RED HAT®
CONSULTING

# Fiserv EAP 7 Architecural Review

Toufic Arabi

# Table of Contents

# Purpose

Fiserv has engaged Red Hat to assist them with an architectural review of their current Weblogic High Availability (HA) - Load Balance + Failover - setup with a desire to migration away from the Oracle provided Enterprise Edition container to Red Hat's latest EAP 7 Java EE container.

In response to Fiserv's request, Red Hat engaged with Fiserv in an architectural review, to assist them in defining how their current architectural requirements can be moved into Red Hat JBoss EAP 7 running on Red Hat Enterprise Linux 7.

In the rest of this document we will start by showing Fiserv's current architectural design, their target architecture and attempt to provide a process by which the Red Hat JBoss EAP 7 containers can be deployed and configured in the most automated fashion.

# Current Architecture

# 1. Situational Awareness

The systems engineering team at Fiserv is responsible for setting the IT standards that the rest of the business units at Fiserv must follow. This also means that the container that is approved for usage is driven by the systems engineering team and must meet HA and Load Balancing capabilities and criteria before it is approved for development, QA and production usage.

Fiserv's systems engineering team provisions its VMs using VMware technologies as their hypervisors. They are currently running Weblogic server on Red Hat Linux and that was the result of a previous migration from Solaris to RHEL 7. Lower Fiserv environments as expect have fewer Weblogic containers (as expected) than higher environments closer to production. Below we present a lower - development - environment and a prod like environment architecture setup where each Weblogic admin server manages a separate domain:

# 2. Architecture of Lower and Prod Like Environments

The following two images show the architecture of two main types of environments that Fiserv's Systems Engineering team supports for its clients. The non production architecture

The non production envrionment is usually used for development and testing. In both a non production like environment and a prod like environment the following constraints exist:

1. The current Apache HTTPD and F5 load balancers must be in the DMZ
2. The load balancers must be highly available
3. There has to be two clusters load balanced by a set of two different
4. In a maintenance type scenario, one load balancer set would be taken out of load for Weblogic server maintenance and the other set succombs to the same process once the maintenance is completed on the first set
5. The Weblogic servers and clusters sit in the protected area where firewall restrictions are less strict
6. One Weblogic admin server managers both Weblogic Clusters and resides in the protected area

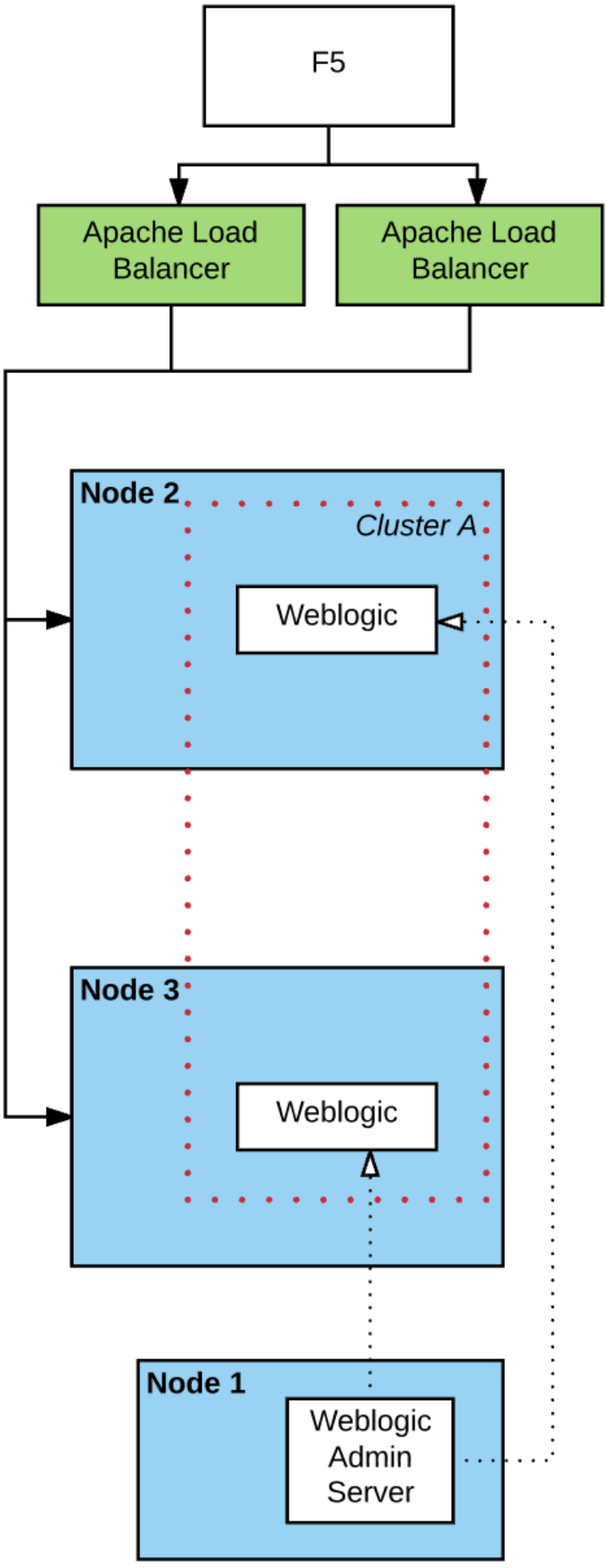
F5
Apache Load Balancer
Apache Load Balancer
Node 2
Cluster A
Weblogic
Node 3
Weblogic
Node 1
Weblogic Admin Server

