



Oracle Database 11g on Amazon EC2 Implementation Guide

Abdul Sathar Sait

November 2013

(Please consult <http://aws.amazon.com/whitepapers/> for the latest version of this guide.)

Table of Contents

Abstract	3
Introduction	3
AWS and Oracle Technologies Used In the Architecture Scenarios.....	4
Key AWS Services Used in the Architecture Scenarios	4
Key Oracle Technologies Used in the Architecture Scenarios.....	4
Oracle Database on Amazon EC2: Reference Architecture Scenarios	5
Standard Architecture Scenario	5
Enterprise Class Architecture Scenario	5
Large Enterprise Class Architecture Scenario	6
High Performance Architecture Scenario.....	6
Reference Architectures at a Glance.....	8
Implementation of Oracle Database Architecture Scenarios on Amazon EC2	9
Standard Database Architecture Scenario Implementation	9
Enterprise Class Database Architecture Scenario Implementation	17
Large Enterprise Class Database Architecture Scenario Implementation	20
High Performance Database Architecture Scenario Implementation	25
Launch Oracle Database Architecture Scenario.....	31
Prerequisites for Launching the Scenarios.....	31
Startup Tasks Common to All Four Architecture Scenarios	32
Conclusion	37
References.....	38
Appendix: Install Oracle Secure Backup Module	39

Abstract

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying enterprise-grade solutions in a rapid, reliable, and cost effective manner. Oracle database is a widely used relational database management system that is deployed in enterprises of all sizes to run enterprise resource planning (ERP) applications, reporting systems, business intelligence and analytics applications, ecommerce systems, repositories, and data warehouses.

AWS offers its customers the flexibility to run Oracle database on Amazon Relational Database Service (Amazon RDS), the fully managed database service in the cloud, as well as Amazon Elastic Compute Cloud (Amazon EC2). While each option has its distinct advantages, in this guide, we dive deeper into Oracle database on EC2 option. This implementation guide is targeted at enterprise solution architects and database administrators and discusses four typical architecture scenarios. It includes implementation artifacts in the form of AWS CloudFormation templates so you can customize based on your requirements and quickly and reliably deploy Oracle database 11g on AWS.

Introduction

Most customers prefer to use Amazon Relational Database Service (RDS) for Oracle as it provides an easy, managed option to running an Oracle database on AWS without you having to think about infrastructure provisioning or installing and maintaining database software. Alternatively, you can run Oracle database directly on Amazon EC2, which allows you full control over setup of the entire infrastructure and database environment. This option provides a familiar approach to running Oracle database on AWS, but also requires you to appropriately set up, configure, manage, and tune all the components, such as Amazon EC2 Instances, volumes, scalability networking, security, as appropriate (based on AWS architecture best practices). For more information, please refer to parent document “[RDBMS in the Cloud: Oracle on AWS](#)”, which will give you a good technical understanding of the details and advantages of each option to running Oracle database on AWS. It reviews in detail the different approaches to provision and monitor your Oracle database, and how to manage scalability, performance, backup and recovery, high availability and security in both Amazon RDS and Amazon EC2.

This implementation guide focuses on the ‘Oracle database on EC2’ option. It provides best practices guidance and resources for architecting and implementing a set of characteristic reference scenarios of deploying Oracle database on Amazon EC2. This guide describes a set of common architectural scenarios with varying degrees of performance, scale, high availability, and disaster recovery balanced against different cost profiles for common Oracle database use cases and success criteria. The guide also includes AWS CloudFormation sample templates for each scenario that allows you to automate and rapid deploy the necessary cloud resources of each of the architecture scenarios in a reliable way.

AWS and Oracle Technologies Used In the Architecture Scenarios

Before jumping into the architecture scenarios and details, let's review the key AWS and Oracle technologies used in these architecture scenarios. The aforementioned "[RDBMS in the Cloud: Oracle on AWS](#)" whitepaper provides a detailed explanation of each of these, their details, options, etc. but they are mentioned here as well to provide a background for the scenario details to follow.

Key AWS Services Used in the Architecture Scenarios

Key AWS services used within the architecture scenarios include:

- [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) – provides the core elastic compute service for running the Oracle database server(s). EC2 provides a variety of instance types that accommodate Oracle database resource requirements for CPU, RAM, storage, and networking.
- [Amazon Elastic Block Store \(Amazon EBS\)](#) – provides highly available and reliable block level storage volumes for use with Amazon EC2 instances. Amazon EBS provides two volume types – Standard, and PIOPS
 - **Standard EBS volumes** - offer persistent storage for applications with moderate or bursty I/O requirements.
 - [Provisioned IOPS EBS volumes](#) – are designed to deliver predictable high performance for I/O intensive workloads, such as database applications, which rely on consistent and fast response times.
 - [Amazon EBS-optimized EC2 instance types](#) - provides a configuration that is optimized for use with PIOPS Amazon EBS volumes, providing dedicated network capacity for communicating between the Amazon EC2 instance and associated PIOPS Amazon EBS volume.
- [Amazon Virtual Private Cloud \(Amazon VPC\)](#) – a logically isolated section of AWS virtual network that provides complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.
- [Amazon Simple Storage Service \(Amazon S3\)](#) – provides object storage for EBS snapshots.
- [AWS CloudFormation](#) – use JSON format declarative code that automates the instantiation of the AWS resources that make up the scenarios.

Key Oracle Technologies Used in the Architecture Scenarios

The architecture scenarios also make use of the following Oracle technologies, in addition to Oracle database.

- **Oracle Automatic Storage Management (ASM)** - a volume manager and a file system for Oracle database that provides an alternative to conventional volume managers, file systems, and raw devices. You can use it to stripe multiple Amazon EBS volumes together to take advantage of cumulative IOPS and to enable larger disk volumes needed for the database files.
- **Oracle Secure Backup Cloud Module.** - enables an Oracle database to send its backups to Amazon S3 using the Oracle Recovery Manager (RMAN) SBT interface. This allows external backup libraries to be seamlessly integrated with RMAN, so that existing backup tools – Enterprise Manager, RMAN and other scripts, etc. – can be used to perform cloud backups.
- **Oracle Data Guard and Active Data Guard.** Oracle Data Guard to achieve high availability, data protection, and disaster recovery for enterprise data. Oracle Active Data Guard provides the management, monitoring, and automation software to create and maintain one or more synchronized replicas (standby databases) of a production database (primary database). An Active Data Guard standby database is an exact copy of the primary database that is read-only while it continuously applies changes transmitted by the primary database.

Oracle Database on Amazon EC2: Reference Architecture Scenarios

This section introduces and describes a set of typical scenarios for running Oracle database on Amazon EC2. There are four distinct Oracle database architecture scenarios described here. Each is designed to emphasize a particular combination of business requirements and tradeoffs to accommodate distinct solution types, sizes, and cost profiles. The scenarios are described below in sequence, beginning with the most basic (and lowest cost) and with each subsequent scenario building on the one that precedes it.

Standard Architecture Scenario

This architecture scenario addresses general-purpose database deployment. It emphasizes good performance, optimized for lower overall cost. Oracle database deployments such as dev/test environments, small production application deployments, custom projects, and reporting systems are most applicable for this scenario.

- **Scalability.** This architecture scenario is highly scalable, and you can migrate it to one of the following architecture scenarios if your requirements change. This scenario is scalable in three dimensions: architectural, instance size, and database size. You can start with a small Amazon EC2 instance, such as m1.small, and scale up to the largest instance type available as the needs change. You do this by stopping the database instance and restarting it with a new instance type. Database size can also be increased as needed, up to 1 TB. You can get started with a small size database and scale up the database size as the amount of data increases. To do this, you create a snapshot of the existing Amazon EBS volume used for data files and restore the snapshot into a new larger volume attached to the instance.
- **Size.** This architecture scenario is suitable for database sizes 10 GB to 1 TB.
- **Performance:** This architecture scenario is designed for moderate overall database performance, but you can make it perform better by choosing larger Amazon EC2 instance types to run the database. This architecture scenario provides moderate I/O performance of 100-4000 IOPS.
- **High Availability.** To keep the cost low, this architecture scenario does not incorporate any specific database-level high availability (HA), such as secondary/failover database instances, etc.
- **Disaster Recovery.** This architecture scenario does include disaster recovery (DR) but in order to yield low cost, disaster recovery for this architecture scenario involves a manual process, providing a Recovery Time Objective (RTO) that is longer than the other three scenarios. Amazon EBS snapshots of the backup volume are copied to a different region using the snapshot copy API. A new database instance can be set up manually from the snapshots.
- **Security.** This architecture scenario is designed to provide a secure, isolated environment using Amazon VPC, and instantiates the database Amazon EC2 instance within a private subnet, with applicable Amazon EC2 security group rules.

Enterprise Class Architecture Scenario

This architecture scenario is designed for applications that require enterprise class performance, reliability, high availability, and fault tolerance, that is, for systems that require better performance and larger database size than the Standard Architecture Scenario.

- **Scalability.** This architecture scenario is highly scalable, providing the ability to start small and scale out to larger capacity as load dictates. You can also migrate this scenario to one of the following architecture scenarios if your requirements increase over time. Database size is highly scalable as well. Since the storage used is a striped set of Amazon EBS volumes managed by Oracle ASM, you can easily add Amazon EBS volumes to achieve larger capacity as the database grows.

- **Size.** This architecture scenario is suitable for database sizes 500 GB to 5 TB.
- **Performance.** This architecture scenario is designed to provide a moderate to high level of instance and I/O performance (2500 to 10,000 IOPS).
- **High Availability.** High availability is supported by setting up a secondary database server in another Availability Zone, and using Oracle Data Guard to facilitate data synchronization. Data Guard requires using Oracle Database Enterprise Edition. This same architecture scenario can be used for Oracle Database Standard or Standard One Editions if you use third-party tools that provide similar functionality to Oracle Data Guard. Note that the sample code provided uses Oracle Data Guard.
- **Disaster Recovery.** This architecture scenario provides the ability to recover from an AWS region-wide event. Database data is backed up to another AWS region using OSB and RMAN. In this scenario the disaster recovery process is manual: we provide scripts that manually trigger the creation of a database Amazon EC2 instance from the AMI and restore database data from the RMAN backup in the other region. The timing and state of the recovered database (RPO) depends on the backup frequency.
- **Security.** As in the previous scenario, isolation is provided using Amazon VPC, and database Amazon EC2 instances instantiated in private subnets, with applicable security group rules. You can use Oracle Transparent Data Encryption (TD) to make use of encryption at rest.

Large Enterprise Class Architecture Scenario

This architecture scenario builds on the previous scenario, distributing load across multiple Availability Zones by using Oracle Active Data Guard. It also uses a “Pilot Light” Disaster Recovery pattern for quicker recovery in case of a DR event. The emphasis in this design is on enterprise-level reliability, availability, and performance. This architecture scenario supports almost any database use scenario in an enterprise.

- **Scalability.** This architecture scenario is scalable both in terms of database size (storage) and performance (larger instances and more IOPS).
- **Size.** This architecture scenario is suitable for database sizes 2 TB to 20 TB.
- **Performance.** This architecture scenario is designed to provide a high level of instance performance and I/O performance (8000-40,000 IOPS).
- **Availability.** High availability is implemented using Oracle Active Data Guard in a different Availability Zone. Unlike Data Guard, Active Data Guard allows load distribution by using the replicated instance for read only loads. This architecture scenario allows for multiple read replicas if needed, increasing availability and load distribution, which makes the architecture more scalable.
- **Disaster Recovery.** A “Pilot Light” Disaster Recovery Scenario is used in this architecture, which keeps the cost low while providing very quick failover with minimal or no data loss. (For more details on disaster recovery scenarios, refer to the [Disaster Recovery whitepaper](#).)
- **Security.** This architecture scenario provides a secure, isolated environment using Amazon VPC, and instantiates the database Amazon EC2 instance within a private subnet, with applicable Amazon EC2 security group rules. You can use Oracle Transparent Data Encryption (TD) to make use of encryption at rest.

