Virtualization ,Cloud Computing & Docker

- **Virtualization** is generally accomplished by dividing a single piece of hardware into two or more ‘segments.’

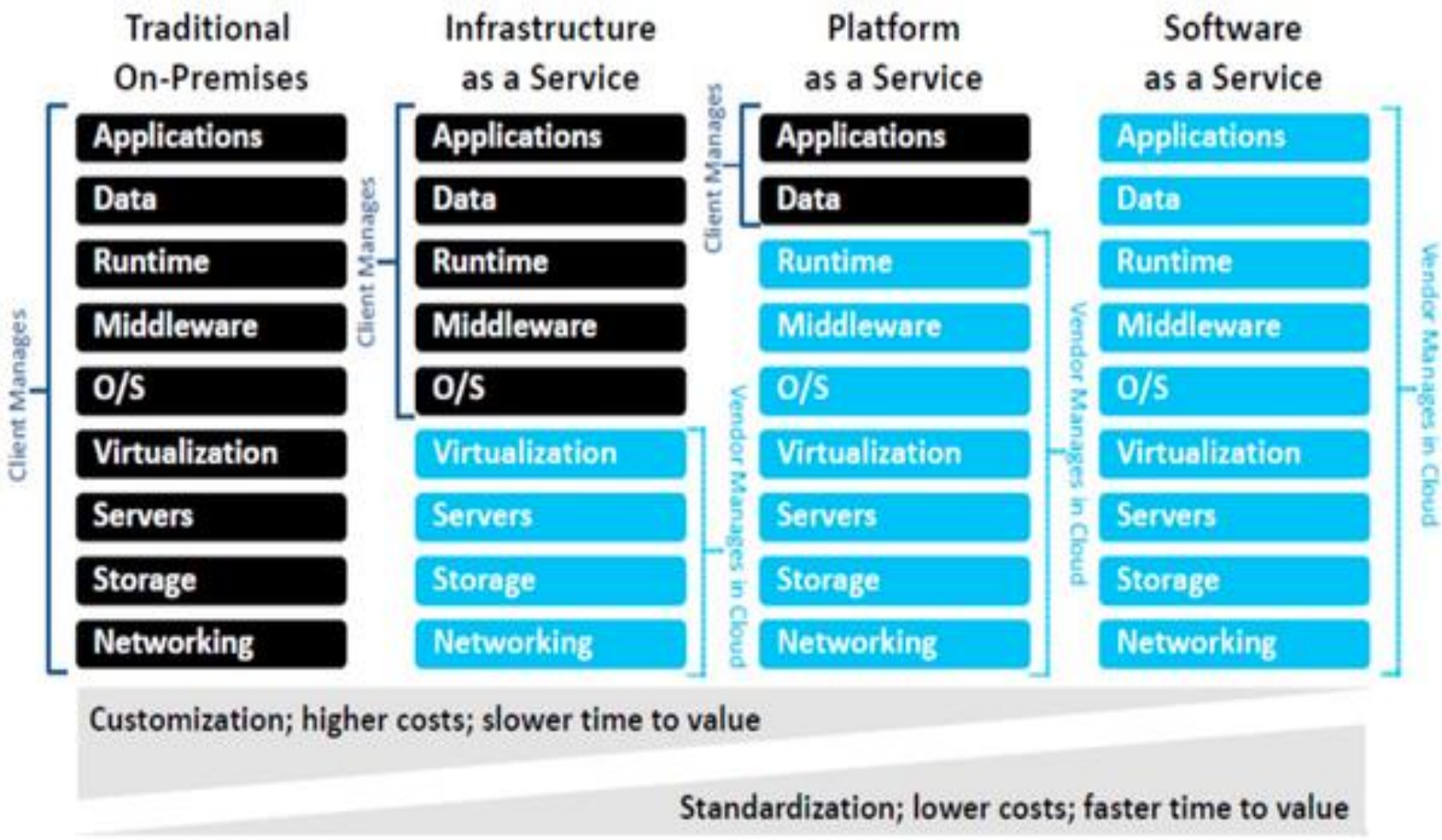
- Each segment operates as its own independent environment.

- For example, server virtualization partitions a single server into a number of smaller virtual servers.

- Essentially, virtualization serves to make computing environments independent of physical infrastructure.

CLOUD COMPUTING

It is shared computing resources, software, or data are delivered as a service and on-demand through the Internet.


Traditional On-Premises
Infrastructure as a Service
Platform as a Service
Software as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Client Manages
Vendor Manages in Cloud
Customization; higher costs; slower time to value
Standardization; lower costs; faster time to value

Virtualization vs Cloud Computing

- Virtualization is a technology
- Cloud computing is a service
- Virtualization can exist without the cloud, but cloud computing cannot exist without virtualization

Cloud Data Centre is just a normal data centre that has been dedicated to co-locating and managing the kit (servers + storage) to provide Cloud services.

So Microsofts Azure Data Centres are 'Cloud Data Centres', Pivotal Cloud Foundry and the same for AWS and so on.

Virtual Data Centre: A pool of virtualised resources made available to a single customer. More than a virtual server, the pool is available - and the customer may scale up and down their requirements within that capacity.

In reality, this is a marketing term used by some hosting / cloud providers, any Cloud service will give you flexibility on resource deployment and scaling.

Cloud computing infrastructure includes the following components:

Servers - physical servers provide "host" machines for multiple virtual machines (VMs) or "guests'

Virtualization - virtualization technologies abstract physical elements and location. IT resources – servers, applications, desktops, storage, and networking – are uncoupled from physical devices and presented as logical resources.

Storage - SAN, network attached storage (NAS), and unified systems provide storage for primary block and file data, data archiving, backup, and business continuance.

Network - switches interconnect physical servers and storage.

Management - cloud infrastructure management includes server, network, and storage orchestration, configuration management, performance monitoring, storage resource management, and usage metering

Security - components ensure information security and data integrity, fulfill compliance and confidentiality needs, manage risk, and provide governance.

Backup & recovery - virtual servers, NAS(network attached storage), and virtual desktops are backed up automatically.

Infrastructure systems - pre-integrated software and hardware, such as complete backup systems with de-duplication and pre-racked platforms containing servers, hypervisor, network, and storage, streamline cloud infrastructure deployment and further reduce complexity.

Docker vs VM

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Docker is a bit like a virtual machine.

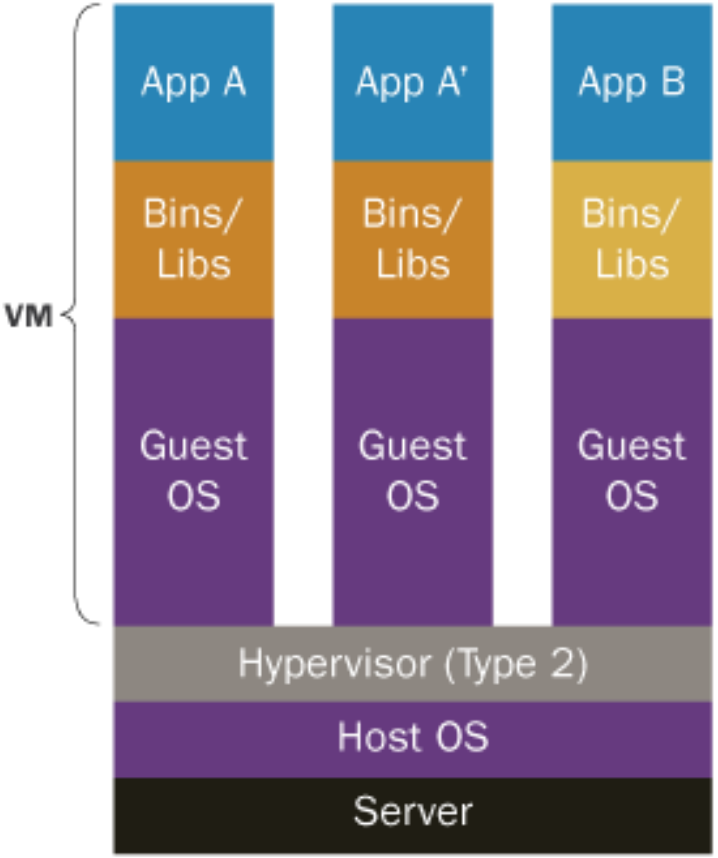
But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer.

This gives a significant performance boost and reduces the size of the application.

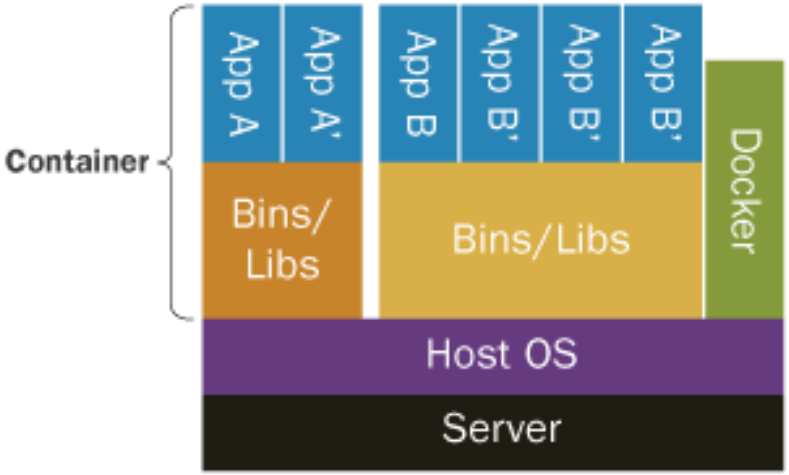
Docker is not a virtualization technology, it’s an application delivery technology

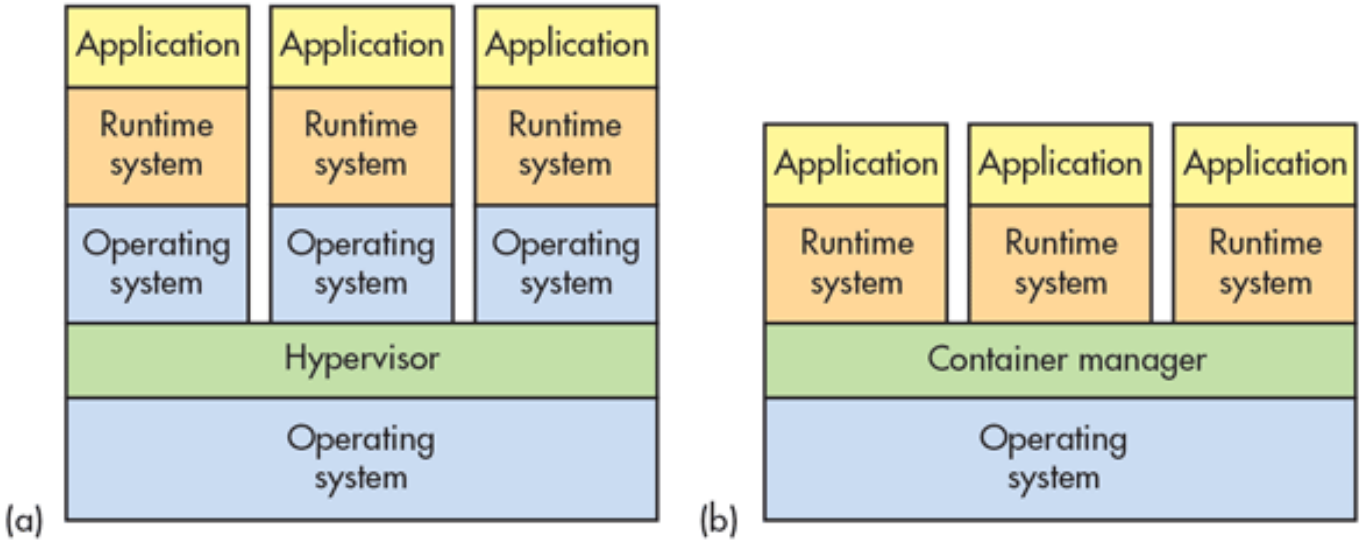
PCF uses Garden container. Docker is another option

