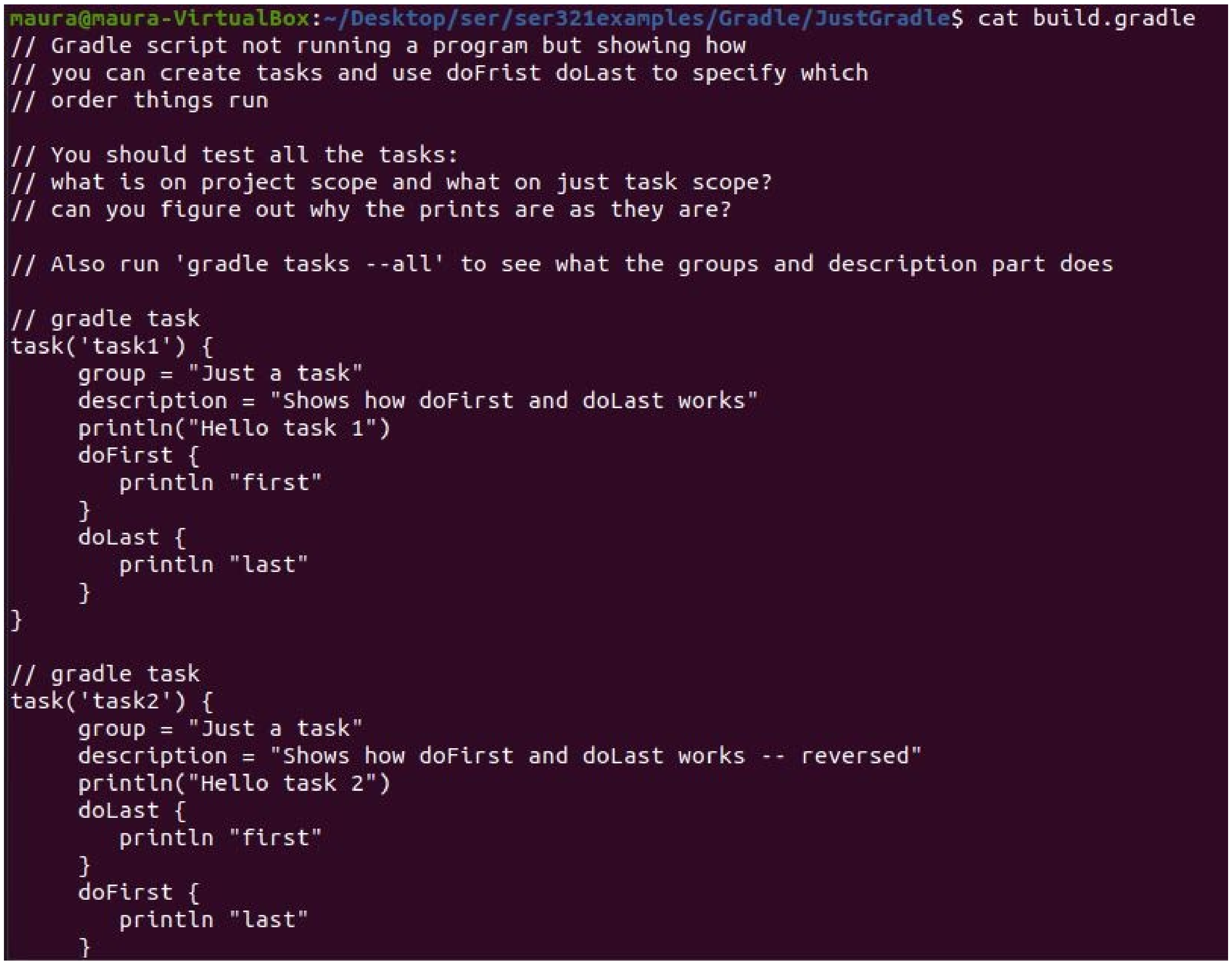
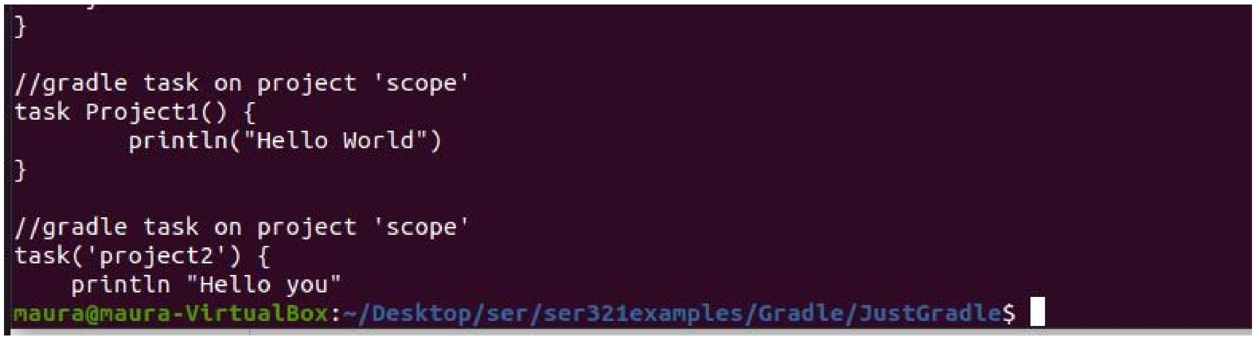
Part 1
https://github.com/mpeter56/ser321-spring2022-A-mpeter56.git
2.1
Linux:

1. mkdir cli_assignment
2. cd cli_assignment
3. touch stuff.txt
4. cat >stuff.txt
   Hello
   World
   What a beautiful day
   Ctrl d
5. wc -w -l stuff.txt
6. cat >>stuff.txt
   It is so bright and sunny
   Ctrl d
7. mkdir draft
8. mv stuff.txt draft
9. touch .secret.txt
10. cd ..
    cp -r draft final
11. mv draft draft.remove
12. mv draft.remove final
13. ls -R -a -l
14. zcat NASA_access_log_Aug95.gz
15. gunzip NASA_access_log_Aug95.gz
16. mv NASA_access_log_Aug95 logs.txt
17. mv logs.txt /home/maura/Desktop/ser/week1/cli_assignment
18. head -n 100 logs.txt
19. head -n 100 logs.txt | tee logs_top_100.txt
20. tail -n 100 logs.txt
21. tail -n 100 logs.txt | tee logs_bottom_100.txt
22. cat logs_top100.txt logs_bottom_100.txt > ./logs_snapshot.txt
23. cat >> logs_snapshot.txt
    mpeter56: This is a great assignment 1/13/2022
    Ctrl d
24. less logs.txt
    Use down key till you are done or press Q to exit
25. tail -n +2 marks.csv | cut -d “%” -f 1
26. tail -n +2 marks.csv | cut -d “%” -f 4 | sort
27. tail -n +2 marks.csv | awk -F’%’ ‘{sum+=$2} END {print(sum/NR)}’
28. tail -n +2 marks.csv | awk -F’%’ ‘{sum+=$2} END {print(sum/NR)}’ | tee done.txt
29. mv done.txt final
30. cd final

mv done.txt average.txt
2.2
Example 1 JustGradle
This example shows how to use the doFirst and doLast and also shows project scope vs task scope.
This is the code in build.gradle

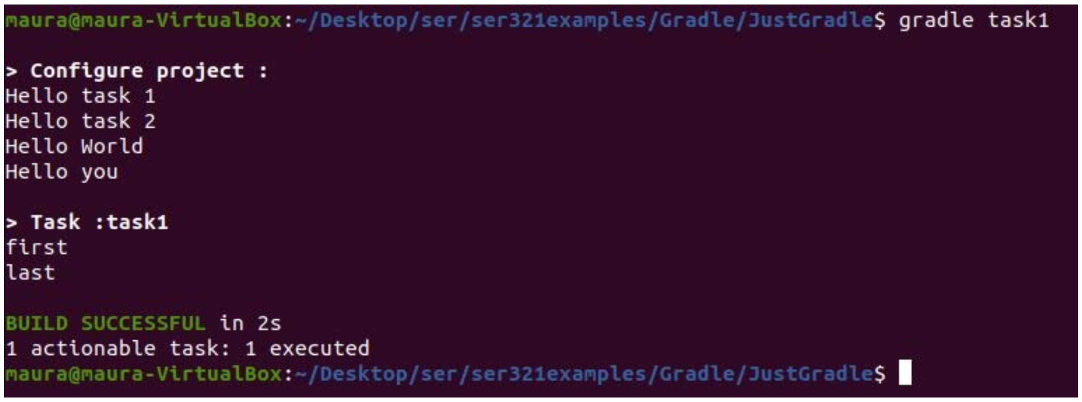


```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ cat build.gradle
// Gradle script not running a program but showing how
// you can create tasks and use doFrist doLast to specify which
// order things run

// You should test all the tasks:
// what is on project scope and what on just task scope?
// can you figure out why the prints are as they are?

// Also run 'gradle tasks --all' to see what the groups and description part does

// gradle task
task('task1') {
     group = "Just a task"
     description = "Shows how doFirst and doLast works"
     println("Hello task 1")
     doFirst {
        println "first"
     }
     doLast {
        println "last"
     }
}

// gradle task
task('task2') {
     group = "Just a task"
     description = "Shows how doFirst and doLast works -- reversed"
     println("Hello task 2")
     doLast {
        println "first"
     }
     doFirst {
        println "last"
     }
```



```
}

//gradle task on project 'scope'
task Project1() {
        println("Hello World")
}

//gradle task on project 'scope'
task('project2') {
    println "Hello you"
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$
```

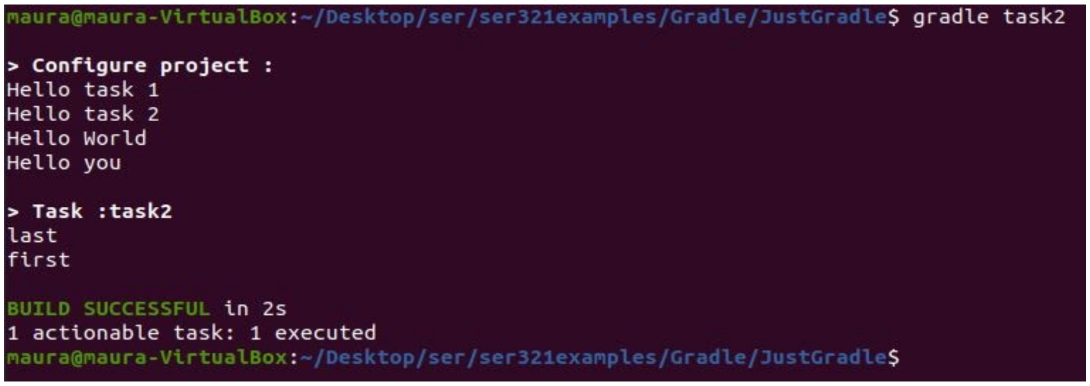
This is the result of running task 1



```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ gradle task1

> Configure project :
Hello task 1
Hello task 2
Hello World
Hello you

> Task :task1
first
last

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$
```

