

## Hosting Static Web Application in AWS (IaC)

Jordan L. Sumner

Western Governors University

## Table of Contents

Proposal Overview.....	5
Problem Summary .....	5
IT Solution .....	6
Implementation Plan .....	7
Review of Other Work.....	7
Project Rationale.....	13
Current Project Environment.....	14
Methodology .....	15
Project Goals, Objectives, and Deliverables.....	16
Project Timeline with Milestones.....	19
Outcome.....	21
References.....	23
Appendix A.....	24
Design .....	24
Appendix B.....	25
Version Terraform File .....	25
Appendix C .....	26
Variables Terraform File .....	26
Appendix D.....	27
ACM Terraform File.....	27
Appendix E .....	28
CodePipeline Terraform File I.....	28
Appendix F .....	29
CodePipeline Terraform File II .....	29

Appendix G.....	30
CloudFront Terraform File I.....	30
Appendix H.....	31
CloudFront Terraform File II.....	31
Appendix I .....	32
AWS Console ACM .....	32
Appendix J .....	33
AWS Console CloudFront.....	33
Appendix K.....	34
AWS Console CodePipeline.....	34
Appendix L .....	35
AWS Console S3 .....	35
Appendix M.....	36
AWS Console S3 Contents.....	36
Appendix N.....	37
CloudFront Servers.....	37
Appendix O.....	38
Web Application.....	38
Appendix P .....	39
CodePipeline Error Fix I.....	39
Appendix Q.....	40
CodePipeline Error Fix II .....	40
Appendix R.....	41
CodePipeline Error Fix III.....	41
Appendix S .....	42
CodePipeline Error Fix IV.....	42

Appendix T .....	43
CodePipeline Error Fix V .....	43
Appendix U .....	44
CodePipeline Error Fix VI .....	44
Appendix V .....	45
CodePipeline Error Fix VII .....	45
Appendix W .....	46
CodePipeline Error Fix VIII .....	46
Appendix X .....	47
CodePipeline Error Fix IX .....	47
Appendix Y .....	48
Success Metrics .....	48

## Proposal Overview

### Problem Summary

An open-source web application developer has created a geography guessing game named “Sum Cloud,” registered with Route53 on Amazon Web Services (AWS) under SumCloud.net. This game was developed in their apartment to help combat boredom under pandemic restrictions. The developer has been showcasing his open-source code for their game on their personal GitHub account and has been receiving constant push updates from other users to enhance the game's code.

This developer has been hosting their static web application on their home laptop using a virtual machine and has seen a struggle on their hardware with increased traffic to their website. The developer has also received personal messages from users from other parts of the country complaining about the extended load time to visit the website.

The developer has decided to leverage cloud computing technology and has asked a third-party company to develop an environment to host their static web application game on AWS to help relieve the increased traffic and latency from his on-premises server. The developer of Sum Cloud believes that moving their on-premises web applications to a cloud environment would significantly reduce and eliminate some of the complaints they have received from users.

The developer of Sum Cloud has asked a third-party company, Cloud Solutions, to develop the groundwork for hosting their web application game in AWS. They have asked Cloud Solutions to create a deliverable that they can quickly deploy on their AWS account. Cloud Solutions has developed a solution by making the environment with the resources needed as IaC instead of through the AWS Console. This will allow the developers of Sum Cloud to utilize the IaC template and quickly deploy or destroy the cloud environment on their own personal AWS

account.

### IT Solution

Cloud Solutions has developed several essential services that AWS has to provide to eliminate the need for the developer's on-premises laptop to host their web application. First, Cloud Solutions will leverage CloudFront from AWS to resolve the latency issue and offload the on-premises server. CloudFront reduces latency by caching copies of the web application at edge locations in various parts of the country, allowing users to visit the web application quickly. Also, CloudFront is a managed service that automatically scales to meet high-traffic and low-traffic demands. This allows constant performance without having to scale the infrastructure manually.

Cloud Solutions will also use AWS's Simple Storage Service (S3). Amazon S3 is "an object storage service offering industry-leading scalability, data availability, security, and performance." *Amazon S3*. S3 will give the developers ease of mind by allowing continuous updates in the code of their web application game to grow inside the Amazon S3 bucket. To enable continuous updates to their web application from the developer's GitHub account, Cloud Solutions will suggest leveraging AWS CodePipeline. "AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates." *Amazon CodePipeline*. Implementing this allows the developer to manually update their web application game to the most recent code being hosted in their GitHub account.

Lastly, Cloud Solutions will always recommend the best security practices when hosting a web application. To fulfill this, we recommend creating a Transport Layer Security (TLS) certificate generated from AWS Certificate Manager (ACM) and attaching the certificate to the

web application hosted on CloudFront. This will give users peace of mind when visiting SumCloud.net, knowing that any information going to and from this web application is encrypted.

### **Implementation Plan**

To successfully implement this solution, we will break it into three phases. Our first phase is going to create a blueprint of the cloud infrastructure. Cloud Solutions will start by utilizing Draw.IO to give a high-level blueprint of each resource being used and how they are each connected. This will allow anyone looking at this project to get a general idea of its layout.

The second phase will be to develop the infrastructure using Terraform, a provisioning tool used to create AWS services. Each service will be independently produced as IaC, and the result should be CloudFront, S3, Codepipeline, and ACM working cohesively to host the Sum Cloud web application.

The third step will be to test and validate the project. To accomplish this, we will check to see if each service is running in AWS and hosting the web application. We will ensure that the web application can be visited in different spots of the country by using the command prompt and running the command “nslookup sumcloud.net.” This will provide the IP addresses where CloudFront is being hosted. Next, we will test the CodePipeline by updating the current code in GitHub, running the update through the pipeline, and ensuring that S3 receives and stores the updated code.

Going through each of these phases will ensure that Cloud Solutions will deliver a product to the developer of Sum Cloud.

### **Review of Other Work**

One source that will be utilized when reviewing other people’s work to create this project

will be “AWS Whitepaper for Secure Content Delivery with Amazon CloudFront.” This document will broadly explain how CloudFront, a content delivery network (CDN), securely delivers content over the Internet. It focuses on its ability to provide low latency and secure data transportation to different parts of the country and globally. This article explains the security model between AWS and the customer by detailing how AWS secures the infrastructure of CloudFront and how the customer can implement security in this service.

First, the paper goes over how AWS provides security for CloudFront. Since CloudFront is a global infrastructure, AWS will apply the best security practices and comply with security standards. Not only does AWS implement physical, network, and access controls, but CloudFront also undergoes third-party audits to ensure they stay in different compliance programs. The security that AWS CloudFront provides will give the customer, Sum Cloud, the ease of mind knowing that when users continue to visit their website, CloudFront is providing some of the top security features and policies, whether it is serving HTTPS requests only or allowing the customer to add a Web Application Firewall (WAF) to the front end of their application.

