Oracle WebLogic Server 12c on AWS

December 2018

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This whitepaper provides guidance on how to deploy Oracle WebLogic Server 12c-based applications on AWS. This paper provides a reference architecture and information about best practices for high availability, security, scalability, and performance when you deploy Oracle WebLogic Server 12c-based applications on AWS. Also included is information about cost optimization using AWS Auto Scaling.

The target audience of this whitepaper is Solution Architects, Systems Architects, and System Administrators with a basic understanding of cloud computing, AWS, and Oracle WebLogic 12c.

Introduction

Many enterprises today rely on J2EE application servers for deploying their mission critical applications. Oracle WebLogic Server is a popular Java application server for deploying such applications.

You can use various AWS services to deploy Oracle WebLogic Server 12c-based applications on AWS in a secure, highly available, and cost-effective manner. With auto scaling, you can dynamically scale the compute resources required for your application, thereby keeping your costs low, and using Amazon Elastic File System (EFS) for shared storage.

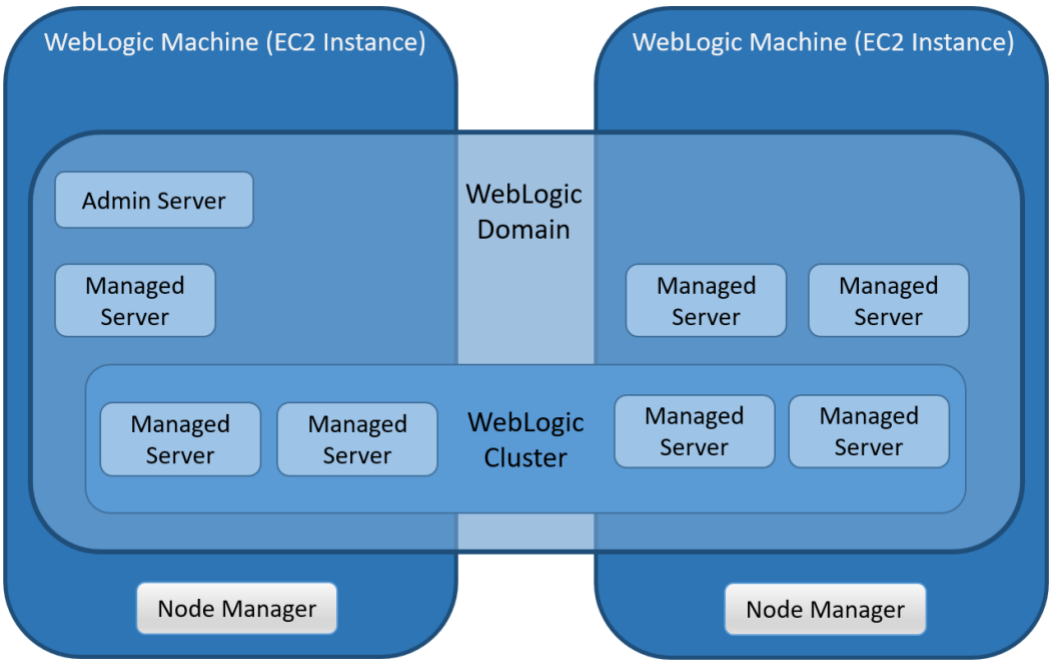
This whitepaper assumes that you have a basic understanding of Amazon Web Services. For an overview of AWS Services, see Overview of Amazon Web Services.

Oracle WebLogic on AWS

It is important to have a good understanding of the architecture of Oracle WebLogic Server 12c (Oracle WebLogic) and the major WebLogic components to successfully deploy and configure it on AWS.

Oracle WebLogic Components

This diagram shows the major components of Oracle WebLogic Application Server.



Each WebLogic deployment has a WebLogic Domain, which typically contains multiple WebLogic Server instances. 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 application.

