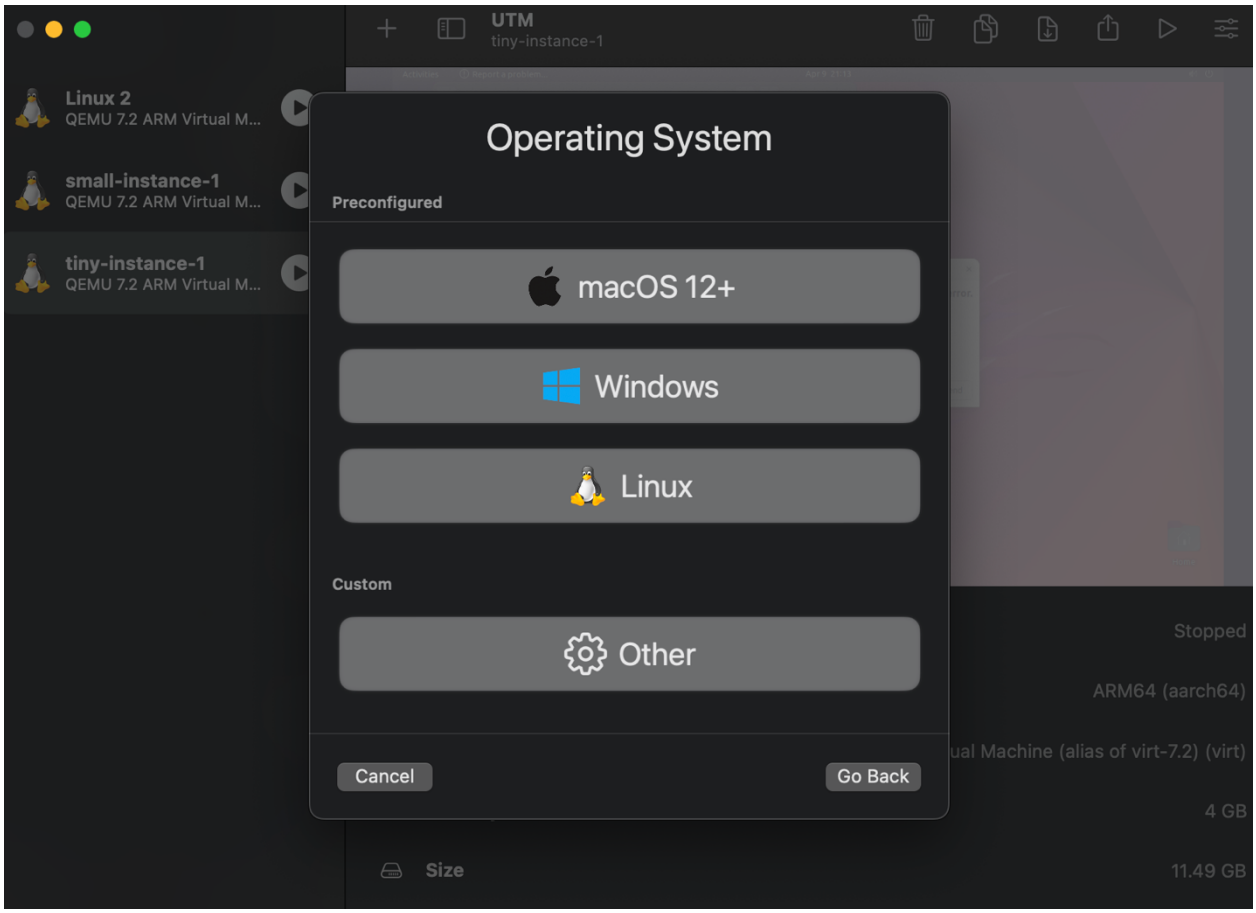
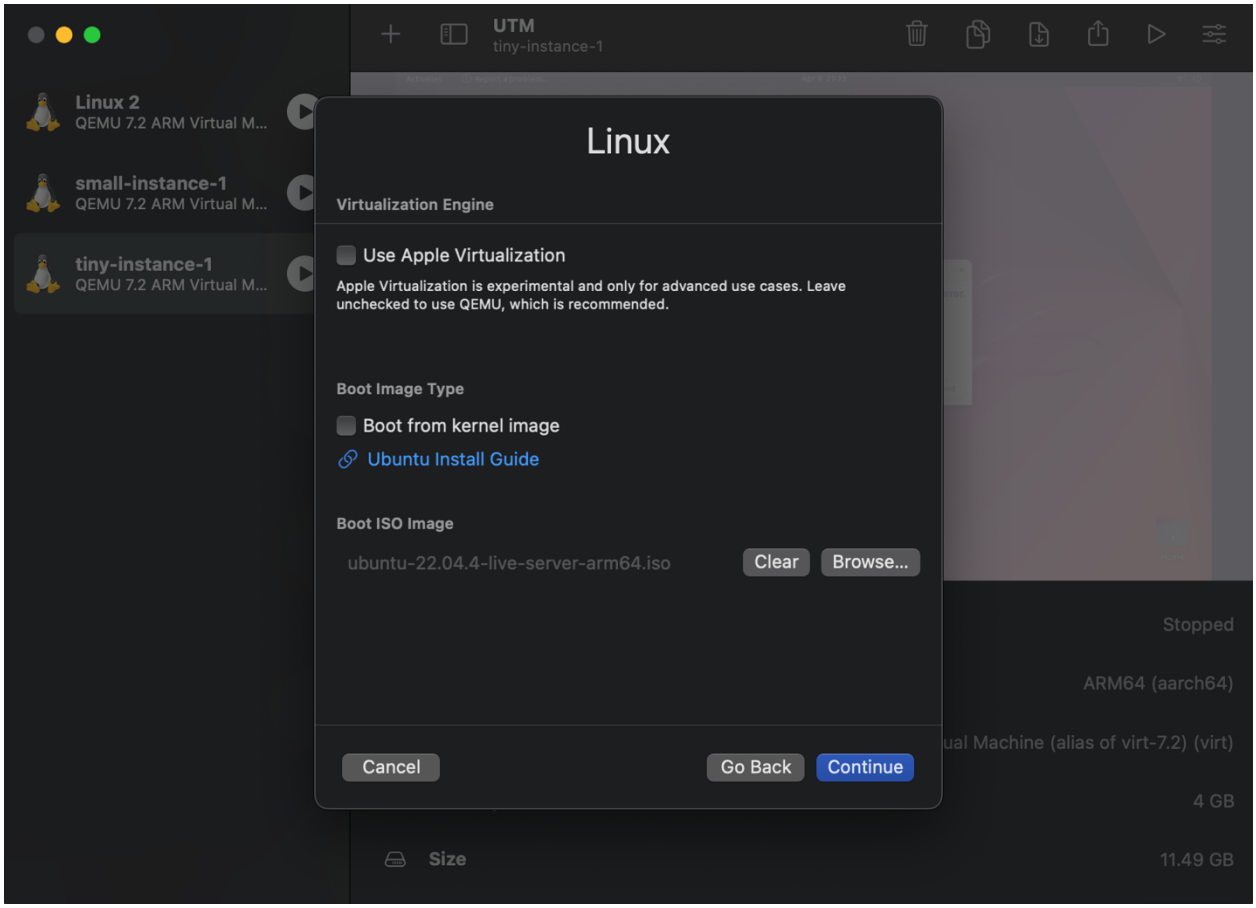
Homework 5 – Hadoop and Spark

