



1 JULY 2016 / MIDDLEWARE

Configuring Amazon RDS as the Oracle SOA Suite Database

TAGS: MIDDLEWARE | DEVOPS | AWS | INTEGRATION

We are proud to announce that RDS support is fully integrated and certified against MyST 3.8.2, which was released on 15/Jun/2016. To know more about MyST visit: <http://myst.rubiconred.com>

1. Overview

We started to provision Oracle Fusion Middleware platforms against AWS in anger about three years ago. From the beginning we took advantage of the ability to create the AWS infrastructure within minutes. We could also use MyST to provision complex Oracle Fusion Middleware EDG (Enterprise Deployment Guide) Compliant platforms in less than an hour.

One challenge that we faced, was that we couldn't use Oracle RDS as the database for our Oracle Fusion Middleware installations. This was primarily because the RDS master user didn't have the database privileges required to run the Oracle Repository Creation Utility (RCU). As a result, we implemented our own automation for provisioning the Oracle Database running on EC2 instances.

Whilst this works for running Dev and Test workloads in AWS,

when it comes to implementing Production workloads, Oracle RDS provides a number of additional benefits. This includes simplified administration tasks, including backups, software patching, monitoring, and hardware scaling.

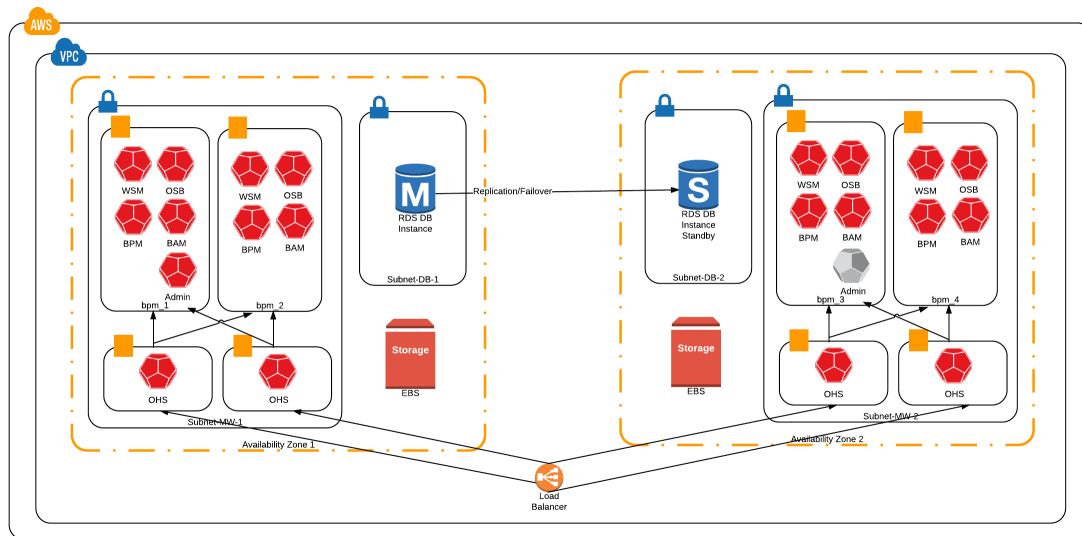
In addition, the Multi-AZ deployment option simplifies the implementation of a highly available architecture, as it contains built-in support for automated fail-over from your primary database to a synchronously replicated secondary database in an alternative Availability Zone in case of a failure.

This all started to change late last year, with a number of our customers looking at running Oracle SOA workloads in Production on AWS. Being an AWS Technology Partner, we provided this feedback to AWS, who in return invited us to collaborate with the RDS team.

We spent the last 4 months of this year working with the AWS Oracle RDS Team (a big thank you to Michael and Jinyoung) to test the RCU capability within MyST. This went extremely well, and the RDS team worked closely with us to support the go-live of our first customer on Oracle SOA 12.2.1 on AWS using RDS – what we believe to be a world first!

Our very first customer go-live on RDS was a few weeks ago in June-2016. More recently, Amazon has now announced that RCU is officially supported by Oracle RDS. This is great news for us and our customers. We can now provision an Oracle Fusion Middleware EDG HA compliant environment within minutes and take advantage of RDS to simplify on-going operations.

The following diagram depicts a typical highly available Oracle Fusion Middleware deployment in AWS. Note that the number of compute nodes, as well as the components may vary depending on the requirements.



2. Oracle RDS Compatibility

Before getting started with Oracle RDS, it's important to check its compatibility with RCU and the corresponding Oracle Fusion Middleware components. The following AWS Oracle RDS edition/version options support RCU:

- Oracle Database (EE / SE Two) 12.1.0.2.v4+
- Oracle Database (SE / SE One) 12.1.0.1.v5+
- Oracle Database (EE / SE / SE One) 11.2.0.4.v8+

Oracle Fusion Middleware Components

The following is the compatibility Matrix we have validated between Oracle Fusion Middleware and Oracle RDS. Most of them are already being used by our customers in Production.

Note: RDS uses Oracle Managed Files, therefore the RCU honorOMF flag must be used.

OFMW VERSION	OFMW COMPONENTS	RCU CC
Integration 12.2.1	• ADF	• IA
	• B2B	• IA
	• BAM	• IA
	• BPM	• M
	• OWSM	• O
	• OSB	• SC
	• SOA	• UC • ST • W
Integration 12.2.1	• MFT	• ES
	• ESS	• IA • IA • IA • M • O • UC • ST • W
Social Business and Collaboration 12.2.1	• ADF	• DI
	• OWSM	• IA
	• Webcenter Portal	• IA • IA • IA • M • O • PC • UC • ST • W

3. Before You Start

The following is a checklist of items to verify before you start creating your RDS Instance.

