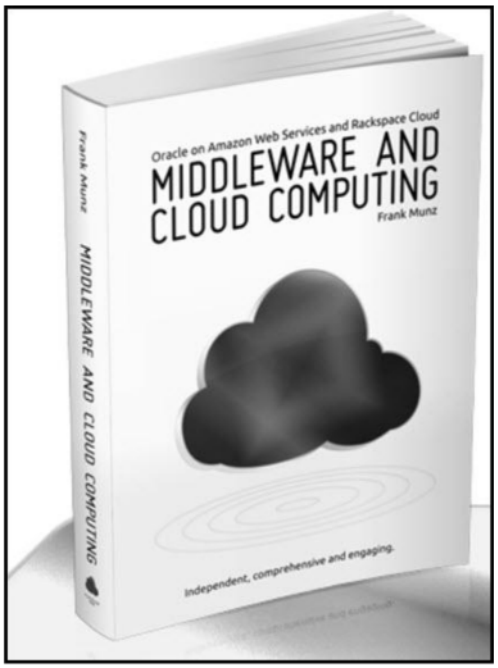
ORACLE MIDDLEWARE AND CLOUD COMPUTING

*An Introduction to **Cloud Computing and Oracle Middleware** on Amazon Web Services (AWS), Rackspace Cloud and the RightScale Cloud Management Platform.*



Reader Reviews:

"Great book: Hands-on tutorial, up-to-date and well researched for Oracle Middleware Architectures in the cloud"

"Clear, objective and brand new - I highly recommend it!"

"Excellent for practical use"

Contents and more information about the book: *http://cloudbook.munzandmore.com/*

Get it from Amazon.com: *http://www.amazon.com/dp/0980798000*

High-end Oracle Middleware/SOA/Cloud Courses and Consulting (worldwide): *http://www.munzandmore.com/courses*

6 Designing for the Cloud

This chapter explores how to use Oracle Fusion Middleware in the cloud. It is the base for the subsequent chapters about databases, scalability, availability and monitoring.

We will start with design principles for the computing cloud. Then follow with a detailed comparison of the main IaaS providers. After that we'll cover different strategies for building cloud images, and how to introduce a design blueprint for distributed WebLogic applications in the cloud.

6.1 Design Principles

You may wonder if a certain design is required or if you have to follow certain principles to build applications for the cloud. Often you read, no, just go ahead and build scalable applications as you would do in a classical, non-cloud environment.

This is only half of the truth. Of course any scalable and distributed application will perform well in the cloud. But then, if your application is based on complex middleware, such as WebLogic Server or Service Bus, you need to understand how to provision these environments in the cloud, and how to benefit from (or when to avoid) built-in middleware features that support availability and scalability.

Designing for Availability

Consider that you can suddenly lose an instance in the cloud. Sometimes this is called "plan for failure". Losing one part of your distributed application shouldn't stop your whole business. This is an important principle; to achieve this, decouple the building blocks of your application and use asynchronous communication.

I will cover the aspects that are vital for understanding availability in chapter 9.

If you lose an instance, you want to be informed about it, so you require a monitoring system. Monitoring is covered in chapter 11.

Designing for Scalability

Sometimes this is simply called "elasticity". I prefer the term scalability here, because it encompasses more; it includes topics such as load balancing, content distribution networks, and auto scaling.

Load balancing is offered as an AWS cloud service to distribute incoming requests. Amazon's Content Distribution Network, CloudFront, provides scalability for static content.

Rapid elasticity is a unique feature of the cloud, based on AWS auto scaling. To provide true elasticity, you have to design a system architecture where new instances can be started quickly and running instances are resilient to restarts.

All these aspects of scalability are covered in chapter 9.

