Blue Planet Orchestrate DevOps Toolkit

Software Skills

Student Lab Guide

Rev. 3

Blue Planet 18.06.4

blueplanet®

LEGAL NOTICES

Contacting Ciena

Corporate headquarters	410-694-5700 or 800-921-1144	www.ciena.com
Customer technical support/warranty		
In North America	1-800-CIENA24 (243-6224) 410-865-4961	
In Europe, Middle East, and Africa	800-CIENA-24-7 (800-2436-2247) +44-207-012-5508	
In Asia-Pacific	800-CIENA-24-7 (800-2436-2247) +81-3-6367-3989 +91-124-4340-600	
In Caribbean and Latin America	800-CIENA-24-7 (800-2436-2247) 410-865-4944 (USA)	
Sales and General Information	410-694-5700	E-mail: sales@ciena.com
In North America	410-694-5700 or 800-207-3714	E-mail: sales@ciena.com
In Europe	+44-207-012-5500 (UK)	E-mail: sales@ciena.com
In Asia +81-3-3248-4680 (Japan)		E-mail: sales@ciena.com
In India	+91-124-434-0500	E-mail: sales@ciena.com
In Latin America	011-5255-1719-0220 (Mexico City)	E-mail: sales@ciena.com
Training	877-CIENA-TD (243-6283) or 410-865-8996	E-mail: techtng@ciena.com

For additional office locations and phone numbers, please visit the Ciena website at www.ciena.com.

Change History

Orchestrate Release	Revision	Publication Date	Reason for Change
16.10	1.0.0	4/9/2017	Initial release.
17.02	2.0.0	6/7/17	Remove labs on linux and upgrade to 17.02
18.06.4	3.0.0	1/15/19	Updates: fixed errors introduced in previous version and updated for Devops TK 18.-6.4

Contents

Lab 1: Create a JSON file

Helpful hints:

File and directory names will be highlighted in the following font: `/home/vagrant`

Commands will be highlighted in **bold text**.

Important notes and warnings are in red text.

Links are in blue text.

Objective: In this lab you will create a valid JSON file.

Task 1: Create the file

1. Open a text editor of your choosing.
2. Create a JSON file that contains an object with the following keys and values:

Key	Value	
Age	An integer value of 25	
Name	A string value of "Bob "	
Spouse	A string value of "Sue"	
Children	An array with the string values of "Nick", "Tim", and "Ann"	
Address	An object that has the following keys and values:	
	Key	**Value**
	Street	A string of "Main St."
	Number	A numeric value of 101
	City	An integer value of "San Diego"
	State	A string value of "CA"
	ZIP	An integer value of 92020

3. Validate the format of your JSON object using http://jsonlint.com.
4. Save the data into a file named `test.json`.

Conclusion: You now have a valid JSON file. Compare your file with the answer file provided in "Answer Keys " folder provided on your USB disk.

End of Lab

Lab 2: Create a JSON Schema

Objective: In this lab you will create a valid JSON schema file.

Task 1: Create the file

1. Open a text editor of your choosing.
2. Create a JSON schema file that can be used to validate the following format:

Key	**Value**	
Age	An integer value greater than 0 and less than 125	
Name	A string value	
Spouse	A string value	
Children	An array with string values, at least one value must be present and each item must be unique	
Address	An object that has the following keys and values:	
	Key	**Value**
	Street	A string
	Number	An integer value
	City	A string value
	State	A string value
	ZIP	An integer value

3. Make sure that the Age and Name Keys are required.
4. Do not allow for any additional Keys .
5. Validate the format of your JSON schema object using http://jsonlint.com.
6. Save the data into a file named **`test-schema.json`**.

Task 2: Use your Schema to validate the JSON object

1. In a web browser, go to the following URL: http://www.jsonschemavalidator.net.
2. Copy the contents of your JSON Schema file (**`test-schema.json`**) and paste this into the text area on the left hand side of the web page.
3. Copy the contents of your JSON file (**`test.json`**) and paste this into the text area on the right hand side of the web page.
4. Verify that the following message appears in the bottom left-hand side of the web page:

✔ No errors found. JSON validates against the schema

Conclusion: You now have a valid JSON Schema file. Compare your file with the answer file provided in "Answer Keys " folder provided on your USB disk.

End of Lab

Lab 3: Create a HOCON file

Objective: In this lab you will create a valid HOCON file.

