Red Hat JBoss Enterprise Application Platform 7.3

Introduction to JBoss EAP

For Use with Red Hat JBoss Enterprise Application Platform 7.3

Last Updated: 2020-03-18

Red Hat JBoss Enterprise Application Platform 7.3 Introduction to JBoss EAP

For Use with Red Hat JBoss Enterprise Application Platform 7.3

Legal Notice

Abstract

This document provides a high-level, conceptual overview of Red Hat JBoss Enterprise Application Platform (JBoss EAP). The document also introduces you to JBoss EAP subsystems and the different operating modes that JBoss EAP offers.

Table of Contents

CHAPTER 1. OVERVIEW OF GENERAL CONCEPTS

Before understanding how Red Hat JBoss Enterprise Application Platform can be configured and deployed, there are some important concepts to understand.

1.1. JAVA

Java is a programming language and a computing platform that incorporates concepts such as object-orientation, classes, and concurrency. Java applications are compiled down to bytecode and are run inside a Java Virtual Machine (JVM).

1.2. APPLICATION SERVERS

An application server, or app server, is software that provides an environment to run web applications. Most app servers also provide functionality to web applications running in their environment through a set of APIs. For example, an app server can provide an API for connecting to a database.

1.3. JAKARTA EE 8

Jakarta EE 8, maintained by the Eclipse Foundation, is the exact equivalent of Java Enterprise Edition 8.

For information about Jakarta EE 8, see About Jakarta EE.

1.4. JAVA ENTERPRISE EDITION 8

Java Platform, Enterprise Edition (Java EE) is a standards-based enterprise platform that provides both an API and runtime environment for running and developing Java applications. The goal is to improve developer productivity by providing rich enterprise capabilities in easy to consume frameworks that eliminate boilerplate and reduce technical burden. The frameworks that compose Java EE are heavily tested in combination.

Java EE 8, which is based on JSR 366, builds upon Java EE 7. The primary focus of this release is to improve API and programming models needed for today's applications. In addition to the Java EE (JSR 366) specification, the following specifications are new or updated in Java EE:

- JSR 250: Common Annotations 1.3
- JSR 338: Java Persistence 2.2
- JSR 356: Java API for WebSocket 1.1
- JSR 365: Contexts and Dependency Injection (CDI) 2.0
- JSR 367: The Java API for JSON Binding (JSON-B) 1.0
- JSR 369: Java Servlet 4.0
- JSR 370: Java API for RESTful Web Services (JAX-RS) 2.1
- JSR 372: JavaServer Faces (JSF) 2.3
- JSR 374: Java API for JSON Processing (JSON-P) 1.1
- JSR 375: Java EE Security API 1.0

- JSR 380: Bean Validation 2.0
- JSR 919: JavaMail 1.6

CHAPTER 2. OVERVIEW OF JBOSS EAP

2.1. ABOUT JBOSS EAP 7

Red Hat JBoss Enterprise Application Platform (JBoss EAP) 7.3 is a Jakarta EE 8 compatible implementation for both Web Profile and Full Platform specifications and is also a certified implementation of the Java Enterprise Edition (Java EE) 8 specification. Major versions of JBoss EAP are forked from the WildFly community project at certain points when the community project has reached the desired feature completeness level. After that point, an extended period of testing and productization takes place in which JBoss EAP is stabilized, certified, and enhanced for production use. During the lifetime of a JBoss EAP major version, selected features may be cherry-picked and back-ported from the community project into a series of feature enhancing minor releases within the same major version family.

JBoss EAP provides preconfigured options for features such as high-availability clustering, messaging, and distributed caching. It also enables users to write, deploy, and run applications using the various APIs and services that JBoss EAP provides.

JBoss EAP includes a modular structure that allows service enabling only when required, improving startup speed. The web-based management console and management command line interface (CLI) make editing XML configuration files unnecessary and add the ability to script and automate tasks. In addition, JBoss EAP includes APIs and development frameworks for quickly developing secure and scalable Java EE applications.