Of course, everyone wants to keep their website up and running with no issues; unfortunately, we don't live in a perfect world. CloudFront provides resilience and availability so customers' web applications can stay up and running. Distributed Denial of Service (DDoS) attacks are common in today's cyber world. DDoS attacks can bring down web applications so users can't visit them. Luckily, CloudFront “can further improve the DDoS resilience of your application by applying Shield Standard DDoS protection.” *AWS Whitepaper*. AWS Shield Standard is applied to every user of CloudFront at no additional cost, which is a further advantage when presenting this to the customer.



Lastly, this article explains the logging and monitoring that CloudFront has to offer. CloudFront integrates with multiple services, such as CloudWatch, WAF, and CloudTrail, for monitoring and logging. The developers of Sum Cloud can leverage this service offered by CloudFront to detect anomalies and incidents and ensure compliance with any security policies they create.

In conclusion, this whitepaper tells anyone working with AWS CloudFront the importance of understanding the security measures AWS Employs and the features available in CloudFront. By leveraging CloudFront's features and incorporating other AWS services, the customer can build robust security and highly available low latency applications globally.

My second Review of other people's work will be an article called "A Simple CI/CD Pipeline with AWS CodePipeline " by Osusara Kammalawatta. This article goes over his step-by-step process in applying AWS CodePipeline for Continuous Integration and Continuous Delivery (CI/CD).

The first part of the article explains why implementing a CI/CD pipeline is one of the best practices a developer can integrate into their project. As we provided earlier in this document, we wanted to add AWS CodePipeline to this project to allow the developer of Sum Cloud to have an agile methodology by applying iterative integration to their web application. The article explains that they will be creating a YAML file that is going to be integrated into their web application. Since we reviewed Sum Clouds files and what programming language was used, we can safely bypass this phase. The developers of Sum Cloud wrote the web application in Hyper Text Markup Language (HTML) with some JavaScript, so a YAML file is not needed.

In step two the author states briefly that a project needs to be pushed to the GitHub repository. Luckily, the developer of Sum Cloud has already completed this step due to his

project being open source for other developers to contribute to their project. The reason why we want the developer's project to be in a GitHub account is that it will allow us to attach a pipeline from that account to their AWS account. This will allow the user to make updates on their local computer to the web application code, and then push that updated code to GitHub. That pipeline will then deliver any updates from the GitHub account to their AWS account where the web application is being hosted.

In the next step, the author of this article provides a resource in AWS that will need to be created in addition to the CodePipeline. This resource is going to be an Amazon S3 bucket. S3 acts as an object storage area where all the files for the web application are going to be placed and hosted. This article explains that when an S3 bucket is going to be working in conjunction with CodePipeline, the permissions tab where the bucket policy is located needs to be updated. The policy the author suggests is going to allow our S3 bucket to be public, allowing it to receive objects from the GitHub account.

The next steps provided by the author are going to go over the Graphical User Interface (GUI) of creating the CodePipeline in the AWS console. This is helpful for Cloud Solutions because when we create our Terraform template for CodePipeline we can make a side-by-side comparison to help with filling out some of the key areas in the IaC.

In conclusion, this article gives us the knowledge on creating a CodePipeline from a GitHub account to AWS. Even though our project is going to be focused on creating the environment as IaC this article gives us the ease as if we were writing the code within the AWS console.

The third article that I will be reviewing is called, "Build S3 Static Website and CloudFront Using Terraform and Gitlab" written by Theara Seng who is a DevOps engineer.

The article is a guide that will walk you through the setup of an S3 static website along with CloudFront using Terraform. The article does mention Gitlab, but this will not be covered in the project.

This article lays down the foundation for the entire project itself. The whole infrastructure will be built using IaC using Terraform as the programming language. One might wonder what IaC is. Well, this article briefly describes it as “manages and providing an infrastructure through code instead of through manual processes” (Seng).

The article starts by giving us directions on how to structure Terraform files. One key point to this is to keep all files located within the same Terraform folder. The five different files that will be created in this folder will be Version, CloudFront, Codepipeline, ACM, and a Variables file.

Next, the article goes over what needs to be typed out in each file starting with variables and ending with CloudFront. Now, like I said earlier, that article goes over Gitlab which is a service that we will not be adding to this project. Instead, we will be adding the CodePipeline. Since no information is available in this article that describes how we would write this out, we will have to visit the CodePipeline Terraform website. This will give us insight into the kind of code that is needed to create this resource.

At the end of this article, it goes over creating an index page that is then added to the S3 bucket that was created in Terraform. For this project, our CodePipeline will be attaching our GitHub and AWS account and uploading those files to our S3 bucket which was explained earlier in this paper.

The final paper that I reviewed is called “Cloud versus On-Premises Computing” written by Cameron Fisher from Massachusetts Institute of Technology (MIT). The reason why I

decided to review this study was because our client was looking to move their web application to the cloud from on-premises. This article can help with determining the pros and cons of both environments and which is best.

Some pros to on-premises computing are control, customization, and security. Control and customization allow organizations to maintain total control over their equipment allowing the user to make significant customization to their software. Next security on-premises allows the user to keep any sensitive data they might have on-site, which will reduce exposure to potential breaches from nefarious actors.

Some cons to on-premises computing are high initial costs, scalability, and maintenance. The article tells us that on-premises systems can require a significant upfront capital expenditure with expensive hardware and setup. Next, scalability is limited unless you're in a company that has large amounts of capital that can be spent on the infrastructure. This can also lead to overprovisioning or underutilization of resources during different parts of the year. Maintenance and upgrades can be burdensome. This might include upgrading software, security patches, and just the hardware itself which would not be ideal for the developer of Sum Cloud.

Some pros of Cloud computing are the flexibility and scalability of being on-demand and customizable. The article describes that resources can be scaled up or down dynamically based on the demand. Next, the cost efficiency is a pay-as-you-go model. This is a staple of cloud computing as it allows users to only pay for the resources that they choose to use and for however long they choose to use it.

Some cons to cloud computing that the article covers include dependency on internet connectivity and data security. Of course, an internet connection will always be needed when accessing a website from a cloud provider, so ensuring they have a stable and available

connection is critical. If connections go dark, this could impact business operations. A business or organization also needs to be able to trust in the cloud provider to keep any data that is being sent in and out to be secure. The article describes cloud providers having robust security measures set in place as data is stored off-site.

The Developers of Sum Cloud want to move their web application game from on-premises to the cloud, and by showing them the details of what this study has to offer we can hopefully persuade them.

### Project Rationale

The reason why this project was chosen was to show how quick, easy, and affordable it is to move a web application from on-premises to the cloud. Every year more and more businesses are moving their applications and data to the cloud and reducing their capital expenditure. What makes this project interesting is that it can easily be written as code. If you are new to writing IaC it can take a couple of days to write and test, if you're experienced this can take you less than a day.

The reason why I am implementing this project is because we have a customer, Sum Cloud, who wants to move their web application from one premise to the cloud. The reason for this is that they are hosting their web application on old and outdated hardware. The developers have told us that they are looking to host their web application in the cloud because it will provide scalability due to an increased amount of traffic to their application and they also wanted to reduce the latency for some users who are in different parts of the country.

