**Lab2 Notes**

Assignment 2 is now available on canvas. Due on Sep 17, Tuesday, 11:59 EST

Bonus problem and submission template also available.

AMI available for HW 2 - **Harvard-e88-HW2-2019-new** in the “*US EAST Ohio”* region.

It has an ‘input\_files’ directory in the **'centos'** user home directory and iotop and htop utility installed.

**Agenda:**

**Introduction - Single threaded, multi-threaded and multi-process**

**Demo – python**

**AWS – instance created from Harvard-e88-HW2-2019-new**

**Iotop**

**htop**

**Demo – java**

1. **Python:**
   * single-threaded vs multi-threaded vs multi-process
   * demo in PyCharm
   * shared variables between processes:

<https://www.geeksforgeeks.org/multiprocessing-python-set-2/>

<https://eli.thegreenplace.net/2012/01/04/shared-counter-with-pythons-multiprocessing>

* instance created from AMI “**Harvard-e88-HW2-2019-new “. Has multiple cores**
* Move your code to an EC2 instance - into the 'centos' home dir or any other dir you like:

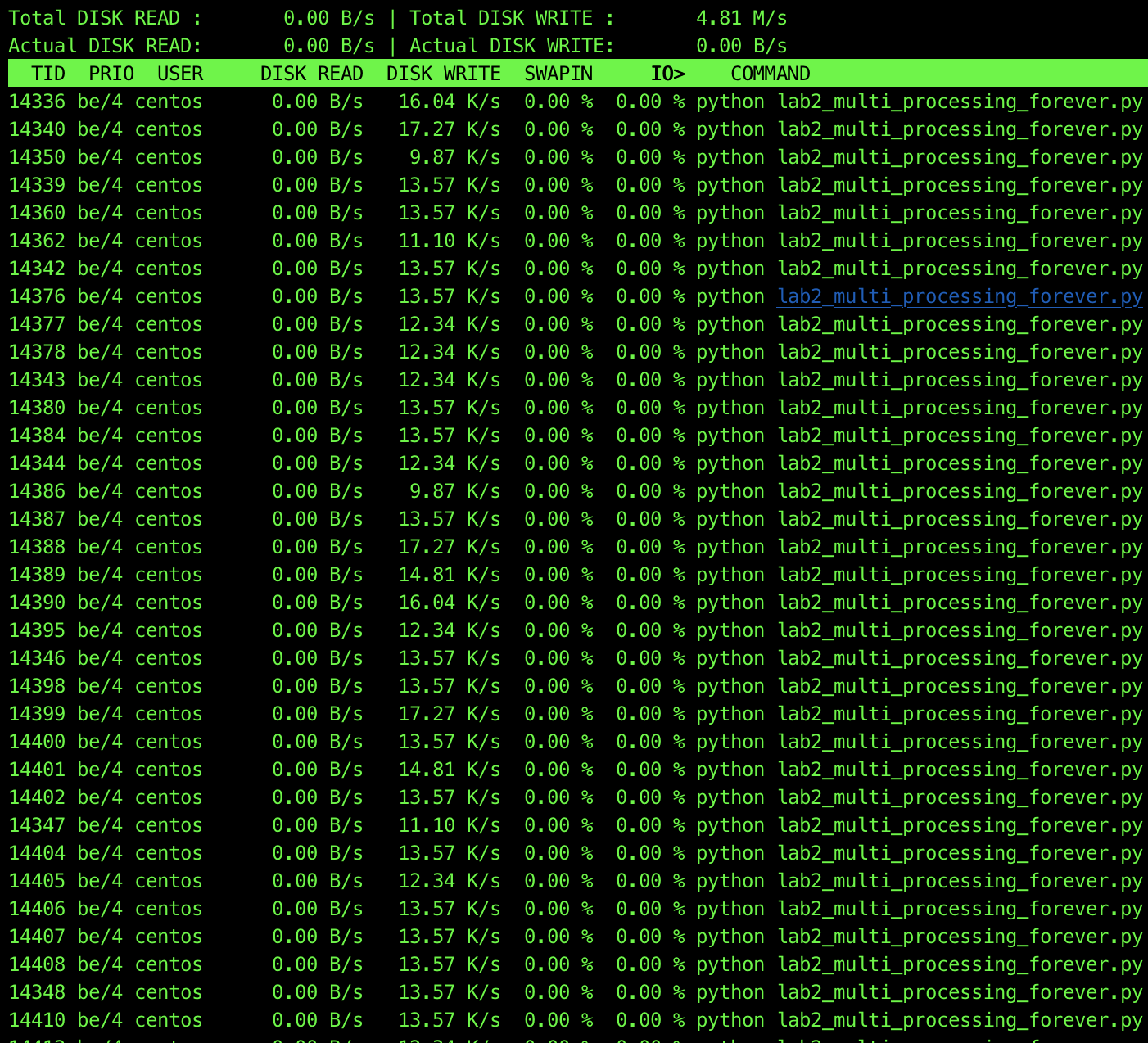
***scp -i awsKey1.pem -r /Users/pavithra/PycharmProjects/harvard-labs-2019 centos@ec2-18-220-162-101.us-east-2.compute.amazonaws.com:python\_files/***