6.2 IaaS Platform Choice

Searching the web reveals dozens of postings from people evaluating Amazon Web Service and Rackspace Cloud, and also includes some Amazon vs. Rackspace comparisons. I encourage you to do your own Google search to get an overview of these stories. I am trying to keep it neutral and stick to an analysis of the different features of the platforms. Few users have explored both platform options. Sometimes, the initial decision to go with one or the other IaaS platform is based on a particular killer feature (or its absence), but in general you should compare the entire offerings including all provided services and non-functional aspects -such as costs and SLAs.

I have summarized the differences in Table 3 and marked the better option in bold letters, wherever this was possible.

Comparing the performance of both platforms in an objective way would be an interesting and challenging task, but it's also walking on thin ice. The actual performance difference that you could observe for a particular application depends on many factors and cannot be quantified easily and in general terms. We analyze some of these factors in more detail later on in the chapter about scalability. In general, nobody will be able to easily predict the performance for an arbitrary application running on a cloud platform.

If you try to conduct such a comparison yourself, then don't forget that application performance per dollar spent is more important than raw system performance (measured by a more or less synthetic benchmark). So, first of all, you have to understand your application. Also, consider the non-functional requirements such as availability, which

again depend on whether you start a number of small and low performing instances or one big instance. All these factors are difficult to isolate since they depend on each other.

EC2 and Rackspace Cloud Server

Looking at the features of both platforms, there is an interesting difference for applications with a small footprint. RSC offers smaller instances where the smallest instance costs less than the smallest AWS instance. If you are on a budget or you are not running a full-blown WebLogic installation, with lots of heap assigned to the JVM, this can be the better choice because RSC instances guarantee a floor limit, but allow CPU bursting. Also, an increased number of small instances results in increased overall availability.

On the high end, AWS is unbeatable with 68.4 GB of RAM -that is four times the maximum RAM you could squeeze out of the biggest RSC instance. Although using that much heap for a single JVM is only possible if you are running a 64-bit JVM.

Whereas RSC claims it wants to earn your business every month, you can get cheaper pre-paid or spot instances from AWS.

Location could be the killer criteria: There is no RSC location in Europe, but more often than not, European companies cannot use IT infrastructure in the US. Often this is due to legal issues, since it is not possible to send data out of the country of origin or region of some countries. Sometimes it is because of the risk of espionage: the more important your data is, the more likely it is that some other person, company, or even country is trying hard to get a copy of it. Latencies are another reason. Brokers try to minimize the distance to the stock exchange, because for real-time trading, fractions of a millisecond can decide millions.

The worldwide availability of cached information on a content delivery network is provided by AWS and RSC as well. Remember, that CDNs only help with reducing the latencies for the delivery of static content. Concerning presence at different geographic locations, Amazon is the clear winner, managing to cover three different cultural regions.

AWS supports a number of other cloud services, e.g. for message passing, notifications, databases, load balancing and auto scaling. All these services are off-instance and don't require any installation.

Table 3: Comparison Cloud Instances and Machines

Cloud Instances and Machines		
	Amazon Web Services EC2	**Rackspace Cloud Server**
ID Verification	Text message based	A person is calling you
Invoice based billing	Yes	
One-on-one Support	400$	**Excellent, free and 24/7**
Default access	Certificate based	Root login
CPU	Upper limit	Floor limit (with bursting)
Minimum cost in US for on demand Linux server	0.02 $ / hour (0.613 GB)	**0.015$ / hour** (256 MB RAM)
GB / $ at min RAM in US	**30,6**	17,1
Max RAM	**68.4 GB (at 2,4$ / h)**	15.8 GB (at 0,96$ /h)
GB /$ at max RAM in US	**28,5**	16,5
Cheaper prepaid instances	**Yes**	No
Locations North America	**2 regions = 6 availability zones**	2 data centers
Locations Europe	**1 region = 2 availability zones**	Planned for 2010
Locations Asia	**1 region = 2 availability zones**	-
Choice of geographically distributed machines	**Yes**	No
Resizable instances	Yes (but not via GUI)	**Yes**
Max emails / day	Limit for new accounts (can be raised upon request)	5000/day 250 per 20 mins
Instance bandwidth limit	No	**Yes**
Additional IP addresses	No	**Yes** $2.00 per month per IP
Floating IPs for WebLogic	No	No
NAT used	Yes	No
Reverse DNS	No	**Yes**
Support for IP Multicast	No	No
External Cloud Firewall	Yes: Security Groups	No (iptables, Windows FW)
Cloud Loadbalancer	**Yes**	No

