REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

About Projects

When talking about projects for learning, there are three main types of projects you can do. First, there's the hand-holder. This is the type that tells you what to do and you follow along. Second, there's the type that gives you an end result with some initial guidance and guardrails, but leaves the implementation to you. And third, there's projects that you conceive of yourself and work through, developing whatever skills you need along the way.

Naturally the third kind is the most valuable to go through. It shows initiative, and you may find that things you wanted to do are impossible, and have to find alternatives. That's what builds experience. Sounds great, right? Well, the problem is that these sorts of projects are a lot of hard work and you may feel that you don't have the experience to achieve anything at all without some guidance. In this project, we've tried to give you some guidance without giving it to you on a silver platter - you're going to have to step outside your comfort zone, do your own research, and fail a lot. But it will be worth it for the learning and knowledge you'll gain from it in the end. You may just get through Level 1 in this project, and that's okay - as long as you learnt something new. There is no final grade on this project. It's for you to learn as much as you can. You be the judge of that for yourself.

Just remember; this is not going to be easy. And that's exactly how it should be.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

About this Project

Pre-requisites:

- You have completed the classroom work of the first three weeks in the Summer of AWS course.
- You are familiar with programming/scripting. You don't need to be a master programmer or anything like that - but you do need to have a grasp of the basics and know how to troubleshoot on your own. A lot.
- IMPORTANT: You are willing to dedicate time to overcome complex issues.
- You have an AWS Account and a marginal amount of money to spend improving your skills. The project should be mostly free, but there may be a few small costs over the course of the project for some AWS uses.

How to use this guide:

- This is not a step by step how-to guide.
- It is up to you to take each goal and figure it out on your own. We have included hints to point you in the right direction, but you should expect to do plenty of your own research and development.
- Google is your friend. AWS Documentation is your friend. Stack Overflow is your friend.
- Find out and implement the "right way", not the quick way. Or at least do the quick way first then refactor to the right way before moving on.
- Shut down or de-provision as much as you can between learning sessions. You should be able to do everything in this guide for literally less than $50 using the AWS Free Tier. Rebuilding often will reinforce concepts anyway.
- Instead of waiting till the end of the project, make sure you read the Cost Analysis and Automation sections first and have them in the back of your mind as you work through the goals.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

About the Client

Cotiss is built specifically to help organisations be more effective in how they buy & supply goods and services. They're a B2B company that creates software for companies to track their spending and make better business decisions.

The Honest-Feedback project:

As Cotiss has grown over the years, the need for collecting feedback internally and externally has become a more prominent issue. Cotiss leadership is looking for a simple website where employees can anonymously submit feedback. To encourage employees to share honestly, they want a randomly selected piece of existing feedback to be displayed on the site on each page load.

The UX design is simple; a random piece of feedback displayed on each page load, and a box at the bottom of the page with a submit button to add new feedback to the feedback bank.

Our advice:

While the project itself is simple, the technology behind it has many, many layers. In the spirit of agile ways of working, we've devised an outline of the project that results in iterative versions of the solution, each increasing in complexity.

There is no minimum requirement for how much of the project you need to complete. Do as much as you can for the more you do, the more you'll learn, and the better your chances of passing the exam. This can also be used as a project for interviews and job applications. The more of the project you do, the more you'll stand out. Moral of the story: do as much as you physically can in this time period. It will pay off big time.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Level 1 - Off we go!

Account Basics
GOAL: You can use the AWS CLI to retrieve information about your AWS account.

- Create an IAM user for your personal use.
- Set up MFA for your root user, turn off all root user API keys.
- Set up Billing Alerts for anything over a few dollars.
- Configure the AWS CLI for your user using API credentials.

  Checkpoint: You can use the AWS CLI to retrieve information about your AWS account.

Web Hosting Basics
GOAL: Create a simple HTML page served from your EC2 instance.

- Deploy a EC2 virtual machine (VM) and host a simple static "Honest-Feedback Coming Soon" web page.
- Take a snapshot of your VM, delete the VM, and deploy a new one from the snapshot (ie. enable disaster recovery)

  Checkpoint: You can view a simple HTML page served from your EC2 instance.

Auto Scaling
GOAL: You can view a simple HTML page served from both of your EC2 instances. You can turn one off and your website is still accessible.

- Create an Amazon Machine Image from your virtual machine and put it in an autoscaling group so one virtual machine always exists.
- Put an Elastic Load Balancer in front of that virtual machine and load balance between two Availability Zones (one EC2 in each Availability Zones).

  Checkpoint: You can view a simple HTML page served from both of your EC2 instances. You can turn one off and your website is still accessible.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

External Data
GOAL: Your auto scaled website can now load/save data to a database between users and sessions

- Create a DynamoDB table and experiment with loading and retrieving data manually, then do the same via a script on your local machine.
- Refactor your static page into your Honest Feedback website (feel free to use Node, PHP, Python, or whichever programming language you like) which reads/updates a list of Feedback entries in the AWS DynamoDB table. *(Hint: EC2 Instance Role)*

  Checkpoint: Your Highly Available/AutoScaled website can now load/save data to a database between users and sessions

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Level 2 - It's getting serious

Web Hosting Platform-as-a-Service
GOAL: The same as before, except you don't have to manage the servers, web server software, website deployment, or the load balancer.

