Deploying Oracle SOA Suite 12c on AWS

AWS Whitepaper

Deploying Oracle SOA Suite 12c on AWS: AWS Whitepaper

Table of Contents

Deploying Oracle SOA Suite 12c on AWS

Publication date: **October 19, 2022** (*Document revisions* (p. 21))

Abstract

This whitepaper provides guidance for deploying Oracle SOA Suite 12c applications on Amazon Web Services (AWS). This whitepaper provides a reference architecture and information about best practices for high availability, security, scalability, and performance when you deploy Oracle SOA Suite 12c-based applications on AWS.

The target audience of this whitepaper is solutions architects, systems architects, and system administrators with a basic understanding of cloud computing, AWS, and Oracle SOA Suite 12c.

This whitepaper assumes that you have a basic understanding of AWS. For an overview of AWS services, refer to Overview of Amazon Web Services.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Many enterprises today rely on application and data integration to deliver new services and capabilities to their customers. Oracle SOA Suite 12c is a popular application integration and service-oriented architecture (SOA) solution that enables you to build, deploy, and manage integrations. Some of the benefits of deploying Oracle SOA Suite on AWS include:

- **Low cost** – Resources are billed by the hour and only for the duration they are used.
- **Eliminate the need for large capital outlays** – Replace large, upfront expenses with low variable payments that only apply to what you use.
- **High availability** – Achieve high availability by deploying Oracle SOA Suite 12c in a Multi-AZ configuration.
- **Flexibility** – Add compute capacity elastically to cope with demand.
- **Testing** – Add test environments, use them for short durations, and pay only for the duration they are used.

You can use various AWS services to deploy Oracle SOA Suite 12c-based applications on AWS in a secure, highly-available, and cost-effective manner. With automatic scaling, you can dynamically scale the compute resources required for your application, thereby keeping your costs low. You can use Amazon Elastic File System (Amazon EFS) as shared storage to store Oracle SOA Suite artifacts. Amazon EFS provides a scalable, fully-managed elastic network file system (NFS) with the ability to grow and shrink automatically as you add and remove files.

Use cases for Oracle SOA Suite 12c on AWS

Oracle SOA Suite 12c customers use AWS for a variety of use cases, including:

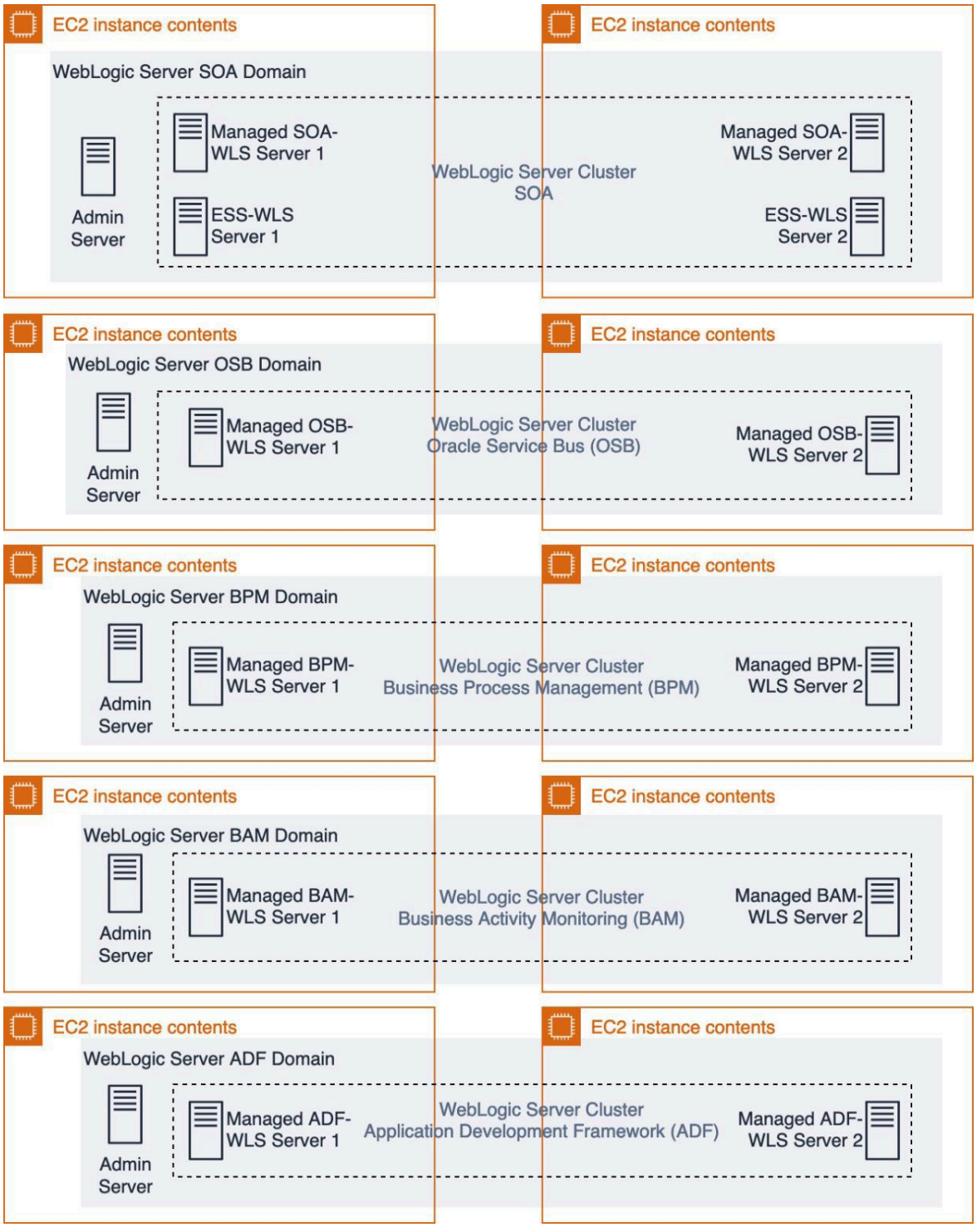
- Migration of existing Oracle SOA Suite production environments.
- Implementation of new Oracle SOA Suite production environments.
- Implementation of disaster recovery environments.
- Running Oracle SOA Suite development, test, demonstration, proof of concept (POC), and training environments.
- Temporary environments for migrations and testing upgrades.
- Temporary environments for performance testing.

