

**WA2543 DevOps Boot Camp**

**Student Labs**

**Web Age Solutions Inc.**

## Table of Contents

Lab 1 - Configuration Management.....	3
Lab 2 - Version Control - Git.....	13
Lab 3 - Version Control – SVN.....	19
Lab 4 - Continuous Integration.....	23
Lab 5 - Install Prerequisites.....	35
Lab 6 - Continuous Code Quality - SonarQube.....	41
Lab 7 - Automation (Shell Scripting).....	45
Lab 8 - Tomcat Application Deployment using Chef.....	54
Lab 9 - Continuous Monitoring - Nagios.....	59
Lab 10 - Containerization – Docker .....	68
Lab 11 - Scripting 101 – Python (OPTIONAL).....	73

# Lab 1 - Configuration Management

**Note:** This document can be found on the Desktop of your computer to allow you copy and paste the code during the exercises.

In this chapter you will explore Chef basics. You will create recipes that utilizes various Chef resources.

## Part 1 - Launch terminal

In this part you will launch the Linux terminal.

\_\_1. Open the **Terminal** window.



Alternatively, you can press Ctrl+Alt+T to access the terminal

or

In the task bar, click Search button. In search text box, type in terminal and hit enter key on the keyboard.



\_\_2. Run following command to switch to **Documents** directory.

```
cd ~/Documents
```

## Part 2 - Copy lab scripts

In this part you will copy lab scripts to Documents directory.

\_\_1. Copy scripts.

```
cp -r ~/Downloads/labs ~/Documents/
```

\_\_2. Switch to **chef** directory.

```
cd ~/Documents/labs/chef
```

## Part 3 - Install Chef server core, Chef manage, and Chef Development Kit

In this part you will install Chef server, Chef manage, and Chef development kit.

**Note:** If prompted to enter password during the labs, enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_1. View chef\_install.sh file contents.

```
cat chef_install.sh | more
```

\_\_2. Run the install script.

```
bash chef_install.sh
```

Note: Installation will take a few minutes complete.

Notice it executes a script and shows various messages, such as, action create, action run, action delete, action create\_if\_missing, and action restart. Actions are part of a cookbook recipe and will be covered later in the course.

Syntax for creating Chef admin user is as follows:

```
chef-server-ctl user-create USER_NAME FIRST_NAME LAST_NAME  
EMAIL_ADDRESS PASSWORD --filename PRIVATE_KEY
```

Each chef server must belong to an organization. For complete docs on all the chef server control commands and syntax reference [https://docs.chef.io/ctl\\_chef\\_server.html](https://docs.chef.io/ctl_chef_server.html)

## Part 4 - Verify Chef server installation

In this part you will verify Chef server installation.

\_\_1. Run following command to view the chef configuration. This dumps out the chef JSON configuration object.

```
sudo chef-server-ctl show-config
```

\_\_2. Run following command to view the services running for chef. This dumps out the chef JSON configuration object.

```
sudo chef-server-ctl service-list
```

\_\_3. Run following command to view status of each running chef service.

```
sudo chef-server-ctl status
```

Note: The status command can also be invoked with a specific service, e.g. chef-server-ctl status nginx.

\_\_4. Run following command to view the service logs for all running chef services.

```
sudo chef-server-ctl tail
```

\_\_5. Hit Ctrl + Z to exit.

\_\_6. Similar to the service-list command you can view only the last entries for a specific

service log.

```
sudo chef-server-ctl tail erchef
```

Note: The output may not show the result if there are no error messages in the log file. This is the expected message:

```
wasadmin@ubuntu:~/Downloads$ sudo chef-server-ctl tail erchef
find: `/var/log/opscode/erchef': No such file or directory
tail: cannot follow '-' by name
```

The chef server control has a built in process supervisor that ensures all of the required services are in the appropriate state at any given time. The supervisor starts two processes per service and provides the following sub-commands for managing services: hup, int, kill, once, restart, service-list, start, status, stop, tail, and term.

\_\_ 7. Run following command to stop the embedded nginx service.

```
sudo chef-server-ctl stop nginx
```

\_\_ 8. Run following command to verify the embedded nginx service is stopped.

```
netstat -an | grep 0.0.0.0:443
```

\_\_ 9. Run following command to start the embedded nginx service.

```
sudo chef-server-ctl restart nginx
```

## Part 5 - Verify Chef Manage installation

In this part you will verify chef manage installation. Chef manage is used for various reasons, such as, viewing nodes, editing run list, viewing reports and managing policies, and administration.

\_\_ 1. To verify chef manage is installed successfully launch Firefox from the Linux task bar.

\_\_ 2. In the URL bar enter:

```
https://localhost
```

Note: The SSL certificate will not be recognized and we will have to provide a confirmation of the security exception

\_\_ 3. Click **Advanced** button.

\_\_ 4. Click **Add Exception** button.

\_\_ 5. Click **Get Certificate** button.

\_\_ 6. Click **Confirm Security Exception** button.

\_\_7. In user name enter "wasadmin" and in password enter "wasadmin".

Note: enter credentials without quotes. If you are running this command in your organization environment then enter the user/password that your instructor provided to you.

\_\_8. Click close (x) if it prompts you to remember credentials.

Note: There are currently no nodes connected to our chef server. You will add a node later in the lab.

\_\_9. Keep the browser open and switch back to the terminal.

## Part 6 - Verify Chef Development Kit installation

In this part you will verify Chef development kit installation.

\_\_1. Run following command to verify chef development kit is installed.

```
chef verify
```

Notice it shows status of various chef components

\_\_2. Run following command to get version of various chef development kit components.

```
chef --version
```

\_\_3. Run following command to output the chef server version and verify the "knife" tool is available.

```
knife --version
```

## Part 7 - Create a simple standalone chef recipe

In this part you will create a standalone recipe that downloads and installs cowsay, fortune, and tree packages. It also creates a hello.txt file.

\_\_1. Open a new **Terminal** window.

\_\_2. Run following commands and notice they all display error message that corresponding package is not installed.

```
cowsay  
fortune  
tree
```

\_\_3. Run "cat /tmp/hello.txt" and verify it displays a message saying the file doesn't exist.

\_\_4. Run following command to switch to **Documents** directory.

```
cd ~/Documents
```

\_\_5. Run following command to create a directory named **recipes**.

```
mkdir recipes
```

\_\_6. Run following command to switch to **recipes** directory.

```
cd recipes
```

\_\_7. Run following command to launch nano text editor.

```
sudo nano setup.rb
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_8. Type following text.

```
package "cowsay" do
  action :install
end

package 'fortune'

package 'tree'

file '/tmp/hello.txt' do
  content 'Hello World!'
end
```

\_\_9. Press Ctrl+O to save the file and press Enter (do this every time you need to save a file using nano).

\_\_10. Press Ctrl+X to exit to the terminal.

\_\_11. Run the following command to cook the recipe in local mode.

```
sudo chef-client -z /home/wasadmin/Documents/recipes/setup.rb
```

Alternatively, you can use `sudo chef-client --local-mode setup.rb`

Notice it downloads cowsay, fortune, and tree packages and installs it. It also creates hello.txt file.

If the package installation fails then run following command:

```
sudo apt-get update
```

\_\_12. Run following shell command to verify cowsay is installed.

```
cowsay 'Hello World'
```

The following should be displayed



Note. If you get a path error then include /usr/games/ before the command in the following steps, for example:

```
/usr/games/cowsay 'Hello World'
```

\_\_13. Run following command to verify "fortune" package is installed.

```
fortune
```

Notice it displays a random message each time you run it.

\_\_14. Run following command to utilize fortune and cowsay together.

```
fortune | cowsay
```

\_\_15. Run following command to verify "tree" package is installed.

```
tree /home
```

Notice it displays directory tree.

\_\_16. Run following command to verify hello.txt file exists.

```
cat /tmp/hello.txt
```

It shows display Hello World!

Notice chef created /tmp/hello.txt file. Chef recipes can be used for creating configuration files, such as, app.config, web.config, settings.xml, and other similar files.

\_\_17. Run following command to create launch nano text editor.

```
sudo nano remove.rb
```

\_\_18. Type following code.

```
package 'fortune' do
  action :remove
end
```



```
package 'cowsay' do
  action :remove
end

file '/tmp/hello.txt' do
  action :delete
end
```

Notice for packages you use remove action and for file you use delete action. You are purposefully not removing the "tree" package since it will be used later in the lab

\_\_19. Press Ctrl+O to save the file and press Enter.

\_\_20. Press Ctrl+X to exit to the terminal.

\_\_21. Run following command to run the recipe in local mode.

```
sudo chef-client -z remove.rb
```

\_\_22. Run following commands and notice packages do not exist.

```
fortune
cowsay
```

\_\_23. Run "cat /tmp/hello.txt" and verify the file no longer exists.

## Part 8 - Create a chef cookbook and organize recipes

In this part you will create a cookbook and organize recipes.

\_\_1. Launch a new Linux terminal or keep in the same Terminal.

\_\_2. Switch to Documents directory.

```
cd ~/Documents
```

\_\_3. Create a directory for storing cookbooks.

```
mkdir cookbooks
```

\_\_4. Review what **chef** tool can do.

```
sudo chef --help
sudo chef generate --help
```

\_\_5. Generate a cookbook.

```
sudo chef generate cookbook cookbooks/my_cookbook
```

\_\_6. Switch to cookbooks directory.

```
cd cookbooks
```

\_\_7. Get directory list.

```
ls
```

Notice there's my\_cookbook directory

\_\_8. Switch to my\_cookbook directory.

```
cd my_cookbook
```

\_\_9. Get directory list.

```
ls
```

\_\_10. Notice there are various files and folders.

\_\_11. View Berksfile.

```
cat Berksfile
```

Notice it contains source which points to <https://supermarket.chef.io> by default. That's where all the packages are downloaded by default.

\_\_12. View chefignore.

```
cat chefignore
```

Notice it contains various files and directories. These are ignored by chef client. You can add more entries to the file, if required.

\_\_13. View metadata.rb.

```
cat metadata.rb
```

Notice it contains author name, cookbook name, description, license, and version.

\_\_14. View README.md.

```
cat README.md
```

Notice you can add description of your cookbook here.

\_\_15. Switch to recipes directory.

```
cd recipes
```

\_\_16. Get directory list.

```
ls
```

\_\_17. Notice there's default.rb. You can edit this file and define custom recipe, if required.

\_\_18. View default.rb.

```
cat default.rb
```

Notice it contains comments by default.

\_\_19. Switch to recipes directory.

```
cd ~/Documents/cookbooks/my_cookbook/recipes
```

\_\_20. Copy previously created standalone recipes to the recipes directory.

```
sudo cp ~/Documents/recipes/*.rb  
~/Documents/cookbooks/my_cookbook/recipes
```

\_\_21. Get directory list.

```
ls
```

Notice there are five ruby files: default.rb, remove.rb, fortune.rb, cowsay.rb, setup.rb, and tree.rb

\_\_22. Switch to Documents directory.

```
cd ~/Documents
```

\_\_23. Run setup.rb cookbook recipe.

```
sudo chef-client -z -r "recipe[my_cookbook::setup]"
```

Notice it installs fortune, cowsay, and tree packages. It also creates hello.txt file.  
You can also use --recipe instead of -r

\_\_24. Run tree command.

```
tree cookbooks/my_cookbook
```

Notice it displays my\_cookbook's directory tree.

\_\_25. Run remove.rb cookbook recipe.

```
sudo chef-client -z -r "recipe[my_cookbook::remove]"
```

Notice it removes fortune, cowsay, and tree packages. It also deletes hello.txt file.

\_\_26. Run default.rb cookbook recipe.

```
sudo chef-client -z -r "recipe[my_cookbook]"
```

Notice it doesn't do anything since it's an empty recipe.

\_\_27. Close all Terminal windows.

## Part 9 - Review

In this lab you installed and configured chef server, chef development kit, and chef manage web user interface. You also created simple recipes to utilize various resources, such as, package and file.

## Lab 2 - Version Control - Git

In this chapter you will install, configure, and use Git.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

\_\_1. Open the **Terminal** window.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

\_\_2. Run following command to switch to **Downloads** directory.

```
cd ~/Downloads
```

### Part 2 - Install Git by creating a chef cookbook

In this part you will remove and install Subversion and git by creating a chef cookbook.

\_\_1. Switch to cookbooks directory.

```
cd /home/wasadmin/Documents/cookbooks
```

\_\_2. Create a cookbook.

```
sudo chef generate cookbook git_cookbook
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_3. Edit default.rb recipe.

```
sudo nano git_cookbook/recipes/default.rb
```

\_\_4. Append following text.

```
package "git" do
  action :install
end
```

\_\_5. Press Ctrl+O to save the file and hit enter.

