Chapter 47.

November 2020

Welcome to the November2020 edition of DataStax Developer's Notebook (DDN). This month we answer the following question(s);

> My company is finally going cloud. We wish to run performance tests and more for both virtual machine hosting, and then also containers, Kubernetes. We want to see performance implications and also make ready to update our run-book. Can you help ?
>
> *Excellent question ! We've expertly done each; virtualization, and containers. We'll begin a series of articles, in response to this/your question. First, here, we'll detail virtualization. We'll share a number of techniques we use when automating tests and similar when using virtual machines. All of this work will be done on GCP. After this article, on virtualization, we'll move to containers; a series of articles with first an overview, and then a number of recipes when on Kubernetes.*

Software versions

The primary DataStax software component used in this edition of DDN is DataStax Enterprise (DSE), currently release 6.8.*, or DataStax Astra (Apache Cassandra version 4.0.0.*), as required. All of the steps outlined below can be run on one laptop with 16 GB of RAM, or if you prefer, run these steps on Amazon Web Services (AWS), Microsoft Azure, or similar, to allow yourself a bit more resource.

For isolation and (simplicity), we develop and test all systems inside virtual machines using a hypervisor (Oracle Virtual Box, VMWare Fusion version 8.5, or similar). The guest operating system we use is Ubuntu Desktop version 18.04, 64 bit.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

47.1 Terms and core concepts

As stated above, ultimately the end goal is to become proficient using Apache Cassandra when using virtualization, and then containers. When we say virtualization, we mean;

- Operating Apache Cassandra inside virtual machines (VMs), atop an IaaS provider of some sort; aka, atop a level-1 or level-2 hypervisor.
- Treatment of this topic could obviously take weeks, so we'll complete many of the steps below in less than optimal, but complete means, meaning; we will use a lot of brute force methods below.
- Running these VMs with locally attached storage, and using solid state drives, is still not as fast (probably) as using a bare metal system.

  But .. you can not argue against the accelerated application development times when using virtualization or containers.

We will detail all of these steps using Google Cloud Platform (GCP), but of course, this all translates to Amazon, Azure, other.

If you are going to use Google

You will need an account on,

```
console.cloud.google.com
```

We will be here long enough, and doing enough things that you will need a billable account. So make one of these.

On Google there are 3 primary, initial choices,

- Compute engine (Google Compute Engine, GCE)
- Application engine
- Kubernetes engine (Google Kubernetes Engine, GKE)

Engine here basically means, a virtual machine (VM). For our work testing Apache Cassandra using VMs, we want the 'compute engine'. Comments;

- The application engine could be viewed as a superset of the compute engine. The application engine will offer to (seed) our VMs with software we would need to host an application (Node, Python, ..), and it will offer to put a load balancer and more in front of our application nodes (VMs).

  We don't want or need any of that when hosting Apache Cassandra.
- The Kubernetes engine is for containers, which we cover in later editions of this article.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

You can use the Web interface to GCE, however, we will be performing many of these operations dozens if not hundreds of times for our eventual testing. As such, we will script what we are doing; E.g., write them in programs for re-use.

Note: The Web interface to GCE will, in many cases, show you the equivalent command you would run when using their CLI, or API interfaces.

- CLI, command line interface, scripting
- API, programmed within; Python, Java, other

So, if you get stuck on some command, consider prototyping the (object) in the Web UI, and then observe the command it would take to create same.

To install the GCP (GCE, and more) CLI, see,

- `https://cloud.google.com/sdk`
- `https://cloud.google.com/sdk/docs/install`

On Ubuntu, at least, the GCP CLI seemed to prefer or require Python/3.

Note: The primary program we use inside the CLI is titled, glcoud.

Generally we will use gcloud to make and manage (server objects). We then use standard Linux commands to affect the contents of these server objects.

'gcloud' is Google proprietary, and each similar vendor has a similar command. E.g., VMware vSphere Tanzu's equivalent command is titled, tkg, and operates in nearly the same manner as gcloud.

Generally, we install gcloud on Ubuntu using these steps,

From,

`https://cloud.google.com/sdk/docs/install#deb`

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list
```

apt-get install apt-transport-https ca-certificates gnupg

curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -

apt-get update && sudo apt-get install google-cloud-sdk

The first step after install is to validate your userid/password against Google Cloud. From the command line,

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
gcloud auth login
# Above opens a browser TAB to complete authentication


gcloud config set project gkeXXXXXXev
# After authenticating, the above sets your default GCP/GKE/other
project (destination) for server objects
```

Creating SSH keys

Now that we are getting ready to have at least one node (VM), we need encryption keys to be able to SSH, SCP, and more with said node. These keys can be made independently of the node, then added to the node as 'metadata'.

Note: In this context, 'metadata' is a keyword, an actual entity added to a node.

