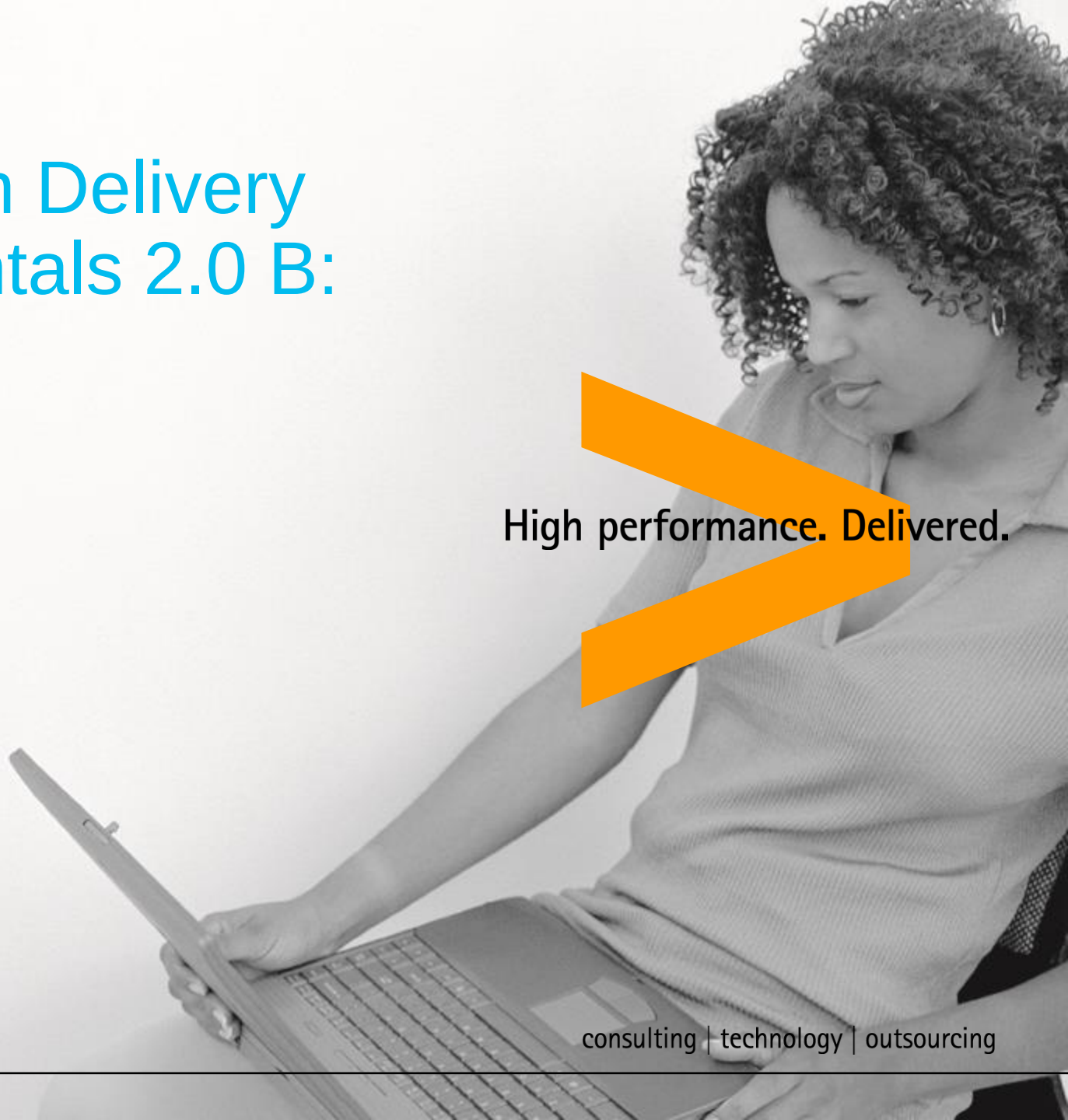
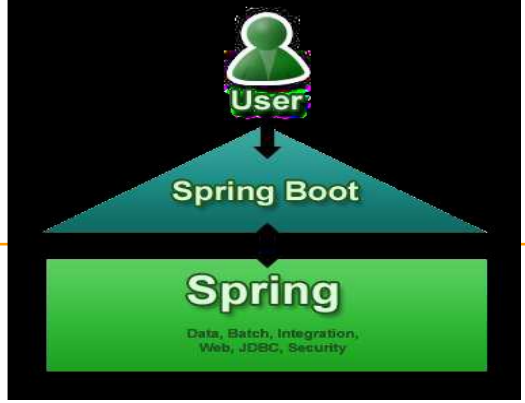
Application Delivery Fundamentals 2.0 B: Java

Spring Boot



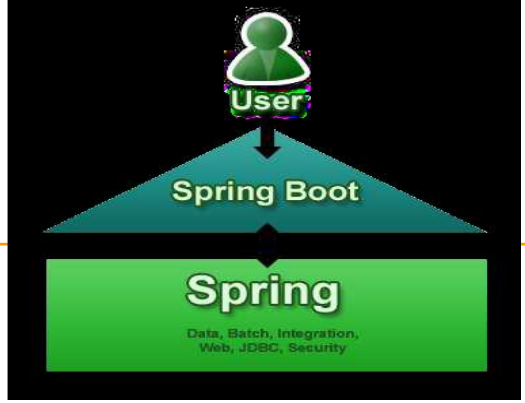
Spring Boot



Goals

- Provide a radically faster and widely accessible getting started experience
- Be opinionated out of the box, but get out of the way quickly as requirements start to diverge from the defaults
- Provide a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration)
- Absolutely no code generation and no requirement for XML configuration

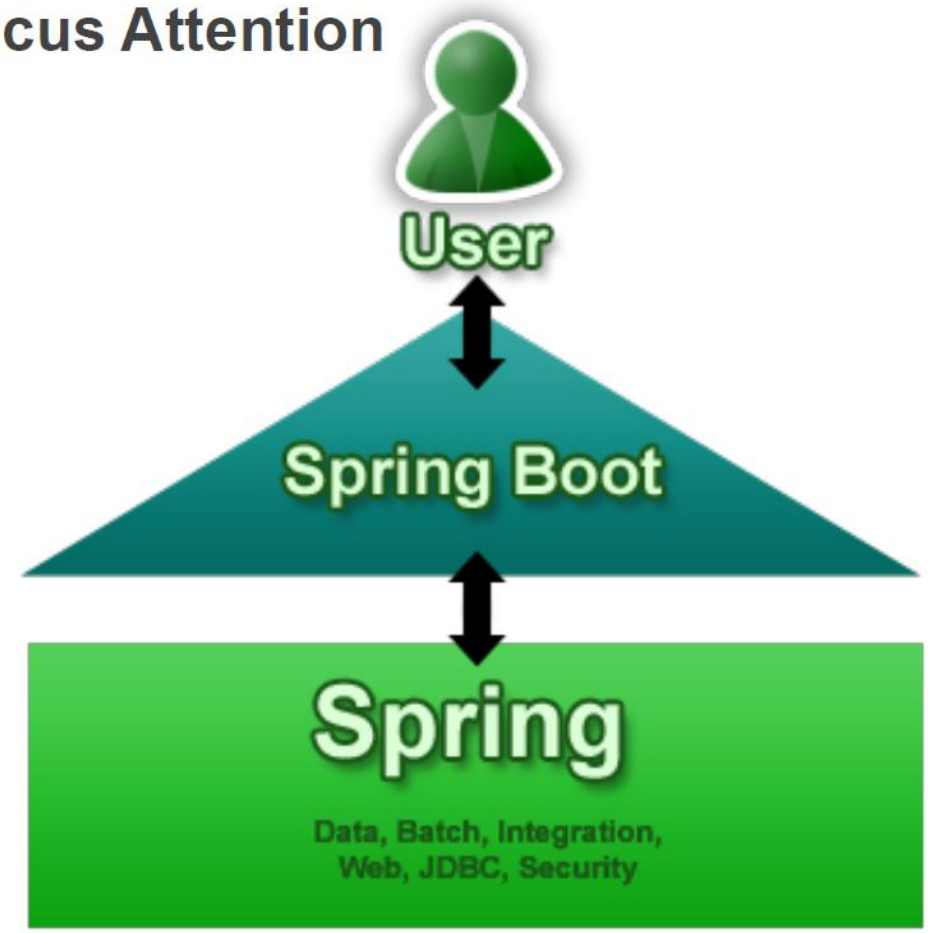
Spring Boot



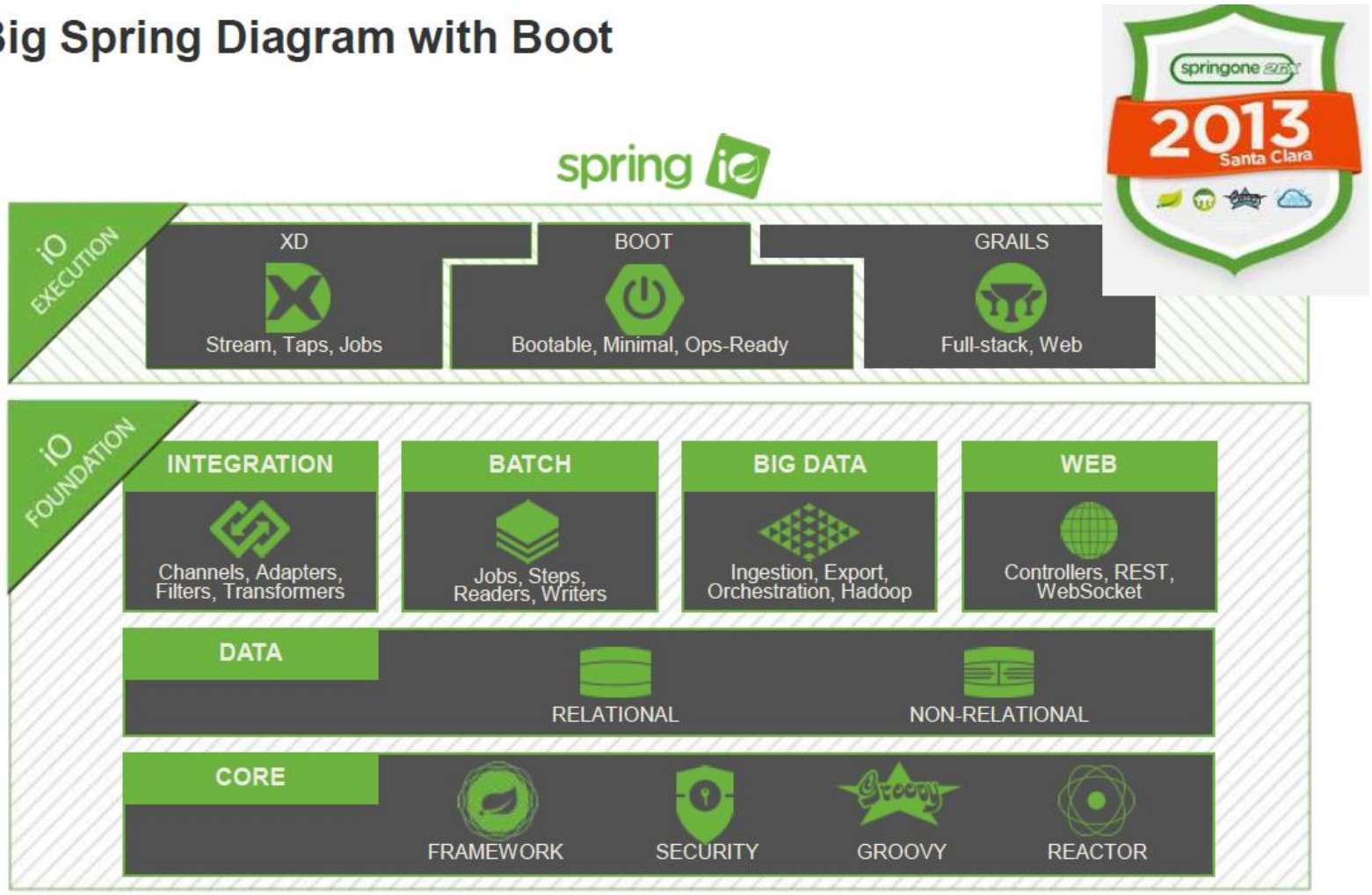
Goals

- To avoid defining more Annotation Configuration(It combined some existing Spring Framework Annotations to a simple and single Annotation)
- Spring Boot avoid writing lots of import statements
- To provide some defaults to quick start new projects within no time.

Spring Boot: Focus Attention

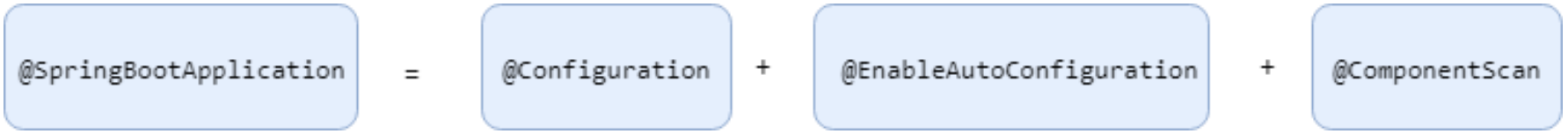


Spring Boot

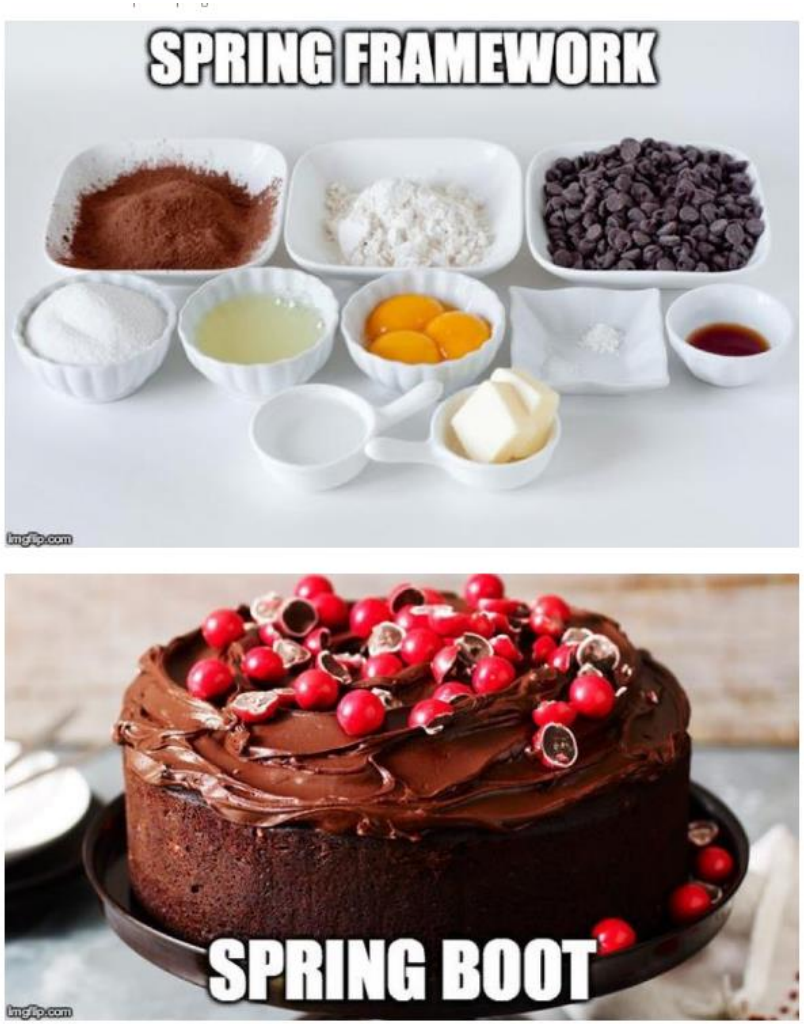

Big Spring Diagram with Boot
springone 2GX
2013
Santa Clara
spring io
IO EXECUTION
XD
Stream, Taps, Jobs
BOOT
Bootable, Minimal, Ops-Ready
GRAILS
Full-stack, Web
IO FOUNDATION
INTEGRATION
Channels, Adapters, Filters, Transformers
BATCH
Jobs, Steps, Readers, Writers
BIG DATA
Ingestion, Export, Orchestration, Hadoop
WEB
Controllers, REST, WebSocket
DATA
RELATIONAL
NON-RELATIONAL
CORE
FRAMEWORK
SECURITY
GROOVY
REACTOR

SpringBoot = a Sub-Project of SpringFramework

SpringBoot = a Sub-Project of SpringFramework

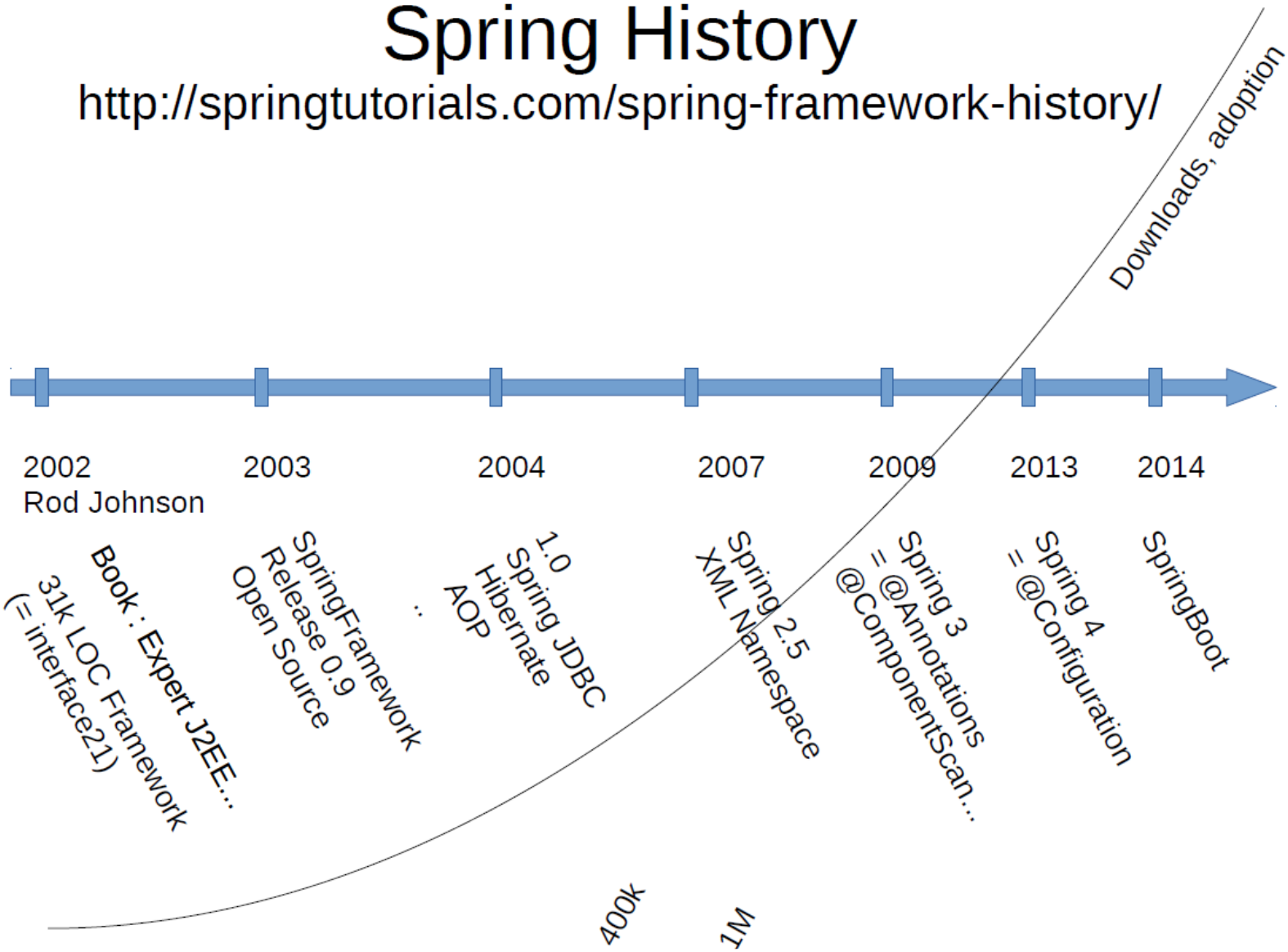


SpringBoot = a Sub-Project of SpringFramework

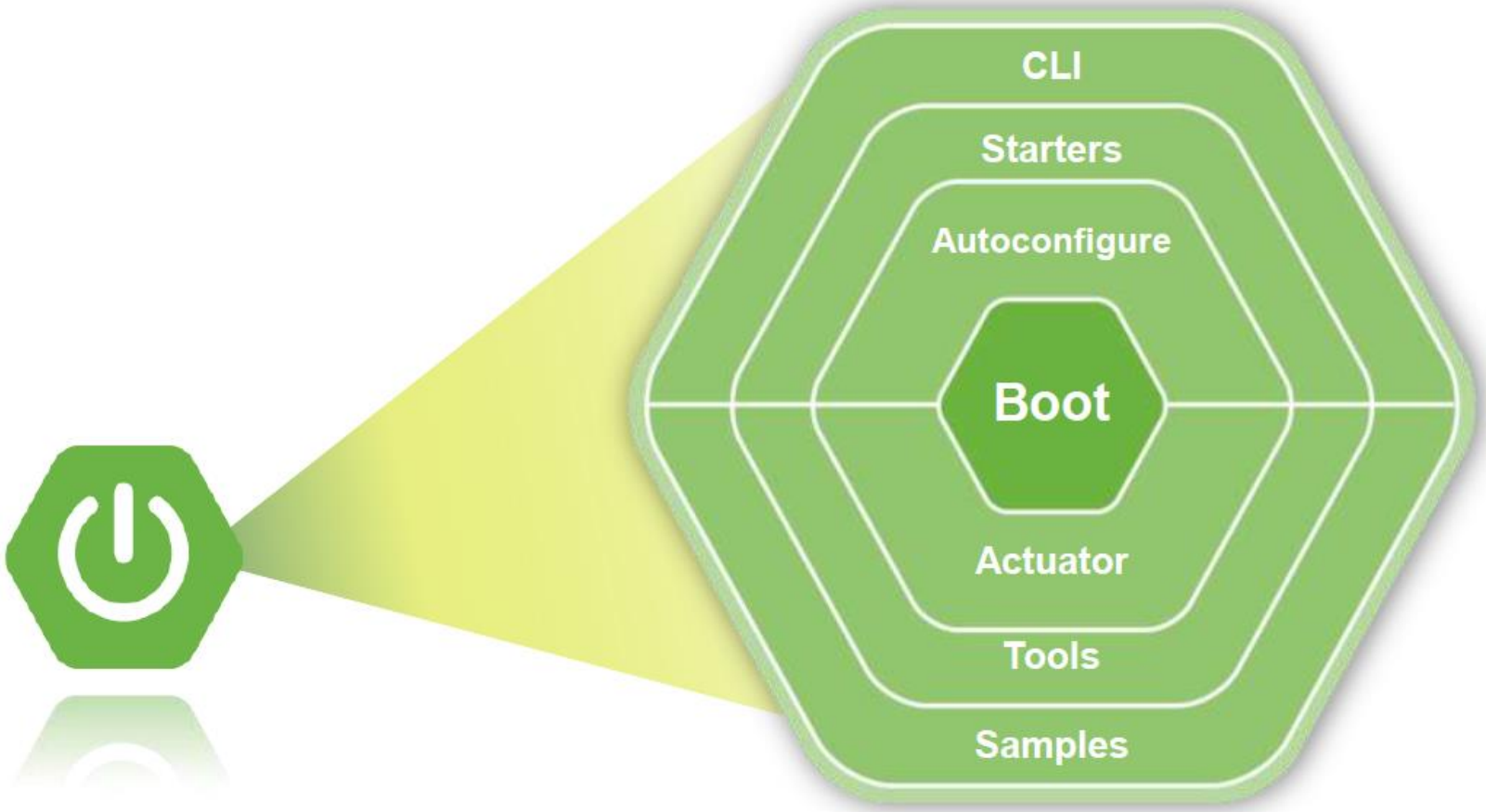


Spring History

http://springtutorials.com/spring-framework-history/



Spring Boot Modules



Spring Boot Components

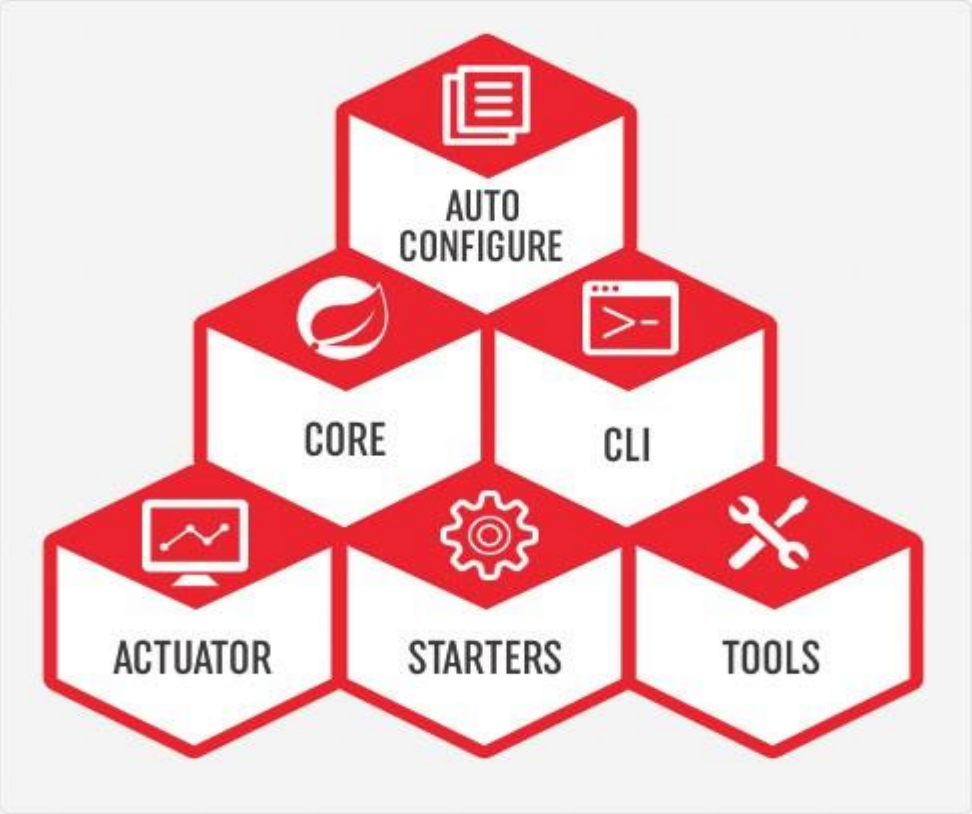
- Spring Boot Auto Configure
- Module to auto configure a wide range of Spring projects.
- It will detect availability of certain frameworks (Spring Batch, Spring Data JPA, Hibernate, JDBC).
- When detected it will try to auto configure that framework with some sensible defaults, which in general can be overridden by configuration in an application.properties/yml file.

Spring Boot Components

- Spring Boot Core
- The base for other modules, but it also provides some functionality that can be used on its own, eg. using command line arguments and YAML files as Spring Environment property sources and automatically binding environment properties to Spring bean properties (with validation).

Spring Boot Components

- Spring Boot CLI
- A command line interface, based on ruby, to start/stop spring boot created applications.