The print “first” was in the doFirst so prints first and the print “last” was in doLast so prints last

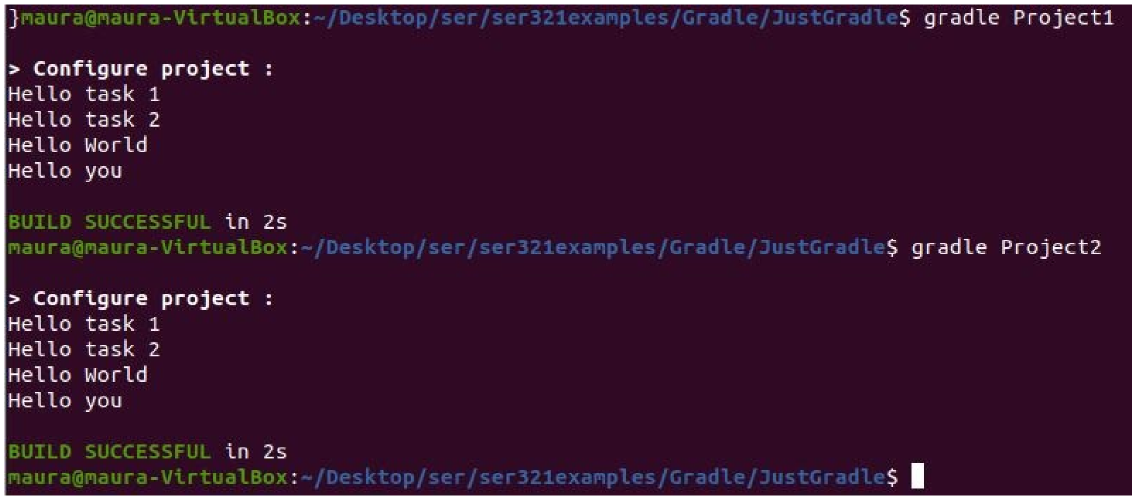
This is the result of running task 2



```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ gradle task2

> Configure project :
Hello task 1
Hello task 2
Hello World
Hello you

> Task :task2
last
first

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$
```

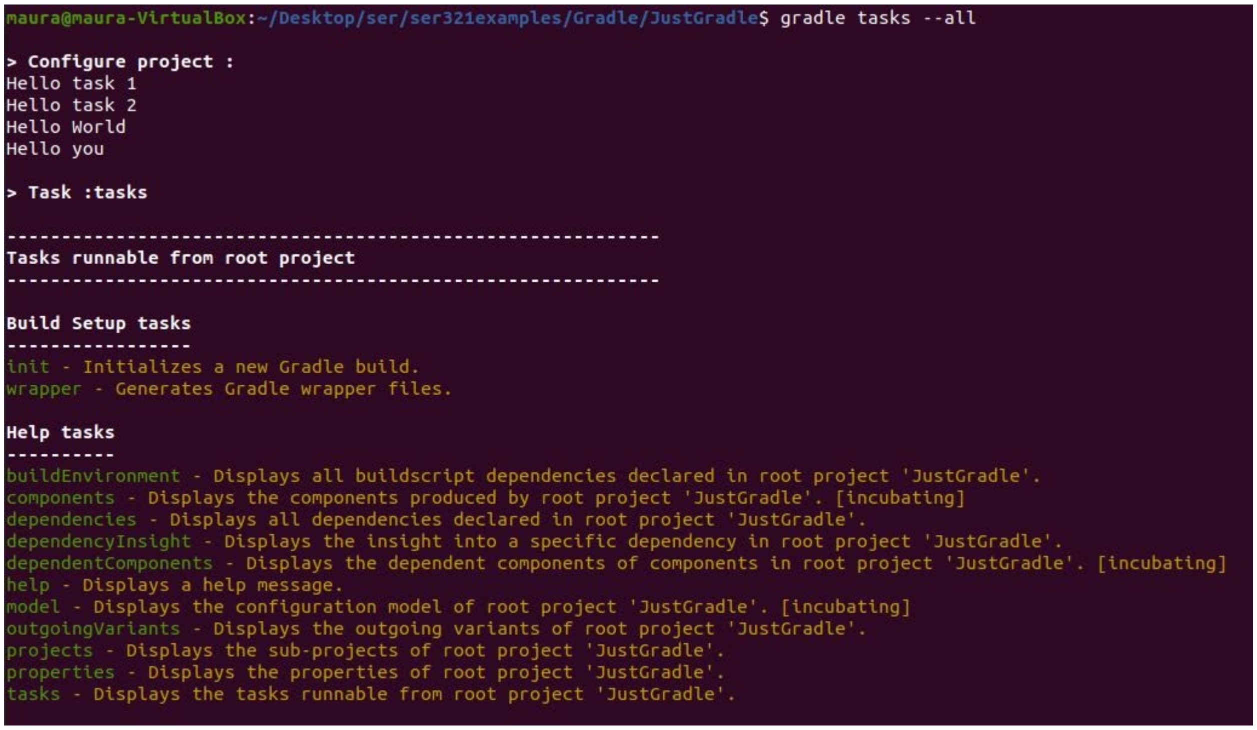
The print “last” was in the doFirst so prints first and the print “first was in the doLast so prints last
The print statements in project scope appear in both because project scope always prints, but the tasks only print for the current task

Running both Project1 and Project2 print the same because project scope always prints

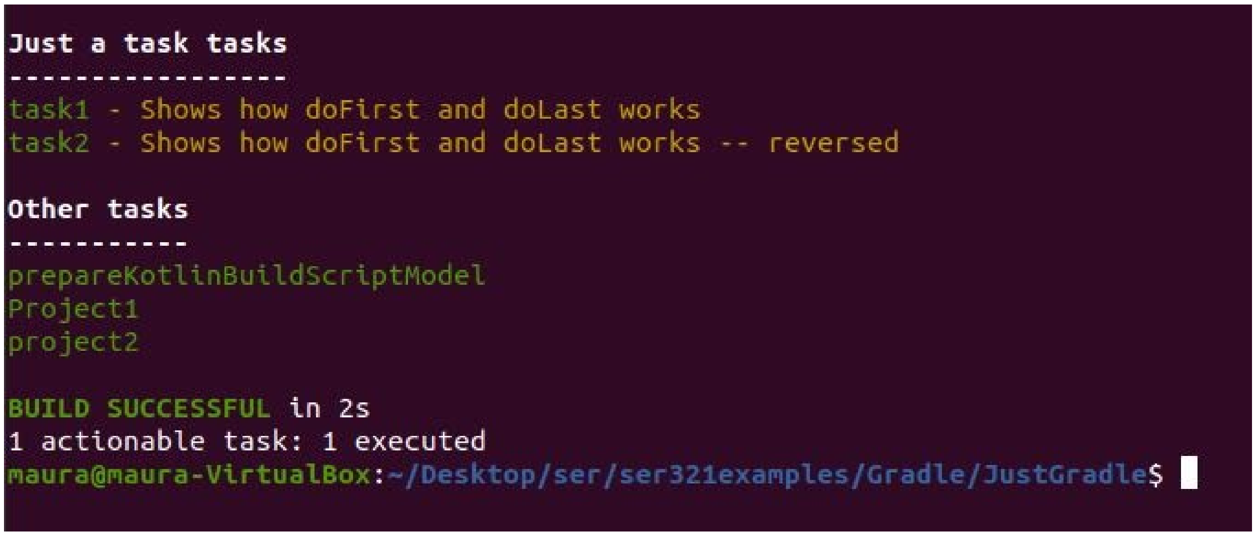


```
}maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ gradle Project1

> Configure project :
Hello task 1
Hello task 2
Hello World
Hello you

BUILD SUCCESSFUL in 2s
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ gradle Project2

> Configure project :
Hello task 1
Hello task 2
Hello World
Hello you

BUILD SUCCESSFUL in 2s
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$
```

This shows running gradle tasks - -all



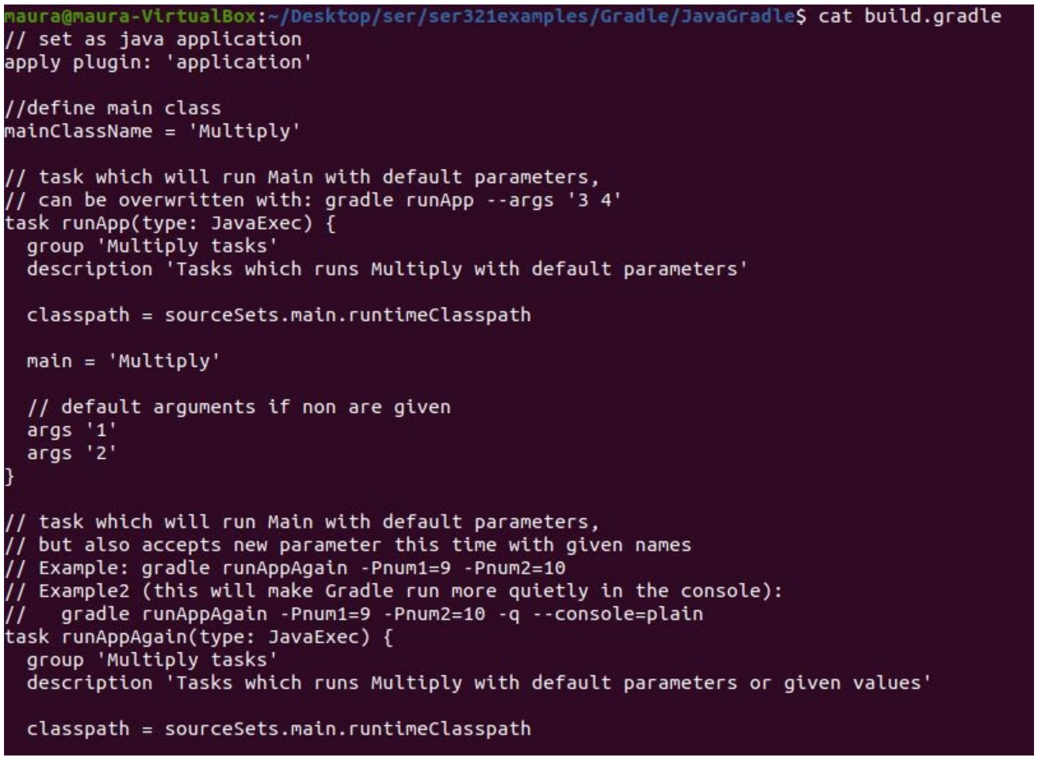
```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ gradle tasks --all

> Configure project :
Hello task 1
Hello task 2
Hello World
Hello you

> Task :tasks

------------------------------------------------------------
Tasks runnable from root project
------------------------------------------------------------

Build Setup tasks
-----------------
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'JustGradle'.
components - Displays the components produced by root project 'JustGradle'. [incubating]
dependencies - Displays all dependencies declared in root project 'JustGradle'.
dependencyInsight - Displays the insight into a specific dependency in root project 'JustGradle'.
dependentComponents - Displays the dependent components of components in root project 'JustGradle'. [incubating]
help - Displays a help message.
model - Displays the configuration model of root project 'JustGradle'. [incubating]
outgoingVariants - Displays the outgoing variants of root project 'JustGradle'.
projects - Displays the sub-projects of root project 'JustGradle'.
properties - Displays the properties of root project 'JustGradle'.
tasks - Displays the tasks runnable from root project 'JustGradle'.
```



