Deploying Scalable Oracle RAC on Amazon EC2

This article provides details on how to build Oracle Real Application Clusters (RAC) on the Amazon Elastic Compute Cloud (EC2). The configuration used here is just one of many ways to deploy RAC.

Details

Submitted By: Jeremiah Wilton, Prashant Bondada

AWS Products Used: Amazon EC2, Amazon Route 53, Amazon VPC

Created On: October 9, 2015 3:42 AM GMT

Last Updated: May 23, 2016 10:03 PM GMT

The following tutorial relies on the AWS Command Line Interface (CLI). You can run the AWS CLI on your desktop, such as a PC, Mac or Linux machine. To find out more about the AWS CLI, see the AWS Command Line Interface User Guide. Although all steps accomplished with the CLI can be accomplished using the AWS Console, the instructions are limited to the CLI for reasons of clarity and space. In command line examples, commands that you issue are in bold, values you must substitute are in red, and values that you should note for later use are underlined.

This tutorial implements Oracle RAC, which is a complex enterprise-class database technology. It is designed to be followed by an experienced DBA, database engineer or systems engineer with strong knowledge and understanding of AWS, RAC, storage, networking and Linux. It does not implement a managed or fully automated service. Customers implementing RAC using these techniques will likely want to complete the automation components that would be required for any enterprise-class production system.

- Skip directly to the tutorial outline
- Skip directly to the frequently asked questions

Oracle RAC Overview

Oracle Real Application Clusters are a shared-everything database cluster technology from Oracle. RAC allows a single database (a set of data files) to be concurrently accessed and served by one or many database server instances. Multiple instances provide high availability though redundancy at the database server level, and scaling through the addition of instances.

Introducing RAC on EC2

It has been technically possible for some time to run Oracle RAC on EC2. Continual improvements such as placement groups, cluster compute instances, 10Gb Ethernet, local SSD storage and dedicated instances have made running RAC on EC2 increasingly attractive. This tutorial demonstrates building a 2-node RAC cluster using high-performance EC2 infrastructure, but is not meant to be a specific solution for any specific use case. Although the architecture presented here implements features to maximize performance and availability, it is only an example. Customers seeking to deploy Oracle RAC on EC2 will need to architect a solution against their own use cases and requirements. Customers should validate and address all elements of production implementation, including testing failure cases, deploying backups, verifying recovery and testing at scale.

There are numerous potential use cases for deploying RAC on EC2. Developers building software that uses a production RAC database might want to be able to do their development and testing against a RAC database as well. With RAC on EC2, every developer can have his or her own RAC database, which will allow detection of performance and availability regressions related to the RAC architecture in their software.

With RAC, customers can increase or decrease the number of nodes based on demand. However, most customers implement RAC on large, expensive, physical servers. That means that scale out and scale back requires deploying or decommissioning hardware, which can hardly be described as "elastic." On EC2, adding a node to the RAC cluster is as easy as a few API calls and commands. It can be done in minutes, rather than weeks. In this respect, RAC is well matched to the elastic characteristics of AWS.

Some Oracle customers migrating to EC2 want to know that if they needed to, they could scale beyond the capacity of the largest compute instances that AWS provides. With RAC, Oracle customers can have peace of mind, knowing that they can scale beyond the capacity of EC2’s largest instances by adding RAC nodes to their cluster. For customers using RAC on conventionally-hosted infrastructure, EC2 can still provide a compelling use case. These customers can test the extent to which their workload will scale on RAC, by determining the number of nodes at which their workload reaches a point of diminishing returns, and begins to scale inversely.

From an availability perspective, RAC on EC2 provides the same excellent availability characteristics as RAC on conventionally-hosted infrastructure. RAC provides redundancy at the node and instance level, and by implementing complementary client-side availability features such as Application Continuity, applications can survive server and instance failures with little or no impact to customers.
Finally, there are some customers who simply can’t or don’t want to re-architect their Oracle scaling and availability strategy just to move to AWS. These customers may be running third-party software packages for which RAC is a stated requirement. Or they may just be looking to lift and shift their existing architecture into the cloud immediately, then engage gradually over time in re-architecting their applications to take advantage of the next generation of cloud-native databases.

**Historical Obstacles**

**Shared Storage**

Customers thinking of running RAC on EC2 have long come up short upon realizing that Elastic Block Store (EBS) volumes cannot currently be attached to more than one EC2 instance at a time. This means that EBS cannot fulfill the shared storage requirement of RAC. To overcome this limitation, we have built Amazon Machine Images (AMIs) that allow you to launch EC2 instances that can act as network attached storage (NAS) devices. Using iSCSI, these instances can be used to present storage to multiple RAC instances. You could use one of these NAS devices to serve up any amount of EBS storage you like. However, we were not satisfied with the I/O characteristics resulting from the multiple round trips required for serving EBS storage via iSCSI. The RAC node would have to make an I/O call over the network to the NAS instance, but to fulfill that request, the NAS instance would have to make another I/O request to EBS. Only when that request returned could the NAS instance return the original I/O request to the RAC node.

In order to provide high I/O performance, we chose to build the NAS instances on EC2’s i2.8xlarge instance size, which has 6.25 terabytes of local ephemeral solid state disk (SSD). With the i2.8xlarge instances, we can present those SSD volumes via iSCSI to the RAC nodes, and avoid the extra round trip to EBS. It is not required to use this configuration, and some customers may find that using EBS volumes is acceptable, especially for data that does not have stringent I/O requirements, such as historical or archived data.

The SSD volumes that come with the i2.8xlarge instance are ephemeral disk. That means that upon stop, termination or failure of the instance, the data on those volumes will be lost. This is obviously unacceptable to most customers, so we recommend deploying two or three of these NAS instances, and mirroring the storage between them using ASM Normal or High redundancy. This tutorial demonstrates deployment with two NAS instances. Note that we have not yet provided automation within this tutorial to implement automatic configuration, addition and rebalancing of a new storage node. We recommend that production systems take advantage of three NAS instances, or deploy a commercial virtual SAN appliance solution that provides data protection natively for EC2 ephemeral volumes, such as EMC ScaleIO or SoftNAS.

**Multicast IP**

On EC2, the network does not natively support multicast IP. However, RAC needs to broadcast packets over multicast during cluster configuration and reconfiguration events. To provide a discrete network for the interconnect, and enable it for multicast, we deploy a point-to-point VPN among the RAC nodes, using Ntop N2N. N2N allows a group of hosts to join a VPN “community,” and also supports multicast IP among the members. Members of the community run the N2N component called “Edge.” N2N Edge registers with, and obtains information on other members from, a single “supernode” which for the purposes of this tutorial, you will deploy on the storage servers. With N2N, we are able to overcome the multicast networking limitations of EC2 networking.

