Workshop: Understanding the CLI

Workshop
Understanding the CLI

In this Workshop, follow along with the instructor as you navigate through the CLI.

Important

One outcome of this workshop is the creation of a new server group, called **qa-group**. You will need this group later in the course.

☐ 1. Start the CLI

☐ 1.1. Open a terminal window and change directories to ***EAP_HOME*/bin**.

☐ 1.2. You start the CLI by running the **jboss-cli.sh** script (or **jboss-cli.bat** batch file) in the **bin** folder of EAP. Use the **--controller** property to specify the host and port of the EAP instance you want to connect to.

On RHEL, enter the following command (replacing *xx* with your IP address):

```
[student@station bin]$ ./jboss-cli.sh --connect --
controller=192.168.0.xx:9999
```

On Windows:

```
jboss-cli.bat --connect --controller=192.168.0.xx:9999
```

The **-c** flag is a shortcut for **--connect**. If you forgot the **--connect** in the previous step then you can issue a **connect** command at the CLI prompt.

☐ 1.3. When you attempt to connect to a remote Domain controller, you will be prompted for credentials:

```
$ ./jboss-cli.sh --connect --controller=192.168.0.254:9999
Authenticating against security realm: ManagementRealm
Username: admin
Password:
[domain@192.168.0.254:9999 /]
```

☐ 1.4. Enter **help** to view the various commands. Notice at the end of the help output that a description and an example are given of how to input operation requests.

☐ 2. Read a Resource

A common and useful task with the CLI is reading resource properties to discover or verify your configurations.

- 2.1. The CLI input is a hierarchical structure that starts at the Domain level. (In Standalone mode, the hierarchy starts at the Server level.) For example, to view the settings at the top level, enter the following command:

```
/:read-resource
```

- 2.2. The forward slash "/" is used to separate levels. For example, the following command reads the resources of a Host:

```
/host=host2:read-resource
```

- 2.3. You can keep drilling down to further levels in the hierarchy using the forward slash. For example:

```
/host=host2/server=production-server-A:read-resource
```

Insight

If you are familiar with Linux **bash** CLI shortcuts, then you will find a number of them are available with-in the JBoss CLI too. For example, **Ctrl+U** clears from the cursor to the beginning of the line, **Ctrl+K** clears from the cursor to the end of the line, **Up Arrow** & **Down Arrow** step forward and backward through the CLI history. Feel free to explore which other key combinations maybe applicable.

- 3. Using Tab Completion
  - 3.1. A nice feature of the CLI is **Tab completion**, which shows all possible commands available at any point in your current command. For example, enter / then hit **Tab** to view all possible values you can enter next after the /.

```
[domain@192.168.0.254:9999 /] /

deployment                     extension                      host
          interface
management-client-content      path                           profile
          server-group
socket-binding-group           system-property
```

  - 3.2. Start typing in "**profile**" after the /, and hit **Tab**. Notice how the CLI not only completes the **profile** sub-level, but it adds an equals sign, because an equals sign is the only possible value after "/profile".

```
/profile=
```

  - 3.3. Hit **Tab** again and all of the profiles will display:

```
[domain@192.168.0.254:9999 /] /profile=
```

```
default   full      full-ha   ha
[domain@192.168.0.254:9999 /] /profile=
```

☐ 3.4. Start typing "**default**" and hit **Tab** to auto-complete. Then enter a **/** and hit **Tab**. The following will appear:

```
/profile=default/subsystem=
```

This is because the only level below profile is subsystem. Hit **Tab** again to view the available subsystems:

```
[domain@192.168.0.254:9999 /] /profile=default/subsystem=

configadmin          datasources          ee                   ejb3
      infinispan
jaxrs                jca                  jdr                  jmx
      jpa
logging              mail                 naming               osgi
      pojo
remoting             resource-adapters    sar                  security
      threads
transactions         web                  webservices          weld
[domain@192.168.0.254:9999 /] /profile=default/subsystem=
```

