

Post Sum Cloud Application Implementation

Jordan L. Sumner

Western Governors University

Table of Contents

Summary	5
Review of Other Work	6
Changes to the Project Environment	9
Methodology	10
Project Goals and Objectives	12
Project Timeline	13
Unanticipated Requirements	15
Conclusions	16
Project Deliverables	17
References	19
Appendix A	20
Design Phase from Waterfall Methodology	20
Appendix B	21
Telemetry of request to Sum Cloud and error rate in AWS	21
Appendix C	22
Locations of users visiting Sum Cloud Web Application	22
Appendix D	23
Version.tf file	23
Appendix E	24
Variables.tf file	24
Appendix F	25

ACM Resource, IaC.....	25
Appendix G.....	26
WAF for CloudFront, IaC.....	26
Appendix H.....	27
CloudFront, IAC	27
Appendix I	28
Route53 in CloudFront, IaC	28
Appendix J	29
CodePipeline, IaC	29
Appendix K.....	30
CodePipeline, IaC II	30
Appendix L	31
CloudFront AWS Console.....	31
Appendix M.....	32
S3 AWS Console	32
Appendix N.....	33
CodePipeline AWS Console.....	33
Appendix O.....	34
ACM AWS Console	34
Appendix P	35
WAF AWS Console	35

Appendix Q.....	36
Nslookup sumcloud.net	36
Appendix R.....	37
HTTPS website of sumcloud.net	37

Summary

Cloud Solutions, a third-party company that helps clients move web applications to cloud environments, was asked by a client, the developers of Sum Cloud, to move their web application game from an on-premises environment to Amazon Web Services (AWS). Our client at Sum Cloud has been hosting the web application game on an old laptop running a type one hypervisor with a Linux guest operating system for years. This piece of hardware had insufficient memory, CPU, and network capabilities to keep up with the increased demand of users visiting the web application.

Cloud Solutions visited with the developer of Sum Cloud and started examining their current environment to gather details on how they would proceed with a cloud solution. After discussing possible solutions and financial capabilities with the client, it was determined that moving to the AWS cloud was the perfect solution.

After meeting with the client, Cloud Solutions started developing a high-level blueprint of the cloud environment that would be created. They started finalizing the resources that would be used in the AWS environment. Cloud Solutions decided to implement AWS CloudFront, AWS Certificate Manager (ACM), Simple Storage Solution (S3), and CodePipeline.

Cloud Solutions believes writing the environment as Infrastructure as Code (IaC) would be best. This would be the best solution since it is a deliverable that can be handed over to our clients for use in their AWS accounts. Also, if Sum Cloud wanted to destroy and recreate the cloud environment, it could do so almost instantly. If Sum Cloud decided to expand and create multiple web applications, they could use this blueprint to deploy those new services.

The different AWS resources were created in Terraform, a cloud procurement programming tool for creating AWS, Azure, and Google Cloud environments. These resources

were developed and adjusted throughout the project to meet our client's needs, with slight adjustments to security discovered later, which will be explained in this document. Tests were completed to show that each resource was created successfully and that each resource depended on the other AWS resources to host the web application.

After each of these resources was created, configured, and tested, Cloud Solutions believed that they had developed a deliverable that would be sufficient to deliver to the developers of Sum Cloud. Before Cloud Solutions handed over the deliverable, they reviewed the Terraform files containing the resources in AWS and explained what each of them did. It also discussed the basic operations of the Terraform program and how to deploy, destroy, and make edits to the files for future uses. When the client had no further questions, the deliverable was handed off, and the project was signed off.

Review of Other Work

Four previous resources were reviewed to start implementing this project. The first previous resource was *Secure Content Delivery with Amazon CloudFront*. The Whitepaper discusses how to secure content delivery when using Amazon CloudFront. The document covers the security measures users can implement to protect their web applications from malicious actors. Some other topics discussed were HTTPS delivery, protecting content with geo-restrictions, and mitigating threats like Distributed Denial of Service attacks (DDoS). This document gave Cloud Solutions a general idea of how they would like to create CloudFront in Terraform and what features they wanted to implement in CloudFront.

The second previously reviewed article was *A Simple CI/CD Pipeline with AWS CodePipeline* by Osusara Kammalawatta. This article describes the step-by-step process of applying AWS CodePipeline for Continuous Integration and Continuous Delivery (CI/CD). This

article gave ideas on how CodePipeline works with continuous integration from a local computer in a repository in GitHub. It then explains its advantages when connecting it to a cloud provider to deliver the most updated code to a web application.

The third previously reviewed article, *Build S3 Static Website and CloudFront Using Terraform and GitLab*, was written by Theara Seng, a DevOps Engineer. This article laid down the foundation for the project. This article walked you through the setup of an S3 Static website along with CloudFront using Terraform.

The final previously reviewed article was *Cloud Versus On-Premises Computing*, by Cameron Fisher from the Massachusetts Institute of Technology (MIT). This article went into detail on the pros and cons of hosting a web application in an on-premises environment and the pros and cons of hosting a web application in the cloud.

New resources reviewed to complete this project included the article *Creating CloudFront Distribution in AWS using Terraform*, written by Sharma, Arnav. This article gave Cloud Solution several examples of how Terraform could create different types of resources in AWS. One of the services that the author showed was how to create a CloudFront distribution. We could use the example that the author has to provide as a reference for how Cloud Solutions would lay out the Terraform code to create CloudFront. This document also gave insight into creating an S3 bucket in Terraform and creating the dependency between the two resources for CloudFront to grab any files from the bucket to host and view web applications.

The second article that was reviewed was an AWS Whitepaper called *Guidelines for Implementing AWS WAF*. This document goes over the purpose of a web application firewall (WAF) and how it can reduce the impact on a creator's web application, such as DDoS attacks, web application attacks, and bad bots. Cloud Solutions leveraged this Whitepaper because a

change to our project environment was recognized, and a WAF was needed to continue high availability and enhanced security for our client's web applications. The article briefly describes WAF's purpose when handling DDoS attacks:

For HTTP floods, you can use AWS WAF rate limiting rules to block clients from specific IP addresses that are sending an abusive number of requests to your application.