Currently, this tutorial’s use of N2N has a number of drawbacks about which customers should be aware. Performance of network communications via N2N is significantly lower than non-VPN traffic over the same EC2 network. In addition, Ntop, the makers of N2N, are no longer developing the N2N product. These factors may preclude running production-class RAC workloads over N2N. Currently, we are developing new approaches using clusterware-managed dynamic creation of GRE tunnels to serve the cluster’s multicast needs. We will provide details on this approach and revise this tutorial accordingly before August 2016.

**Tutorial**

**Overview**

This tutorial allows you to build an Oracle RAC 12c cluster on EC2. If you need to run a version of Oracle other than 12c, then you should note the approaches used in this design, feel free to use the components we have built into the 12c AMI, but build your 10g or 11g cluster from a bare-bones O/S, from the ground up.

The design consists of two iSCSI target NAS devices serving their local SSD storage, and two RAC nodes. To minimize latency and maximize throughput, this tutorial deploys all components on cluster compute EC2 instances with 10Gbe network capabilities. To further enhance performance, this tutorial deploys all components in a single EC2 placement group, which minimizes network latencies between the instances. To provide DNS resolution for the RAC-related network names, such as the SCAN address and the VIPs, this tutorial deploys an AWS Route 53 private hosted zone.
Obtaining the AMIs

The AMIs used in this tutorial are available free through AWS Marketplace. There is no cost for the AMIs. You only pay standard rates for EC2 usage. You can search for “Oracle RAC” there, or go to the specific detail pages for the two components, the storage servers and the RAC nodes. The URLs for these products are:

- **Storage Servers**: aws.amazon.com/marketplace/pp/B0169D8CT2
- **RAC Nodes**: aws.amazon.com/marketplace/pp/B0169D8GCA

When you order these products via marketplace, you should choose the “Manual Launch” option. This will allow you access to the AMIs in the region you choose via the AWS Command Line Interface (CLI). Be sure to note the AMI IDs you obtain via AWS Marketplace, as you will use these AMI IDs to launch instances in this tutorial.

**Tutorial Outline**

1. **Step 1: Create a VPC and network components**
2. **Step 2: Set up DNS**
3. **Step 3: Setting up security group and key pair**
4. **Step 4: Setting Up shared storage**
5. **Step 5: Create the first RAC node**
6. **Step 6: Download Oracle software to your EC2 instance**
7. **Step 7: Install Oracle Grid Infrastructure**
8. **Step 8: Configure Oracle Cluster Ready Services (CRS)**
9. **Step 9: Add VIP and SCAN services to Oracle CRS**
10. **Step 10: Set Oracle Automatic Storage Management (ASM) parameters**
11. **Step 11: Create disk groups**
12. **Step 12: Database software installation**
13. **Step 13: Database creation**
14. **Step 14: Adding a second node**
15. **Step 15: (Optional) Configuring Flash Cache**

**Step 1: Create VPC and Network Components**

First, create a VPC in the region you want to use and assign the VPC a name that you will associate with this tutorial. If you want to build the RAC cluster within an existing VPC, make sure you adjust the IP address ranges to suit your VPC. You will also change the VPC attributes so that you can use Route 53 DNS with the VPC. The following commands create a VPC with a CIDR rule of 10.0.0.0/16, tag the VPC, and then enable DNS support. Be sure to note the VPC ID that is created.

**Command:**

```bash
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

**Sample output:**

```
VPC 10.0.0.0/16 dopt-82eefee0 default pending vpc-1951227c
```

**Commands:**

```bash
$ aws ec2 create-tags --tags Key=Name,Value=MyRacVPC --resources vpc-1951227c
$ aws ec2 modify-vpc-attribute --vpc-id vpc-1951227c --enable-dns-support
$ aws ec2 modify-vpc-attribute --vpc-id vpc-1951227c --enable-dns-hostnames
```

If you want to be able to connect to your VPC from the Internet, add an Internet gateway, and attach it to your VPC. The following command adds an Internet gateway to the VPC created in the previous section.

**Command:**

```bash
$ aws ec2 create-internet-gateway
```

**Sample output:**

```
INTERNETGATEWAY igw-928503f7
```
Command:

```bash
$ aws ec2 attach-internet-gateway --internet-gateway-id igw-928503f7 --vpc-id vpc-1951227c
```

Next, determine the route table ID for your VPC using the `describe-route-tables` command:

Command:

```bash
$ aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-1951227c"
```

Sample output:

```
ROUTETABLES
  rtb-66c5bd03  vpc-1951227c...
```

Add a route to the VPC's route table so that network traffic can flow through the Internet gateway. The following command adds the CIDR block 0.0.0.0/0 to the route table of the VPC:

Command:

```bash
$ aws ec2 create-route --route-table-id rtb-66c5bd03 --destination-cidr-block 0.0.0.0/0 \  --gateway-id igw-928503f7
```

Next, create a subnet within your VPC, and tag it with a name that you will associate with this tutorial. If you are using a preexisting subnet, you will need to use a CIDR block appropriate to that subnet. The following commands create a subnet in the VPC, assign the CIDR block 10.0.0.0/24, and then add a tag to the subnet.

Command:

```bash
$ aws ec2 create-subnet --vpc-id vpc-1951227c --cidr-block 10.0.0.0/24
```

Sample output:

```
SUBNET  us-west-2a  251 10.0.0.0/24 pending subnet-74307211 vpc-1951227c
```

Command:

```bash
$ aws ec2 create-tags --resources subnet-74307211 --tags Key=Name,Value=MyRacSubnet
```

In order to minimize latency between the components, create a placement group for the RAC cluster. When you deploy EC2 cluster compute instances in an EC2 placement group, it allows you to take advantage of the 10Gb Ethernet networking capabilities of these instances. The following command creates an EC2 placement group:

Command:

```bash
$ aws ec2 create-placement-group --group-name MyRACPlacementGroup --strategy cluster
```

### Step 2: Set up DNS

The initial RAC cluster will have two nodes. Below is a table of the DNS names and addresses for the cluster. In this step, you will create an AWS Route 53 private hosted zone to serve these DNS names privately within your VPC. You should use addresses within your subnet’s range.

<table>
<thead>
<tr>
<th>DNS Name</th>
<th>IP Address(es)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCAN</td>
<td>myrac-scan.myrachostedzone.net</td>
</tr>
<tr>
<td>SCAN 1</td>
<td>myrac-scan1.myrachostedzone.net</td>
</tr>
<tr>
<td>SCAN 2</td>
<td>myrac-scan2.myrachostedzone.net</td>
</tr>
<tr>
<td>SCAN 3</td>
<td>myrac-scan3.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 1 Private</td>
<td>myrac01-priv.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 1 VIP</td>
<td>myrac01-vip.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 1</td>
<td>myracnode01.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 2 Private</td>
<td>myrac02-priv.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 2 VIP</td>
<td>myrac02-vip.myrachostedzone.net</td>
</tr>
<tr>
<td>Node 2</td>
<td>myracnode02.myrachostedzone.net</td>
</tr>
<tr>
<td>iSCSI Target 1</td>
<td>myiscsitarget01.myrachostedzone.net</td>
</tr>
<tr>
<td>iSCSI Target 2</td>
<td>myiscsitarget02.myrachostedzone.net</td>
</tr>
</tbody>
</table>

**Tab. 1: RAC DNS names and IP addresses**

If you already use a private hosted zone for your subnet, then you don’t need to create one. The following command creates an AWS Route 53 hosted zone and associates it with the VPC. Choose a value for `caller-reference` that you will associate with this tutorial:

Command:

