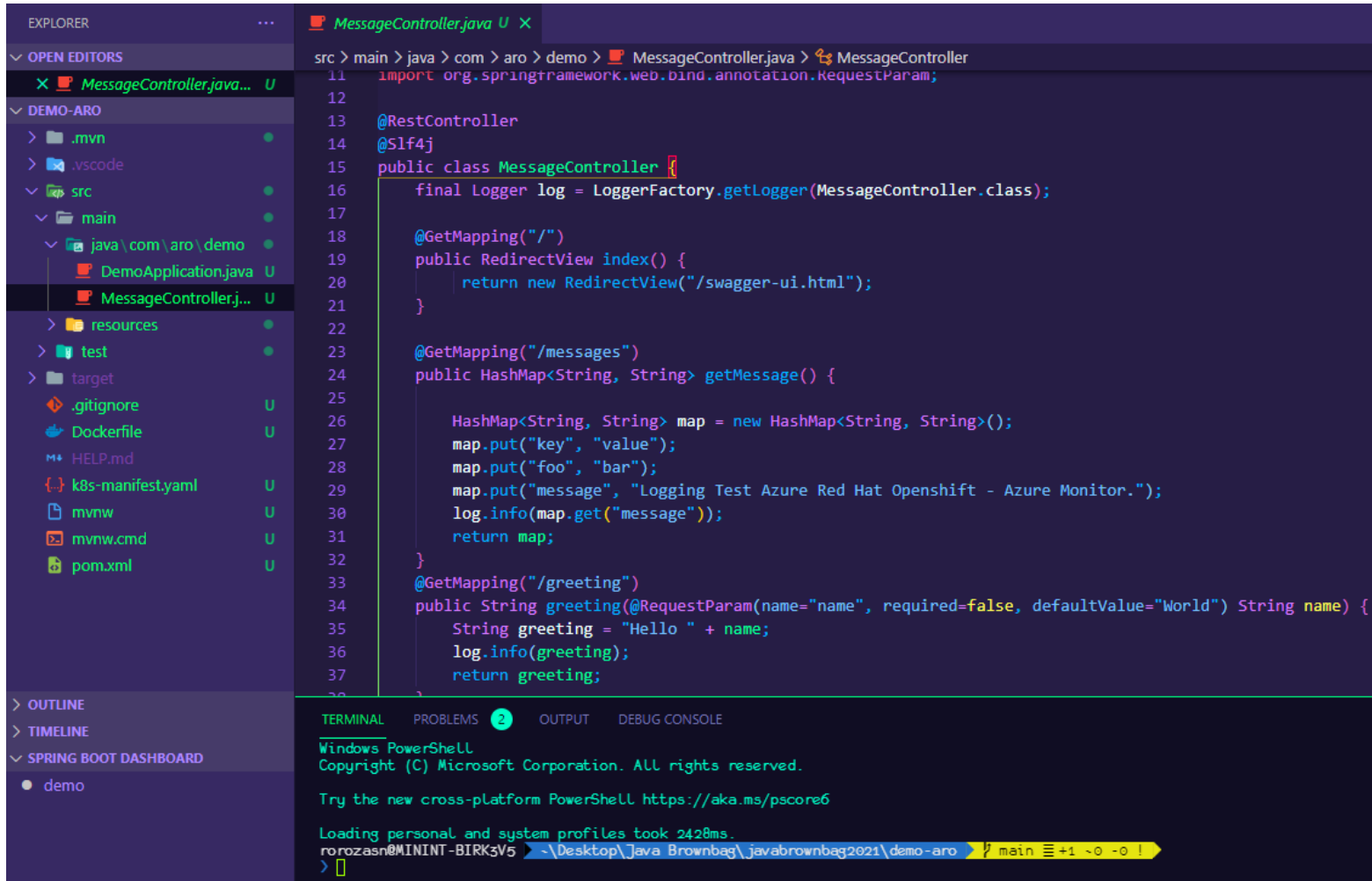
**Extension Pack (5)**

- Spring Boot Tools**  
Provides validation and content...  
Pivotal
- Concourse CI Pipeline Editor**  
Provides validation and content...  
Pivotal
- Cloudfoundry Manifest YML...**  
Adds linting, content assist and ...  
Pivotal
- Spring Initializr Java Support**  
A lightweight extension based ...  
Microsoft
- Spring Boot Dashboard**  
Spring Boot Dashboard for VS ...  
Microsoft

