VERITAS™

High Availability Configuration in AWS Cloud using InfoScale Enterprise

Who should read this paper

Administrators who want to implement HA architectures for unmanaged application services in the cloud

Table of Contents

Introduction

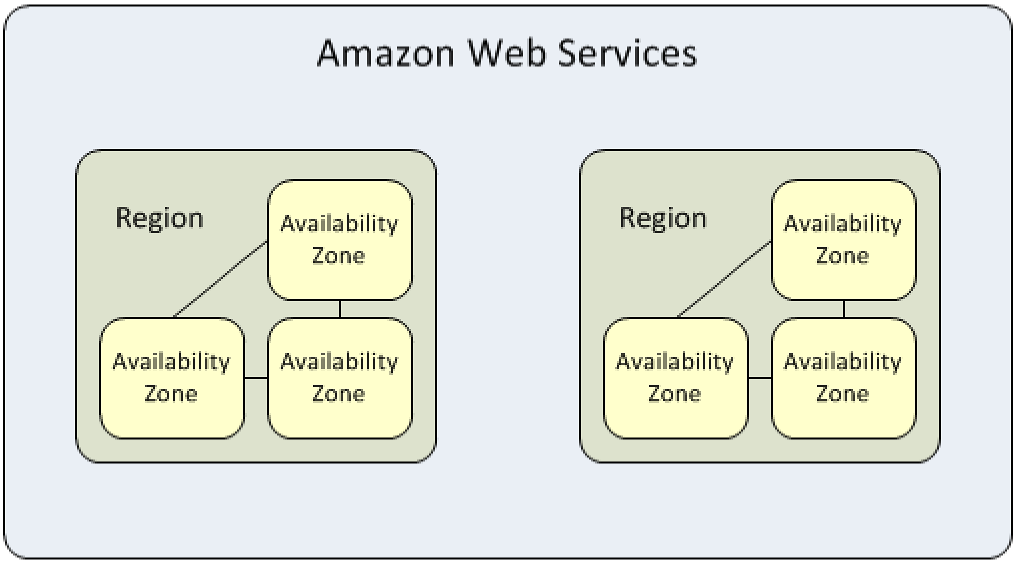
The Amazon Web Services (AWS) cloud has become ubiquitous and organizations are deploying traditional as well as cloud-native workloads on AWS. AWS provides a native RDS (Relational Database Service) solution for relational databases, which is a managed database service. RDS supports common databases like MySQL, SQL Server, Oracle, and others with built-in high availability (HA). However, RDS does not let users control the operating system (OS) environment in which a database runs. When users have fine control over the OS, they can perform advanced tuning for the databases that they control. Similarly, RDS supports typical configurations, but certain kinds of advanced configurations may not be possible.

As an alternative to RDS, customers have the option of deploying a database in an EC2 compute node environment. For more information, see “References” on page 10. Such configurations are called unmanaged configurations, where the end users are responsible for the availability of the database. In this document, we discuss an HA architecture for a database in an unmanaged EC2 environment using Veritas InfoScale Enterprise and AWS Elastic Load Balancer (ELB). The architecture is typical of HA configurations in the AWS cloud. While this document focuses on Oracle, the general idea can be extended to any application supported by InfoScale Enterprise.

We discuss setting up Oracle in AWS in a multi-availability-zone (multi-AZ) for HA. The configuration comprises an Elastic Load Balancer for routing traffic to the active Oracle Node and performs a failover of the network traffic from clients when the underlying database instance fails over. InfoScale Enterprise features like Cluster Server (VCS) and Volume Replicator (VVR) are used to provide HA and synchronous replication respectively. Coordination Point (CP) Server-based fencing is deployed. This configuration protects against application/server failures and also failure of an entire AZ.

Note: The configurations mentioned in this document have been tested using small instances. For actual production-quality performance, make sure to use appropriate-sized instances and to provision the required network bandwidth from AWS.