From a business perspective, this will be a smart choice to pursue. The developers of Sum Cloud don't have a steady revenue of capital due to the application being free for users to

play. This would make it very difficult for them to start investing in on-premises hardware just so more users can visit their web application. The smarter move would be to eliminate the need for capital expenditure and implement a pay-as-you-go service from AWS.

Again, the rationale for choosing this project is that it is not relatively excessive for the work needed to move the customer from on-premise hardware to the cloud. Also, it allows the developers at Sum Cloud to have peace of mind knowing their web application will be hosted in the cloud with minimum expenditure that is scalable, highly available, and that users can access quickly if needed.

### Current Project Environment

Currently, the developer is working in an on-premises environment where their web application is being hosted on an old laptop in their network server rack. Right now, they don't have the resources needed to expand with the amount of traffic that is going to their web application and are currently looking for a solution.

Sum Cloud is running its web application from a laptop that is acting as a type one hypervisor and does not have the capabilities to scale up or out. Right now, the hypervisor is running a non-graphical user interface of a Linux distribution with HTTPD installed. HTTPD is a software program that acts like a server using HTTP and HTTPS and can be used to host a web application. The laptop is running an older generation processor with a low amount of RAM and one network connection. When the developers at Sum Cloud configured the guest operating system on the hypervisor they were already met with limited resources.

The problem with this current state is that the web application itself is having an influx of users visiting the site, this in turn is adding a large amount of strain in the current hardware that

is being used to host the web application. When reviewing the current setup, we found that the CPU and memory utilization was maxing out to what was set when creating the Linux guest operating system. Also, since the laptop has one ethernet connection that allows users to view the web application for one specific part of the country, it is causing them to either have high latency or have the connection time out making it inaccessible.

Cloud Solutions is going to provide a deliverable to Sum Cloud to move their web application from the current on-premises environment to a cloud-based environment. The services that are going to be used in the AWS cloud environment are going to allow the web application Sum Cloud to have large amounts of users access the site with reduced latency from different parts of the country. An additional advantage to this is since Sum Cloud isn't currently collecting any data from the users, no storage must be transferred. Since the application is constantly being updated, the only connection that is needed is a pipeline from GitHub to AWS S3 to transfer the files from the repository to the bucket storage.

### Methodology

Since this project is going to be delivered as a full upfront solution, the methodology that will be used is Waterfall. Waterfall is a linear project management workflow that breaks down into several different pieces providing a full deliverable upfront to our customer.

The waterfall methodology is going to be broken up into six different pieces. First, we are going to go over the requirements that are going to be needed for this project. This has been evaluated throughout the project documentation where we need a cheap, highly available, and scalable solution to host the web application. Next, we are going to go over the design of the cloud environment which is going to be created in Draw.IO (Appendix A).

Next, Cloud Solutions will be developing the cloud environment as IaC. This will be

created in five different files (Version, Variables, ACM, CloudFront, and CodePipeline). The files will be created using Visual Studio Code (Appendix B – H).

In the testing phase, we will execute the terraform files that will build the cloud infrastructure. We can validate the infrastructure in the AWS console by looking at each resource. (Appendix I – O). After the testing phase, we will deploy the Web Application with the IaC and provide any maintenance until the project is officially closed. During the closing of the project, all deliverables will be transferred over to the developer on Sum Cloud for them to use to build upon.

### **Project Goals, Objectives, and Deliverables**

This project is going to contain four different goals with several objectives to complete a deliverable. The deliverable at the end of this project should be a cloud environment where a user can host their web application that can be highly available and scalable due to the amount of traffic.

The first goal is to view the current hardware that is being utilized to host the web application along with finances and determine what resources to use in AWS. The objective of this goal is to inspect the type of hardware that is hosting the web application. This was determined to be a laptop running a type one hypervisor with a Linux guest operating system. The computer had an outdated CPU with a low amount of memory, bottlenecking users from visiting the web application. During this goal, Cloud Solutions is going to sit down with the developer of Sum Cloud and gather information to determine what components will be used in AWS to fit their budget. The outcome of this goal give insight into what resources are going to be used in the project and the budget the developer of Sum Cloud is willing to spend.

It's been determined that CloudFront, ACM, CodePipeline, and S3 will be used to host



the web application. We decided to go this route because the client doesn't have experience with cloud infrastructure, so we wanted to pick resources that AWS would manage.

The next goal was to build out the infrastructure. We started by creating a high-level blueprint of the cloud environment using the application Draw.IO to give a general overview of the resources. Since we determined the resources that are going to be used in this project while meeting with the client, we were able to pull the Terraform template for each service. We first started with creating the version.tf file. This file declared what version and cloud provider we were going to use when creating the environment. Also, the regions were declared in this file, which were going to be US-West-2 and US-East-1.

The next three files that were created were going to be ACM.tf, CloudFront.tf and CodePipeline.tf files. The ACM.tf file consisted of the actual certificate which was being created in US-East-1 and showed that it was going to be applied to a specific Route53 site, SumCloud.net. The CloudFront.tf file was set up to use the S3 bucket as the origin access to grab the web application files. This file also sets up the redirection of HTTP requests to HTTPS for secure connections. Lastly, this file is only going to allow edge locations to be created in the United States giving users quicker access to the web application from different parts of the country. For the CodePipeline.tf file a pipeline connection was created to connect the client's GitHub account to the AWS account. This file also created several policies that would allow the connection to be complete. The S3 bucket was also created in this file for simplicity.

Our third goal includes our testing phase, where we are going to be running the command "Terraform Apply" in the command prompt in Visual Studio Code. When we run the command, we should receive feedback on any errors found when the program reviews the code. If no errors are found, then we will get a response for all the resources that are going to be created with a

green plus next to them. When we get a response in the command prompt saying “Apply Complete” with the number of resources completed, we can visit the AWS Console and ensure they have been created. The only manual adjustment we need to complete in the AWS console is to authorize the connection from AWS to GitHub (Appendix P – X). At the end of the test, we can visit the website [sumcloud.net](http://sumcloud.net) and verify that it is up and running. We can also go to the command prompt on a PC and type in the command “nslookup sumcloud.net” and see the different CloudFront servers that are hosting the web application. (Appendix N).

The final goal is Post-Deployment and Transfer. At this point, the Terraform project will be handed over to the customer. The client will now have access to the IaC and can either continue to add or destroy the environment as many times as they want without having to manually create the resources individually in the AWS console. Any documentation will be provided to show how to deploy the environment and configure the settings.

The final deliverable to our client is going to be a cloud environment that is created using IaC. The resources used in this infrastructure will allow the web application to be hosted in different parts of the country along with managed scalability by AWS.

	Goal	Supporting objectives	Deliverables enabling the project objectives
1	Determine current state of environment	1.a. Inspect current hardware	1.a.i. Determine that current hardware that is used to host the web application
			1.a.ii. Determine client’s financial status to host the web application
		1.b. Document information.	1.b.i. Document the current environment that is hosting the web application.
			1.b.ii. Document the cloud environment that needs to be created and discuss with the client.
2	Build Infrastructure	2.a. Determine Resources and Document in Blueprint	2.a.i. Finalize the AWS resources that are going to be used in the project and create a high-level blueprint of the environment.

