INTRODUCTION TO SPRING BOOT & CLOUD FOUNDRY

Spring Boot

- Stand-alone, production-grade applications

  Ready to run.
- “Opinionated” view of the Spring platform & 3rd party libraries
  Minimizes manual Spring configuration.
- Don’t need to concerned with every aspect of its lifecycle including deployment and management.
- Spring Boot Definition

Spring Boot

"Menu-based" creation of stand-alone, production-grade Apps

Spring Boot applications need a reduced amount of Spring configuration.

Pre-configured POM files to assist your initial project configuration.

Eliminates some configuration work and has no requirements for XML

Support for security, metrics, and health checks - production-ready apps

Dev-Ops friendly - developers focus on business features not on infrastructure.

Designed to make microservices a resource-conscious, developer-focused process.

Assists in decomposing monolithic services into distributed microservices

Spring Boot Reference

Spring Boot Application Properties

Spring Boot Developer Tools

Spring Boot & Microservice Architecture

A Microservice Architecture typically does NOT have a built in UI component [Presentation Tier].

It is a distributed “business” service architecture [RE: RESTful Web Services].

By default, Spring (Boot) applications have only one ApplicationContext.

There's no distinction between a root web application context and a servlet web application context in Spring Boot.

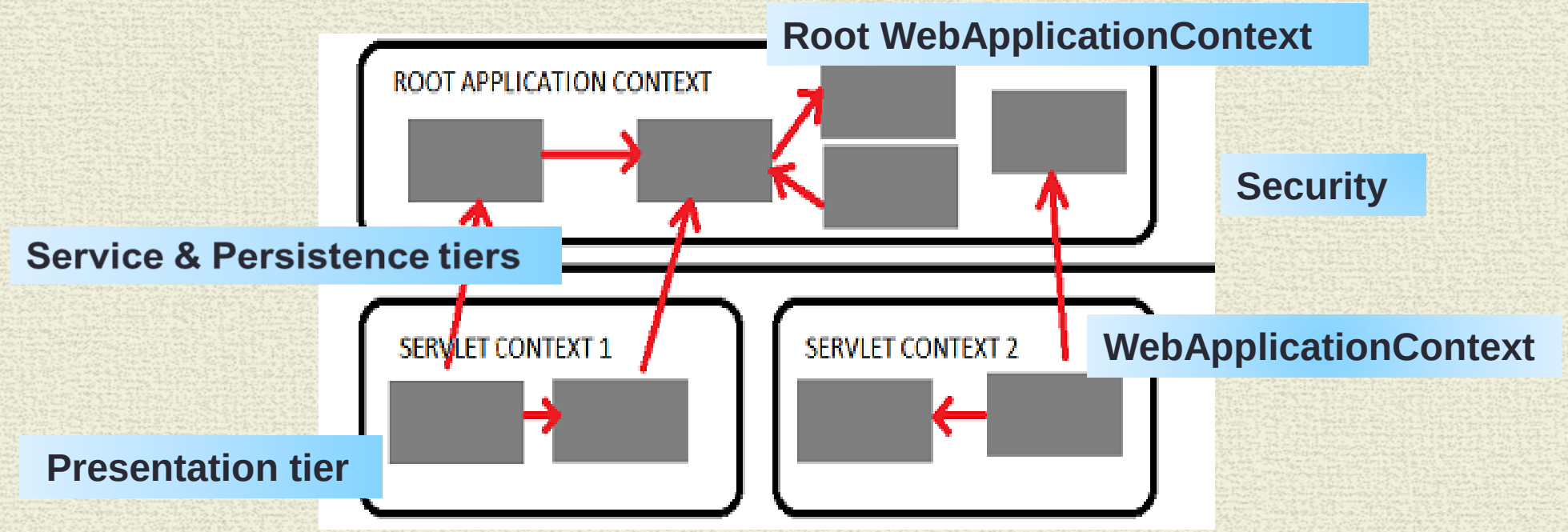
So,

by default, a Spring Boot application with “built in” Server Side Rendering [HTML page delivery] is a Monolith…

Web Application Context

Spring has multilevel application context hierarchies.

Web apps by default have two hierarchy levels, root and servlet contexts:



Spring Boot requires custom Configuration to separate the Servlet Web Application Context from the Root WebApplication Context.

Application Start

- The @SpringBootApplication annotation is equivalent to using @Configuration, @**EnableAutoConfiguration** and @ComponentScan with their default attributes

- It is on “main” method in ROOT package…

```
@SpringBootApplication
public class HelloSpringThymeBootApplication {
  public static void main(String[] args) {
   SpringApplication.run(HelloSpringThymeBootApplication.class, args);
  }
}
```

Customization Example:
@SpringBootApplication(scanBasePackages = “edu.mum")

- @SpringBootApplication allows for customization of the attributes of @EnableAutoConfiguration and @ComponentScan.

@AutoConfiguration

Guess and configure beans that you are likely to need.