Procedures to generate SSH when on GCP:

```
# From,
# https://cloud.google.com/compute/docs/instances/adding-removing-ss
h-keys
# Generating ssh data
gcloud compute instances describe --project gkeZZZZdev --zone
us-central1-a farrell-vm1
# Subsequent re-use of these keys will fail, so remove them
rm -r $HOME/.ssh/*


#  Actual command to generate keys, we use no pass phrase below
gcloud compute config-ssh --remove


# Copy the generated files to a local spot; ease of use, copying,
other
cp $HOME/.ssh/google_compute_engine* .


# Some ofthe Google commands complain about formatting.
# Add  "root: to beginning of *.pub file
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
# And these files should be readable by the current user only
chmod 400 google_compute_engine*


# Adding these keys to the VM
gcloud compute instances add-metadata --zone us-central1-a
farrell-vm1 --project gkZZZv --metadata-from-file
ssh-keys=google_compute_engine.pub


# If we can SSH, then these steps workd
ssh -i google_compute_engine root@34.70.12.163
```

Creating GCP server objects

At this point, we can move to (scripts). E.g., shell script (Bash), so we can execute these steps time over time, as our needs dictate.

Note: Another advantage of scripts ?

The variableness of GCP/GKE.

these platforms are moving (improving) rapidly. (Stuff) that worked yesterday may not work today.

Having these steps in scripts gives us solid documentation that it's not you that's crazy. It's them.

Example 47-1 is the script we call to make nodes (VMs). A code review follows.

Example 47-1 Shell script to make GCP VMs

```
#!/bin/bash



#  .  Assume already authenticated against GCP
#  .  Make (n) VM nodes on GCP
#  .  Copy in all relevant software

GCP_PROJECT=gkCCCCCCCCCev
GCP_ZONE=us-central1-a
   #
DISK_SIZE=1000GB
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#################################################################
#################################################################


[ -r google_compute_engine.pub ] || {
   echo ""
   echo "ERROR: There needs to be a readable file titled
'google_compute_engine.pub'"
   echo "   in the current working directory, which contains the
SSH public keys."
   echo ""
   echo ""
   exit 1
}


clear
   #
echo ""
echo "Calling 'gcloud' to make virtual machines ..."
echo "========================================="
echo ""
sleep 10


#  If we get command line arguments, use that list to make
#  named VMs, else, make just 2 with hard coded names
[ "$#" -eq 0 ]  && m_args="vm1 vm2" || m_args=${@}

#  Hack, Yes. Sidesteps reusing same IP over iterative tests
#
rm -f ${HOME}/.ssh/*
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#  Supresses "yes|no" prompt first time we ssh or other-
#
echo "StrictHostKeyChecking no" > ${HOME}/.ssh/config


#############################################################


#  Loop thru the args
#
for l_vm in ${m_args}
   do

   echo ">>>> Virtual machine: farrell-${l_vm}  in  ${GCP_ZONE}"
   echo ""
   echo "Calling to create "${l_vm}" ..."
   echo "-----------------------------------------"
   echo "-----------------------------------------"

   #  As of 2020-11-10, there were 4 hard disk types avail for
   #  zone "us-central1-a", and only 1 SSD type could be a boot
   #  disk, pd-ssd
   #
   #  local-ssd    us-central1-a              zone    375GB-375GB
   #  pd-balanced  us-central1-a             zone    10GB-65536GB
   #  pd-ssd       us-central1-a             zone    10GB-65536GB
   #  pd-standard  us-central1-a             zone    10GB-65536GB
   #
   #  Lastly; this VM type would only support a 10GB initial disk
   #  size

   gcloud beta compute --project=gkXXXXXXXXXev instances create \
      farrell-${l_vm}  \
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
      --zone=${GCP_ZONE} \
      --machine-type=n2-standard-8 \
      --subnet=default  \
      --network-tier=PREMIUM \
      --maintenance-policy=MIGRATE  \

--service-account=10DDDDDDDD1-compute@developer.gserviceaccount.
com  \

--scopes=https://www.googleapis.com/auth/devstorage.read_only,ht
tps://www.googleapis.com/auth/logging.write,https://www.googleap
is.com/auth/monitoring.write,https://www.googleapis.com/auth/ser
vicecontrol,https://www.googleapis.com/auth/service.management.r
eadonly,https://www.googleapis.com/auth/trace.append  \
      --image=debian-10-buster-v20201112 \
      --image-project=debian-cloud \
      --boot-disk-size=10GB \
      --boot-disk-type=pd-ssd \
      --boot-disk-device-name=farrell-${l_vm}  \
      --no-shielded-secure-boot \
      --shielded-vtpm \
      --shielded-integrity-monitoring  \
      --reservation-affinity=any
         #
   sleep 30

   echo ""
   echo "Calling to add SSH keys "${l_vm}" ..."
   echo "-------------------------------------------"
   echo "-------------------------------------------"
      #
   gcloud compute instances add-metadata --zone ${GCP_ZONE}
farrell-${l_vm} --project ${GCP_PROJECT} --metadata-from-file
ssh-keys=google_compute_engine.pub
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
         #
   sleep 10

   echo ""
   echo "Calling to re-size disk (step 1 of 2) "${l_vm}" ..."
   echo "----------------------------------------"
   echo "----------------------------------------"
      #
   gcloud compute disks resize farrell-${l_vm} -q --size
${DISK_SIZE} --zone ${GCP_ZONE} --project ${GCP_PROJECT}
         #
   sleep 10

   echo ""
   echo "Calling to re-size disk (step 2 of 2) "${l_vm}" ..."
   echo "----------------------------------------"
   echo "----------------------------------------"
      #
   #  Need external IP address
   #
   l_ext_ip=` gcloud compute instances describe farrell-${l_vm} \
      --zone ${GCP_ZONE}  \
     --format='get(networkInterfaces[0].accessConfigs[0].natIP)'
`
         #
   ssh -i google_compute_engine root@${l_ext_ip} "apt -y install
cloud-guest-utils; growpart /dev/sda 1; resize2fs /dev/sda1; df
-Th"
         #
   sleep 10

   #  This block copies given binaries into our VM nodes.
   #
   #  There are more files we need to copy later (metadata of all
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
nodes, other)
   #
   echo ""
   echo "Copying in given binary programs "${l_vm}" ..."
   echo "-----------------------------------------"
   echo "-----------------------------------------"
      #
   rm -f 81_SoftwareToCopyIn/92_IntIP_addresses.txt
   rm -f 81_SoftwareToCopyIn/93_ExtIP_addresses.txt
      #
   for t in `ls 81_SoftwareToCopyIn`
      do
      echo "File: "${t}
      scp -i google_compute_engine 81_SoftwareToCopyIn/${t}
root@${l_ext_ip}:/opt
      done
         #
      ssh -i google_compute_engine root@${l_ext_ip} "chmod 777
/opt/4*; /opt/40_step01_install.sh "
         #
   sleep 10

   echo ""
   echo ""

   done


#############################################################


echo ""
echo "Building local IP address list (will be used by DSE/C*)
..."
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
echo "-----------------------------------------"
echo "-----------------------------------------"
   #
gcloud compute instances list --project ${GCP_PROJECT} | grep -i
"^farrell" | \
   awk '{print $4}' > 81_SoftwareToCopyIn/92_IntIP_addresses.txt
gcloud compute instances list --project ${GCP_PROJECT} | grep -i
"^farrell" | \
   awk '{print $5}' > 81_SoftwareToCopyIn/93_ExtIP_addresses.txt
echo ""
echo ""


echo "Done: (Now, next steps)"
echo ""
echo "   .  vi  ./81_SoftwareToCopyIn/92_IntIP_addresses.txt"
echo "         and"
echo "      vi  ./81_SoftwareToCopyIn/93_ExtIP_addresses.txt"
echo ""
echo "      to remove any IPs you wish to use for purposes other
than C*/DSE."
echo ""
echo "      (Internal IP address in this file are later used to
configure C*/DSE.)"
echo ""
echo "   .  Run  ./\"36 Configure But Not Boot DSE.sh\""
echo ""
echo ""
```