Oracle SOA Suite components

Oracle SOA Suite 12c is deployed on Oracle WebLogic Server and includes the following major components:

- Oracle SOA
- Oracle Service Bus (OSB)
- Oracle Business Process Management (Oracle BPM)
- Oracle Business Activity Monitoring (Oracle BAM)
- Oracle Application Development Framework (Oracle ADF)

The following diagram shows these components of Oracle SOA Suite when deployed on an Oracle WebLogic Server.



Oracle SOA Suite components deployed on Oracle WebLogic Server

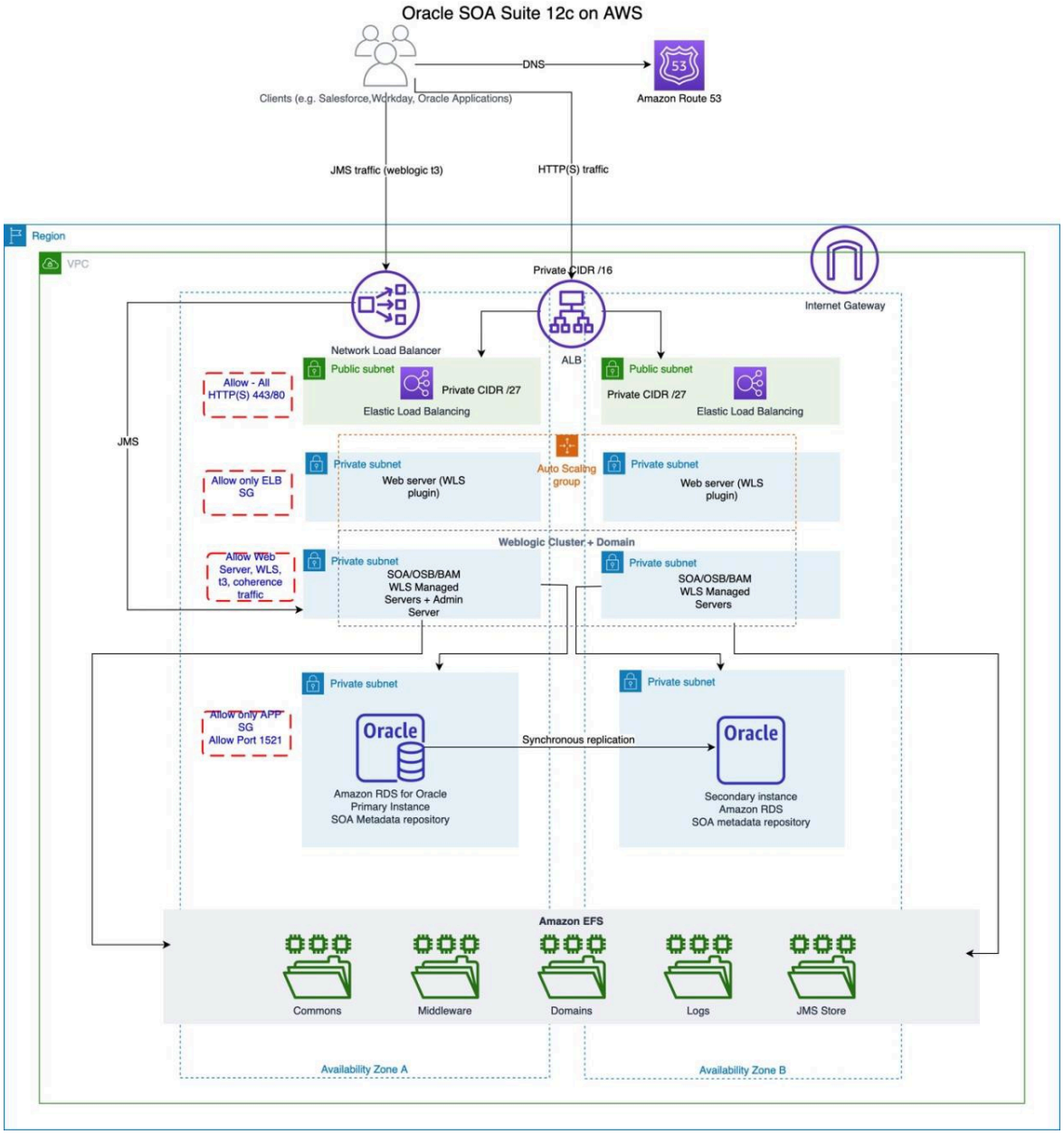
Each WebLogic Server deployment has a WebLogic domain, which typically contains multiple WebLogic Server instances (Managed Servers). A WebLogic domain is the basic unit of administration for WebLogic Server instances: it is a group of logically- related WebLogic Server resources. For example, you can have one WebLogic domain for each component of Oracle SOA Suite.

WebLogic Server instances can run on physical or virtual servers (such as Amazon Elastic Compute Cloud (Amazon EC2) or in containers. You can create a group of multiple WebLogic Managed Servers, known as a WebLogic Server Cluster. WebLogic Server Clusters support load balancing and failover and are required for high availability and scalability of your production deployments. You should deploy your WebLogic Server Cluster across multiple WebLogic Server Machines (Amazon EC2 instances) so that the loss of a single WebLogic Server Machine does not affect the availability of your application.