☐ 3.5. Start typing in the name of a subsystem and it will auto-complete the subsystem's name. Invoke **read-resource** on the **logging** subsystem:

```
[domain@192.168.0.254:9999 /] /profile=default/subsystem=logging:read-
resource
{
    "outcome" => "success",
    "result" => {
        "async-handler" => undefined,
        "console-handler" => undefined,
        "custom-handler" => undefined,
        "file-handler" => undefined,
        "size-rotating-file-handler" => undefined,
        "logger" => {
            "jacorb" => undefined,
            "com.arjuna" => undefined,
            "org.apache.tomcat.util.modeler" => undefined,
            "jacorb.config" => undefined,
            "sun.rmi" => undefined
        },
        "periodic-rotating-file-handler" => {"FILE" => undefined},
        "root-logger" => {"ROOT" => undefined}
    }
}
[domain@192.168.0.254:9999 /]
```

☐ 3.6. You can keep going to deeper levels. For example, the **web** subsystem has several levels:

```
[domain@192.168.0.254:9999 /] /profile=default/subsystem=web/

configuration    connector        virtual-server
```

```
[domain@192.168.0.254:9999 /] /profile=default/subsystem=web/
```

- ☐ 3.7. Notice the **web** subsystem has a sub-level for connectors. You can invoke **read-resource** at any level. To view the properties of the **http** connector:

```
/profile=default/subsystem=web/connector=http:read-resource
```

- ☐ 3.8. You can use an asterisk as a wildcard. For example, to view all the connectors of the **web** subsystem of the **default** profile:

```
/profile=default/subsystem=web/connector=*:read-resource
```

- ☐ 4. Invoking Operations
  - ☐ 4.1. The colon is used to invoke an operation, as you saw with **read-resource**. If you just enter a colon then hit **Tab**, you will see all the available commands for whatever level you are at. For example, at the root level enter a colon and hit **Tab** to view the operations at the root level:

```
[domain@192.168.0.254:9999 /] :

add-namespace                  add-schema-location            delete-
snapshot
full-replace-deployment        list-snapshots                 read-
attribute
read-children-names            read-children-resources        read-
children-types
read-config-as-xml             read-operation-description     read-
operation-names
read-resource                  read-resource-description      remove-
namespace
remove-schema-location         resolve-expression-on-domain   restart-
servers
start-servers                  stop-servers                   take-
snapshot
undefine-attribute             upload-deployment-bytes        upload-
deployment-stream
upload-deployment-url          validate-address               validate-
operation
whoami                         write-attribute
[domain@192.168.0.254:9999 /] :
```

  - ☐ 4.2. Notice there is an operation called **read-operation-names**, which shows a similar output. Enter the following command and compare its output to the output in the previous step:

```
:read-operation-names
```

  - ☐ 4.3. To view a description of an operation, use the **:read-operation-description** operation. This operation requires a **name** parameter, which must be set to a name of an operation. As you type in the following command, notice that the attribute will not auto-complete - you have to type it in:

```
[domain@192.168.0.254:9999 /] :read-operation-description(name=restart-
servers)
{
    "outcome" => "success",
    "result" => {
        "operation-name" => "restart-servers",
        "description" => "Restarts all servers currently running in the
 domain.",
        "reply-properties" => {},
        "read-only" => false
    }
}
[domain@192.168.0.254:9999 /]
```

☐ 4.4. When you invoke an operation, the CLI displays the output in DMR format. Try invoking the **`restart-servers`** operation:

```
[domain@192.168.0.254:9999 /] :restart-servers
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => undefined
}
[domain@192.168.0.254:9999 /]
```

Check the output in the terminal windows of **`host2`** and **`host3`**. All of your servers should have restarted, and notice in the result above that the "**`outcome`**" value is "**`success`**".

☐ 5. Read a Resource Recursively

☐ 5.1. Add the **`recursive`** flag at the end of **`:read-resource`** to view a resource and all of its sub-levels. For example, compare the output of the following operations:

```
/profile=default/subsystem=web:read-resource
/profile=default/subsystem=web:read-resource(recursive=true)
```

Notice with the **`recursive`** flag set to **`true`** that all child elements of the **`web`** resource are displayed, and their child elements are displayed, and so on.