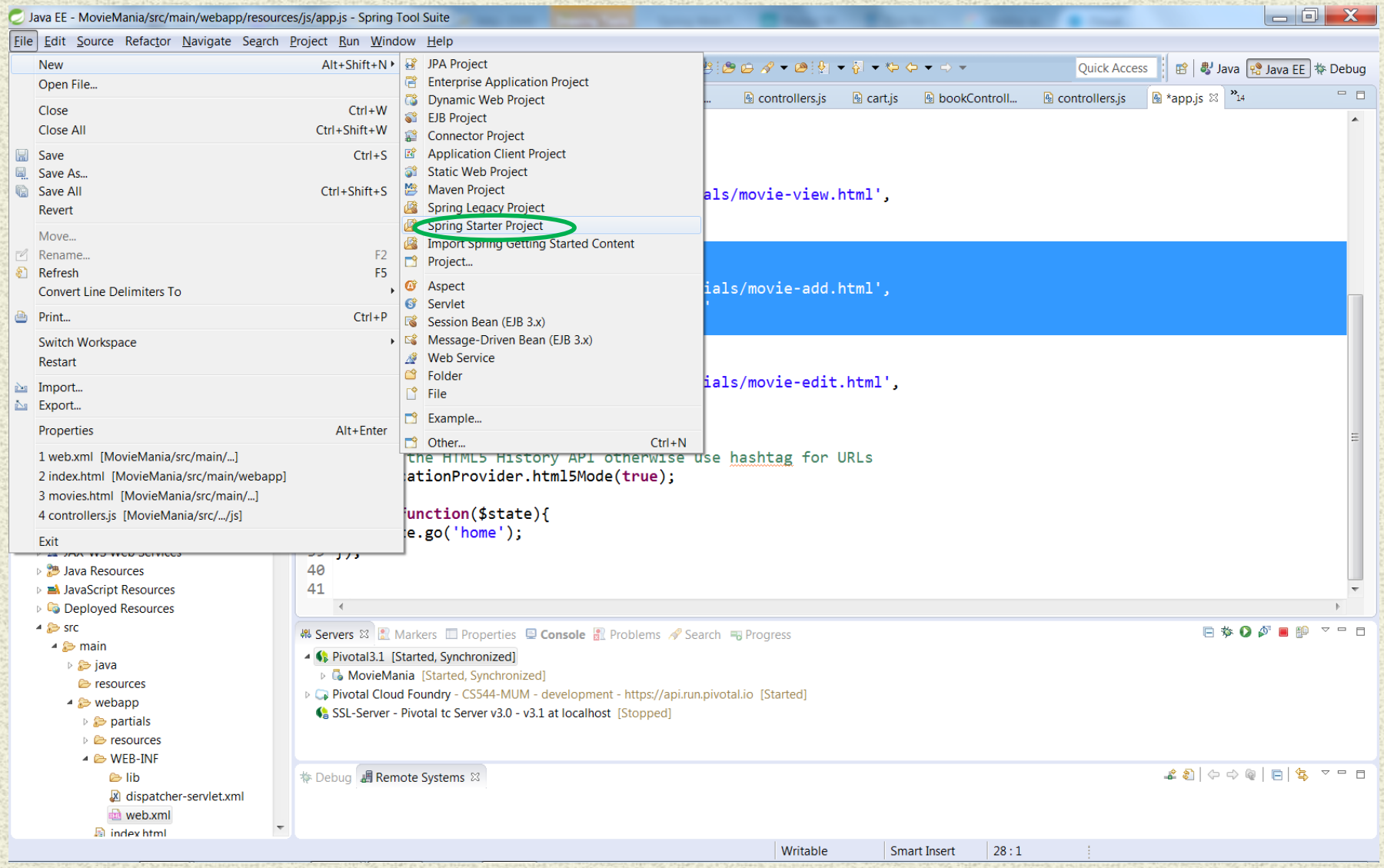
Auto-configuration classes are usually applied based on your classpath and what beans you have defined.

Auto-configuration tries to be as intelligent as possible and ***will back-away as you define more of your own configuration.***

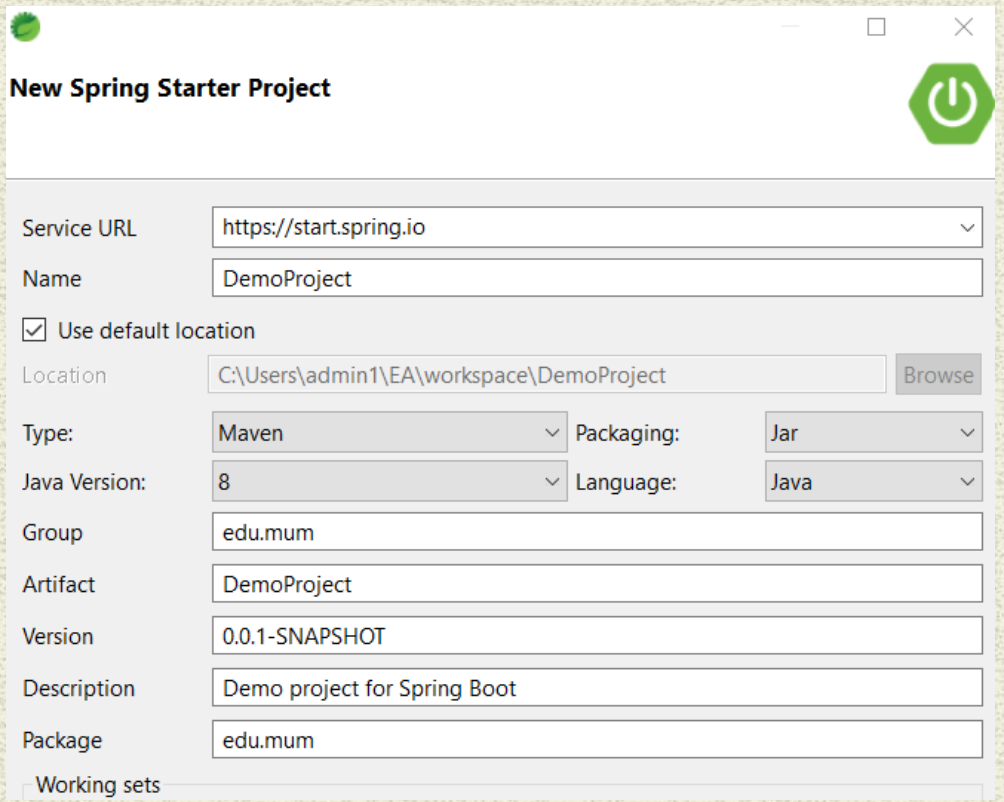
Customization of Boot configured beans is through:

Spring Application Properties

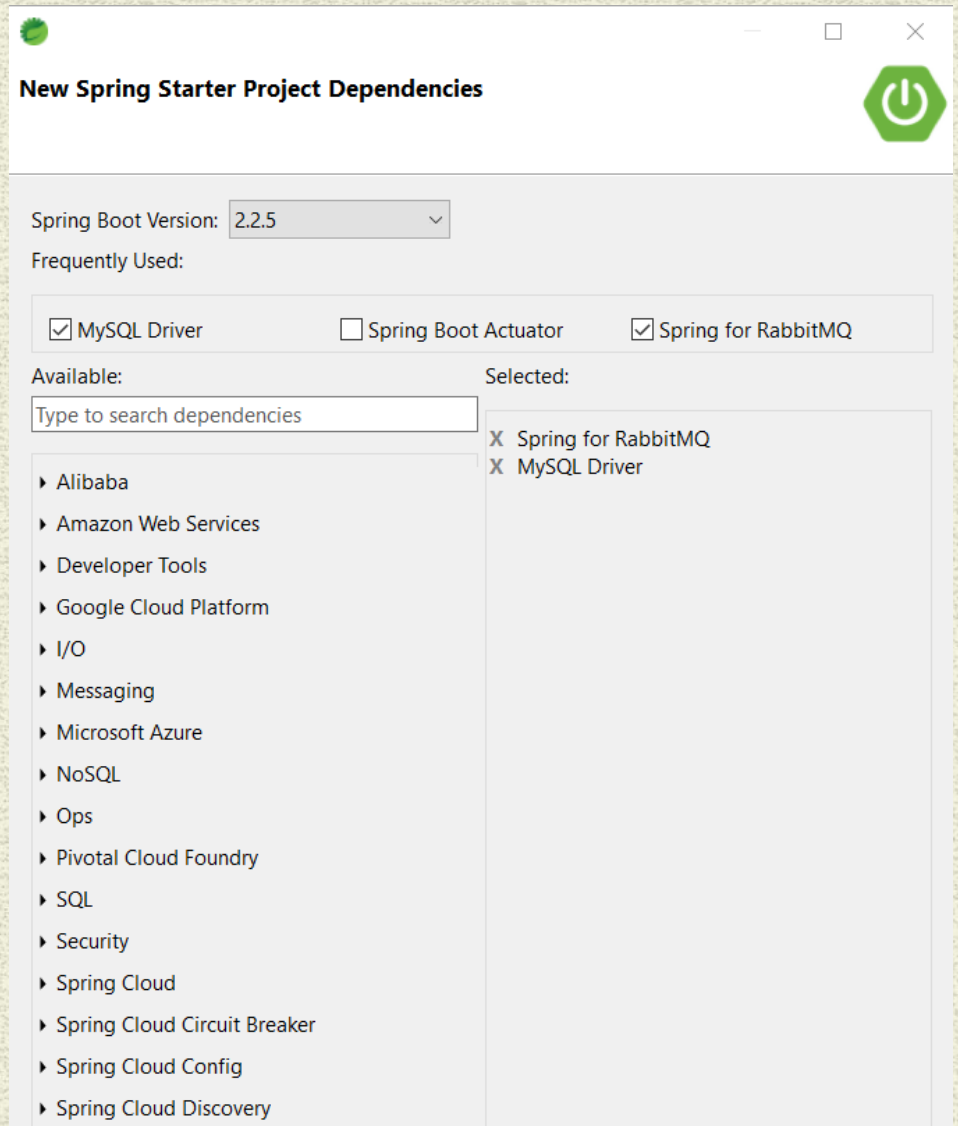
Start a Project with Spring Boot



New Spring Boot Project



- Menu Selection



Spring Boot
Custom Configuration

Register Jersey Rest Services
Set Application Path

```
@Configuration
@ApplicationPath("/MongoJerry")
public class JerseyConfig extends ResourceConfig {

    public JerseyConfig() {

        register(RequestContextFilter.class);
        register(ProductRestService.class);
    }
}
```

Application Properties

name	value
spring.data.mongodb.database	eacore

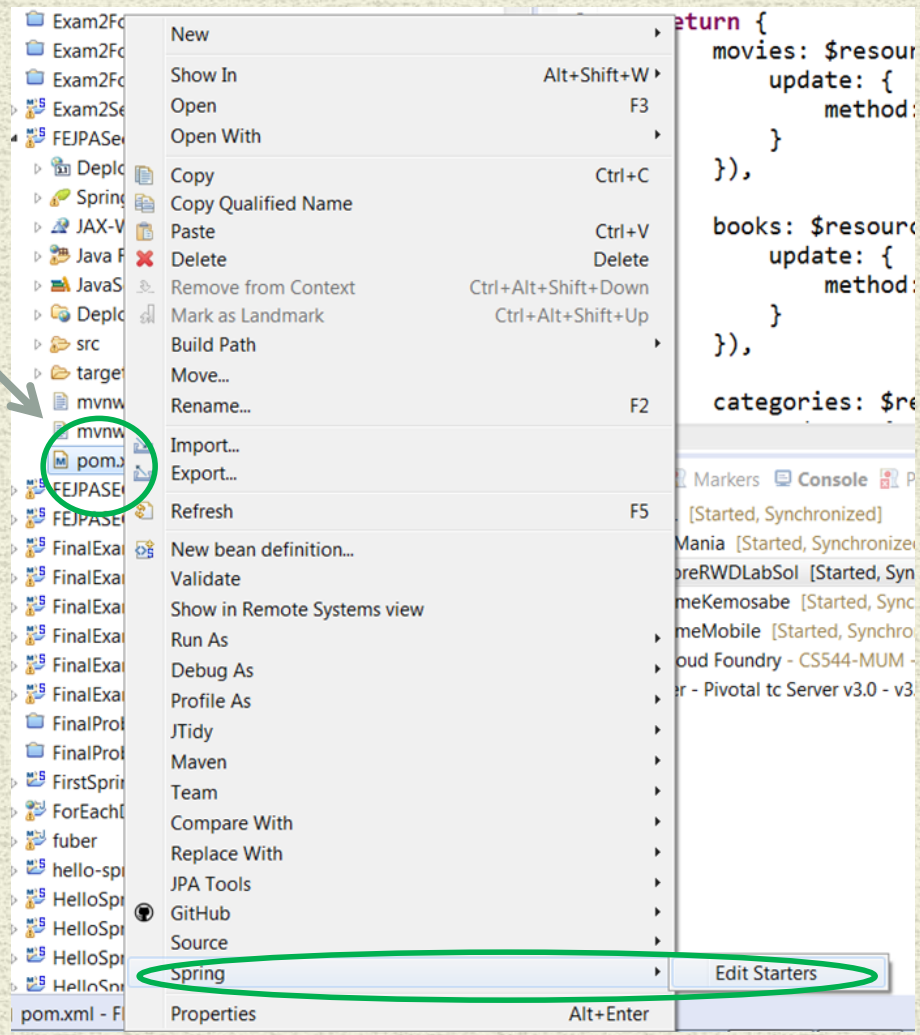
Security Authentication
Default users & Authorization

```
@Configuration
    @EnableGlobalMethodSecurity(prePostEnabled = true)
    @EnableWebSecurity
    class SecurityConfig extends WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http.authorizeRequests().anyRequest().fullyAuthenticated();
            http.httpBasic();
            http.csrf().disable();
        }

        @Autowired
        public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
            auth.inMemoryAuthentication()
                    .withUser("user").password("password").roles("USER").and()
                    .withUser("admin").password("admin").roles("ADMIN", "USER");
        }
    }
```