There are two types of WebLogic Server instances in a domain: a single Administration Server, and one or more Managed Servers. Each WebLogic Server instance runs its own Java Virtual Machine (JVM) and can be configured individually. You deploy and run the components of Oracle SOA Suite on the Managed Server instances in a WebLogic Server Cluster. The Administration Server is used to configure, manage, and monitor the resources in the domain, including the Managed Server instances.

Oracle SOA Suite reference architecture

The following reference architecture shows how you can deploy Oracle SOA Suite 12c on AWS.



High-level architecture of Oracle SOA Suite 12c on AWS

This reference architecture includes a separate WebLogic domain for each component of Oracle SOA Suite (Oracle SOA, OSB, Oracle BPM, Oracle BAM, and Oracle ADF). Each WebLogic domain has one

Administrative Server and multiple Managed Servers grouped into a WebLogic Server Cluster. The SOA metadata repository (MDS) is deployed on Amazon Relational Database Service (Amazon RDS). Amazon EFS is used for shared storage.

Traffic distribution

Amazon Route 53 domain name system (DNS) directs users to the application deployed on Oracle SOA Suite. Elastic Load Balancing (ELB) distributes incoming requests across the Oracle HTTP Servers and the WebLogic Managed Servers (for T3 traffic). The Oracle HTTP Server (OHS) along with the WebLogic Server Plugin serves as the reverse proxy routing the traffic to the respective WebLogic Managed Server instance. The WebLogic Managed Server instances host the various components of Oracle SOA Suite.

The Application Load Balancer load balances HTTP(S) traffic and the Network Load Balancer load balances T3 traffic.

