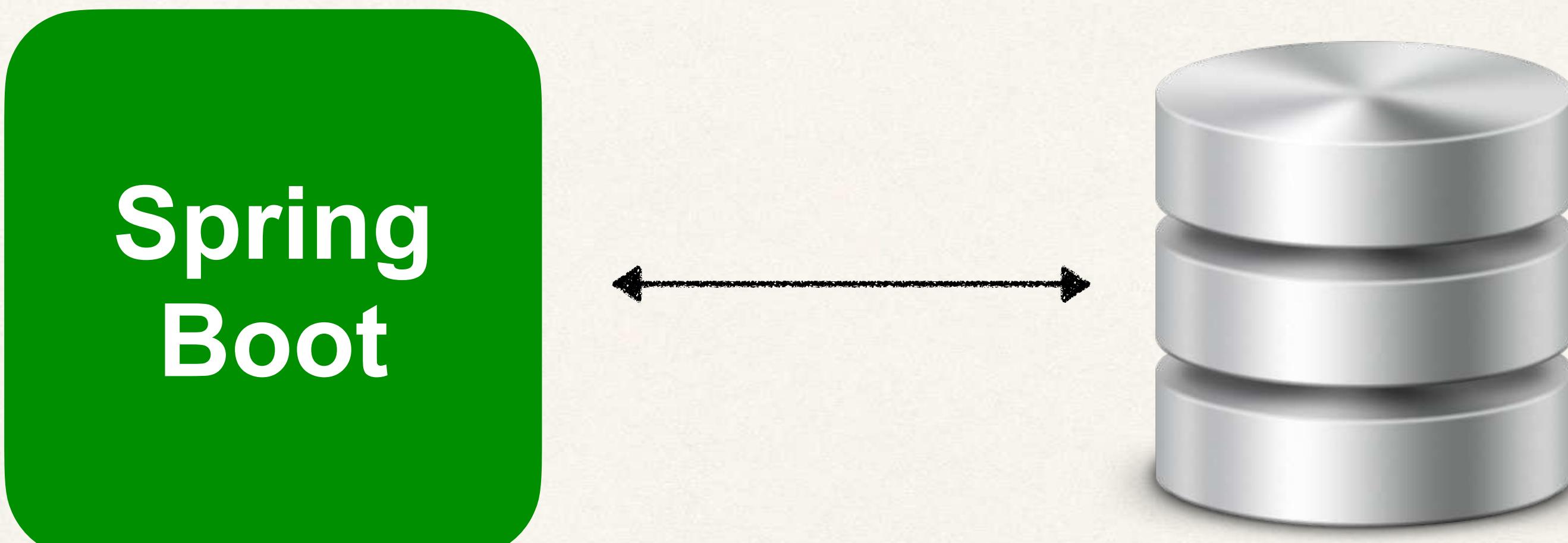


Course Introduction



Local Development

localhost:8080



<< your domain name >>



Main Objectives

- Learn how to deploy your Spring Boot application online
- Live Internet access for your Spring Boot application
- Great way to showcase your skills to the hiring manager

Cloud Hosting

- We will use Amazon Web Services (AWS) for cloud hosting
- Free developer account for 6 months
- No prior experience with AWS is required
- I will give an overview in later videos :-)



Course Road Map

- Amazon Web Services (AWS) Overview
- Create a free AWS Developer Account
- Deploy Spring Boot app on AWS Cloud
- Add database support on AWS Cloud
- Register a custom domain name

www.myspringbootapp.com

Prerequisites

- You should have basic Spring Boot experience: Spring MVC + DB
- Should have these dev tools installed:
 - Java IDE (IntelliJ, VS Code, Eclipse etc ...)
 - MySQL Workbench

Source Code

- Our classroom Spring Boot app is **already developed** and can run locally
- No major coding is involved in this course
- The focus of this course is **deploying code** ... not developing it

Source Code and PDFs

All source code is available for download

All PDFs of slides are available for download

Questions / Help

If you have questions or need help...

Post the question in the classroom discussion forum

AWS Overview



What is Amazon Web Services (AWS)?

- At a high-level, you can think of AWS as
 - Online hosting service
 - Deploy web applications
 - Deploy databases
- Once deployed, your apps are available online!





What is Amazon Web Services (AWS)?

AWS is a full service cloud platform

Over 90+ services!

With AWS Cloud - you get ...

- On-demand delivery of IT resources via the Internet
 - Spin up servers on-demand (choose your OS etc)
 - Deploy databases in the cloud (choose your DB)
- Pay-as-you-go pricing (**FREE** developer accounts available!)

Go Global in Minutes - Data Centers Worldwide



Select your Services ... then Deploy!

The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, user 'Chad Darby', location 'Oregon', and 'Support' dropdown.

The main area is titled 'AWS services' and features a search bar: 'Find a service by name or feature (for example, EC2, S3 or VM, storage.)'. Below the search bar, there are sections for 'Recently visited services' (Route 53, Elastic Beanstalk, Lambda, CloudWatch, Simple Notification Service) and 'All services' categorized into Compute, Management Tools, Mobile Services, Storage, Media Services, AR & VR, Application Integration, and others.

To the right, there's a 'Helpful tips' sidebar with two items: 'Manage your costs' (with a graph icon) and 'Create an organization' (with a cube icon). Below that is an 'Explore AWS' section featuring 'Amazon Relational Database Service (RDS)' and 'Real-Time Analytics with Amazon Kinesis'.

AWS Free Tier

- For developers, six months FREE
- Enough resources for us to deploy Spring App and DB for FREE

<https://aws.amazon.com/free/>

Practical Results

- Cover the essential AWS services to deploy our Spring App online
- Not an A to Z reference of AWS ... for that, you can see the **AWS Documentation**

<https://aws.amazon.com/documentation/>

AWS Resources

- AWS YouTube Videos (official)

<https://www.youtube.com/user/AmazonWebServices>

- Plenty of AWS courses on [udemy.com](#)
- Search for AWS and review the courses with **Best Seller** tag

AWS Key Services



AWS Key Services

Elastic Cloud Compute (EC2)

Remote VM

Elastic Beanstalk

Deploy web applications

Relational Database Service (RDS)

Database in the cloud

Route 53

Route custom domain names

Elastic Cloud Compute (EC2)

- Remote virtual machine based on your specs
- Select the Operating System (MS Windows or Linux)
- You get the Operating System pre-installed ... that's it
- Manually install your own apps on top of OS (Tomcat, DB, etc ...)



Can also use
custom
Amazon “Images”

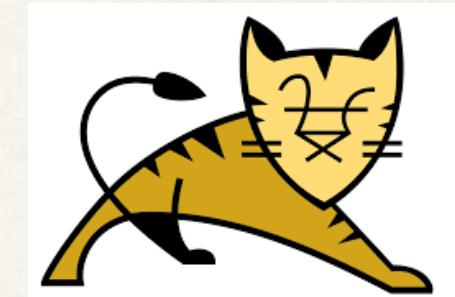
Elastic Beanstalk

- Quickly deploy web applications with Elastic Beanstalk



JAVA

- Select pre-configured virtual machine for your web app stack



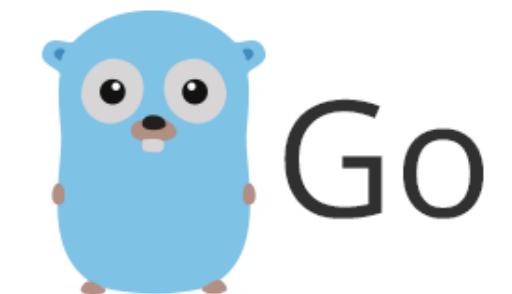
- No software to install on virtual machine



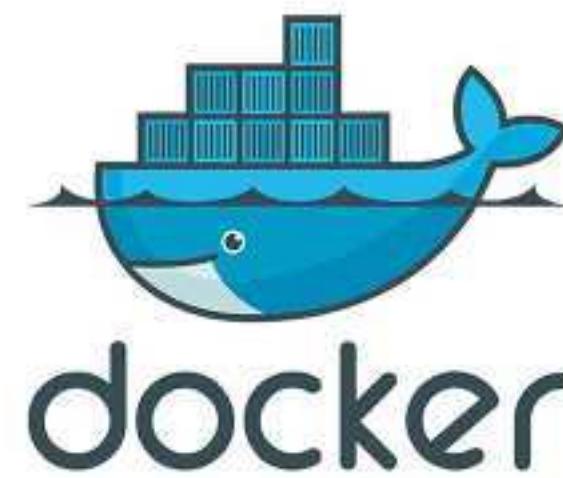
GlassFish



- Just deploy your code (zip, war etc)



Go

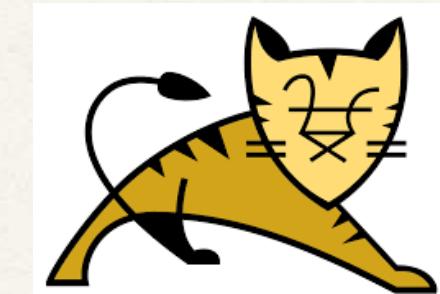


docker

Java on AWS

- You can use regular Java EE APIs
- Can use 3rd-party frameworks: Spring, Hibernate etc
- All based on standard Java

JAVA



THERE ARE NO PROPRIETARY AMAZON HOOKS

Java Web Application Archive (WAR)

- Java Web App specification defines standard deployment file
 - Web Application ARchive (WAR file)
- Zipped archive of your web app with **.war** extension
- Create the WAR file using your IDE or Maven
- Deploy your WAR file to Elastic Beanstalk

mycoolwebapp.war

- **src/main/webapp**
 - **WEB-INF**
 - **WEB-INF/classes**
 - **WEB-INF/lib**

Relational Database Service (RDS)

- Quickly deploy a relational database in the cloud
- You can manage it using normal admin tool:
 - MySQL Workbench
 - Oracle SQL Developer, etc ...
- AWS also has support for NoSQL databases
 - MongoDB, etc ...

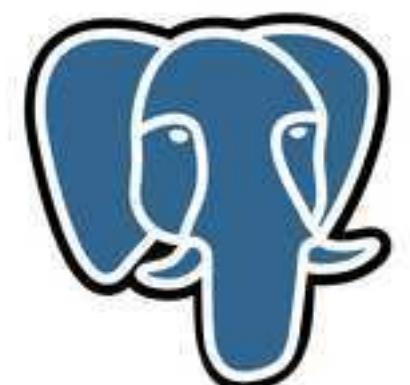


MySQL

Oracle



PostgreSQL



Route 53

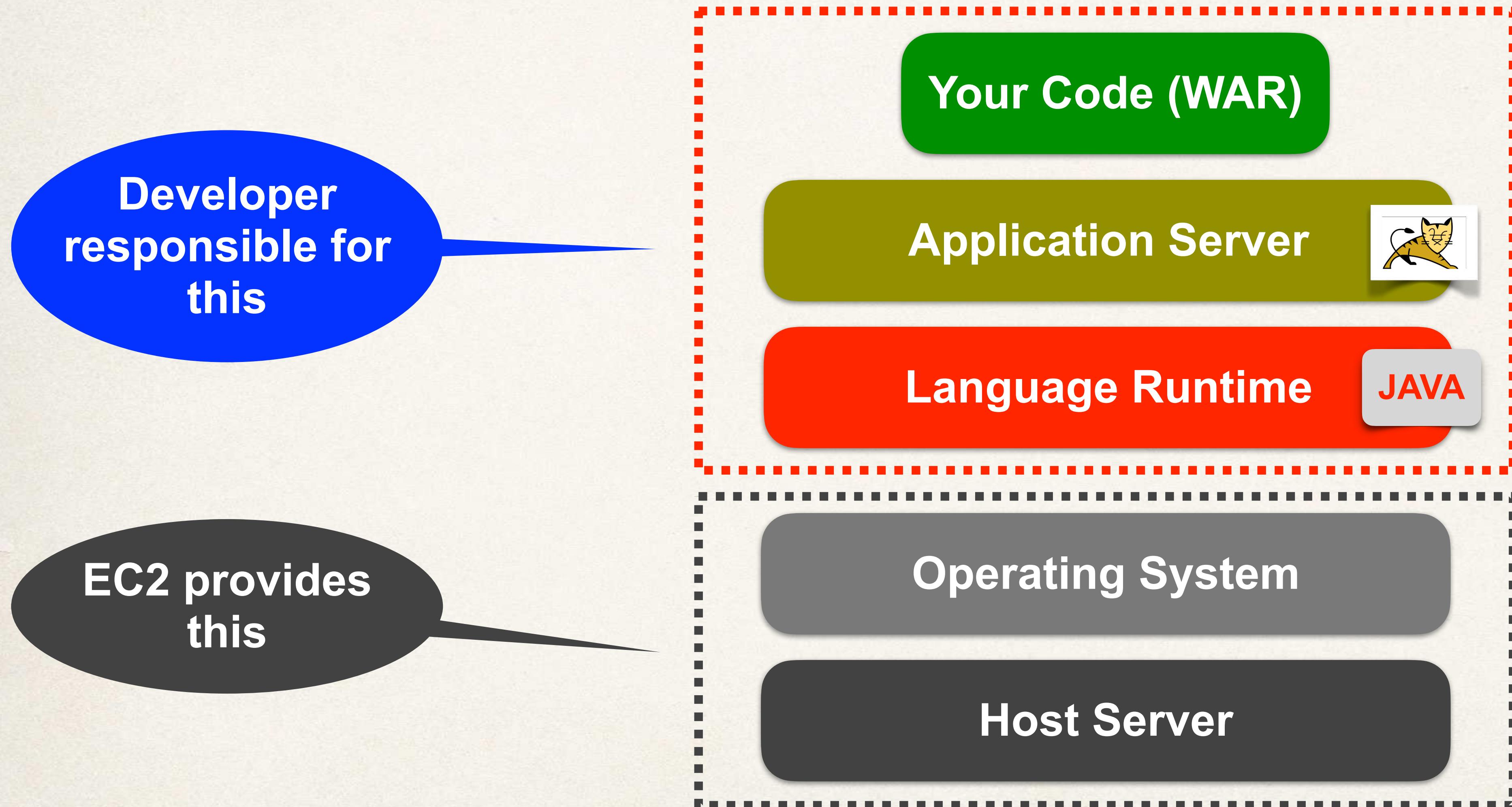


- Routes your custom domain name to your app on AWS
- Configure Route 53 to send www.myspringbootapp.com to your AWS app
- This is the AWS Domain Name System (DNS)
- We'll use it later in the course for our custom domain name :-)

Comparing EC2 to Elastic Beanstalk

- EC2 is a Do-It-Yourself solution
- All you get is the operating system
 - MS Windows or Linux
- If you want other apps
 - You have to manually install them
 - JDK, Tomcat, MySQL etc ...