```bash
$ aws route53 create-hosted-zone --name myrachostedzone.net \  --vpc VPCRegion=us-west-1, VPCId=vpc-1951227c \  --caller-reference myrac-2015-09-29-a
```
To populate the DNS records for Route 53, you will transpose the above names and addresses into a JSON payload that the AWS CLI can use. Using a text editor, copy the following JSON code into a file in your local directory on the machine where you run the CLI. Name the file MyRACDNSRecords.json.

```json
{
  "Changes": [
    {"Action": "UPSERT",
     "ResourceRecordSet": {
       "Name": "myrac-scan.myrachostedzone.net","Type": "A","TTL": 300,
       "ResourceRecords": ["Value": "10.0.0.31"],"Value": "10.0.0.32"],"Value": "10.0.0.33"}
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan1.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan2.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan3.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan4.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan5.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan6.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan7.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan8.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan9.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan10.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan11.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan12.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan13.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan14.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan15.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan16.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan17.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan18.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan19.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan20.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan21.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan22.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan23.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan24.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"]
    }
    ,
    {"Action": "UPSERT",
    "ResourceRecordSet": {
    "Name": "myrac-scan25.myrachostedzone.net","Type": "A",TTL": 300,
    "ResourceRecords": ["Value": "10.0.0.31"
```

With this JSON file, you can configure the AWS Route 53 private hosted zone. The following command passes the JSON file you created to AWS CLI and populates the DNS records:

```
$ aws route53 change-resource-record-sets --hosted-zone-id Z353E5Q1II928R \
  --change-batch file://MyRACDNSRecords.json
```

Sample output:

```
```

**Step 3: Setting up security group and key pair**

Create a security group to control access, and allow all traffic between member instances of that security group. The following commands create an EC2 security group and authorize network communication between all instances associated with the security group:

Command:

```
```
$ aws ec2 create-security-group --group-name MyRACSecurityGroup \\ --description "My RAC Security Group" --vpc-id vpc-1951227c

Sample output:

GROUPID sg-a7cbb6

Command:

$ aws ec2 authorize-security-group-ingress --group-id sg-a7cbb6 --protocol -1 --port -1 \\ --source-group sg-a7cbb6

You may also need to authorize access into the instances from the IP address from which you are working. For more information on authorizing inbound traffic, see the Amazon EC2 User Guide for Linux Instances.

Create a key pair, so you can log in to the instances you will create. The following command creates a key pair that you will use in the next step:

Command:

$ aws ec2 create-keypair --key-name MyRACKeyPair

Sample output:

BEGIN RSA PRIVATE KEY
MIIEogIBAAKCAQEAu6bnCXumXMMYxjZdPDdPHxa57HrB4VQi2rpmFExaS2EEEPu81/cXaxs4WXEB0
...
3EGAD4t7x4YfTD3K4EUFGZG6FmWxN8x7e23WFSYgHb4bG7JkWdcgG9y2VfA9c/sLyj8=
END RSA PRIVATE KEY

Copy everything from BEGIN RSA PRIVATE KEY through END RSA PRIVATE KEY into a file on your local host called MyRACKeyPair.pem, using a text editor, then change the permissions on the file. The following command changes the permissions on the key pair file you created:

$ chmod 600 MyRACKeyPair.pem

If you plan to use PuTTY terminal software for Windows to connect to your hosts, the .pem file can be converted into a .ppk file using the PuTTY program puttygen.exe. The following command converts the key pair file you created to a .ppk file that can be used to connect using PuTTY:

C:\> puttygen.exe MyRACKeyPair.pem -o MyRACKeyPair.ppk

Step 4: Setting up shared storage

To prepare to build the iSCSI target instances, you start by creating a couple of files on our local host where you are running the AWS CLI. The first file is a user-data script that will be used during EC2 instance creation to configure the storage servers on first boot. This file will place the SSD storage under LVM control, stripe the storage and present it via the iSCSI target service, and set the hostname. You will create one file for each of the two storage servers (01 and 02).

Using a text editor, copy the following code into a file named user-data-tgt01.sh.

```
#!/bin/bash
ORDINAL=01
/var/opt/first_boot.sh $ORDINAL
```

Repeat for the second storage server. Using a text editor, copy the following code into a file called user-data-tgt02.sh.

```
#!/bin/bash
ORDINAL=02
/var/opt/first_boot.sh $ORDINAL
```

Next, you will use these user-data scripts to create and tag two iSCSI storage servers on dedicated tenancy high-end I/O optimized instances with 10g network. The AMI ID you will substitute below should be the AMI ID for the "iSCSI Target Server for Oracle RAC" which you obtained via AWS Marketplace.

This tutorial is designed for i2.8xlarge storage server instances only. You can choose any instance type you like, but because storage sizes and presentations vary by instance class, you will have to modify the /var/opt/first_boot.sh file on the instance, and create a new AMI from your instance, which includes volume mapping appropriate to the instance you have chosen.

This script partitions the storage, presents it to LVM, and makes it available via iSCSI. You do not have to build the cluster on dedicated tenancy. This tutorial does so in order to provide maximum performance. The following commands each create two EC2 i2.8xlarge instances using the user-data files, placement group, VPC, subnet, and key pairs you created previously. The third and fourth commands assign tags to each of the instances. Be sure to substitute the AMI ID with the iSCSI Target AMI you have obtained from AWS Marketplace.

Command:
Step 5: Create the first RAC node

With the VPC, DNS and iSCSI targets in place, you are ready to launch the first RAC node. You will begin with a single node, and add a second node later. As with the iSCSI targets, you need to create a user-data script to correctly configure the RAC node on first boot. The AMI ID you will substitute below should be the AMI ID for the “Compute Node for Oracle RAC” which you obtained via AWS Marketplace.

Note: This tutorial and the scripts included are designed to create RAC nodes on c3.8xlarge instances only. You can build the cluster on any size instances you like, but like the storage servers, you will have to change the first_boot scripts in /var/opt/ on the RAC host to configure the host appropriately. In addition, you will need to change memory settings in the response files in /home/oracle/dbca.rsp.

Use a text editor to copy the following code and then paste it into a file named user-data-rac01.sh. Be sure to substitute your own AWS access keys. The keys are used to configure the AWS CLI on the RAC nodes, which is required by RAC to move the VIP and SCAN IPs within your VPC.