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries





Virtual machines (VM) are managed by a hypervisor and utilize VM hardware

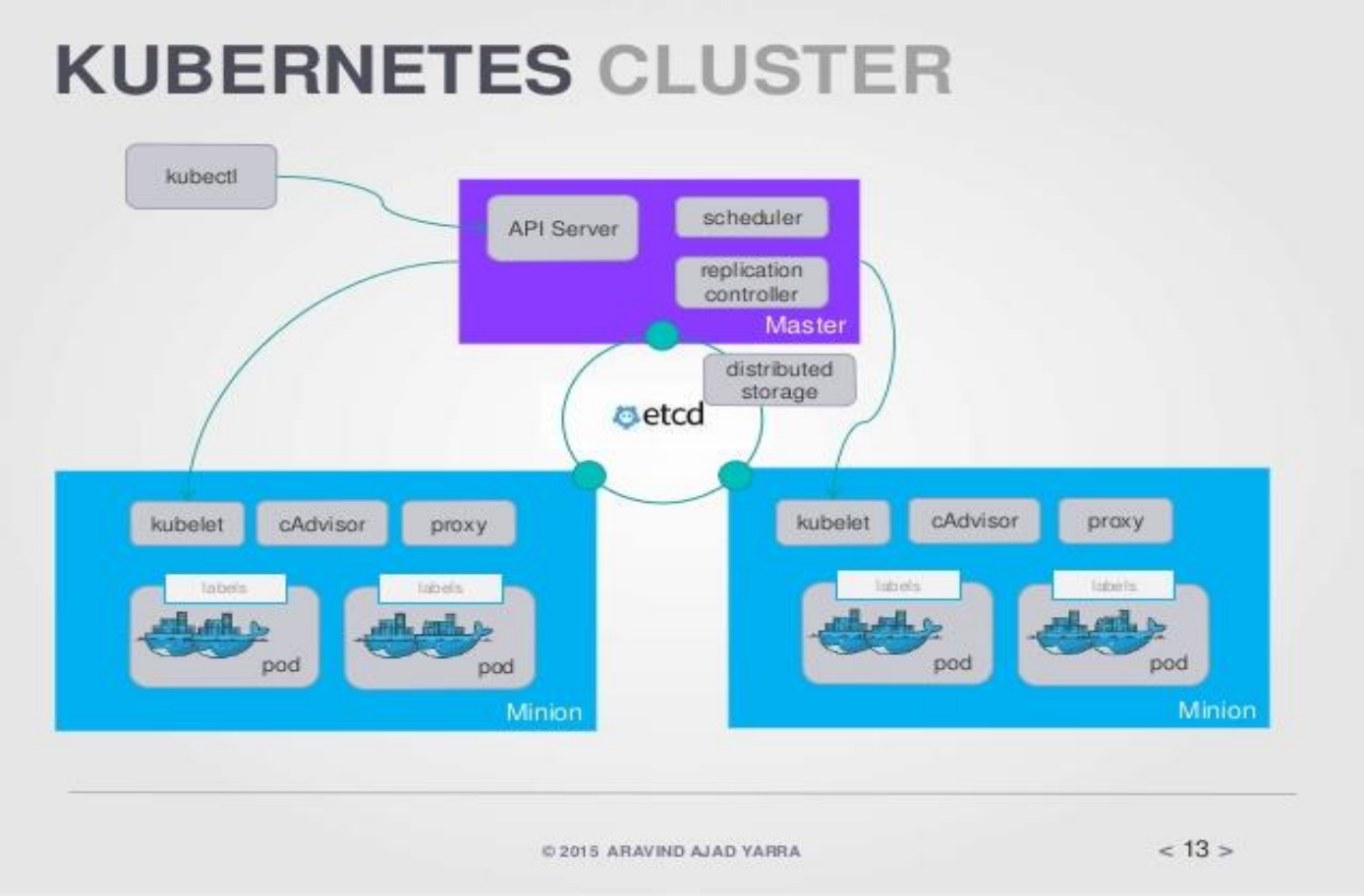
Container systems provide operating system services from the underlying host and isolate the applications using virtual-memory hardware.

Docker primarily focuses on automating the deployment of applications inside application containers.

Application containers are designed to package and run a single service.

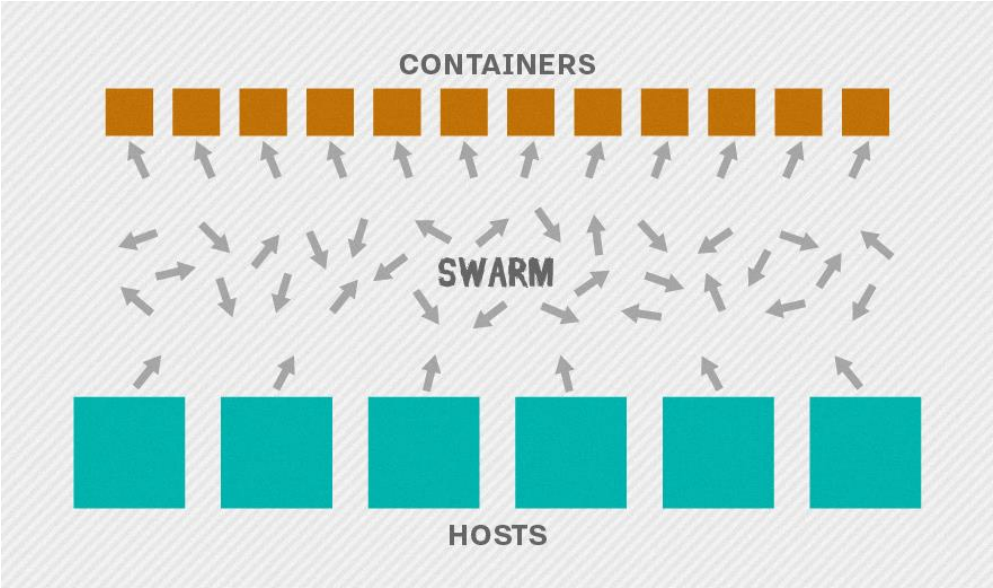
System containers are designed to run multiple processes, like virtual machines.

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.



Docker Swarm is a clustering and scheduling tool for Docker containers. With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system

Swarm mode also exists natively for Docker Engine, the layer between the OS and container images. Swarm mode **integrates** the orchestration capabilities of Docker Swarm into **Docker Engine 1.12** and newer releases.



Pods

In Kubernetes pod is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed.

Amazon EC2

What Is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud.

Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

We can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

Amazon EC2 enables us to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Features of Amazon EC2

Amazon EC2 provides the following features:

Virtual computing environments, known as instances

Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits we need for our server (including the operating system and additional software)

Various configurations of CPU, memory, storage, and networking capacity for our instances, known as instance types

Secure login information for our instances using key pairs (AWS stores the public key, and we store the private key in a secure place)

Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes

Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones

A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups

Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses

Metadata, known as tags, that you can create and assign to your Amazon EC2 resources

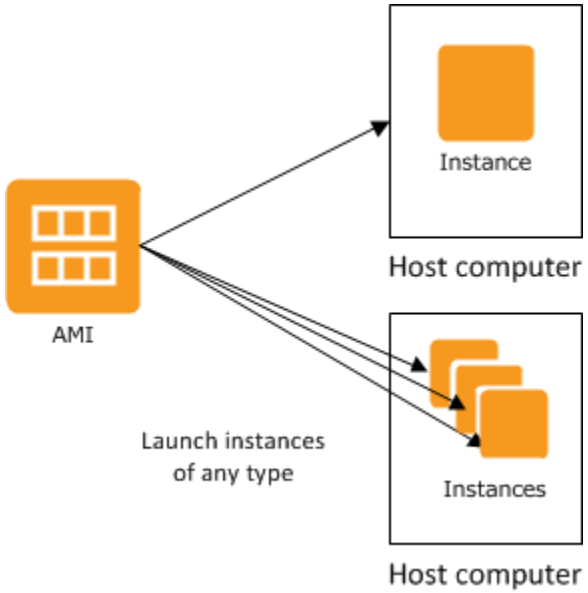
Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

Instances and AMIs

An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications).

From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud.

We can launch multiple instances of an AMI, as shown in the following figure.



Amazon S3 Buckets

Amazon S3 is cloud storage for the Internet.

To upload your data (photos, videos, documents etc.), we first create a bucket in one of the AWS Regions.

We can then upload any number of objects to the bucket.

Creating a Bucket

Amazon S3 provides APIs for us to create and manage buckets. By default, we can create up to 100 buckets in each of our AWS accounts.

Within each bucket, we can store any number of objects.

We can create a bucket using any of the following methods:

- ❑ Create the bucket using the console.
- ❑ Create the bucket programmatically using the AWS SDKs.

Spring Boot & Amazon Web Services (EC2, RDS & S3)

EC2 - Amazons Elastic Cloud Compute provides on demand virtual server instances that can be quickly provisioned with the operating system and software stack of your choice. We'll be using Amazons own Linux machine image to deploy our application.

Relational Database Service - Amazons database as a service allows developers to provision Amazon managed database instances in the cloud. A number of common database platforms are supported but we'll be using a MySQL instance.

S3 Storage - Amazons Simple Storage Service provides simple key value data storage which we'll be using to store image files.

S3 Access :

We can uploaded images to S3 storage.

AWS provides an SDK that makes it easy to integrate with S3, so all we need to do is write a simple Service that uses that SDK to save and retrieve files.

```
@Service
public class FileArchiveService {

   @Autowired
   private AmazonS3Client s3Client;
```

```
private static final String S3_BUCKET_NAME = “AWS-S3 Demo";

s3Client.putObject(new PutObjectRequest(S3_BUCKET_NAME, key,
fileToUpload));
```

XML Resource Configuration for AWS

In order to access protected resources using Amazons SDK an access key and a secret key must be supplied

```
<beans>

  <aws-context:context-credentials>
    <aws-context:simple-credentials access-key="${accessKey:}" secret-key="${secretKey:}"/>
  </aws-context:context-credentials>

  <aws-context:context-resource-loader/>

</beans>
```

AWS Elastic Beanstalk makes it even easier for developers to quickly deploy and manage applications in the AWS Cloud.

Developers simply upload their application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

EC2 vs Elastic Beanstalk

EC2 by itself is not PAAS. It is more like IAAS (Infrastructure as a Service). We still have to take care of the server instances, install software on them, keep them updated, etc.

Elastic Beanstalk is a PAAS system. So are App Engine and Azure among many others.

Elastic Beanstalk does more than just load balancing, monitoring, and autoscaling.

1) Manages application versions by storing and managing different versions of our application, allowing us to easily switch back and forth between different versions of our applications.

2) Has the concept of "environments" for each application, allowing us to deploy different versions of our application in each environment.
This is handy for example if you want to set up separate QA and DEV environments, and you want to easily deploy a build first in DEV then deploy the same version of the application in QA when your QA team is ready for the next build.