		2. b. Build IaC	2. b.i Create a Terraform environment 2.b.ii Create AWS resource files
3	Test and Monitor	3.a. Test deployment of IaC	3.a.i Check Terraform command line is deploying resources
			3.a.ii Visit AWS Console and review each resource that is created.
			3.a.iii Fix error with CodePipeline to make the connection between GitHub and AWS
		3.b. Ensure the Web App is running	3.b.i Visit SumCloud.net and view the address bar to see if the web application is using HTTPS
		3.c. Review Servers hosting the Web application	3.c.i Open the command prompt on a computer and type command “nslookup sumcloud.net”
			3.c.ii Review the different IP addresses that CloudFront is using to host the web application.
4	Post-Deployment	4.a Transfer deliverable along with and documentation. Close out Project.	4.a.i Transfer the Terraform files (deliverable) to client to host the web application.
			4.a.ii Transfer any documentation associated with the environment. (Draw.IO blueprint).
			4.a.iii Officially close-out the project.

### Project Timeline with Milestones

This project is going to have six different milestones that will be met each day and created in five to six days if no problems arise:

- The first milestone is going to be viewing the current environment in which the application is being hosted and discussing with the client the cloud environment that we plan to move the web application to.
- The second milestone is finishing the selection of resources that are going to be used in AWS and creating a high-level blueprint which should take three to four hours to create.
- The third milestone is to create the Version, ACM, and variables Terraform files.

Resources will be created inside these files and tested. This process should take three to four hours and will be successful if the resources are created in the AWS console with no errors.

- Fourth, we will create the CodePipeline and CloudFront Terraform files along with finishing the variables file. This milestone will be completed when the resources are created in the AWS console and regression testing is completed and stable against the previously created resources.
- The day before handing the client the deliverable we will go over testing to make sure the entire Terraform environment is deployed to AWS and is hosting the website. The milestone will be met when the web application can be visited from the web from multiple electronic devices, the connection is HTTPS, and CloudFront is using multiple servers to host the web application.
- The final milestone, which is the IaC and documentation, are to be handed over to the client for them to utilize at their discretion. A demo of how the Terraform files work along with any questions is answered. This milestone is met when the client signs off on the project.

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
Current/Future Environment Evaluation	One day	September 22, 2024	September 22, 2024
Resource Selection and Blueprint of	3 - 4 hours	September 23, 2024	September 23, 2024

Cloud Environment			
Version, ACM, and Variables Terraform Files Created and Tested	4 - 5 hours	September 24, 2024	September 24, 2024
CloudFront and CodePipeline Created. Variables file completed	1 day	September 25, 2024	September 25, 2024
Test and Make Adjustment	1 day	September 26, 2024	September 26, 2024
Transfer Deliverable	3 hours	September 27, 2024	September 27, 2024

### Outcome

Once the project is completed and transferred over to the client, they can view the telemetry of the CloudFront distribution. The monitoring tab under Telemetry in CloudFront will give our client insight into the number of requests and Error rates occurring within our distribution.

We can use “Requests” to measure the success of our project by reviewing the total number of viewers that have viewed our web application via CloudFront. This will measure any HTTP or HTTPS request. This is measured on a run chart, and we can continually see the number of users visiting our web application. If the request continues to stay elevated, we can assume that our web application is running, automatically scaling, and users can play Sum

Cloud.

Now if “Requests” start to descend on the chart it can mean two things: Users are no longer interested in our web application game and are refusing to visit the site, or we are having either server-side or client-side errors. This metric can be measured in the “Error Rate” chart in the monitoring tab under Telemetry. This chart can give our client insight into users running into client-side or server-side errors.

Three things need to be determined for this project to be successful:

1. The Terraform files deploy a web application that is being hosted in AWS through CloudFront. This can be confirmed by viewing the AWS console and visiting CloudFront, CodePipeline, ACM, and S3 bucket and making sure the resources are created and the site SumCloud.net can be visited.
2. The metrics in our CloudFront distribution can be continually monitored and reviewed to see the number of requests that are coming to our web application.
3. If we continue to have zero error rates, along with continuous requests, we know that users can successfully visit SumCloud.net (Appendix Y).

