INDEX

Scanned with OKEN Scanner

INDEX

Scanned with OKEN Scanner

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Outcomes

1. Describe various cloud computing platforms and services providers.
2. Illustrate the significance of various types of virtualization.
3. Identify the architecture., delivery models and industrial platforms for cloud computing based applications
4. Analyze the role of security aspects in cloud computing
5. Demonstrate cloud applications in various fields using suitable cloud platforms.

Scanned with OKEN Scanner

Experiment 1.

1. Exploring AWS cloud shell and the AWS cloud9 IDE

Objective: To familiarize yourself with AWS cloudshell by exploring their features, capabilities and use cases.

Step 1: Sign up with AWS free tier.
Step 2: Login to AWS management console.
Step 3: Search for cloudshell icon.
↳ AWS will provide a command line Interface environment.
Step 4: Familiarize with cloudshell.
↳ Pre-Installed tools: python, node.js, git.
aws --version.
python 3 --version.
node --version.
git --version.
test them by.
Step 5: Create a file and verify persistent storage in AWS cloudshell.

a) Create a file.

→ In a cloudshell terminal run a file command and to create a file named test.txt
command - echo "Hello, cloudshell." > text.txt.
→ This command write the text "Hello, cloudshell" into a file called test.txt

b) Verify the file:
→ To confirm that file was created and contains the correct text.
Run → Cat test.txt.
→ This will display the contents of the file.
o/p → Hello, cloudshell.

c) Verify the persistent storage
→ Close the cloudshell tab. Or type exit to close the session
→ Reopen the cloudshell by clicking the cloud shell icon.
→ Run the following command to check whether the file still exist.
command → cat test.txt.
→ If the file still persist you should see shoud as Hello, cloud Shell.

Experiment 2.

2. Working with amazon S3. & orchestrating a serverless function with AWS Step function.

Objective: 1) Learn to create and manage a S3 bucket.
2) Understand how to use AWS Step function to orchestrate serverless workflows.

PART 1.

Step 1: Login to AWS management console.
Step 2: Search for S3 in the search bar and select it.
Step 3: Click create bucket.
Step 4: Create a unique bucket name.
Step 5: Choose a region.
Step 6: Leave all other setting as default and click create bucket.
Step 7: Select the bucket you just created
Step 8: Click upload.
Step 9: Add a file from your local machine.
Step 10: Click Upload.
Step 11: In the bucket click on uploaded file.
Step 12: Select open option to verify the file is accessible.
Step 13: Delete the file and bucket.

Scanned with OKEN Scanner

PART-2 : Orchestrating serverless function with AWS step function.

Step 1: Search for AWS lambda service in the search bar.

Step 2: Create the first lambda function.

Name: functionA

Runtime: Python 3.x

```
import JSON
def lambda_handler (event, context)
    print ("Function A Executed.")
    return {
        status code: 200,
        'body' : json.dump ("Hello from function A")
    }
```

Step 3: Deploy (click) to save function.

Step 4: Create a second lambda function.

Name : Function.B

Runtime : Python 3.x

```
import JSON
def lambda-hadler (event, context);
    print ("Function B executed")
    return {
        Status code: 200,
        'body' : json.dump ("Hello from function B")
```

Step 5: click deploy.
Step 6: In the AWS Management console search for step function and select it.
Step 7: Click create state machine.
Step 8: Click on create your own.
Step 9: Select code option

```
{
  "Comment": A simple workflow to invoke function A & function B.";
  "StartA": "Invoke function A";
  "StartB": {
    "Invoke function A": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "FunctionB";
        "Payload": {}
      }
    }
    "End": true
  }
}
```

Experiment 3.

3. Working with Amazon DynamoDB.

Objective: The objective of the experiment is to familiarize yourself with Amazon DynamoDB, a fully managed NoSQL database service provided by AWS. You will learn how to create a DynamoDB table, insert data and query data.

Step 1: Sign in to AWS console.
1. Go to the AWS Management console.
2. Sign in with your AWS account.

Step 2: Create a DynamoDB Table.
1. In the AWS Management console, search for DynamoDB and select it.
2. Click Create Table.
3. Enter the following details.
   - Table name: Music_collection.
   - Partition key: Artist.
   - Sort key: Song title
4. Leave other settings as default
5. Click create table.

Step 3: Insert Data into the table.
1. Once the table is created go to the Items tab.
2. Click Create item.