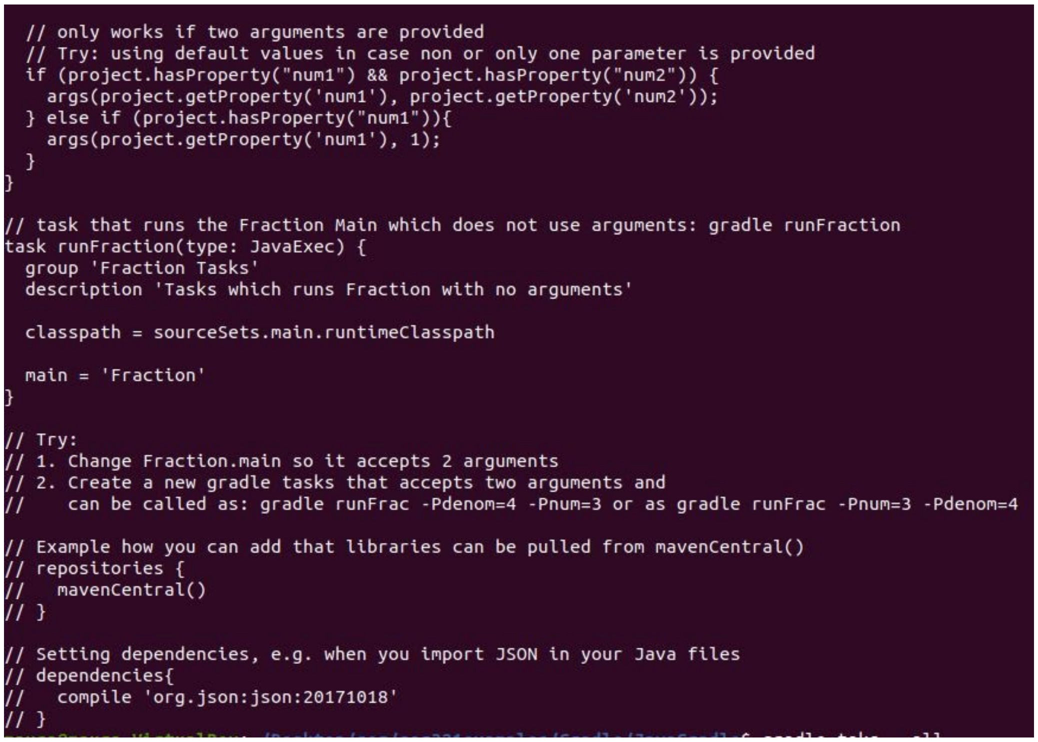
```
Just a task tasks
-----------------
task1 - Shows how doFirst and doLast works
task2 - Shows how doFirst and doLast works -- reversed

Other tasks
-----------
prepareKotlinBuildScriptModel
Project1
project2

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JustGradle$ 
```

This shows all available tasks in the build and the descriptions of what they do, also shows all generic tasks. It also prints the print statements in project scope.

Example 2:
JavaGradle
This program has two groups of tasks Multiply tasks and Fraction tasks. Tasks in these groups will run the Main for their group.

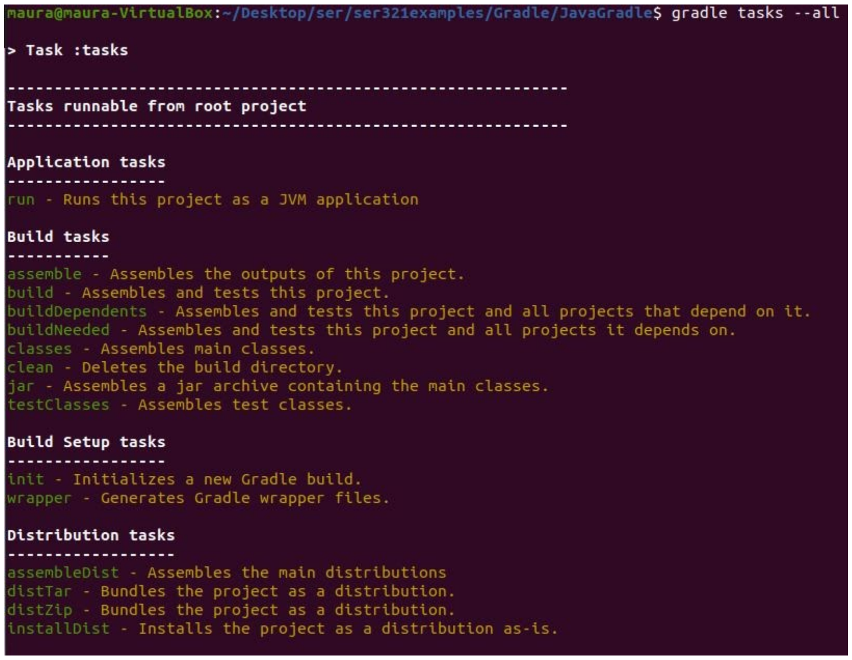


```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ cat build.gradle
// set as java application
apply plugin: 'application'

//define main class
mainClassName = 'Multiply'

// task which will run Main with default parameters,
// can be overwritten with: gradle runApp --args '3 4'
task runApp(type: JavaExec) {
  group 'Multiply tasks'
  description 'Tasks which runs Multiply with default parameters'

  classpath = sourceSets.main.runtimeClasspath

  main = 'Multiply'

  // default arguments if non are given
  args '1'
  args '2'
}

// task which will run Main with default parameters,
// but also accepts new parameter this time with given names
// Example: gradle runAppAgain -Pnum1=9 -Pnum2=10
// Example2 (this will make Gradle run more quietly in the console):
//   gradle runAppAgain -Pnum1=9 -Pnum2=10 -q --console=plain
task runAppAgain(type: JavaExec) {
  group 'Multiply tasks'
  description 'Tasks which runs Multiply with default parameters or given values'

  classpath = sourceSets.main.runtimeClasspath
```

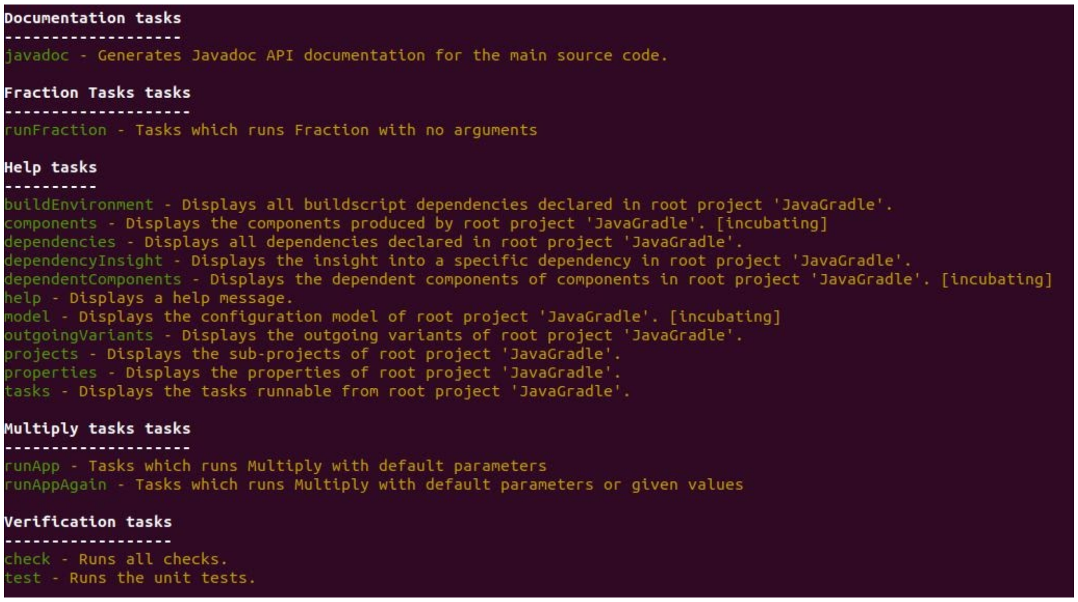


```
  // only works if two arguments are provided
  // Try: using default values in case non or only one parameter is provided
  if (project.hasProperty("num1") && project.hasProperty("num2")) {
    args(project.getProperty('num1'), project.getProperty('num2'));
  } else if (project.hasProperty("num1")){
    args(project.getProperty('num1'), 1);
  }
}

// task that runs the Fraction Main which does not use arguments: gradle runFraction
task runFraction(type: JavaExec) {
  group 'Fraction Tasks'
  description 'Tasks which runs Fraction with no arguments'

  classpath = sourceSets.main.runtimeClasspath

  main = 'Fraction'
}

// Try:
// 1. Change Fraction.main so it accepts 2 arguments
// 2. Create a new gradle tasks that accepts two arguments and
//    can be called as: gradle runFrac -Pdenom=4 -Pnum=3 or as gradle runFrac -Pnum=3 -Pdenom=4

// Example how you can add that libraries can be pulled from mavenCentral()
// repositories {
//   mavenCentral()
// }

// Setting dependencies, e.g. when you import JSON in your Java files
// dependencies{
//   compile 'org.json:json:20171018'
// }
```