*Figure 1. Fiserv Non Production Architecture Per Business Unit*

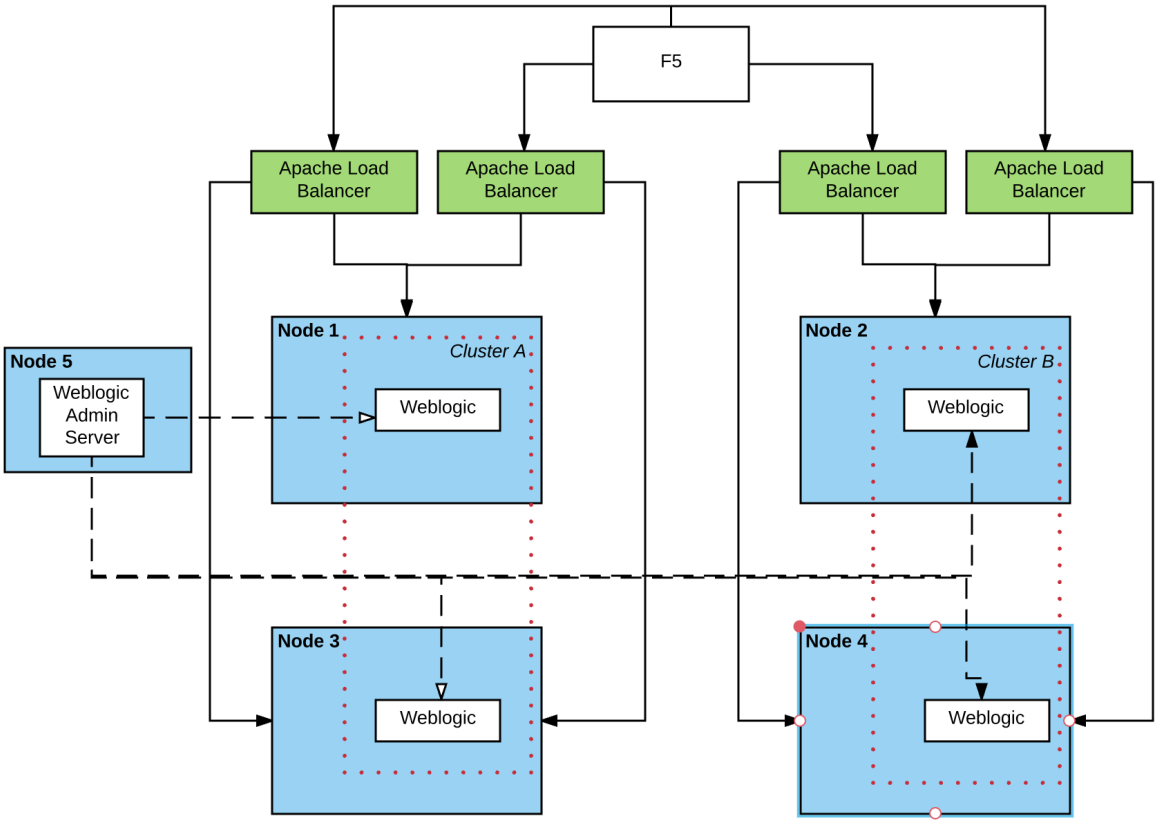


*Figure 2. Fiserv Production Architecture Per Business Unit*

# Introduction to High Availability

# 3. Definitions and Clarification

The words "High Availability", "Failover", and "clustering" are being used interchangeably nowadays and that causes architectural conversations to be mislead. It is important that we define what these terms means and provide a proper context on what they offer when using them for a Java EE application and with regards to the JBoss EAP container.

JBoss EAP provides the following high availability services to guarantee the availability of deployed Java EE applications.

1. **Load balancing** : This allows a service to handle a large number of requests by spreading the workload across multiple servers. A client can have timely responses from the service even in the event of a high volume of requests.
2. **Failover** : This allows a client to have uninterrupted access to a service even in the event of hardware or network failures. If the service fails, another cluster member takes over the client's requests so that it can continue processing. Clustering is a term that encompasses all of these capabilities. Members of a cluster can be configured to share workloads (load balancing) and pick up client processing in the event of a failure of another cluster member (failover).

JBoss EAP supports high availability at several different levels using various components. Some of those components of the runtime and your applications that can be made highly-available are:

1. Instances of the application server Web applications, when used in conjunction with the internal JBoss Web Server, Apache HTTP Server, Microsoft IIS, or Oracle iPlanet Web Server.
2. Stateful and stateless session Enterprise JavaBeans (EJBs) Single sign-on (SSO) mechanisms
3. HTTP sessions
4. JMS services and message-driven beans (MDBs)
5. Singleton MSC services
6. Singleton deployments