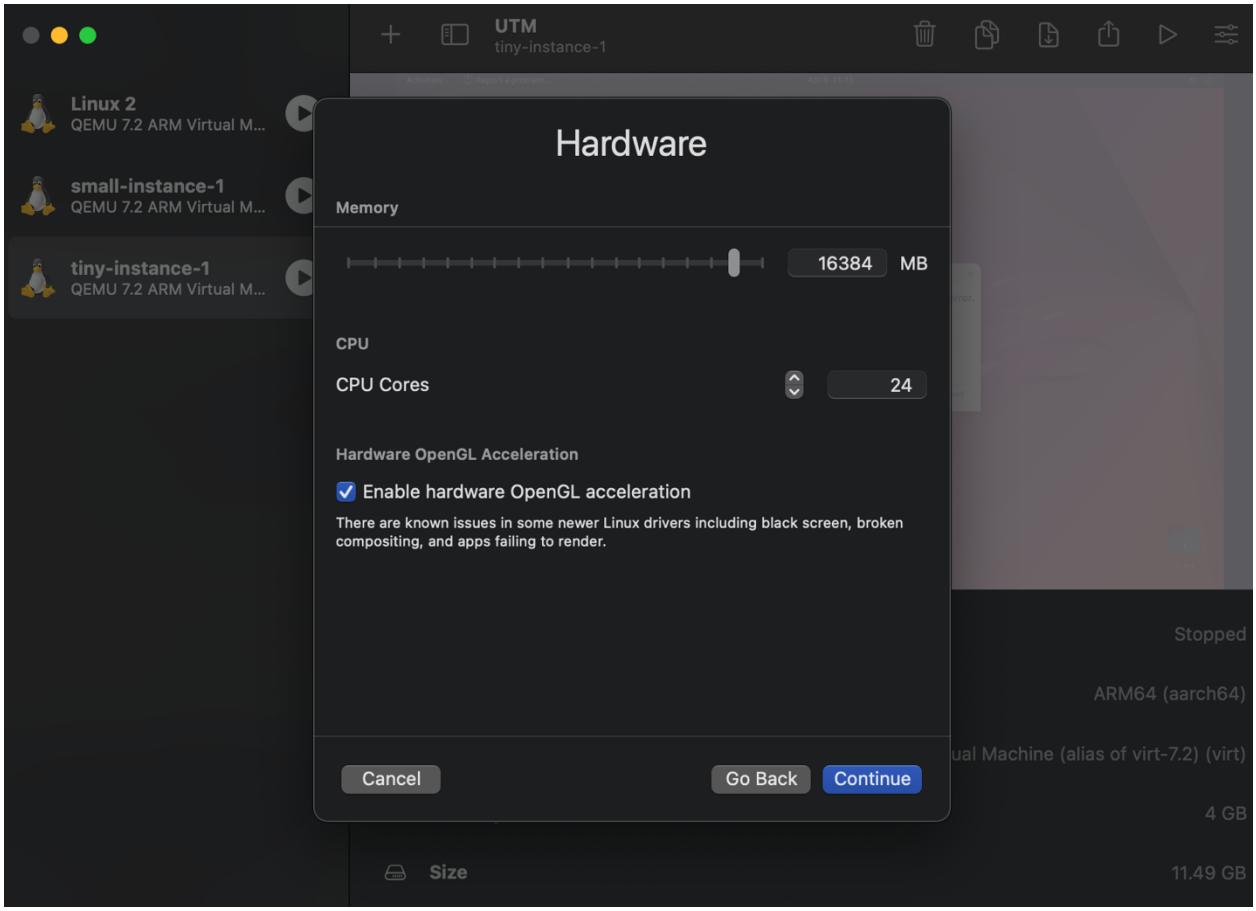
- Darshan S Nair

VM Setup (In my personal device using UTM)

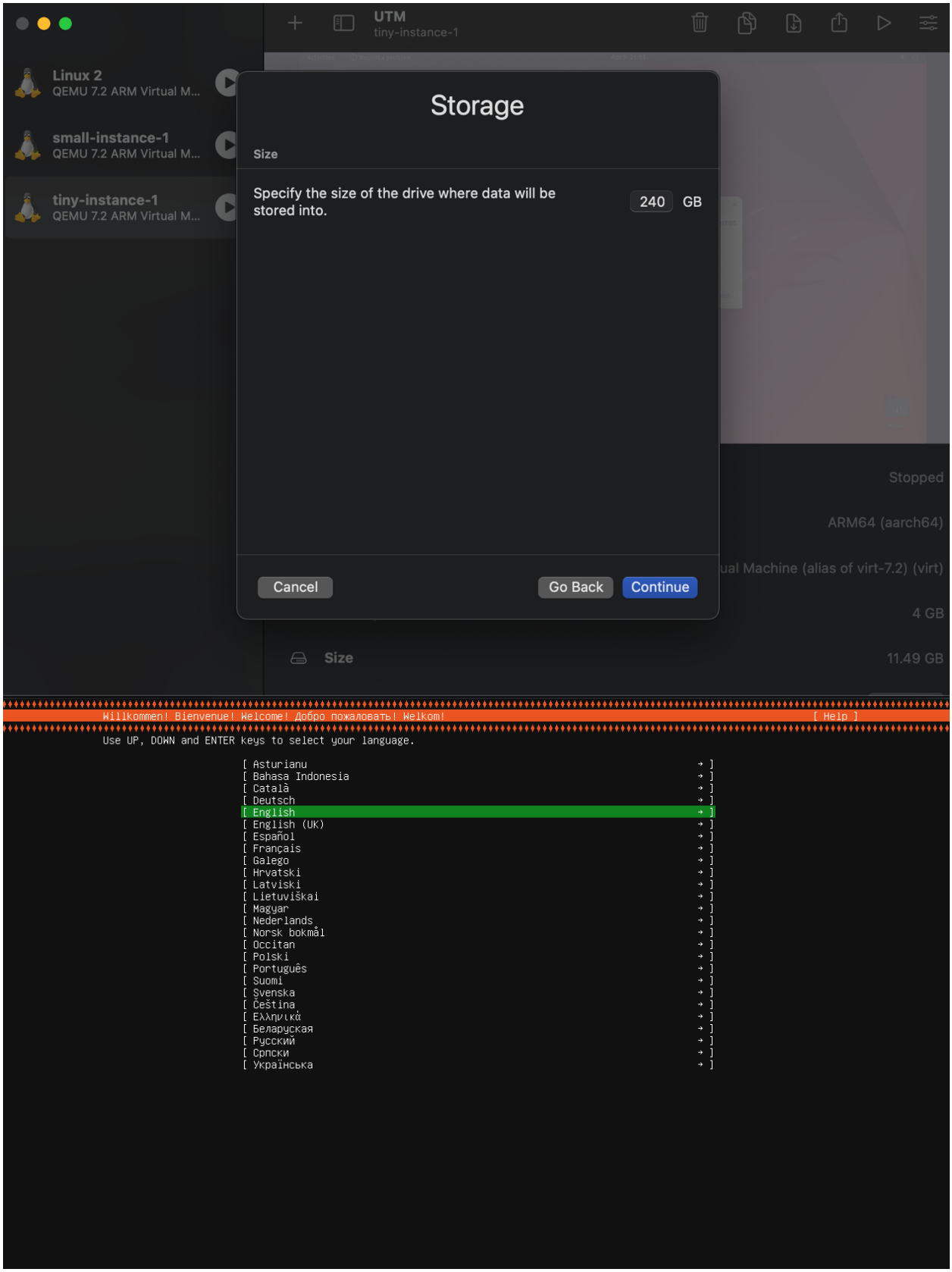
Below is the VM setup for just big instance, all the other instance were created similarly