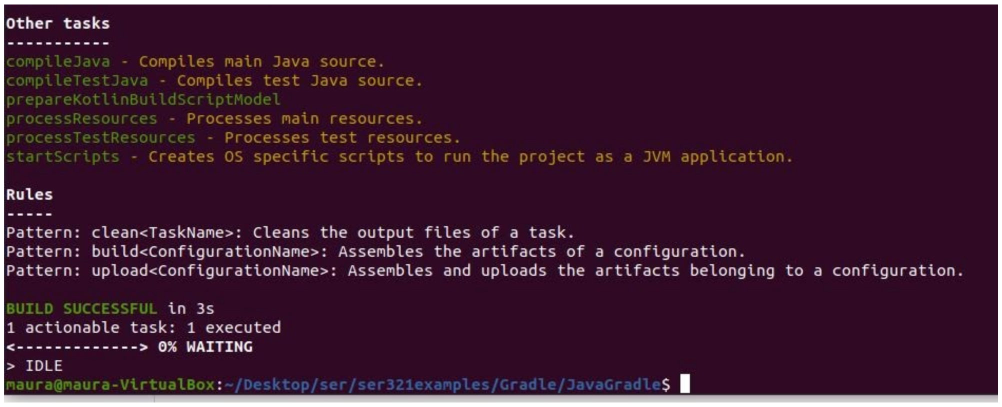
This shows all available tasks for this application



```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle tasks --all

> Task :tasks

------------------------------------------------------------
Tasks runnable from root project
------------------------------------------------------------

Application tasks
-----------------
run - Runs this project as a JVM application

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Distribution tasks
------------------
assembleDist - Assembles the main distributions
distTar - Bundles the project as a distribution.
distZip - Bundles the project as a distribution.
installDist - Installs the project as a distribution as-is.
```

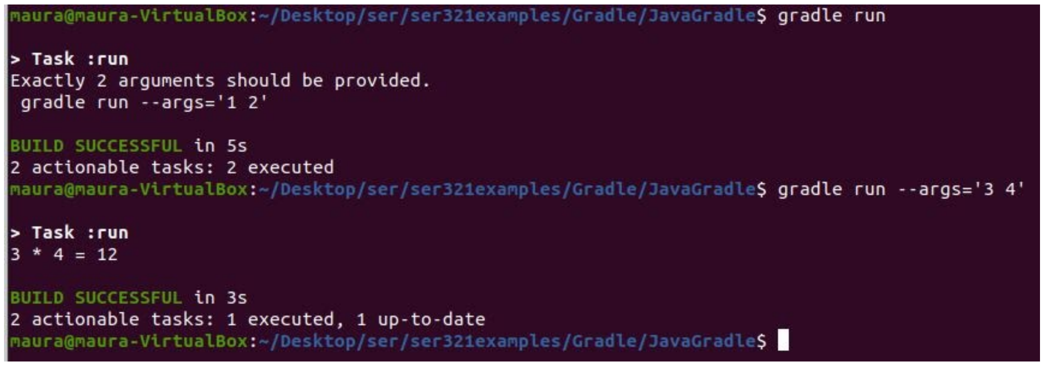


```
Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Fraction Tasks tasks
--------------------
runFraction - Tasks which runs Fraction with no arguments

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'JavaGradle'.
components - Displays the components produced by root project 'JavaGradle'. [incubating]
dependencies - Displays all dependencies declared in root project 'JavaGradle'.
dependencyInsight - Displays the insight into a specific dependency in root project 'JavaGradle'.
dependentComponents - Displays the dependent components of components in root project 'JavaGradle'. [incubating]
help - Displays a help message.
model - Displays the configuration model of root project 'JavaGradle'. [incubating]
outgoingVariants - Displays the outgoing variants of root project 'JavaGradle'.
projects - Displays the sub-projects of root project 'JavaGradle'.
properties - Displays the properties of root project 'JavaGradle'.
tasks - Displays the tasks runnable from root project 'JavaGradle'.

Multiply tasks tasks
--------------------
runApp - Tasks which runs Multiply with default parameters
runAppAgain - Tasks which runs Multiply with default parameters or given values

Verification tasks
------------------
check - Runs all checks.
test - Runs the unit tests.
```



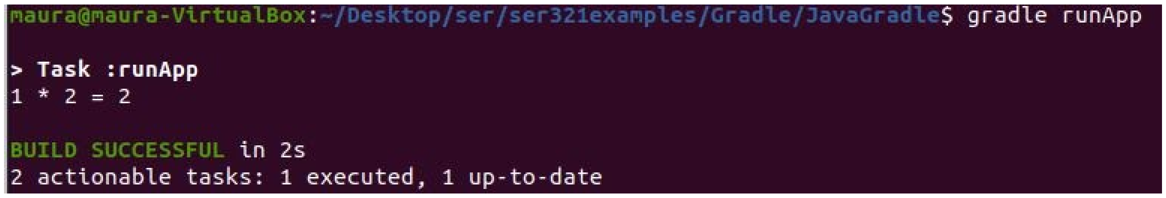
```
Other tasks
-----------
compileJava - Compiles main Java source.
compileTestJava - Compiles test Java source.
prepareKotlinBuildScriptModel
processResources - Processes main resources.
processTestResources - Processes test resources.
startScripts - Creates OS specific scripts to run the project as a JVM application.

Rules
-----
Pattern: clean<TaskName>: Cleans the output files of a task.
Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.
Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.

BUILD SUCCESSFUL in 3s
1 actionable task: 1 executed
<-------------> 0% WAITING
> IDLE
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ 
```

This shows that when gradle run is used it lets you know how the arguments need to be formatted and the results of running the program which runs the multiply task



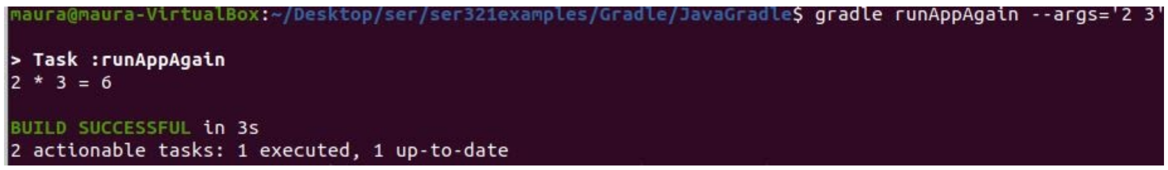
```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle run

> Task :run
Exactly 2 arguments should be provided.
 gradle run --args='1 2'

BUILD SUCCESSFUL in 5s
2 actionable tasks: 2 executed
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle run --args='3 4'

> Task :run
3 * 4 = 12

BUILD SUCCESSFUL in 3s
2 actionable tasks: 1 executed, 1 up-to-date
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ 
```

This shows runApp which does not require any arguments and uses default arguments



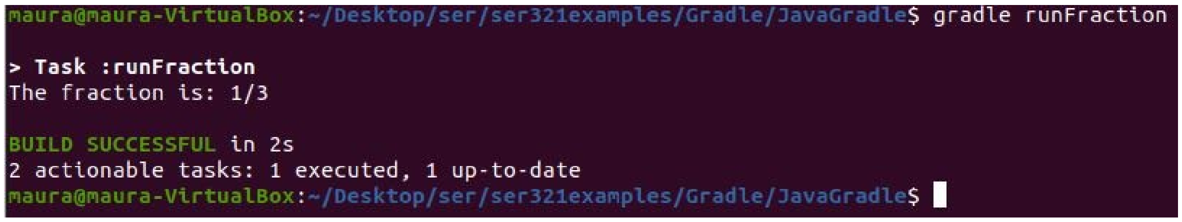
```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle runApp

> Task :runApp
1 * 2 = 2

BUILD SUCCESSFUL in 2s
2 actionable tasks: 1 executed, 1 up-to-date
```

runAppAgain allows you to run the app with new arguments



```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle runAppAgain --args='2 3'

> Task :runAppAgain
2 * 3 = 6

BUILD SUCCESSFUL in 3s
2 actionable tasks: 1 executed, 1 up-to-date
```

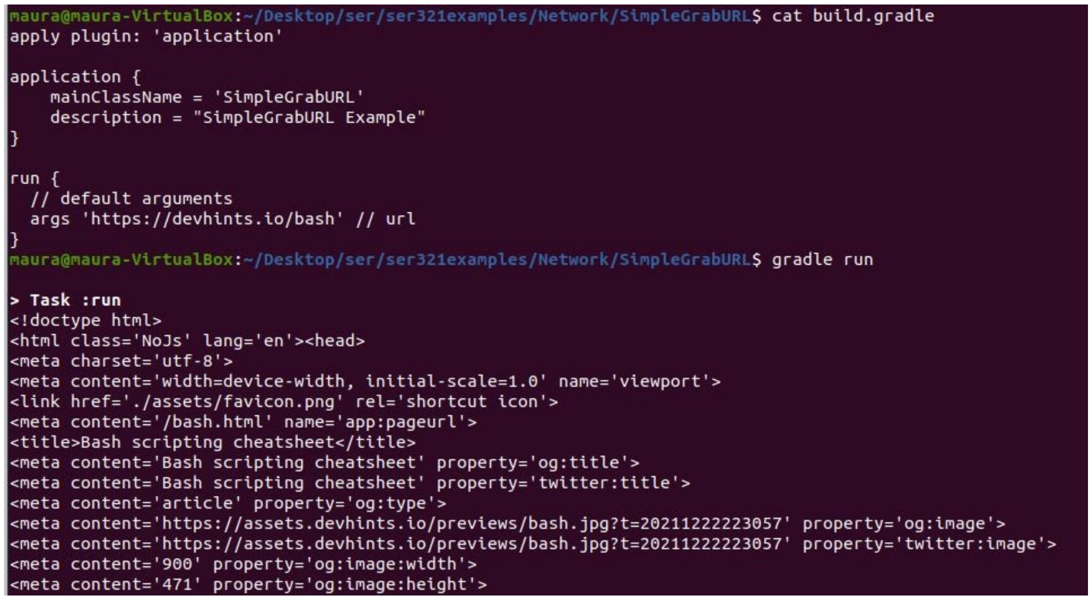
runFraction requires no arguments and prints “The fraction is:⅓”



```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ gradle runFraction

> Task :runFraction
The fraction is: 1/3

BUILD SUCCESSFUL in 2s
2 actionable tasks: 1 executed, 1 up-to-date
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Gradle/JavaGradle$ 
```

Example 3:
SimpleGrabURL

This program seems to print out the code from that url "https://devhints.io/bash"

