redhat. TRAINING

CHAPTER 12

IMPLEMENTATION OF CLUSTER APPLICATIONS

General description	
Meta	Configure different subsystems to support clustered application deployment. •
Goals	Describe the benefits of clusters and the JBoss EAP subsystems that support clustered application deployments. • Configure Infinispan and JGroups to support clusters. • Configure Undertow as a load balancer. • Configure and implement a highly available application.
sections	• Exploration of clustered applications (and quiz) • Configuring subsystems that support clustered applications (and guided exercise) • Load balancing configuration (and guided exercise) • Implementation of HA Singleton applications (and questionnaire)
Laboratory work	• Deployment of clustered applications

Clustered application exploration

Goals

After completing this section, students should be able to do the following:

- **Describe the benefits of clusters and the JBoss EAP subsystems they support. clustered application deployments.**
- **Describe examples of clustered application topologies.**

clustered applications

A cluster is a grouping of servers that communicate with each other in ways that improve the availability of services by providing the following capabilities:

- **High availability (HA): a service has a very high probability of being available.**
- **Scalability: A service can respond to a large number of requests. requests by distributing the workload across multiple servers.**
- **Failover: If a service fails, the client can continue to process its tasks while another member of the cluster handles the client's requests.**
- **Fault tolerance: a server can guarantee its correct behavior even if a failure occurs.**

The most common way to achieve scalability and high availability is to use the following together:

- **Load Balancer – Often a piece of hardware or a service such as Apache httpd.**
- **Data replication services: a service like memcached or a framework (Infinispan).**

A cluster is made available to an EAP instance through two subsystems: jgroups and infinispan. The ha and full-ha profiles have these subsystems enabled by default.

In EAP 6, a cluster is like a group of servers that are identically configured and that communicate with each other to ensure the provision of HA, failover, and other cluster capabilities. Although users can run in standalone or domain managed mode, these operating modes only dictate how the servers are managed. Clustered servers can be configured in EAP in both modes of operation.

On a standalone server, since each EAP instance has exactly one server, a cluster is a grouping of running EAP instances:

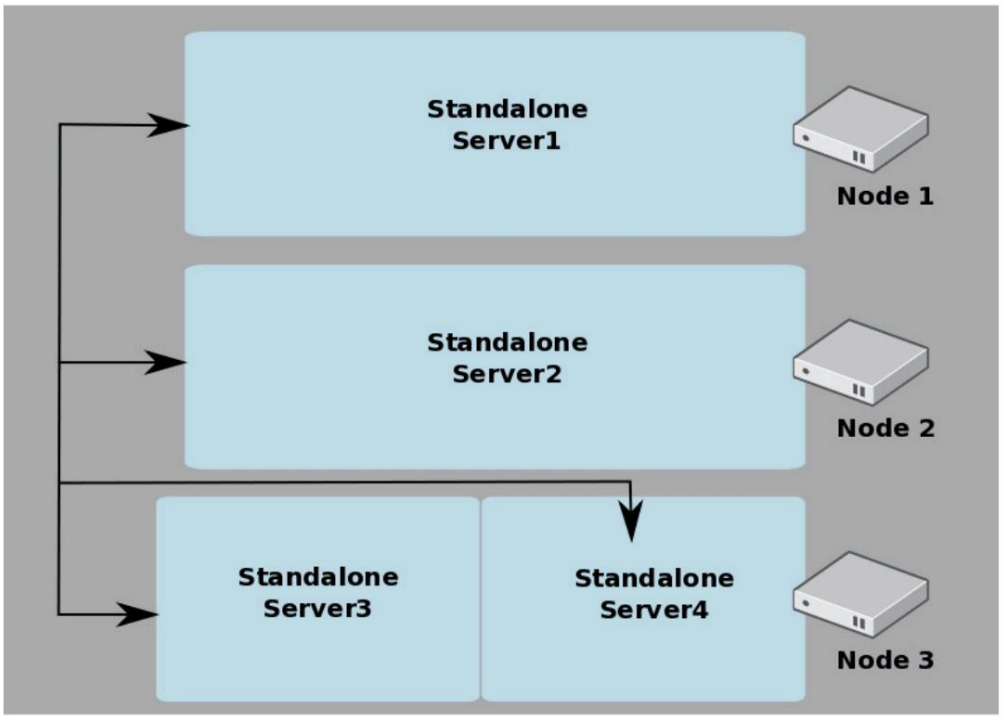


Figure 12.1: Standalone servers in a cluster.

In a managed domain, a cluster is a grouping of servers in a server group, and each server in the server group represents the cluster nodes:

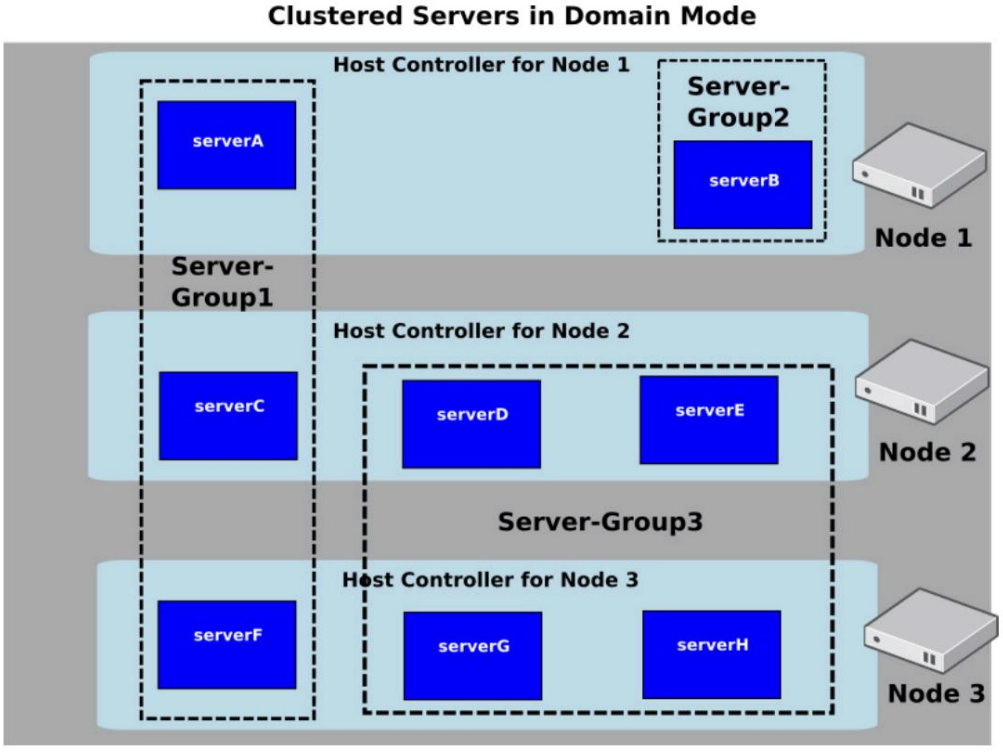


Figure 12.2: Clustered servers in a managed domain.

The diagram above is an example of a clustered managed domain. There are three nodes hosting eight combined servers that are divided into three different groups.

use

In a managed domain, a cluster is a grouping of servers. Also, a cluster can consist of two or more server groups, and the servers in the clustered server groups represent the cluster nodes. As can be seen in the graph above, users can have multiple clusters in a domain and have servers in a non-cluster pool.

Architecture of a cluster

A cluster is obtained through the Infinispan and JGroups subsystems.

• Infinispan – Architecture used for object caching and also for object replication between caches. The Infinispan subsystem provides caching, state replication, and state distribution support.

• JGroups: framework for nodes to communicate with each other using UDP or TCP.

use

Infinispan and JGroups are two great frameworks that contain many capabilities and configuration settings. This section requires only the basics of Infinispan and JGroups. Further configuration of these subsystems is beyond the scope of this course.

use

JBoss has a supported version of EAP with Infinispan backed by a NoSQL database called JBoss Data Grid (JDG). For more details, see Red Hat's JB450 course.

Make an application distributable

To take advantage of the high availability features of cluster servers, an application must first be marked as distributable. To do this, include the <distributable> tag in the web.xml for the application. The following is an example of the web.xml of an application that is configured for clustered storage:

```
<?xml version="1.0"?>
<web-app ...>
<distributable/>
</web-app>
```

Also, users can further tune replication by overriding the default settings in the application's jboss-web.xml file by adding

a <replication-config> tag. The following is an example configuration that defines full session replication (note that this is the true default behavior for EAP):

```
<jboss-web ...>
    <replication-config>
        <replication-granularity>SESSION</replication-granularity> </replication-config>
    </jboss-web>
```

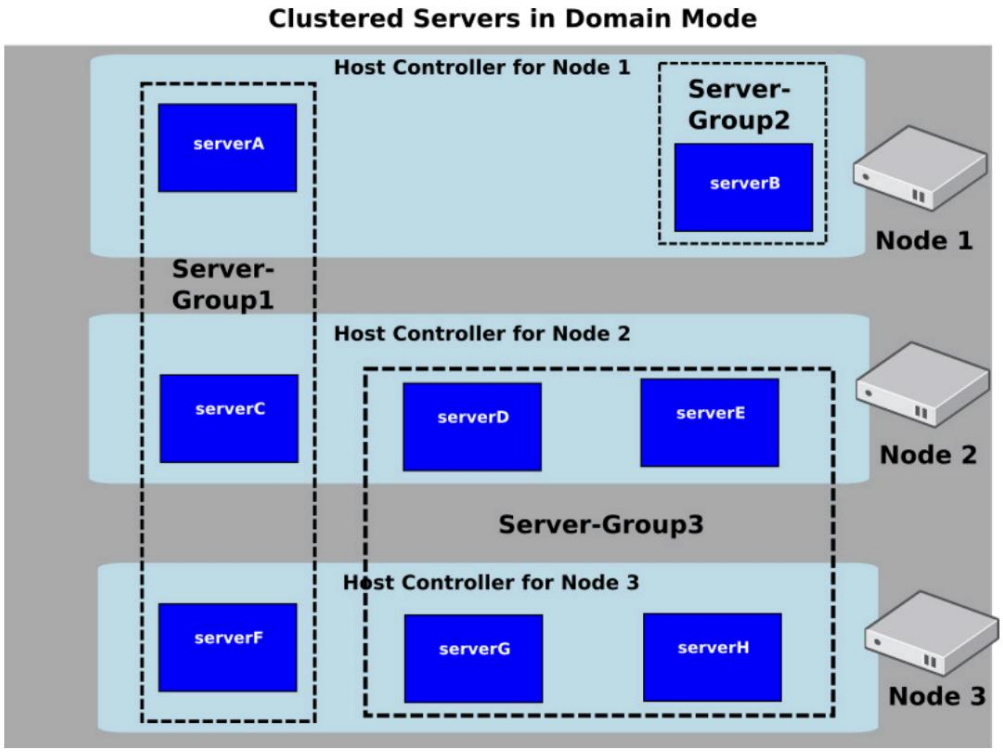
There are two possible values for <replication-granularity>: •

SESSION – The default option. A session level of replication granularity implies that the entire session object is replicated whenever any attribute changes.

• ATTRIBUTE – Only attributes changed in a session are replicated.