3) Externalizes the important container configuration properties (Tomcat memory settings, for example) to the Elastic Beanstalk console and API. Because of this we can easily save the settings and copy them between environments.

4) View application log files through the console and automatically roll and archive log files to S3.

Amazon EC2 Container Service(ECS)

ECS is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.

Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure. With simple API calls, we can launch and stop container-enabled applications, query the complete state of your cluster, and access many familiar features like security groups, Elastic Load Balancingetc.,

Deploying a Spring Boot Microservice To Docker / AWS Elastic Beanstalk

(lab : spring-boot-docker-jaxrs-demo-master)

Working With Elastic Beanstalk

Once we have a working Docker image of the application – there are 2 basic approaches to getting it up and running on Elastic Beanstalk.

The first option is to create an archive and upload it directly, the second is to upload it to a docker repository (e.g. Docker Hub), and reference the image directly from there.

Step 1 – Create the Archive

As it's a single file upload to Amazon – we will need to create a zip file containing the Spring Boot JAR and the associated Dockerfile.

For Maven, add the Maven Assembly plugin to the pom.xml (and assembly.xml descriptor to the project) to handle this through the maven packaging process.

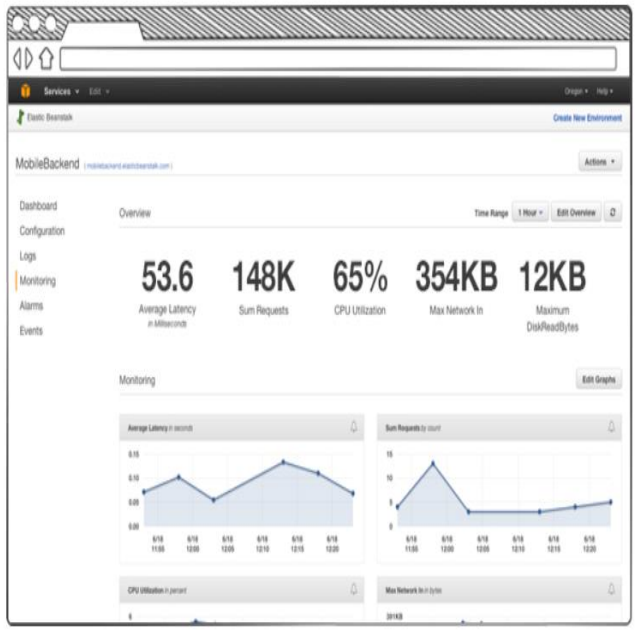
Pom.xml

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
```

Step2 - mvn clean package

aws Services Resource Groups jbossramana Ohio Support

Elastic Beanstalk Create New Application



Welcome to AWS Elastic Beanstalk

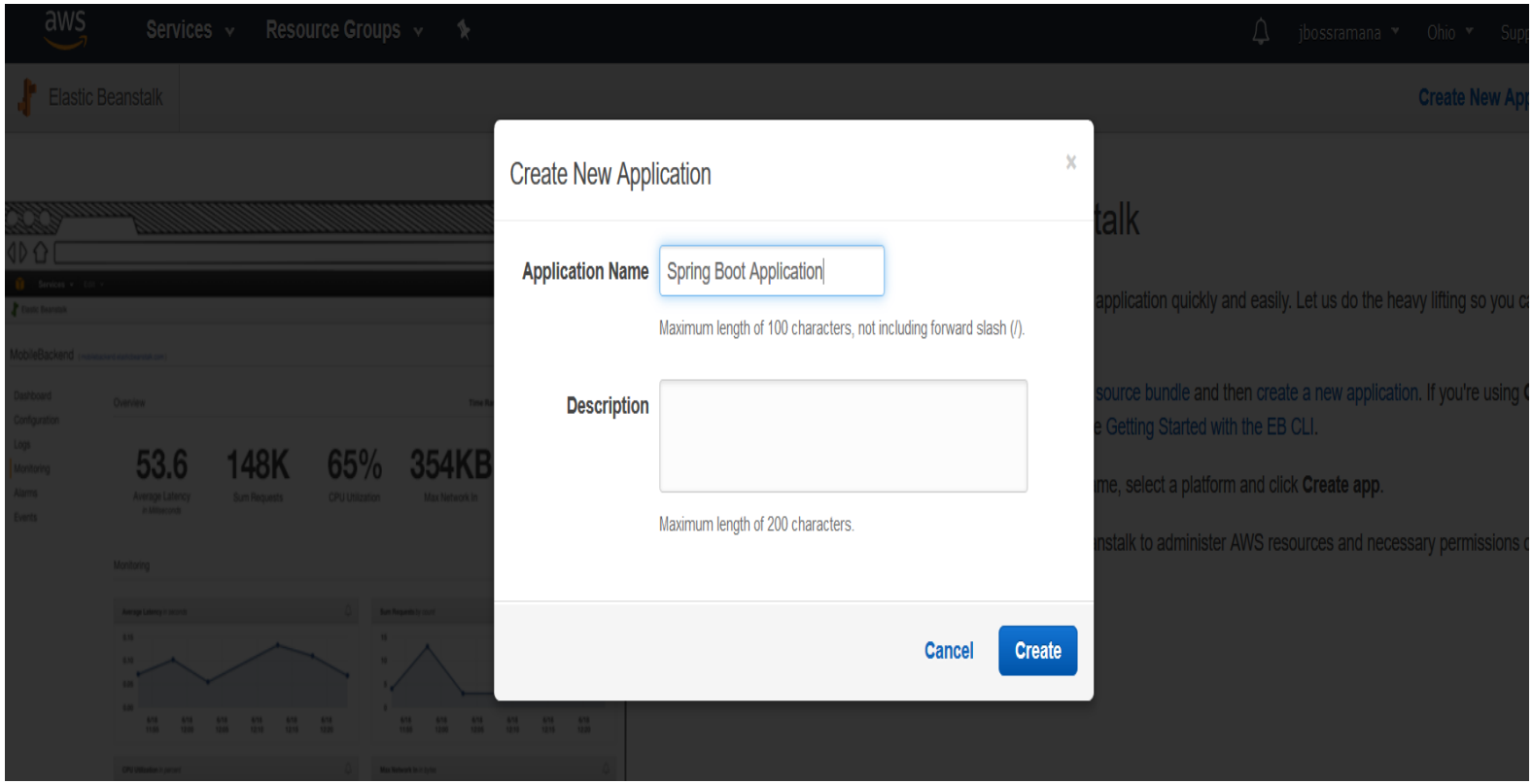