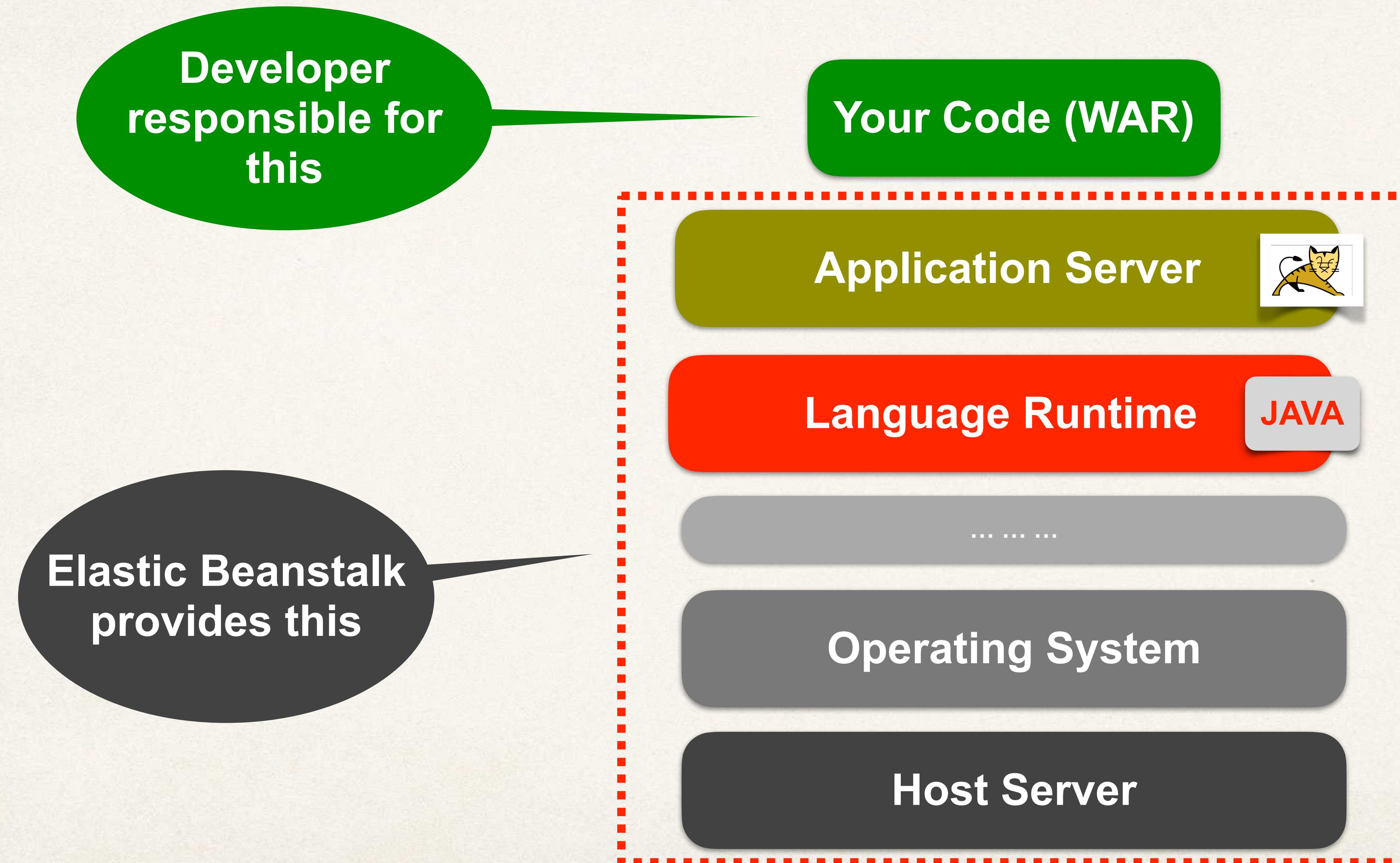
Elastic Cloud Compute (EC2)



Comparing EC2 to Elastic Beanstalk

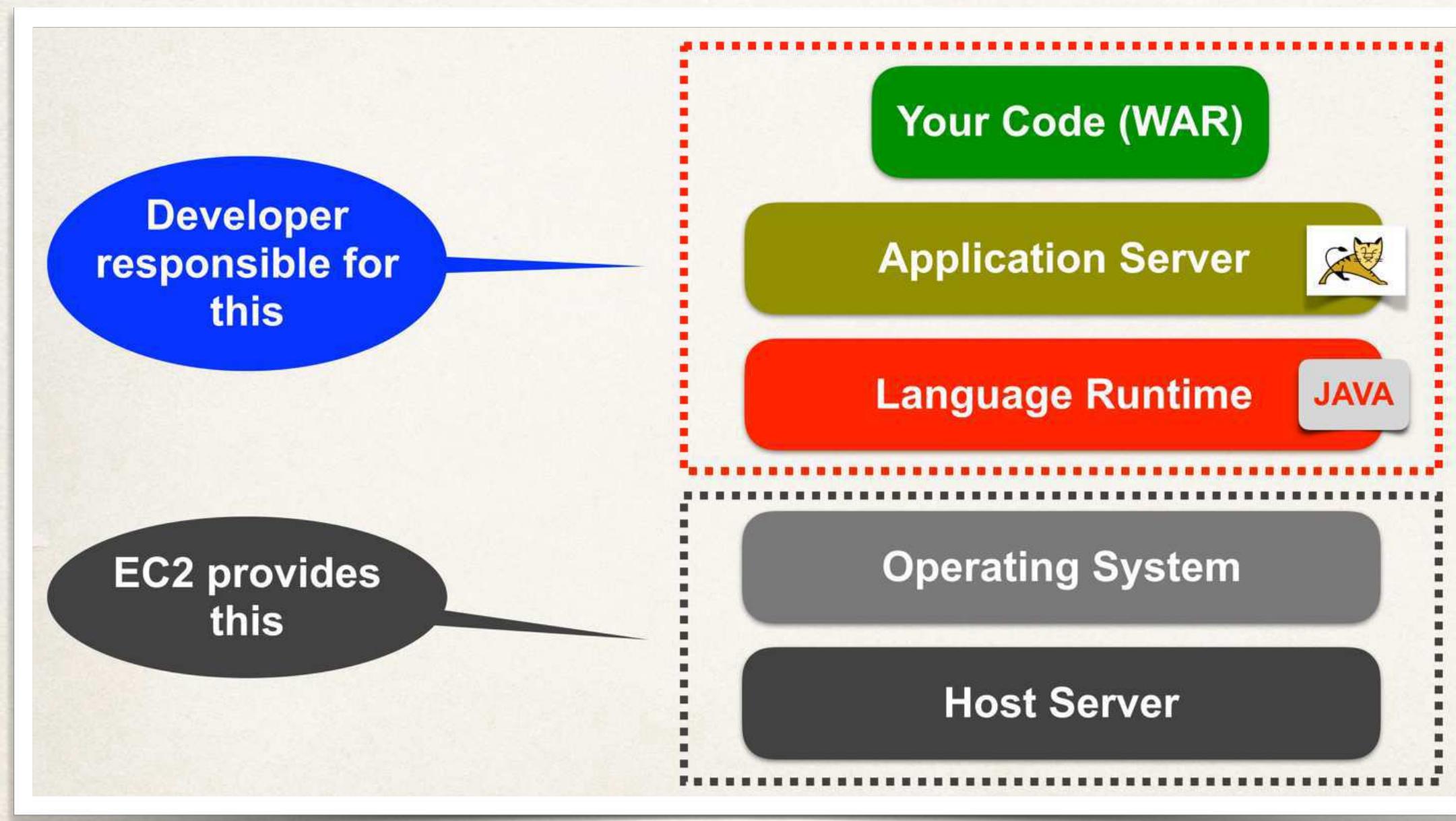
- Elastic Beanstalk is a pre-packaged platform
 - Ideal for deployments on a web stack
 - Simply select the services you need
- Known as Platform-as-a-Service (PaaS)
- All you have to deploy is **Your Code**