* Experiment with the number of processes running/threads vs. number of CPUs of your instance:

*python* [*lab2\_multi\_processing\_forever.py*](http://lab2_multi_processing_forever.py) *-fc 2*

1. **EC2 setup**
   * install iotop
     + # yum install iotop

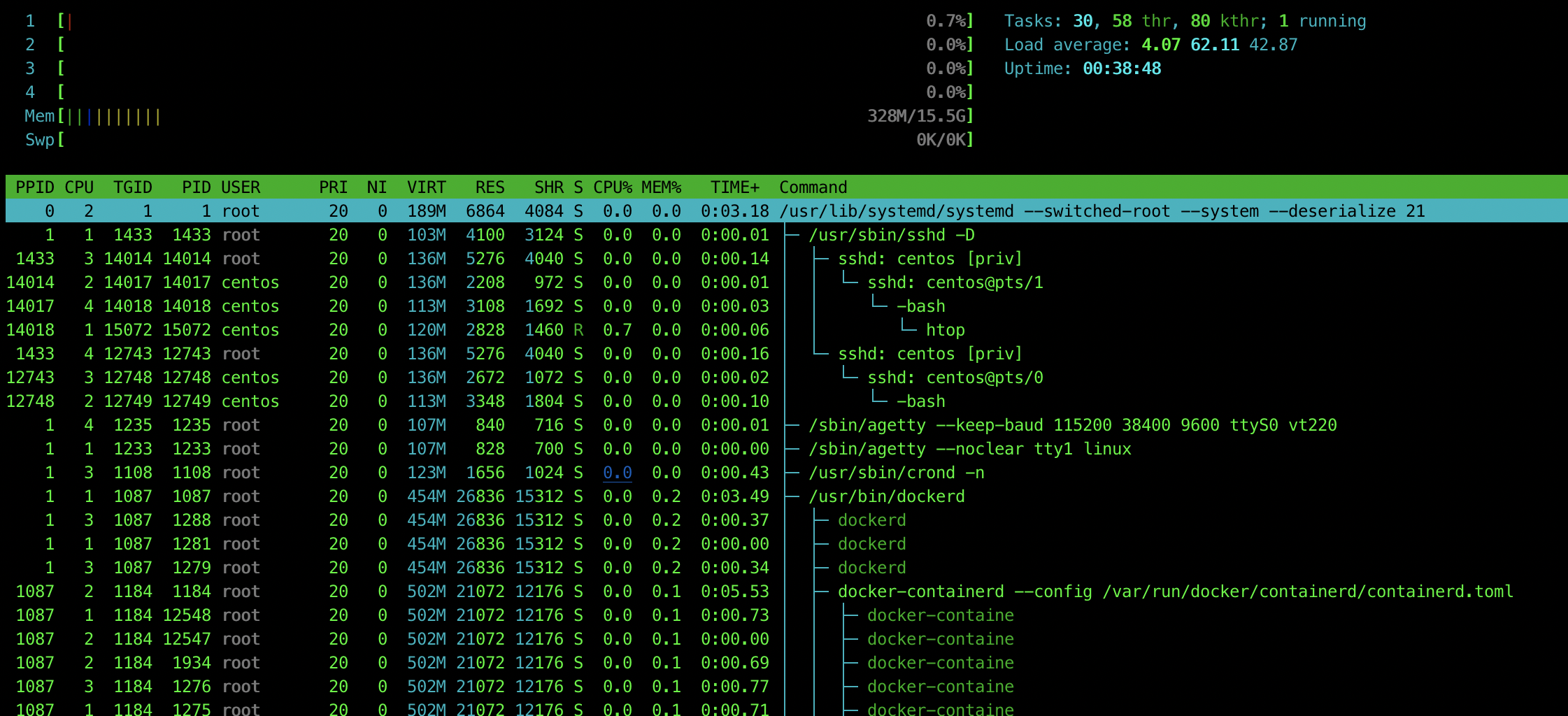
Iotop result example:



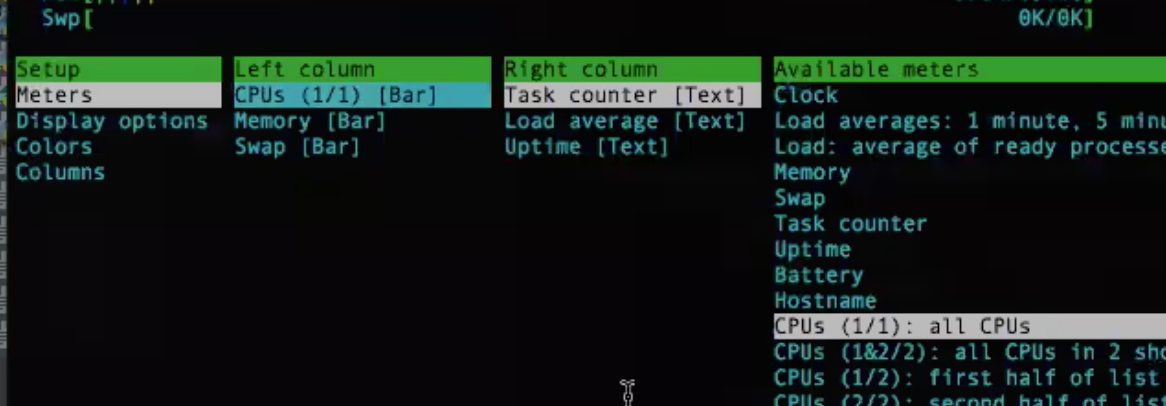
* htop –

To install : <https://www.webhostingbuzz.com/wiki/install-htop-linux/>

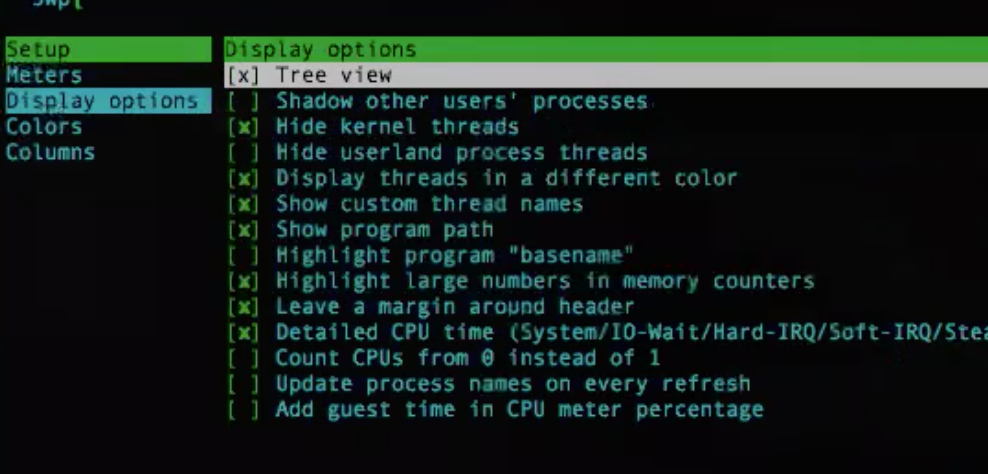
htop result example:



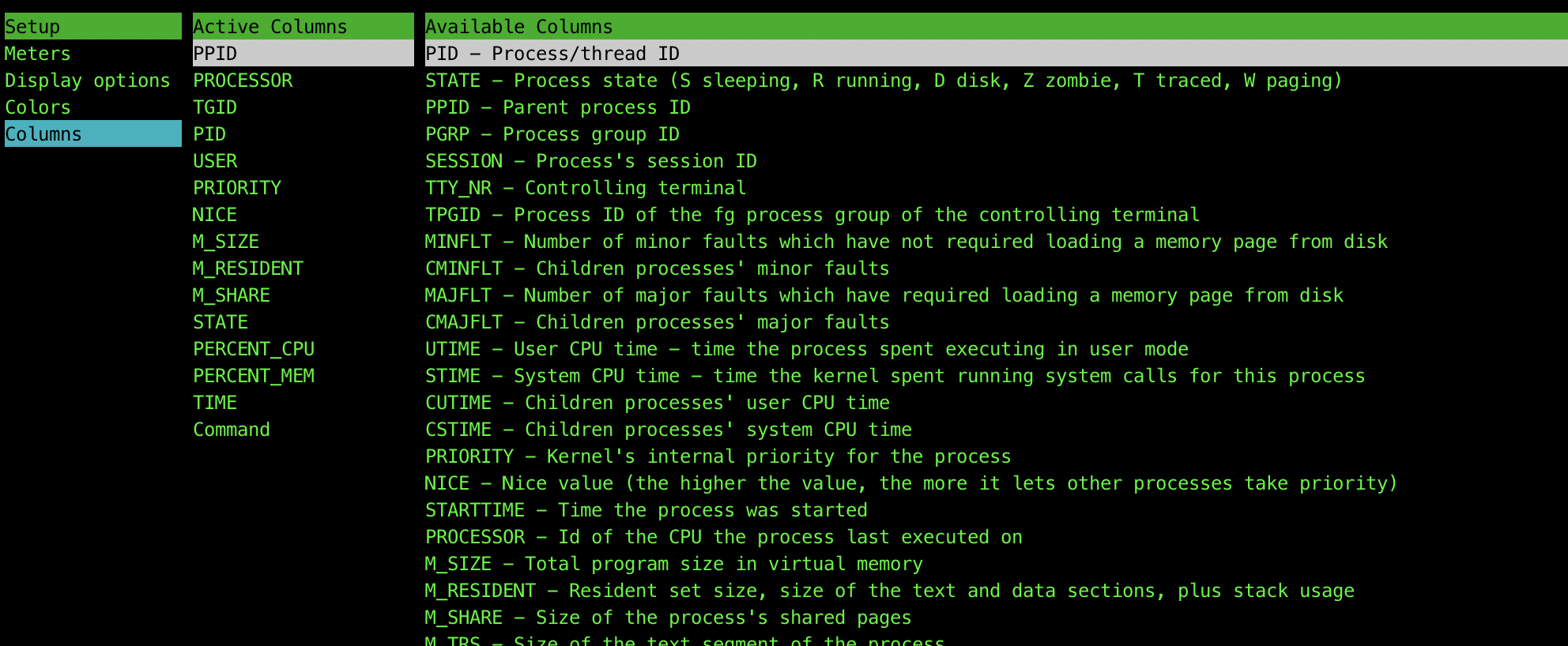
* install htop; configuration options:
  + Meters: CPU (1/1) - all:



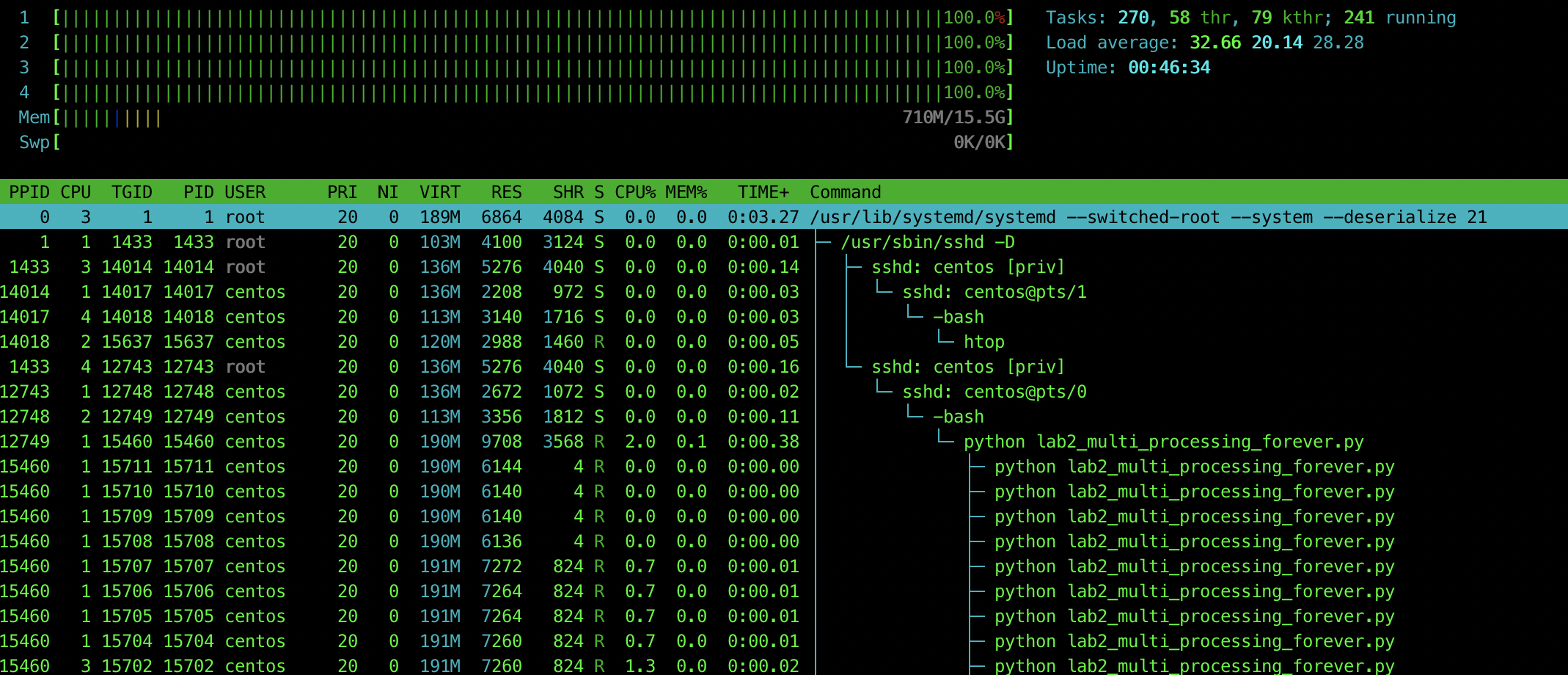
* + Display options: Tree view



* + Columns: PPID, Processor, CPU ….