High Performance Architecture Scenario

This architecture scenario is designed primarily for high I/O performance. It leverages local (ephemeral) SSD drives to take advantage of very high IOPS and avoid network latency. Ephemeral instance storage is used for the database along with replication to another instance in the same Availability Zone, based on Amazon EBS volumes for data safety and reliability. If the primary instance fails, this architecture rapidly switches over to the second instance, making the database continuously available and preventing data loss.

- **Scalability.** Scalability of this architecture scenario is limited because it can be used only in a Hi I/O instance type and the database size is limited to the maximum size of instance storage available (currently 2 TB).
- **Size.** This architecture is suitable for database sizes 5 GB to 2 TB.
- **Performance.** This architecture scenario leverages Amazon EC2 ephemeral SSD disks to take advantage of very high IOPS and reduced network latency (up to 200,000 IOPS).
- **High Availability.** High availability is implemented using two Oracle Active Data Guard instances, one in the same Availability Zone as the primary database, and another in a different Availability Zone. This is also done for reliability: the primary instance is based on volatile instance storage; replicated instances are based on Amazon EBS storage. Oracle Active Data Guard also allows load distribution by using the replicated instance for read only loads.
- **Disaster Recovery.** A “Pilot Light” Disaster Recovery Scenario is used in this architecture, which keeps the cost low while providing very quick failover with minimal or no data loss.
- **Security.** This architecture scenario is designed to provide a secure, isolated environment using Amazon VPC, and instantiates the database Amazon EC2 instance within a private subnet, with applicable Amazon EC2 security group rules. You can use Oracle Transparent Data Encryption (TD) to make use of encryption at rest.

This implementation guide explains how each architecture scenario maps to one of the provided AWS CloudFormation sample templates and the rationale behind how the template is structured. It then provides a walkthrough of each template and explains how to customize the templates to suit your specific requirements.

Reference Architectures at a Glance

Table 1 shows a quick comparison of the four architecture scenarios discussed in the above section

Architecture Scenario	Standard	Enterprise Class	Large Enterprise Class	High Performance
Recommended Database Size	10GB - 1 TB	500 GB - 5 TB	2 TB - 20 TB	5 GB - 2 TB
I/O Performance	Moderate (500-1000 IOPS)	Moderately High (2500-10,000 IOPS)	High (8000 to 20,000 IOPS)	Very High (Up to 200,000 IOPS)
High Availability	None	AZ Failover (Seconds to couple of minutes)	Same AZ Switchover and Multi-AZ Failover (Seconds)	Multi-AZ Failover
Replication	None	Oracle Data Guard	Oracle Active Data Guard	Oracle Active Data Guard
Load Distribution	None	None	Active Data Guard Read Replicas	Active Data Guard Read Replicas
Backup	RMAN to Disk then Snapshot to Amazon S3	RMAN + OSB to Amazon S3	RMAN + OSB to Amazon S3	RMAN + OSB to Amazon S3
DR	Cross Region Amazon S3 Backup Using Amazon EBS Snapshot Copy	Cross Region Amazon S3 Backup Using RMAN+OSB	Pilot Light DR	Pilot Light DR
Scalability	Moderate		High	Low
Recommended Instance Types	m1.large m1.xlarge m2.xlarge m2.2xlarge m2.4xlarge	m2.2xlarge m2.4xlarge	m2.4xlarge hi1.4xlarge hs1.8xlarge	hi1.4xlarge
Recommended Oracle Database Edition	Standard/Standard One	Enterprise	Enterprise	Enterprise
Infrastructure Cost	\$\$\$	\$\$\$\$	\$\$\$\$\$	\$\$\$\$

Table 1: Quick comparison of Four Reference Architecture Scenarios (With Sample Cost Estimate)

Actual cost of implementation varies based on the specific implementation. Cost calculation provided for the architectures in the table above is based on a typical environment. Click a \$ link in the final row to go to the Simple Monthly Calculator to modify the parameters to calculate the cost more appropriately for your specific scenarios.

Implementation of Oracle Database Architecture Scenarios on Amazon EC2

This section describes the details of each architecture scenario and the template that you can use to deploy the architecture. We'll discuss the details of each template and where you might consider customizing it.

We recommend that you read through each architecture scenario before you begin working with the sample templates. For information about how to work with the templates, see the Implementation section later in this guide.

Standard Database Architecture Scenario Implementation

This architecture scenario is best suited for repositories, custom applications, reporting, development and test environments, and small size production applications. It emphasizes low cost.

The Standard database architecture has two variations, one using Amazon EBS snapshot, the other using Oracle Secure Backup Cloud Module for backing up data to Amazon S3.

This scenario leverages the following key AWS resources:

- Amazon VPC provides a private, isolated environment to hold and run the Oracle database.
- An Amazon EC2 instance to host the Oracle database. For this scenario, applicable Amazon EC2 instance types range from m1.large to m2.4xlarge.
- A PIOPS Amazon EBS volume (attached to the Amazon EC2 instance) for durable Database Server storage.
- An Amazon EC2 instance in VPC public subnet as a bastion to provide secure ingress to the rest of the setup.
- A NAT Amazon EC2 instance in VPC public subnet to provide outbound-only communications for internal, private instances.
- Use of a secondary AWS region for disaster recovery purposes (to support continuity in case of primary region issues).

The Amazon EBS and Amazon S3 backup is faster than Oracle Secure Backup, because the system is backed up to a local disk, and the Amazon S3 snapshot is taken without touching the database files.

The latest RMAN backup is available in the backup volume, but if you need to go to a previous backup, you restore the backup volume from the snapshot before you restore the database.

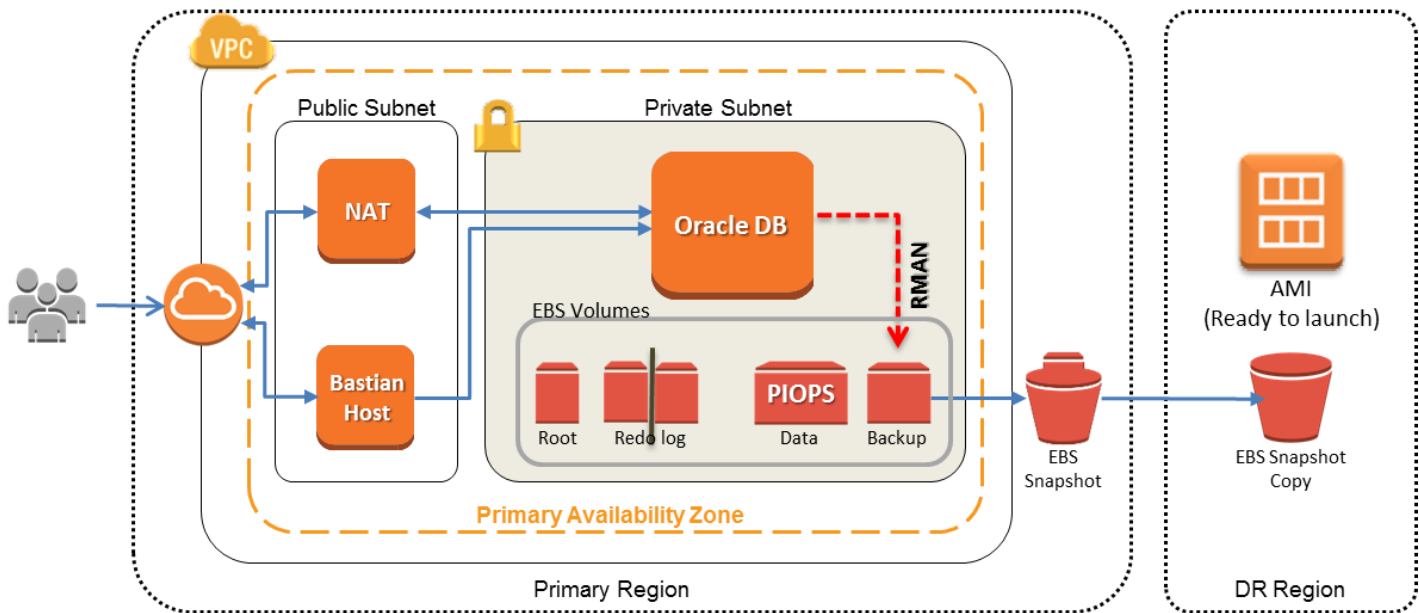


Figure 1: Standard Database Reference Architecture, Amazon EBS Snapshot Version

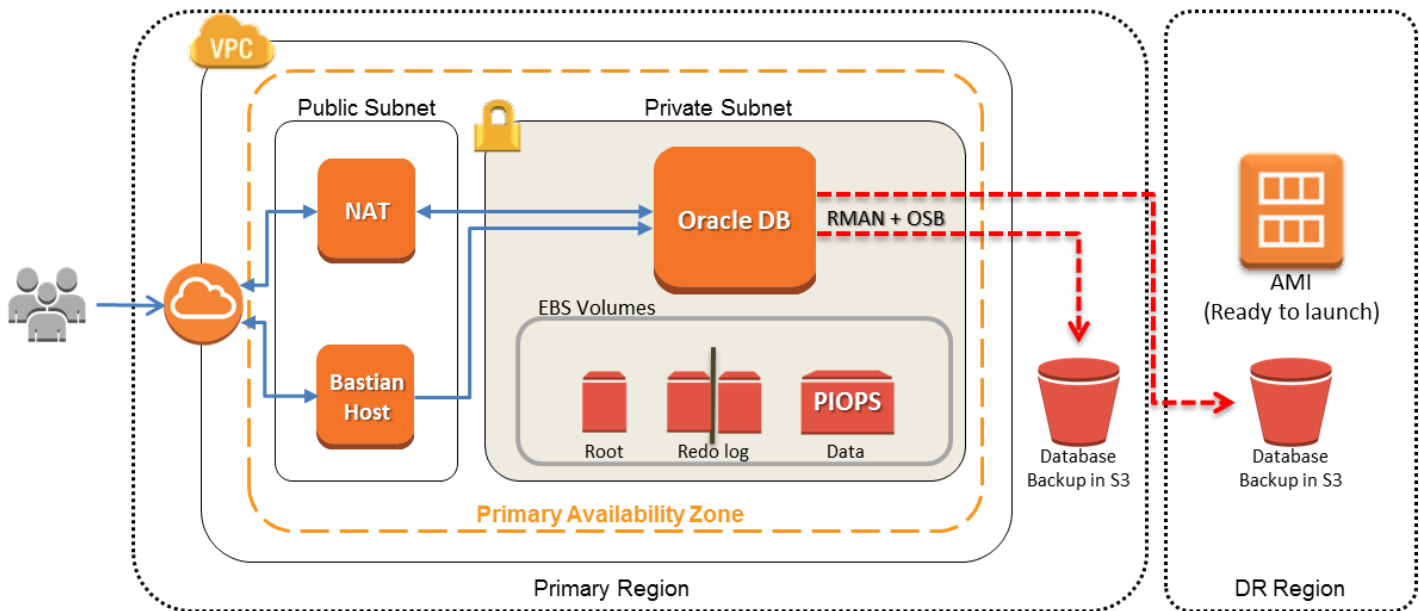


Figure 2: Standard Database Reference Architecture, Oracle Secure Backup Version

Template Overview

Here are the actions this template ([Standard Database](#)) performs:

- Create and set up the Virtual Private Cloud
 - Create VPC
 - Create an Internet gateway for the VPC
 - Attach the Internet gateway to the VPC
 - Create route table for the VPC
 - Create a route for the route table
 - Create a network ACL for the VPC
 - Create inbound network ACL
 - Create outbound network ACL
- Create a public subnet in the first Availability Zone
 - Associate the public subnet to the Route Table
 - Associate public subnet to Network ACL
 - Create security group to be used for Bastion host
 - Create Elastic IP address for Bastion host
 - Create a Micro Amazon EC2 Windows Instance as Bastion host
- Create a private subnet in the first Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
- Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
 - Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored Redo Log Volumes
 - Execute script to install and configure Oracle database

About Template Parameters

The Parameter section of the template defines parameters that will appear in the dialog box that opens when you launch the AWS CloudFormation stack in the AWS Management Console. The values bound to these parameters are used later in the template.