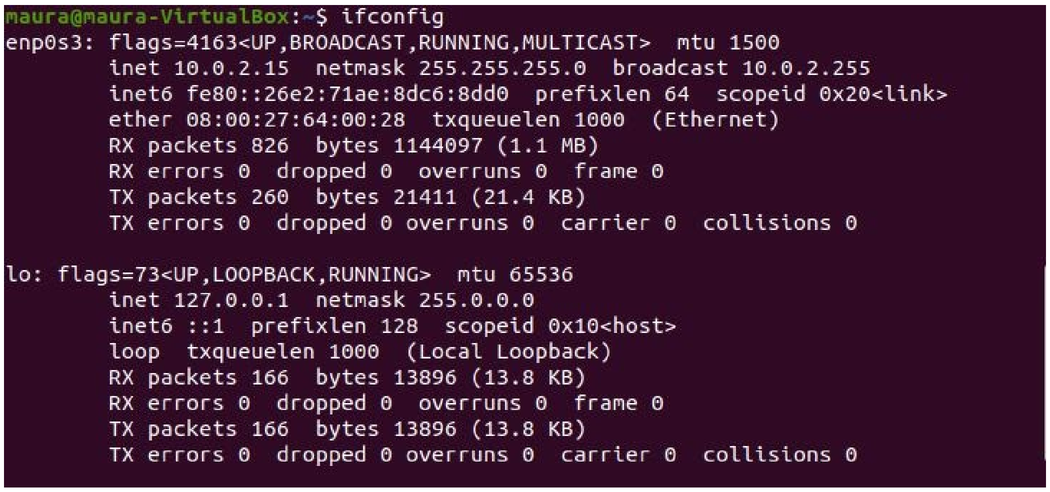


```
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Network/SimpleGrabURL$ cat build.gradle
apply plugin: 'application'

application {
    mainClassName = 'SimpleGrabURL'
    description = "SimpleGrabURL Example"
}

run {
  // default arguments
  args 'https://devhints.io/bash' // url
}
maura@maura-VirtualBox:~/Desktop/ser/ser321examples/Network/SimpleGrabURL$ gradle run

> Task :run
<!doctype html>
<html class='NoJs' lang='en'><head>
<meta charset='utf-8'>
<meta content='width=device-width, initial-scale=1.0' name='viewport'>
<link href='./assets/favicon.png' rel='shortcut icon'>
<meta content='/bash.html' name='app:pageurl'>
<title>Bash scripting cheatsheet</title>
<meta content='Bash scripting cheatsheet' property='og:title'>
<meta content='Bash scripting cheatsheet' property='twitter:title'>
<meta content='article' property='og:type'>
<meta content='https://assets.devhints.io/previews/bash.jpg?t=20211222223057' property='og:image'>
<meta content='https://assets.devhints.io/previews/bash.jpg?t=20211222223057' property='twitter:image'>
<meta content='900' property='og:image:width'>
<meta content='471' property='og:image:height'>
```

2.4
Main system: Linux
Second system: AWS

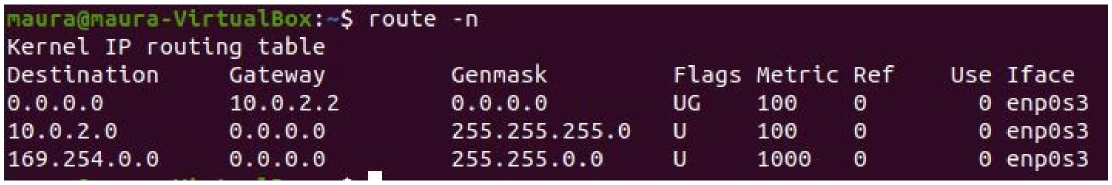
https://youtu.be/nstoxSBUsOM

Part 2
3.1

1. 

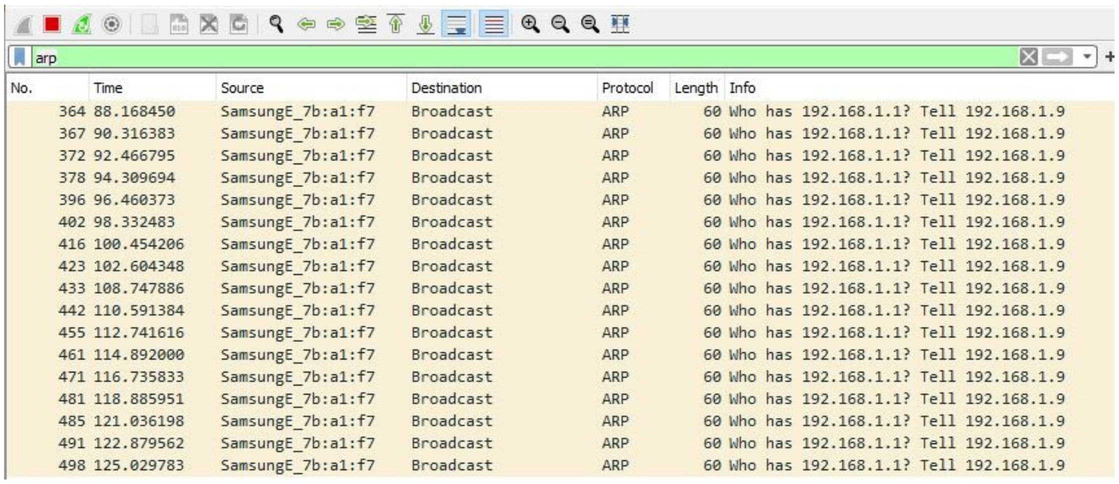
```
maura@maura-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::26e2:71ae:8dc6:8dd0  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:64:00:28  txqueuelen 1000  (Ethernet)
        RX packets 826  bytes 1144097 (1.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 260  bytes 21411 (21.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 166  bytes 13896 (13.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 166  bytes 13896 (13.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2.



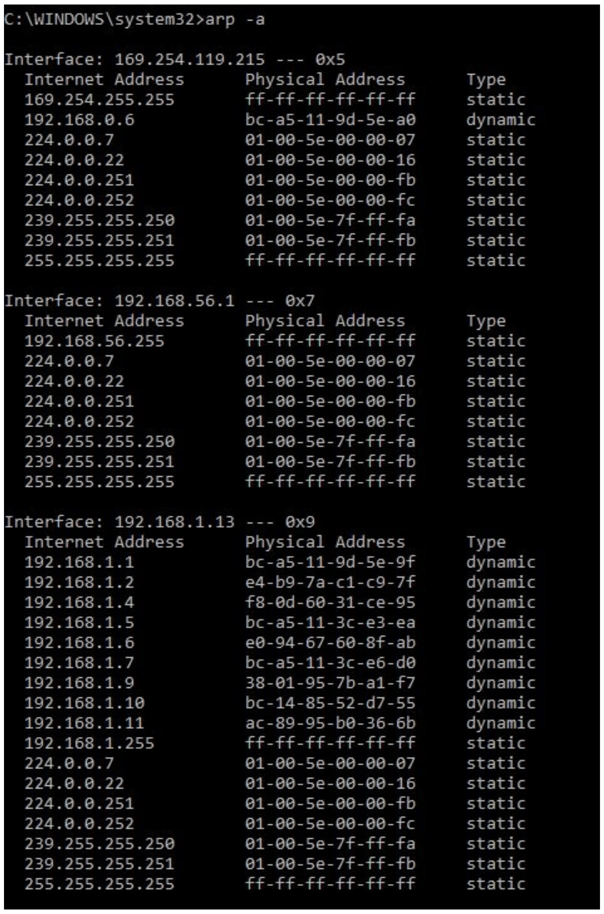
```
maura@maura-VirtualBox:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

3.

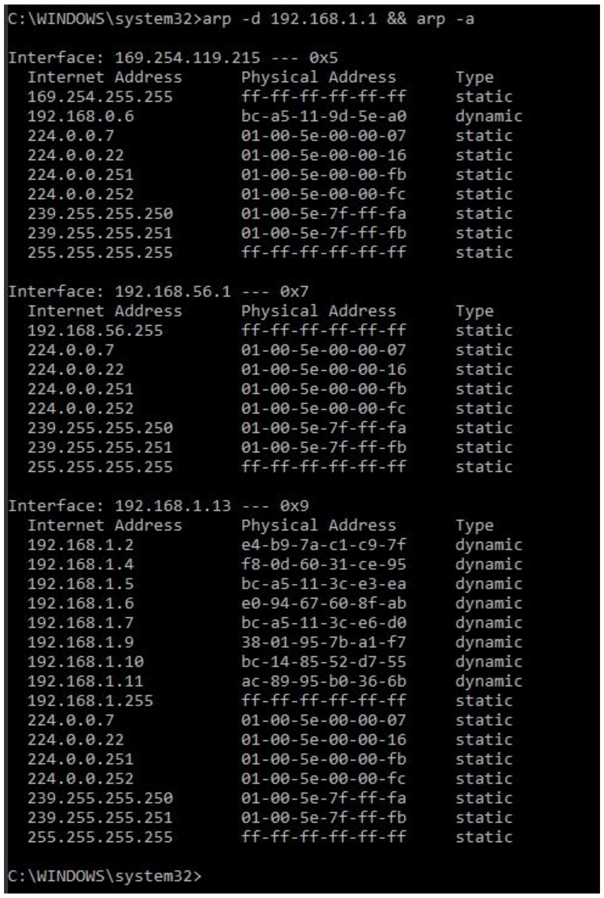


```
arp
No.   Time        Source              Destination   Protocol Length Info
  364 88.168450   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  367 90.316383   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  372 92.466795   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  378 94.309694   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  396 96.460373   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  402 98.332483   SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  416 100.454206  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  423 102.604348  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  433 108.747886  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  442 110.591384  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  455 112.741616  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  461 114.892000  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  471 116.735833  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  481 118.885951  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  485 121.036198  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  491 122.879562  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
  498 125.029783  SamsungE_7b:a1:f7   Broadcast     ARP       60 Who has 192.168.1.1? Tell 192.168.1.9
```

4.