Java Message Service (JMS) integration and T3 load balancing

Remote Method Invocation (RMI) communications in the WebLogic Server use the T3 protocol to transport data between WebLogic Servers and other Java programs, including clients and other WebLogic Server instances. When deploying on AWS, you can use a Network Load Balancer for load balancing T3 requests. WebLogic Server clients that use RMI can interoperate with a load balancer by tunneling T3 over HTTP/HTTPS or using T3 directly with the load balancer. Refer to WebLogic RMI Integration with Load Balancers on the Oracle Help Center for more information.

Storage

If you use file-based persistence, you must have storage for the Oracle WebLogic Server product binaries, common files and scripts, domain configuration files, logs, and persistence stores for Java Message Service (JMS) and Java Transaction API (JTA).

You can either use shared storage or Amazon Elastic Block Store (Amazon EBS) volumes to store these files.

Shared storage

Using Oracle Fusion Middleware, you can configure multiple WebLogic Server domains from a single Oracle home. This configuration allows you to install the Oracle home in a single location on a shared volume and reuse the Oracle home for multiple host installations.

Amazon EFS provides a simple, scalable, fully-managed elastic network file system (NFS). Amazon EFS can store the artifacts for Oracle Fusion Middleware, including Oracle home, Domain Home, Application Home, JTA Transaction Logs, and JMS Stores for persisting the JMS messages.

The reference architecture uses Amazon EFS for shared storage. The Oracle WebLogic product binaries, common files and scripts, domain configuration files, and logs are stored in Amazon EFS, which includes the *commons*, *domains*, *middleware*, and *logs* file systems.

Amazon EFS has two throughput modes for your file system: Bursting Throughput and Provisioned Throughput. With Bursting Throughput mode, throughput on Amazon EFS scales as your file system grows. With Provisioned Throughput mode, you can instantly provision the throughput of your file system in MiB/s, independent of the amount of data stored. For better performance, we recommend you select Provisioned Throughput mode while using Amazon EFS. With Provisioned Throughput mode,

you can provision up to 1024 MiB/s of throughput for your file system. You can change the file system throughput in Provisioned Throughput mode at any time after you create the file system.

We recommend mounting the Amazon EFS on Amazon EC2 with a DNS name and the recommended NFS mount options. For more information, refer to Mounting on Amazon EC2 with a DNS name and Recommended NFS mount options.

High availability

The Managed Servers for each domain are grouped into a WebLogic Server Cluster that spans two Availability Zones (AZs). Use Availability Zones to operate production applications and databases that are more highly available, fault tolerant, and scalable than is possible from a single data center. In the unlikely event of failure of one AZ, user requests are routed to your application instances in the second zone. This ensures that your application continues to remain available at all times.

You can add and remove Oracle HTTP Server (OHS) instances from your load balancer as your needs change, either manually or with Amazon EC2 Auto Scaling, without disrupting the overall flow of information. ELB ensures that only healthy instances receive traffic by detecting unhealthy instances and rerouting traffic across the remaining healthy instances. If an instance fails, ELB automatically reroutes the traffic to the remaining running instances. If a failed instance is restored, ELB restores the traffic to that instance.

Oracle Metadata Services (MDS) Repository database contains the metadata for Oracle Fusion Middleware components, such as the Oracle Application Developer Framework. The MDS is hosted on Amazon RDS for Oracle. Amazon RDS for Oracle makes it easy to use replication to enhance availability and reliability for production workloads. Using the Multi-AZ deployment option, you can run MDS with high-availability and a built-in automated failover from your primary database to a synchronously-replicated secondary database in case of a failure. Since the endpoint for your database instance remains the same after a failover, applications can resume database operations as soon as the failover is complete without the need for manual administrative intervention. You can run Amazon RDS for Oracle under two different licensing models: License Included or Bring-Your-Own-License (BYOL). In the License Included service model, you do not need to purchase Oracle licenses separately; the Oracle Database software has been licensed by AWS.

Amazon EFS is designed to be highly available and durable. Your data in Amazon EFS is redundantly stored across multiple AZs, which means that your data is available in the unlikely event of an AZ failure.

State management

For the stateful components of Oracle SOA Suite such as Oracle BAM web services, you can configure Oracle WebLogic Server to replicate the HTTP session state in memory to another Managed Server in the Oracle WebLogic Server Cluster. Oracle WebLogic Server uses a cookie to track the location of the Managed Servers hosting the primary and the replica copies of the session state. If the Managed Server hosting the primary copy of the session state fails, Oracle WebLogic Server can retrieve the HTTP session state from the replica. For more information about HTTP session state replication, refer to Oracle WebLogic Server 12c: Managing HTTP Sessions in a Cluster.

High availability and state management of Oracle SOA Suite 12c components

The following table lists the various Oracle SOA Suite 12c components and the strategies used for failure protection, high availability, and state management.

Table 1: High availability and state management of Oracle SOA Suite 12c components

Domain	Oracle SOA Suite component	High availability	State management
Oracle SOA	SOA Service Infrastructure	Active-Active	Stateless
	SOA Admin	Singleton (Active-Passive)	Stateless
	SOA BPEL	Active-Active	Stateless
	BPM Suite	Active-Active	
	B2B UI	Active-Active	Stateful (HTTP Session)
	Mediator	Active-Active	Stateless
	Human workflow	Active-Active	Stateless
	Oracle WSM	Active-Active	Stateless
	Oracle UMS	Active-Active	
	Oracle JCA Adapters	Active-Active	Stateless
Oracle BAM	Web Apps	Active-Active	BAM web applications are stateful and need session replication for high availability.
	BAM Server	Singleton Active-Passive	Stateless
OSB		Active-Active	

Failure scenarios

The following section describes how failure scenarios are addressed for the various components in the architecture.

Oracle HTTP server failure

You can also configure an Auto Scaling group with EC2 instances running Oracle HTTP Server in multiple AZs. Amazon EC2 Auto Scaling spins up new instances in case of failure of the Oracle HTTP Server.

If the underlying host for the OHS experiences a failure, you can also configure automatic recovery for Amazon EC2 instances to recover the failed server instances. When using automatic recovery for Amazon EC2 instances, several system status checks monitor the instance and the other components required for your instance. Among other things, the system status checks monitor for loss of network connectivity, loss of system power, software issues on the physical host, and hardware issues on the physical host. If a system status check of the underlying hardware fails, the instance is rebooted (on new hardware if necessary), but it retains its instance ID, IP address, Elastic IP addresses, EBS volume attachments, and other configuration details.

WebLogic Server node failure

Because SOA components run in an Oracle WebLogic Server Cluster, the components remain highly available as long as the components are running in Active-Active mode and deployed across multiple WebLogic Managed Servers.

WebLogic managed server failure

In the event of a Managed Server failure, Oracle WebLogic Node Manager restarts the WebLogic Managed Server.

Administration server failure

The Administration Server is used to configure, manage, and monitor the resources in the domain, including the Managed Server instances. Because the failure of the Administration Server does not affect the functioning of the Managed Servers in the domain, the Managed Servers continue to run, and your application is still available.

However, if the Administration Server fails, the WebLogic Server Administration Console is unavailable and you cannot make changes to the domain configuration.

If the underlying host for the Administration Server experiences a failure, you can use the automatic recovery for Amazon EC2 instances to recover the failed server instances.

Another option is to put the Administration Server instances in an Amazon EC2 Auto Scaling group that spans multiple AZs, and set the minimum and maximum size of the group to one. Automatic scaling ensures that an instance of the Administration Server is running in the selected AZs. This configuration ensures high availability of the Administration Server if an AZ failure occurs.

MDS database failure

The database instance failure can be handled by using Amazon RDS for Oracle with active-standby setup using Multi-AZ deployment. Refer to Multi-AZ deployments for high availability for more details on Multi-AZ deployments.

Considerations for deploying Oracle SOA Suite 12c on AWS

The following are some points to consider when deploying Oracle SOA Suite 12c on AWS.

- Configure Oracle WebLogic Server in unicast mode and make sure to allow the port number in the AWS Security Group configuration.
- Make sure to set the appropriate value of the Java property `networkaddress.cache.ttl` so that appropriate caching policy is set in the JVM for successful DNS lookups. See *Setting the JVM TTL for DNS Name Lookups* in Multi-AZ deployments for high availability for more information.
- Allow appropriate inbound and outbound traffic in the application security groups including the ports configured for T3, HTTP(S), Internal WebLogic Server Cluster intercommunication, and Internal Coherence cluster communication.
- If SOA nodes are not synchronizing after deploying the BPEL process, tune the Linux kernel parameters `net.core.rmem_max` and `net.core.wmem_max`. Refer to the Oracle Support document Nodes Not Syncing In 12C (Doc ID 2315273.1) for more information.

Scalability

When you use AWS, you can scale your application easily because of the elastic nature of the cloud. You can scale your application vertically and horizontally.