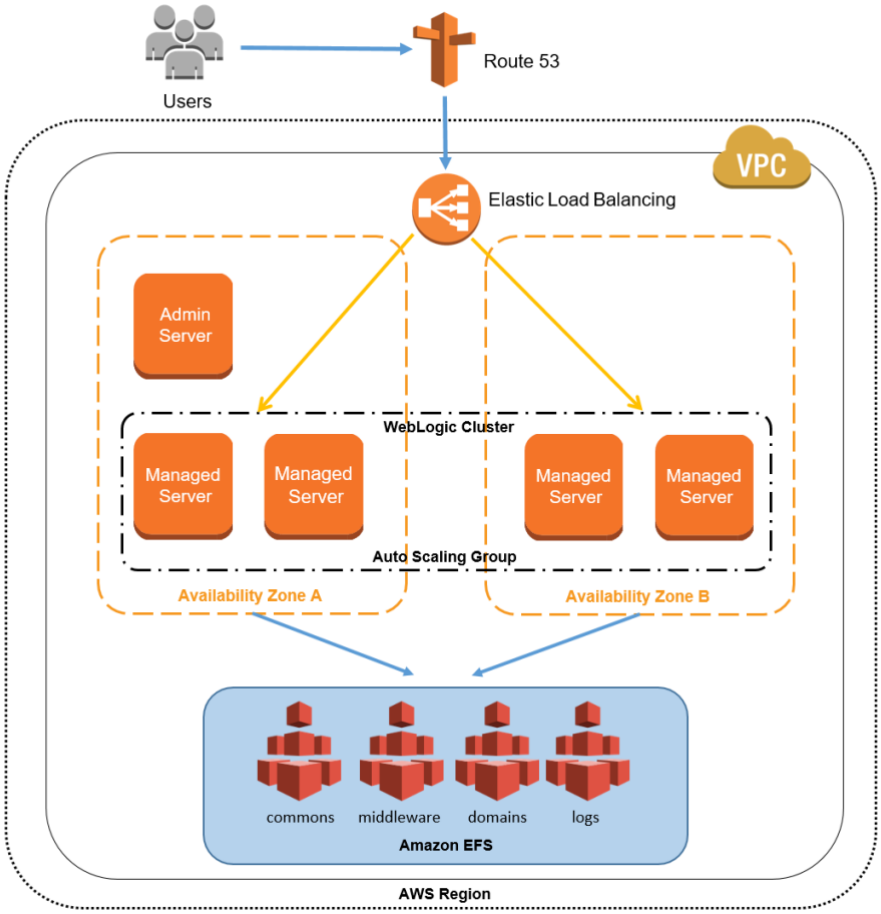
There are two types of WebLogic Server instances in a domain: a single Administration Server and one or more Managed Servers. Each WebLogic Server instance runs its own Java Virtual Machine (JVM) and can be configured individually. You deploy and run your web applications, EJBs, and other resources on the Managed Server instances. The Administration Server is used

to configure, manage, and monitor the resources in the domain, including the Managed Server instances.

WebLogic Server instances, referred to as WebLogic Server Machines, can run on physical or virtual servers (such as Amazon EC2) or in containers. The Node Manager is a utility used to start, stop, or restart the Administration server or Managed Server instances. You can create a group of multiple WebLogic Managed Servers, which is known as a WebLogic cluster. WebLogic clusters support load balancing and failover and are required for high availability and scalability of your production deployments. You should deploy your WebLogic cluster across multiple WebLogic Machines so that the loss of a single WebLogic Machine does not impact the availability of your application.

Oracle WebLogic Architecture on AWS

This reference architecture diagram shows how you can deploy a web application on Oracle WebLogic on AWS.

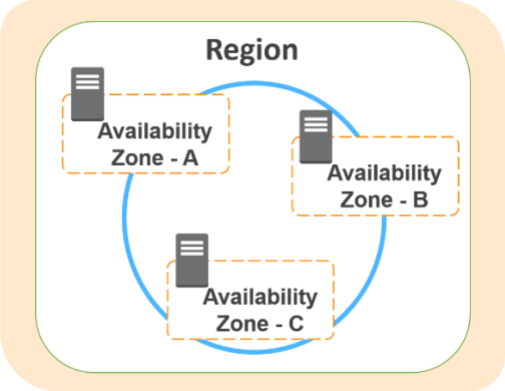


This is a basic, combined-tier architecture, with static HTTP pages, servlets, and EJBs that are deployed together in a single WebLogic cluster. You can also deploy the static HTTP pages and servlets to a separate WebLogic cluster, and the EJBs to another WebLogic cluster. For more information about WebLogic architectural patterns, see the Oracle WebLogic Server documentation.

