Lab Work: Deploying Clustered Applications

In this lab work, you will set up a cluster of EAP servers in a managed domain and implement a cluster-aware application.

Resources	
Files	/opt/domain /opt/standalone /tmp/cluster.war /tmp/new-tcp-stack.cli
app url	http://172.25.250.254:9080/cluster

Result

You should be able to configure a cluster of EAP server instances and deploy a cluster-aware application to test load balancing and switching.

before you start

Use the following command to download the relevant lab files, create the required lab folders, and ensure that the managed domain is set up correctly:

```
[student@workstation ~]$ lab clustering-lab-final setup
```

You can use the EAP 7 management console or the JBoss EAP CLI to achieve your goals, keeping in mind that the EAP CLI is the preferred option in production environments.

An EAP administrator has configured a managed domain with two host controllers running the servera and serverb virtual machines, respectively, and the domain controller on the workstation. The domain and host configuration files are stored in the /opt/domain folder on all three machines. You will start the managed domain and set up a two-node cluster (servera.1 and serverb.1).
You'll also run a mod_cluster load balancer on a separate server instance, running on a workstation virtual machine.

1. **Open several ports on the workstation, servera, and serverb virtual machines for this lab work. Briefly review the firewall settings of the workstation virtual machine in the /tmp/jgroups-firewall-rules workstation.sh file.**

   Run the following script on the workstation virtual machine:

   ```
   [student@workstation ~]$ sudo sh /tmp/jgroups-firewall-rules-workstation.sh
   ```

Open a new terminal on the server virtual machine. Briefly review the server virtual machine's firewall settings in the /tmp/jgroups firewall-rules-servera.sh file.

Run the following script on the server virtual machine:

```
[student@servera ~]$ sudo sh /tmp/jgroups-firewall-rules-servera.sh
```

Open a new terminal on the serverb virtual machine. Briefly review the serverb virtual machine's firewall settings in the /tmp/jgroups firewall-rules-servera.sh file.

Run the following script on the serverb virtual machine:

```
[student@serverb ~]$ sudo sh /tmp/jgroups-firewall-rules-serverb.sh
```

2. Verify that the ports on the firewall are open by running the following command on all three (ALL) virtual machines:

```
[student@workstation ~]$ firewall-cmd --list-all --zone=public
```

3. Start the domain controller on the workstation virtual machine. Because domain controller configuration files are kept in the /opt/domain folder on workstation, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also notice that the host file for the domain controller is named host-master.xml and is located in the /opt/domain/configuration folder. (Tip: Pass the --host-config=host-master.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so you must start the domain controller using sudo -u jboss /opt/jboss-eap-7.0/ bin/domain.sh ...

4. Start the load balancer instance on the workstation virtual machine. The setup script created the load balancer configuration in the /opt/standalone folder on workstation. Use /opt/standalone as the value of the jboss.server.base.dir argument that you pass to the standalone.sh startup script.
Also note that it will start load balancing with standalone-ha.xml and a port offset of 1000 to avoid port collisions with the domain controller running on the workstation. Since the load balancer will be the entry point for the application, it must be configured to listen on the public IP of the workstation (172.25.250.254). (Hint: Send the argument -c standalone-ha.xml -Djboss.socket.binding.port-offset=1000 - Djboss.bind.address=172.25.250.254 to standalone.sh.)

Note that the /opt/standalone directory is owned by the jboss user, so you should start the load handler using sudo -u jboss /opt/jboss eap-7.0/bin/standalone.sh ...

5. You have been instructed to set up a cluster of two EAP server instances with the following following specifications:

- You must use the dynamic load balancer based on mod_cluster and the load balancer must load balancing requests between the servers in Group1 (servera.1 and serverb.1).
- The cluster.war application must be deployed on the Group1 server group.
- TCP-based cluster storage must be configured for the cluster (for communication between clusters between EAP server instances).
- Auto-discovery based on the front-end load balancer's default UDP of back-end EAP server instances should be disabled.
  EAP server instances must be configured with a static list of proxies (load balancers).
- The failure of a server instance must be handled transparently and the Cluster test application should continue to run without interruption (high availability and failover).

5.1. Set up a TCP-based cluster of EAP servers in the domain managed.
Use the EAP CLI to define a new TCP stack configuration. Open the /tmp/new-tcp-stack.cli file on the workstation virtual machine and review the commands listed there. Note that this must be edited as the jboss user.

Edit the initial_hosts property and replace the values with the hostname and ports of the two EAP server instances that should be part of the cluster.