☐ 5.2. If you want to view all resources on the server, simply start at the top level and turn on recursion:

```
/:read-resource(recursive=true)
```

The output will show you the settings of your entire Domain configuration, which is about 5,000 lines of output! You can pipe the output to a file for better viewing:

```
[domain@192.168.0.254:9999 /] :read-resource(recursive=true) > output.txt
[domain@192.168.0.254:9999 /]
```

Important

In the following screen, and later screens in this lab, the commands are very long. They should be entered on one line, even though they appear to break across two lines in the screens.

```
[domain@192.168.0.254:9999 / #] /host=host2/server-config=dev-server-
two:add(group=dev-group,socket-binding-group=standard-sockets)
#1 /host=host2/server-config=dev-server-two:add(group=dev-group,socket-
binding-group=standard-sockets)
[domain@192.168.0.254:9999 / #] /server-group=dev-group/system-
property=opt.folder:add(value=/home/student/JB248/opt)
#2 /server-group=dev-group/system-property=opt.folder:add(value=/home/
student/JB248/opt)
```

If you make a mistake and need to start over, execute the **discard-batch** command to exit **batch** mode (and lose any batch commands entered).

Insight

Notice that defining a new Server is unique using the CLI because instead of invoking the **add** operation at the **server** level, there is a special **server-config** level that is used instead. You also use the **server-config** to edit attributes of existing servers.

☐ 5.3. To execute a batch, use the **run-batch** command:

```
[domain@192.168.0.254:9999 / #] run-batch
The batch executed successfully.
[domain@192.168.0.254:9999 /]
```

If you get an error, enter the **discard-batch** command, then try defining and running the batch again.

☐ 5.4. Verify your new server is defined:

```
/host=host2/server-config=dev-server-two:read-resource
```

Alternately, go to the **Server** page of the Management Console and view the list of **Server Configurations** for **host2**.

☐ 5.5. You can use the CLI to start your new Server. Enter the following command:

```
[domain@192.168.0.254:9999 /] /host=host2/server-config=dev-server-
two:start
{
    "outcome" => "success",
    "result" => "STARTING"
}
```

Notice no output appears in the CLI, but you should see a new file named **`output.txt`** (in your ***`EAP_HOME`*`/bin`** folder) with the entire settings of your Domain in the DMR syntax.

- ☐ 6. Changing Directories
  - ☐ 6.1. Browsing the resources in the CLI is similar to navigating a folder system from a Linux command prompt. For example, if you need to enter multiple commands on the subsystem level, you can **`cd`** to that level. For example, if you know you are going to invoke multiple operations on **`host2`**, then **`cd`** to that level. Enter the following commands:

```
[domain@192.168.0.254:9999 /] cd host=host2
[domain@192.168.0.254:9999 host=host2] ./

core-service       interface          jvm                path
 server             server-config
system-property
[domain@192.168.0.254:9999 host=host2] ./
```

Notice the use of the **`./`** to display the sub-levels relative to the current level.

  - ☐ 6.2. You can also use the **`ls`** command to view a more detailed list of the resources at the current level you are on:

```
ls
```

  - ☐ 6.3. At the **`host2`** level, start typing in the following command and hit **`Tab`** after the equals sign:

```
[domain@192.168.0.254:9999 host=host2] cd server=

dev-server-one         production-server-A
[domain@192.168.0.254:9999 host=host2] cd server=
```

Change levels into **`production-server-A`** and enter **`ls`** to view its contents:

```
[domain@192.168.0.254:9999 host=host2] cd server=production-server-A
[domain@192.168.0.254:9999 server=production-server-A] ls
core-service                                deployment
    extension
interface                                   path
    socket-binding-group
subsystem                                   system-property
    launch-type=DOMAIN
management-major-version=1                  management-minor-version=1
    name=production-server-A
namespaces=[]                               process-type=Server
    product-name=EAP
product-version=6.0.0.Beta2                 profile-name=ha
    release-codename=Dronies
release-version=7.1.1.Final-redhat-1        running-mode=NORMAL
    schema-locations=[]
server-state=running
[domain@192.168.0.254:9999 server=production-server-A]
```

Notice how the command prompt changes to display the current level that you are on.