AWS WAF also provides the ability to block known malicious IP addresses using the Amazon IP reputation list from the AWS Managed Rules or by subscribing to AWS Partner IP reputation lists from the AWS Marketplace. (p. 5)

This article gave us knowledge on the purpose of Amazon WAF, and we concluded that it would be highly recommended to add it to our cloud environment that was being created for Sum Cloud.

Our third reviewed source is *20 Terraform Best Practices to Improve your TF Workflow*, written by Flavius Dinu and Ioannis Moustakis. This article goes over key practices when using Terraform. We leveraged this article to show us best practices for structuring our Terraform Project. The source states:

If we are dealing with a small project with limited infrastructure components, it's not a bad idea to keep our Terraform configuration as simple as possible. In these cases, we can configure only the necessary files for our root module, which are the configuration files that exist in the root directory. A small project can contain just these files main.tf, variables.tf, README.md. Some other files that you might find handy to use are the outputs.tf to define the output values of your project, versions.tf to gather any pinned versions for the configurations, and providers.tf to configure options related to the providers you use, especially if there are multiple. (para. 16)

Taking the information from this article gave us better insight and best practices when creating small projects and showed us how our files should be structured for ease and future use for Cloud Solutions and our clients.

Changes to the Project Environment

As stated in our “Review of Other Work” section, it was brought to Cloud Solutions' attention that adding a WAF to our client's Cloud infrastructure would be necessary. Before transferring our solution to our client, we reviewed our findings for Sum Cloud's current on-premises environment. Further, we discussed whether they had any additional security that might have been missed. It was discovered that no additional software or hardware was being leveraged on-premises to help with web application security.

Cloud Solutions discussed with their client the possibility of adding a WAF to their cloud environment. We brought to their attention that this could reduce the number of malicious actors trying to access the web application, which in turn could create a better experience for their user base accessing the web application. We also discussed growing threats when it came to hosting a web application on the internet and how a WAF would mitigate or eliminate a wide range of threats.

After discussing the monthly benefits and expenditures with Cloud Solutions, they agreed that adding a WAF to their cloud environment would be best. After evaluation in a change control process, it was determined that the project scope would stay the same. Minimal time and no additional costs would be added to the project to include this with our CloudFront resource.

Changes to our current environment included an AWS resource that needed to be added to our main.tf file in Visual Studio. We obtained a template from the Terraform website of a WAF and placed it on our main page.tf file. One of the benefits of running a Terraform IaC is

that even if you have resources currently running with a web application, you can continue adding additional resources to your AWS environment without having to destroy and restart the process. The AWS WAF was created in the main.tf file, and that resource was attached to our CloudFront resource in the AWS console. This WAF was now placed in front of our web application, scanning traffic coming into our environment and looking for potential malicious actors.

With the addition of a WAF along with CloudFront, ACM, CodePipeline, and S3, we then moved the Sum Cloud web application from the on-premises environment to the cloud environment. Since the developers and community members of Sum Cloud had already been uploading changes to their source code in GitHub this was an easy transition from on-premises to the cloud. With CodePipeline built into the AWS infrastructure we gave authorization for CodePipeline to make a connection between S3 and GitHub. When this connection was approved, S3 pulled the source code for GitHub and uploaded it to S3, and then CloudFront presented the source code as a web application in different parts of the country. When a user typed in sumcloud.net, they were presented with the web application now being hosted in AWS instead of our client's on-premises hardware.

Methodology

Since this project was delivered as a complete upfront solution, the methodology that was used is Waterfall. Waterfall is a linear project management workflow that breaks down into several pieces, providing a complete upfront deliverable to our customers.

The waterfall methodology was broken up into six different pieces. First, we went over the requirements for this project. During the implementation, we gathered the requirements by looking at Sum Cloud's current setup and reviewed the hardware, software, and operating system

that was currently in use. We then discussed with Sum Cloud what feedback they received from their users visiting their web application and the issues they were running into. After gathering the information, Cloud Solutions was able to determine what resources were needed to create a highly available, scalable, and secure with a CI/CD pipeline. The resources that were determined to be used to fulfill these requirements were ACM, CodePipeline, and CloudFront. A WAF was determined later in the project and added for enhanced security.

Next, we created a high-level design overview of the AWS cloud infrastructure that we believed was the best approach to running a successful web application in the cloud. We used DrawIO, free software that enabled us to map out the AWS account along with the regions, what resources were created, and where they were placed. (Appendix A)

The next phase completed was development. Cloud Solutions created the environment as IaC. Cloud Solutions created three files (Version, Variables, and Main). The main.tf file contained all the resources deployed into AWS to host the web application. The variables.tf file consisted of the AWS Domain Name variable and Alternative AWS Domain Name for ACM, as well as a separate Domain and Alternative Domain for the CloudFront distribution. This file also contained the Route53 Zone ID variable for SumCloud.net. The version.tf file listed the Terraform version and the AWS regions used to deploy our resources.

In the building phase, we built and tested the Terraform files and compared them to those created in AWS. When we executed the Terraform files, several errors occurred with resources improperly configured and not depending on each other. After we correctly configured the resources, we moved into the closing phase. During the project's closing, all deliverables were transferred over to the developer of Sum Cloud for them to use to host their web application or to further improve on.

Project Goals and Objectives

This project consists of 4 goals that need to be accomplished to deliver a successful product to our client:

- The first goal was determining the current hardware to host the Sum Cloud web application. This goal was met at the beginning stages of our project by collecting data on the laptop components and software used for external users to see the website. This goal was briefly revisited due to additional security practices that we believe would benefit our customers, and we were able to implement that feature into the deliverable.
- Our second goal was to build the IaC in Terraform, which will use AWS as the cloud provider. This goal was met by creating several different files in our Terraform project. The Version.tf file consisted of the programming language, hashicorp, the version that would run when deploying the resources, and the region location to which each resource would be deployed. The Main.tf file was created to store all the AWS resources that would be used to host the web application. This file comprised CloudFront, ACM, CodePipeline, S3, and WAF.
- The third goal was to test and monitor the resources created from the Terraform environment. This was accomplished by building out the infrastructure. When each resource was completed, the command “terraform plan” was executed to check if the resources would work with AWS. The following command was executed: “terraform apply.” This command would start creating the resources in an AWS account. Some manual configurations in the AWS console needed to be completed, like providing authorization for our GitHub account to connect to our AWS account because the Terraform code did not yet offer the capabilities at the time of this writing.