AWS Region

You must decide which of the available AWS Regions you will be using for your workload. Choosing a region can be influenced (but not restricted) by the following factors:

- Latency between the end users and the AWS region
- Latency between your datacenter and the AWS region. This is one of the most critical factors in case you have Oracle Fusion Middleware running on-cloud and backends running on-premises
- AWS cost: The AWS service cost varies depending on the region
- Legislation and Compliance: The data from your customers might have restrictions on which country the data must be stored

You can find more details about the available regions at:

<https://aws.amazon.com/about-aws/global-infrastructure/>.

VPC

Next, you need to define your VPC, subnets and Oracle Fusion Middleware EC2 instance security groups.

Oracle Fusion Middleware version

Thirdly, you need to check the Fusion Middleware version is supported along three aspects:

- You are using a supported RCU version by RDS. At this stage RCU 12.2.1 works out of the box, due to support of honorOMF flag. There are some options on how to support RCU 12.1.3, but that is out of scope for this post.
- The RCU version must match your Oracle Fusion Middleware version.
- You must also make sure that the RCU components are supported by RDS.

The Oracle Database is compliant with OFMW

You must make sure that the Oracle database edition and version are supported by the Oracle Fusion Middleware version you are installing.

The supported configurations can be found here:

<http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html>

Oracle Database License

You can select either "Bring your own license" or "License Included". Only "Standard Edition One" supports "License Included". Make sure you have a clear understanding of which licenses your organisation has, before creating the database instances.

AWS Permissions

You must have enough grants in your AWS account to perform the actions of this post.

You will need permissions to the following services under your AWS account:

- VPC
- EC2
- RDS
- Route53

Oracle SOA EDG Database Requirements

The Oracle EDG mentions requirements about the number of processes to be set in the database, as follows:

COMPONENT	PROCESSES
SOA	300+
BAM	100+
SOA and BAM	400+
SOA and OSB	800+

The above numbers are quite generic. You must also take into consideration:

- The number of connections configured in the Weblogic JDBC Connection Pool
- The number of Weblogic managed services
- If JMS/TLOGs database persistence is enabled, Weblogic will require to establish more connections to the database

Considerations about the number of DB processes

For an Oracle Database, the parameter **process** is calculated dynamically by default. You will find the following expression in the RDS DB Parameter group:

```
GREATEST({DBInstanceClassMemory/9868951}, 80)
```

In the table below, you can see some examples of how many maximum number of processes you will have by default, depending on the Database instance class:

DBINSTANCECLASS	RAM	PROCESSES
db.m1.small	1.7GB	159
db.m3.medium	3.75GB	386
db.m3.large	7.5GB	794
db.m2.xlarge	17.1GB	1838

You can override the number of processes with an absolute value, but be aware that your DB instance could run out of memory. You can monitor both memory consumption and number of processes from the RDS console.

AWS CLI

If you use the AWS CLI, make sure that you have the CLI installed, and you have either an access/secret key, or an EC2 instance with a role associated to it (recommended).

4. Configuring a Highly Available Oracle RDS instance

We recommend that you script any AWS configuration, by either

using CloudFormation templates, or by using the AWS APIs (either CLI, Python, Java, etc). This will enable you to automate the configuration, and you could use the same set of scripts for multiple projects.

For the purpose of this blog post, we will show how to configure via both: AWS console, and the AWS CLI. We recommend that you go through the AWS Console part even if you use the CLI, as you will get a better context. The AWS CLI commands in the post map directly to the tasks executed using the AWS Console.

4.1 Define the security groups for the database

A security group allows us to control access to specific inbound/outbound ports, as well as restrict access based on source address.

Source addresses can be represented either as CIDR (for example, 192.168.1.1/32 represents a single IP address, whereas 192.168.1.0/24 represents 254 IP addresses under the 192.168.1.0 network) or another security group.

We will use another security group to define access to source addresses, as this provides more flexibility and is easier to manage. In that way, we can simply give network access from OFMW-TEST security group to the TEST database, rather than having to specify each of the source IP addresses.

Using the security group is one of the ways to make sure that only the required ports are exposed to specific hosts. That helps to prevent, for example, a PROD Oracle SOA Suite instance trying to access a TEST SOA Suite instance Database.

We recommend creating the following security groups

SECURITY GROUP NAME	INBOUND RULES	SOURCE
TEST-RDS-OEM-SG	TCP 5500	VPN Gateway Security Group (sg-***
TEST-RDS-SG	TCP 1521	OFMW-TEST Security Group (sg-****
		Gateway Security Group (sg-*****69

Create the TEST Database OEM Security Group

Using the AWS console

Perform the following steps:

- Select Services -> VPC -> Security Groups -> Create Security Group .
- Specify the Name Tag, Group Name and Description. For our example we will define the same value.
- Select the VPC to which the database will be deployed.