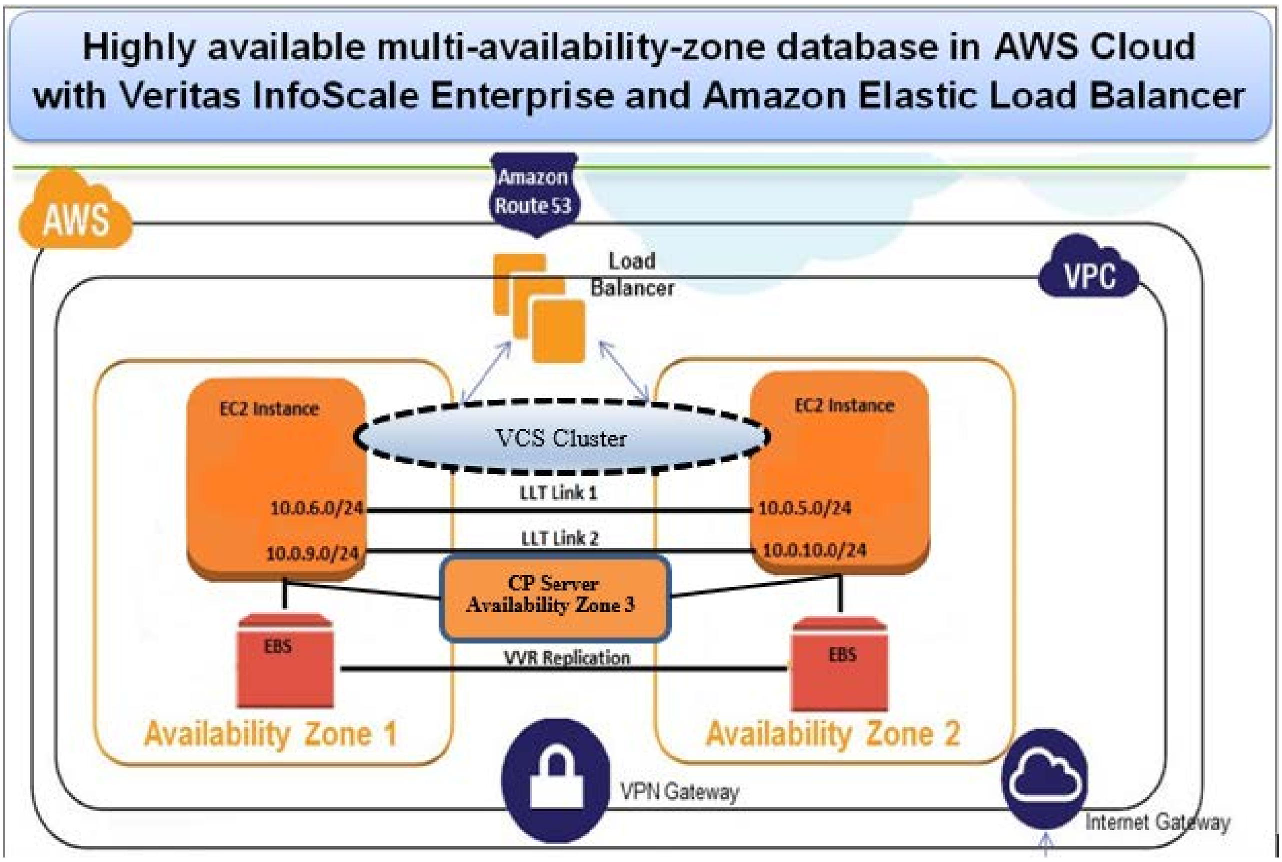
AWS terminology



Term	Description
Region	A region is a separate geographic area, for example: us-east.
Availability Zone	Each region has multiple, isolated locations known as availability zones, which are connected by high-speed, redundant, low-latency links. For example: us-east-1a.
Virtual Private Cloud (VPC)	Amazon VPC lets you provision a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define.
Subnet	After creating a VPC, you add one or more subnets in each availability zone. When you create a subnet, you specify the CIDR block for the subnet, which is a subset of the VPC CIDR block. Each subnet must reside entirely within one availability zone and cannot span zones.
Elastic Load Balancer (ELB)	Elastic load balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.
Elastic Network Interface (ENI)	An ENI is a virtual network interface that you can attach to an instance in a VPC. ENIs are available only for instances running in a VPC.

