Guided Exercise: Securing Connections HTTP

In this lab work, you will configure an SSL connection to allow secure connections from a browser, increase the number of HTTP connections supported by EAP, and enable an AJP listener to support communication with the load balancer.

Resources	
Files:	/home/student/JB248/labs/standalone /home/student/JB248/labs/https
App URL:	http://localhost:8080/version https://localhost:8443/version
Resources	/home/student/JB248/labs/server-listeners/ version.war

Results

You should be able to access EAP using a secure browser connection, enable AJP to allow the load balancer to distribute load with this server, and improve EAP's responsiveness by increasing the number of concurrent HTTP connections.

before you start

Before beginning the guided exercise, run the following command to verify that EAP was installed in /opt/jboss-eap-7.0, that no EAP instances are running, to verify that the Configure JDBC Drivers guided exercise ran successfully, and to download the app version.war:

```
[student@workstation ~]$ lab server-listeners setup
```

1. Start the standalone EAP server.

Use the following command to start an EAP instance so that you can access the management console:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./standalone.sh \
-Djboss.server.base.dir=/home/student/JB248/labs/standalone/
```

Before proceeding, wait for the server to finish starting up.

2. Deploy the app version.

2.1. Open a new terminal window and run the following commands to start the CLI connection to your EAP instance:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh -c
```

2.2. Deploy the version.war application available in the /home/student/ JB248/labs/server-listeners folder:

```
[standalone@localhost:9990 /] deploy \ /home/
student/JB248/labs/server-listeners/version.war
```

23. Open a web browser on the workstation and navigate to http://localhost:8080/version to validate the implementation. The application version should appear.

3. Generate a self-signed certificate.

A self-signed certificate will be used to encrypt communication between the server and the browser. As a side effect, the certificate may raise security alerts when used, since the certificate was not issued by a trusted source. Some companies offer a trusted certificate that overcomes this limitation.

In this lab work, you will create a certificate to encrypt the communication between the server and the client. The HTTPS listener will use this certificate.

3.1. In a new terminal window, run the following to switch to the /home/student/JB248/labs/server-listeners folder:

```
[student@workstation ~]$ cd /home/student/JB248/labs/server-listeners
```

3.2. Create the certificate with the keytool command with the following parameters:

- **other: appserver**
- **large type: jks**
- **keyalg: RSA**
- **keysize: 2048**
- **keystore: identity.jks**
- **Name and surname: System administrator**
- **Organizational unit: GLS**
- **Name of the organization: Training**
- **City: Raleigh**
- **Status: NC**
- **Country: US**
- **Password: changeit**
- **Key appserver password: changeit**

```
[student@workstation server-listeners]$ keytool -genkeypair -alias appserver \
```

```
-storetype jks -keyalg RSA -keysize 2048 -keystore identity.jks
```

3.3. Verify that the identity.jks keystore has been created:

```
[student@workstation server-listeners]$ ls
```

The following result is expected:

```
identity.jks version.war
```

3.4. Check the certificates available in the identity.jks keystore:

```
[student@workstation server-listeners]$ keytool -list -v -keystore identity.jks
```

Use changeit as password. A similar result is expected:

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: appserver
Creation date: May 9, 2016
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=System Administrator, OU=GLS, O=Training, L=Raleigh, ST=NC, C=US
Issuer: CN=System Administrator, OU=GLS, O=Training, L=Raleigh, ST=NC, C=US

… OUTPUT OMMITED...
```

4. Create the SSL connection settings.

The SSL connection is configured by creating a new listener in the undertow subsystem and can be configured using the CLI. To configure this listener, a new security domain must be created to store the information about the certificate.

4.1. A new domain must be created to configure the HTTPS listener. return to the CLI and, in the core-service named management, add a new security realm named HTTPSRealm:

```
[standalone@localhost:9990 /] /core-service=\ management/
security-realm=HTTPSRealm:add()
```

4.2. Configure the HTTPSRealm realm to load the keystore file /home/student/ JB248/labs/server-listeners/identity.jks. Use the password changeit and open the certificate named appserver. The keystore file was created earlier in this guided exercise.