\_\_6. Press Ctrl+X to exit to the terminal.

\_\_7. Run the recipe.

```
sudo chef-client -z -r "recipe[git_cookbook]"
```

## Part 3 - Verify Git is installed

In this part you will see how to verify Git installation.

\_\_1. In the terminal run following command to find git version number.

```
git --version
git help -g
git help -a
```

## Part 4 - Using Git

In this part you will use Git to perform various operations, such as, check in, check status etc.

\_\_1. Switch to the "Documents" directory.

```
cd ~/Documents
```

\_\_2. Create a directory.

```
mkdir -p workspace/git
```

\_\_3. Switch to the "git" directory.

```
cd workspace/git
```

\_\_4. Initialize repository.

```
git init
```

\_\_5. Tell Git who you are.

```
git config --global user.name "Alice Smith"
git config --global user.email alice@smith.com
```

Note: One interesting aspect of Git is that it separates user identity in the repository from any sort of authentication or authorization. Because a distributed repository will generally be maintained by many separate individuals or systems, the identity of the committer must be contained in the repository – it can't just be supplied as a user id when we do the commit. So, even if we're not connected to any central repository, we need to tell Git who we are. The identity that we supply will be recorded whenever we commit to a repository.

\_\_6. Create a text file.

```
sudo nano sample.txt
```

\_\_7. Enter following text.

```
First Version!
```

\_\_8. Press Ctrl+O to save the file and hit enter.

\_\_9. Press Ctrl+X to exit to the terminal.

\_\_10. Get Git status.

```
git status
```

Notice sample.txt is listed under untracked files.

\_\_11. Add the files to tracked.

```
git add .
```

Note: Here you are adding the current directory. You could also add the file using "git sample.txt".

\_\_12. Get Git status again.

```
git status
```

Notice sample.txt is tracked.

\_\_13. Commit changes.

```
sudo git commit
```

Notice it launched text editor automatically which lists operations that will get performed when files are committed. Here you can add detailed description that will get saved when you commit the changes.

\_\_14. Add following text in the first line.

```
Added sample.txt
```

\_\_15. Press Ctrl+O to save the file and hit enter.

\_\_16. Press Ctrl+X to exit to the terminal.

\_\_17. Get Git status.

```
git status
```

Notice it says there's nothing to commit since you have already committed all changes.

\_\_18. Modify sample.txt.

```
sudo nano sample.txt
```

\_\_19. Change "First Version!" to "Second Version!"

\_\_20. Press Ctrl+O to save the file and hit enter.

\_\_21. Press Ctrl+X to exit to the terminal.

\_\_22. Get Git status.

```
git status
```

Notice it says the file is modified.

\_\_23. View changes.

```
git diff
```

Notice it shows old text in red and new text in green.

\_\_24. Create another file.

```
sudo nano another.txt
```

\_\_25. Add the following text.



Hello World!

\_\_26. Press Ctrl+O to save the file and hit enter.

\_\_27. Press Ctrl+X to exit to the terminal.

\_\_28. Add all files.

```
git add .
```

\_\_29. Get Git status.

```
git status
```

Notice it's showing 1 file as modified and 1 file as a newly added file.

\_\_30. Commit changes.

```
sudo git commit -m "Made 2 changes"
```

Notice when you pass -m switch, you can store a simple single line comment.

\_\_31. Get Git status.

```
git status
```

Notice there's nothing to commit.

\_\_32. Delete sample.txt

```
sudo rm sample.txt
```

\_\_33. Recover file.

```
git checkout sample.txt
```

\_\_34. View sample.txt

```
cat sample.txt
```

Note: It restored latest version by default.

\_\_35. Delete file again.

```
sudo rm sample.txt
```

\_\_36. View all versions of a file.

```
git log
```

Notice it shows user, commit id, date time, and comment.

\_\_37. Copy the **commit id** for the first version.

\_\_38. View changes between current and the first version.

```
git diff <commit_id>
```

\_\_39. Restore the older version.

```
git checkout <commit_id> sample.txt
```

\_\_40. View file content.

```
cat sample.txt
```

Notice it's the first version.

\_\_41. Close the terminal.

## Part 5 - Review

In this chapter you installed and used Git.

## Lab 3 - Version Control – SVN

In this chapter you will install, configure, and use SVN.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

\_\_1. Open the **Terminal** window.

Alternatively, you can press Ctrl+Alt+T to access the terminal

\_\_2. Run following command to switch to **Downloads** directory.

```
cd ~/Downloads
```

### Part 2 - Install Subversion by creating a chef cookbook

In this part you will remove and install Subversion by creating a chef cookbook.

\_\_1. Switch to cookbooks directory.

```
cd /home/wasadmin/Documents/cookbooks
```

\_\_2. Create a cookbook.

```
sudo chef generate cookbook svn_cookbook
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_3. Edit default.rb recipe.

```
sudo nano svn_cookbook/recipes/default.rb
```

\_\_4. Append following text.

```
package "subversion" do
  action :install
end
```

\_\_5. Press Ctrl+O to save the file and hit enter.

\_\_6. Press Ctrl+X to exit to the terminal.

\_\_7. Run the recipe.

```
sudo chef-client -z -r "recipe[svn_cookbook]"
```

## Part 3 - Verify Subversion is installed

In this part you will see how to verify git and subversion.

\_\_1. In the terminal run following command to find Subversion version number.

```
svn --version  
svn help
```

## Part 4 - Using Subversion

In this part you will use Subversion.

\_\_1. Switch to "workspace" directory.

```
cd ~/Documents/workspace
```

\_\_2. Create SVN repository.

```
svnadmin create svn
```

Note: svn is the repository name.
-----------------------------------

\_\_3. Create source directory where you will place your project files.

```
mkdir -p ~/Documents/workspace/src
```

\_\_4. Change to directory.

```
cd src
```

\_\_5. Checkout a working copy of the repository.

```
svn co file:///home/wasadmin/Documents/workspace/svn  
~/Documents/workspace/src
```

\_\_6. Get history.

```
svn log
```

Notice there's nothing to display since no operation operation has been performed
---

\_\_7. Create a text file.

```
sudo nano sample.txt
```

\_\_8. Enter following text.

```
First Version
```

\_\_9. Press Ctrl+O to save the file and hit enter.

\_\_10. Press Ctrl+X to exit to the terminal.

\_\_11. Add the file to the repository.

```
svn add sample.txt
```

Notice it shows "A sample.txt". "A" means a file has been added.

\_\_12. Get the difference.

```
svn diff
```

Notice it shows sample.txt is available in the working copy and it also shows the text that has been added to the file.

\_\_13. Commit changes.

```
svn commit -m "Added sample.txt"
```

\_\_14. Update the local changes from the repository.

```
svn update
```

\_\_15. Get the log.

```
svn log
```

Notice it shows date time and comment for the previous commit.

\_\_16. Modify the sample.txt file.

```
sudo nano sample.txt
```

\_\_17. Change the text to as shown below.

```
Second Version
```

\_\_18. Press Ctrl+O to save the changes and hit enter.

\_\_19. Press Ctrl+X to exit to the terminal.

\_\_20. Get status.

```
svn status
```

Notice it shows "M sample.txt". "M" means the file was modified.

\_\_21. Commit changes.

```
svn commit -m "Modified sample.txt"
```

\_\_22. Update the local changes.

```
svn update
```

\_\_23. Get the log.

```
svn log
```

Notice it shows both commits.

\_\_24. Close the terminal.

## **Part 5 - Review**

In this chapter you installed and used Git.

## Lab 4 - Continuous Integration

In this chapter you will install and configure Java, Jenkins, Maven, and various SCM plugins for Jenkins.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

1. Open the **Terminal** windows.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

2. Run following command to switch to **Documents** directory.

```
cd ~/Documents
```

### Part 2 - Install Java other prerequisites.

In this part you will install Java and other prerequisites.

1. View the script that will install Java and other prerequisites.

```
cat ~/Documents/labs/prereqs/java.sh
```

2. Run the script.

```
bash ~/Documents/labs/prereqs/java.sh
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Note: Press **Y** if it prompts you to do so.

3. Verify Java is installed.

```
java -version
```

### Part 3 - Install Maven

In this part you will install Maven.

1. Switch to the home directory in the terminal.

```
cd ~
```

2. Create a directory for storing Maven files.

```
mkdir maven
```

\_\_3. Switch to the directory.

```
cd maven
```

\_\_4. Run following command in the terminal to extract Maven archive.

```
sudo tar xvf /home/wasadmin/Downloads/apache-maven-3.3.9-bin.tar.gz -C  
/home/wasadmin/maven --strip-components=1
```

\_\_5. Switch to the bin directory.

```
cd bin
```

\_\_6. Verify Maven is installed.

```
./mvn --version
```

\_\_7. Add maven's bin directory to the path.

```
export PATH=$PATH:/home/wasadmin/maven/bin
```

Note: In this part you installed Maven using the offline install file. If you prefer to do an online install then run following command.

```
sudo apt-get install maven
```

It gets installed in /usr/share/maven directory.



## Part 4 - Using Maven

In this part you will see how to build a simple Java project.

\_\_ 1. Switch to the "Documents" directory.

```
cd ~/Documents
```

\_\_ 2. Unzip sample Java project.

```
unzip ~/Downloads/SimpleGreeting.zip
```

\_\_ 3. Switch to the project folder.

```
cd SimpleGreeting
```

\_\_ 4. Get directory tree.

```
tree
```

Notice there's pom.xml, main/java/com/simple/Greeting.java, and main/java/com/simple/TestGreeting.java.

\_\_ 5. View pom.xml

```
sudo nano pom.xml
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Note: pom.xml is a "Project Object Model" file used by Maven. Here are some prominent entries in the pom.xml file:

packaging: jar

dependencies: defines junit dependency

build plugin: specifies build plugins

Configuration source/target: JDK/JRE version

\_\_ 6. Press Ctrl+X to exit to the terminal.

\_\_ 7. View Greeting.java

```
sudo nano src/main/java/com/simple/Greeting.java
```

Review the code.

\_\_ 8. Press Ctrl+X to exit to the terminal.

\_\_ 9. View the unit test file: TestGreeting.java

```
sudo nano src/test/java/com/simple/TestGreeting.java
```

Review the code.

\_\_ 10. Press Ctrl+X to exit to the terminal.

\_\_ 11. Verify and download Maven plugins.

```
mvn verify
```

Note. Notice it downloads plugins required to build and unit test a Java project.

\_\_ 12. Build and unit test the project.

```
mvn install
```

Notice it builds the jar file and also runs unit tests.

\_\_ 13. Switch to "target" directory.

```
cd target
```

\_\_ 14. Get directory tree of the "target" directory.

```
tree
```

Notice there's SimpleGreeting-1.0-SNAPSHOT.jar file.

\_\_ 15. Go back 1 level.

```
cd ..
```

\_\_ 16. Execute the jar file.

```
java -cp target/SimpleGreeting-1.0-SNAPSHOT.jar com.simple.Greeting
```

Notice it displays "GOOD" on the screen.

\_\_ 17. Clean the project.

```
mvn clean
```

\_\_ 18. Get directory list.

```
ls
```

Notice "target" folder is deleted.

## Part 5 - Install Jenkins

In this part you will install Jenkins using dpkg / apt-get.

\_\_1. In the terminal switch to Downloads directory.

```
cd ~/Downloads
```

\_\_2. Install "daemon" prerequisite package.

```
sudo dpkg -i daemon_0.6.4-1_amd64.deb
```

\_\_3. Install Jenkins package.

```
sudo dpkg -i jenkins_2.9_all.deb
```

\_\_4. Install additional prerequisites, if any.

```
sudo apt-get -f install
```

Note: If you don't have the offline jenkins package then you can follow these steps to install Jenkins.

To use the Debian package repository of Jenkins, first add the key to your system:

```
wget -q -O - http://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

Then add the following entry in your /etc/apt/sources.list:

```
deb http://pkg.jenkins.io/debian binary/
```

Update your local package index, then finally install Jenkins:

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

Note: Following Chef recipe also can be used for installing Jenkins:

```
apt_repository "jenkins" do
```

```
  uri "http://pkg.jenkins-ci.org/debian"
```

```
  key "http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key"
```

```
  components ["binary/"]
```

```
  action :add
```

```
end
```

```
package "jenkins" do
  action :install
end
service "jenkins" do
  action [[:enable, :start]]
end
```

## Part 6 - Configure Jenkins and start the service

In this part you will modify the service configuration file to start Jenkins as "root". Although it's not recommended to do so in production environment but you will do so to keep the setup a bit simpler. Detailed security configuration is out of scope of this course.