Configuration

The following graphic depicts a cluster configuration with two VCS nodes running on Amazon EC2 instance in different availability zones, AZ1 and AZ2:



- The EC2 instances have Oracle databases installed and running, which are managed by VCS. Both the EC2 instances are in a private subnet with no access to the internet.
- Each EC2 instance has two network interfaces, eth0 and eth1. The EC2 instance in AZ1 has subnet 10.0.6.0/24 on eth0 and subnet 10.0.9.0/24 on eth1. Similarly, the EC2 instance on AZ2 has subnet 10.0.5.0/24 on eth0 and subnet 10.0.10.0/24 on eth1.
- Because eth1 on both the instances are on different subnets, they are not reachable via the default gateway which is on eth0. To make them reachable, we need to define a new routing table for eth1 on both the instances.

  The following procedure illustrates how to achieve such a configuration using a Linux instance:

  1. Create a routing table for your new device:

     ```
     # echo 2 eth1_rt >> /etc/iproute2/rt_tables
     ```

  2. Add the eth1_rt route table, so that it also points to the gateway by default:

     ```
     # ip route add default via <default gateway for eth1> dev eth1 table
     eth1_rt
     ```

Example:

```
# ip route add default via 10.0.9.1 dev eth1 table eth1_rt
```

3. Add a rule, to address the situation when there is traffic from the eth1 IP address:

   ```
   # ip rule add from <IP address of eth1> lookup eth1_rt prio 1000
   ```

   Example:

   ```
   # ip rule add from 10.0.9.178 lookup eth1_rt prio 1000
   ```

4. Add destination-specific route so that all the traffic to subnet 10.0.10.0/24 goes through eth1:

   ```
   # ip rule add to <subnet of destination interface> lookup eth1_rt prio
   1000
   ```

   Example:

   ```
   # ip rule add to 10.0.10.0/24 lookup eth1_rt prio 1000
   ```

5. The previous commands let you create temporary rules and routes. To make them persistent across system restarts, create files for each interface.

   Sample rule file content:

   ```
   # cat /etc/sysconfig/network-scripts/rule-eth1
   from  10.0.9.178 lookup eth1_rt prio 1000
   to 10.0.10.0/24  lookup eth1_rt prio 1000
   ```

   Sample route file content:

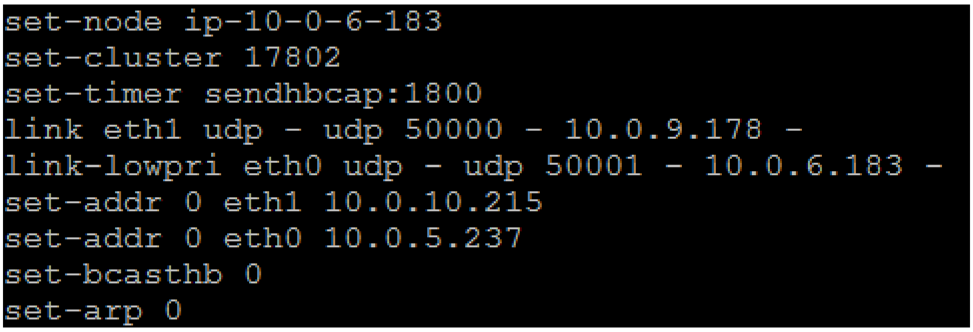
   ```
   # cat /etc/sysconfig/network-scripts/route-eth1
   default via 10.0.9.1 dev eth1 table eth1_rt
   ```

   Note: The examples pertain to RHEL 7. You may create equivalent files for other platforms.

Perform this procedure on the other EC2 instance too.

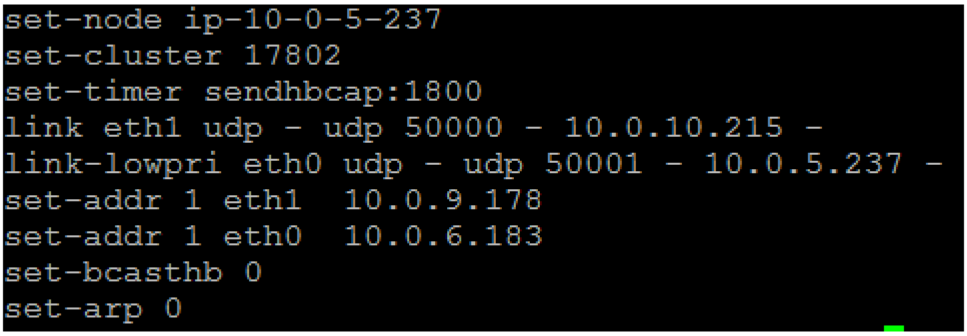
- The LLT links are configured to use both these subnets as a part of the intra-cluster communication.