With Elastic Beanstalk, you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

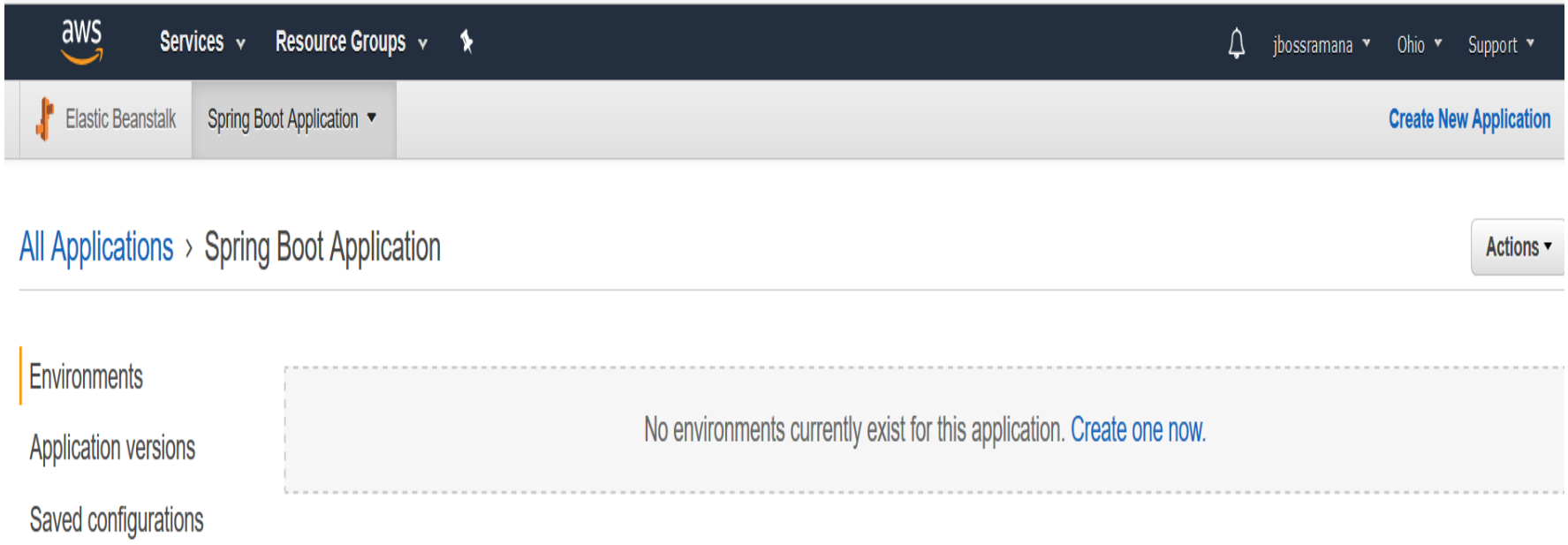
To deploy your **existing web application**, create an application source bundle and then create a new application. If you're using **Git** and would prefer to use it with our command line tool, please see Getting Started with the EB CLI.

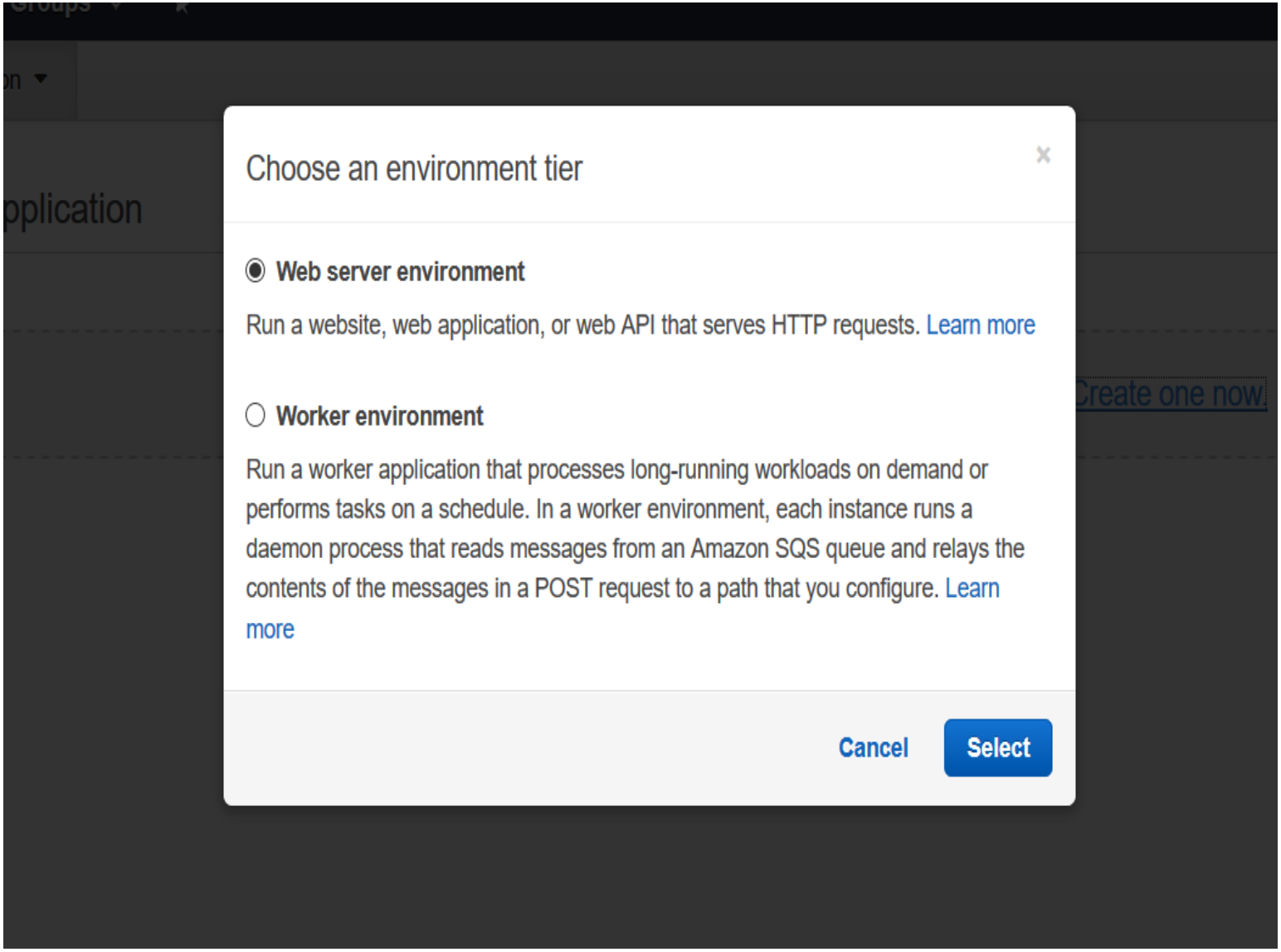
To deploy a **sample application**, click **Get started**, choose a name, select a platform and click **Create app**.

By launching the sample application, you allow AWS Elastic Beanstalk to administer AWS resources and necessary permissions on your behalf. Learn more

Get started


aws
Services
Resource Groups
jbossramana
Ohio
Elastic Beanstalk
Create New App
Create New Application
Application Name
Spring Boot Application
Maximum length of 100 characters, not including forward slash (/).
Description
Maximum length of 200 characters.
Cancel
Create
53.6
148K
65%
354KB
talk
application quickly and easily. Let us do the heavy lifting so you c
source bundle and then create a new application. If you're using
e Getting Started with the EB CLI.
me, select a platform and click Create app.
nstalk to administer AWS resources and necessary permissions


aws
Services
Resource Groups
jbossramana
Ohio
Support
Elastic Beanstalk
Spring Boot Application
Create New Application
All Applications › Spring Boot Application
Actions
Environments
Application versions
Saved configurations
No environments currently exist for this application. Create one now.


Choose an environment tier
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. In a worker environment, each instance runs a daemon process that reads messages from an Amazon SQS queue and relays the contents of the messages in a POST request to a path that you configure. Learn more