Quiz: Exploring Clustered Applications

In managed domain mode, a cluster is a grouping of servers in a server group, and each server in the server group represents the cluster nodes:



Choose the correct answer to the following questions, based on the graph of clustered servers in a managed domain:

1. How many server groups are in this domain? (Choose one option).

to. One b.
Two
c. Three
d. Four
It is. None

2. How many servers are in Server-Group1? (Choose one option).

to. One b.
Of the
c. Three
d. Four
It is. None

3. If a distributable application is deployed to Server-Group1, and Server Group1 uses the ha profile, do you create a cluster between serverA, serverC and serverF? (Choose one option).

a. **Yes. When the distributable application is deployed to the pool, they will automatically create a cluster.**

b. **Yes. But cluster storage needs to be manually enabled on the server side.**

c. **No. The server needs additional configuration in the *-ha profile.**

d. **No. The application needs to be deployed on an individual server, not a group of servers.**

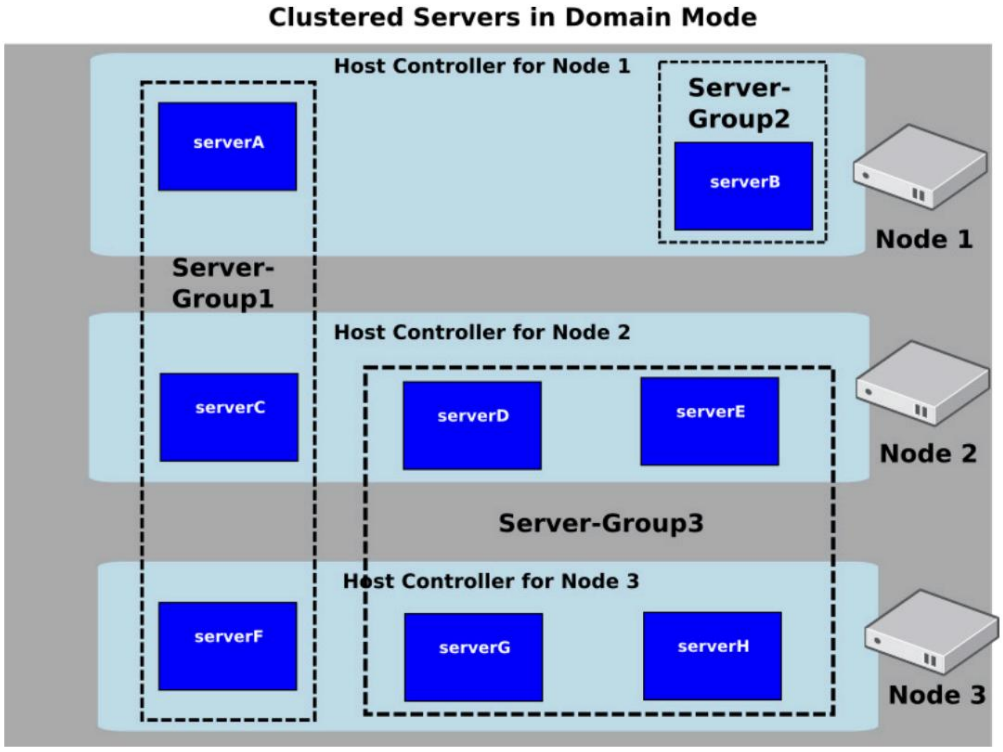
4. Suppose app1 is deployed to Server-Group1 and app3 is deployed to Server-Group3 (and both apps are distributable). What happens to those applications if node 3 fails? (Choose one option).

to. Both applications will report errors to users. b. app1 will continue to work by failing over to serverA and serverC. app3 will fail, since it is not a cluster and has no servers available.

c. **The serverA and serverC instances continue to function, so requests sent to app1 will fail over to either of them. Similarly, serverD and serverE will handle failover of requests sent to app3. d. app3 will continue to work by failing over to serverE and serverD. app1 will fail as it is not a cluster and has no servers available.**

Solution

In managed domain mode, a cluster is a grouping of servers in a server group, and each server in the server group represents the cluster nodes:



Choose the correct answer to the following questions, based on the graph of clustered servers in a managed domain:

1. How many server groups are in this domain? (Choose one option).

to. One **b.**
Of the
c. Three
d. Four
It is. None

2. How many servers are in Server-Group1? (Choose one option).

to. One **b.**
Of the
c. Three
d. Four
It is. None

3. If a distributable application is deployed to Server-Group1, and Server Group1 uses the ha profile, do you create a cluster between serverA, serverC and serverF? (Choose one option).

a. **Yes. When the distributable application is deployed to the pool, they will automatically create a cluster.**

b. Yes. But cluster storage needs to be manually enabled on the server side.

c. No. The server needs additional configuration in the *-ha profile.

d. No. The application needs to be deployed on an individual server, not a group of servers.

4. Suppose app1 is deployed to Server-Group1 and app3 is deployed to Server-Group3 (and both apps are distributable). What happens to those applications if node 3 fails? (Choose one option).

to. Both applications will report errors to users. **b.** app1 will continue to work by failing over to serverA and serverC. app3 will fail, since it is not a cluster and has no servers available.

c. **The serverA and serverC instances continue to function, so requests sent to app1 will fail over to either of them. Similarly, serverD and serverE will handle failover of requests sent to app3.**

d. app3 will continue to work by failing over to serverE and serverD. app1 will fail as it is not a cluster and has no servers available.

Configuring subsystems that support clustered applications

Goals

After completing this section, students should be able to do the following:

• Configure Infinispan and JGroups to support clusters.

• Start a cluster on a stand-alone server or in a managed domain.

