Red Hat JBoss Enterprise Application Platform 7.2

Using JBoss EAP in Microsoft Azure

For Use with Red Hat JBoss Enterprise Application Platform 7.2

Last Updated: 2020-10-09

Red Hat JBoss Enterprise Application Platform 7.2 Using JBoss EAP in Microsoft Azure

For Use with Red Hat JBoss Enterprise Application Platform 7.2

Legal Notice

Abstract

This book is a guide for using Red Hat JBoss Enterprise Application Platform in Microsoft Azure, including high-availability configuration.

Table of Contents

CHAPTER 1. INTRODUCTION

JBoss EAP 7 can be used with the Microsoft Azure platform, as long as you use it within the specific supported configurations for running JBoss EAP in Azure. If you are configuring a clustered JBoss EAP environment, you must apply the specific configurations necessary to use JBoss EAP clustering features in Azure.

NOTE

You can also run JBoss EAP 7.2 on Azure App Service, a managed hosting service for web and API applications, as public preview. For more information, see Public Preview: A managed JBoss EAP experience on Azure.

This guide details the supported configurations of using JBoss EAP in Microsoft Azure, as well as the specific JBoss EAP configuration required to enable JBoss EAP clustering in Azure. All other JBoss EAP features not mentioned in this guide operate normally in Azure as with any other JBoss EAP installation. See the other JBoss EAP documentation for non-Azure-specific configuration instructions.

1.1. ABOUT RED HAT CLOUD ACCESS

Red Hat Cloud Access is a Red Hat subscription feature that provides support for JBoss EAP on Red Hat certified cloud infrastructure providers, such as Amazon EC2 and Microsoft Azure. Red Hat Cloud Access allows you to move your subscriptions between traditional servers and public cloud-based resources in a simple and cost-effective manner.

You can find more information about Red Hat Cloud Access on the Customer Portal .

CHAPTER 2. SUPPORTED CONFIGURATIONS

The only virtual machine operating systems supported for using JBoss EAP in Microsoft Azure are:

- Red Hat Enterprise Linux 7
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016

The Red Hat Cloud Access program allows you to use a JBoss EAP subscription to install JBoss EAP on your own Azure virtual machine or one of the above On-Demand operating systems from the Microsoft Azure Marketplace. Note that virtual machine operating system subscriptions are separate from a JBoss EAP subscription.

Other than the above operating system restrictions, see the Customer Portal for further information on supported configurations for JBoss EAP, such as supported Java Development Kit (JDK) vendors and versions.

2.1. UNSUPPORTED FEATURES

There are some unsupported features when using JBoss EAP in a Microsoft Azure environment.

Managed Domains

JBoss EAP managed domains are not supported in Microsoft Azure. Only standalone JBoss EAP server instances are supported. Note that configuring JBoss EAP clusters using standalone JBoss EAP servers is supported in Azure.

ActiveMQ Artemis High Availability Using a Shared Store

JBoss EAP messaging high availability using Artemis shared stores is not supported in Microsoft Azure. To configure JBoss EAP messaging high availability in Azure, see the instructions in the ActiveMQ Artemis High Availability section.

mod_custer Advertising

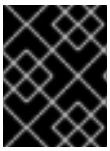
If you want to use JBoss EAP as an Undertow mod_cluster proxy load balancer, the mod_cluster advertisement functionally is unsupported because of Azure UDP multicast limitations. You must configure mod_cluster load balancing to use a proxy list instead of advertising .

CHAPTER 3. CREATING YOUR MICROSOFT AZURE ENVIRONMENT

Create the virtual machines that will host your JBoss EAP instances in your Microsoft Azure environment. The virtual machines must use an Azure size of **Standard_A2** or higher.

You can use either the Azure On-Demand premium images to create your virtual machines or create your own virtual machines manually.