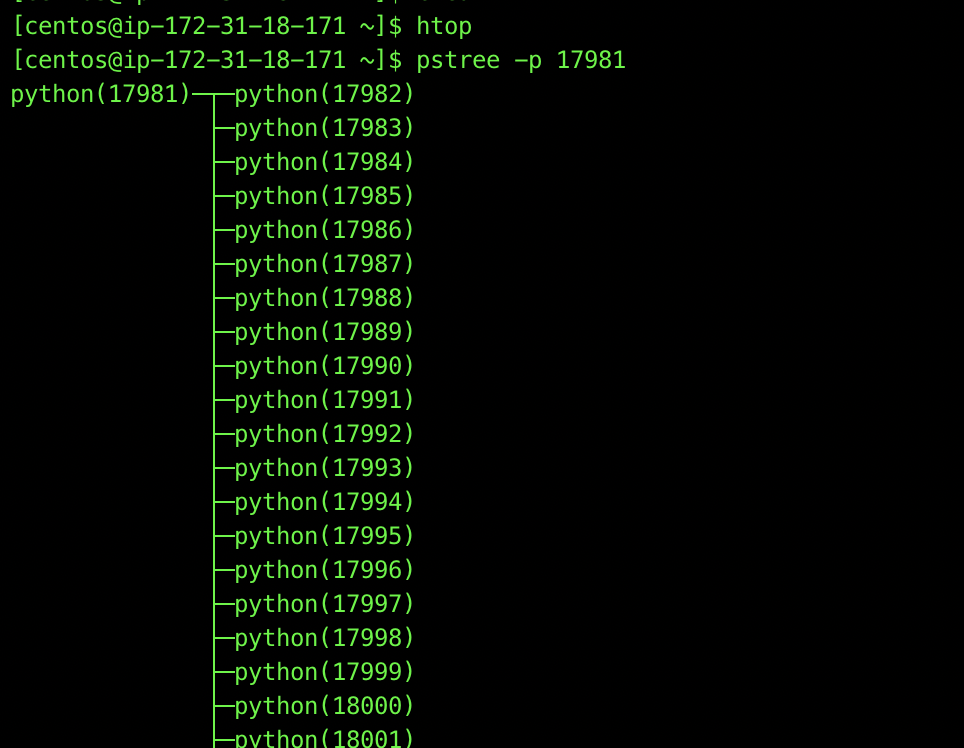
* + htop exploring
    - see sub-processes of the main one
    - see different CPUs being used



* + pstree -p mainPID - see a different view of the process hierarchy

to install pstree:

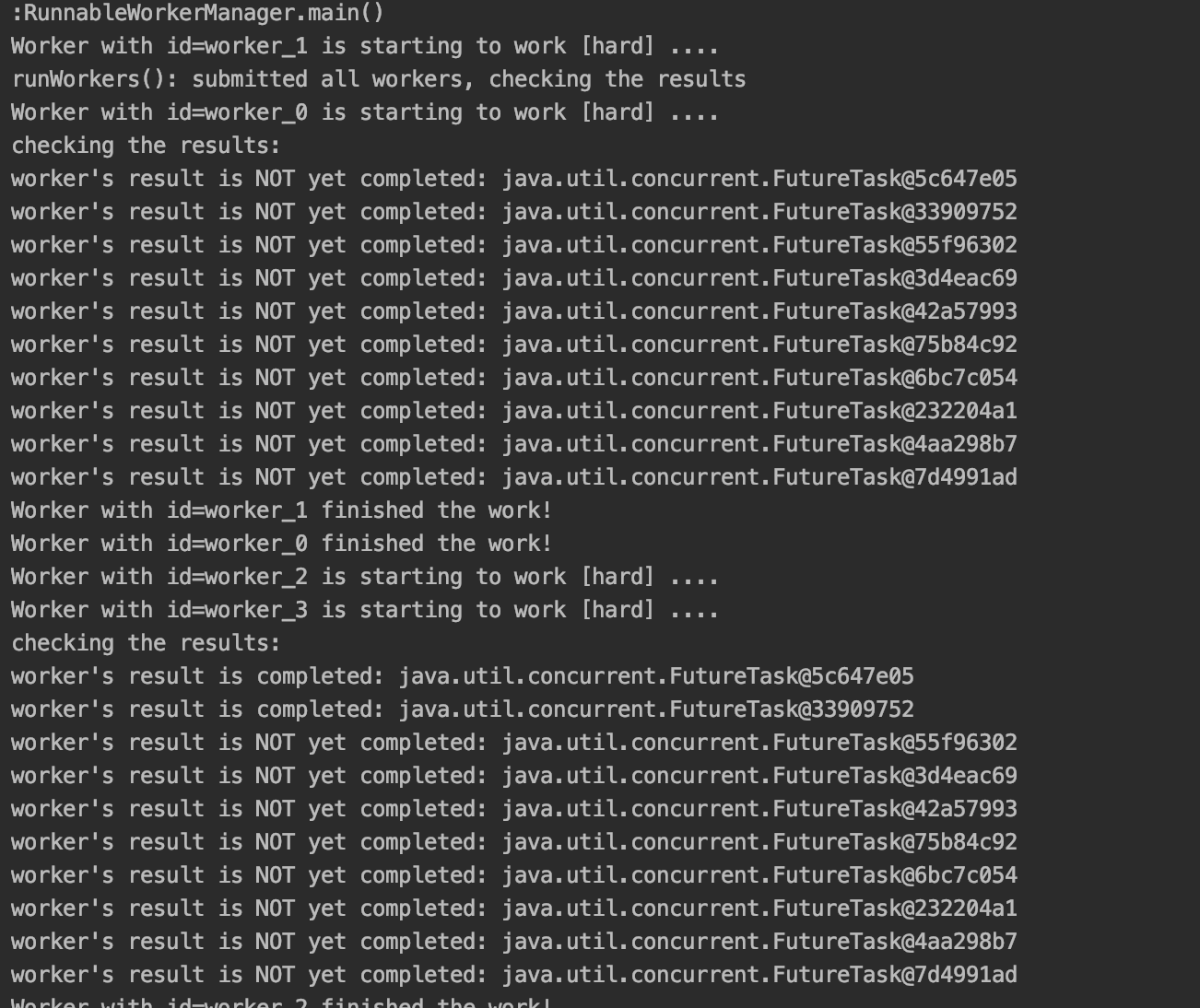
yum install psmisc



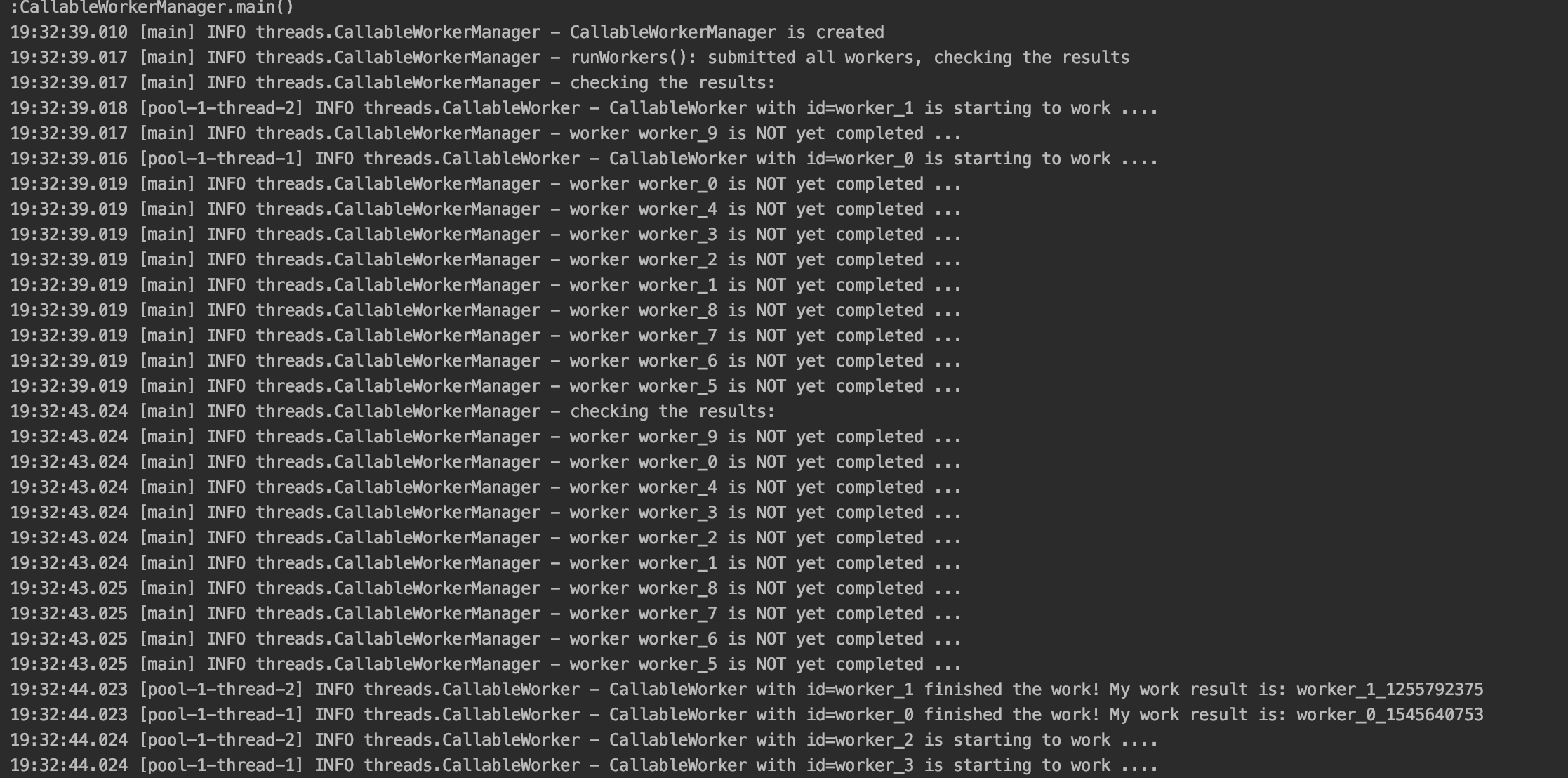
**Java**

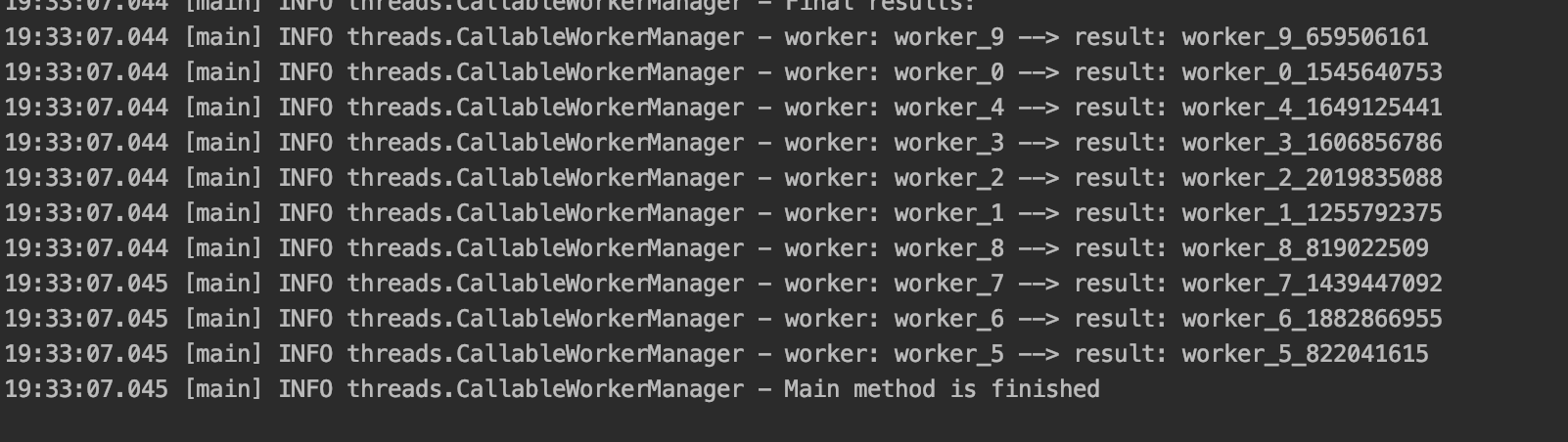
* single-threaded : main() thread
* thread pools via Executor service - why we need pools of threads
* multi-threaded - Runnable vs Callback interface

Output of RunnableWorkerManager:



Output of CallableWorkerManager:

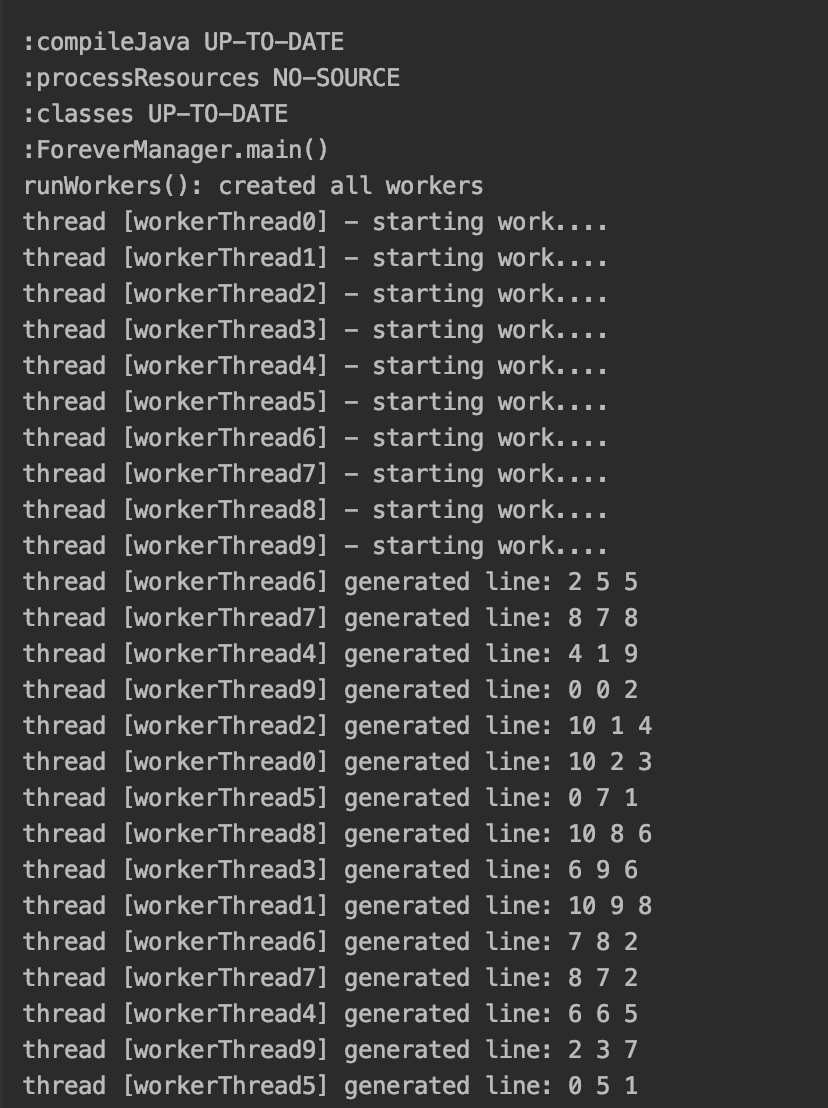




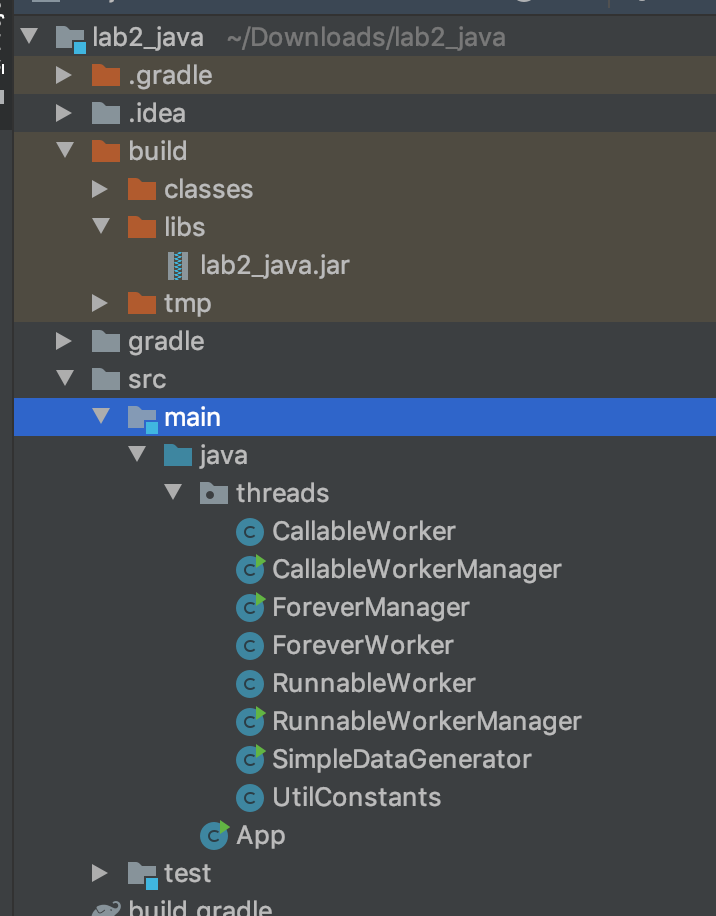
* ForeverWorker - generating random numbers forever - to view resource utilization by threads

(unlike the other programs, arguments are not hard coded. Enter Program arguments- in this case, an integer = number of threads)

Output of ForeverManager:



* How to create a basic JAR - from Intellij IDEA , eclipse
  + For Intellij IDEA, since we are using gradle –
  + In the Intellij terminal – run ./gradlew jar
  + A jar file will be created under build directory >libs



More info : <https://stackoverflow.com/questions/1082580/how-to-build-jars-from-intellij-properly>

* + For eclipse : <https://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Ftasks%2Ftasks-33.htm>
* Scp your jar to your EC2 instance:

*scp -i ./cscie88-2018.pem ./Labs/Lab2/lab2.jar* *centos@ec2-18-223-159-243.us-east-2.compute.amazonaws.com:/home/centos/lab2\_java*

* ssh to your EC2 instance:

*ssh -i .../cscie88-2018.pem* *centos@ec2-18-223-159-243.us-east-2.compute.amazonaws.com*