This reference architecture includes a WebLogic domain with one Administrative Server and multiple Managed Servers. These Managed Servers are part of a WebLogic cluster and are deployed on EC2 instances (WebLogic Machines) across two Availability Zones for high availability. The application is deployed to the Managed Servers in the cluster that spans the two Availability Zones. Amazon EFS is used for shared storage.

AWS Availability Zones

The AWS Cloud infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple, physically separated and isolated Availability Zones which are connected with low latency, high throughput, and highly redundant networking. Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, and housed in separate facilities as shown in the following diagram.



These Availability Zones enable you to operate production applications and databases that are more highly available, fault tolerant, and scalable than is possible from a single data center. You can deploy your application on EC2 instances across multiple zones. In the unlikely event of failure of one Availability Zone, user requests are routed to your application instances in the second zone. This ensures that your application continues to remain available at all times.

Traffic Distribution and Load Balancing

Amazon Route 53 DNS is used to direct users to your application deployed on Oracle WebLogic on AWS. Elastic Load Balancing (ELB) is used to distribute incoming requests across the WebLogic Managed Servers deployed on Amazon EC2 instances in multiple Availability Zones. The load balancer serves as a single point of contact for client requests, which enables you to increase the availability of your application.

You can add and remove WebLogic Managed Server instances from your load balancer as your needs change, either manually or with 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.

Use Multiple Availability Zones for High Availability

Each Availability Zone is isolated from other Availability Zones and runs on its own physically distinct, independent infrastructure. The likelihood of two Availability Zones experiencing a failure at the same time is relatively small. To ensure high availability of your application, you can deploy your WebLogic Managed Server instances across multiple Availability Zones.

You then deploy your application on the Managed Servers in the WebLogic cluster, which spans two Availability Zones. In the unlikely event of an Availability Zone failure, user requests to the zone with the failure are routed by Elastic Load Balancing to the Managed Servers deployed in the second Availability Zone. This ensures that your application continues to remain available, regardless of a zone failure.

You can configure WebLogic to replicate the HTTP session state in memory to another Managed Server in the WebLogic cluster. WebLogic tracks the location of the Managed Servers hosting the primary and the replica of the session state using a cookie. If the Managed Server hosting the primary copy of the session state fails, WebLogic can retrieve the HTTP session state from the replica. For more information about HTTP session state replication, see the Oracle WebLogic documentation.

For shared storage, you can use Amazon EFS, which is designed to be highly available and durable. Your data in Amazon EFS is redundantly stored across multiple Availability Zones, which means that your data is available if there is an Availability Zone failure. For information about how to use Amazon EFS for shared storage, see the Shared Storage section.

Administration Server High Availability

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 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 Amazon EC2 Auto Recovery feature to recover the failed server instances. When using Amazon EC2 Auto Recovery, several system status checks monitor the instance and the other components that need to be running for your instance to function as expected. Among other things, the system status checks look 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 will be rebooted (on new hardware if necessary) but will retain its instance ID, IP address, Elastic IP addresses, EBS volume attachments, and other configuration details.

Another option is to put the Administration Server instances in an Auto Scaling group that spans multiple Availability Zones, and set the minimum and maximum size of the group to one. Auto Scaling ensures that an instance of the Administration Server is running in the selected Availability Zones. This solution ensures high availability of the Administration Server if a zone failure occurs.

Storage

If you use file-based persistence, you must have storage for the WebLogic product binaries, common files and scripts, the domain configuration files, logs, and persistence stores for JMS and JTA. You can either use shared storage or Amazon EBS volumes to store these files.