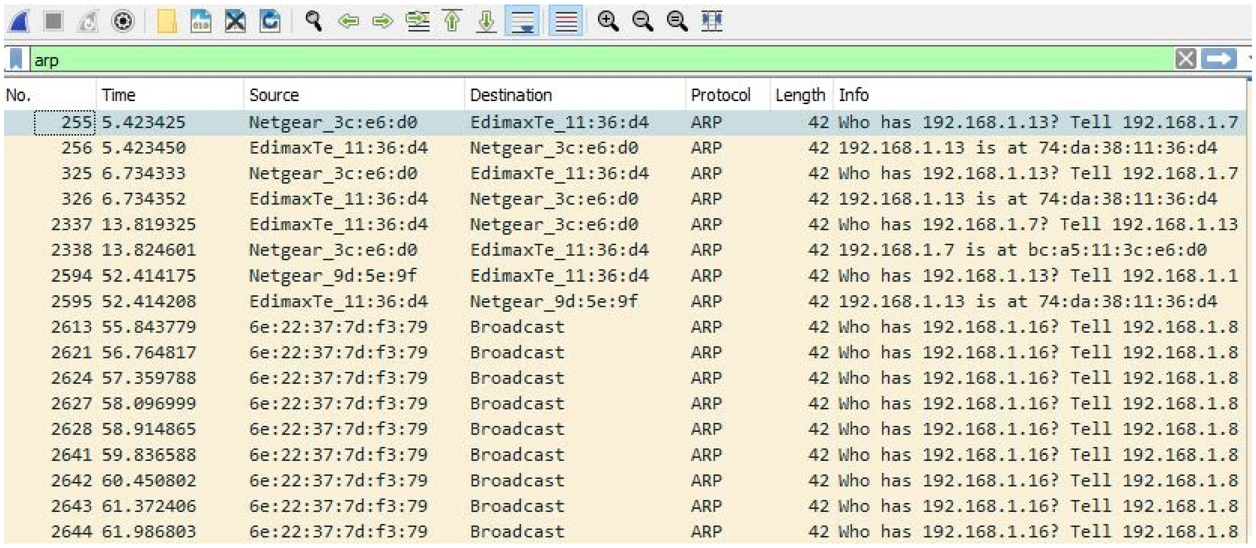


```
C:\WINDOWS\system32>arp -a

Interface: 169.254.119.215 --- 0x5
  Internet Address      Physical Address      Type
  169.254.255.255       ff-ff-ff-ff-ff-ff     static
  192.168.0.6           bc-a5-11-9d-5e-a0     dynamic
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x7
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.13 --- 0x9
  Internet Address      Physical Address      Type
  192.168.1.1           bc-a5-11-9d-5e-9f     dynamic
  192.168.1.2           e4-b9-7a-c1-c9-7f     dynamic
  192.168.1.4           f8-0d-60-31-ce-95     dynamic
  192.168.1.5           bc-a5-11-3c-e3-ea     dynamic
  192.168.1.6           e0-94-67-60-8f-ab     dynamic
  192.168.1.7           bc-a5-11-3c-e6-d0     dynamic
  192.168.1.9           38-01-95-7b-a1-f7     dynamic
  192.168.1.10          bc-14-85-52-d7-55     dynamic
  192.168.1.11          ac-89-95-b0-36-6b     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```



```
C:\WINDOWS\system32>arp -d 192.168.1.1 && arp -a

Interface: 169.254.119.215 --- 0x5
  Internet Address      Physical Address      Type
  169.254.255.255       ff-ff-ff-ff-ff-ff     static
  192.168.0.6           bc-a5-11-9d-5e-a0     dynamic
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x7
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.13 --- 0x9
  Internet Address      Physical Address      Type
  192.168.1.2           e4-b9-7a-c1-c9-7f     dynamic
  192.168.1.4           f8-0d-60-31-ce-95     dynamic
  192.168.1.5           bc-a5-11-3c-e3-ea     dynamic
  192.168.1.6           e0-94-67-60-8f-ab     dynamic
  192.168.1.7           bc-a5-11-3c-e6-d0     dynamic
  192.168.1.9           38-01-95-7b-a1-f7     dynamic
  192.168.1.10          bc-14-85-52-d7-55     dynamic
  192.168.1.11          ac-89-95-b0-36-6b     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.7             01-00-5e-00-00-07     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  239.255.255.251       01-00-5e-7f-ff-fb     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\WINDOWS\system32>
```

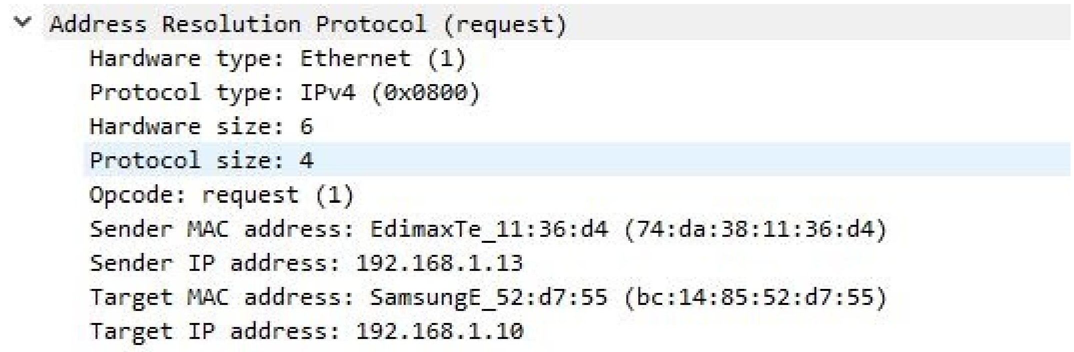
5.



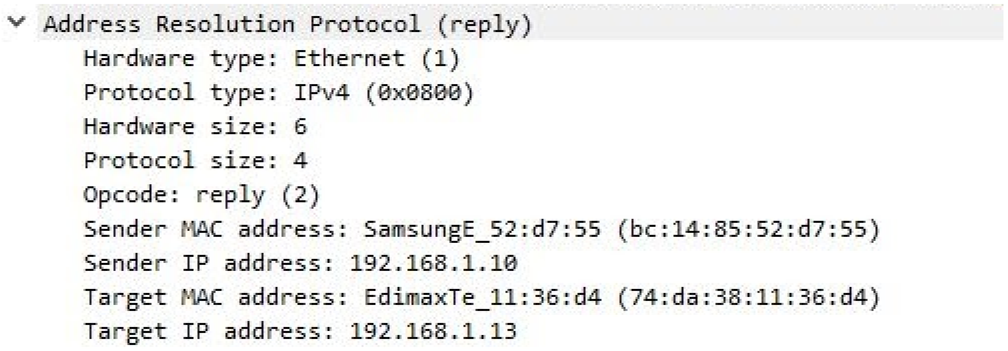
arp

No.	Time	Source	Destination	Protocol	Length	Info
255	5.423425	Netgear_3c:e6:d0	EdimaxTe_11:36:d4	ARP	42	Who has 192.168.1.13? Tell 192.168.1.7
256	5.423450	EdimaxTe_11:36:d4	Netgear_3c:e6:d0	ARP	42	192.168.1.13 is at 74:da:38:11:36:d4
325	6.734333	Netgear_3c:e6:d0	EdimaxTe_11:36:d4	ARP	42	Who has 192.168.1.13? Tell 192.168.1.7
326	6.734352	EdimaxTe_11:36:d4	Netgear_3c:e6:d0	ARP	42	192.168.1.13 is at 74:da:38:11:36:d4
2337	13.819325	EdimaxTe_11:36:d4	Netgear_3c:e6:d0	ARP	42	Who has 192.168.1.7? Tell 192.168.1.13
2338	13.824601	Netgear_3c:e6:d0	EdimaxTe_11:36:d4	ARP	42	192.168.1.7 is at bc:a5:11:3c:e6:d0
2594	52.414175	Netgear_9d:5e:9f	EdimaxTe_11:36:d4	ARP	42	Who has 192.168.1.13? Tell 192.168.1.1
2595	52.414208	EdimaxTe_11:36:d4	Netgear_9d:5e:9f	ARP	42	192.168.1.13 is at 74:da:38:11:36:d4
2613	55.843779	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2621	56.764817	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2624	57.359788	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2627	58.096999	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2628	58.914865	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2641	59.836588	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2642	60.450802	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2643	61.372406	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8
2644	61.986803	6e:22:37:7d:f3:79	Broadcast	ARP	42	Who has 192.168.1.16? Tell 192.168.1.8

Step 2



```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: EdimaxTe_11:36:d4 (74:da:38:11:36:d4)
    Sender IP address: 192.168.1.13
    Target MAC address: SamsungE_52:d7:55 (bc:14:85:52:d7:55)
    Target IP address: 192.168.1.10
```