☐ 6.4. When you have changed directories to a specific level, you can now invoke an operation at that level without specifying the full path. For example, if you want to invoke **read-resource** on the **/host=host2/server=production-server-A** level and you are already at that level, simply enter **:read-resource**.

```
:read-resource
```

☐ 6.5. Commands like the following also work:

```
cd .. (move up one level)
cd /  (move to the root level)
```

☐ 7. Change a Resource Attribute

Up until now, you have only read information using the CLI. However, you can use the **write-attribute** operation to modify a resource's attributes using the CLI. To demonstrate this operation, you will change the **min-pool-size** attribute of the **ExampleDS** datasource resource. We will discuss datasource configuration in Unit 6, *The Datasource Subsystem*, but for now just note that this attribute defines the minimum number of connections held open for a datasource.

☐ 7.1. To view the **min-pool-size** attribute's current value, enter the following command (use **Tab** completion to avoid typing errors):

```
[domain@192.168.0.254:9999 data-source=ExampleDS] cd /profile=default/
subsystem=datasources/data-source=ExampleDS
[domain@192.168.0.254:9999 data-source=ExampleDS] :read-resource
```

There are a lot of values of the ExampleDS datasource. To view a specific attribute, use the read-attribute operation:

```
[domain@192.168.0.254:9999 data-source=ExampleDS] :read-
attribute(name=min-pool-size)
{
    "outcome" => "success",
    "result" => undefined
}
```

Notice the value of **min-pool-size** is **undefined**.

☐ 7.2. Use the **write-attribute** command to change the **min-pool-size** value to **5**:

```
[domain@192.168.0.254:9999 data-source=ExampleDS] :write-
attribute(name=min-pool-size,value=5)
{
    "outcome" => "success",
    "result" => undefined,
```

```
    "server-groups" => {"dev-group" => {"host" => {"host2" => {"dev-
server-one" => {
        "response" => {
            "outcome" => "success",
            "result" => undefined
        }
    }}}}
}
```

- ☐ 7.3. Use the **:read-attribute** command and verify the change occurred:

```
[domain@192.168.0.254:9999 data-source=ExampleDS] :read-
attribute(name=min-pool-size)
{
    "outcome" => "success",
    "result" => 5
}
```

- ☐ 7.4. Go to the **Profiles** page of the Management Console. Click on the **Datasources** link in the **Connector** section. In the middle of the page is a collection of tabs. Click on the **Pool** tab and verify that the **Min Pool Size** is now 5. Notice the Management Console is instantly aware of the change made in the CLI.

- ☐ 7.5. View the **domain.xml** file of your Domain controller (in the folder **machine1/domain/configuration**). Notice in the **ExampleDS** definition in the **datasource** subsystem of the **default** profile that the **<min-pool-size>** element is **5**, and you have now seen that the Management Console, the CLI and the underlying XML files are all synchronized.

Important

Not all resource attributes are writeable. Use the **:read-resource-description** command to check if the **access-type** of the attribute you want is **read-write**. Other valid types include **read-only** and **metric**.

- ☐ 8. The add Operation
  - ☐ 8.1. The CLI **add** operation is used to add new resources to a configuration. To add a new configuration, you start by typing in the name of the new resource at the level where the resource is to appear. For example, the **system-property** element is at the root level. The following statement adds a new system property named **x** whose value is **25**:

```
/system-property=x:add(value=25)
```

    Enter the command above, then go the **System Properties** page found on the **Profiles** page of the Management Console. The value of **x** should appear in the list of **System Properties**.

  - ☐ 8.2. You will want to use tab completion as much as possible to simplify using the **add** operation. For example, suppose you want to define a new Server Group, which is defined at the root level.