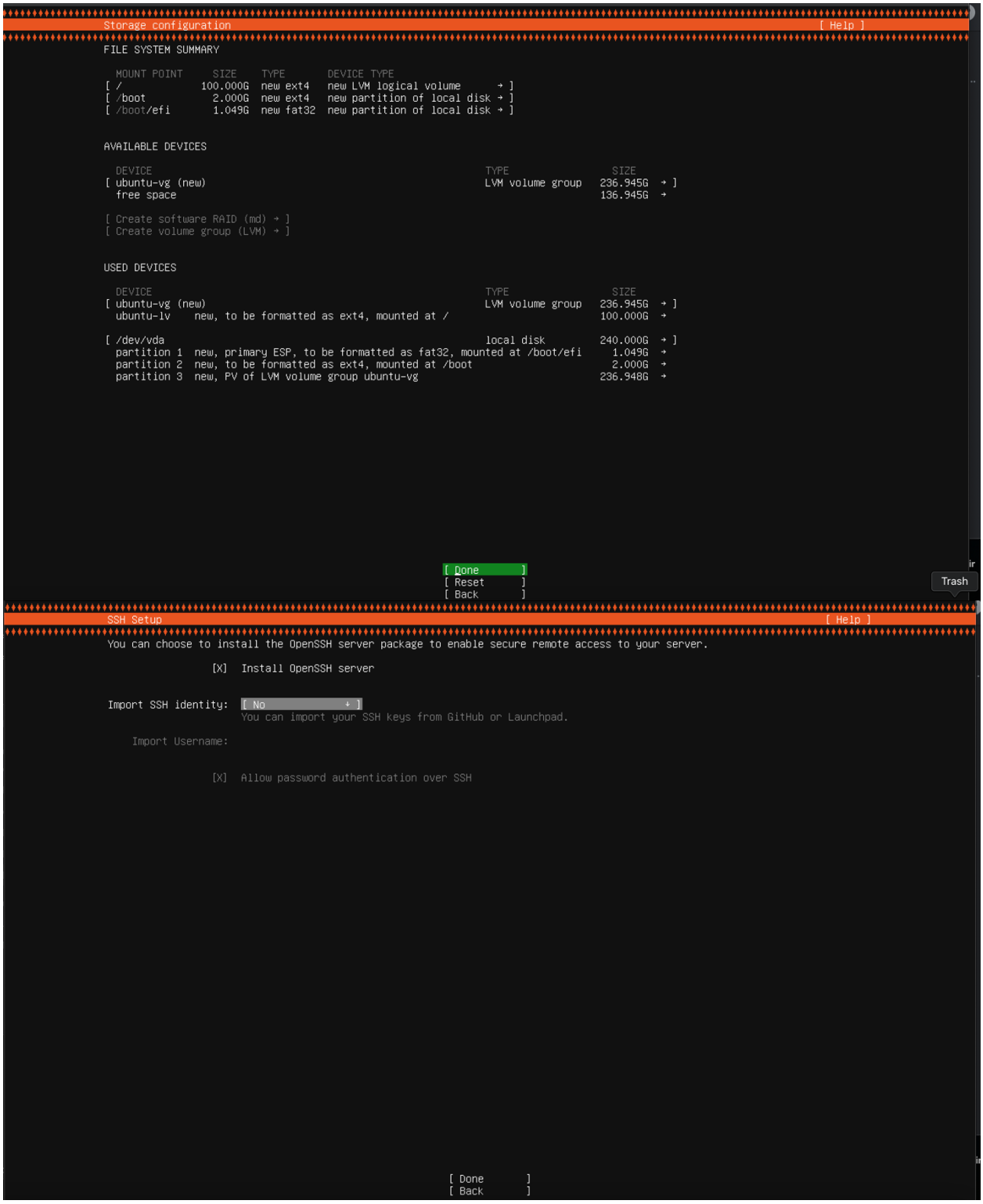

UTM
tiny-instance-1
Linux 2
QEMU 7.2 ARM Virtual M...
small-instance-1
QEMU 7.2 ARM Virtual M...
tiny-instance-1
QEMU 7.2 ARM Virtual M...
Linux
Virtualization Engine
Use Apple Virtualization
Apple Virtualization is experimental and only for advanced use cases. Leave unchecked to use QEMU, which is recommended.
Boot Image Type
Boot from kernel image
Ubuntu Install Guide
Boot ISO Image
ubuntu-22.04.4-live-server-arm64.iso
Clear
Browse...
Cancel
Go Back
Continue
Stopped
ARM64 (aarch64)
ual Machine (alias of virt-7.2) (virt)
4 GB
Size
11.49 GB

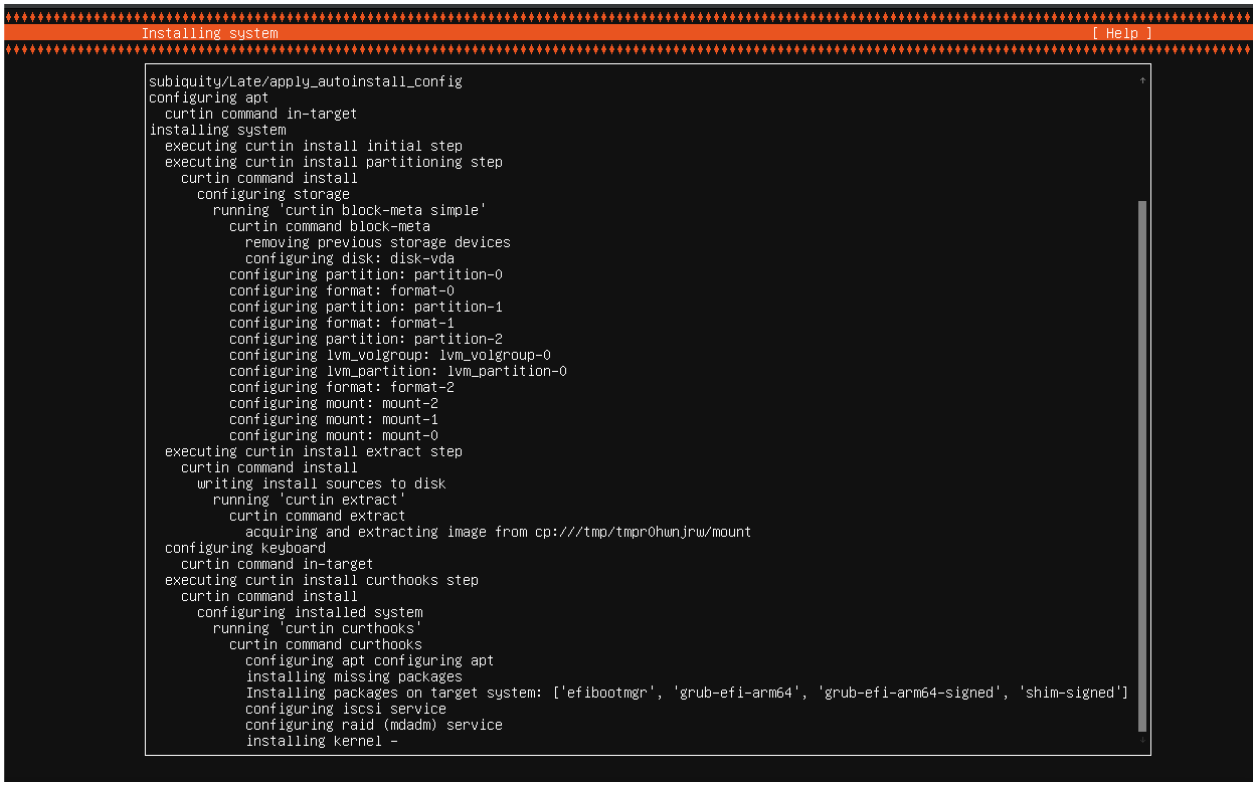


Note that the memory size is not 24GB it is much smaller. This is due to some specification limit on my UTM application. I use the correct amount in my chameleon instance when initializing VM using lxd.


UTM
tiny-instance-1
Linux 2
QEMU 7.2 ARM Virtual M...
small-instance-1
QEMU 7.2 ARM Virtual M...
tiny-instance-1
QEMU 7.2 ARM Virtual M...
Storage
Size
Specify the size of the drive where data will be stored into.
240
GB
Cancel
Go Back
Continue
Stopped
ARM64 (aarch64)
ual Machine (alias of virt-7.2) (virt)
4 GB
Size
11.49 GB
Willkommen! Bienvenue! Welcome! Добро пожаловать! Welkom!
[Help]
Use UP, DOWN and ENTER keys to select your language.
[Asturianu
[Bahasa Indonesia
[Català
[Deutsch
[English
[English (UK)
[Español
[Français
[Galego
[Hrvatski
[Latviski
[Lietuviškai
[Magyar
[Nederlands
[Norsk bokmål
[Occitan
[Polski
[Português
[Suomi
[Svenska
[Čeština
[Ελληνικά
[Беларуская
[Русский
[Српски
[Українська