```
#!/bin/bash

ORDINAL=01
PRIV=10.1.0.21
PUB=10.0.0.11
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=

/aws/opt/first_node_first_boot.sh $ORDINAL $PRIV $PUB $AWS_ACCESS_KEY_ID $AWS_SECRET_ACCESS_KEY
```

Using this user-data script, create and tag the first RAC node instance. Be sure to use the AMI ID from AWS Marketplace for Oracle RAC Nodes. The first command creates the first RAC node, and the second command assigns a tag to the RAC node:

```
aws ec2 run-instances --image-id ami-40b25773 --instance-type c3.8xlarge --key-key MyRACKeyPair
  --placement AvailabilityZone=us-west-2a,GroupAMIName=MyRACPlacementGroup.Tenancy=dedicated
  --associate-public-ip-address --subnet subnet-74307211 --private-ip-address 10.0.0.11
  --security-group-ids sg-a27ccbc6 --user-data file://user-data-rac01.sh
```

Sample output:
```
34329338502 r-dhe70431
INSTANCES 0 x86_64 None False xen ami-40b25773
```

Commands:
```
aws ec2 create-tags --resources i-793210bc --tags Key=Name,Value=MyRACNode01
```

Note the network interface ID (eni-xxxxx) and instance ID in the response from the instance creation above. You will now add the VIP and SCAN IP addresses to the instance using the network interface ID. The following commands add the VIP and SCAN IP addresses to the RAC node instance:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.21
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.31
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.32
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.33
```

Sample output:
```
34329338502 r-dhe70431
INSTANCES 0 x86_64 None False xen ami-40b25773
```

Commands:
```
aws ec2 create-tags --resources i-793210bc --tags Key=Name,Value=MyRACNode01
```

Note the network interface ID (eni-xxxxx) and instance ID in the response from the instance creation above. You will now add the VIP and SCAN IP addresses to the instance using the network interface ID. The following commands add the VIP and SCAN IP addresses to the RAC node instance:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.21
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.31
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.32
aws ec2 assign-private-ip-addresses --network-interface-id eni-01el6a47 --private-ip-addresses 10.0.0.33
```
Step 6: Download Oracle software to your EC2 instance

The next task is to download software from Oracle. Log into your Oracle account at Oracle eDelivery. In the Product search box, type Oracle Database Enterprise, then click on the suggested option, Oracle Database Enterprise Edition.

Fig. 2: Searching for Oracle software on Oracle eDelivery

Next, click on "Select Platform," and select "Linux x86-64," and click "Continue."

Fig. 3: Selecting the Linux x86-64 platform

In the following page, eDelivery will list Oracle Database Enterprise Edition 12.1.0.2.0... Click the triangle at the left to expand this item, and reveal the component parts. Uncheck all items except Oracle Database... and Oracle Database Grid Infrastructure. Click Continue. The site will prompt you to accept terms. If you have accepted the terms, click Continue.

Fig. 4: Selecting Database and Grid Infrastructure software

On the next screen, you will see a list of .zip files to download. Instead of downloading them to your desktop, click on WGET Options at the bottom:

Fig. 5: Zip files

This will bring up a pane explaining the wget utility. In this pane, click Download .sh. This will download a shell script called wget.sh, containing the wget commands that will download the software you need onto your EC2 instance.

Edit the wget.sh file using a text editor. Add your Oracle login credentials to the beginning of the script, as in the following example. In some cases, your user name but not your password may already be in the file.
# SSO username and password
SSO_USERNAME=foo@halcyon.com
SSO_PASSWORD=6sdd0bbYY

Next, note the files in the file list from the eDelivery website (Fig. 5) comprising the media for Standard Edition 2 (SE2). They are V77388-01_1of2.zip and V77388-01_2of2.zip in the above example. Remove or comment out those lines from the wget.sh script.

You need one more piece of information before you can use the contents of the wget.sh file. Use the command line to describe the RAC node instance. Note the public IP address; you will need this value later. The following command describes the instance:

**Command:**

```
$ aws ec2 describe-instances --instance-id i-7e7375bb
```

**Sample output:**

```
RESERVATIONS 343299325021 r-doa6822b
INSTANCES 0 x86_64 None False xen-ami-b95f4789 i-7e7375bb 12.8xlarge MyRACKeyPair
ec2-52-88-78-69.us-west-2.compute.amazonaws.com 52.88.78.69 /dev/xvda ebs True None ...
```

Copy the wget.sh file to your RAC node instance, using the utilities scp or pscp. The following commands can be used to copy the wget.sh file to your RAC node instance:

**Commands:**

```
$ scp -i MyRACKeyPair.pem wget.sh ec2-user@52.88.78.69:
```

or

```
C:\> pscp -i MyRACKeyPair.ppk wget.sh ec2-user@52.88.78.69:
```

Use SSH or PuTTY to connect to the host’s command line. The following SSH and PuTTY commands allow you to connect to your instance via SSH.

**Commands:**

```
$ ssh -i MyRACKeyPair.pem ec2-user@52.88.78.69
```

or

```
C:\> putty -i MyRACKeyPair.ppk ec2-user@52.88.78.69
```

Move the wget.sh file to a directory called media under the oracle user’s home directory, and change it to be owned by the oracle user account. The following commands move the file to the media directory and change the file’s ownership to oracle:

**Commands:**

```
myracnode01$ sudo mv wget.sh ~oracle/media/
myracnode01$ sudo chown oracle:oinstall ~oracle/media/wget.sh
```

Login using the oracle user account, change directory to the media directory, and start the download. The following commands log in as oracle, change the directory and execute the wget.sh script file:

**Commands:**

```
ome2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ cd media
oracle@myracnode01$ sh wget.sh &
```

The download will be logged to a file in /home/oracle/media/ called wgetlog-log. You can monitor progress by tailing that log:

**Commands:**

```
oracle@myracnode01$ tail -f wgetlog-09-22-15:58:08.log
```

**Sample output:**

```
2015-09-22 15:58:17 (2.01 MB/s) - './V46096-01_2of2.zip' saved [646972897/646972897]
```

When the download is complete, you should have four media files, something like these:

**Command:**