**VS Code Spring Boot Application Development Extension Pack**  
Collection of extensions for developing and deploying Spring Boot Application

<https://marketplace.visualstudio.com/items?itemName=Pivotal.vscode-boot-dev-pack>

# Opening the Project on VS Code

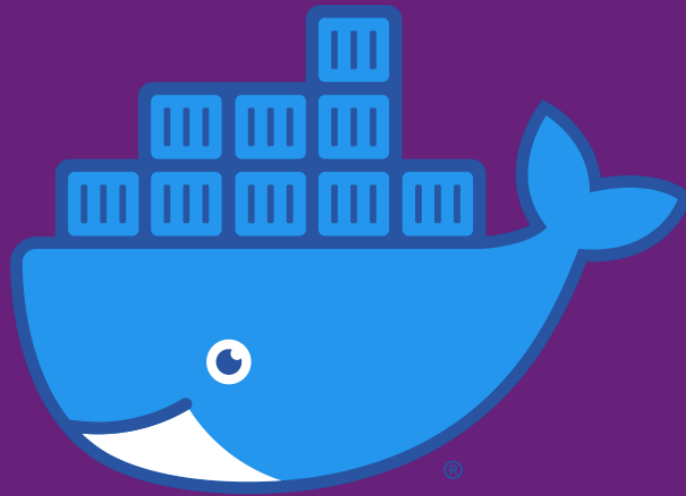


The screenshot displays the Visual Studio Code interface with a Java project. The Explorer sidebar on the left shows the project structure, including the `src` directory, `main` package, and the `MessageController.java` file. The main editor area shows the code for `MessageController.java`, which is a REST controller using Spring Framework annotations. The code includes imports for `org.springframework.web.bind.annotation.RequestMapping` and `org.springframework.web.bind.annotation.RestController`. The `MessageController` class is annotated with `@RestController` and `@Slf4j`. It contains two methods: `index()` which returns a `RedirectView` to `/swagger-ui.html`, and `getMessage()` which returns a `HashMap` containing a message. The `getMessage()` method also logs the message using `log.info()`. The `greeting()` method is also present, returning a greeting string.

```
src > main > java > com > aro > demo > MessageController.java > MessageController
11 import org.springframework.web.bind.annotation.RequestMapping;
12
13 @RestController
14 @Slf4j
15 public class MessageController {
16     final Logger log = LoggerFactory.getLogger(MessageController.class);
17
18     @GetMapping("/")
19     public RedirectView index() {
20         return new RedirectView("/swagger-ui.html");
21     }
22
23     @GetMapping("/messages")
24     public HashMap<String, String> getMessage() {
25
26         HashMap<String, String> map = new HashMap<String, String>();
27         map.put("key", "value");
28         map.put("foo", "bar");
29         map.put("message", "Logging Test Azure Red Hat Openshift - Azure Monitor.");
30         log.info(map.get("message"));
31         return map;
32     }
33
34     @GetMapping("/greeting")
35     public String greeting(@RequestParam(name="name", required=false, defaultValue="World") String name) {
36         String greeting = "Hello " + name;
37         log.info(greeting);
38         return greeting;
39     }
40 }
```

The TERMINAL panel at the bottom shows the Windows PowerShell prompt, indicating the current directory is `~\Desktop\Java Brownbag\javabrownbag2021\demo-aro`.