This template parameterizes key details to allow user-defined customization. For example, the EC2 instance type to use for the database instances defaults to m1.xlarge, but also allows you to specify a different instance type within the range of other applicable instance types specified with the parameter declaration. Also, if you want to launch the AWS CloudFormation stack in more than one AWS region, the ID of your AMI will be different in each region (since AMIs are specific to a region). In that case, you would enter the applicable region-specific AMI ID for the “AMI” parameter. Alternately, you can add an entry in the ‘Mappings’ section of the CloudFormation template that maps the applicable AMI ID for each region, allowing the template to select the appropriate AMI ID based on the region it is running in.

Parameters can be configured to automatically use the default value if no value is provided through the dialog box. The Allowed Values value restricts the value that can be provided for this parameter during launch.

Parameters

Here is a description of the parameters used in the template.

Input Parameter	Description
InstanceType	Amazon EC2 instance type to be used for the database instance. Refer to the architecture details for recommended instance types.
KeyName	Name of the Amazon EC2 key pair that you created when you created your custom AMI. You need to use this key pair to launch the components of this implementation, for example, RefImplKeyPair.
AMI	ID of the customized AMI you created previously. You will use this AMI to launch the Amazon EC2 instance. Keep in mind that the AMI ID will change when you copy it to a different region. You need to specify the correct AMI ID to be used, for example, ami-3d749c59.
DatafileStorageSize	Size of storage in gigabytes to be allocated for database files. This architecture scenario is recommended for databases less than 1 TB in size. Choose a slightly bigger storage size than your currently database size to allow for immediate growth.
IOPS	Number of IO operations this database should be able to handle. The value you choose here is critical for the performance of the database. Maximum number of IOPS provisioned for a volume is also limited by the size of the volume. You can provision up to ten times the volume size up to 4000 PIOPS per volume. IOPS provisioned for the volume cannot be changed once the volume is created, so choose the number of IOPS you need based on your requirements. If you have to change provisioned IOPS after the volume is in use, you will have to create a new volume with the desired PIOPS, snapshot the existing volume and restore that to the new volume.
DatabaseSID	SID for the Oracle database to be created.
DatabaseServiceName	Service Name for the Oracle database to be created.
Passwords	Passwords Grid Control and Database accounts. Choose secure passwords with enough complexity based on security practices. We recommended changing these passwords after the installation is complete.
SSHOriginLocation	IP range from which you can SSH into the Amazon EC2 instance. This is for additional security, and restricts SSH access to the Amazon EC2 instance to a specific range of IP numbers. If you don't need the additional level of security, provide the value 0.0.0.0/0 to allow access from any IP address. To allow access only from a single IP address, specify that IP address /32, for example, you could use IP address range 172.67.108.0/24, for a single IP address, use 172.67.108.38/32.

About AWS Resources

This section of the AWS CloudFormation template describes all the AWS resources that the template will launch as part of the stack. Each resource has a logical name unique within the template. Some of the resources used in this AWS CloudFormation template include an Amazon EC2 instance, network resources like Amazon VPC, subnets, Network

Gateway, routing tables, Network ACL and Elastic IP addresses, as well as storage resources like Amazon EBS and Amazon S3.

About Network Resources

This section of the AWS CloudFormation template sets up various parts of the network infrastructure required for the installation, instantiation, and usage of the database in a secure manner.

The first entry creates an Amazon VPC within the region where you launch the AWS CloudFormation stack. We recommend that you place your instances within an Amazon VPC as a security best practice.

In this architecture scenario, our Amazon VPC spans only a single Availability Zone. The CidrBlock property specifies the range of IP addresses you specify for this Amazon VPC. You can choose your own private IP range; which could be part of your corporate IP address block.

Your Amazon VPC is your own isolated network space, so you don't have to worry about your IP addresses conflicting with those of other customers; you just have to avoid creating conflicting addresses within your own Amazon VPC.

```
"VPC" : {
  "Type" : "AWS::EC2::VPC",
  "Properties" : {
    "CidrBlock" : "172.22.0.0/16",
    "Tags" : [{
      "Key" : "Application",
      "Value" : {
        "Ref" : "AWS::StackName"
      }
    }
  ]
}
```

As a security best practice, we want to isolate the database instance in a subnet that does not have direct access from outside the Amazon VPC. To do so, we will create two subnets within the Amazon VPC: a private subnet, the instances within which cannot be accessed from outside the Amazon VPC, and a public subnet, which can access and be accessed from the Internet. The database instance will be in the private subnet, making it well fortified, and an application server or a bastion host will be in the public subnet. Only the application server or the bastion host will be allowed to access the database.

The Fn::GetAZs function selects the Availability Zone to be used within the Amazon VPC. In this case, the subnets are created in the first Availability Zone in the list. You can modify the script to choose another one from the list, or even hard-code an Availability Zone if you know that you will always launch the stack only in one region. The CidrBlock allocated for each subnet will be within what is allocated for the Amazon VPC where we are creating these subnets.

```
"PublicSubnet" : {
  "Type" : "AWS::EC2::Subnet",
  "Properties" : {
    "VpcId" : {
      "Ref" : "VPC"
    },
    "CidrBlock" : "172.22.1.0/24",
    "AvailabilityZone" : {
      "Fn::Select" : ["1", {
        "Fn::GetAZs" : ""
      }
    ]
  }
}
```

```

        ],
        "Tags" : [{
            "Key" : "Application",
            "Value" : {
                "Ref" : "AWS::StackName"
            }
        }
    ]
},
"PrimaryDBSubnet" : {
    "Type" : "AWS::EC2::Subnet",
    "Properties" : {
        "VpcId" : {
            "Ref" : "VPC"
        },
        "CidrBlock" : "172.22.2.0/24",
        "AvailabilityZone" : {
            "Fn::Select" : ["1", {
                "Fn::GetAZs" : ""
            }
        ]
    },
    "Tags" : [{
        "Key" : "Application",
        "Value" : {
            "Ref" : "AWS::StackName"
        }
    }
    ]
},
}
},

```

By default, an Amazon VPC is a fully isolated network. Nothing from outside the Amazon VPC can communicate to anything within the Amazon VPC, so next we need to provide just enough access, so that we keep the instances inside the Amazon VPC highly secure. AWS security design authorizes access strictly on a least-privileged basis.

We create an Internet Gateway and attach it to the Amazon VPC so that the instances that need access to the Internet can access it through the Internet Gateway. The Internet Gateway is attached to the Amazon VPC, not to each subnet.

```

"InternetGateway" : {
    "Type" : "AWS::EC2::InternetGateway",
    "Properties" : {
        "Tags" : [{
            "Key" : "Application",
            "Value" : {
                "Ref" : "AWS::StackName"
            }
        }
    ]
},
"AttachGateway" : {
    "Type" : "AWS::EC2::VPCGatewayAttachment",
    "Properties" : {
        "VpcId" : {
            "Ref" : "VPC"
        },
        "InternetGatewayId" : {
            "Ref" : "InternetGateway"
        }
    }
},

```

We also create routing tables and Network ACLs and attached them to the subnets. The routing tables and Network ACLs give us fine-grain control on egress and ingress, allowing only the specific IP range, port ranges and protocols that we approve to have access to and from our instances.

```

"RouteTable" : {
  "Type" : "AWS::EC2::RouteTable",
  "Properties" : {
    "VpcId" : {
      "Ref" : "VPC"
    },
    "Tags" : [{
      "Key" : "Application",
      "Value" : {
        "Ref" : "AWS::StackName"
      }
    }
  ]
},
"Route" : {
  "Type" : "AWS::EC2::Route",
  "Properties" : {
    "RouteTableId" : {
      "Ref" : "RouteTable"
    },
    "DestinationCidrBlock" : "0.0.0.0/0",
    "GatewayId" : {
      "Ref" : "InternetGateway"
    }
  }
},
"SubnetRouteTableAssociation" : {
  "Type" : "AWS::EC2::SubnetRouteTableAssociation",
  "Properties" : {
    "SubnetId" : {
      "Ref" : "PublicSubnet"
    },
    "RouteTableId" : {
      "Ref" : "RouteTable"
    }
  }
},
"SubnetRouteTableAssociation" : {
  "Type" : "AWS::EC2::SubnetRouteTableAssociation",
  "Properties" : {
    "SubnetId" : {
      "Ref" : "PrimaryDBSubnet"
    },
    "RouteTableId" : {
      "Ref" : "RouteTable"
    }
  }
},
"NetworkAcl" : {
  "Type" : "AWS::EC2::NetworkAcl",

```

```

    "Properties" : {
      "VpcId" : {
        "Ref" : "VPC"
      },
      "Tags" : [{
        "Key" : "Application",
        "Value" : {
          "Ref" : "AWS::StackName"
        }
      }
    ]
  }
},
"InboundNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {
      "Ref" : "NetworkAcl"
    },
    "RuleNumber" : "100",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "false",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {
      "From" : "0",
      "To" : "65535"
    }
  }
},
"OutBoundNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {
      "Ref" : "NetworkAcl"
    },
    "RuleNumber" : "110",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "true",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {
      "From" : "0",
      "To" : "65535"
    }
  }
},
"SubnetNetworkAclAssociation" : {
  "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
  "Properties" : {
    "SubnetId" : {
      "Ref" : "PublicSubnet"
    },
    "NetworkAclId" : {
      "Ref" : "NetworkAcl"
    }
  }
}

```



```
    },
    "SubnetNetworkAclAssociation" : {
      "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
      "Properties" : {
        "SubnetId" : {
          "Ref" : "PrimaryDBSubnet"
        },
        "NetworkAclId" : {
          "Ref" : "NetworkAcl"
        }
      }
    }
  },
},
```

Enterprise Class Database Architecture Scenario Implementation

This architecture scenario is best suited for line of business (LOB) applications or other deployments where the emphasis is on high availability and load distribution.

Here are some of the key differences in this architecture from the previous one.

Multiple Provisioned IOPS Amazon EBS volumes. Although you can add and mount multiple Amazon EBS volumes separately to be used individually for data files, we strongly recommend that you stripe across multiple volumes to achieve much higher performance through cumulative PIOPS. We have achieved up to 40,000 IOPS by striping together 22 Amazon EBS volumes provisioned at 2000 PIOPS each. Theoretically, with 4000 PIOPS volumes you could achieve an even higher total IOPS when many Amazon EBS volumes are striped together. The effect of cumulative IOPS diminishes after about 20-22 volumes, so we don't recommend going beyond 20-22 TB overall size for the database. Because this particular architecture scenario is targeted at a medium-sized enterprise class database, we recommend using fewer than 10 total volumes.