Table 2.1. Features of JBoss EAP

Feature	Description
Jakarta EE compatible	Jakarta EE 8 compatible implementation for both Web Profile and Full Platform specifications.
Java EE compliant	Java Enterprise Edition 8 full platform and Web Profile certified.
Managed Domain	Centralized management of multiple server instances and physical hosts, while a standalone server allows for a single server instance. Per-server group management of configuration, deployment, socket bindings, modules, extensions, and system properties. Centralized and simplified management of application security (including security domains).
Management console and management CLI	New domain or standalone server management interfaces. The management CLI also includes a batch mode that can script and automate management tasks. Directly editing the JBoss EAP XML configuration files is not recommended.
Simplified directory layout	The modules directory contains all application server modules. The domain and standalone directories contain the artifacts and configuration files for domain and standalone deployments respectively.

Feature	Description
Modular class-loading mechanism	Modules are loaded and unloaded on demand. This improves performance, has security benefits, and reduces start-up and restart times.
Streamlined datasource management	Database drivers are deployed like other services. In addition, datasources are created and managed using the management console and management CLI.
Unified security framework	Elytron provides a single unified framework that can manage and configure access for both standalone servers and managed domains. It can also be used to configure security access for applications deployed to JBoss EAP servers.

2.2. SUBSYSTEMS

Many of the APIs and capabilities that are exposed to applications deployed to JBoss EAP are organized into subsystems. These subsystems can be configured by administrators to provide different behavior, depending on the goal of the application. For instance, if an application requires a database, a datasource can be configured in the **datasources** subsystem and accessed by that application after it is deployed to that JBoss EAP server or domain.

2.3. HIGH AVAILABILITY

High availability (HA) in JBoss EAP refers to multiple JBoss EAP instances working together to provide applications that are more resistant to fluctuations in traffic, server load, and server failure. HA incorporates concepts such as scalability, load balancing, and fault tolerance.