\_\_ 1. Edit Jenkins service configuration file.

```
sudo nano /etc/init.d/jenkins
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_ 2. Press Ctrl+\ to access search & replace option.

\_\_ 3. Enter "\$JENKINS\_USER" (without quotes) in "to replace" and hit enter.

\_\_ 4. Enter "root" (without quotes) in "replace with" and hit enter.

\_\_ 5. Click A to replace All and hit enter.

\_\_ 6. Press Ctrl+O to save the file and hit enter.

\_\_ 7. Press Ctrl+X to exit to the terminal.

\_\_ 8. Start Jenkins service.

```
sudo service jenkins restart
```

## Part 7 - Launch Jenkins, install plugins, and configure Maven

\_\_ 1. Launch Firefox from the task bar and enter following URL.

```
http://localhost:8080
```

Notice Jenkins is currently locked. It needs to be unlocked. Make note of the directory displayed on the page

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

\_\_ 2. Get the access key.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Notice it displays random text as shown in picture.

```
student@student-VirtualBox:~/chef-repo/cookbooks/mywrapper/recipes$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
[sudo] password for student:
a08bab9fae0644cd872f2cd4e0a1d2d5
```

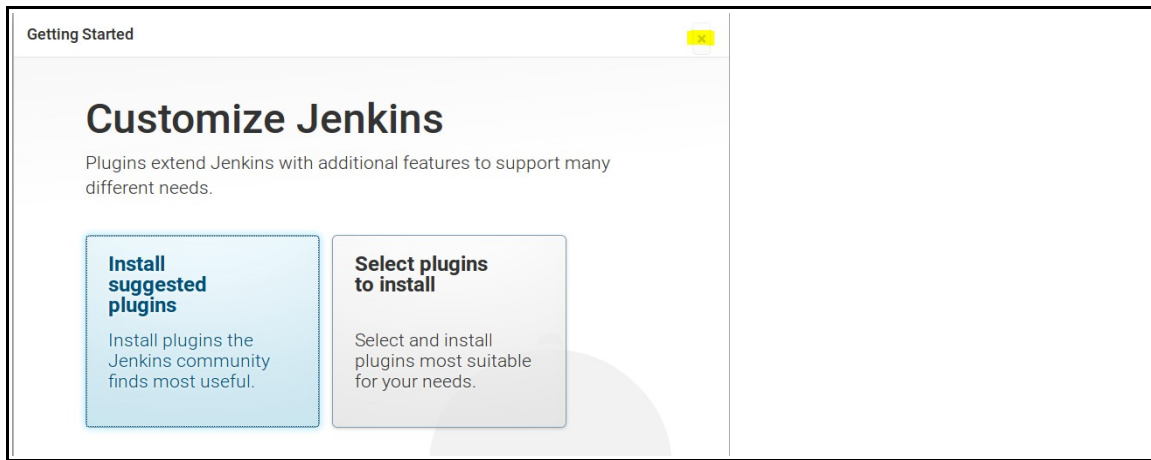
\_\_ 3. Highlight the text and copy it to the clipboard.

\_\_ 4. Paste the text in the "Administrative password" text box and press continue.

\_\_ 5. Close the save password window.

\_\_ 6. Click x on "Customize Jenkins" page.

Note: You will install plugins later in this lab.



- \_\_ 7. Click "Start using Jenkins" button.
- \_\_ 8. On left side of the page, click "Manage Jenkins".
- \_\_ 9. Click "Manage Plugins".
- \_\_ 10. Click "Available" tab.
- \_\_ 11. In "Filter" text box, enter "maven integration plugin".
- \_\_ 12. Select check box in front of "Maven Integration plugin".
- \_\_ 13. Click "Install without restart".

Note: Stay on the page until it shows "Success" status as shown below.



- \_\_ 14. On left side the page click "Manage Plugins".
- \_\_ 15. Click "Available" tab.
- \_\_ 16. In the "Filter" text box, enter "SVN".
- \_\_ 17. Select check box in front of "SVN Publisher plugin" and click "Install without restart".

Note: Stay on the page until it shows "Success" status for plugins / dependencies.

- \_\_ 18. On left side of the page, click "Manage Jenkins".

- \_\_19. Click "Global Tool Configuration".
- \_\_20. Under Maven click "Add Maven".
- \_\_21. Uncheck "Install Automatically" checkbox.
- \_\_22. In "Name" text box, enter "Maven".
- \_\_23. In "MAVEN\_HOME" text box, enter.

`/home/wasadmin/maven`

- \_\_24. Click "Save" button.

## **Part 8 - Create and Run a Jenkins Job**

In this part you will see how to create and run a Jenkins job.

- \_\_1. In Firefox type in following URL.


`http://localhost:8080`

- \_\_2. On left side of the page, click "New Item".
- \_\_3. In "Enter an item name" enter "Maven Test" and select "Maven project", as shown below, and click OK button.

## Enter an item name


Maven Test

» Required field




**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.




**Maven project**

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



**External Job**

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).



**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

\_\_4. Scroll down on the page and locate "Build" section.

\_\_5. In "Root POM" text box, enter.

```
/home/wasadmin/Documents/SimpleGreeting/pom.xml
```

\_\_6. In "Goals and options" text box, enter "clean verify install" (without quotes).

\_\_7. Click "Save" button.

\_\_8. Click "Build Now".

Notice the status of the build.

## Part 9 - Integrating Git with Jenkins

In this part you will integrate Git with Jenkins.

\_\_1. In the terminal switch to the SimpleGreeting folder.

```
cd ~/Documents/SimpleGreeting
```

\_\_2. Initialize Git repository.



```
git init
```

\_\_3. Add content to the repository.

```
git add .
```

\_\_4. Commit changes.

```
git commit -m "Added files"
```

\_\_5. Enter this commands.

```
cd /var/lib/jenkins  
sudo nano config.xml
```

\_\_6. Change <useSecurity> from true to **false**.

\_\_7. Press Ctrl+O to save the file and hit enter.

\_\_8. Press Ctrl+X to exit to the terminal.

\_\_9. Restart Jenkins by running following command in the terminal.

```
sudo service jenkins restart
```

\_\_10. Launch Firefox from the task bar and enter following URL.

```
http://localhost:8080
```

\_\_11. Click **Manage Jenkins**.

\_\_12. Click **Manage Plugins**.

\_\_13. Click **Available** tab.

\_\_14. In filters search for **git plugin**

\_\_15. Scroll down and check box for **Git plugin**.

\_\_16. Click **Install without restart**.

\_\_17. Wait until installation is completed.

\_\_18. Click **Back to Dashboard**.

\_\_19. Click "Maven Test" job that you created in Jenkins exercise.

\_\_20. On left side the page, click "Configure".

\_\_21. Click "Source Code Management" tab.

\_\_22. Select "Git" radio button.

\_\_23. In "Repository URL" enter "file:///home/wasadmin/Documents/SimpleGreeting"

(without quotes).

\_\_24. Click "Build Triggers" tab.

\_\_25. Select check box in front of "Poll SCM".

\_\_26. In "Schedule" enter "\* \* \* \* \*" (without quotes and with spaces between the stars).

Note: It will make Jenkins poll the SCM every minute

\_\_27. Click "Save" button.

\_\_28. In the terminal edit Greeting.java file.

```
sudo nano
~/Documents/SimpleGreeting/src/main/java/com/simple/Greeting.java
```

\_\_29. Change the message "GOOD" to "VERY GOOD".

\_\_30. Press Ctrl+O to save the file and hit enter.

\_\_31. Press Ctrl+X to exit to the terminal.

```
cd ~/Documents/SimpleGreeting
```

\_\_32. Add changes to the repository.

```
git add .
```

\_\_33. Commit the changes.

```
git commit -m "updated message"
```

\_\_34. Go back to the Jenkins page in Firefox.

Wait for a minute or so and notice another build shows up in Build History.

\_\_35. On left side of the page click "Git Polling Log".

Note: Here you can see when the Git repository was polled.

\_\_36. On left side of the page, click the latest build in Build History.

\_\_37. On left side of the page, click "Console Output".

Notice Jenkins connected to Git, compiled the code, and ran unit tests.

\_\_38. Close the web browser and terminal.

## Part 10 - Review

In this lab you installed and configured Java, Jenkins, Maven, and various plugins for Jenkins. You also integrated Git with Jenkins.

## Lab 5 - Install Prerequisites

In this lab you will install the Apache web server, MySQL, and PHP. They are required by Continuous Code Quality (SonarQube) and Continuous Application Monitoring (Nagios) tools.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

- \_\_\_ 1. In the task bar, click **Search** button.
- \_\_\_ 2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_\_ 3. Run following command to switch to **Downloads** directory.

```
cd ~/Downloads
```

### Part 2 - Install Apache web server

In this part you will install Apache web server.

- \_\_\_ 1. Update APT repository.

```
sudo apt-get update
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

- \_\_\_ 2. Install Apache web server.

```
sudo apt-get install apache2
```

Note: Press Y to continue installation.

Since we have a service running on port 80, it will show errors. If this were unexpected, you could run `netstat -an | grep 80` to verify what is running and take appropriate action.

- \_\_\_ 3. Edit ports.conf file.

```
sudo nano /etc/apache2/ports.conf
```

You will modify the default HTTP and HTTPS ports.

- \_\_\_ 4. Change "80" to "1080" and "443" to "10443"
- \_\_\_ 5. Press Ctrl+O to save the file and hit Enter.
- \_\_\_ 6. Press Ctrl+X.

\_\_7. Edit the virtualhost file.

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Next you will modify the default http port.

\_\_8. Change 80 to 1080.

\_\_9. Press Ctrl+O to save the file and hit Enter.

\_\_10. Press Ctrl+X to exit to the terminal.

\_\_11. Edit apache2.conf.

```
sudo nano /etc/apache2/apache2.conf
```

\_\_12. Append following text to the bottom of the file to allow the server to identify itself and eliminate DNS binding errors for hostname.

```
ServerName localhost
```

\_\_13. Press Ctrl+O to save the file and hit Enter.

\_\_14. Press Ctrl+X to exit to the terminal.

\_\_15. Restart Apache web server.

```
sudo service apache2 restart
```

\_\_16. Launch nano text editor and create a test page.

```
sudo nano /var/www/html/test.html
```

\_\_17. Type in following text.

```
<h1>Hello world!</h1>
```

\_\_18. Press Ctrl+O to save the file and hit Enter.

\_\_19. Press Ctrl+X to exit to the terminal.

\_\_20. Launch Firefox web browser from the task bar and type in following URL.

```
http://localhost:1080/test.html
```

Notice the test page shows up.

If you want to install Apache web server using a chef cookbook then use following recipe.

```
# install Apache web server
```

```

package "apache2" do
  action :install
end
# start the web server
service "apache2" do
  action [:enable, :start]
end

# create a test file
file "/var/www/test.html" do
  content "<html><head><title>Test
Page</title></head><body><h1>Hello World!
</h1></body></html>"
end

```

### Part 3 - Install MySQL

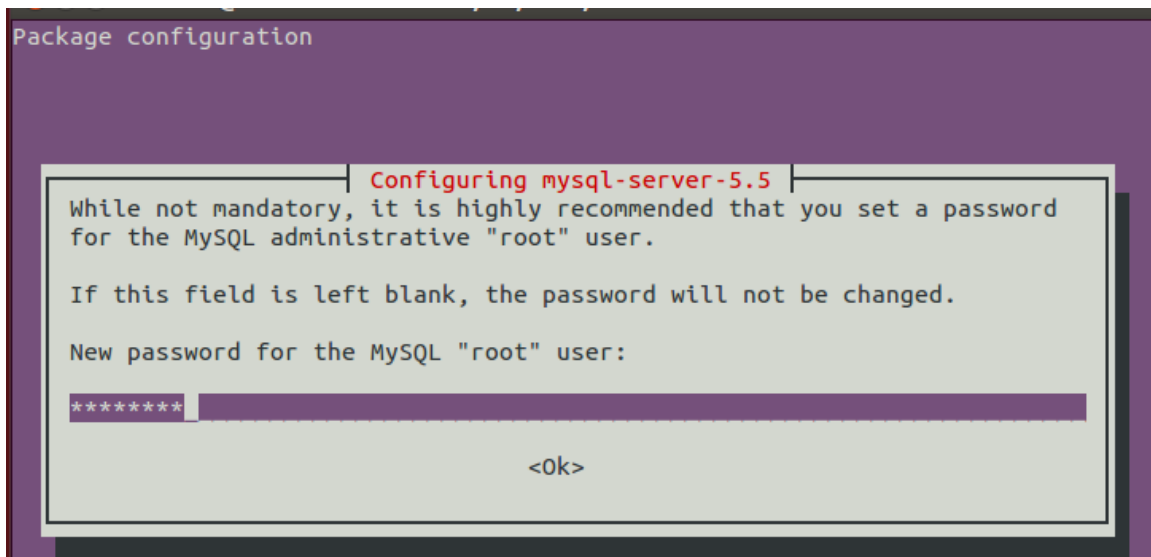
In this part you will install MySQL.