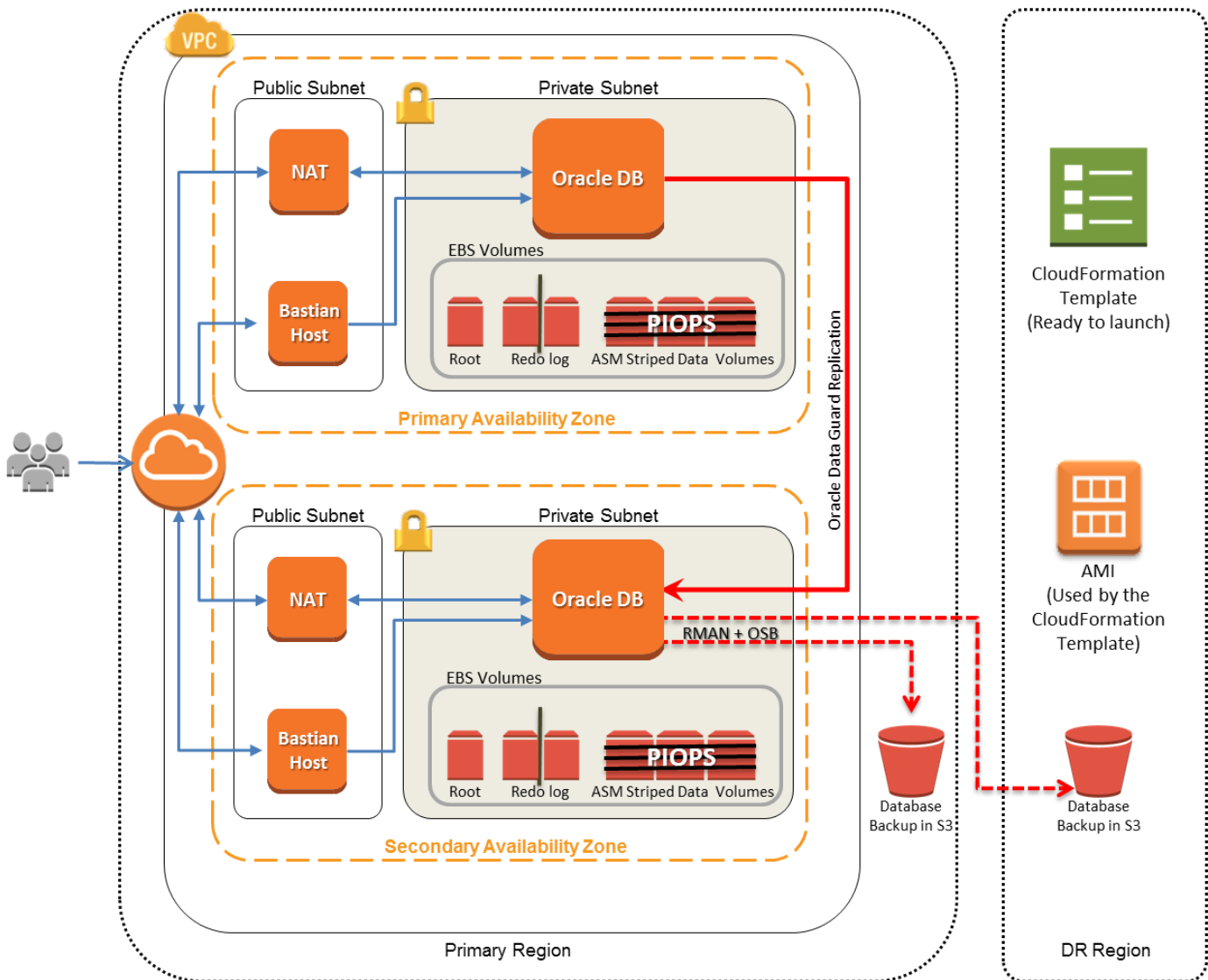


Figure 3: Enterprise Class Database Architecture

Template Overview

Here are the actions this template ([Enterprise Class Database](#)) performs:

- Create and set up a Virtual Private Cloud
 - Create an Internet gateway for the VPC
 - Attach the Internet gateway to the VPC
 - Create route table for the VPC
 - Create a route for the route table
- Create a network ACL for the VPC
 - Create inbound network ACL
 - Create outbound network ACL
 - Create a public subnet in the first Availability Zone
 - Associate the public subnet to the Route Table
- Associate public subnet to Network ACL

- Create security group to be used for Bastion host
- Create Elastic IP address for Bastion host
- Create a Micro Amazon EC2 Windows Instance as Bastion host
- Create a private subnet in the first Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
- Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored Redo Log Volumes
 - Execute script to Stripe PIOPS volumes for Data Files
 - Execute script to install and configure Oracle database
 - Execute script to setup Oracle Secure Backup Cloud Module to backup database to Amazon S3
 - Execute script to create Standby Database in the Standby Instance after making sure that the instance configuration is fully complete
- Execute script to install Oracle Data Guard
- Create a private subnet in the second Availability Zone
 - Associate Private Subnet to the Route Table
 - Associate Private Subnet to Network ACL
 - Create Security Group to be used for Oracle DB instance
- Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for Database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored Redo Log Volumes
 - Execute script to Stripe PIOPS volumes for Data Files
 - Execute script to install Oracle Data Guard
 - Execute script to configure Data Guard Broker
 - Execute script to setup Oracle Secure Backup Cloud Module to backup database to Amazon S3 in the DR Region

This architecture scenario introduces using a mirror image of the entire setup in a secondary Availability Zone and replicating to the database in that Availability Zone using Oracle Data Guard.

The CloudFormation template code to setup the Amazon EC2 two primary and secondary DB instances is shown below:

```
"ORCLDBInstance" : {
  "Type" : "AWS::EC2::Instance",
  "Metadata" : {
    "AWS::CloudFormation::Init" : {
      "config" : {
        "sources" : {
          "/home/oracle/scripts/refarch/" :
            "https://s3.amazonaws.com/orclrefarch/scripts/orcl_refarch_sd_scripts.zip"},
        "commands" : {
          "kickoff-install" : {
```

```

    "command" : { "Fn::Join" : [ "", [
      "./home/oracle/scripts/refarch/refarch_install.sh",
      { "Ref" : "DatabaseSID" }, { "Ref" : "DatabaseServiceName" },
      { "Ref" : "SysASMPassword" }, { "Ref" : "SysPassword" },
      { "Ref" : "SystemPassword" }, { "Ref" : "SysManPassword" },
      { "Ref" : "DBSNMPPassword" }, { "Ref" : "APEXPassword" },
      { "Ref" : "DatabaseSize" }, { "Ref" : "IOPS" }]]]
  "StandbyIPAddress" : {
    "Type" : "AWS::EC2::EIP",
    "Properties" : {
      "Domain" : "vpc",
      "InstanceId" : {
        "Ref" : "ORCLInstanceStandby"
      }
    }
  }
  "ORCLInstanceStandby" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "ImageId" : "ami-d37a19ba",
      "InstanceType" : {
        "Ref" : "InstanceType"
      },
      "KeyName" : {
        "Ref" : "KeyName"
      },
      "Monitoring" : "false",
      "PrivateIpAddress" : "172.22.2.11",
      "SecurityGroupIds" : [{
        "Ref" : "ORCLInstanceSecurityGroup"
      }],
      "SubnetId" : {
        "Ref" : "SubnetTwo"
      },
      "Tags" : [{
        "Key" : "Application",
        "Value" : {
          "Ref" : "AWS::StackName"
        }
      }, {
        "Key" : "Instance",
        "Value" : "LinuxInstance"
      }
    ]
  }

```

Large Enterprise Class Database Architecture Scenario Implementation

This architecture scenario is best suited for business-decision support systems or other environments where the emphasis is on high reliability.

This architecture scenario builds on the previous Enterprise Class Architecture scenario, but it is targeted at more significant enterprise workloads with larger database size and heavier processing requirements. For this scenario, we recommend using larger Amazon EC2 instances and more PIOPS volumes for better throughput. We use Active Data Guard for replication instead of Data Guard so that the replicated instance can be used for read-only load sharing as well. Since 80 percent of database usage in most enterprises is for read rather than for write, many of the read-only applications—Reporting Applications, BI Dashboards, the ETL tool and Data-Mining Applications, for example—can be redirected to use the replicated instance.

This architecture scenario provides high availability through failover to the replicated instance as well as load distribution by making the replicated instance available for read transactions.

This architecture scenario also includes a full-fledged disaster recovery (DR) implementation in a different region using “Pilot Light” DR architecture model. The pilot light is an analogy that comes from the gas heater. In a gas heater, a small idle flame that’s always on can quickly ignite the entire furnace to heat up a building as needed. Like the furnace's pilot

light, this architecture includes a small Amazon EC2 instance type in the DR region just powerful enough to run the database. Data is replicated to this database from the primary database using Data Guard. In case of a disaster the instance type can be scaled up immediately to match the production instance type. The pilot light method offers a shorter recovery time than the Backup and Restore method, because the system is configured and running. All you have to do to switch over to production is scale up the instance and necessary DNS propagation.

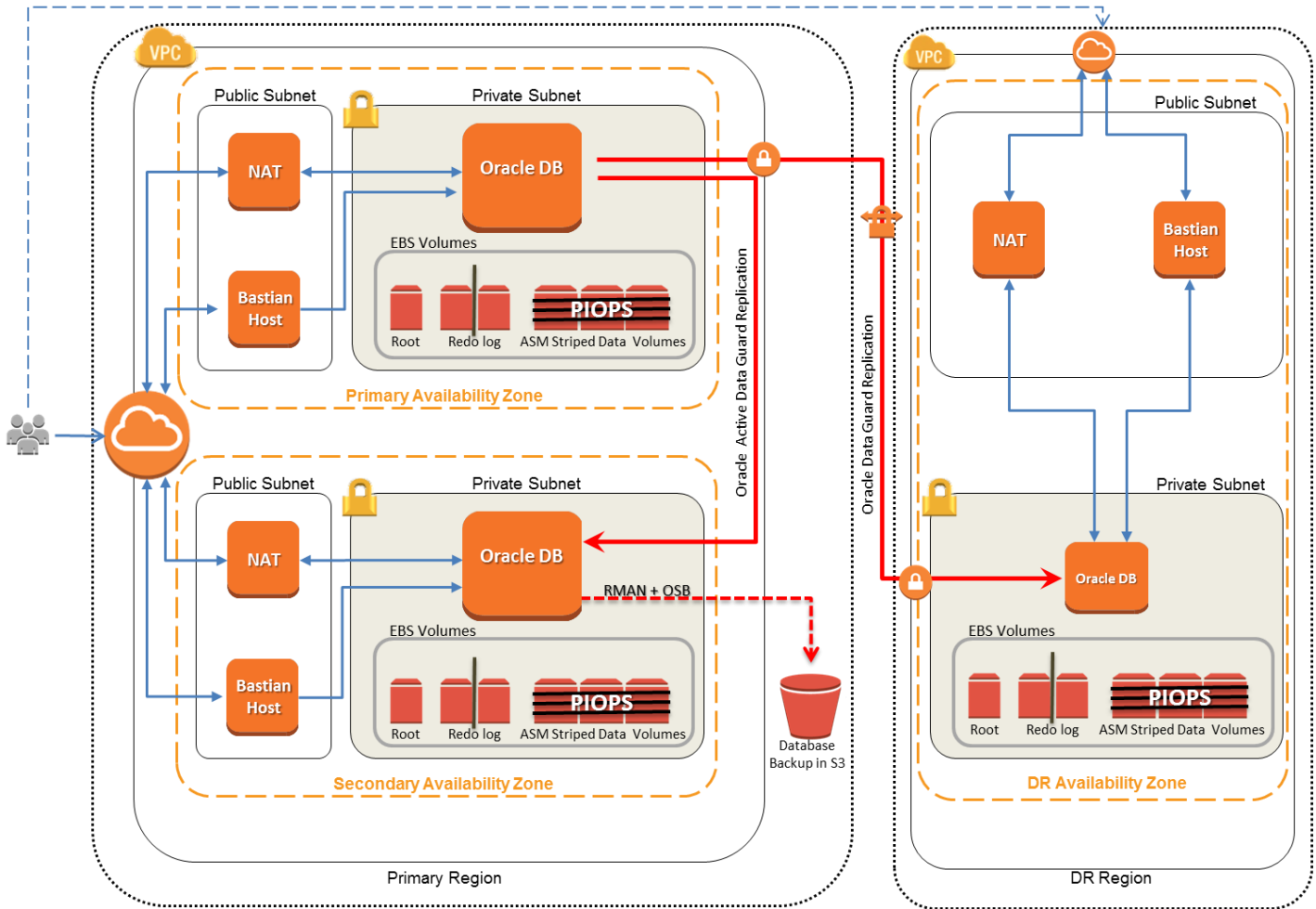


Figure 4: Large Enterprise Class Database Architecture

Template Overview

Here are the actions this template ([Large Enterprise Class Database](#)) performs:

- Create and set up a Virtual Private Cloud
 - Create an Internet gateway for the VPC
 - Attach the Internet gateway to the VPC
 - Create route table for the VPC
 - Create a route for the route table
- Create a network ACL for the VPC
- Create inbound network ACL
- Create outbound network ACL
- Create a public subnet in the first Availability Zone
 - Associate the public subnet to the Route Table
 - Associate public subnet to Network ACL
 - Create security group to be used for bastion host
 - Create Elastic IP address for bastion host
 - Create a Micro Amazon EC2 Windows Instance as bastion host
- Create a private subnet in the first Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
 - Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored Redo Log Volumes
 - Execute script to Stripe PIOPS volumes for Data Files
 - Execute script to install and configure Oracle database
 - Execute script to set up Oracle Secure Backup Cloud Module to backup database to Amazon S3
 - Execute script to create Stand by Database in the Standby Instance after making sure that the instance configuration is fully complete.
 - Execute script to install Oracle Active Data Guard
 - Execute script to configure Data Guard Broker
- Launch DR template to create DR stack in the DR region
- Create a private subnet in the second Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
 - Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored Redo Log Volumes
 - Execute script to Stripe PIOPS volumes for Data Files
 - Execute script to install Oracle Data Guard

- Execute script to configure Data Guard Broker

One of the main difference in this scenario from the previous one is using larger instances, more Amazon EBS volumes and using Active Data Guard instead of Oracle Data Guard so that the replicated instances can be used for read only loads thus providing some load distribution. Neither of these changes substantially changes the AWS CloudFormation template for this scenario compared to the previous one. One difference you will see in the AWS CloudFormation template is that 10 Amazon EBS volumes are being added to the instances in this case as shown below. You can increase the number of volumes even more if you need larger capacity for data volume by adding more Amazon EBS volumes.

```
"Volumes" : [
{ "VolumeId" : { "Ref" : "RedoLogVolume" }, "Device" : "/dev/sde" },
{ "VolumeId" : { "Ref" : "RedoLogVolumeMirror" }, "Device" : "/dev/sdf" },
{ "VolumeId" : { "Ref" : "DataVolume01" }, "Device" : "/dev/sdi" },
{ "VolumeId" : { "Ref" : "DataVolume02" }, "Device" : "/dev/sdj" },
{ "VolumeId" : { "Ref" : "DataVolume03" }, "Device" : "/dev/sdk" },
{ "VolumeId" : { "Ref" : "DataVolume04" }, "Device" : "/dev/sdl" },
{ "VolumeId" : { "Ref" : "DataVolume05" }, "Device" : "/dev/sdm" },
{ "VolumeId" : { "Ref" : "DataVolume06" }, "Device" : "/dev/sdn" },
{ "VolumeId" : { "Ref" : "DataVolume07" }, "Device" : "/dev/sdo" },
{ "VolumeId" : { "Ref" : "DataVolume08" }, "Device" : "/dev/sdp" },
{ "VolumeId" : { "Ref" : "DataVolume09" }, "Device" : "/dev/sdq" },
{ "VolumeId" : { "Ref" : "DataVolume10" }, "Device" : "/dev/sdr" },],
```

Each volume is defined separately based on the size specified in the Parameter as shown below

```
"DataVolume01" : {
  "Type" : "AWS::EC2::Volume",
  "Properties" : {
    "Size" : {
      "Ref" : "DBFileEBSVolumeSize"
    },
    "VolumeType" : "io1",
    "Iops" : {
      "Ref" : "VolumeIOPS"
    },
    "AvailabilityZone" : {
      "Fn::Select" : ["1", {"Fn::GetAZs" : ""}]
    }
  }
},
```

One other difference in this scenario is usage of *Pilot Light* architecture for the Disaster Recovery setup. This is not done automatically. You must run the Disaster Recovery AWS CloudFormation template ([ORCL Ref Arch LECDB DR.template](#)) in the region of your choice to set up the Disaster Recovery environment. From the AWS CloudFormation template you can see that the Disaster Recovery environment is set up in a way similar to the Standard Database Architecture Scenario with the exception of number of Amazon EBS volumes. In the DR environment also, ten Amazon EBS volumes are used for data files, and they are striped using ASM.

```
"ORCL_DR_DBInstance" : {
  "Type" : "AWS::EC2::Instance",
  "Metadata" : {
    "AWS::CloudFormation::Init" : {
```

```

        "config" : {
            "sources" : {
                "/home/oracle/scripts/refarch/":
"https://s3.amazonaws.com/orcl-refarch/scripts/orcl_refarch_ecdb_dr_scripts.zip"
            }
            "commands" : {
                "kickoff-install" : {
                    "command" : { "Fn::Join" : [ "", [
                        "/home/oracle/scripts/refarch/refarch_install.sh
", { "Ref" : "DatabaseSID" }, { "Ref" : "DatabaseServiceName" },
                        { "Ref" : "SysASMPassword" }, { "Ref" :
"SysPassword" }, { "Ref" : "SystemPassword" }, { "Ref" : "SysManPassword" },
                        { "Ref" : "DBSNMPPassword" }, { "Ref" :
"APEXPassword" }, { "Ref" : "DBFileEBSVolumeSize" }, { "Ref" : "VolumeIOPS" }
                    ] ] }
                }
            }
        }
    },
    "Properties" : {
        "ImageId" : {
            "Ref" : "AMI"
        },
        "InstanceType" : {
            "Ref" : "InstanceType"
        },
        "KeyName" : {
            "Ref" : "KeyName"
        },
        "Monitoring" : "false",
        "EbsOptimized" : "true",
        "BlockDeviceMappings" : [
            {
                "DeviceName" : "/dev/sdc",
                "VirtualName" : "ephemeral0"
            },
            {
                "DeviceName" : "/dev/sdd",
                "VirtualName" : "ephemeral1"
            }
        ],
        "Volumes" : [
            { "VolumeId" : { "Ref" : "RedoLogVolume" }, "Device" : "/dev/sde" },
            { "VolumeId" : { "Ref" : "RedoLogVolumeMirror" }, "Device" : "/dev/sdf" },
            { "VolumeId" : { "Ref" : "DataVolume01" }, "Device" : "/dev/sdi" },
            { "VolumeId" : { "Ref" : "DataVolume02" }, "Device" : "/dev/sdj" },
            { "VolumeId" : { "Ref" : "DataVolume03" }, "Device" : "/dev/sdk" },
            { "VolumeId" : { "Ref" : "DataVolume04" }, "Device" : "/dev/sdl" },
            { "VolumeId" : { "Ref" : "DataVolume05" }, "Device" : "/dev/sdm" },
            { "VolumeId" : { "Ref" : "DataVolume06" }, "Device" : "/dev/sdn" },
            { "VolumeId" : { "Ref" : "DataVolume07" }, "Device" : "/dev/sdo" },
            { "VolumeId" : { "Ref" : "DataVolume08" }, "Device" : "/dev/sdp" },
            { "VolumeId" : { "Ref" : "DataVolume09" }, "Device" : "/dev/sdq" },
            { "VolumeId" : { "Ref" : "DataVolume10" }, "Device" : "/dev/sdr" }
        ],
    },

```



```

    "PrivateIpAddress" : "172.22.2.10",
    "SecurityGroupIds" : [{
        "Ref" : "ORCLInstanceSecurityGroup"
    }
    ],
    "SubnetId" : {
        "Ref" : "DRDBSubnet"
    },
    "Tags" : [{
        "Key" : "Application",
        "Value" : {
            "Ref" : "AWS::StackName"
        }
    }, {
        "Key" : "Instance",
        "Value" : "LinuxInstance"
    }
    ]
}
},

```

After completing this you need to set up a VPN connection between the VPC of the primary region and the VPC of the Disaster Recovery region. Follow the instructions in [Amazon VPC Connectivity Options whitepaper](#) to establish the connection between VPCs.

After the VPC-VPC connection is set up, you need to set up and configure Oracle Data Guard Replication between the primary region and the Disaster Recovery region to replicate data. Do test runs to make sure that the data is being replicated properly.

High Performance Database Architecture Scenario Implementation

This architecture scenario is best suited for use with online transaction processing (OLTP) loads where high performance is the crucial measure of success.

The three architecture scenarios we've discussed so far cover most Oracle database use cases on the AWS platform.

To attain high IOPS for running an OLTP application in this architecture scenario, we use local/ephemeral SSD-based volumes that are provided with the Amazon EC2 instance itself. Because these are ephemeral disks, there is the potential to lose the entire database if the instance fails. To prevent any potential for data loss and help achieve reliability, this architecture scenario employs a second instance in the same Availability Zone. It uses Oracle Data Guard to replicate data to this instance from the primary instance.

This implementation sets up Data Guard for Fast Start Failover, so that the failover to standby instance can be achieved quickly. The primary instance uses Elastic Network Interface (ENI), which can be leveraged for an even faster failover by swapping the ENI from the primary instance to the standby instance, because both instances are in the same Availability Zone. This requires a third observer instance to monitor the primary instance and swap the ENI in case of a failure.

We also configure Data Guard in Maximum Availability Mode to help ensure there will be no data loss in case of a failover. Maximum Availability Mode slightly increases commit latency but is worth it to protect data. Because most database systems serve a substantially higher percentage of reads than writes, this shouldn't have much impact on performance.

The Amazon EC2 Hi I/O instance has SSD-based Instance Storage, so we use it to implement this architecture scenario. Because the Hi I/O instance comes with 2 TB of SSD storage, the database size cannot be larger than 2 TB. This architecture scenario can be extended to use in a hybrid scenario where some data files are on the instance storage.