# Testing on Docker



# Dockerfile



```
1 FROM openjdk:8-jdk-alpine
2 LABEL maintainer="@AshWilliams"
3 COPY target/demo-0.0.1-SNAPSHOT.jar message-api.jar
4 EXPOSE 8080
5 ENTRYPOINT ["java","-jar","/message-api.jar"]
```

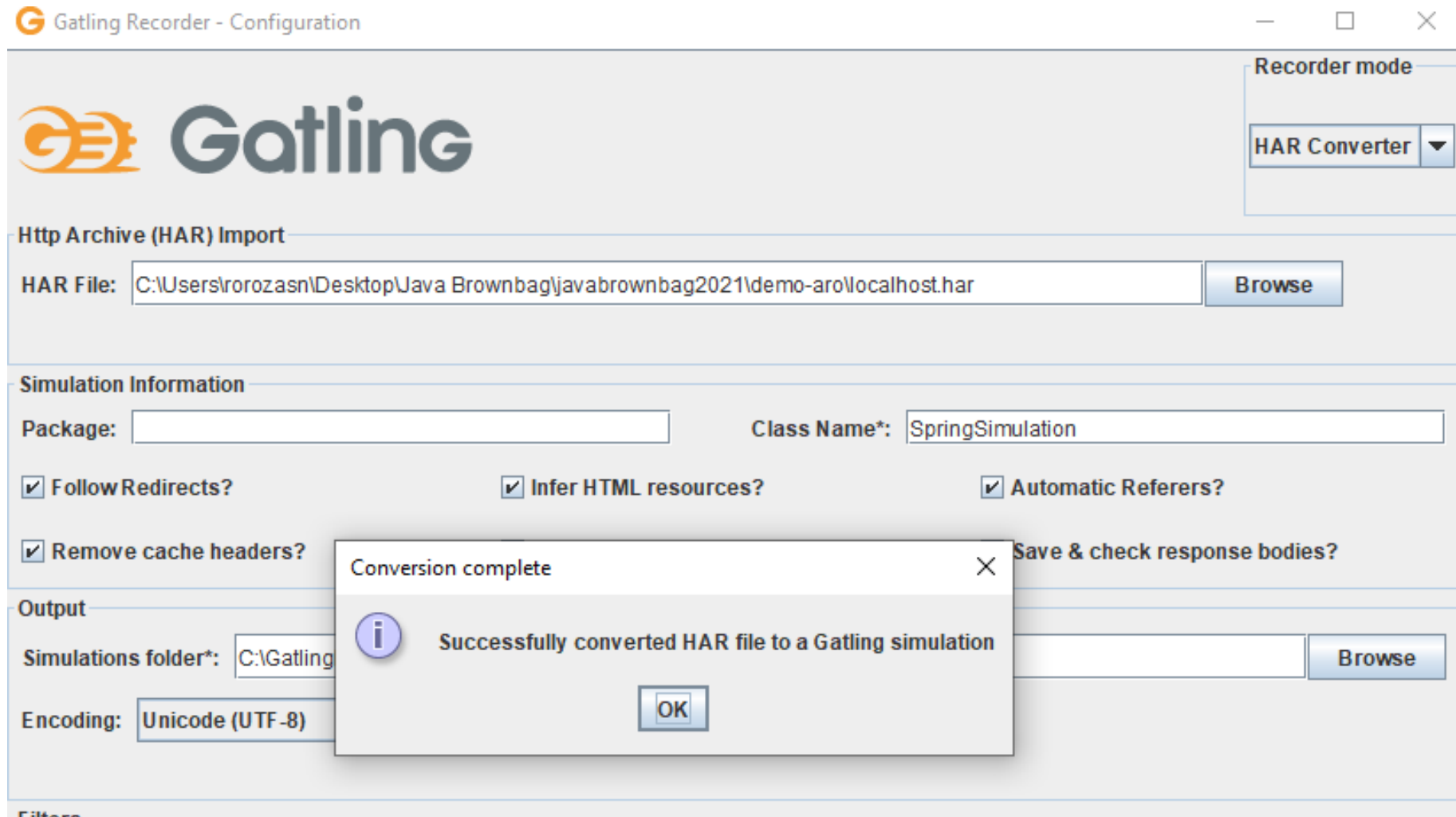
# docker run && docker stats

```
rorozasn@MININT-BIRK3V5 ~\Desktop\Java Brownbag\javabrownbag2021\demo-aro
> docker run -d -p 8085:8080 ashwilliams/aro-spring-log:v2
eb0231d9d63087fc81d44c5bcae0a44297360874e517962efb8cdf411460b034
rorozasn@MININT-BIRK3V5 ~\Desktop\Java Brownbag\javabrownbag2021\demo-aro
> docker stats
```