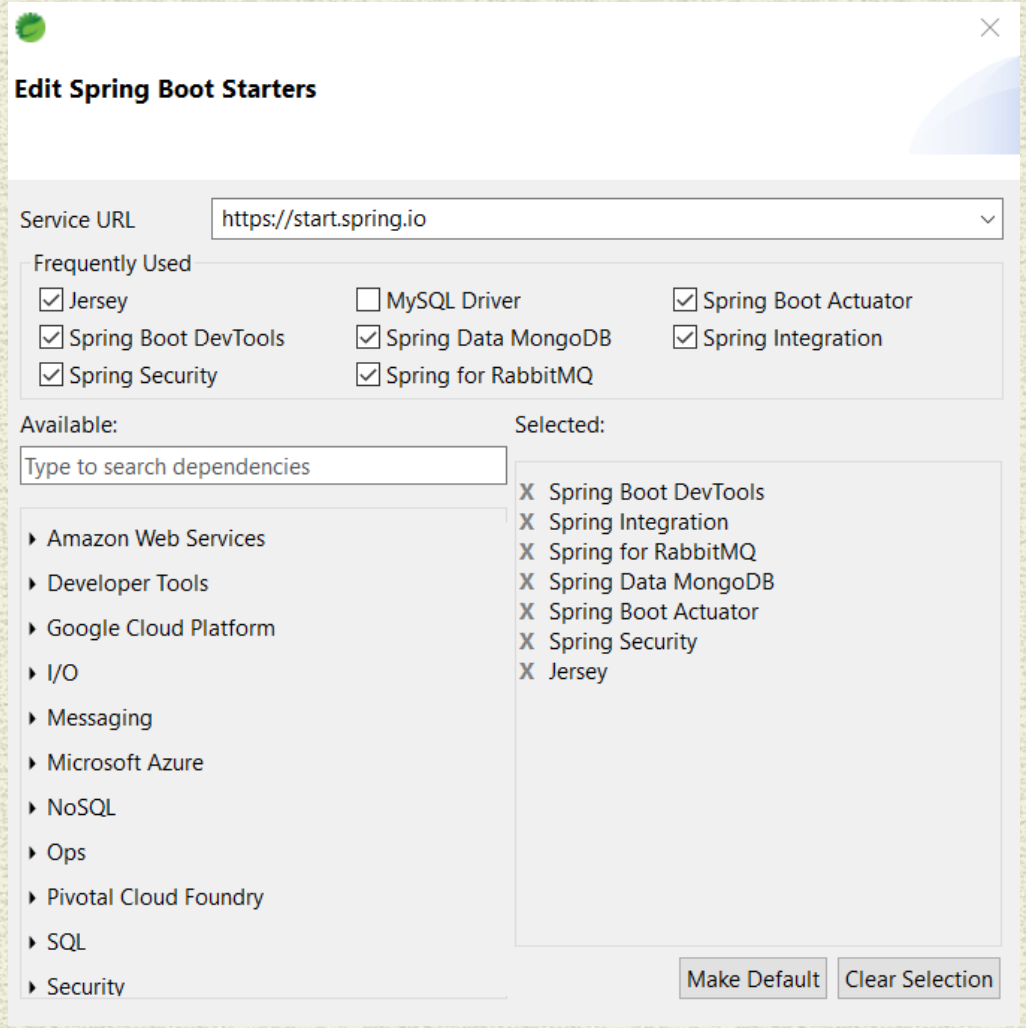
See Demo for More Details

Edit an Existing Boot Project

- The *Edit Starters* wizard is accessible by right-clicking a Maven project's pom.xml, from menu *Spring* >> *Edit Starters*.



Edit Starters Screen



The Cloud

- Very broad concept
- covers just about every possible sort of online service
- USUALLY refers to

Software as a Service (SaaS)

3rd party software [SalesForce, DropBox, etc.]

Platform as a Service (PaaS)

Platform on which software is developed/deployed.

Infrastructure as a Service (IaaS).

Automated and scalable computing resources, cloud storage and networks. Client control over infrastructure

Cloud Migration Strategies

Lift-and-shift [Rehosting]

Reduction of cost and improved performance and resiliency.
applications are easier to optimize/re-architect once they're already running in the cloud.

Replatforming ["lift-tinker-and-shift."]

Database-as-a-service
Embedded Tomcat

Refactoring / Re-architecting

AKA use cloud-native features AKA use microservices
driven by a strong business need – use case
most expensive, but, if you have a good product-market fit

“how to build a successful app” by Heruku 12 factor apps
…adopted by … major software platforms and frameworks***

12 Factor [Cloud] Apps

- **Codebase**

  One codebase tracked in revision control, many deploys

  [Use SVN;github] **

- **Dependencies**

  Explicitly declare and isolate dependencies
  [Maven;gradle] ***

- **Config**

  Store config in the environment

  [anything that varies between environments.]

- **Backing_Services**

  Treat backing services as attached resources

  [CLOUD – Attach services]**

12 Factors

- **Build, release, run**

  Strictly separate build and run stages
- **Processes**

  Execute the app as one or more stateless processes
- **Port binding**

  Export services via port binding
- **Concurrency**

  Scale out via the process model
- **Disposability**

  Maximize robustness - fast startup and graceful shutdown
- **Development – Production Parity**

  Keep development, staging, and production similar
- **Logs**

  Treat logs as event streams
- **Admin processes**

  Run admin/management tasks as one-off processes

Cloud Foundry
A multi-cloud Approach

Cloud Foundry is an application PaaS, and let's you concentrate on your application and its associated services instead of the infrastructure itself.

Can be built on a range of cloud providers. [AWS, Microsoft Azure, Google Cloud Platform]

- Single platform for application development.
- Removes grunt work of running the infrastructure
- No infrastructure platform lock in.
- Easily move applications from one cloud to another (or use multiple clouds simultaneously)
- Open Source Foundation [70+ members] includes Microsoft, Google, IBM
-

Cloud Foundry, Pivotal Cloud Foundry Pivotal Web Services

- **DEFINITIONS:**

- **Cloud Foundry** open source software is the community maintained , community supported software and tools required to run Cloud Foundry on your own infrastructure.
- https://cloudfoundry.org

- **Pivotal Cloud Foundry** is a Pivotal branded installation of Cloud Foundry on your own infrastructure (AWS, VMware, OpenStack, Azure, etc) which has enterprise grade support. It also offers an array of services like MySQL,, RabbitMQ, etc. that can leverage their cloudfoundry installation and have applications that use these services.