- The fourth goal was to transfer the Terraform file over to our client so they could deploy their web application. This goal was accomplished, and our clients could open the file on their computer and fill in the variables for the domain, alternative domain, and AWS Zone ID for their personal Route53 configuration. This file was then executed, CodePipeline gained authorization from their own GitHub account, and the files that contained the code for the web application Sum Cloud were transferred to the S3 bucket. The web application was accessible when visiting sumcloud.net.

Project Timeline

The previously estimated timeline involved six milestones that would be met each day and created in five to six days if no problems arose. The original plan was to complete the current and future environment in one day, which stayed the same with no changes. The resource selection and blueprint milestone were combined with the start of the creation of Version and ACM Terraform Files. This milestone was supposed to be two days and was condensed into one day. The Variables file was created during the CloudFront and CodePipeline, along with testing and adjustment of settings. This milestone was condensed from two days of work into one.

We reduced the project timeline by integrating Microsoft Copilots into Visual Studio Code, where our Terraform files were being created. Whenever an error appeared in the Terraform files or the developers didn't know how to fill out a specific resource, we could type to our Copilot, who would then review our code and suggest what needed to be added. Another advantage was that when we asked for recommendations and fixes to our errors, Copilot would scan our current working file and reproduce the infrastructure as code with the adjustments. It was as easy as copying and pasting the updated code into our environment. Due to the

adjustments and introduction of Copilot into our environment, we reduced the project timeline from six to four days, allowing Cloud Solutions to deliver the project ahead of schedule.

Milestone or deliverable	Planned Duration (Hours or Days)	Actual Duration (hours or days)	Projected Start Date	Projected End Date	Actual Start Date	Actual End Date
Current/Future Environment Evaluation	One day	One Day	09/22/2024	09/22/2024	9/22/2024	09/22/2024
Resource Selection, Blueprint	3 – 4 Hours	One Day	09/23/2024	09/23/2024	09/23/2024	09/23/2024
Resource Creation	Two days	One Day	09/24/2024	09/25/2024	09/24/2024	09/24/2024
Testing	One Day	One Day	09/26/2024	09/26/2024	09/25/2024	09/25/2024
Transfer of Deliverable	3 Hours	3 Hours	09/27/2024	09/27/2024	09/26/2024	09/26/2024

Unanticipated Requirements

A component that was not anticipated during the initiation of this project and that ended up presenting itself during the development of the cloud environment was Identity Access Management (IAM). This added some complexity to our project because the IaC at the time was being created with the account of Cloud Solutions rather than Sum Cloud. This, of course, needed to be communicated with Cloud Solutions, and additional policy resources were added to several different components of AWS resources. This service in AWS helps administrators determine who can access AWS resources and what they can do with them. An administrator can create AWS users and groups and then set permissions for who and what is allowed or denied access to a resource.

Several essential AWS resources would be used to add policy in our IaC. These policies could range from an S3 bucket granting origin access from CloudFront to “Get Object” to getting permission to create the resources in a user’s account. Cloud Solutions always recommends using best security practices by implementing the least privilege to users in an AWS environment. Since this project is being created for one user, we recommend that our client from Sum Cloud set up a root account in AWS and then create their account separately from the root. Currently, they can use the Administrator Access policy attached to their account. This would allow them to execute the resources in our Terraform files without it kicking back an error saying that they do not have sufficient permissions.

This unanticipated requirement was brought to the project manager's attention and was analyzed and evaluated to determine any impact on our project. After careful evaluation, it was determined that the project's scope slightly increased. However, the time required to complete

this project did not adjust, and the change was documented and communicated to the project team.

Conclusions

The results from this project presented us with a working web application hosted with AWS resources. When executing the files in Terraform with the command “Terraform Apply,” we created a CloudFront distribution, a WAF attached to the CloudFront, an Amazon S3 bucket hosting our files from GitHub, a CodePipeline connecting our GitHub account to our AWS S3 bucket to transfer objects, and an ACM to attach to our web application to allow HTTPS requests.

When these resources were created in AWS, we immediately started to see a positive impact of users beginning to visit our web application with no bottlenecking effect like the on-premises environment was causing. If we visited our CloudFront distribution, looked under “Telemetry,” and clicked on the “Monitoring” tab, we could see the number of requests, data transfers, and error rates occurring in near real-time for the web application. Additionally, we could look under the “Reports and Analytics.” In the “Viewers” tab, we can see what devices are being used to view our web application, the browsers, operating systems, and the location from where the user is located. (Appendix B, C)

The future of this project can go in many different directions. If the developers of Sum Cloud wanted to expand the CloudFront distribution to various parts of the globe, they could add other countries to the accepted list in our CloudFront distribution. Also, this deliverable transferred to Sum Cloud could be used to add additional resources to adapt to changes in their web application. If Sum Cloud wanted to add a database to the back end of the web application to start tracking the scores of individual users, then from a high-level viewpoint, they can begin

pulling Terraform templates for Elastic Cloud Compute (EC2) and SQL databases and start hosting their web application through those services. If used appropriately, this template can become very modular for whoever wants to expand their web applications in different ways.

Project Deliverables

This project had several components that comprised one primary deliverable for Sum Cloud. This deliverable was a cloud-based infrastructure that allowed Sum Cloud to host their web application in a scalable, highly available AWS environment, CI/CD, and secure. Sum Cloud could take this Terraform file, execute it, and have a working application within minutes. Listed here are the key components that made up this deliverable.