Create Security Group

Name tag: TEST-RDS-OEM-SG

Group name: TEST-RDS-OEM-SG

Description: TEST-RDS-OEM-SG

VPC: vpc-af (10.4.0.0/16) | DEVTEST-VPC

Cancel Yes, Create

- Under Inbound Rules , set the TCP port to 5500, and define

in the source either the security group which you want to give access to, or the CIDR.

sg-e2c97e84 | TEST-RDS-OEM-SG

Summary **Inbound Rules** Outbound Rules Tags

Cancel Save

Type	Protocol	Port Range	Source	Remove
Custom TCP Rule	TCP (6)	5500	sg-69	

Add another rule

Using the AWS CLI

The following three commands execute respectively:

- Creates the DB OEM Security Group and assign the Security Group ID from the response to the OS environment variable called SGOEM (this is linux specific)
- Adds an ingress rule, which authorizes access to TCP port 5500 coming from the VPN Security Group
- Tags the Security group setting its name

```
SGOEM=$(aws ec2 create-security-group --group-name TEST-RDS-OEM-SG --d
aws ec2 authorize-security-group-ingress --group-id $SGOEM --protocol
aws ec2 create-tags --resources $SGOEM --tags "Key=Name,Value=TEST-RDS
```

Create the TEST RDS Data access Security Group

Using the AWS Console

- Repeat the previous steps, this time for the Security Group which will be used to give access to SQL clients, as follows:

Create Security Group [X]

Name tag: TEST-RDS-SG [i]

Group name: TEST-RDS-SG [i]

Description: TEST SG for DB client traffic [i]

VPC: vpc-af (10.4.0.0/16) | DEVTEST-VPC [i]

[Cancel] [Yes, Create]

- Add the rules to TCP Port 1521, granting access to the TEST Oracle Fusion Middleware Security group as a minimum. Optionally, you can give access to other security groups, in case you want operators/developers to be able to connect to the database using a SQL Client.

Summary Inbound Rules Outbound Rules Tags

[Cancel] [Save]

Type	Protocol	Port Range	Source	Remove
Oracle (1521)	TCP (6)	1521	sg-b8	[i] [X]
Oracle (1521)	TCP (6)	1521	sg-69	[i] [X]

[Add another rule]

Using the AWS CLI

Similarly to the previous step:

```
SGDB=$(aws ec2 create-security-group --group-name TEST-RDS-SG --descri  
  
aws ec2 authorize-security-group-ingress --group-id $SGDB --protocol t  
  
aws ec2 authorize-security-group-ingress --group-id $SGDB --protocol t  
  
aws ec2 create-tags --resources $SGDB --tags "Key=Name,Value=TEST-RDS-
```

4.2 Database Subnet Group

Before you can create an RDS instance, you must define a Subnet Group. A subnet group maps to a collection of subnets from your VPC. For RDS Oracle database, you must select 2 subnets in different availability zones.

Create using the AWS Console

To create a new DB Subnet Group, access from the AWS console:

Services -> RDS -> Subnet Groups -> Create DB Subnet Group

You must provide the following details:

- **Name/Description**
- **VPC ID:** The VPC which the database will be deployed to
- **Subnets:** Two subnets in different availability zones. Make sure that you select the correct subnets, in case you have multiple subnets per availability zone.

RDS Dashboard

- Instances
- Clusters
- Reserved Purchases
- Snapshots
- Security Groups
- Parameter Groups
- Option Groups
- Subnet Groups** ¹
- Events
- Event Subscriptions
- Notifications

Create DB Subnet Group

To create a new Subnet Group give it a name, description, and select an existing VPC below. Once you select an existing VPC, you will be able to add subnets related to that VPC.

Name DEVTEST-DB-SubnetGroup ²

Description Subnet Group for DEV/TEST ²

VPC ID DEVTEST-VPC (vpc-af) ²

Add Subnet(s) to this Subnet Group. You may add subnets one at a time below or [add all the subnets](#) related to this VPC. You may make additions/edits after this group is created. A minimum of 2 subnets is required.

Availability Zone	Subnet ID	CIDR Block	Action
us-west-2b	subnet-264ab642 (10.4.11.0/24)		Add
us-west-2b	subnet-42	10.4.11.0/24	Remove
us-west-2a	subnet-8e	10.4.1.0/24	Remove

[Cancel](#) [Create](#)

Create using the AWS CLI

You can create the DB Subnet Group using the AWS CLI as follows (Note that the Subnet ID values are masked):

```
aws rds create-db-subnet-group --db-subnet-group-name devtest-db-subne
```

4.3 Option Group

An Option Group persists additional feature definitions that you might want to add to your Oracle database. You can for example, set the database Timezone, enable Oracle Enterprise Manager feature for your database, or enable STATSPACK for more detailed monitoring.

RDS provides out of the box default option groups, but you cannot change them. For this reason, we will create a new Option group and add a few features (options). You can assign one Option group to multiple RDS instances.

Creating an Option Group

Using the AWS Console

To create a new Option Group, from the AWS console select Services -> RDS -> Option Groups -> Create Option Group , and define:

- **Name/Description**
- **Engine:** You must select which database edition you will use in your RDS instances
- **Major Engine version:** You must select the database version you will use

The screenshot shows the AWS Management Console interface for creating an Option Group. On the left, the 'Option Groups' menu item is highlighted with a red box and a circled '1'. The main panel displays the 'Create Option Group' form. The form contains the following fields:

- Name:** fusioncloud-test-optiongroup-oracle-se1
- Description:** Test OG for TEST Oracle SE1
- Engine:** oracle-se1
- Major Engine Version:** 12.1

The 'Description' field is highlighted with a red box and a circled '2'. At the bottom right of the form are 'Cancel' and 'Create' buttons.

Using the AWS CLI

You can create the DB Subnet Group using the AWS CLI as follows

```
aws rds create-option-group --option-group-name fusioncloud-test-optio
```

Adding options to an Option Group

Once the option group is created, you can add options to it.