Start by entering the following and hitting **Tab**:

```
/server-group=
```

You should have two Server Groups defined:

```
[domain@192.168.0.254:9999 /] /server-group=

dev-group           production-group
[domain@192.168.0.254:9999 /] /server-group=
```

☐ 8.3. To add a **server-group**, type in a new **server-group** name that does not already appear, followed by the **add** operation. Carefully type in the following and see what tab completion displays:

```
[domain@192.168.0.254:9999 /] /server-group=qa-group:add(

jvm=                                management-subsystem-endpoint=
profile=                            socket-binding-group=
socket-binding-port-offset=
[domain@192.168.0.254:9999 /] /server-group=qa-group:add(
```

Tab complete is showing you the available attributes that you can define for a new **server-group**, which you will do in the next step.

☐ 8.4. Attributes are defined in an **add** operation as a comma-separated list of **name=value** pairs. For example, the following command defines the **profile** and **socket-binding-group** attributes for the new **qa-group**. (These two attributes are required.)

```
[domain@192.168.0.254:9999 /] /server-group=qa-group:add(profile=full-
ha,socket-binding-group=full-ha-sockets)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => undefined
}
[domain@192.168.0.254:9999 /]
```

☐ 8.5. Verify the new server group definition by invoking **read-resource** on it:

```
[domain@192.168.0.254:9999 /] /server-group=qa-group:read-resource

{
    "outcome" => "success",
    "result" => {
        "deployment" => undefined,
        "jvm" => undefined,
        "management-subsystem-endpoint" => false,
        "profile" => "full-ha",
        "socket-binding-group" => "full-ha-sockets",
        "system-property" => undefined
    }
}
```

```
[domain@192.168.0.254:9999 /]
```

You can also verify the new Server Group definition by viewing the **Server Groups** page, found on the **Server** page of the Management Console.

- ☐ 9. The CLI GUI
  - ☐ 9.1. Exit the CLI by entering exit at the prompt:

```
[domain@192.168.0.254:9999 /] exit
```

  - ☐ 9.2. Restart the CLI, but this time add the **--gui** property:

```
./jboss-cli.sh  connect --controller=192.168.0.xx:9990 --gui
```

  - ☐ 9.3. With the CLI GUI, you can drill-down to a level, then right-click on a level to select an operation to invoke. For example, expand **profile=full**, then click on **subsystem=logging**. Notice the entry in the **cmd>** textfield:

```
/profile=full/subsystem=logging/
```

  - ☐ 9.4. Right-click on the **logging** subsystem and select **read-resource** from the pop-up dialog. A pop-up window appears allowing you to select options for the **read-resource** operation. Check the **recursive** box, then click the **OK** button. View the command in the **cmd>** textfield.
  - ☐ 9.5. Notice you can use the CLI GUI to write commands for you, then copy-and-paste them into a running CLI or into a script file. You can also run the command right from the GUI. Click the **Submit** button and the **Output** tab will appear with the result of the operation.

Insight

The GUI CLI is a very handy tool! Use it to generate your CLI commands for you, helping you avoid typing mistakes. It is especially helpful when executing long commands and/or writing script files.

Lab 05_01: Using the CLI

Performance Checklist

Lab Overview: In this lab, you will perform a multitude of administrative tasks using the CLI.

Lab Resources/Configuration:

Lab Files Location:	n/a
Application URL:	n/a

Success Criteria: After completing this exercise, you will be more familiar with navigating and using the CLI.

Lab Outline:

1. Reading Resources
2. Add a New Path Resource
3. Change an Attribute
4. Take a Snapshot
5. The Batch Command
6. Executing a CLI Script File

Before you begin...

Connect to your Domain using the CLI tool.

☐ 1. Reading Resources

Use the **`read-resource`** operation of the CLI to answer the following questions. These questions are intentionally difficult, in order to help you learn how to navigate through the CLI!