- CloudFront: AWS manages This content delivery network (CDN). This means that AWS manages the service themselves, giving application availability to different parts of a region and parts of the world. The CloudFront distribution also allows for scalability, which is managed independently depending on the traffic being visited. (Appendix H, I, L)
- WAF: Our WAF component will cover the security side of our web application. The WAF provides security at the application layer in our cloud environment, detecting and preventing malicious attacks such as SQL injection and cross-site scripting, and assists with DDoS attacks by limiting the number of requests that can be made from a single source. (Appendix G and P)
- S3: The S3 bucket is our web application storage component, holding the HTML code and JavaScript information. These files are stored in our bucket, and given the right policy and permissions, our CloudFront distribution can get the source code and

use it to display our web application from different parts of the region. (Appendix J and M)

- CodePipeline: Our CodePipeline component connects us to our GitHub account, where a central repository is created for users to update the code for our web application and AWS S3 account. This connection allows transferring updated files in near real-time or schedule to provide users with a revised application. (Appendix J, K, and N)
- ACM: Our ACM component gives our web application an additional security feature, turning our HTTP web application into an HTTPS. This allows any data traveling to and from our web application to be encrypted, giving our users peace of mind that no information is being transmitted in plain text. (Appendix F, O, and R)

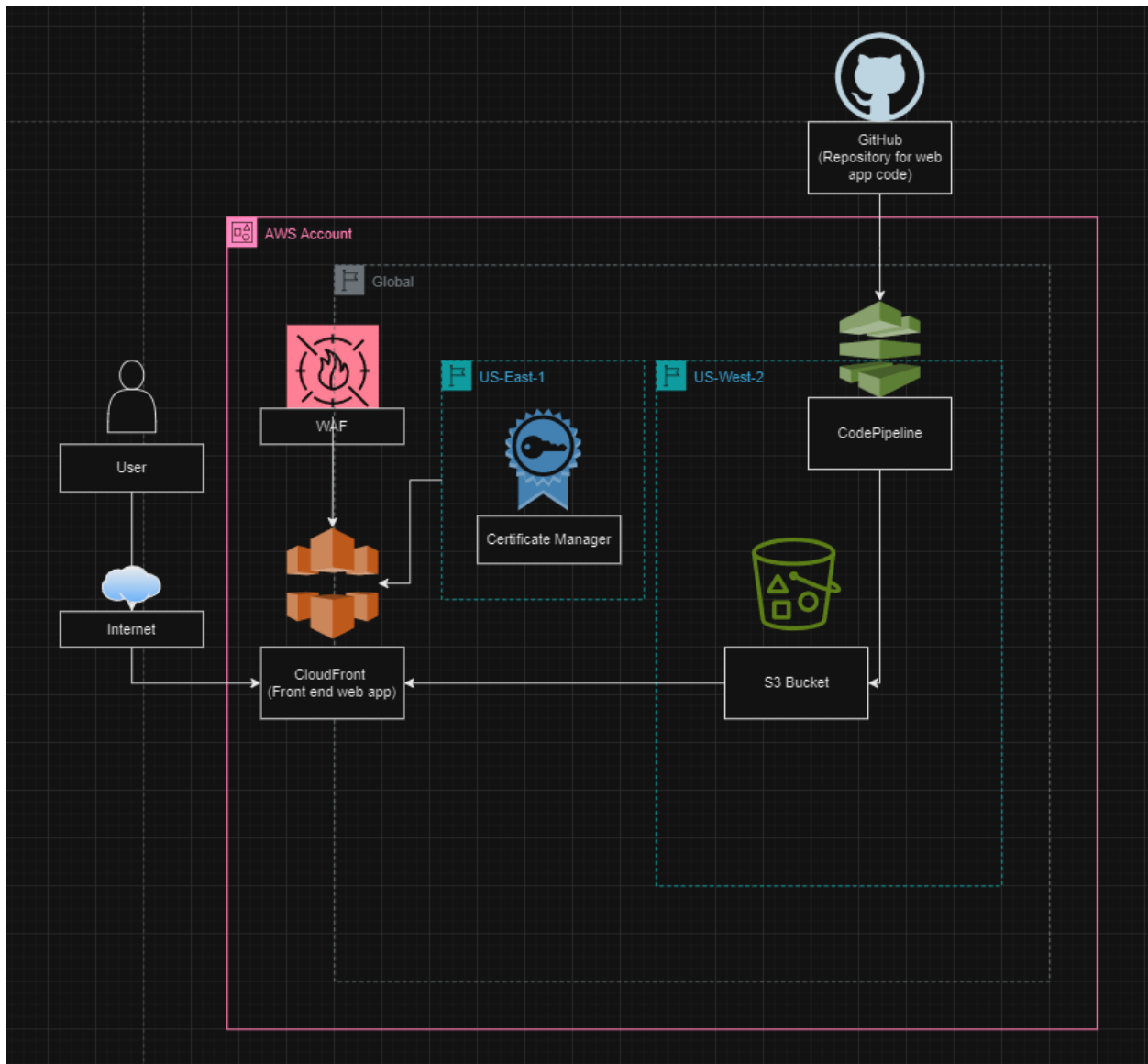
These five components give us a deliverable of a working cloud environment to host a stable web application. The Terraform file can also be modular, allowing our client to update resources and easily create and destroy additional resources.

References

- Amazon Web Services. (2024). *Guidelines for implementing AWS WAF*. AWS Whitepaper. Retrieved from <https://docs.aws.amazon.com/pdfs/whitepapers/latest/guidelines-for-implementing-aws-waf/guidelines-for-implementing-aws-waf.pdf>
- Amazon Web Services. (2024, April 26). *Secure content delivery with Amazon CloudFront*. AWS Whitepaper. <https://docs.aws.amazon.com/pdfs/whitepapers/latest/secure-content-delivery-amazon-cloudfront/secure-content-delivery-amazon-cloudfront.pdf>
- Dinu, F., & Moustakis, I. (2024, September 23). *20 Terraform best practices to improve your TF workflow*. Spacelift. <https://spacelift.io/blog/terraform-best-practices>
- Fisher, C. (2018). *Cloud versus on-premise computing*. American Journal of Industrial and Business Management, 8, 1991-2006. doi: [10.4236/ajibm.2018.89133](https://doi.org/10.4236/ajibm.2018.89133)
- Kammalawatta, O. (2020, July 9). *A simple CI/CD pipeline with AWS CodePipeline*. Medium. <https://medium.com/@osusara/a-simple-ci-cd-pipeline-with-aws-codepipeline>
- 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>
- Sharma, A. (2020, June 13). *Creating CloudFront distribution in AWS using Terraform*. Medium. <https://medium.com/@arnav.sharma/creating-cloudfront-distribution-in-aws-using-terraform>