Pivotal Cloud Foundry

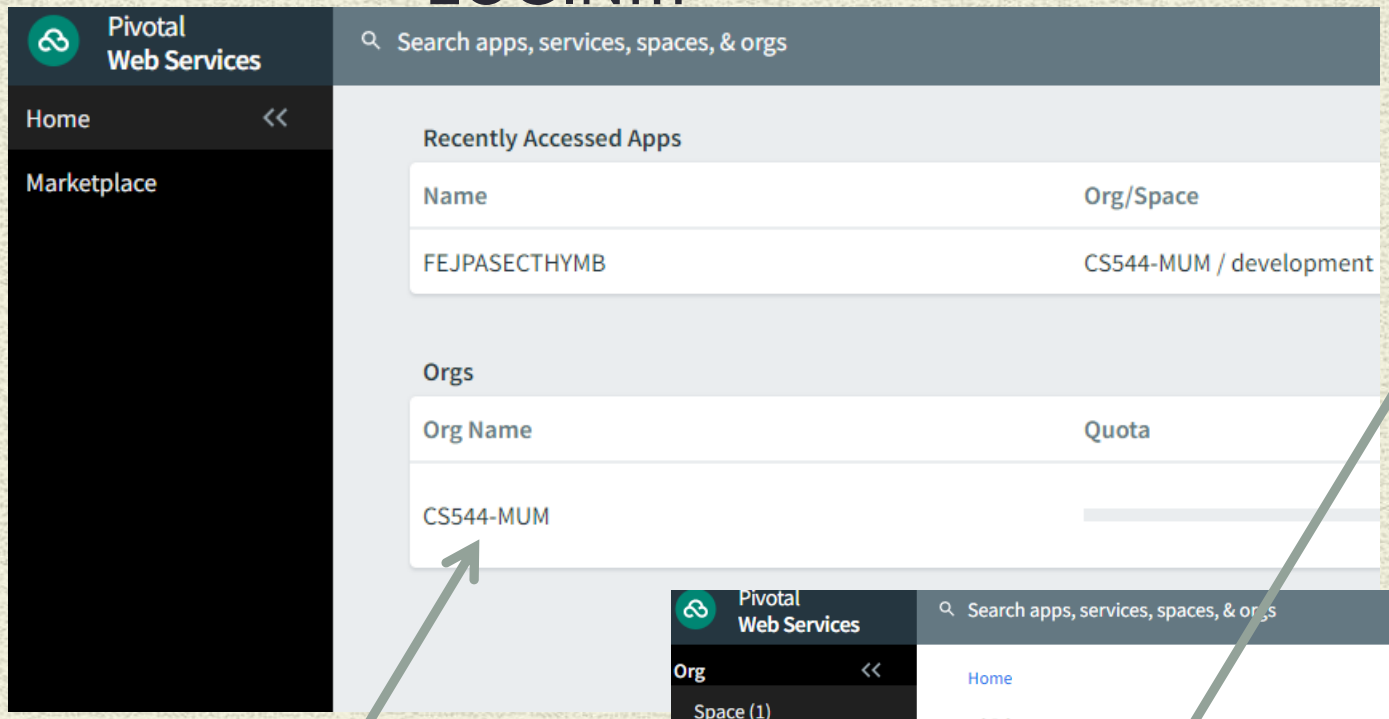
- **Pivotal Web Services** is a Pivotal's hosted Cloudfoundry for people and companies who want to develop cloudfoundry applications but don't want to operate and manage their own infrastructure and Cloud Foundry installation. ***Deployed on AWS***
- https://run.pivotal.io
-