Elastic Beanstalk

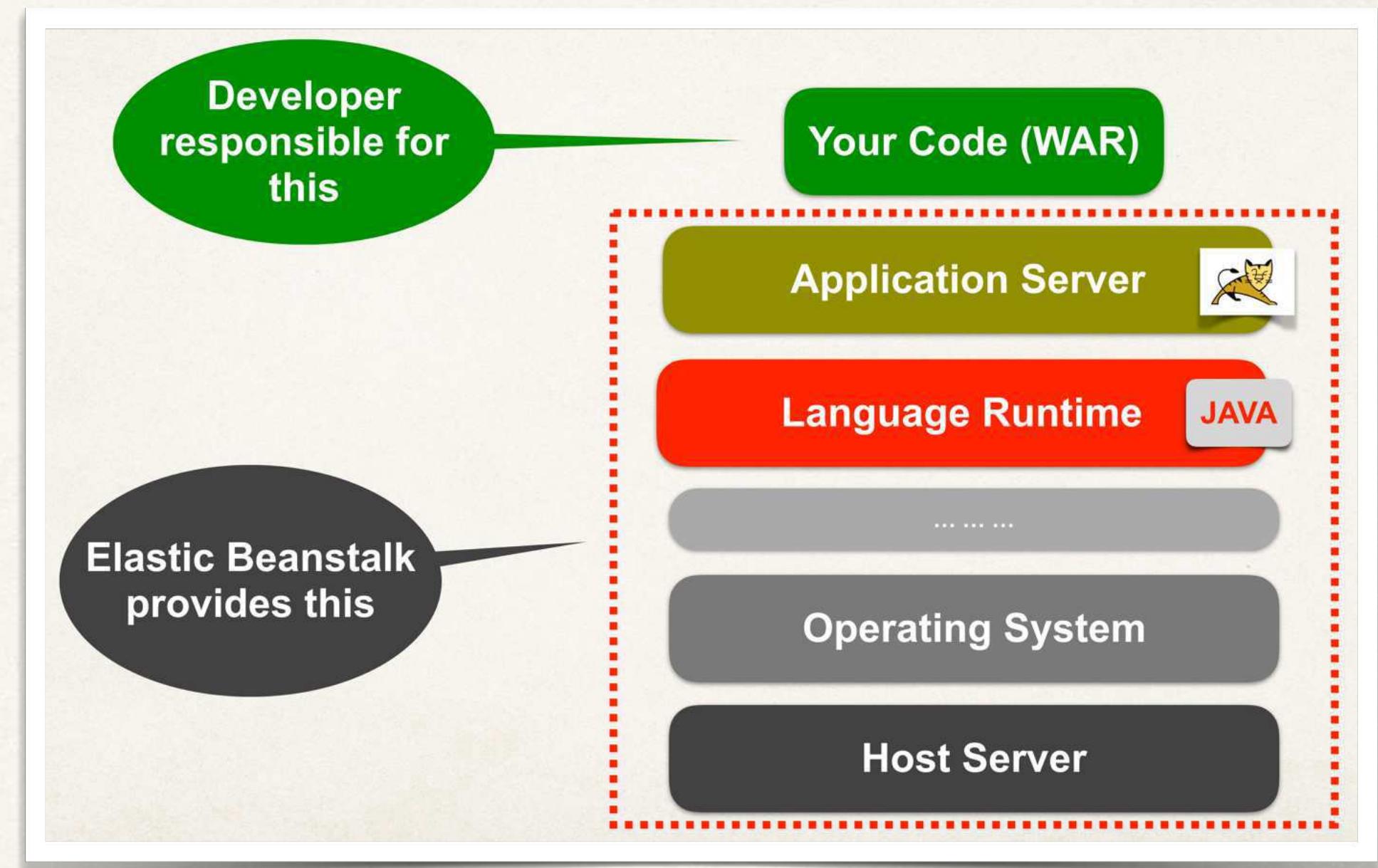


EC2 vs Elastic Beanstalk

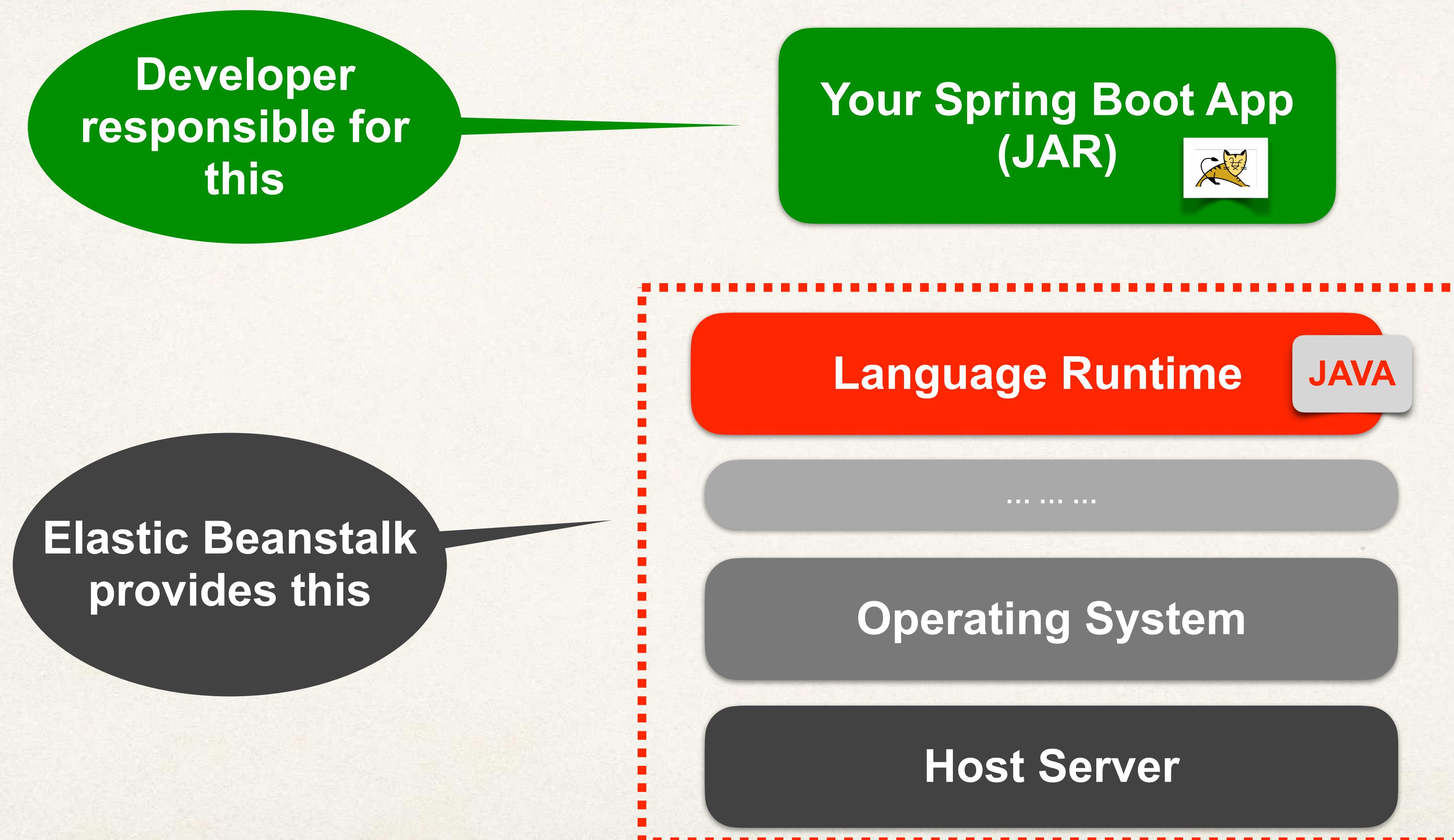
EC2



Elastic Beanstalk



Spring Boot JAR - includes embedded Tomcat



Use Case

- For your apps, start with Elastic Beanstalk



- Quickly get started with deploying your apps
- Leverage pre-configured web stacks

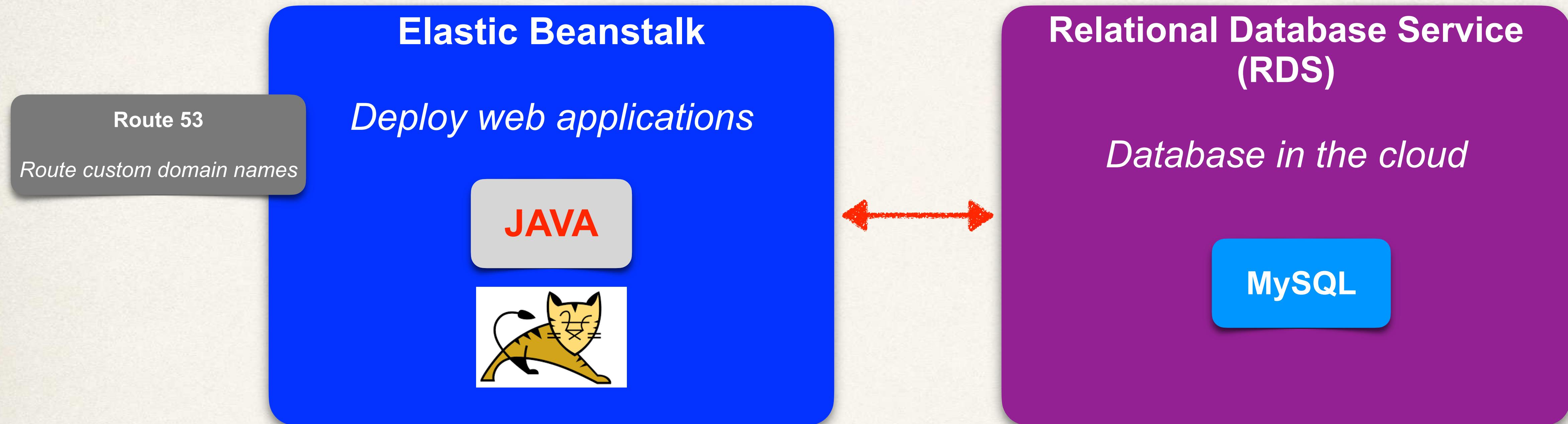
- Use EC2 if you need low-level control



- You want to deploy using a different version of Java not supported by Elastic Beanstalk
- You want to use a different Java app server (WebLogic, WebSphere etc ...)
- Any other OS specific customizations you need to make

Can also create custom
Elastic Beanstalk
“templates”

Our App Architecture



<< your domain name >>



More Cool Features of Elastic Beanstalk

- Capacity provisioning
- Load balancing
- Auto scaling
- Health monitoring, alarms and notifications
- Many more



AWS Documentation

<https://aws.amazon.com/documentation/>

Service	Docs
Elastic Cloud Compute (EC2)	https://aws.amazon.com/documentation/ec2/
Elastic Beanstalk	https://aws.amazon.com/documentation/elastic-beanstalk/
Relational Database Service (RDS)	https://aws.amazon.com/documentation/rds/
Route 53	https://aws.amazon.com/documentation/route53/

Create a Free AWS Account



Deploy HelloWorld Spring Boot App



HelloWorld - Spring Boot

Spring
Boot



Spring Boot - HelloWorld

- As Proof-of-Concept, deploy HelloWorld Spring Boot app on Elastic Beanstalk
- Simple Spring Boot app
- Focus on the deployment process to Elastic Beanstalk

We'll do advanced Spring Boot with
Database CRUD in later videos ...

Development Process

Step-By-Step

1. Download the app and import into Java IDE
2. Package the application using Maven
3. Create a new application in Elastic Beanstalk
4. Upload the JAR file to Elastic Beanstalk

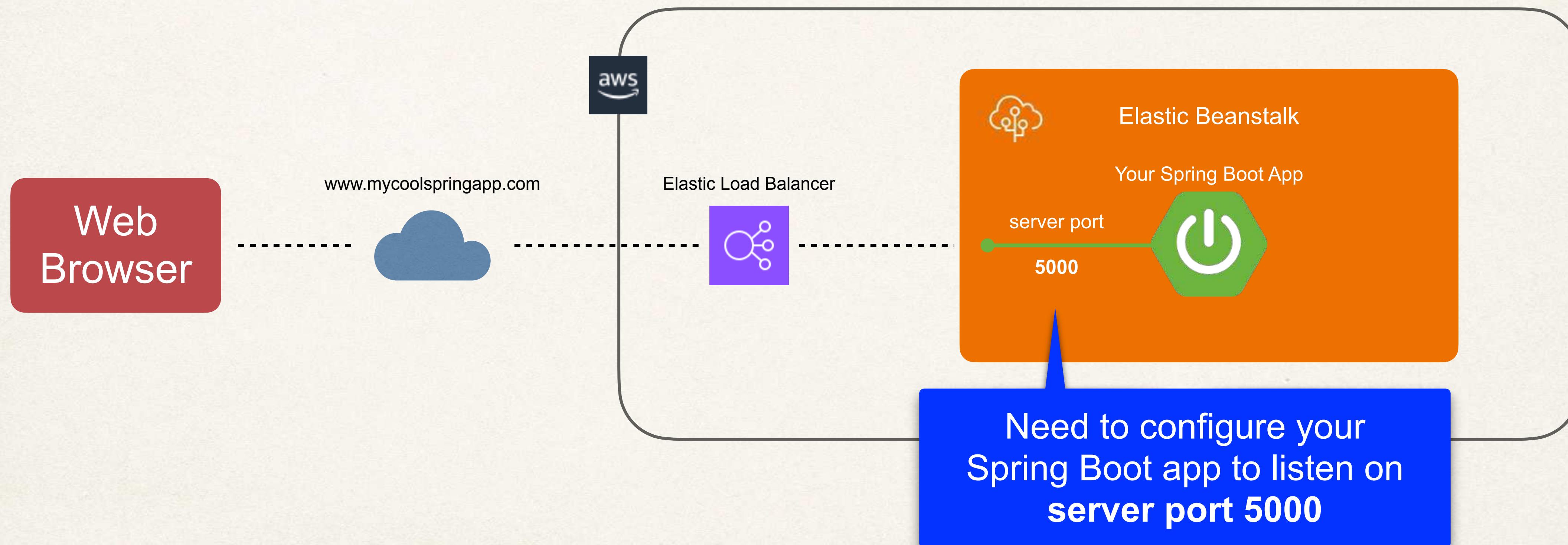
Spring Boot - REST API Deployment

- We'll use MVC app in this course, but we can also deploy REST API
- For REST API ... we can follow the exact same deployment steps
- This works the same for `@RestController` ... just like `@Controller`

Same deployment steps!