SpringBoot CLI

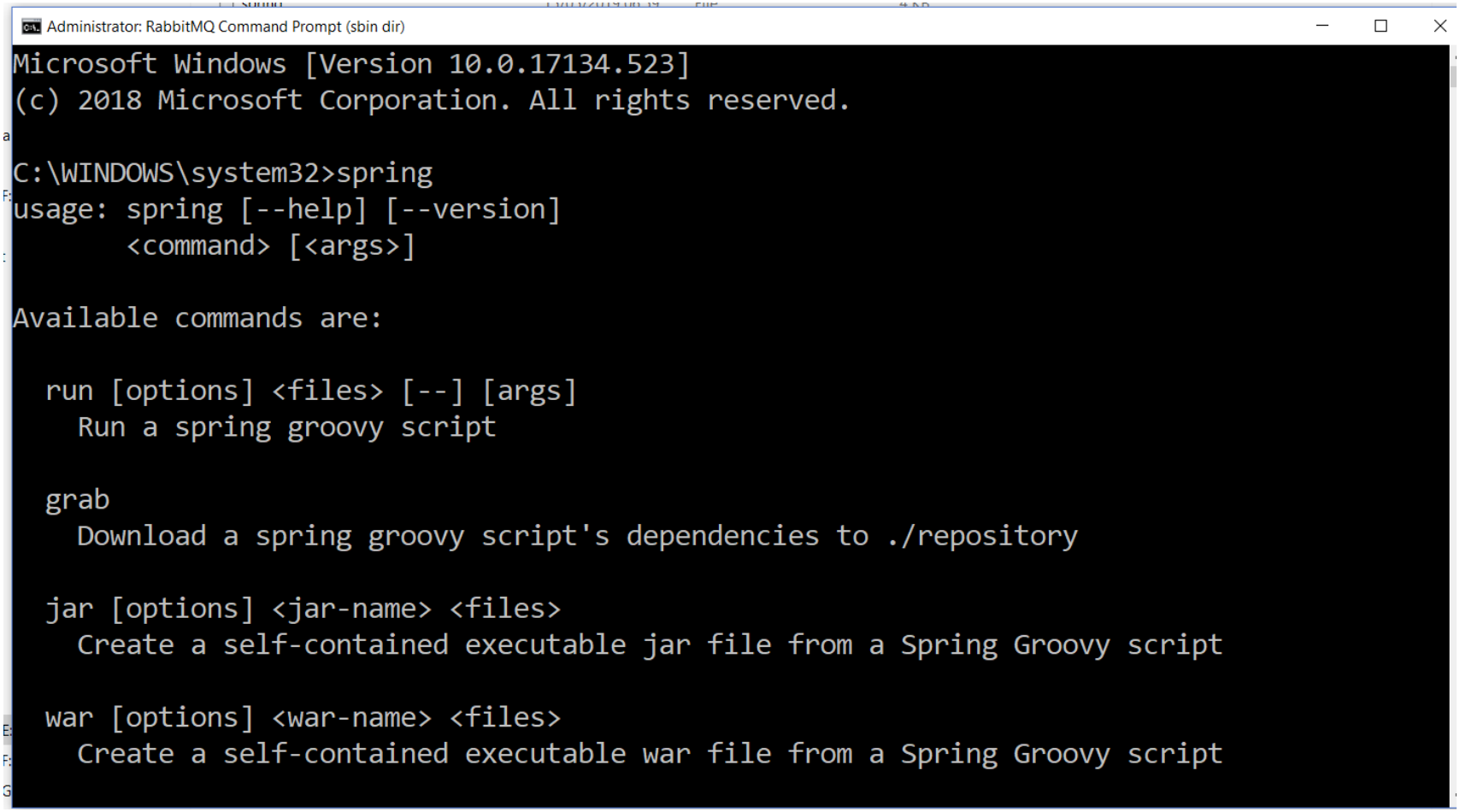
- To work with Spring boot CLI, first we need to install it in our system. There are many ways to install Spring boot CLI.
  1. Manual installation
  2. Installation with SDKMAN!
  3. OSX Homebrew installation
  4. MacPorts installation
  5. Command-line completion

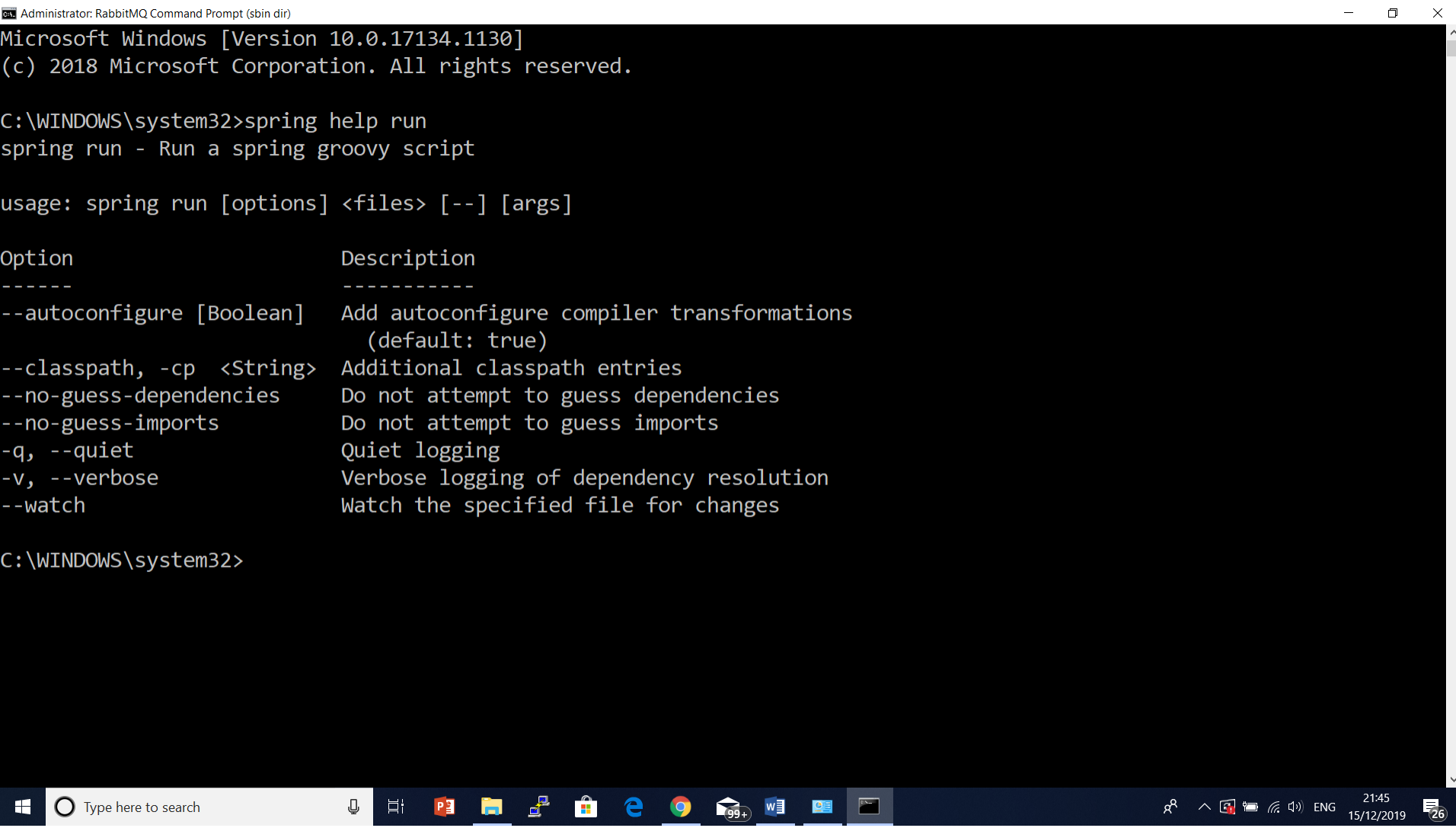
SpringBoot CLI

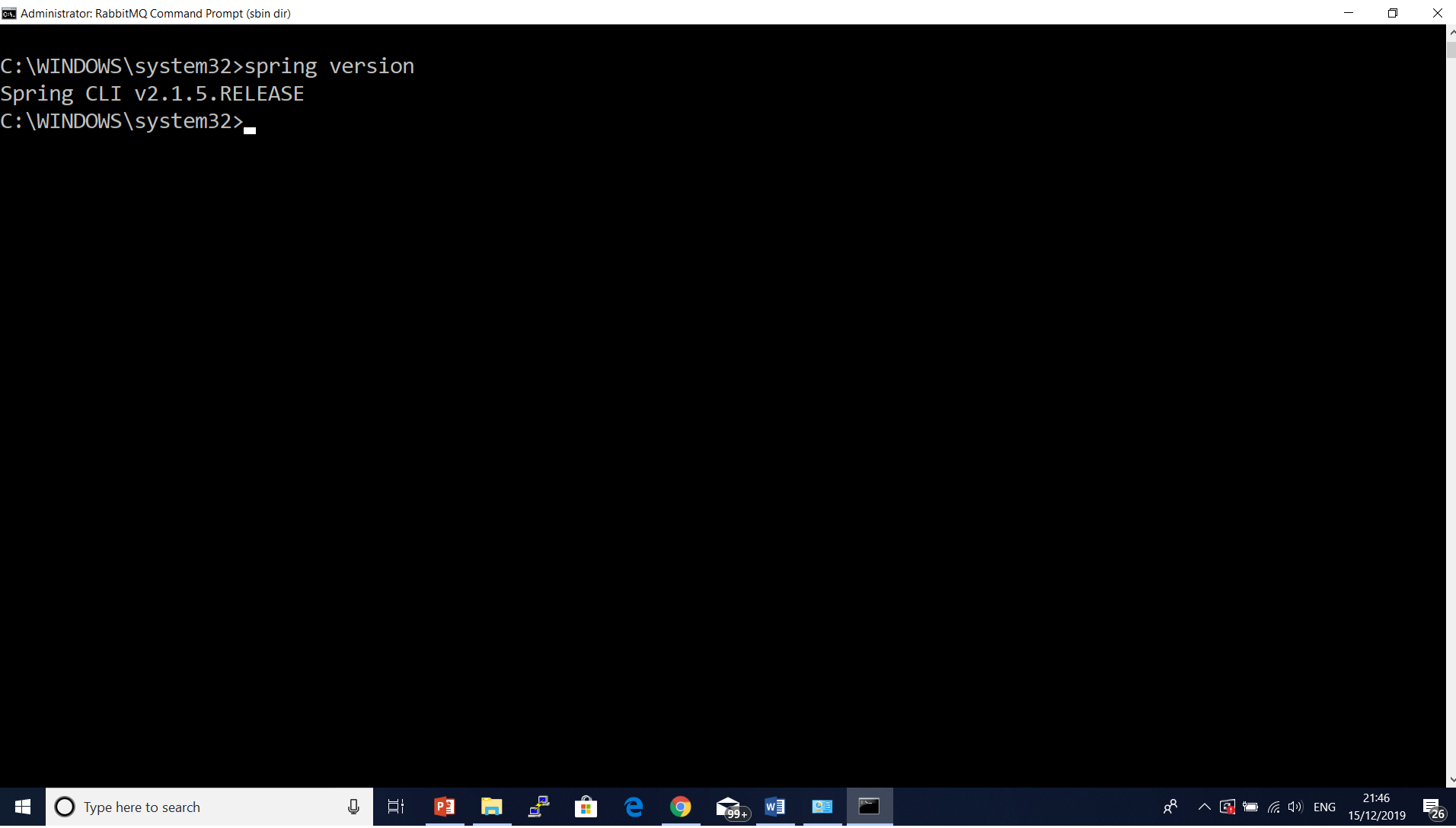
- Spring boot CLI using manual installation in my Windows 7 OS. Find the below steps.
  Step 1: Download Spring boot CLI using below link spring-boot-cli-{version}.RELEASE-bin.zip.

  Step 2: Unzip it and keep somewhere in your system. Suppose I have kept it as follows.
  C:\spring-1.4.3.RELEASE
  Now we need to set following environment variables in our system.
  1. **SPRING_HOME** with the value C:\spring-1.4.3.RELEASE
  2. **PATH** with the value C:\spring-1.4.3.RELEASE\bin
  Step 3: Now we will test our installation. Open the command prompt and type the command **spring**

SpringBoot CLI

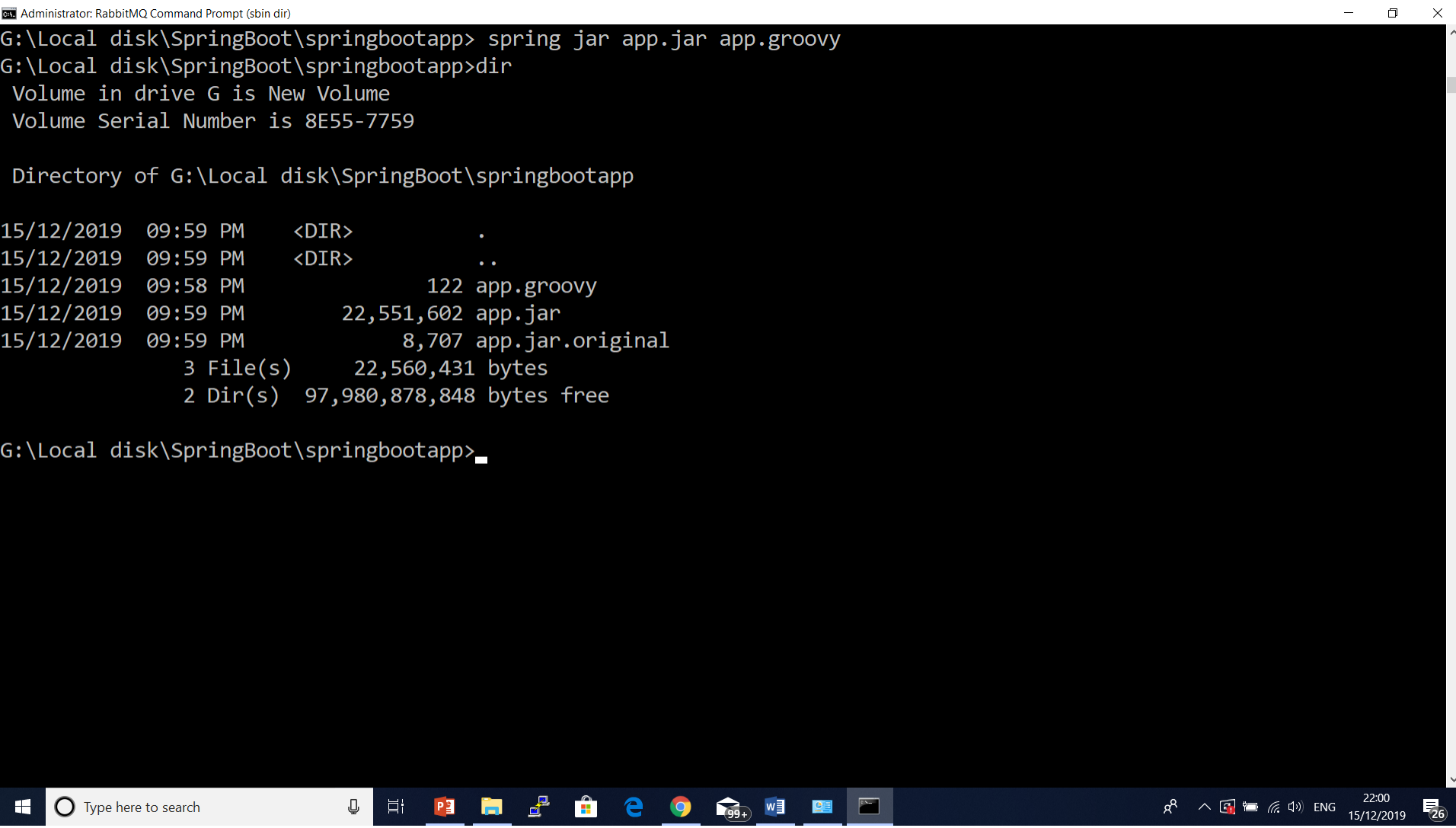


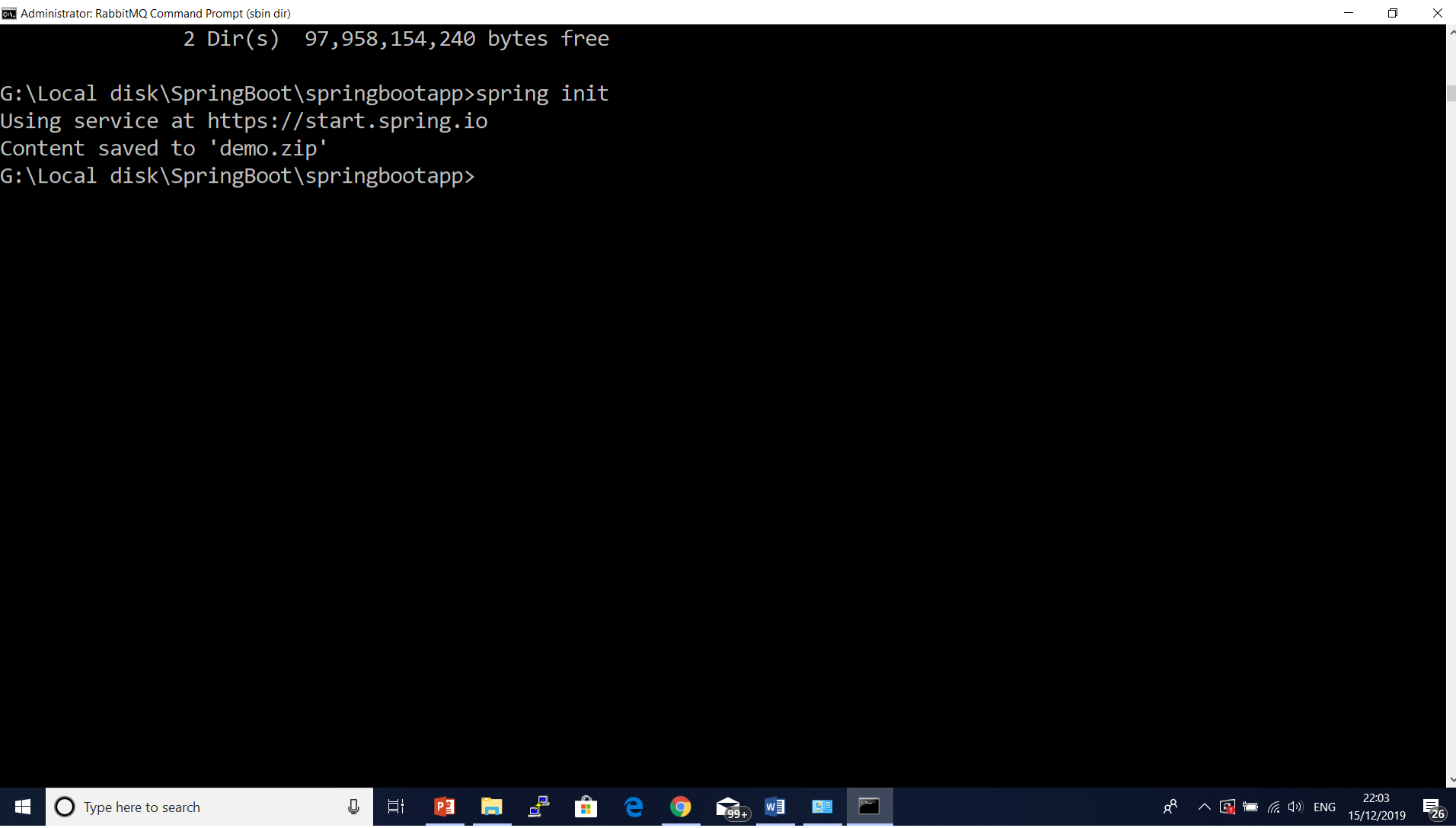

Administrator: RabbitMQ Command Prompt (sbin dir)
Microsoft Windows [Version 10.0.17134.1130]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>spring help run
spring run - Run a spring groovy script
usage: spring run [options] <files> [--] [args]
Option Description
------ -----------
--autoconfigure [Boolean] Add autoconfigure compiler transformations
(default: true)
--classpath, -cp <String> Additional classpath entries
--no-guess-dependencies Do not attempt to guess dependencies
--no-guess-imports Do not attempt to guess imports
-q, --quiet Quiet logging
-v, --verbose Verbose logging of dependency resolution
--watch Watch the specified file for changes
C:\WINDOWS\system32>
Type here to search
ENG
21:45
15/12/2019

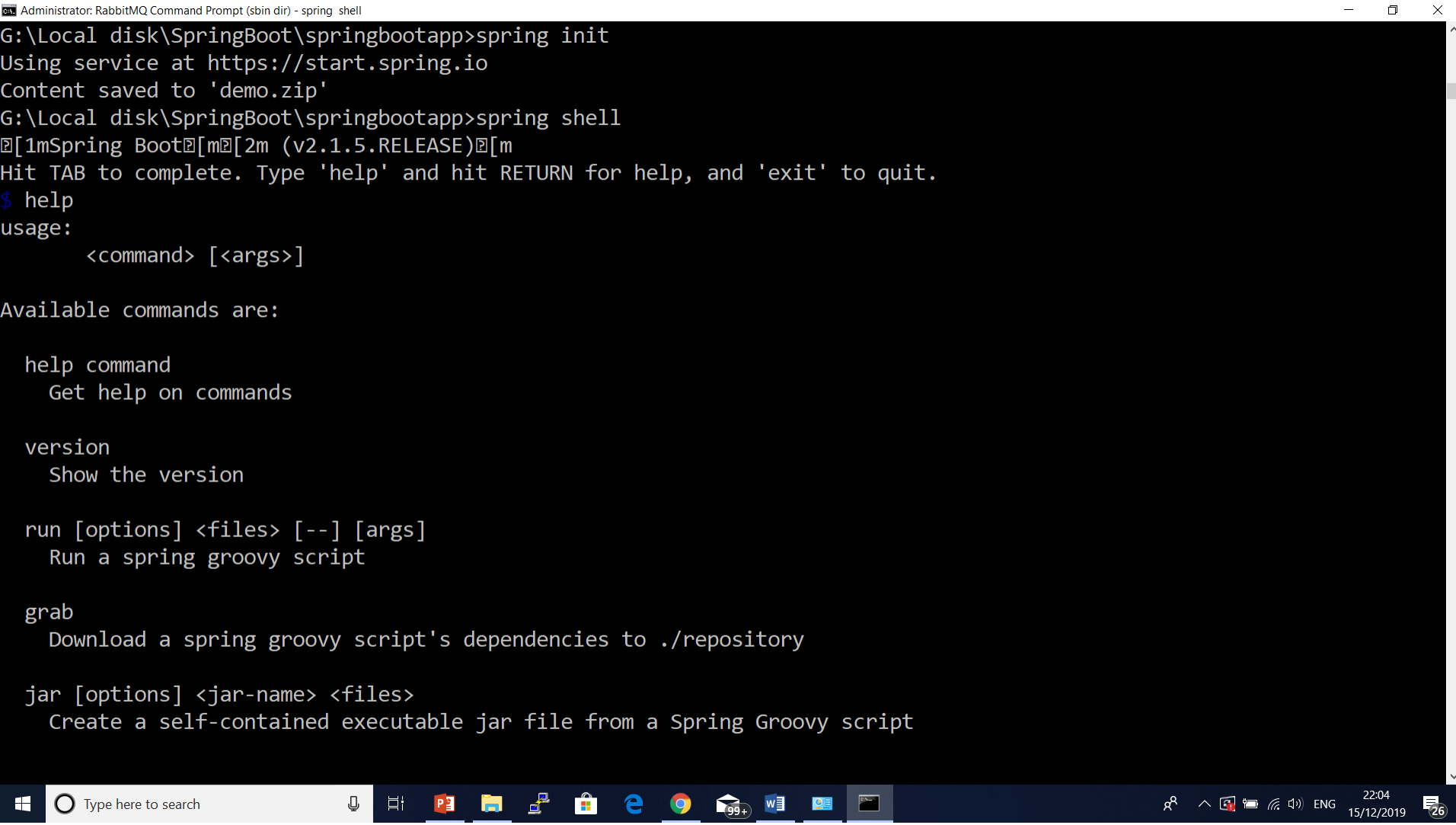

Administrator: RabbitMQ Command Prompt (sbin dir)
C:\WINDOWS\system32>spring version
Spring CLI v2.1.5.RELEASE
C:\WINDOWS\system32>
Type here to search
ENG
21:46
15/12/2019

Administrator: RabbitMQ Command Prompt (sbin dir) - spring run app.groovy

```
        at groovy.lang.GroovyClassLoader.loadClass(GroovyClassLoader.java:860)
        ... 31 more
G:\Local disk\SpringBoot\springbootapp>spring run app.groovy


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.5.RELEASE)

2019-12-15 21:58:53.862  INFO 22704 --- [       runner-0] o.s.boot.SpringApplication               : Starting applicat
ion on DESKTOP-55AGI0I with PID 22704 (started by Balasubramaniam in G:\Local disk\SpringBoot\springbootapp)
2019-12-15 21:58:53.867  INFO 22704 --- [       runner-0] o.s.boot.SpringApplication               : No active profile
 set, falling back to default profiles: default
2019-12-15 21:58:55.742  INFO 22704 --- [       runner-0] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialize
d with port(s): 8080 (http)
2019-12-15 21:58:55.809  INFO 22704 --- [       runner-0] o.apache.catalina.core.StandardService   : Starting service
[Tomcat]
2019-12-15 21:58:55.810  INFO 22704 --- [       runner-0] org.apache.catalina.core.StandardEngine  : Starting Servlet
engine: [Apache Tomcat/9.0.19]
2019-12-15 21:58:55.886  INFO 22704 --- [       runner-0] org.apache.catalina.loader.WebappLoader  : Unknown class loa
der [org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader@5f02f1e2] of class
[class org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader]
2019-12-15 21:58:55.927  INFO 22704 --- [       runner-0] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spri
ng embedded WebApplicationContext
```


Administrator: RabbitMQ Command Prompt (sbin dir)
G:\Local disk\SpringBoot\springbootapp> spring jar app.jar app.groovy
G:\Local disk\SpringBoot\springbootapp>dir
 Volume in drive G is New Volume
 Volume Serial Number is 8E55-7759
 Directory of G:\Local disk\SpringBoot\springbootapp
15/12/2019 09:59 PM   <DIR>   .
15/12/2019 09:59 PM   <DIR>   ..
15/12/2019 09:58 PM   122 app.groovy
15/12/2019 09:59 PM   22,551,602 app.jar
15/12/2019 09:59 PM   8,707 app.jar.original
               3 File(s)   22,560,431 bytes
               2 Dir(s) 97,980,878,848 bytes free
G:\Local disk\SpringBoot\springbootapp>
Type here to search
ENG
22:00
15/12/2019


Administrator: RabbitMQ Command Prompt (sbin dir)
2 Dir(s) 97,958,154,240 bytes free
G:\Local disk\SpringBoot\springbootapp>spring init
Using service at https://start.spring.io
Content saved to 'demo.zip'
G:\Local disk\SpringBoot\springbootapp>
Type here to search
ENG
22:03
15/12/2019


Administrator: RabbitMQ Command Prompt (sbin dir) - spring shell
G:\Local disk\SpringBoot\springbootapp>spring init
Using service at https://start.spring.io
Content saved to 'demo.zip'
G:\Local disk\SpringBoot\springbootapp>spring shell
[1mSpring Boot[m[2m (v2.1.5.RELEASE)[m
Hit TAB to complete. Type 'help' and hit RETURN for help, and 'exit' to quit.
$ help
usage:
<command> [<args>]
Available commands are:
help command
Get help on commands
version
Show the version
run [options] <files> [--] [args]
Run a spring groovy script
grab
Download a spring groovy script's dependencies to ./repository
jar [options] <jar-name> <files>
Create a self-contained executable jar file from a Spring Groovy script
Type here to search
ENG
22:04
15/12/2019

SpringBoot CLI

- Create a New Project using Spring Boot CLI
- spring init --dependencies=web,thymeleaf my-app.zip
- In case we want to use Gradle build tool that will build a WAR file with any specific java version then we can run the command as follows.
- spring init --build=gradle --java-version=1.8 --dependencies=web,thymeleaf --packaging=war my-app.zip

Using Embedded Shell

- Spring boot has command line completion scripts for BASH and Zsh shells. If we are using WINDOWS, spring boot provides **shell** command to launch an integrated shell. If using WINDOWS then launch integrated shell using the following command.
- spring shell
- Now we can directly run command without using spring keyword such as
- $ version
- $ test hello.groovy tests.groovy
- $ run hello.groovy

Jetty server

- Maven
- <dependency>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-web</artifactId>
- <exclusions>
- <exclusion>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-tomcat</artifactId>
- </exclusion>
- </exclusions>
- </dependency>
- <dependency>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-jetty</artifactId>
- </dependency>

Jetty server

- gradle
- configurations {
- compile.exclude module: "spring-boot-starter-tomcat"
- }
-
- dependencies {
- compile("org.springframework.boot:spring-boot-starter-web:2.0.0.BUILD-SNAPSHOT")
- compile("org.springframework.boot:spring-boot-starter-jetty:2.0.0.BUILD-SNAPSHOT")
- }

Spring Boot 2.0 Configuration

```
@Bean
    public ConfigurableServletWebServerFactory webServerFactory()
    {
        JettyServletWebServerFactory factory = new JettyServletWebServerFactory();
        factory.setPort(9000);
        factory.setContextPath("/myapp");
        factory.addErrorPages(new ErrorPage(HttpStatus.NOT_FOUND, "/notfound.html"));
        return factory;
    }
```

HTTPS Configuration

- Create your own self signed SSL certificate
- To get SSL digital certificate for our application we have two options –
- to create a self-signed certificate
- to obtain SSL certificate from certification authority(CA) we call it CA certificate.