Relative to Example 47-1, the following is offered:

- As stated in the comments, this program makes one or more VMs, and expects that you are already authenticated against GCP.
- In the first block, we set 3 modular variables, used as identifiers, below. This program is written in Bash(C).

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

- The second block begins with a check for the google_compute_enginer.pub file-
  - If this file is not present readable, we throw and error message and exit.
  - This file is used to set the SSH keys for any newly created VMs, below.
  - We clear the screen, and sleep 10 seconds.

    The sleep is a safety valve; did we enter this program by mistake. You can CONTROL-C, before proceeding.
  - After the sleep, we check for command line arguments. If present, these names are used to build specific VM machine names, used in later identification.
  - The **"**`rm -f`**"** of any previous SSH keys is a hack. IF older contents of SSH keys are present, then any new SSH or SCP command will fail.
  - The `StrictHostKeyChecking` line prevents warnings that will be produced because this, our test area, is not likely to have valid CA certificates.
- The third blocks begins with a for loop; a loop through the command line arguments, the names of the virtual machines to create.
  - First, some echoes to the operator.
  - The documentation that follows states there was only 1 bootable disk of type SSD in this GCP region. Oddly, there was a very small initial size limitation on this disk size, and we must manually resize this disk below.
  - And the gcloud command proper to make a net new virtual machine (VM).

    We didn't produce this command. Instead, we used the GCP Web UI to prototype (generate) this command for us.
  - Calls to the GCP CLI are asynchronous, so we follow this call with a sleep 30, to ensure that the VM is in a likely usable state before proceeding.
  - The "`gcloud .. add-metadata`" line puts our manually generated SSH keys into this VM.
  - Resizing the disk to a larger size is a 2 step process; first from a GCP perspective (allow for a larger disk), then inside the VM proper, to add the additional disk space to the Linux root filesystem.

    The VM does not need to rebooted as the result of this work.

    As we need to run commands inside the VM, here we see the first use of `ssh(C)` are the keys we added to the VM metadata.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

- And embedded for loop inside our current/parent loop, here we copy given software into the VM.