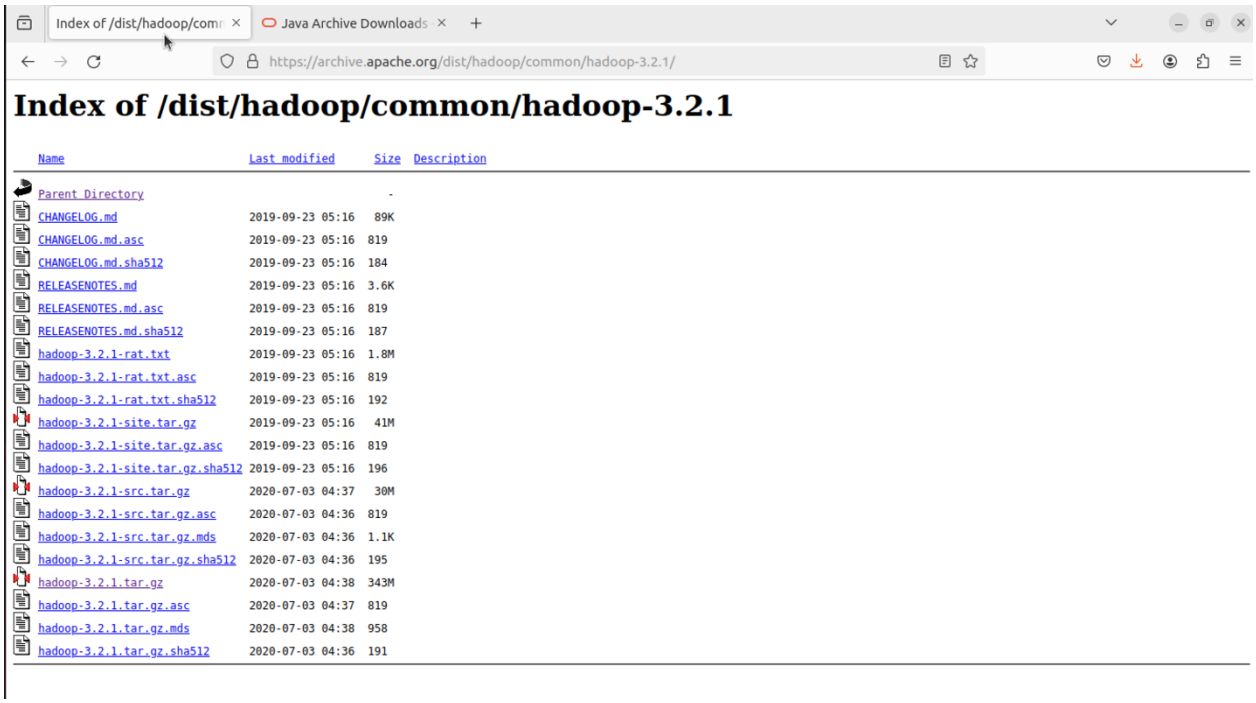

Choose type of install
[Help]
Choose the base for the installation.
(X) Ubuntu Server
The default install contains a curated set of packages that provide a comfortable experience for operating your server.
( ) Ubuntu Server (minimized)
This version has been customized to have a small runtime footprint in environments where humans are not expected to log in.
Additional options
[ ] Search for third-party drivers
This software is subject to license terms included with its documentation. Some is proprietary. Third-party drivers should not be installed on systems that will be used for FIPS or the real-time kernel.
[Done]
[Back]
Guided storage configuration
[Help]
Configure a guided storage layout, or create a custom one:
(X) Use an entire disk
[/dev/vda local disk 240.000G ▾]
[X] Set up this disk as an LVM group
[ ] Encrypt the LVM group with LUKS
Passphrase:
Confirm passphrase:
[ ] Also create a recovery key
The key will be stored as ~/recovery-key.txt in the live system and will be copied to /var/log/installer/ in the target system.
( ) Custom storage layout
[Done]
[Back]

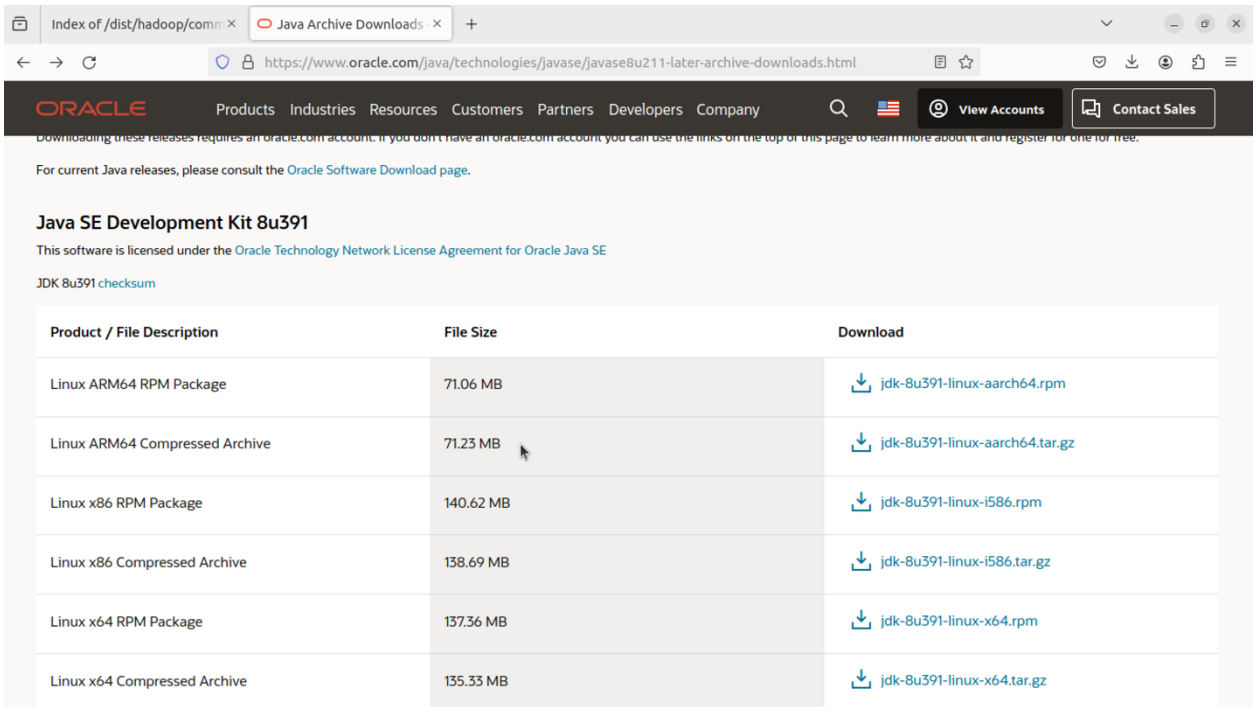

Storage configuration
[Help]
FILE SYSTEM SUMMARY
MOUNT POINT SIZE TYPE DEVICE TYPE
[/ 100.000G new ext4 new LVM logical volume →]
[/boot 2.000G new ext4 new partition of local disk →]
[/boot/efi 1.049G new fat32 new partition of local disk →]
AVAILABLE DEVICES
DEVICE TYPE SIZE
[ubuntu-vg (new) LVM volume group 236.945G →]
free space 136.945G →
[Create software RAID (md) →]
[Create volume group (LVM) →]
USED DEVICES
DEVICE TYPE SIZE
[ubuntu-vg (new) LVM volume group 236.945G →]
ubuntu-lv new, to be formatted as ext4, mounted at / 100.000G →
[/dev/vda local disk 240.000G →]
partition 1 new, primary ESP, to be formatted as fat32, mounted at /boot/efi 1.049G →
partition 2 new, to be formatted as ext4, mounted at /boot 2.000G →
partition 3 new, PV of LVM volume group ubuntu-vg 236.948G →
[Done]
[Reset]
[Back]
Trash
SSH Setup
[Help]
You can choose to install the OpenSSH server package to enable secure remote access to your server.
[X] Install OpenSSH server
Import SSH identity: [No →]
You can import your SSH keys from GitHub or Launchpad.
Import Username:
[X] Allow password authentication over SSH
[Done]
[Back]


Installing system
[Help]
subiquity/Late/apply_autoinstall_config
configuring apt
curtin command in-target
installing system
executing curtin install initial step
executing curtin install partitioning step
curtin command install
configuring storage
running 'curtin block-meta simple'
curtin command block-meta
removing previous storage devices
configuring disk: disk-vda
configuring partition: partition-0
configuring format: format-0
configuring partition: partition-1
configuring format: format-1
configuring partition: partition-2
configuring lvm_volgroup: lvm_volgroup-0
configuring lvm_partition: lvm_partition-0
configuring format: format-2
configuring mount: mount-2
configuring mount: mount-1
configuring mount: mount-0
executing curtin install extract step
curtin command install
writing install sources to disk
running 'curtin extract'
curtin command extract
acquiring and extracting image from cp:///tmp/tmpr0hwnjrw/mount
configuring keyboard
curtin command in-target
executing curtin install curthooks step
curtin command install
configuring installed system
running 'curtin curthooks'
curtin command curthooks
configuring apt configuring apt
installing missing packages
Installing packages on target system: ['efibootmgr', 'grub-efi-arm64', 'grub-efi-arm64-signed', 'shim-signed']
configuring iscsi service
configuring raid (mdadm) service
installing kernel -