Table 4: Cloud Images

Cloud Images		
	AWS EC2	**RS Cloud Server**
Images with WebLogic pre-installed	**Yes, WLS 10.3.3** but only in US	No
Images with custom OS	**Yes**	No
Support for Linux	Yes	Yes
Support for Windows	Yes	Yes
Available server images	**Many**	Few
Sharable server images	**Yes**	No

At RSC you cannot create share your images or create your own image with your particular operating system installed. I recommend that you first check if a RSC machine image exists with the operating system that suits your needs. There are a huge number of different images for AWS but a rather limited list for RSC.

RSC has a clear advantage over AWS because RSC lets you configure reverse DNS, so your email sender contains the correct DNS name, which will pass SPAM filters more likely than the AWS DNS entry. RSC also publishes a clear statement about the number of emails that can be sent per day.

Instance Storage

The approach that both platforms take regarding instance storage is quite different. Local storage at RSC is a RAID 10 and therefore it's persistent and highly available.

AWS has ephemeral local storage but offers SAN-based EBS as persistent storage which can also be used to boot the instances. EBS continues to be available when an instance is stopped and can be mounted to another instance, but only to one at a time.

At RSC you can configure an instance, create a snapshot and stop the instance. Snapshots belonging to an instance will be removed when the instance itself is removed, but they can be persisted to Cloud Files.

Cloud Storage

At first glance, the cloud storage offering of both platforms is rather similar in concept. Both platforms offer a content distribution network. Both can store metadata with objects, but only Amazon's S3 provides versioning for objects.

Maybe the most important difference, and a major drawback for the RSC offering, is that you cannot share your RSC, whereas you can store and share your EC2 AMIs on S3.

There are a few other differences, such as the shorter length of bucket- or object names, which are not practically relevant in most cases.

Table 5: Cloud Storage

Cloud Storage		
	S3	**Cloud Files**
Grouping element	Bucket	Container
Grouping element length	63	256
Identifier length	1024	4096
Max size	5 GB	5 GB
Metadata	Yes	Yes
SLA availability	99.95 for each region	99.9
SLA durability	99.999999999% 99.99% with reduced redundancy	
Versioning	Yes	No
CDN	**CloudFront**	**Limelight**
CDN edges	8 in US 4 in Europe 3 Asia	**70 worldwide**
Server images on cloud storage	Yes	Yes

Conclusion

To conclude this summary, AWS is the more versatile and more powerful platform with more data centers at different locations.

RSC is a great choice if you prefer an easier approach combined with excellent and free support service, in contrast, the possibilities offered by AWS come with a certain level of complexity. At the moment, RSC is a compelling choice if a cloud platform located exclusively in the US, with a limited number of non-sharable machine images and a worldwide CDN, suits your needs.

On the other hand, AWS provides a fast growing, unparalleled set of cloud services such as auto scaling, elastic load balancing, monitoring, relational database server, as well as

queuing and notification services. We will examine these services in the remainder of the book.

Since AWS is the more complex and expansive platform, I will focus mainly on AWS for the remaining chapters. Whenever it eems interesting enough we will have a look at RSC, too.

6.3 AMI Design

When moving to the cloud you have to think about the right AMIs for your installation. The type of AMIs you are using, how you build them, and what to include with them are the core decisions. It is definitely worth thinking about a strategy that suits you best, since changing it later comes at a high cost.