HTTPS Configuration

- keytool -genkey -alias selfsigned_localhost_sslserver -keyalg RSA -keysize 2048 -validity 700 -keypass changeit -storepass changeit -keystore ssl-server.jks.

- To view what is inside this keystore we can again use the keytool -list command as bellow.

- keytool -list -keystore ssl-server.jks

Logging

- logging.level.org.springframework=DEBUG
- logging.level.com.howtodoinjava=DEBUG
-
- #output to a temp_folder/file
- logging.file=${java.io.tmpdir}/application.log
-
- # Logging pattern for the console
- logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %msg%n
-
- # Logging pattern for file
- logging.pattern.file= %d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%

Actuator

- Spring boot’s module Actuator allows you to monitor and manage application usages in production environment, without coding and configuration for any of them.

Actuator

ENDPOINT	USAGE
/env	Returns list of properties in current environment
/health	Returns application health information.
/auditevents	Returns all auto-configuration candidates and the reason why they ‘were’ or ‘were not’ applied.
/beans	Returns a complete list of all the Spring beans in your application.

Actuator

/trace	Returns trace logs (by default the last 100 HTTP requests).
/dump	It performs a thread dump.
/metrics	It shows several useful metrics information like JVM memory used, system CPU usage, open files, and much more.

http://localhost:7070/browser/index.html#/actuator/metrics/system.cpu.usage

Logging

```
private final Logger LOGGER = LoggerFactory.getLogger(this.getClass());

@RequestMapping("/")
public String home(Map<String, Object> model) {

    LOGGER.debug("This is a debug message");
    LOGGER.info("This is an info message");
    LOGGER.warn("This is a warn message");
    LOGGER.error("This is an error message");

    model.put("message", "HowToDoInJava Reader !!");
    return "index";
}
```

Spring Cache

- The Spring Framework provides support for transparently adding caching to an application. At its core, the abstraction applies caching to ecutions based on the information available in the cache.
- This support is available from Spring Framework 3.1 and there is significant improvement provided in the Spring 4.1.
- This is an abstract framework where Spring only provides the layer where other third party caching implementations can be easily plugged for storing data means cache storage is not implemented by Spring, where as enabling and caching is supported by spring out of the box.
- Caching is supported for the methods and it works well if the methods return the same result for the given input for the multiple invocations.

Supported cache providers in Spring

- Generic
- JCache (JSR-107)
- EhCache 2.x
- Hazelcast
- Infinispan
- Couchbase
- Redis
- Caffeine
- Guava
- Simple

Use Caching With Annotations

- After enabling the cache we can use following list of declarative annotations.
- @Cacheable
- @CacheEvict
- @CachePut
- @Caching
- @CacheConfig

Use Caching With Annotations

- @EnableCaching to enable caching because caching in spring is not enabled by default. The caching feature can be declaratively enabled by simply adding the @EnableCaching annotation to any of the configuration classes.
- @Cacheable: It is one of the most important and common annotation for caching the requests.
- If you annotate a method with @Cacheable, if multiple requests are received by the application, then this annotation will not execute the method multiple times, instead it will send the result from the cached storage.

Use Caching With Annotations

- @CacheEvict:
- If we annotate all methods with @Cacheable then the size of cache may be some problem.
- We don’t want to populate the cache with values that we don’t need often.
- Caches can grow quite large, quite fast, and we could be holding on to a lot of stale or unused data.
- @CacheEvict annotation is used for removing a single cache or clearing the entire cache from the cache storage so that fresh values can be loaded into the cache again:

Use Caching With Annotations

- **@CachePut:**
- @CachePut annotation helps for updating the cache with the latest execution without stopping the method execution. The difference between @Cacheable and @CachePut is that @Cacheable will skip running the method, whereas @CachePut will actually run the method and then put its results in the cache.

Use Caching With Annotations

- **@Caching:**
- What if you want to use multiple annotations of the same type for caching a method? @Caching annotation used for grouping multiple annotations of the same type together when one annotation is not sufficient for the specifying the suitable condition. For example, you can put multiple @CacheEvict or @CachePut annotation inside @Caching to narrow down your conditions as you need.

Use Caching With Annotations

- **@CacheConfig:**
- You can annotate @CacheConfig at the class level to avoid repeated mentioning in each method. For example, in the class level you can provide the cache name and in the method you just annotate with @Cacheable annotation.
- .

Spring Boot Components

- Spring Boot Actuator
- This project, when added, will enable certain enterprise features (Security, Metrics, Default Error pages) to your application.
- As the auto configure module it uses auto detection to detect certain frameworks/features of your application.
- For an example, you can see all the REST Services defined in a web application using Actuator

Hateos

- The Spring HATEOAS project is a library of APIs that we can use to easily create REST representations that follow the principle of HATEOAS (Hypertext as the Engine of Application State).
- **Spring HATEOAS offers three abstractions for creating the URI – *ResourceSupport, Link,* and *ControllerLinkBuilder***.

Spring Boot Components

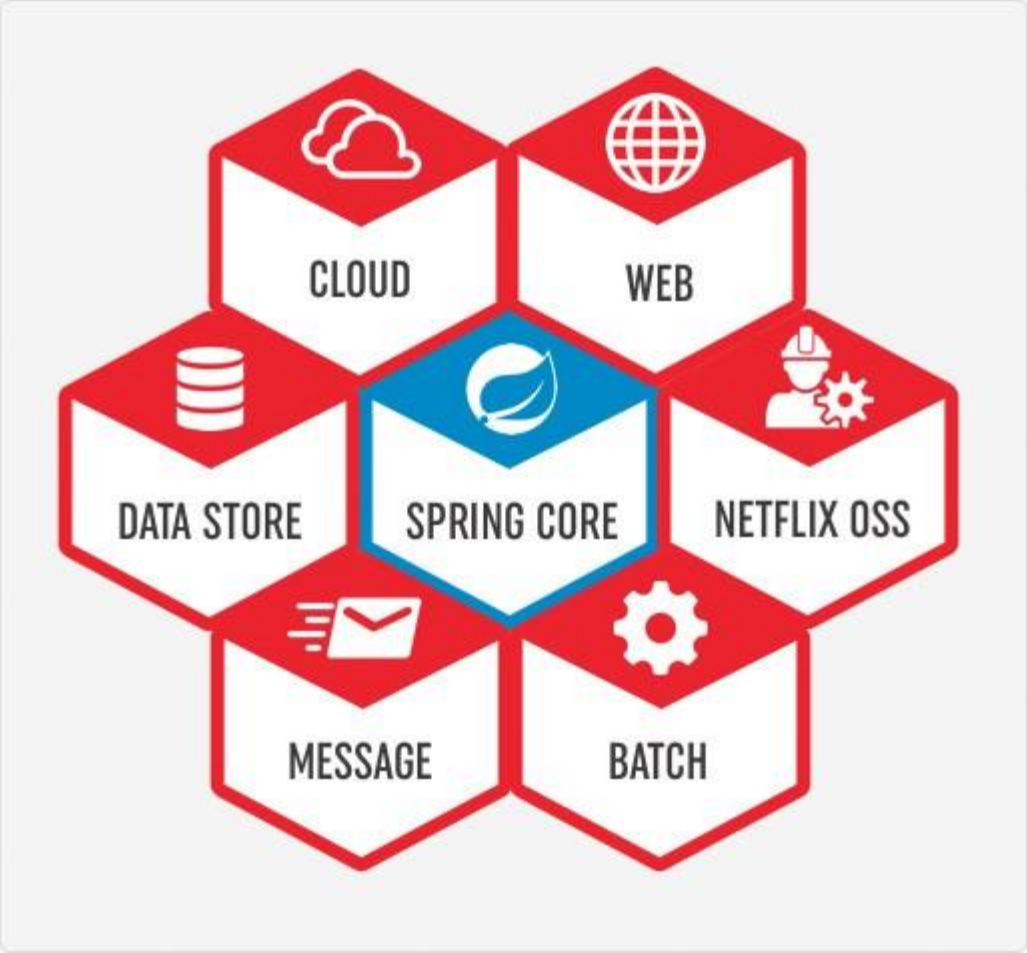
- Spring Boot Starters
- Different quick start projects to include as a dependency in your maven or gradle build file.
- It will have the needed dependencies for that type of application. Currently there are many starter projects and many more are expected to be added.

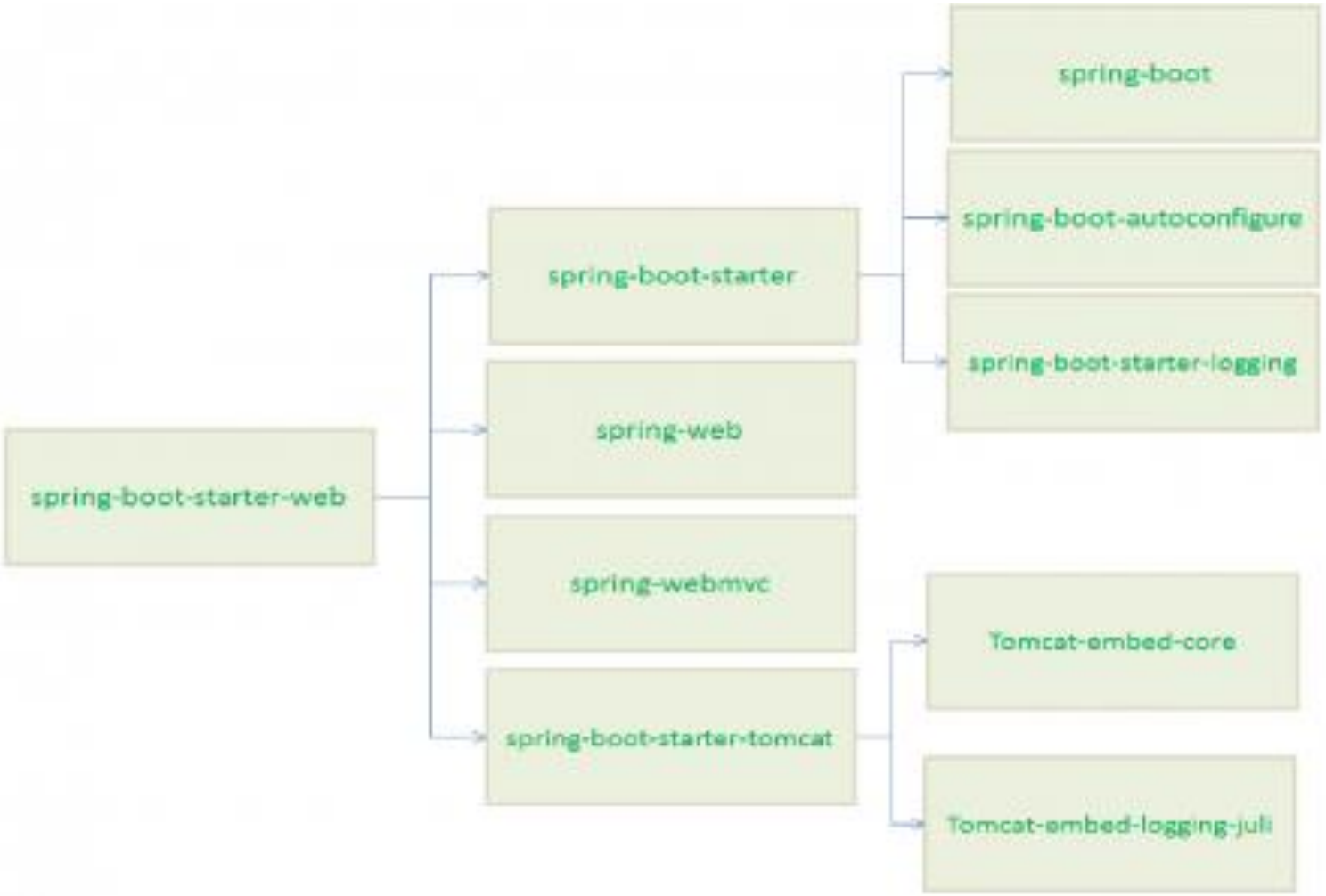
Spring Boot Components

- For example, if you go to a student page, you will see
- Student profile
- Links to Edit and Delete Student details
- Links to see details of other students
- Link to see details of the courses and grades of the student
- HATEOAS brings the same concepts to RESTful Web Services.

Spring Boot Components

- Spring Boot Starters




spring-boot
spring-boot-autoconfigure
spring-boot-starter
spring-boot-starter-logging
spring-web
spring-boot-starter-web
spring-webmvc
Tomcat-embed-core
spring-boot-starter-tomcat
Tomcat-embed-logging-juli

Spring Boot Components

- Spring Boot Tools
- The Maven and Gradle build tool as well as the custom Spring Boot Loader (used in the single executable jar/war) is included in this project.

Spring Boot Installation

Requirements:

- Java (>=1.6) + (for Java projects)
- Maven 3.2+ or Gradle 1.12+

! **Spring Tool Suite has some nice features for Java projects**

! **Download: https://start.spring.io/spring.zip**

! **Unzip the distro (approx. 10MB), and find bin/ directory**

! **(You can also install Spring Boot CLI with gvm, brew or MacPorts)**

Springboot 2.x Features

1.Java 8 is minimum version

2.Tomcat version 8.5 is minimum

3.Hibernate version 5.2 is minimum

4.Gradle version 3.4 is minimum

5.Added SpringBoot Starters for WebFlux and reactive support for Cassandra, MongoDB and Redis.

6.AutoConfiguration

Springboot 2.x Features

- Actuator Endpoint change:

- Before 2.*: http://localhost:8080/business-customer/profile/env will give the details.

- From 2.*: http://localhost:8080/business-customer/profile/actuator/env will give the details.

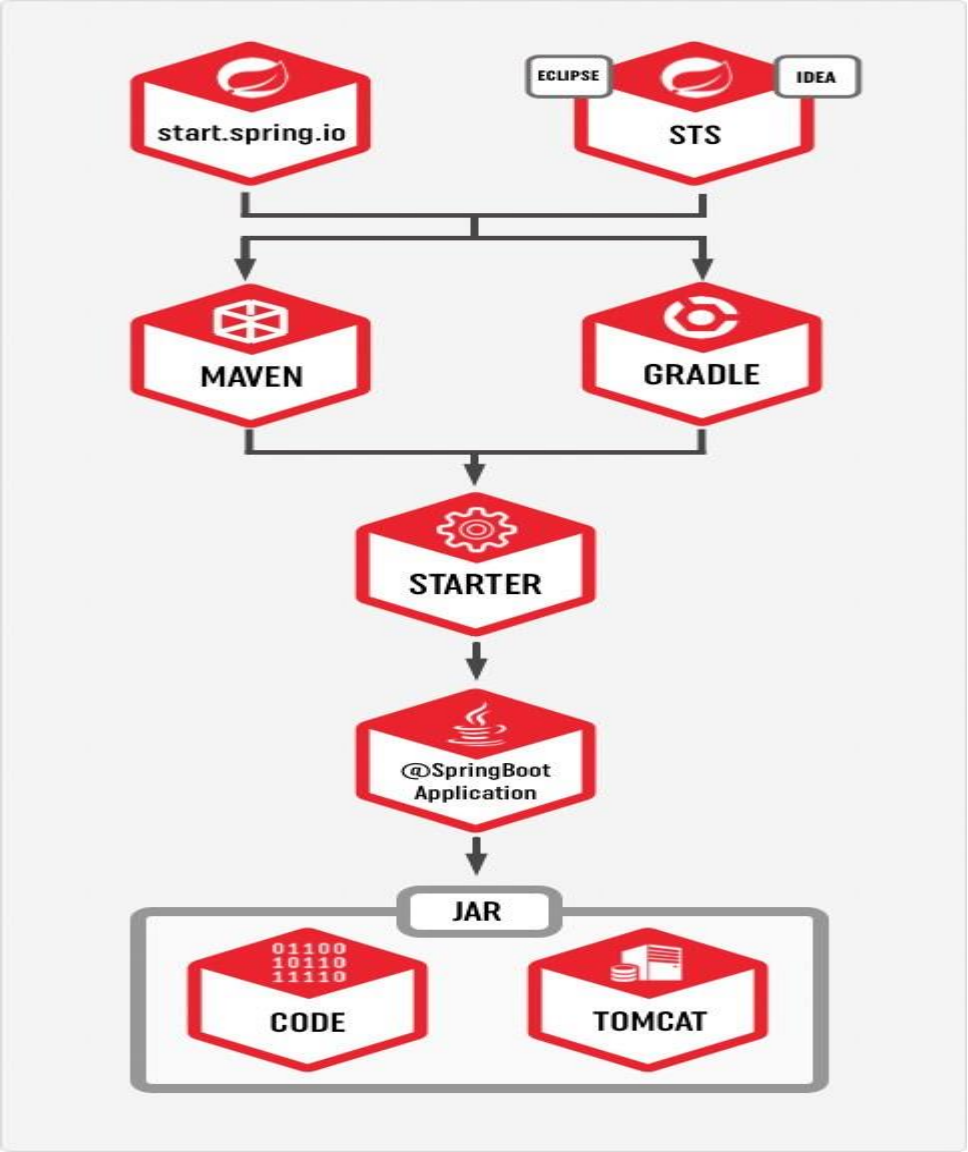
Springboot 2.x Features

- Endpoint properties in application.properties (to enable all endpoints)
- management.endpoints.web.exposure.include=* management.endpoints.web.exposure.exclude=loggers
- Connection Pool by default:
- Before 2.*: tomcat CP
- After 2.: HikariCP (from SpringBoot 2. You don't need to add HikariCP dependency and its config bean creation and its properties changes.)
- Migration: https://spring.io/blog/2018/03/12/upgrading-start-spring-io-to-spring-boot-2

Servlet Containers

Name	Servlet Version
Tomcat 8.5	3.1
Jetty 9.4	3.1
Undertow 1.4	3.1

How to use Spring Boot



gradle

- gradlew build --refresh-dependencies

Profile

- application-dev.properties
- application-qa.properties
- application-stage.properties
- application-prod.properties
- At the time of launching the Java application
- -Dspring.profiles.active=qa - in the VM properties, OR
- Do the following in the application.properties file
- spring.application.profiles=qa.

Annotations

- **OracleConfiguration will only be loaded if the class DataSource is present, in which case we can assume the application will use a database:**
- @Configuration
- @ConditionalOnClass(DataSource.class)
- class OracleAutoconfiguration {
- //...
- }

- **This bean to be created if a bean called *dataSource* is present and if a bean called *entityManagerFactory* is not already defined:**
- @Bean
- @ConditionalOnBean(name = "dataSource")
- LocalContainerEntityManagerFactoryBean entityManagerFactory() {
- // ...
- }

Annotations

- *transactionManager* bean that will only be loaded if a bean of type *JpaTransactionManager* is not already defined:
- @Bean
- @ConditionalOnMissingBean(type = "JpaTransactionManager")
- JpaTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {
- JpaTransactionManager transactionManager = new JpaTransactionManager();
- transactionManager.setEntityManagerFactory(entityManagerFactory);
- return transactionManager;
- }

Annotations

- @Bean
- @ConditionalOnProperty(
- name = "usemysql",
- havingValue = "local"
-)
- DataSource dataSource() {
- // ...
- }

Annotations

- @ConditionalOnResource(resources = "classpath:mysql.properties")
- Properties additionalProperties() {
- 	// ...
- }
- @Bean
- @ConditionalOnExpression("${usemysql} && ${mysqlserver == 'local'}")
- DataSource dataSource() {
- 	// ...
- }

Annotations

- @Conditional(HibernateCondition.class)
- Properties additionalProperties() {
- 	//...
- }

What is Pivotal

Vmware and EMC spin out

- vFabric from vmware
- Greenplum MPP DB and Hadoop from EMC
- Agile development acquisitions Pivotal Labs and Xtreme Labs

Paul Maritz CEO

- Former CEO vmware.

Custodian of various OSS projects including Spring

- and RabbitMQ, Redis, CloudFondry, Groovy/Grails, …

Commercial interests in Big/Fast Data, PaaS, Enterprise tooling

- Spring is an enabling technology

Installing the Spring Boot CLI

- The Spring Boot CLI (Command Line Interface) is a command line tool that you can use to quickly prototype with Spring.
- It lets you run Groovy scripts, which means that you have a familiar Java-like syntax without so much boilerplate code.
- You do not need to use the CLI to work with Spring Boot, but it is definitely the quickest way to get a Spring application off the ground.

Manual Installation

- You can download the Spring CLI distribution from the Spring software repository:
- spring-boot-cli-2.0.4.RELEASE-bin.zip
- spring-boot-cli-2.0.4.RELEASE-bin.tar.gz

Manual Installation

- set PATH=D:\spring-boot-cli-1.2.3.RELEASE\bin;%PATH%

```
spring –version
Spring --help
```

```
@RestController
class HelloWorld {
  @RequestMapping("/")
  String hello() {
    "Hello JournalDev World."
  }
}
```

Name this file as HelloWorld.groovy. Here ".groovy" extension is mandatory.

Manual Installation

```
10-09-2018  12:33    <DIR>          .metadata
10-09-2018  12:59    <DIR>          .recommenders
10-09-2018  12:33    <DIR>          RemoteSystemsTempFiles
10-09-2018  20:45    <DIR>          Servers
10-09-2018  21:00    <DIR>          SpringBootApp
10-09-2018  12:34    <DIR>          SpringCoreDemo
10-09-2018  20:49    <DIR>          Test
11-09-2018  00:11               133 Test.groovy
               1 File(s)            133 bytes
               9 Dir(s)  99,302,350,848 bytes free

F:\Scope_SpringBoot_2018>spring run Test.groovy
Resolving dependencies.......

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.4.RELEASE)
```

Manual Installation

```
See 'spring help <command>' for more information on a specific command.

F:\Scope_SpringBoot_2018>spring run Test.groovy  -- --server.port=8484


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.4.RELEASE)

2018-09-11 00:22:00.164  INFO 25272 --- [       runner-0] o.s.boot.SpringApplication
        : Starting application on DESKTOP-55AGI0I with PID 25272 (started by Balasubraman
iam in F:\Scope_SpringBoot_2018)
2018-09-11 00:22:00.170  INFO 25272 --- [       runner-0] o.s.boot.SpringApplication
        : No active profile set, falling back to default profiles: default
2018-09-11 00:22:00.441  INFO 25272 --- [       runner-0] ConfigServletWebServerApplicatio
nContext : Refreshing org.springframework.boot.web.servlet.context.AnnotationConfigServlet
WebServerApplicationContext@5e25c1f1: startup date [Tue Sep 11 00:22:00 IST 2018]; root of
 context hierarchy
```

Packaging For Production

Maven plugin (using spring-boot-starter-parent):

- **Gradle plugin:**

```
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-
plugin</artifactId>
</plugin>
```

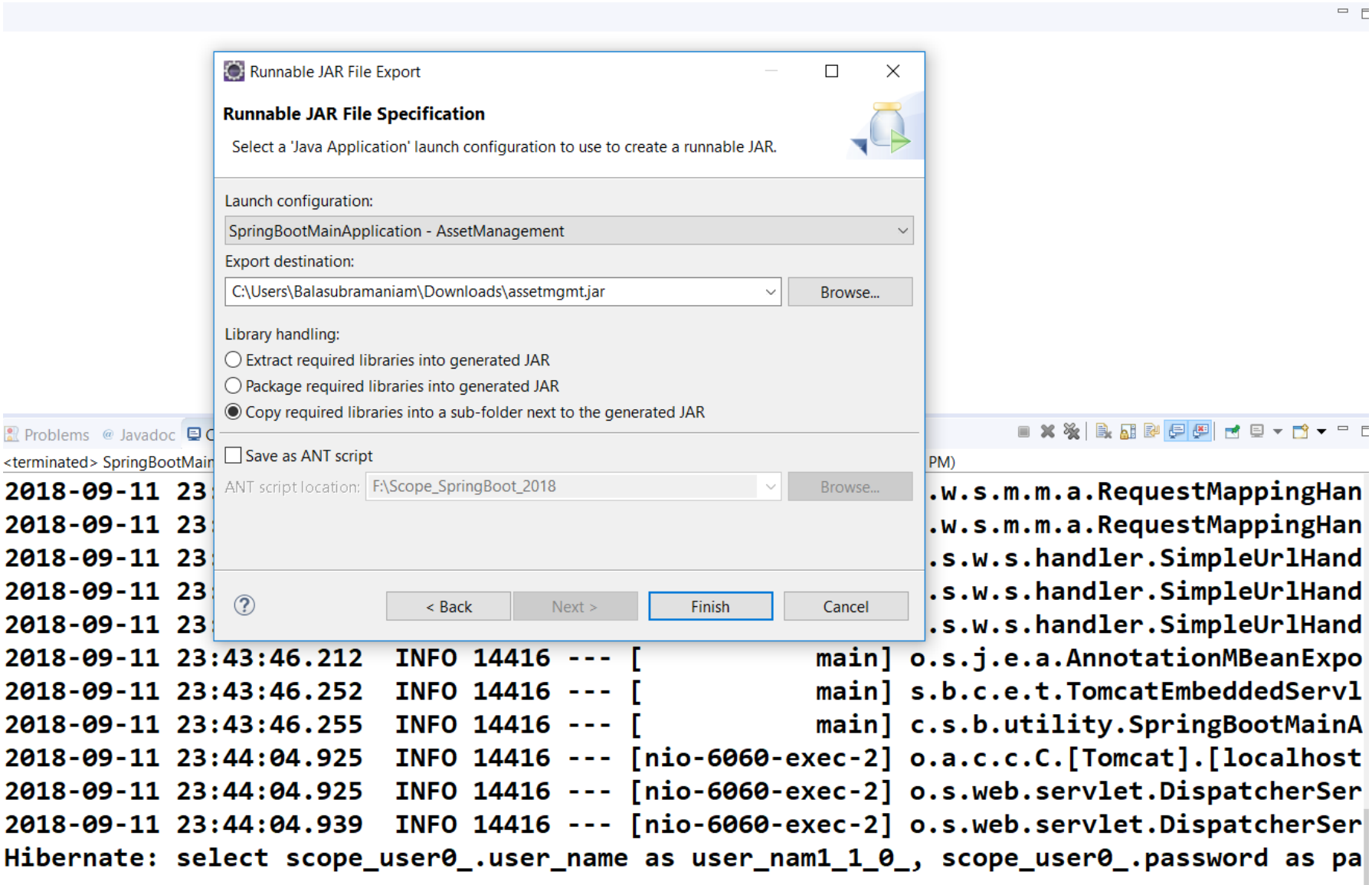
```
$ mvn package
```

```
apply plugin: 'spring-boot'
```

```
$ gradle build
```

Packaging For Production

- **Packaging For Production**
- $ java -jar yourapp.jar
- ! **Easy to understand structure**
- ! **No unpacking or start scripts required**
- ! **Typical REST app ~10Mb**
- ! **Cloud Foundry friendly (works & fast to upload)**


Runnable JAR File Export
Runnable JAR File Specification
Select a 'Java Application' launch configuration to use to create a runnable JAR.
Launch configuration:
SpringBootMainApplication - AssetManagement
Export destination:
C:\Users\Balasubramaniam\Downloads\assetmgmt.jar
Browse...
Library handling:
Extract required libraries into generated JAR
Package required libraries into generated JAR
Copy required libraries into a sub-folder next to the generated JAR
Save as ANT script
ANT script location:
F:\Scope_SpringBoot_2018
Browse...
< Back
Next >
Finish
Cancel
Problems
Javadoc
<terminated> SpringBootMain
PM)
2018-09-11 23:43:46.212 INFO 14416 --- [main] o.s.j.e.a.AnnotationMBeanExpo
2018-09-11 23:43:46.252 INFO 14416 --- [main] s.b.c.e.t.TomcatEmbeddedServl
2018-09-11 23:43:46.255 INFO 14416 --- [main] c.s.b.utility.SpringBootMainA
2018-09-11 23:44:04.925 INFO 14416 --- [nio-6060-exec-2] o.a.c.c.C.[Tomcat].[localhost
2018-09-11 23:44:04.925 INFO 14416 --- [nio-6060-exec-2] o.s.web.servlet.DispatcherSer
2018-09-11 23:44:04.939 INFO 14416 --- [nio-6060-exec-2] o.s.web.servlet.DispatcherSer
Hibernate: select scope_user0_.user_name as user_nam1_1_0_, scope_user0_.password as pa

```
C:\Users\Balasubramaniam\Downloads>java -jar assetmgmt.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  (v1.5.16.BUILD-SNAPSHOT)

2018-09-11 23:50:56.827  INFO 12020 --- [           main] c.s.b.utility.SpringBootMainApplication  : Starting SpringBo
otMainApplication on DESKTOP-55AGI0I with PID 12020 (C:\Users\Balasubramaniam\Downloads\assetmgmt.jar started by Balas
ubramaniam in C:\Users\Balasubramaniam\Downloads)
2018-09-11 23:50:56.830  INFO 12020 --- [           main] c.s.b.utility.SpringBootMainApplication  : No active profile
 set, falling back to default profiles: default
2018-09-11 23:50:56.908  INFO 12020 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing org.sp
ringframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@641147d0: startup date [Tue Sep 11 2
3:50:56 IST 2018]; root of context hierarchy
2018-09-11 23:50:57.843  INFO 12020 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springf
ramework.transaction.annotation.ProxyTransactionManagementConfiguration' of type [org.springframework.transaction.anno
tation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$7788aebc] is not eligible for getting processed
 by all BeanPostProcessors (for example: not eligible for auto-proxying)
2018-09-11 23:50:58.366  INFO 12020 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialize
d with port(s): 6060 (http)
2018-09-11 23:50:58.403  INFO 12020 --- [           main] o.apache.catalina.core.StandardService   : Starting service
```

Logging Configuration in Spring Boot

- Spring Boot provides support for logging and uses Commons Logging for all internal logging, but leaves the underlying log implementation open.
- Default configurations are provided for **Java Util Logging, Log4J2 and Logback**.
- In each case loggers are pre-configured to use console output with optional file output also available.