Appendix A

Design Phase from Waterfall Methodology



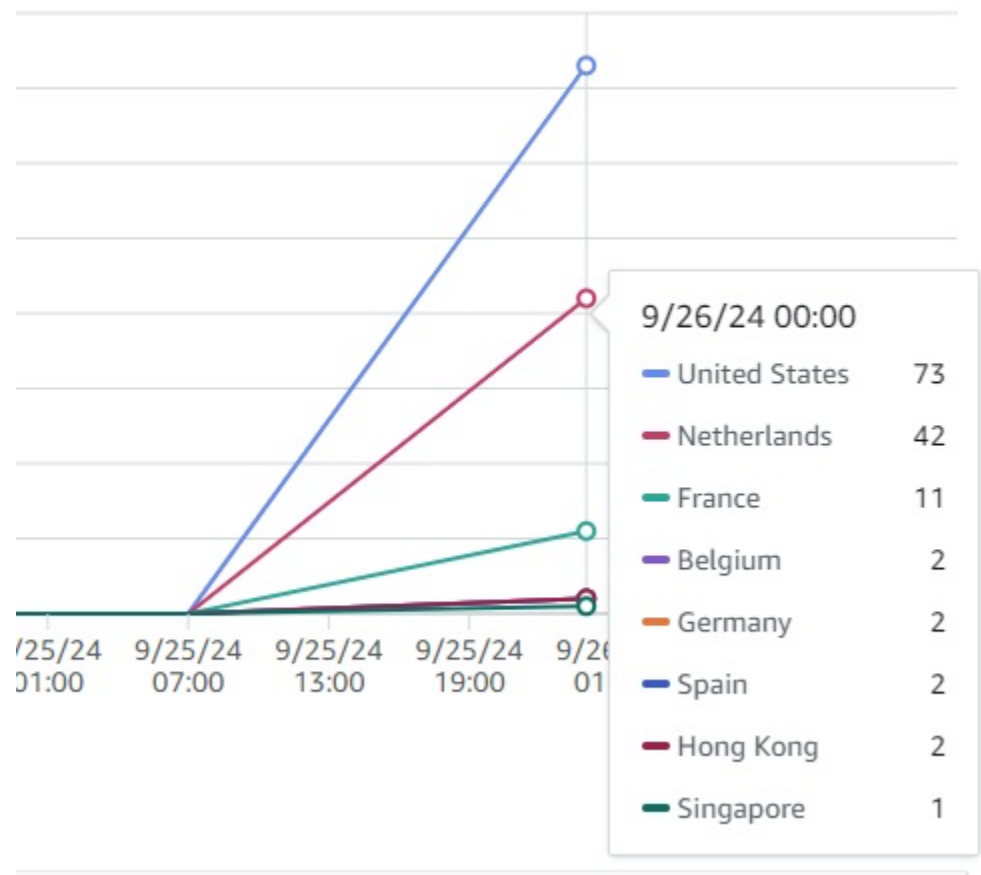
Appendix B

Telemetry of request to Sum Cloud and error rate in AWS



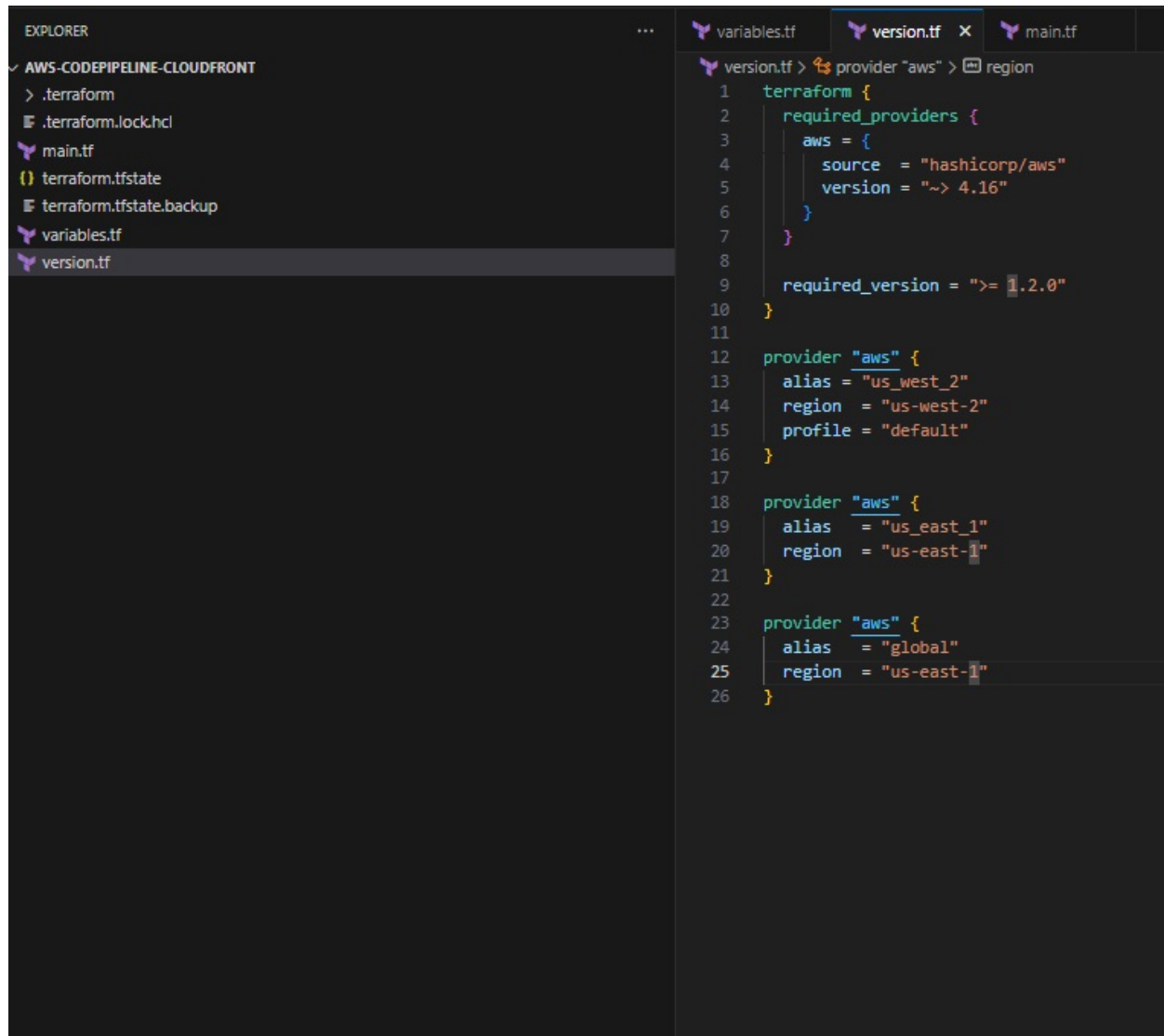
Appendix C

Locations of users visiting Sum Cloud Web Application



Appendix D

Version.tf file

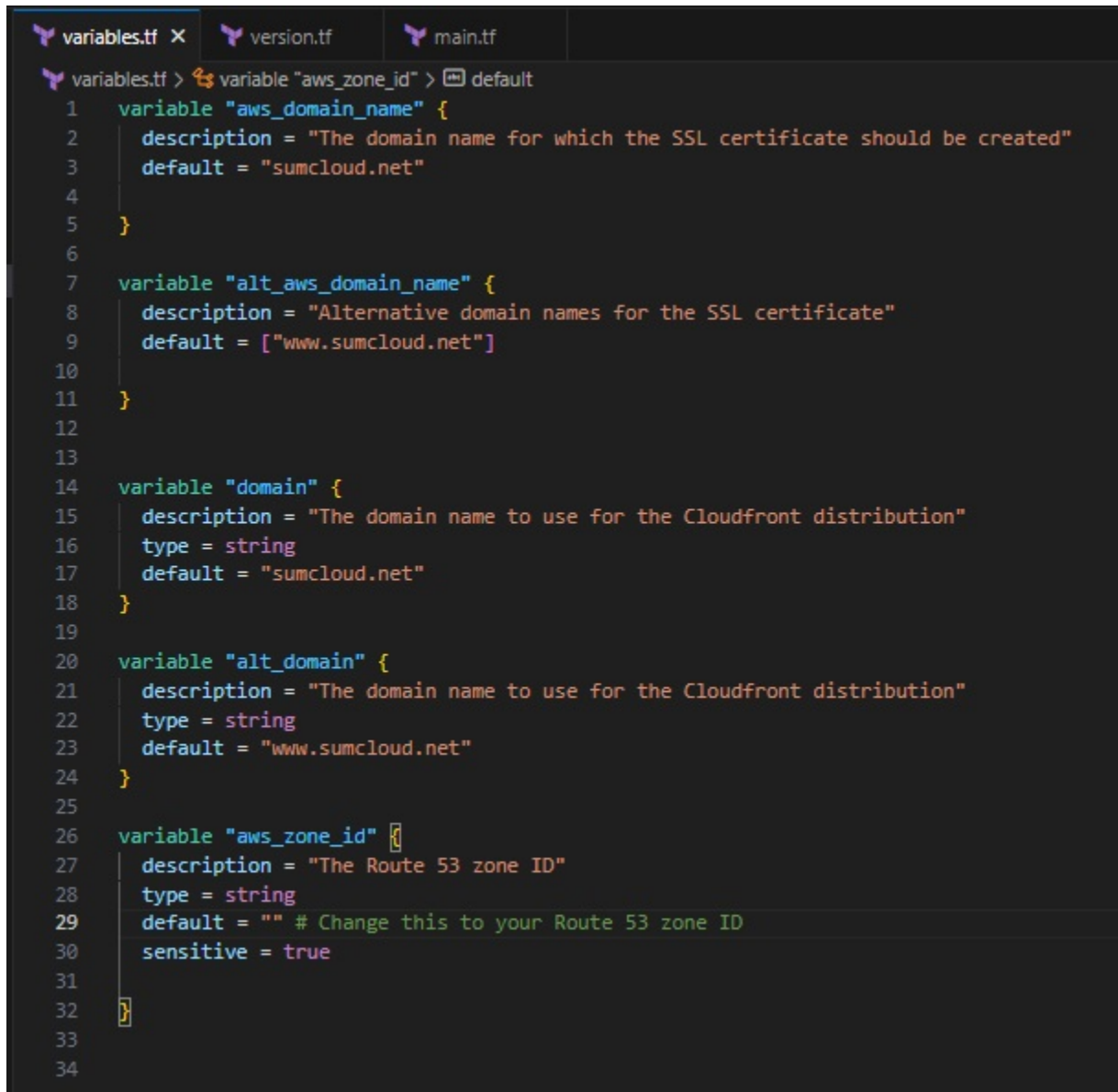


The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'AWS-CODEPIPELINE-CLOUDFRONT'. The files listed are: '.terraform', '.terraform.lock.hcl', 'main.tf', 'terraform.tfstate', 'terraform.tfstate.backup', 'variables.tf', and 'version.tf'. The 'version.tf' file is selected and highlighted. The main editor area shows the content of 'version.tf', which is a Terraform configuration file. It defines the required providers for AWS, specifying the source and version for the 'aws' provider, and defines three provider aliases for different AWS regions: 'us-west-2', 'us-east-1', and 'global'. The code is as follows:

```
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 }
22
23 provider "aws" {
24   alias = "global"
25   region = "us-east-1"
26 }
```

Appendix E

Variables.tf file



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


Appendix F

ACM Resource, IaC

```

main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