  We use a folder titled, 81_SoftwareToCopyIn, and SCP any files located there.

  After the copy, we run a given install/personalization program, of our own design.
- One more piece; all of these VMs will have the internal and external IP addresses of their neighbors. Those files are titled with a numeric 92 and 93, for easier identification. In this loop, we erase those files before file copy.

  Below we will copy these files into the VMs.

Note: We haven't created all of the VMs that need to be created yet. Thus, we can not know their IP addresses until we create this first parent loop first.

- The fourth block begins with an echo, **"**`Building local IP ..`**"**.

  We don't need to loop here, since gcloud with a text processing command will give us that data.
- We finish with a message of "Done", and suggested next steps.

Note: So this is our (create net new VMs) shell script. Not ready for production, but good enough for trials, created quickly.

And, we made this program in under 10 minutes; quick and effective.

Deleting GCP server objects

Example 47-2 will delete the VMs made above. A code review follows.

Example 47-2 Shell script to delete GCP VMs

```
#!/bin/bash


#  .  Assume already authenticated against GCP
#  .  Delete (n) VM nodes on GCP, any labeled with 'farrell'

GCP_PROJECT=gkXXXXXXXXXev
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
############################################################


#  data will look like,
#
#  $gcloud compute instances list
#
#  NAME                                                ZONE
MACHINE_TYPE    PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP     STATUS
#  farrell-vm1
us-central1-a  e2-standard-8               10.128.0.41
35.184.68.224   RUNNING
#  farrell-vm2
us-central1-a  e2-standard-8               10.128.0.42
35.192.69.238   RUNNING
#  ddac-gcp-24x7-striim-dont-remove-ddac-seed-1
us-east1-b    n1-standard-4             10.8.0.4
TERMINATED
#  instance-oracle-centos
us-east1-b     n1-standard-16              10.142.0.9
35.237.200.11   RUNNING
#  striim-gcp-marketplace-vm-solution-1-vm
us-east1-b    n1-highmem-8              10.142.0.11
TERMINATED
#  ddac-gcp-24x7-striim-dont-remove-ddac-non-seed-node-gg7k
us-east1-d     n1-standard-4               10.8.0.5
35.190.190.142  RUNNING
#  ddac-gcp-24x7-striim-dont-remove-ddac-seed-0
us-east1-d    n1-standard-4             10.8.0.3
TERMINATED
#  oss-201104t072735-cassandra-non-seed-node-6m78
us-west1-b     n1-standard-4               10.8.0.5
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
34.83.98.244    RUNNING
#  oss-201104t072735-cassandra-seed-1
us-west1-c     n1-standard-4                10.8.0.4
34.83.142.102   RUNNING


echo ""
echo "Calling 'gcloud' to delete virtual machines ..."
echo ""
sleep 10

while read line
   do
   l_vm=`echo ${line} | cut -d " " -f1`
   l_zn=`echo ${line} | cut -d " " -f2`

   echo "Found virtual machine: ${l_vm}  in  ${l_zn}"
   echo ""
   echo "Calling to delete ..."
      #
   gcloud compute instances delete ${l_vm} -q --zone ${l_zn}
--project ${GCP_PROJECT}
   echo ""
      #
   sleep 30
      #
   done < <( gcloud compute instances list | grep -i "^farrell" )

echo ""
echo ""
```

Relative to Example 47-2, the following is offered:

- Standard to each of these programs, we begin with a sleep 10, in the event you called this program by accident.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

- Because we are parsing lines of input, we use 'while .. read line' as the loop construct.
- The -q flag to 'gcloud .. delete' suppresses any, 'Are you sure ?' prompting.

Example 47-3 lists any VMs we have in our GCP project. A code review follows.

Example 47-3 Shell script to list VMs

```
#!/bin/bash


#  .  List all nodes

GCP_PROJECT=gkZZZZZZev


################################################################


echo ""
echo "Calling 'gcloud' to report status ..."
echo ""

gcloud compute instances list --project ${GCP_PROJECT}

echo ""
echo ""
```

Relative to Example 47-3, the following is offered:

- 'compute instance list' provides the necessary output.
- The --project flag acts as a filter.

Example 47-4 calls to run a configurations program on each of the VMs. A code review follows.