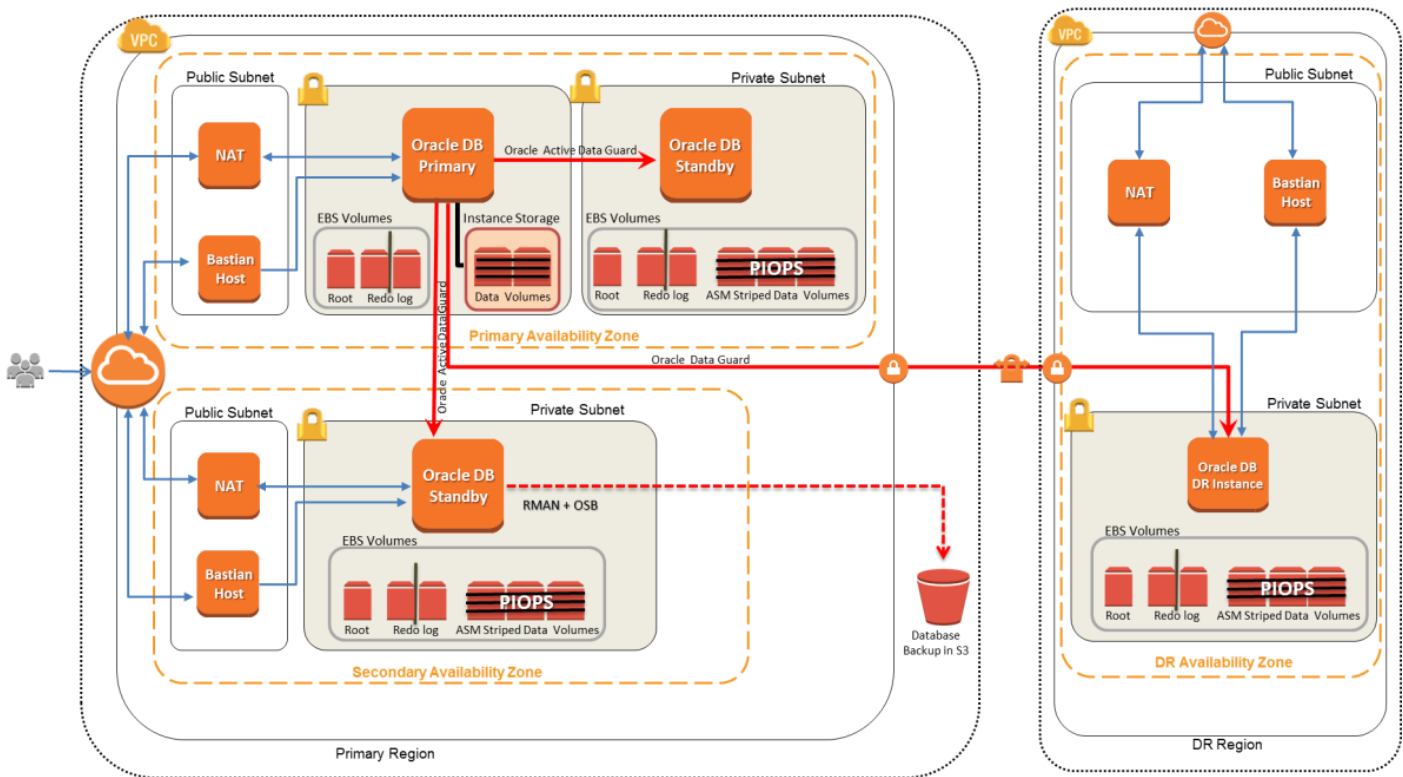


Figure 5: High Performance Database Architecture

Template Overview

Here are the actions this template ([High-Performance Database](#)) performs:

- Create and set up Virtual Private Cloud
 - Create an Internet gateway for the VPC
 - Attach the Internet gateway to the VPC
 - Create route table for the VPC
 - Create a route for the route table
- Create a network ACL for the VPC
 - Create inbound network ACL
 - Create outbound network ACL
- Create a public subnet in the first Availability Zone
 - Associate the public subnet to the Route Table
 - Associate public subnet to Network ACL
 - Create security group to be used for bastion host
 - Create Elastic IP address for bastion host
 - Create a Micro Amazon EC2 Windows Instance as bastion host
- Create a private subnet in the first Availability Zone
 - Associate private subnet to the route table

- Associate private subnet to network ACL
- Create security group to be used for Oracle DB instance
- Create Elastic IP address for Oracle Instance
- Create PIOPS Amazon EBS volumes for database files based on input by user
- Create three standard volumes for redo log and backup
- Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored redo log volumes
 - Execute script to stripe ephemeral volumes for data files
 - Execute script to install and configure Oracle database
 - Execute script to set up Oracle Secure Backup Cloud Module to back up database to Amazon S3
 - Execute script to create standby Database in the first standby instance in the same Availability Zone after making sure that the instance configuration is fully complete
 - Execute script to create standby database in the second standby instance in the second Availability Zone after making sure that the instance configuration is fully complete
 - Execute script to install Oracle Active Data Guard
 - Execute script to configure Data Guard Broker
- Launch DR template to create DR stack in the DR region
- Create a second private subnet in the first Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
 - Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored redo log volumes
 - Execute script to stripe PIOPS volumes for data files
 - Execute script to install and configure Oracle database
 - Execute script to set up Oracle Secure Backup Cloud Module to back up database to Amazon S3
 - Execute script to create standby database in the standby instance after making sure that the instance configuration is fully complete.
 - Execute script to install Oracle Active Data Guard
 - Execute script to configure Data Guard Broker
- Launch DR template to create DR stack in the DR region
- Create a private subnet in the second Availability Zone
 - Associate private subnet to the route table
 - Associate private subnet to network ACL
 - Create security group to be used for Oracle DB instance
 - Create Elastic IP address for Oracle Instance
 - Create PIOPS Amazon EBS volumes for database files based on input by user
 - Create three standard volumes for redo log and backup
 - Create Database Amazon EC2 instance from custom AMI ID input by the user
- Download installation scripts from Amazon S3 bucket to instance volume
 - Execute script to create mirrored redo log volumes
 - Execute script to stripe PIOPS volumes for data files
 - Execute script to install Oracle Data Guard

- Execute script to configure Data Guard Broker

The major difference in this scenario from the previous ones is using ephemeral instance storage for the data files of the primary instance. The ephemeral disks are mapped in the AWS CloudFormation template as shown below to be later used by the database installation scripts

```
"BlockDeviceMappings" : [
  {
    "DeviceName" : "/dev/sdc",
    "VirtualName" : "ephemeral10"
  },
  {
    "DeviceName" : "/dev/sdd",
    "VirtualName" : "ephemeral11"
  }
]
```

This scenario also uses an additional replicated instance in the same Availability Zone to add reliability as the primary database is stored on ephemeral disks. This database is in its own private subnet in the same Availability Zone.

```
"SecondaryDBSubnet" : {
  "Type" : "AWS::EC2::Subnet",
  "Properties" : {
    "VpcId" : {
      "Ref" : "VPC"
    },
    "CidrBlock" : "172.22.3.0/24",
    "AvailabilityZone" : {
      "Fn::Select" : ["1", {
        "Fn::GetAZs" : ""
      }]
    },
    "Tags" : [{
      "Key" : "Application",
      "Value" : {
        "Ref" : "AWS::StackName"
      }
    }]
  }
},
```

Database instance is created in this subnet with Amazon EBS volumes as the data storage so that in the case of a failure of the primary instance the data is safely persisted on the Amazon EBS volume-based database instance.

```
"ORCLSecondaryDBInstance" : {
  "Type" : "AWS::EC2::Instance",
  "Metadata" : {
    "AWS::CloudFormation::Init" : {
```

```

        "config" : {
            "sources" : {
                "/home/oracle/scripts/refarch/":
"https://s3.amazonaws.com/orcl-
refarch/scripts/orcl_refarch_hpdb__sec_scripts.zip"
            }
            "commands" : {
                "kickoff-install" : {
                    "command" : { "Fn::Join" : [ "", [
"./home/oracle/scripts/refarch/refarch_install.sh ",
{ "Ref" : "DatabaseSID" }, { "Ref" : "DatabaseServiceName" },
{ "Ref" : "SysASMPassword" }, { "Ref" : "SysPassword" }, { "Ref" :
"SystemPassword" }, { "Ref" : "SysManPassword" }, { "Ref" :
"DBSNMPPassword" }, { "Ref" : "APEXPassword" }, { "Ref" :
"DBFileEBSVolumeSize" }, { "Ref" : "VolumeIOPS" }
                    ]]}
                }
            }
        }
    },
    "Properties" : {
        "ImageId" : {
            "Ref" : "AMI"
        },
        "InstanceType" : {
            "Ref" : "InstanceType"
        },
        "KeyName" : {
            "Ref" : "KeyName"
        },
        "Monitoring" : "false",
        "EbsOptimized" : "true",
        "BlockDeviceMappings" : [
            {
                "DeviceName" : "/dev/sdc",
                "VirtualName" : "ephemeral0"
            },
            {
                "DeviceName" : "/dev/sdd",
                "VirtualName" : "ephemeral1"
            }
        ],
        "Volumes" : [
            { "VolumeId" : { "Ref" : "RedoLogVolume" }, "Device" : "/dev/sde" },
            { "VolumeId" : { "Ref" : "RedoLogVolumeMirror" }, "Device" :
"/dev/sdf" },
            { "VolumeId" : { "Ref" : "DataVolume01" }, "Device" : "/dev/sdi" },
            { "VolumeId" : { "Ref" : "DataVolume02" }, "Device" : "/dev/sdj" },
            { "VolumeId" : { "Ref" : "DataVolume03" }, "Device" : "/dev/sdk" },
            { "VolumeId" : { "Ref" : "DataVolume04" }, "Device" : "/dev/sdl" },
            { "VolumeId" : { "Ref" : "DataVolume05" }, "Device" : "/dev/sdm" },
        ],
        "PrivateIpAddress" : "172.22.2.10",
    }
}

```

```
    "SecurityGroupIds" : [{
        "Ref" : "ORCLInstanceSecurityGroup"
    }
],
"SubnetId" : {
    "Ref" : "PrimaryDBSubnet"
},
"Tags" : [{
    "Key" : "Application",
    "Value" : {
        "Ref" : "AWS::StackName"
    }
}, {
    "Key" : "Instance",
    "Value" : "LinuxInstance"
}
]
}
},
```

Launch Oracle Database Architecture Scenario

Prerequisites for Launching the Scenarios

Now that you're reviewed the rationales and sample templates for the scenarios, let's set up the necessary prerequisites to launch these scenarios using the sample templates provided.

You must choose the edition of Oracle database that's best suited for the scenario and template you want to use. We'll briefly introduce you to some of the technologies that the AWS CloudFormation templates rely upon to create your Oracle environment, and then you'll perform some setup tasks, such as creating an AWS account if you don't have one, creating an Amazon Machine Image (AMI), and installing the AWS CloudFormation helper scripts.

Choosing an Oracle Database Edition

We recommend using Oracle Database Standard One Edition for the first scenario and Oracle Database Enterprise Edition for scenarios two through four.

Each Oracle database edition provides specific levels of processing power and support for other features. AWS supports all Oracle database editions, versions 9i release 2 and later, but this guide discusses only Oracle 11g Standard One and Enterprise, because they are the best choices to run with the sample templates.

Enterprise Edition supports an unlimited number of processors, places no limit on database size, includes Data Guard, Active Data Guard, and 64-bit support, all of which are required by the most complex scenario described in this paper. Standard Edition and Standard One Edition place no limit on database size, limit the numbers of processors, and don't come with Data Guard or Active Data Guard. Express Edition is so limited that we don't recommend using it for business production.

The following table compares Oracle database 11g editions.

	Express	Standard One	Standard	Enterprise
Maximum CPU	1 CPU	2 sockets	4 sockets	No limit
Maximum RAM	1 GB	OS max	OS max	OS max
Maximum Database Size	4 GB	No limit	No limit	No limit
Data Guard	No	No	No	Yes
Active Data Guard	No	No	No	Yes
64-Bit Support	No	Yes	Yes	Yes

Table 2: Comparison of Oracle DB 11g Licenses/Editions

For licensing purposes, Oracle considers the virtual core count of Amazon EC2 instances equal to a physical core count. You can license Oracle Database Standard on Amazon EC2 instances up to 16 virtual cores, and Standard Edition One on Amazon EC2 instances up to eight virtual cores. In Standard Edition or Standard Edition One, Amazon EC2 instances with four or fewer virtual cores count as one socket. In Enterprise Edition, Amazon EC2 instances with two or fewer cores count as one socket.