```
oracle@myracnode01$ ls -l V4609*.zip
```

**Sample output:**

```
-rw-rw-r-- 1 oracle oracle 646972897 May 14 15:58 V46096-01_1of2.zip
-rw-rw-r-- 1 oracle oracle 646972897 May 14 15:58 V46096-01_2of2.zip
-rw-rw-r-- 1 oracle oracle 646972897 May 14 15:58 V46097-01_1of2.zip
-rw-rw-r-- 1 oracle oracle 646972897 May 14 15:58 V46097-01_2of2.zip
```
Unzip the files:

Command:
```
oracle@myracnode01$ unzip 'V4609*.zip'
```

Sample output:
```
Archive:  V46095-01_1of2.zip
     creating: database/
     creating: database/rpm/
     ...
```

In addition, you should see two new directories, database and grid, in the media directory. Now you are ready to install the Oracle Grid Infrastructure software.

**Step 7: Install Oracle Grid Infrastructure**

Throughout this tutorial, we have provided Oracle Universal Installer response files pre-configured to correctly deploy software and services. These files are located on the RAC node in /home/oracle/. We have pre-populated the passwords for privileged users in these response files with the value v5uPT6JJ. We strongly advise you to set your own passwords. To do this at install time, simply edit the response files with the editor of your choice, and replace all instances of v5uPT6JJ with your own password values.

To install the Oracle Grid Infrastructure software, change directories to the grid directory (under the media directory), and run the installer. We have provided a working response file for the installation. The following commands change the directory and run the installer:

Commands:
```
oracle@myracnode01$ cd grid
oracle@myracnode01$ ./runInstaller -silent -ignoreprereq -responsefile /home/oracle/grid.rsp
```

Sample output
```
Starting Oracle Universal Installer...  
Checking Temp space: must be greater than 415 MB.  Actual 80120 MB    Passed  
Checking swap space: must be greater than 150 MB.  Actual 20479 MB    Passed  
Preparing to launch Oracle Universal Installer from /tmp/OraInstall2015-09-24_12-17-58AM.  
Please wait ... You can find the log of this install session at: 
/u01/app/oraInventory/logs/installActions2015-09-24_12-17-58AM.log
```

To monitor the progress of the installation, you can tail the log file:

Commands:
```
oracle@myracnode01$ tail -100f /u01/app/oraInventory/logs/installActions2015-09-24_12-17-58AM.log
```

Sample output:
```
INFO: Setting variable 'OPTIONAL_CONFIG_TOOLS' to '[]'. Received the value from a code block.  
INFO: Setting variable 'OPTIONAL_CONFIG_TOOLS' to '[]'. Received the value from a code block.  
...  
INFO: Exit Status is 0  
INFO: Shutdown Oracle Grid Infrastructure 12c Release 1 Installer
```

When the install is complete, you should see the final two INFO lines above near the end of the log. Now log out of the oracle account and run the following scripts using sudo.

Commands:
```
oracle@myracnode01$ exit
ec2-user@myracnode01$ sudo /u01/app/oraInventory/orainstRoot.sh
```

Sample output:
```
Changing permissions of /u01/app/oraInventory.  
Adding read,write permissions for group.  
Removing read,write,execute permissions for world.  
Changing groupname of /u01/app/oraInventory to oinstall.  
The execution of the script is complete.
```

Command:
```
ec2-user@myracnode01$ sudo /u01/app/12.1.0/grid/root.sh
```

Sample output:
```
Check /u01/app/12.1.0/grid/install/root_myracnode01_2015-09-24-00-27-41.log for the output of root script
```

You may move immediately on to the next step once these commands complete.

**Step 8: Configure Oracle Cluster Ready Services (CRS)**
Now you are ready to configure Oracle CRS on the first RAC node. In this step, you will log in as the oracle user account and run the Cluster Configuration Assistant using the provided response file. The Oracle installer uses the values in the response file to provide answers to some or all of the Oracle installer prompts. The following commands login as the oracle user account, change directory, and then run the Oracle installer using the response file:

Commands:
```
ec2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ cd /u01/app/12.1.0/grid/crs/config
oracle@myracnode01$ ./config.sh -silent -ignoreprereq -responsefile /home/oracle/gridconfig.rsp
```

When the configuration is complete, log out of the oracle user, and run the following scripts using sudo:

Commands:
```
oracle@myracnode01$ exit
ec2-user@myracnode01$ sudo /u01/app/12.1.0/grid/root.sh &
```

Sample output:
```
CRS-2672: Attempting to start 'ora.asm' on 'myracnode01'
CRS-2676: Start of 'ora.asm' on 'myracnode01' succeeded
Preparing packages...
```

You should see the final ... succeeded message near the end of the log.

**Step 9: Add VIP and SCAN services to Oracle CRS**

Although Oracle CRS will bring IP addresses down and up within the cluster when VIP and SCAN addresses move, Oracle CRS has no native capability to change the EC2 instance with which those IP addresses are associated. You must add services to Oracle CRS that will make calls to the EC2 API, so that the VIP addresses will be moved to the correct ENI within your VPC. Login using the oracle user account, source your environment for the +ASM1 instance, and run the commands:

Commands:
```
ec2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] ? +ASM1
oracle@myracnode01$ crsctl add resource myrac01-vip.ec2 -type cluster_resource -attr "CHECK_INTERACTION_SCRIPT=/etc/oracle/ec2_vip_failover.sh,PLACEMENT=favored,HOSTING_MEMBERS=myracnode01,AUTO_START_DEPENDENCIES=hard(intermediate:ora.myracnode01.vip),RELOCATE_BY_DEPENDENCY=1,STOP_DEPENDENCIES=hard(intermediate:ora.myracnode01.vip)"
oracle@myracnode01$ crsctl add resource myrac-scan1.ec2 -type cluster_resource -attr "CHECK_INTERACTION_SCRIPT=/etc/oracle/ec2_vip_failover.sh,PLACEMENT=favored,HOSTING_MEMBERS=myracnode01,AUTO_START_DEPENDENCIES=hard(intermediate:ora.scan1.vip),RELOCATE_BY_DEPENDENCY=1,STOP_DEPENDENCIES=hard(intermediate:ora.scan1.vip)"
oracle@myracnode01$ crsctl add resource myrac-scan2.ec2 -type cluster_resource -attr "CHECK_INTERACTION_SCRIPT=/etc/oracle/ec2_vip_failover.sh,PLACEMENT=favored,HOSTING_MEMBERS=myracnode01,AUTO_START_DEPENDENCIES=hard(intermediate:ora.scan2.vip),RELOCATE_BY_DEPENDENCY=1,STOP_DEPENDENCIES=hard(intermediate:ora.scan2.vip)"
oracle@myracnode01$ crsctl add resource myrac-scan3.ec2 -type cluster_resource -attr "CHECK_INTERACTION_SCRIPT=/etc/oracle/ec2_vip_failover.sh,PLACEMENT=favored,HOSTING_MEMBERS=myracnode01,AUTO_START_DEPENDENCIES=hard(intermediate:ora.scan3.vip),RELOCATE_BY_DEPENDENCY=1,STOP_DEPENDENCIES=hard(intermediate:ora.scan3.vip)"
```

You should see the final ... succeeded message near the end of the log.

**Step 10: Set Oracle Automatic Storage Management (ASM) parameters:**

Several Oracle ASM instance parameters need to be set, such as cluster_interconnects and memory parameters. We have supplied a script. The following command runs the provided script to set the Oracle ASM parameters:

Commands:
```
oracle@myracnode01$ sqlplus / as sysasm @asm_parameters
```

To allow ASM to take the new parameter value, you must log out of the oracle user, become the root user using sudo, then restart Oracle CRS:

Commands:
```
oracle@myracnode01$ exit
ec2-user@myracnode01$ sudo su -
root@myracnode01$ . oraenv
ORACLE_SID = [oracle] ? +ASM1
root@myracnode01$ crsctl stop crs
root@myracnode01$ crsctl start crs
root@myracnode01$ crsctl check crs
```

Check `/u01/app/12.1.0/grid/install/root_myracnode01_2015-09-24_19-35-15.log` for the output of root script.
Before proceeding, make sure the command crsctl check crs shows that Oracle CRS is up. This could take a few minutes.

**Step 11: Create disk groups**

You will store the database files in an ASM disk group called DATA, which uses normal Oracle ASM redundancy to mirror the storage across the two storage nodes. You will also create a disk group called RECO for the recovery area.

We have provided a script to create these disk groups. The following commands run this script:

Commands:

```
root@myracnode01# exit
e2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] ?ASM1
oracle@myracnode01$ sqlplus / as sysasm @add_diskgroups
```

**Step 12: Database software installation**

To install the Oracle Database software, login using the oracle user account, then change to the database directory (under the media directory), and run the Oracle installer. We have provided a working response file for the installation.

The following commands run the Oracle installer using the provided response file:

Commands:

```
oracle@myracnode01$ cd media/database
oracle@myracnode01$ ./runInstaller -silent -ignoreprereq -responsefile /home/oracle/db.rsp
```

Sample output:

```
Starting Oracle Universal Installer...
Checking Temp space: must be greater than 500 MB. Actual 81829 MB Passed
Checking swap space: must be greater than 150 MB. Actual 20479 MB Passed
Preparing to launch Oracle Universal Installer from /tmp/OraInstall2015-09-24_07-51-49PM. Please see the readme file for details.
You can find the log of this install session at:
/u01/app/oraInventory/logs/installActions2015-09-24_07-51-49PM.log
```

To monitor the progress of the database software installation, you can tail the log file:

Command:

```
oracle@myracnode01$ tail -f /u01/app/oraInventory/logs/installActions2015-09-24_07-51-49PM.log
```

Sample output:

```
INFO: Dispose the install area control object
INFO: Update the state machine to STATE_CLEAN
INFO: Finding the most appropriate exit status for the current application
INFO: Exit Status is 0
INFO: Shutdown Oracle Database 12c Release 1 Installer
INFO: Unloading Setup Driver
```

When the install is complete, you should see the final Exit Status is 0 message near the end of the log. Then log out of the oracle user and run the following scripts using sudo:

Commands:

```
oracle@myracnode01$ exit
e2-user@myracnode01$ sudo /u01/app/oracle/product/12.1.0/dbhome_1/root.sh
```

Sample output:

```
/u01/app/oracle/product/12.1.0/dbhome_1/install/root_myracnode01_2015-09-24_19-58-57.log
```

**Step 13: Database creation**

Now you will create a database. We have provided a working response file for the installation. First, become the oracle user and source your environment for the database ORACLE_HOME:

Commands:

```
e2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] ?
ORACLE_HOME = [/home/oracle] ? /u01/app/oracle/product/12.1.0/dbhome_1
```

You will make a small modification to the database creation template that comes with the database software. The following command increases the redo log file size to 2G

```
/oracle@myracnode01$ sed -i -e "s/<filesize unit="KB">51200</fileSize>/<fileSize unit="KB">2097152</fileSize>/g" $ORACLE_HOME/assistants/dbca/templates/General_Purpose.dbc
```

Now run the database creation assistant, using our response file. The following command runs the database creation assistant using the provided response file:

```
/oracle@myracnode01$ dbca -silent -responsefile dbca.rsp
```
Step 14: Adding a second node

Now you're ready to add the second node. You will start RAC node 2 from the same AMI you used for RAC node 1. The only differences will be the IP addresses and the user-data script called. Using a text editor copy the following code into a file named user-data-rac02.sh. Be sure to substitute your own AWS access keys. The keys are used to configure the AWS CLI on the RAC nodes, which is required by RAC to move the VIP and SCAN IPs within your VPC.

```
#!/bin/bash
ORDINAL=02
PRIV=10.1.0.22
PUB=10.0.0.12
AWS_ACCESS_KEY_ID=xxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxx
/var/opt/next_node_first_boot.sh $ORDINAL $PRIV $PUB $AWS_ACCESS_KEY_ID $AWS_SECRET_ACCESS_KEY
```

On your local machine, where you run the AWS CLI, create and tag the second RAC node instance. Be sure to substitute the AMI ID for Oracle RAC Nodes that you obtained from AWS Marketplace. You should note the instance ID and ethernet interface ID (ENI) that are returned from this command. The following command creates the second RAC node instance:

```
Command:
$ aws ec2 run-instances --image-id ami-40b25773 --instance-type c3.8xlarge --key MyRACKeyPair 
  --place=AvailabilityZone=west-2a,GroupName=MyRACPlacementGroup,Tenancy=dedicated 
  --associate-public-ip-address --subnet subnet-74307211 --private-ip-address 10.0.0.12 
  --security-group-ids sg-b726f72 --user-data file://user-data-rac02.sh
```

Sample output:

```
RESERVATIONS 343299325021 r-a1354156
INSTANCES 0 x86_64 None False xen ami-f90c16c9 i-b7726f72
NETWORKINTERFACES None 02:0d:32:ec:31:ab eni-60327f06
...```

```
Command:
$ aws ec2 create-tags --tags Key=Name,Value=MyRACNode02 --resources i-b7726f72
```

Note the instance ID and network interface ID (eni-xxxxx) in the response. You will need these values later. You may wish to wait a few minutes so that the second RAC node is completely up. Now you will add the VIP IP address to the instance. The following command adds the VIP address to the RAC node instance:

```
Command:
$ aws ec2 assign-private-ip-addresses --network-interface-id eni-60327f06 
  --private-ip-addresses 10.0.0.22
```