Task 1: Create the file

1. Open a text editor of your choosing.
2. Create a HOCON file that contains an object with the following keys and values (use HOCON format instead of JSON format whenever possible):

Key	**Value**	
Age	An integer value of 25	
Name	A string value of "Bob "	
Spouse	A string value of "Sue"	
Children	An array with the string values of "Nick", "Tim", and "Ann"	
Address	An object that has the following keys and values:	
	Key	**Value**
	Street	A string of "Main St."
	Number	A numeric value of 101
	City	An integer value of "San Diego"
	State	A string value of "CA"
	ZIP	An integer value of 92020

3. Validate the format of your HOCON object using http://www.hoconlint.com.
4. Save the data into a file named **`test.hocon`**.

Conclusion: You now have a valid HOCON file. Compare your file with the answer file provided in "Answer Keys " folder provided on your USB disk.

End of Lab

Lab 4: Create a YAML file

Objective: In this lab you will create a valid YAML file.

Task 1: Create the file

1. Open a text editor of your choosing.
2. Create a YAML file that contains an object with the following keys and values:

Key	Value	
Age	An integer value of 25	
Name	A string value of "Bob "	
Spouse	A string value of "Sue"	
Children	An array with the string values of "Nick", "Tim", and "Ann"	
Address	An object that has the following keys and values:	
	Key	**Value**
	Street	A string of "Main St."
	Number	A numeric value of 101
	City	An integer value of "San Diego"
	State	A string value of "CA"
	ZIP	An integer value of 92020

3. Validate the format of your HOCON object using http://www.yamllint.com.
4. Save the data into a file named **`test.yaml`**.

Conclusion: You now have a valid YAML file. Compare your file with the answer file provided in "Answer Keys " folder provided on your USB disk.

End of Lab

Lab 5: Setup the Blue Planet DevOps Toolkit

Objective: In this lab you will install and configure the software that is required for the Devops Toolkit.

Task 1: Install Vagrant and Virtualbox

To use the Vagrant box you must install Vagrant and Virtualbox on your host machine. Both are free downloads. Check the DevOps Exchange for details on the version of Vagrant and Virtualbox to use with the DevOps Toolkit.

- Vagrant version 1.9 available here: https://releases.hashicorp.com/vagrant/1.9.0/
- Oracle VirtualBox version 5.1.30 available here: https://www.virtualbox.org/wiki/Download_Old_Builds_5_1

NOTE: Hardware virtualization **must** be enabled on your PC. See your PC documentation for instructions on enabling hardware virtualization through the BIOS.

The DevOps Toolkit provides a self-contained environment for developing and testing Blue Planet resource adapters. The toolkit is available on the Blue Planet Developers Exchange at http://community.ciena.com. The toolkit is deployed on your host computer using Vagrant and Virtualbox. The DevOps Vagrant box is a self-contained Blue Planet development environment that includes:

- Docker
- Git
- bpocore-dev docker image
- orchestrate-ui-dev docker image
- RASDK – Resource Adapter Development Kit

For details on system requirements see the information posted on the Blue Planet Developers Exchange.

When developing RAs and service templates you can use your IDE of choice, or a text editor.

If you are using Windows you must have an SSH utility. One solution is Cygwin. It can be downloaded from https://cygwin.com/install.html. When installing Cygwin be sure to install the **OpenSSH** package as this is not selected by default. Cygwin provides a unix-like terminal for Windows.

Note that Cygwin installs other Unix-based software. If you only want a SSH client utility, consider installing PUTTY from http://www.putty.org/.

Task 2: Setup the Vagrant Project

1. Open a command-line window on your computer.
2. Navigate to your home directory or another directory that you can easily get to. The DevOps Toolkit can be in any directory on your computer.
3. Create a directory for the DevOps toolkit, and make it the current directory:

```
$ mkdir ra_devops_toolkit
$ cd ra_devops_toolkit
```

   You can also use Explorer (Windows) or Finder (Mac) to create the directory.

   This is the directory where the DevOps Toolkit will be installed.

   This will be referred to as the *DevOps Toolkit directory* for the rest of the course.