Shared Storage

To store the shared files related to your WebLogic deployment, you can use Amazon EFS, which supports NFSv4 and will be mounted by all the instances that are part of the WebLogic cluster. In the reference architecture, we use Amazon EFS for shared storage. The WebLogic product binaries, common files and scripts, the domain configuration files, and logs are stored in Amazon EFS, which includes the *commons*, *domains*, *middleware*, and *logs* file systems. This table describes each of these file systems.

File System	Description
commons	For common files, such as installation files, response files, and scripts.
domains	For WebLogic Domain files, such as configuration, runtime, and temporary files.
middleware	For binaries, such as Java VM and Oracle WebLogic installation.
logs	For log files.

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.

If you are deploying your application in a region where Amazon EFS is not yet available, there are several third-party products by vendors such as NetApp and SoftNAS available on the AWS Marketplace that offer a shared storage solution on AWS.

Amazon EBS Volumes

In this reference architecture, we use Amazon EFS for shared storage. You can also deploy Oracle WebLogic on AWS without using shared storage. Instead, you can use Amazon EBS volumes attached to your Amazon EC2 instances for storage. Make sure to select the General Purpose (gp2) volume type for storing the WebLogic product binaries, common files and scripts, the domain configuration files, and logs. GP2 volumes are backed by solid-state drives (SSDs) designed to offer single-digit millisecond latencies and are suitable for use with Oracle WebLogic.

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, or scale up, your application simply by changing the EC2 instance type on which your WebLogic Managed Servers are deployed to a larger instance type, and then increasing the 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 r4.large instance with 2 vCPUs and 15 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, see the Amazon EC2 Instance Types page on the AWS website.

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, or scale out, your application by adding more Managed Servers to your WebLogic 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 WebLogic cluster, and *register* your instances with the ELB.

You can automate this process with AWS Auto Scaling and WebLogic scripting. For more information, see the Auto Scaling your Oracle WebLogic Cluster section.

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

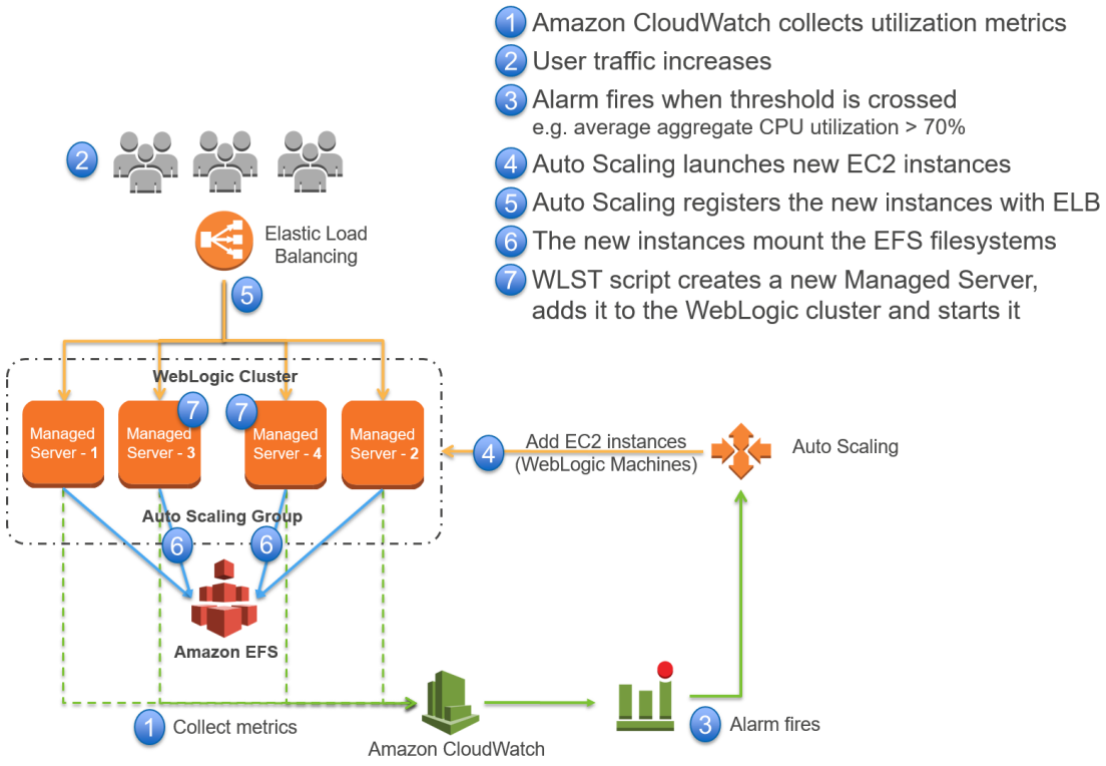
Standby Instances

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 preinstalled standby instances provide you the flexibility to meet additional capacity when you need it.