- For Red Hat Enterprise Linux virtual machines, see the instructions on the Customer Portal:
  - Using the On-Demand Marketplace Red Hat Enterprise Linux 7 image in Azure .
  - Manually creating and provisioning a Red Hat Enterprise Linux 7 image for Azure .
- For Microsoft Windows Server virtual machines, see the Microsoft Azure documentation for instructions on creating a Windows Server virtual machine in Microsoft Azure.



IMPORTANT

If you are configuring JBoss EAP high availability, you must create your virtual machines inside the same virtual network.

CHAPTER 4. INSTALLING JBOSS EAP

IMPORTANT

If you are using a Red Hat Enterprise Linux On-Demand virtual machine from the Microsoft Azure Marketplace, you must install JBoss EAP using the ZIP or installer methods. You must not register a Red Hat Enterprise Linux On-Demand virtual machine to Red Hat Subscription Management, as you will be billed twice for that virtual machine.

Installing JBoss EAP on a virtual machine in a Microsoft Azure environment is no different from a normal JBoss EAP installation. See the JBoss EAP *Installation Guide* for instructions.

CHAPTER 5. CONFIGURING JBOSS EAP SUBSYSTEMS TO WORK ON CLOUD PLATFORMS

Some JBoss EAP subsystems must be configured to work properly on cloud platforms, such as Amazon EC2 and Microsoft Azure. This is required because a JBoss EAP server is usually bound to a cloud virtual machine's private IP address, for example: **10.x.x.x**, which is only visible from within the cloud platform. For certain subsystems, this address must also be mapped to a server's public IP address, which is visible from outside the cloud.

5.1. WEB SERVICES

When a client makes a web service request using **Service.create(wsdlURL, serviceName);**, the user connects to the server public IP address, but is subsequently redirected to an address defined in the server configuration files in the **webservices** subsystem. By default, this address is **${jboss.bind.address:127.0.0.1}**, which means that on a cloud platform, the caller will be redirected to the server's private IP address and will be unable to resolve the request. The server's public IP address has to be configured in the **wsdl-host** element, using the following command:

```
/subsystem=webservices:write-attribute(name=wsdl-host,value=PUBLIC_IP_ADDRESS)
```

5.2. MESSAGING

When using messaging on a cloud platform, the connection factory that the client uses must have a connector pointing to the server's public IP address.

For this reason a new connector and socket binding must be created for JBoss EAP servers running a **full** profile.

1. The referenced **http-public** socket binding must be created within the **socket-binding-group**:

   ```
   /socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=http-public:add(host=PUBLIC_IP_ADDRESS,port=${jboss.http.port:8080})
   ```

2. Create the new **http-connector** element in the **messaging** subsystem:

   ```
   /subsystem=messaging-activemq/server=default/http-connector=http-public-connector:add(endpoint=http-acceptor, socket-binding=http-public)
   ```

3. Set the **connectors** in the **connection-factory**, which will be used by clients. For example, configuration of **RemoteConnectionFactory** as the default connection will be:

   ```
   /subsystem=messaging-activemq/server=default/connection-factory=RemoteConnectionFactory:write-attribute(name=connectors, value=["http-public-connector"]
   ```

5.3. REMOTING CONFIGURATION FOR HIGH AVAILABILITY

If you are using JBoss EAP HA features with clustered EJBs on a cloud platform, some extra configuration for the **remoting** subsystem is required to ensure EJB clients can receive cluster view updates.

This is done by configuring **client-mappings** for the **remoting** subsystem socket binding:

```
/socket-binding-group=standard-sockets/socket-binding=http:write-attribute(name=client-
mappings,value=[{ "destination-address" => "PUBLIC_IP_ADDRESS", "destination-port" => "8080" }])
```

CHAPTER 6. CONFIGURING JBOSS EAP FOR MICROSOFT AZURE

Starting JBoss EAP as a Service at Boot on Red Hat Enterprise Linux

If you are starting JBoss EAP as a service on Red Hat Enterprise Linux in Microsoft Azure, you must change the following configuration of the JBoss EAP service to ensure that it starts after the Azure virtual machine host name has been initialized.