## References

**Amazon Web Services.** (n.d.). *Amazon S3*. Amazon. Retrieved 09/21/2024, from

<https://aws.amazon.com/s3/>

**Amazon Web Services.** (n.d.). *AWS CodePipeline*. Amazon. Retrieved 09/21/2024, from

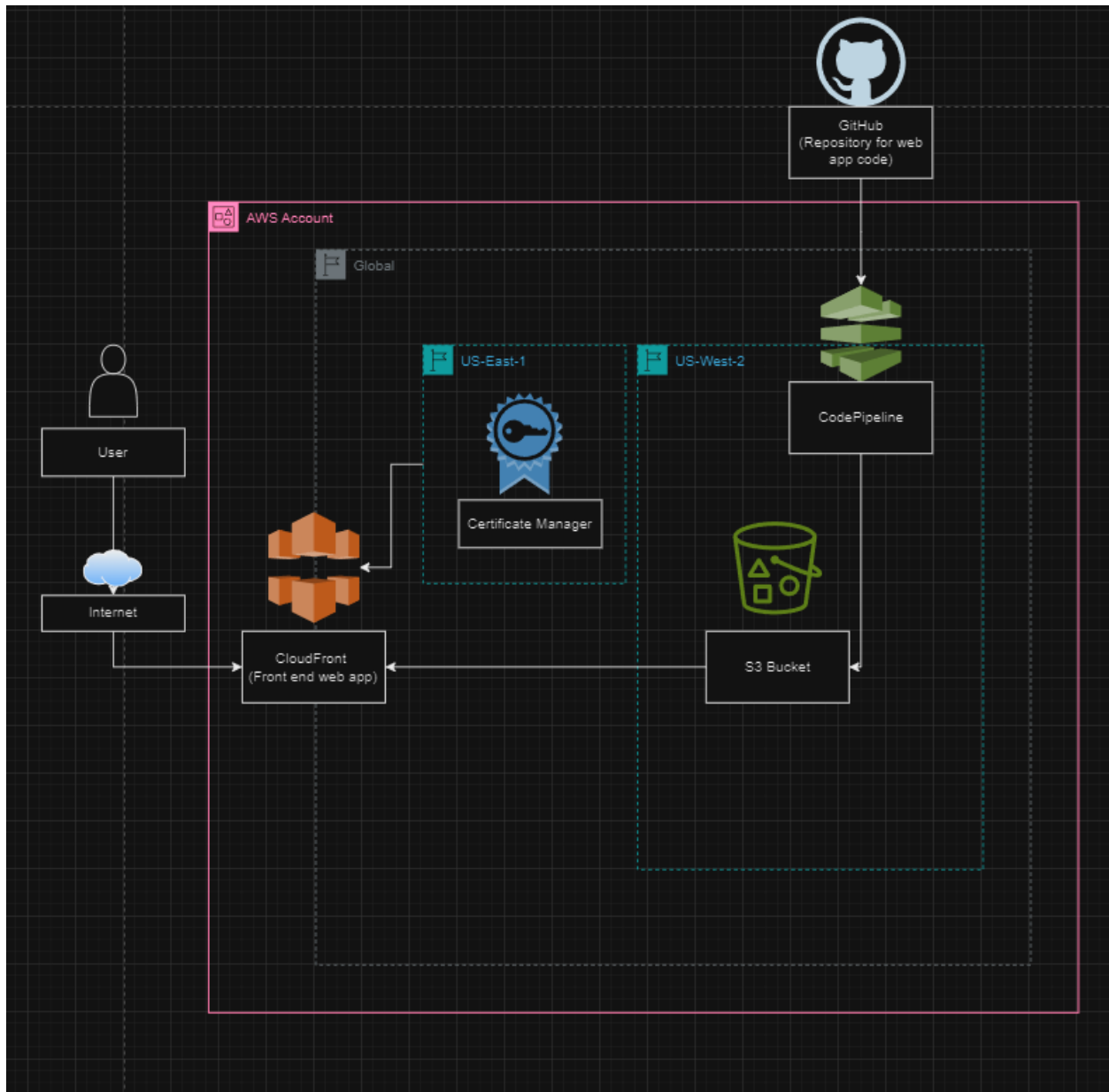
<https://aws.amazon.com/codepipeline/>

**Amazon Web Services.** (2024, April 26). *Secure content delivery with Amazon CloudFront* (AWS Whitepaper). Amazon. <https://docs.aws.amazon.com/pdfs/whitepapers/latest/secure-content-delivery-amazon-cloudfront/secure-content-delivery-amazon-cloudfront.pdf>

**Seng, Theara.** (2022, November 26). *Build S3 static website and CloudFront using Terraform and GitLab*. Medium. <https://medium.com/@thearaseng/build-s3-static-website-and-cloudfront-using-terraform-and-gitlab>

## Appendix A

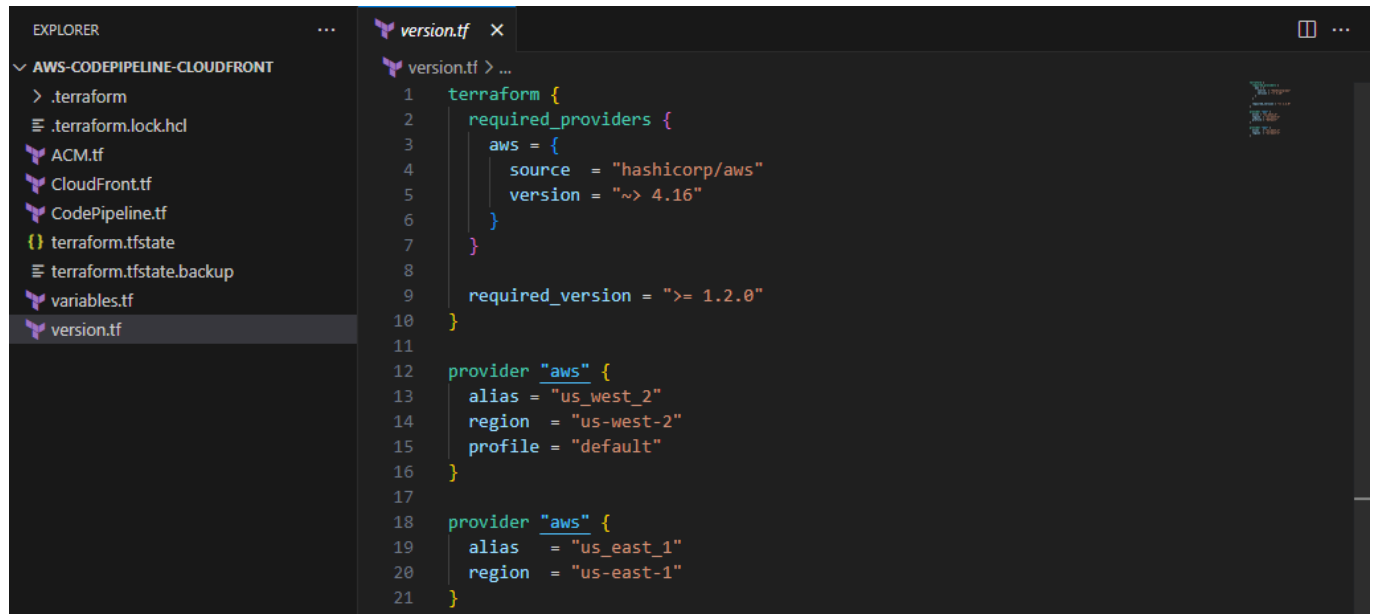
### Design





## Appendix B

## Version Terraform File



```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 4.16"
6     }
7   }
8
9   required_version = ">= 1.2.0"
10 }
11
12 provider "aws" {
13   alias = "us-west-2"
14   region = "us-west-2"
15   profile = "default"
16 }
17
18 provider "aws" {
19   alias = "us-east-1"
20   region = "us-east-1"
21 }
```

## Appendix C

## Variables Terraform File

```
variables.tf > variable "aws_zone_id"
1  variable "aws_domain_name" {
2      description = "The domain name for which the SSL certificate should be crea
3      default = "" #domain name for SSL certificate
4
5  }
6
7  variable "alt_aws_domain_name" {
8      description = "Alternative domain names for the SSL certificate"
9      default = [""] #alternative domain name
10
11  }
12
13
14  variable "domain" {
15      description = "The domain name to use for the Cloudfront distribution"
16      type = string
17      default = "" #domain name for cloudfront distribution
18  }
19
20  variable "alt_domain" {
21      description = "The domain name to use for the Cloudfront distribution"
22      type = string
23      default = "" #alternative domain name for cloudfront distribution
24  }
25
26  variable "aws_zone_id" {
27      description = "The Route 53 zone ID"
28      type = string
29      default = "" # Route 53 zone ID for the domain
30      sensitive = true
31  }
32  }
33
34
```

## Appendix D

## ACM Terraform File

```
ACM.tf > resource "aws_acm_certificate_validation" "certificate_validation"
1  #ACM
2
3  resource "aws_acm_certificate" "acm-certificate" {
4      provider          = aws.us_east_1 # ACM is only available in us-east-1
5      domain_name       = var.aws_domain_name
6      subject_alternative_names = var.alt_aws_domain_name
7      validation_method = "DNS"
8
9
10     lifecycle {
11         create_before_destroy = true
12     }
13 }
14
15 data "aws_route53_zone" "Route53-data" {
16     provider = aws.us_west_2
17     name     = var.aws_domain_name
18     private_zone = false
19 }
20
21 resource "aws_route53_record" "cert-validation" {
22     provider = aws.us_west_2
23     for_each = {
24         for dvo in aws_acm_certificate.acm-certificate.domain_validation_options : dvo.domain_name => {
25             name     = dvo.resource_record_name
26             record   = dvo.resource_record_value
27             type     = dvo.resource_record_type
28         }
29     }
30
31     allow_overwrite = true
32     name            = each.value.name
33     records         = [each.value.record]
34     ttl             = 60
35     type            = each.value.type
36     zone_id         = data.aws_route53_zone.Route53-data.zone_id
37 }
38
39 resource "aws_acm_certificate_validation" "certificate_validation" {
40     provider = aws.us_east_1
41     certificate_arn = aws_acm_certificate.acm-certificate.arn
42     validation_record_fqdns = [for record in aws_route53_record.cert-validation : record.fqdn]
43 }
```