Server Port 5000

- Elastic Beanstalk expects for your **Spring Boot** app to listen on **server port 5000**



Server Port 5000

- Elastic Beanstalk expects for your app to listen on server port 5000
- Two options
 - Option 1: Set **environment variable** in Elastic Beanstalk
 - **SERVER_PORT=5000**
- Option 2: Set property in **application.properties**
- **server.port=5000**

We'll cover Option 1 for first deployment: HelloWorld

We'll cover Option 2 for second deployment: CRUD MVC

Can also use Spring profiles

AWS Free Tier and Budgets



AWS Free Tier

- Six months free available to new AWS customers
- Specific limitations on service and instance type

<http://aws.amazon.com/free>

AWS Free Tier

- Six months free does not give you unlimited access
- You can not run unlimited number of apps for free for six months

<http://aws.amazon.com/free>

AWS Free Tier

- Based on free tier limits, you can safely run
 - 1 Spring Boot app on Elastic Beanstalk
 - 1 Database instance on Amazon Relational Database Service (RDS)
- Run this scenario for free for six months - 24/7

AWS Free Tier

- You can experiment with other services
- However, if you are not actively using the services
 - You should stop/delete the services
 - If you don't stop / delete the services
 - You will be charged if usage exceeds free tier limits

Budget Alerts

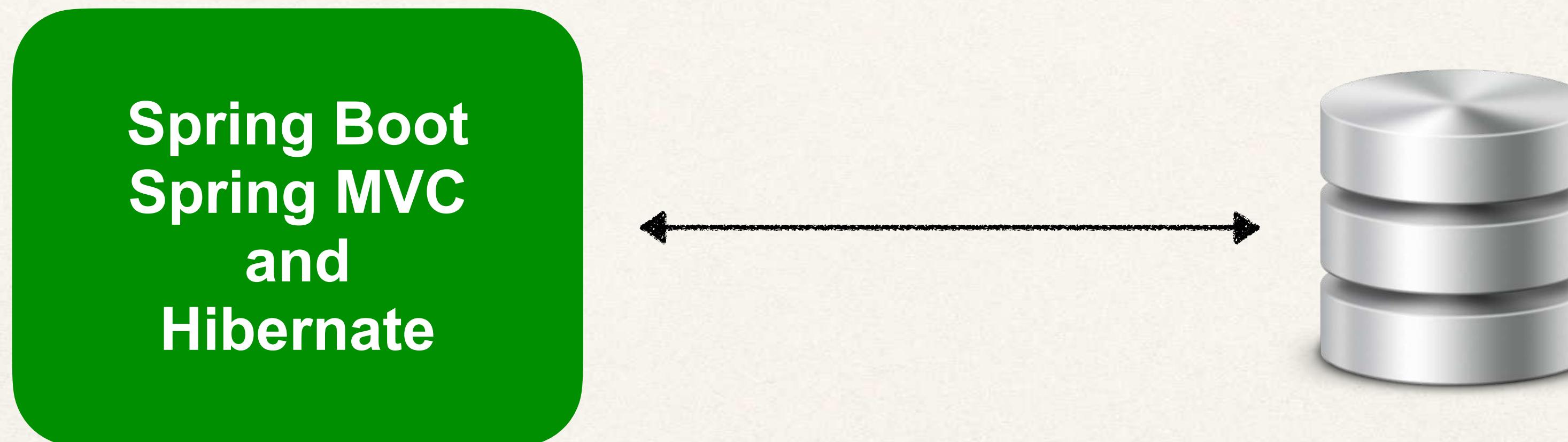
- To help monitor charges, you can set up **Budget Alerts**
- Helps keep you in the Free Tier
- AWS will send an email before you exceed capacity / usage
- Normally sends the alert when you are at 75% capacity / usage
- Gives you a chance to stop the services before charges are incurred

Deploy Spring Boot Spring MVC CRUD App



Spring Boot - Spring MVC CRUD Project

**Spring Boot - Spring MVC and JPA/Hibernate
application that connects to a database**



Employee Directory

- List Employees
- Add a new Employee
- Update an Employee
- Delete an Employee

Employee Directory			
Add Employee			
First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete
Emma	Baumgarten	emma@luv2code.com	Update Delete
Avani	Gupta	avani@luv2code.com	Update Delete
Yuri	Petrov	yuri@luv2code.com	Update Delete
Juan	Vega	juan@luv2code.com	Update Delete

About the code

- All of the coding is complete ... we'll focus on deployment
- We'll download the full source code for this app
- Run the app locally using a local database

Create a database on AWS

- Once we have app running locally, then in following videos
 - Create a database on AWS
 - Configure the app to talk our database deployed in AWS
 - Deploy the app to AWS Elastic Beanstalk
- We'll do that later ... at the moment, let's focus on running the app locally

Two Database Scripts

1. Folder: **sql-scripts**

- **01-create-user.sql**
- **02-employee-directory.sql**

Details on downloading the SQL file
provided in next section

About: 01-create-user.sql

1. Create a new MySQL user for our application

- user id: **springstudent**
- password: **springstudent**

About: 02-employee-directory.sql

1. Create a new database table: **employee**

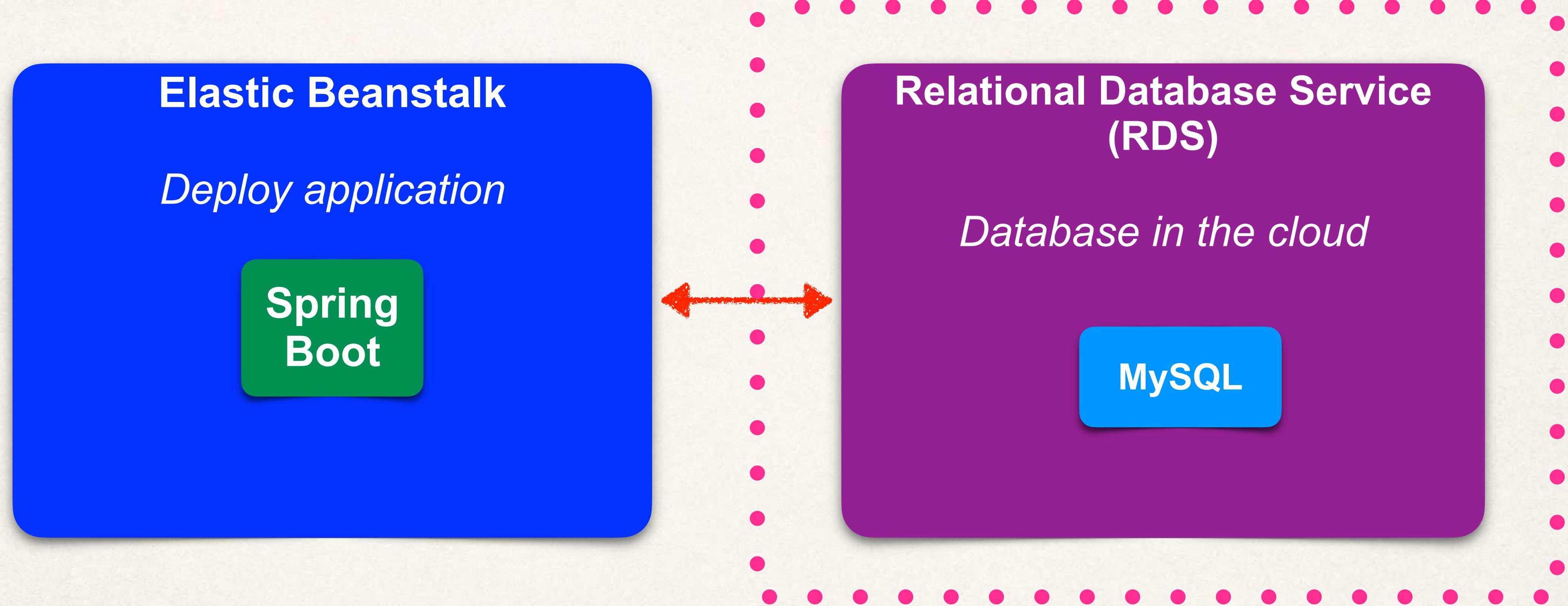
2. Load table with sample data

employee	
!	id INT(11)
◆	first_name VARCHAR(45)
◆	last_name VARCHAR(45)
◆	email VARCHAR(45)
Indexes	

Deploy Relational Database Service (RDS)



Our App Architecture





Relational Database Service (RDS)

- Quickly deploy a relational database in the cloud
- You can manage it using normal admin tool:
 - MySQL Workbench
 - Oracle SQL Developer, etc ...
- AWS also has support for NoSQL databases
 - MongoDB, etc ...

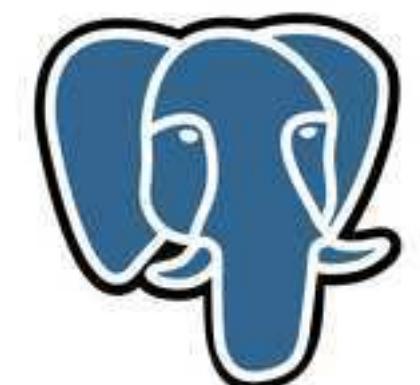


MySQL

Oracle



PostgreSQL



RDS Documentation

<https://aws.amazon.com/documentation/rds>

Our Game Plan

- Deploy MySQL database instance in AWS Cloud
- Once deployed, we will have our database hostname, userid & password
- Connect to our database with MySQL Workbench to verify connectivity



Development Process

1. Select Engine
2. Choose Use Case
3. Specify Database Details
4. Configure Advanced Settings

Step-By-Step

Configure Relational Database Service (RDS)



Current Status - Database in the Cloud

- We created a database server instance ...
- But that's it ... it is empty
- No database schema, no tables ...
- No nothing ... LOL!

**Relational Database Service
(RDS)**

Database in the cloud

MySQL

We need to perform some configuration work

Development Process

1. Configure security for Inbound Connection Rules
2. Test database connection with MySQL Workbench
3. Run SQL Scripts to create table and load sample data

Step-By-Step

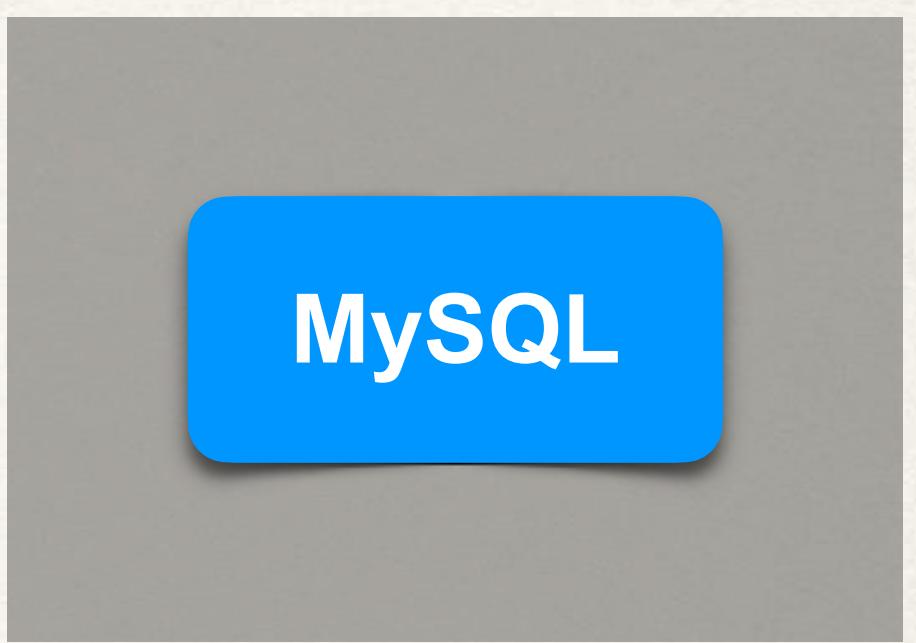
Application Properties with Spring Profiles



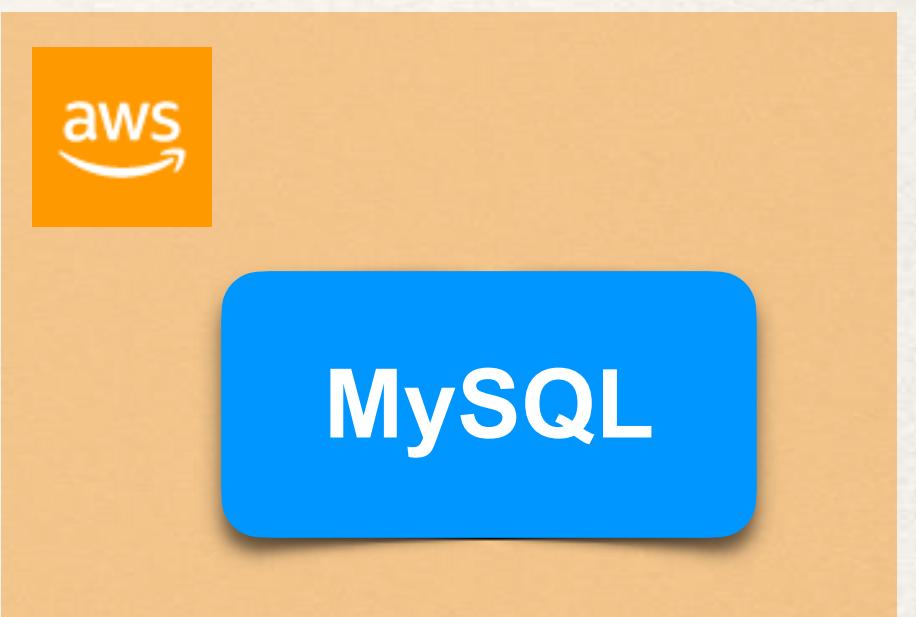
The Challenge - Multiple Environments

- Currently, we have multiple databases running
 - A local MySQL database
 - A remote MySQL database in AWS cloud
- Which one will our application connect to???