Open **/etc/systemd/system/multi-user.target.wants/eap7-standalone.service** and add a dependency on **waagent.service**:

Replace the following line:

```
After=syslog.target network.target
```

with:

```
After=syslog.target network.target waagent.service
Requires=waagent.service
```

Load Balancing with mod_cluster

As a result of mod_cluster advertising not being supported in Azure , if you are configuring a JBoss EAP load balancing environment, you must ensure that all balancers and workers are bound to IP addresses that are accessible on your internal Microsoft Azure virtual network.

NOTE

Balancers and workers must be bound to the actual internal Azure virtual IP address, for example **172.28.0.2**, and not **0.0.0.0**.

Additionally, because of the unavailability of mod_cluster advertising, if you are using JBoss EAP as an Undertow mod_cluster proxy load balancer, you must configure each worker node to use a proxy list containing each balancer's IP address and port, as shown below:

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=proxy-one:add(host="BALANCER_IP_ADDRESS", port="BALANCER_PORT")
/subsystem=modcluster/mod-cluster-config=configuration:list-add(name=proxies,value=proxy-one)
```

ActiveMQ Artemis High Availability

JBoss EAP messaging high availability using Artemis shared stores is not supported in Microsoft Azure. To configure JBoss EAP messaging high availability in Azure, you must use a replicated journal with the AZURE_PING JGroups discovery protocol using a TCP JGroups stack. For more information on using a replicated journal, see Data Replication in *Configuring Messaging* for JBoss EAP.

After configuring the TCP JGroups stack with AZURE_PING, you must add the JGroups stack to the **discovery-group** and **broadcast-group jgroups-cluster** configurations in the **messaging-activemq** subsystem. For details, see Server Discovery in *Configuring Messaging* for JBoss EAP.

CHAPTER 7. USING JBOSS EAP HIGH AVAILABILITY IN MICROSOFT AZURE

Microsoft Azure does not support JGroups discovery protocols that are based on UDP multicast. Although you may use other JGroups discovery protocols (such as a static configuration (**TCPPING**), a shared database (**JDBC_PING**), shared file system-based ping (**FILE_PING**), or **TCPGOSSIP**), we strongly recommend that you use the shared file discovery protocol specifically developed for Azure: **AZURE_PING**.

7.1. CONFIGURING AZURE_PING FOR JBOSS EAP HIGH AVAILABILITY

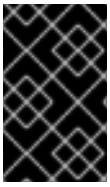
This section describes configuring your JBoss EAP cluster to use the AZURE_PING JGroups discovery protocol. Ensure that you meet the prerequisites when creating your virtual machines.

AZURE_PING uses a common blob container in a Microsoft Azure storage account. If you do not already have a blob container that AZURE_PING can use, create one that your virtual machines can access.

After creating your blob container, you will need the following information to configure AZURE_PING:

- **storage_account_name**: the name of the Microsoft Azure storage account that contains your blob container.
- **storage_access_key**: the secret access key of the storage account.
- **container**: the name of the blob container to use for **PING** data.

To configure JBoss EAP to use AZURE_PING as the JGroups discovery protocol, you can either use a preconfigured example JBoss EAP configuration file, or modify an existing configuration.



IMPORTANT

The following instructions configure AZURE_PING using a UDP JGroups stack. If you will be configuring JBoss EAP messaging high availability in Azure, you must configure AZURE_PING in a TCP JGroups stack instead.

7.1.1. Using the Example Configuration File

JBoss EAP includes example configuration files for configuring clustering of standalone servers in Microsoft Azure. These files are located in **EAP_HOME/docs/examples/configs/** and are **standalone-azure-ha.xml** and **standalone-azure-full-ha.xml**.

NOTE

See the JBoss EAP *Configuration Guide* for an explanation of the differences between the server profiles.

These sample configuration files are preconfigured for using clustering in Microsoft Azure, and all that is needed is to specify the values for your Azure storage account and blob container.

Copy your desired example configuration file to **EAP_HOME/standalone/configuration/**.

7.1.2. Modifying an Existing Configuration

If you are modifying an existing JBoss EAP high availability configuration, the following changes to the **jgroups** subsystem are required.

1. Launch the management CLI and embed a server to make offline changes to your chosen configuration file. For example:

```
$ EAP_HOME/bin/jboss-cli.sh
[disconnected /] embed-server --server-config=standalone-ha.xml
```

2. By default, JGroups uses the UDP stack. If you were using another stack, change back to using the UDP stack:

```
[standalone@embedded /] /subsystem=jgroups/channel=ee:write-
attribute(name=stack,value=udp)
```

3. Execute the following batch of commands to remove the existing UDP stack and insert a new UDP stack configured for Microsoft Azure:

```
batch
/subsystem=jgroups/stack=udp:remove
/subsystem=jgroups/stack=udp:add()
/subsystem=jgroups/stack=udp/transport=UDP:add(socket-binding=jgroups-udp,properties=
{ip_mcast=false})
/subsystem=jgroups/stack=udp/protocol=azure.AZURE_PING:add(properties=
{storage_account_name="${jboss.jgroups.azure_ping.storage_account_name}",
storage_access_key="${jboss.jgroups.azure_ping.storage_access_key}",
container="${jboss.jgroups.azure_ping.container}"})
/subsystem=jgroups/stack=udp/protocol=MERGE3:add
/subsystem=jgroups/stack=udp/protocol=FD_SOCK:add(socket-binding=jgroups-udp-fd)
/subsystem=jgroups/stack=udp/protocol=FD:add
/subsystem=jgroups/stack=udp/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=udp/protocol=pbcast.NAKACK2:add(properties=
{use_mcast_xmit=false,use_mcast_xmit_req=false})
/subsystem=jgroups/stack=udp/protocol=UNICAST3:add
/subsystem=jgroups/stack=udp/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=udp/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=udp/protocol=UFC:add
/subsystem=jgroups/stack=udp/protocol=FRAG2:add
run-batch
```

NOTE

If you want to store the values of your Microsoft Azure storage account and blob container in your configuration file, replace the system property references in the above configuration with the values from your Azure environment. In the following command, examples for starting JBoss EAP, the system properties are used.

The stack XML in your configuration file should look like the following:

```
<stack name="udp">
    <transport type="UDP" socket-binding="jgroups-udp">
        <property name="ip_mcast">
            false
```

```
            </property>
        </transport>
        <protocol type="azure.AZURE_PING">
            <property name="storage_account_name">
                ${jboss.jgroups.azure_ping.storage_account_name}
            </property>
            <property name="storage_access_key">
                ${jboss.jgroups.azure_ping.storage_access_key}
            </property>
            <property name="container">
                ${jboss.jgroups.azure_ping.container}
            </property>
        </protocol>
        <protocol type="MERGE3"/>
        <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
        <protocol type="FD"/>
        <protocol type="VERIFY_SUSPECT"/>
        <protocol type="pbcast.NAKACK2">
            <property name="use_mcast_xmit">
                false
            </property>
            <property name="use_mcast_xmit_req">
                false
            </property>
        </protocol>
        <protocol type="UNICAST3"/>
        <protocol type="pbcast.STABLE"/>
        <protocol type="pbcast.GMS"/>
        <protocol type="UFC"/>
        <protocol type="FRAG2"/>
    </stack>
```

4. Stop the embedded server and exit the management CLI:

```
[standalone@embedded /] stop-embedded-server
[disconnected /] exit
```

7.2. STARTING JBOSS EAP HIGH AVAILABILITY

To start JBoss EAP using high availability in Microsoft Azure, you must:

- use a configuration file that has been configured with the AZURE_PING discovery protocol and specify the required values of your Microsoft Azure storage account and blob container.
- bind the **private** interface to the Microsoft Azure internal IP address that is used for clustering traffic. You can do this at startup, as shown below, or as a set configuration shown in the JBoss EAP *Configuration Guide*.

WARNING

For security reasons, you must ensure that you do not expose clustering traffic to unintended networks.

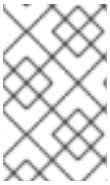
You can do this by restricting the endpoints to your Microsoft Azure virtual network or by creating a dedicated virtual network and dedicated virtual machine NICs for clustering traffic.

Start your JBoss EAP high availability instance using the following command. If you stored your Microsoft Azure storage account and blob container values in your configuration file, you can omit the **-Djboss.jgroups.azure_ping** system property definitions.