Configuring the JGroups subsystem

The *JGroups subsystem* provides all the communication mechanisms that define how servers in a cluster communicate with each other. The EAP platform comes pre-configured with two JGroups stacks:

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

2. tcp: The nodes in the cluster use a Transmission Control Protocol (TCP) to communicate with each other.

Users can use any of these preconfigured stacks, or they can define and use a new stack to suit the specific needs of the environment. By default, the UDP protocol is used to communicate between nodes in clusters on the JGroups ee channel. The following EAP CLI command adjusts the JGroups ee channel to use a tcp stack instead of udp:

```
/subsystem=jgroups/channel=ee:write-
attribute( name=stack,value=tcp)
```

By default, the tcp stack uses multicast to discover other members of a cluster. Users can further customize the tcp stack by changing the protocol to TCPPING or TCPGOSSIP.

• TCPPING – A protocol that uses a static list to define cluster members and uses unicast as an alternative to multicast. The following parameters are specific to this protocol: ÿ initial_hosts – A list of hosts available and known to seek membership of clusters.

ÿ port_range – The range that the protocol will use to search for hosts based on the initial port. For example, a port range of two on an initial port of 7600 causes the TCPPING protocol to look for a viable host on ports 7600 and 7601 to add to the membership.

References

You can find more information about the parameters to customize TCPPING in: Configure

> **TCPPING_stack** https://access.redhat.com/documentation/en/red-hat-jboss-enterprise application-platform/7.0/configuration-guide/chapter-21- configuring-high availability#configure_tcpping_stack

• TCPGOSSIP: Discover members of a cluster using an external gossip router.

The instructor will now demonstrate how to configure the JGroups UDP and TCP stacks using the EAP admin console:

Demo: Configuring the JGroups stack

1. Open a terminal window on the workstation virtual machine (Applications > Favorites > Terminal) and run the following command to create the lab directory, download the files required for the demo, and verify that EAP is installed and not running on this moment:

```
[student@workstation ~]$ demo clustering-jgroups setup
```

2. Users often want to run multiple independent instances without need to install EAP multiple times on the same host. The configure script created two directories to represent the two independent EAP instances, which we will cluster in this demo.

Verify that you can see the following three folders inside the /home/student/JB248/labs/jgroups-cluster1 and /home/student/JB248/labs/jgroups-cluster2 folders:

• configuration

• deployments

• lib

3. You must open several ports on the workstation virtual machine for this demo. Briefly review the firewall settings in the /home/student/JB248/labs/clustering-jgroups/jgroups-firewall-rules .sh file. Run the script as follows:

```
[student@workstation standalone]$ sudo sh \ /home/
student/JB248/labs/clustering-jgroups/jgroups-firewall-rules.sh
```

4. Verify that the ports on the firewall are open by running the following command:

```
[student@workstation standalone]$ firewall-cmd --list-all --zone=public public (default, active)
```

```
  interfaces: eth0 eth1
  sources:
  services: dhcpv6-client http ldap mountd mysql nfs rpc-bind ssh ports: 54300/tcp 55200/
  udp 55300/udp 23364/udp 54200/tcp 7600/tcp 45688/udp 7700/ tcp

  ...
```

5. Run the following command to start the first instance of the EAP server using the standalone.sh script in the original EAP installation, while using the new EAP configuration files from jgroups-cluster1.

Note that we will be clustering the two instances, so you should start the instances with the standalone-full-ha.xml configuration, as this has the relevant JGroups subsystems defined. We also start each instance with a unique node name to avoid clustered subsystem warning messages when running two instances on the same server.

```
[student@workstation standalone]$ cd /opt/jboss-eap-7.0/bin [student@workstation
bin]$ ./standalone.sh \ -Djboss.server.base.dir=/home/
student/JB248/labs/jgroups-cluster1/ \ -Djboss.node.name=jgroups-cluster1 \ -c standalone-
full-ha.xml
```

Verify that the first instance started without errors by looking at the /home/student/JB248/labs/jgroups-cluster1/log/server.log file, using the tail -f command.

use

In the standalone-full-ha.xml file, there is no console handler defined by default, hence the need to use the tail command to view the server logs.

Similarly, start the second instance of the EAP server using the jgroups-cluster2 configuration files. Notice that to avoid port collisions with the first instance, we start the second instance with a port-offset value of 100:

```
[student@workstation standalone]$ cd /opt/jboss-eap-7.0/bin [student@workstation
bin]$ ./standalone.sh \ -Djboss.server.base.dir=/home/
student/JB248/labs/jgroups-cluster2/ \ -Djboss.socket.binding.port-offset=100 \
-Djboss.node.name=jgroups-cluster2 \

-c standalone-full-ha.xml
```

Verify that the second instance started without errors by reviewing the /home/student/JB248/labs/jgroups-cluster2/log/server.log file, using the tail -f command.

6. Observe the log file for jgroups-cluster1 after the jgroups-cluster2 instance starts successfully. You should see the following entries showing that there are now two members in the cluster:

```
2016-05-25 03:08:57,593 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups
cluster1) ISPN000094: Received new cluster view for channel server: [jgroups  cluster1|1] (2) [jgroups-
cluster1, jgroups-cluster2]
2016-05-25 03:08:57,596 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups  cluster1) ISPN000094:
Received new cluster view for channel web: [jgroups-cluster1| 1] (2) [jgroups-cluster1, jgroups-cluster2]

2016-05-25 03:08:57,613 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups  cluster1) ISPN000094:
Received new cluster view for channel ejb: [jgroups-cluster1| 1] (2) [jgroups-cluster1, jgroups-cluster2]

2016-05-25 03:08:57,614 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups  cluster1) ISPN000094:
Received new cluster view for channel hibernate: [jgroups  cluster1|1] (2) [jgroups-cluster1, jgroups-cluster2]
```

use

You may see some WARN messages, such as the following, in the server logs:

```
2016-05-30 13:13:19,132 WARN
  [org.infinispan.topology.ClusterTopologyManagerImpl] (transport
thread--p13-t2) ISPN000197: Error updating cluster member list:
  org.infinispan.util.concurrent.TimeoutException: Replication timeout for jgroups-cluster2
```

This is a recurring issue in the EAP 7 GA release (see https://issues.jboss.org/browse/JBEAP-794) and can be safely ignored. It will be fixed in a later release of EAP 7.