Local Computer



AWS Cloud



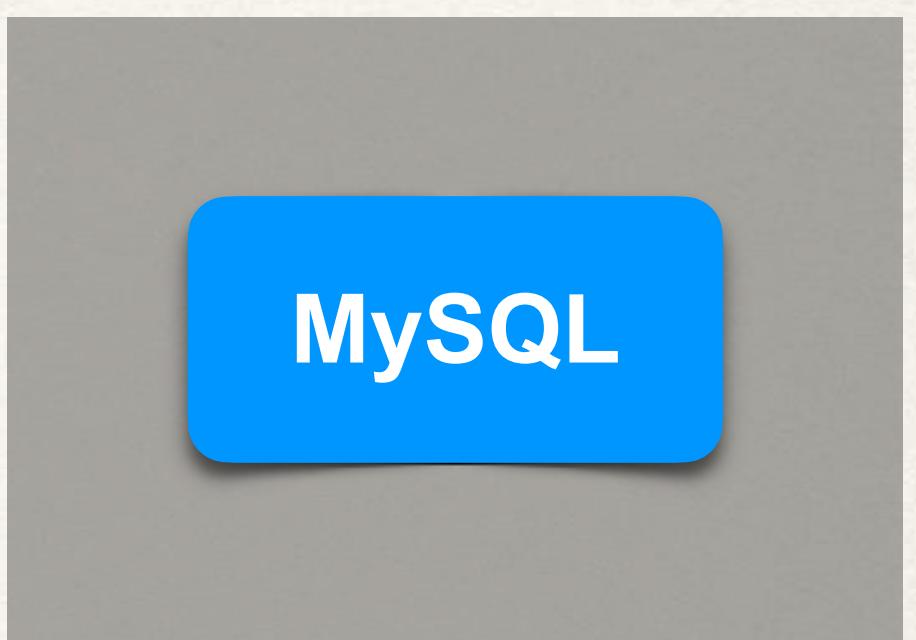
The Challenge - Multiple Environments

- Our database configuration is in our config file

File: application.properties

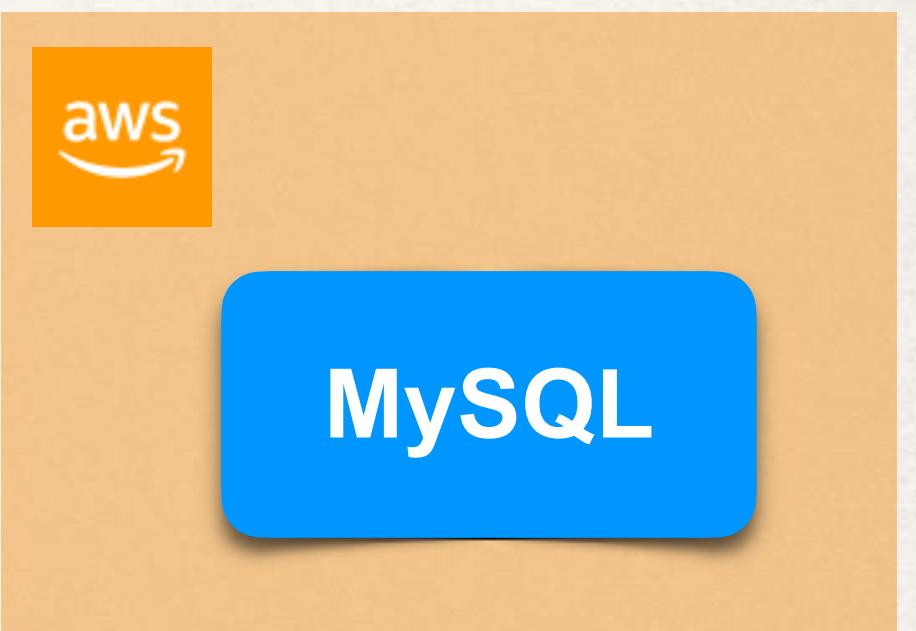
```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory  
spring.datasource.username=springstudent  
spring.datasource.password=springstudent
```

Local Computer



- The AWS database will have different **url, username and password**
- Wouldn't it be great if I could have multiple config files??
- Just tell Spring Boot which one to use???

AWS Cloud



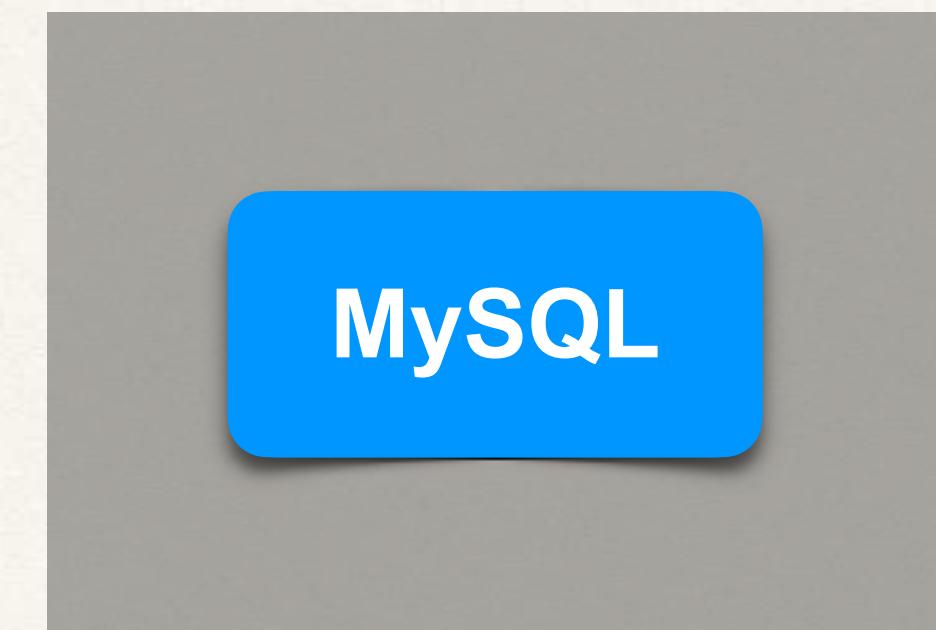
The Solution

- Spring Boot has support for **Profiles**
 - Allows you to have multiple configurations
 - Tell Spring Boot which profile is active

application properties for **dev**

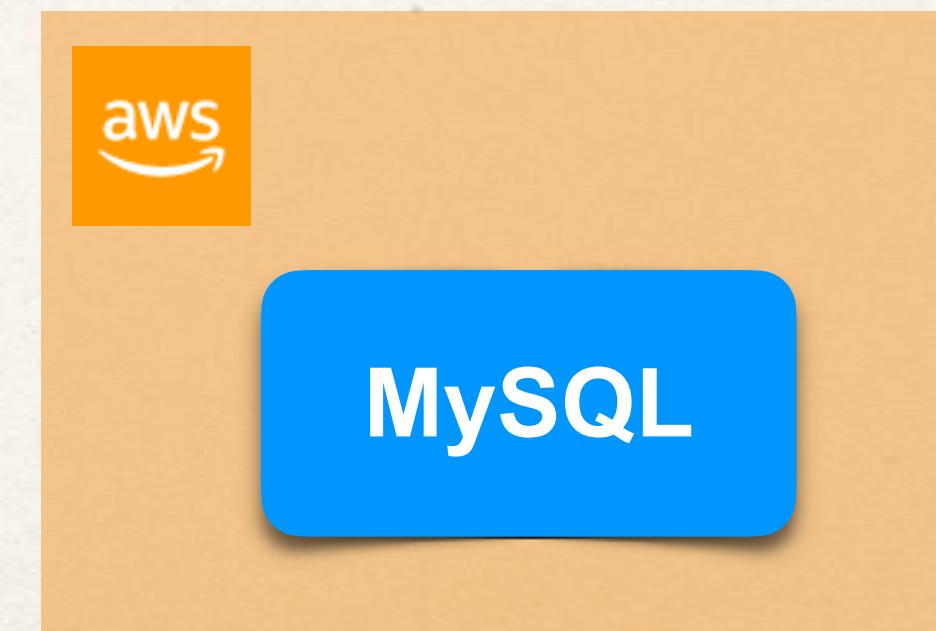
application properties for **prod**

Local Computer



MySQL

AWS Cloud



MySQL

Profiles

- You can have as many profiles as you need
- You can give the profiles any name

application properties for **dev**

application properties for **qa**

application properties for **qa-fun-test**

application properties for **prod**

You can give
any name for
profiles

Profiles - Naming Convention

- **application-<profile name>.properties**

application-dev.properties

You can give
any name for
profiles

application-qa.properties

application-qa-fun-test.properties

YAML files are also
supported

application-prod.properties

Profiles - the default profile

File: application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory  
spring.datasource.username=springstudent  
spring.datasource.password=springstudent
```

Since no profile given,
this is the **default** profile

```
: Starting CruddemoApplication using Java  
  
: No active profile set, falling back to 1 default profile: "default"  
: Bootstrapping Spring Data JPA repositories in DEFAULT mode.  
: Finished Spring Data repository scanning in 5 ms.
```

Profiles - the files

Naming convention
application-<profile name>.properties

File: application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory
spring.datasource.username=springstudent
spring.datasource.password=springstudent
```

Since no profile given,
this is the default profile

File: application-qa.properties

```
spring.datasource.url=jdbc:mysql://testing.example.com:3306/employee_directory
spring.datasource.username=qateam
spring.datasource.password=testsquad
```

Notice, each config file can have
different values based on the
profile / environment

File: application-qa-fun-test.properties

```
spring.datasource.url=jdbc:mysql://190.167.139.61:3306/employee_directory
spring.datasource.username=demo
spring.datasource.password=demo123
```

For example:
spring.datasource.url = ...

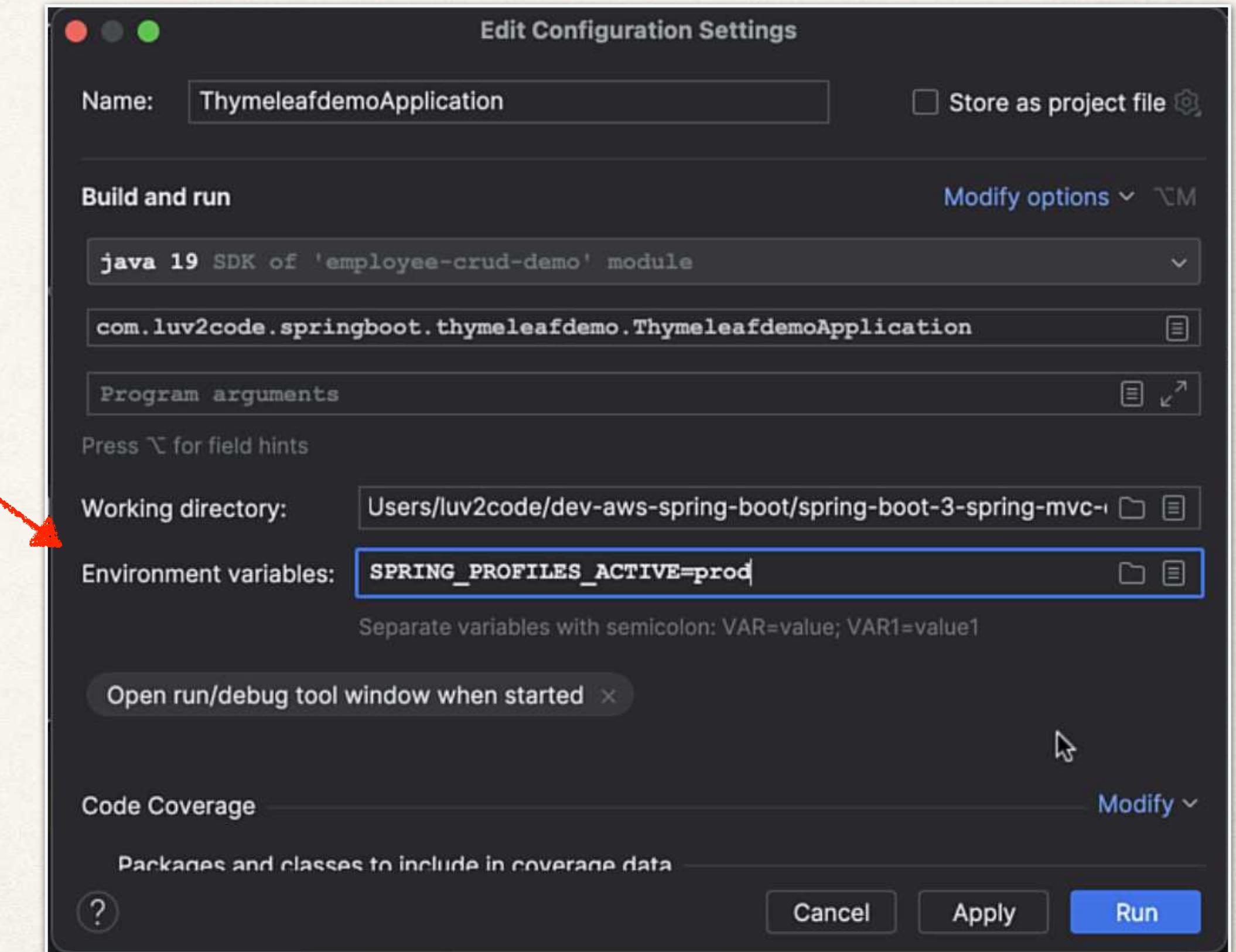
File: application-prod.properties

```
spring.datasource.url=jdbc:mysql://useast2-ohio.aws.amazon.com:3306/employee_directory
spring.datasource.username=springstudent
spring.datasource.password=SuperTopSecret123#
```

Tell Spring Boot which Profile to use

- One option, set an environment variable: **SPRING_PROFILES_ACTIVE**
- IntelliJ Runtime configuration settings

```
SPRING_PROFILES_ACTIVE=prod
```