We will add the following options to the our Option Group:

OPTION	ADDITIONAL CONFIG
Timezone	Time Zone: This is the timezone which the database associated
OEM	Port: The TCP port which the Database OEM will listen to Security Group: This is the security group we created previously
STATSPACK	Notes: As Oracle Standard Edition One does not support AWR r

The selection **Apply Immediately** is relevant when you are adding options to an option group which has RDS Instances running, so you have the choice to apply options straight away. This is not relevant at this stage, as we haven't created the RDS instances yet.

Adding a new Option

The screenshot displays the AWS RDS console interface. In the top navigation bar, the 'Add Option' button is highlighted with a red box and a circled '2'. In the left-hand navigation menu, the 'Option Groups' link is highlighted with a red box. The main content area shows a list of option groups. The second option group, 'fusioncloud-test-optiongroup-oracle-se1', is selected and highlighted with a red box and a circled '1'. Below the list, the 'Option Group Properties' section is visible, showing details for the selected option group:

- Option Group Name:** fusioncloud-test-optiongroup-oracle-se1
- Option Group Description:** Test OG for TEST Oracle SE1
- Engine Name:** oracle-se1
- Major Engine Version:** 12.1

At the bottom of the console, there are two sections: 'Options' and 'Associated Instances and Snapshots'. The 'Options' section shows a table with columns: Name, Persistent, Permanent, Port, Security Groups, and Settings. It currently displays 'No options added'. The 'Associated Instances and Snapshots' section shows a table with columns: Resource and Type, currently displaying 'None'.

Adding the Timezone option using the AWS Console

Adding the Timezone option using the AWS CLI

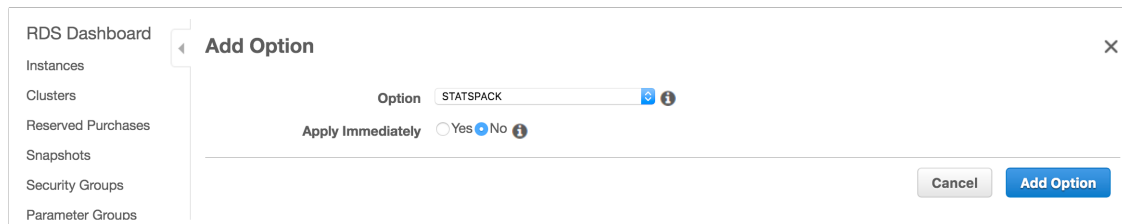
```
aws rds add-option-to-option-group --option-group-name fusioncloud-tes
```

Adding OEM Option using the AWS Console

Adding OEM Option using the AWS CLI

```
aws rds add-option-to-option-group --option-group-name fusioncloud-tes
```

Adding STATSPACK Option using the AWS Console



Adding STATSPACK Option using the AWS CLI

```
aws rds add-option-to-option-group --option-group-name fusioncloud-tes
```

4.4 Parameter Group

A parameter group is a container of parameters which are applied to your database engine.

A default parameter group is created out of the box when you launch an RDS instance. However, you cannot update the default parameter group. For this reason, we will create a new Parameter Group for the TEST environment.

We recommend that you create a new parameter group, even if you don't need to customize any parameters at this point. It is highly likely that at some stage you will need to customize some of the parameters.

We also recommend that you plan how you will be reusing the parameter groups among multiple RDS instances. It will depend on how the DB parameters could differ among the different database instances.

Create a Parameter Group using the AWS Console

To create a new Parameter Group, from the AWS console select
Services -> RDS -> Parameter Groups -> Create Parameter Group ,
and define:

- **Name/Description**
- **Engine:** You must select which database edition you will use in your RDS instances

RDS Dashboard
Instances
Clusters
Reserved Purchases
Snapshots
Security Groups
Parameter Groups 1
Option Groups
Subnet Groups

Create Parameter Group X

To create a Parameter Group, select a Parameter Group Family, then name and describe your Parameter Group.

Parameter Group Family oracle-se1-12.1
Group Name fusioncloud-test-parametergroup-oracle- 2
Description fusioncloud-test-parametergroup-oracle- 2

Cancel Create

Create a Parameter Group using the AWS CLI

```
aws rds create-db-parameter-group --db-parameter-group-name fusioncloud
```

Changing a parameter

Note that some of the parameters are calculated dynamically. For example, the parameter **processes** has the out of the box expression as: `GREATEST({DBInstanceClassMemory/9868951}, 80)`

For a db.m3.medium instance type, we should get 386 processes. If we want to increase the number of processes, we could either change the DB Class, or set an absolute value. For the purpose of

this post, we will set the number of processes to 400 in the Parameter Group we created.

Changing the parameter processes using the AWS Console

Once you select the Parameter Group you have created, and clicked on "Edit Parameters", you will be directed to the following screen. Update the number of processes to 400, and save the changes.

The screenshot shows the AWS Management Console interface for editing parameters of a Parameter Group named 'fusioncloud-test-parametergroup-oracle-se1'. The 'Parameters' tab is selected. A filter 'processes' is applied. The 'processes' parameter is highlighted with a red box and a circled '1'. The 'Save Changes' button is highlighted with a red box and a circled '2'.