Vertical scaling

You can vertically scale (scale up) your application by changing the Amazon EC2 instance type on which your Oracle WebLogic Managed Servers are deployed to a larger instance type, and then increasing the Oracle WebLogic JVM heap size. You can modify the Java heap size with the `-Xms` (initial heap size) and `-Xmx` (maximum heap size) parameters. Ideally, you should set both the initial heap size (`-Xms`) and the maximum heap size (`-Xmx`) to the same value to minimize garbage collections and optimize performance.

For example, you can start with an `r5.large` instance with 2 vCPUs and 16 GiB RAM, and scale up all the way to an `x1e.32xlarge` instance with 128 vCPUs and 3,904 GiB RAM. For the most updated list of Amazon EC2 instance types, refer to Amazon EC2 Instance Types.

After you select a new instance type, you simply restart the instance for the changes to take effect. Typically, the resizing operation is completed in a few minutes, the Amazon EBS volumes remain attached to the instances, and no data migration is required.

Horizontal scaling

You can horizontally scale (scale out) your application by adding more Managed Servers to your Oracle WebLogic Server Cluster depending on the user traffic or on a particular schedule. You launch new EC2 instances to deploy, and configure additional Managed Servers, add them to the Oracle Server Cluster, and register your instances with ELB.

You can automate this process with AWS Auto Scaling and Oracle WebLogic scripting. For more information, refer to the next section in this document, *Configure automatic scaling on your Oracle WebLogic server cluster*.

AWS Auto Scaling for scaling out your Oracle WebLogic Server Cluster also requires scripting, which can be an additional technical investment. Although we recommend that you use AWS Auto Scaling, sometimes you may not have the time or the technical resources to implement it while migrating your Oracle WebLogic Server application to AWS. A simpler alternative might be to use *standby* instances.

Standby instances

To meet extra capacity requirements, additional instances of the WebLogic Managed Servers are preinstalled and configured on EC2 instances. These standby instances can be shut down until the extra capacity is required. You do not incur compute charges when instances are shut down, you incur only Amazon Elastic Block Store (Amazon EBS) storage charges. These pre-installed standby instances provide you with the flexibility to meet additional capacity when you need it.

Configure automatic scaling on your Oracle WebLogic server cluster

You can use AWS Auto Scaling to horizontally scale your applications based on demand. This helps you to maintain steady, predictable performance at the lowest possible cost. For example, you can configure AWS Auto Scaling to automatically create and add more Managed Servers to your Oracle WebLogic Server Cluster as the traffic increases, and to stop and remove Managed Servers from the Oracle WebLogic Server Cluster as the traffic decreases. For more information about AWS Auto Scaling, refer to the AWS Auto Scaling Documentation.

To configure automatic scaling for your Oracle WebLogic Server Cluster on AWS, complete these major steps:

1. **Install and configure Oracle WebLogic Server** – The first step is to configure Amazon EFS for shared storage, install Oracle WebLogic, and configure the Oracle WebLogic Domain and the Oracle WebLogic Server Cluster. Amazon EFS stores the WebLogic product binaries, common files and scripts, domain configuration files, and logs.
2. **Configure AWS Auto Scaling** – Next, you have to configure AWS Auto Scaling to launch and terminate EC2 instances—or Oracle WebLogic Machines—based on the application workload.
3. **Configure WebLogic scaling scripts** – Finally, you create WebLogic Scripting Tool (WLST) scripts. These scripts create and add or remove the Managed Servers from the WebLogic Server Cluster when AWS Auto Scaling launches or terminates EC2 instances in the Auto Scaling group.

Migrating Oracle SOA Suite 12c to AWS

When migrating Oracle SOA Suite 12c to AWS, consider the architecture described in this whitepaper, perform a trial migration and validate it, then iterate one or two times before migrating to production. Review the migration approach to ensure the least amount of downtime for the business during the production cutover. Before starting the migration, you must create an Amazon Virtual Private Cloud (Amazon VPC) with the required subnets on AWS. You must also set up network connectivity from on-premises to AWS using a VPN connection or AWS Direct Connect.

The following sections describe the process to migrate the various Oracle SOA Suite 12c components to AWS.

Migrating the web and application tier

You can migrate the Oracle SOA Suite 12c Web and Application tier components to AWS using the following approaches.