## Appendix E

## CodePipeline Terraform File I

```

CodePipeline.tf > resource "aws_codestarconnections_connection" "git_cloud" > name
1  # Codepipeline
2
3  resource "aws_s3_bucket" "codepipeline_bucket" {
4      bucket = "Capstone-test-bucket-WGU"
5  }
6
7  resource "aws_s3_bucket_policy" "my_bucket_policy" {
8      bucket = aws_s3_bucket.codepipeline_bucket.id
9
10     policy = jsonencode({
11         Version = "2012-10-17"
12         Statement = [
13             {
14                 Effect = "Allow"
15                 Principal = {
16                     AWS = aws_cloudfront_origin_access_identity.oai.iam_arn
17                 }
18                 Action = "s3:GetObject"
19                 Resource = "${aws_s3_bucket.codepipeline_bucket.arn}/*"
20             }
21         ]
22     })
23 }
24
25 resource "aws_s3_bucket_public_access_block" "codepipeline_bucket_pab" {
26     bucket = aws_s3_bucket.codepipeline_bucket.id
27
28     block_public_acls       = true
29     block_public_policy     = true
30     ignore_public_acls     = true
31     restrict_public_buckets = true
32 }
33
34 resource "aws_codestarconnections_connection" "git_cloud" {
35     name           = "Capstone-Pipeline-Demo"
36     provider_type = "GitHub"
37 }
38
39 resource "aws_iam_role" "codepipeline_role" {
40     name = "codepipeline-role"
41     assume_role_policy = jsonencode({
42         Version = "2012-10-17"
43         Statement = [
44             {
45                 Action = "sts:AssumeRole"
46                 Effect = "Allow"
47                 Principal = {
48                     Service = "codepipeline.amazonaws.com"
49                 }
50             }
51         ]
52     })
53 }
54

```

## Appendix F

## CodePipeline Terraform File II

```

55 resource "aws_iam_role_policy" "codepipeline_policy" {
56   role = aws_iam_role.codepipeline_role.id
57
58   policy = jsonencode({
59     Version = "2012-10-17"
60     Statement = [
61       {
62         Effect = "Allow"
63         Action = [
64           "s3:*",
65           "codestar-connections:UseConnection",
66           "codebuild:*",
67           "codepipeline:*",
68           "iam:PassRole"
69         ],
70         Resource = "*"
71       }
72     ]
73   })
74 }
75
76 resource "aws_codepipeline" "codepipeline" {
77   name = "my-capstone-codepipeline"
78   role_arn = aws_iam_role.codepipeline_role.arn
79
80   artifact_store {
81     location = aws_s3_bucket.codepipeline_bucket.bucket
82     type = "S3"
83   }
84
85   stage {
86     name = "Source"
87
88     action {
89       name = "Source"
90       category = "Source"
91       owner = "AWS"
92       provider = "CodeStarSourceConnection"
93       version = "1"
94       output_artifacts = ["source_output"]
95
96       configuration = {
97         ConnectionArn = aws_codestarconnections_connection.git_cloud.arn
98         FullRepositoryId = "JordanSum/Sum-Cloud-V0.10"
99         BranchName = "main"
100       }
101     }
102   }
103
104   stage {
105     name = "Deploy"
106
107     action {
108       name = "S3Deploy"
109       category = "Deploy"
110       owner = "AWS"
111       provider = "S3"
112       input_artifacts = ["source_output"]
113       version = "1"
114
115       configuration = {
116         BucketName = aws_s3_bucket.codepipeline_bucket.bucket
117         Extract = "true"
118       }
119     }
120   }
121 }

```

## Appendix G

## CloudFront Terraform File I

```

CloudFront.tf > resource "aws_cloudfront_distribution" "s3_distribution" > default_cache_behavior
1  # CloudFront
2
3  resource "aws_cloudfront_origin_access_identity" "oai" {
4    comment = "OAI for my S3 bucket"
5  }
6
7  resource "aws_cloudfront_distribution" "s3_distribution" {
8    origin {
9      domain_name      = aws_s3_bucket.codepipeline_bucket.bucket_regional_domain_name
10     origin_id         = aws_s3_bucket.codepipeline_bucket.bucket
11
12     s3_origin_config {
13       origin_access_identity = aws_cloudfront_origin_access_identity.oai.cloudfront_access_identity_path
14     }
15   }
16
17   enabled              = true
18   is_ipv6_enabled      = true
19   comment              = "Capstone Cloudfront Distribution"
20   default_root_object  = "index.html"
21
22   aliases = [var.domain, var.alt_domain]
23
24   # Cache behavior with precedence 0
25   default_cache_behavior {
26     allowed_methods = ["GET", "HEAD"]
27     cached_methods  = ["GET", "HEAD"]
28     target_origin_id = aws_s3_bucket.codepipeline_bucket.bucket
29
30
31     forwarded_values {
32       query_string = false
33
34       cookies {
35         forward = "none"
36       }
37     }
38
39     min_ttl            = 0
40     default_ttl        = 86400
41     max_ttl            = 31536000
42     compress           = true
43     viewer_protocol_policy = "redirect-to-https"
44   }
45
46   restrictions {
47     geo_restriction {
48       restriction_type = "whitelist"
49       locations        = ["US"] # preferred location
50     }
51   }
52
53   tags = {
54     Environment = "development"
55   }
56
57   viewer_certificate {
58     acm_certificate_arn = aws_acm_certificate.acm-certificate.arn
59     ssl_support_method  = "sni-only"
60     minimum_protocol_version = "TLSv1.2_2021"
61   }
62 }

```

## Appendix H

## CloudFront Terraform File II

```
64 resource "aws_route53_record" "cloudfront_alias" {
65     zone_id = var.aws_zone_id
66     name     = var.domain
67     type     = "A"
68
69     alias {
70         name           = aws_cloudfront_distribution.s3_distribution.domain_name
71         zone_id        = aws_cloudfront_distribution.s3_distribution.hosted_zone_id
72         evaluate_target_health = false
73     }
74 }
75
76 resource "aws_route53_record" "cloudfront_alt_alias" {
77     zone_id = var.aws_zone_id
78     name     = var.alt_domain
79     type     = "A"
80
81     alias {
82         name           = aws_cloudfront_distribution.s3_distribution.domain_name
83         zone_id        = aws_cloudfront_distribution.s3_distribution.hosted_zone_id
84         evaluate_target_health = false
85     }
86 }
```

Appendix I

AWS Console ACM

[AWS Certificate Manager](#) > Certificates

Certificates (2)

Delete

Manage expiry events

Import

Request

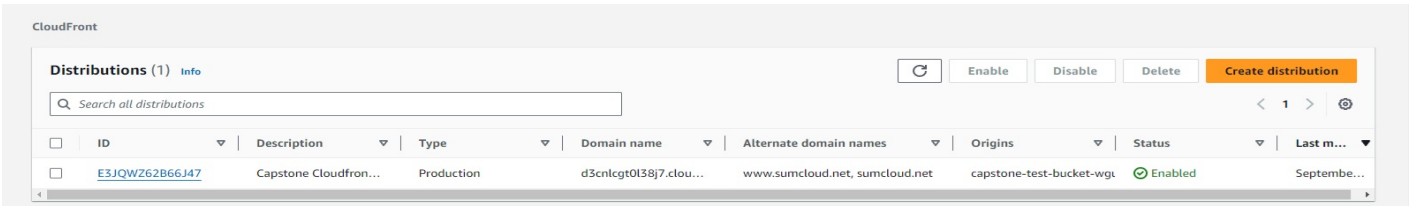
< 1 > ⚙

<input type="checkbox"/>	Certificate ID	Domain name	Type	Status	In use	Renewal eligibility	Key algorithm
<input type="checkbox"/>	<a href="#">2a135b7a-bf93-4961-891b-b3240ada7f20</a>	sumcloud.net	Amazon Issued	Issued	Yes	Eligible	RSA 2048



Appendix J

AWS Console CloudFront






## Appendix K

### AWS Console CodePipeline

Developer Tools > CodePipeline > Pipelines

Pipelines [Info](#) [Refresh](#) [Notify](#) [View history](#) [Release change](#) [Delete pipeline](#) [Create pipeline](#)

	Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
	<a href="#">my-capstone-codepipeline</a> <small>(Type: V1   Execution mode: SUPERSEDED)</small>	 Succeeded	Source - <a href="#">798d9f87</a> <a href="#">View</a> update for capstone purpose	11 minutes ago	 <a href="#">View details</a>

Appendix L

AWS Console S3



## Appendix M

## AWS Console S3 Contents

Amazon S3 > Buckets > capstone-test-bucket-wgu

capstone-test-bucket-wgu [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

**Objects (11)** [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">.github/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">.vscode/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">favicon.ico</a>	ico	September 23, 2024, 13:47:17 (UTC-07:00)	72.2 KB	Standard
<input type="checkbox"/>	<a href="#">index.html</a>	html	September 23, 2024, 13:47:17 (UTC-07:00)	2.6 KB	Standard
<input type="checkbox"/>	<a href="#">landmark.jpg</a>	jpg	September 23, 2024, 13:47:17 (UTC-07:00)	916.8 KB	Standard
<input type="checkbox"/>	<a href="#">my-capstone-codepipe/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">script.js</a>	js	September 23, 2024, 13:47:17 (UTC-07:00)	5.7 KB	Standard
<input type="checkbox"/>	<a href="#">script2.js</a>	js	September 23, 2024, 13:47:17 (UTC-07:00)	1.6 KB	Standard
<input type="checkbox"/>	<a href="#">site.css</a>	css	September 23, 2024, 13:47:17 (UTC-07:00)	341.0 B	Standard
<input type="checkbox"/>	<a href="#">style.css</a>	css	September 23, 2024, 13:47:17 (UTC-07:00)	7.1 KB	Standard
<input type="checkbox"/>	<a href="#">winner.html</a>	html	September 23, 2024, 13:47:17 (UTC-07:00)	343.0 B	Standard

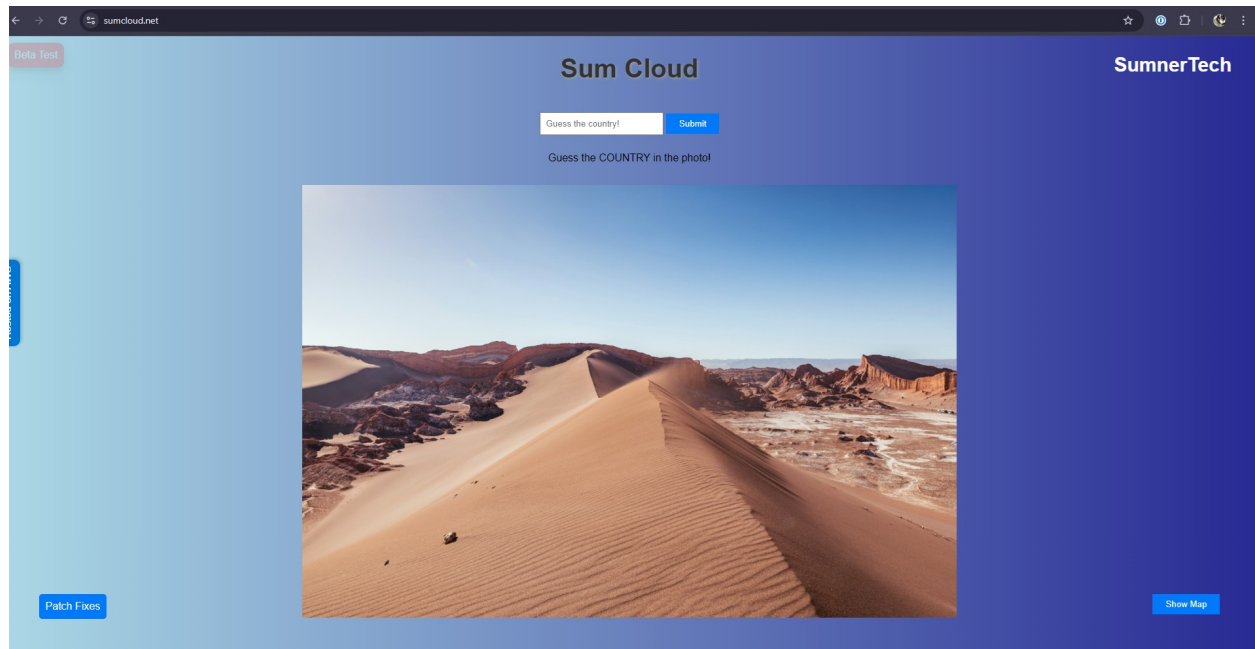
## Appendix N

## CloudFront Servers

```
Non-authoritative answer:  
Name:      sumcloud.net  
Addresses: 18.239.199.90  
           18.239.199.84  
           18.239.199.33  
           18.239.199.98
```