Name	Edit Values	Allowed Values	Is Modifiable	Source	Apply Type	D
aq_tm_processes	<input type="text"/>	0-40	true	engine-default	dynamic	in
background_core_dump	<engine-default>		true	engine-default	static	st
db_writer_processes	<input type="text"/>	1-100	true	engine-default	static	in
gcs_server_processes	<input type="text"/>		false	engine-default	static	in
job_queue_processes	50	0-1000	true	system	dynamic	in
log_archive_max_processes	<input type="text"/>	1-30	true	engine-default	dynamic	in
processes	400	80-20000	true	system	static	in
recovery_parallelism	<input type="text"/>		false	engine-default	static	in
shadow_core_dump	<engine-default>		true	engine-default	static	st

Changing the parameter processes using the AWS CLI

Note that because the parameter **processes** is a static parameter, we need to specify the method as "pending-reboot".

```
aws rds modify-db-parameter-group --db-parameter-group-name fusionclo
```

4.5 Creating the Database Instance using the AWS Console

In our example, we will create an "Oracle SE One" database, and demonstrate how simple it is to create a highly available database across two different availability Zones.

Note that depending on your SLA requirements, you might want to run a database in a single Availability zone. Running the database in Multi-AZ increases the costs.

The screenshot shows the 'Step 1: Select Engine' screen in the AWS Management Console. On the left, there is a vertical list of database engine logos: Amazon Aurora, MySQL, MariaDB, PostgreSQL, ORACLE, and Microsoft SQL Server. The 'ORACLE' logo is highlighted with a blue bar. To the right of the logos, the 'Select Engine' section displays a list of Oracle database options. Each option includes the engine name, the edition, and a brief description. A red rectangular box highlights the 'Select' button for 'Oracle SE One'.

Engine	Edition	Description	Select Button
Oracle EE	Oracle Database Enterprise Edition	Oracle Database Enterprise Edition is an efficient, reliable, and secure database management system that delivers comprehensive high-end capabilities for mission-critical applications and demanding database workloads.	Select
Oracle SE	Oracle Database Standard Edition	Oracle Database Standard Edition is an affordable and full-featured database management system supporting up to 32 vCPUs.	Select
Oracle SE One	Oracle Database Standard Edition One	Oracle Database Standard Edition One is an affordable and full-featured database management system supporting up to 16 vCPUs.	Select (highlighted)
Oracle SE Two	Oracle Database Standard Edition Two		Select

Selecting the availability

We will select the Multi-AZ deployment option. As a result, a secondary DB instance will be created in a separate location, and will be in standby. Provisioned IOPS will also be enabled by default.

Step 1: [Select Engine](#)

Step 2: Production?

Step 3: [Specify DB Details](#)

Step 4: [Configure Advanced Settings](#)

Do you plan to use this database for production purposes?

Production

☒ Oracle SE One

Use [Multi-AZ Deployment](#) and [Provisioned IOPS Storage](#) as defaults for high availability and fast, consistent performance.

Dev/Test

☐ Oracle SE One

This instance is intended for use outside of production or under the [RDS Free Usage Tier](#).

Billing is based on [RDS pricing](#).

[Cancel](#) [Previous](#) [Next Step](#)

Specifying the DB Details

- **License model:** For Oracle SE One, you can select between "Bring your own license" and "License Included".
- **DB Engine Version:** We will select 12.1.0.1.v5, since this version onwards supports RCU 12.2.1.
- **DB Instance Class:** We will select a relatively small DB Instance Class, as the load will be just for test purposes for one Oracle Fusion Middleware platform. You can change the DB Instance Class at any point. That will require the Database instance to be restarted.
- **Multi-AZ Deployment:** We will select "Yes", so that we will have a second standby instance running on a second Availability Zone.
- **Storage Type:** Because we have selected Multi-AZ, by default the Storage Type will be set to "Provisioned IOPS", which provides a more consistent throughput but has a higher cost associated. We will change it to "General Purpose (SSD)", as we are not too interested in consistent performance for the dev/test environment. We suggest you run performance tests, to make sure the selected option meets your requirements.
- **Allocated Storage:** We will allocate 100GB. Note that for General Purpose SSD, the number of IOPS is proportional

to volume size (it's a 3:1 ratio).

- **DB Instance Identifier:** This identifier will be used to define the database hostname.
- **Master Username:** This is the equivalent of a sys user, but with less privileges, since RDS does not allow you to use either sys user or sysdba role.
- **Master Password:** The password for your master user.

Step 1: [Select Engine](#)
 Step 2: [Production?](#)
Step 3: Specify DB Details
 Step 4: [Configure Advanced Settings](#)

The following selections disqualify the instance from being eligible for the free tier:

- Multi-AZ Deployment
- Allocated Storage > 20GB
- DB Instance Class

You will be charged normal RDS Prices. [Learn More](#).

Estimate your monthly costs for the DB Instance using the [RDS Instance Cost Calculator](#).

Specify DB Details

Instance Specifications

DB Engine	oracle-se1
License Model	bring-your-own-license
DB Engine Version	12.1.0.1.v5
DB Instance Class	db.m3.medium — 1 vCPU, 3.75 GiB RAM
Multi-AZ Deployment	Yes
Storage Type	General Purpose (SSD)
Allocated Storage*	100 GB

Select Yes to have Amazon RDS maintain a synchronous standby replica in a different Availability Zone than the DB instance. Amazon RDS will automatically fail over to the standby in the case of a planned or unplanned outage of the primary. [Learn More](#).

Settings

DB Instance Identifier*	rdstest
Master Username*	admin
Master Password*
Confirm Password*

* Required

[Cancel](#) [Previous](#) [Next Step](#)

Database Instance Advanced Settings

- **VPC:** The VPC where the Database will be deployed.
- **Subnet Group:** This is the subnet group we created in the very beginning of this post. The database instance will be deployed against the subnets associated with the Subnet Group.
- **Publicly Available:** If you want to allow external access to the database. Usually this will be set to "No", as you want to give access only to addresses in your private subnets.