It is important to understand the shared responsibility model of the AMIs. Amazon doesn't care what kind of operating system you are running in an AMI. They don't mind if you install security patches. AWS will run any of your AMIs. So you have to make sure that they are secure and up to date.

S3 vs. EBS AMIs

This is a good opportunity to revisit EBS- and S3-backed AMIs again. See Table 6 for an overview of the differences.

As already explained in the chapter about AWS, the EBS-based AMIs are newer and nowadays they are used more frequently. They support a root image of up to 1 TByte. Even if your instance crashed, you could remount the EBS volume and save the data.

When you use the AWS management console to start an EBS-backed instance, it will only show the EBS volume. Local storage is available for EBS-backed images, too. With the command-line it is possible to attach the local storage when the instance is started. However, you cannot assign local storage to EBS-backed instances when they are already running. On the other hand, S3-backed instances start with the local instance storage already attached. There is one exception though; micro instances don't support local storage at all.

Table 6: S3 vs. EBS AMIs

	S3-backed AMI	EBS-backed AMI
Available since	Start of AWS	December 2009
Park instance?	No. Instance is running or terminated.	Yes. Instance can be stopped.
Data persistence	Data on instance storage not persisted in case of failure or termination.	Data persisted on EBS when instance is stopped or in case of failure. EBS persistence after termination can be set.
Instance storage mounted per default	Yes	No
Max size of root device	10 GByte	1 TByte
Boot time	Few seconds up to one minute	Less than 5 minutes
Create new AMI	Requires AMI tools installation and a script of several commands to rebundle AMI.	One command
AWS management based creation of new AMI	No	Yes
Charges	Instance +S3 for AMI storage	Instance usage + EBS volume usage + EBS snapshot for AMI
Upgrading	Instance attributes fixed for the life of an instance	Instance type, kernel, RAM disk, and user data can be changed while the instance is stopped.
Paid AMIs	Yes	No

EBS-backed AMIs boot quickly - a really convenient advantage. In my case, the AWS management console shows a customized EBS-backed AWS Linux image as running only a few seconds after this instance was started.

Only EBS-backed instances allow you to change the instance type, kernel and RAM disk while the instance is stopped (see the example in section **3.4** titled "Resize an Instance").

I guess it is obvious that EBS-backed images have a lot of advantages, but is there any room left for S3-backed AMIs? When should you still consider them? Here are some examples when to consider S3-backed images.

- Instances which are stateless and don't need any EBS file system. E.g. a full-baked or scripted AMI containing an application server that uses a remote database.
- AWS DevPay is an online billing and account management service that enables S3-backed AMIs that the client has to pay for. DevPay isn't supported by EBS-backed images.

KIBIS AND GIBIS

Don't be surprised to encounter kibis and gibis in the AWS documentation. AWS laudably uses the new binary prefixes, such as GiByte instead of the old GByte or GB when indicating a power of 2 (according to the IEC 60027-2 Amendment 2 and ISO/IEC 80000 standards - in case you are interested).

The second syllable is derived from the word "binary" -shortened to "bi", so gibi is short for "gigabinary" or 2^{30} which is in the ballpark of 1.05×10^{6}.

For a detailed discussion, take a look at the Wikipedia site:

http://en.wikipedia.org/wiki/Binary_prefix

User Data and Startup Scripts

You can pass any file or startup data when starting a new instance. When using the EC2 command-line tools simply add a `--user-data-file` parameter as shown in the example below.

```
ec2-run-instances ami-ID -n 3 --user-data-file payload.zip
```

Using the AWS management console, the user data can be passed as well as shown in the screenshot in Figure 53.