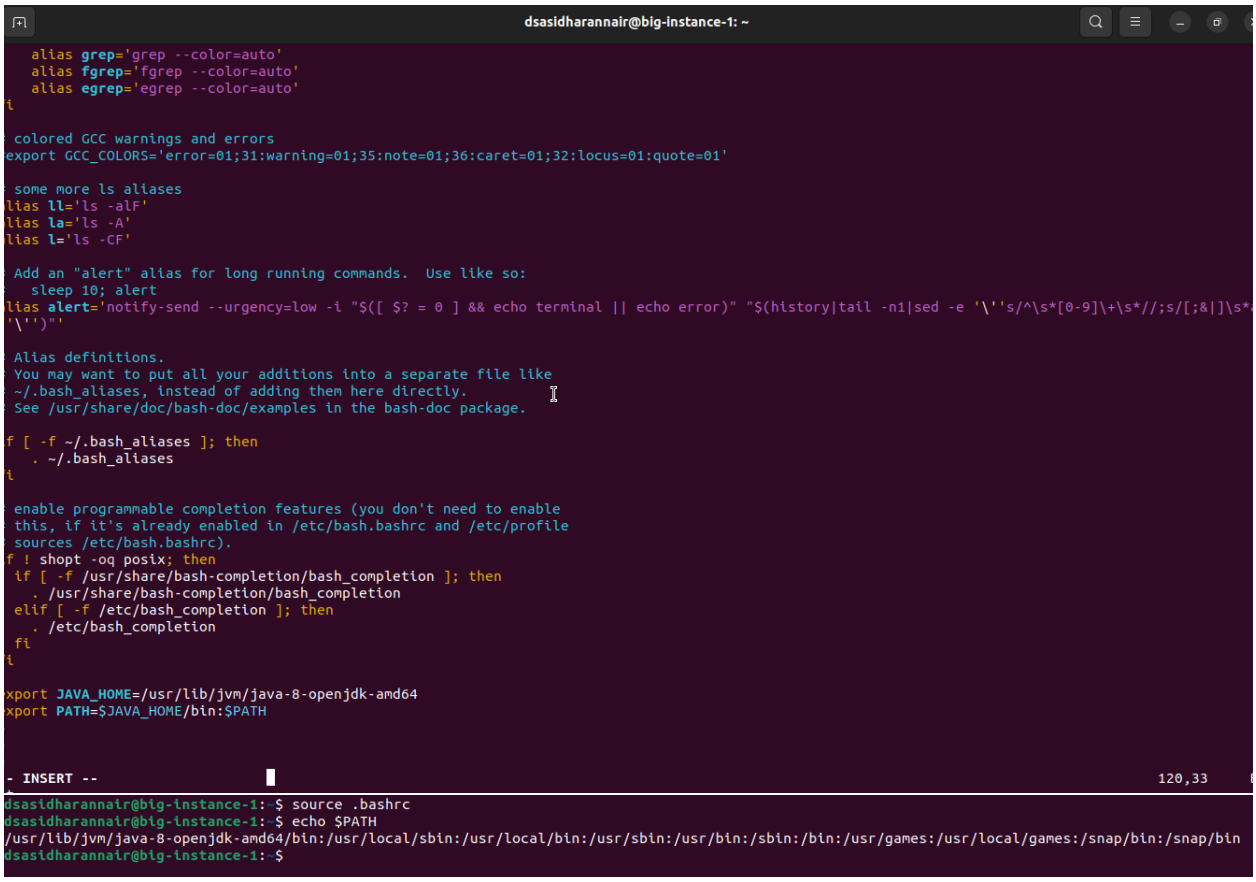

Ubuntu 22.04.4 LTS big-instance-1 tty1
big-instance-1 login: dsasidharannair
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-102-generic aarch64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Wed Apr 10 10:58:44 PM UTC 2024
System load: 0.32421875
Usage of /: 7.0% of 97.87GB
Memory usage: 1%
Swap usage: 0%
Processes: 304
Users logged in: 0
IPv4 address for enp0s1: 192.168.64.7
IPv6 address for enp0s1: fd1a:4243:52d:790d:ccba:28ff:fecd:9a73
Expanded Security Maintenance for Applications is not enabled.
18 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
dsasidharannair@big-instance-1:~$ ls
dsasidharannair@big-instance-1:~$

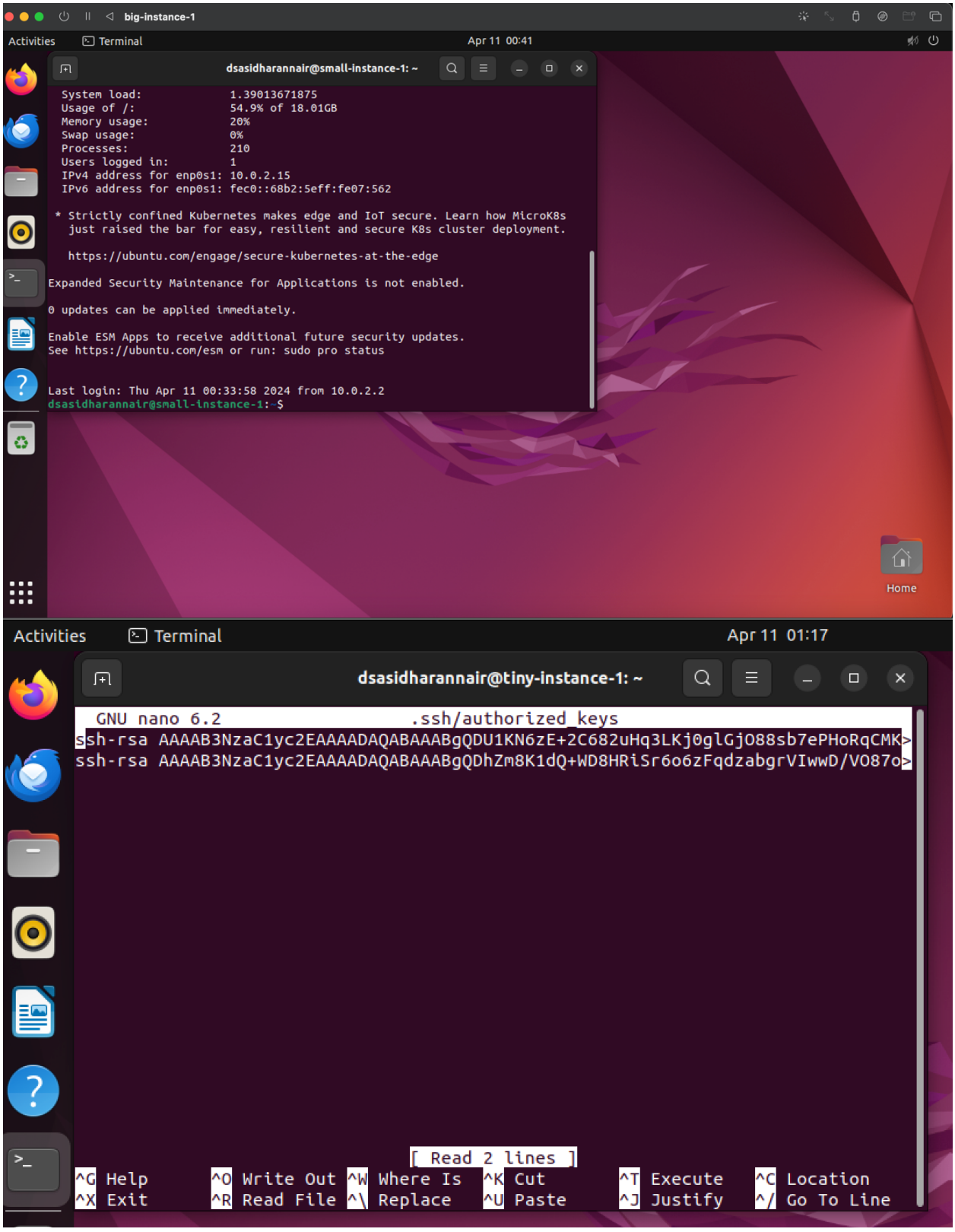
Download, install, configure, and start the HDFS system

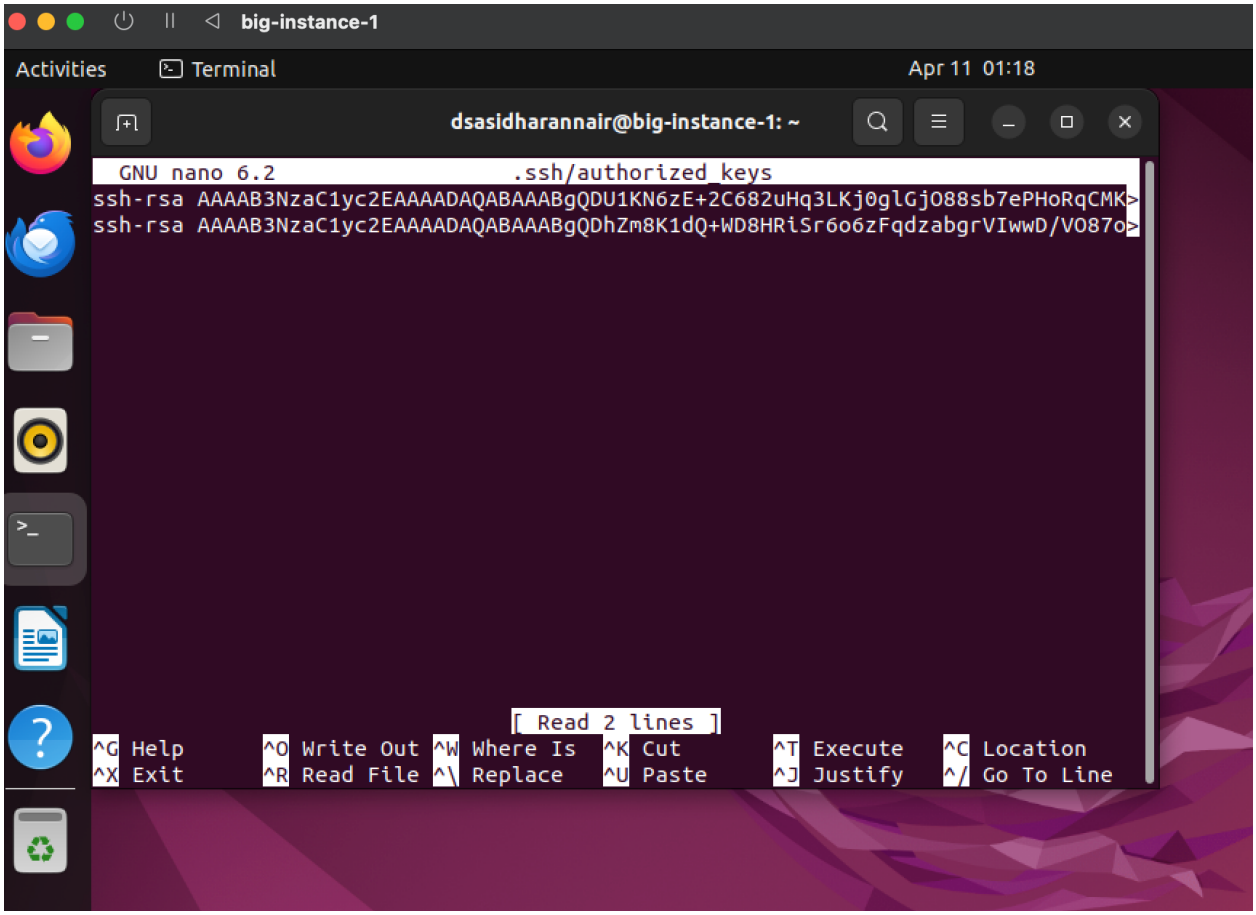




```
dsasidharannair@big-instance-1:~$ sudo apt install openjdk-8-jdk
```




