Day 16 Internship Report

CloudFront optimization, validation improvements, RDS Terraform updates.

Intern Name:	Arsalan Sharief
Company:	Signiance Technology
Role:	DevOps Intern
Day:	16

1. Investigation and Resolution of CloudFront Cache Miss Issue

Objective

To analyze why CloudFront was returning a cache miss status from the CDN instead of a cache hit in the browser Developer Tools, ensuring efficient content delivery.

Issue Identified

While inspecting network requests in the browser, the response header indicated that the object was not being served from the cache:

```
X-Cache: Miss from cloudfront
```

Since CloudFront was configured for caching, the expected behavior was to return:

```
X-Cache: Hit from cloudfront
```

Analysis Performed

- Inspected response headers in browser DevTools to confirm the behavior across multiple requests.
- Reviewed CloudFront Distribution Behavior settings in the AWS Console.
- Checked the specific Cache Policy configuration attached to the behavior.
- Identified that the "Caching Disabled" policy was mistakenly applied instead of an optimized caching policy.

Resolution Implemented

- Updated CloudFront behavior settings.
- Changed Cache Policy from Caching Disabled to CachingOptimized.
- Verified configuration for the S3 bucket origin.
- Tested the application again to confirm headers.

Result

After implementing the changes, the response header correctly returned:

```
X-Cache: Hit from cloudfront
```

This confirmed that CloudFront caching was functioning properly.

2. Frontend Contact Number Validation Implementation

Objective

To ensure the contact number field accepts only numeric values and enforces exactly 10 digits to prevent data entry errors.

Changes Implemented

- Restricted input field to numeric values only.
- Enforced an exact 10-digit limit.
- Prevented alpha-character and special symbol input.
- Updated validation logic in `index.html`.

Code Enhancements

- Applied numeric-only input restriction using JavaScript event listeners.
- Set maximum length attribute to 10 digits.
- Tested for edge cases including manual typing and invalid input pasting.

Deployment

- Updated `index.html` locally.
- Uploaded the modified file to the S3 bucket.
- Verified functionality in the live environment after deployment.

3. Architecture Diagram Update

Objective

To update the infrastructure architecture diagram to accurately reflect recent changes and additions to the stack.

Modifications Made

- Added Amazon RDS component to the data layer.
- Updated application flow to visualize the path: CloudFront → ALB → Backend → RDS.
- Reflected current deployed infrastructure.
- Ensured diagram accuracy according to the production setup.

4. Terraform Module Creation for RDS

Objective

To create a reusable and modular Terraform configuration for provisioning Amazon RDS instances, promoting Infrastructure as Code (IaC) best practices.

Tasks Completed

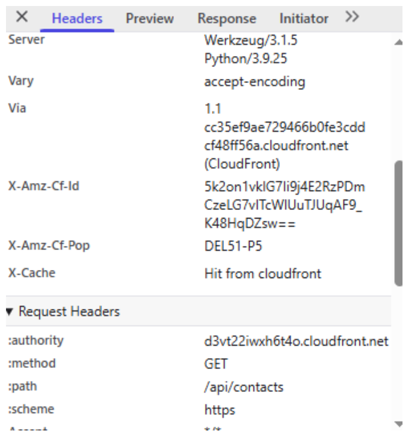
- Created a modular Terraform directory structure for RDS.
- Defined input variables including:
    - Database name
    - Instance class
    - Storage configuration
    - Subnet group
    - Engine configuration
- Configured associated resources:
    - Security Groups

        - DB Subnet Groups
    - Integrated the RDS module into the main Terraform configuration file.
    - Ensured modular and reusable infrastructure design for future scalability.