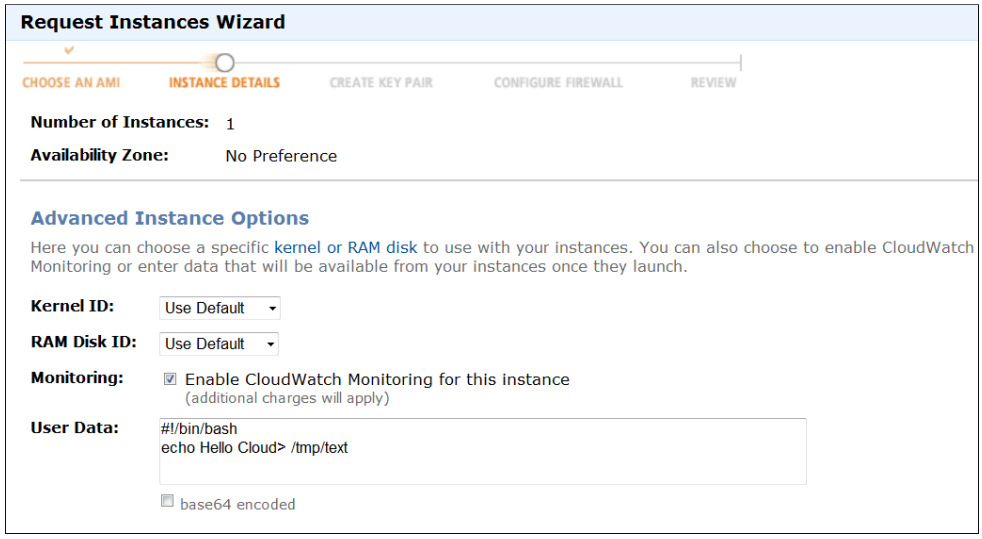


Figure 53: User Data in AWS Console

In general there is no automatic mechanism that retrieves the user data on the instance. However ,you can retrieve the data via a REST call using wget.

```
wget http://169.254.169.254/1.0/user-data -O /tmp/payload.zip
```

AMIs from Alestic and Ubuntu and the AWS Linux, implement useful additional functionality. The instance runs user-data starting with the two characters #! as the root user on the first boot.

Running a script at startup of an instance is the core mechanism to provide scripted configuration for AMIs.

runurl

Eric Hammond from Alestic published a little utility that extends the user-data mechanism even further. The runurl utility runs a remote script which is specified by a URL with the following syntax.

```
runurl URL [ARGS]...
```

The runurl utility can be installed with the following commands:

```
sudo wget -qO/usr/bin/runurl run.alestic.com/runurl
sudo chmod 755 /usr/bin/runurl
```

Centralized scripting

The advantage of this approach is that you can run any configuration script from a centralized script repository, which could be stored on any web server or on S3 as shown in Figure 54. There is no need to pass the whole script to the starting instance, just the sequence of commands for the configuration.

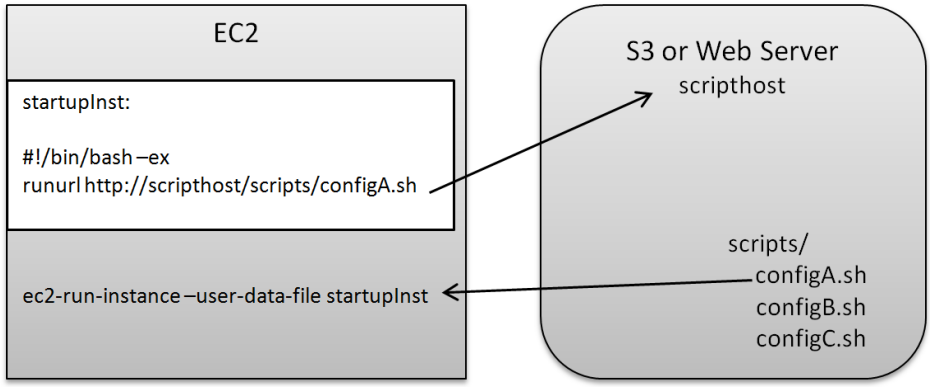


Figure 54: runurl Script for Startup Automation

Auto scaling

Passing only the name of the script but not the script itself, also facilitates the auto scaling configuration where the startup parameters are specified in a launch configuration. When passing the name of the script only, the script itself can be changed anytime without changing the launch configuration.