- **Availability Zone:** This option is disabled for Multi-AZ deployment. The option becomes available whenever you deploy a database instance against a single AZ.
- **VPC Security Group:** This is the security group which will be associated with your Database instance. That security group will provide access to the database listener. This security group was created in the beginning of this post.
- **Database Name:** The Database Service Name, which your database clients will use to connect.
- **Database Port:** The TCP port which the database listener will listen on.
- **DB Parameter Group:** The database engine parameters. We will select the "fusioncloud-test-parametergroup-se1" DB Parameter Group, which we created earlier.
- **DB Option Group:** The features which will be enabled in the database. We will select the "fusioncloud-test-optiongroup-se1" DB Option Group, which we created earlier.
- **Copy Tags to Snapshots:** When this option is enabled, it copies any tag associated to your database instance to the database snapshots. This is useful to track usage and costs.
- **Character Set:** The character set for your database
- **Enable Encryption:** Enable this option if you want your database data to be encrypted.

Step 1: [Select Engine](#)
Step 2: [Production?](#)
Step 3: [Specify DB Details](#)
Step 4: Configure Advanced Settings

Configure Advanced Settings

Network & Security

VPC* DEVTEST-VPC (vpc-af)

Subnet Group devtest-db-subnetgroup

Publicly Accessible No

Availability Zone No Preference

VPC Security Group(s) TEST-RDS-OEM-SG (VPC)
TEST-RDS-SG (VPC)
default (VPC)
devtest-region-vpc-sgNAT-91TUTW8

Database Options

Database Name SOA

Database Port 1521

DB Parameter Group fusioncloud-test-paramete...

Option Group fusioncloud-test-optiongr...

Copy Tags To Snapshots ☒

Character Set Name AL32UTF8

Enable Encryption No

4.6 Creating the Database Instance using the AWS CLI

AWS CLI command based on the previous section walkthrough

```
aws rds create-db-instance --db-name SOA --db-instance-identifier rdst
```

4.7 DNS Alias for the Database Instance

When you create an Oracle RDS Instance, a unique DNS hostname is created for your instance (e.g: `rdstest.c27ee1mbdsas.us-west-2.rds.amazonaws.com`). You can use that hostname to connect to the database.

However, you have no control over the hostname. As a result, you

end up with a database URL that is not easy to remember.

Also, you might need to restore the database from a snapshot. That might be required, for example if an operator makes a mistake in manipulating the data, or if your application corrupts the data due to a bug.

When you restore the database from a snapshot, a new database instance is created. You cannot restore a snapshot to an existing RDS instance. As a result, a new database hostname will be generated.

The last thing you want is to impact existing applications and have to update their database endpoints. For this reason we strongly recommend that you create a DNS Alias for your RDS instance.

Depending on your architecture, you might register the DNS alias either in your corporate DNS server running on-premises, or in a DNS server running in AWS.

We will show now how to register a DNS alias in AWS using Route53, as a private zone. When you use a private zone, only hosts in your VPC will be able to resolve the DNS names for your database. There is a way to extend the name resolution outside of the VPC, but that's out of scope of this blog post.

Create a Route53 Private Zone

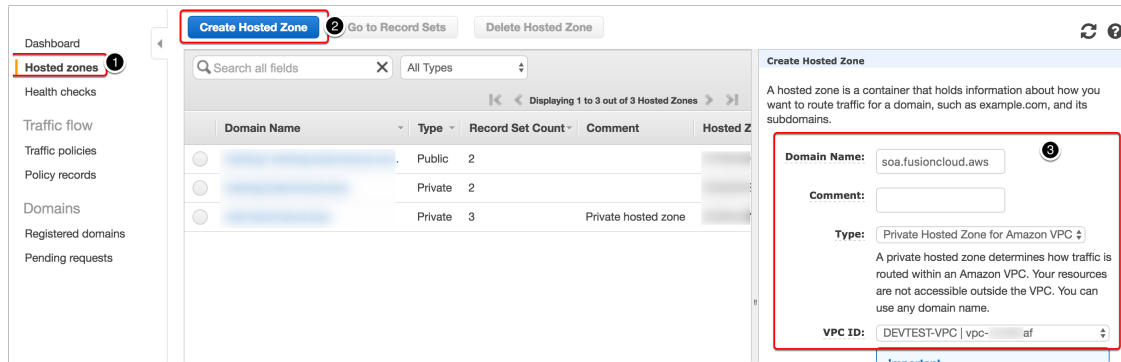
If one doesn't already exist, you need to create a Private Zone.

Using the AWS Console

- In the AWS console, access `Services -> Route53`
- Click on `Hosted Zones -> Create Hosted Zone`

Enter the details of your Hosted Zone:

- **Domain Name:** The private domain name
- **Type:** In this case it will be "Private Hosted Zone for Amazon VPC"
- **VPC ID:** The VPC used by your SOA infrastructure in AWS



Using the AWS CLI

```
aws route53 create-hosted-zone --name soa.fusioncloud.aws --vpc '{"VPC
```