# 4. Clustering JBoss EAP 7

**Clustering** is made available to JBoss EAP by the JGroups, Infinispan, and mod_cluster subsystems. The ha and full-ha profiles have these systems enabled. In JBoss EAP, these services start up and shut down on demand, but they will only start up if an application configured as distributable is deployed on the servers. Clustering is the act of grouping servers together to act as a single entity. We have to be clear and understand that clustering servers is not really failover, and its defnitely not load balancing. The act of using a single group of servers to achieve load balancing, failover and redundancy is clusteing, giving us then a Highly Available architecture.

## 4.1. JGroups

JGroups is a toolkit for reliable messaging and can be used to create clusters whose nodes can send messages to each other. The JGroups subsystem provides group communication support for high availability services in JBoss EAP. It allows you to configure named channels and protocol stacks as well as view runtime statistics for channels. The JGroups subsystem is available when using a configuration that provides high availability capabilities, such as the ha or full-ha profile in a managed domain, or the standalone-ha.xml or standalone-full-ha.xml configuration file for a standalone server. JBoss EAP is preconfigured with two JGroups stacks:

1. udp

The nodes in the cluster use User Datagram Protocol (UDP) multicasting to communicate with each other. This is the default stack.

1. tcp

The nodes in the cluster use Transmission Control Protocol (TCP) to communicate with each other. Although TCP tends to be slower than UDP, there are use cases for it, for example, when UDP is not available in a certain environment. You can use the preconfigured stacks or define your own to suit your system's specific requirements.

## 4.2. Infinispan

Infinispan is a Java data grid platform that provides a JSR-107-compatible cache interface for managing cached data. The Infinispan subsystem provides caching support for JBoss EAP. It allows you to configure and view runtime metrics for named cache containers and caches. When using a configuration that provides high availability capabilities, such as the ha or full-ha profile in a managed domain, or the standalone-ha.xml or standalone-full-ha.xml configuration file for a standalone server, the Infinispan subsystem provides caching, state replication, and state distribution support. In non-high-availability configurations, the Infinispan subsystem provides local caching support.