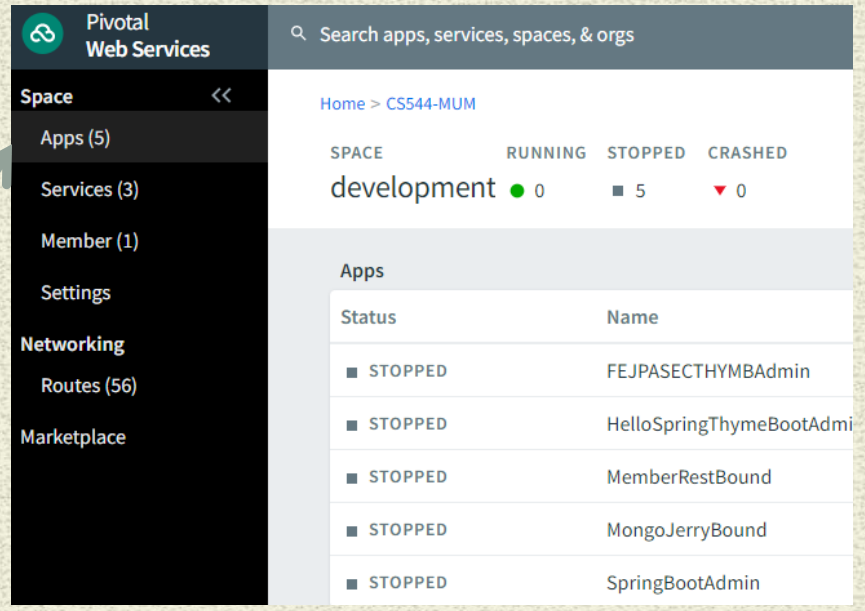
Spring Cloud Support

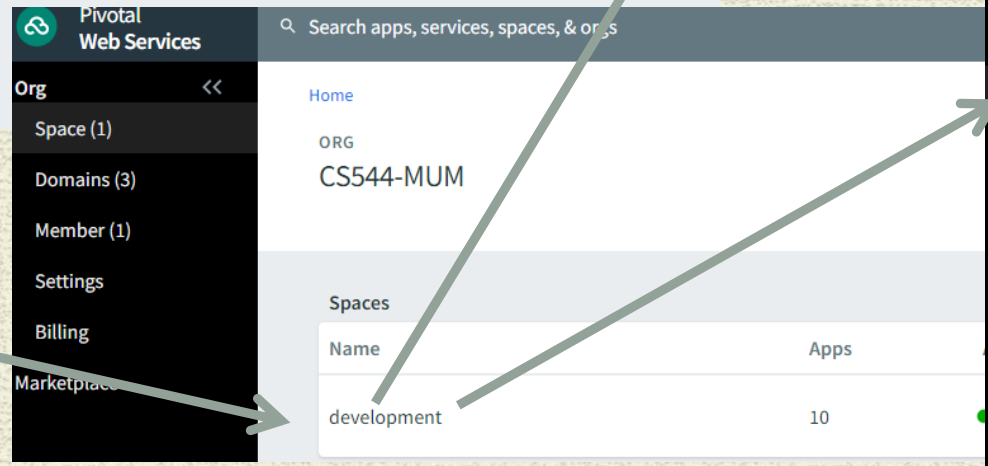
- Spring Cloud Config
- Centralized external configuration management backed by a git repository. The configuration resources map directly to Spring `Environment` but could be used by non-Spring applications if desired.
- Spring Cloud Netflix
- Integration with various Netflix OSS components (Eureka, Hystrix, Zuul, Archaius, etc.)
- Spring Cloud Bus
- An event bus for linking services and service instances together with distributed messaging. Useful for propagating state changes across a cluster (e.g. config change event)
- Spring Cloud for Cloud Foundry
- Integrates your application with Pivotal Cloudfoundry. Provides a service discovery implementation and also makes it easy to implement SSO and OAuth2 protected resources, and also to create a Cloud Foundry service broker.
- Spring Cloud Cluster
- Leadership election and common stateful patterns with an abstraction and implementation for Zookeeper, Redis, Hazelcast, Consul.
- Spring Cloud Consul
- Service discovery and configuration management with Hashicorp Consul.
- **ETC…, ETC..., ETC…**

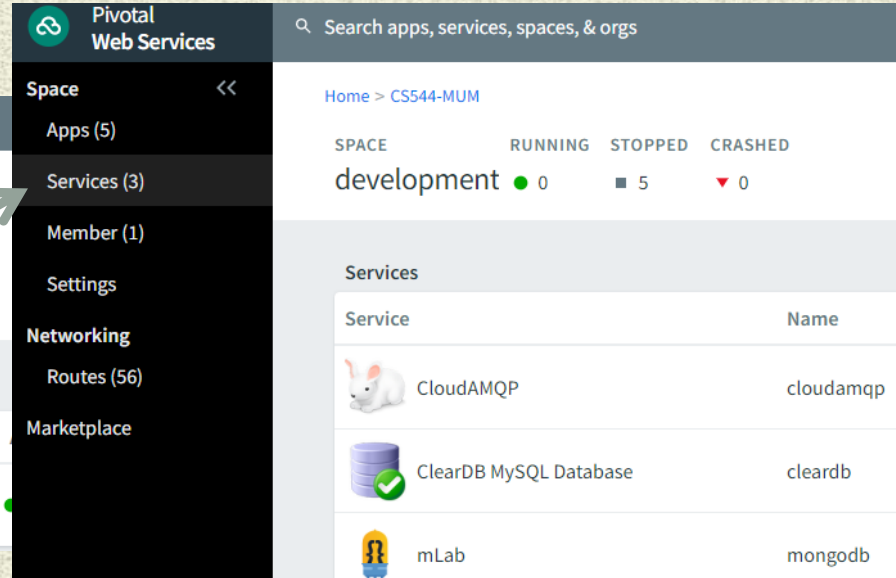
Pivotal Cloud Foundry

- Pivotal Web Services Home
- LOGIN…









Click On Org
Click on Space

Cloud Foundry CLI Login

[Cloud Foundry Command Line Interface]

Get endpoint…[version #]

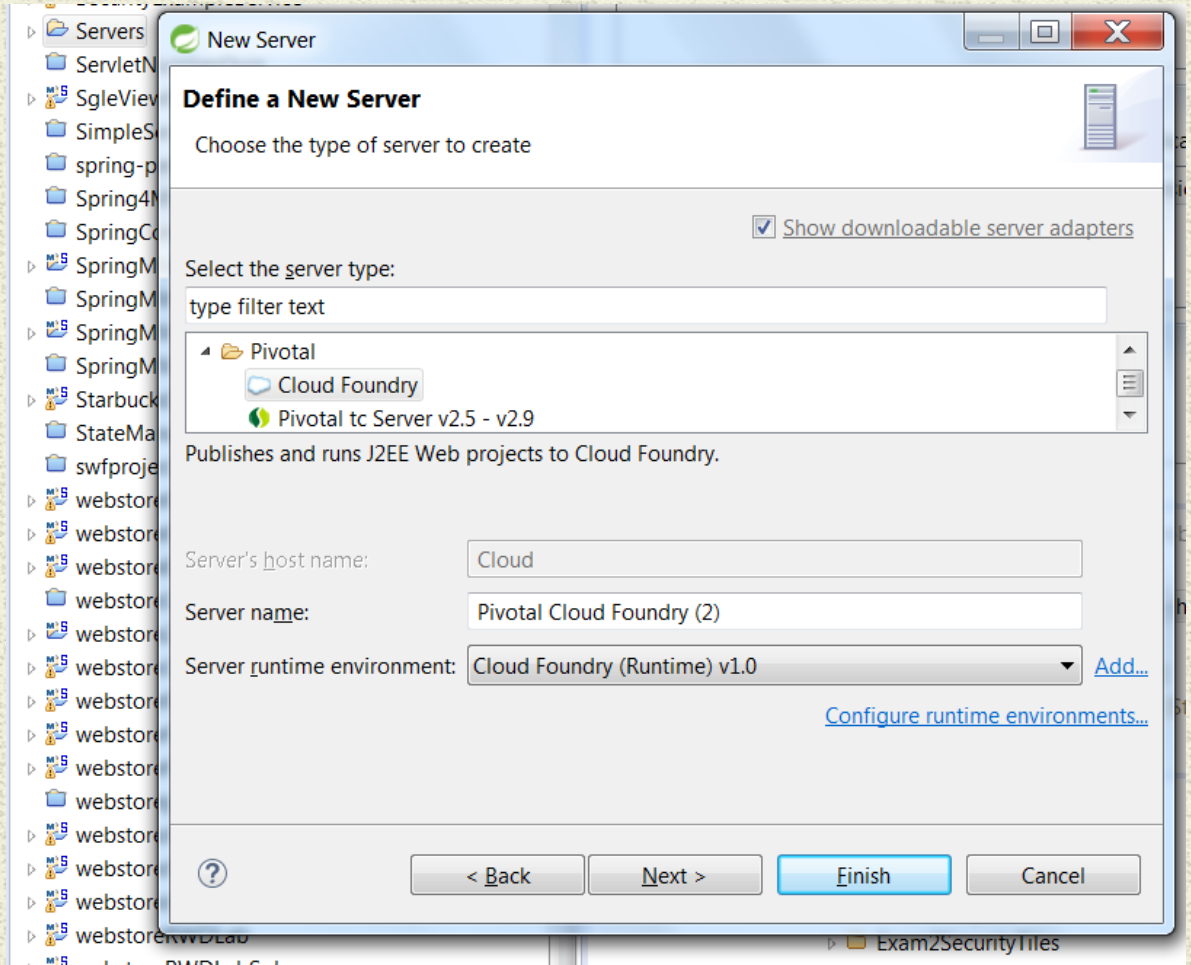
```
c:\Program Files\Cloud Foundry>cf api
api endpoint:   http://api.run.pivotal.io
api version:    2.85.0

c:\Program Files\Cloud Foundry>cf login
API endpoint: http://api.run.pivotal.io
Warning: Insecure http API endpoint detected: secure https API endpoints are recommended


Email> jbruen@mum.edu

Password>
Authenticating...
OK

Targeted org CS544-MUM

Targeted space development



API endpoint:   http://api.run.pivotal.io (API version: 2.85.0)
User:           jbruen@mum.edu
Org:            CS544-MUM
Space:          development
```

