DEPLOYING JAVA WEB APPLICATIONS

"How do I quickly and easily deploy a Java web application?"

Overview

Amazon Web Services (AWS) is a great platform for running Java workloads, and offers several options for deploying and managing both off-the-shelf and custom applications. When planning to deploy a Java web application to AWS for the first time, new customers are sometimes unclear as to which method will allow them to quickly get their application up and running. This document provides guidance for deploying Java web applications quickly and easily using either AWS-provided or self-managed IIS web servers.

The following sections assume basic knowledge of Amazon Elastic Compute Cloud (Amazon EC2), Java application development, load balancing, and MySQL databases.

General Best Practices

The size and installation complexity of web applications can vary greatly, therefore there is rarely a one-size-fits-all solution for deploying and hosting Java applications. However, there are some universal best practices to consider when deploying any web application:

- Understand the deployment, installation, and configuration characteristics of the application.
- Understand application expectations from initial deployment to future scalability, availability, and backup and recovery requirements.
- Use automation whenever possible for deployment and other tasks where consistency is important.
- Leverage source code or application repositories to protect your application.

Application on the AWS Platform

AWS offers several tools and services to enable both AWS-managed and customer-managed Java application deployment. The table below is a high-level reference to help identify the most appropriate option for a specific scenario. The following sections describe these different approaches and their applicable use cases in more detail.

Application Characteristics	Packaging Tools	Deployment Mechanism	Deployment Method/ Environment
Custom Java applications developed in Eclipse	Eclipse	Single-click deployment from within Eclipse	**AWS Toolkit for Eclipse**
Java web applications deployed as a JAR, WAR, or ZIP file, and requiring minimal OS changes	JAR, WAR, or ZIP	Automated deployment of packaged application using AWS Elastic Beanstalk	**AWS Elastic Beanstalk**
Any Java application or server configuration, especially those needing customized OS or third-party installers	Existing custom installers, application archive (JAR, WAR, ZIP), manual file copy, etc.	Existing software deployment tools and processes or automated deployment services, such as AWS CodeDeploy or AWS OpsWorks.	**EC2 Instances**

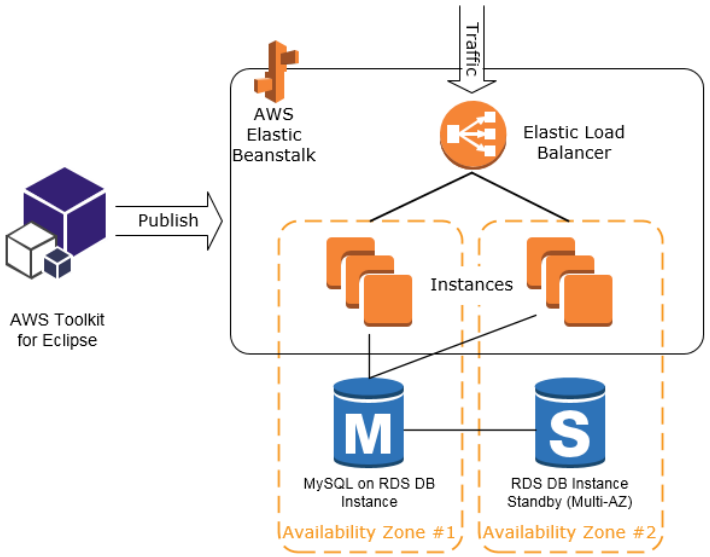
AWS Toolkit for Eclipse

The AWS Toolkit for Eclipse provides the easiest and most straightforward way to deploy custom Java applications to AWS. This approach allows developers to deploy Java code directly to AWS, and to create supporting resources, such as AWS-managed RDS databases, directly from within Eclipse. For production deployments, customers can use Eclipse to create Java Archive (JAR) or Web application ARchive (WAR) files that are then versioned and deployed through more controlled change management processes using the AWS Elastic Beanstalk (Elastic Beanstalk) console, Elastic Beanstalk Command Line Interface (EB CLI), or Elastic Beanstalk API calls.