Cancel
Select

Domain Leave blank for autogenerated value .us-east-2.elasticbeanstalk.com Check availability

Description

Base configuration

Tier Web Server (Choose tier)

Platform Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

Application code

Docker

-- Choose a platform --
Preconfigured
Go
Node.js
Java
Ruby
Tomcat
PHP
Packer
Python
.NET (Windows/IIS)
Preconfigured – Docker
GlassFish
Go
Python
Generic
Multi-container Docker
Docker

oring Boot Application.

Upload a source bundle from your computer or copy one from Amazon S3.

Upload ZIP or WAR

Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Source code origin
(Maximum size 512 MB)

- (•) Local file
  - Browse... spring-boot-docker-demo.zip
- ( ) Public S3 URL
  - https://s3.us-east-2.amazonaws.com

Version label

spring boot application-source

Unique name for this version of your application code.

Cancel Upload

Application code

- ○ Sample application

  Get started right away with sample code.

- ○ Existing version

  Application versions that you have uploaded for **Spring Boot Application**.

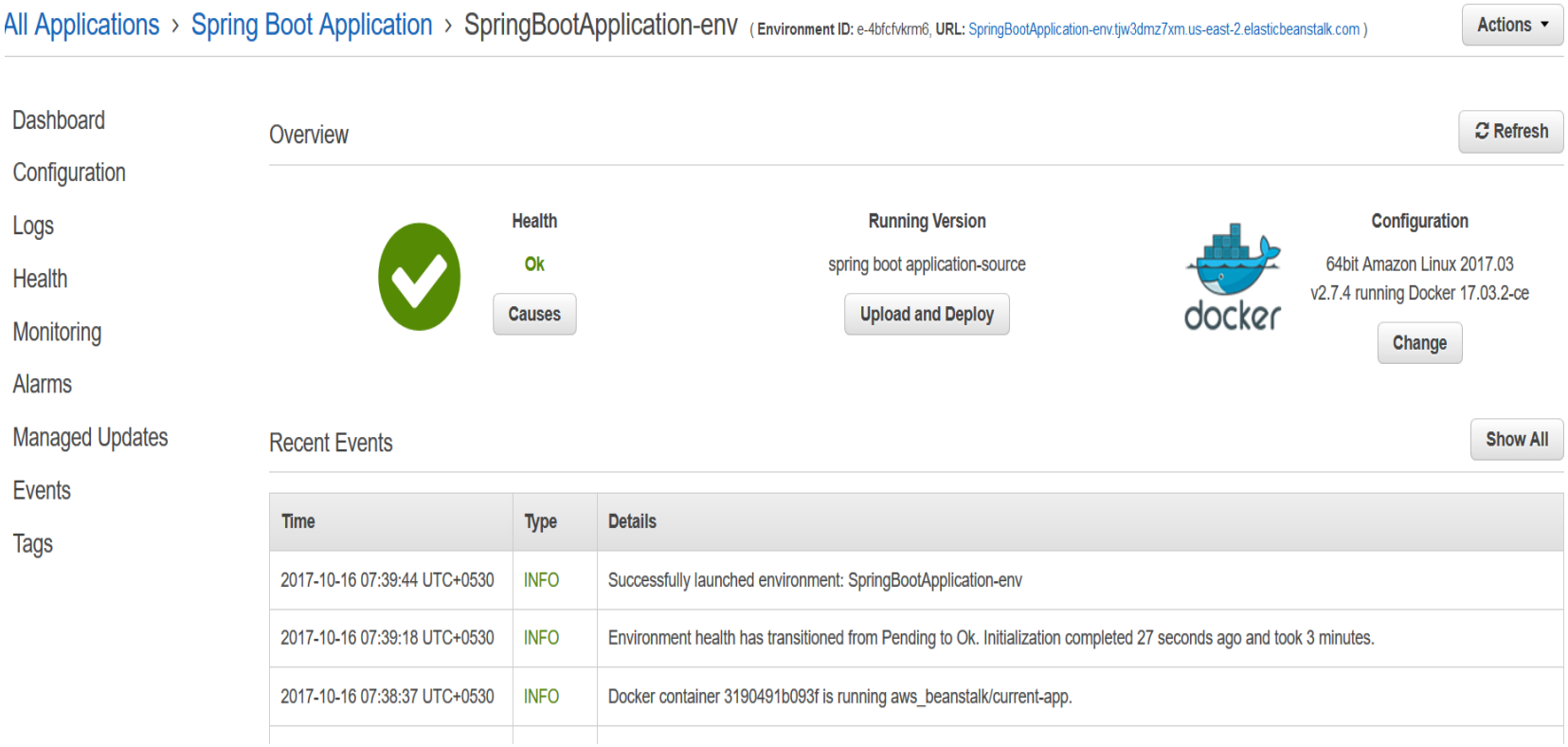
  -- Choose a version --

- ◉ Upload your code

  Upload a source bundle from your computer or copy one from Amazon S3.

  Upload spring boot application-source

Cancel Configure more options Create environment

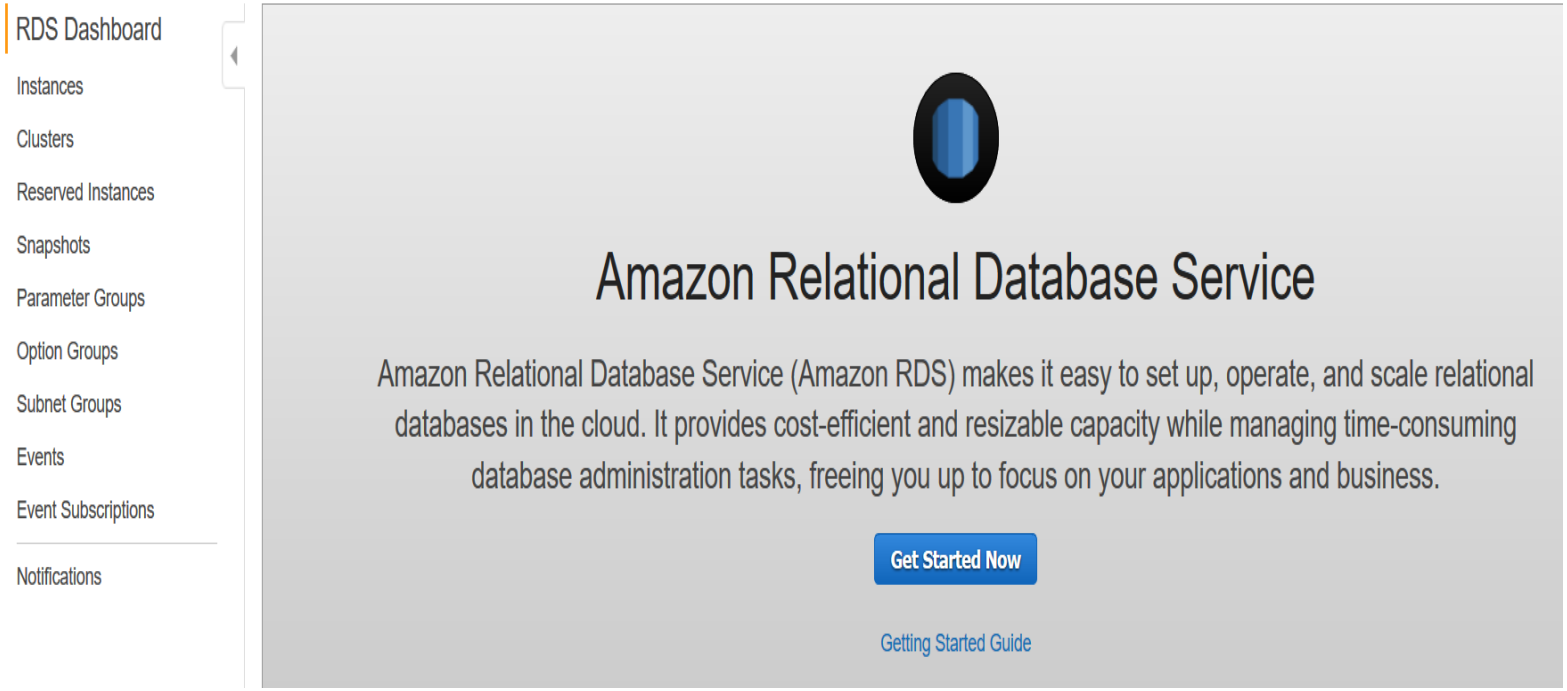


From the Brower access : <URL>/mytestapp/demo/hello

Deploying a Spring Boot Microservice using Postgres Database To Tomcat / AWS Elastic Beanstalk (lab : spring-boot-aws-postgres)

Setting up Postgres in AWS

AWS console-> RDS service -> Get Started Now



aws Services Resource Groups

Step 1: Select Engine

...

☑ Free tier eligible only

Select Engine

To get started, choose a DB Engine below and click Select.

Amazon Aurora

MySQL