- ☐ 1.1. What is the value of the **`system-property`** named **`java.net.preferIPv4Stack`**? _____
- ☐ 1.2. How many **`cache-containers`** are defined in the **`infinispan`** subsystem of the **`ha`** profile? _____
- ☐ 1.3. Does the **`mail-session`** named **`java:jboss/mail/Default`** in the **`mail`** subsystem of the **`default`** profile use SSL? (**Hint:** Use the **`recursive`** flag to view the SSL setting of **`mail-session`**.) _____
- ☐ 1.4. In the **`web`** subsystem of the **`full-ha`** profile, there is a **`virtual-server`** named **`default-host`** which serves up requests made to the local IP address. Is the **`welcome-content`** application (the default Welcome page for EAP 6) enabled for **`default-host`**? _____
- ☐ 1.5. What is the maximum POST size of the **`http`** connector in the **`web`** subsystem of the **`full`** profile? _____

- ☐ 2. Add a New Path Resource
  - ☐ 2.1. A *path* is a variable that refers to a file or folder location on your filesystem. Using paths allows you to avoid hard-coding system-dependant paths and use variable names instead. The Management Console does not provide a mechanism for defining new paths, but the CLI does. Enter the following command and hit **Tab**:

```
/path=
```

    There are no currently-defined paths, so tab completion doesn't do anything.

  - ☐ 2.2. To add a path, you start by entering the name you want to give the path followed by the **add** operation. Enter the following and hit **Tab** again:

```
[domain@192.168.0.254:9999 /] /path=home.folder:add(

path=           relative-to=
[domain@192.168.0.254:9999 /] /path=home.folder:add(
```

  - ☐ 2.3. On RHEL, set the path attribute to **/home/student**:

```
/path=home.folder:add(path=/home/student)
```

    On Windows, use **c:\JB248**:

```
/path=home.folder:add(path="c:\JB248")
```

  - ☐ 2.4. Verify the path is defined appropriately by invoking **read-resource** on **home.folder**:

```
[domain@192.168.0.254:9999 /] /path=home.folder:read-resource
{
    "outcome" => "success",
    "result" => {
        "name" => "home.folder",
        "path" => "/home/student",
        "read-only" => false,
        "relative-to" => undefined
    }
}
```

- ☐ 3. Change an Attribute
  - ☐ 3.1. Enter the following command to view the settings of the **default** JVM on **host2**:

```
/host=host2/jvm=default:read-resource
```

  - ☐ 3.2. What is the maximum heap size of the **default jvm**? _____
  - ☐ 3.3. What is the size of the permanent generation? _____

- ☐ 3.4. Using the **write-attribute** operation, set the value of **stack-size** on the **default jvm** to **128k**.

- ☐ 3.5. To verify the **stack-size** was changed appropriately, enter the following command and make sure your output matches the output here:

```
[domain@192.168.0.254:9999 /] /host=host2/jvm=default:read-
attribute(name=stack-size)
{
    "outcome" => "success",
    "result" => "128k"
}
```

Insight

The default stack size of a thread is typically too big for many production environments that use a lot of threads. If you have too many threads, you can actually run out of memory because the stack size is set too high! In those situations, lowering the **stack-size** to a value like **128k** can help avoid the out-of-memory issues.

- ☐ 4. Take a Snapshot
  - ☐ 4.1. Enter the following command at the root level to take a snapshot of your current configuration files:

```
:take-snapshot
```

  - ☐ 4.2. Look in your **machine1/domain/configuration/domain_xml_history/snapshot** folder and verify the snapshot worked successfully.

- ☐ 5. The Batch Command
  - ☐ 5.1. The CLI has a **batch** command that allows you to enter multiple commands that execute as one atomic unit. If at least one of the commands or operations fails, all the other successfully executed commands and operations in the batch are rolled back. Execute the command **batch** to enter batch mode:

```
[domain@192.168.0.254:9999 /] batch

[domain@192.168.0.254:9999 / #]
```

    It is a subtle difference, but notice the hash symbol now appears in the prompt.

  - ☐ 5.2. Now enter the following two commands, which define a new Server on **host2** and also adds a **system-property** to the Server Group named **dev-group**. Notice that when you are in batch mode, each command is assigned a number.