Configuration Model

There are two different ways to set up your AMIs. When building your first AMI you will most likely configure everything, then create a new image and happily use it.

But then, you will have to create a new AMI for every change. Be it a newer version of the Apache web server, or a new UNIX password (since you realized that the first name of your spouse isn't as safe as you thought it was, when you created the image at 2.30 am). Those fully configured AMIs are sometimes called full-baked AMIs.

The other approach is to start with a basic template containing the OS only. Based on this template you can use a script to install and configure all necessary packages.

Let's say, you start with an AWS Linux AMI, then run two different sets of scripts. One would be to set up web server AMIs, install Apache, create the users, and get the web content. And the other script would be the one that installs WebLogic, creates a domain, configures resources such as JMS queues, JDBC data sources, and finally deploys your application.

To explore things at home I have a set of full-baked AMIs that I wouldn't want to lose. For larger systems I definitely recommend a scripted approach since the maintenance overhead is lower, which in the end saves you a lot of time, and therefore reduces costs.

Table 7: Full-Baked vs. Scripted AMIs

	Full-baked AMI	Scripted AMI
Initial setup time	Low	High
Maintenance effort	High	Low
Time to start new image	Low	High
Reusability	Low	High

There are tools available that follow the scripted AMI path such as RightScale. We will examine this in more detail in the chapter 8 about cloud management.

Provisioning of Oracle Software

You most probably get your Oracle software by using a web browser and downloading it from the download section of the Oracle web site.

For a scripted setup of an image, this is rather impractical because you won't have a web browser on a server image and you cannot script the download.

I found that you can obtain the software with a simple `wget` as well. E.g. to download WebLogic 11g with Oracle Enterprise Pack for Eclipse, you run the following command:

```
wget --user=your_OTN_username --password=your_OTN_password
http://download.oracle.com/otn/linux/middleware/11g/
wls1033_oepe111150_linux32.bin
```

Use your web browser to interactively explore the download URLs of other products. Because of the high-speed internet connection of the EC2 instances, this approach is usually much faster than downloading the product installer from Oracle to a local repository and then uploading it again to a particular EC2 instance.

Security

After you create a template image, it will be started hundreds of times, so you'd better enforce tight security right from the start of the design process.

Consider the following items as a checklist. If you feel uncomfortable implementing them yourself consider the professional service of a specialist.

- Harden the operating system. It is best to start with the smallest possible installation, e.g. Oracle JeOS or AWS Linux, and then add only the essential packages for your installation.
- Make sure there are no unnecessary processes running.
- Use restrictive security groups. Don't open port ranges for future usage when a single port is sufficient.
- Consider encrypting your file systems. Many operating systems support encryption.
- Consider installing a virus scanner for a Windows AMI. Even if it is not running permanently for performance reasons, it can still run full system scans periodically.
- Consider installing a network intrusion detection system (NIDS) to detect unusual network patterns such as port scans or denial of service attacks. Examples for NIDS are:
    - Bro (see `http://www.bro-ids.org`)
    - Snort (see `http://www.snort.org`).
- Install a host intrusion detection (HIDS) tool for monitoring and alerting on specific file change. Examples for HIDS are:
    - TripWire (see `http://www.tripwire.com`)
    - OSSEC (see `http://www.ossec.net`).

Cleaning Up before Creating an Image

After the customization, but before the creation of a new AMI, make sure that you don't leave anything behind. During the customization of an image it is good practice not to use features of the image that won't be included in the customized image.