How to Deploy Spring Boot Application to Cloud Foundry Platform

- Cloud Foundry is an open-source platform as a service (PaaS) that provides you with a choice of clouds, developer frameworks, and application services.
- It is open source and it is governed by the Cloud Foundry Foundation.
- The original Cloud Foundry was developed by VMware and currently it is managed by Pivotal, a joint venture company by GE, EMC and VMware.

Cloud Foundary Providers

- Pivotal Cloud Foundry
- IBM Bluemix
- HPE Helion Stackato 4.0
- Atos Canopy
- CenturyLink App Fog
- GE Predix
- Huawei FusionStage
- SAP Cloud Platform
- Swisscom Application Cloud

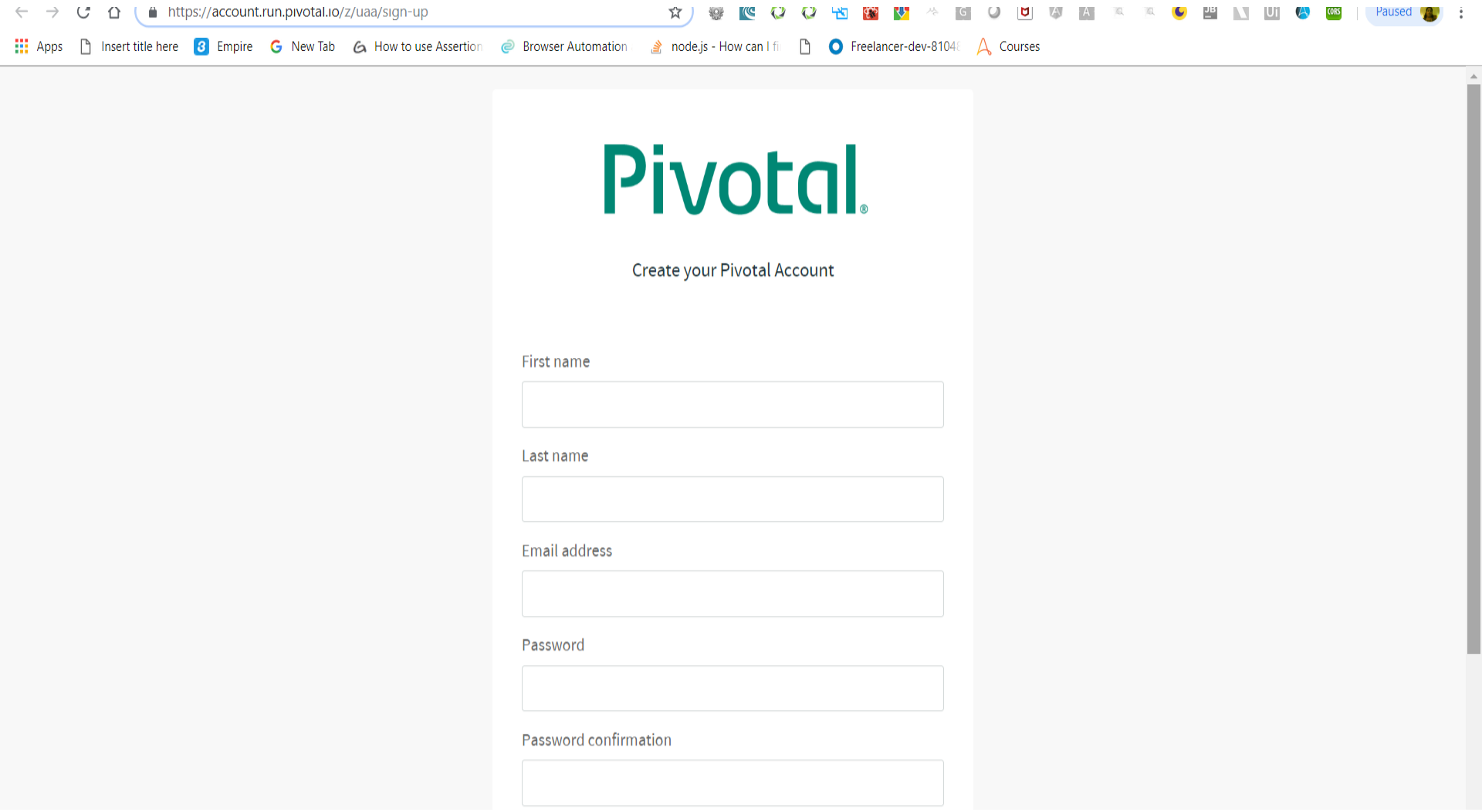
Cloud Foundry Installation for Windows

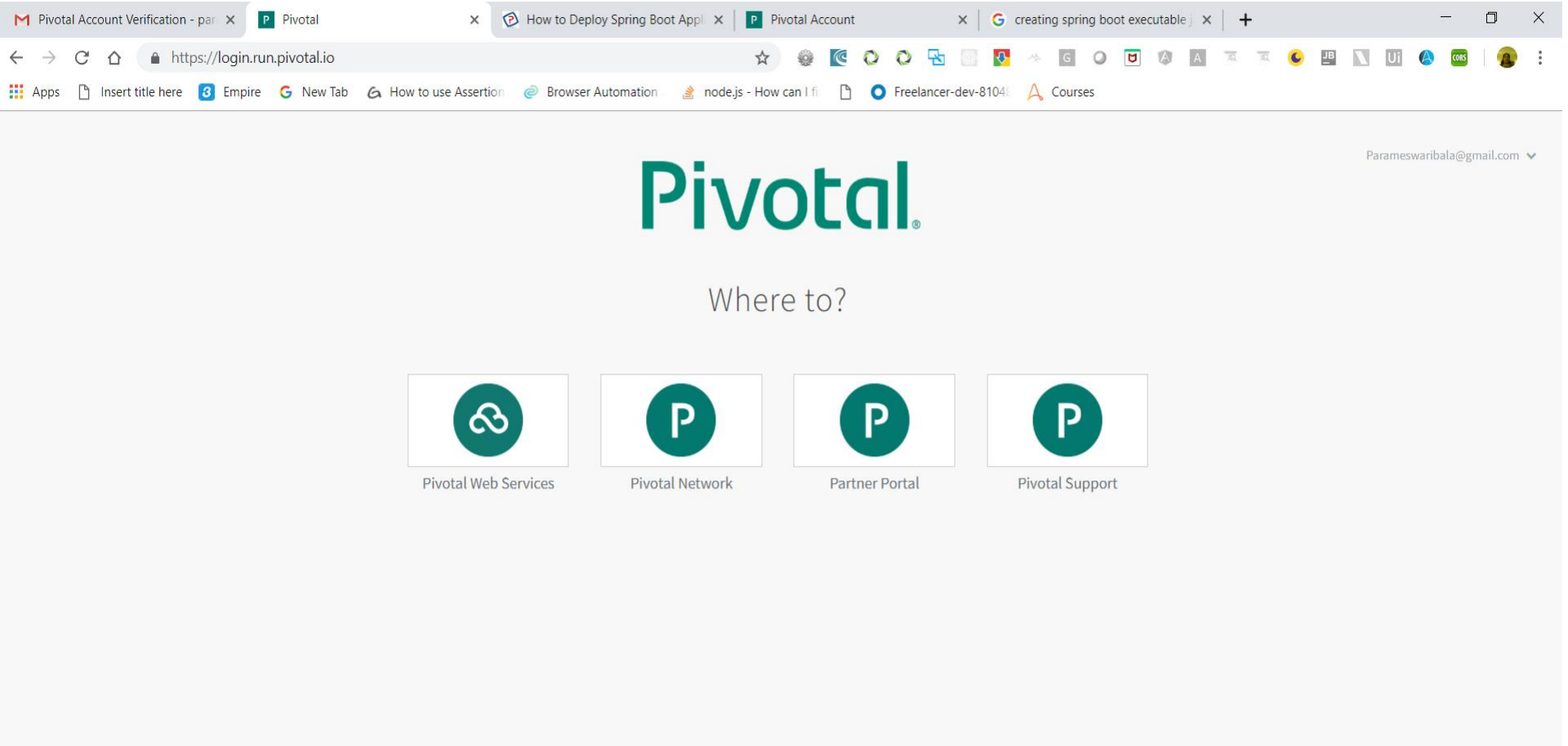
- Download the CF Windows installer. It will prompt for the download. Save the zip file distribution.
- Unpack the zip file to a suitable place in your workstation.
- After successfully **unzip** operation, double cick on the cf CLI executable.
- When prompted, click **Install**, then Close.

Cloud Foundry Installation for Windows

- Verify the installation by opening a terminal window and type cf. If your installation was successful, the cf CLI help listing appears.
- This indicates that you are ready to go with any cloud foundry platform from your local workstation.

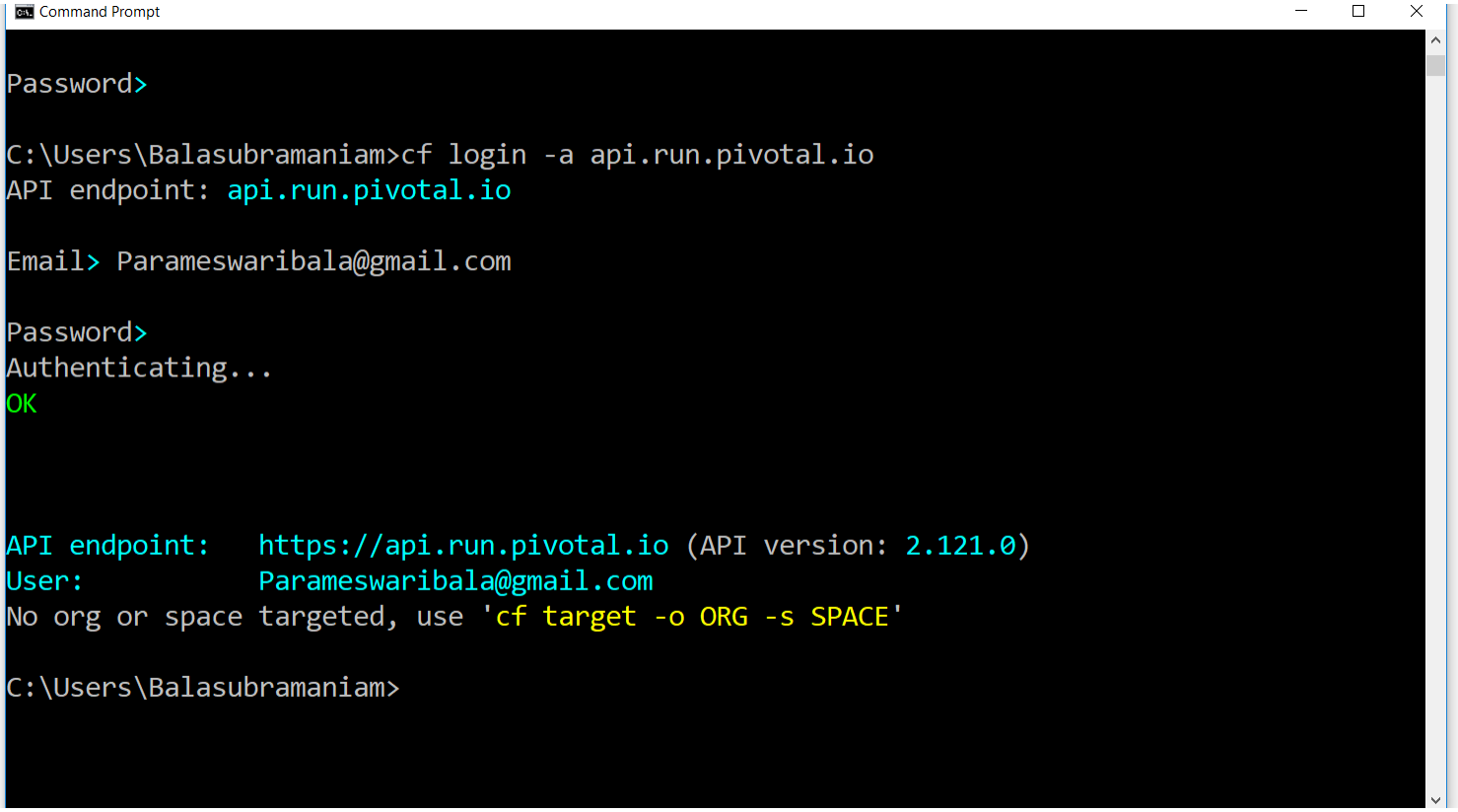
```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Balasubramaniam>cf
cf version 6.38.0+7ddf0aadd.2018-08-07, Cloud Foundry command line tool
Usage: cf [global options] command [arguments...] [command options]

Before getting started:
  config    login,l      target,t
  help,h    logout,lo

Application lifecycle:
  apps,a        run-task,rt    events
  push,p        logs           set-env,se
  start,st      ssh            create-app-manifest
  stop,sp       app
  restart,rs    env,e
  restage,rg    scale

Services integration:
  marketplace,m        create-user-provided-service,cups
  services,s           update-user-provided-service,uups
```

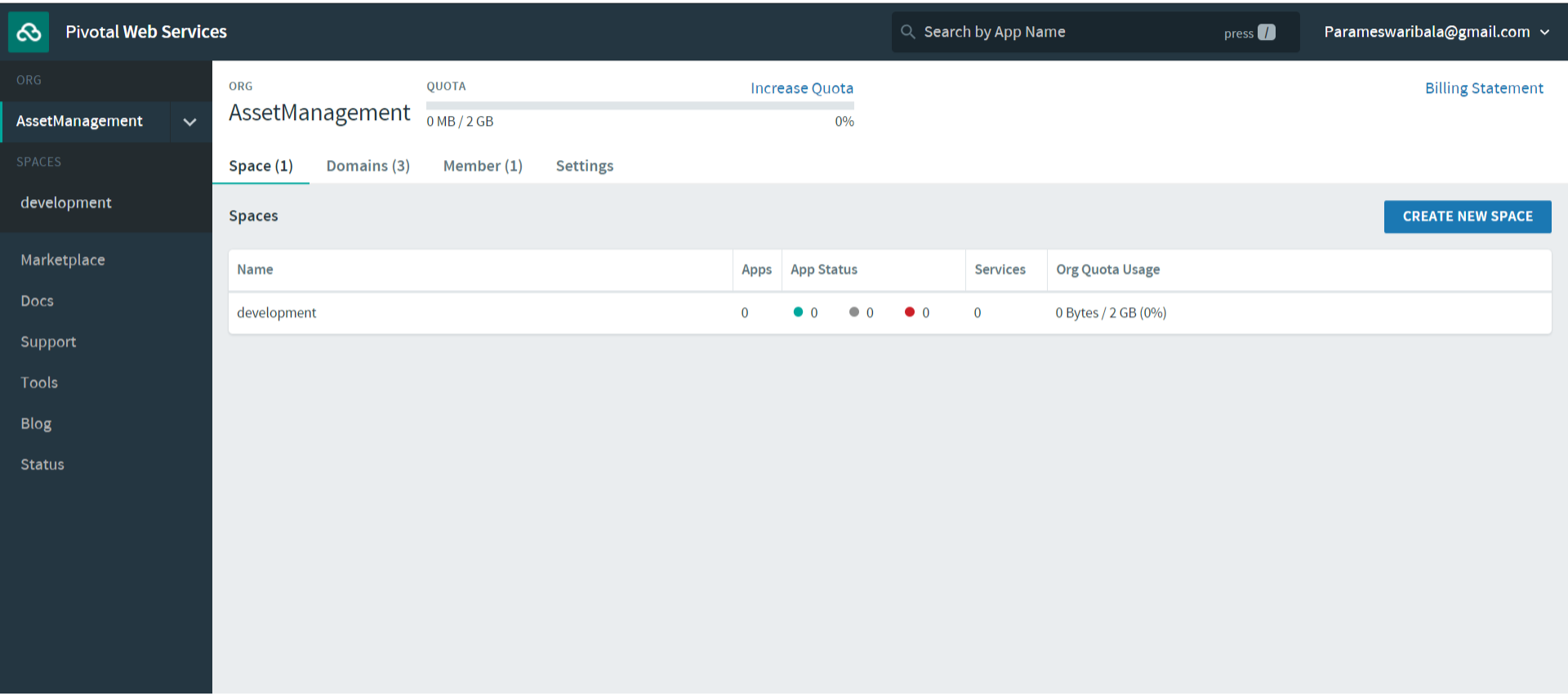

https://account.run.pivotal.io/z/uaa/sign-up
Paused
Apps
Insert title here
Empire
New Tab
How to use Assertion
Browser Automation
node.js - How can I fi
Freelancer-dev-8104
Courses
Pivotal
Create your Pivotal Account
First name
Last name
Email address
Password
Password confirmation


Pivotal Account Verification - par
Pivotal
How to Deploy Spring Boot Appl
Pivotal Account
creating spring boot executable j
https://login.run.pivotal.io
Apps
Insert title here
Empire
New Tab
How to use Assertion
Browser Automation
node.js - How can I fi
Freelancer-dev-8104
Courses
Parameswaribala@gmail.com
Pivotal
Where to?
Pivotal Web Services
Pivotal Network
Partner Portal
Pivotal Support

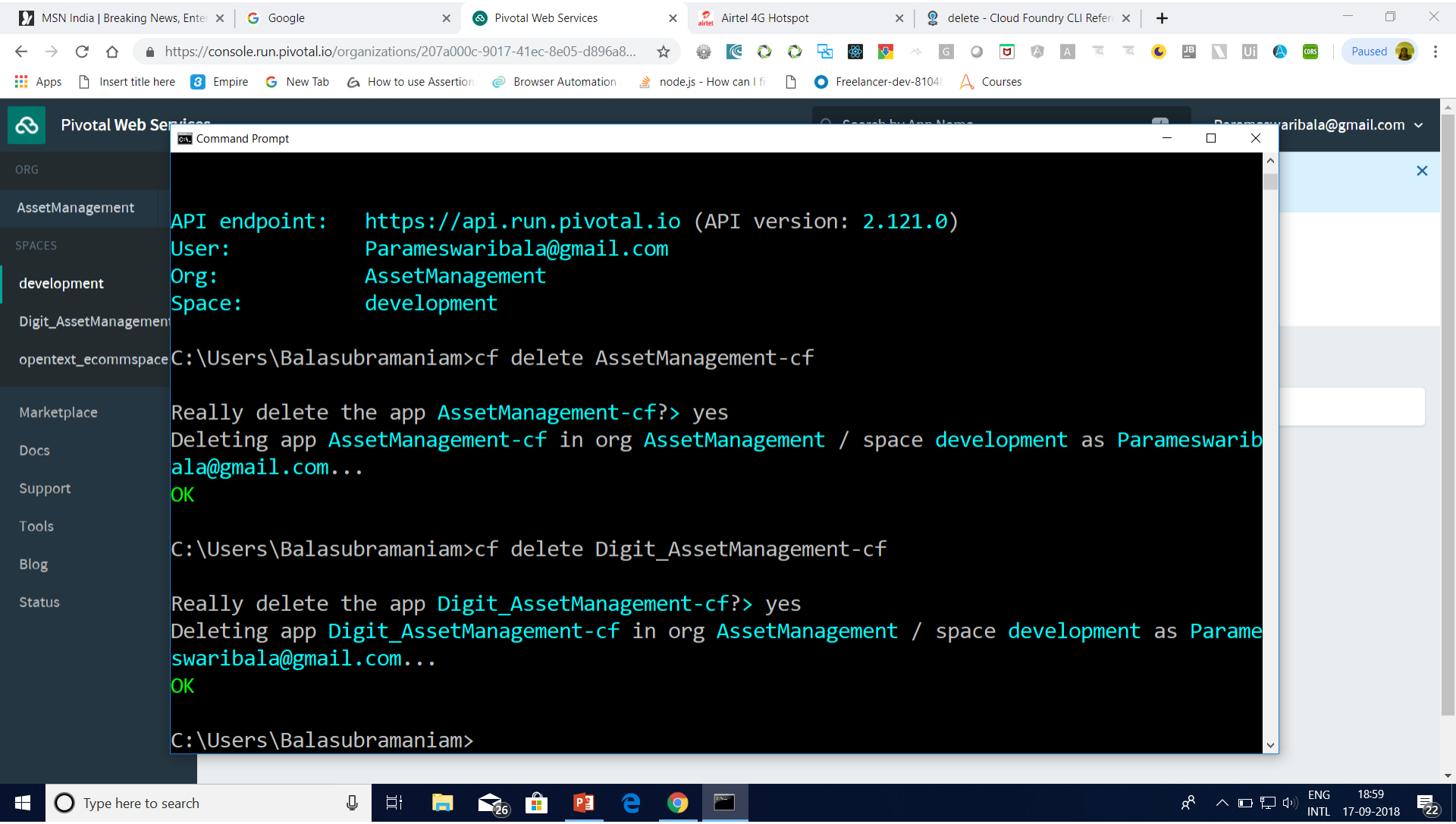
Login and logout from PWS Console using CLI

- Login to PWS – We will use cf login -a api.run.pivotal.io
- Logout from PWS Console – We will use command cf logout to logout from the platform, once we have all the work done for that session.