\_\_\_1. Install MySQL.

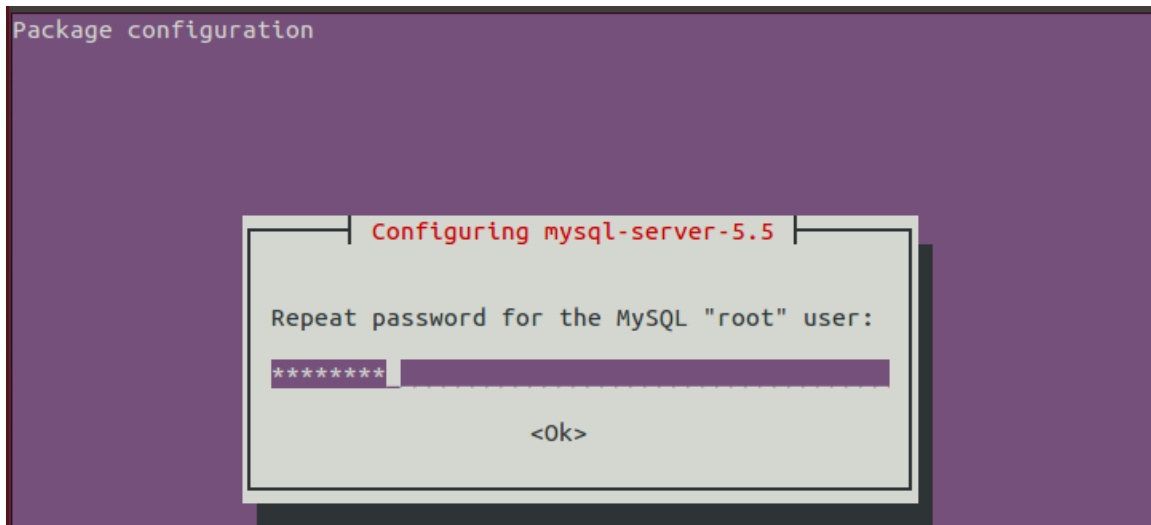
```
sudo apt-get install mysql-server-5.6 php5-mysql
```

\_\_\_2. Press Y to continue installation.

\_\_\_3. Enter **wasadmin** when prompted to enter the "root" password.



\_\_\_4. Enter **wasadmin** again when prompted to reenter the "root" password.



\_\_5. Secure the installation by running following command.

```
sudo mysql_secure_installation
```

\_\_6. Enter "**wasadmin**" (without quotes) when prompted to enter the "root" password.

\_\_7. Press "n" when prompted to change the "root" password.

\_\_8. Press "Y" when prompted to remove anonymous users.

\_\_9. Press "Y" when prompted to disallow remote root login.

\_\_10. Press "Y" when prompted to remove test database.

Ignore error messages. You don't have a test database.

\_\_11. Press "Y" when prompted to reload privileges table.

\_\_12. Restart MySQL service.

```
sudo service mysql restart
```

\_\_13. Connect to the MySQL server.

```
sudo mysql -u root -p
```

\_\_14. Enter "wasadmin" (without quotes) when prompted to enter the password.

\_\_15. At "mysql" prompt, run following command to obtain MySQL version.

```
SELECT VERSION();
```

Note: It should 5.6

\_\_16. Run following command to get list of databases.

```
show databases;
```

Notice it shows result like this.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.00 sec)
```

\_\_17. Type "quit" to exit to the terminal.

## Part 4 - Install PHP

In this part you will install and configure PHP.

\_\_1. Install PHP and helper packages.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Press Y to continue installation.

\_\_2. Edit Apache web server's dir.conf file to change default file.

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

\_\_3. Change the line that has "DirectoryIndex" so that index.php appears before index.html

\_\_4. Press Ctrl+O to save the file and hit Enter.

\_\_5. Press Ctrl+X to exit to the terminal.

\_\_6. Restart Apache web server.

```
sudo service apache2 restart
```

\_\_7. Launch nano text editor to create a same PHP file.

```
sudo nano /var/www/html/index.php
```


\_\_8. Type following text.

```
<?php phpinfo(); ?>
```

- \_\_ 9. Press Ctrl+O to save the file and hit Enter.
- \_\_ 10. Press Ctrl+X to exit to the terminal.
- \_\_ 11. Launch Firefox from the task bar and enter following URL.

`http://localhost:1080/index.php`

Notice it shows a page like this.

<b>PHP Version 5.5.9-1ubuntu4.17</b>	
	
<b>System</b>	Linux student-VirtualBox 4.2.0-27-generic #32~14.04.1-Ubuntu SMP Fri Jan 22 15:32:26 UTC 2016 x86_64
<b>Build Date</b>	May 19 2016 19:05:33
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	disabled
<b>Configuration File (php.ini) Path</b>	/etc/php5/apache2
<b>Loaded Configuration File</b>	/etc/php5/apache2/php.ini
<b>Scan this dir for additional .ini files</b>	/etc/php5/apache2/conf.d
<b>Additional .ini files parsed</b>	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-readline.ini
<b>PHP API</b>	20121113

- \_\_ 12. Remove the test file.

```
sudo rm /var/www/html/index.php
```

- \_\_ 13. Close the Terminal and web browser.

## Part 5 - Review

In this lab you installed and configured Apache web server, MySQL, and PHP.



## Lab 6 - Continuous Code Quality - SonarQube

In this lab you will install, configure, and use SonarQube server, SonarQube Scanner, and Maven to check code quality.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

- \_\_\_ 1. In the task bar, click **Search** button.
- \_\_\_ 2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_\_ 3. Run following command to switch to **Downloads** directory.

```
cd ~/Downloads
```

### Part 2 - Create SonarQube database and user

In this part you will create SonarQube database and user.

- \_\_\_ 1. Connect to MySQL.

```
sudo mysql -u root -p
```

- \_\_\_ 2. Enter twice "wasadmin" (without quotes) when prompted to enter the "root" password.

- \_\_\_ 3. Create database.

```
CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;
```

- \_\_\_ 4. Create a user.

```
CREATE USER 'wasadmin' IDENTIFIED BY 'wasadmin';
```

- \_\_\_ 5. Grant permission on the database to the user.

```
GRANT ALL ON sonar.* TO 'wasadmin'@'%' IDENTIFIED BY 'wasadmin';  
GRANT ALL ON sonar.* TO 'wasadmin'@'localhost' IDENTIFIED BY 'wasadmin';
```

- \_\_\_ 6. Reload privileges.

```
FLUSH PRIVILEGES;
```

- \_\_\_ 7. Type the following command and press enter to exit to the terminal.

```
exit
```

### Part 3 - Extract SonarQube archive

In this part you will extract SonarQube archive.

\_\_1. Switch to the home directory.

```
cd ~
```

\_\_2. Unzip the archive.

```
sudo unzip /home/wasadmin/Downloads/sonarqube-5.5.zip
```

\_\_3. Move the archive to "opt" directory.

```
sudo mv sonarqube-5.5 /opt/sonar
```

### Part 4 - Configure SonarQube

In this part you will configure SonarQube.

\_\_1. Edit sonar.properties.

```
sudo nano /opt/sonar/conf/sonar.properties
```

\_\_2. Find sonar.jdbc.username and uncomment it by removing # in front of the property. Also set user name. It should read as follows.

```
sonar.jdbc.username=wasadmin
```

\_\_3. Find sonar.jdbc.password and uncomment it by removing # in front of the property. Also set user name. It should read as follows.

```
sonar.jdbc.password=wasadmin
```

\_\_4. In MySQL section, uncomment sonar.jdbc.url by removing # in front of the property.

It should read as follows.

```
sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?  
useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfig  
s=maxPerformance
```

\_\_5. In "Web Server" of the configuration file find sonar.web.host then uncomment and set it as follows.

```
sonar.web.host=127.0.0.1
```

\_\_6. In "Web Server" of the configuration file find sonar.web.context then uncomment and set it as follows.

```
sonar.web.context=/sonar
```

Note: /sonar will act as the virtual directory for accessing SonarQube's web user interface.

\_\_7. In "Web Server" of the configuration file find sonar.web.port then uncomment and set it as follows.

```
sonar.web.port=1090
```

\_\_8. Press CTRL+O to save the file and hit Enter.

\_\_9. Press CTRL+X to exit.

## **Part 5 - Run SonarQube server and connect to the web user interface**

In this part you will start SonarQube server and connect its web user interface.

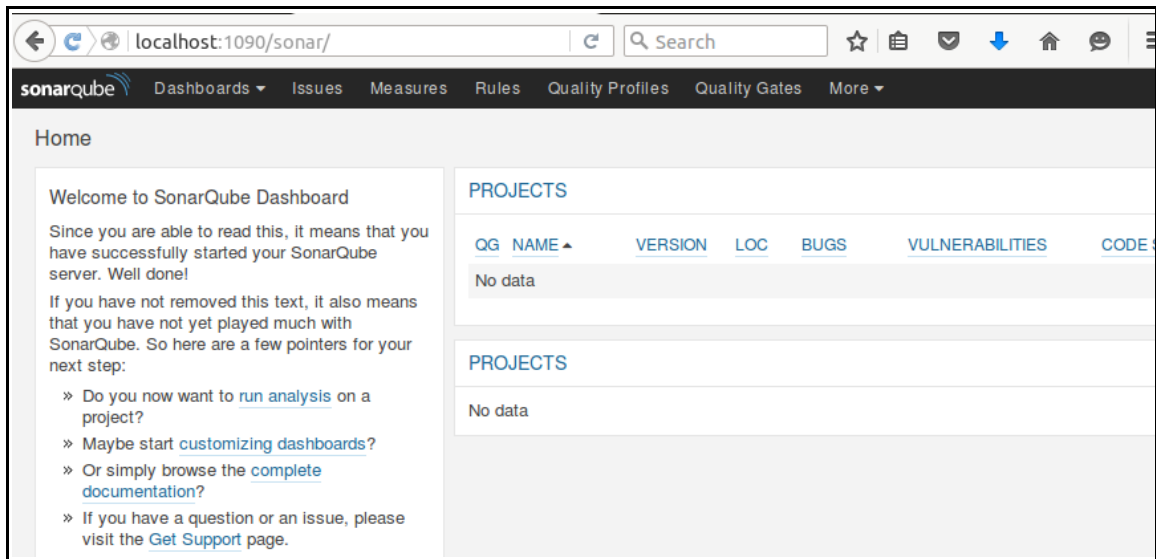
\_\_1. Start SonarQube server.

```
sudo /opt/sonar/bin/linux-x86-64/sonar.sh start
```

\_\_2. Start Firefox from the task bar and enter following URL (it may take a while to load the page).

```
http://localhost:1090/sonar
```

Notice the web page shows up like this.



\_\_\_3. Close the web browser and Terminal.

## Part 6 - Review

In this lab you installed, configured, and used SonarQube Serve, SonarQube Scanner, and Maven to check code quality.

## Lab 7 - Automation (Shell Scripting)

In this chapter you will explore basics of Bash shell scripting.

### Part 1 - Launch the shell terminal and create a folder for saving scripts

In this part you will launch the Linux terminal.

- \_\_\_ 1. In the task bar, click **Search** button.
- \_\_\_ 2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_\_ 3. Run following command to switch to **Documents** folder.

```
cd ~/Documents
```

- \_\_\_ 4. Run following command to create a folder for storing lab files.

```
mkdir -p labs
```

- \_\_\_ 5. Copy scripts from **Downloads** directory.

```
cp -r ~/Downloads/labs/shell ~/Documents/labs/shell
```

- \_\_\_ 6. Switch to **shell** directory.

```
cd labs/shell
```

- \_\_\_ 7. Get directory list.

```
ls
```

Notice there are a few .sh files in the directory.

### Part 2 - Execute shell commands in interactive mode without creating a script file

In this part you will run shell commands without creating a script file.

- \_\_\_ 1. Run following command to display "Hello World!" message.

```
echo "Hello World!"
```

- \_\_\_ 2. Run following command to display "Hello World!" in fancy colors.

```
echo -e "\e[31;43m Hello World! \e[0m"
```

Note: Syntax: `echo -e "\e[COLOR1;COLOR2m<YOUR TEXT HERE>\e[0m"`

More details are available on: [https://wiki.archlinux.org/index.php/Color\\_Bash\\_Prompt](https://wiki.archlinux.org/index.php/Color_Bash_Prompt)

\_\_3. Run following to create a variable, display it's value, and unset the variable.

