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. section title

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:

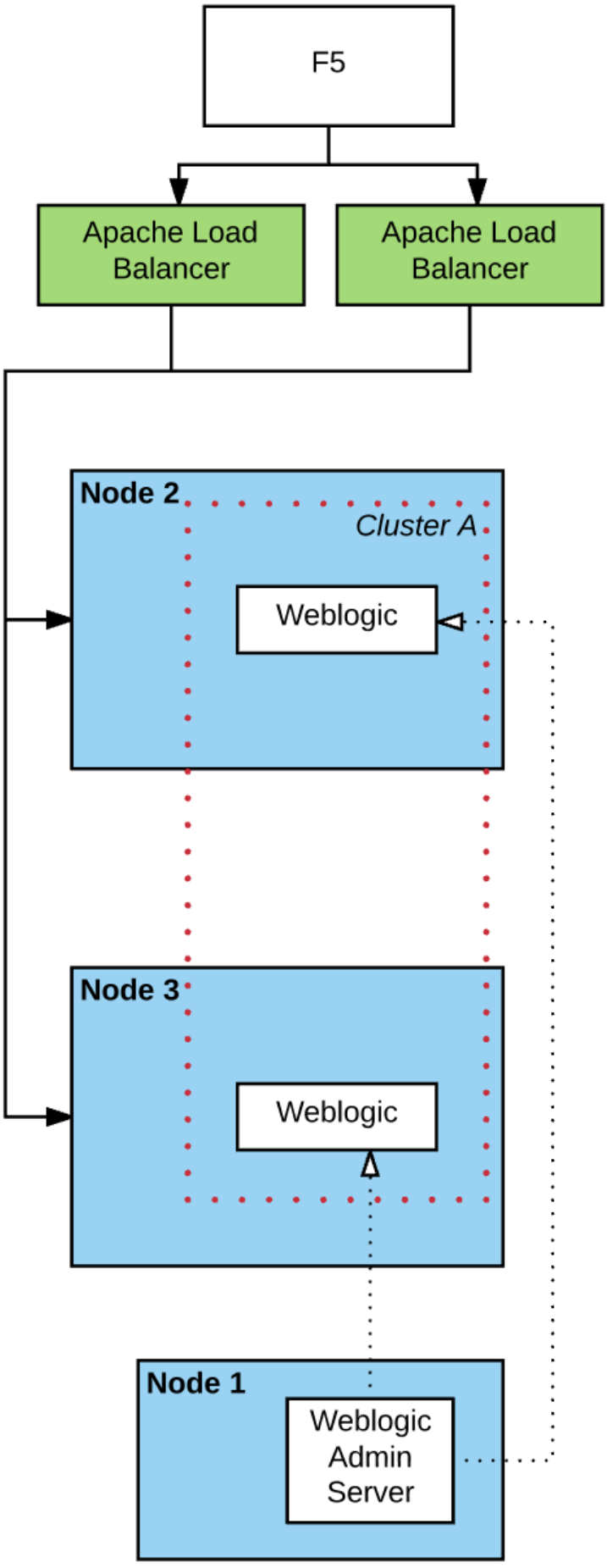
F5
Apache Load Balancer
Apache Load Balancer
Node 2
Cluster A
Weblogic
Node 3
Weblogic
Node 1
Weblogic Admin Server

Figure 1. Fiserv Non Production Architecture Per Business Unit

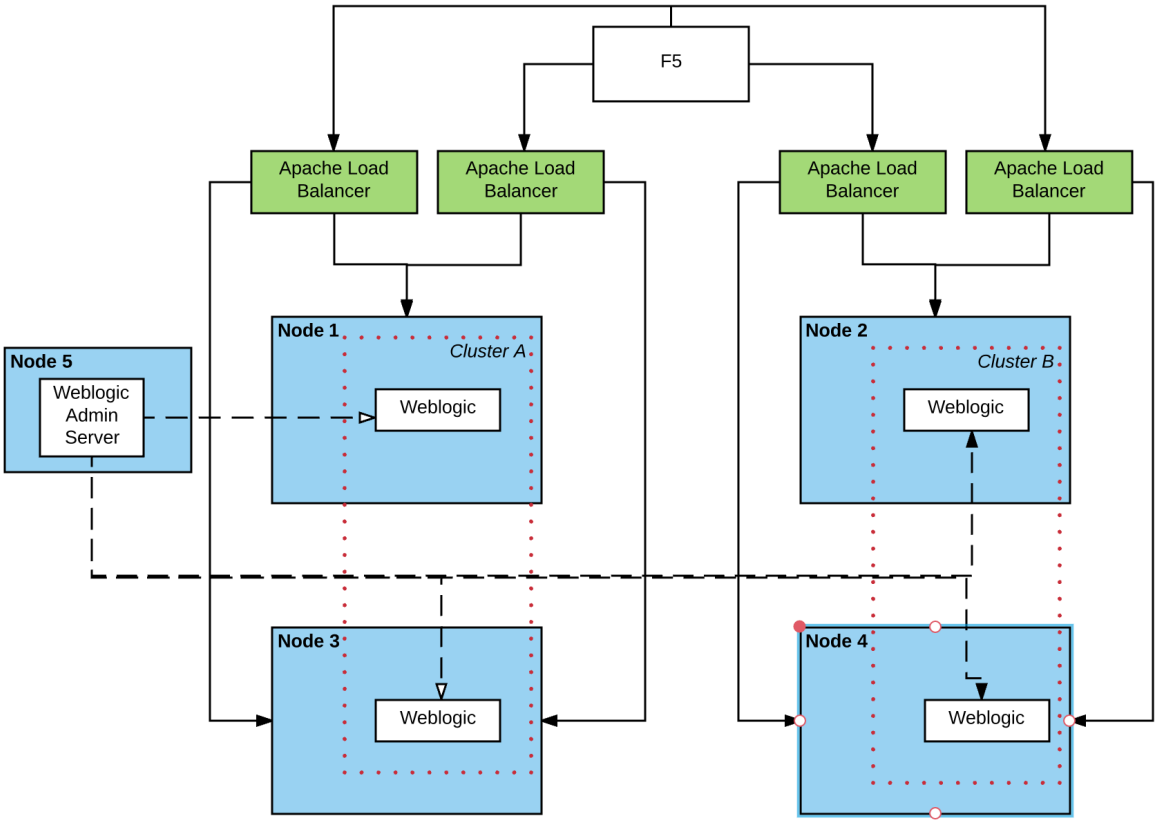

Figure 2. Fiserv Production Architecture Per Business Unit

Introduction to High Availability

2. HA

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

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.

There are two main caching policies using Infinispan that can setup: distributed vs replicated. You are highly encouraged to read about these strategies in the JBoss EAP 7 configuration guide and the SYNC vs ASYNC mode they can be configured with.

Implementing High Availability with JBoss EAP 7 & Undertow

3. section here

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
2. clustering with jgroups and infinispan
    a. basic cluster with UDP , can switch to TCP with -→ put link here
    b. clustering EJBs: https://access.redhat.com/solutions/136963
    c. clustering messaging subsystem, data rep vs shared store: http://www.mastertheboss.com/jboss-server/jboss-jms/jms-clustering-in-wildfly-and-jboss-eap
    d. http session replication - </distributable> -→ https://access.redhat.com/solutions/24898

    ```
    https://access.redhat.com/sites/default/files/attachments/eap7_1.pdf
    ```

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

Desired Future Architecture

4. title

Installation Mechanisms & Configuration Considerations

5. title