  The following graphic depicts a sample /etc/llttab file for the EC2 instance in AZ1:



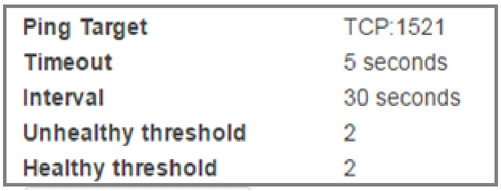
```
set-node ip-10-0-6-183
set-cluster 17802
set-timer sendhbcap:1800
link eth1 udp - udp 50000 - 10.0.9.178 -
link-lowpri eth0 udp - udp 50001 - 10.0.6.183 -
set-addr 0 eth1 10.0.10.215
set-addr 0 eth0 10.0.5.237
set-bcasthb 0
set-arp 0
```

The following graphic depicts a sample /etc/llttab file for the EC2 instance in AZ2:



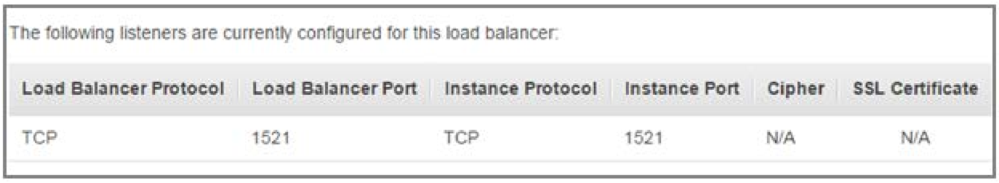
```
set-node ip-10-0-5-237
set-cluster 17802
set-timer sendhbcap:1800
link eth1 udp - udp 50000 - 10.0.10.215 -
link-lowpri eth0 udp - udp 50001 - 10.0.5.237 -
set-addr 1 eth1  10.0.9.178
set-addr 1 eth0  10.0.6.183
set-bcasthb 0
set-arp 0
```

- Two EBS volumes are attached, one to each EC2 instance, which act as the data volumes for the Oracle database. The volume is replicated using VVR synchronous replication.
- A CP server is configured in a third availability zone, which takes care of fencing in case there is a network partition between the two cluster nodes.
- The ELB is used to redirect traffic to the EC2 instance where the Oracle service group is running. To achieve this, the ELB is configured to do health check on the TCP port 1521 on which the Oracle listener is listening. The following graphic illustrates this configuration:



Ping Target	TCP:1521
Timeout	5 seconds
Interval	30 seconds
Unhealthy threshold	2
Healthy threshold	2

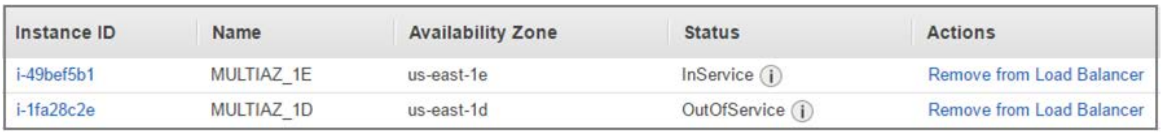
The following graphic depicts the listener that is configured for the ELB:



The following listeners are currently configured for this load balancer:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
TCP	1521	TCP	1521	N/A	N/A

The following graphic illustrates how the ELB is configured for two instances (in a two-node VCS cluster):



Instance ID	Name	Availability Zone	Status	Actions
i-49bef5b1	MULTIAZ_1E	us-east-1e	InService ⓘ	Remove from Load Balancer
i-1fa28c2e	MULTIAZ_1D	us-east-1d	OutOfService ⓘ	Remove from Load Balancer

The status of one of the instances is OutOfService, because the Oracle service group is a failover group, which can be online on any one instance.

