# Final Project for Graduates

Final Project for Graduates for ITMO 544

## Objectives

* Display understanding of AWS Lambda Serverless App Construction
* Demonstrate the use of a NoSQL based database in a cloud native application
* Demonstrate successful deployment of an SNS Notification topic and service

### Create Script

The list of variables and their order for create-env.sh is as follows:

1. ImageID
2. Count
3. Subnet 1 (availability zone a for your region) for load balancer
4. Subnet 2 (availability zone b for your region) for load balancer
5. subnet-id for EC2 launch instance (availability zone a) (may or may not be used depending on your design)
6. Security Group ID
7. Load balancer name
8. Target Group name
9. Key-pair name
10. auto-scaling-group-name
11. launch-configuration-name
12. vpc-id You can have the user prompt this or you can retrieve it
13. S3 bucket name raw
14. IAM profile name
15. DynamoDB table name
16. S3 bucket name for thumbnail images – new element
17. AWS Lambda user role (will be presented as: arn:aws:iam::548002151864:role/service-role/jrn-inclass-lambda-role-veorp4bo)

* Create the SQS Message Queue, hardcode the name
* Create a SNS Topic, hardcode the name

#### Specifications

* Create a DynamoDB table
  + Table should reuse any fields from the previous SQL schema, but you can add additional attributes as needed (for instance finished S3 URL)
* Create a Lambda function (assume that the Lambda IAM role has already been created)
  + Function name is **EditorFunction**
  + <https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/create-function.html>
* Include the compiled Python Lambda package code as editor.zip – including the Pillow package (will require local installation for packaging)
  + <https://docs.aws.amazon.com/lambda/latest/dg/lambda-python-how-to-create-deployment-package.html#python-package-dependencies>

#### editor.zip

* In this code the Lambda function will execute the “backend” image processing
  + Name the file for the Python Lambda editor.py
    - Name the handler: --handler editor.handler
    - <https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html>
  + Retrieve Queue message with UUID
  + Retrieve DynamoDB record
  + Subscribe the phone number to the SNS Topic (Describe SNS topics)
    - <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html#SNS.Client.subscribe>
    - Protocol is *sms*
    - Endpoint is full US phone: 16305551212
  + Retrieve image from S3 raw bucket
  + Process image to thumbnail image
    - Using PIL or Pillow render a thumbnail of the original image
    - Save the file as thumbnail-UUID.jpg where UUID is a unique UUID generated via the UUID library
  + Place the rendered image file, thumbnail-UUID.jpg into the S3 bucket finished bucket
  + Update DynamoDB record (add finished URL, change stat from 0 to 1)
  + Publish SNS message to subscribed members - stating image editing is done
    - <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html#SNS.Client.publish>
    - Use TopicArn and PhoneNumber
    - Message Body: “Hello {Insert Name}. Your image has been rendered”
  + Delete SQS Message

#### Lambda Trigger

There are two ways to trigger the Lambda function by using the [create-event-source-mapping](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/create-event-source-mapping.html)

* Using SQS to trigger the Lambda
  + <https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html>
* Using DynamoDB to trigger the Lambda
  + <https://docs.aws.amazon.com/lambda/latest/dg/with-ddb-example.html>
  + <https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html>
* <https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/create-event-source-mapping.html>

### Install Script

The install-env.sh script will remain the same for this exercise as the previous week. You will be adding an additional files:

* index.html
  + <https://github.com/illinoistech-itm/jhajek/>
* app.js
  + Create an additional route call /gallery to display the contents of DynamoDB

#### index.html

The form will collect:

* Name
* Email
* Phone Number
* [How to set a default value](https://www.w3schools.com/tags/att_input_value.asp)

#### app.js

Reuse your existing app.js, with this final addition:

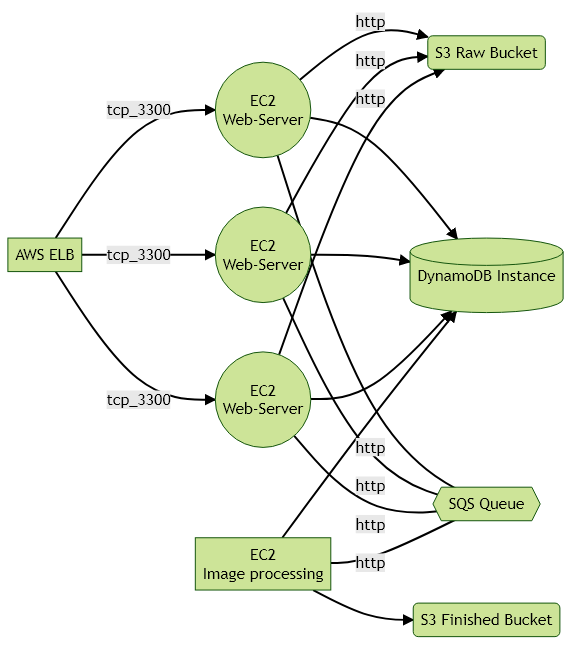
/\* you should add this function to the end of your app.js \*/  
/\* This will create a new route/page located at /gallery \*/  
app.get('/gallery', function (req, res) {  
  
/\* List Content of the DyanmoDB database here or you can list the objects in S3 Bucket\*/  
  
 res.write("Gallery");  
 /\* list the content in a simple list via the res.write() function \*/  
 res.end();  
  
});

### Destroy Script

The destroy-env.sh script’s purpose is to peel back what you have created and delete or terminate all of the launched resources.

* Deregister targets attached to target-group
* Use elbv2 wait command for targets to be deregistered
* Delete listener
* Delete target-group
* Delete Load-balancer
* Use elbv2 wait command for load-balancers to be deleted
* Use EC2, terminate all three instances
* Use the EC2 wait command until all instances are terminated
* Delete DynamoDB tables (use waiter)
* Delete SQS Queue
* Delete the AutoScaling group launch configuration and delete-auto-scaling-group
* Delete AWS Lambda
* Delete Raw and Finished S3 buckets
* [Delete event source mapping](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/delete-event-source-mapping.html)
* Delete SNS Topic (Force)
* Print a message that all resources have been terminated/deleted

## Diagram



*FlowChart of Application*

## Deliverable

Push all code and the following deliverables to your GitHub repo in a folder named **final**. Submit this URL to Blackboard.

* create-env.sh
* install-env.sh
* destroy-env.sh
* index.html
* app.js
* editor.py
* editor.zip
* Readme.md (give any instructions or modifications needed to run your project)
  + Show the command you use to launch the install
* Plus the additional sample code used to build and start your images (include the folder structure)