```
[standalone@localhost:9990 /] /core-service=management/security-realm=\ HTTPSRealm/server-
identity=\ ssl:add(keystore-path=/
home/student/JB248/labs/server-listeners/identity.jks, \ keystore-password=changeit, alias=appserver)
```

Important

In a production environment, it is recommended to use a store to protect the password of the certificate.

4.3. A top-up is required to activate the new domain:

```
[standalone@localhost:9990 /] reload
```

5. Configure el listener HTTPS.

5.1. Add a new HTTPS listener using the configured HTTPSRealm domain:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
https-listener=https:\ add(socket-binding=https,
security-realm=HTTPSRealm)
```

5.2. Open a web browser and navigate to https://localhost:8443/version to test the SSL connection.

use

You should get a warning about an untrusted certificate.
Accept the certificate and you should see the version apply over SSL.
Remember that if the certificate was authorized by a trusted source, you will not receive a warning about an untrusted certificate.

5.3. The SSL connection is available to all applications, including the root application. Navigate to https://localhost:8443/ and test the SSL connection for the root app.

6. Enable the AJP protocol.

6.1. To enable the AJP protocol, the AJP listener must be configured. Run the following command to enable the AJP protocol:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
ajp-listener=ajp:add(socket-binding=ajp)
```

6.2. Verify that the AJP listener is configured by reading its properties:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
ajp-listener=ajp:read-resource
```

A similar result is expected:

```
{
```

```
        "outcome" => "success", "result"
        => {
                "allow-encoded-slash" => false, "allow-equals-
                in-cookie-value" => false, "always-set-keep-alive" => true,
                "buffer-pipelined-data" => true, "buffer-pool"
                => "default", "decode-url" => true, "disallowed-
                methods" => ["TRACE"], "enabled" =>
                true, "max-buffered-request-
                size" => 16384, "max-connections" => undefined,



... OUTPUT OMMITED...
```

7. Adjust the HTTP listener.

After analyzing the number of connections coming from the server's HTTP listener, the total number of connections has increased, resulting in HTTP 500 errors.
Increase connections as much as possible for the HTTP listener to avoid errors.

7.1. The number of connections can be updated using the undertow subsystem's default server. Increase the maximum possible connections to 200 on the http-listener to improve responsiveness:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/http-
listener=default:\ write-attribute(name=max-
connections, value=200)
```

use

You can also define the maximum number of connections to https-listener and ajp-listener to improve the responsiveness of these protocols.

7.2. A recharge is required to activate the new maximum connections value:

```
[standalone@localhost:9990 /] reload
```

7.3. Verify that the max-connections attribute has been modified:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/http-
listener=default:read-attribute(name=max-connections)
```

The following result is expected:

```
{
        "outcome" => "success", "result"
        => 200
}
```

8. Perform cleaning.

8.1. Undeploy the application version.war:

```
[standalone@localhost:9990 /] undeploy version.war
```

8.2. Quite el listener AJP:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
ajp-listener=ajp:remove
```

8.3. Exit the EAP CLI:

```
[standalone@localhost:9990 /] exit
```

8.4. Stop the EAP instance by pressing Ctrl+C in the terminal window that is running EAP.

This concludes the guided exercise.

Lab Work: Configuring the Web Subsystem

In this lab assignment, you will configure the Undertow subsystem in an EAP managed domain that enables the HTTPS and AJP protocols. It will also deploy the bookstore app, making it the root web app.

Resources	
Files	/opt/domain /opt/keystore
app url	http://172.25.250.10:8080/bookstore http://172.25.250.11:8080/bookstore https://172.25.250.10:8443/bookstore https://172.25.250.11:8443/bookstore
Resources	bookstore.war

Result

You must be able to configure the Undertow web subsystem of an EAP managed domain that enables the HTTPS and AJP protocols. You should also be able to deploy the bookstore app, making it the root web app.

before you start

Use the following command to download the relevant lab files and ensure that the managed domain is set correctly:

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

1. Generate a self-signed certificate.

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.

The bookstore application will be deployed on this domain and must be accessible using the HTTPS protocol. To activate this protocol, you need to create a certificate using the keytool command. Generate a self-signed certificate by creating a keystore called identity.jks in the /opt/keystore folder on servera and serverb. The key repository must have the following characteristics:

- **other: appserver**
- **large type: jks**
- **keyalg: RSA**
- **keysize: 2048**
- **Password: changeit**

• Key appserver password: changeit

Use the values you want for the organization name and location. The jboss user must be the owner of the keystore. Finally, the keytool will confirm if the information is correct. Just type yes.

2. Start the domain controller.

Start the domain controller on workstation. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

3. Start the host controllers.

The two host controllers on servera and serverb connect to the domain controller in the previous step and get the latest domain configuration. Start the two host controllers on servera and serverb.

3.1. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

3.2. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

3.3. 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.

4. Create the SSL connection settings.

4.1. The SSL connection is configured by creating a new listener in the undertow subsystem and can be configured using the CLI. To configure this listener, a new security domain must be created for each server to store the information about the certificate.

In the management section of each host controller configuration file, create a new security domain named HTTPSRealm.

Because this configuration can only be created using the CLI, and the admin property is not directly available on a host, you can use the main service property instead of setting the admin attribute.

4.2. After creating this domain, configure it to load the keystore file /opt/keystore/identity.jks using the password changeit and open the certificate named appserver. The keystore file was created previously in this lab work.

5. Reload the host controllers to activate the domain.

6. Configure el listener HTTPS.

The HTTPS listener is responsible for enabling the HTTPS protocol in the undertow subsystem. Because all servers on all host controllers run using the full-ha profile, you must customize the undertow subsystem for that profile. Add a new https-listener named https, which refers to the socket binding named https (which listens for requests on port 8443) and should use the certificate associated with the HTTPSRealm security domain to encrypt the content. This listener must be linked to the default-server of the undertow configuration.

7. Configure el firewall.

A firewall configuration is required to enable communication using the HTTPS protocol. Open TCP port 8443 on servers servera and serverb.

8. Set the HTTPS listener.

To support a larger number of requests, the number of HTTPS connections is not enough. Increase the maximum connections to 150 in the undertow subsystem in the full-ha profile.

9. Deploy the bookstore application.

You are now ready to deploy the bookstore application WAR file to the managed domain. The bookstore.war file is available in the /tmp/ folder of the workstation. Deploy the application to Group1. Verify that you can use the bookstore application with the HTTP and HTTPS protocols at:

- **http://172.25.250.10:8080/bookstore**
- **http://172.25.250.11:8080/bookstore**
- **https://172.25.250.10:8443/bookstore**
- **https://172.25.250.11:8443/bookstore**

10. Set the bookstore app as the root app.

By default, the root application is the Welcome to JBoss application. This app should be replaced by the bookstore app. Replace the root application by changing the default-web-module attribute of the host named default-host.

This host is located in the full-ha profile, on the server named default-server in the undertow subsystem.

11. Perform cleaning and grading.

11.1.Exit the CLI:

```
[domain@172.25.250.254:9990 /] exit
```

11.2.Press Ctrl+C to stop the domain and host controllers.

11.3.Run the following workstation command to grade the assignment:

```
[student@workstation bin]$ lab undertow-lab-final grade
```

This concludes the lab work.

Solution

In this lab assignment, you will configure the Undertow subsystem in an EAP managed domain that enables the HTTPS and AJP protocols. It will also deploy the bookstore app, making it the root web app.

Resources	
Files	/opt/domain /opt/keystore
app url	http://172.25.250.10:8080/bookstore http://172.25.250.11:8080/bookstore https://172.25.250.10:8443/bookstore https://172.25.250.11:8443/bookstore
Resources	bookstore.war

Result

You should be able to configure the Undertow web subsystem of an EAP managed domain that enables the HTTPS and AJP protocols. You should also be able to deploy the bookstore app, making it the root web app.

before you start

Use the following command to download the relevant lab files and ensure that the managed domain is set correctly:

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

1. Generate a self-signed certificate.

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.

The bookstore application will be deployed on this domain and must be accessible using the HTTPS protocol. To activate this protocol, you need to create a certificate using the keytool command. Generate a self-signed certificate by creating a keystore called identity.jks in the /opt/keystore folder on servera and serverb. The key repository must have the following characteristics:

• other: appserver

• large type: jks

• keyalg: RSA

• keysize: 2048

• Password: changeit

• Key appserver password: changeit

Use the values you want for the organization name and location. The jboss user must be the owner of the keystore. Finally, the keytool will confirm if the information is correct. Just type yes.

1.1. On the servera server, open a new terminal window and change to the /opt/keystore folder:

```
[student@server ~]$ cd /opt/keystore
```

1.2. Generate the self-signed certificate:

```
[student@servera keystore]$ sudo -u jboss keytool -genkeypair \ -alias appserver \
-storetype jks -keyalg
RSA -keysize 2048 -keystore identity.jks
```

1.3. Check the certificates available in the identity.jks keystore:

```
[student@servera keystore]$ keytool -list -v -keystore identity.jks
```

1.4. Copy the servera virtual machine certificate to serverb in the /tmp directory.