MariaDB

PostgreSQL

PostgreSQL

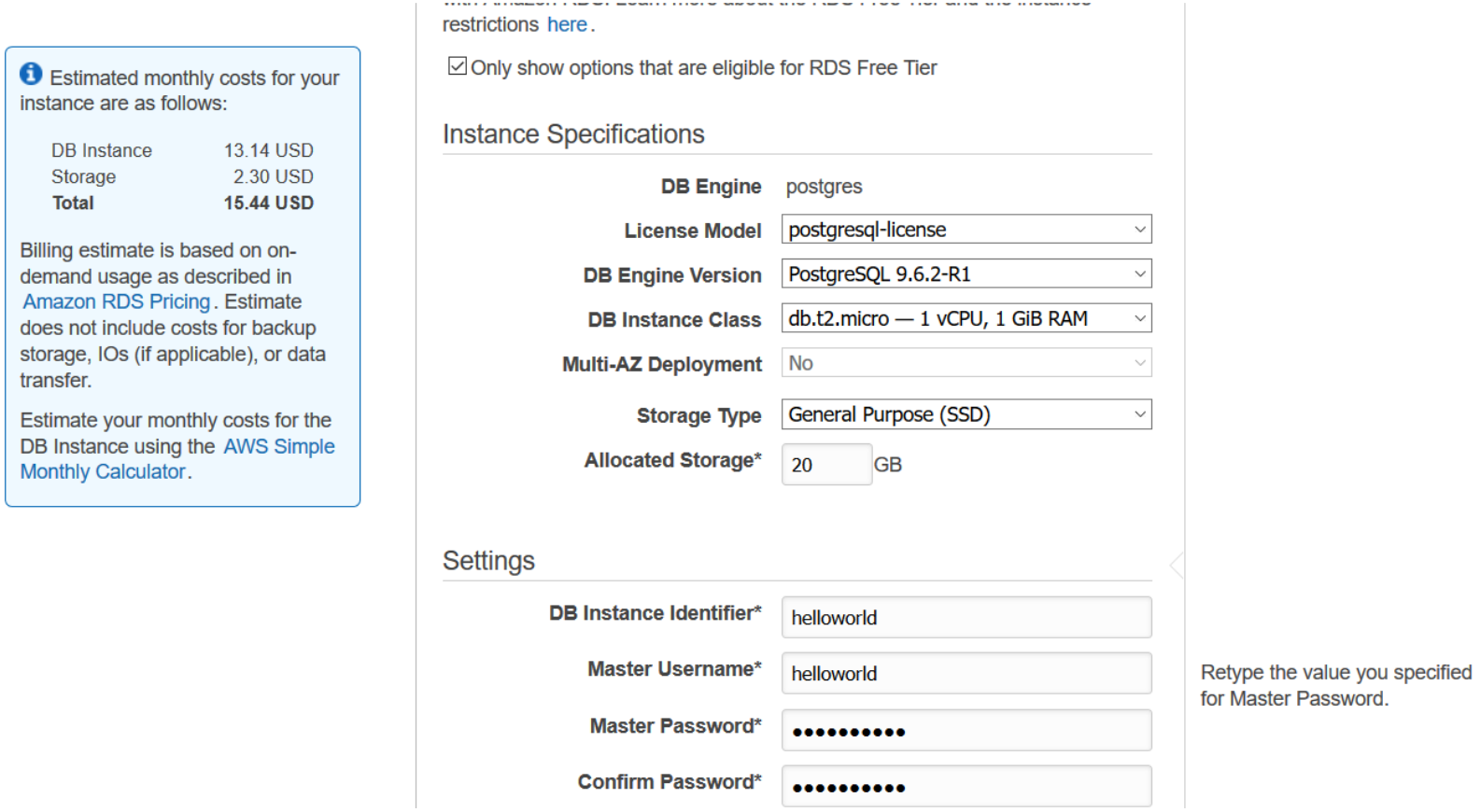
Select

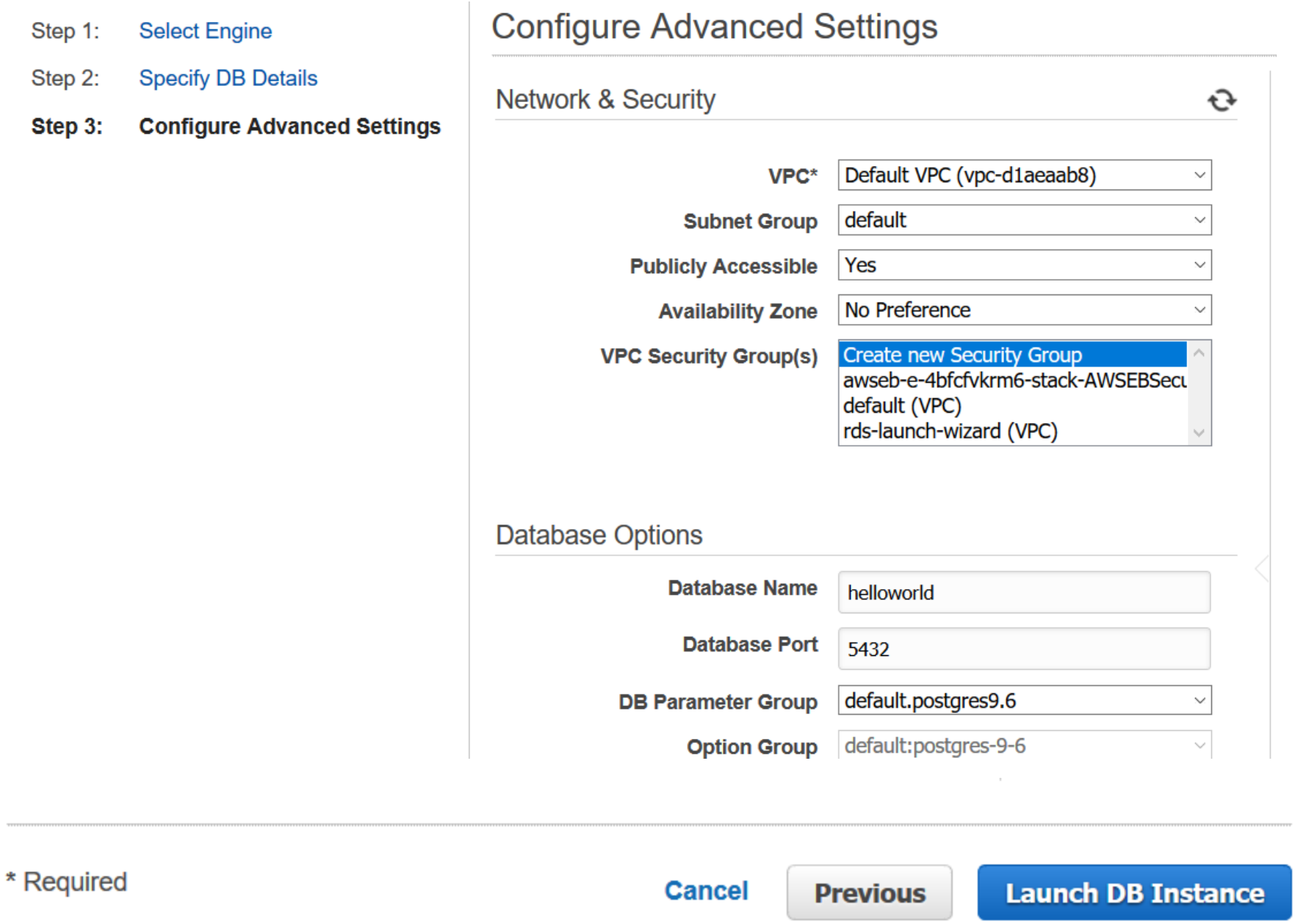
PostgreSQL is a powerful, open-source object-relational database system with a strong reputation of reliability, stability, and correctness.

- High reliability and stability in a variety of workloads.
- Advanced features to perform in high-volume environments.
- Vibrant open-source community that releases new features multiple times per year.
- Supports multiple extensions that add even more functionality to the database.
- The most Oracle-compatible open-source database.
- Free tier eligible

db instance, username, and database name -> helloworld

Step2 : Specify DB details




Step 1: Select Engine
Step 2: Specify DB Details
Step 3: Configure Advanced Settings
Configure Advanced Settings
Network & Security
VPC*
Default VPC (vpc-d1aeaab8)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
awseb-e-4bfcfvkrm6-stack-AWSEBSecu
default (VPC)
rds-launch-wizard (VPC)
Database Options
Database Name
helloworld
Database Port
5432
DB Parameter Group
default.postgres9.6
Option Group
default:postgres-9-6
* Required
Cancel
Previous
Launch DB Instance

Step 1: Select Engine

Step 2: Specify DB Details

Step 3: Configure Advanced Settings

Your DB Instance is being created.

Note: Your instance may take a few minutes to launch.

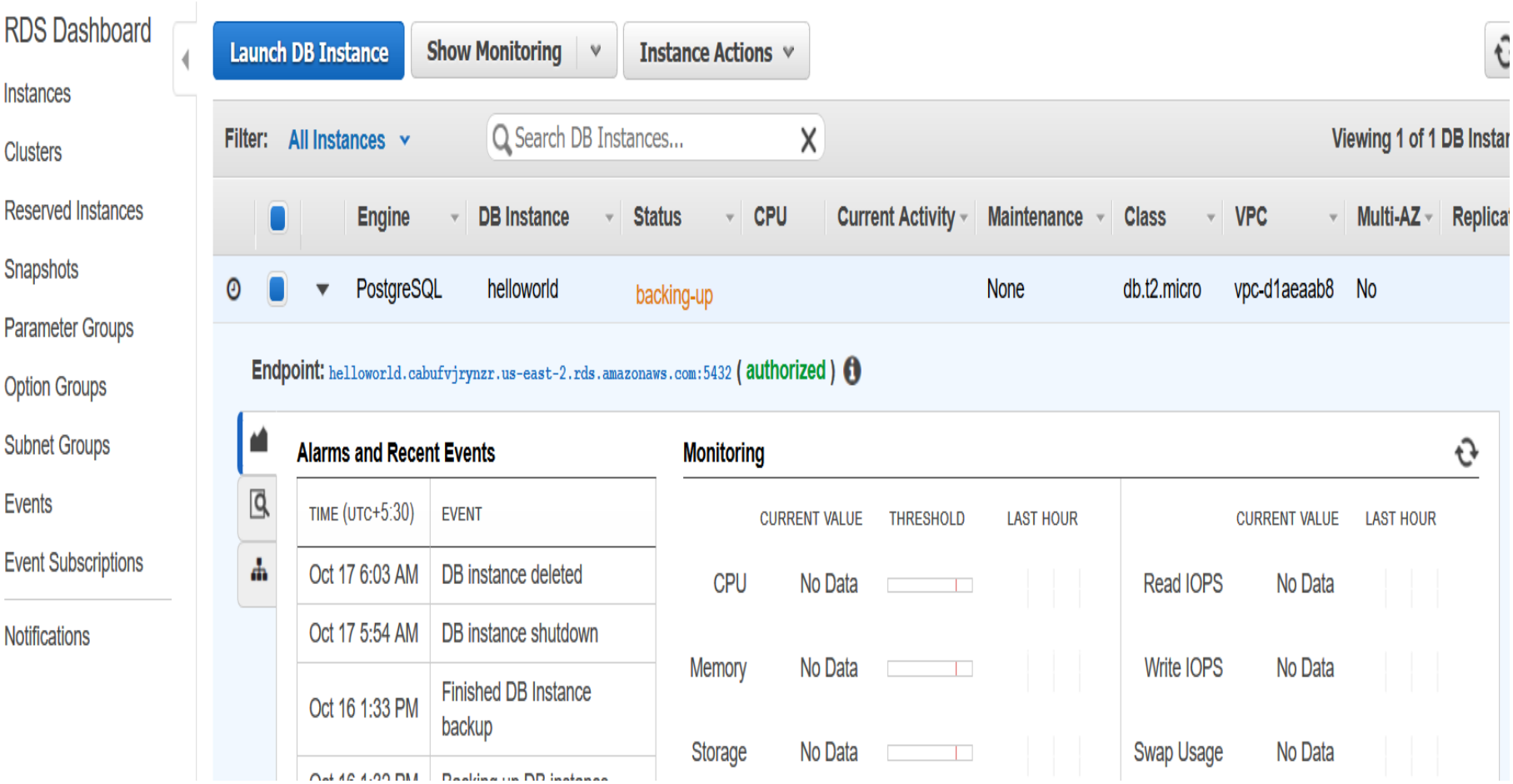
Connecting to your DB Instance

Once Amazon RDS finishes provisioning your DB instance, you can use a SQL client application or utility to connect to the instance.

Learn about connecting to your DB instance

View Your DB Instances

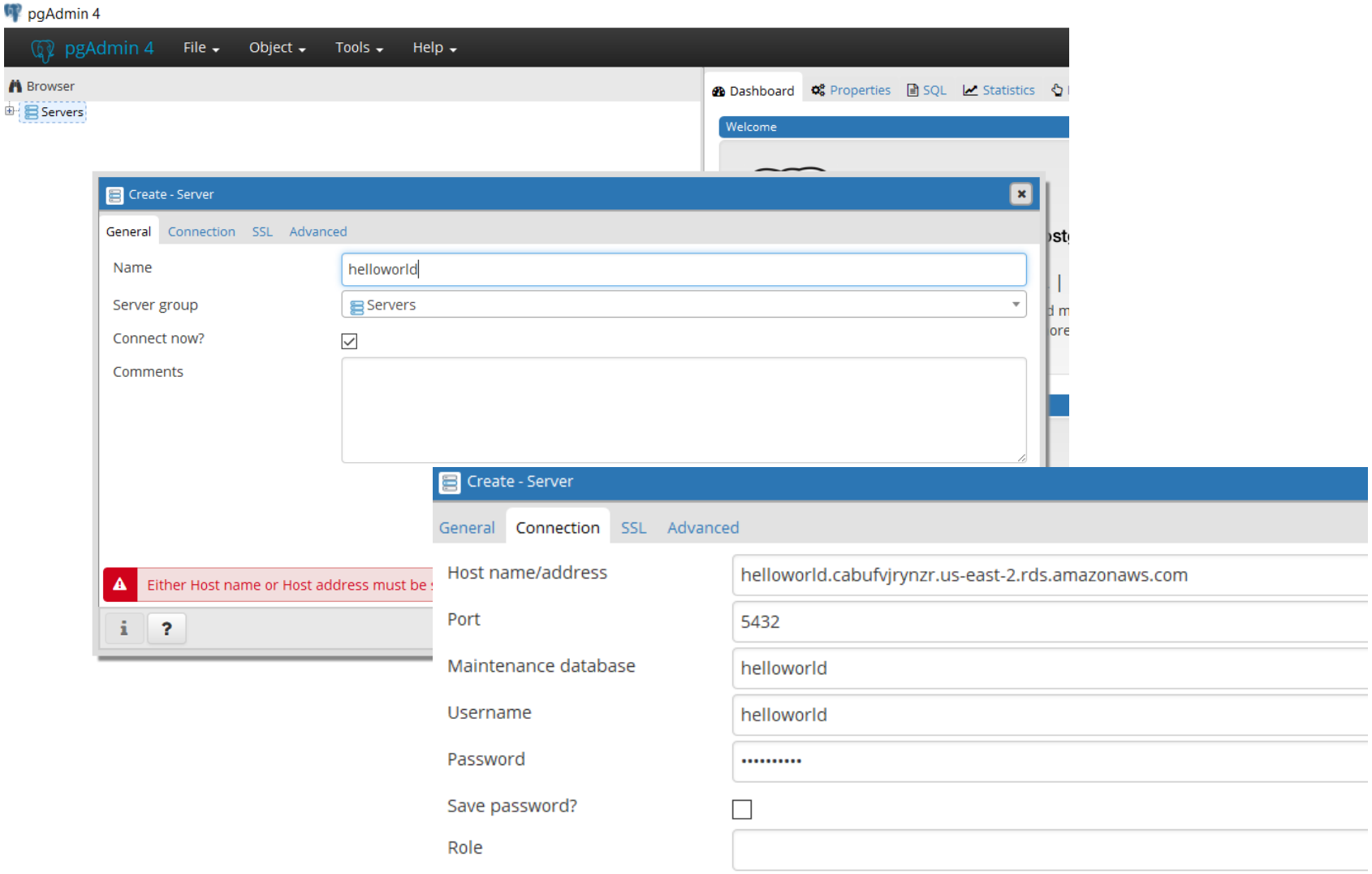
Refer to Endpoint

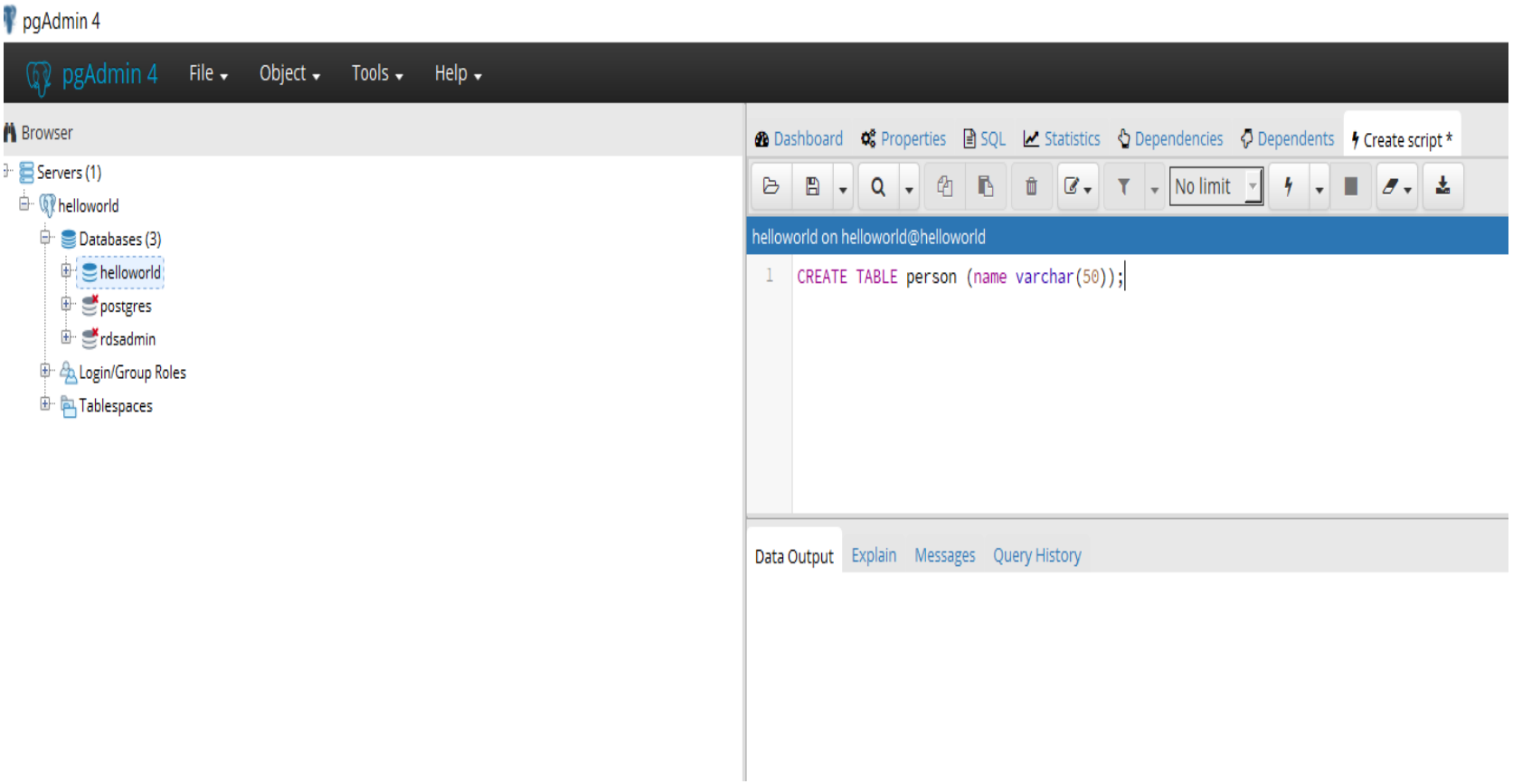


Refer To :

spring-boot-aws-postgres/src/main/resources/application.properties file:

```
spring.database.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://YOUR_AWS_RDS_ENDPOINT:5432
/helloworld
spring.datasource.username=helloworld
spring.datasource.password= helloworld
```


pgAdmin 4
pgAdmin 4
File
Object
Tools
Help
Browser
Servers
Dashboard
Properties
SQL
Statistics
Welcome
Create - Server
General
Connection
SSL
Advanced
Name
helloworld
Server group
Servers
Connect now?
Comments
Either Host name or Host address must be
Create - Server
General
Connection
SSL
Advanced
Host name/address
helloworld.cabufvjrynzr.us-east-2.rds.amazonaws.com
Port
5432
Maintenance database
helloworld
Username
helloworld
Password
Save password?
Role


pgAdmin 4
pgAdmin 4
File
Object
Tools
Help
Browser
Servers (1)
helloworld
Databases (3)
helloworld
postgres
rdsadmin
Login/Group Roles
Tablespaces
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Create script *
No limit
helloworld on helloworld@helloworld
1
CREATE TABLE person (name varchar(50));
Data Output
Explain
Messages
Query History

eclipse project-> spring-boot-aws-postgres > mvn spring-boot:run

From the browser, access the below url

http://localhost:8080/greetings?YOUR_NAME

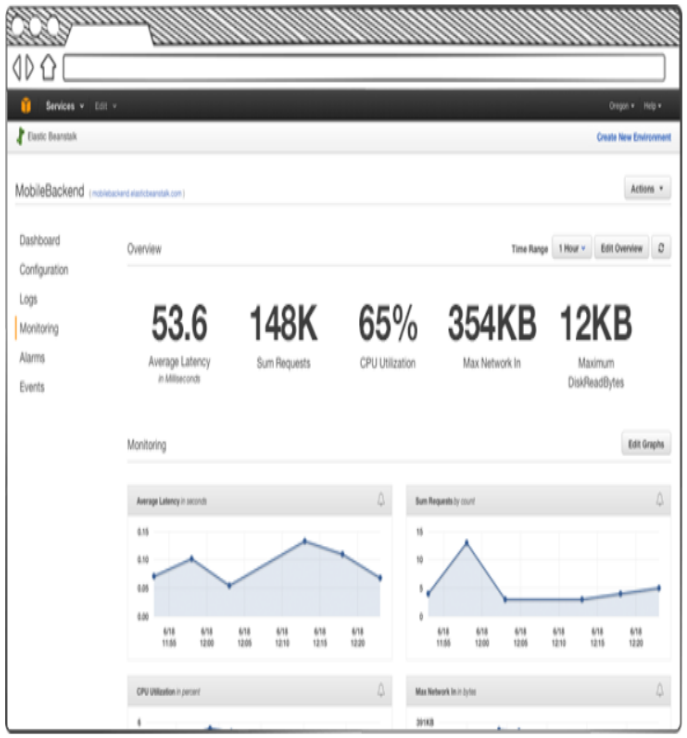
DEPLOYING THE APPLICATION IN AWS

In EBS create a new application using default settings and then create a new environment within the new application.

When prompted to choose a web environment tier, choose Web server environment. You'll be prompted to choose a platform, select Tomcat and leave all other fields as default.

AWS will then take about 10 minutes to prepare the environment.

Select -> Create New Application



Welcome to AWS Elastic Beanstalk

With Elastic Beanstalk, you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

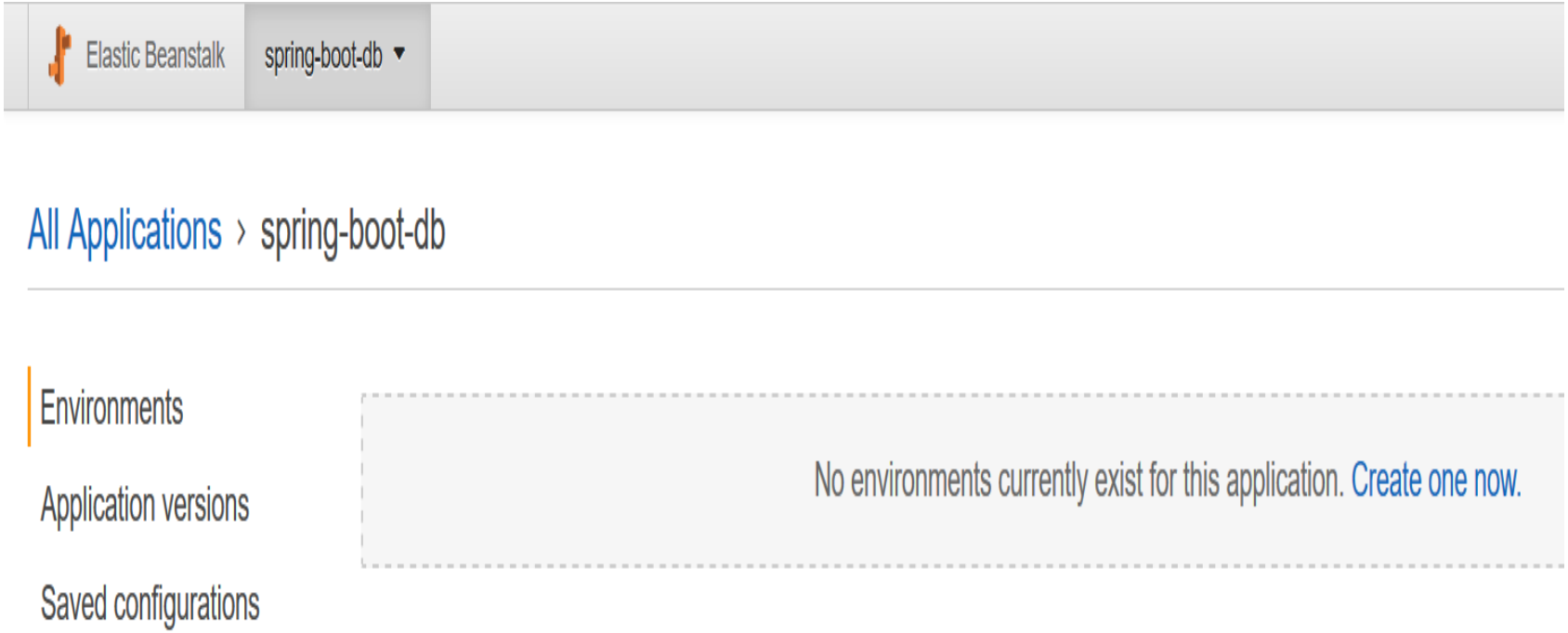
To deploy your **existing web application**, create an application source bundle and then create a new application. If you're using **Git** and would prefer to use it with our command line tool, please see Getting Started with the EB CLI.

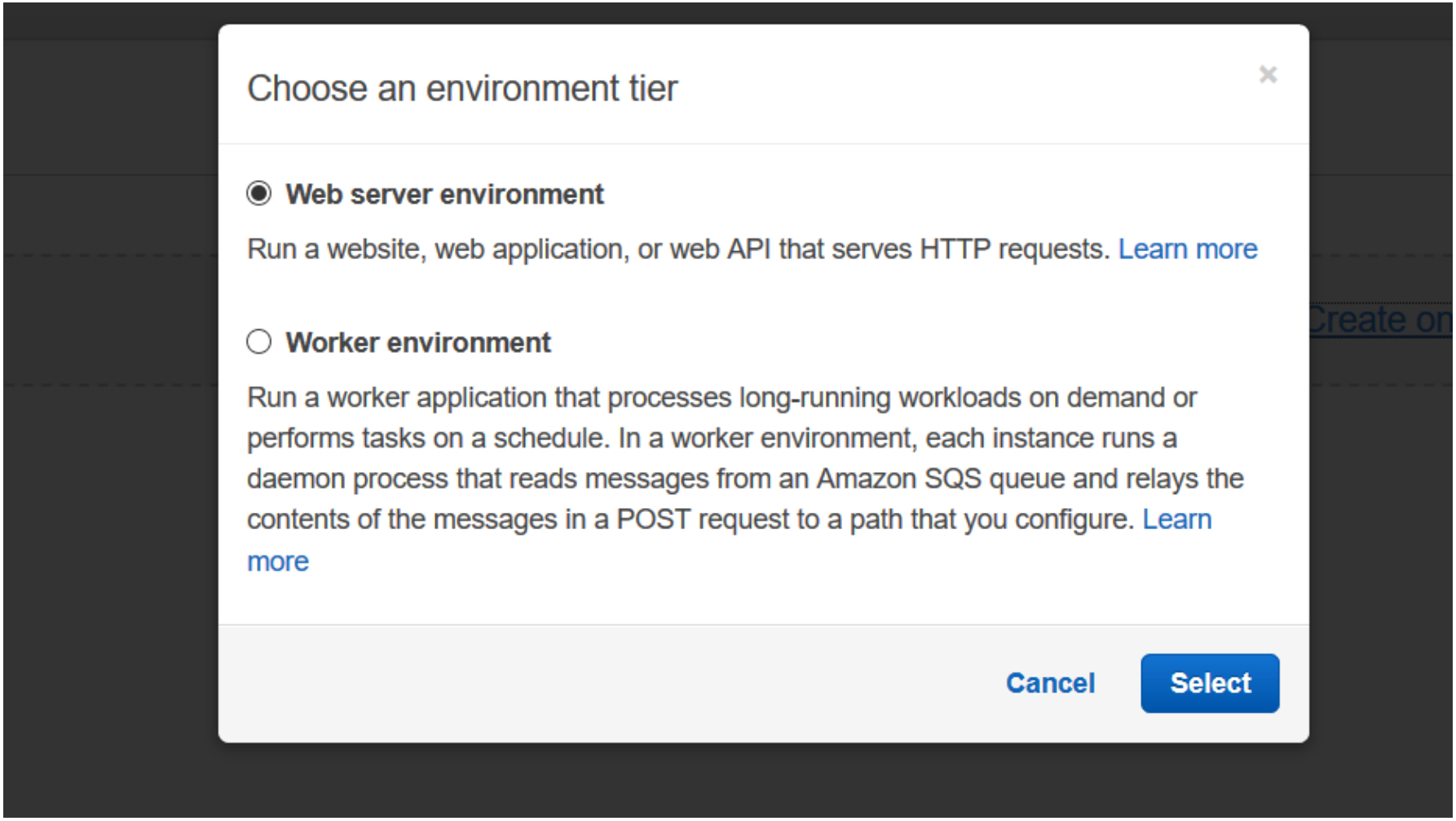
To deploy a **sample application**, click **Get started**, choose a name, select a platform and click **Create app**.

By launching the sample application, you allow AWS Elastic Beanstalk to administer AWS resources and necessary permissions on your behalf. Learn more

Get started

create a new environment




Choose an environment tier
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. In a worker environment, each instance runs a daemon process that reads messages from an Amazon SQS queue and relays the contents of the messages in a POST request to a path that you configure. Learn more
Cancel
Select

Create a new environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. Learn more

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Application name spring-boot-db

Environment name SpringBootDb-env

Domain Le -east-2.elasticbeanstalk.com Check availability

Description

-- Choose a platform --
Preconfigured
Go
Node.js
Java
Ruby
Tomcat
PHP
Packer
Python
.NET (Windows/IIS)
Preconfigured – Docker
GlassFish
Go
Python

Base configuration

Tier We

Application code

○ Sample application

Get started right away with sample code.

○ Existing version

Application versions that you have uploaded for **spring-boot-db**.

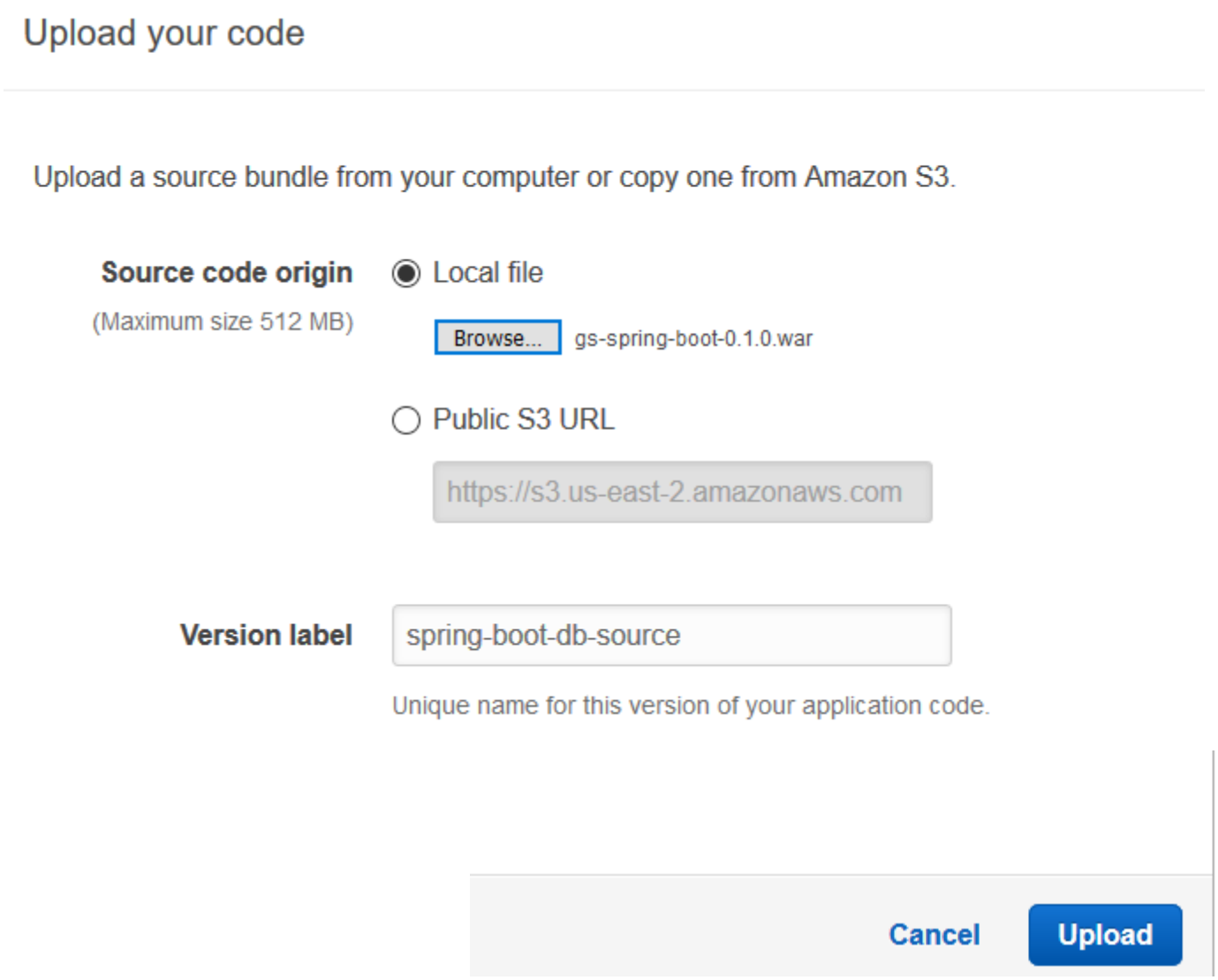
-- Choose a version --