```
A=1
echo $A
unset A
echo $A
```

\_\_4. Run following to create variables, add them up, and display result.

```
A=5
B=6
expr $A + $B
```

### Part 3 - Execute a basic script that prints "Hello world!"

In this part you will execute your first shell script that prints "Hello world!" message.

\_\_1. View hello.sh file content.

```
cat hello.sh
```

Notice there's a variable and a statement for displaying variable's value on the console.

\_\_2. Run following command to execute the script and verify it prints "Hello World!" message.

```
bash hello.sh
```

### Part 4 - Alternative way of executing shell scripts

In this part you will execute shell script without utilizing "bash" as part of the command line.

\_\_1. View hello2.sh file content.

```
cat hello2.sh
```

\_\_2. Run following command to edit hello2.sh.

```
sudo nano hello2.sh
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Notice there is `#!/bin/bash` in the first line. This allows execution of bash shell scripts without specifying bash as part of the command-line.

\_\_ 3. Hit Ctrl + X to exit.

\_\_ 4. Run following command to execute the script.

```
./hello2.sh
```

Notice it displays Permission denied message. You have to grant appropriate permissions to the script before it can be executed.

\_\_ 5. Run following command to obtain security permissions for your script.

```
ls -l
```

Note: Permissions are listed in the first column.

The first character states what the item is: a 'd' means directory and a '-' means file.

The next 3 characters tell whether the owner has given himself permission to read, write, or execute the file. First character 'r' means read permission is granted, '-' means read permission is denied. Second character 'w' means write permission is granted, '-' means write permission is denied. Third character 'x' means execute permission is granted, '-' means execute permission is denied. In case of a directory, 'x' means directory is searchable.

The next 3 characters tell you what permission the user has granted to other members of his group.

The last 3 characters give permissions that rest of the people has.

The first 3 characters for hello2.sh should be -rw which means it's a file with read and write permission granted to the wasadmin user.

\_\_ 6. Run following command to grant your script read, write, and execute permission.

```
chmod +x hello2.sh
```

Note: If you want your script to be private (i.e. only you can read and execute), use `u=+x` instead

\_\_ 7. Grant execute permission to all shell scripts in the directory.

```
sudo chmod +x *.sh
```

Note you are granting execute permission to all scripts since you will be executing them later in the lab.

\_\_ 8. Run following command to obtain security permissions for your script.

```
ls -l
```

Note: After executing the command the first 4 characters should be -rwx which means it's a file with read, write, and execute permissions.

\_\_ 9. Run following command to execute the script and verify it prints "Hello World!".

```
./hello2.sh
```

\_\_ 10. Run following command to try executing the script without specifying path.

```
hello2.sh
```

Notice it prints 'command not found' message. If you want to execute a script without specifying path then it should be added to the path.

\_\_ 11. Run following command to obtain directories in path.

```
echo $PATH
```

\_\_ 12. Run following command to add /home/wasadmin/Documents/shell directory to the path.

```
export PATH=$PATH:/home/wasadmin/Documents/labs/shell
```

\_\_ 13. Run following command to obtain directories in path and verify your custom shell directory is in the path.

```
echo $PATH
```

\_\_ 14. Run following command to execute your script.

```
hello2.sh
```

## Part 5 - Get input from user

In this part you will execute a shell script that utilizes variables, gets input from user, creates a directory, lists directory contents, and delete the directory.

\_\_ 1. Launch the terminal if it's not already running.

\_\_ 2. Run following command to switch to **shell** directory.

```
cd ~/Documents/labs/shell
```



\_\_3. View my\_dir.sh file content.

```
cat my_dir.sh
```

Notice the script prompts user to enter directory name, creates the directory then deletes the directory.

\_\_4. Type following to execute the script.

```
my_dir.sh
```

\_\_5. When prompted to enter directory name, type "test" (without quotes) and press enter key on the keyboard.

Notice it creates the test directory, list the contents of current directory, deletes the directory and relists the contents of the current directory.

## Part 6 - Execute a script that utilizes functions and return keyword

In this part you will execute a script that utilizes functions. You will see how to return numeric and string output.

\_\_1. View functions.sh file content.

```
cat functions.sh
```

Notice the script has various function definitions and function calls.

\_\_2. Run following to execute the script.

```
functions.sh
```

Notice it prints 11 and Hello World! Messages.

## Part 7 - Conditional Statements

In this part you will utilize if..else if..else and case conditional statements.

\_\_1. View if.sh file content.

```
cat if.sh | more
```

Notice the script reads command-line arguments and use if-else to decide what to print on the screen.!

\_\_2. Run following command and notice it displays the usage for the script.

```
if.sh -invalid invalid
```

\_\_3. Run following command notice it displays directory contents.

```
if.sh -p l
```

\_\_4. Run following command notice it displays the date.

```
if.sh -p d
```

\_\_5. Run following command notice it displays the calendar.

```
if.sh -p c
```

\_\_6. View case.sh file content.

```
cat case.sh | more
```

Notice it use case syntax to display directory tree, date, or calendar depending on command-line argument passed to the script.

\_\_7. Run following command and notice it displays the usage for the script.

```
case.sh -invalid invalid
```

\_\_8. Run following command notice it displays directory contents.

```
case.sh -p l
```

\_\_9. Run following command notice it displays the date.

```
case.sh -p d
```

\_\_10. Run following command notice it displays the calendar.

```
case.sh -p c
```

## Part 8 - Arrays and Loops

In this part you will create a script that utilizes arrays and loops.

\_\_1. View for1.sh file content.

```
cat for1.sh | more
```

Notice it uses `for` loop to display directory list.

\_\_2. Run following command to execute the script.

```
for1.sh
```

Notice it displays directory contents of `/home/wasadmin`

\_\_3. View `for1.sh` file content.

```
cat for2.sh | more
```

Notice it uses `for` loop to display array contents.

\_\_4. Run following command to execute the script.

```
for2.sh
```

Notice "`for`" loop displays contents of the "`DIRS`" array.

\_\_5. View `for1.sh` file content.

```
cat while.sh | more
```

Notice it uses `for` loop to display array contents.

\_\_6. Run following command to execute the script.

```
while.sh
```

Notice "`while`" loop displays contents of the "`DIRS`" array.

\_\_7. View `for1.sh` file content.

```
cat until.sh | more
```

Notice it uses `until` loop to display array contents.

\_\_8. Run following command to execute the script.

```
until.sh
```

Notice "`until`" loop displays contents of the "`DIRS`" array.

\_\_ 9. View for1.sh file content.

```
cat select.sh | more
```

Notice it uses `until` loop to display array contents.

\_\_ 10. Run following command to execute the script.

```
select.sh
```

Notice "select" loop is used to display directory list, date, or calendar depending on user input selection.

\_\_ 11. Press 1 and notice it displays directory contents.

\_\_ 12. Press 2 and notice it displays date.

\_\_ 13. Press 3 and notice it displays calendar.

\_\_ 14. Press 7 and notice it displays error message.

\_\_ 15. Press 4 and notice it exits to the terminal.

## Part 9 - Redirection

In this part you will write a script that utilizes "df" command to retrieve file system usage and write script output to a .html file.

\_\_ 1. Run the disk usage command to view file system usage for a folder.

```
du -h
```

\_\_ 2. Run the disk free command to view file system utilization.

```
df -h
```

Note: In this part you will retrieve values for columns 1, 2, and 5.

\_\_ 3. View html.sh file contents.

```
cat html.sh
```

Notice shell scripting is used to redirect output of shell commands to HTML file.

\_\_ 4. Run following command to execute the script.

```
html.sh
```

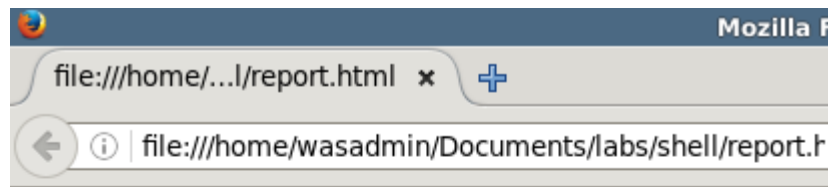
\_\_ 5. In the Linux task bar, click **Files** to launch file system explorer.

\_\_ 6. Browse to Documents/labs/shell or use the complete path as

/Computer//home/wasadmin/Documents/labs/shell

\_\_\_7. Double click report.html (Note: ensure that you have Firefox or some other web browser installed and set as default web browser)

\_\_8. Verify the output looks as show below. (Note: values will be different)



Filesystem usage for host **trn263**  
Last updated: **Wed Oct 26 13:15:22 CDT 2016**

Filesystem	Size	Use %
udev	1.5G	1%
tmpfs	301M	1%
/dev/sda1	66G	17%
none	4.0K	0%
none	5.0M	1%
none	1.5G	1%
none	100M	1%

\_\_9. Close the web browser and the terminal.

## Part 10 - Review

In this lab you used various Linux shell commands. You also saw how to write shell scripts and utilize redirection to write to a .html file.

## Lab 8 - Tomcat Application Deployment using Chef

In this lab you will install Tomcat and use Chef to deploy a sample Tomcat application.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

- \_\_ 1. Open the **Terminal** windows.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_ 2. Run following command to switch to **Downloads** directory.

```
cd ~/Documents
```

### Part 2 - Install Tomcat.

In this part you will install Tomcat.

- \_\_ 1. View the script that will install Tomcat.

```
cat ~/Documents/labs/tomcat/install.sh
```

- \_\_ 2. Run the script.

```
bash ~/Documents/labs/tomcat/install.sh
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Note: Press **Y** if it prompts you to do so.

### Part 3 - Edit Tomcat's server.xml file and start the service

In this part you will edit Tomcat's configuration file and start the service.

- \_\_ 1. Run following command to edit server.xml.

```
sudo nano /opt/tomcat/conf/server.xml
```

- \_\_ 2. Find text "Connector port" and change "8080" to "8082".

- \_\_ 3. Press Ctrl+O to save the file and hit enter.

- \_\_ 4. Press Ctrl+X to exit to the terminal.

- \_\_ 5. Start Tomcat by running following command in the terminal.

```
cd /opt/tomcat/bin
```

```
sudo ./catalina.sh start
```

\_\_6. Verify Tomcat is running.

```
sudo ./catalina.sh version
```

\_\_7. Launch Firefox from the task bar and enter following URL.

```
http://localhost:8082
```

\_\_8. Click **Manage App** button.

Notice it prompts you to enter credentials. You currently don't have a user configured. You will do it in next step.

\_\_9. Click Cancel.

\_\_10. Edit tomcat-users.xml file.

```
sudo nano /opt/tomcat/conf/tomcat-users.xml
```

\_\_11. In <tomcat-users> tag enter following text.

```
<user username="wasadmin" password="wasadmin" roles="manager-gui,
admin-gui"/>
```

\_\_12. Press Ctrl+O to save the file and hit enter.

\_\_13. Press Ctrl+X to exit to the terminal.

\_\_14. Restart Tomcat by running following command in the terminal.

```
cd /opt/tomcat/bin
sudo ./catalina.sh stop
sudo ./catalina.sh start
```

\_\_15. Launch Firefox from the task bar and enter following URL.

```
http://localhost:8082
```

\_\_16. Click **Manage App** button.

\_\_17. In user name enter "wasadmin" and in password enter "wasadmin". Do not save the password.

Notice Tomcat's Application Manager page shows up.

Note: If you want to install Tomcat 8 using chef then following cookbook can be used:  
<https://supermarket.chef.io/cookbooks/tomcat>.



```
tomcat_install 'my_tomcat' do
  version '8.0.36'
end
tomcat_service 'my_tomcat' do
  action [:enable, :start]
end
```

The recipe installs Tomcat 8.0.36 instance named 'my\_tomcat' to /opt/tomcat\_my\_tomcat\_8\_0\_36/ with a symlink at /opt/tomcat\_my\_tomcat/

## Part 4 - Creating a Chef cookbook that deploys a sample Tomcat application

In this part you will create a Chef cookbook that deploys a Tomcat application.

\_\_1. Switch to the "cookbooks".

```
cd ~/Documents/cookbooks
```

\_\_2. Create a cookbook.

```
sudo chef generate cookbook my_tomcat_app_deploy
```

\_\_3. Create a recipe which will deploy Tomcat application.

```
sudo chef generate recipe my_tomcat_app_deploy deploy.rb
```

\_\_4. Edit the recipe.

```
sudo nano my_tomcat_app_deploy/recipes/deploy.rb
```

\_\_ 5. Enter following code.

