**DevOps lab 4.3 Test Coverage Tools**

Setup a Mocking framework environment.

**Step-1:**

Open the Cloud Platform Console at [https://console.cloud.google.com](https://console.cloud.google.com/).

Click on the three horizontal bars at the left most side of the blue bar near the top of the browser window. *Select Compute Engine*.

Select *VM Instances*. You should see the virtual machine you created earlier.

Click on the checkbox to the left of the VM name and then select *START*. It will take a few moments to start.

Click on *SSH* to start a terminal window.

**Change the host name to student:** Find the icon that looks like a gear in the upper right-hand corner of this terminal browser window and select *Change Linux User Name*. Enter *student* and *click Change*. Now, notice the prompt that says "student@lab:~$"



**Step 2**

We will use the same code as in the FizzBuzz project.

*cd  
cd devops-lesson-4/lab-4.1*

**Step 3**

Start the vncserver.

*vncserver*

**Step 4**

Start the VNC client on your local machine. You will need the external IP address of your virtual machine. Enter the following into the VNC client, replacing x.x.x.x with your IP address. Note that the external IP address can change every time you start the virtual machine.

x.x.x.x:5901

If you are using a Mac, enter the following into Safari, replacing x.x.x.x with your IP address.

vnc://x.x.x.x:5901

Dismiss any warnings about insecure connections.

The password is simplilearn.

You will be connected to the graphical desktop of the virtual machine.

**Step 5**

Start and use Eclipse.

In your VNC window, start the *File System* from the desktop icon.  
Select *Student* in the left column.  
Open the *eclipse* folder.  
Open the java-neon\_ folder.  
Open the *eclipse* folder.  
Start eclipse from the diamond shaped icon.  
Accept the default workspace location.

Select File->New->Java Project.  
Uncheck *Use default location*.  
Click on the *Browse* button. Select *Home*.  
Open the devops-lesson-4 folder.  
Open the lab-4.1 folder.  
Select the FizzBuzz folder and hit *OK* at the bottom of the dialog.  
Click on the *Finish* button at the bottom of the dialog.

You should see the FizzBuzz project on the left. The test folder is marked in red as there is a missing dependency.

Right click on the FizzBuzz project. A menu will appear. Select *Properties* at the bottom.

Select *Java Build Path*.  
Select the *Libraries* tab.  
Click on the *Add Library* button.  
Select *JUnit* and then hit *Next*.  
Make sure that *JUnit 4* is selected and then hit *Finish*. Hit *OK.* The project errors should disappear.

Open up the source and test folders and the DevOps packages. Double click on FizzBuzz.java and FizzBuzzTest.java. Editor windows will open.

**Step 6**

Right click on FizzBuzzTests.java.

Select *Run as* and then then select *JUnit Test*.

The unit tests should run successfully.

Implement the numberThreeReturnsFizz() test and run it. It should fail.

Implement the code and run it. Make changes until the tests pass.

If you have time, implement more tests.

**Step 7**

You will see code coverage now. This means you have code but it does not execute during run time. Perhaps it is dead code or is not being called properly.

We will use ECLEMMA for this. It’s a free java code coverage tool.

Go to help, Eclipse marketplace, and search for ECLEMMA. Install it.

Accept terms and proceed to install. It will ask you to restart Eclipse. Go ahead and restart it.

Right click on FizzBuzz project and Select *Run Under Coverage as* and then select *JUnit Test*.

Now, look at the code coverage results. You can now click on the percentage coverage and the code file under src folder to see where there is less code coverage.

Aim for 100% code coverage in your code. Anything above 90% is good.

Any extra code not being executed will be highlighted in red.

If you have time, implement more code and re-run.

**Step 8**

You will see coding mistakes/coding standards you may want to use in your project.

Use PMD for this. It’s an open-source tool for coding standards. It analyzes idiom mistakes, conversion mistakes, etc.

It’s a rule-based system, but you can turn off the rules. It’s hard for your code to pass all rules, but you can turn rules off and then turn on rules when you and your team make more mistakes. Finally, your team can decide which rules you always want to turn on.

Go to help in Eclipse marketplace. Search for PMD and install it.

Accept terms and proceed to install. You will be asked to restart Eclipse; go ahead restart it.

Set up PMD with rules, enable them, disable them, try a few rules, and scan your code.

If you have time, implement more code and re-run.

**Step 9**

Exit eclipse by selecting *File*->*Exit*.

Close the VNC viewer.

You will need to stop the lab computer at the end of each day to prevent it from accumulating costs during the evening and night.

From the Web UI, you can navigate to the Compute Engine section and select your lab computer. When it is selected, click on the icon representing the "Stop" operation as shown below:

