

**EC2 (Elastic Cloud Compute):**  
It's one of the first services that came out when AWS was released

You basically select the operating system Microsoft Windows or Linux, and then you get the operating system pre-installed.

If you want any additional software, you have to manually install it on top of the OS.  
So if you want the JDK you have to install Java, install Tomcat database, so on. So it's almost like you're getting a brand new laptop that only has the operating system, and you need to install your own tools on top of it.

You can also use custom Amazon images

**Elastic bean stalk:** Quickly deploy your web applications  
select a pre-configured virtual machine for your given web stack, like Java  
and Tomcat.

 There’s no software to install on the virtual machine. You simply deploy your code, your zip or war file

**WAR(web application archive):**

So now let's talk about deploying our application or deploying our code.

So, the Java Web App specification defines a standard deployment file.

It's called a web application archive or War file

So, this is just a zipped archive of your web app that has a dot war extension.

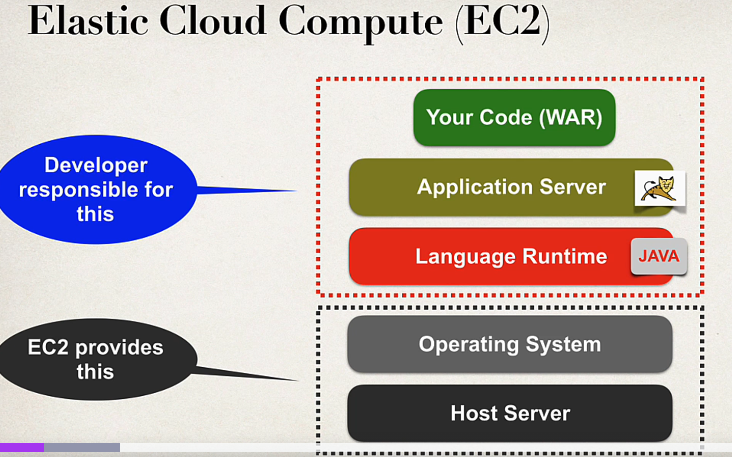
So basically, all of your source code, you know, source main web app, your web inf, your classes, your lib, basically all your compiled code, compiled application and associated assets are all zipped up into this war file and then you can deploy it.

**Relational Database Service (RDS):** This allows you to quickly deploy a relational database in the cloud. And it has support for a lot of the popular databases MySQL, Oracle, Microsoft SQL server and so on. And you can manage these tools using your normal admin tools. So, if you're using MySQL, you can use MySQL workbench.  
AWS also has support for NoSQL databases such as MongoDB and so on.  
So basically all the major database features that you need, you can find it in AWS with the support of the relational database service

Route 53: This allows you to route your custom domain name to your application on AWS.  
So basically, we'll configure route 53 to send my www.myspringbootapp.com to your AWS application. So, this is effectively the AWS Domain Name System or the DNS.

**Comparing EC2 to Elastic Beanstalk**

So first off, EC2 is a do-it-yourself solution. All you get is the operating system Windows or Linux. And if you want other apps you must manually install them. So, you need to manually install the JDK, Tomcat, MySQL and so on.

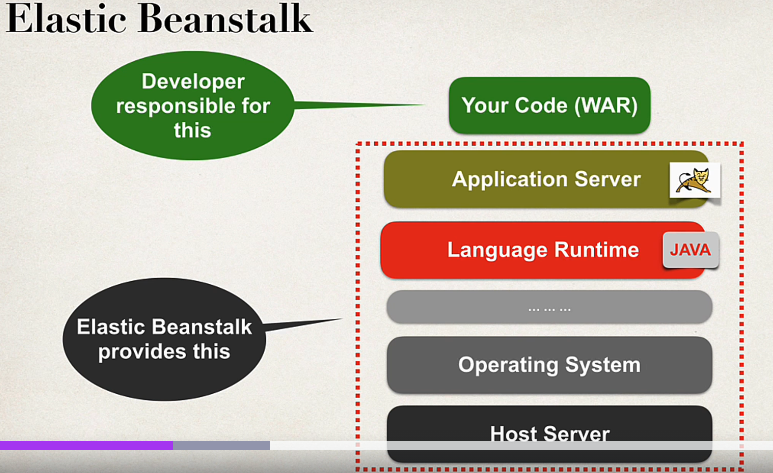


EC2-> So, you start off with the actual host server. You get the actual host operating system.

And EC2 provides this portion for you. Now as a developer, if you want to add additional software like Java app servers in your code, then you will be responsible for adding all these items. So again, just like I mentioned earlier, this is almost like joining a new company.

They give you a laptop, but they say, hey, this is just a very basic laptop**.** It only has windows installed. If you need any of your Java developer tools, you have to go through and install that yourself**.**

Elastic Bean Stalk-> So Elastic Beanstalk is a pre-packaged platform. It's great for deployments on a web stack. And you simply select the services that you need, and it will set it up for you. And this is known as platform as a service or PaaS



So you have your host server, your operating system, and some other items. And then they also give you the language runtime and the application server. So Elastic Beanstalk provides all of this for you as the developer. All you need to do is focus on deploying your code or your actual war file.

***Spring boot Jar - Includes embedded tomcat***

Now, when you deploy spring boot applications, you could deploy it as a war file, or you could deploy it as a Jar file. The recommended approach, or the preferred approach, is deploying a spring boot application as a Jar file. When you do this, it actually includes the embedded Tomcat. So you don't have to depend on AWS to provide the application server.  
It's already included in our Jar file our spring boot jar file. So here your spring boot jar has the embedded Tomcat inside of it.

**Use Case**: So, for your apps, I always recommend starting with Elastic Beanstalk because you can quickly get started with deploying your application by leveraging those pre-configured web stacks out of the box. Use EC2 if you need some low-level control, so you may want to use a different version of Java that's not supported by Elastic Beanstalk.

Or you may want to use a different Java app server like WebLogic or WebSphere, or some other app server, any other OS specific customizations you need to make. So EC2 will give you that low level control. And you can also achieve this by creating custom Elastic Beanstalk templates. If you don't want to use the low level EC2.

**Our App Architecture:**

A screenshot of a computer

AI-generated content may be incorrect.

So, in our application architecture we're going to use Elastic Beanstalk for deploying our  
web applications. We're going to have our Java app running on Tomcat.

We'll make use of the relational database service RDS. That's our database in the cloud using MySQL. And then we'll make use of route 53 to route our custom domain names to our application. That's hosted on AWS

And so, the big picture again is that we're going to take our application and deploy it in the cloud. So, our spring app with the database running in the Amazon cloud, we'll have our *own custom domain name* that anyone can access on their browser or whatever your domain name is. You can just dream up a creative, fancy, cool domain name, and we'll have it connected to our application.

More Cool features of Elastic Beanstalk:

For capacity provisioning you can say, hey, create 5 servers for me out of the box at the start instead of just one, create 5 or 10 or 20.

Um, you can also set up load balancing. So now that you have those 20 servers running, you can set up the load balancing rules between those different servers.

You could also set up auto scaling saying, hey, if we meet a certain threshold of traffic, then go ahead and spin up X number of additional servers to handle the rush hour for hour, I don't know, e-commerce site or whatever. And they also provide support for health monitoring, alarms and notifications and so on. A lot of additional features that are available with Elastic Beanstalk that you don't get out of the box with EC2.