```
[student@servera keystore]$ scp /opt/keystore/identity.jks \ student@serverb:/
tmp
```

1.5. Copy the file transferred to serverb in the /tmp directory to the /opt/ directory keystore.

```
[student@serverb ~]$ cp /tmp/identity.jks /opt/keystore
```

1.6. Change the keystore owner to the jboss user on the serverb virtual machine.

```
[student@serverb ~]$ sudo chown jboss:jboss /opt/keystore/identity.jks
```

2. Start the domain controller.

Start the domain controller on workstation. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

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

3. Start the host controllers.

The two host controllers on servera and serverb connect to the domain controller in the previous step and get the latest domain configuration. Start the two host controllers on servera and serverb.

3.1. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

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

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

3.2. 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 ...

Also change the directory to the /opt/jboss-eap-7.0/bin directory to avoid permission errors.

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

```
[student@serverb ~]$ cd /opt/jboss-eap-7.0/bin
```

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

3.3. 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.

4. Create the SSL connection settings.

4.1. The SSL connection is configured by creating a new listener in the undertow subsystem and can be configured using the CLI. To configure this listener, a new security domain must be created for each server to store the information about the certificate.

In the management section of each host controller configuration file, create a new security domain named HTTPSRealm.

Because this configuration can only be created using the CLI, and the admin property is not directly available on a host, you can use the main service property instead of setting the admin attribute.

Access the JBoss EAP CLI. 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
```

Add a new security realm named HTTPSRealm on the host serve:

```
[domain@172.25.250.254:9990 /] /host=servera/core-service=\ management/
security-realm=HTTPSRealm:add()
```

Add a new security realm named HTTPSRealm on host serverb:

```
[domain@172.25.250.254:9990 /] /host=serverb/core-service=\ management/
security-realm=HTTPSRealm:add()
```

4.2. After creating this domain, configure it to load the keystore file /opt/keystore/identity.jks using the password changeit and open the certificate named appserver. The keystore file was created previously in this lab work.

Configure the HTTPSRealm domain to load the identity.jks keystore on the servera host:

```
[domain@172.25.250.254:9990 /] /host=servera/core-service=\ management/
security-realm=HTTPSRealm/server-identity=\
```

```
ssl:add(keystore-path=/opt/keystore/identity.jks, \ keystore-
password=changeit, alias=appserver)
```

Configure the HTTPSRealm domain to load the identity.jks keystore on the host serverb:

```
[domain@172.25.250.254:9990 /] /host=serverb/core-service=\ management/
security-realm=HTTPSRealm/server-identity=\ ssl:add(keystore-path=/
opt/keystore/identity.jks, \ keystore-password=changeit,
alias=appserver)
```

5. Reload the host controllers to activate the domain.

Reload the host servera to activate the domain:

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

Reload the host serverb to activate the domain:

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

6. Configure el listener HTTPS.

The HTTPS listener is responsible for enabling the HTTPS protocol in the undertow subsystem. Because all servers on all host controllers run using the full-ha profile, you must customize the undertow subsystem for that profile. Add a new https-listener named https, which refers to the socket binding named https (which listens for requests on port 8443) and should use the certificate associated with the HTTPSRealm security domain to encrypt the content. This listener must be linked to the default-server of the undertow configuration.

6.1. Add https-listener:

```
[domain@172.25.250.254:9990 /] /profile=full-ha/subsystem=undertow/server=\ default-server/
https-listener=https:\ add(socket-binding=https,
security-realm=HTTPSRealm)
```

7. Configure el firewall.

A firewall configuration is required to enable communication using the HTTPS protocol. Open TCP port 8443 on servers servera and serverb.

7.1. On servera, open TCP port 8443:

```
[student@servera ~]$ sudo firewall-cmd --zone=public --add-port 8443/tcp \ --permanent
```

7.2. Reload the firewall configuration on servera:

```
[student@servera ~]$ sudo firewall-cmd --reload
```

7.3. Check open ports on servera:

```
[student@servera ~]$ sudo firewall-cmd --list-all
```

You should see port 8443 available in the ports attribute.