MINGW64:/c/Users/rorozasn

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
eb0231d9d630	vigorous_dewdney	0.17%	299.5MiB / 5.068GiB	5.77%	946B / 0B	0B / 0B	29

# docker testing with GlatlingIO



# docker testing with GlatlingIO

C: > GatlingIO > user-files > simulations > SpringSimulation.scala

```
9
10 val httpProtocol = http
11   .baseUrl("http://localhost:8085")
12   .inferHtmlResources()
13   .acceptHeader("*/*")
14   .acceptEncodingHeader("gzip, deflate")
15   .acceptLanguageHeader("es-419,es;q=0.9")
16   .userAgentHeader("Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36")
17
18 val headers_0 = Map(
19   "Sec-Fetch-Dest" -> "empty",
20   "Sec-Fetch-Mode" -> "cors",
21   "Sec-Fetch-Site" -> "same-origin",
22   "sec-ch-ua" -> "" Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90""",
23   "sec-ch-ua-mobile" -> "?0")
24
25
26
27 val scn = scenario("SpringSimulation")
28   .exec(http("request_0")
29     .get("/messages")
30     .headers(headers_0))
31
32 setUp(scn.inject(atOnceUsers(1000)).protocols(httpProtocol))
33 }
```

# docker testing with GatlingIO

```
> .\gatling.bat
GATLING_HOME is set to "C:\GatlingIO"
JAVA = "C:\Program Files\Java\jdk-11.0.2\bin\java.exe"
Choose a simulation number:
  [0] RecordedSimulation
  [1] SimulacionClinicaVeterinaria
  [2] SpringSimulation
  [3] computerdatabase.BasicSimulation
  [4] computerdatabase.advanced.AdvancedSimulationStep01
  [5] computerdatabase.advanced.AdvancedSimulationStep02
  [6] computerdatabase.advanced.AdvancedSimulationStep03
  [7] computerdatabase.advanced.AdvancedSimulationStep04
  [8] computerdatabase.advanced.AdvancedSimulationStep05
2
Select run description (optional)
spring 1000 users at once
Simulation SpringSimulation started...
```

2021-06-02 16:02:57

---- Requests ----

> Global

> request\_0

---- SpringSimulation ----

[#####

waiting: 0

MINGW64:/c/Users/rorozasn

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
eb0231d9d630	vigorous_dewdney	198.36%	381.2MiB / 5.068GiB	7.34%	964kB / 8.38MB	967kB / 0B	219



# docker testing with GatlingIO

```
Simulation SpringSimulation completed in 7 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

=====
---- Global Information -----
> request count                1000 (OK=1000   KO=0     )
> min response time            882 (OK=882    KO=-     )
> max response time            7421 (OK=7421   KO=-     )
> mean response time           4932 (OK=4932   KO=-     )
> std deviation                1619 (OK=1619   KO=-     )
> response time 50th percentile 5184 (OK=5184   KO=-     )
> response time 75th percentile 6232 (OK=6232   KO=-     )
> response time 95th percentile 7160 (OK=7160   KO=-     )
> response time 99th percentile 7339 (OK=7339   KO=-     )
> mean requests/sec            125 (OK=125    KO=-     )
---- Response Time Distribution -----
> t < 800 ms                   0 ( 0%)
> 800 ms < t < 1200 ms        34 ( 3%)
> t > 1200 ms                  966 ( 97%)
> failed                       0 ( 0%)
```

# docker testing with GatlingIO



# Kubernetes Manifest



# Final K8s Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-demo-deployment
  labels:
    app: spring-demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: spring-demo
  template:
    metadata:
      labels:
        app: spring-demo
    spec:
      containers:
        - name: spring-demo
          image: ashwilliams/aro-spring-log:v2
          resources:
            requests:
              memory: "300Mi"
              cpu: "180m"
            limits:
              memory: "390Mi"
              cpu: "2000m"
          ports:
            - containerPort: 8080
```

```
apiVersion: v1
kind: Service
metadata:
  name: svc-spring-demo
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: spring-demo
```



# IT'S'DEMO



**OPENSIFT<sup>®</sup>**  
by Red Hat<sup>®</sup>



Azure



Microsoft



# Q&A