```
remote_file "Deploy Tomcat Application" do
  source "file:///home/wasadmin/Downloads/sample.war"
  path "/opt/tomcat/webapps/sample.war"
  owner 'root'
  group 'root'
  mode 0755
end

bash "Deploy Tomcat Application - Technique 2" do
  code <<-EOL
  cp ~/Downloads/sample.war ~/Documents/sample.war
  EOL
end
```

\_\_ 6. Press Ctrl+O to save the file and hit enter.

\_\_ 7. Press Ctrl+X to exit to the terminal.

\_\_ 8. Run the cookbook recipe.

```
sudo chef-client -z -r "recipe[my_tomcat_app_deploy::deploy]"
```

\_\_ 9. Verify sample.war exists in Documents directory.

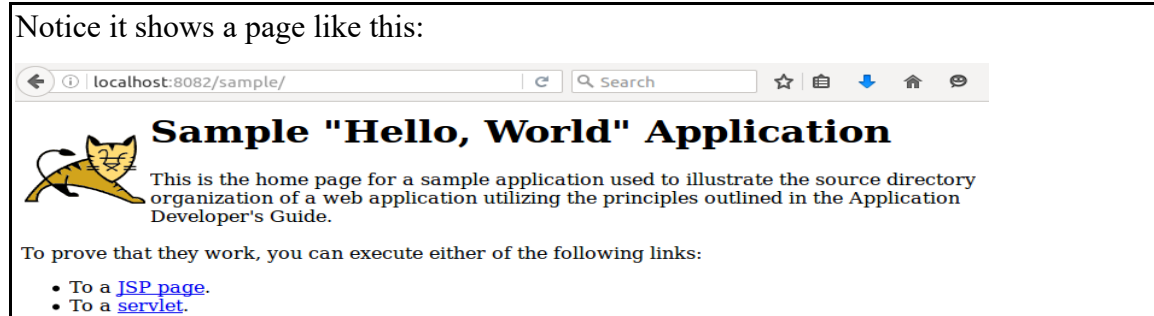
```
ls ~/Documents/sample.war
```

\_\_ 10. Verify sample.war exists in Tomcat's webapps directory.

```
ls /opt/tomcat/webapps/sample.war
```

\_\_ 11. Launch Firefox from the task bar and enter following URL to verify sample Hello World application is accessible.

```
http://localhost:8082/sample
```



\_\_ 12. Close all.

## Part 5 - Review

## Lab 9 - Continuous Monitoring - Nagios

In this lab you will install, configure, and use Nagios to monitor servers and services

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

- \_\_ 1. In the task bar, click **Search** button.
- \_\_ 2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_ 3. Run following command to switch to **Downloads** directory.

```
cd ~/Downloads
```

### Part 2 - Execute script for installing Nagios and Nagios plugins.

In this part you will execute a script that will create user account for using Nagios, install Nagios, and install Nagios plugins.

- \_\_ 1. View installation script.

```
cat ~/Documents/labs/nagios/install.sh | more
```

- \_\_ 2. Run the script.

```
bash ~/Documents/labs/nagios/install.sh
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Enter **Y** if prompted to do so.

### Part 3 - Configure Nagios

In this part you will configure Nagios.

- \_\_ 1. Open nagios.cfg file in text editor.

```
sudo nano /usr/local/nagios/etc/nagios.cfg
```

- \_\_ 2. Uncomment the line `#cfg_dir=/usr/local/nagios/etc/servers` by removing `#` symbol.

Note: This is the directory where you will create a configuration file for each host you want to monitor. You will use this directory later in this lab.

- \_\_ 3. Press Ctrl+O to save the file and hit enter.
- \_\_ 4. Press Ctrl+X to exit to the terminal.

\_\_5. Create directory where you will store configuration file for each server that you will monitor.

```
sudo mkdir /usr/local/nagios/etc/servers
```

\_\_6. Review contacts.cfg.

```
sudo nano /usr/local/nagios/etc/objects/contacts.cfg
```

Scroll down and notice there's define\_contact statement that defines email address and other properties.

\_\_7. Press Ctrl+X to exit to the terminal.

## Part 4 - Configure Apache

In this part you will execute a script which will configure Apache web server so you can access Nagio's web interface.

\_\_1. View script that you will execute for configuring Apache web server.

```
cat ~/Documents/labs/nagios/configure.sh
```

\_\_2. Execute the script.

```
bash ~/Documents/labs/nagios/configure.sh
```

If it prompts you to enter password, enter "wasadmin" (without quotes).

Note: Default admin user is nagiosadmin. Since you have used a non-default admin name so you need to edit cgi.cfg file as well. You will do that in the next step.

\_\_3. Edit cgi.cfg file.

```
sudo nano /usr/local/nagios/etc/cgi.cfg
```

\_\_4. Press Ctrl+\

\_\_5. In "Search (to replace)" type "nagiosadmin" (without quotes) and press the Enter key on the keyboard.

\_\_6. In "replace with" type "wasadmin" (without quotes) and press the Enter key on the keyboard.

\_\_7. Press "A" to replace all instances of nagiosadmin with wasadmin.

\_\_8. Press Ctrl+O to save the file and hit enter.

\_\_9. Press Ctrl+X to exit to the terminal.

\_\_10. View script that you will execute for configuring Apache web server.

```
cat ~/Documents/labs/nagios/restart.sh
```

\_\_11. Execute the script.

```
bash ~/Documents/labs/nagios/restart.sh
```

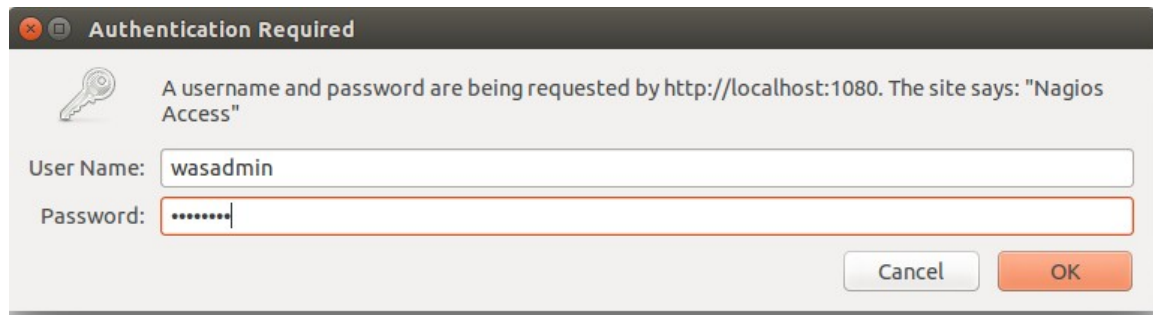
## Part 5 - Access the Nagios Web Interface

In this part you will access the Nagios web interface.

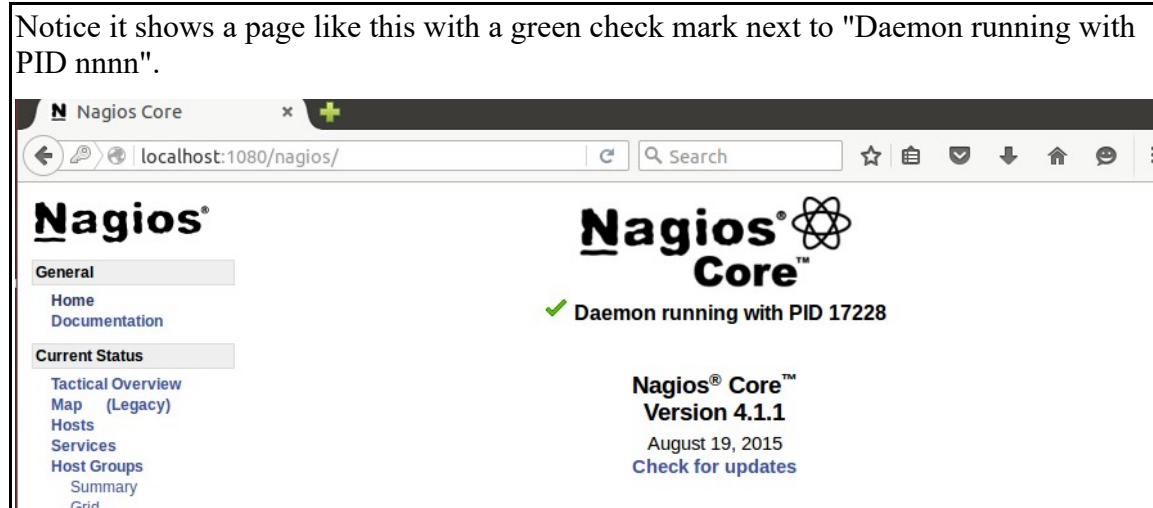
\_\_1. Launch firefox from the task bar and enter following URL.

```
http://localhost:1080/nagios
```

\_\_2. Enter "wasadmin" as user name and password.



\_\_3. Press x when it prompts you to save the credentials.



\_\_4. Under "Current Status", click "Services".

Notice it shows a page like this

General

Home

Documentation

Current Status

Tactical Overview

Map (Legacy)

Hosts

Services

Host Groups

Summary

Grid

Service Groups

Summary

Grid

Problems

Services (Unhandled)

Hosts (Unhandled)

Network Outages

Quick Search:

Reports

Availability

Trends (Legacy)

Alerts

History

Summary

Current Network Status

Last Updated: Tue Jul 19 07:23:12 MDT 2016

Updated every 90 seconds

Nagios® Core™ 4.1.1 - [www.nagios.org](http://www.nagios.org)

Logged in as wasadmin

View History For all hosts

View Notifications For All Hosts

View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems

All Types

0	1
---	---

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
7	0	0	1	0

All Problems

All Types

1	8
---	---

Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	07-19-2016 07:21:33	0d 1h 1m 39s	1/4	OK - load average: 0.44, 0.50, 0.50
	Current Users	OK	07-19-2016 07:22:11	0d 1h 1m 1s	1/4	USERS OK - 2 users currently logged in
	HTTP	OK	07-19-2016 07:22:48	0d 1h 0m 24s	1/4	HTTP OK: HTTP/1.1 301 Moved Permanently - 374 bytes in 0.000 second response time
	PING	OK	07-19-2016 07:18:26	0d 0h 59m 46s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	07-19-2016 07:19:03	0d 0h 59m 9s	1/4	DISK OK - free space: / 20854 MB (67% inode=80%):
	SSH	CRITICAL	07-19-2016 07:18:18	0d 0h 58m 31s	4/4	connect to address 127.0.0.1 and port 22: Connection refused
	Swap Usage	OK	07-19-2016 07:20:18	0d 0h 57m 54s	1/4	SWAP OK - 100% free (8676 MB out of 8676 MB)
	Total Processes	OK	07-19-2016 07:20:56	0d 0h 57m 16s	1/4	PROCS OK: 110 processes with STATE = RSZDT

The status of various services is obtained via plugins that you will explore in detail later in this lab. Refresh the page until the status is updated.

## Part 6 - Using Nagios Plugins

In this part you will utilize some of the Nagios plugins you installed in Part 6 of this lab.

\_\_ 1. Switch to the plugins directory.

```
cd /usr/local/nagios/libexec
```

\_\_ 2. Get a list of files.

```
ls
```

Notice there are various check\_\* files. These plugins can be used to check disk storage, checking whether ftp/http is up and running or not, ping a host etc.

\_\_ 3. Check Google's http status.

```
./check_http -H www.google.ca
```

\_\_ 4. Check Apache web server's http status on local machine.

```
./check_http -H localhost -p 1080
```

If you have internet access then notice it returns HTTP OK: HTTP/1.1 200 OK message.

\_\_ 5. Try http status for an invalid host.

```
./check_http -H invalidtest
```

Notice it displays "Name or service not known" message.

\_\_6. Get host up-time.

```
./check_uptime
```

Notice it displays the uptime.

\_\_7. Get disk storage.

```
df -k
```

Make a note of "Use%" for /. Subtract the value from 100%.

\_\_8. Get disk storage and set threshold using Nagios plugin.

```
./check_disk -w 10% -c 5%
```

Note: -w will display a warning if disk storage is less than 10% free. -c shows critical message if disk storage is less than 5% free.

To -w and -c pass a value greater than the number you noticed in step 10 to see if it shows warning and critical message.

## Part 7 - Adding Hosts to Monitor

In this part you will review how to create a configuration file for each of the remote hosts that you want to monitor.

\_\_1. Create a configuration file for a new host.

```
sudo nano /usr/local/nagios/etc/servers/server2.cfg
```

\_\_2. Add following text.

```
define host {
    use                linux-server
    host_name          yourhost
    alias              My Apache server
    address            10.132.234.52
    max_check_attempts 5
    check_period       24x7
    notification_interval 30
    notification_period 24x7
}
```



Notice you can specify host\_name, alias, and address. You can also specify other properties, such as, max\_check\_attempts, check\_period, notification\_interval, and notification\_period.