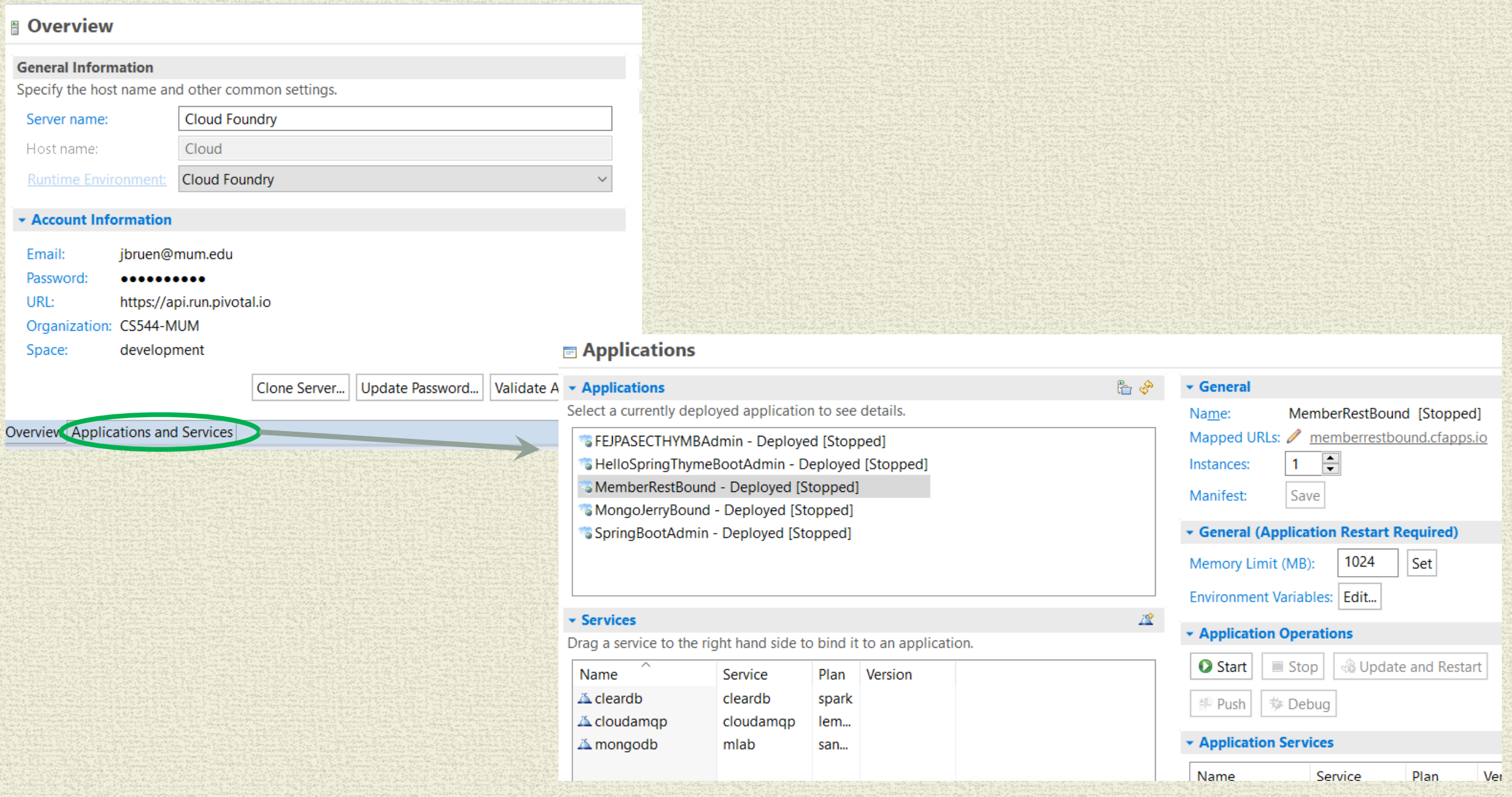
View Apps & Services

```
c:\Program Files\CloudFoundry>cf apps
Getting apps in org CS544-MUM / space development as jbruen@mum.edu...
OK

name                 requested state   instances   memory   disk   urls
FEJPASECB            stopped           0/1         512M     1G     FEJPASECB.cfapps.io
FEJPASECTHYMB        stopped           0/1         512M     1G     FEJPASECTHYMB.cfapps.io
HelloSpringBoot1     stopped           0/1         1G       1G     hellospringboot1.cfapps.io
HelloSpringThymeBoot stopped           0/1         512M     1G     HelloSpringThymeBoot.cfapps.io
MongoJerry           stopped           0/1         512M     1G     mongojerry.cfapps.io

c:\Program Files\CloudFoundry>cf services
Getting services in org CS544-MUM / space development as jbruen@mum.edu...
OK

name      service   plan      bound apps                last operation
cleardb   cleardb   spark     FEJPASECB, FEJPASECTHYMB  create succeeded
mongodb   mlab      sandbox   MongoJerry                create succeeded
```