Scanned with OKEN Scanner

3. Add the following attributes
   - Artist.
   - Songtitle.
   - year.
4. Click Save.
5. Add more items with different Artist and Song Title values to populate the table.

Step 4: Query Data from the table.
1. In the Items tab., click Query
2. Select the artist attribute and enter John as the value.
3. Click start search.
4. You should see items where.
   Artist = John Doe.
5: Experiment with queries using the sort key (eg. filtering by song title).

Scanned with OKEN Scanner

Experiment 4.

4. Developing REST APIs with Amazon API Gateway.

Step1: Login to AWS Management Console.
Step2: Search for AWS Lambda function.
Step3: Create a function.
function Name - My-API Lambda.
Runtime.: Python 3.13.
Click on Create function.
Step4: code:

```
import JSON.
def lambda-handler (event, context);
returns.
'statuscode' : 200,
'body': json.dumps ("Hello from ILE")
}.
```

Click on Deploy.
Step5: Search for API Gateway.
Step6: Click on Create an API
Step7: Select REST API and click on build.
Step8: New API
API name - my-first-API
Click on Create API
Step9: Click on create method.

Step10: Method type: GET.

Scanned with OKEN Scanner

Step 11: Integration type.- Lambda function.

Step 12: Lambda function : Choose the function created by you previously.

Step 13: Click on create Method.

Step 14: Click on Deploy API
Stage : New Stage.
Stage name : newAPI

Step 15: Click on deploy.

Step 16: Copy Invoke URL and paste in Browser.

Step 17: Output should be visible new.

Step 18: Delete all the created items previously.

Scanned with OKEN Scanner

Experiment 5

5. Creating Lambda Function using the AWS SDK. for python.

Accessing the AWS Managment console.

1. At the top of these instruction., choose start lab to launch your lab. A start lab panel opens and it displays the lab status
2. Wait until you see the message Lab status ready then close the start lab panely by choosing the X
3. At the top of these instructions chose AWS.
4. Arrange the AWS Management console tab so that it displays along side these instructions Ideally, you will be able to see both browser labs at the same time. so that you can follow the lab steps more easily.

Task 1: Configuring the development environment.

5. Connect to the VS code IDE.

Scanned with OKEN Scanner

→ At the top of these instructions choose details followed by AWS : Show.

→ Copy values from the table similar to the following and paste it into an editor of your choice for use later.
- Lab IDE URL
- Lab IDE Password.

→ In a new browser tab, paste the value for LABIDURL to open the VS code IDE.

6. Download and extract the files that you will need for this lab.

7. Run the script that re-creates the work that you completed in earlier labs into this AWS account.

8. To verify the SDK for Python is installed, run the following command in the VS code IDE terminal:
   pip3 show boto3.

9. Take a moment to see what resources the script created.

10. Copy the invoke URL for the API to your clipboard.

11. Update the website's config.js file.

12. Update and then run the update_config.py script:

13. Load the latest cafe webpage with the developer console views exposed.

14. Observe and edit the Python code that you will used in the lambda function.

15. Test the code locally in VS code IDE

16. Modify a setting in the code and test it again.

17. Reverse the change that you made to the code

18. Comment out the last line of the code & save the file

19. Locate the IAM role that the Lambda function will use and copy the role Amazon Resource Number (ARN) value.

20. Edit the wrapper code that you will use to create the lambda function.

Experiment 6.

6. Migrating a Web-Application to Dockers containers.

1. At the Top of these instructions, choose Start lab to launch your lab. A start lab panel opens and it displays the lab status.

2. Wait until you see the message Lab status: ready then close the start lab panel by choosing the X.

3. At the top of these instructions, choose AWS

4. Arrange the AWS Management console tab so that it displays along side these instructions. Ideally you will be able to see both ~~boto~~ browser tabs at the same time so that you can follow the lab steps more easily.

5. Connect to the VS code IDE.

6. Download and extract the files that you will need for this lab.

7. Run a script that assures you can get full credit when you choose to submit this lab.

Scanned with OKEN Scanner

* It also will upgrade the version of python and the AWS CLI that are installed on the VS code IDE.
8. Verify the version of AWS CLI installed.
9. Verify that the SDK for python is installed.
10. Open the existing coffee supplier application in a browser tab.
11. Test the web application functionality.
12. Analyze the web application code.
13. Create a working directory for the node application and move the source code into the new directory.
14. Create a Dockerfile.
15. Build an image from the Dockerfile.
16. Verify that the Docker image was created.
17. Create and run a Docker container based on the Docker image.
18. Verify that the node application is now running in the container.
19. Adjust the security group of the VS code IDE instance to allow network traffic on port 3000 from your computer.
20. Access the web interface of the application which is now running in a container
21. Analyze the database connection issue.