Tell Spring Boot which Profile to use

- One option, set an environment variable: **SPRING_PROFILES_ACTIVE**
- IntelliJ Runtime configuration settings

SPRING_PROFILES_ACTIVE=prod

File: **application-prod.properties**

```
spring.datasource.url=jdbc:mysql://useast2-ohio.amazonaws.com:3306/employee_directory
spring.datasource.username=springstudent
spring.datasource.password=SuperTopSecret123#
```

Naming convention
application-<profile name>.properties

AWS Cloud



MySQL

Profiles - Advanced Features

- Can have multiple profiles active
 - Define common properties in default profile: app title etc
 - Define environment specific info in profiles: jdbc url etc
- If multiple profiles define **same property**, last one wins
 - Profiles listed as environment variable, left to right (last one wins)

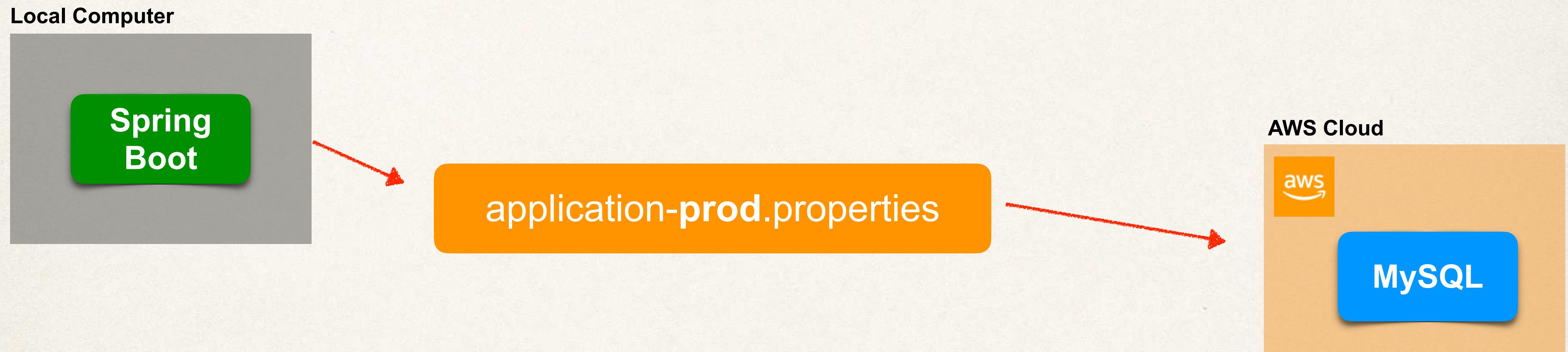
```
SPRING_PROFILES_ACTIVE=dev,qa,prod
```

Profiles - Advanced Features

<http://www.luv2code.com/spring-boot-profiles>

Our Game Plan

- Run our app locally ... and connect to AWS database in the cloud

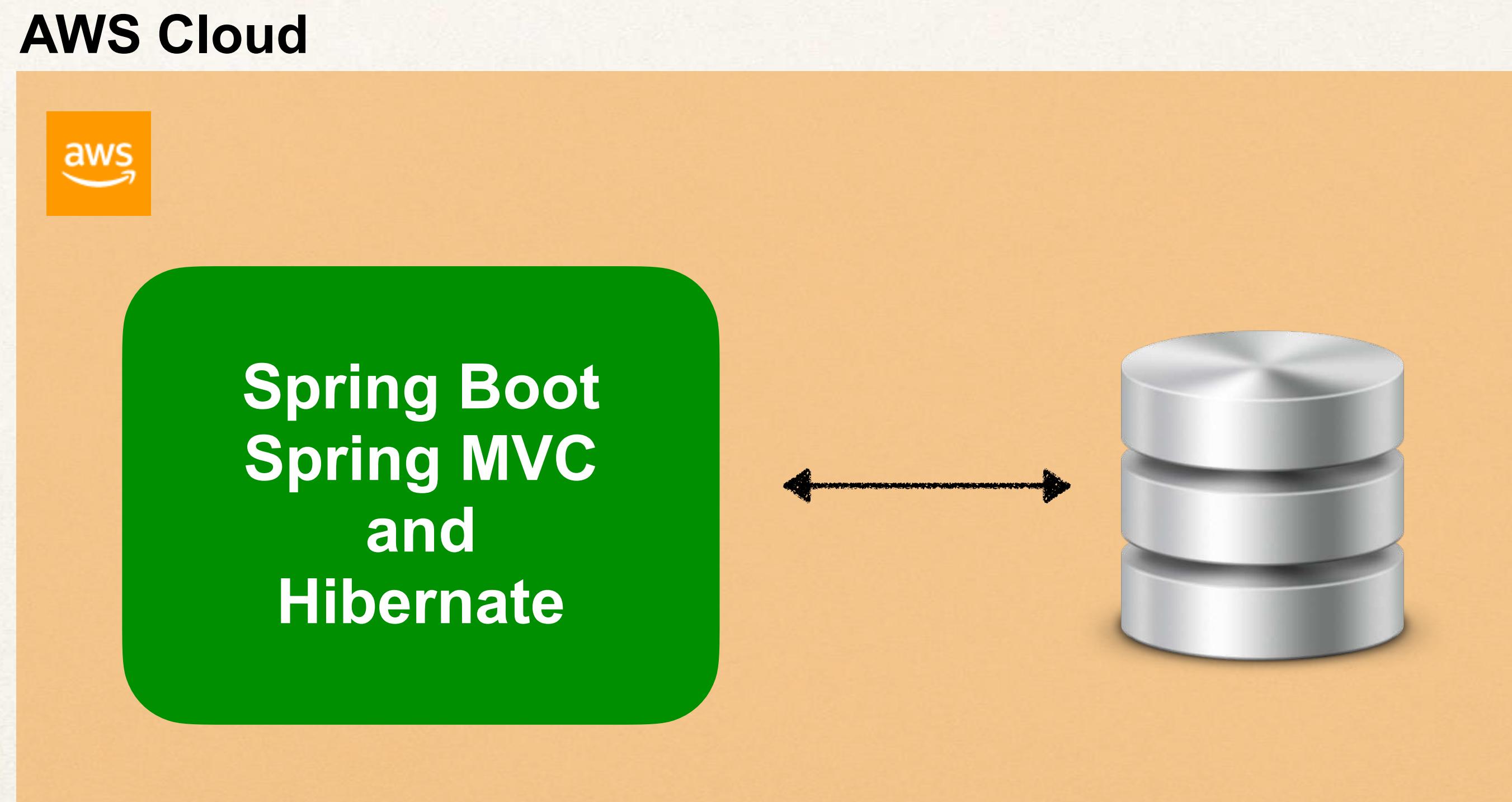


Deploy Spring Boot app on AWS



Application Set Up

- Our database is already deployed in the AWS Cloud
- Now, let's deploy our Spring Boot app in the AWS Cloud



Development Process

1. Package the Spring Boot app using Maven

Step-By-Step

2. Create a new application in Elastic Beanstalk

3. Upload the Spring Boot JAR file to Elastic Beanstalk

4. Set Environment variables in AWS

- **SERVER_PORT=5000**
- **SPRING_PROFILES_ACTIVE=prod**

Use **prod** profile to load
application-**prod**.properties
to connect to AWS database

Environment Properties in AWS



We have a little problem ...

File: application-prod.properties

```
spring.datasource.url=jdbc:mysql://useast2-ohio.amazonaws.com:3306/employee_directory  
spring.datasource.username=springstudent  
spring.datasource.password=SuperTopSecret123#
```

AWS Cloud



MySQL

- We have PRODUCTION database info in our props file :-(
- What if a developer accidentally checked this into the git repo??
- This data is not secure ... not good!!!



The Solution

- Remove database connection info from application-prod.properties
- Use Environment Properties / Environment Variables in AWS

Environment Properties

Defined in AWS

- Spring Boot can read appropriate environment properties

application.properties	Environment Property
spring.datasource.url	→ SPRING_DATASOURCE_URL
spring.datasource.username	→ SPRING_DATASOURCE_USERNAME
spring.datasource.password	→ SPRING_DATASOURCE_PASSWORD

To convert props name: Make all caps and replace “.” With “_”

Environment Properties

Defined in AWS

Environment properties

The following properties are passed in the application as environment properties. [Learn more](#)

Name	Value	Remove
GRADLE_HOME	/usr/local/gradle	Remove
M2	/usr/local/apache-maven/bin	Remove
M2_HOME	/usr/local/apache-maven	Remove
SERVER_PORT	5000	Remove
SPRING_DATASOURCE_URL	jdbc:mysql://useast2-ohio.aws.amazon.c	Remove
SPRING_DATASOURCE_USERNAME	springstudent	Remove
SPRING_DATASOURCE_PASSWORD	SuperTopSecret123#	Remove

Add environment property

Cancel Continue Apply

Database
Connection
Configs

This info is no longer in
application-prod.properties

Environment Properties - Higher Priority

- What if property is defined twice??
 - Property defined in application-<profile>.properties
 - AWS environment property is set
- The environment property has higher priority and will override

Development Process

1. Set Environment properties in AWS

Step-By-Step

SERVER_PORT	5000
SPRING_DATASOURCE_URL	jdbc:mysql://useast2-ohio.aws.amazon.c
SPRING_DATASOURCE_USERNAME	springstudent
SPRING_DATASOURCE_PASSWORD	SuperTopSecret123#

2. Remove the application-prod.properties file from Spring Boot app
3. Package the Spring Boot app using Maven
4. Upload the Spring Boot JAR file to Elastic Beanstalk

Encrypted Environment Properties in AWS



AWS Environment Properties

- The environment properties are NOT encrypted at rest
- Stored as plain text

SERVER_PORT	5000
SPRING_DATASOURCE_URL	jdbc:mysql://useast2-ohio.aws.amazon.c
SPRING_DATASOURCE_USERNAME	springstudent
SPRING_DATASOURCE_PASSWORD	SuperTopSecret123#

Security Requirements

- Database connection info should be
 - Encrypted at rest
 - Stored in an encrypted format
- This applies to database URL, username and password

Solution: AWS Systems Manager Parameter Store

- Provides secure solution for storing configuration data
 - Supports storing data as plain text or encrypted data
 - Adds auditing and security controls for accessing data
 - Centralized solution for managing configuration data
-
- Note: Official service name is **AWS Systems Manager Parameter Store**
 - I'll just call it **AWS Parameter Store** ... most developers use the short name

Free service

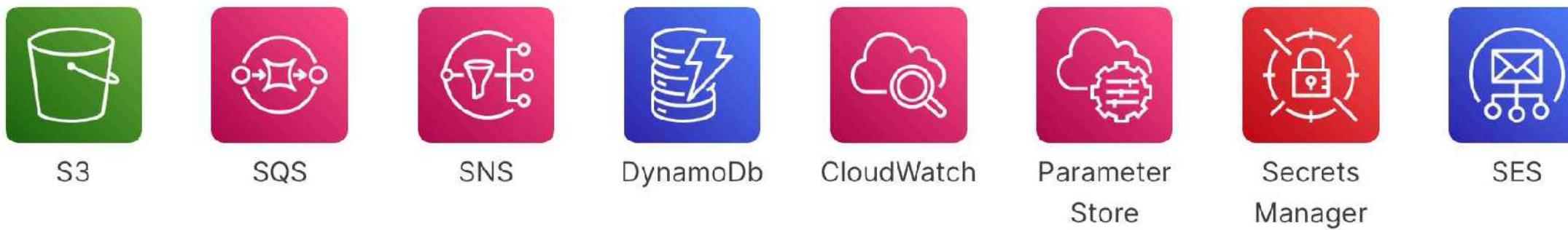
Spring Cloud AWS

Spring Cloud AWS

Spring Boot & Spring Cloud integration with AWS Cloud

[Reference Documentation](#) [View on Github](#)

Spring Cloud AWS simplifies using AWS managed services in a Spring Framework and Spring Boot applications. It offers a convenient way to interact with AWS provided services using well-known Spring idioms and APIs.