Note: This discussion of Oracle licensing policies and Oracle licensing costs is for informational purposes only and is based on the information available at the time of writing. Ensure that you are in compliance with Oracle licensing guidelines by visiting the [Oracle Licensing Policies](http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf) website at <http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf>.

Resources Used by the Sample Implementations

The reference implementation scenarios use three different types of resources:

- **Customized version of an Oracle Supplied Database Amazon EC2 AMI.** The Database Amazon EC2 instance is started based on this AMI. This AMI is based on Oracle Linux and already includes all the installation files necessary for the database install.
- **AWS CloudFormation Template.** The AWS CloudFormation template we provide creates the AWS infrastructure components, launches the Amazon EC2 instance using the custom AMI, and executes shell scripts within the Amazon EC2 instance.
- **Shell Scripts.** This guide provides a set of UNIX shell scripts for installing and configuring Oracle software. These scripts are located in an Amazon S3 bucket as bundled ZIP files, one for each architecture. Each ZIP file contains all scripts used by that particular architecture. The AWS CloudFormation template downloads the ZIP file that corresponds to the scenario you choose and extracts the scripts into the `/home/oracle/scripts/refarch` directory of the root volume of the Amazon EC2 instance. After the scripts are extracted, they are executed by AWS CloudFormation by passing the appropriate parameters.

Startup Tasks Common to All Four Architecture Scenarios

Before you can work with any of the four sample templates provided in this guide, you need to perform the following tasks. (If you already have an AWS account, you can skip the first one.)

1. Sign up for an AWS account, if you don't already have one.
2. Build an Oracle database custom AMI.
3. Launch the AWS CloudFormation Stack for the scenario using the sample template provided.

Sign Up for an AWS Account

If you do not already have an AWS account, take the following steps:

1. Go to <http://aws.amazon.com>.
2. Click **Sign Up**.
3. Follow the on-screen instructions to complete the sign up process.

Amazon Web Services automatically signs up your new account for all services offered through AWS, including Amazon EC2. (You are charged only for services that you use.)

Building an Oracle Database Custom AMI

Before you can launch the AWS CloudFormation sample templates provided in this guide, you must build a custom Amazon Machine Image (AMI) that includes Oracle database to use for implementing the reference architecture scenarios.

These reference architecture examples are implemented using a custom AMI derived from one provided by Oracle, based on the Oracle Linux operating system. The Oracle-supplied AMIs currently available on AWS does not include certain features that the sample templates in this paper rely on, so you must build a custom AMI. You can implement the same architecture on any Oracle-supported operating system, including Red Hat Enterprise Linux, SUSE Linux, or Microsoft Windows, but the AWS CloudFormation sample templates and shell scripts provided with this guide are specific to the Oracle Linux operating system. To build a custom Oracle database AMI for these architecture scenarios, execute the following steps.

1. Go to the **AWS Management Console** and click **EC2**.
2. Click **Launch Instance**.
3. On the **Create a New Instance** screen, after naming your instance and choosing or creating a key pair, in the **Choose a Launch Configuration** box, click **More Amazon Machine Images** and then click **Continue**.
4. Search for the following AMIs based on the Oracle Database Edition you plan to use. For example, if you are launching your AWS CloudFormation stack in the U.S. East (Northern Virginia) Region, choose from among the following:
 - a. Oracle Database 11.2.0.1 64-bit Standard Edition One
 - b. Oracle Database 11.2.0.1 64-bit Standard Edition
 - c. Oracle Database 11.2.0.1 64-bit Enterprise Edition
5. Click the appropriate AMI for your version of Oracle Database, and then click **Continue**.

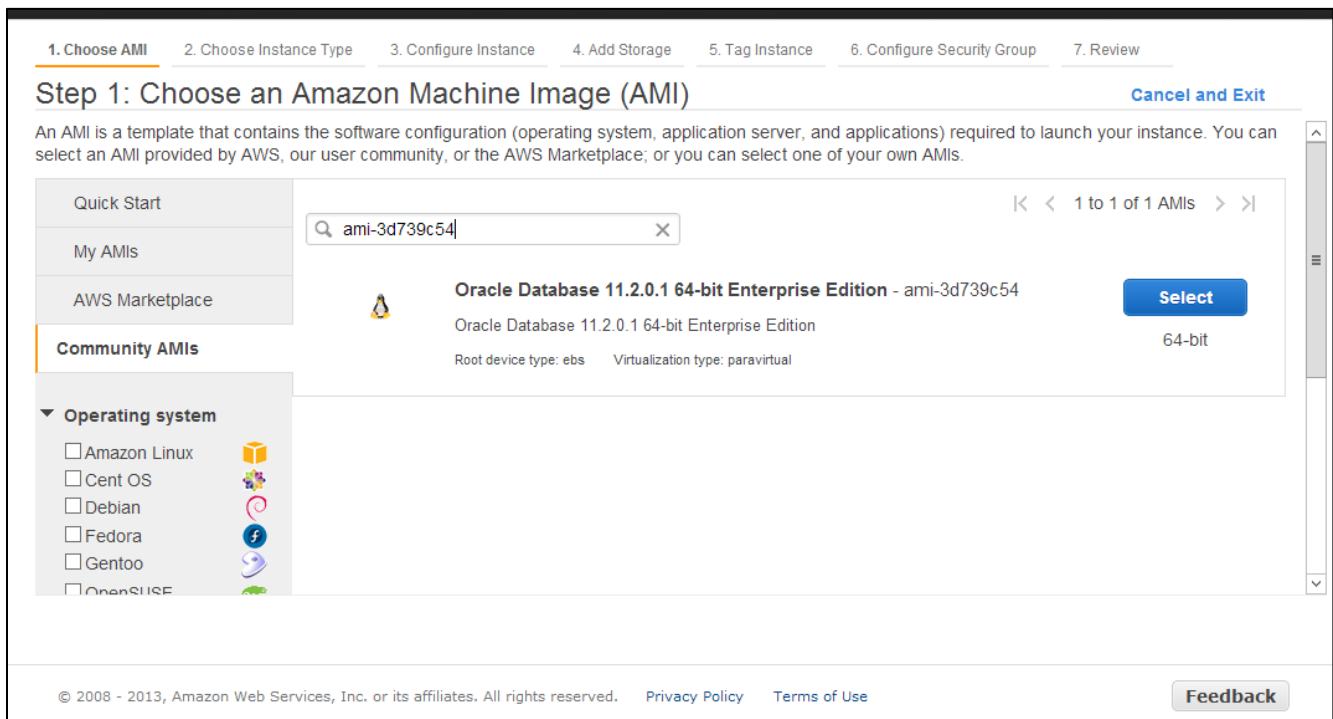


Figure 8. Oracle Database 11.2.0.1 64-bit Enterprise Edition AMI.

6. After the instance becomes available, log into it using an SSH client like Putty or the Java Client available from the AWS Management Console.
7. Accept the Oracle EULA and click **No** when you are prompted to **Install Database**.

Download Oracle Grid Infrastructure Binaries

The architecture scenarios in this guide use ASM for striping multiple Amazon EBS volumes together to accommodate the database size and to achieve better performance. You must download the Grid Infrastructure binaries, as shown in Figure 8.

1. From the Oracle Technology Network website at <http://www.oracle.com/technetwork/index.html>, download the Grid Infrastructure binaries as a ZIP file based on OTN licensing or your own licensing.
2. Copy the file `linux.x64_11gR2_grid.zip` to `/u01/app` folder in the Oracle database Instance you started in the previous section.

Oracle Technology Network > Database > Database 12c > Downloads

Database 12c
Multitenant
Application Development
Big Data
Data Warehousing
Database Cloud
Engineered Systems
High Availability
Manageability
Performance
Security
Storage Management
Unstructured Data
Windows
Database Technology Index

Overview Downloads Documentation Learn More Community

Oracle Database 11g Release 2 (11.2.0.1.0)

Standard Edition, Standard Edition One, and Enterprise Edition

You must accept the [OTN License Agreement](#) to download this software.
 Accept License Agreement | Decline License Agreement

Oracle Database 11g Release 2 (11.2.0.1.0) for Linux x86-64

- linux.x64_11gR2_database_1of2.zip (1,239,269,270 bytes) (cksum - 3152418844)
- linux.x64_11gR2_database_2of2.zip (1,111,416,131 bytes) (cksum - 3669256139)

Directions

1. All files are in the .zip format. There is an unzip utility here if you need one.
2. Download and unzip both files to the same directory.
3. Installation guides and general Oracle Database 11g documentation are here.
4. Review the certification matrix for this product here.

Oracle Database 11g Release 2 Client (11.2.0.1.0) for Linux x86-64

- linux.x64_11gR2_client.zip (706,187,979 bytes) (cksum - 3654981652)

Contains the Oracle Client Libraries for Linux. Download if you want the client libraries only

Oracle Grid Infrastructure 11g Release 2 (11.2.0.1.0) for Linux x86-64

- linux.x64_11gR2_grid.zip (1,052,897,657 bytes) (cksum - 3369676398)

Contains the Grid Infrastructure Software including Oracle Clusterware, Automated Storage Management (ASM), and ASM Cluster File System. Download and install prior to installing Oracle Real Application Clusters, Oracle Real Application Clusters One Node, or other Oracle software in a Grid Environment

Figure 9. Download the Grid Infrastructure binaries.

Upgrading Python to 2.6 +

This AMI has Python 2.4.3 installed. To work with the AWS CloudFormation template, you need to upgrade to Python 2.6+. Follow these steps, replacing x with the latest version number.

1. Use SSH to log into the computer where you plan to install Python.
2. Download the latest version of Python into `/opt/aws/python`:

```
wget http://www.python.org/ftp/python/2.6.x/Python-2.6.x.tgz
```

3. Extract the files:

```
tar -zxvf Python-2.6.x.tgz
```

4. Enter the Python directory:

```
cd Python-2.6.x
```

5. Set the run-time library search path:

```
LD_RUN_PATH=/usr/local/lib; export LD_RUN_PATH
```

6. Prepare the build environment:

```
make clean
```

7. Create the build configuration (the `-enable-shared` option is needed by `mod_wsgi`):

```
./configure --enable-shared
```

8. Build and install Python:

```
make
```

```
make install
```

```
./install-sh
```

9. When your SSH session is terminated, log back in and run `python` to test the new installation.

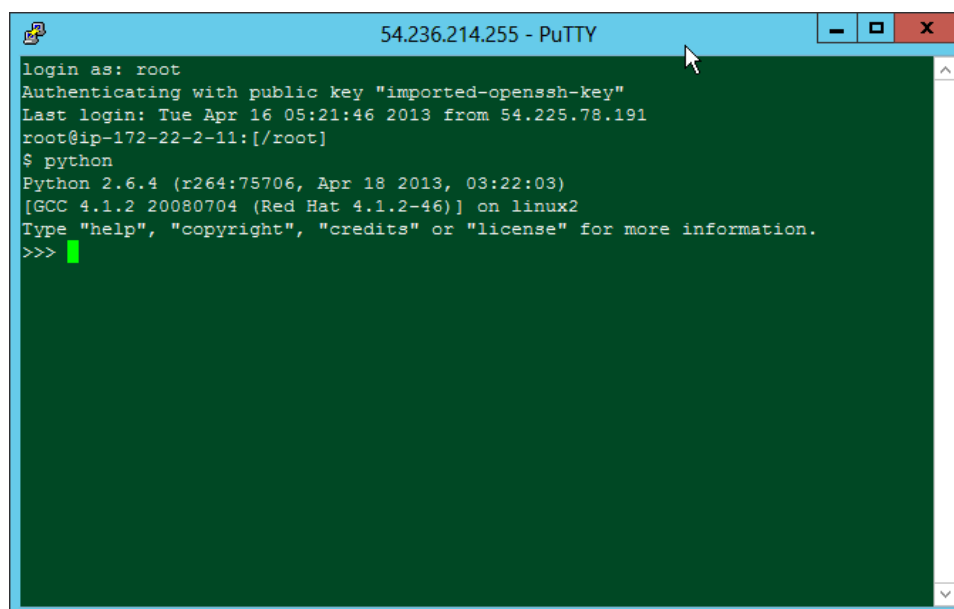
A screenshot of a PuTTY terminal window titled "54.236.214.255 - PuTTY". The terminal shows the following text: "login as: root", "Authenticating with public key 'imported-openssh-key'", "Last login: Tue Apr 16 05:21:46 2013 from 54.225.78.191", "root@ip-172-22-2-11: [/root]", "\$ python", "Python 2.6.4 (r264:75706, Apr 18 2013, 03:22:03)", "[GCC 4.1.2 20080704 (Red Hat 4.1.2-46)] on linux2", "Type 'help', 'copyright', 'credits' or 'license' for more information.", and ">>>". A green cursor is visible after the last prompt. The terminal window has a blue title bar and standard window controls (minimize, maximize, close) on the right side.

