Disclaimer
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

Questions or comments on this paper? Please email: quickstart_us_grp@oracle.com.
Table of Contents

Cassandra on Oracle Cloud Infrastructure overview.............................................. 2
Assumptions.......................................................................................................... 3
Instance shape selection....................................................................................... 3
Replication factor and Availability Domains .......................................................... 3
Configuring local NVMe storage............................................................................ 4
Create the required network resources ................................................................. 5
Create the required compute instances ............................................................... 5
Update, configure, install Cassandra on the three nodes ...................................... 7
Create the Cassandra cluster................................................................................ 8
Start the Cassandra cluster .................................................................................. 9
Cassandra on Oracle Cloud Infrastructure overview

This white paper is designed as a quick start reference guide for deploying the Apache Software Foundation’s Cassandra NoSQL database on the Oracle Cloud Infrastructure platform.

Apache Cassandra is a leading NoSQL database providing scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on cloud infrastructure makes it a compelling solution for mission-critical bid data workloads. Cassandra supports replicating across multiple datacenters, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

Oracle Cloud Infrastructure offers hourly metered Bare Metal instances; by eliminating the hypervisor, Oracle can deliver better performance at a lower cost than traditional IaaS providers. In addition to compute unencumbered by a hypervisor, Oracle Cloud Infrastructure offers instances with up to 28TBs of locally attached NVMe storage. Each 28TB instance is capable of over 3 million 4K IOPs/sec, the perfect platform for a Cassandra workload.

Instances in the Oracle Cloud Infrastructure are attached via a 10Gb non-blocking network with no oversubscription. Each Cassandra node has access to the full performance of the interface, there are no “noisy neighbors” or a hypervisor to share resources with. Instances in the same region are always less than 1ms from each other, this means your Cassandra queries will be completed in less time, at a lower cost than any other IaaS provider.

To support highly available Cassandra deployments, Oracle Cloud Infrastructure builds regions with at least 3 Availability Domains (AD). Each AD is a fully independent datacenter with no fault domains shared across ADs. A Cassandra cluster replicated across two ADs is highly available; a Cassandra cluster replicated across three ADs increases data durability and ensures the reliability of the cluster and data set.
Assumptions

Consumers of this document should:

» Be familiar with the fundamentals of the Oracle Cloud Infrastructure:
   » https://docs.us-phoenix-1.oraclecloud.com/

» The Oracle Cloud Infrastructure walkthrough is highly recommended if this is the first time you have used the platform:
   » https://docs.us-phoenix-1.oraclecloud.com/Content/GSG/Reference/overviewworkflow.htm

» Have a basic understanding of Cassandra:
   » Cassandra key concepts
   » http://cassandra.apache.org/

Planning a Cassandra Deployment on Oracle Cloud Infrastructure

A successful Cassandra deployment requires the administrator to weigh multiple factors: initial volume of data, typical workload, replication factor, future data growth. This information is used to select the right instance shape and the number of ADs to deploy across.

Instance shape selection

For Cassandra workloads, customers should use an instance shape that includes local NVMe storage. There are currently two options:

<table>
<thead>
<tr>
<th>BM.HighIO1.512</th>
<th>BM.DenseIO1.512</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.8TBs local NVMe storage</td>
<td>28.8TBs local NVMe storage</td>
</tr>
<tr>
<td>36 cores (72 threads)</td>
<td>36 cores</td>
</tr>
<tr>
<td>512GBs of memory</td>
<td>512GBs of memory</td>
</tr>
</tbody>
</table>

It’s common for customers to start on the HighIO instance type (12.8TB) and eventually outgrow them. Running Cassandra on Oracle Cloud Infrastructure makes it trivial to move to the larger DenseIO (28.8TB) instances without any downtime on your Cassandra cluster. If your data size shrinks over time, it’s also easy to move to the smaller instances.

Replication factor and Availability Domains

Cassandra is designed to replicate data across fault domains. On Oracle Cloud Infrastructure, each Availability Domain (AD) is its own fault domain. It’s important that customers replicate across at least two ADs. Customers who want to increase their data durability should replicate across three ADs. The number of ADs a Cassandra cluster is replicated across impacts the availability and durability of a cluster as well as driving different storage configuration options.
Configuring local NVMe storage

The local NVMe storage on each instance is unprotected; by default there is no redundant RAID. Customers are responsible for making sure the data on those drives is properly protected. Using Cassandra to replicate the data to additional nodes is one way to protect that data, customers can add an additional layer of data protection by creating a redundant RAID array across the NVMe drives.

On each instance the customer will create a RAID array across the NVMe drives, either RAID 10 or RAID 6. RAID 10 provides protection against the failure of a NVMe device but reduces the available storage by 50%. RAID 6 provides parity based protection and, on the BM.Dense instance type will use fewer drives for redundancy than RAID 10 but won’t be as fast.

NOTE: 

Anti-Patterns

Don’t use Oracle Cloud Infrastructure Remote Block Storage LUNs for primary storage. The Block service can be used effectively as a backup target.