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 }
44

```

Appendix G

WAF for CloudFront, IaC

```
variables.tf  version.tf  main.tf  x
main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
39 resource "aws_acm_certificate_validation" "certificate_validation" {
43 }
44
45 # WAF for CloudFront
46
47 resource "aws_wafv2_web_acl" "web_acl" {
48   provider = aws.global
49   name      = "sum-cloud-web-acl-capstone"
50   description = "WAF Web ACL For CloudFront Capstone Project"
51   scope      = "CLOUDFRONT"
52   default_action {
53     allow {}
54   }
55
56   rule {
57     name      = "bad-bot-rule"
58     priority  = 1
59     action {
60       block {}
61     }
62
63     statement {
64       byte_match_statement {
65         search_string = "bad-bot"
66         field_to_match {
67           uri_path {}
68         }
69       }
70       text_transformation {
71         priority = 0
72         type     = "NONE"
73       }
74       positional_constraint = "CONTAINS"
75     }
76   }
77
78   visibility_config {
79     sampled_requests_enabled = true
80     cloudwatch_metrics_enabled = true
81     metric_name = "WAFMetric"
82   }
83
84   visibility_config {
85     cloudwatch_metrics_enabled = true
86     metric_name = "CloudWatchMetric"
87     sampled_requests_enabled = true
88   }
89 }
```

