Project – Build Your Website in the AWS cloud

Due to recent university policy, **you are required**

1. to make a 15-minute recording explaining the project and how you did it.
2. Also, commit daily to Git during the project. In the **readme** file, write what you did for the day.

Sample projects:

- https://meijuan-long.click/
- https://sdbappi.com/
- https://tanvirmahboob.click/

Benefits of the project

1. You will build out your own website that impresses people and recruiters. You will stand out from other candidates during the job search.
2. Gain hands-on experience with modern cloud technologies. I assure the technologies you learned in this class help you shine during the job interview and at work.

Goals

1. Build a fault-tolerant, highly scalable, production-ready API on the cloud.
2. Create a front-end app that utilizes the API – It will help you understand how back-end and front-end developers build a whole application.

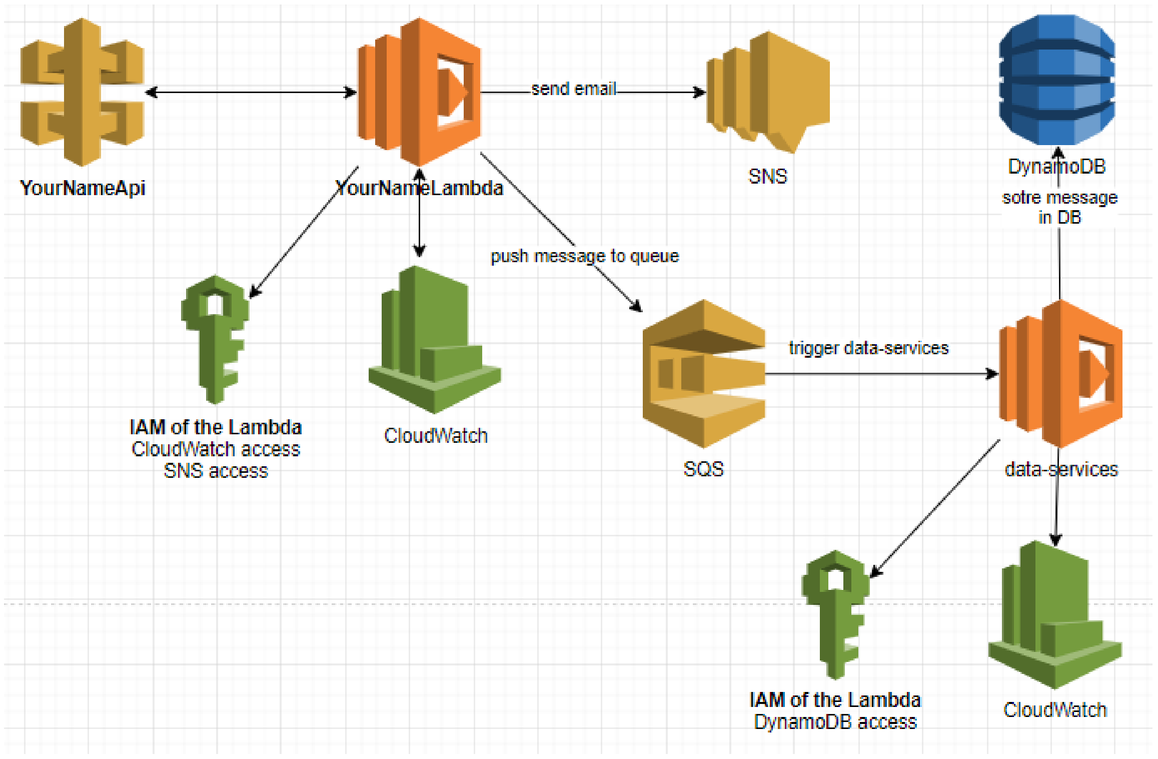
Tasks

1. Build the front end using React, Angular, or any front-end libraries and framework. Or you can use templates online. Deploy it in S3. Put **CloudFront** in front of the S3 bucket.
2. Build the back-end API using serverless services. For example, Lambda, DynamoDB, API Gateway, SNS, SQS, and/or S3, AWS Step Functions.
3. Buy a domain name (or get one for free from GitHub) and configure that in **Route53**. Buying a domain on AWS would be much easier for you to integrate whereas free domains sometimes don't work as it is not all yours! You can get a free certificate from **ACM** (Amazon Certificate Manager).
4. Come up with an interesting idea and implement it. **Blog** how you came up with and implemented that interesting idea on your web.