- Clean up log files. This means all kinds of logs. Installation logs, logs remaining from when you started WebLogic to see if it was working, syslogs from the operating system, etc.
- Delete the shell history since it will contain IP addresses and possibly even user names and passwords.
- It is best not to have a UNIX desktop such as Gnome or KDE installed, but you should at least consider removing it, if you installed one. Your image configuration should be completed by now, but even if you need the desktop again, it's only a question of minutes to reinstall it.
- Delete all browser histories in case you used the browser. Don't build images with browser links that show where you buy your stocks, go for dinner, make new friends, or which terms you have searched at Google.

6.4 Architecture Blueprint

I am well aware that you are expecting a blueprint for what to build, how Oracle Fusion Middleware looks in the cloud, and a diagram that shows all the bits and pieces of a distributed WebLogic domain in the cloud.

Distributed WLS Application

There is no single, general blueprint fitting every possible design, but let me start with a blueprint for an application built on a distributed WebLogic domain in the cloud. When building such an application, you can take advantage of the AWS cloud services; these will be introduced in the following chapters.

All these services are off-instance, they don't require any installation or maintenance, and they are inherently scalable and highly available. You can just use them without worrying about the technical details of their implementation (Please let me remind you, however, that you have to pay for them, but we will come to that later).

When using all these services for a WebLogic domain and managed servers you get around installing a load balancer, web servers and a database, as well as dealing with the WebLogic configurations for JMS and JTA availability and clustering in the cloud.

Dive cruises and AWS services

Try to visualize the blueprint in Figure 55. It shows a fictional booking system for dive cruises, based on a distributed WebLogic domain running in Amazon's Europe cloud.

The clients are located in various European countries and access the entry page of the application by its DNS name. The entry page is delivered from the nearest cache of the CloudFront content distribution network. In the entry page, the URLs for dynamic content point to the Elastic Load Balancer that is distributing the requests to the WebLogic instances across both of these two availability zones (eu-west-1a and eu-west-1b).

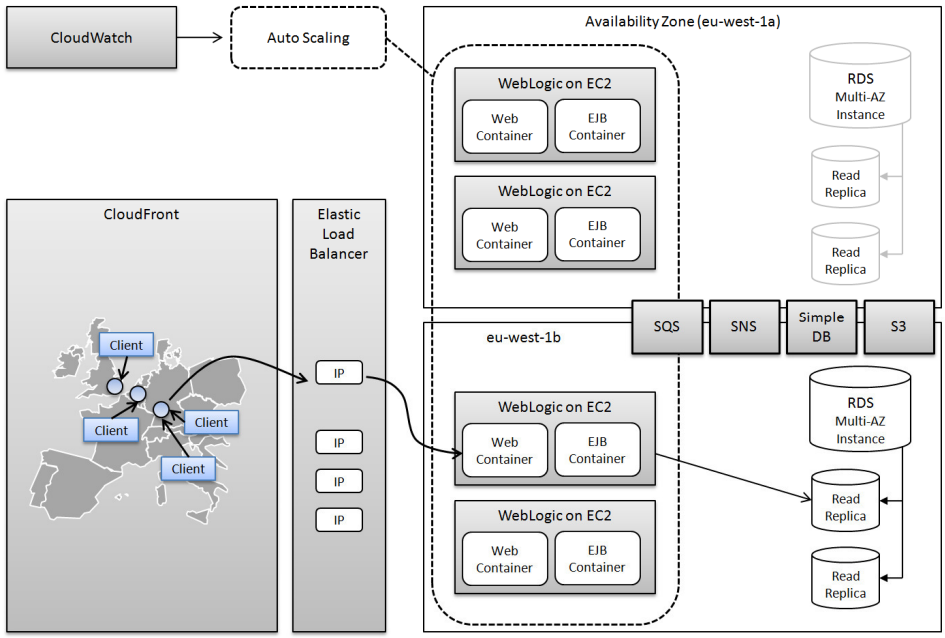


Figure 55: WLS Domain in the Cloud Blueprint

The WebLogic application uses the AWS relational database service to store the bookings, and a set of read replicas to speed up the retrieval read only data, when browsing destination catalogues of Australia, Fiji, the Philippines, and South Africa.