Appendix H

CloudFront, IAC

```

main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
89 }
84 visibility_config {
88 |
89 }
90
91 # Cloudfront
92
93 resource "aws_cloudfront_origin_access_identity" "oai" {
94 | comment = "OAI for my S3 bucket"
95 }
96
97 resource "aws_cloudfront_distribution" "s3_distribution" {
98 | origin {
99 |   domain_name      = aws_s3_bucket.codepipeline_bucket.bucket_regional_domain_name
100 |   origin_id        = aws_s3_bucket.codepipeline_bucket.bucket_id
101 |
102 |   s3_origin_config {
103 |     origin_access_identity = aws_cloudfront_origin_access_identity.oai.cloudfront_access_identity_path
104 |   }
105 | }
106
107 enabled          = true
108 is_ipv6_enabled  = true
109 comment          = "Capstone Final Cloudfront Distribution"
110 default_root_object = "index.html"
111
112 aliases = [var.domain, var.alt_domain]
113
114 # Cache behavior with precedence 0
115 default_cache_behavior {
116 | allowed_methods = ["GET", "HEAD"]
117 | cached_methods  = ["GET", "HEAD"]
118 | target_origin_id = aws_s3_bucket.codepipeline_bucket.bucket_id
119 |
120 | forwarded_values {
121 |   query_string = false
122 |
123 |   cookies {
124 |     forward = "none"
125 |   }
126 | }
127
128 min_ttl          = 0
129 default_ttl      = 86400
130 max_ttl          = 31536000
131 compress         = true
132 viewer_protocol_policy = "redirect-to-https"
133
134 restrictions {
135 | geo_restriction {
136 |   restriction_type = "whitelist"
137 |   locations        = ["US"] # preferred location
138 | }
139 }
140
141 tags = {
142 | Environment = "development"
143 }
144
145 viewer_certificate {
146 | acm_certificate_arn = aws_acm_certificate.acm-certificate.arn
147 | ssl_support_method  = "sni-only"
148 | minimum_protocol_version = "TLSv1.2_2021"
149 }
150
151 web_acl_id = aws_wafv2_web_acl.web_acl.arn
152 }
153
154 resource "aws_route53_record" "cloudfront_alias" {

```

Appendix I

Route53 in CloudFront, IaC

```
variables.tf  version.tf  main.tf  x
main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
97  resource "aws_cloudfront_distribution" "s3_distribution" {
154  }
155
156
157
158  resource "aws_route53_record" "cloudfront_alias" {
159    zone_id = var.aws_zone_id
160    name     = var.domain
161    type     = "A"
162
163    alias {
164      name           = aws_cloudfront_distribution.s3_distribution.domain_name
165      zone_id        = aws_cloudfront_distribution.s3_distribution.hosted_zone_id
166      evaluate_target_health = false
167    }
168  }
169
170  resource "aws_route53_record" "cloudfront_alt_alias" {
171    zone_id = var.aws_zone_id
172    name     = var.alt_domain
173    type     = "A"
174
175    alias {
176      name           = aws_cloudfront_distribution.s3_distribution.domain_name
177      zone_id        = aws_cloudfront_distribution.s3_distribution.hosted_zone_id
178      evaluate_target_health = false
179    }
180  }
181
```

Appendix J

CodePipeline, IaC

```

variables.tf  version.tf  main.tf  X
main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
181
182 # Codepipeline
183
184 resource "aws_s3_bucket" "codepipeline_bucket" {
185     bucket = "capstone-final-bucket-wgu"
186 }
187
188 resource "aws_s3_bucket_policy" "my_bucket_policy" {
189     bucket = aws_s3_bucket.codepipeline_bucket.id
190
191     policy = jsonencode({
192         Version = "2012-10-17"
193         Statement = [
194             {
195                 Effect = "Allow"
196                 Principal = {
197                     AWS = aws_cloudfront_origin_access_identity.oai.iam_arn
198                 }
199                 Action = "s3:GetObject"
200                 Resource = "${aws_s3_bucket.codepipeline_bucket.arn}/*"
201             }
202         ]
203     })
204 }
205
206 resource "aws_s3_bucket_public_access_block" "codepipeline_bucket_pab" {
207     bucket = aws_s3_bucket.codepipeline_bucket.id
208
209     block_public_acls       = true
210     block_public_policy     = true
211     ignore_public_acls     = true
212     restrict_public_buckets = true
213 }
214
215 resource "aws_codestarconnections_connection" "git_cloud" {
216     name           = "Capstone-Final-Pipeline-Demo"
217     provider_type = "GitHub"
218 }
219
220 resource "aws_iam_role" "codepipeline_role" {
221     name = "codepipeline-role"
222     assume_role_policy = jsonencode({
223         Version = "2012-10-17"
224         Statement = [
225             {
226                 Action = "sts:AssumeRole"
227                 Effect = "Allow"
228                 Principal = {
229                     Service = "codepipeline.amazonaws.com"
230                 }
231             }
232         ]
233     })
234 }