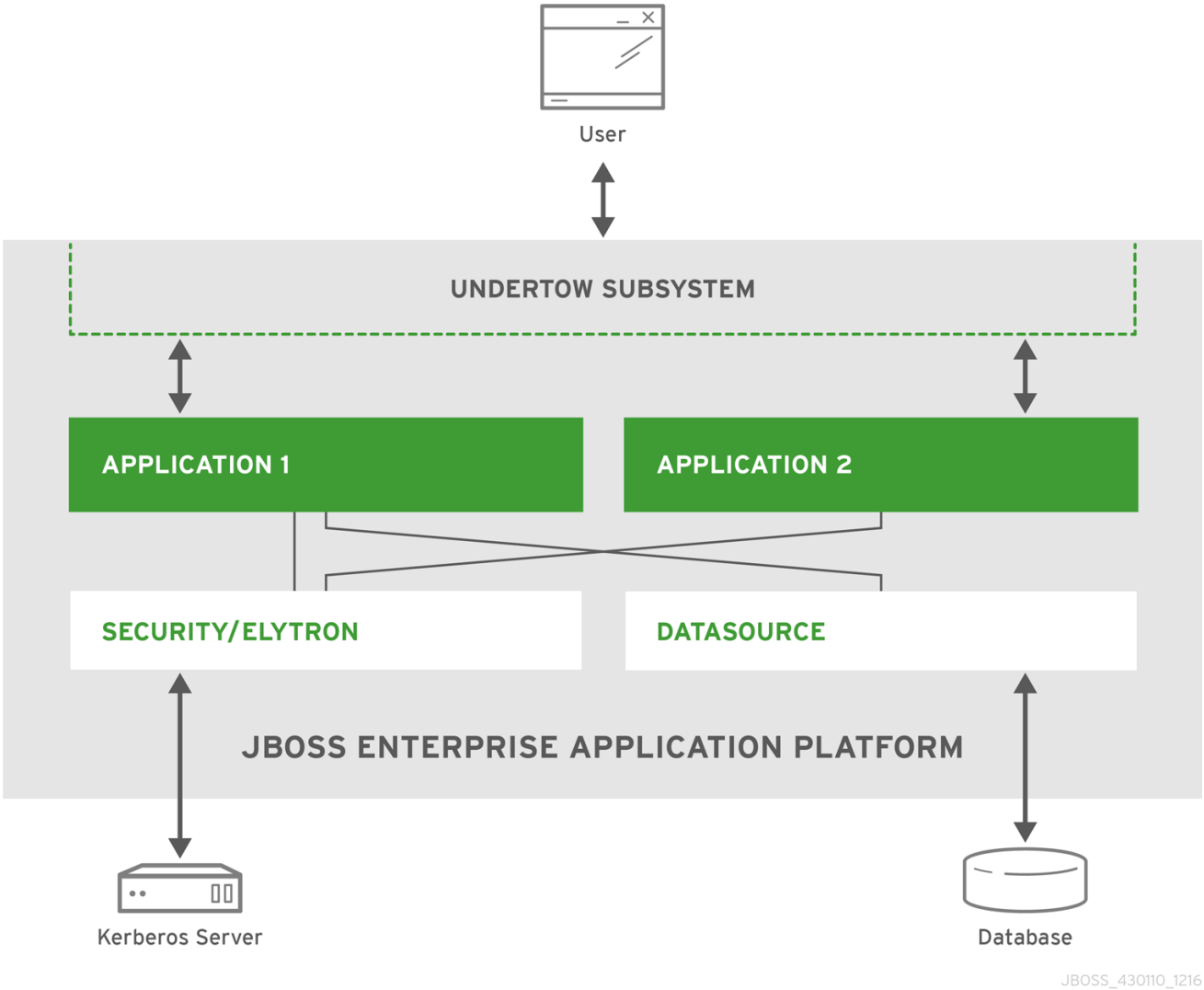
2.4. OPERATING MODES

In addition to providing functionality and APIs to its applications, JBoss EAP has powerful management capabilities. These management capabilities differ depending on which operating mode is used to start JBoss EAP. JBoss EAP offers a *standalone server* operating mode for managing discrete instances and a *managed domain* operating mode for managing groups of instances from a single control point.