big-instance-1
Activities
Terminal
Apr 11 00:41
dsasidharannair@small-instance-1: ~
System load: 1.39013671875
Usage of /: 54.9% of 18.01GB
Memory usage: 20%
Swap usage: 0%
Processes: 210
Users logged in: 1
IPv4 address for enp0s1: 10.0.2.15
IPv6 address for enp0s1: fec0::68b2:5eff:fe07:562
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Thu Apr 11 00:33:58 2024 from 10.0.2.2
dsasidharannair@small-instance-1:~$
Home
Activities
Terminal
Apr 11 01:17
dsasidharannair@tiny-instance-1: ~
GNU nano 6.2 .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDU1KN6zE+2C682uHq3LKj0glGjO88sb7ePHoRqCMK>
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDhZm8K1dQ+WD8HRiSr6o6zFqdzabgrVIwwD/VO87o>
[Read 2 lines]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line



Setting Up Hadoop specifications – For Large + Tiny

Before Installing the ssh keys when working on a chameleon machine create a new user and then work on that user for the rest of the rest of the project

Log in as root user

Create new user using

sudo adduser dsasidharannair

and also do

sudo apt update && sudo apt install sysstat && sudo apt install openjdk-8-jdk

1. Setup Java by installing jdk using

sudo apt install openjdk-8-jdk

2. Add these two lines to the .bashrc file

export HADOOP_HOME=/home/dsasidharannair/hadoop
export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin

3. Then add this line to the .profile

PATH=/home/dsasidharannair/hadoop/bin:/home/dsasidharannair/hadoop/sbin:$PATH

4. Open the tar file for Hadoop using

tar -xzf hadoop-3.2.1.tar.gz

5. Go into Hadoop-3.2.1/etc/Hadoop and add

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-arm64/jre

For Chameleon instance

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

to mapred-env.sh, hadoop-env.sh, yarn-env.sh

6. Do this for all the VMS in the cluster

7. When this is done, we need to now modify the Hadoop workload specifications (i.e how work is distributed within the Hadoop cluster)
8. Then go into core-site.xml and add the below, setting tiny-instance as the master node

```
<configuration>
      <property>
         <name> fs.defaultFS</name>
         <value>hdfs://tiny-instance-1:9000</value>
      </property>
</configuration>
```

9. Then go into yarn-site.xml and add the below, which sets the resource manager as the tiny instance. Note: The memory management if different for this cluster and the 6 small instance cluster.

```
<configuration>

<!-- Site specific YARN configuration properties -->
   <property>
         <name>yarn.acl.enable</name>
         <value>0</value>
```

```
  </property>

  <property>
      <name>yarn.resourcemanager.hostname</name>
      <value>192.168.64.9</value>
  </property>

  <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

10. Then go into hdfs-site.xml and add the below, which creates files for collecting meta data.

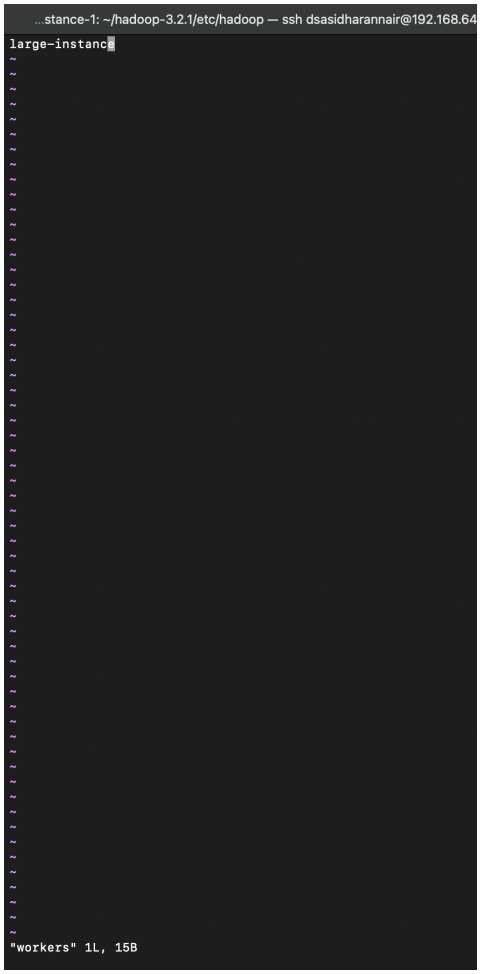
```
<configuration>
  <property>
      <name>dfs.namenode.name.dir</name>
      <value>/home/dsasidharannair/hadoop/data/nameNode</value>
  </property>

  <property>
      <name>dfs.datanode.data.dir</name>
      <value>/home/dsasidharannair/hadoop/data/dataNode</value>
  </property>

  <property>
      <name>dfs.replication</name>
      <value>1</value>
  </property>
</configuration>
```

11. Then go into mapred-site.xml and add the below,

```
<configuration>
  <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
  </property>
  <property>
      <name>yarn.app.mapreduce.am.env</name>
      <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
  <property>
      <name>mapreduce.map.env</name>
```

```
        <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
    </property>
    <property>
        <name>mapreduce.reduce.env</name>
        <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
    </property>
</configuration>
```

12. Now only for the worker file, edit the the workers file and add all the hostnames, This is how my workers file looks like



13. Also go to sudo nano /etc/hosts to add the hostnames of the other nodes. An example is below. Do this step for all the nodes

```
127.0.0.1 localhost
192.168.64.8 large-instance
192.168.64.9 tiny-instance-1

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

14. Then do the below command the copy all the Hadoop config files to the slave nodes,

scp ~/hadoop/etc/hadoop/* large-instance:/home/dsasidharannair/hadoop/etc/hadoop/

15. Now go back to the primary "hadoop-3.2.1" directory. And run

hdfs namenode -format

start-dfs.sh
start-yarn.sh

16. You can run jps on all your nodes to see if it worked

yarn node -list

Setting Up Spark specifications – For Large + Tiny

1. Download the spark 3 .tar file from the spark official website.

```
[dsasidharannair@tiny-instance-1:~$ ls
 alice.txt  data  hadoop  hadoop-3.2.1.tar.gz  spark  spark-3.1.3-bin-hadoop3.2.tgz
```

2. Execute the below command to open the .tar file

tar -xvf spark-3.1.3-bin-hadoop3.2.tgz
mv spark-3.1.3-bin-hadoop3.2.tgz spark

3. Edit the .profile file in the main directory and add these lines

PATH=/home/dsasidharannair/spark/bin:$PATH
export HADOOP_CONF_DIR=/home/dsasidharannair/hadoop/etc/hadoop
export SPARK_HOME=/home/dsasidharannair/spark

export
LD_LIBRARY_PATH=/home/dsasidharannair/hadoop/lib/native:$LD_LIBRARY_PATH

4. Restart the session by logging out and logging in again

5. Implement this line to rename the sparks configs file

mv $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-defaults.conf

6. Edit the spark-defaults.conf file in the config directory of spark

vi $SPARK_HOME/conf/spark-defaults.conf

and add,

spark.master yarn

7. The spark setup is now complete, now to record the program logs, add these lines to the above spark-defaults.conf

spark.eventLog.enabled true
spark.eventLog.dir hdfs://tiny-instance-1:9000/spark-logs
spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.logDirectory hdfs://tiny-instance-1:9000/spark-logs
spark.history.fs.update.interval 10s
spark.history.ui.port 18080