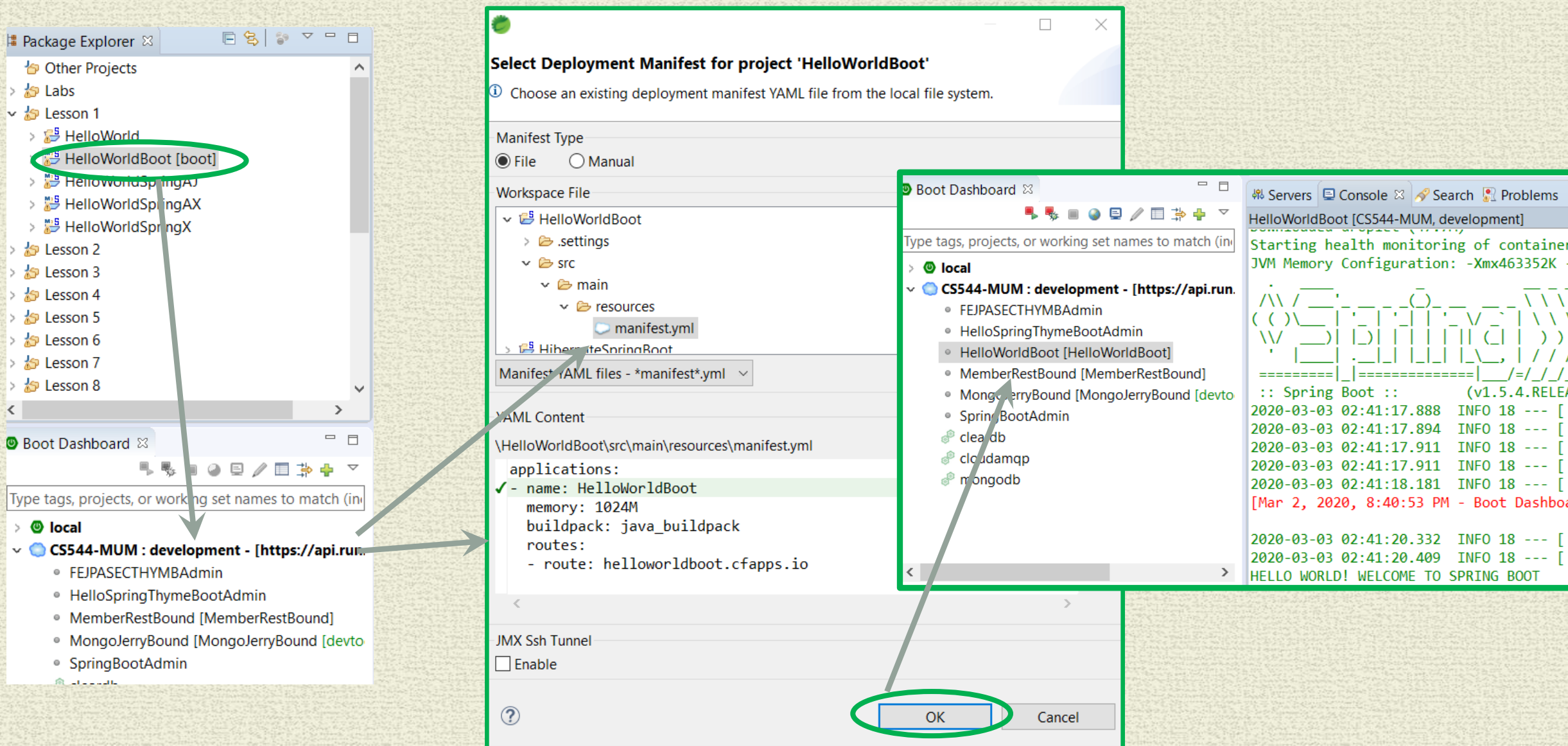
Using Eclipse Plugin

- Cloud Foundry Eclipse Plugin
- Start new server right click servers [project explorer]
  new >> other >> servers >> server …Pivotal >> Cloud Foundry




Overview
General Information
Specify the host name and other common settings.
Server name: Cloud Foundry
Host name: Cloud
Runtime Environment: Cloud Foundry
Account Information
Email: jbruen@mum.edu
Password: ••••••••••
URL: https://api.run.pivotal.io
Organization: CS544-MUM
Space: development
Clone Server...
Update Password...
Validate A
Overview
Applications and Services
Applications
Applications
Select a currently deployed application to see details.
FEJPASECTHYMBAdmin - Deployed [Stopped]
HelloSpringThymeBootAdmin - Deployed [Stopped]
MemberRestBound - Deployed [Stopped]
MongoJerryBound - Deployed [Stopped]
SpringBootAdmin - Deployed [Stopped]
Services
Drag a service to the right hand side to bind it to an application.
Name
Service
Plan
Version
cleardb
cleardb
spark
cloudamqp
cloudamqp
lem...
mongodb
mlab
san...
General
Name: MemberRestBound [Stopped]
Mapped URLs: memberrestbound.cfapps.io
Instances: 1
Manifest: Save
General (Application Restart Required)
Memory Limit (MB): 1024
Set
Environment Variables: Edit...
Application Operations
Start
Stop
Update and Restart
Push
Debug
Application Services
Name
Service
Plan

Drag & Drop Boot Dashboard



Cloud Deployment Types
[W/R to Application Servers]

No External Server

Package the application, with all its dependencies, into a single “fat” JAR file - can include an embedded framework with optional third-party libraries that will be compatible.

{Spring Boot w/Embedded Tomcat}

Cloud provided Server

Package a **container [e.g** Java EE,Tomcat **]** and its service implementation in a Container [Linux - Docker,Heruku]. Layered, cached dependencies; Linux type/version agnostic **{Spring Boot w/WAR file}**

CLOUD AMQP Setup

1. Add cloudampq in STS
2. Use plan Lemur [free]

3. ***From pivotal services login YOU can access Rabbit admin:***
4. From “HOME”
5. Click on org name [cs544-MUM]
6. Click on development [space]
7. Click on MongoJerryBound
8. Click on Services
9. Click on cloudamqp
10. Click on manage
11. Click on RabbitMQ Manage…[upper left]
12. THERE YOU ARE!!!