Auto Scaling your Oracle WebLogic 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 WebLogic cluster as the traffic increases, and to stop and remove Managed Servers from the WebLogic cluster as the traffic decreases. For more information about Auto Scaling, see the [Amazon EC2 Auto Scaling] documentation.

This diagram shows how AWS Auto Scaling works with Oracle WebLogic. In this example, we use Amazon EFS for shared storage.



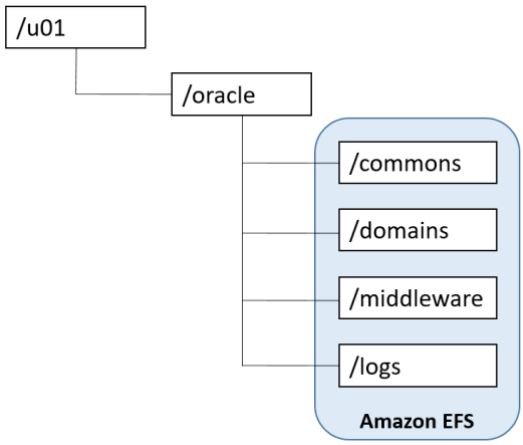
To Auto Scale your WebLogic cluster on AWS you must complete these major steps.

1. Install and configure WebLogic – The first step is to configure Amazon EFS for shared storage, install Oracle WebLogic, and configure the WebLogic Domain and the WebLogic cluster. Amazon EFS is used to store the WebLogic product binaries, common files and scripts, the 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 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 cluster when AWS Auto Scaling launches or terminates EC2 instances in the auto scaling group.

Configure Oracle WebLogic

To configure Oracle WebLogic and setup shared storage, you must complete these high-level steps.

1. Create the *commons*, *domains*, *middleware*, and *logs* file systems on Amazon EFS, as described in the Shared Storage section.
2. Create an EC2 instance for deploying the WebLogic Administration Server and mount the EFS file systems. In the reference architecture, we have created the following directory structure to store the WebLogic binaries, domain configurations, common scripts and logs.



3. Install Oracle WebLogic. The ORACLE_HOME directory should be located on a shared folder (/middleware) on EFS.
4. Create the WebLogic domain. You can use the *Basic WebLogic Server Domain Template* in the /templates/wls/wls.jar' directory to create the domain.
5. Create a WebLogic cluster in the domain and set the cluster messaging mode to *Unicast*.

Configure AWS Auto Scaling

To configure AWS Auto Scaling to launch and terminate EC2 instances (or *WebLogic Machines*) based on the application load, you must complete the following high-level steps. For more details on Auto Scaling, see the Amazon EC2 Auto Scaling documentation on the AWS website.

1. Create a launch configuration and an Auto Scaling group.
2. Create the scale in and scale out policies. For example, you can create a scaling policy to add an instance when the CPU utilization is >80 % and to remove an instance when the CPU utilization is <60 %.
3. If you are using in-memory session persistence, Oracle WebLogic replicates the session data to another Managed Server in the cluster. You should ensure that the Auto Scaling scale down process terminates only one Managed Server at a time, to make sure you do not destroy the master and the replica of the session at the same time.

For detailed, step-by-step instructions on how to configure Auto Scaling, see the Amazon EC2 Auto Scaling documentation on the AWS website.

Configure WebLogic Scaling Scripts

Based on the traffic to your application, Auto Scaling can create and add new EC2 instances (scaling out), or remove existing EC2 instances (scaling in) from your auto scaling group. You must create the following scripts to automate the configuration of WebLogic in an auto-scaled environment.