5.2. Run the EAP CLI script file on the domain controller.

5.3. Run the EAP CLI and connect to the domain controller to configure the servers in the managed domain.

5.4. Configure the mod_cluster subsystem. By default, EAP is set to advertise its status to load balancers using UDP multicast.
Disable publishing to the mod_cluster subsystem for the full-ha profile.

5.5. Because you disabled publishing, you need to configure the backend nodes from EAP with a list of proxies (load balancers). Configure the EAP back-end nodes to communicate with the load balancer running on the workstation virtual machine. Be sure to add an outgoing socket binding that indicates the load balancer's IP address and port (172.25.250.254:9080).

5.6. To change the default stack, you must reload the configuration of the managed domain. Reload the domain controller using the EAP CLI.

6. Configure the load balancer.

6.1. Run the EAP CLI and connect to the load balancer instance on workstation.

6.2. Configure the modcluster subsystem to act as a front-end load balancer. Set a unique password for advertise-security-key.

6.3. Configure the mod_cluster filter. Publish the load balancer using the modcluster socket binding and use the HTTP protocol for management socket binding. Make sure the security-key attribute matches the advertise-security-key you configured in the previous step.

6.4. As a final step, bind the modcluster filter to the undertow default-server.

6.5. Reload the load balancer configuration.

7. The two host controllers on servera and serverb connect to the host controller domain and get the latest configuration of the domain. Start the two host controllers on servera and serverb.

7.1. Start the host controller on servera. Because the host controller configuration files are kept in the /opt/domain folder on servera, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also notice that the host file for the host controller is named host-slave.xml and is located in the /opt/domain/configuration folder. (Hint: Pass the --host config=host-slave.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so start the host controller using sudo -u jboss /opt/jboss eap-7.0/bin/domain.sh ...

7.2. Start the host controller on serverb. Because the host controller configuration files are kept in the /opt/domain folder on serverb, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also notice that the host file for the host controller is named host-slave.xml and is located in the /opt/domain/configuration folder. (Hint: Pass the --host config=host-slave.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so start the host controller using sudo -u jboss /opt/jboss eap-7.0/bin/domain.sh ...

7.3. Verify that both host controllers connect to the domain controller and form a managed domain. Look at the console window in which you started the domain controller and verify that both servera and serverb are registered as slaves to the domain controller.

7.4. Observe the terminal window of the load balancer and verify that this Record the two server instances (nodes) in Group1 that you are using in this lab work.

8. Deploy and test the cluster test application.

8.1. Using the EAP CLI, stop the servers in the Group2 server group, because They are not used in this lab work.

8.2. Deploy the cluster test application to the Group1 server group. It is available in /tmp/cluster.war on the workstation virtual machine.

8.3. Observe the load balancer's terminal window and verify that both server instances, servera.1 and serverb.1, have registered with the load balancer and are ready to serve the /cluster context.

8.4. Navigate to the load balancer at http://172.25.250.254:9080/cluster via a browser on the workstation virtual machine. You should see the cluster application. Refresh the browser several times and notice that each request is processed by the same server (due to session stickiness). Determine which server instance is handling your current request by looking at the JBoss EAP node name is in the application tag.

8.5. Stop the host from actively responding to requests by entering Ctrl+C in the appropriate terminal window to completely shut down the host controller.

8.6. Observe the load balancer's terminal window and verify that the unregistered the server you terminated and the /cluster context for that load balancer server instance.

8.7. Return to the web browser and refresh the page. The load balancer fails its request to the remaining server without losing the current hits value.

9. Perform cleaning and grading.

9.1. Press Ctrl+C in the terminal window in which you started the drivers host and domain controller to stop the managed domain. (Alternatively, you can shut down the domain controller using the JBoss EAP CLI command /host=master:shutdown()).

9.2. Press Ctrl+C to shut down the load balancer instance in the dialog window. terminal where you started it.

9.3. Press Ctrl+C to exit the EAP CLI if you used the CLI in your lab work. (Alternatively, you can leave the CLI by typing exit.)

9.4. Run the following command on workstation to grade the assignment:

```
[student@workstation ~]$ lab clustering-lab-final grade
```

This concludes the lab work.

Solution

In this lab work, you will set up a cluster of EAP servers in a managed domain and implement a cluster-aware application.

Resources	
Files	/opt/domain /opt/standalone /tmp/cluster.war /tmp/new-tcp-stack.cli
app url	http://172.25.250.254:9080/cluster

Result

You should be able to configure a cluster of EAP server instances and deploy a cluster-aware application to test load balancing and switching.

before you start

Use the following command to download the relevant lab files, create the required lab folders, and ensure that the managed domain is set up correctly:

```
[student@workstation ~]$ lab clustering-lab-final setup
```

You can use the EAP 7 management console or the JBoss EAP CLI to achieve your goals, keeping in mind that the EAP CLI is the preferred option in production environments.