The booking system is decoupled from the other modules, such as billing, catalogue, and the customer loyalty program. Every module uses the simple queue service to asynchronously communicate with other modules based on persisted messages. Once a booking is completed, an email notification is sent via the simple notification service, and the booking itself is stored as a PDF file in S3.

The EC2 instances running the WebLogic domain are configured as an auto scaling group. The launch configuration of the auto scaling group specifies the setup scripts, which are executed when a new instance is created. These scripts are centrally administered and hosted off-instance in an S3 bucket. Application specific configuration data is stored in SimpleDB, and retrieved by the booking application during startup.

An auto scaling trigger defines when the number of instances is increased or decreased depending on CloudWatch monitoring metrics. CloudWatch also graphically displays the metrics of EC2 instances, the EBS volumes, and the load balancer.

I will cover every one of these services, alternatives, design-tradeoffs, and their costs in the following chapters. Use this blueprint as a reference if you build an application for the cloud.

Middleware Features and AWS Services

In addition to all the AWS services represented by the boxes in the blueprint above, you have to understand the following design trade-offs where similar functionality is provided by an AWS service and a WebLogic feature:

- Simple Notification Service vs. WebLogic JMS topics or WebLogic Diagnostic Framework Notifications
- Simple queue service vs. WebLogic JMS queues
- AWS Elastic Load Balancing vs. WebLogic web server plugin
- AWS auto scaling vs. WebLogic clustering

Middleware in the Cloud

Operating middleware products in the cloud is more challenging if the product itself requires WebLogic clustering or distributed JMS queues. Middleware products like Oracle Service Bus or SOA Suite depend on such features (for details see section 5.4).

However, an AWS service (e.g. simple queue service) cannot replace middleware features (e.g. WebLogic JMS queues) and you need to understand the technical details of highly available JMS destinations and a WebLogic cluster in the cloud. This is why the peculiarities of middleware products in the cloud are also covered in the following chapters.

- Buzzword-free, comprehensive and vendor-neutral.
- Clearly written by an independent IT professional, who has delivered successful projects to some of the world's leading companies.
- Learn why running Oracle WebLogic Server and Fusion Middleware in the cloud is often easier, sometimes cheaper and typically more reliable than in your own data center.
- Take advantage of the industry's best: Amazon, Rackspace, RightScale and Oracle.
- Understand what it takes to achieve availability, scalability, monitoring and management of middleware in the cloud.
- Get started straightaway with a free Micro Instance from Amazon.
- Discover key features and showstoppers.
- Develop your expertise by reading this book.

MIDDLEWARE AND CLOUD COMPUTING

Dr. Frank Munz is an expert in middleware and distributed computing. He has more than 15 years experience working with top middleware vendors such as Sun, BEA, TIBCO and Oracle, throughout Europe and Australia. Frank is the founder of munz & more - a cutting-edge consultancy focusing on Oracle middleware and cloud computing - and also runs his own high-end training program. He loves to talk about features and showstoppers and frequently speaks at conferences all over the world.

Oracle WebLogic Server and **Fusion Middleware** in the **cloud.**
Amazon Web Services (AWS), **Rackspace Cloud** and **RightScale Cloud** Management.
Outlook on **Oracle VM** in the **cloud.**

This book contains all the new and cool AWS stuff: Micro Instances, Tags, S3 Reduced Redundancy Storage, EBS-backed AMIs, MySQL RDS Read Replicas and Multi-Availability Zone Instances, Simple Notification Service (SNS), Simple Queue Service (SQS), Auto Scaling, Elastic Load Balancing with SSL Termination, the brand new Amazon Linux and much, much more!

munz & more publishing

ISBN 978-0-9807980-0-5
90000
9 780980 798005