Infinispan is delivered as a private module in JBoss EAP to provide the caching capabilities of JBoss EAP. Infinispan is not supported for direct use by applications.

Clustering can be configured in two different ways in JBoss EAP using Infinispan. The best method for your application will depend on your requirements. There is a trade-off between availability, consistency, reliability and scalability with each mode. Before choosing a clustering mode, you must identify what are the most important features of your network for you, and balance those

### 4.2.1. Cache Modes

**Replicated**

Replicated mode automatically detects and adds new instances on the cluster. Changes made to these instances will be replicated to all nodes on the cluster. Replicated mode typically works best in small clusters because of the amount of information that has to be replicated over the network. Infinispan can be configured to use UDP multicast, which alleviates network traffic congestion to a degree.

**Distributed**

Distributed mode allows Infinispan to scale the cluster linearly. Distributed mode uses a consistent hash algorithm to determine where in a cluster a new node should be placed. The number of copies (owners) of information to be kept is configurable. There is a trade-off between the number of copies kept, durability of the data, and performance.

The more copies that are kept, the more impact on performance, but the less likely you are to lose data in a server failure. The hash algorithm also works to reduce network traffic by locating entries without multicasting or storing metadata.

You should consider using distributed mode as a caching strategy when the cluster size exceeds 6-8 nodes. With distributed mode, data is distributed to only a subset of nodes within the cluster, as opposed to all nodes.

### 4.2.2. Synchronous and Asynchronous Replication

Replication can be performed either in synchronous or asynchronous mode, and the mode chosen depends on your requirements and your application.

**Synchronous replication**

With synchronous replication, the thread that handles the user request is blocked until replication has been successful. When the replication is successful, a response is sent back to the client, and only then is the thread is released. Synchronous replication has an impact on network traffic because it requires a response from each node in the cluster. It has the advantage, however, of ensuring that all modifications have been made to all nodes in the cluster.

**Asynchronous replication**

With asynchronous replication, Infinispan uses a thread pool to carry out replication in the background. The sender does not wait for replies from other nodes in the cluster. However, cache reads for the same session will block until the previous replication completes so that stale data is not read. Replication is triggered either on a time basis or by queue size. Failed replication attempts are written to a log, not notified in real time.

# Implementing High Availability with JBoss EAP 7 & Undertow

# 5. Load Balancing with Undertow

1. Load Balancing With Undertow
    1. undertow as a dynamic load balancer  using default u get static LB, and using HA u get dynamic with mod cluster doing dynamic is better: https://www.quora.com/What-is-the-difference-between-static-balancing-and-dynamic-balancing

the filters have to be created on the undertow subsystem in the load balancer container

the multicast is done on both the LB and the app containers in mod cluster subsystem and u configure the shared key on both the lb and the app server

1. undertow remains a part of the domain

# 6. Clustering with JGroups and Infinispan

1. clustering with jgroups and infinispan
    a. basic cluster with UDP , can switch to TCP with -→ put link here The change we made to the domain.xml on the domain controller to get UDP clustering to work, is to alter the full-ha profile and change this line from

```
<socket-binding name="jgroups-udp" interface="private" port="55200" multicast-address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
```

TO

```
<socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
```

The same can be accomplished via CLI - please make sure you are editing the right sockets group - full-ha-sockets

/socket-binding-group=full-ha-sockets/socket-binding=jgroups-udp:undefine-attribute(name=interface)

a. clustering EJBs: https://access.redhat.com/solutions/136963
b. clustering messaging subsystem, data rep vs shared store: http://www.mastertheboss.com/jboss-server/jboss-jms/jms-clustering-in-wildfly-and-jboss-eap
c. http session replication - </distributable> -→ https://access.redhat.com/solutions/24898

```
https://access.redhat.com/sites/default/files/attachments/eap7_1.pdf
--> failover of the above resources
```

1. Failover of the domain controller: https://access.redhat.com/solutions/1247783

# Desired Future Architecture

## 7. title

# Installation Mechanisms & Configuration Considerations

# 8. title