◉ Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Upload ZIP or WAR

Cancel | Configure more options | Create environment


Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.
Source code origin
(Maximum size 512 MB)
Local file
Browse...
gs-spring-boot-0.1.0.war
Public S3 URL
https://s3.us-east-2.amazonaws.com
Version label
spring-boot-db-source
Unique name for this version of your application code.
Cancel
Upload

Platform

◉ Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

Tomcat

○ Custom platform NEW

Platforms created and owned by you. Learn more

-- Choose a custom platform --

Application code

○ Sample application

Get started right away with sample code.

○ Existing version

Application versions that you have uploaded for **spring-boot-db**.

-- Choose a version --

◉ Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Upload spring-boot-db-source

Cancel | Configure more options | Create environment

Elastic Beanstalk | spring-boot-db ▾ | Create New

All Applications › spring-boot-db › SpringBootDb-env (**Environment ID:** e-xirgyme279, **URL:** SpringBootDb-env.j7dw3mt42i.us-east-2.elasticbeanstalk.com)

- Dashboard
- Configuration
- Logs
- Health
- Monitoring
- Alarms
- Managed Updates
- Events
- Tags

Overview

Health

Ok

Causes

Running Version

spring-boot-db-source

Upload and Deploy

Configuration

64bit Amazon Linux 2017.
v2.6.5 running Tomcat 8 Ja

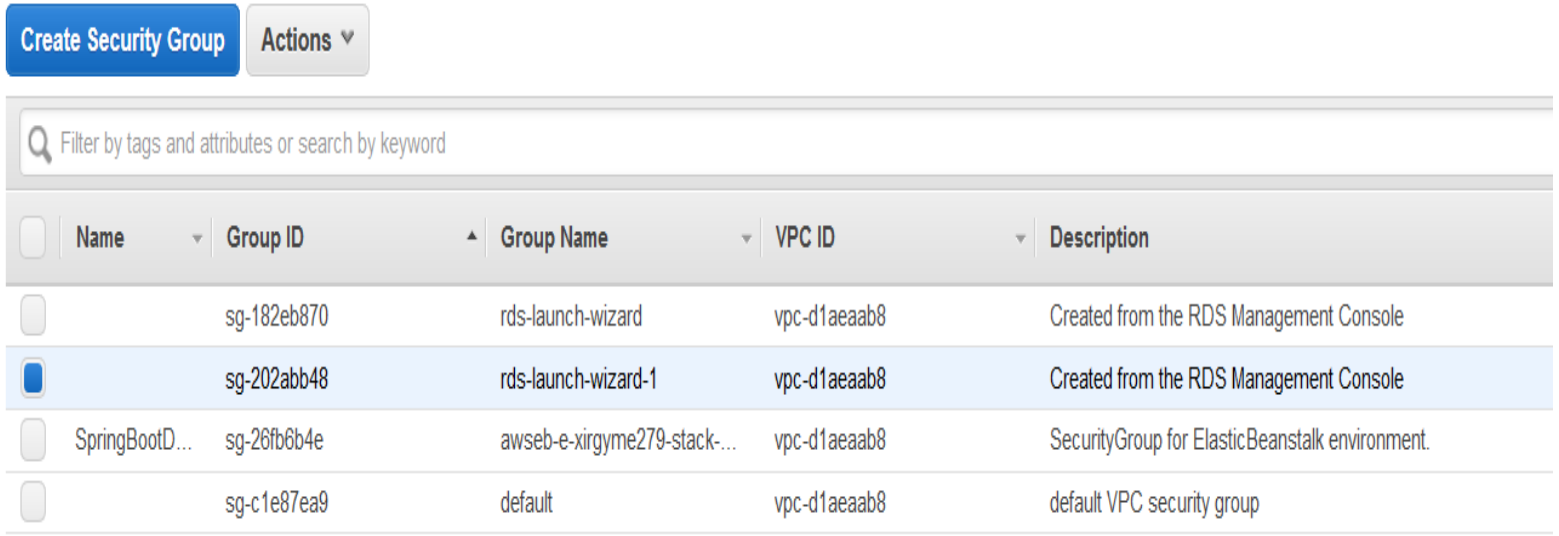
Change

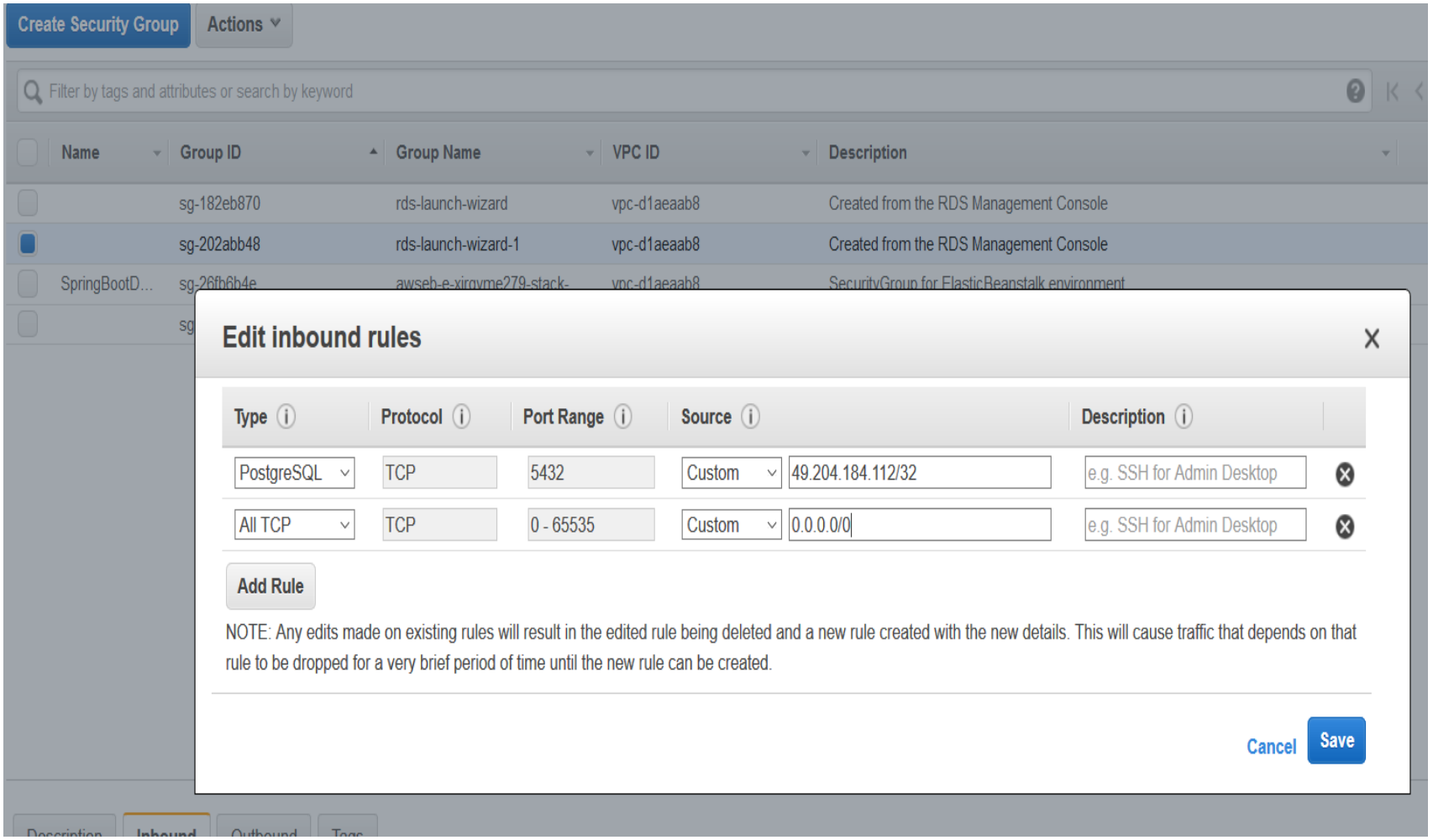
Recent Events

Time	Type	Details
2017-10-17 10:37:34 UTC+0530	INFO	Successfully launched environment: SpringBootDb-env
2017-10-17 10:36:56 UTC+0530	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 10 seconds ago and took 2 minutes.
2017-10-17 10:36:17 UTC+0530	INFO	Waiting for EC2 instances to launch. This may take a few minutes.

Configuring the Security Group :

Select EC2 -> NETWORK & SECURITY -> Security Groups




Create Security Group
Actions
Filter by tags and attributes or search by keyword
Name
Group ID
Group Name
VPC ID
Description
sg-182eb870
rds-launch-wizard
vpc-d1aeaab8
Created from the RDS Management Console
sg-202abb48
rds-launch-wizard-1
vpc-d1aeaab8
Created from the RDS Management Console
SpringBootD...
sg-26fb6b4e
awseb-e-xirqvme279-stack-
vpc-d1aeaab8
SecurityGroup for ElasticBeanstalk environment
Edit inbound rules
Type
Protocol
Port Range
Source
Description
PostgreSQL
TCP
5432
Custom
49.204.184.112/32
e.g. SSH for Admin Desktop
All TCP
TCP
0 - 65535
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
Add Rule
NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.
Cancel
Save
Description
Inbound
Outbound
Tags

Elastic Beanstalk | spring-boot-db ▾

All Applications › spring-boot-db › SpringBootDb-env (**Environment ID:** e-xirgyme279, **URL:** SpringBootDb-env.j7dw3mt42i.us-east-2.elasticbeanstalk.com)

- Dashboard
- Configuration
- Logs
- Health
- Monitoring
- Alarms
- Managed Updates
- Events
- Tags

Overview

Health

Ok

Causes

Running Version

spring-boot-db-source

Upload and Deploy

Recent Events

Time	Type	Details
2017-10-17 10:37:34 UTC+0530	INFO	Successfully launched environment: SpringBootDb-env
2017-10-17 10:36:56 UTC+0530	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 10 seconds ago and took 2 mi

springbootdb-env.j7dw3mt42i.us-east-2.elasticbeanstalk.com/greeting

Hello, World!

Visitors so far:

World