Figure 10: Python Upgrade Completed.

Install the AWS CloudFormation Helper Scripts (cfn-init)

1. Download the latest version of the AWS CloudFormation helper scripts:

```
wget https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
```

2. Untar the tar ball:

```
tar -zxvf aws-cfn-bootstrap-latest.tar.gz
```

You have created a new folder named aws-cfn-bootstrap-1.3 and unzipped all the files into it.

3. Switch to that folder and run:

```
python setup.py install
```

Your instance is now ready to create the custom AMI.

4. Go to the **AWS Management Console**, right-click the Amazon EC2 instance, and then choose **Create Image (EBS AMI)**. Enter the appropriate information on the next page of the wizard and then choose **Yes, Create**.

After the AMI is created, it becomes available under AMIs in your AWS account.

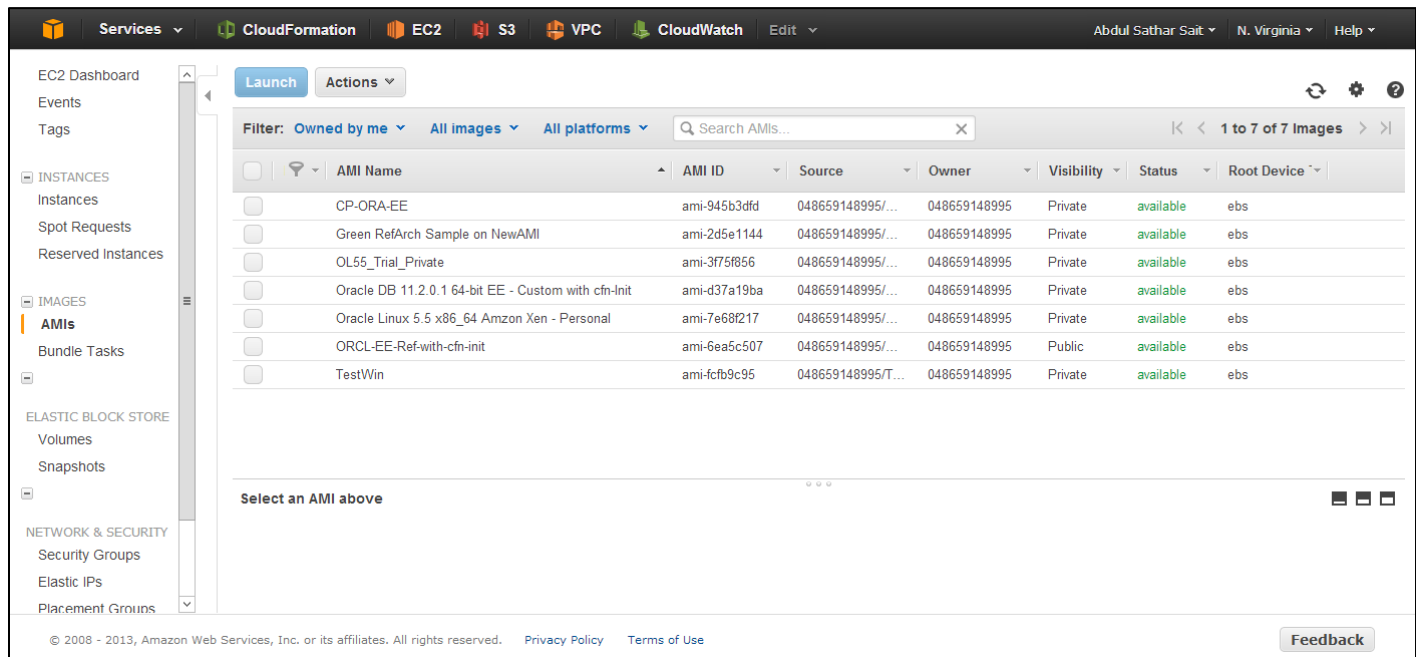


Figure 11. AMIs Available From the Left Navigation Pane.

Note the AMI ID. You will use it as an input to the AWS CloudFormation script.

Click one of the links below to download the AWS CloudFormation template that you want to use.

- [Standard Database](#)
- [Enterprise Class Database](#)

- [Large Enterprise Class Database](#)
- [High-Performance Database](#)

Launch the AWS CloudFormation stack

1. Go to the **AWS Management Console** and click **CloudFormation**.
2. In the next page, choose **Create New Stack**, and then click **Continue**.
3. Provide a name for the stack and select the template file for the scenario you want to use.

Follow the instructions onscreen to launch the AWS CloudFormation stack. After launch, on the **Events** tab, click **View Active Stack** to see the execution of each event in CloudFormation. After the stack is fully created, follow the instructions in the appendix to install OSB in the database instance to backup data to Amazon S3.

Conclusion

Amazon Web Services (AWS) provides a secure, reliable, resilient, cost-effective and highly flexible cloud computing platform for running Oracle database at scale. Using one of the four scenarios in this guide, you can implement an Oracle database environment quickly, cost-effectively, and with minimal effort, leveraging services from AWS such as Amazon EC2 and Amazon VPC without having to worry about creating and configuring the underlying AWS infrastructure.

References

- AWS Services Details
 - <http://aws.amazon.com/products/>
 - <http://aws.amazon.com/documentation/>
 - <http://aws.amazon.com/whitepapers/>
- Oracle on AWS
 - <http://aws.amazon.com/oracle/>
 - <http://aws.amazon.com/rds/oracle/>
- Oracle Test Drives on AWS
 - <http://aws.amazon.com/solutions/global-solution-providers/oracle/labs/>
- Oracle Licensing on AWS
 - <http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf>
- Oracle on AWS FAQ
 - <http://www.oracle.com/technetwork/topics/cloud/faq-098970.html>
- Getting Started with Oracle Secure Backup and RMAN with Amazon S3
 - <http://aws.amazon.com/backup-storage/gsg-oracle-rman/>
- AWS Case Study: Amazon.com Oracle DB Backup to Amazon S3
 - <http://aws.amazon.com/solutions/case-studies/amazon-oracle/>

Appendix: Install Oracle Secure Backup Module

To install and configure Oracle Secure Backup (OSB) Module, you must have Oracle Technical Network credentials. After you have Oracle Technical Network credentials, follow the steps below. (For a video on RMAN backup, see <http://aws.amazon.com/backup-storage/gsg-oracle-rman/>)

To install and configure Oracle Secure Backup to back up Oracle database directly to Amazon S3 using RMAN

1. Download the OSB Cloud Module for Amazon S3 installer (osbws_installer.zip) from <http://www.oracle.com/technetwork/products/secure-backup/secure-backup-s3-484709.html> to the database server.
2. Copy and unzip the OSB Cloud Module for Amazon S3 installer archive to a staging directory as the Oracle software owner.
3. Create a directory for the secure Oracle wallet. The Oracle wallet will be created by the installer and used to store your Amazon S3 credentials.

The authentication method used by Amazon S3 relies on the client's system time being similar to Amazon S3's time. In this case, the client is the machine where you run the OSB Web Services library. Amazon S3 time is Coordinated Universal Time (UTC), so you must ensure that the system time on your client is within a few minutes of UTC.

To Install and Configure Oracle Secure Backup Cloud Module for Amazon S3

1. Verify the environment by listing the valid set of input parameters for the OSB Cloud Module for Amazon S3 installer along with their descriptions.

```
$ cd ~/software/osbws
$ java -jar osbws_install.jar
```

You can also review the osbws_readme.txt file for more details.

2. Create a script that will invoke the OSB Cloud Module for Amazon S3 installer with the required input parameter values.

```
$ vi osbws_install.sh

java -jar osbws_install.jar \
-AWSID nnnnnnnnnnnnnnnn \
-AWSKey nnnnnnnnnnnnnnnnn \
-otnUser Richard.roe@example.com \
-otnPass xxxxxxxx \
-walletDir $ORACLE_HOME/dbs/osbws_wallet \
-libDir $ORACLE_HOME/lib \
-configFile $ORACLE_HOME/dbs/osbws<ORACLE_SID>.ora

$ chmod +x osbws_install.sh
```

Here are the OSB parameters:

Parameter	Description	Required?
[-AWSID] and [-AWSKey]	Supply your AWS access key and secret key which serve the purpose of ID and Password to access Amazon S3.	(Mandatory)
[-otnUser]	Your OTN username, which the installer uses to identify the customer.	(Mandatory)
[-otnPass]	Your OTN password.	(Mandatory)
[-walletDir]	Directory where you want the installer to create a secure wallet containing your Amazon S3 credentials and proxy information (if a proxy server is specified).	(Mandatory)
[-libDir]	Directory where you want the installer to download the OSB Cloud Module for Amazon S3 software library. The installer detects the underlying platform and downloads the appropriate port of the software library. Although this is an optional parameter, if you omit it, the installer will not download the library. Usually you should download the library. Omit downloading the library only if you are using the install tool to regenerate the wallet and configuration file in an Oracle Home where the OSB Cloud Module for Amazon S3 has previously been installed.	(Optional)
[-configFile]	The name of the initialization parameter file that will be created by the install tool. This parameter file will be referenced during your RMAN jobs for a particular database. If this parameter is not specified then the initialization parameter file will be created in a system-dependent default location and filename based on the ORACLE_SID, for example: \$ORACLE_HOME/dbs/osbws<ORACLE_SID>.ora.	(Optional)

Run OSB Cloud Module for Amazon S3 Installer Script

Run the OSB Cloud Module for Amazon S3 installer script that you created in the previous step. The installer downloads the software library and configures the environment for running Oracle database backups to the cloud (Amazon S3):

```
$ ./osbws_install.sh
Oracle Secure Backup Database Web-Service Install Tool, build 2011-02-04.0001
AWS credentials are valid.
Creating new registration for this Amazon S3 user.
Created new log bucket.
Registration ID: 5b2c25ef-74cd-4b1d-a6d2-792e78c759c3
Amazon S3 Logging Bucket: oracle-log-idevjhun-1
Validating log bucket location ...
Validating license file ...
Create credential oracle.security.client.connect_string1
OSB web-services wallet created in directory
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/osbws_wallet.
OSB web-services initialization file
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/osbwstestdb1.ora created.
Downloading OSB Web Services Software Library from file osbws_linux64.zip.
Downloaded 20744341 bytes in 10 seconds. Transfer rate was 2074434 bytes/second.
Download complete.
```

```
Extracted file /u01/app/oracle/product/11.2.0/dbhome_1/lib/libosbws12.so
```