VM or server level migration

You can migrate or clone the entire virtual machine (VM) or server hosting the Oracle SOA Suite 12c components using tools such as AWS Application Migration Service (AWS MGN), the next generation of CloudEndure Migration. The following steps provide an overview of the migration process:

1. Use AWS MGN (or a similar tool) to migrate the servers or VMs from the on-premises deployment. AWS MGN ensures the data is synchronized in real time and minimizes the cutover window.
2. Use Amazon EFS as the shared storage mounted on the application tier nodes. Reconfigure Oracle WebLogic Server to use Amazon EFS for shared data such as the artifacts for Oracle Fusion Middleware, including Oracle home, Domain Home, Application Home, JTA Transaction Logs, and JMS Store for persisting the JMS messages.
3. Migrate the Oracle Metadata Services (MDS) Repository as described in the *Migrating the Oracle MDS repository* section of this document.
4. Update the metadata data source in WebLogic with the new database connection information of the Oracle Metadata Services (MDS) Repository.
5. Open the WebLogic Administration Console to verify that the Domain, cluster, and managed servers are configured correctly.
6. Configure the Web servers and ELB (Network Load Balancer and Application Load Balancer).
7. Start the servers and test.

Migration using Oracle Fusion Middleware utilities

The second option is to migrate the Oracle SOA Suite 12c components using the built-in Oracle Fusion Middleware utilities such as copy binary/copy config, and paste binary/paste config. The following steps provide a high-level overview of migration. For more details, refer to Overview of Procedures for Moving from a Source to a Target Environment on the Oracle Help Center.

1. Launch the EC2 instances with Oracle Linux or a certified operating system for hosting the Oracle SOA Suite 12c components.
   a. Apply the required prerequisite patches for Oracle SOA Suite 12c, including the security patches.
   b. Create an Amazon EBS-backed Linux Amazon Machine Image (AMI) and preserve the copy for future use.
2. Install a new Oracle Home. You can also copy the Oracle Home binaries from the existing on-premises instance.
3. Use Copy binary and Copy Config commands on each application node to back up Oracle SOA Suite.
4. Back up the web servers on each node.
5. Back up the files on the on-premises NFS drive shared between the application tier nodes.
6. Copy the backup of the Oracle SOA Suite files and the Web Servers to the respective instances on AWS.
7. Restore the Oracle SOA Suite components using the paste library and paste config commands on the application tier nodes on AWS.
8. Complete the Fusion Middleware steps for Moving from a Test to a Production Environment. The RCU step is not required if you have already migrated the Oracle MDS Repository (refer to the *Migrating the Oracle MDS Repository* (p. 15) section later in this document).
9. Restore the Web Server backup on AWS.
10. Mount the Amazon EFS share on the application tier nodes and copy the shared data from on-premises.
11. Migrate the Oracle MDS Repository. (Refer to the *Migrating the Oracle MDS repository* section of this document.)
12. Update the metadata data source in WebLogic with the new database connection information of the MDS Repository.
13. Open the WebLogic Administration Console to verify that the domain, cluster, and managed servers are configured correctly.
14. Configure the Web servers and ELB (Network Load Balancer and Application Load Balancer).
15. Start the servers and test.

Migration using new installation and configuration

1. Launch the EC2 instances with Oracle Linux or a certified operating system for hosting the Oracle SOA Suite 12c components.
   a. Apply the required prerequisite patches for Oracle SOA Suite 12c, including the security patches.
   b. Create an Amazon EBS-backed Linux AMI and preserve the copy for future use.
2. Migrate the Oracle MDS Repository. (Refer to the *Migrating the Oracle MDS repository* section of this document.)
3. Install the Oracle Home.
4. Mount the Amazon EFS share on the application tier nodes.
5. Install Oracle SOA Suite 12c software on a single node.
6. Configure an Oracle SOA Suite domain for each component (Oracle SOA, OSB, Oracle BPM, Oracle BAM, and Oracle ADF).
7. Horizontally scale out to multiple nodes as per the requirements and complete the domain configuration.
8. Configure data sources and JMS connection pools in WebLogic Server.
9. Install the Web server and configure it (you can copy the configuration files from on-premises).
10. Configure the Web servers and ELB (Network Load Balancer and Application Load Balancer).
11. Start the servers and test.

Migrating the Oracle MDS repository

The steps for MDS Repository migration differ depending upon your database configuration.

For databases on Amazon EC2

For initial data migration, copy the regular Oracle Recovery Manager (Oracle RMAN) backup files along with archive log files for recovery. Then, use AWS Database Migration Service (AWS DMS) to synchronize the databases using change data capture (CDC). Refer to Migrate an on-premises Oracle database to Oracle on Amazon EC2 for prescriptive guidance.