7. On workstation, navigate to http://localhost:9990 to access the EAP administration console for the jgroups-cluster1 instance.

Log in with the username jbossadm and the password JBoss@RedHat123.

8. In the EAP administration console, the JGroups subsystem is configured in the Configuration > Subsystems > JGroups section.

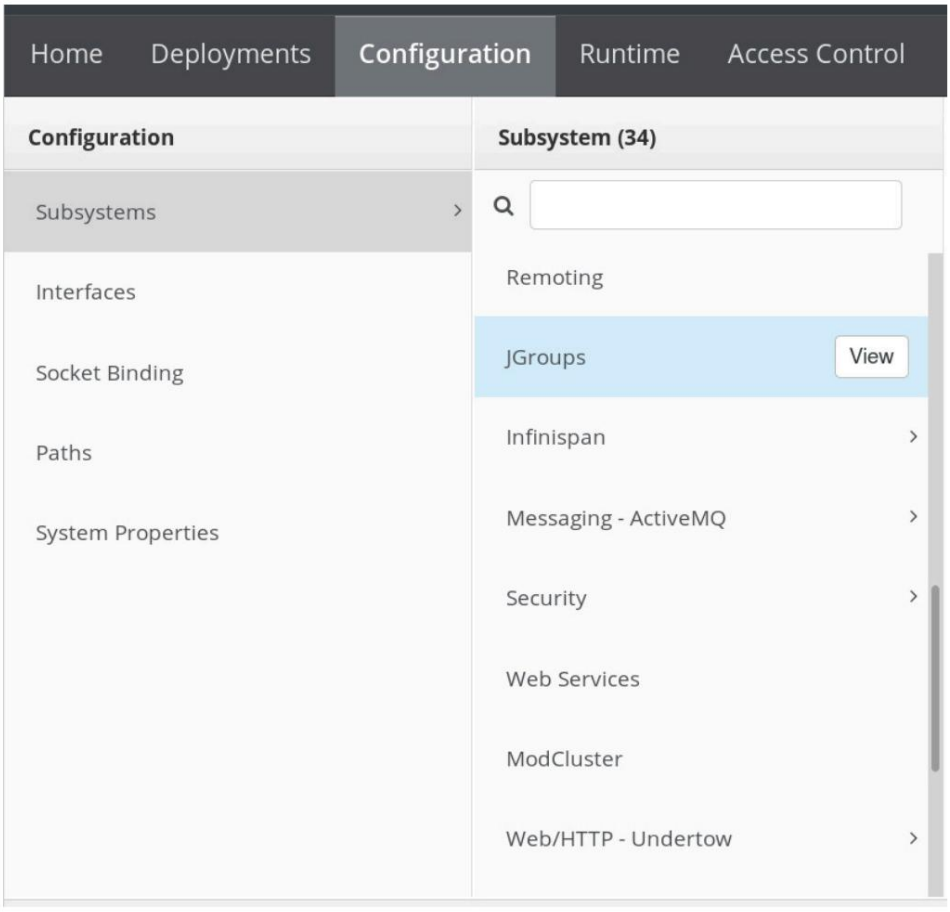


Figure 12.4: JGroups Subsystem

Navigate to Configuration > Subsystems > JGroups in the EAP management console and click View next to JGroups to bring up the JGroups subsystem configuration page.

9. On the JGroups subsystem configuration page, there are two stacks defined so default:

- **udp – This is the default JGroups stack, which uses UDP multicast and auto-discovery to form an EAP cluster.**
- **tcp: JGroups uses the TCP protocol to form an EAP cluster. This is useful in scenarios where UDP traffic is not allowed within the network segment running the EAP cluster. In contrast to the UDP stack, where EAP nodes can be added to a cluster automatically, in the TCP stack configuration, the EAP manager manually manages the IP addresses of cluster nodes.**

10. Click View next to the UDP stack on the subsystem configuration page JGroups, and briefly review the configuration of the UDP stack. DO NOT change any of the values as it is configured for clustering by default.

11. Go back to the Protocol Stack page and click View next to TCP Stack to view the JGroups TCP stack configuration. Briefly review the configuration of the TCP stack. DO NOT change any of the values, as we will be defining a new TCP stack that will configure the cluster with values specific to our demo environment.

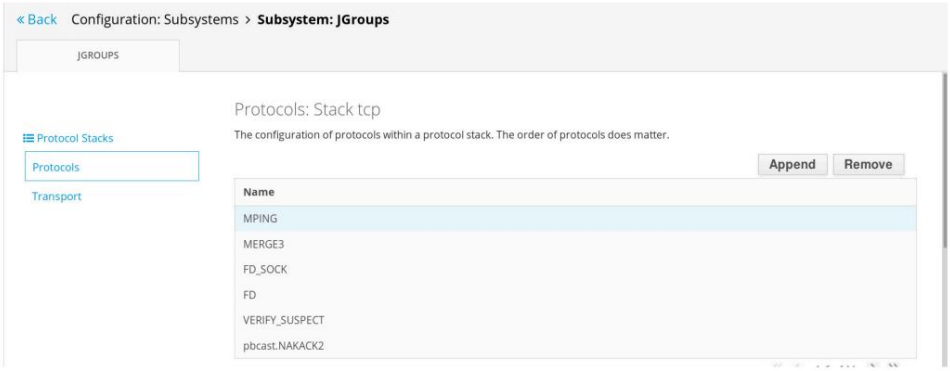


Figure 12.5: JGroups TCP Stack

use

The standalone-full-ha.xml file already defines a tcp stack that uses the TCP protocol to replicate data between EAP nodes, but uses the MPING protocol for automatic node discovery. MPING is based on UDP multicast and may not work in many network environments where UDP multicast traffic is disabled. Therefore, we need to define a new TCP stack that uses a TCP-based protocol called TCPPING, which relies on node discovery using a static list of IP addresses and ports for each of the cluster nodes.