- EC2 configuration scripts – These scripts mount the EFS filesystems, invoke the WLST scripts to configure and start the WebLogic Managed Server on the startup of the EC2 instance, and invoke the WLST scripts to stop the WebLogic Managed Server on shutdown of the EC2 instance.

You can pass this script with the EC2 user data. For detailed information, see the Amazon EC2 documentation on the AWS website.

- WebLogic Scripting Tool (WLST) scripts – WLST is a command-line scripting interface used to manage WebLogic Server instances and domains. These scripts create and add the Managed Server to your WebLogic cluster when Auto Scaling adds a new EC2 instance to the Auto Scaling group. These scripts also stop and remove the Managed Server from your WebLogic cluster when Auto Scaling removes the EC2 instance from the Auto Scaling group. For more information, see the Oracle WLST documentation.

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 WebLogic environment and the application you deployed on WebLogic.

You can use Fusion Middleware Control, the Oracle WebLogic Server Administration Console or the command line (using the WSLT state command) to monitor your Oracle WebLogic 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, see the Oracle WebLogic documentation.

You can also use Amazon CloudWatch to monitor AWS Cloud resources and the applications you run on AWS. Amazon CloudWatch enables you to monitor your AWS resources in near real-time, including Amazon EC2 instances, Amazon EBS volumes, Amazon EFS, ELB load balancers, and Amazon RDS DB instances. Metrics such as CPU utilization, latency, and request counts are provided automatically for these AWS resources. You can also supply your own logs or custom application and system metrics, such as memory usage, transaction volumes, or error rates, which Amazon CloudWatch will also monitor.

With Amazon CloudWatch alarms, you can set a threshold on metrics and trigger an action when that threshold is exceeded. For example, you can create an alarm that is triggered when the CPU utilization on an EC2 instance crosses a threshold. You can also configure a notification of the event to be sent through SMS or email. Real-time alarms for metrics and events enable you to minimize downtime and potential business impact.

If your application uses a database deployed on Amazon RDS, you can use the Enhanced Monitoring feature of Amazon RDS 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 DB instance and their related metrics, including percentage of CPU usage and memory usage.

AWS Security and Compliance

The AWS Cloud security infrastructure has been architected to be one of the most flexible and secure cloud computing environments available today. Security on AWS is very similar to security in your on-premises data center, but without the costs and complexities involved in protecting facilities and hardware. 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, see the AWS Security Center.

AWS Compliance enables customers to understand the robust controls in place at AWS to maintain security and data protection in the cloud. AWS engages with external certifying bodies and independent auditors to provide customers with extensive information regarding the policies, processes, and controls established and operated by AWS. To learn more about AWS Compliance, see the AWS Compliance Center.

The AWS Security Model

The AWS infrastructure has been architected to provide an extremely scalable, highly reliable platform that enables you to deploy applications and data quickly and securely.

Security in the cloud is different than security in your on-premises data centers. When you move computer systems and data to the cloud, security responsibilities become shared between you and your cloud service provider. In the AWS cloud model, AWS is responsible for securing the underlying infrastructure that supports the cloud, and you are responsible for securing workloads that you deploy in AWS. This shared security responsibility model can reduce your operational burden in many ways, and gives you the flexibility you need to implement the most applicable security controls for your business functions in the AWS environment.

