

Practice week 04

Complex services deployment on AWS using CloudFormation

Objectives

In this practice you will experiment with the manual services deployment and configuration and using CloudFormation template for automatic service deployment.

The following resources to be deployed and interconnected

- Launch EC2 instances
- Deploy a web server on EC2 instance
- Create a RDS server
- Create a Snapshot for a backup
- Create Elastic Load Balancer

Block 1. Manual creation and configuration of the LAMP Webserver

Part A. Launch EC2 instance

Launch an EC2 instance with “Amazon Linux 2 AMI (HVM)” if you don’t have one running.

(You can choose any other OS instances, if you are comfortable.)

Make sure of your Security Group policies are open for HTTP (port 80). In case it is not open, edit Security Group policy and add HTTP - TCP rule for your specific IP for testing or 0.0.0.0/0 for everyone to access. (This help only when you have something running, can not test now. Wait until **Part B - step d**, to test it)

Part B. Create a web server (LAMP server)

- a) Connect to ec2 instance through ssh (So, all commands below are for the deployed **EC2** instance)
- b) Install Apache webserver with PHP (for testing)
\$ sudo yum update -y
\$ sudo yum install -y httpd.x86_64 php56 php56-mysqlnd
- c) Start the HTTP server
\$ sudo service httpd start
- d) Open web browser to check if server is running, if HTTP port is closed nothing will come, else it will show some test page for HTTP server.
---- > <http://xxxxxxxx.compute-1.amazonaws.com> (replace xx with appropriate path for ec2 instance)
- e) Auto start the webserver server with each restart of instance
\$ sudo chkconfig httpd on
- f) Setting permissions for the Apache web server
\$ sudo groupadd www
\$ sudo usermod -a -G www ec2-user
\$ sudo chown -R root:www /var/www
\$ sudo chmod 2775 /var/www

```
$ find /var/www -type d -exec sudo chmod 2775 {} +  
$ find /var/www -type f -exec sudo chmod 0664 {} +
```

Part C. Create a RDS instance

Use the following steps to setup a MySQL database using AWS RDS service (RDS - Relational Database Service)

- a) Go to RDS console <https://console.aws.amazon.com/rds/> and select MySQL and select RDS Free Tier (as alternative to Production and Dev/Test).
Note: Be aware not requesting creation AuroraDB: It is very expensive because it is large scale relational database




Investigate DB price options for different templates: check at the bottom "Estimated monthly costs"

- b) Step 2: Specify DB details -> You can choose DB engine version. In Settings fill the following details:
DB instance identifier:
Master username: -- remember these details
Master password: -- remember these details
- c) Step 3: Configure advanced settings
Network & Security tab: Leave the default options
Database options tab: choose a Database name and IAM DB authentication : NO
Backup tab: No preference (change to 0/1 days - to reduce the charges for logs)
Monitoring tab: Enhanced monitoring -- Choose Disable
Log exports tab: select General log
Maintenance tab: select "Enable auto minor version upgrade" and "No preference"
Deletion protection: you can choose to protect database or leave it. As this just for testing, you can uncheck so that Database can be deleted.
- d) Create database will take some time. Check the status in RDS -> Databases -> "You_Database_name"

RDS > Databases > mysql-us-east-instance1

mysql-us-east-instance1

Summary

DB Name mysql-us-east-instance1	CPU  1.33%	Info  Available
Role Instance	Current activity  0 Connections	Engine MySQL

[Connectivity & security](#) | [Monitoring](#) | [Logs & events](#) | [Configuration](#) | [Maintenance & backups](#) | [Tags](#)

Connectivity & security

<h4>Endpoint & port</h4> <p>Endpoint mysql-us-east-instance1.ccb9mdfj5j6.us-east-1.rds.amazonaws.com</p> <p>Port 3306</p>	<h4>Networking</h4> <p>Availability zone us-east-1a</p> <p>VPC vpc-6b0cbe11</p> <p>Subnet group</p>
---	---

- e) Once it is created, note down the endpoint and port from “connectivity & Security” tab

Part D. Connect RDS DB instance to Web Server

Follow the steps to create a page on EC2 instance

```
$ cd /var/www
$ mkdir inc
$ cd inc
```

- a) Create a file with your DB credentials to connect “dbinfo.inc” on EC2 instance at the following path ‘/var/www/inc/dbinfo.inc’. Change ‘*endpoint*’ with endpoint noted in **Part C-e** and ‘*master password*’ with password used to create the database in **Part C-b** (Master password)

```
<?php

define('DB_SERVER', 'endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');

?>
```