Command Prompt
Password>
C:\Users\Balasubramaniam>cf login -a api.run.pivotal.io
API endpoint: api.run.pivotal.io
Email> Parameswaribala@gmail.com
Password>
Authenticating...
OK
API endpoint: https://api.run.pivotal.io (API version: 2.121.0)
User: Parameswaribala@gmail.com
No org or space targeted, use 'cf target -o ORG -s SPACE'
C:\Users\Balasubramaniam>


Pivotal Web Services
Search by App Name
press /
Parameswaribala@gmail.com
ORG
AssetManagement
SPACES
development
Marketplace
Docs
Support
Tools
Blog
Status
ORG
AssetManagement
QUOTA
Increase Quota
0 MB / 2 GB
0%
Billing Statement
Space (1)
Domains (3)
Member (1)
Settings
Spaces
CREATE NEW SPACE
Name
Apps
App Status
Services
Org Quota Usage
development
0
0
0
0
0
0 Bytes / 2 GB (0%)

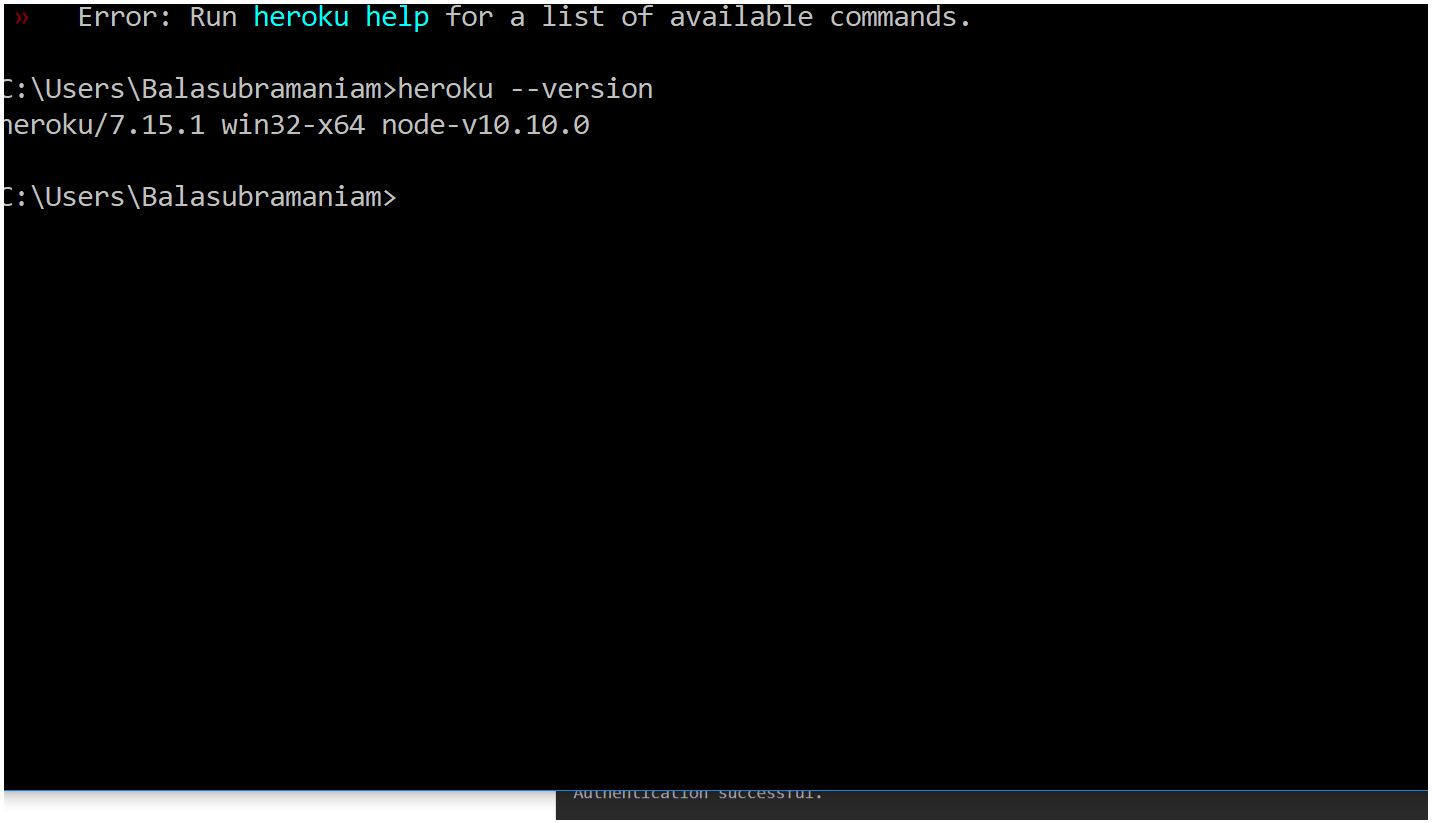
```
Targeted space development



API endpoint:   https://api.run.pivotal.io (API version: 2.121.0)
User:           Parameswaribala@gmail.com
Org:            AssetManagement
Space:          development

C:\Users\Balasubramaniam>cf push my-app
Pushing app my-app to org AssetManagement / space development as Parameswaribala@gmail.com
...
Getting app info...
The app cannot be mapped to route my-app.cfapps.io because the route exists in a different
 space.
FAILED

C:\Users\Balasubramaniam>cf push my-app.cfapps
Pushing app my-app.cfapps to org AssetManagement / space development as Parameswaribala@gm
ail.com...
Getting app info...
```

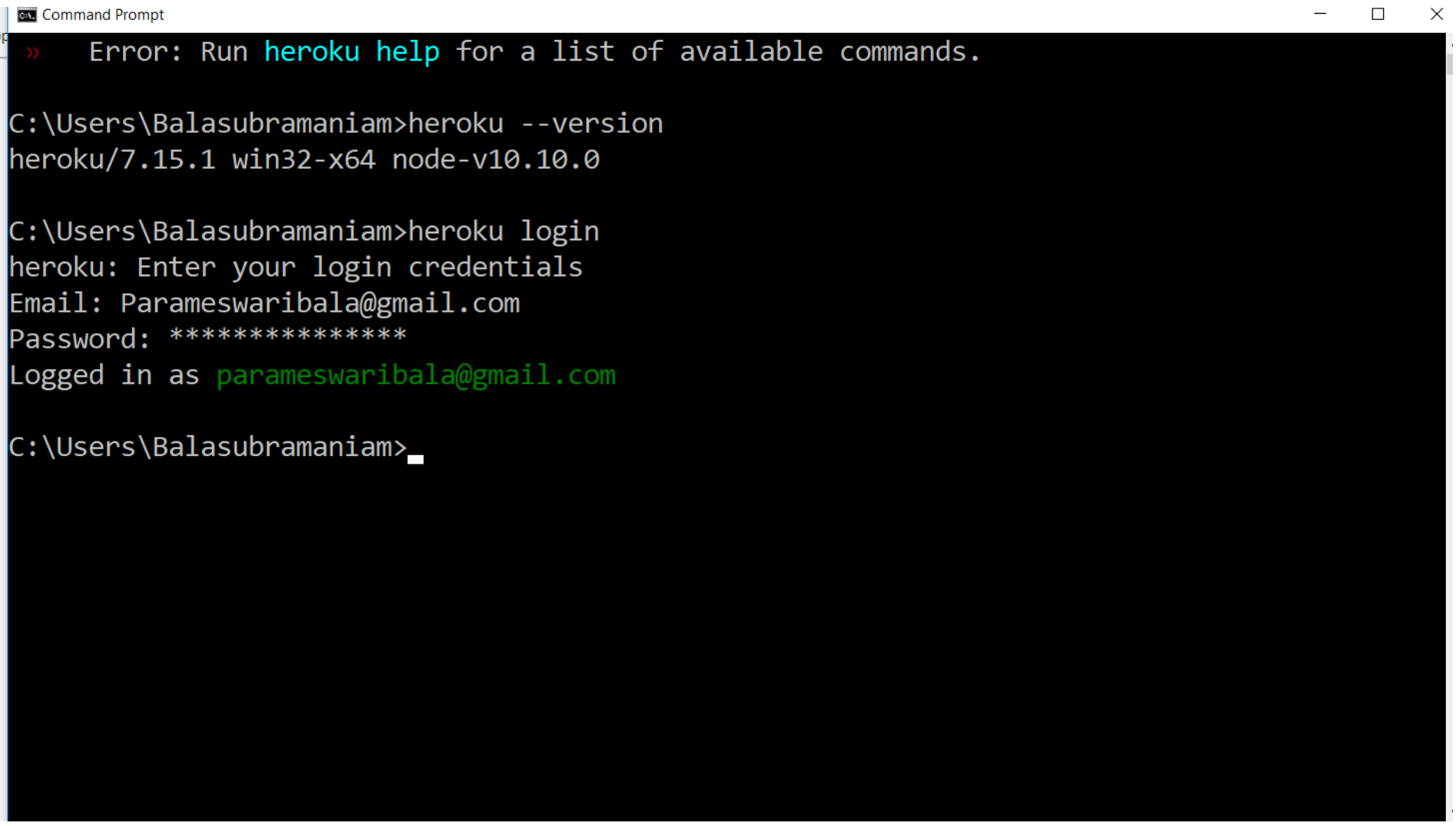

MSN India | Breaking News, Ente
Google
Pivot
Airtel 4G Hotspot
delete - Cloud Foundry CLI Refer
https://console.run.pivotal.io/organizations/207a000c-9017-41ec-8e05-d896a8...
Paused
Apps
Insert title here
Empire
New Tab
How to use Assertion
Browser Automation
node.js - How can I fi
Freelancer-dev-8104
Courses
Pivotal Web Services
Parameswaribala@gmail.com
ORG
AssetManagement
SPACES
development
Digit_AssetManagement
opentext_ecommspace
Marketplace
Docs
Support
Tools
Blog
Status
Command Prompt
API endpoint: https://api.run.pivotal.io (API version: 2.121.0)
User: Parameswaribala@gmail.com
Org: AssetManagement
Space: development
C:\Users\Balasubramaniam>cf delete AssetManagement-cf
Really delete the app AssetManagement-cf?> yes
Deleting app AssetManagement-cf in org AssetManagement / space development as Parameswaribala@gmail.com...
OK
C:\Users\Balasubramaniam>cf delete Digit_AssetManagement-cf
Really delete the app Digit_AssetManagement-cf?> yes
Deleting app Digit_AssetManagement-cf in org AssetManagement / space development as Parameswaribala@gmail.com...
OK
C:\Users\Balasubramaniam>
Type here to search
ENG
INTL
18:59
17-09-2018

Command Prompt - cf push opentext-cf -p EcommerceDemo-0.0.1-SNAPSHOT.jar

```
   2018-09-17T19:36:17.45+0530 [CELL/0] OUT Cell fafaa66c-635e-45b1-af3f-ee96cedf6965 destroying container for instanc
e 74eef495-c752-4843-5703-a8d8
   2018-09-17T19:36:17.61+0530 [PROXY/0] OUT Exit status 137
   2018-09-17T19:36:17.91+0530 [CELL/0] OUT Cell fafaa66c-635e-45b1-af3f-ee96cedf6965 successfully destroyed container
 for instance 74eef495-c752-4843-5703-a8d8

C:\Users\Balasubramaniam\Downloads\test>cf delete opentext-cf

Really delete the app opentext-cf?> yes
Deleting app opentext-cf in org AssetManagement / space opentext_ecommspace as Parameswaribala@gmail.com...
OK

C:\Users\Balasubramaniam\Downloads\test>cf push opentext-cf -p EcommerceDemo-0.0.1-SNAPSHOT.jar
```

Heroku

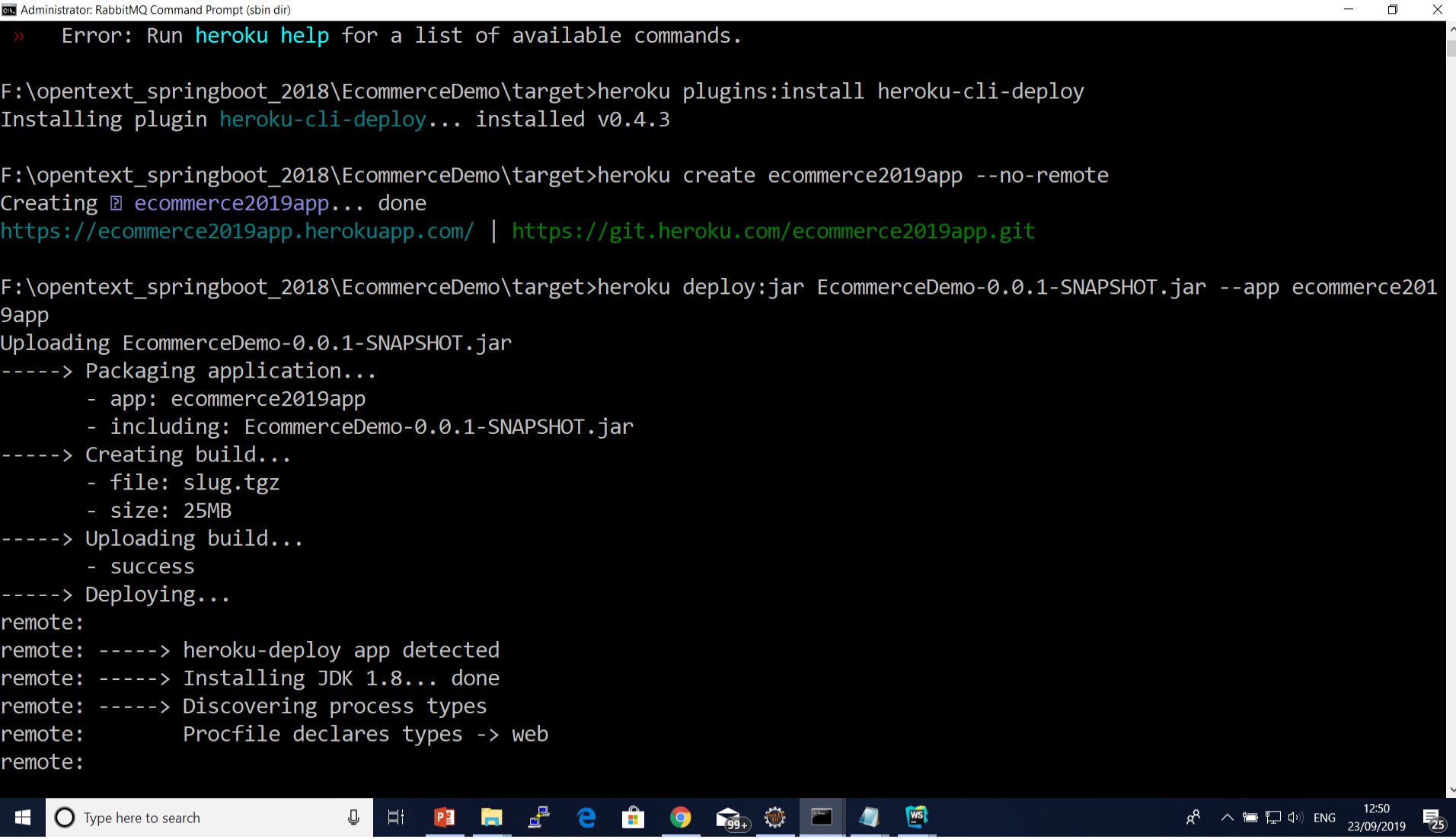

» Error: Run heroku help for a list of available commands.
C:\Users\Balasubramaniam>heroku --version
heroku/7.15.1 win32-x64 node-v10.10.0
C:\Users\Balasubramaniam>


Command Prompt
» Error: Run heroku help for a list of available commands.
C:\Users\Balasubramaniam>heroku --version
heroku/7.15.1 win32-x64 node-v10.10.0
C:\Users\Balasubramaniam>heroku login
heroku: Enter your login credentials
Email: Parameswaribala@gmail.com
Password: ***************
Logged in as parameswaribala@gmail.com
C:\Users\Balasubramaniam>

```
heroku: Enter your login credentials
Email: Parameswaribala@gmail.com
Password: ***************
Logged in as parameswaribala@gmail.com

C:\Users\Balasubramaniam>d:

D:\>md heroku-app

D:\>cd heroku-app

D:\heroku-app>heroku create
Creating app... \
```

```
heroku: Enter your login credentials
Email: Parameswaribala@gmail.com
Password: ***************
Logged in as parameswaribala@gmail.com

C:\Users\Balasubramaniam>d:

D:\>md heroku-app

D:\>cd heroku-app

D:\heroku-app>heroku create
Creating app... done, ⬀ immense-lowlands-97845
https://immense-lowlands-97845.herokuapp.com/ | https://git.heroku.com/immense-lowlands-97
845.git

D:\heroku-app>git init
Initialized empty Git repository in D:/heroku-app/.git/

D:\heroku-app>git add .

D:\heroku-app>
```

```
Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Balasubramaniam@DESKTOP-55AGI0I.(none)')

D:\heroku-app>git config --global user.email "Parameswaribala@gmail.com"

D:\heroku-app>git config --global user.name "Parameswaribala"

D:\heroku-app>git commit -am "make it better"
[master (root-commit) 7660c5e] make it better
 56 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 assetmgmt.jar
 create mode 100644 assetmgmt_lib/antlr-2.7.7.jar
 create mode 100644 assetmgmt_lib/aspectjweaver-1.8.13.jar
 create mode 100644 assetmgmt_lib/classmate-1.3.4.jar
```


Administrator: RabbitMQ Command Prompt (sbin dir)
» Error: Run heroku help for a list of available commands.
F:\opentext_springboot_2018\EcommerceDemo\target>heroku plugins:install heroku-cli-deploy
Installing plugin heroku-cli-deploy... installed v0.4.3
F:\opentext_springboot_2018\EcommerceDemo\target>heroku create ecommerce2019app --no-remote
Creating ecommerce2019app... done
https://ecommerce2019app.herokuapp.com/ | https://git.heroku.com/ecommerce2019app.git
F:\opentext_springboot_2018\EcommerceDemo\target>heroku deploy:jar EcommerceDemo-0.0.1-SNAPSHOT.jar --app ecommerce201
9app
Uploading EcommerceDemo-0.0.1-SNAPSHOT.jar
-----> Packaging application...
- app: ecommerce2019app
- including: EcommerceDemo-0.0.1-SNAPSHOT.jar
-----> Creating build...
- file: slug.tgz
- size: 25MB
-----> Uploading build...
- success
-----> Deploying...
remote:
remote: -----> heroku-deploy app detected
remote: -----> Installing JDK 1.8... done
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote:
Type here to search
ENG
12:50
23/09/2019

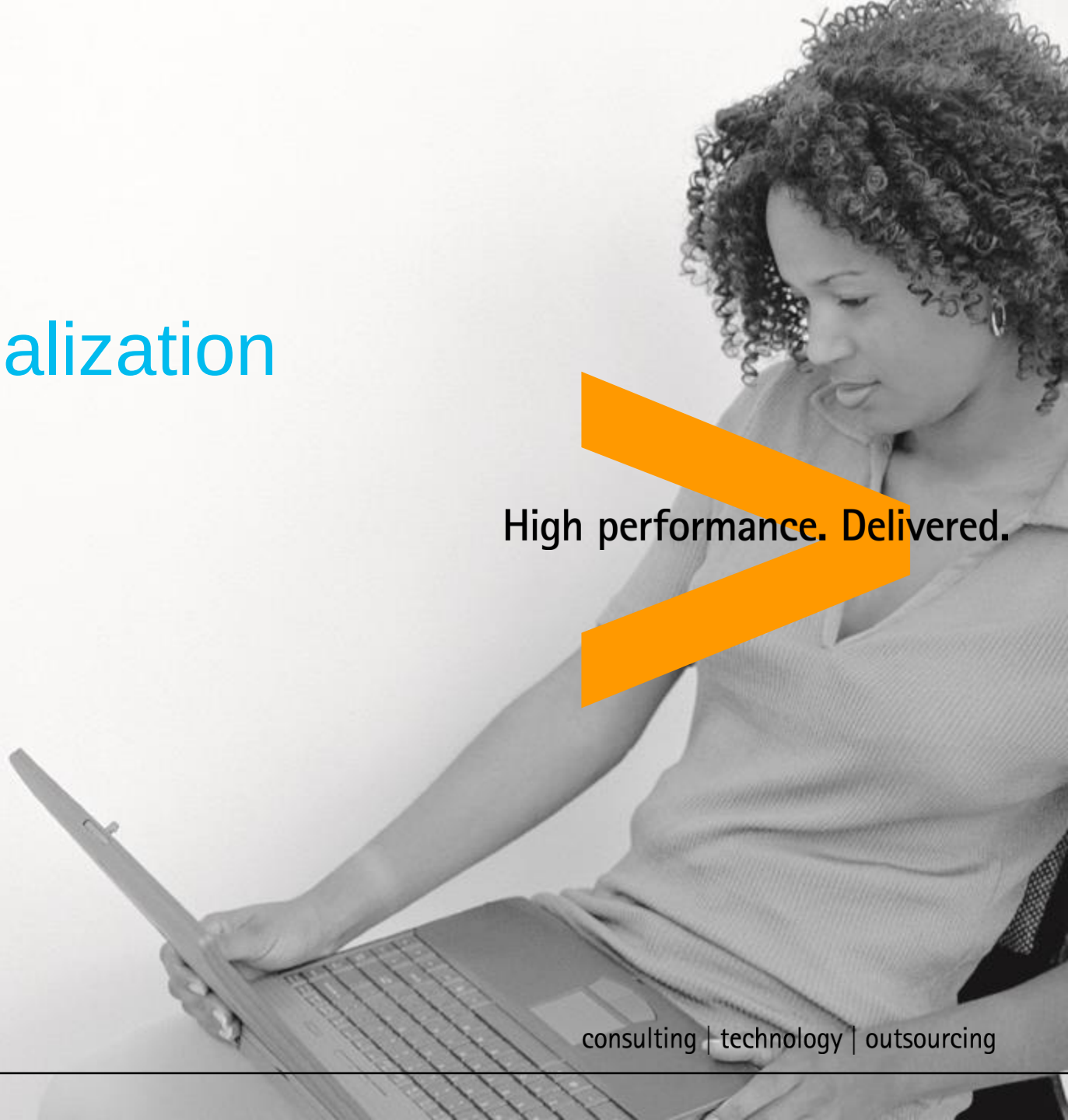
Using the Heroku Deploy CLI plugin

- heroku addons:create heroku-postgresql:hobby-dev
- heroku plugins:install heroku-cli-deploy
- heroku create --no-remote
- heroku deploy:jar target/my-app.jar --app sushi

Logging Configuration in Spring Boot

- **Date and Time**: Millisecond precision and easily sort able.
- **Log Level:** ERROR, WARN, INFO, DEBUG or TRACE.
- **Process ID.**
- A **—** Separator to distinguish the start of actual log messages.
- **Thread name:** Enclosed in square brackets (may be truncated for console output).
- **Logger name:** This is usually the source class name (often abbreviated).

Internationalization



Currently Available AutoConfiguration Behaviour

- **Data grid: Spring Data Gemfire, Solr and Elasticsearch**
- ! **Websocket**
- ! **Web services**
- ! **Mobile & Social (Facebook, Twitter and LinkedIn)**
- ! **Reactor for events and async processing**
- ! **Jersey**
- ! **JTA**
- ! **Email, CRaSH, AOP (AspectJ)**
- ! **Actuator features (Security, Audit, Metrics, Trace)**

The Actuator

- Actuator brings production-ready features to our application.
- **Monitoring our app, gathering metrics, understanding traffic or the state of our database becomes trivial with this dependency.**
- Actuator is mainly used to **expose operational information about the running application** – health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.
- Once this dependency is on the classpath several endpoints are available for us out of the box. As with most Spring modules, we can easily configure or extend it in many ways.

The Actuator

Adds common non-functional features to your application and exposes endpoints to interact with them (REST, JMX)

- Secure endpoints: /env, /metrics, /trace, /dump, /shutdown, /beans, /

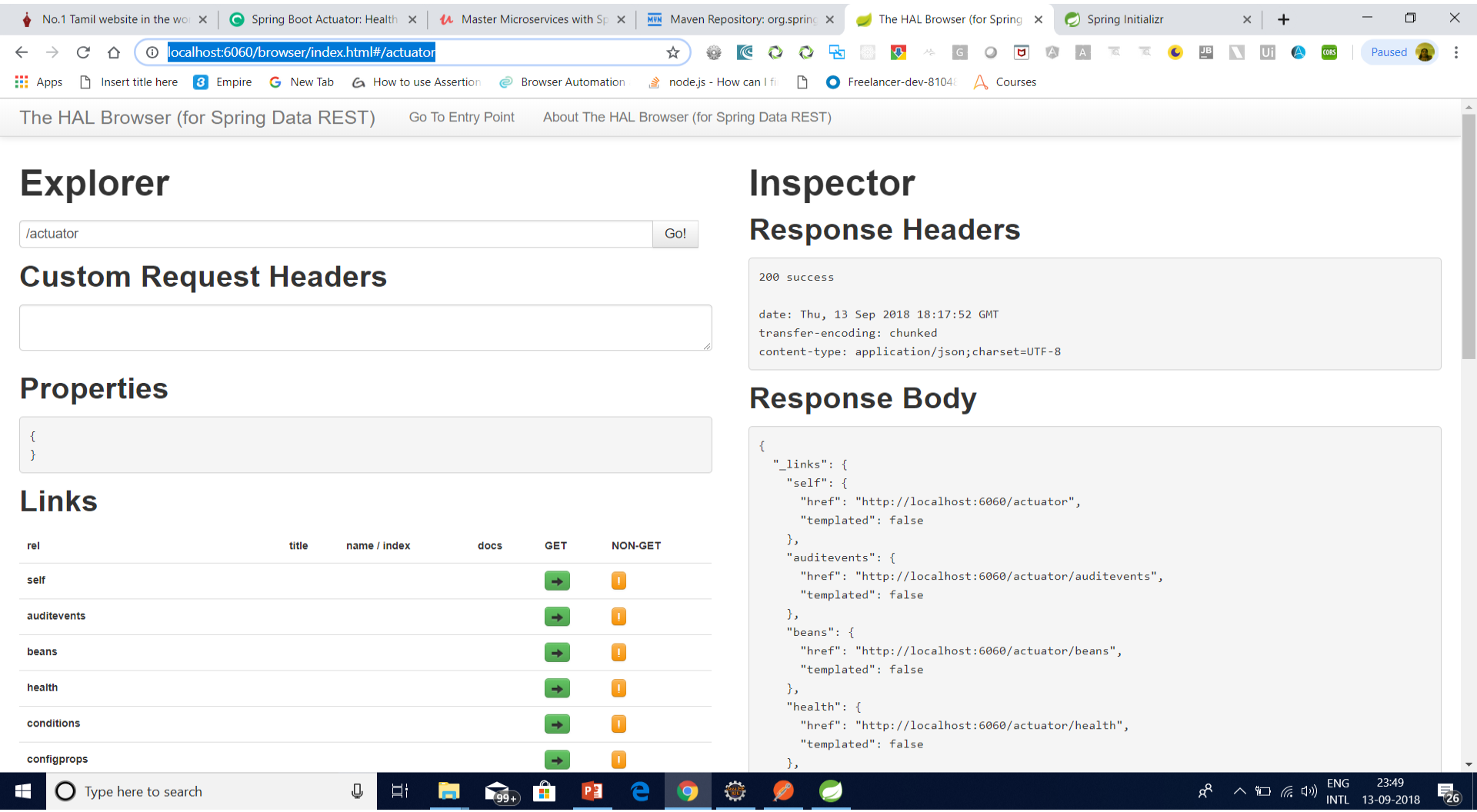
  autoconfig, /configprops, /mappings
- /info
- /health
- Audit

If embedded in a web app or web service can use the same port or a different one

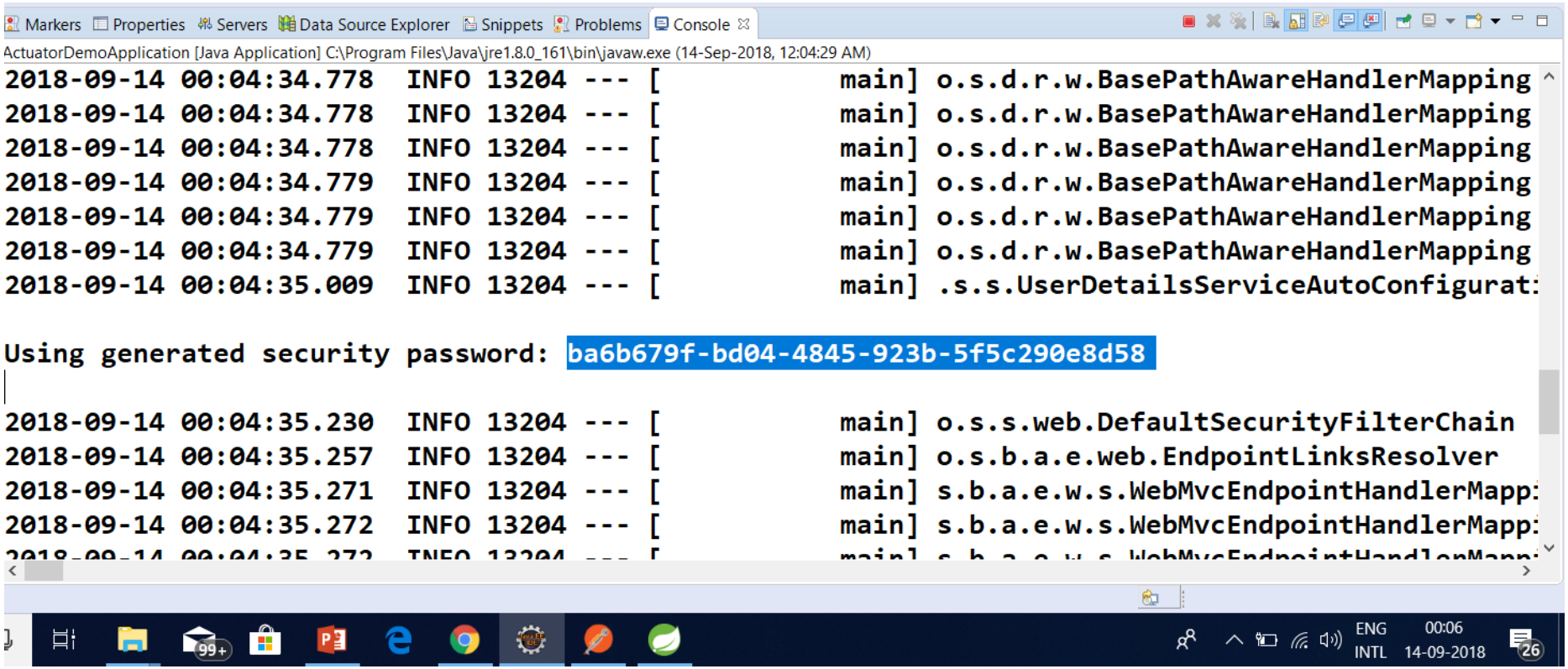
(management.port) and/or a different network interface (management.address)

and/or context path (management.context-path).