Important

In the following screen, and later screens in this lab, the commands are very long. They should be entered on one line, even though they appear to break across two lines in the screens.

```
[domain@192.168.0.254:9999 / #] /host=host2/server-config=dev-server-
two:add(group=dev-group,socket-binding-group=standard-sockets)
#1 /host=host2/server-config=dev-server-two:add(group=dev-group,socket-
binding-group=standard-sockets)
[domain@192.168.0.254:9999 / #] /server-group=dev-group/system-
property=opt.folder:add(value=/home/student/JB248/opt)
#2 /server-group=dev-group/system-property=opt.folder:add(value=/home/
student/JB248/opt)
```

If you make a mistake and need to start over, execute the **discard-batch** command to exit **batch** mode (and lose any batch commands entered).

Insight

Notice that defining a new Server is unique using the CLI because instead of invoking the **add** operation at the **server** level, there is a special **server-config** level that is used instead. You also use the **server-config** to edit attributes of existing servers.

☐ 5.3. To execute a batch, use the **run-batch** command:

```
[domain@192.168.0.254:9999 / #] run-batch
The batch executed successfully.
[domain@192.168.0.254:9999 /]
```

If you get an error, enter the **discard-batch** command, then try defining and running the batch again.

☐ 5.4. Verify your new server is defined:

```
/host=host2/server-config=dev-server-two:read-resource
```

Alternately, go to the **Server** page of the Management Console and view the list of **Server Configurations** for **host2**.

☐ 5.5. You can use the CLI to start your new Server. Enter the following command:

```
[domain@192.168.0.254:9999 /] /host=host2/server-config=dev-server-
two:start
{
    "outcome" => "success",
    "result" => "STARTING"
}
```

Verify it is has started by viewing the output in terminal window of **host2**.

- ☐ 5.6. Verify that the **system-property** named **opt.folder** was also defined in the **batch**. Use tab completion as much as possible to enter the following command:

```
[domain@192.168.0.254:9999 /] /host=host2/server=dev-server-two/system-
property=opt.folder:read-resource
{
    "outcome" => "success",
    "result" => {"value" => "/home/student/JB248/opt"}
}
```

Your output should be similar to the output above.

- ☐ 6. Executing a CLI Script File
  - ☐ 6.1. You can enter CLI commands in a text file, then pass the text file to the CLI tool, thereby allowing you to define scripts for repetitive tasks. To demonstrate how to define a script file, start by exiting the CLI:

```
[domain@192.168.0.254:9999 /] exit
```

  - ☐ 6.2. Using a text editor, define a new file named **deploy_example_app.cli** in your ***LABS*** folder.
  - ☐ 6.3. Add the following commands to **deploy_example_app.cli**, which deploy **example.war** onto the **production-group** and restart the server instances:

```
batch
deploy /home/student/JB248/labs/Lab02_01/example.war --server-
groups=production-group
:restart-servers
run-batch
```

On Windows, use the appropriate path to **example.war**.

  - ☐ 6.4. Save the file.
  - ☐ 6.5. Execute the **deploy_example_app.cli** script by passing it into the CLI runtime command using the **--file** option:

```
$ ./jboss-cli.sh --connect --controller=192.168.0.254:9999 --file=LABS/
deploy_example_app.cli
Authenticating against security realm: ManagementRealm
Username: admin
Password:
#1 deploy --name=example.war --server-groups=production-group
#2 /:restart-servers
The batch executed successfully.
```

  - ☐ 6.6. Verify the servers restarted by looking in the output of the terminal windows for **host2** and **host3**.

- ☐ 6.7. Verify the **`example.war`** application deployed successfully by pointing your browser to `http://192.168.0.xx:48080/example/`.
- ☐ 6.8. Stop here! You have performed a lot of different tasks using the CLI. The goal of performing these tasks was more than just to show you how to do them, but also for you to become familiar with navigating around and invoking operations using the CLI. As we go through the remainder of the course, you will see how to configure various subsystems and settings using both the Management Console and the CLI.