For databases on Amazon RDS

For initial migration, copy the export dump files taken from the on-premises database to restore on Amazon RDS using the import tools. Then, use AWS DMS to synchronize the databases using change data capture (CDC). Refer to Migrate an on-premises Oracle database to Amazon RDS for Oracle for prescriptive guidance.

Monitoring your infrastructure

After you migrate your Oracle WebLogic applications to AWS, you can continue to use the monitoring tools you are familiar with to monitor your Oracle SOA Suite 12c environment.

You can use Oracle Enterprise Manager Fusion Middleware Control, the Oracle WebLogic Server Administration Console, or the command line (using the WebLogic Scripting Tool (WLST) state command) to monitor your Oracle WebLogic Server infrastructure components. This includes WebLogic domains, Managed Servers, and clusters. You can also monitor the Java applications deployed and get information such as the state of your application, the number of active sessions, and response times. For more information about how to monitor Oracle WebLogic Server, refer to the Oracle WebLogic Server documentation.

You can also use Amazon CloudWatch to monitor AWS Cloud resources and the Oracle SOA Suite 12c components. Amazon CloudWatch enables you to monitor your AWS resources in near real-time, including Amazon EC2 instances, Amazon EBS volumes, Amazon EFS, ELB, and Amazon RDS instances. Metrics such as CPU utilization, latency, and request counts are provided automatically for these AWS resources. Amazon CloudWatch can also monitor your own logs or custom application and system metrics, such as memory usage, transaction volumes, or error rates.

If your Oracle SOA Suite deployment uses a database deployed on Amazon RDS, you can use Amazon RDS Enhanced Monitoring to monitor your database. Enhanced Monitoring gives you access to over 50 metrics, including CPU, memory, file system, and disk I/O. You can also view the processes running on the database instance and their related metrics, including percentage of CPU usage and memory usage. You can also set up custom metrics in Amazon CloudWatch and then set up alarms, dashboards, widgets, and more.

Securing your environment

AWS provides a secure global infrastructure, plus a range of features that you can use to help secure your systems and data in the cloud. To learn more about AWS Security, refer to the AWS Security Center.

AWS security group configuration

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. For each security group, you add rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. For more information on security groups, refer to Amazon EC2 security groups for Linux instances.

The following table includes some of the recommended security group rules related to Oracle SOA Suite 12c. This list is not an exhaustive list of inbound rules it is only meant to serve as a reference.

Table 2: Suggested Security Group rules for Oracle SOA Suite 12c

Security Group	Inbound traffic type	Inbound port (example*)
Load Balancer Security Group	ELB http	80
	ELB https (SSL)	443
Application Security Group	SOA Admin Server	7001
	SOA Managed servers	7003, 7005, 7007
	OSB Admin server	8001
	OSB Managed servers	8003
	ADF Admin server	9001
	ADF Managed servers	9003, 9005
	BPM Admin Server	10001
	BPM Managed Servers	10003, 10005, 10007
	Any other SOA component	<Add as per your configuration>
	Oracle Coherence port	9991
	WebLogic Node manager	5555 - 5558
Database Security Group	Oracle Database	1521
Web Security Group	Oracle HTTP Server (OHS)	7777

**The port numbers might change depending on your configuration.*

Conclusion

AWS can be an extremely cost-effective, secure, scalable, high-performing, and flexible option for deploying Oracle SOA Suite 12c components. By deploying Oracle SOA Suite 12c applications on the AWS Cloud, you can reduce costs and simultaneously enable capabilities that may not be possible or cost-effective if you deployed your application in an on-premises data center.

Further reading

For additional information, refer to:

- Database Migration Service Step-by-Step Walkthroughs
- Using the Oracle Repository Creation Utility on Amazon RDS for Oracle
- AWS MGN compared with CloudEndure Migration

Contributors

Contributors to this document include:

- Dhawalkumar Patel, Senior Solutions Architect, Amazon Web Services
- Ashok Sundaram, Senior Solutions Architect, Amazon Web Services
- Bharath Terala, Senior Partner Solutions Architect, Amazon Web Services
- Ravikiran Dundigalla, Senior Principal Consultant, Apps Associates LLC

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Document updated (p. 21)	Minor updates.	October 19, 2022
Initial publication (p. 21)	Whitepaper published.	April 1, 2020

Notices

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.