```
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: SamsungE_52:d7:55 (bc:14:85:52:d7:55)
    Sender IP address: 192.168.1.10
    Target MAC address: EdimaxTe_11:36:d4 (74:da:38:11:36:d4)
    Target IP address: 192.168.1.13
```

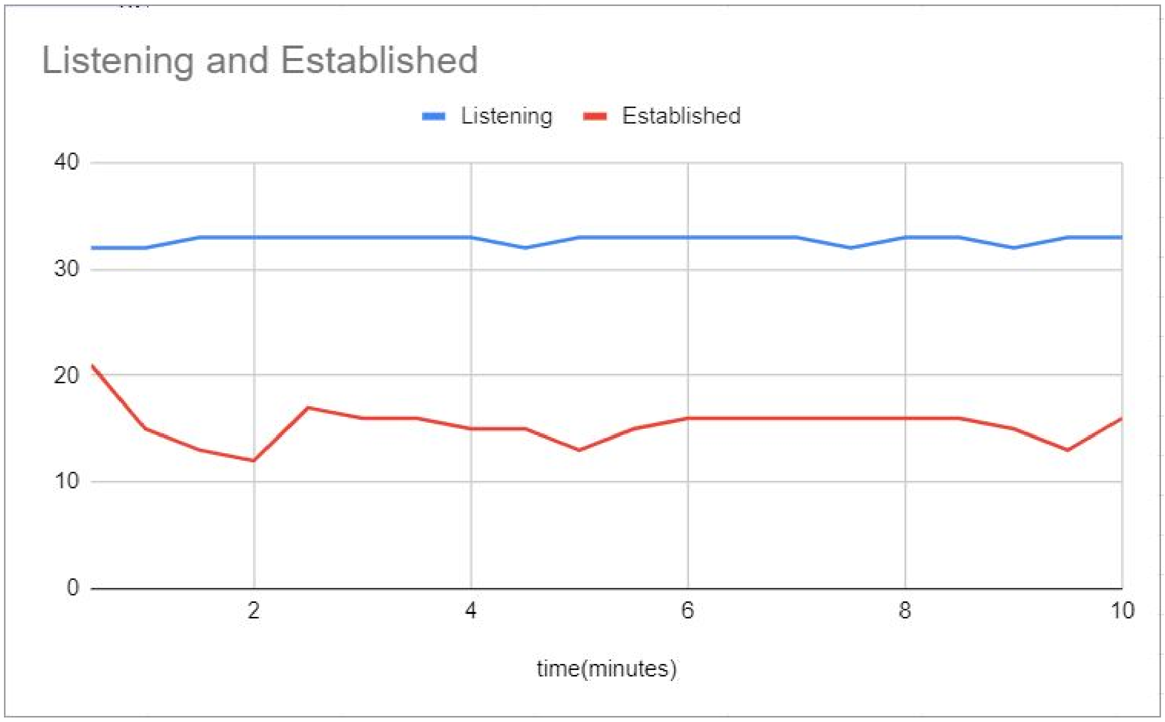
Step 3

1. Request:1 Reply:2
2. The request header is 46 bytes with 18 bytes of padding and the reply has the same size header.
   source: https://www.practicalnetworking.net/series/arp/traditional-arp/
3. 00:00:00_00:00:00

4. 0x806

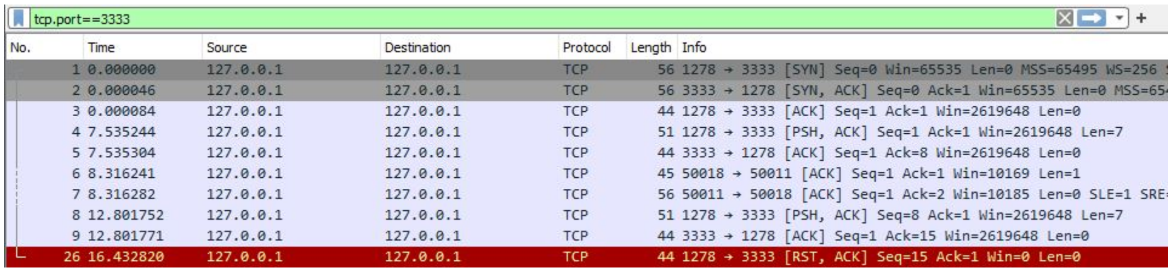
3.2

Command used: netstat -n 30 -q | findstr "ESTABLISHED LISTEN"



3.3

TCP



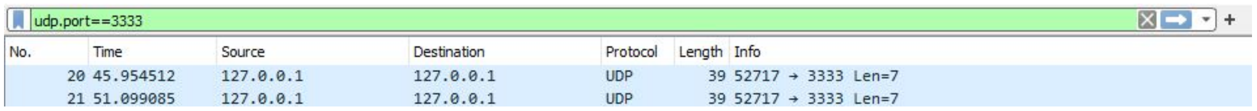
```
tcp.port==3333
No.     Time        Source       Destination  Protocol  Length Info
      1 0.000000    127.0.0.1    127.0.0.1    TCP       56 1278 → 3333 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256
      2 0.000046    127.0.0.1    127.0.0.1    TCP       56 3333 → 1278 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65
      3 0.000084    127.0.0.1    127.0.0.1    TCP       44 1278 → 3333 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
      4 7.535244    127.0.0.1    127.0.0.1    TCP       51 1278 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=7
      5 7.535304    127.0.0.1    127.0.0.1    TCP       44 3333 → 1278 [ACK] Seq=1 Ack=8 Win=2619648 Len=0
      6 8.316241    127.0.0.1    127.0.0.1    TCP       45 50018 → 50011 [ACK] Seq=1 Ack=1 Win=10169 Len=1
      7 8.316282    127.0.0.1    127.0.0.1    TCP       56 50011 → 50018 [ACK] Seq=1 Ack=2 Win=10185 Len=0 SLE=1 SRE
      8 12.801752   127.0.0.1    127.0.0.1    TCP       51 1278 → 3333 [PSH, ACK] Seq=8 Ack=1 Win=2619648 Len=7
      9 12.801771   127.0.0.1    127.0.0.1    TCP       44 3333 → 1278 [ACK] Seq=1 Ack=15 Win=2619648 Len=0
     26 16.432820   127.0.0.1    127.0.0.1    TCP       44 1278 → 3333 [RST, ACK] Seq=15 Ack=1 Win=0 Len=0
```

a) First command: ncat.exe -k -l 3333
    1) ncat.exe is the command to run netcat on windows.
    2) -k tells Netcat to wait for a new connection and must be used with -l
    3) -l 3333 puts it in to listen and server mode for port 3333

   Second command: ncat.exe 127.0.0.1 3333
   SER321
   Rocks!
    1) Sends the message SER321\n Rocks!\n to the listener.

b) Frames: 26
c) Packets: 7
d) Packets:10
e) 491 bytes
f) 31.77%

UDP



udp.port==3333

No.	Time	Source	Destination	Protocol	Length	Info
20	45.954512	127.0.0.1	127.0.0.1	UDP	39	52717 → 3333 Len=7
21	51.099085	127.0.0.1	127.0.0.1	UDP	39	52717 → 3333 Len=7

a) First command: ncat.exe -u -l 3333
    1) ncat.exe is the command to run netcat on windows.
    2) -u puts it in UDP mode instead of TCP
    3) -l 3333 puts it in to listen and server mode for port 3333

   Second command: ncat.exe -u 127.0.0.1 3333

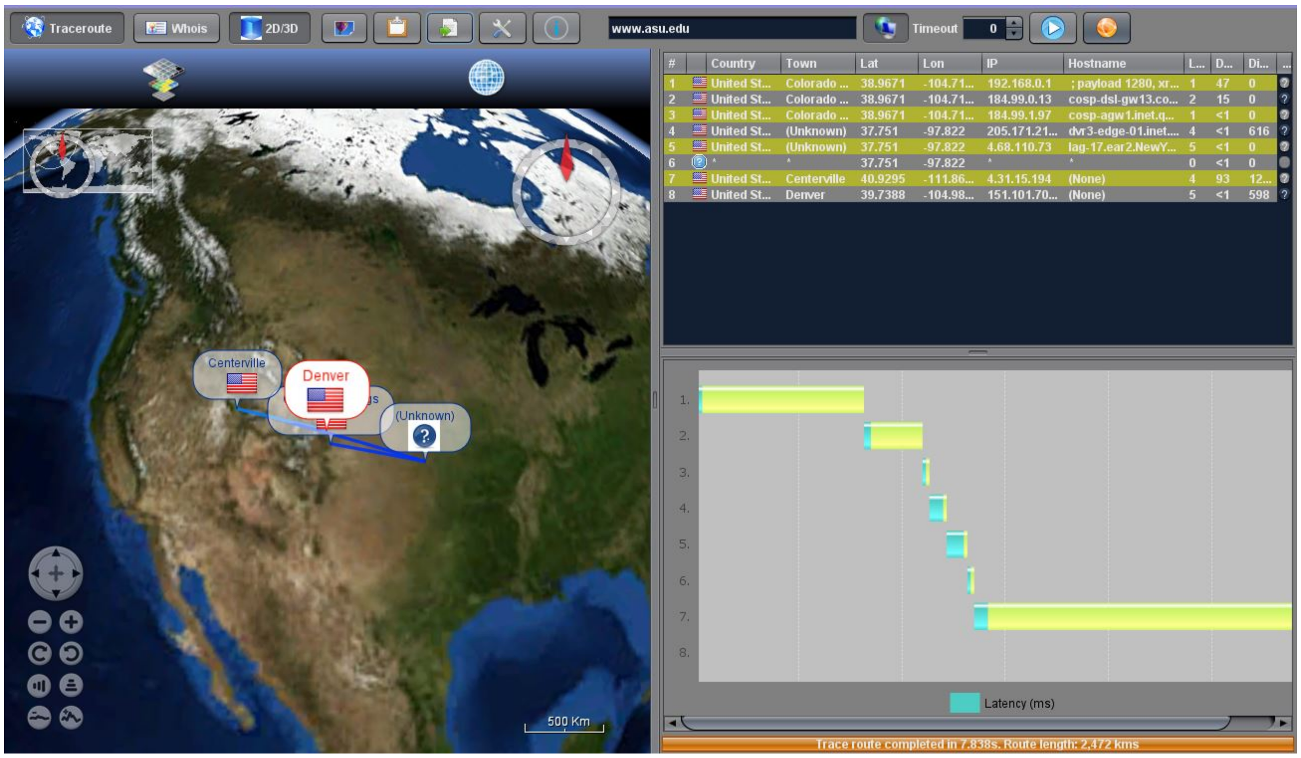
   SER321
   Rocks!

    4) Sends the message SER321\n Rocks!\n to the listener in UDP mode.

b) Frames: 21
c) Packets:2
d) Packets:2
e) D
    i) Total bytes:78
    ii) Overhead: 0%
f) TCP has 31.77% overhead while UDP has 0%. This difference is due to TCP being a connection-oriented protocol. TCP has feedback mechanisms that result in more overhead, while UDP is a connectionless, simple internet protocol that just sends the data and doesn't worry if the other side gets it.

3.4

Main Network



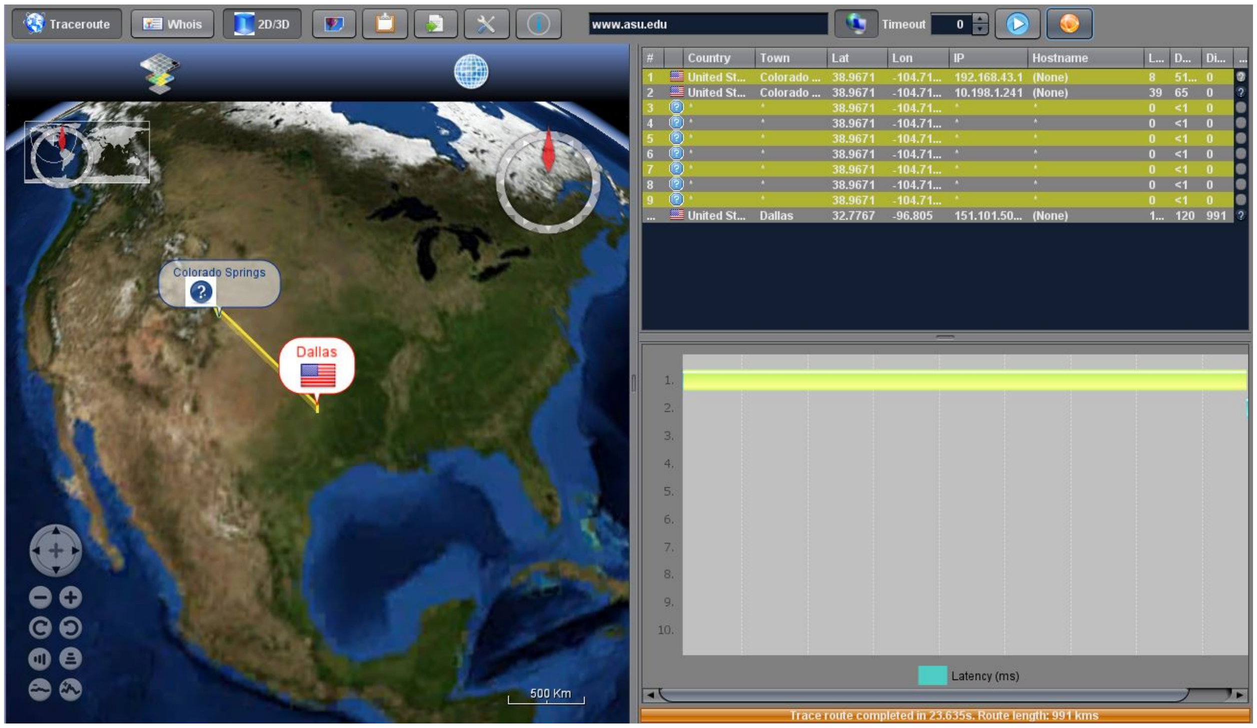
www.asu.edu

Timeout 0

#		Country	Town	Lat	Lon	IP	Hostname	L...	D...	Di...
1		United St...	Colorado ...	38.9671	-104.71...	192.168.0.1	; payload 1280, xr...	1	47	0
2		United St...	Colorado ...	38.9671	-104.71...	184.99.0.13	cosp-dsl-gw13.co...	2	15	0
3		United St...	Colorado ...	38.9671	-104.71...	184.99.1.97	cosp-agw1.inet.q...	1	<1	0
4		United St...	(Unknown)	37.751	-97.822	205.171.21...	dvr3-edge-01.inet....	4	<1	616
5		United St...	(Unknown)	37.751	-97.822	4.68.110.73	lag-17.ear2.NewY...	5	<1	0
6		*	*	37.751	-97.822	*	*	0	<1	0
7		United St...	Centerville	40.9295	-111.86...	4.31.15.194	(None)	4	93	12...
8		United St...	Denver	39.7388	-104.98...	151.101.70...	(None)	5	<1	598

Trace route completed in 7.838s. Route length: 2,472 kms

Mobile hotspot

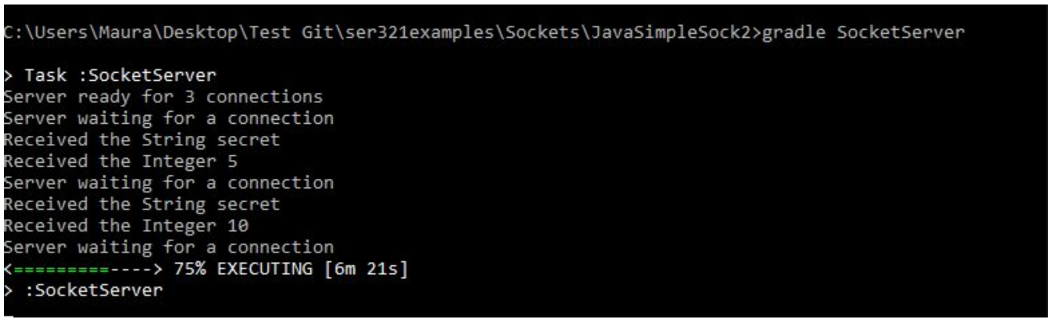


#		Country	Town	Lat	Lon	IP	Hostname	L...	D...	Di...	...
1		United St...	Colorado ...	38.9671	-104.71...	192.168.43.1	(None)	8	51...	0	?
2		United St...	Colorado ...	38.9671	-104.71...	10.198.1.241	(None)	39	65	0	?
3	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
4	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
5	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
6	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
7	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
8	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
9	?	*	*	38.9671	-104.71...	*	*	0	<1	0	●
...		United St...	Dallas	32.7767	-96.805	151.101.50...	(None)	1...	120	991	?

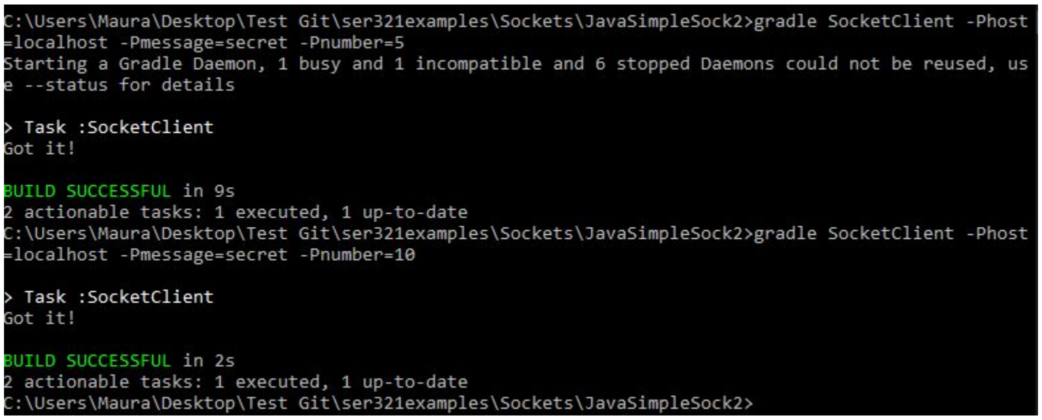
a) The main network was much faster at 7.838s compared to 23.635s on the mobile hotspot.
b) The mobile hotspot only had one hop to Dallas while the main network had 3 hops.

3.5.1

https://youtu.be/vfFmJLSydaQ

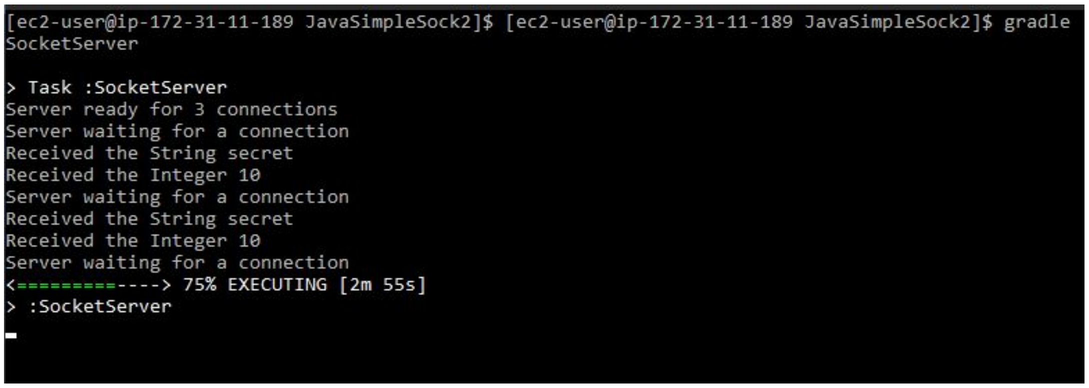