You have a single server setup provided as part of the course so you won't be able to add more hosts.. Here you are just seeing how a configuration file for additional hosts will look like.

\_\_\_ 3. Press Ctrl+O to save the file and hit enter.

\_\_\_ 4. Press Ctrl+X to exit to the terminal.

\_\_\_ 5. Reload the configuration.

```
sudo service nagios reload
```

\_\_\_ 6. Launch firefox from the task bar and access Nagios web user interface by entering following URL.

```
http://localhost:1080/nagios
```

\_\_\_ 7. Enter "wasadmin" both in user name and password if prompt.

\_\_\_ 8. On left side of the page, click "Hosts".

Notice it shows a page like this:

Host	Status	Last Check	Duration	Status Information
localhost	UP	07-21-2016 05:27:23	1d 3h 27m 10s	PING OK - Packet loss = 0%, RTA = 0.05 ms
yourhost	DOWN	07-21-2016 05:29:52	0d 0h 0m 4s	PING CRITICAL - Packet loss = 100%

On the page you can see "yourhost" server is down. Refresh the page until the status is updated.

## Part 8 - Monitoring Services on Host



In this part you will mimic a second host by reusing 127.0.0.1 address as a different host.

\_\_\_ 1. In the web browser type in following URL.

```
http://localhost:1080/nagios
```

\_\_\_ 2. On left side of the page, click "Services".

Notice it shows up as follows

Limit Results: 100 ▾						
Host ↕	Service ↕	Status ↕	Last Check ↕	Duration ↕	Attempt ↕	Status Information
localhost	Current Load	OK	07-21-2016 07:33:36	1d 5h 34m 34s	1/4	OK - load average: 0.27, 0.35, 0.43
	Current Users	OK	07-21-2016 07:36:42	1d 5h 33m 56s	1/4	USERS OK - 4 users currently logged in
	HTTP 	OK	07-21-2016 07:36:06	1d 5h 33m 19s	1/4	HTTP OK: HTTP/1.1 301 Moved Permanently - 374 bytes in 0.001 second response time
	PING	OK	07-21-2016 07:37:21	1d 5h 32m 41s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	07-21-2016 07:34:14	1d 5h 32m 4s	1/4	DISK OK - free space: / 19300 MB (62% inode=80%):
	SSH 	CRITICAL	07-21-2016 07:35:28	1d 5h 31m 26s	4/4	connect to address 127.0.0.1 and port 22: Connection refused
	Swap Usage	OK	07-21-2016 07:36:43	1d 5h 30m 49s	1/4	SWAP OK - 100% free (8676 MB ou of 8676 MB)
	Total Processes	OK	07-21-2016 07:34:45	1d 5h 30m 11s	1/4	PROCS OK: 117 processes with STATE = RSZDT

Services are Nagios plugins and the status is displayed in Status Information column

\_\_ 3. Review localhost configuration file.

```
sudo nano /usr/local/nagios/etc/objects/localhost.cfg
```

Notice it defines the "localhost" as host and it also contains "define service" entries where it's executing the plugins.

\_\_ 4. Press Ctrl+X to exit to the terminal.

\_\_ 5. Review commands configuration file.

```
sudo nano /usr/local/nagios/etc/objects/commands.cfg
```

Notice it defines various plugins that can be executed as commands

\_\_ 6. Press Ctrl+X to exit to the terminal.

\_\_ 7. Duplicate localhost configuration file to mimic different servers.

```
sudo cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/servers/server2.cfg
```

```
sudo cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/servers/server3.cfg
```

\_\_ 8. Edit server2.cfg.

```
sudo nano /usr/local/nagios/etc/servers/server2.cfg
```

\_\_ 9. In "define host" section, change host\_name and alias to "Server2" (without quotes).

\_\_ 10. Remove the entire "define hostgroup" section.

- \_\_ 11. Press Ctrl+\ to access search & replace option.
- \_\_ 12. In "Search (to replace)" enter "localhost" (without quotes).
- \_\_ 13. In "replace with" enter "Server2" (without quotes).
- \_\_ 14. Press "A" to replace all occurrences.
- \_\_ 15. Press Ctrl+O to save the file and hit enter.
- \_\_ 16. Press Ctrl+X to exit to the terminal.
- \_\_ 17. Edit server3.cfg.

```
sudo nano /usr/local/nagios/etc/servers/server3.cfg
```

- \_\_ 18. In "define host" section, change host\_name and alias to "Server3" (without quotes).
- \_\_ 19. Remove the entire "define hostgroup" section.
- \_\_ 20. Press Ctrl+\ to access search & replace option.
- \_\_ 21. In "Search (to replace)" enter "localhost" (without quotes).
- \_\_ 22. In "replace with" enter "Server3" (without quotes).
- \_\_ 23. Press "A" to replace all occurrences.
- \_\_ 24. Press Ctrl+O to save the file and hit enter.
- \_\_ 25. Press Ctrl+X to exit to the terminal.
- \_\_ 26. Reload Nagios configuration.

```
sudo service nagios reload
```

- \_\_ 27. In the web browser type in following URL.

```
http://localhost:1080/nagios
```

- \_\_ 28. On left side of the page, click "Services".

Notice it shows localhost, Server2, and Server3 as shown below. If the status shows "PENDING" then wait for 2-3 minutes and refresh the page.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
Server2	Current Load	OK	07-21-2016 07:55:03	0d 0h 7m 50s	1/4	OK - load average: 0.13, 0.21, 0.28
	Current Users	OK	07-21-2016 07:56:36	0d 0h 6m 17s	1/4	USERS OK - 4 users currently logged in
	HTTP	OK	07-21-2016 07:53:10	0d 0h 4m 43s	1/4	HTTP OK: HTTP/1.1 301 Moved Permanently - 374 bytes in 0.000 second response time
	PING	OK	07-21-2016 07:53:08	0d 0h 4m 45s	1/4	PING OK - Packet loss = 0%, RTA = 0.05 ms
	Root Partition	OK	07-21-2016 07:55:22	0d 0h 7m 31s	1/4	DISK OK - free space: / 19378 MB (62% inode=80%):
	SSH	CRITICAL	07-21-2016 07:54:55	0d 0h 5m 58s	4/4	connect to address 127.0.0.1 and port 22: Connection refused
	Swap Usage	OK	07-21-2016 07:53:28	0d 0h 4m 25s	1/4	SWAP OK - 100% free (8676 MB out of 8676 MB)
	Total Processes	OK	07-21-2016 07:54:13	0d 0h 8m 40s	1/4	PROCS OK: 114 processes with STATE = RSZDT
Server3	Current Load	OK	07-21-2016 07:56:13	0d 0h 1m 40s	1/4	OK - load average: 0.15, 0.22, 0.28
	Current Users	OK	07-21-2016 07:57:15	0d 0h 0m 38s	1/4	USERS OK - 4 users currently logged in
	HTTP	PENDING	N/A	0d 0h 2m 42s+	1/4	Service check scheduled for Thu Jul 21 07:58:18 MDT 2016
	PING	PENDING	N/A	0d 0h 2m 42s+	1/4	Service check scheduled for Thu Jul 21 07:59:20 MDT 2016
	Root Partition	OK	07-21-2016 07:56:37	0d 0h 1m 16s	1/4	DISK OK - free space: / 19378 MB (62% inode=80%):
	SSH	CRITICAL	07-21-2016 07:57:25	0d 0h 1m 28s	2/4	connect to address 127.0.0.1 and port 22: Connection refused
	Swap Usage	OK	07-21-2016 07:57:28	0d 0h 0m 25s	1/4	SWAP OK - 100% free (8676 MB out of 8676 MB)
	Total Processes	PENDING	N/A	0d 0h 2m 42s+	1/4	Service check scheduled for Thu Jul 21 07:58:30 MDT 2016
localhost	Current Load	OK	07-21-2016 07:53:36	1d 5h 54m 37s	1/4	OK - load average: 0.10, 0.24, 0.30
	Current Users	OK	07-21-2016 07:56:42	1d 5h 53m 59s	1/4	USERS OK - 4 users currently logged in
	HTTP	OK	07-21-2016 07:56:06	1d 5h 53m 22s	1/4	HTTP OK: HTTP/1.1 301 Moved Permanently - 374 bytes in 0.000 second response time
	PING	OK	07-21-2016 07:57:21	1d 5h 52m 44s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	07-21-2016 07:54:14	1d 5h 52m 7s	1/4	DISK OK - free space: / 19378 MB (62% inode=80%):
	SSH	CRITICAL	07-21-2016 07:55:28	1d 5h 51m 29s	4/4	connect to address 127.0.0.1 and port 22: Connection refused

\_\_29. Close the Terminal and web browser.

## Part 9 - Review

In this lab you installed, configured, and used Nagios to monitor servers and services.

## Lab 10 - Containerization – Docker

In this chapter you will install and configure Docker.

### Part 1 - Launch terminal

In this part you will launch the Linux terminal.

- \_\_1. In the task bar, click **Search** button.
- \_\_2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_3. Run following command to switch to **Documents** directory.

```
cd ~/Documents
```

### Part 2 - Install Docker

In this part you will create a Chef cookbook that installs Docker.

- \_\_1. View key.sh file contents.

```
cat ~/Documents/labs/docker/key.sh
```

Notice it installs the GPG key required for installing Docker. Alternatively, you can run "sudo apt-get -allow-unauthenticated update", but, it's not recommended.

- \_\_2. Run the script to install GPG key.

```
bash ~/Documents/labs/docker/key.sh
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

- \_\_3. Switch to the cookbooks folder.

```
cd ~/Documents/cookbooks
```

- \_\_4. Create the new docker cookbook.

```
sudo chef generate cookbook my_docker
```

- \_\_5. Copy a recipe that installs docker prerequisites.

```
sudo cp ~/Documents/labs/docker/dockerPrereqs.rb
```

```
~/Documents/cookbooks/my_docker/recipes
```

\_\_6. View the recipe's contents.

```
cat ~/Documents/cookbooks/my_docker/recipes/dockerPrereqs.rb
```

\_\_7. Update the package manager to ensure we install the correct version of docker.

```
sudo apt-get update
```

\_\_8. Execute the Chef recipe for installing the prerequisites and ensure there are no errors.

```
sudo chef-client -z -r "recipe[my_docker::dockerPrereqs]"
```

Note: It might take a while. Wait until it returns to the prompt.

To purge as a safety precaution in the event that someone has installed a deprecated docker package.

```
sudo apt-get purge lxc-docker
```

To update the package manager cache policy for the docker engine software.

```
apt-cache policy docker-engine
```

\_\_9. Copy recipe for installing docker.

```
sudo cp ~/Documents/labs/docker/dockerInstall.rb  
~/Documents/cookbooks/my_docker/recipes
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

\_\_10. View dockerInstall.rb recipe.

```
cat ~/Documents/cookbooks/my_docker/recipes/dockerInstall.rb
```

Notice it installs docker and starts the service.

\_\_11. Add docker dependency to the metadata for your my\_docker cookbook usage.

```
sudo nano ~/Documents/cookbooks/my_docker/metadata.rb
```

\_\_12. Append this line to the end.

```
depends 'docker', '~> 2.0'
```

\_\_13. Press Ctrl+O to save the file and hit enter.

\_\_14. Press Ctrl+X to exit to the terminal.

\_\_15. Set **cookbooks** directory as the "vendor" directory.

```
cd ~/Documents/cookbooks/my_docker
```

```
sudo berks vendor ~/Documents/cookbooks
```

Alternatively, you can download all the dependencies manually from <https://supermarket.chef.io> and unpack them in the cookbooks directory.

\_\_16. Verify dependencies are downloaded.

```
ls ~/Documents/cookbooks
```

Notice there are 2 extra folders: compat\_resource, docker

\_\_17. Update APT repository.

```
sudo apt-get update
```

\_\_18. Execute the chef recipe for the installation.

```
sudo chef-client -z -r "recipe[my_docker::dockerInstall]"
```

\_\_19. Configure user security.

```
sudo usermod -aG docker wasadmin
```

\_\_20. Verify Docker is installed.

```
docker --version  
sudo status docker
```

\_\_21. Download and run hello-world container.

```
sudo docker run hello-world
```

You will see something like this:

```
wasadmin@ubuntu:~/Documents/cookbooks/my_docker$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c04b14da8d14: Pull complete
Digest: sha256:0256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdc9f9431a1feb60fd9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

\_\_22. Get a list of containers.

```
sudo docker ps -a
```

Notice it shows hello-world container.

\_\_23. Remove hello-world container.

```
sudo docker rm <CONTAINER ID>
```

Note: Replace CONTAINER ID by the id displayed on the console.

## Part 3 - Install nginx web server and add it to docker container

\_\_1. Copy recipe for installing and configuring nginx web server.