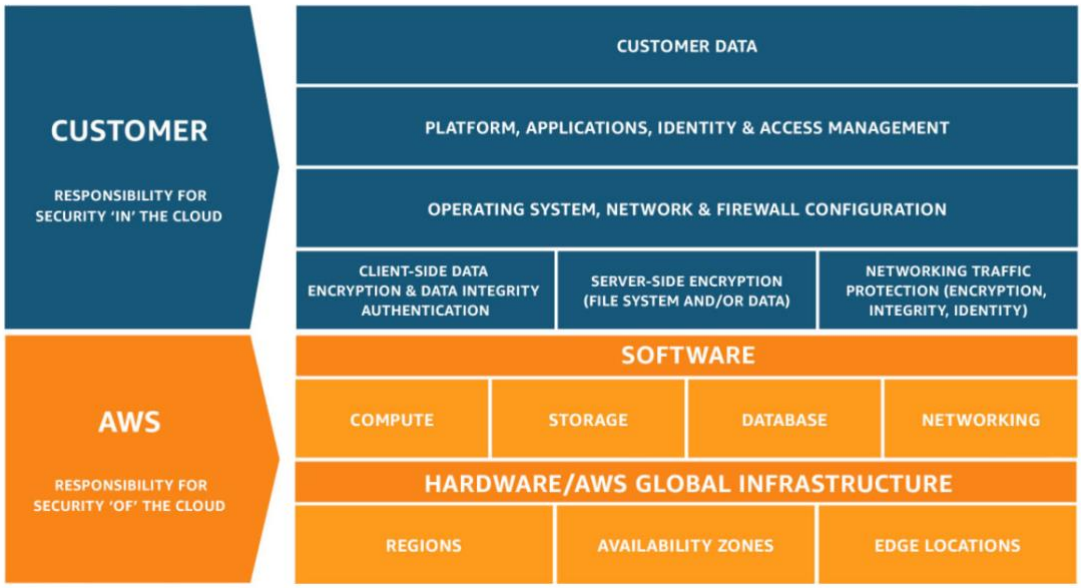


Figure 6: The AWS shared responsibility model

When you deploy Oracle WebLogic applications on AWS, we recommend that you take advantage of the various security features of AWS, such as AWS Identity and Access Management, monitoring and logging, network security, and data encryption.

AWS Identity and Access Management

With AWS Identity and Access Management (IAM), you can centrally manage your users and their security credentials, such as passwords, access keys, and permissions policies, which control the AWS services and resources that users can access. IAM supports multifactor authentication (MFA) for privileged accounts, including options for hardware-based authenticators and support for integration and federation with corporate directories to reduce administrative overhead and improve end-user experience.

Monitoring and Logging

AWS CloudTrail is a service that records AWS API calls for your account and delivers log files to you. The recorded information in the log files includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. This provides deep visibility into API calls, including who, what, when, and from where calls were made. The AWS API call history produced by

CloudTrail enables security analysis, resource change tracking, and compliance auditing.

Network Security and Amazon Virtual Private Cloud

In each Amazon Virtual Private Cloud (VPC), you create one or more subnets. Each instance you launch in your VPC is connected to one subnet. Traditional layer 2 security attacks, including MAC spoofing and ARP spoofing, are blocked. You can configure network ACLs, which are stateless traffic filters that apply to all inbound or outbound traffic, from a subnet within your VPC.

These ACLs can contain ordered rules to allow or deny traffic based on IP protocol, by service port, and by source and destination IP address.

Security groups are a complete firewall solution that enable filtering on both ingress and egress traffic from an instance. Traffic can be restricted by any IP protocol, by service port, as well as source and destination IP address (individual IP address or classless inter-domain routing (CIDR) block).

Data Encryption

AWS offers you the ability to add a layer of security to your data at rest in the cloud, by providing scalable and efficient encryption features. Data encryption capabilities are available in AWS storage and database services, such as Amazon EBS, Amazon S3, Amazon Glacier, Amazon RDS for Oracle, Amazon RDS for SQL Server, and Amazon Redshift. Flexible key management options allow you to choose whether to have AWS manage the encryption keys using the AWS Key Management Service o (AWS KMS) or to maintain complete control over your keys. Dedicated, hardware-based cryptographic key storage options (AWS CloudHSM) are available to help you satisfy compliance requirements.

For more information, see the Introduction to AWS Security and AWS Security Best Practices whitepapers.

Oracle WebLogic on AWS Use Cases

Oracle WebLogic customers use AWS for a variety of use cases, including these environments:

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

Conclusion

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

Some of the benefits of deploying Oracle WebLogic 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 WebLogic 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.

Contributors

The following individuals and organizations contributed to this document:

Ashok Sundaram, Solutions Architect, Amazon Web Services

Document Revisions

Date	Description
December 2018	First publication