You will then get a confirmation similar to the one below:

```
{
  "ChangeInfo": {
    "Status": "PENDING",
    "SubmittedAt": "2016-06-28T00:22:08.317Z",
    "Id": "/change/C2R8IM38ABCDE"
  },
  "HostedZone": {
    "ResourceRecordSetCount": 2,
    "CallerReference": "dnsfc01",
    "Config": {
```

```
        "PrivateZone": true
      },
      "Id": "/hostedzone/Z2VRA761ABCDEF",
      "Name": "soa.fusioncloud.aws."
    },
    "Location": "https://route53.amazonaws.com/2013-04-01/hostedzon
    "VPC": {
      "VPCId": "vpc-*****af",
      "VPCRegion": "us-west-2"
    }
  }
}
```

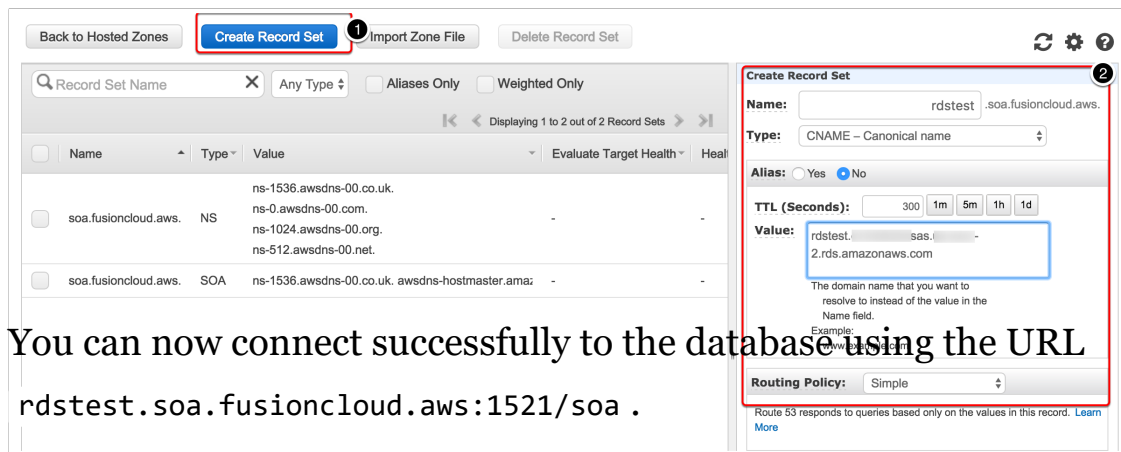
You will need to take notes of the HostedZone Id, to use to create the Record sets in the next section. In the above snippet, the Hosted Zone Id is: Z2VRA761ABCDEF

Create a DNS Alias

Using the AWS Console

Click on "Create Record Set", then enter the following details:

- **Name:** The fully qualified domain name. The value you enter will prefix the domain name. In our example, it will be: `rdstest.soa.fusioncloud.aws`
- **Type:** Select CNAME
- **Alias:** Select No
- **Value:** The RDS instance hostname (make sure you don't add the port information `:1521`)



Using the AWS CLI

```
aws route53 change-resource-record-sets --hosted-zone-id Z2VRA761ABCDE
```

5. Running RCU to create the OFMW Schemas

Running RCU in Interactive Mode

Note: This step is not required if you are using [MyST](#), as MyST will handle this under the hood.

You can run RCU in Interactive mode, using the UI. You have to make sure that you provide the `-honorOMF` flag, otherwise RCU will not be able to create the tablespaces in RDS.

Assuming that your Middleware home is under
`/uo1/app/oracle/product/fmw1221`

You can invoke:

```
./rcu -createRepository -honorOMF -interactive -dbUser admin -dbRole N
```

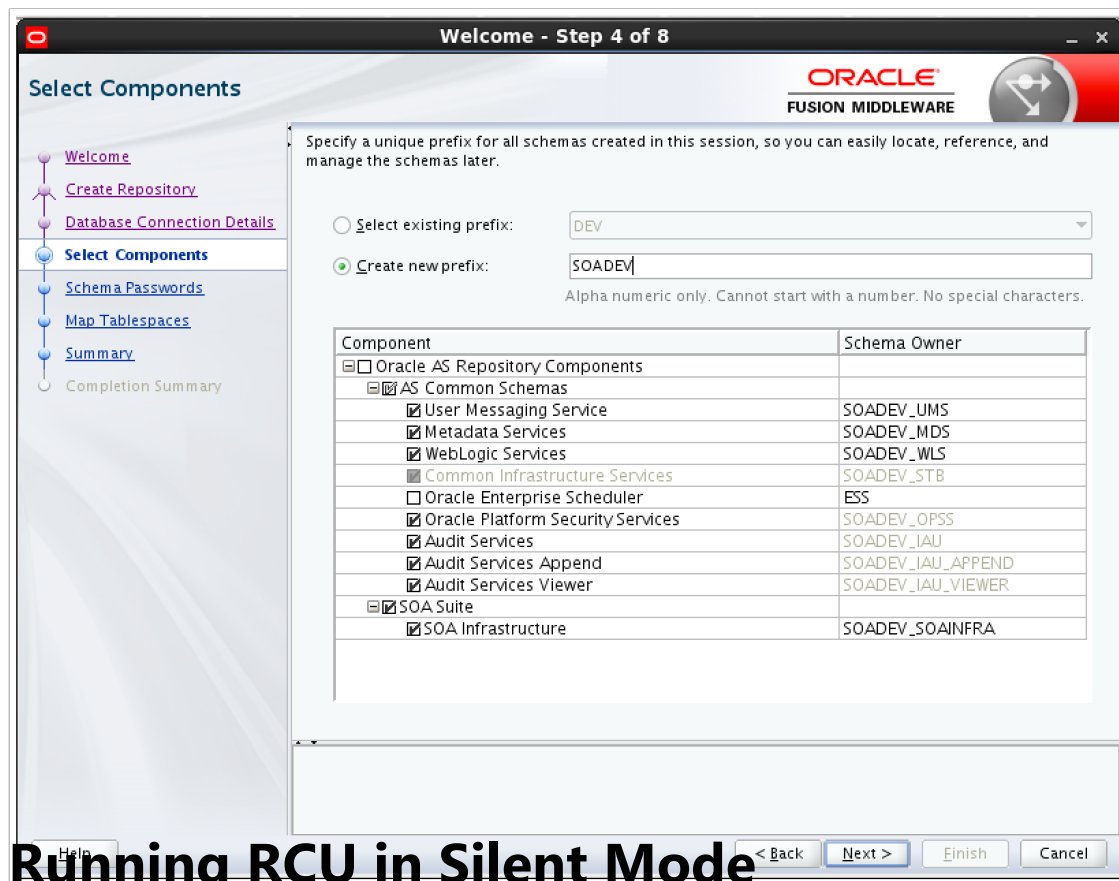
The database details will be automatically populated (except the database password field), as follows:

The screenshot shows the 'Repository Creation Utility' window, titled 'Welcome - Step 3 of 8'. The left sidebar contains a navigation tree with the following items: 'Welcome', 'Create Repository', 'Database Connection Details' (highlighted), 'Select Components', 'Schema Passwords', 'Map Tablespaces', 'Summary', and 'Completion Summary'. The main area displays the 'Database Connection Details' form. The form includes the following fields and values:

- Database Type:** Oracle Database (dropdown menu)
- Host Name:** rdtest.soa.fusioncloud.aws (text field). Below the field, it says: 'For RAC database, specify VIP name or one of the Node name as Host name. For SCAN enabled RAC database, specify SCAN host as Host name.'
- Port:** 1521 (text field)
- Service Name:** SOA (text field)
- Username:** admin (text field). Below the field, it says: 'User with DBA or SYSDBA privileges. Example:sys'
- Password:** (text field with masked characters '••••••••')
- Role:** Normal (dropdown menu). Below the field, it says: 'One or more components may require SYSDBA role for the operation to succeed.'

At the bottom of the window, there are four buttons: 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

You can then define the new Prefix, as well as selecting the RCU components as follows:



Running RCU in Silent Mode

Note: This step is not required if you are using [MyST](#), as MyST will handle this under the hood.

You can also invoke RCU in silent mode, so you can automate the schema creation without using a UI.

This is an example of how you can invoke RCU in silent mode:

```
cd /u01/app/oracle/product/fmw1221/oracle_common/bin
```

```
rcu -silent -createRepository -databaseType ORACLE -honorOMF -connect
```

RCU Related Configuration from MyST Studio

If you use [MyST](#) Studio to do the entire provisioning, you don't need to run RCU manually.

For more information, visit: <http://myst.rubiconred.com> .

MyST will simplify the process, auto computing the required RCU components, and by selecting the "Generic Data Source (AWS RDS)" JDBC Data Source Type, MyST will invoke RCU with the required flags, as follows (snapshot taken from MyST Studio Platform Model wizard):

New Platform Model - true
Configure your Middleware settings.

General Advanced

Domain Name	soa_domain
WebLogic Admin User	weblogic
WebLogic Admin Password	*****
JDBC Data Source type	Generic Data Source (AWS RDS)
RCU Components	OPSS,STB,MDS,IAU_VIEWER,IAU_APPEND,IAU,WLS,SOAINFRA,UCSUMS
RCU Database URL	jdbc:oracle:thin:@rdstest.soa.fusioncloud.aws:1521/soa
RCU Prefix	SOADEV
RCU Database Password	*****
RCU SYS User	admin
RCU SYS Password	*****

6. Conclusion

In this post we covered:

- The benefits of using RDS as the Oracle Fusion Middleware data repository
- The RDS database compatibility with RCU
- The OFMW components which we have covered so far
- How to create an HA Oracle RDS instance from scratch using the AWS Console
- How to run RCU against RDS

The next steps would be to either install Oracle Fusion Middleware products, create the domain and configure according to EDG (Enterprise Deployment Guide) guidelines manually, or use an automation tool such [MyST](#) to do the heavy lifting for you.

In the next blog post, we will cover how to provision an Oracle Fusion Middleware environment in AWS using MyST within minutes.



Fabio Douek

Read more posts by this author.

[Read More](#)

— Cloud Native Application Development,
Integration, Process Automation and DevOps

Middleware

DEVOPS

AUG 01, 2016

Automating the Provisioning of Oracle SOA Suite on AWS

1. Overview Rubicon Red MyST
delivers automated platform
provisioning and continuous delivery
for Oracle Middleware, enabling users

WebLogic Log Inspector: Automated
Error Checking using WebLogic
Admin REST APIs

Tips for Working with Stored
Procedures in OSB 12c

Effective Use Cases for XQuery Library
in OSB 12c

[See all 31 posts →](#)

to deliver a consistent and reliable
platform in minutes. MyST also
provides infrastructure independence;
enabling customers



12 MIN READ

MIDDLEWARE

JUN 28, 2016

Using Panel Grid Layout in Oracle ADF Faces UI for perfect alignment

While working on a number of client projects recently, I've discovered the Panel Grid Layout in ADF is the perfect choice for laying out screens when there are complex data items of



8 MIN READ

Cloud Native Application Development, Integration, Process Automation and DevOps © 2019

[Latest Posts](#) [Twitter](#) [Ghost](#)