4. Download the toolkit bundle and put it in the DevOps Toolkit directory.

5. Create a **`Vagrantfile`** in the current directory:

```
$ vagrant init ra-18.06.4 devops-toolkit-orch-18.06.4.box
```

`devops-toolkit-orch-18.02.2.box` is the name of the toolkit **`.box`** file, which is available on the Blue Planet community page. **You must use the exact name of the toolkit .box file**.

Vagrant creates a **`Vagrantfile`** in the directory. For details on Vagrantfiles, see Vagrantfiles in the Vagrant documentation.

Task 3: Edit your Vagrantfile

You need to customize your environment by editing the Vagrantfile. For more information and an example of customizing VirtualBox, see VBoxManage in the VirtualBox documentation. These instructions provide the minimum customizations needed to run the DevOps Toolkit.

1. Open the Vagrantfile in a text editor. Be sure the text editor will not use "smart quotes". **Do not use Wordpad or other Rich Text Format editors.**
2. Around line 28 uncomment the line below to create a private network that allows host-only access. Remove the # at the start of the line to uncomment the line:

```
config.vm.network "private_network", ip: "192.168.33.10"
```

3. **NOTE**: Be sure the IP address parameter is included. On some Windows systems this may not be present.
4. Around line 45-51 is a block that is commented out that starts with:

```
# config.vm.provider "virtualbox" do |vb|
```

5. and ends with:

```
#end
```

6. Uncomment both of those lines.
7. Between those lines add the following:

```
vb.customize ["modifyvm", :id, "--memory", "4096"]
```

8. When you're done the block should look like this:

```
config.vm.provider "virtualbox" do |vb|
          vb.customize ["modifyvm", :id, "--memory", "4096"]
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
  #   vb.memory = "1024"
end
```

9. Just before the block of code starting around line 45, add the follow lines to create a shared directory for your host machine and the virtual machine.

```
config.vm.synced_folder "./shared", "/home/vagrant/shared", type: "nfs"
config.vm.synced_folder ".", "/vagrant", disabled: true
```

10. Save the file.

Task 4: Create a 'shared' Directory

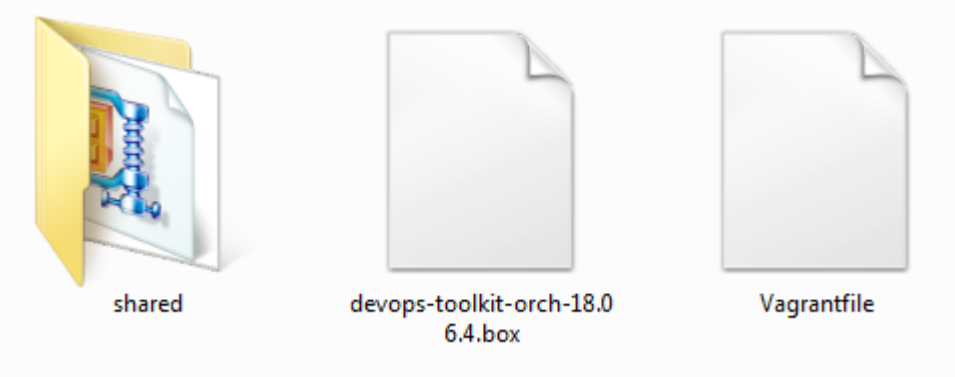
1. In a terminal window on your host computer navigate to DevOps Toolkit directory.
2. Create a **`shared`** directory:

```
$ mkdir shared
```

3. You can also use Explorer (Windows) or Finder (Mac) to create the directory.

When you're done the DevOps Toolkit directory should have:

- **`Vagrantfile`**
- DevOps Toolkit .box file (file name depends on the version)
- **`shared`** directory



Task 5: Starting the VM

1. Once you have modified the **`Vagrantfile`** and have created the **`shared`** directory you can start the virtual machine and login.
2. In a terminal window on your host computer go to the DevOps Toolkit directory. **For Windows users, make sure you open a DOS prompt with Administrative rights.**
3. Start the VM:

```
$ vagrant up
```

4. Note that when the VM is brought up for the first time, it may take a while (10-15 minutes). If any errors occur, look closely at the error messages. Often the errors will be the result of a type in your **`Vagrantfile`**, but they could also indicate missing software (Virtualbox), the wrong version of the software, the lack of administrative rights or the lack of BIOS virtualization.
5. Once the VM has booted, connect to the machine.