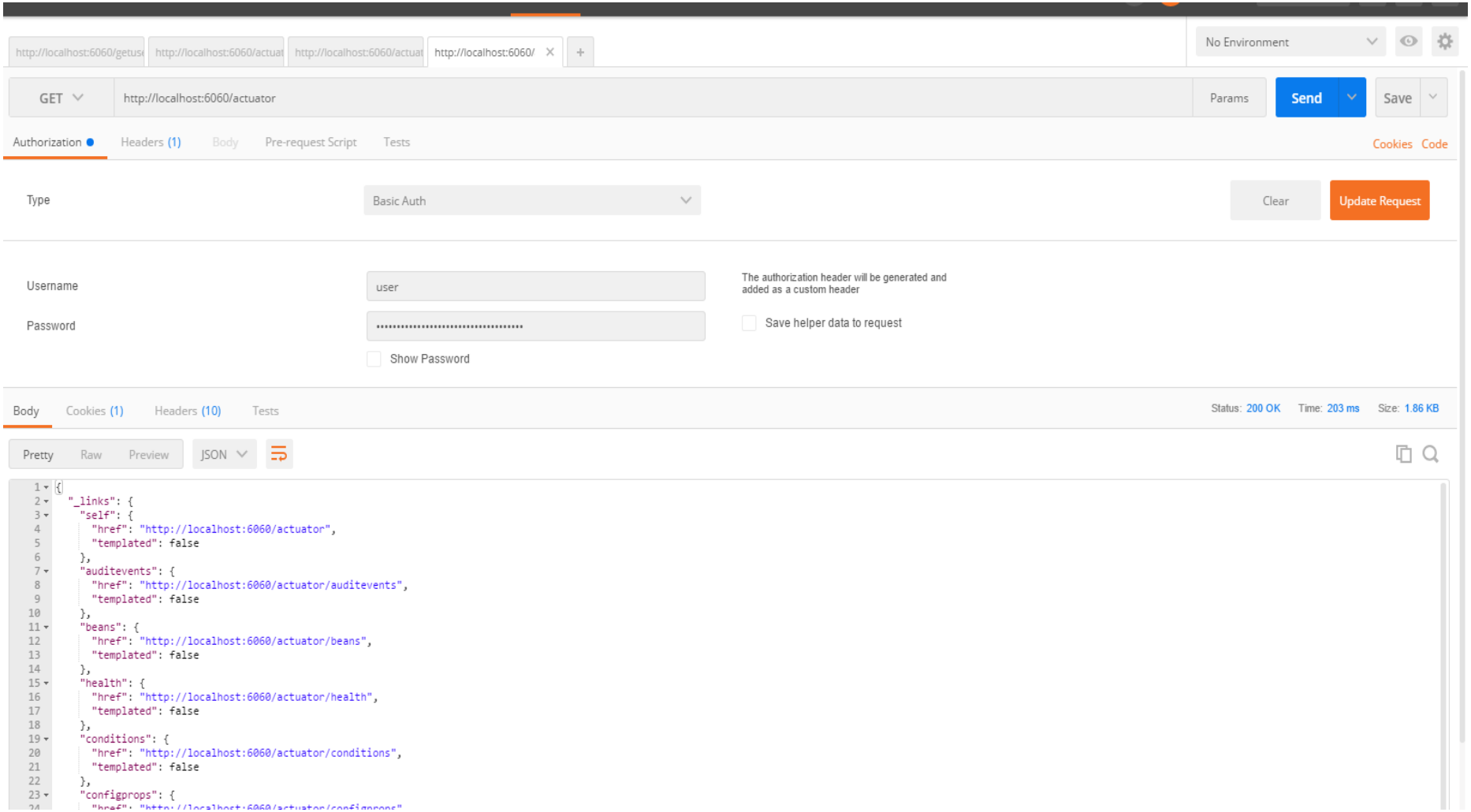
The Actuator – spring –data-hal-browser http://localhost:6060/browser/index.html



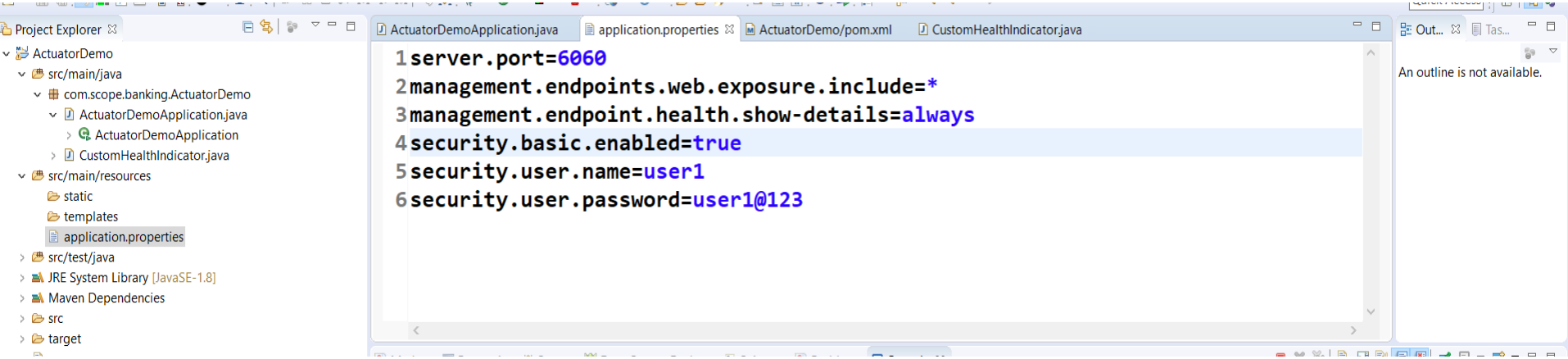
Spring Boot Security



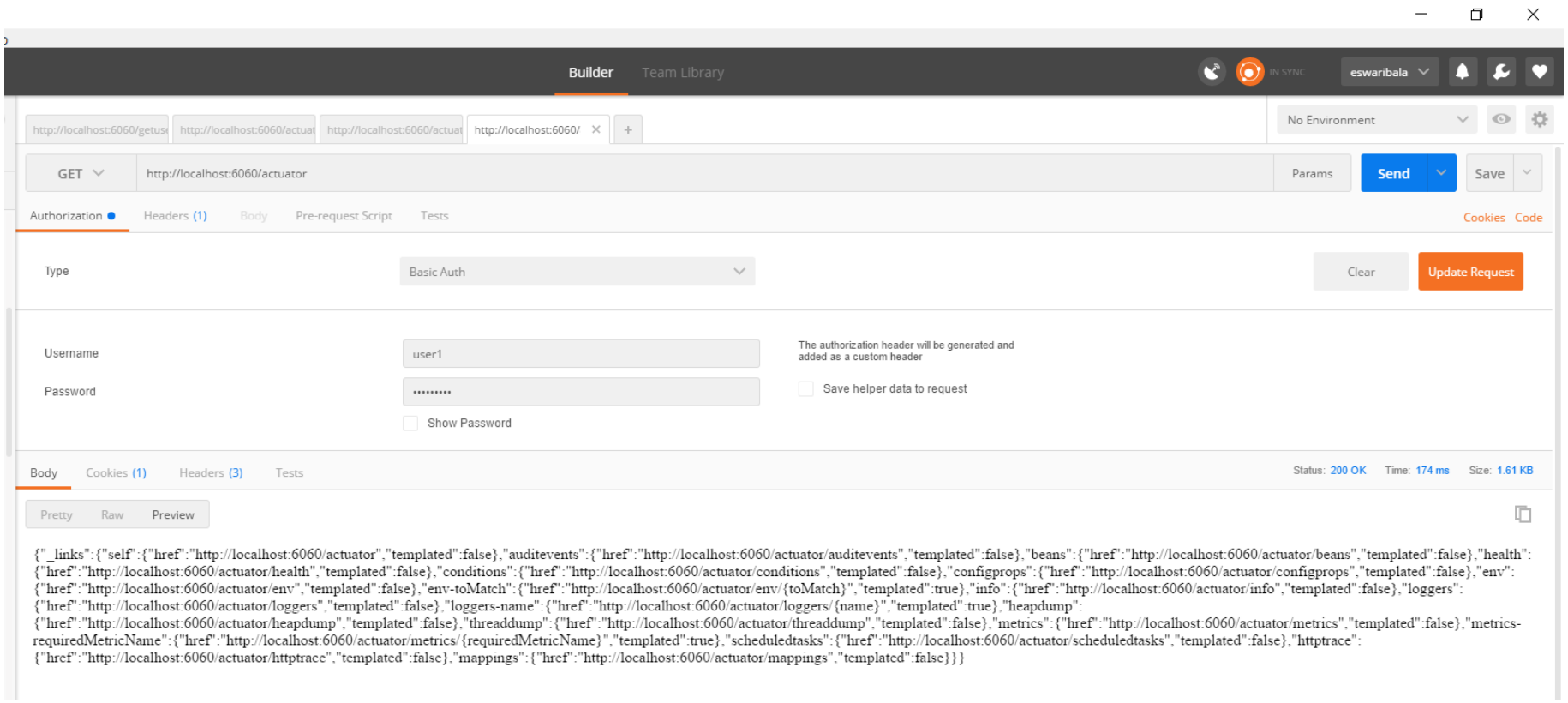
Spring Boot Security

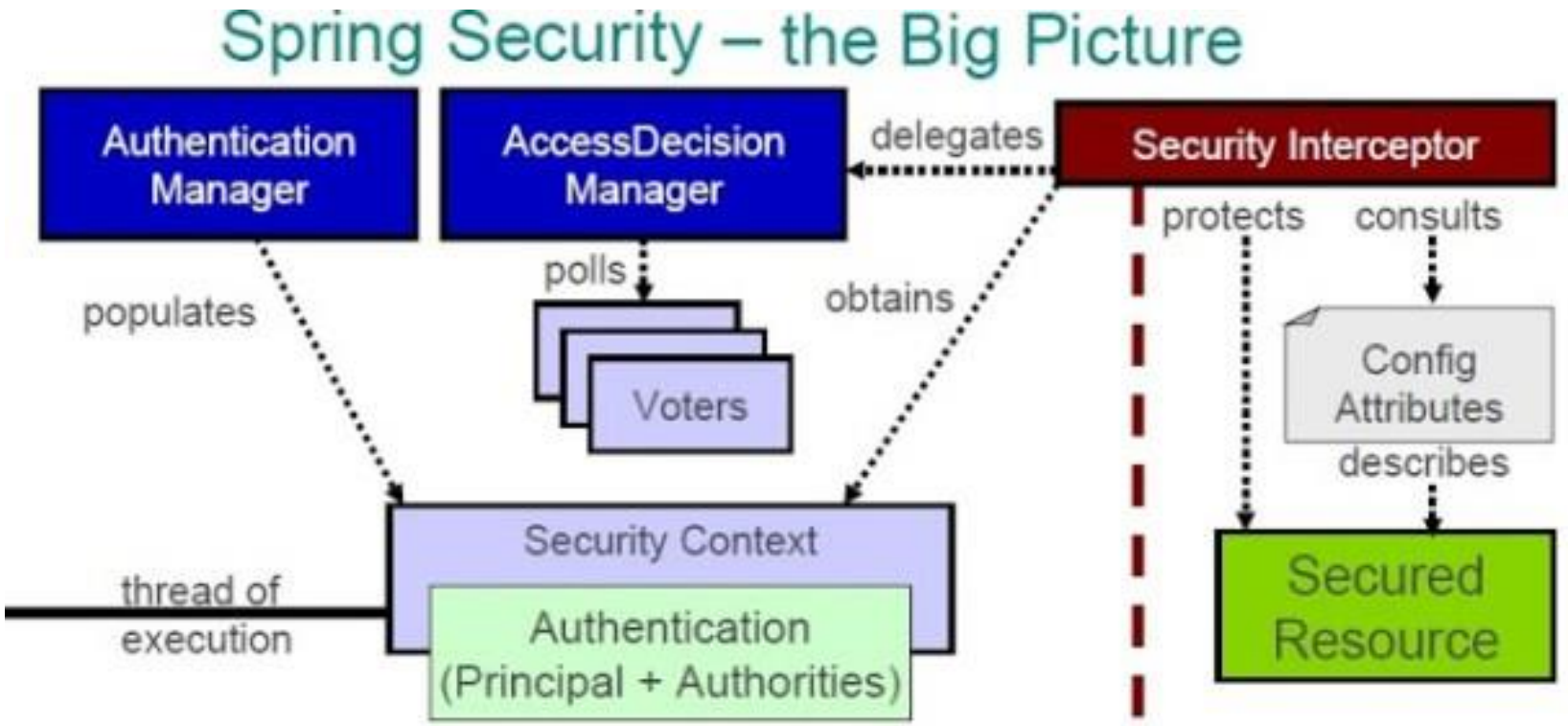


Spring Boot Security



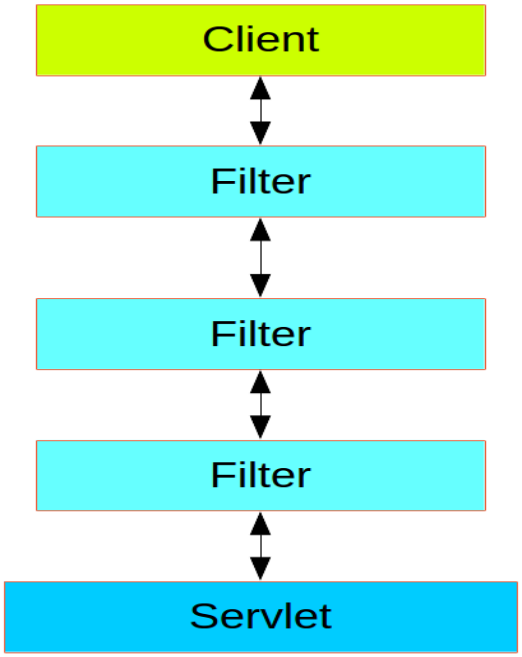
Spring Boot Security

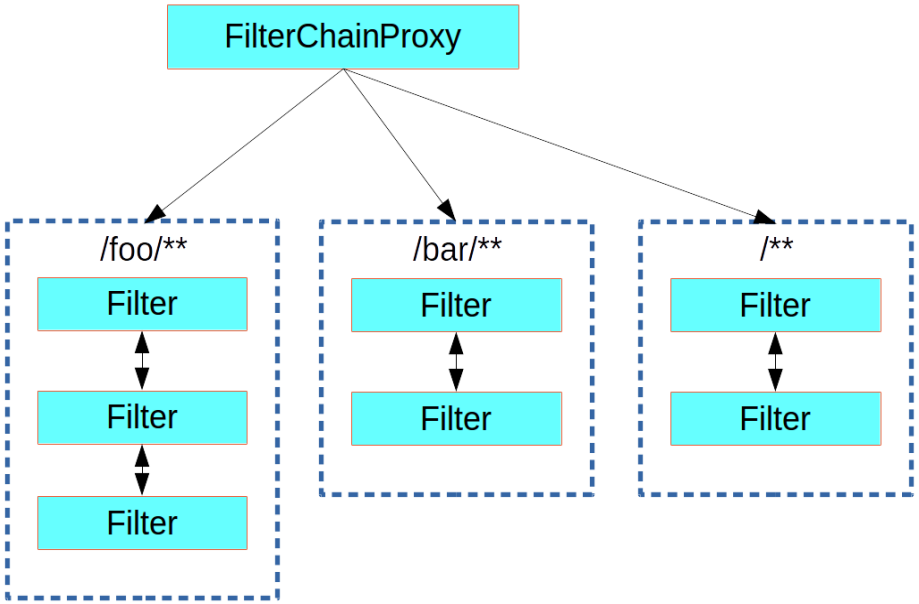


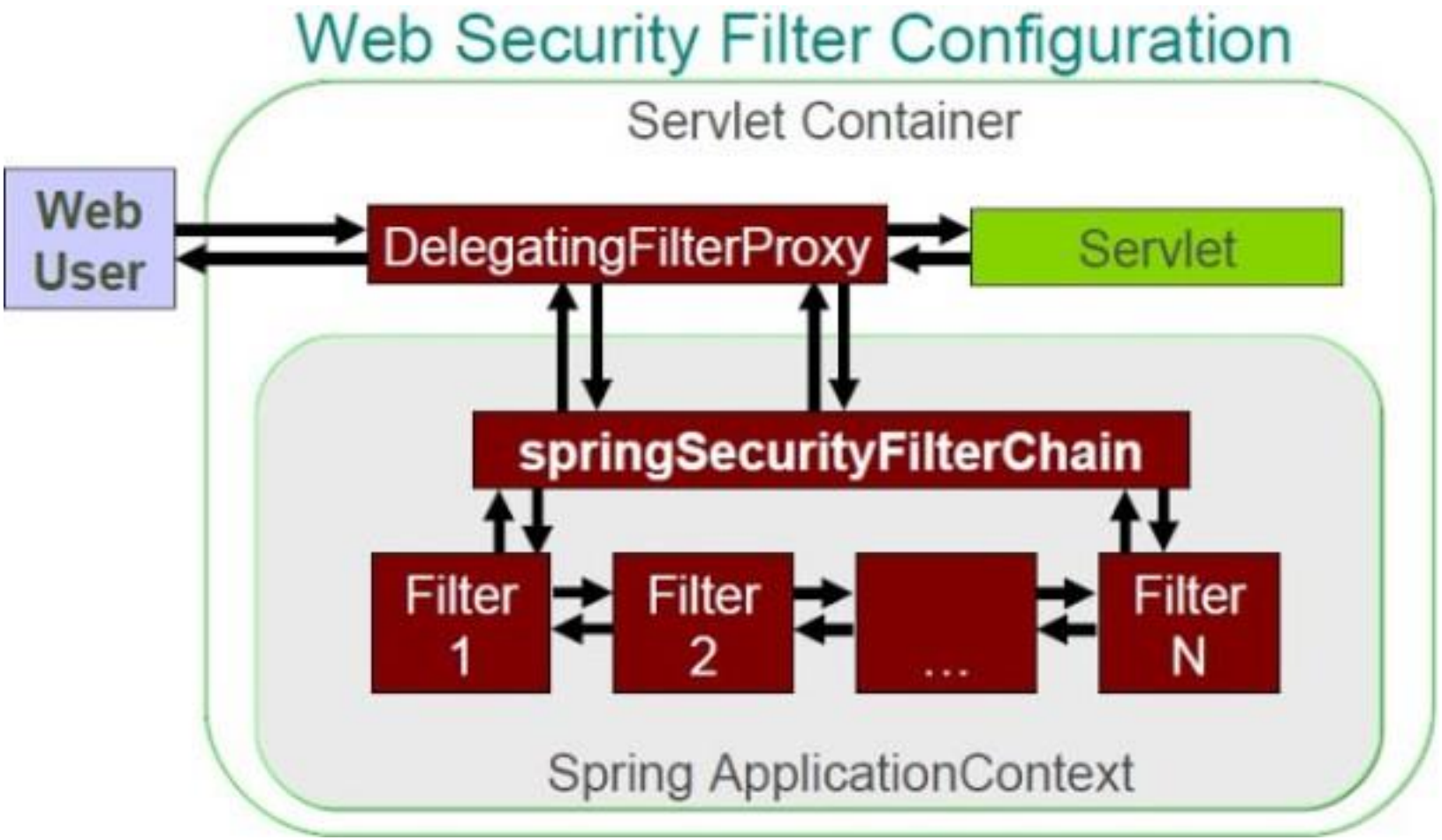

Spring Security – the Big Picture
Authentication Manager
AccessDecision Manager
delegates
Security Interceptor
protects
consults
polls
obtains
populates
Voters
Config Attributes
describes
Security Context
thread of execution
Authentication (Principal + Authorities)
Secured Resource

EnableGlobalMethodSecurity vs Enable Web Security

- EnableGlobalMethodSecurity provides AOP security on methods, some of annotation it will enable are PreAuthorize PostAuthorize


Client
Filter
Filter
Filter
Servlet

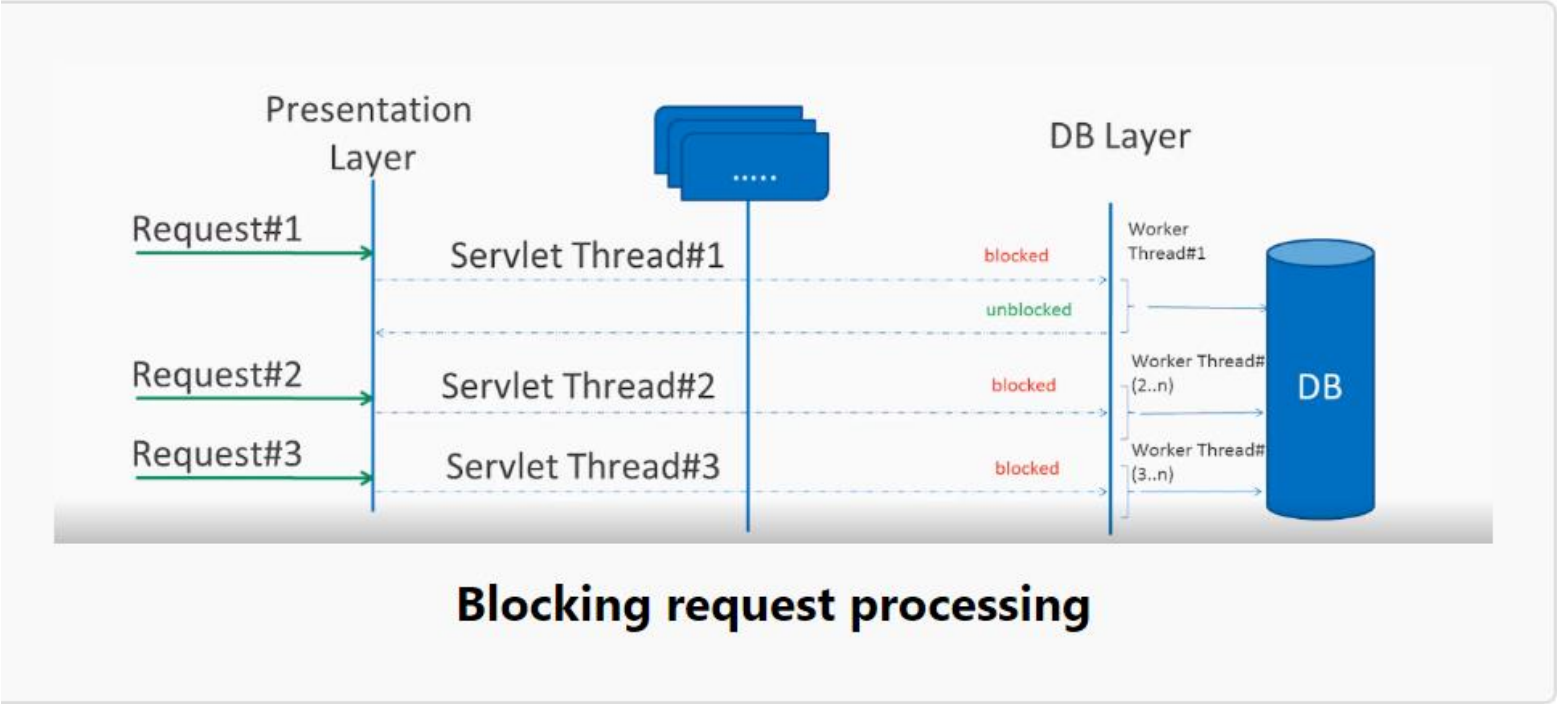

FilterChainProxy
/foo/**
Filter
Filter
Filter
/bar/**
Filter
Filter
/**
Filter
Filter


Web Security Filter Configuration
Servlet Container
Web User
DelegatingFilterProxy
Servlet
springSecurityFilterChain
Filter 1
Filter 2
...
Filter N
Spring ApplicationContext

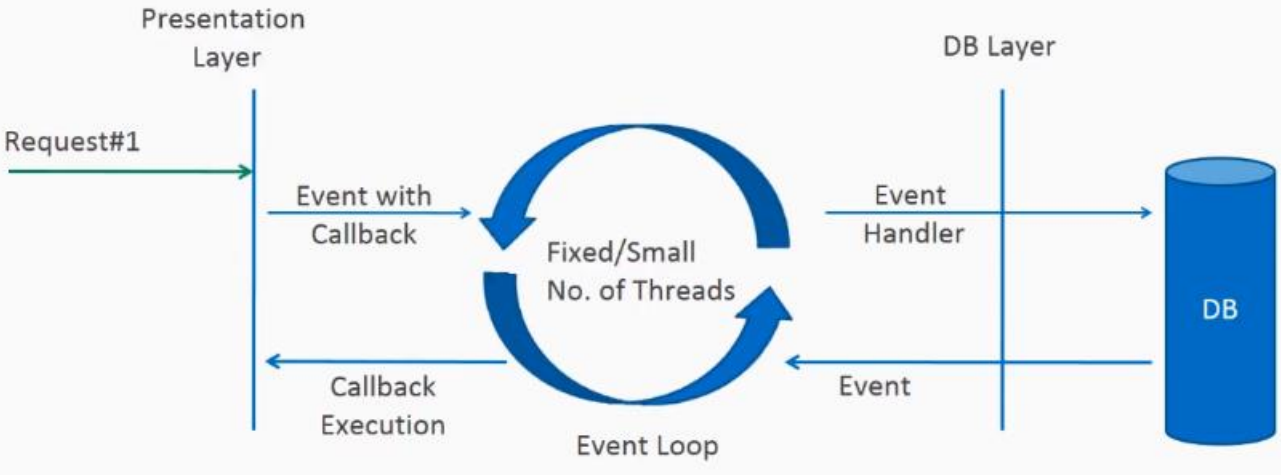
Reactive java programming

- Reactive programming is a programming paradigm that promotes an asynchronous, non-blocking, event-driven approach to data processing.
- Reactive programming involves modeling data and events as observable data streams and implementing data processing routines to react to the changes in those streams.

Blocking request processing



Non-blocking request processing



Non-blocking request processing

What is reactive programming?

- The term, “reactive,” refers to programming models that are built around reacting to changes.
- It is build around publisher-subscriber pattern (observer pattern).
- In reactive style of programming, we make a request for resource and start performing other things.
- When the data is available, we get the notification along with data inform of call back function.
- In callback function, we handle the response as per application/user needs.

What is reactive programming?

- One important thing to remember is back pressure.
- In non-blocking code, it becomes important to **control the rate of events** so that a fast producer does not overwhelm its destination.
- Reactive web programming is great for applications that have streaming data, and clients that consume it and stream it to their users.
- It is not great for developing traditional CRUD applications.
- If you're developing the next *Facebook* or *Twitter* with lots of data, a reactive API might be just what you're looking for.

Reactive Streams API

- The new Reactive Streams API was created by engineers from Netflix, Pivotal, Lightbend, RedHat, Twitter, and Oracle, among others and is now part of Java 9.
- It defines four interfaces:
- Publisher: Emits a sequence of events to subscribers according to the demand received from its subscribers. A publisher can serve multiple subscribers.
- It has a single method:

- Publisher.java
- public interface Publisher<T>
- {
- public void subscribe(Subscriber<? super T> s);
- }

Reactive Streams API

- Subscriber: Receives and processes events emitted by a Publisher. Please note that no notifications will be received until Subscription#request(long) is called to signal the demand.
- It has four methods to handle various kind of responses received.