```
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>gradle SocketServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 10
Server waiting for a connection
<=========----> 75% EXECUTING [6m 21s]
> :SocketServer
```

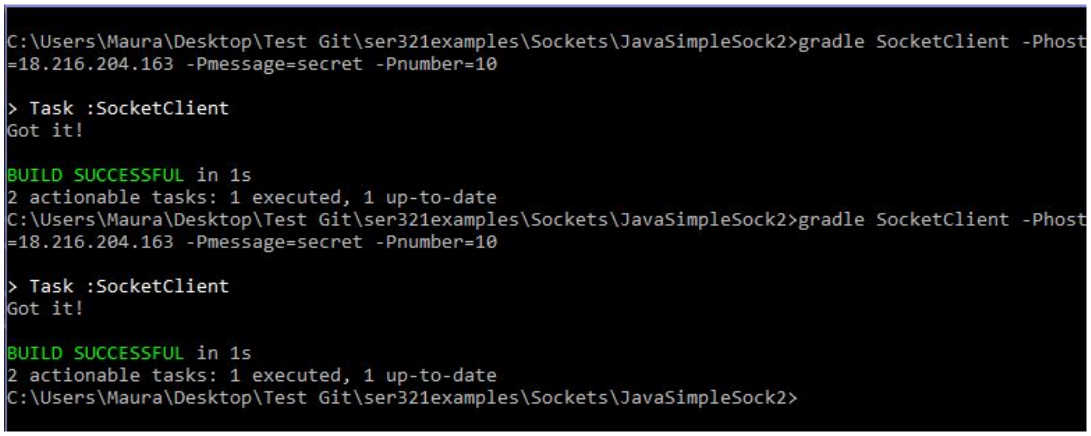


```
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>gradle SocketClient -Phost
=localhost -Pmessage=secret -Pnumber=5
Starting a Gradle Daemon, 1 busy and 1 incompatible and 6 stopped Daemons could not be reused, us
e --status for details

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 9s
2 actionable tasks: 1 executed, 1 up-to-date
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>gradle SocketClient -Phost
=localhost -Pmessage=secret -Pnumber=10

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 2s
2 actionable tasks: 1 executed, 1 up-to-date
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>
```

3.5.2



```
[ec2-user@ip-172-31-11-189 JavaSimpleSock2]$ [ec2-user@ip-172-31-11-189 JavaSimpleSock2]$ gradle
SocketServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String secret
Received the Integer 10
Server waiting for a connection
Received the String secret
Received the Integer 10
Server waiting for a connection
<=========----> 75% EXECUTING [2m 55s]
> :SocketServer
```



```
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>gradle SocketClient -Phost
=18.216.204.163 -Pmessage=secret -Pnumber=10

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>gradle SocketClient -Phost
=18.216.204.163 -Pmessage=secret -Pnumber=10

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
C:\Users\Maura\Desktop\Test Git\ser321examples\Sockets\JavaSimpleSock2>
```

When running locally the host is localhost, when running on AWS the host became the IP address of the AWS. On Wireshark, the data sent to the server is condensed into a single packet that holds both the message and the number as well as the headers, whereas locally they were spread out across several packets.

3.5.3
This does not work, you can not do it in the same way because the personal computer has a router that blocks it whereas AWS does not.

3.5.4
The local IP address is the IP address of the computer and not the public-facing IP address. The public IP address will send the information to the router which will send the information to the local IP address. The router can be set up to forward port 8888 by logging into the router account and changing the settings to forward port 8888 to your local IP address.