CHAPTER 3. EXAMPLES

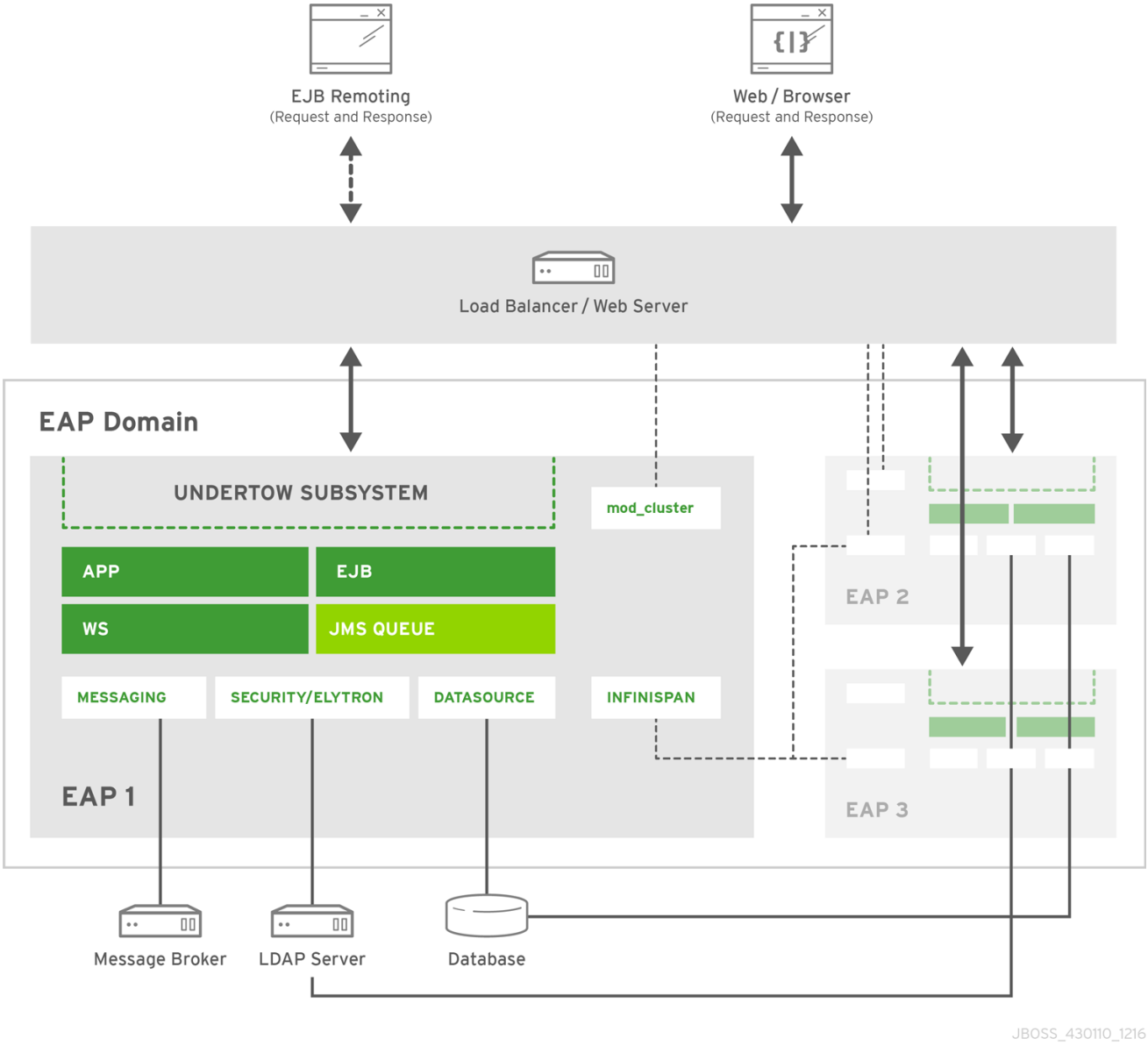
Below are several examples to illustrate how JBoss EAP works and where it fits into different environments.

3.1. SIMPLE EXAMPLE



This example shows a simple JBoss EAP setup. The JBoss EAP instance has two applications deployed to it. It is also configured to connect to a database using the **datasources** subsystem and a Kerberos server which can use either the legacy **security** subsystem or the **elytron** subsystem. These connections are exposed to the deployed applications. The JBoss EAP instance handles requests through the **undertow** subsystem and directs those requests to the appropriate application. The applications use the APIs exposed by JBoss EAP to connect to the database and Kerberos server, and perform their implemented business logic. After completion, the applications send a response back to the requester through the **undertow** subsystem.

3.2. EXPANDED EXAMPLE



This example illustrates a more complex setup involving three JBoss EAP instances arranged in a managed domain with either a load balancer or a web server. The three instances are also configured to support high availability through load balancing using mod_cluster and session replication using Infinispan. All three JBoss EAP instances have a web application, a web service, and EJB deployed. One JBoss EAP instance has a JMS queue configured through the **messaging-activemq** subsystem. All three JBoss EAP instances have connections to a database through the datasource. They also have a connection to the LDAP server using either the legacy **security** subsystem or the **elytron** subsystem. In addition, one JBoss EAP instance is configured to connect to an external message broker through the **messaging-activemq** subsystem. Those configured connections are exposed to the applications, web services, EJBs, and JMS queues deployed to that respective instance.

All inbound requests intended for the application, web service, or EJB are first received by the load balancer or web server. Based on the configured load balancing algorithm and the information provided by each JBoss EAP instance, the web server or load balancer directs that request to the appropriate JBoss EAP instance. The JBoss EAP instance handles requests through the **undertow** subsystem and directs those requests to the appropriate application. The applications use the APIs exposed by JBoss EAP to connect to the database and Kerberos server, and perform their implemented business logic. After completion, the applications send a response back to the requester through the **undertow** subsystem. Any non-persisted information, for example session information, is propagated among the JBoss EAP instances through the **infinispan** subsystem.

Revised on 2020-03-18 10:11:09 UTC