- Subscriber.java
- public interface Subscriber<T>
- {
- public void onSubscribe(Subscription s);
- public void onNext(T t);
- public void onError(Throwable t);
- public void onComplete();
- }

Reactive Streams API

- Subscription: Defines a one-to-one relationship between a Publisher and a Subscriber. It can only be used once by a single Subscriber. It is used to both signal desire for data and cancel demand (and allow resource cleanup).
- Subscription.java
- public interface Subscription<T>
- {
- public void request(long n);
- public void cancel();
- }

Reactive Streams API

- Processor: Represents a processing stage consisting of both a Subscriber and a Publisher and obeys the contracts of both.
- Processor.java
- public interface Processor<T, R> extends Subscriber<T>, Publisher<R>
- {
- }

Reactive Streams API

- Two popular implementations of reactive streams are **RxJava** and **Project Reactor**

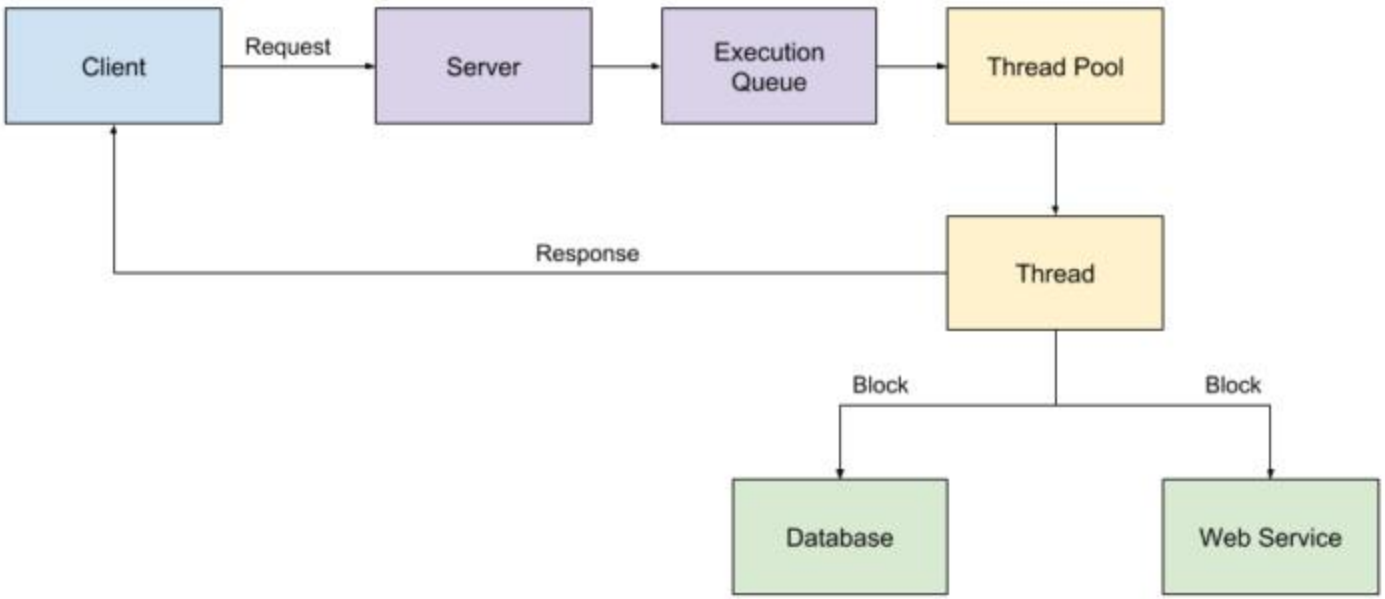
What is Spring WebFlux

- Spring WebFlux is parallel version of Spring MVC and supports fully non-blocking reactive streams.
- It support the back pressure concept and uses Netty as inbuilt server to run reactive applications.
- If you are familiar with Spring MVC programming style, you can easily work on webflux also.
- Spring webflux uses project reactor as reactive library. Reactor is a Reactive Streams library and, therefore, all of its operators support non-blocking back pressure. It is developed in close collaboration with Spring

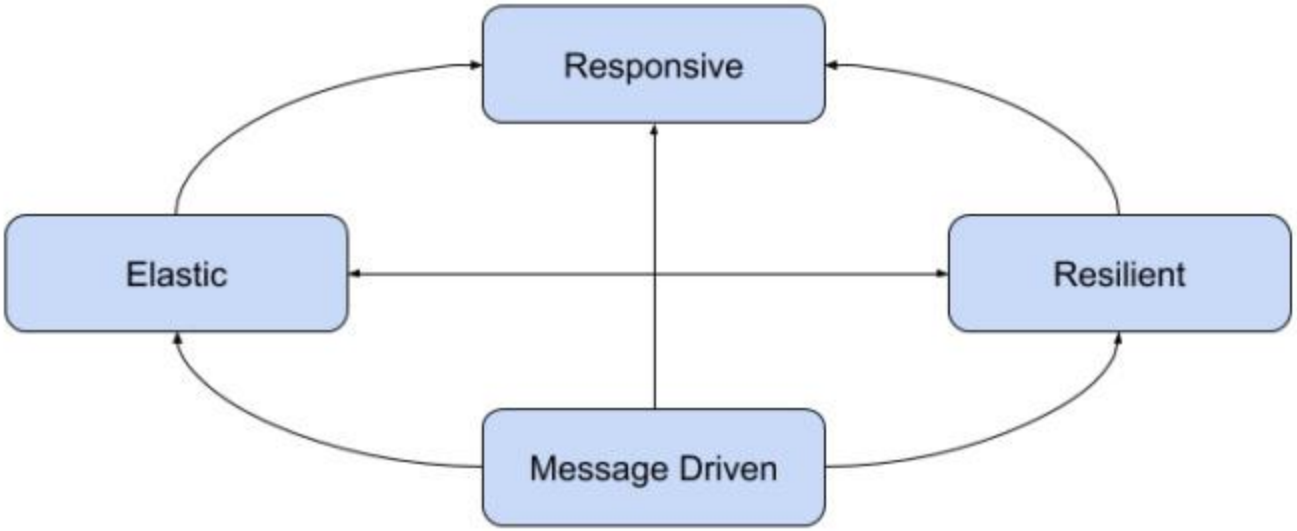
Spring MVC vs WebFlux

Traditional Stack	Reactive Stack
Spring Web MVC	Spring **WebFlux**
Controller and Handler Mapping	Router Functions
Servlet API	HTTP/ Reactive Stream
Servlet Containers	Any servlet container with support for Servlet 3.1+, Tomcat 8.x,Jetty, Netty, UnderTow

Traditional



WebFlux



WebFlux

- Reactive systems are **responsive**, meaning that they respond in a timely manner, in all possible circumstances. They focus on providing rapid and consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service.

WebFlux

- Reactive systems are **resilient**, meaning that they remain responsive in the face of failure. Resilience is achieved by the techniques of replication, containment, isolation, and delegation. By isolating application components from each other, you can contain failures and protect the system as a whole.

WebFlux

- Reactive systems are **elastic**, meaning that they stay responsive under varying workloads. This is achieved by scaling application components elastically to meet the current demand.

WebFlux

- Reactive systems are **message-driven**, meaning that they rely on asynchronous message passing between components. This allows you to create loose coupling, isolation, and location transparency.

Characteristics of a reactive system

- Reactive systems are built by creating isolated components that communicate with one another asynchronously and can scale quickly to meet the current load. Components still fail in reactive systems, but there are defined actions to perform as a result of that failure, which keeps the system as a whole functional and responsive.

Characteristics of a reactive system

- **Data streams**: A *stream* is a sequence of events ordered in time, such as user interactions, REST service calls, JMS messages, and results from a database.
- **Asynchronous**: Data stream events are captured asynchronously and your code defines what to do when an event is emitted, when an error occurs, and when the stream of events has completed.

Characteristics of a reactive system

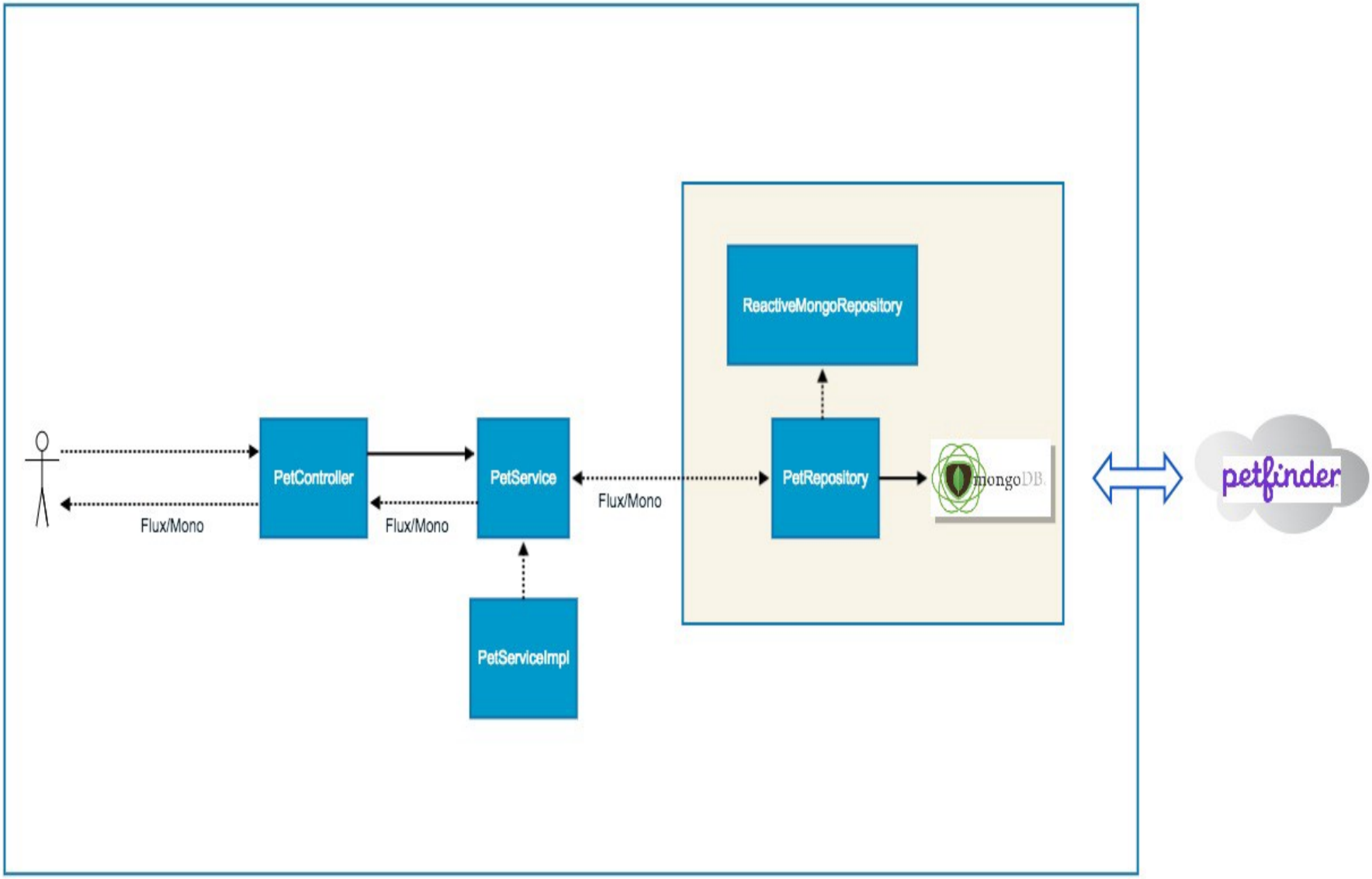
- **Non-blocking**: As you process events, your code should not block and perform synchronous calls; instead, it should make asynchronous calls and respond as the results of those calls are returned.
- **Back pressure**: Components control the number of events and how often they are emitted. In reactive terms, your component is referred to as the *subscriber* and events are emitted by a *publisher*. This is important because the subscriber is in control of how much data it receives and thus will not overburden itself.

Characteristics of a reactive system

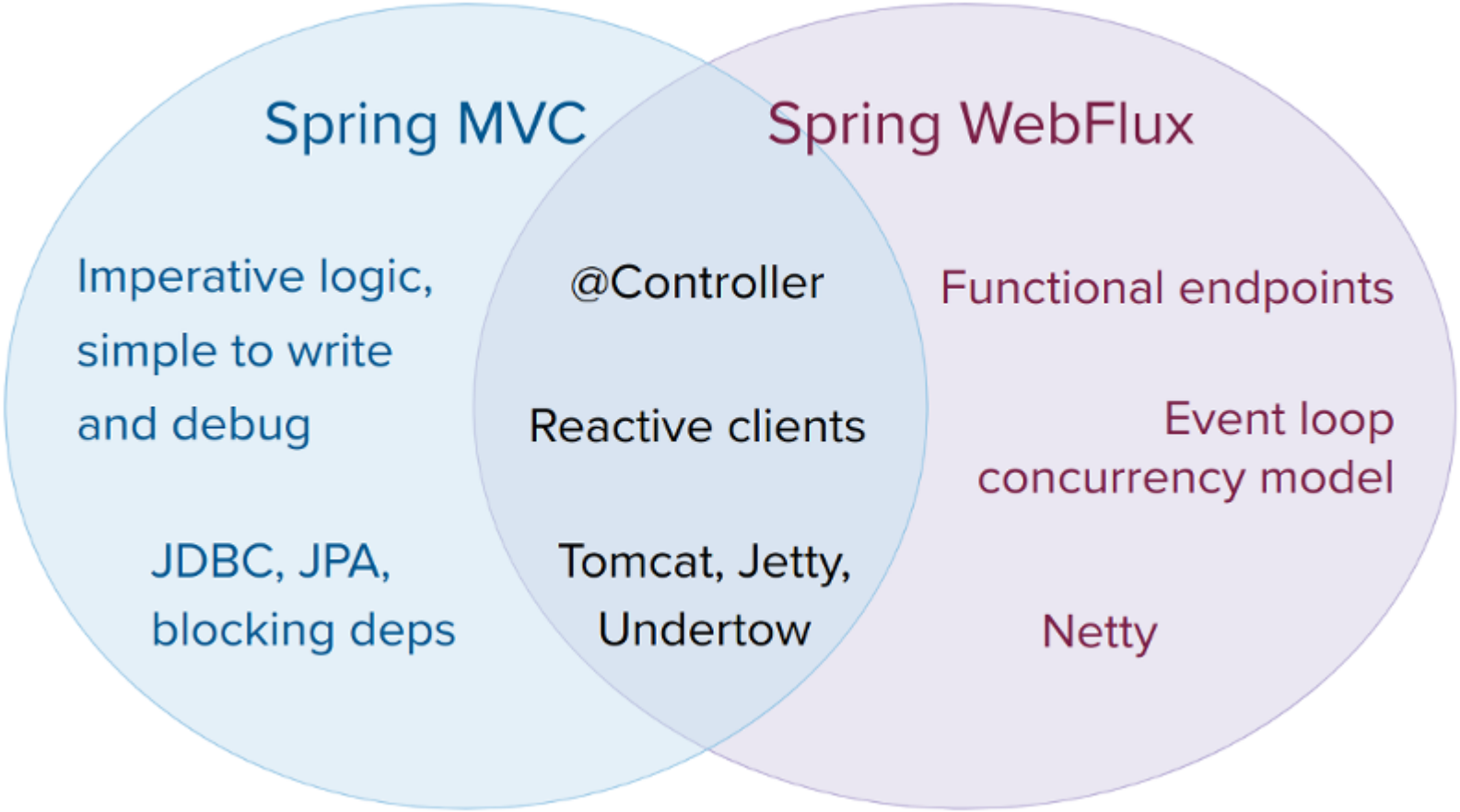
- **Failure messages**: Instead of components throwing exceptions, failures are sent as messages to a handler function. Whereas throwing exceptions breaks the stream, defining a function to handle failures as they occur does not.

The Reactive Streams API

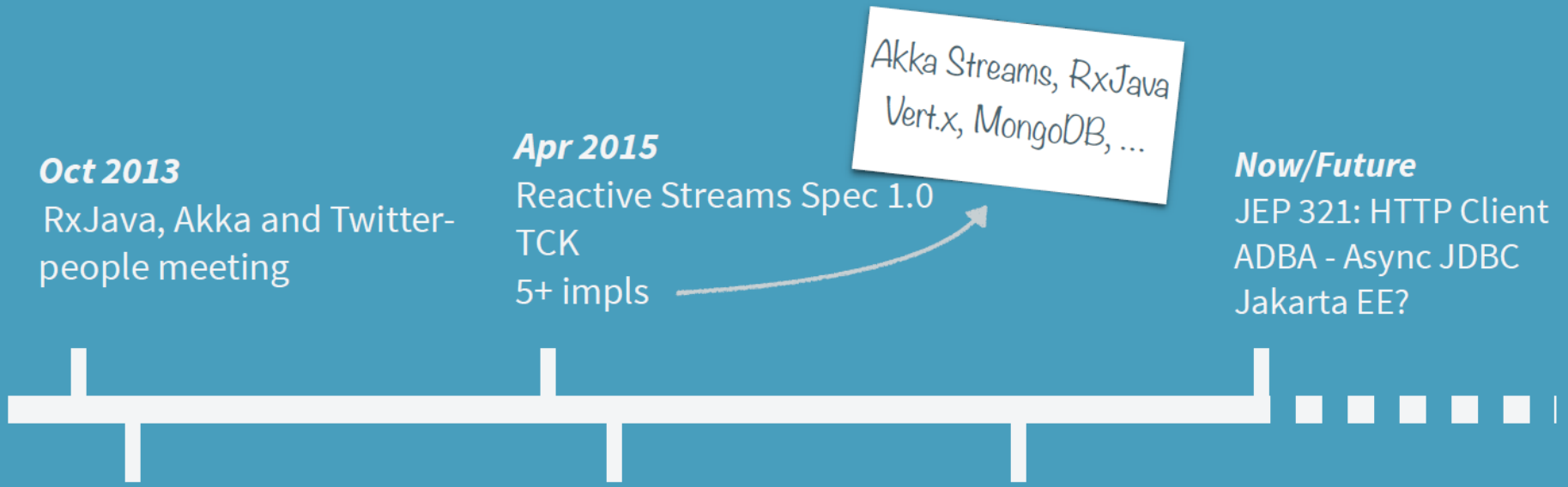
- The new Reactive Streams API was created by engineers from Netflix, Pivotal, Lightbend, RedHat, Twitter, and Oracle, among others. Published in 2015, the Reactive Streams API is now part of Java 9. It defines four interfaces:
- **Publisher**: Emits a sequence of events to subscribers.
- **Subscriber**: Receives and processes events emitted by a Publisher.
- **Subscription**: Defines a one-to-one relationship between a Publisher and a Subscriber.
- **Processor**: Represents a processing stage consisting of both a Subscriber and a Publisher and obeys the contracts of both.


ReactiveMongoRepository
PetController
PetService
PetRepository
mongoDB.
Flux/Mono
Flux/Mono
Flux/Mono
PetServiceImpl
petfinder

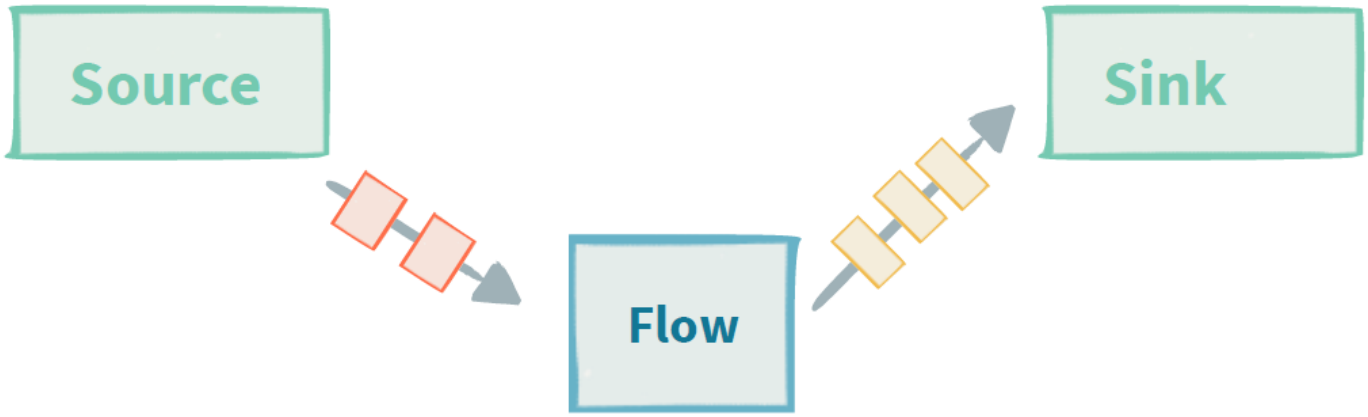
Spring MVC vs WebFlux



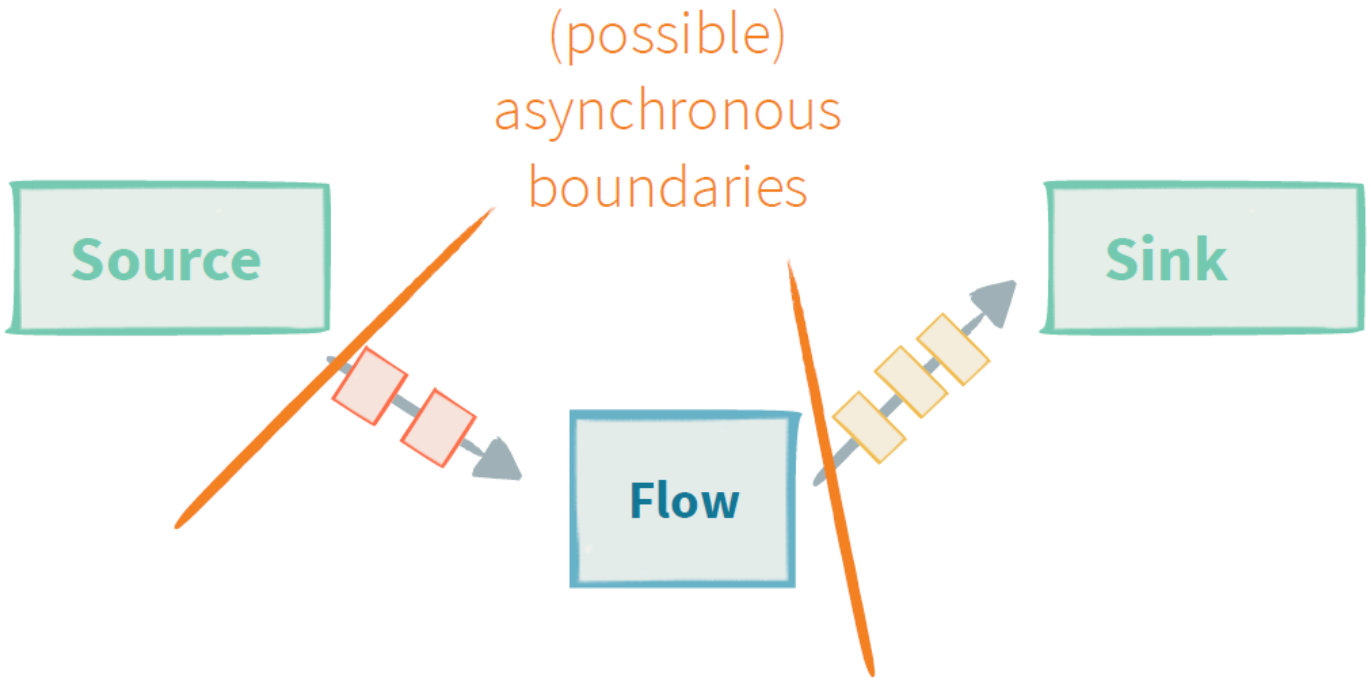
Reactive Streams timeline



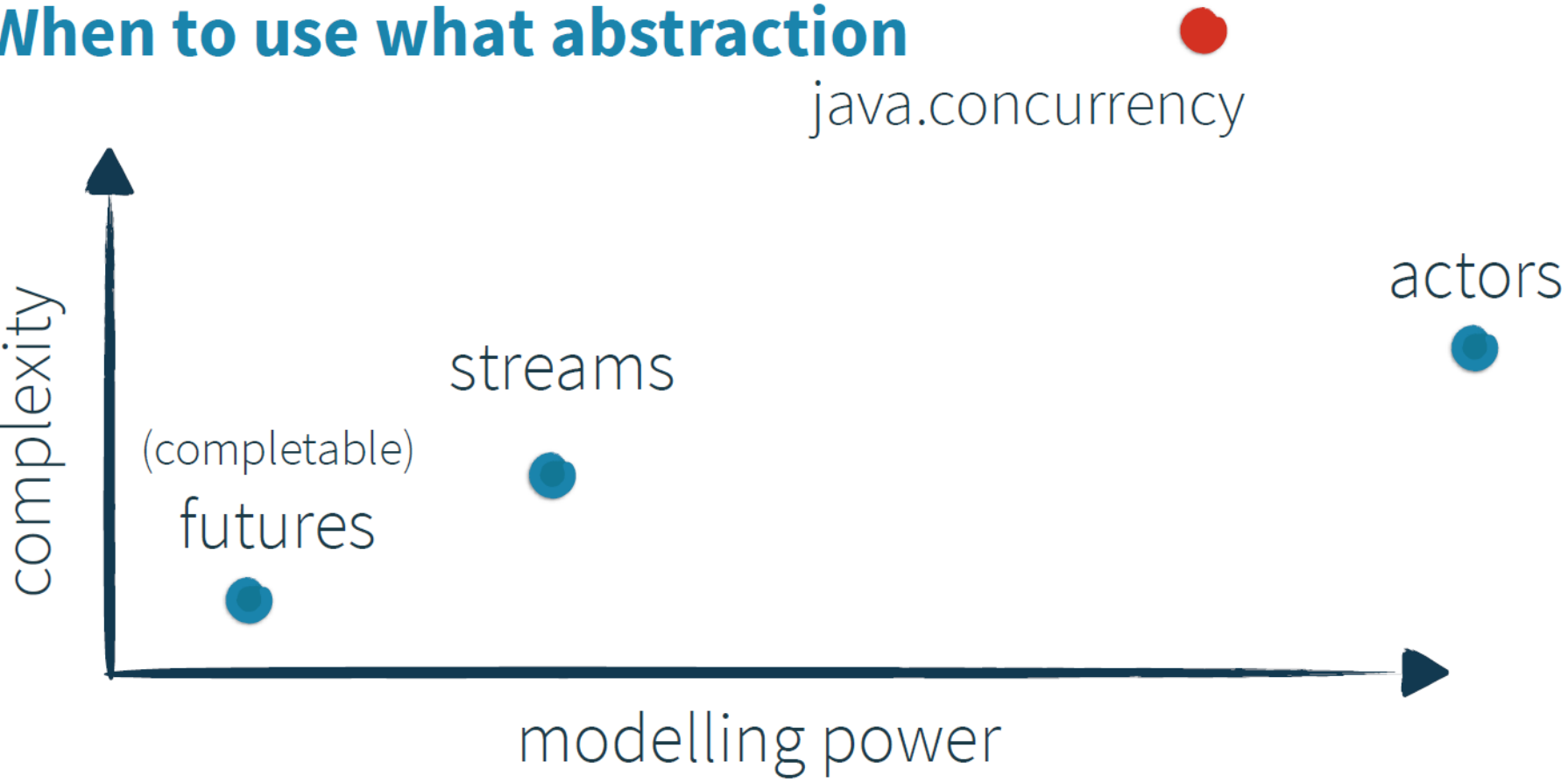
Stream processing



Asynchronous stream processing



When to use what abstraction



Why Streams?

- In software development, there can be cases where we need to handle the potentially large amount of data.
- So while handling these kinds of scenarios there can be issues such as `out of memory` exceptions so we should divide the data in chunks and handle the chunks independently.

What is the Akka Framework?

- Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant applications on the JVM.
- Akka is written in Scala, with language bindings provided for both Scala and Java.
- Akka's approach to handling concurrency is based on the Actor Model.
- In an actor-based system, everything is an actor, in much the same way that everything is an object in object-oriented design.
- A key difference is that the Actor Model was specifically designed and architected to serve as a concurrent model whereas the object-oriented model is not.
- More specifically, in a Scala actor system, actors interact and share information, without any presupposition of sequentiality.
- The mechanism by which actors share information with one another, and task one another, is message passing.

What is the Akka Framework?

- Akka creates a layer between the actors and the underlying system such that actors simply need to process messages.
- All the complexity of creating and scheduling threads, receiving and dispatching messages, and handling race conditions and synchronization, is relegated to the framework to handle transparently.

What is the Akka Framework?

- Akka strictly adheres to the Reactive Manifesto.
- Reactive applications aim at replacing traditional multithreaded applications with an architecture that satisfies one or more of the following requirements:
- Event-driven. Using Actors, one can write code that handles requests asynchronously and employs non-blocking operations exclusively.
- Scalable. In Akka, adding nodes without having to modify the code is possible, thanks both to message passing and location transparency.