```
EAP_HOME/bin/standalone.sh -b IP_ADDRESS -bprivate IP_ADDRESS --server-config=EAP_CONFIG_FILE.xml -Djboss.jgroups.azure_ping.storage_account_name=STORAGE_ACCOUNT_NAME -Djboss.jgroups.azure_ping.storage_access_key=STORAGE_ACCESS_KEY -Djboss.jgroups.azure_ping.container=CONTAINER_NAME
```

For example:

```
EAP_HOME/bin/standalone.sh -b 172.28.0.2 -bprivate 172.28.0.2 --server-config=standalone-azure-ha.xml -Djboss.jgroups.azure_ping.storage_account_name=my_storage_account -Djboss.jgroups.azure_ping.storage_access_key=y7+2x7P68pQse9MNh58Bkk5po9OGzeJc+0IRqYcQ9Cr/Sp4xiUFJVIbY+MGXJRNx3syksikwm4tOYIFgjvoCmw== -Djboss.jgroups.azure_ping.container=my_blob_container
```



NOTE

As JBoss EAP subsystems only start when needed, you must deploy a distributable application to your JBoss EAP servers to start the high availability JBoss EAP subsystems.

After you start a second JBoss EAP instance in a cluster, you should see logs similar to the following in the console log of the first server in the cluster:

```
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-2,ee,eap-server-1) ISPN000094: Received new cluster view for channel server: [eap-server-1|1] (2) [eap-server-1, eap-server-2]
```

7.3. TROUBLESHOOTING JBOSS EAP HIGH AVAILABILITY

If you are having trouble getting clustering to work in Microsoft Azure, verify that you have completed all the requirements in the following list.

- Ensure that the Microsoft Azure virtual machines hosting your JBoss EAP servers are using the same virtual network.

- Ensure that you have a blob container for AZURE_PING to use.
- Ensure that you are using a JBoss EAP configuration file with the AZURE_PING discovery protocol configured in the **jgroups** subsystem.
- Ensure that you are binding both the public and private interfaces to the correct Microsoft Azure IP addresses.
- Ensure that you have the correct values for your Microsoft Azure storage account and blob container and that you are either:
  - storing them in your configuration file, or
  - setting the correct system properties when starting JBoss EAP.
- Ensure that you have your distributable Java application deployed to all the JBoss EAP servers in your cluster.

7.3.1. Cleaning Stale Discovery Files in Your Blob Container

If a JBoss EAP cluster that uses AZURE_PING is shut down abnormally, for example, using **kill -9** to end the JBoss EAP process, some stale discovery files may be left in your blob container.

These files are usually cleaned up in a graceful cluster shutdown, but if left there from an abnormal shutdown, it may impact startup performance of cluster members attempting to contact nodes that are no longer online.

If this is a problem for you, you can set the following configuration to make the cluster coordinator remove and refresh all discovery files whenever the cluster view changes.

```
/subsystem=jgroups/stack=udp/protocol=azure.AZURE_PING/property=remove_all_files_on_view_char
ge:add(value=true)
```

NOTE

Alternatively, if cleaning your container on each view change is not ideal, you can reduce the number of join attempts for a node attempting to join a cluster. The default number of join attempts is **10**. For example, to set the number of join attempts to **3**:

```
/subsystem=jgroups/stack=udp/protocol=pbcast.GMS/property=max_join_attempts:add(v
alue=3)
```

The stale discovery files will still be present, but a node attempting to join a cluster will not spend as much time attempting to contact nodes that are no longer online.

Revised on 2020-10-09 09:38:41 UTC