S3 SQS SNS DynamoDb CloudWatch Parameter Store Secrets Manager SES

- Open-source project provides integration with AWS Cloud
- Simplifies using AWS services in Spring Boot applications
- We'll use Spring Cloud AWS for easy integration with AWS Parameter Store

<https://awspring.io/>

Development Process

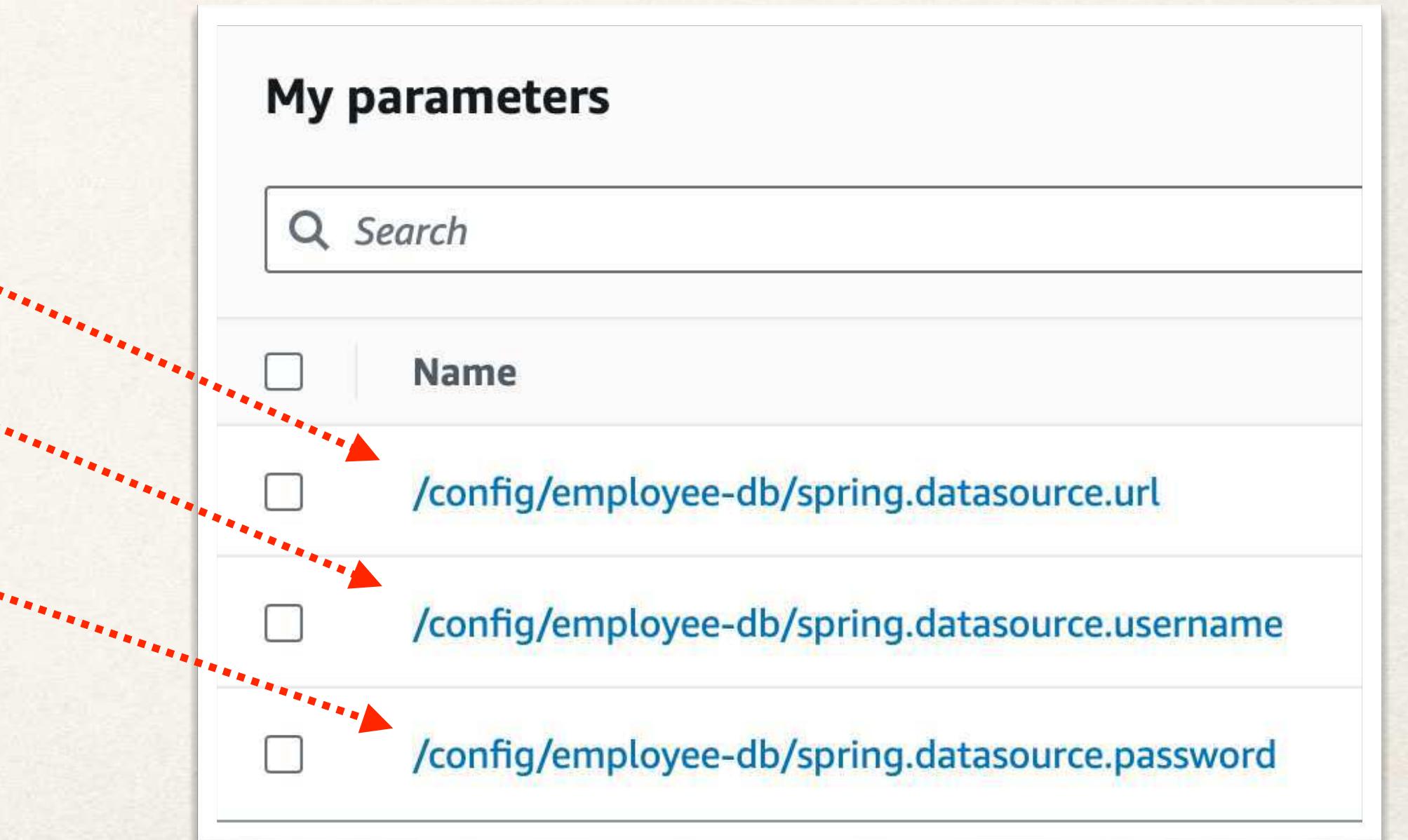
Step-By-Step

1. Create parameters in AWS Parameter Store
2. Attach policy to AWS role: AmazonSSMReadOnlyAccess
3. Edit application-prod.properties file in Spring Boot app
4. Update Maven pom.xml to support Parameter Store
5. Package the Spring Boot app using Maven
6. Update AWS Environment Properties
7. Upload the Spring Boot JAR file to Elastic Beanstalk

Step 1: Create parameters in AWS Parameter Store

- We can define a path to our parameters: /config/employee-db
- Use the standard property names for Spring Boot

- `spring.datasource.url`
- `spring.datasource.username`
- `spring.datasource.password`



Step 1: Create parameters in AWS Parameter Store

We can define any path and property name

Name
When naming a parameter, you can use forward slashes (/) to organize it into a hierarchy. [Learn more about hierarchies](#)

Description — *Optional*

Tier
Parameter Store offers standard and advanced parameters.

Standard
Limit of 10,000 parameters. Parameter value size up to 4 KB. Parameter policies are not available. No additional charge.

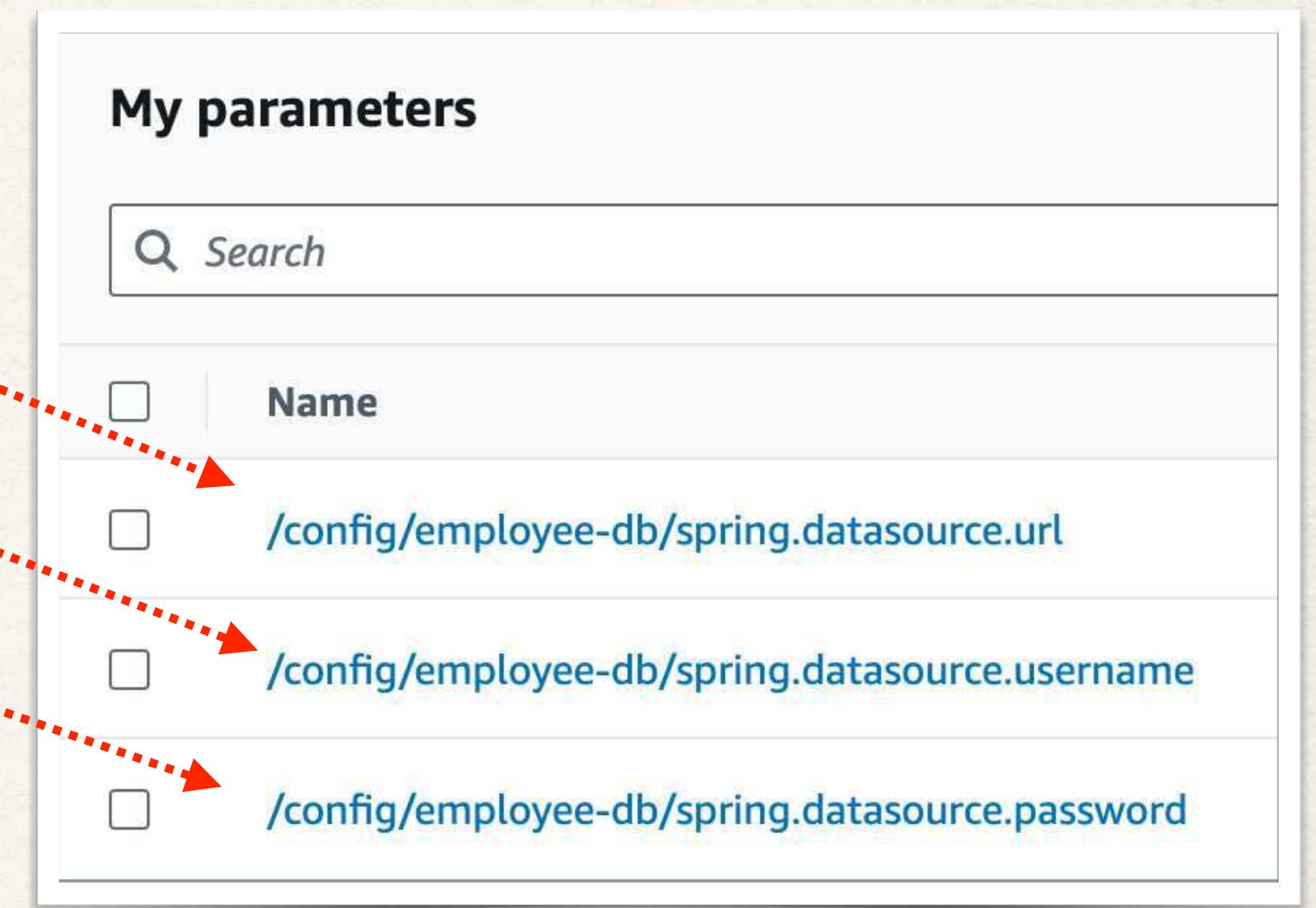
Type
 String
Any string value.
 StringList
Separate strings using commas.
 SecureString
Encrypt sensitive data using KMS keys from your account or another account.

For encrypting the data
Very important!

Step 1: Create parameters in AWS Parameter Store

- Repeat the process for standard property names for Spring Boot

- `spring.datasource.url`
- `spring.datasource.username`
- `spring.datasource.password`



Step 2: Attach policy to AWS role: AmazonSSMReadOnlyAccess

- Allow our app to read config data from AWS Parameter Store

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	 AmazonSSMReadOnlyAccess	AWS managed	Provides read only access to Amazon SSM.

Step 3: Edit application-prod.properties file in Spring Boot app

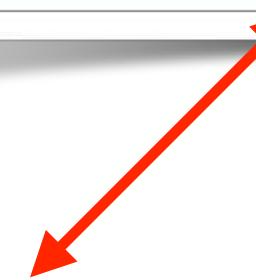
File: application-prod.properties

```
#  
# Load DataSource properties  
# from AWS Parameter Store  
#  
spring.config.import=aws-parameterstore:/config/employee-db
```

For importing additional configuration

Fetch additional configuration from AWS Parameter Store via Spring Cloud AWS libraries

<input type="checkbox"/>	Name
<input type="checkbox"/>	/config/employee-db/spring.datasource.url
<input type="checkbox"/>	/config/employee-db/spring.datasource.username
<input type="checkbox"/>	/config/employee-db/spring.datasource.password



Step 3: Edit application-prod.properties file in Spring Boot app

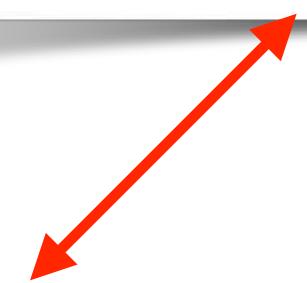
File: application-prod.properties

```
#  
# Load DataSource properties  
# from AWS Parameter Store  
#  
spring.config.import=aws-parameterstore:/config/employee-db
```

For importing additional configuration

Fetch additional configuration from AWS Parameter Store via Spring Cloud AWS libraries

<input type="checkbox"/>	Name
<input type="checkbox"/>	/config/employee-db/spring.datasource.url
<input type="checkbox"/>	/config/employee-db/spring.datasource.username
<input type="checkbox"/>	/config/employee-db/spring.datasource.password



Step 4: Update Maven pom.xml to support Parameter Store

- We will use the Bill of Materials (BOM) for Spring Cloud AWS
 - Defines the recommended versions for each dependency in Spring Cloud AWS
 - Similar concept to Spring Boot Starter Parent

File: pom.xml

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.awspring.cloud</groupId>
      <artifactId>spring-cloud-aws-dependencies</artifactId>
      <version>3.1.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Note: This entry is in
<dependencyManagement> section
...
not regular <dependencies> section

Step 4: Update Maven pom.xml to support Parameter Store

- Add dependency entry for Spring Cloud AWS Starter: Parameter Store
- No need to list version number
 - Since it is defined in Spring Cloud AWS Bill of Materials entry from previous slide

File: pom.xml

Regular <dependencies> section

```
<dependencies>

    <dependency>
        <groupId>io.awspring.cloud</groupId>
        <artifactId>spring-cloud-aws-starter-parameter-store</artifactId>
    </dependency>
    ...
</dependencies>
```

Easy step

5. Package the Spring Boot app using Maven

Step 6: Update AWS Environment Properties

- Set Environment property in AWS
- **SPRING_PROFILES_ACTIVE=prod**

File: application-prod.properties

```
#  
# Load Spring DataSource properties  
# from AWS Parameter Store  
#  
spring.config.import=aws-parameterstore:/config/employee-db
```

Use **prod** profile to load
application-prod.properties
to access
AWS Systems Manager Parameter Store

Easy step

7. Upload the Spring Boot JAR file to Elastic Beanstalk

Options

We'll use the free service

- **AWS Parameter Store**
 - Provides support for encrypted storage of configuration data
 - This service is **free** for storing up to 10,000 parameters / per region / per account
- **AWS Secrets Manager**
 - This is a **paid** service (not eligible for free tier)
 - Rotate secrets automatically based on your compliance policy
 - Replicate secrets to handle disaster recovery and applications spanning multiple regions
 - Monitor and audit the use of secrets
 - *more ...*

Create a Custom Domain Name



Following steps are optional

- We create a custom domain name which requires a purchase (\$13 USD)


At the time of this recording
- This is optional, if you don't want to purchase custom domain name
- Share the **generated** AWS domain name with friends / employers
- **Generated** AWS domain name is free ... no charge