Example 47-4 Call to copy additional configuration programs to the VM

```
#!/bin/bash
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
# .  Assume already authenticated against GCP
# .  Delete (n) VM nodes on GCP, any labeled with 'farrell'

GCP_PROJECT=gkZZZZZZZZZZv


#################################################################
####


echo ""
echo "At this point, you have all software on all nodes that you"
echo "called to previously create."
echo ""
echo "[ This program ] will:"
echo ""
echo "   .  Push (2) more files out to the nodes;"
echo "        81_SoftwareToCopyIn/92_IntIP_addresses.txt"
echo "        81_SoftwareToCopyIn/93_ExtIP_addresses.txt"
echo ""
echo "      Which nodes get these files ?"
echo "      Any nodes listed in the 93* file."
echo ""
echo "      So, if you have 6 nodes, and only want 5 to operate"
echo "      C*/DSE, delete lines from both the 92* and 93* files."
echo ""
echo "   .  All nodes will be placed in 1 DC. If you wish to have"
echo "      multiple DCs, you must edit the cassandra-topology.properties"
echo "      file on each node after this program completes."
echo ""
echo "   .  After these files are pushed, C*/DSE is not booted."
echo ""
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
echo "      To boot C*/DSE, run file 37*."
echo ""
echo "**  You have 30 seconds to cancel before proceeding."
echo ""
echo ""
sleep 30


#################################################################
####
#################################################################
####


#  Loop thru the external IPs, just file mung
#
for l_ext_ip in `cat ./81_SoftwareToCopyIn/93_ExtIP_addresses.txt`
  do

  echo "Copying and configuring files for host: "${l_ext_ip}
  echo "----------------------------------------"
  scp -i google_compute_engine
81_SoftwareToCopyIn/92_IntIP_addresses.txt root@${l_ext_ip}:/opt
  scp -i google_compute_engine
81_SoftwareToCopyIn/93_ExtIP_addresses.txt root@${l_ext_ip}:/opt
  echo ""
  ssh -i google_compute_engine root@${l_ext_ip}
"/opt/44_step04_massageConfigFiles.sh"
  echo ""

  done

  ssh -i google_compute_engine root@35.232.93.34
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#########################################################################
####


echo ""
echo ""
echo "Done: (Now, next steps)"
echo ""
echo "   .  If you are okay with all nodes being in one C*/DSE DC,"
echo "      proceed to run file 37* in this same folder."
echo ""
echo "      File 37* calls to boot C*/DSE."
echo ""
echo "   .  If you wish multiple DCs, edit the
cassandra-topology.properties"
echo "      on each node. (The topology file, obviously, assigns DCs to
each"
echo "      node. Generally, this file is the same on each node.)"
echo ""
echo ""
```

Relative to Example 47-4, the following is offered:

- This program loops through a list of external IP addresses we generated for any VMs we wish to run Apache Cassandra on.
- We copy in 3 files; the IP list, both internal and external, and the local SSH key file.
- As stated in the program comments, the default (and generated) behavior is to run all Apache Cassandra nodes within 1 Cassandra 'DC'. If you wish another topology, that is do-able, as a manual configuration.

Example 47-5 calls to boot Apache Cassandra. A code review follows.

Example 47-5 Booting Apache Cassandra

```
#!/bin/bash
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#  .  Assume already authenticated against GCP
#  .  Delete (n) VM nodes on GCP, any labeled with 'farrell'

GCP_PROJECT=gke-launcher-dev


################################################################


echo ""
echo "At this point, you are ready to boot C*/DSE."
echo ""
echo "**  You have 30 seconds to cancel before proceeding."
echo ""
echo ""
sleep 30


################################################################
################################################################


#  Loop thru the external IPs, boot DSE
#
for l_ext_ip in `cat
./81_SoftwareToCopyIn/93_ExtIP_addresses.txt`
   do

   echo "Booting DSE for host: "${l_ext_ip}
   echo "----------------------------------------"
   echo ""
   # ssh -i google_compute_engine root@${l_ext_ip} "cd
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
/opt/dse_68/node1/bin; ./dse cassandra -R -k"
   ssh -i google_compute_engine root@${l_ext_ip} "cd
/opt/dse_68/node1/bin; ./cassandra -R"
   echo ""

   done


#################################################################


l_first_int_IP=`head -1
81_SoftwareToCopyIn/92_IntIP_addresses.txt`
l_first_ext_IP=`head -1
81_SoftwareToCopyIn/93_ExtIP_addresses.txt`

echo ""
echo ""
echo "Done: (Now, next steps)"
echo ""
echo "   .  ssh into any box. For example,"
echo ""
echo "         ssh -i google_compute_engine
root@"${l_first_ext_IP}
echo ""
echo "   .  Once there, run a nodetool. For example,"
echo ""
echo "         /opt/dse_68/node1/bin/nodetool status"
echo ""
echo "         /opt/dse_68/node1/bin/cqlsh "${l_first_int_IP}
echo ""
echo "         export PATH=$PATH:/opt/dse_68/node1/bin"
echo ""
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

Relative to Example 47-5, the following is offered:

- Per the comments, this program assumes-
  - That you've already authenticated against GCP.
  - That this is a new install of any VMs, and the software they contain.
- As this program is run from a jump (control) box, we need the external IP addresses to SSH into any target VMs.
- Running as 'root', we boot Apache Cassandra with a 'cassandra -R'.
- And we finish by listing possible next steps.

Example 47-6 is the last of the programs we would operate from our jump box. A code review follows.

Example 47-6 SSH into each node, in turn