- b) Create a new file in the html ‘SampleLAMPPage.php’, in the following path ‘/var/www/html/SampleLAMPPage.php’ with sample content below:

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
```

<?php

```
/* Connect to MySQL and select the database. */
$connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
mysqli_connect_error();

$database = mysqli_select_db($connection, DB_DATABASE);

/* Ensure that the Employees table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the Employees table. */
$employee_name = htmlentities($_POST['Name']);
$employee_address = htmlentities($_POST['Address']);

if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
}
?>
```

```
<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
    <table border="0">
        <tr>
            <td>Name</td>
            <td>Address</td>
        </tr>
        <tr>
            <td>
                <input type="text" name="Name" maxlength="45" size="30" />
            </td>
            <td>
                <input type="text" name="Address" maxlength="90" size="60" />
            </td>
            <td>
                <input type="submit" value="Add Data" />
            </td>
        </tr>
    </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
    <tr>
        <td>ID</td>
        <td>Name</td>
        <td>Address</td>
    </tr>
```

<?php

```
$result = mysqli_query($connection, "SELECT * FROM Employees");

while($query_data = mysqli_fetch_row($result)) {
    echo "<tr>";
    echo "<td>",$query_data[0], "</td>";
    echo "<td>",$query_data[1], "</td>";
```

```

        "<td>",$query_data[2], "</td>";
    echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

    mysqli_free_result($result);
    mysqli_close($connection);

?>

</body>
</html>

<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
    $n = mysqli_real_escape_string($connection, $name);
    $a = mysqli_real_escape_string($connection, $address);

    $query = "INSERT INTO `Employees` (`Name`, `Address`) VALUES ('$n',
'$a')";

    if(!mysqli_query($connection, $query)) echo("<p>Error adding employee
data.</p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
    if(!TableExists("Employees", $connection, $dbName))
    {
        $query = "CREATE TABLE `Employees` (
            `ID` int(11) NOT NULL AUTO_INCREMENT,
            `Name` varchar(45) DEFAULT NULL,
            `Address` varchar(90) DEFAULT NULL,
            PRIMARY KEY (`ID`),
            UNIQUE KEY `ID_UNIQUE` (`ID`)
        ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1";

        if(!mysqli_query($connection, $query)) echo("<p>Error creating
table.</p>");
    }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
    $t = mysqli_real_escape_string($connection, $tableName);
    $d = mysqli_real_escape_string($connection, $dbName);

    $checktable = mysqli_query($connection,
        "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
'$t' AND TABLE_SCHEMA = '$d'");

```

```

if(mysqli_num_rows($checktable) > 0) return true;