Building a Cassandra cluster on Oracle Cloud Infrastructure

In this walkthrough, we will build a 3-node Cassandra cluster in the PHX region that spans three ADs and replicates across all three ADs. We will isolate the Cassandra nodes from the Internet by using security lists.
Create the required network resources

1. A new Virtual Cloud Network
   a. Name – Cassandra_example
   b. CIDR block – 10.0.0.0/27

2. A new internet gateway
   a. Name - Cassandra_IG

3. A new route table
   a. Name - Cassandra_Internet_route
   b. CIDR block – 0.0.0.0/0
   c. Target – Cassandra_IG

4. Three new subnets
   a. Name – AD1_Cassandra_private
   b. AD – PHX-AD-1
   c. CIDR block – 10.0.0.0/29
   d. Route table – Cassandra_Internet_route
   e. Name – AD2_Cassandra_private
   f. AD – PHX-AD-2
   g. CIDR block – 10.0.0.8/29
   h. Route table – Cassandra_Internet_route
   i. Name – AD3_Cassandra_private
   j. AD – PHX-AD-3
   k. CIDR block – 10.0.0.16/29
   l. Route table – Cassandra_Internet_route

Create the required compute instances

1. Three new instances
   a. Name – Cassandra_AD1_0
   b. Image – Oracle-Linux-7.2-2016.10.05-0
   c. Shape – BM.DenseIO1.512
   d. AD – PHX-AD-1
   e. Cloud Network – Cassandra_example
   f. Subnet – AD1_Cassandra_private
   g. SSH Key – <public half of your key pair>
For each instance note the public and RFC1918 IP addresses –

<table>
<thead>
<tr>
<th>Instance</th>
<th>Public IP</th>
<th>RFC1918 IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra_AD1_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra_AD2_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra_AD3_0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: All inter-instance communication should be across the RFC1918 address of the instances, not the public IP. Using the public IP adds latency to the connection and limits the bandwidth. Using the RFC1918 IP guarantees access to the full network bandwidth and the lowest possible latency.
Update, configure, install Cassandra on the three nodes

Repeat these steps on each instance.

1. SSH into the instance -

   $ ssh -i <private key> opc@<public IP>

2. Patch the instance -

   $ sudo yum update -y

3. Install Java and a few other tools we'll use -

   $ sudo yum install java mdadm screen dstat -y

4. Create a RAID 6 array across all 9 NVMe drives, create an XFS filesystem on the array and mount the filesystem -

   $ sudo mdadm --create /dev/md0 --chunk=256 --raid-devices=9 --level=6 /dev/nvme0n1
   /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1 /dev/nvme6n1
   /dev/nvme7n1 /dev/nvme8n1
   $ sudo mdadm --detail --scan | sudo tee -a /etc/mdadm.conf >> /dev/null
   $ screen
   $ sudo mkfs.xfs -s size=4096 -d su=262144 -d sw=6 /dev/md0
   $ sudo mkdir /mnt/cassandra
   $ sudo mount /dev/md0 /mnt/cassandra

   NOTE: We use screen here in case we get disconnected from the instance while the mkfs command is running.

5. Open up the operating system firewall to allow Cassandra to communicate between instances. We limit communication on the Cassandra ports to the VCN subnet.

   $ sudo firewall-cmd --permanent --zone=public \ 
   --add-rich-rule="rule family="ipv4" source address="10.0.0.0/27" \ 
   port protocol="tcp" port="7000-7001" accept"
   $ sudo firewall-cmd --permanent --zone=public \ 
   --add-rich-rule="rule family="ipv4" source address="10.0.0.0/27" \ 
   port protocol="tcp" port="7199" accept"
6. Add the DataStax repo, using yum to install Cassandra is easier than the alternatives

```
$ echo -e "[DataStax]name=DataStax Repo for Apache
        baseurl=http://rpm.datastax.com/datastax-ddc/3.2
        enabled=1
        gpgcheck=0" | sudo tee /etc/yum.repos.d/datastax.repo
$ sudo yum update -y
$ sudo yum install datastax-ddc -y
```

7. Set the cluster name, use the NVMe backed filesystem for data, and a few more details.

```
$ sudo chown cassandra.cassandra /mnt/cassandra/
$ sudo vi /etc/cassandra/conf/cassandra.yaml
line 10: cluster_name: 'ORCL BMC ROCKS'
line 71: hints_directory: /mnt/cassandra/hints
line 170: - /mnt/cassandra/data
line 175: commitlog_directory: /mnt/cassandra/commit_log
line 287: saved_caches_directory: /mnt/cassandra/saved_caches
line 343: -seeds: <comma delimited list of all three RFC 1918 IPs>
line 472: listen_address: <the RFC 1918 IP of the local instance>
line 801: endpoint_snitch: GossipingPropertyFileSnitch
NOTE: To turn on line numbers in vi: :set number
```

Create the Cassandra cluster

Now we configure the Cassandra so that every node belongs to the same cluster. To create a cluster while using the GossipingPropertyFileSnitch endpoint snitch we have to create a different /etc/cassandra/conf/cassandra-rackdc.properties file on each instance.

By default the GossipingPropertyFileSnitch always loads cassandra-topology.properties when it is present. Remove the file from each node.

1. On Cassandra_AD1_0 -

```
$ sudo rm /etc/cassandra/conf/cassandra-topology.properties
$ sudo vi /etc/cassandra/conf/cassandra-rackdc.properties

dc=AD1
rack=RAC1
```
2. On Cassandra_AD2_0 -

```bash
$ sudo rm /etc/cassandra/conf/cassandra-topology.properties
$ sudo vi /etc/cassandra/conf/cassandra-rackdc.properties

dc=AD2
rack=RAC1
```

3. On Cassandra_AD3_0 –

```bash
$ sudo rm /etc/cassandra/conf/cassandra-topology.properties
$ sudo vi /etc/cassandra/conf/cassandra-rackdc.properties

dc=AD3
rack=RAC1
```

Start the Cassandra cluster

On each node –

```bash
$ sudo service cassandra start
```

Check the status of the cluster –

```bash
$ nodetool status
```

Congratulations!

You have a triple redundant, high performance Cassandra cluster up and running in the OBMCS.

Next steps

A Cassandra cluster isn’t much good without data. Here’s a great list of public data sets -
https://www.opensciencedatacloud.org/publicdata/. Your cluster can support about 20TBs of data.

The US Census Dataset is a great place to get started –
https://www.opensciencedatacloud.org/publicdata/us_census/