```
cd ~/Documents/cookbooks

sudo cp ~/Documents/labs/docker/nginxSample.rb
~/Documents/cookbooks/my_docker/recipes
```

\_\_2. View the recipe's contents.

```
cat ~/Documents/cookbooks/my_docker/recipes/nginxSample.rb
```

Notice the recipe installs nginx and listens on port 80

\_\_3. Invoke the chef recipe for the nginx installation and configuration.

```
sudo chef-client -z -r "recipe[my_docker::nginxSample]"
```

\_\_4. Run hello-world container.

```
sudo docker run hello-world
```

\_\_5. Get a list of containers.



```
sudo docker ps -a
```

Notice it shows hello-world container.

\_\_ 6. Remove nginx container.

```
sudo docker rm <CONTAINER ID>
```

Note: Replace CONTAINER ID by the id displayed on the console.

\_\_ 7. Close the Terminal.

## Part 4 - Review

In this lab you installed and configured Docker.

# Lab 11 - Scripting 101 – Python (OPTIONAL)

In this chapter you will explore basics of Python scripting.

## Part 1 - Launch the shell terminal and create a folder for saving scripts

In this part you will launch the Linux terminal.

- \_\_ 1. In the task bar, click **Search** button.
- \_\_ 2. In search text box, type in **terminal** and hit enter key on the keyboard.

Alternatively, you can press Ctrl+Alt+T to access the terminal.

- \_\_ 3. Run following command to switch to **python** directory

```
cd ~/Documents/labs/python
```

## Part 2 - Basic python programming using interactive mode

In this part you will run python commands without creating a script file.

- \_\_ 1. Run following command to launch python.

```
python
```

- \_\_ 2. Run following command to display "Hello World" message.

```
print ("Hello World! ")
```

- \_\_ 3. Run following to create a variable, display it's value, and unset the variable.

```
a=1
print (a)
del a
print (a)
```

- \_\_ 4. Run following to create variables, add them up, and display result.

```
a=5
b=6
print (a + b)
```

- \_\_ 5. Exit to the terminal.

```
quit()
```

### Part 3 - Execute a basic script that prints "Hello world!"

In this part you will create your first python script that prints "Hello world!" message.

\_\_1. View hello.py contents.

```
cat hello.py
```

Notice it defines a variable and prints its value on the screen.

\_\_2. Run following command to execute the script and verify it displays "Hello World!".

```
python hello.py
```

### Part 4 - Alternative way of executing python scripts

In this part you will execute python script without utilizing "python" as part of the command line.

\_\_1. View hello2.py contents.

```
cat hello2.py
```

Notice `#!/usr/bin/python` in the first line. It defines the interpreter that will be used to run the script.

\_\_2. Run following command to execute the script.

```
./hello2.py
```

Notice it displays Permission denied message. You have to grant appropriate permissions to the script before it can be executed.

\_\_3. Run following command to obtain security permissions for your script.

```
ls -l
```

Note: Permissions are listed in the first column.

The first character states what the item is: a 'd' means directory and a '-' means file.

The next 3 characters tell whether the owner has given himself permission to read, write, or execute the file. First character 'r' means read permission is granted, '-' means read permission is denied. Second character 'w' means write permission is granted, '-' means write permission is denied. Third character 'x' means execute permission is granted, '-' means execute permission is denied. In case of a directory, 'x' means directory is searchable.

The next 3 characters tell you what permission the user has granted to other members of his group.

The last 3 characters give permissions that rest of the people has.

The first 3 4 characters for hello2.py should be -rw- which means it's a file with read and write permission granted to the wasadmin user.

\_\_4. Run following command to grant your script read, write, and execute permission.

```
sudo chmod +x hello2.py
```

Enter wasadmin or the password provided by your organization. Contact your instructor to get this information.

Note: If you want your script to be private (i.e. only you can read and execute), use u+=x instead.

\_\_5. Run following command to obtain security permissions for your script.

```
ls -l
```

Note: After executing the command the first 4 characters should be -rwx which means it's a file with read, write, and execute permissions.

\_\_6. Run following command to execute the script and verify it prints "Hello World!".

```
./hello2.py
```

\_\_7. Run following command to grant all python scripts the execute permission.

```
sudo chmod +x *.py
```

\_\_8. Run following command to try executing the script without specifying path.

```
hello2.py
```

Notice it prints 'command not found' message. If you want to execute a script without specifying path then it should be added to the path.

\_\_9. Run following command to obtain directories in path.

```
echo $PATH
```

\_\_10. Run following command to add /home/wasadmin/Documents/labs/python directory to the path.

```
export PATH=$PATH:/home/wasadmin/Documents/labs/python
```

\_\_11. Run following command to obtain directories in path and verify your custom shell directory is in the path.

```
echo $PATH
```

\_\_12. Run following command to execute your script.

```
hello2.py
```

## Part 5 - Get input from user

In this part you will execute a shell script that utilizes variables, gets input from user, creates a directory, lists directory contents, and delete the directory.

\_\_1. View my\_dir.py contents.

```
cat my_dir.py | more
```

Notice it's using **os** module to manipulate directories. It is also using `raw_input` function to get user input.

\_\_2. Run the script.

```
my_dir.py
```

\_\_3. When prompted to enter directory name, type "test" (without quotes) and press enter key on the keyboard.

Notice it creates the test directory, list the contents of current directory, deletes the directory and relists the contents of the current directory.

## Part 6 - Execute a script that utilizes functions and return keyword

In this part you will execute a script that utilizes functions. You will see how to return numeric and string output.

\_\_1. View functions.py contents.

```
cat functions.py | more
```

Notice it's using various functions for displaying date, calendar, and other information. **def** keyword is used to define a custom function. **lambda** allows creation of anonymous functions.

\_\_2. Run the script.

```
functions.py
```

Notice it prints date time, calendar, and sum of 5 and 6.

## Part 7 - Conditional Statements

In this part you will execute a script that utilizes if..else if..else.

\_\_1. View if.py contents.

```
cat if.py | more
```

Notice it's using if-else code to display directory contents, date, or calendar depending on command-line arguments passed to the script.

\_\_2. Run following command and notice it displays the usage for the script.

```
if.py -invalid invalid
```

\_\_3. Run following command notice it displays directory contents.

```
if.py -p l
```

\_\_4. Run following command notice it displays the date.

```
if.py -p d
```

\_\_5. Run following command notice it displays the calendar.

```
if.py -p c
```

## Part 8 - Lists and Loops

In this part you will execute a script that utilizes lists and loops.

\_\_1. View list\_loop.py contents.

```
cat list_loop.py | more
```

Notice it's using for and while loop to add items to an array and display its contents.

\_\_2. Run following command to execute the script.

```
list_loop.py
```

Notice it displays the array contents.

## Part 9 - Tuple, filing, and exception handling

In this part you will execute a script that utilizes tuple, filing, and exception handling to write file list to a .html file.

\_\_1. View `html.py` contents.

```
cat html.py | more
```

Notice it's using `os` module to to create an HTML file which displays directory contents.

\_\_2. Run following command to execute the script.

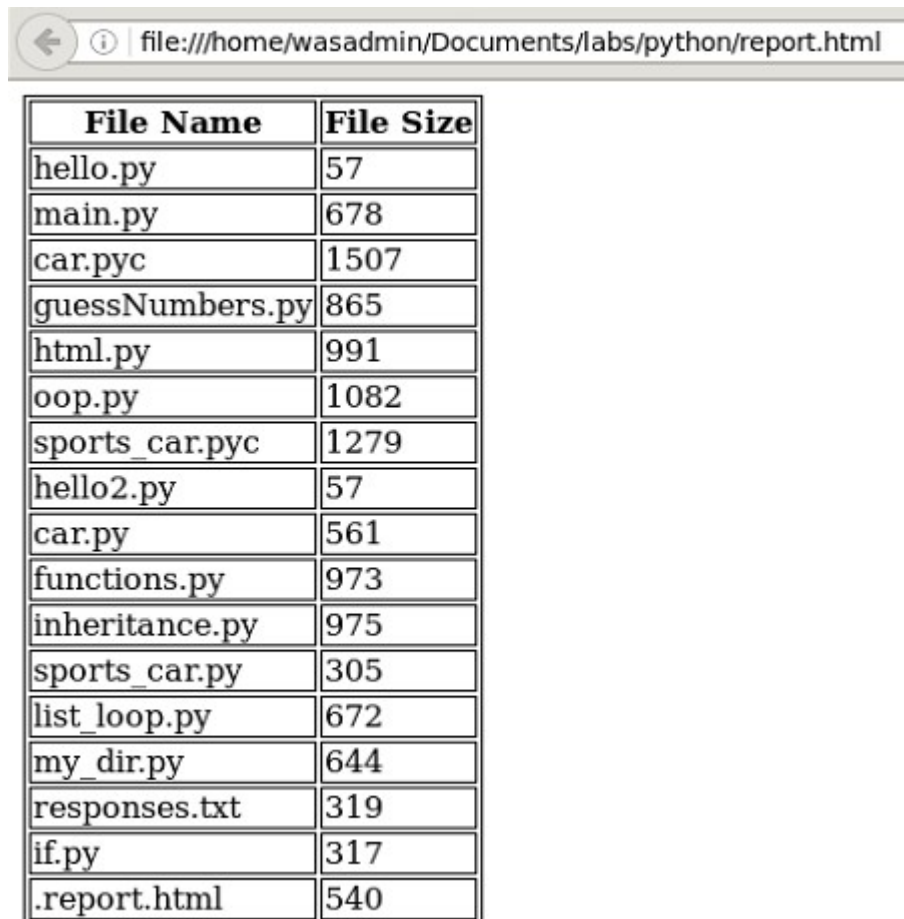
```
html.py
```

\_\_3. In the Linux task bar, click **Files** to launch file system explorer.

\_\_4. Browse to `~/Documents/labs/python`

\_\_5. Double click **report.html** (Note: ensure that you have Firefox or some other web browser installed and set as default web browser).

\_\_6. Verify the output looks as show below (Note: values will be different).



The screenshot shows a web browser window with the address bar displaying `file:///home/wasadmin/Documents/labs/python/report.html`. Below the address bar is a table with two columns: **File Name** and **File Size**. The table lists 18 files and their sizes in bytes.

File Name	File Size
hello.py	57
main.py	678
car.pyc	1507
guessNumbers.py	865
html.py	991
oop.py	1082
sports_car.pyc	1279
hello2.py	57
car.py	561
functions.py	973
inheritance.py	975
sports_car.py	305
list_loop.py	672
my_dir.py	644
responses.txt	319
if.py	317
.report.html	540

## Part 10 - Filing basics

In this part you will execute a script that demonstrates the basics of filing and user input in python.

\_\_1. View guessNumbers.py contents.

```
cat guessNumbers.py | more
```

Notice it's using os module to read a text file. It's using **random.randrange** to generate a random number then using a while loop to let the user guess the number.

\_\_2. Run following command to execute the script.

```
guessNumbers.py
```

\_\_3. Enter any number until to get the Congratulations message.

Notice it displays a random message each time you make a wrong guess.

## Part 11 - Basics of object oriented programming in python

In this part you will execute a script that utilizes object oriented programming concepts, such as, class, property, constructor, and inheritance.

\_\_1. View oop.py contents.

```
cat oop.py | more
```

Notice it's creating a class named **car**. The class contains a class-level variable, a few instance-level properties, a constructor, a destruction, and few custom methods. It also creates instances of the class.

\_\_2. Run following command to execute the script.

```
oop.py
```

\_\_3. Verify it shows result as shown below.

```
Car constructed
Car constructed
Starting Ford...
Starting Toyota...
110
100
Ford
Toyota
Car destroyed
Car destroyed
```

\_\_4. View inheritance.py contents.

```
cat inheritance.py | more
```

Notice there's SportsCar class which inherits Car class.

\_\_5. Run following command to execute the script.



inheritance.py

```
=====testing inheritance=====
SportsCar constructed
SportsCar.start
SportsCar destroyed
```

## Part 12 - Modules in python

In this part you will execute a script which utilizes modules. Rather than keeping code in a single script, you will notice the code has been split into 3 files: car.py, sports\_car.py, and main.py.

\_\_1. View car.py contents.

```
cat car.py | more
```

Notice it contains car class definition.

\_\_2. View sports\_car.py contents.

```
cat sports_car.py | more
```

Notice it's importing car module and contains SportsCar class definition.

\_\_3. View main.py contents

```
cat main.py | more
```

Notice it imports car and sports\_car modules. It's also creating instances of classes.

\_\_4. Run following command to execute the script.

```
main.py
```

Notice it's result is same as previous part.

\_\_5. Close the Web Browser and Terminal.

## Part 13 - Review

In this lab you learned the basics of Python scripting.