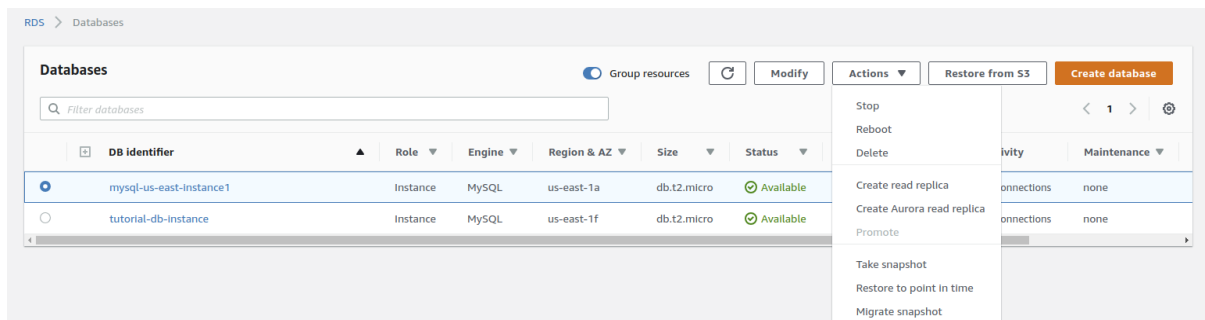
return false;
}
?>

```

- c) Check the website you just created, by following the path
<http://xxxxxxxxx.compute-1.amazonaws.com/SampleLAMPPage.php>
- d) If everything is configured correctly, data entered in this sample page will be saved in the MySQL DB on AWS.

Part E. Create a Snapshot for a backup for RDS

- a) Go to RDS console -> Databases ---> then select "You_Database_name" check box
- b) From "Actions" on top right drop down menu select "Take snapshot" to save the database to Snapshots



- c) Give a name to snapshot and save it. (it will take some time depending on the size of database)
- d) "You can not restore the database to same name - if the database already exists in your AWS account"
 - To restore the database from snapshot. Go to RDS console -> snapshots
 - 1. Select appropriate snapshot --> on top right "Actions" --> "Restore snapshot"
 - 2. Remember to double to check the all the parameters:
 - Change "DB Instance Class": appropriate size (In real time scenarios it might be not a good option to change, just to make sure you will not be charged anything for creating large DB instance)
 - Choose other parameters appropriately to restore the database.
- e) %% Did not find anything to save snapshots to S3
- f) ## This works to shit the existing database to Amazon services, "You can not restore the database to same name"
 - select "Restore from S3"
 - 1. Select engine - Select MySQL (As we have created MySQL RDS server)
 - 2. Specify source backup details:
 - Choose sql version, S3 bucket from which backup needs to be restored.
 - IAM role: **NO**

Part F. Create Elastic Load Balancer

- a) Create another EC2 instance as you did in **Part A**
- b) LAMP webserver installation as you did in **Part B**
- c) Connect RDS DB instance to Web Server as you did in **Part D**

- d) Then follow the Elastic Load Balancer steps from **Week 1 - Part E**

Block 2. Using CloudFormation for creation and configuration of the Sample Webserver

Part G. Using CloudFormation with the Template for Webserver

G.1. Create a Webserver using CloudFormation template

- a) Go to CloudFormation -> Create new Stack
If you want to design own template you can go to Design template and start working.

But for this exercise go with second option:

“Choose a template”-> “Upload a template to Amazon S3” -> Choose file “practice04_cloudformation-sample.json” to upload (practice04_cloudformation-sample.json file is provided on Canvas and shared GoogleDrive folder). If you select this option new S3 bucket will be created to store your template. Be aware of the extra charges.

Note: If you choose to upload file, CloudFormation still creates S3 bucket to put file there.

(Or)

“Choose a template”-> “Specify an Amazon S3 template URL” and give the following path

https://cf-templates-nia2aye4gy8j-us-east-1.s3.amazonaws.com/2020056Jg2-practice04_cloudformation-sample.json

(Note. This link will be working or accessible till the end of course, then it will be deleted)

- b) Specify Details Tab:
Choose the appropriate details for Database
- | | |
|-----------------------|--|
| Stack name | % To identify Stack Created |
| DBName | % Name of the DB to access or store |
| DBPassword | % Password to connect to DB |
| DBRootPassword | % root password for DB |
| DBUser | % DB user - debuser |
| InstanceType | % Choose required resources for EC2 instance |
| KeyName | % used to connect to EC2 instance created |
- c) Next -> Create
d) At the CloudFormation Stacks wait/refresh table

If **CREATE_COMPLETE**, go to console and start using resource

If **ROLLBACK_COMPLETE**, start from the beginning and possibly change template or instance size or location (be aware about costs).

- e) Check CloudFormation output for Webserver URL: put webserver URL into browser and display frontpage

G.2. Experiment with template modification

- a) Make changes to template, e.g. (i) modify Welcome page by changing text and adding image; or (ii) add creation of ‘SampleLAMPPage.php’ in the template like in Part D (b).

Part H. Compare manual services deployment and configuration and using CloudFormation template - in the report.

Self-study questions

Note: To answer self-study questions you may use lecture as a starting point for further study, however additional research and readings will be required.

1. How is section Parameters is used in the CloudFormation template? Do you need to put passwords in the template? Is it secure?
 2. What operations the CloudFormation executes automatically and what instruction you need to define yourself? How do you do this?
 3. Why and for what purpose you need to specify SSH key? How to you do this? Why CloudFormation require existing key?
- A. You need to create a fleet of webserver and other resources. What is the structure of CloudFormation template for this case? What are solutions for this with using CloudFormation and/or other cloud automation tools? Provide example what functionalities available in Ansible (e.g. modules and/or tasks).
- B. Which CloudFormation functions can done in Ansible and which cannot? Provide example of both. How do Ansible and CloudFormation interact during deployment process?

Reporting

Prepare your report and include the following information:

- 1) Document your experiments in Parts B-F, including configuration of your instances and services and few screenshots demonstrating webserver access via Internet.
- 2) The same for Part G Webserver deployment with the CloudFormation template.
- 3) Compare two approaches: manual and with the CloudFormation.
- 4) Provide summary of experience and observations
- 5) Answer at least 3 Self-study questions including at least one question from groups {1,2,3} and {A,B}

Submit your practice report to Canvas Assignment week 4 link by the end of Monday in week 5.