   For Mac and Linux systems use the Vagrant SSH command:

```
$ vagrant ssh
```

For Windows the above command will work if you are using Cygwin. If not, open an SSH application and connect to the virtual machine using the IP address 192.168.33.10. Use the following credentials:

- Username: *vagrant*
- Password: *vagrant*

Once you have login you should see the default Ubuntu prompt:

```
vagrant@vagrant:~$
```

For the rest of this course the virtual machine is referred to as the *Vagrant development environment*. In future labs, you will be directed to go to the "Vagrant bash prompt".

Conclusion: You now have a configured Devops Toolkit environment.

End of Lab

Lab 6: Starting the Orchestrate Docker Containers

Objective: In this lab you will start the docker containers needed for the development environment and then navigate to your Orchestrate web UI to verify the Vagrant development environment is running.

You will be opening three SSH sessions connected the Vagrant development environment:

- One for bpocore-dev
- One for orchestrate-ui-dev
- One for normal development work

Task 1: Verify Image Versions

1. From the Vagrant bash prompt verify the version of the Docker containers:

```
$ docker images
```

2. This will display all available images with the version. The version is listed in the Tag column.

```
$ docker images
REPOSITORY                                                      TAG
   IMAGE ID            CREATED             SIZE
blueplanet/racisco                                              latest
   f8eb83cc650e        5 weeks ago         758.8 MB
artifactory.ciena.com/blueplanet/script-dev                     2.1.21
   e2d032141c21        5 weeks ago         842.9 MB
artifactory.ciena.com/blueplanet/bpocore-dev          2.1.1-8949-c7cd94b
   fa4853a861a6        7 weeks ago         738.9 MB
artifactory.ciena.com/blueplanet/frost-orchestrate-ui-dev    1806.0.15
   7c1da1385a62        7 weeks ago         248.8 MB
artifactory.ciena.com/blueplanet/base-image-devops-toolkit   20180223
   e96fb895d1d0        7 months ago        631 MB
```

Note the version for frost-orchestrate-ui-dev and bpocore-dev. In the above output the versions are:

frost-orchestrate-ui-dev:1806.0.15

bpocore-dev:2.1.1-8949-c7cd94b

Task 2: Start the Docker Containers

1. At the Vagrant bash prompt start bpocore-dev using the following command. Enter the command on a single line (also read "Notes" below):

```
$ docker run --name bpocore-dev -it --rm --publish 8181:8181 -v
   /bp2:/var/log/orchestrate/
   artifactory.ciena.com/blueplanet/bpocore-dev:2.1.1-8949-c7cd94b
```

Notes:

- The tag (2.1.1-8949-c7cd94b in the command above) must match the tag displayed for the image when you ran the **docker images** command.
- The ␣ represents a space character. There should be no space character between "artifactory.ciena.com/blueplanet/bpocore-dev:" and "2.1.1-8949-c7cd94b".

Wait for bpocore-dev to start. A series of log messages will display. Once they stop you can continue. No return prompt will display.

Leave this terminal window open.

2. Open a new SSH session to the Vagrant environment.
3. From the Vagrant bash prompt start the Orchestrate UI using the **docker run** command. Enter the command on a single line:

```
$ docker run --rm -it -p 9980:80 --name orchestrate-ui-dev --link␣
   bpocore-dev:bpocore-dev␣
   artifactory.ciena.com/blueplanet/frost-orchestrate-ui-dev:1806.0.15
```

Notes:

- The tag (1806.0.15 in bold in the command above) must match the tag displayed for the image when you ran the **docker images** command.
- The ␣ represents a space character.

No log messages or return prompt will display.

Leave this terminal window open.

4. Open a new SSH session to the Vagrant environment.
5. From the Vagrant bash prompt, list the currently running docker containers to verify that the bpocore and front-orchestrate-ui containers are running:

```
docker ps

CONTAINER ID        IMAGE
   COMMAND             CREATED             STATUS              PORTS
   NAMES

c3f3ca585154
   artifactory.ciena.com/blueplanet/frost-orchestrate-ui-dev:1806.0.15
   "nginx"             15 minutes ago      Up 15 minutes
   0.0.0.0:9980->80/tcp                       orchestrate-ui-dev

76897b1a5ca2
   artifactory.ciena.com/blueplanet/bpocore-dev:2.1.1-8949-c7cd94b
   "/main.py"          15 minutes ago      Up 15 minutes       22/tcp,
   9092/tcp, 0.0.0.0:8181->8181/tcp   bpocore-dev
```

6. You can now navigate to the orchestrate web UI using the IP address of the Vagrant virtual machine. In a web browser go to **http://192.168.33.10:9980/orchestrate/**. You should see the Blue Planet interface.