12. Test clustering using the UDP stack configured in a manner default in the standalone-full-ha.xml file. Use the test application named cluster.war, which is available in the /home/student/JB248/labs/clustering-jgroups folder.

13. To verify that the two EAP instances in the cluster are using the protocol UDP for communication, use the tcpdump tool to monitor the traffic on the workstation virtual machine. Communication takes place at the multicast address 230.0.0.4 and port 45688 by default.

Open a new terminal window and run the following command:

```
[student@workstation ~]$ sudo tcpdump -i lo udp port 45688 -vvv
```

You will see output that looks like the following:

```
21:30:07.951124 IP (tos 0x8, ttl 2, id 43372, offset 0, flags [DF], proto UDP (17), length 4196) localhost.55300 >
  230.0.0.4.45688:
[bad udp cksum 0x7567 -> 0xba48!] UDP, length
  4168
```

```
21:30:08.163847 IP (tos 0x8, ttl 2, id 43373, offset 0, flags [DF], proto UDP (17), length 4196) localhost.55200 >
  230.0.0.4.45688:
[bad udp cksum 0x7567 -> 0x5712!] UDP, length
  4168
21:30:09.386034 IP (tos 0x8, ttl 2, id 43374, offset 0, flags [DF], proto UDP (17), length 68) localhost.55300 >

230.0.0.4.45688: [bad udp cksum 0x6547 -> 0xaf03!] UDP, length 40 21:30:09.598657 IP (tos 0x8, ttl 2, id 43375,
offset 0, flags [DF], proto UDP (17), length 68) localhost.55200 > 230.0.0.4.45688: [bad udp cksum 0x6547 -> 0x07ff!]
  UDP, length
40 21:30:09.951433 IP (tos 0x8, ttl 2, id 43376, offset 0, flags [DF], proto UDP (17), length 4196)
```

To stop getting information from tcpdump, press Ctrl+C in the terminal window where you started it.

14. Open two terminals and run the EAP CLI to connect to both instances.
Connect to the jgroups-cluster1 instance using the following command:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh --connect --controller=localhost:9990 [standalone@localhost:9990 /]
```

Connect to the jgroups-cluster2 instance using the following command. Notice that the jgroups-cluster2 instance is running with a port-offset value of 100:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh --connect --controller=localhost:10090 [standalone@localhost:10090 /]
```

15. Deploy the cluster.war file to the jgroups-cluster1 instance:

```
[standalone@localhost:9990 /] deploy \
   /home/student/JB248/labs/clustering-jgroups/cluster.war
[standalone@localhost:9990 /]
```

16. Similarly, deploy the cluster.war file to the jgroups instance cluster2:

```
[standalone@localhost:10090 /] deploy \ /home/student/
   JB248/labs/clustering-jgroups/cluster.war [standalone@localhost:10090 /]
```

17. In the log files of both instances, verify that the application was successfully deployed. You should see a message similar to the following:

```
2016-05-25 03:41:24,749 INFO [org.jboss.as.server] (management-handler-thread - 5)
  WFLYSRV0010: Deployed "cluster.war" (runtime-name : "cluster.war")
```

18. Navigate to http://localhost:8080/cluster to see the test application running on the jgroups-cluster1 instance. Refresh the page a few times and notice that the counter increases by one each time you refresh the page.

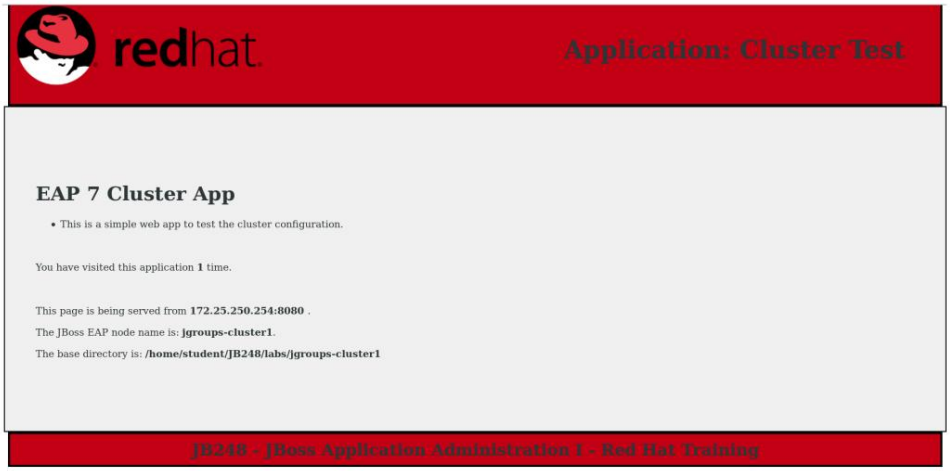


Figure 12.6: The cluster test application.

19. **Navigate to http://localhost:8180/cluster to see the test application running on the jgroups-cluster2 instance. Notice that the counter value was not reset, but increased by one. Refresh the page a few times and notice that the counter increases by one each time you refresh the page.**

20. **Shut down the jgroups-cluster2 instance by pressing Ctrl+C in the terminal window in which you started the instance. Navigate to http://localhost:8080/cluster to see the test application running on the jgroups-cluster1 instance.**
    Notice that the counter value has not been reset, but reflects the last value (plus one), as noted before shutting down jgroups-cluster2.