```
#!/bin/bash


############################################################


echo ""
echo "SSH into all nodes, then do whatevs."
echo ""
echo "**  You have 10 seconds to cancel before proceeding."
echo ""
echo ""
sleep 10


############################################################
############################################################


#  Loop thru the external IPs, ssh into each
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#
for l_ext_ip in `cat
./81_SoftwareToCopyIn/93_ExtIP_addresses.txt`
   do

   echo "ssh into host: "${l_ext_ip}
   echo "----------------------------------------"
   ssh -i google_compute_engine root@${l_ext_ip}
   echo ""

   done


#################################################################


echo ""
echo ""
```

Relative to Example 47-6, the following is offered:

- Here we loop through the data file containing our external IP addresses.
- The SSH functions because we had previously generated SSH keys, and placed metadata into the VM containing same.

Example 47-7 This program 'installs' a number of programs we want to operate on each node. A code review follows.

Example 47-7 Node software install program

```
#!/usr/bin/bash


#  These file IDs come from my DS Google Drive
#
ID_DSE=17yaZZZZZZZZZZZf0y
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
ID_NB=15SuGGGGGGGGGGGknp
ID_ZEPP=1CK3HHHHHHHHHHHHDKy6V


#############################################################

echo "Install JRE"
apt install default-jre -y &> /dev/null
   #
# echo "Install Python-Pip"
# apt install python-pip -y &> /dev/null
# echo "Install GDown (Google-drive file copy)"
# pip install gdown        &> /dev/null

cd /opt


#  On 2020/11/20, G-Drive was being super flakey
#
# gdown https://drive.google.com/uc?id=${ID_DSE}
# sleep 5
# gdown https://drive.google.com/uc?id=${ID_NB}
# sleep 5
# gdown https://drive.google.com/uc?id=${ID_ZEPP}
# sleep 5
#
#  Just DSE
#
# tar xf dse-6.8.5-bin.tar.gz
# rm -f  dse-6.8.5-bin.tar.gz
#    #
# mkdir dse_68
# mv dse-6.8.5 dse_68/node0
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#  This is the workaround to G-Drive (for now)
#
#  DL DSE
# curl -L
https://downloads.datastax.com/enterprise/dse-6.8.5-bin.tar.gz |
tar xz
# mkdir dse_68
# mv dse-6.8.5 dse_68/node0
#
curl -L
https://downloads.apache.org/cassandra/4.0-beta3/apache-cassandr
a-4.0-beta3-bin.tar.gz | tar xz
mkdir dse_68
mv apache-cassandra-4.0-beta3 dse_68/node0


#  Just nb
#
# chmod 777 nb
# mv nb zzz
# mkdir nb
# mv zzz nb/nb

#  Just Zeppelin
#
# tar xf zeppelin-0.9.0-preview2-bin-all.tgz
# rm -f  zeppelin-0.9.0-preview2-bin-all.tgz
# mv zeppelin-0.9.0-preview2-bin-all zeppelin
```

Relative to Example 47-7, the following is offered:

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

- These Apache Cassandra nodes are being operated on GCP. As such, we were copying many large binaries from our Google Drive account for speed. Like, really good speed.

  But .. .. reliability was bad. the 'gdown' file copy utility would fail, often, over days.

  So we moved back to Curl.
- We were, for a period of time, copying in; DataStax Enterprise, Apache Zeppelin, and more.

  Most of that programming is now disabled via commenting, and we leave only the operating instructions to copy in apache Cassandra.
- curl(ing) to a '`tar xz`' is pretty useful. We know of no better means to curl binaries and have them unpacked, than with this method.
- As a convention, we copy the Cassandra binaries to 'node0', a folder we never use, and then source from that directory over and over, as needed.

Example 47-8 localizes our installed files. A code review follows.

Example 47-8 'localizing' installed files.

```
#!/usr/bin/bash


#  A virgin C* 4.X exists in /opt/dse_68/node0
#
#  Copy that [ distro ] to node1, and edit files as needed in
#  order to boot DSE.
#
#  Because we are playing with replacement/other of data files,
#  all of those go in /opt2 for easy reference.

#  Moved from DSE to C*.
#
#  File 45* is the version of this same file before I made that
#  change.


############################################################
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#############################################################


#  Mung directories and such

rm -fr /opt/dse_68/node1
rm -fr /opt2
   #
mkdir /opt/dse_68/node1
   #
mkdir -p /opt2/data
   #
mkdir -p /opt2/metadata
mkdir -p /opt2/commitlog
mkdir -p /opt2/saved_caches
mkdir -p /opt2/hints
mkdir -p /opt2/cdc_raw
   #
chmod -R 777 /opt/dse_68/node1
chmod -R 777 /opt2
   #
cp -R /opt/dse_68/node0/* /opt/dse_68/node1


#############################################################


#  Variables we will need later

#  To get external IP address;  curl ifconfig.co
#
#  But we need the internal IP address

l_intIpAddr=` ip addr | grep "inet " | grep -v "127\.0\.0\.1" |
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
head -1 | awk '{print $2}' | sed 's/...$//' `