AWS Elastic Beanstalk

Elastic Beanstalk is an easy-to-use service for deploying and scaling Java web applications. Elastic Beanstalk supports several platform configurations for Java applications, including multiple versions of Java with the Apache Tomcat application server and Java-only configurations for applications that do not use Tomcat.[1] The Java-only option allows customers to include any required library JAR files in the source bundle for Java web applications that don't use a web container or use a different one, such as Jetty or GlassFish. Once deployed, Elastic Beanstalk automatically manages capacity provisioning, load balancing, and Auto Scaling. This approach is appropriate for companies deploying Java applications that include the following criteria:

- Require minimal OS changes[2]
- Either run in Apache Tomcat 7 or 8, or are packaged with their own web container



Elastic Beanstalk supports the following packaging and deployment mechanisms:

- Custom applications developed and deployed directly to Elastic Beanstalk using Eclipse and the *AWS Toolkit for Eclipse*
- Applications packaged into a JAR, WAR, or ZIP file,[3] then deployed with the Elastic Beanstalk console, EB CLI, or Elastic Beanstalk API calls

EC2 Instances

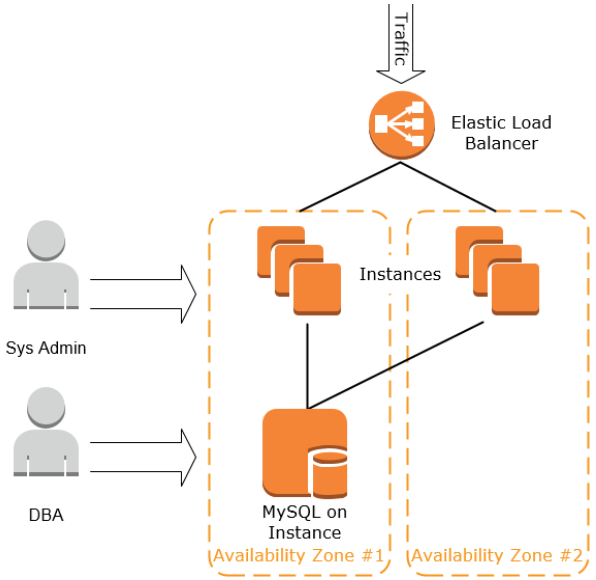
Amazon EC2 Windows instances allow customers to deploy Java applications to AWS using their existing application deployment tools and processes, or to integrate Java application deployment with automated deployment tools and services such as AWS CodeDeploy or AWS OpsWorks. AWS CodeDeploy and AWS OpsWorks make it easier to rapidly release new features, and automate the configuration, deployment, management, and updates of your services and applications on both EC2 instances and on-premises servers.

Self-managed Amazon EC2 instances offer the flexibility to choose specific operating system, Java, and Java web container versions that an application or a company's IT standards require. Administrator or root access to EC2 virtual machines allows customers to install and use existing infrastructure software, Java platform, and deployment tools to support their Java web applications. This option is best suited for companies with the following requirements:

1 For a list of supported platforms, see http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.platforms.html#concepts.platforms.java
2 Note that Elastic Beanstalk configuration files support advanced platform and OS configuration options. However this requires additional Elastic Beanstalk packaging effort and expertise.
3 To deploy multiple applications to one Elastic Beanstalk environment, customers can bundle multiple WAR files into a single ZIP file.

- Want to leverage their existing Java platform expertise and standards
- Need to integrate their AWS environment into existing on-premises development and deployment processes
- Require the most flexibility in OS and Java platform selection, version, and configuration
- Plan to deploy third-party Java applications with sophisticated installers, manual license configuration, or extensive external dependencies

This approach offers tremendous flexibility, but also requires advanced AWS knowledge and service configuration, such as creating and configuring Elastic Load Balancing load balancers, Auto Scaling groups, and self-managing the Java platform and underlying OS as well as implementing custom application-deployment processes.



Resources

Web Application Hosting in the AWS Cloud: Best Practices	http://aws.amazon.com/whitepapers/web-application-hosting-best-practices/ AWS whitepaper on how to create a highly available, scalable web application
Overview of Deployment Options	http://d0.awsstatic.com/whitepapers/overview-of-deployment-options-on-aws.pdf AWS whitepaper on deployment services, features, and strategies
AWS Elastic Beanstalk Documentation	http://aws.amazon.com/documentation/elastic-beanstalk/ AWS webpage with links to Elastic Beanstalk technical documentation including the *Developer Guide* and detailed information about getting started with web applications.
AWS Toolkit for Eclipse	http://aws.amazon.com/eclipse/
AWS CodeDeploy	http://aws.amazon.com/codedeploy
AWS OpsWorks	https://aws.amazon.com/opsworks/
AWS and Java	http://aws.amazon.com/java/