    Refresh the page a few times and notice that the counter increases by one each time you refresh the page. The EAP clustering component uses the JGroups stack (UDP by default) to replicate the counter value to all nodes in the cluster.

21. **Shut down the jgroups-cluster1 instance by pressing Ctrl+C in the terminal window in which the instance started.**

22. **Use the EAP CLI to define a new TCP stack configuration. Open the /home/student/JB248/labs/clustering-jgroups/new-tcp-stack.cli file in a text editor of your choice and review the commands listed there. The content of the file should look like this:**

```
# Add a new TCP stack called "tcpping" batch /

subsystem="jgroups"/stack="tcpping":add() /
subsystem="jgroups"/stack="tcpping":add-protocol(type="TCPPING") /
subsystem="jgroups"/stack="tcpping"/transport="TRANSPORT":add(socket
binding="jgroups-tcp",type="TCP") run-
batch

# Customize the protocol settings for tcpping batch /

subsystem="jgroups"/stack="tcpping"/protocol="TCPPING"/
property="initial_hosts":add(value="localhost[7600],localhost[7700]")
```

```
/subsystem="jgroups"/stack="tcpping"/protocol="TCPPING"/
property="port_range":add(value="10") /
subsystem="jgroups"/stack="tcpping":add-protocol(type="MERGE2") /subsystem="jgroups"/
stack="tcpping":add-protocol(socket-binding="jgroups-tcp  fd",type="FD_SOCK") /subsystem="jgroups"/
stack="tcpping":add-
protocol(type="FD") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="VERIFY_SUSPECT") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="BARRIER") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="pbcast.NAKACK") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="UNICAST2") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="pbcast.STABLE") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="pbcast.GMS") /subsystem="jgroups"/stack="tcpping":add-protocol(type="UFC") /
subsystem="jgroups"/stack="tcpping":add-protocol(type="MFC") /subsystem="jgroups"/
stack="tcpping":add-protocol(type="FRAG2") /subsystem="jgroups"/stack="tcpping":add-
protocol(type="RSVP") /subsystem=jgroups/channel=ee:write-
attribute(name=stack,value=tcpping) run-batch :reload
```

The initial_hosts property is a comma-separated list of server instances and the ports (in brackets) that will be part of the cluster.

23. Start both EAP instances as described in step 5.

24. Run the following command in a new terminal on the workstation virtual machine to create the new TCP stack on the jgroups-cluster1 instance:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh --connect --controller=localhost:9990 \
    --file=/home/student/JB248/labs/clustering-jgroups/new-tcp-stack.cli
The batch executed successfully
...
```

25. Similarly, run the following command in a new terminal on the workstation virtual machine to create the new TCP stack on the jgroups-cluster2 instance:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh --connect --controller=localhost:10090 \ --file=/home/student/JB248/
            labs/clustering-jgroups/new-tcp-stack.cli The batch executed successfully

...
```

26. Verify that the new TCP stack named tcpping is defined in the /home/student/JB248/labs/ jgroups-clusterX/configuration/standalone-full ha.xml file of both instances, where 'X' denotes the instance number :

```
<stack name="tcpping">
    <transport type="TCP" socket-binding="jgroups-tcp"/>
        <protocol type="TCPPING">
            <property name="initial_hosts">
                localhost[7600],localhost[7700] </property>
            <property
            name="port_range"> 10

            </property>
        </protocol>
```

```
            <protocol type="MERGE2"/>
                                    ...
</stack>
```

Also check that the default stack has been set to the new tcpping definition:

```
<channels default="ee">
      <channel name="ee" stack="tcpping"/> </channels>
```

27. Repeat steps 17-20 to test the new TCP stack. Check that the application test behaves similarly to how it did when you configured the instances with the default UDP stack.

28. Monitor the output of the tcpdump command as described in Step 13 and verify that you are NOT seeing UDP traffic on port 45688 because the nodes are using the TCP protocol for communication. Observe the TCP traffic on localhost port 7600 when both instances are running using the tcpdump command:

```
[student@workstation ~]$ sudo tcpdump -i lo tcp port 7600 -vvv
```

You should see output similar to the following:

```
04:42:11.670370 IP (tos 0x0, ttl 64, id 16573, offset 0, flags [DF], proto TCP (6), length 52) localhost.localdomain.55718
  >
    localhost.localdomain.7600: Flags [.], cksum 0xfe28 (incorrect -> 0xfb75), seq 62613, ack 59711, win
  32742, options [nop,nop,TS val 1811776 ecr 1811776], length 0 04:42:11.718546 IP (tos 0x0, ttl 64, id 65391, offset
  0, flags [DF], proto TCP (6), length 178)
localhost.localdomain.7600 > localhost.localdomain.55718: Flags [P.], cksum


  0xfea6 (incorrect -> 0xa79b), seq 59711:59837, ack 62613, win 10, options [nop,nop,TS val 1811824
  ecr 1811776], length 126
...
```

29. Stop the two EAP instances by pressing Ctrl+C in the terminal window in which started the instances.

Close the EAP CLI sessions by typing quit or press Ctrl+C in the terminal window in which you started the CLI sessions.

Exit the tail and tcpdump command sessions by pressing Ctrl+C in the terminal window in which you started the commands.

This concludes the demo.

Infinispan Subsystem Configuration

The Infinispan subsystem provides caching support for JBoss EAP, offering clustered server high availability features.

