**Capstone - Exercise 3 Camel Integration**

Use Apache Camel to get Docker events.

**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:~$"



We need to change the Docker startup script so that it listens on TCP port 2375.

*sudo vi /etc/systemd/system/multi\*/docker.service*

Add a second -H switch to the ExecStart line so that it reads:

*ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375*

Reload the daemon processes.

*sudo systemctl daemon-reload*

Restart Docker.

*sudo systemctl restart docker*

Check that there is a listener on port 2375:

*netstat -an | grep 2375*

Install telnet.

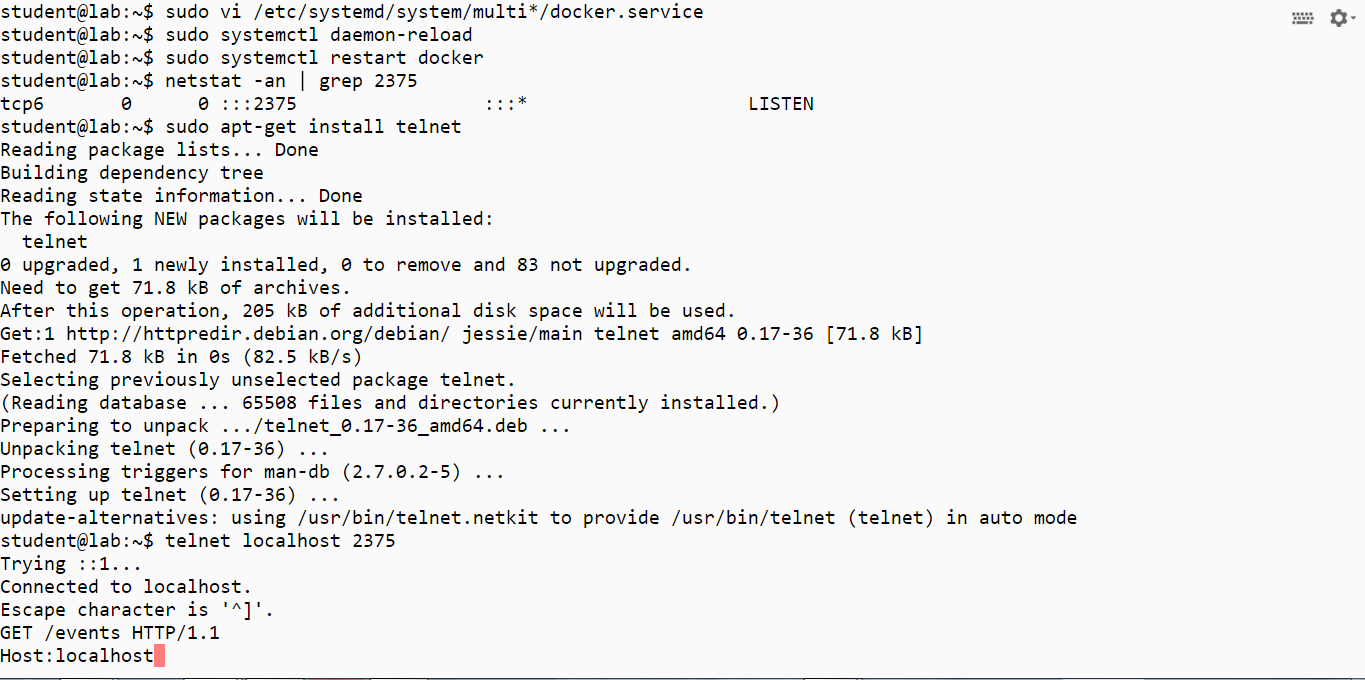
*sudo apt-get install telnet*

Connect to Docker using telnet:

*telnet localhost 2375*

Now type the following two HTTP headers.

*GET /events HTTP/1.1  
Host: localhost*



Hit ENTER twice.

You should see a response from Docker similar like below:   


Exit telnet by typing control-] followed by quit.

**Step 2**

Start the VNC server.

*vncserver*

Start the VNC client on your local machine and enter x.x.x.x.:5901, replacing x.x.x.x with the external IP address of your virtual machine.

Start Eclipse.

Open the DevOps project and the src and test folders. Open the files DockerRouting.java and DockerRoutingTests.java. The first file uses Apache Camel to read event data from Docker. The second file uses the Apache Camel test framework to run routes.

Run DockerRoutingTests.java as a JUnit test.

Go to your SSH window and generate events.

*docker run -it --rm centos /bin/bash*

Exit with control-D.

Go back to Eclipse. The integration test should have passed.  
Look through the console output and see what events were reported. The actual Docker message is the text starting *Event*.

**Step 3**

Open the files DockerData.java and DockerDataTests.java in Eclipse. We are populating a Docker object from Event data. The unit test artificially generated the event.  
Run the unit tests in DockerDataTests.java. They should pass.

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:

