IAM and SERVERLESS Lab

You need to create a simple REST API using serverless in AWS. The API must have API Gateway and two Lambda functions. The REST API must have two end points (/users and /upload-file). The /users endpoint must retrieve dummy data from a dummy API and the /upload-file endpoint must allow you to upload files in an already created bucket (accedia-aws-training-files-bucket) via a simple frontend page.

\*NOTE: The code for the Lambda functions is already provided in a git repository

\*NOTE2: All resources must be with names with the following pattern: “lab-<your-name>-<name-of-resource>”, example: “lab-vladimir-vladimirov-rest-api”

\*NOTE3: Delete all the resources you created for the lab (described below)

1. Create public REST API in the AWS API Gateway - “lab-<your-name>-rest-api”
   1. Go to Services -> API Gateway -> Create API
   2. Choose REST API -> Build
   3. Pick API name -> Create API
2. Create Lambda Function to retrieve users - “lab-<your-name>-get-users”
   1. Go to Services -> Lambda -> Create function
   2. Choose Author from scratch -> pick a Function name -> pick a Runtime “Node.js 12.x” -> Create function
   3. Choose Code entry type “Upload a .zip file”
   4. Upload “GetUsers.zip” -> Save
3. Create AMI Policy with permission to put objects in “accedia-aws-training-files-bucket” - “lab-<your-name>-s3-putObject“
   1. Go to Services -> IAM
   2. Choose Policies -> Create policy
   3. Choose service S3
   4. Choose action Write -> PutObject
   5. Choose Resources -> Specific -> Add ARN -> Pick Bucket Name “accedia-aws-training-files-bucket” and Object name “Any” or “\*”
   6. Review policy -> Create policy
4. Create Lambda Function to upload files in S3 bucket with permissions to put objects in “accedia-aws-training-files-bucket” - “<your-name>-lab-upload-file”
   1. Go to Services -> Lambda -> Create function
   2. Choose Author from scratch -> pick a Function name -> pick a Runtime “Node.js 12.x” -> Create function
   3. Choose Code entry type “Upload a .zip file” or “Edit code inline”
   4. Upload “UploadFile.zip” or copy-paste the code -> Save
   5. Basic settings -> Edit -> View the <your-name>-upload-file-role”
   6. Attach policies -> find your policy from the previous step -> Attach policies
5. Create /users resource with GET method in your REST API in API Gateway
   1. Go to Services -> API Gateway -> select your API
   2. Actions -> Create Resource
   3. Pick resource name -> pick resource path -> Create Resource
   4. Actions -> Create Method -> GET
   5. Choose Integration type “Lambda Function”
   6. Tick “User Lambda Proxy integration -> choose Lambda Function “<your-name>-lab-get-users” -> Save
   7. Actions -> Deploy API -> Create new stage “dev”
   8. Test your endpoint in the browser - <your-api-url>/dev/users
6. Create /upload-file resource with POST method in your REST API in API Gateway
   1. Actions -> Create Resource
   2. Pick resource name -> pick resource path -> tick Enable API Gateway CORS -> Create Resource
   3. Actions -> Create Method -> POST
   4. Choose Integration type “Lambda Function”
   5. Tick “User Lambda Proxy integration -> choose Lambda Function “<your-name>-lab-upload-file” -> Save
   6. Actions -> Deploy API -> Pick “dev” stage
   7. Test your endpoint - open the index.html file and upload a file successfully in the S3 “accedia-aws-training-files-bucket”

Delete the following resources:

1. REST API in the API Gateway
2. Two lambda functions
3. The lambda functions roles
4. The policy lab-<your-name>-s3-putObject in IAM Policies
5. The roles lab-<your-name>-get-users-role and lab-<your-name>-upload-file-role in IAM Roles
6. The log groups /aws/lambda/lab-<your-name>-get-users and /aws/lambda/lab-<your-name>-upload-file in CloudWatch Log groups