CP server configuration

The coordination point server (CP server) is a software solution that runs on a remote system or cluster. CP server provides arbitration functionality by allowing the VCS cluster nodes to perform the following tasks:

- Self-register to become a member of an active VCS cluster (registered with CP server) with access to the data drives.
- Check which other nodes are registered as members of this active VCS cluster.
- Self-unregister from this active VCS cluster.
- Forcefully unregister other nodes (preempt) as members of this active VCS cluster.

In short, the CP server functions as another arbitration mechanism that integrates within the existing I/O fencing module. Note that the CP server depicted in the graphic in the "Configuration" section is in a different availability zone.

You can configure a CP server by invoking the `/opt/VRTS/install/installer -configcps` command.

Sample VCS configuration file (main.cf)

```
include "types.cf"
include "OracleTypes.cf"

cluster AWS_AGENTS (
      UserNames = { admin = dqrPqnQn }
      Administrators = { admin }
      UseFence = SCSI3
      HacliUserLevel = COMMANDROOT
      )

system ip-10-0-5-237 (
      )

system ip-10-0-6-183 (
      )

group VVR (
      SystemList = { ip-10-0-5-237 = 0, ip-10-0-6-183 = 1 }
      Parallel = 2
      AutoStartList = { ip-10-0-5-237, ip-10-0-6-183 }
      SystemZones = { ip-10-0-6-183 = 0, ip-10-0-5-237 = 1 }
      )

DiskGroup dgres (
      DiskGroup = oradg
      )

IP ipres (
      Device = eth0
      Address @ip-10-0-5-237 = "10.0.5.238"
      Address @ip-10-0-6-183 = "10.0.6.184"
      NetMask = "255.255.254.0"
      )
```

```
NIC nicres (
      Device = eth0
      )

RVG rvgres (
      RVG = rvg
      DiskGroup = oradg
      )

ipres requires nicres
rvgres requires dgres
rvgres requires ipres

// resource dependency tree
//
//    group VVR
//    {
//          RVG rvgres
//          {
//                DiskGroup dgres
//                IP ipres
//                {
//                      NIC nicres
//                }
//          }
//    }

group testgrp (
      SystemList = { ip-10-0-5-237 = 0, ip-10-0-6-183 = 1 }
      )

Mount mountres (
      MountPoint = "/u01"
      BlockDevice = "/dev/vx/dsk/oradg/oravol"
      FSType = vxfs
      FsckOpt = "-y"
      )

Netlsnr netlsnr (
      Owner = oracle
      Home = "/u01/app/oracle/product/11.2.0/dbhome_1"
      )

Oracle orares (
      Sid = orclaws
      Owner = oracle
      Home = "/u01/app/oracle/product/11.2.0/dbhome_1"
      User = vcsuser
      Pword = drjPgrGjeJfjG
      Table = vcs
      )
```

```
RVGPrimary rvgp (
      RvgResourceName = rvgres
      )

requires group VVR online local hard
mountres requires rvgp
netlsnr requires orares
orares requires mountres

// resource dependency tree
//
//    group testgrp
//    {
//          Netlsnr netlsnr
//          {
//                Oracle orares
//                {
//                      Mount mountres
//                      {
//                            RVGPrimary rvgp
//                      }
//                }
//          }
//    }

group vxfen (
      SystemList = { ip-10-0-5-237 = 0, ip-10-0-6-183 = 1 }
      AutoFailOver = 0
      Parallel = 1
      )

CoordPoint coordpoint (
      ActionOnCoordPointFault = RefreshRegistrations
      )

RES_phantom_vxfen (
      )

// resource dependency tree
//
//    group vxfen
//    {
//          Phantom RES_phantom_vxfen
//          CoordPoint coordpoint
//    }
```

References

https://d0.awsstatic.com/whitepapers/best-practices-for-running-oracle-database-on-aws.pdf

About Veritas Technologies LLC

Veritas Technologies LLC enables organizations to harness the power of their information, with solutions designed to serve the world's largest and most complex heterogeneous environments. Veritas works with 86 percent of Fortune 500 companies today, improving data availability and revealing insights to drive competitive advantage.

For specific country offices
and contact numbers,
please visit our website.

Veritas World Headquarters
500 East Middlefield Road
Mountain View, CA 94043
+1 (650) 933 1000
www.veritas.com