Project – Build Your Website in the AWS cloud

Sample projects:

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

Benefits of the project

- You will build out your own website that impresses people and recruiters. You will stand out from other candidates during the job search.
- 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

- Build a fault-tolerant, highly scalable, production-ready API on the cloud.
- 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.

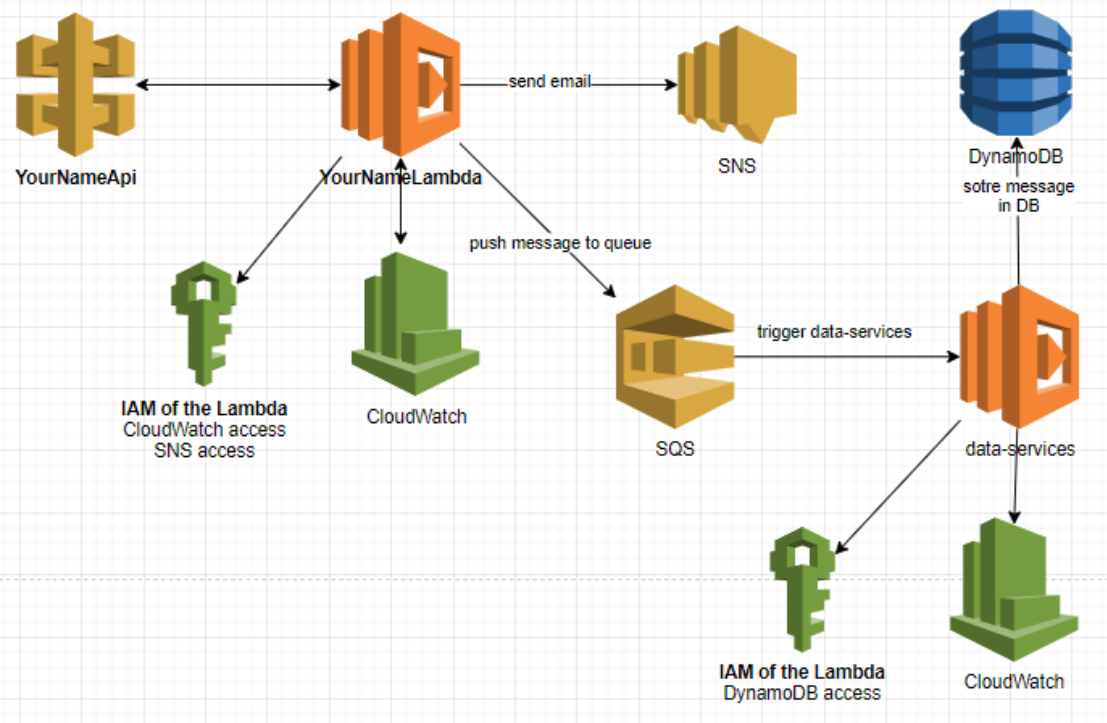
Tasks

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 i.e., Lambda, DynamoDB, API Gateway, SNS, SQS, and/or S3, AWS Step Functions. There must be:
    a. API Gateway
    b. Lambda
    c. DynamoDB.
3. Buy a domain name in **Route53**. Otherwise, you would need to manually register the NS records in the other domain name provider. To check if that succeeded, you can use tools like https://www.nslookup.io/ that must resolve the NS records in Route53.
4. **Blog** an interesting technical topic on your web. It can redirect to your Medium blog.

Examples of interesting 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
- 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)

Score breakdown

Project score breakdown:

1. Domain name and certificate - 4 points
2. CloudFront – 4 points
3. Back-end – 4 points
4. Front-end (styling and content customization) – 2 points
5. Blog something interesting – 2 points

Reference

The domain name and CloudFront setup

- Some students said you can get a free domain on GitHub. It is harder to set up a free domain as there could be restrictions. Highly recommend you buy your own domain on Route53. In case, if you bought the domain name from another domain name provide:
    - 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 https://www.nslookup.io/ and enter your domain that must resolve the AWS NS records before proceeding to the next step.
- 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 – I got a script that automatically does all these things at https://engineering.handpro.mn/cloud-5-cicd-and-cdk-d0b624f9472e with AWS CDK. But that is okey, you can manually do it:
    - 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.