Now you will copy the oracle user's SSH keys from the first node to the second, by way of an S3 bucket. On the first RAC node, log in using the oracle user account. Then run the following commands to create an S3 bucket with a unique name, and copy the contents of the .ssh/ directory to the bucket:

```
Commands:
ec2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ aws s3 mb s3://mybucket_`dd bs=8 if=/dev/urandom count=1 2>/dev/null | base64 | tr -d [:punct:]
make_bucket: s3://mybucket_rTIz3D7C7s
oracle@myracnode01$ aws s3 cp --recursive .ssh s3://mybucket_rT1z307C7s
upload: .ssh/known_hosts to s3://mybucket_rT1z307C7s/known_hosts
upload: .ssh/authorized_keys to s3://mybucket_rT1z307C7s/authorized_keys
upload: .ssh/id_rsa to s3://mybucket_rT1z307C7s/id_rsa
upload: .ssh/id_rsa.pub to s3://mybucket_rT1z307C7s/id_rsa.pub
```

Next, log into the second RAC node as the oracle user, and use the following commands to copy the SSH keys from your S3 bucket to the .ssh/ directory, then delete the bucket:

```
Commands:
ec2-user@myracnode02$ sudo su - oracle
oracle@myracnode02$ aws s3 cp --recursive s3://mybucket_rT1z307C7s .ssh/
download s3://mybucket_rT1z307C7s/authorized_keys to .ssh/authorized_keys
download s3://mybucket_rT1z307C7s/known_hosts to .ssh/known_hosts
download s3://mybucket_rT1z307C7s/id_rsa to .ssh/id_rsa
download s3://mybucket_rT1z307C7s/id_rsa.pub to .ssh/id_rsa.pub
```

```
oracle@myracnode02$ aws s3 rb --force s3://mybucket_rT1z307C7s
delete s3://mybucket_rT1z307C7s/authorized_keys
delete s3://mybucket_rT1z307C7s/id_rsa
```
Now you are ready to add the second node to the cluster. Log into the first node as the oracle user, then source your environment for the +ASM1 instance, and run the Node Addition Assistant. The following commands change to the oracle user account and run the Node Addition Assistant:

```bash
ec2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] +ASM1
oracle@myracnode01$ /u01/app/12.1.0/grid/addnode/addnode.sh -silent -ignoreprereq -CLUSTER_NEW_NODES={myracnode02} -CLUSTER_NEW_VIRTUAL_HOSTNAMES={myrac02-vip}
```

When the Node Addition Assistant completes, you will need to describe the second node to get the public IP address, so you can log in. The following command describes the instance:

```bash
$ aws ec2 describe-instances --instance-id i-552c3890
```

Sample output:

```
RESERVATIONS 343299325021 r-al354156
INSTANCES 0 x86_64 None False xen ami-f90c16c9 i-b772672 c3.xlarge MyRACKeyPair
2015-09-30T18:11:50.000Z i-10-0-0-13.us-west-2.compute.amazonaws.com 52.88.7.255 ec2-52-88-7-255.us-west-2.compute.amazonaws.com
... 
```

Log into the second RAC node, and run the following scripts using sudo. The following commands log into the second node, and run the root scripts:

```bash
ec2-user@myracnode02 $ sudo /u01/app/oraInventory/orainstRoot.sh
```

Sample output:

```
Changing permissions of /u01/app/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.
Changing groupname of /u01/app/oraInventory to oinstall.
The execution of the script is complete.
```

```bash
ec2-user@myracnode02 $ sudo /u01/app/12.1.0/grid/root.sh &
```

Sample output:

```
Check /u01/app/12.1.0/grid/install/root_myracnode02_2015-09-28_17-02-16.log for the output of root.sh
```

To monitor the progress of the Oracle CRS configuration on the second node, you can tail the log file:

```bash
ec2-user@myracnode02 $ sudo tail -f /u01/app/12.1.0/grid/install/root_myracnode02_2015-09-28_17-02
```

Sample output:

```
Operation successful.
Preparing packages...
cvuqdisk-1.0.9-1.x86_64 2015/10/01 20:17:51
CLSRSC-325: Configure Oracle Grid Infrastructure for a Cluster ... succeeded
```

You should see the "succeeded" message if CRS has been configured correctly on node 2.

Like you did for Node 1, we must also create a cluster service to handle moving the EC2 IP addresses within the VPC when CRS moves the VIP. We also must change the existing EC2 IP address services for the node 1 VIP and the SCAN addresses so that they include the full list of nodes. As the oracle user account on node 1, source your environment for the +ASM1 instance, and run the commands:

```bash
ec2-user@myracnode01 $ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] +ASM1
oracle@myracnode01$ crsctl add resource myrac02-vip.ec2 -type cluster_resource -attr "CHECK_INTERVAL=10,ACTION_SCRIPT=/etc/oracle/ec2_vip_failover.sh,PLACEMENT=favored,HOSTING_MEMBERS=myracnode01,myracnode02,AUTO_START=always,START_DEPENDENCIES=hard(intermediate:ora.myracnode02.vip),RELOCATE_BY_DEPENDENCY=1,"" 
```

Check /u01/app/12.1.0/grid/install/root_myracnode02_2015-09-28_17-02.log for the output of crsctl add resource command.
Next you will copy the Oracle database installation from the first RAC node to the second. The following command uses a pipeline of the tar and ssh utilities to copy /u01/app/oracle/product/ to myracnode02.

Command:
```
oracle@myracnode01$ tar czvf -/u01/app/oracle/product | ssh myracnode02 "tar xzvf - -C /
```

Sample output:
When the copy is complete, log into the second node, and complete installation of the software on that host.

The following commands log into the second node, register the software with the local Oracle inventory database, run the root scripts, set the environment for the database home, and relink the binaries.

Commands:
```
ec2-user@myracnode02$ sudo su - oracle
oracle@myracnode02$ . oraenv
ORACLE_SID = [oracle] ?
ORACLE_HOME = /home/oracle? /u01/app/oracle/product/12.1.0/dbhome_1 ORACLE_HOME_NAME="dbhome_1" ORACLE_BASE="/u01/app/oracle"
```

Sample output:
Starting Oracle Universal Installer...
The cloning of dbhome_1 was successful.
Please check '/u01/app/oraInventory/logs/cloneActions2015-10-01_10-39-34PM.log' for more details.

Command:
```
ec2-user@myracnode02$ sudo /u01/app/oracle/product/12.1.0/dbhome_1/root.sh
```

Sample output:
Check /u01/app/oracle/product/12.1.0/dbhome_1/install/root/myracnode02_2015-10-01_22-57-11.log for...