What is the Akka Framework?

- ***Resilient.*** Any application will encounter errors and fail at some point in time. Akka provides “supervision” (fault tolerance) strategies to facilitate a self-healing system.
- ***Responsive.*** Many of today’s high performance and rapid response applications need to give quick feedback to the user and therefore need to react to events in an extremely timely manner. Akka’s non-blocking, message-based strategy helps achieve this.

What is an Actor in Akka?

- An actor is essentially nothing more than an object that receives messages and takes actions to handle them.
- It is decoupled from the source of the message and its only responsibility is to properly recognize the type of message it has received and take action accordingly.

What is an Actor in Akka?

- Upon receiving a message, an actor may take one or more of the following actions:
- Execute some operations itself (such as performing calculations, persisting data, calling an external web service, and so on)
- Forward the message, or a derived message, to another actor
- Instantiate a new actor and forward the message to it.
- Alternatively, the actor may choose to ignore the message entirely (i.e., it may choose inaction) if it deems it appropriate to do so.

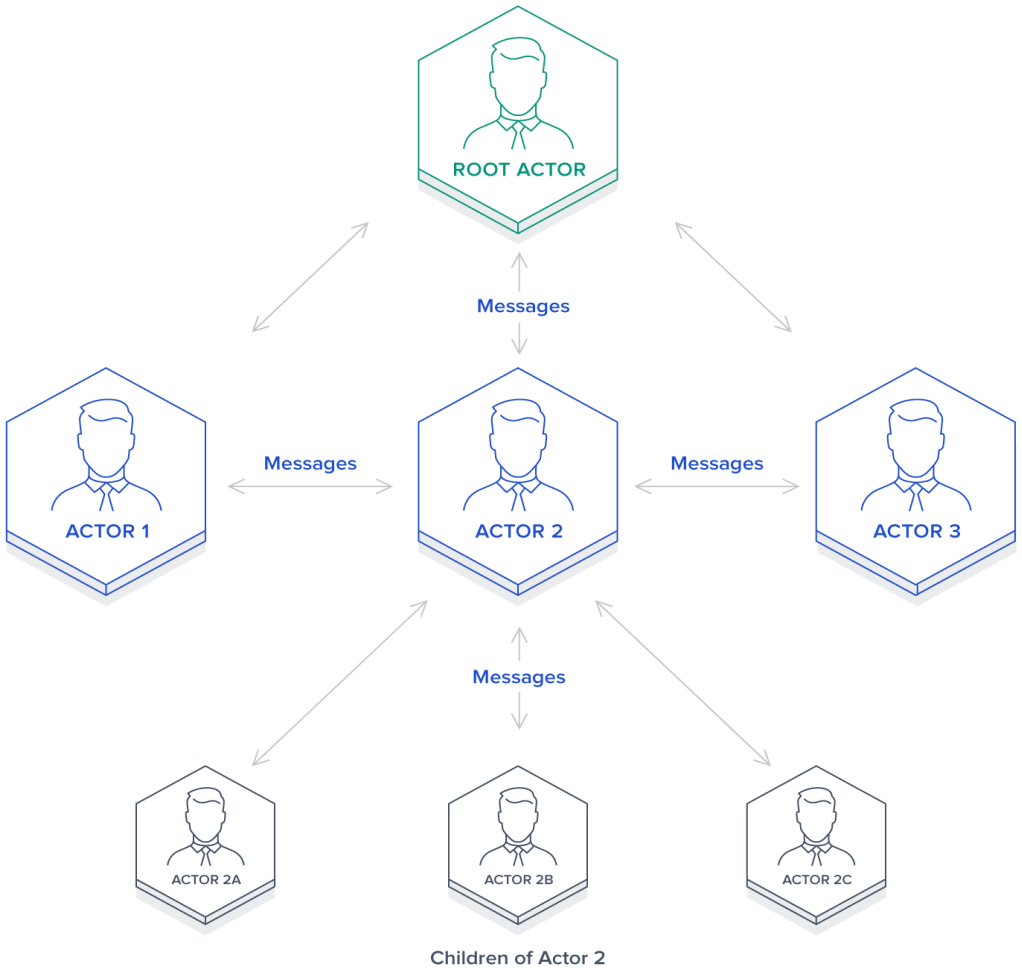
The Actor System

- Taking a complex problem and recursively splitting it into smaller sub-problems is a sound problem solving technique in general.
- This approach can be particularly beneficial in computer science (consistent with the Single Responsibility Principle), as it tends to yield clean, modularized code, with little or no redundancy, that is relatively easy to maintain.

The Actor System

- In an actor-based design, use of this technique facilitates the logical organization of actors into a hierarchical structure known as an Actor System.
- The actor system provides the infrastructure through which actors interact with one another.

The Actor System



The Actor System

- In Akka, the only way to communicate with an actor is through an ActorRef.
- An ActorRef represents a reference to an actor that precludes other objects from directly accessing or manipulating that actor's internals and state.
- Messages may be sent to an actor via an ActorRef using one of the following syntax protocols:
- ! ("tell") – sends the message and returns immediately
- ? ("ask") – sends the message and returns a Future representing a possible reply

The Actor System

- Each actor has a mailbox to which its incoming messages are delivered.
- There are multiple mailbox implementations from which to choose, with the default implementation being FIFO.
- An actor contains many instance variables to maintain state while processing multiple messages.
- Akka ensures that each instance of an actor runs in its own lightweight thread and that messages are processed one at a time.
- In this way, each actor's state can be reliably maintained without the developer needing to explicitly worry about synchronization or race conditions.

The Actor System

- Each actor is provided with the following useful information for performing its tasks via the Akka Actor API:
- sender: an ActorRef to the sender of the message currently being processed
- context: information and methods relating to the context within which the actor is running (includes, for example, an actorOf method for instantiating a new actor)
- supervisionStrategy: defines the strategy to be used for recovering from errors
- self: the ActorRef for the actor itself

Supervision

- supervision describes a dependency relationship between actors: the supervisor delegates tasks to subordinates and therefore must respond to their failures.
- When a subordinate detects a failure (i.e. throws an exception), it suspends itself and all its subordinates and sends a message to its supervisor, signaling failure.
- Depending on the nature of the work to be supervised and the nature of the failure, the supervisor has a choice of the following four options:

Supervision

- Resume the subordinate, keeping its accumulated internal state
- Restart the subordinate, clearing out its accumulated internal state
- Stop the subordinate permanently
- Escalate the failure, thereby failing itself

What Restarting Means

- When presented with an actor which failed while processing a certain message, causes for the failure fall into three categories:
- Systematic (i.e. programming) error for the specific message received
- (Transient) failure of some external resource used during processing the message
- Corrupt internal state of the actor

What Restarting Means

- suspend the actor (which means that it will not process normal messages until resumed), and recursively suspend all children
- call the old instance's preRestart hook (defaults to sending termination requests to all children and calling postStop)
- wait for all children which were requested to terminate (using context.stop()) during preRestart to actually terminate; this—like all actor operations—is non-blocking, the termination notice from the last killed child will effect the progression to the next step
- create new actor instance by invoking the originally provided factory again
- invoke postRestart on the new instance (which by default also calls preStart)
- send restart request to all children which were not killed in step 3; restarted children will follow the same process recursively, from step 2
- resume the actor

The Actor System

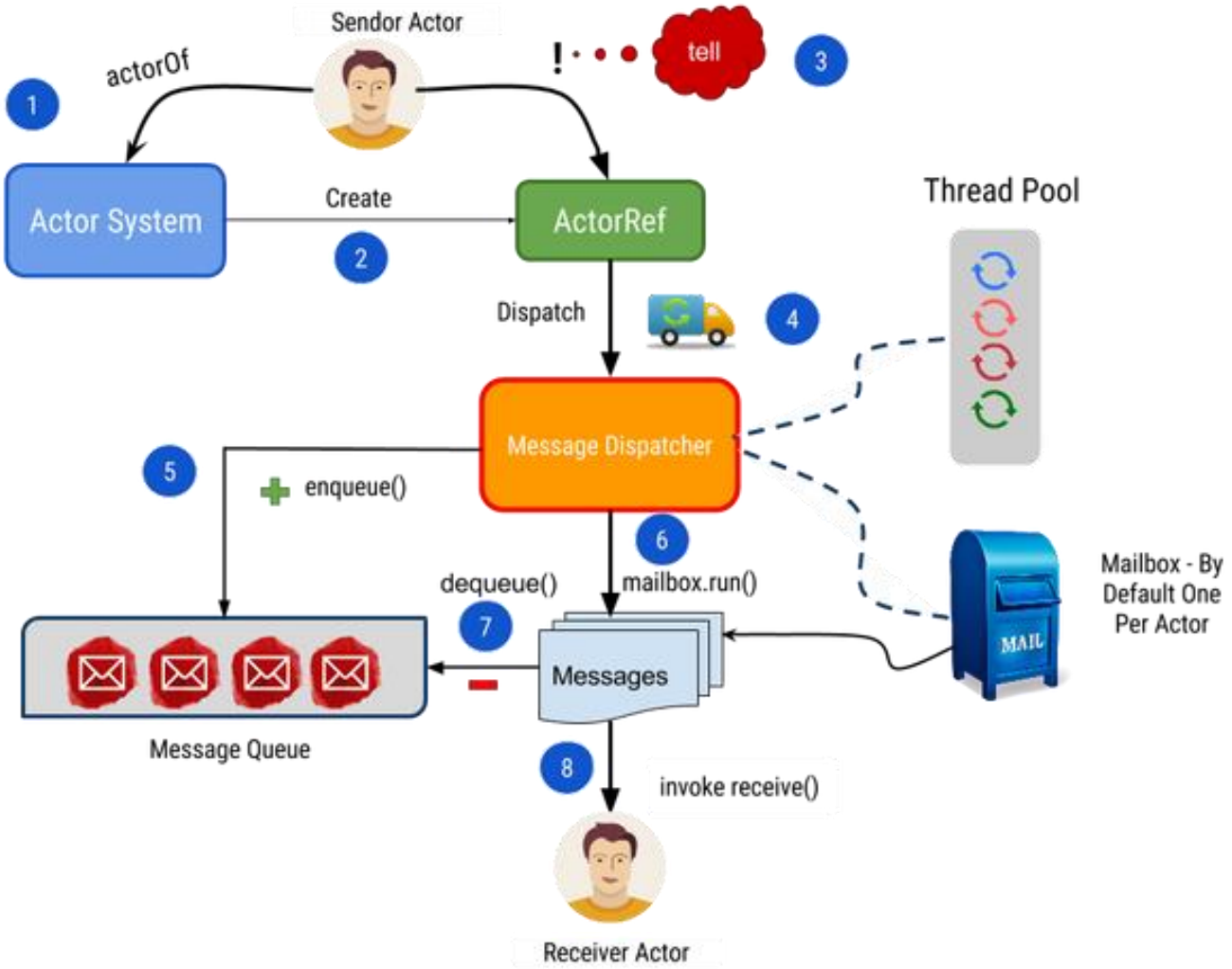
- Akka ensures that each instance of an actor runs in its own lightweight thread and that messages are processed one at a time.
- In this way, each actor's state can be reliably maintained without the developer needing to explicitly worry about synchronization or race conditions.

Actor System

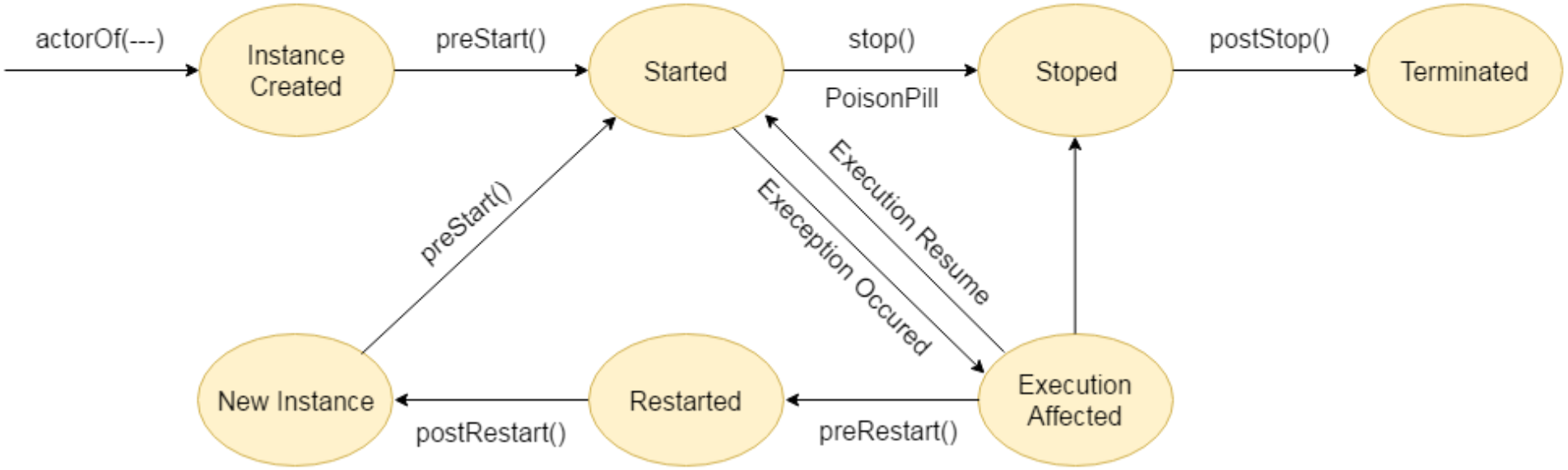
- Actors are objects which encapsulate state and behavior, they communicate exclusively by exchanging messages which are placed into the recipient's mailbox.
- In a sense, actors are the most stringent form of object-oriented programming, but it serves better to view them as persons:
- while modeling a solution with actors, envision a group of people and assign sub-tasks to them, arrange their functions into an organizational structure and think about how to escalate failure.
- The result can then serve as a mental scaffolding for building the software implementation.

Akka System

- An actor is a container for State, Behavior, a Mailbox, Child Actors and a Supervisor Strategy.
- All of this is encapsulated behind an Actor Reference.
- One noteworthy aspect is that actors have an explicit lifecycle, they are not automatically destroyed when no longer referenced;


Sendor Actor
actorOf
tell
1
3
Actor System
Create
ActorRef
2
Thread Pool
Dispatch
4
Message Dispatcher
5
enqueue()
6
dequeue()
mailbox.run()
7
Messages
Mailbox - By Default One Per Actor
MAIL
Message Queue
8
invoke receive()
Receiver Actor

Akka Actor Life Cycle



Akka System

- **State**
- Actor objects will typically contain some variables which reflect possible states the actor may be in.
- This can be an explicit state machine, or it could be a counter, set of listeners, pending requests, etc.
- These data are what make an actor valuable, and they must be protected from corruption by other actors.
- The good news is that Akka actors conceptually each have their own light-weight thread, which is completely shielded from the rest of the system.
- This means that instead of having to synchronize access using locks you can write your actor code without worrying about concurrency at all.

Akka System

- Behavior
- Every time a message is processed, it is matched against the current behavior of the actor.
- Behavior means a function which defines the actions to be taken in reaction to the message at that point in time, say forward a request if the client is authorized, deny it otherwise.
- This behavior may change over time, e.g. because different clients obtain authorization over time, or because the actor may go into an “out-of-service” mode and later come back.
- These changes are achieved by either encoding them in state variables which are read from the behavior logic, or the function itself may be swapped out at runtime, see the become and unbecome operations.

Akka System

- Mailbox
- An actor’s purpose is the processing of messages, and these messages were sent to the actor from other actors (or from outside the actor system).
- The piece which connects sender and receiver is the actor’s mailbox: each actor has exactly one mailbox to which all senders enqueue their messages.
- Enqueuing happens in the time-order of send operations, which means that messages sent from different actors may not have a defined order at runtime due to the apparent randomness of distributing actors across threads.
- Sending multiple messages to the same target from the same actor, on the other hand, will enqueue them in the same order.

Akka System

- Child Actors
- Each actor is potentially a supervisor: if it creates children for delegating sub-tasks, it will automatically supervise them.
- The list of children is maintained within the actor's context and the actor has access to it.
- Modifications to the list are done by creating (context.actorOf(...)) or stopping (context.stop(child)) children and these actions are reflected immediately.
- The actual creation and termination actions happen behind the scenes in an asynchronous way, so they do not "block" their supervisor.

Akka System

- Supervisor Strategy
- The final piece of an actor is its strategy for handling faults of its children.
- Fault handling is then done transparently by Akka, applying one of the strategies in Supervision and Monitoring for each incoming failure.
- As this strategy is fundamental to how an actor system is structured, it cannot be changed once an actor has been created.

Akka System

- When an Actor Terminates
- Once an actor terminates, i.e. fails in a way which is not handled by a restart, stops itself or is stopped by its supervisor, it will free up its resources, draining all remaining messages from its mailbox into the system's "dead letter mailbox" which will forward them to the EventStream as DeadLetters.
- The mailbox is then replaced within the actor reference with a system mailbox, redirecting all new messages to the EventStream as DeadLetters.
- This is done on a best effort basis, though, so do not rely on it in order to construct "guaranteed delivery".

Akka System

- Akka is a set of open-source libraries for designing scalable, resilient systems that span processor cores and networks.
- Akka allows you to focus on meeting business needs instead of writing low-level code to provide reliable behavior, fault tolerance, and high performance.
- Many common practices and accepted programming models do not address important challenges inherent in designing systems for modern computer architectures.

Akka System

- To be successful, distributed systems must cope in an environment where components crash without responding, messages get lost without a trace on the wire, and network latency fluctuates.

Akka System

- Akka provides:
- Multi-threaded behavior without the use of low-level concurrency constructs like atomics or locks — relieving you from even thinking about memory visibility issues.
- Transparent remote communication between systems and their components — relieving you from writing and maintaining difficult networking code.
- A clustered, high-availability architecture that is elastic, scales in or out, on demand — enabling you to deliver a truly reactive system.

Akka Streams

- Akka-Streams provide a higher-level abstraction over Akka's existing actor model.
- The Actor model provides an excellent primitive for writing concurrent, scalable software, but it still is a primitive;

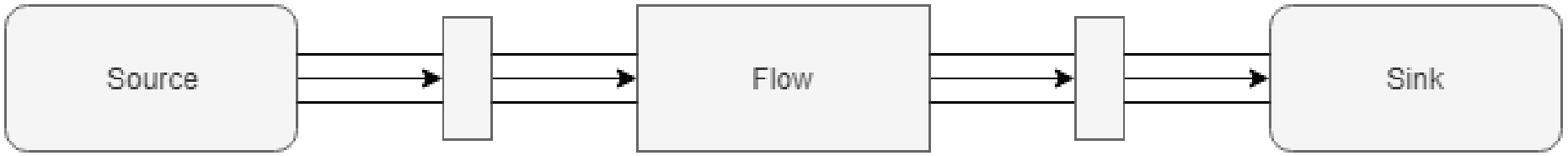
Akka Streams basics

- In Akka Streams, we represent data processing in a form of data flow through an arbitrary complex graph of processing stages.
- Stages have zero or more inputs and zero or more outputs.
- The basic building blocks are Sources (one output), Sinks (one input) and Flows (one input and one output).
- Using them, we can build arbitrary long linear pipelines.

Akka Streams basics

- Akka streams consist of 3 major components in it – Source, Flow, Sink – and any non-cyclical stream consist of at least 2 components Source, Sink and any number of Flow element.
- Source – this is the Source of data. It has exactly one output. We can think of Source as Publisher.
- Sink – this is the Receiver of data. It has exactly one input. We can think of Sink as Receiver.
- Flow – this is the Transformation that acts on the Source. It has exactly one input and one output.

Akka Streams basics



What are Reactive Streams?

- Applications developed using streams can run into problems if Source is generating data too fast than the Sink can handle.
- This causes Sink to buffer the data – but the problem is if data is too large then Sink buffer will also grow and can lead to memory issues.
- So to handle this Sink need to communicate with the Source – to slow down the generation of data until it finished handling of current data.
- This handle of communication between Publisher and Receiver is called as Backpressure handling. And Streams that handle this mechanism are called Reactive Streams.
- Example using Akka Stream:

What are Reactive Streams?

- MATERIALIZER
- A materializer is optional in Akka-streams
- We can use one if we want our Flow to have some side effects like logging or saving results.
- However, we can pass NotUsed alias as a Materializer to denote that our Flow should not have any side effects

What are Reactive Streams?

- RUNNABLE GRAPH
- A graph is a whole set of Outlet/FlowShape/Inlet linked together, that is closed.
- After a stream is properly terminated by having both a source and a sink, it will be represented by the Runnable Graph type, indicating that it is ready to be executed.

BroadCasting In Akka Streams

- One of the most straightforward ways to partition a stream is to simply broadcast every upstream element to a set of downstream Akka Streams: Streaming Live consumers.
- Every downstream consumer will receive exactly the same stream of messages and can independently apply filtering, transformations, aggregations, or side-effects.
- Basically, **Broadcast** ingests events from one input and emits duplicated events across more than one output stream.

What shall users of Akka Streams expect?

- Akka is built upon a conscious decision to offer APIs that are minimal and consistent—as opposed to easy or intuitive.
- Akka Streams:
- all features are explicit in the API, no magic
- supreme compositionality: combined pieces retain the function of each part
- exhaustive model of the domain of distributed bounded stream processing

What shall users of Akka Streams expect?

- This means that we provide all the tools necessary to express any stream processing topology, that we model all the essential aspects of this domain (back-pressure, buffering, transformations, failure recovery, etc.) and that whatever the user builds is reusable in a larger context.

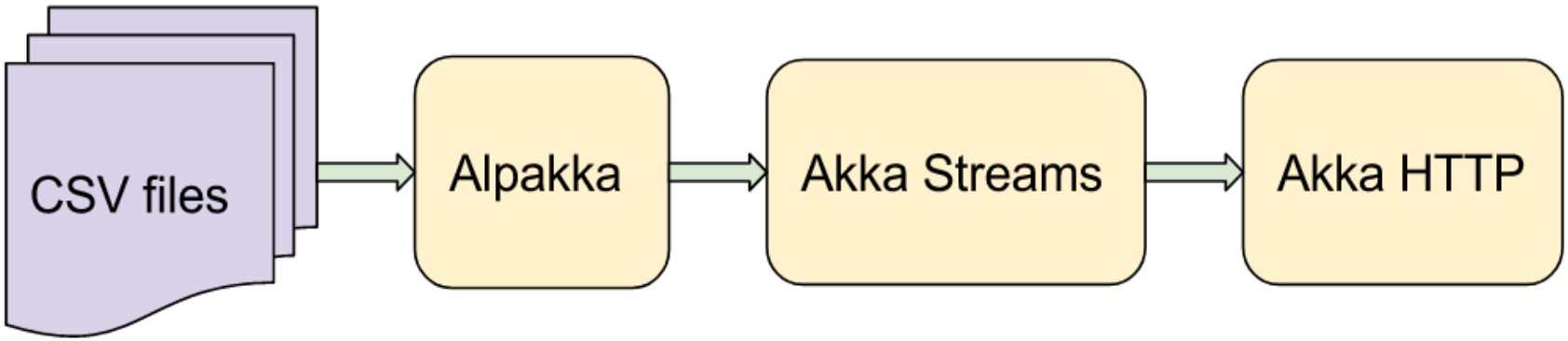
Use Case

- Let’s say we have a directory that we want to watch for new CSV files, then process every file in a streaming fashion, perform some on-the-fly aggregations and send the aggregated results to a websocket (in real time).
- Additionally, we want to define some kind of threshold for the aggregated data to trigger email notifications.
- In our case the CSV lines will contain some (id, value) pairs, with the id changing every second line, e.g.:
- 370582,0.1787070024725666
- 370582,0.5262255382633264
- 441876,0.3099802526590945 7
- 441876,0.3141591265785087
- 722246,0.7334219632071504
- 722246,0.5310146239777006

Use Case

- We want to compute an average value of every two lines sharing the same id and only send it to the websocket when it’s greater than 0.9.
- Moreover, we want to send a notification email on every 5th value sent to the websocket.
- Finally, we want to read and display the data received from the websocket with a trivial JavaScript frontend.

The architecture



The architecture

- The core will of course be Akka Streams, which lets us process the data in real time and in a streaming fashion.
- To read the CSV files, we're going to use Alpakka, which is a set of Akka Streams connectors to various technologies, protocols or libraries.
- Since Akka Streams are Reactive Streams, the entire Alpakka ecosystem is also accessible to any other RS implementation - this is the benefit of interoperability that the RS interfaces set out to achieve.
- Finally, we're going to use Akka HTTP to expose a websocket endpoint.
- The nice thing here is that Akka HTTP seamlessly integrates with Akka Streams (which it actually uses under the hood), so exposing a stream as a websocket is trivial.

Pros and Cons Of Using Akka Streams

- Pros :
- Akka streams implement something called reactive manifesto which is great to achieve really low latency.
- Akka streams provide a lot of operators to write declaratively transformations over streams easily.
- One of the best approach to use when working with a real-time or reactive category.
- In case you have more of custom needs such as running a ton of varying jobs with different actors/flows, then Akka streams can help you in that.
- Akka streams are quite economical in some cases when it comes to threads.

Pros and Cons Of Using Akka Streams

- Cons:
- Using Akka streams imposes some overhead in getting up and running. It requires learning the DSL and also understanding what's going on under the hood to some extent.
- Implementation hard to read
- Conceptually very complicated
- Unlike heavier "streaming data processing" frameworks, Akka Streams are not "deployed" nor automatically distributed.

Kotlin

- **Kotlin** is a general-purpose, statically typed, and open-source programming language.
- It runs on JVM and can be used anywhere Java is used today. It can be used to develop Android apps, server-side apps and much more.

Kotlin

- History of Kotlin
- **Kotlin** was developed by JetBrains team. A project was started in 2010 to develop the language and officially, first released in February 2016. Kotlin was developed under the Apache 2.0 license.

Kotlin

- Features of Kotlin
- **Concise:** Kotlin reduces writing the extra codes. This makes Kotlin more concise.
- **Null safety:** Kotlin is null safety language. Kotlin aimed to eliminate the NullPointerException (null reference) from the code.Interoperable.
- **Interoperable:** Kotlin easily calls the Java code in a natural way as well as Kotlin code can be used by Java.
- **Smart cast:** It explicitly typecasts the immutable values and inserts the value in its safe cast automatically.
- **Compilation Time:** It has better performance and fast compilation time.
- **Tool-friendly:** Kotlin programs are build using the command line as well as any of Java IDE.
- **Extension function:** Kotlin supports extension functions and extension properties which means it helps to extend the functionality of classes without touching their code.
-

Using Kotlin for Server-side Development

- **Expressiveness**: Kotlin's innovative language features, such as its support for type-safe builders and delegated properties, help build powerful and easy-to-use abstractions.
- **Scalability**: Kotlin's support for coroutines helps build server-side applications that scale to massive numbers of clients with modest hardware requirements.
- **Interoperability**: Kotlin is fully compatible with all Java-based frameworks, which lets you stay on your familiar technology stack while reaping the benefits of a more modern language.
- **Migration**: Kotlin supports gradual, step by step migration of large codebases from Java to Kotlin. You can start writing new code in Kotlin while keeping older parts of your system in Java.

Using Kotlin for Server-side Development

- Tooling: In addition to great IDE support in general, Kotlin offers framework-specific tooling (for example, for Spring) in the plugin for IntelliJ IDEA Ultimate.
- Learning Curve: For a Java developer, getting started with Kotlin is very easy. The automated Java to Kotlin converter included in the Kotlin plugin helps with the first steps.
- Kotlin Koans offer a guide through the key features of the language with a series of interactive exercises.

Kotlin Data Type

- In Kotlin, everything is an object, which means we can call member function and properties on any variable.

- Kotlin built in data type are categorized as following different categories:

- Number
- Character
- Boolean
- Array
- String

Questions

Module Summary

- Spring Integration Framework.
- Message, Channel and Adapter
- Understood the different Component Integration
- Understood the Event-Driven Architecture