7.4. Open the same port in serverb:

```
[student@serverb ~]$ sudo firewall-cmd --zone=public --add-port 8443/tcp \ --permanent
```

7.5. Reload the firewall configuration in serverb:

```
[student@serverb ~]$ sudo firewall-cmd --reload
```

7.6. Check the open ports in serverb:

```
[student@serverb ~]$ sudo firewall-cmd --list-all
```

7.7. Verify that you can access the root application with SSL at https://172.25.250.10:8443 and https://172.25.250.11:8443.

use

You should get a warning about an untrusted certificate.
Accept the certificate and you should see the version apply over SSL.
Remember that if the certificate was authorized by a trusted source, you will not receive a warning about an untrusted certificate.

8. Set the HTTPS listener.

To support a larger number of requests, the number of HTTPS connections is not enough. Increase the maximum connections to 150 in the undertow subsystem in the full-ha profile.

8.1. The number of connections can be updated using the undertow subsystem's default server in the full-ha profile. Increase the maximum number of connections possible:

```
[domain@172.25.250.254:9990 /] /profile=full-ha/subsystem=undertow/server=\ default-server/
https-listener=https:\ write-attribute(name=max-
connections, value=150)
```

8.2. A recharge is required to activate the new maximum connections value:

```
[domain@172.25.250.254:9990 /] /:reload-servers
```

8.3. Verify that the max-connections attribute has been modified:

```
[domain@172.25.250.254:9990 /] /profile=full-ha/subsystem=undertow/server=\ default-server/
https-listener=https:\ read-attribute(name=max-
connections)
```

9. Deploy the bookstore application.

You are now ready to deploy the bookstore application WAR file to the managed domain. The bookstore.war file is available in the /tmp/ folder of the workstation. Deploy the application to Group1. Verify that you can use the bookstore application with the HTTP and HTTPS protocols at:

- **http://172.25.250.10:8080/bookstore**
- **http://172.25.250.11:8080/bookstore**
- **https://172.25.250.10:8443/bookstore**
- **https://172.25.250.11:8443/bookstore**

9.1. Deploy the app:

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

9.2. Verify that you can use the bookstore application using the provided URLs.

10. Set the bookstore app as the root app.

By default, the root application is the Welcome to JBoss application. This app should be replaced by the bookstore app. Replace the root application by changing the default-web-module attribute of the host named default-host.

This host is located in the full-ha profile, on the server named default-server in the undertow subsystem.

10.1.Replace the root app:

```
[domain@172.25.250.254:9990 /] /profile=full-ha/subsystem=undertow/server=\ default-server/
host=default-host:write-attribute(name=\ default-web-
module,value=bookstore.war)
```

10.2.A reload is required to activate the new root app:

```
[domain@172.25.250.254:9990 /] /:reload-servers
```

10.3.Test the root application using one of the following URLs:

- **http://172.25.250.10:8080**
- **http://172.25.250.11:8080**
- **https://172.25.250.10:8443**

- **https://172.25.250.11:8443**

use

If the bookstore app doesn't show up, try clearing your browser's cache and refreshing the page.

11. Perform cleaning and grading.

11.1.Exit the CLI:

```
[domain@172.25.250.254:9990 /] exit
```

11.2.Press Ctrl+C to stop the domain and host controllers.

11.3.Run the following workstation command to grade the assignment:

```
[student@workstation bin]$ lab undertow-lab-final grade
```

This concludes the lab work.

Summary

In this chapter, you learned the following:

- The JBoss web server was replaced by Undertow.
- Undertow is written in Java, using the XNIO API.
- New features are provided by Undertow:
  - ÿ Undertow can act as a load balancer.
  - ÿ Reduce the number of ports.
  - ÿ Supports web sockets.
  - ÿ Supports JSON-P.
  - ÿ Supports HTTP/2.
- Undertow has five main components:
  - ÿ Buffered caches
  - ÿ Server
  - ÿ Servlet Container
  - ÿ Handlers
  - ÿ Filters
- It is possible to define the default application in two ways:
  - ÿ Changing the welcome-content file handler.
  - ÿ Changing the default-web-module attribute.
- It is possible to configure a server with three listeners:
  - ÿ HTTP
  - ÿ HTTPS
  - ÿ AJP
- To allow an SSL connection, a certification file is required and it is possible to generate a self-signed certificate with the keytool command.