## Appendix O

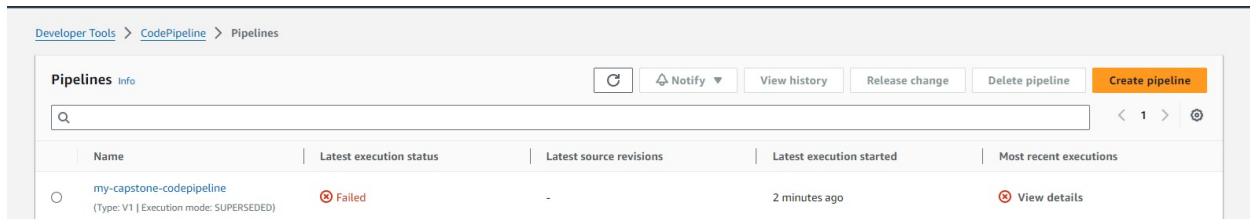
### Web Application



## Appendix P

## CodePipeline Error Fix I

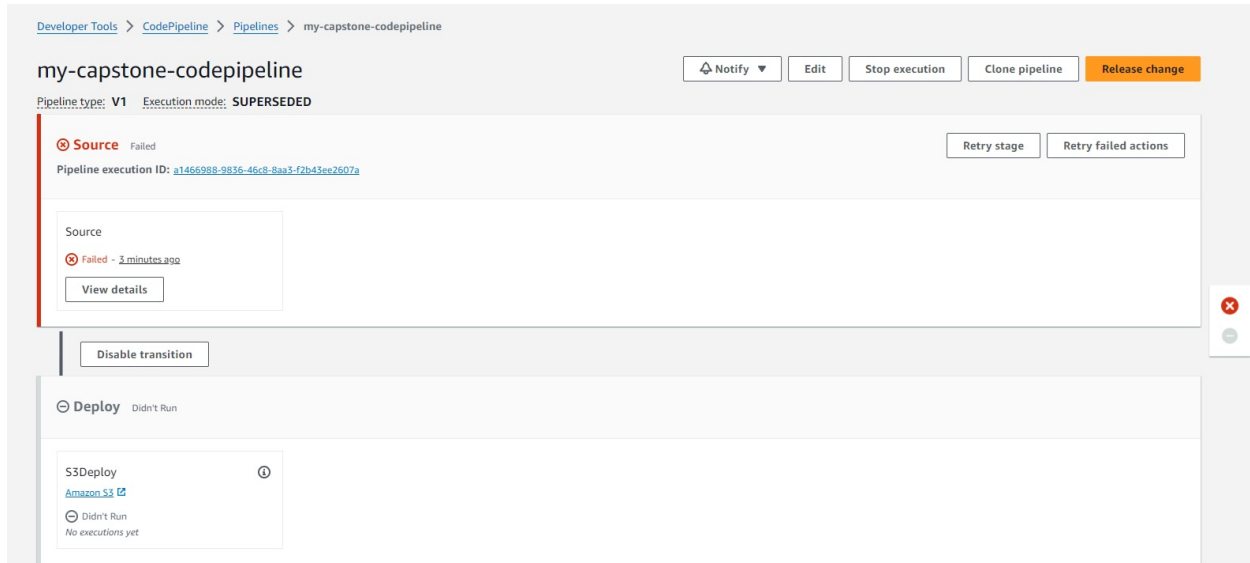
When deploying the IaC, CodePipeline is not going to make the connection from GitHub to AWS, this must be done manually.



## Appendix Q

### CodePipeline Error Fix II

When viewing the source connection (GitHub) it will show failed. At this point client needs to navigate to connections tab in CodePipeline to fix this.





## Appendix R

### CodePipeline Error Fix III

Connection is going to show as pending. Click on the connection name to complete the setup.

Connections <a href="#">info</a>			
<div> <input type="text"/> <span>&lt; 1 &gt; ⌵</span> </div>			
Connection name	Provider	Status	ARN
<input type="radio"/> <a href="#">Capstone-Pipeline-Demo</a>	GitHub	<span>⌵ Pending</span>	arn:aws:codestar-connections:us-west-2:975955791414:connection/30d02d30-7d3b-477b-b590-ea69b64d4a27


## Appendix S

### CodePipeline Error Fix IV


User is going to click on “Update pending connection.”

Developer Tools > Connections > 30d02d30-7d3b-477b-b590-ea69b64d4a27

#### Capstone-Pipeline-Demo


**Finish creating your connection**  
 Your connection status is Pending. Choose Update pending connection.
 

Update pending connection

Connection settings			
Name	Provider	Status	Arn
Capstone-Pipeline-Demo	GitHub	 Pending	arn:aws:codestar-connections:us-west-2:975955791414:connection/30d02d30-7d3b-477b-b590-ea69b64d4a27

## Appendix T

## CodePipeline Error Fix V

User is going to install the AWS application to their GitHub account.

---

[Developer Tools](#) > [Connections](#) > Create connection

## Connect to GitHub

**GitHub connection settings** [Info](#)

Connection name

**App installation - optional**  
Install GitHub App to connect as a bot. Alternatively, leave it blank to connect as a GitHub user, which can be used in AWS CodeBuild projects.

or **Install a new app**

[Connect](#)

## Appendix U

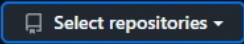
## CodePipeline Error Fix VI

Select the repository you want to pull the source code from.



**Repository access**



☐ **All repositories**  
This applies to all current *and* future repositories owned by the resource owner.  
Also includes public repositories (read-only).

☒ **Only select repositories**  
Select at least one repository.  
Also includes public repositories (read-only).

 Select repositories ▾

Selected 2 repositories.

 JordanSum/Sum-Cloud	×
 JordanSum/Sum-Tech	×

 Save  Cancel

## Appendix V

## CodePipeline Error Fix VII

Click “Connect”

[Developer Tools](#) > [Connections](#) > [Create connection](#)

**Information** Beginning July 1, 2024, the console will create connections with codeconnections in the resource ARN. Resources with both service prefixes will continue to display in the console. [Learn more](#)

## Connect to GitHub

### GitHub connection settings [Info](#)

Connection name

Capstone-Pipeline-Demo

**App installation - optional**  
Install GitHub App to connect as a bot. Alternatively, leave it blank to connect as a GitHub user, which can be used in AWS CodeBuild projects.

or


## Appendix W

### CodePipeline Error Fix VIII

User is going to see the “Status” change from “Pending” to “Available”

[Developer Tools](#) > [Connections](#) > 30d02d30-7d3b-477b-b590-ea69b64d4a27

Capstone-Pipeline-Demo

Connection settings			
Name	Provider	Status	Arn
Capstone-Pipeline-Demo	GitHub	<div>  Available                 </div>	arn:aws:codestar-connections:us-west-2:975955791414:connection/30d02d30-7d3b-477b-b590-ea69b64d4a27

## Appendix X

### CodePipeline Error Fix IX

The user can run the stage that failed again and the pipeline should work.

The screenshot displays the AWS CodePipeline console interface for a pipeline named 'my-capstone-codepipeline'. The pipeline type is 'V1' and the execution mode is 'SUPERSEDED'. The pipeline execution ID is 'a1466988-9836-46c8-8aa3-f2b43ec2607a'. The pipeline consists of two stages: 'Source' and 'Deploy', both of which are marked as 'Succeeded'.

**Source Stage:** The 'Source' stage is shown with a 'Succeeded' status. It includes a 'View details' button and a 'Disable transition' button. The stage's execution ID is '7288d9f87'. The stage's description is 'Source: update for capstone purpose'.

**Deploy Stage:** The 'Deploy' stage is shown with a 'Succeeded' status. It includes a 'View details' button and a 'Disable transition' button. The stage's execution ID is '7288d9f87'. The stage's description is 'Source: update for capstone purpose'.

At the top right of the console, there are buttons for 'Notify', 'Edit', 'Stop execution', 'Clone pipeline', and 'Release change'.

Appendix Y

Success Metrics