- Retire that simple website and re-deploy it on Elastic Beanstalk.
- Create a S3 Static Website Bucket, upload some sample static pages/files/images. Add those assets to your Elastic Beanstalk website.
- Register a domain (or re-use an existing one. It's a good experience if you haven't done it already! Just get a cheap one). Set Route53 as the Nameservers and use Route53 for DNS. Make *www.yourdomain.com* go to your Elastic Beanstalk website. Make *static.yourdomain.com* serve data from the S3 bucket.
- Enable SSL for your Static S3 Website. This isn't exactly trivial. (Hint: CloudFront + AWS Certificate Manager (ACM))
- Enable SSL for your Elastic Beanstalk Website.

  Checkpoint: Your AutoScaled website now serves all data over HTTPS. The same as before, except you don't have to manage the servers, web server software, website deployment, or the load balancer.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Level 3 - Put on an extra pair of underpants

Microservices
GOAL: Your Elastic Beanstalk deployment is only used to retrieve data from your database. All of your UI and application logic is served from the S3 Bucket via CloudFront. By doing this, you can support many more users since you're no longer using expensive servers to serve your website's static data.

- Refactor your Elastic Beanstalk website into ONLY providing an API. It should only have a POST/GET to update/retrieve that specific data from DynamoDB. Get rid of *www.yourdomain.com* and serve this Elastic Beanstalk as *api.yourdomain.com*
- Move most of the UI piece of your Elastic Beanstalk website into your Static S3 website and use Javascript (or another language of choice) to retrieve the data from your *api.yourdomain.com URL* on page load. Send data to the Elastic Beanstalk URL to have it update the DynamoDB. Get rid of *static.yourdomain.com* and change your S3 bucket to serve from *www.yourdomain.com*.

  Checkpoint: Your Elastic Beanstalk deployment is now only a structured way to retrieve data from your database. All of your UI and application logic is served from the S3 Bucket (via CloudFront). You can support many more users since you're no longer using expensive servers to serve your website's static data.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Level 4 - 10 - Living through the fear

Serverless
GOAL: You have no EC2 instances deployed. All work is done by AWS services and billed as consumed. Your API Gateway and S3 Bucket are fronted by CloudFront.

- Write an AWS Lambda function to email you a list of all of the Feedback entries in the DynamoDB table every night. Implement Least Privilege security for the Lambda Role. *(Hint: Lambda using Python 3, Boto3, Amazon SES, scheduled with CloudWatch)*
- Refactor the above app into a **Serverless app.** This is where it gets a little more abstract and you'll have to do a lot of research, experimentation on your own.
    - The architecture: Static S3 Website Front-End calls API Gateway which executes a Lambda Function which reads/updates data in the DynamoDB table.
- Use your SSL enabled bucket as the primary domain landing page with static content.
- Create an AWS API Gateway, use it to forward HTTP requests to an AWS Lambda function that queries the same data from DynamoDB as your EB Microservice.
- Your S3 static content should make Javascript calls to the API Gateway and then update the page with the retrieved data.
- Once you have the "Get Feedback" API Gateway + Lambda working, do the "New Feedback" API.

  Checkpoint: Your API Gateway and S3 Bucket are fronted by CloudFront with SSL. You have no EC2 instances deployed. All work is done by AWS services and billed as consumed.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Level 11 - 20 - You’re a Cloud Computing Physco

Automation
GOAL: Stay sane.

- These technologies are the most powerful when they're automated. You can make a Development environment in minutes and experiment and throw it away without a thought. This stuff isn't easy, but it's where the really skilled people excel.
- Automate the deployment of the architectures above. Use whatever tool you want. The popular ones are AWS CloudFormation or Terraform. Store your code in AWS CodeCommit or on GitHub. You can automate the deployment of ALL of the above with native AWS tools.

Continuous Delivery
GOAL: Stay sane

- As you become more familiar with automating deployments you should explore and implement a Continuous Delivery pipeline.
- Develop a CI/CD pipeline to automatically update a dev deployment of your infrastructure when new code is published, and then build a workflow to update the production version if approved. Travis CI is a decent SaaS tool, but Jenkins has a huge following too. If you want to stick with AWS-specific technologies you'll be looking at CodePipeline.

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

No matter where you get to: Finishing it off

Cost Analysis

- Explore the AWS pricing models and see how pricing is structured for the services you've used.
- Answer the following for each of the main architectures you built:
    - Roughly how much would this have cost for a month?
    - How would I scale this architecture and how would my costs change?
- Architectures
    - **Basic Web Hosting**: Highly Available EC2 Instances serving static web page behind Elastic Load Balancing.
    - **Microservices**: Elastic Beanstalk SSL Website for only API + S3 Static Website for all static content + DynamoDB Table + Route53 + CloudFront SSL
    - **Serverless**: Serverless Website using API Gateway + Lambda Functions + DynamoDB + Route53 + CloudFront SSL + S3 Static Website for all static content

REAL-WORLD PROJECT BRIEF

Summer of AWS 2022 - 2023

Final Thoughts

This project is one you can build and show off as proof of your skills. To get maximum returns on your efforts, open source it on GitHub, make professional documentation, comment as much as is reasonable, and host a demo of the website. Add links to your LinkedIn, reference it on your resume, work it into interview answers….the whole thing. When in a job interview you'll be able to answer all kinds of real-world questions because you've been-there-done-that with most of AWS' major services.