22. Stop and remove the container that has the database connectivity issue.

23. Launch a new container. This time you will pass an environment variable to tell the node application the correction location of database.

Experiment 7.

7. Caching Caching - Application Data with ElastiCache, Caching with - Amazon Cloud Front, caching strategies.

1. At the Top of these instruction, choose start lab.
2. To connect to the AWS management conside, choose the AWS link in the upper-left corner, above the terminal window.
3. If you see a message.
   The credentials in your login link were invalid. To logout click were link then double-click the AWS link above these instructions again.
4. Arrange the AWS Management console tab so that it display along side these instructions
5. Connect to the VS code IDE
6. Download and extract the files that you need for this lab.
7. Run a script to upgrade the AWS CLI that are installed on the VS code IDE It will also recreate the work that you completed in earlier labs into this AWS account.
8. Verify the version of AWS CLI installed.
9. Verify that the SDK for python is installed.

Scanned with OKEN Scanner

10. Configure the security groups that the Aurora serverless instances will use.
11. Configure the database subnet group that the Aurora serverless instance will use.
12. Create an Elastic Cache for Memcached instance.
13. To change the directory that holds the proof of concept scripts return to the VS code IDE. Bash terminal and run the following command.
    cd ~/environment/python3.
14. To install the libraries that Python needs to interact with the database and the cache run the following commands. sudo dnf install -y mariadb-devel gcc python-devel -1.
    sudo pip 3 install python PyMySQL.
    sudo pip 3 install pymemcache.
15. Find the Elasticache for Memcached endpoint.
16. Find the Aurora serverless clusters endpoint.
17. Review the script that will be used to test loading data into the cache.
18. Update the script to define the database & cache connections.
19. Test the find_all.py script.

Scanned with OKEN Scanner

20. Review the script that will be ~~new~~ used to test an update to a record in the Aurora database
21. Test the update item.py script.
22. Now, to test the cache refresh.
    python3 find_all.py.

Scanned with OKEN Scanner

Experiment 8.

8. Implementing CloudFront for Caching and Application Security.

Preparing the development environment.

1. Verify that the AWS CloudFormation stack creation process for the lab has successfully completed.
2. Connect to the VS code IDE.
3. Download and extract the files that you need for this lab.
4. Run a script to upgrade the ~~various~~ version of python and the AWS CLI that are installed on the VS code IDE. The script also ~~installed~~ recreates the work that you completed in earlier labs into this AWS account.
5. Verify the version of AWS CLI installed.
6. Verify that the SDK for Python is installed

   pip show boto3.
7. Note the metadata settings on the objects stored in the S3 bucket & then verify that you can access the cafe website.
8. Begin to configure a cloud-front distribution. for the configure a cloud-front distribution for the cafe website hosted on Amazon S3.

9. Configure the origin settings for the distribution
10. Configure the Default cache behaviour settings.
11. Configure the cache key and origin requests settings.
12. Configure the settings section of the distribution.
13. Finally, chose create distribution
14. Update the bucket policy.
15. Retest access to the cafe website after the update to the ~~cloud front distribution~~ bucket policy.
16. Verify that the cloud front distribution is now enabled.
17. Test the cloudfront distribution
18. First create an IP set for your IP address.
19. Begin to create a web ACL
20. Add a rule to the web ACL configuration to allow requests from the office IP set.
21. Update the new web ACL rule to block any requests that don't match the rule.
22. Set the rule priority, configure metrics and create the web ACL

23. Confirm the web ACL configuration has been applied to the cloud front distribution.
24. Test the AWS WAF configuration that was applied to the cloud front distribution.
25. Create a regional AWS WAF IP set.
26. Begin to create a regional web ACL.
27. Add a rule to the web ACL configuration to allow requests from API Gateway.
28. Update the new web ACL rule to block any requests that don't match the rule.
29. Set the rule priority, configure metrics and create the web ACL.
30. When the web ACL creation process is complete check the resources associated with the ACL.