Once you have verified that you can access the Orchestrate web UI, leave the containers running and the SSH session windows open. You will need them for the next lab exercise.

Conclusion: You now have had some practice running basic docker commands.

End of Lab

Lab 7: Suspending and Restarting the Development Environment

Objective: If you reboot or shut down your host machine, you could stop the docker containers running bpocore-dev and orchestrate-ui-dev. This means you will have to restart both docker containers and will lose any changes made to the bpocore-dev database. This lab shows you how to properly suspend your development environment before shutting down your host OS.

You should have three terminal windows open:

- One for bpocore-dev – this should be showing log messages
- One for orchestrate-ui-dev – this should be showing log messages
- One at the Ubuntu prompt

Task 1: Suspend the Development Environment

1. If you are in the Vagrant environment, you will need to exit the vagrant environment. At the Vagrant bash prompt, type **exit** and press enter to exit the Vagrant development environment. This will return you to your host machine, for example:

```
$ exit
logout
Connection to 127.0.0.1 closed.
```

When you exit the development environment you will see the logout message and a message indicating the SSH connection was closed.

Note: If you are using Putty or other similar tools to connect to the development environment the connection will be closed, but you will need to open a command prompt window and navigate to the directory containing the **`Vagrantfile`** and **`.box`** file.

2. At the prompt for your host machine suspend the Vagrant:

```
$ vagrant suspend
```

You will see message indicating the VM is suspending. Once the VM is finished suspending you will see the normal prompt for your host machine.

The terminal windows running bpocore-dev and orchestrate-ui-dev will also be returned to the prompt for your host machine. (For Windows systems, the terminal windows may automatically close.)

3. Go to the web browser window running the Orchestrate web UI and reload the page. The connection should timeout, indicating the development environment has been suspended.

Task 2: Restart the Development Environment

1. In the terminal on your host machine navigate to the directory containing the **`Vagrantfile`** and the **`.box`** file.
2. Restart the development environment:

```
$ vagrant up
```


3. After the Vagrant VM starts up, go to the web browser window running the Orchestrate web UI and reload the page. The UI should display.
4. Connect to the Vagrant virtual machine.

   For Mac and Linux systems use the Vagrant SSH command:

```
$ vagrant ssh
```

   For Windows open an SSH application and connect to the virtual machine using the IP address 192.168.33.10. Use the following credentials:

- Username: *vagrant*
- Password: *vagrant*

   Once you login you should see the default Ubuntu prompt:

```
vagrant@vagrant:~$
```

5. Verify the docker containers are running:

```
$ docker ps
```

   This will list the current containers. The bpocore-dev and orchestrate-ui-dev containers should be listed, and the STATUS column show an UP time.

Task 3: Stopping Docker Containers

This procedure shows you how to stop and restart docker containers.

1. Verify the container is running using the following command:

```
$ docker ps –a
```

2. This displays the running containers:

```
CONTAINER
   ID        IMAGE
          COMMAND              CREATED              STATUS               PORTS
                               NAMES

f2afa2438477        artifactory.ciena.com/blueplanet/frost-orchestrate-ui-d
   ev:1806.0.15   "nginx"              10 minutes ago      Up 10
   minutes       0.0.0.0:9980->80/tcp              orchestrate-ui-dev

630a27979194        artifactory.ciena.com/blueplanet/bpocore-dev:1.6.1-7983
   -d0ba3e1      "/main.py"           12 minutes ago      Up 12
   minutes       22/tcp, 0.0.0.0:8181->8181/tcp   bpocore-dev
```

3. Stop the running container using the container ID (in bold in the example above):

```
$ docker stop f2afa2438477
```

4. Repeat for the other container.

Later when you need to start the containers use the **docker run** command.

Troubleshooting Docker

This section is included for your reference. There is no need to execute any of these commands at this time.