An EAP administrator has configured a managed domain with two host controllers running the servera and serverb virtual machines, respectively, and the domain controller on the workstation. The domain and host configuration files are stored in the /opt/domain folder on all three machines. You will start the managed domain and set up a two-node cluster (servera.1 and serverb.1).
You will also run a mod_cluster load balancer on a separate server instance, running on a workstation virtual machine.

1. Open several ports on the workstation, servera, and serverb virtual machines for this lab work. Briefly review the firewall settings of the workstation virtual machine in the /tmp/jgroups-firewall-rules workstation.sh file.

Run the following script on the workstation virtual machine:

```
[student@workstation ~]$ sudo sh /tmp/jgroups-firewall-rules-workstation.sh
```

Open a new terminal on the server virtual machine. Briefly review the server virtual machine's firewall settings in the /tmp/jgroups firewall-rules-servera.sh file.

Run the following script on the server virtual machine:

```
[student@servera ~]$ sudo sh /tmp/jgroups-firewall-rules-servera.sh
```

Open a new terminal on the serverb virtual machine. Briefly review the serverb virtual machine's firewall settings in the /tmp/jgroups firewall-rules-servera.sh file.

Run the following script on the serverb virtual machine:

```
[student@serverb ~]$ sudo sh /tmp/jgroups-firewall-rules-serverb.sh
```

2. Verify that the ports on the firewall are open by running the following command on all three (ALL) virtual machines:

```
[student@workstation ~]$ firewall-cmd --list-all --zone=public
```

3. Start the domain controller on the workstation virtual machine. Because domain controller configuration files are kept in the /opt/domain folder on workstation, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also note that the host file for the domain controller is named host-master.xml and is located in the /opt/domain/configuration folder. (Tip: Pass the --host-config=host-master.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so you must start the domain controller using sudo -u jboss /opt/jboss-eap-7.0/ bin/domain.sh ...

```
[student@workstation ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/domain.sh \ -Djboss.domain.base.dir=/
opt/domain/ --host-config=host-master. xml
```

4. Start the load balancer instance on the workstation virtual machine. The setup script created the load balancer configuration in the /opt/standalone folder on workstation. Use /opt/standalone as the value of the jboss.server.base.dir argument that you pass to the standalone.sh startup script.
Also note that it will start load balancing with standalone-ha.xml and a port offset of 1000 to avoid port collisions with the domain controller running on the workstation. Since the load balancer will be the entry point for the application, it must be configured to listen on the public IP of the workstation (172.25.250.254). (Hint: Send the argument -c standalone-ha.xml -Djboss.socket.binding.port-offset=1000 - Djboss.bind.address=172.25.250.254 to standalone.sh.)

Note that the /opt/standalone directory is owned by the jboss user, so you should start the load handler using sudo -u jboss /opt/jboss eap-7.0/bin/standalone.sh ...

```
[student@workstation ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/standalone.sh \ -Djboss.server.base.dir=/
opt/standalone/ -Djboss.bind.address=172.25.250.254 \ -Djboss.socket.binding.port-offset=1000 -c
standalone-ha.xml
```

5. You have been instructed to set up a cluster of two EAP server instances with the following following specifications:

- **You must use the dynamic load balancer based on mod_cluster and the load balancer must load balancing requests between the servers in Group1 (servera.1 and serverb.1).**
- **The cluster.war application must be deployed on the Group1 server group.**
- **TCP-based cluster storage must be configured for the cluster (for communication between clusters between EAP server instances).**
- **Auto-discovery based on the front-end load balancer's default UDP of back-end EAP server instances should be disabled.**
  EAP server instances must be configured with a static list of proxies (load balancers).
- **The failure of a server instance must be handled transparently and the Cluster test application should continue to run without interruption (high availability and failover).**

5.1. Set up a TCP-based cluster of EAP servers in the domain managed.
Use the EAP CLI to define a new TCP stack configuration. Open the /tmp/ new-tcp-stack.cli file on the workstation virtual machine and review the commands listed there. Note that this must be edited as the jboss user.