Example ideas from other students:

- Real-time user tracking dashboard https://meijuan-long.click/latestblog/latest.html
- Migrating the project from an AWS Academy account to a personal account with CLI https://sdbappi.com/blog1.pdf
- ChatBot with Lex
- Dynamic Blogging
- Integration with Cognito
- Amplify, SAM
- CICD with Composer, CloudFormation, CodeCommit, CodePipeline
- ChatGPT integration and so on.

Example API architecture



Front-end requirements

Web sections:

- Intro. One statement about yourself.
- About. More detailed. 2 or 3 paragraphs
- Skills. List technologies you know.
- Work experience. Projects you did at MIU if you have no prior experience.
- Education
- Blog about how you implemented your idea. Step by step.
- Contact me section. It will call the back-end API.
  - It should print a success or error message when sending a message.
  - Phone number is optional. There should be client-side validation.
  - Your front-end app will have a form that has 5 fields (MessageTitle, Message, Email, GuestName, Phone [Optional])

Score breakdown

Project score breakdown:

- Domain name and certificate - 4 point
- CloudFront – 3 points
- Front-end (styling and customization) – 3 points
- Client-side validation and success/error messages - 2 points
- Back-end – 4 points
- Your idea – 4 points

The domain name and CloudFront setup

- You can get a free domain at GitHub. But still recommend your own .com domain that would look more legitimate and elegant for you.
- Domain name and certificate
    - Buy a domain, for example, GoDaddy or AWS.
    - Create a hosted zone for your domain in Route 53.
    - Copy 4 NS records and paste them into the domain name provider. So you can manage your domain name in AWS. If the domain name provider accepts 2 NS records, put 2 of them then.
    - Go to ACM (Amazon Certificate Manager) and hit the "request a certificate".
    - Qualified domain names are
        - yourname.com
        - *. yourname.com
    - After the request has been created, click on that and click on "Create records in Route 53". Or manually copy and paste.
- CloudFront
    - Select the bucket in the origin domain.
    - Select OPTIONS and all other HTTP methods in Allowed HTTP methods.
    - Enter a domain name in the Alternate domain name (CNAME).
    - Select the certificate in Custom SSL certificate - If you have the certificate issued by ACM.
    - Enter index.html as the Default root object.
    - In Route53, create an A record. Toggle Alias. Then select the CloudFront distribution. Make sure you enter "the Alternate domain name (CNAME)" in CloudFront.
    - If your app is not replicating the changes after redeploying your front-end on S3 and you have a CloudFront distribution in front of that, go to the "Invalidation" then invalidate all with ("/*")

References

Instructions for setting up CloudFront, Domain name, certificates: React App on AWS S3 with Static Hosting + Cloudfront | Practical AWS Projects #1 on "Be better dev channel".

- Create a certificate for your website with Amazon Certificate Manager. You will use that when creating a CloudFront distribution in the next step. Refer: https://docs.aws.amazon.com/acm/latest/userguide/gs-acm-request-public.html
    - Make sure you selected DNS validation. Once you create the certificate on AWS ACM, it will generate a CNAME record. That you will register in your hosted zone.
- Create an AWS CloudFront distribution for the S3 bucket. That will distribute your code all over the world. Refer https://aws.amazon.com/cloudfront/getting-started/S3/.
    - Select the issued certificate when creating the CloudFront distribution.
- Create a DNS record for the CloudFront distribution on Route 53. Refer: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-cloudfront-distribution.html

- Now your website is accessible from both S3 and CloudFront. The best practice is to access only through CloudFront. Refer: https://aws.amazon.com/premiumsupport/knowledge-center/cloudfront-access-to-amazon-s3/
- CORS error: How do I resolve a CORS error for my API Gateway REST API? there are many more resources on the internet if you google.