Summary of Accomplishments

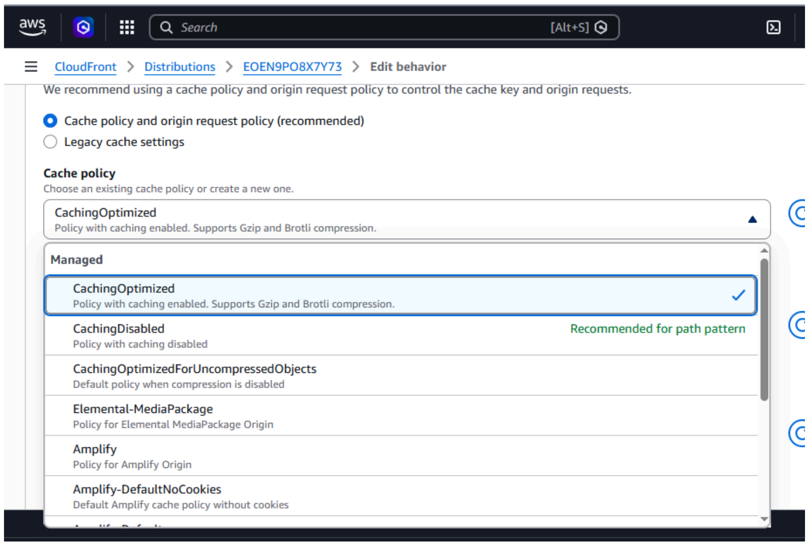
Today's work focused on optimizing CloudFront caching behavior to improve performance and enhancing frontend validation logic to ensure data integrity. Additionally, infrastructure documentation was updated to reflect the current architecture, and a reusable Terraform module for RDS was implemented to streamline future database provisioning. All tasks were successfully completed.

Screenshots



```
Server          Werkzeug/3.1.5
                Python/3.9.25
Vary            accept-encoding
Via             1.1
                cc35ef9ae729466b0fe3cdd
                cf48ff56a.cloudfront.net
                (CloudFront)
X-Amz-Cf-Id     5k2on1vkIG7Ii9j4E2RzPDm
                CzeLG7vITcWIUuTJUqAF9_
                K48HqDZsw==
X-Amz-Cf-Pop    DEL51-P5
X-Cache         Hit from cloudfront
▾ Request Headers
:authority      d3vt22iwxh6t4o.cloudfront.net
:method         GET
:path           /api/contacts
:scheme         https
```

Shows a GET request to `/api/contacts` going through **AWS CloudFront**, with response coming from cache (`X-Cache: Hit from cloudfront`



CloudFront > Distributions > EOEN9PO8X7Y73 > Edit behavior

We recommend using a cache policy and origin request policy to control the cache key and origin requests.

- [✓ Cache policy and origin request policy (recommended)]
- Legacy cache settings

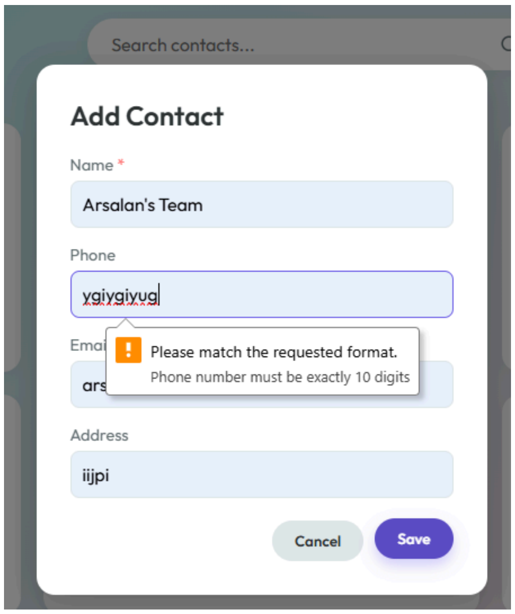
Cache policy
Choose an existing cache policy or create a new one.

CachingOptimized
Policy with caching enabled. Supports Gzip and Brotli compression.

Managed

- CachingOptimized ✓
  Policy with caching enabled. Supports Gzip and Brotli compression.
- CachingDisabled — Recommended for path pattern
  Policy with caching disabled
- CachingOptimizedForUncompressedObjects
  Default policy when compression is disabled
- Elemental-MediaPackage
  Policy for Elemental MediaPackage Origin
- Amplify
  Policy for Amplify Origin
- Amplify-DefaultNoCookies
  Default Amplify cache policy without cookies

CloudFront behavior is set to **CachingOptimized**.