If the docker containers do not shut down properly you may get an error when trying to start the container:

```
docker: Error response from daemon: Conflict. The name "/orchestrate-ui-dev"
   is already in use by container
   d881c733c283835c206dabcdced1d170c5733b2e3fc4553a865ea6f47381854e. You
   have to remove (or rename) that container to be able to reuse that name.
```

This means the container is already running. If bpocore-dev or orchestrate-ui-dev are not working properly you may need to restart the container:

1. Use the **docker ps -a** command to view all containers.
2. Locate the container ID for the container you need restart.
3. Use the **docker stop <container ID>** command to stop the container.
4. Use the **docker start <container ID>** command to start the container.

If you are still experiencing issues starting the containers, you may need to remove the current containers. Use the **docker rm** command to remove the container, then use the **docker run** command to start a new container. For example:

```
$ docker rm ddbae0bfd050


$ docker run --rm -it -p 9980:80 --name orchestrate-ui-dev --link
   bpocore-dev:bpocore-dev
   artifactory.ciena.com/blueplanet/frost-orchestrate-ui-dev:1806.0.15
```

Conclusion: You now have had some practice controlling the vagrant image and docker containers.

End of Lab

Lab 8: Setting up a Basic Resource Adapter

Objective: In this lab you will create a basic Resource Adapter project using the paster and make commands.

Task 1: Render a New RA Project

1. Open an SSH session to the Vagrant development environment.
2. At the Vagrant bash prompt navigate to the **`shared`** directory.

```
$ cd shared
```

The **`shared`** directory is used for RA projects because it is available in both the Vagrant development environment and your host machine. This allows you to use any IDE or text editor on your host machine to create your RA.

3. Run the **paster** command to create a new RA project.

```
$ paster
```

You will be prompted to enter information about your RA. Use the information provided below in *`italics`*.

- **The name of the RA [MyRA]:** *`radocker`*
- **Python package name for your RA [radocker]:** *`radocker`*
- **The author of the RA [Ciena Engineer]:** *`your_name`*
- **The vendor of the RA [Ciena]:** *`your_company`*
- **List the endpoints supported by your RA. [['cli', 'snmp']]:** *`['cli']`*
- **General type or class of things the RA connects to, e.g. Ciena_Optical. [Demo]:** *`NixServer`*
- **List the subgroup or family of devices/domains that your RA manages, e.g. Ciena6500 [['Demo-A']]:** *`NixDockerServer`*
- **List the specific types of devices/domains supported by this RA, e.g. CN6500 [['DEMO-A-1']]:** `docker`
- **Select your RA's configuration optimization settings:** *`1`*
- **Select if you'd like the virtualenv directory initialized for you:** *`1`*
- **Will you be running the RA as an Orchestrate resource provider? [Y/n]:** *`Y`*
- **Organization which maintains the RA. (URL is recommended) [http://ciena.com]:** `http://ciena.com`
- **The vendor's license for the RA. (URL is recommended). [http://ciena.com/license]:** `http://training/license`
- **RP ID (must be a valid UUID or urn such as urn:ciena:bp:ra:ciena6500) [urn:ciena:bp:ra:radocker]:** `urn:ciena:bp:ra:radocker`
- **ID for the RP's first domain (must be a valid UUID or urn such as urn:ciena:bp:domain:ciena6500) [urn:ciena:bp:domain:radocker]:** `urn:ciena:bp:domain:radocker`
- **Maintainer/owner of this RP [Ciena Engineer]:** `Ciena Student`
- **Email for the maintainer [ciena.engineer@email.com]:** `ciena.student@email.com`

Continued on next page…

- **Human-readable title of this RP [radocker]:** `RA Docker`
- **Describe the RP []:** `A manager for docker images`
- **RP version [0.0.1]:** *0.0.1*
- **Select your resource provider type**: `1 - domain manager`

  Once you complete all the prompts, the **paster** command will create the RA project in the **`radocker`** directory.

Task 2: Prepare the RA

1. From the Vagrant bash prompt, move to the **`radocker`** directory:

```
$ cd radocker
```

2. Use the **make** command to prepare the RA:

```
$ make prepare-venv
```

You will see a series of messages as the environment is prepared. This command requires Internet access and make take some time (10-15 minutes) to complete.

3. Verify the software packages were installed by executing the following command:

```
$ ls env/bin
```

You should see a list of Python executables that were installed, including a command you will use in a later lab (the **bpprov-cli** command)

Conclusion: You now have a basic RA project directory. While you won't build Resource Adapters in this course, you will make use of some of the tools provided by Resource Adapters.