8. Create a log directory in hdfs of Hadoop

hdfs dfs -mkdir /spark-logs

9. The run the history server that records the logs,

$SPARK_HOME/sbin/start-history-server.sh

```
[dsasidharannair@tiny-instance-1:~$ yarn node -list
2024-04-20 03:48:59,606 INFO client.RMProxy: Connecting to ResourceManager at /10.10.192.226:8032
Total Nodes:1
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
large-instance:38449            RUNNING large-instance:8042                                  0
dsasidharannair@tiny-instance-1:~$
```

```
large-instance:38449              RUNNING large-instance:8042
[dsasidharannair@tiny-instance-1:~$ hdfs dfsadmin -report
Configured Capacity: 232403804160 (216.44 GB)
Present Capacity: 227962175488 (212.31 GB)
DFS Remaining: 227759714304 (212.12 GB)
DFS Used: 202461184 (193.08 MB)
DFS Used%: 0.09%
Replicated Blocks:
        Under replicated blocks: 8
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 8
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
        Missing block groups: 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (1):

Name: 10.10.192.50:9866 (large-instance)
Hostname: large-instance
Decommission Status : Normal
Configured Capacity: 232403804160 (216.44 GB)
DFS Used: 202461184 (193.08 MB)
Non DFS Used: 4424851456 (4.12 GB)
DFS Remaining: 227759714304 (212.12 GB)
DFS Used%: 0.09%
DFS Remaining%: 98.00%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat Apr 20 03:49:46 UTC 2024
Last Block Report: Sat Apr 20 01:37:42 UTC 2024
Num of Blocks: 83
```

Setting Up Hadoop specifications – For 6 Small + Tiny

Before Installing the ssh keys when working on a chameleon machine create a new user and then work on that user for the rest of the rest of the project

Log in as root user

Create new user using

sudo adduser dsasidharannair

and also do

sudo apt update && sudo apt install sysstat && sudo apt install openjdk-8-jdk

1. Setup Java by installing jdk using

sudo apt install openjdk-8-jdk

2. Add these two lines to the .bashrc file

export HADOOP_HOME=/home/dsasidharannair/hadoop
export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin

3. Then add this line to the .profile

PATH=/home/dsasidharannair/hadoop/bin:/home/dsasidharannair/hadoop/sbin:$PATH

4. Open the tar file for Hadoop using

tar -xzf hadoop-3.2.1.tar.gz

5. Go into Hadoop-3.2.1/etc/Hadoop and add

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-arm64/jre

For Chameleon instance

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

to mapred-env.sh, hadoop-env.sh, yarn-env.sh

6. Do this for all the VMS in the cluster

7. When this is done, we need to now modify the Hadoop workload specifications (i.e how work is distributed within the Hadoop cluster)
8. Then go into core-site.xml and add the below, setting tiny-instance as the master node

```
<configuration>
    <property>
      <name> fs.defaultFS</name>
```

```
        <value>hdfs://tiny-instance-2:9000</value>
      </property>
</configuration>
```

9. Then go into yarn-site.xml and add the below, which sets the resource manager as the tiny instance. Note: The memory management if different for this cluster and the 6 small instance cluster.

```
<configuration>

<!-- Site specific YARN configuration properties -->
  <property>
        <name>yarn.acl.enable</name>
        <value>0</value>
  </property>

  <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>192.168.64.4</value>
  </property>

  <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

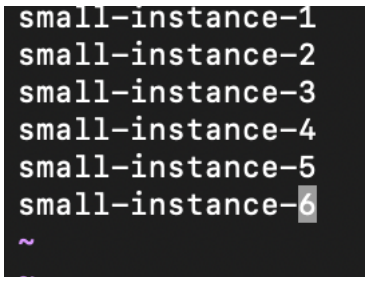
10. Then go into hdfs-site.xml and add the below, which creates files for collecting meta data.

```
<configuration>
  <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/dsasidharannair/hadoop/data/nameNode</value>
  </property>

  <property>
        <name>dfs.datanode.data.dir</name>
        <value>/home/dsasidharannair/hadoop/data/dataNode</value>
  </property>

  <property>
        <name>dfs.replication</name>
        <value>1</value>
  </property>
</configuration>
```

11. Then go into mapred-site.xml and add the below,

```
<configuration>
  <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
  </property>
  <property>
        <name>yarn.app.mapreduce.am.env</name>
        <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
  <property>
        <name>mapreduce.map.env</name>
        <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
  <property>
        <name>mapreduce.reduce.env</name>
        <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>
</configuration>
```

12. Now only for the worker file, edit the the workers file and add all the hostnames,
This is how my workers file looks like



13. Also go to sudo nano /etc/hosts to add the hostnames of the other nodes. An example is below. Do this step for all the nodes

```
  GNU nano 6.2
127.0.0.1 localhost
10.10.192.146 small-instance-1
10.10.192.241 small-instance-2
10.10.192.187 small-instance-3
10.10.192.210 small-instance-4
10.10.192.57 small-instance-5
10.10.192.222 small-instance-6
10.10.192.141 tiny-instance-2

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

14. Then do the below command the copy all the Hadoop config files to the slave nodes,

scp ~/hadoop/etc/hadoop/* small-instance-1:/home/dsasidharannair/hadoop/etc/hadoop/

scp ~/hadoop/etc/hadoop/* small-instance-2:/home/dsasidharannair/hadoop/etc/hadoop/

scp ~/hadoop/etc/hadoop/* small-instance-3:/home/dsasidharannair/hadoop/etc/hadoop/

scp ~/hadoop/etc/hadoop/* small-instance-4:/home/dsasidharannair/hadoop/etc/hadoop/

scp ~/hadoop/etc/hadoop/* small-instance-5:/home/dsasidharannair/hadoop/etc/hadoop/

scp ~/hadoop/etc/hadoop/* small-instance-6:/home/dsasidharannair/hadoop/etc/hadoop/

15. Now go back to the primary “hadoop-3.2.1” directory. And run

hdfs namenode -format

start-dfs.sh
start-yarn.sh

16. You can run jps on all your nodes to see if it worked

yarn node -list

```
[dsasidharannair@tiny-instance-2:~$ yarn node -list
2024-04-19 23:43:15,278 INFO client.RMProxy: Connecting to ResourceManager at /10.10.192.141:8032
Total Nodes:6
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
small-instance-3:46129          RUNNING small-instance-3:8042                             0
small-instance-1:43055          RUNNING small-instance-1:8042                             0
small-instance-5:35819          RUNNING small-instance-5:8042                             0
small-instance-4:44979          RUNNING small-instance-4:8042                             0
small-instance-6:41947          RUNNING small-instance-6:8042                             0
small-instance-2:33841          RUNNING small-instance-2:8042                             0
```

```
dsasidharannair@tiny-instance-2:~$ hdfs dfsadmin -report
Configured Capacity: 231426220032 (215.53 GB)
Present Capacity: 209223086080 (194.85 GB)
DFS Remaining: 209222938624 (194.85 GB)
DFS Used: 147456 (144 KB)
DFS Used%: 0.00%
Replicated Blocks:
        Under replicated blocks: 0
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
        Missing block groups: 0
        Low redundancy blocks with highest priority to recover: 0
        Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (6):

Name: 10.10.192.146:9866 (small-instance-1)
Hostname: small-instance-1
Decommission Status : Normal
Configured Capacity: 38571036672 (35.92 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 3632869376 (3.38 GB)
DFS Remaining: 34921365504 (32.52 GB)
DFS Used%: 0.00%
DFS Remaining%: 90.54%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Fri Apr 19 23:45:10 UTC 2024
Last Block Report: Fri Apr 19 23:41:31 UTC 2024
Num of Blocks: 0


Name: 10.10.192.187:9866 (small-instance-3)
Hostname: small-instance-3
Decommission Status : Normal
Configured Capacity: 38571036672 (35.92 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 3677597696 (3.43 GB)
DFS Remaining: 34876637184 (32.48 GB)
DFS Used%: 0.00%
DFS Remaining%: 90.42%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Fri Apr 19 23:45:10 UTC 2024
Last Block Report: Fri Apr 19 23:41:31 UTC 2024
Num of Blocks: 0
```