Commands:
```
e2-user@myracnode02$ sudo su - oracle
e2-user@myracnode02$ /u01/app/oracle/product/12.1.0/sqldeveloper/sqldeveloper/lib/commons-logging-1.1.1.jar
```

Sample output:
```
Command:
```
chmod 755 /u01/app/oracle/product/12.1.0/dbhome_1/bin
```

Sample output:
```
crsctl modify resource myracnode01-vip.ec2 -attr "CHECK_INTERVAL=10, ACTION_SCRIPT="/etc/oracle/ec2_vip.failover.sh", PLACEMENT=favored, HOSTING_MEMBERS=myracnode01 myracnode02 ORACLE_HOME="/u01/app/oracle/product/12.1.0/dbhome_1", ORACLE_BASE="/u01/app/oracle" AUTO_START=always, START_DEPENDENCIES="hard(intermediate:ora.myracnode01.vip)", STOP_DEPENDENCIES="hard(intermediate:ora.myracnode02.vip)"
```

Next you will copy the Oracle database installation from the first RAC node to the second. The following command uses a pipeline of the tar and ssh utilities to copy /u01/app/oracle/product/ to myracnode02.

Command:
```
oracle@myracnode01$ tar czvf -/u01/app/oracle/product | ssh myracnode02 "tar xzvf - -C /
```

Sample output:
Finally, you can set the database parameters for associating the RAC instances with specific instance numbers and for allowing the instances to communicate. While logged into the first RAC node, run the provided script to set the Oracle database parameters:

```
... chmod 6751 /u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
```

```
Commands:

```
ec2-user@myracnode01$ sudo su - oracle
oracle@myracnode01$ . oraenv
ORACLE_SID = [oracle] ? orcl
oracle@myracnode01$ export ORACLE_SID=orcl1
oracle@myracnode01$ sqlplus / as sysdba @db_parameters
```

To allow the database to take the new parameter values, and to add the second instance to the cluster, you must issue the following `srvctl` commands: Commands:

```
oracle@myracnode01$ srvctl add instance -d orcl -i orcl2 -n myracnode02
oracle@myracnode01$ srvctl start database -d orcl
oracle@myracnode01$ srvctl status database -d orcl
```

**Conclusion**

At this point, you should have a running two-node RAC cluster. To add additional nodes, add the addresses to Route 53 using the JSON file from Step 2 and repeat the procedure for the second node, incrementing the ORDINAL value in the user-data script. You can launch up to a total of nine nodes without modifying the AMIs.

**Step 15: (Optional / Advanced) Configuring Flash Cache**

The provided AMI presents the majority of the c3.8xlarge's 640G of local ephemeral SSD as a logical volume suitable for use with Oracle's Flash Cache feature. The volume is exposed on the instance nodes as /dev/racvg/flashcachelv. Unfortunately, there is a problem with the feature validation for Flash Cache on Oracle Linux 7, and you must apply patch 19504946 from support.oracle.com to the database homes on all nodes where you wish to use flash cache. You will also need to modify the UDEV rules to set the ownership of this volume to oracle:oinstall. Once those prerequisites are completed, you can enable flash cache with the following parameter changes:

```
Commands:

```
SQL> alter system set db_flash_cache_file = '/dev/racvg/flashcachelv' sid='*' scope=sppfile;
SQL> alter system set db_cache_size = 600G sid='*' scope=sppfile;
oracle@myracnode01$ srvctl stop database -d orcl
oracle@myracnode01$ srvctl start database -d orcl
```

**Frequently Asked Questions**

- Will Oracle support RAC on EC2?
- Ntop states that their N2N project is "on hold." Should I be nervous about using an architecture which includes N2N in production?
- Is the Oracle RAC cluster resulting from this tutorial suitable for production use?
- The architecture presented in this tutorial does not provide protection across availability zones. Should I consider this nevertheless to be a high-availability design?
- How can I tell if Oracle RAC on EC2 will provide sufficient capacity for my workloads?
- What about backups?

Q: Will Oracle support RAC on EC2?

A: Oracle's support policy for running Oracle on AWS is in [Oracle support note 1901155.1](https://support.oracle.com/knowledge/technologies/running-oracle-on-ec2.html) (access requires an Oracle Support account), and is similar to their policy regarding non-Oracle VM technologies. In short, they support the Oracle database software on a variety of hypervisors, but will refer you to the VM-vendor (AWS in this case) if they believe the problem is virtualization-related. In addition, using virtual networks for interconnect is standard practice and the subject of several Oracle-authored whitepapers. AWS has not sought from Oracle official certification of AWS in general, or this RAC solution specifically.

Q: Ntop states that their N2N project is "on hold." Should I be nervous about using an architecture which includes N2N in production?

A: Per Ntop, N2N is not at end of life or deprecated, but on hold until they get more time. Because it is an open source tool, some customers may be satisfied with having the source code, and being able make any updates or changes the need. However, customers who choose not to use N2N will find many other VPN products that could be set up to support an overlay multicast VPN.

Q: Is the Oracle RAC cluster resulting from this tutorial suitable for production use?

A: You should use this tutorial to guide you in creating the proper architecture for your own needs. The tutorial provides guidance for a basic RAC architecture, and incorporates a number of features, such as redundant storage, that would be typical in production use cases. However, we encourage you to architect the solution to your own needs, and test performance, availability and recoverability before deploying any database architecture in production.

Q: The architecture presented in this tutorial does not provide protection across availability zones. Should I consider this nevertheless to be a high-availability design?

A: Oracle customers generally follow a broad set of principles called "Maximum Availability Architecture" (MAA) to guarantee availability and recoverability. Under those guidelines, Oracle customers typically implement RAC for local instance and host-level failures, but not for datacenter or region-level failures. Data Guard, a separate technology,
provides protection at the datacenter and region level. Data Guard logs changes synchronously or nearly synchronously to a second system located locally or in a nearby datacenter, and optionally downstream to a remote region. If you want to add Multi-AZ or region-level protection, you may wish to explore adding Data Guard to your design.

Q: How can I tell if Oracle RAC on EC2 will provide sufficient capacity for my workloads?

A: Compare your demand as recorded in AWR (IOPS both read and write), CPU, etc. to the capabilities of the RAC solution you architect on AWS, understanding that you must provide headroom for spikes and future growth.

Q: What about backups?

A: Backup and recovery automation is excluded from the tutorial, and is your responsibility to design. The tutorial's iSCSI target AMIs do present the underlying storage as Linux LVMs, which support copy-on-write snapshots. That means that you can back up a database stored entirely on a set of storage servers by placing the database in backup mode, and taking an LVM snapshot of the volume storing the DATA disk group on one of the mirrored iSCSI servers. You can then copy the snapshot off the S3 for persistent storage. Alternatively, you could use RMAN to back up the database block by block to Amazon S3. Oracle Secure Backup enables Oracle RMAN backup to S3, as does s3fs. Any backup solution must be rigorously tested both for functionality, RTO and RPO under a variety of failure scenarios.