In a clustered environment, similar data is replicated to each cluster node. This data is stored in a cache, and the caching mechanism and features are implemented by a framework called *Infinispan.*

In addition to being able to configure how EAP caches data, Infinispan also offers facilities to view runtime metrics for cache containers and caches.

A cache is defined within a *cache container* or a repository for caches. There are four pre-configured cache containers in the ha and full-ha profiles:

• web: for session replication

• hibernate – for entity caching

• ejb – for stateful session bean replication

• server: for singleton caching

Developers use web, hibernate, and ejb caches for caching Java components. In a cluster, nodes use a configured cluster container cache for efficient and effective object replication across a large cluster of nodes.

There are four different types of caches:

• Local: Entries are not distributed to the rest of the cache; instead, they are stored only on the local node.

• Invalidation – Uses a cache to store inputs and fetches them from the cache when an input needs it.

• Replication – All entries are replicated to each node.

• Distribution: Entries are replicated to only some of the nodes.

Fittingly, there are four different pages in the admin console where each cache type is defined. These pages are included in the Infinispan section of the Subsystem page.

use

We won't cover optimizing the default cache in this course, but it's important to know that it exists and that in some environments users may need to modify the values to suit the application or environment.

The following XML extract shows the default Infinispan configuration:

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
    <cache-container name="server" aliases="singleton cluster" default-cache="default"
  module="org.wildfly.clustering.server"> <transport
        lock-timeout="60000"/> <replicated-cache
        name="default" mode="SYNC"> <transaction mode="BATCH"/
            >
        </replicated-cache>
    </cache-container>
    <cache-container name="web" default-cache="dist"
  module="org.wildfly.clustering.web.infinispan"> <transport lock-
        timeout="60000"/>
```

```
        <distributed-cache name="dist" mode="ASYNC" l1-lifespan="0" owners="2">
            <locking isolation="REPEATABLE_READ"/>
            <transaction mode="BATCH"/> <file-
            store/>
        </distributed-cache>
    </cache-container>
    <cache-container name="ejb" aliases="sfsb" default-cache="dist"
  module="org.wildfly.clustering.ejb.infinispan"> <transport lock-
        timeout="60000"/> <distributed-cache
        name="dist" mode="ASYNC" l1-lifespan="0" owners="2">
            <locking isolation="REPEATABLE_READ"/>
            <transaction mode="BATCH"/> <file-
            store/> </
        distributed-cache>
    </cache-container>
    <cache-container name="hibernate" default-cache="local-query"
  module="org.hibernate.infinispan">
        <transport lock-timeout="60000"/> <local-
        cache name="local-query"> <eviction
            strategy="LRU" max-entries="10000"/> <expiration max-
            idle="100000"/> </local-cache> <invalidation-
        cache name="entity"
        mode="SYNC"> <transaction mode="NON_XA"/> <eviction
            strategy="LRU" max-entries="10000"/
            > <expiration max-idle="100000"/>

        </invalidation-cache>
        <replicated-cache name="timestamps" mode="ASYNC"/>
    </cache-container>
</subsystem>
```

Notice that the subsystem defines the four default cache containers: web, hibernate, ejb, and server. Also notice that each cache container specifies default cache. For example, the hibernate cache container uses the local-query cache, which maps to local-cache.

Use the following process to configure a new cache with the EAP CLI:
1. Create a cache container:

```
/subsystem=infinispan/cache-container=<container-name>:add
```

2. Add a replicated cache:

```
/subsystem=infinispan/cache-container=<container-name>/replicated-cache=<cache
name>:add(mode=<mode>)
```

3. Set the default cache:

```
/subsystem=infinispan/cache-container=<container-name>:write-attribute(name=default  cache,value=<cache-
name>)
```

Start a cluster

To start a cluster in standalone mode or in a managed domain, servers can use the ha or full-ha profile to take advantage of already configured cluster settings.

Standalone or managed domain mode must be started as usual with an additional parameter, the jboss.node.name parameter, and the link address must also be provided. For example, on a standalone server:

```
[student@workstation bin]$ ./standalone.sh \
-Djboss.server.base.dir=/opt/jboss-eap-7.0/standalone/ \
-Djboss.node.name=jgroups-cluster1 \
-Djboss.bind.address=172.25.250.254 \ -c
standalone-full-ha.xml
```

When a node joins a cluster, the following log events are printed:

```
2016-05-25 03:08:57,593 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups  cluster1) ISPN000094: Received
new cluster view for channel server: [jgroups-cluster1|1] (2) [jgroups-cluster1, jgroups-cluster2]

2016-05-25 03:08:57,596 INFO
  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-1,ee,jgroups  cluster1) ISPN000094: Received
new cluster view for channel web: [jgroups-cluster1|1] (2) [jgroups-cluster1, jgroups-cluster2]

...
```

use

When starting additional standalone instances on the same node and attempting to cluster them, be sure to configure a port offset to avoid port conflicts:

```
-Djboss.socket.binding.port-offset=100
```

**To set up a cluster in a managed domain, users need to ensure that the servers in the server groups can communicate with each other through JGroups.
To do this, simply configure a server group to use the ha or full ha profiles.**

A cluster in a managed domain is configured at the server group level. To start a cluster in a managed domain:

1. **Change the desired server-group profile to ha (or full-ha).**
2. **Make sure that server-group is using ha-sockets (or full-ha-sockets) as its socket-binding-group.**
3. **Make sure that each server in the server-group has a unique name.**
4. **Deploy a distributable application to the server group.**

After configuring the domain controller with an appropriate server group, start the host controllers. Any distributable application deployed to the server group will cause all servers in that group to be clustered.

use

Make sure that the necessary ports are open on the firewall, including the ports used by TCP and UDP. If running in a managed domain, all of these ports must be open on each node. Remember to also open offset ports.