Setting Up Spark specifications – For 6 Small + Tiny

1. Download the spark 3 .tar file from the spark official website.

2. Execute the below command to open the .tar file

tar -xvf spark-3.1.3-bin-hadoop3.2.tgz

mv spark-3.1.3-bin-hadoop3.2.tgz spark

3. Edit the .profile file in the main directory and add these lines

```
PATH=/home/dsasidharannair/spark/bin:$PATH
export HADOOP_CONF_DIR=/home/dsasidharannair/hadoop/etc/hadoop
export SPARK_HOME=/home/dsasidharannair/spark
export
LD_LIBRARY_PATH=/home/dsasidharannair/hadoop/lib/native:$LD_LIBRARY_PATH
```

4. Restart the session by logging out and logging in again

5. Implement this line to rename the sparks configs file

```
mv $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-
defaults.conf
```

6. Edit the spark-defaults.conf file in the config directory of spark

```
vi $SPARK_HOME/conf/spark-defaults.conf
```

and add,

```
spark.master    yarn
```

7. The spark setup is now complete, now to record the program logs, add these lines to the above spark-defaults.conf

```
spark.eventLog.enabled  true
spark.eventLog.dir hdfs://tiny-instance-2:9000/spark-logs
spark.history.provider          org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.logDirectory    hdfs://tiny-instance-2:9000/spark-logs
spark.history.fs.update.interval  10s
spark.history.ui.port           18080
```

8. Create a log directory in hdfs of Hadoop

```
hdfs dfs -mkdir /spark-logs
```

9. The run the history server that records the logs,

```
$SPARK_HOME/sbin/start-history-server.sh
```

Chameleon Instance VMS Running with IPV4

```
cc@dsasidharan-instance:~$ sudo lxc list
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
|       NAME       |  STATE  |          IPV4        |                     IPV6                      |      TYPE       | SNAPSHOTS |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| large-instance   | RUNNING | 10.10.192.50 (enp5s0)  | fd42:c13:b5d0:9a42:216:3eff:fe34:d4f4 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-1 | RUNNING | 10.10.192.146 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fe81:1be5 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-2 | RUNNING | 10.10.192.241 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fedf:8b4 (enp5s0)  | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-3 | RUNNING | 10.10.192.187 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fe20:3e96 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-4 | RUNNING | 10.10.192.210 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fe97:a9ff (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-5 | RUNNING | 10.10.192.57 (enp5s0)  | fd42:c13:b5d0:9a42:216:3eff:fe27:3572 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| small-instance-6 | RUNNING | 10.10.192.222 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fe92:a453 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| tiny-instance-1  | RUNNING | 10.10.192.226 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fe39:4c82 (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
| tiny-instance-2  | RUNNING | 10.10.192.141 (enp5s0) | fd42:c13:b5d0:9a42:216:3eff:fef5:485c (enp5s0) | VIRTUAL-MACHINE | 0         |
+------------------+---------+----------------------+-----------------------------------------------+-----------------+-----------+
```

Results Table

Experiment	hashgen	vault	Hadoop Sort	Spark Sort
1 small.instance, 16GB dataset, 2GB RAM	506.72	165.06	N/A	N/A
1 small.instance, 32GB dataset, 2GB RAM	N/A	152.65	N/A	N/A
1 small.instance, 64GB dataset, 2GB RAM	N/A	N/A	N/A	N/A
1 large.instance, 16GB dataset, 16GB RAM	497.86	137.87559818 4	1875.5	
1 large.instance, 32GB dataset, 16GB RAM	1618.677	309.77	2991.22	
1 large.instance, 64GB dataset, 16GB RAM	1919.334	667.89		
6 small.instances, 16GB dataset	N/A	N/A		
6 small.instances, 32GB dataset	N/A	N/A		
6 small.instances, 64GB dataset	N/A	N/A		

MapReduce Sort

My map reduce program makes use of the original map reduce format that is being used in the sample word count program. The unsorted data for input to the program is generated by create_files.c which takes one argument to make sure the nonce being generated are not the same. Each run of create_files generates 1GB of data which is put in hdfs storage. This program is repeatedly executed with each running appending to the previous file within

hdfs till we reach the workload we wish to process. My map reduce program then reads this data using the custom input reader class that I defined which reads chunks of data into a buffer and then passes each record individually to the mapper when called. This greatly reduced the number of I/O operations and allows me to process the binary file into something the map reduce program can take as input. I use the bytesWritable class which is supposed to represent a byte aray making it perfect to represent a record. I then extract the hash from this record and set it as the key in the mapper function. The Hadoop program then sort the program based on the key(hash) and passes it to the reducer which simply writes the entire record to the context as the value of the key value pair. The custom output class I defined also help me write large chunks of binary data to a file using a buffer minimizing the number of I/O operations. I used 16 reducers for this program with each one using 1GB ram with a total of 16GB ram.

MapReduce verify

SparkSort

My spark sort program simply uses sparks built in functionality to read from a binary file. Place it in a RDD array. This RDD array is then mapped to key value pairs similar to the map reduce program. This array is then sorted using the sort by class using the first 10 hash values. I planed to use my custom output class to write to a binary file but I was unable to implement that

Questions to answer

Note that you can set the memory limit to be 2GB for the small.instance and 16GB for the large.instance. What conclusions can you draw?

When setting up Hadoop and spark we set the memory limit to be 2G or 16GB which can be noticed to be lesser than the total memory available to a single node. This is done to improve fault tolerance by allowing more instances to run concurrently while not consuming the entire memory space. This is also an useful tactic to reserve resources for the rest of the system operations or other operational overheads

Which seems to be best at 1 node scale (1 large.instance)?

Large instance beats small instance 10 times out of 10 due to its higher number of cores, higher memory and higher disk space. This is crucial for reducing latency of batch processing tasks in Hadoop and handling larger datasets in memory.

Is there a difference between 1 small.instance and 1 large.instance?

As a large instance has more cores and memory that is available to it, it can handle huge amounts of data more efficiently than a small instance. This would especially be beneficial in spark where in memory computing is highly utilized.

How about 6 nodes (6 small.instance)?

Even though 6 small instances match the large instance in terms of cores, memory and disk space. It is likely to introduce a lot of overhead and network latency due managing multiple nodes simultaneously. Since data is also stored in a distributed manner it would be quite difficult to effectively allocate resources to ensure every node does the same work. This can hugely degrade the performance the 6-node cluster especially in spark as it highly depends on in memory computation which would become very hard to achieve as the data is distributed across the nodes.

What speedup do you achieve with strong scaling between 1 to 6 nodes?

In a strong scaling scenario, where you increase the number of nodes while keeping the workload of the system constant, the performance is not likely to increase as much as 6 times which is what one would think would happen. The performance may not even increase by 2. This is because when adding more nodes there might be more delays in processing the workload due to network delays and the overhead of managing multiple nodes. Even though there might be a small speedup, it is not something that would be worth allocating a lot of resources into.

What speedup do you achieve with weak scaling between 1 to 6 nodes?

In a week scaling scenario, where increase the number of nodes while also increasing the workload, both Hadoop and Spark would scale pretty well as we would be processing a larger amount of data each time, we add a node to the cluster which makes use of the extra memory, processing power and disk space offered by the new node. The node managing overhead, and network delays would be insignificant compared to the speedup that would be achieved.

How many small.instances do you need with Hadoop to achieve the same level of performance as your hashgen or vault programs?

As I was unable to finish running my tests for the 6 small instance, I cannot say conclusively how much small instances would be needed for this case. But it would be fair to assume that at least 50 small instances would be needed to match the performance of the vault program. This way there would be a much higher number of cores and memory that would make using a 50-node cluster beneficially in spite of the overheads involved when managing it.

How about how many small.instances do you need with Spark to achieve the same level of performance as you did with your hashgen/vault?

Considering Spark's great in-memory processing efficiency and speed at handling big datasets, estimating the precise number of tiny instances needed to match the performance of a highly optimized program like vault would be difficult. But it would be safe to assume that it would take a smaller number of nodes than what a Hadoop cluster would take. So that would be around 25 nodes to surpass the speed of the vault program.

Can you predict which would be best if you had 100 small.instances? How about 1000?

Spark and Hadoop might benefit greatly from huge parallelism in the 100 small instance setup that would greatly simplify processing large amounts of data such as 64GB surpassing vaults performance. Although there would be latency issues and overhead these are likely to be overshadowed by the huge performance gain that we would get.

Although the 1000 small instance might seem quite obviously the better choice. There are still a lot of factors to consider in determining its performance for larger workloads. Most probably the huge processing power and memory might end up being overkill for the 64GB workload. Furthermore, it would be very difficult for the name node with the processing power like the tiny instance to manage the working of 1000 worker nodes. The overhead and the network delay would be too much for the system to perform efficiently.