#  This 88* file should include internal IPs for just nodes
#  that are to be part of our DSE cluster
#
l_allIpsInCluster1=` cat /opt/92_IntIP_addresses.txt `
l_allIpsInCluster2=` echo $l_allIpsInCluster1 | sed 's/
/:7000,/g' `


#################################################################


#  cassandra.yaml
#
#  Items we need to change,
#
#     cluster_name: 'Test Cluster'
#     hints_directory: /var/lib/cassandra/hints
#          - /var/lib/cassandra/data
#     metadata_directory: /var/lib/cassandra/metadata
#     commitlog_directory: /var/lib/cassandra/commitlog
#     cdc_raw_directory: /var/lib/cassandra/cdc_raw
#     disk_failure_policy: stop
#     commit_failure_policy: stop
#     saved_caches_directory: /var/lib/cassandra/saved_caches
#               - seeds: "127.0.0.1"
#     listen_address: localhost
#     aggregated_request_timeout_in_ms: 120000
#     endpoint_snitch: com.datastax.bdp.snitch.DseSimpleSnitch
#


echo ""
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
echo "Processing (configuring cassandra.yaml) ..."
echo "----------------------------------------"

cat /opt/dse_68/node1/conf/cassandra.yaml | awk -v
l_intIpAddr=${l_intIpAddr} -v
l_allIpsInCluster=${l_allIpsInCluster2} -F":" '

   {

   #
   #  This block: Set to a single value across all DSE nodes.
   #  ######################################################
   #
   if ($1 == "cluster_name") {
      print("cluster_name: 'my_cluster'")
      print("hints_directory: /opt2/hints")
      print("data_file_directories:")
      print("    - /opt2/data")
      print("commitlog_directory: /opt2/commitlog")
      print("saved_caches_directory: /opt2/saved_caches")
      }
   else if ($1 == "num_tokens") {
      print("num_tokens: 1")
      }
   else if ($1 == "disk_failure_policy") {
      gsub("stop", "die", $0)
      print $0
      }
   else if ($1 == "endpoint_snitch") {
      print("endpoint_snitch: PropertyFileSnitch")
      }
   else if ($1 == "commit_failure_policy") {
      gsub("stop", "die", $0)
      print $0
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
      }
   #
   #  We need these leading spaces, 10 in number.
   #
   else if ($1 == "          - seeds") {
      gsub("127.0.0.1", l_allIpsInCluster, $0)
      print $0
      }
   #
   #  This block: Set specifically for any given node
   #  ###################################################
   #
   else if ($1 == "listen_address") {
      gsub("localhost", l_intIpAddr, $0)
      print $0
      }
   #
   #  Fall through: Everything not munged above
   #
   else
      {
      print $0
      }

   } ' > /tmp/${$}
      #
   mv /tmp/${$} /opt/dse_68/node1/conf/cassandra.yaml


##############################################################
##############################################################


#  So .. .. we are testing 1 DC with 3 nodes, then 2 DCs with
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#  5 nodes each.  Here we'll just punt and set up for all nodes
#  in 1 DC. When we do the 2 DC thing, we'll manually edit this
#  file.
#
#  Sample output that is required,
#
#     192.168.2.200=DC1:RAC1

echo ""
echo "Processing (configuring cassandra-topology.properties):"
echo "---------------------------------------"

#  This file we're starting from is already stepped on, fyi ..
#
cp /opt/91_cassandra-topology.properties
/opt/dse_68/node1/conf/cassandra-topology.properties

for t in ${l_allIpsInCluster1}
   do
   echo ${t}=DC1:RAC1 >>
/opt/dse_68/node1/conf/cassandra-topology.properties
   done


#############################################################


echo ""
echo "Processing (configuring logback.xml):"
echo "---------------------------------------"

cat /opt/dse_68/node1/conf/logback.xml | \
   sed 's/\${cassandra.logdir}/\/opt2/g' > /tmp/${$}
mv /tmp/${$} /opt/dse_68/node1/conf/logback.xml
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
############################################################


# echo ""
# echo "Processing (configuring dse.yaml):"
# echo "-----------------------------------"
#
# cat /opt/dse_68/node1/resources/dse/conf/dse.yaml | awk -F"=" '
#
#    {
#    #
#    #  Setting DSEFS, other
#    #
#    if ($0 == "# dsefs_options:") {
#       print "dsefs_options:";
#       }
#    else if ($0 == "#     keyspace_name: dsefs") {
#       print "    keyspace_name: dsefs";
#       }
#    else if ($0 == "#     work_dir: /var/lib/dsefs") {
#       print "    work_dir: /opt2/dsefs";
#       }
#    else if ($0 == "#     data_directories:") {
#       print "    data_directories:";
#       }
#    else if ($0 == "#         - dir: /var/lib/dsefs/data") {
#       print "         - dir: /opt2/dsefs/data";
#       }
#    else if ($0 == "#           min_free_space: 5368709120") {
#       print "           min_free_space: 1024";
#       }
#    if ($0 == "# alwayson_sql_options:") {
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#       print "alwayson_sql_options:";
#       print "    enabled: true";
#       }
#    else
#       {
#       print $0
#       }
#    } ' > /tmp/${$}
#
# mv /tmp/${$} /opt/dse_68/node1/resources/dse/conf/dse.yaml