Add Contact

Name *
Arsalan's Team

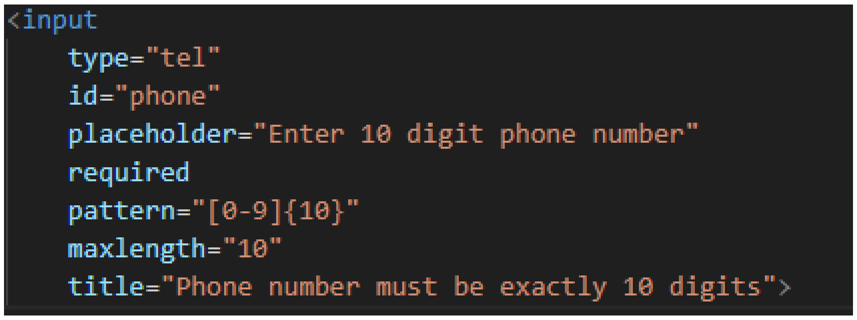
Phone
ygiygiyug

Please match the requested format.
Phone number must be exactly 10 digits

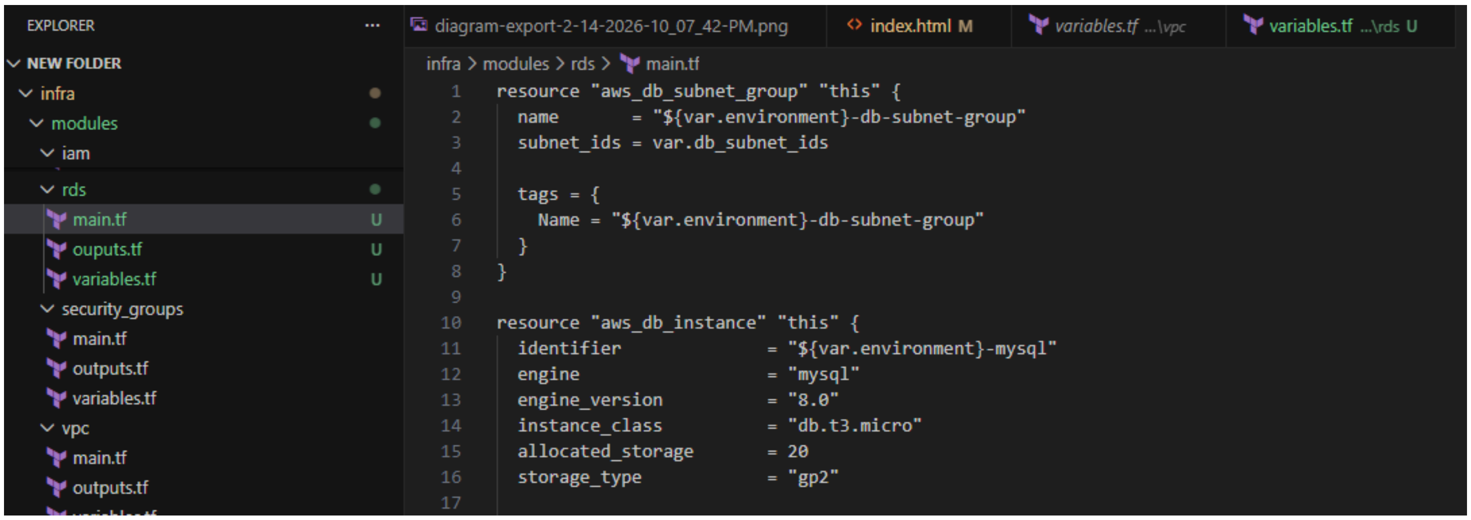
Email
ars

Address
iijpi

Frontend form showing phone validation error when letters are entered. Validation message says phone must be exactly 10 digits.


```html
<input
    type="tel"
    id="phone"
    placeholder="Enter 10 digit phone number"
    required
    pattern="[0-9]{10}"
    maxlength="10"
    title="Phone number must be exactly 10 digits">
```

Input field uses `pattern="[0-9]{10}"` and `maxlength="10"` to allow only exactly 10 digits



```hcl
resource "aws_db_subnet_group" "this" {
  name       = "${var.environment}-db-subnet-group"
  subnet_ids = var.db_subnet_ids

  tags = {
    Name = "${var.environment}-db-subnet-group"
  }
}

resource "aws_db_instance" "this" {
  identifier             = "${var.environment}-mysql"
  engine                 = "mysql"
  engine_version         = "8.0"
  instance_class         = "db.t3.micro"
  allocated_storage      = 20
  storage_type           = "gp2"

```

Terraform Module for RDS

Prepared by: ArsalanSharief
DevOps Intern – Signiance Technology