Scanned with OKEN Scanner

Experiment 9.

9. Orchestrating serverless Functions with AWS step functions.

Step 1: Search for AWS Lambda service in search bar.

Step 2: Create the first lambda function.
Name: Function A.
Runtime: Python 3.X.

```
import JSON.
def lambda_handler (event, context)
print ("Function A. Execute").
return.<
        status code: 200.
        'body': json: dumps ("Hello from.
                              function A")
>.
```

Step 3: Deploy (click) to same function.

Step 4: Create second lambda function.
Name: function B.
Runtime: Python 3.X.

```
import JSON.
def lambda_handler (event, context)
print ("Function B executed")
```

Scanned with OKEN Scanner

```
return {
        statusCode: 200,
        'body': json.dumps("Hello from
          functionB!")
    }
```

Step5: Click deploy.

Step6: In the AWS Management console search for.

Step7: Click create state machine.

Step8: Click on create your own.

Step9: Select code option.

```
{
  "comment": A simple workflow to Invoke
        function A & function B."
  "StartAt": "Invoke function B";
  "State B": {
        "Invoke function A": {
          "Type": "Task"
          "Resource": "arn:aws:states:::
            lambda:invoke";
        "Parameters": {
              "FunctionName": "FunctionA",
        "Payload": {}
```

```
    >
     'Next' : 'Invoke function B".
    >.
     "Invoke function B": <
        "Type": "Task"
       "Reboure": "arn: aws: states:: lambda
        Invoke;
       "Parameters": <.
          "Function.Name." = "FunctionB";
          "Payload" : <>
       >
        "End" : true.
     >.
    >.
  >.
```

Step 10: Click next, click on create.

Step 11: Click on start execution.

Step 12: Delete "roles" state machines and all the functioning.

Scanned with OKEN Scanner

```
>
 'Next' : 'Invoke function B".
>.
 "Invoke function B": <
     "Type": "Task"
     "Resource": "arn: aws: states:: lambda
      Invoke;
     "Parameters": <.
        "Function Name" = "Function B";
        "Payload" : <>
     >
      "End" : true.
   >.
 >.
>.
```

Step 10: Click next, click on create.

Step 11: Click on start execution.

Step 12: Delete roles state machines and all the functioning.

Scanned with OKEN Scanner

Experiment 10:

10. Automating Application Deployment using a CI/CD pipeline.

Steps:

1. Before proceeding to the next step verify that the AWS Cloud formation stack creation process for the lab has successfully completed.
2. Download and extract the files.
3. Run a script that upgrades the version of the AWS CLI installed on the VS code IDE.
4. Create a code commit repository to host the codebase.
5. Create a test file.
6. Review your commit.
7. Add a comment to the committed file.
8. Return to the VS code IDE
9. In the Explorer section expand Environment which is located in the upper

left corner.

10. Expand the resources folder and open the file named cafe-website-front-and-pipeline.jason

11. To create the pipeline run the following commands:
cd ~/environment/resources.
aws codepipeline create-pipeline --cli-input json file://cafe-website-front-and-pipeline.jason.

12. Choose the cafe-website-front-and-pipeline hyperlink and review the pipeline status

13. Retrive the SSH clone URL for your repository.

14. Clone your repository using the SSH URL

15. Explore repository branch management

16. Explore and edit the repository files.

17. Commit your changes to the main branch and review the changes

Scanned with OKEN Scanner

In. codecommit.

18. Return to the VS code IDE tab.

19. On the explorer., Under Environment folder. expand. the front_and_website folder and. delete the test. html file.

20. In VS code IDE bash. & terminal run the following commands.
cd ~/environment.
cd ~/. /resources/website/front-end_website

21. To delete. the website. folder. so that there is one. source. of truth, run the following command.
rm -r.. /resource/website.

22. Commit your changes and return to the browser tab that shows the test. html page.

23. Update the URL be removing test. html from the address and then submit the updated URL.

Scanned with OKEN Scanner

24. Verify that your pipeline applied. the cache control setting.

25. At the top of these instructions choose Submit to record your progress, when prompted choose Yes.

26. If the results don't display after couple. of minutes return to the top of these. Instructions and choose. Grades.

27. Choose End lab at the top of page and then select Yes to confirm that you want to ~~end~~ end the lab.

Scanned with OKEN Scanner