Edit the initial_hosts property and replace the values with the hostname and ports of the two EAP server instances that should be part of the cluster.

```
..."initial_hosts":add(value="servera[7600],serverb[7600]")
```

5.2. Run the EAP CLI script file on the domain controller.

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ sudo -u jboss \ /opt/jboss-
eap-7.0/bin/jboss-cli.sh \ --connect --
controller=172.25.250.254:9990 \ --file=/tmp/new-tcp-
stack.cli The batch executed successfully

...
```

5.3. Run the EAP CLI and connect to the domain controller to configure the servers in the managed domain.

In a new terminal window on workstation, start the EAP CLI and connect to the domain controller as the jboss user:

```
[student@workstation ~]$ sudo -u jboss \ /opt/jboss-
eap-7.0/bin/jboss-cli.sh \ --connect --
controller=172.25.250.254:9990
```

5.4. Configure the mod_cluster subsystem. By default, EAP is set to advertise its status to load balancers using UDP multicast.
Disable publishing to the mod_cluster subsystem for the full-ha profile.

```
[domain@172.25.250.254:9990 /]/profile=full-ha/subsystem=modcluster\ /mod-cluster-
config=configuration/\ :write-
attribute(name=advertise,value=false)
```

5.5. Since you disabled publishing, you need to configure the EAP back-end nodes with a list of proxies (load balancers). Configure the EAP back-end nodes to communicate with the load balancer running on the workstation virtual machine. Be sure to add an outgoing socket binding that indicates the load balancer's IP address and port (172.25.250.254:9080).

```
[domain@172.25.250.254:9990 /]/socket-binding-group=full-ha-sockets\ /remote-destination-
outbound-socket-binding=lb:\ add(host=172.25.250.254,port=9080)
```

Next, add the proxies to the mod_cluster configuration:

```
[domain@172.25.250.254:9990 /]/profile=full-ha/subsystem=modcluster\ /mod-cluster-
config=\ configuration:list-
add(name=proxies,value=lb)
```

5.6. To change the default stack, you must reload the configuration of the managed domain. Reload the domain controller using the EAP CLI.

```
[domain@172.25.250.254:9990 /] reload --host=master
```

6. Configure the load balancer.

6.1. Run the EAP CLI and connect to the load balancer instance on workstation.

In a new terminal window on the workstation, start the EAP CLI and connect to the load balancer as the jboss user:

```
[student@workstation ~]$ sudo -u jboss \ /opt/jboss-
eap-7.0/bin/jboss-cli.sh \
--connect --controller=localhost:10990
```

6.2. Configure the modcluster subsystem to act as a front-end load balancer. Set a unique password for advertise-security-key.

```
[standalone@localhost:10990 /] /subsystem=modcluster/mod-cluster-config=\ configuration:write-
attribute\ (name=advertise-security-key,
value=redhat)
```

6.3. Configure the mod_cluster filter. Publish the load balancer using the modcluster socket binding and use the HTTP protocol for socket binding

management socket. Make sure the security-key attribute matches the advertise-security-key you configured in the previous step.

```
[standalone@localhost:10990 /] /subsystem=undertow/configuration=\
filter/mod-cluster=modcluster:add\
(management-socket-binding=http, advertise-socket-binding=modcluster,\ security-
 key=redhat)
```

6.4. As a final step, bind the modcluster filter to the undertow default-server.

```
[standalone@localhost:10990 /] /subsystem=undertow/server=\ default-server/
host=default-host/filter-ref=modcluster:add
```

6.5. Reload the load balancer configuration.

```
[standalone@localhost:10990 /] :reload
```

7. The two host controllers on servera and serverb connect to the host controller domain and get the latest configuration of the domain. Start the two host controllers on servera and serverb.

7.1. Start the host controller on servera. Because the host controller configuration files are kept in the /opt/domain folder on servera, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also notice that the host file for the host controller is named host-slave.xml and is located in the /opt/domain/configuration folder. (Hint: Pass the --host config=host-slave.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so start the host controller using sudo -u jboss /opt/jboss eap-7.0/bin/domain.sh ...

Open a new terminal window on the server virtual machine and run the following command:

```
[student@servera ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/domain.sh \
-Djboss.domain.base.dir=/opt/domain/ \
-Djboss.domain.master.address= 172.25.250.254 \ --host-
config=host-slave.xml
```

7.2. Start the host controller on serverb. Because the host controller configuration files are kept in the /opt/domain folder on serverb, use /opt/domain as the value of the jboss.domain.base.dir argument that you pass to the domain.sh startup script. Also notice that the host file for the host controller is named host-slave.xml and is located in the /opt/domain/configuration folder. (Hint: Pass the --host config=host-slave.xml argument to domain.sh.)

Note that the /opt/domain directory is owned by the jboss user, so start the host controller using sudo -u jboss /opt/jboss eap-7.0/bin/domain.sh ...

Open a new terminal window on the serverb virtual machine and run the following command:

```
[student@serverb ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/domain.sh \
-Djboss.domain.base.dir=/opt/domain/ \
-Djboss.domain.master.address= 172.25.250.254 \ --host-
config=host-slave.xml
```

7.3. Verify that both host controllers connect to the domain controller and form a managed domain. Look at the console window in which you started the domain controller and verify that both servera and serverb are registered as slaves to the domain controller.

7.4. Observe the terminal window of the load balancer and verify that this Record the two server instances (nodes) in Group1 that you are using in this lab work.

The load balancer registers the two nodes, servera.1 and serverb.1

```
Registering node servera:servera.1, connection: ajp://172.25.250.10:8009/?# Registering node
serverb:serverb.1, connection: ajp://172.25.250.11:8009/?#
```

8. Deploy and test the cluster test application.

8.1. Using the EAP CLI, stop the servers in the Group2 server group, because They are not used in this lab work.

```
[domain@172.25.250.254:9990 /] /server-group=Group2:stop-servers(blocking=true)
```

8.2. Deploy the cluster test application to the Group1 server group. It is available in /tmp/cluster.war on the workstation virtual machine.

```
[domain@172.25.250.254:9990 /] deploy /tmp/cluster.war \ --server-
groups=Group1
```

8.3. Observe the load balancer's terminal window and verify that both server instances, servera.1 and serverb.1, have registered with the load balancer and are ready to serve the /cluster context.

```
Registering context /cluster, for node servera:servera.1 Registering context /
cluster, for node serverb:serverb.1
```

8.4. Navigate to the load balancer at http://172.25.250.254:9080/ cluster via a browser on the workstation virtual machine. You should see the cluster application. Refresh the browser several times and notice that each request is processed by the same server (due to session stickiness). Determine which server instance is handling your current request by looking at the JBoss EAP node name is in the application tag.

8.5. Stop the host from actively responding to requests by entering Ctrl+C in the appropriate terminal window to completely shut down the host controller.

8.6. Observe the load balancer's terminal window and verify that the unregistered the server you terminated and the /cluster context for that load balancer server instance.

```
Unregistering context /cluster, from node servera:servera.1 Removing node
servera:servera.1
```

8.7. Return to the web browser and refresh the page. The load balancer fails its request to the remaining server without losing the current hits value.

9. Perform cleaning and grading.

9.1. Press Ctrl+C in the terminal window in which you started the drivers host and domain controller to stop the managed domain. (Alternatively, you can shut down the domain controller using the JBoss EAP CLI command / host=master:shutdown()).

9.2. Press Ctrl+C to shut down the load balancer instance in the dialog window. terminal where you started it.

9.3. Press Ctrl+C to exit the EAP CLI if you used the CLI in your lab work. (Alternatively, you can leave the CLI by typing exit.)

9.4. Run the following command on workstation to grade the assignment:

```
[student@workstation ~]$ lab clustering-lab-final grade
```

This concludes the lab work.

Summary

In this chapter, you learned the following:

- **A clustered application has the following benefits: ÿ High availability**
  - **ÿ Scalability**
  - **ÿ Failover**
  - **ÿ Error tolerance**
- **Clustered storage is available in EAP using the default settings in the ha or full-ha server configuration files.**
- **Clustered servers can run as servers standalone or in a managed domain.**
- **A cluster is obtained through the Infinispan and JGroups subsystems.**
- **For an application to be distributable, a <distributable> tag must be included in jboss-web.xml.**
- **The JGroups subsystem provides communication between servers stored in clusters. using one of the preconfigured stacks, UDP and TCP.**
- **The Infinispan subsystem provides caching support to maintain high availability.**
- **A cache container is a repository for caches. There are four cache containers defaults: ÿ web**
  - **ÿ hibernate**
  - **ÿ ejb**
  - **ÿ server**
- **A network load balancer directs HTTP requests to EAP server instances that are cluster members.**
- **Session affinity is a mechanism that forwards all requests coming from the same user to the same web server.**
- **mod_cluster creates a dynamic list of cluster members without configuration. manual, but can also be configured to use a static list.**
- **Undertow server can be used as load balancer using mod_cluster as a static or dynamic load balancer.**
- **Undertow uses the session ID cookie to provide persistent sessions.**
- **Alternative load balancers to Undertow are available for use and are supported by Red Hat.**

- A singleton is a design pattern in which a single instance of an object is shared by the entire application and can be configured to be stored in clusters. The singleton implementation and services are configured in the singleton subsystem.