Lab 9: Use TextFSM to parse data

Objective: In this lab you will create an FSM template using sample data. You will then test the template using the **bpprov-cli** command line tool.

The data and template will not be used in your RA. This exercise allows you to practice using FSM and the **bpprov-cli** tool.

The **bpprov-cli** command format is:

env/bin/bpprov-cli fsm-validate test.fsm --input=@fsm_test.txt

Your RA should be running when you issue the command. The input data is provided in the sample files.

Task 1: Create the FSM template file

1. Copy **fsm_test.txt** from the sample files directory into your RA project directory, for example: **~/shared/<ra-name>/fsm_test.txt**
2. Open **fsm_test.txt**. Note the format of the data in the file. You should see three fields:
   - Index
   - ERP Name
   - VID Set
3. In your RA project directory create a new directory called **fsms** in the **model** directory.
4. Create a new text file named **test.fsm** and save it in the **fsms** directory. You will use this new file to create an FSM template.
5. In **test.fsm** create value definitions for each of the fields in **fsm_test.txt**. Use the following format:

```
Value <Value Name> (<insert regular expression>)
```

   For example:

```
Value Index (\S+)
```

   You must determine the correct regular expression for each value:
   - Index: Match any number one or more times
   - ERP Name: Match any character except line return one or more times
   - VID Set: Match a single VLAN or a range of VLANs
6. Below the value definitions create a state definition. State definitions are enclosed between Start and EOF:

```
Start
  <state definitions>


EOF
```

   The state definition must match the format of the text input. Include variables for the value definitions using the format `${ValueName}`.

   End the state definition with `-> Record`.

For example:

```
^${Index} ${Name} -> Record
```

(Note: The example shown above will not work for this exercise. You will need a combination of variables and regular expressions for this exercise.)

7. Once you have the state definition complete save the file.

Task 2: Use the validation tool

1. From the Vagrant bash prompt, use the **bpprov-cli** command to test the FSM template:

```
$ env/bin/bpprov-cli fsm-validate <raname>/model/fsms/test.fsm --input=@fsm_test.txt
```

2. The output should be a list of lists with the values from the sample text file (see below). If your output does not match or you get an error, check the regular expressions in your FSM template.

```
[
    [
        "1",
        "Training ERP                    ",
        "600-650"
    ],
    [
        "2",
        "Iris erp 500-550                ",
        "500-550"
    ],
    [
        "3",
        "Iris ERP 400-450                ",
        "400-450"
    ],
    [
        "4",
        "S1 Iris ERP 100-150             ",
        "100-150"
    ],
    [
        "6",
        "Gene1                           ",
        "300-350"
    ],
    [
        "7",
        "Arek1                           ",
        "200-250"
    ]
]
```

Conclusion: You created a proper FSM template and validated it against test data.

End of Lab

Lab 10: Use Jinja2 to parse device data

Objective: In this lab you will parse data from the output of Resource Adapter commands into a different data structure using

Task 1: Review the original data

1. Using a text editor of your choice, open the **`jinja_data.txt`** file from the samples folder.
2. Review the contents of this file. This data has been created by a Resource Adapter and needs to be converted into a different format.
3. Open a web browser and navigate to the following URL: https://cryptic-cliffs-32040.herokuapp.com/
4. Copy the entire contents of the jinja_data.txt file into the "Values" window in the web site.

Task 2: Create the template and test it

1. Using a text editor of your choice, create a new file.
2. In this new file, create a Jinja Template that will convert the data from the **`jinja_data.txt`** file into data that looks like the following:

```
{
    "type": "linux",
    "resourceType": "docker",
    "serialNumber": "vagrant-ubuntu-trusty-64",
    "deviceVersion": "1.9.1, build b9f10c9",
    "swImage": "GNU-Linux",
    "swType": "x86_64",
    "swVersion": "3.13.0-88-generic"
}
```

3. Copy the contents of this new file into the Template window in the web site that you opened in Task 1.
4. Click the Convert button and review the results in the Render window. If the results don't match up exactly with the data structure shown in step 2, continue to work on your Jinja template.

Conclusion: You created a Jinja template to covert data that originated from a Resource Adapter. Compare your template with the jinja.txt file in the Answers folder of your USB disk.

End of Lab