```


Appendix K

CodePipeline, IaC II

```

variables.tf version.tf main.tf x
main.tf > resource "aws_cloudfront_distribution" "s3_distribution" > tags > Environment
220 resource "aws_iam_role" "codepipeline_role" {
222     assume_role_policy = jsonencode({
230         "Statement": [
231             {
232             }
233         ]
234     })
235 }
236 resource "aws_iam_role_policy" "codepipeline_policy" {
237     role = aws_iam_role.codepipeline_role.id
238
239     policy = jsonencode({
240         Version = "2012-10-17"
241         Statement = [
242             {
243                 Effect = "Allow"
244                 Action = [
245                     "s3:*",
246                     "codestar-connections:UseConnection",
247                     "codebuild:*",
248                     "codepipeline:*",
249                     "iam:PassRole"
250                 ],
251                 Resource = "*"
252             }
253         ]
254     })
255 }
256
257 resource "aws_codepipeline" "codepipeline" {
258     name = "my-capstone-final-codepipeline"
259     role_arn = aws_iam_role.codepipeline_role.arn
260
261     artifact_store {
262         location = aws_s3_bucket.codepipeline_bucket.bucket
263         type = "S3"
264     }
265
266     stage {
267         name = "Source"
268
269         action {
270             name = "Source"
271             category = "Source"
272             owner = "AWS"
273             provider = "CodeStarSourceConnection"
274             version = "1"
275             output_artifacts = ["source_output"]
276
277             configuration = {
278                 ConnectionArn = aws_codestarconnections_connection.git_cloud.arn
279                 FullRepositoryId = "JordanSum/Sum-Cloud-V0.10"
280                 BranchName = "main"
281             }
282         }
283     }
284
285     stage {
286         name = "Deploy"
287
288         action {
289             name = "S3Deploy"
290             category = "Deploy"
291             owner = "AWS"
292             provider = "S3"
293             input_artifacts = ["source_output"]
294             version = "1"
295         }
296     }

```

Appendix L

CloudFront Resource, AWS Console

CloudFront

Distributions (1) [Info](#)

Search all distributions

Enable

Disable

Delete

Create distribution

< 1 >

<input type="checkbox"/>	ID	Description	Type	Domai...	Alterna...	Origins	Status	Last modified
<input type="checkbox"/>	E2D86BEG00AY1G	Capstone Final Cloudfront Distribution	Production	d1zrodpl7...	www.sumcloud	capstone-final-l	Enabled	September 26, 2024 at 4:20:09 AM UTC

Appendix M

S3 Resource, AWS Console

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

capstone-final-bucket-wgu info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (11) info

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	.github/	Folder	-	-	-
<input type="checkbox"/>	vscode/	Folder	-	-	-
<input type="checkbox"/>	favicon.ico	ico	September 25, 2024, 21:22:03 (UTC-07:00)	72.2 KB	Standard
<input type="checkbox"/>	index.html	html	September 25, 2024, 21:22:03 (UTC-07:00)	2.6 KB	Standard
<input type="checkbox"/>	landmark.jpg	jpg	September 25, 2024, 21:22:03 (UTC-07:00)	916.8 KB	Standard
<input type="checkbox"/>	my-capstone-final-co/	Folder	-	-	-
<input type="checkbox"/>	script.js	js	September 25, 2024, 21:22:03 (UTC-07:00)	5.7 KB	Standard
<input type="checkbox"/>	script2.js	js	September 25, 2024, 21:22:03 (UTC-07:00)	1.6 KB	Standard
<input type="checkbox"/>	site.css	css	September 25, 2024, 21:22:03 (UTC-07:00)	341.0 B	Standard
<input type="checkbox"/>	style.css	css	September 25, 2024, 21:22:03 (UTC-07:00)	7.1 KB	Standard
<input type="checkbox"/>	winner.html	html	September 25, 2024, 21:22:03 (UTC-07:00)	343.0 B	Standard

Appendix N

CodePipeline Resource, AWS Console

Developer Tools > CodePipeline > Pipelines > my-capstone-final-codepipeline

my-capstone-final-codepipeline

Pipeline type: V1 Execution mode: SUPERSEDED

[Notify](#) [Edit](#) [Stop execution](#) [Clone pipeline](#) [Release change](#)

Source Succeeded

Pipeline execution ID: [246ea33e-da0f-46dc-9412-a38181a84a79](#)

Source

[GitHub / Version 2](#)

Succeeded - 6 minutes ago

[738d9f87](#)

[View details](#)

[738d9f87](#) Source: update for capstone purpose

[Disable transition](#)

Deploy Succeeded

Pipeline execution ID: [246ea33e-da0f-46dc-9412-a38181a84a79](#)

S3Deploy

[Amazon S3](#)

Succeeded - 6 minutes ago

[View details](#)

[738d9f87](#) Source: update for capstone purpose

Appendix O

ACM Resource, AWS Console

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"/>	32623879-4ccd-43a4-b1e9-7d19fe71ccab	sumcloud.net	Amazon Issued	Issued	Yes	Eligible	RSA 2048

Appendix P

WAF Resource, AWS Console

[AWS WAF](#) > [Web ACLs](#) > sum-cloud-web-acl-capstone

sum-cloud-web-acl-capstone

Download web ACL as JSON

[Traffic overview](#) | [Rules](#) | [Associated AWS resources](#) | [Custom response bodies](#) | [Logging and metrics](#) | [Sampled requests](#) | [CloudWatch Log Insights](#)

You can learn more about the dashboards [here](#). Please provide feedback for the dashboards.

Feedback X

Data filters [info](#)
Select the time range and terminating actions that you want to view in the dashboard. You can select a time range relative to now and you can select an absolute time range.

Terminating rule actions
[Dropdown]

Time range
[Calendar icon] Last 3 hours

Time zone
Local time [Dropdown]

Refresh
[Refresh icon]

Blocked X

Allowed X

Captcha X

Challenge X

[All traffic](#) | Bot Control

▼ **Action totals for the specified time range - all traffic**

Request counts for all traffic during the specified time range. This shows counts for all possible terminating actions, while the rest of the dashboard shows only the actions that you've selected in the filters. If you're filtering on a relative time range, each action also shows the percentage change from the prior, equivalent-length time range. For example, if you've chosen 1 day as the time range, the percentage change reflects the difference between 48-24 hours ago and 24-0 hours ago.

Total	Blocked	Allowed	Captcha	Challenge
5	0	5	0	0
▲ 100%		▲ 100%		

Appendix Q

Nslookup sumcloud.net

```
Non-authoritative answer:  
Name:      sumcloud.net  
Addresses: 3.163.158.71  
           3.163.158.95  
           3.163.158.86  
           3.163.158.92
```

Appendix R

HTTPS website of sumcloud.net