################################################################


# echo ""
# echo "Processing (configuring spark-env.sh):"
# echo "--------------------------------------"
#
#
# cat /opt/dse_68/node1/resources/spark/conf/spark-env.sh | awk
-F"=" '
#    {
#    #
#    #  Setting Spark stuff, other
#    #
#    if ($0 == "# export
SPARK_WORKER_DIR=\"/var/lib/spark/worker\"") {
#       print "export SPARK_WORKER_DIR=\"/opt2/spark/worker\"";
#       }
#    else if ($0 == "# export
SPARK_EXECUTOR_DIRS=\"/var/lib/spark/rdd\"") {
#       print "export SPARK_EXECUTOR_DIRS=\"/opt2/spark/rdd\"";
#       }
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

```
#    else if ($0 == "# export
SPARK_WORKER_LOG_DIR=\"/var/log/spark/worker_log\"") {
#       print "export
SPARK_WORKER_LOG_DIR=\"/opt2/spark/worker_log\"";
#       }
#    else if ($0 == "# export
SPARK_MASTER_LOG_DIR=\"/var/log/spark/master\"") {
#       print "export
SPARK_MASTER_LOG_DIR=\"/opt2/spark/master\"";
#       }
#    else if ($0 == "# export
ALWAYSON_SQL_LOG_DIR=\"/var/log/spark/alwayson_sql\"") {
#       print "export
ALWAYSON_SQL_LOG_DIR=\"/opt2/spark/alwayson_sql\"";
#       }
#    else
#       {
#       print $0
#       }
#    } ' > /tmp/${$}
#
# mv /tmp/${$}
/opt/dse_68/node1/resources/spark/conf/spark-env.sh


############################################################


echo "  "
echo "  "
```

Relative to Example 47-8, the following is offered:

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

- This a our largest file, by volume. In effect, we prepare each node to be able to run Apache Cassandra.

  While each Apache Cassandra node is the same in function (Cassandra is a masterless database server), these largely causes the configuration files on each node to have some amount of unique personalization.

- The first block of code prepares a number of folders. We put all of our files under, /opt2, to make them easy to find. (All files are in one location.)

  As stated above, we always source the Cassandra distribution from node0, and place them in node1. node1 is the only folder we personalize.

- The next block gathers our internal IP addresses of the Cassandra nodes. We need these values to be able to instruct the Cassandra nodes how (where) to find one another.

- The we go file by file through the Cassandra configuration-

  - cassandra.yaml, is the largest file here by function.

    There are settings in this file we must change, settings that will be the same on each node. And, there are changes we make here that will be different on each node.

    Each value we need to change is listed at the head of this block, for reference.

    Effectively we do a Linux cat, to an Awk(C) block.

    The Awk identifies given input paragraphs, and changes values as needed.

  - logback.xml sets our Cassandra node, 'message log file' location; an ASCII text file reporting common server events.

  - The remaining blocks are commented out, as they were specific to DSE.

Our last example, Example 47-9, is used to kill a given Cassandra node, for the purposes of testing. A code review follows.

Example 47-9 Killing a Cassandra node.

```
#!/usr/bin/bash


kill -9 `ps -ef | grep
"/usr/lib/jvm/java-11-openjdk-amd64//bin/java
-Ddse.server_process" | grep -v "grep" | awk '{print $2}'`
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

Relative to Example 47-9, the following is offered:

- Here we grep for the Linux process by a DSE server flag.
- And we "kill -9".

47.2 Complete the following

At this point in this document we have detailed a number of scripts (techniques) we use when operating DataStax Enterprise or Apache Cassandra during benchmarks or similar.

Each installation will vary; GCP or other, Apache Cassandra network topology, or other specific operating needs.

You are invited to harvest from the scripts above to complete your own toolkit.

47.3 In this document, we reviewed or created:

This month and in this document we detailed the following:

- An overview to working on GCP.
- A number of scripts and techniques to make virtual machines, place items inside them, and more.

Persons who help this month.

Kiyu Gabriel, Dave Bechberger, and Jim Hatcher.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

Additional resources:

Free DataStax Enterprise training courses,

```
https://academy.datastax.com/courses/
```

Take any class, any time, for free. If you complete every class on DataStax Academy, you will actually have achieved a pretty good mastery of DataStax Enterprise, Apache Spark, Apache Solr, Apache TinkerPop, and even some programming.

This document is located here,

```
https://github.com/farrell0/DataStax-Developers-Notebook
```

```
https://tinyurl.com/ddn3000
```

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.

© Copyright 2020 All Rights Reserved. US Government Users Restricted Rights-Use, duplication or disclosure restricted by GSA ADP Schedule Contract.