`employee-crud-demo.xxx.us-east-yyy.elasticbeanstalk.com`



`employee-crud-demo.xxx.us-east-yyy.elasticbeanstalk.com`



We want our own
custom domain name ...
not the generated name

<< your domain name >>



myspringbootapp.com



Our own
custom domain name

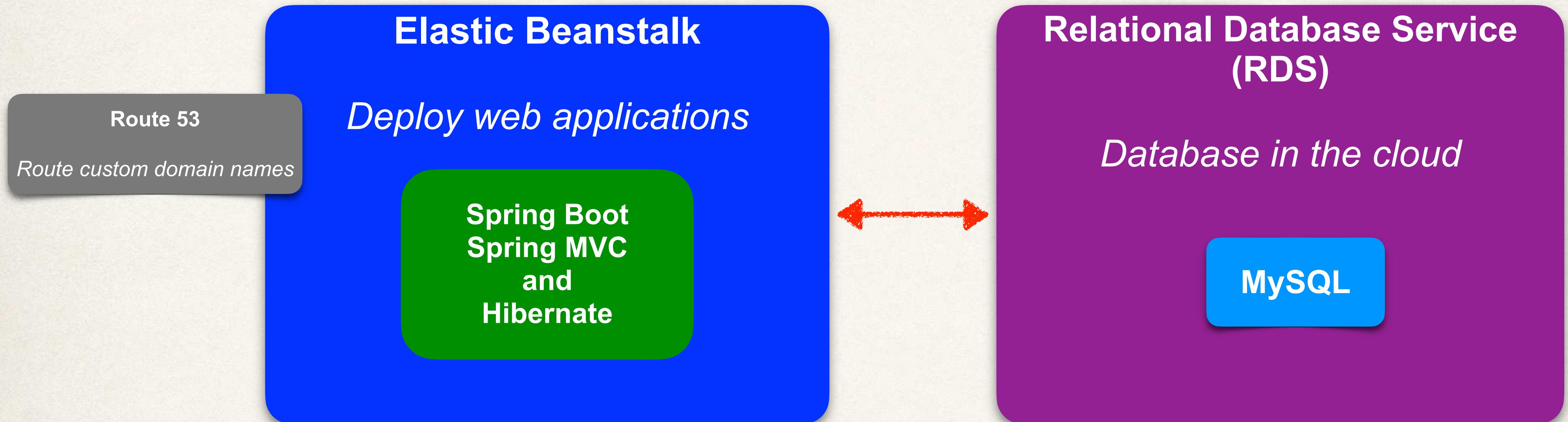
www.myspringbootapp.com

Employee Directory

Add Employee

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	<button>Update</button> <button>Delete</button>
Emma	Baumgarten	emma@luv2code.com	<button>Update</button> <button>Delete</button>
Avani	Gupta	avani@luv2code.com	<button>Update</button> <button>Delete</button>
Yuri	Petrov	yuri@luv2code.com	<button>Update</button> <button>Delete</button>
Juan	Vega	juan@luv2code.com	<button>Update</button> <button>Delete</button>

Our App Architecture



Route 53



- Routes your custom domain name to your app on AWS
- Send www.myspringbootapp.com to your AWS app
- This is the AWS Domain Name System (DNS)
- Trivia: Why the number 53? It is the default port # for name servers :-)

User enters this in web browser

www.myspringbootapp.com

53

Route 53 has an alias for our
Elastic Beanstalk app

employee-crud-demo.xxx.us-east-yyy.elasticbeanstalk.com

Employee Directory			
Add Employee			
First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	<button>Update</button> <button>Delete</button>
Emma	Baumgarten	emma@luv2code.com	<button>Update</button> <button>Delete</button>
Avani	Gupta	avani@luv2code.com	<button>Update</button> <button>Delete</button>
Yuri	Petrov	yuri@luv2code.com	<button>Update</button> <button>Delete</button>
Juan	Vega	juan@luv2code.com	<button>Update</button> <button>Delete</button>

Route 53 Documentation

<https://aws.amazon.com/documentation/route53>

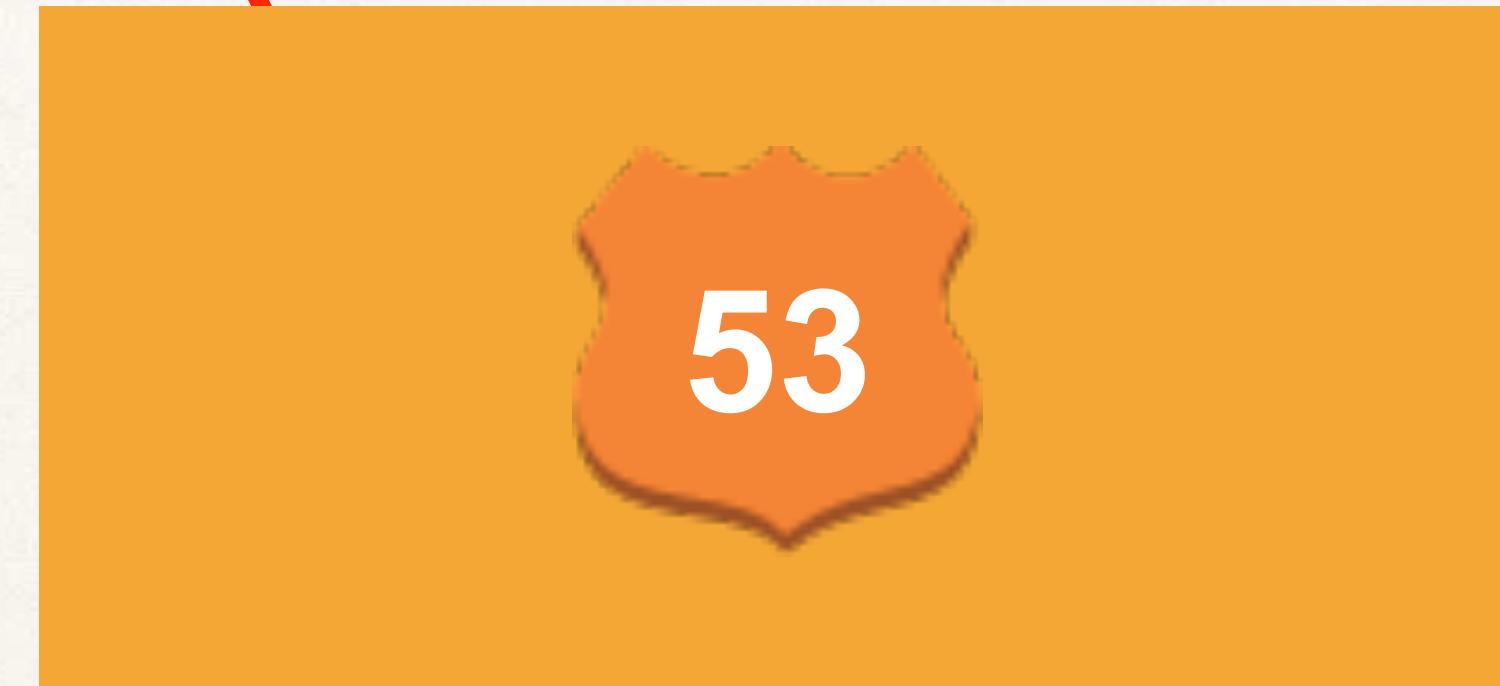
Development Process

Step-By-Step

1. Register your domain name on AWS - \$13 USD annually
2. AWS will perform the registration ... can take up to 3 days
3. Configure domain name to point to your Elastic Beanstalk app

myspringbootapp.com

www.myspringbootapp.com



employee-crud-demo.*.us-east-***.elasticbeanstalk.com**

An alias to redirect to our
www. domain

Employee Directory

Employee Directory			
Add Employee			
First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	<button>Update</button> <button>Delete</button>
Emma	Baumgarten	emma@luv2code.com	<button>Update</button> <button>Delete</button>
Avani	Gupta	avani@luv2code.com	<button>Update</button> <button>Delete</button>
Yuri	Petrov	yuri@luv2code.com	<button>Update</button> <button>Delete</button>
Juan	Vega	juan@luv2code.com	<button>Update</button> <button>Delete</button>

Course Summary



Accomplishments

- Deployed Spring Boot app on AWS Cloud
- Added database support on AWS Cloud
- Registered a custom domain name



Mission
Accomplished



www.myspringbootapp.com

Share Your Accomplishment

- Download your course certificate and share it on social media!

Shivakumar + Follow

Hello Connections,
I am excited to share that I have successfully completed the Spring Boot 3, Spring 6 & Hibernate Course on Udemy!!

This was an extensive course with 396 lectures covering all parts of the JPA / Hibernate framework and Spring Framework covering Spring Boot.

It included Spring Boot fundamentals, Spring Core Concepts, Spring Data, Spring MVC, Spring REST, Spring Security, JPA/ Hibernate basic & advanced concepts and AOP (Aspect Oriented Programming).

Sincere thanks to the instructor [Chad Darby](#) for providing a fastidious and methodical explanation of the concepts along with their application in real projects and making the course intriguing.

I have learned a lot from this course !!

udemy

CERTIFICATE OF COMPLETION

[NEW] Spring Boot 3, Spring 6 & Hibernate for Beginners

Instructors [Chad Darby](#)

Ahmed

I'm happy to share that I've obtained a new certification: Spring Boot 3, Spring 6 & Hibernate from [Udemy](#)!

This course was a game-changer! My knowledge of Spring Boot, Spring MVC, Spring Security, Hibernate, Spring Data JPA, and AOP soared to new heights thanks to the incredible content and engaging instruction.

Finally i have to thank [Chad Darby](#) for the interesting course.



Celebrating a New Certification

Mayank

🚀 Exciting News! 🚀
I'm thrilled to share that I've successfully completed the "Spring Boot 3: Learn Spring 6, Spring Core, Spring REST, Spring MVC, Spring Security, Thymeleaf, JPA, Hibernate, MySQL" course by the amazing [Chad Darby](#) ! 🌟

Link - <https://lnkd.in/dVfrsfCK>

- 🔍 Explored the latest in Spring Boot 3
- 🛠 Mastered Spring Core & REST
- 🌐 Developed with Spring MVC & Security
- 🎨 Explored Thymeleaf for dynamic web pages
- ⌚ Dived deep into JPA & Hibernate
- 📊 Leveraged MySQL for database interactions

Grateful for the comprehensive insights and hands-on experience gained. Ready to apply these skills in real-world projects! 🚀💡

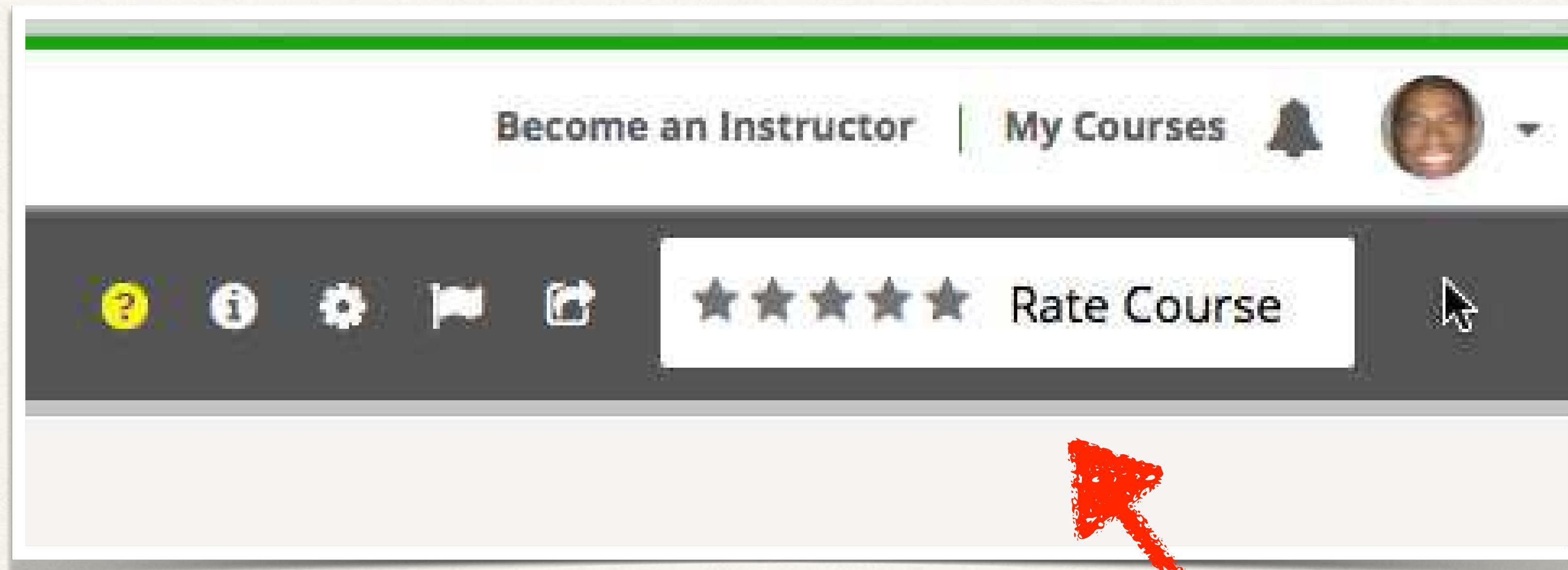
udemy

CERTIFICATE OF COMPLETION

[NEW] Spring Boot 3, Spring 6 & Hibernate for Beginners

Instructors [Chad Darby](#)

Please Rate the Course





Contact Me

darby@luv2code.com

www.luv2code.com