Scaling WordPress Blogs on AWS with Terraform Infrastructure-as-Code

Jacek Roszkowiak

neuefische AWS cloud-24-3

CONTENTS

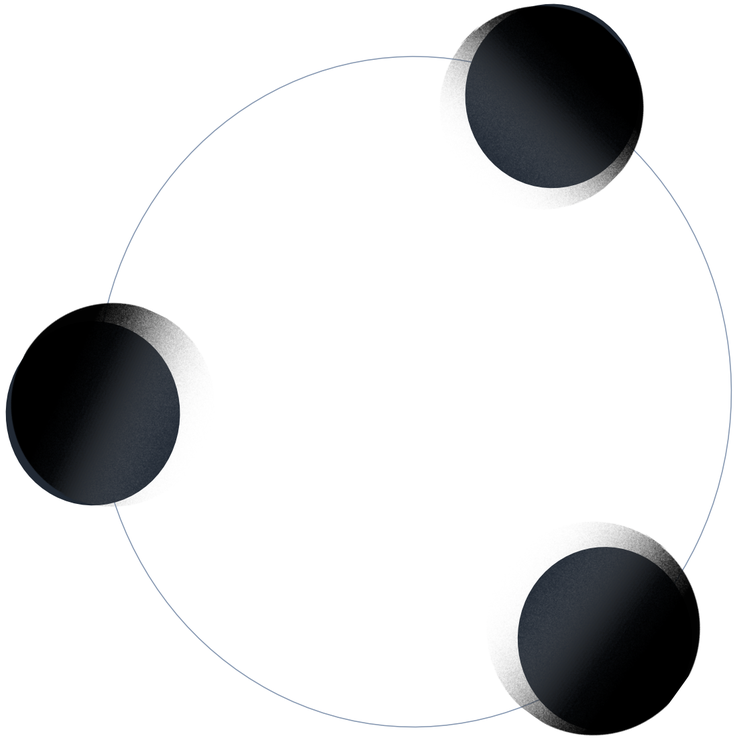
01

Project Overview

Project Objectives

Automate WordPress deployment

The project aims to streamline the deployment process of a WordPress website, reducing manual effort and the potential for errors. This automation will enable quicker updates and consistent configurations for the web application.

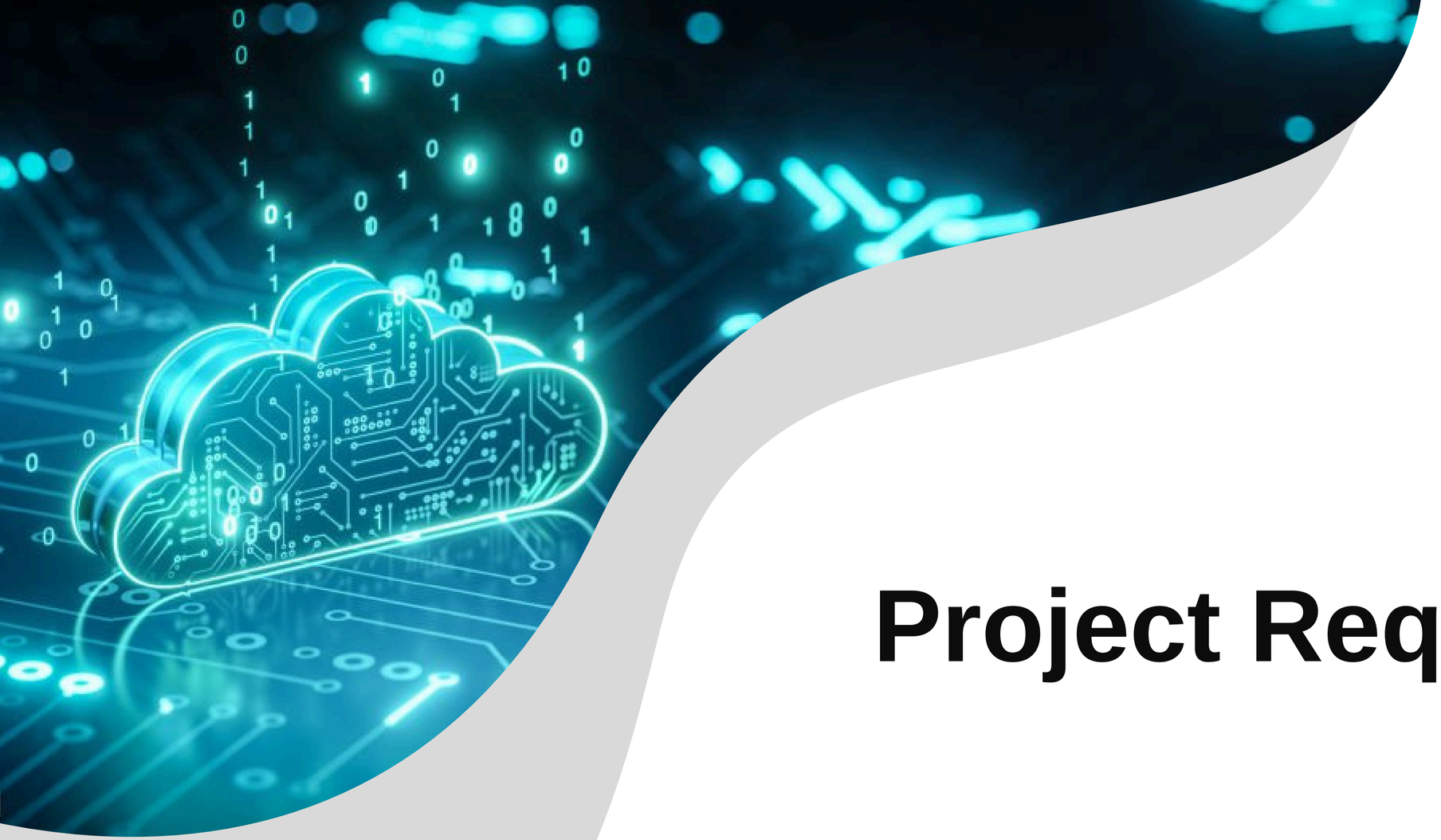


Utilise Terraform for infrastructure.

Terraform will be used as the primary tool for infrastructure as code, allowing for version-controlled and repeatable deployments. Its declarative nature enables easier management of AWS resources through configuration files.

Implement scalable architecture

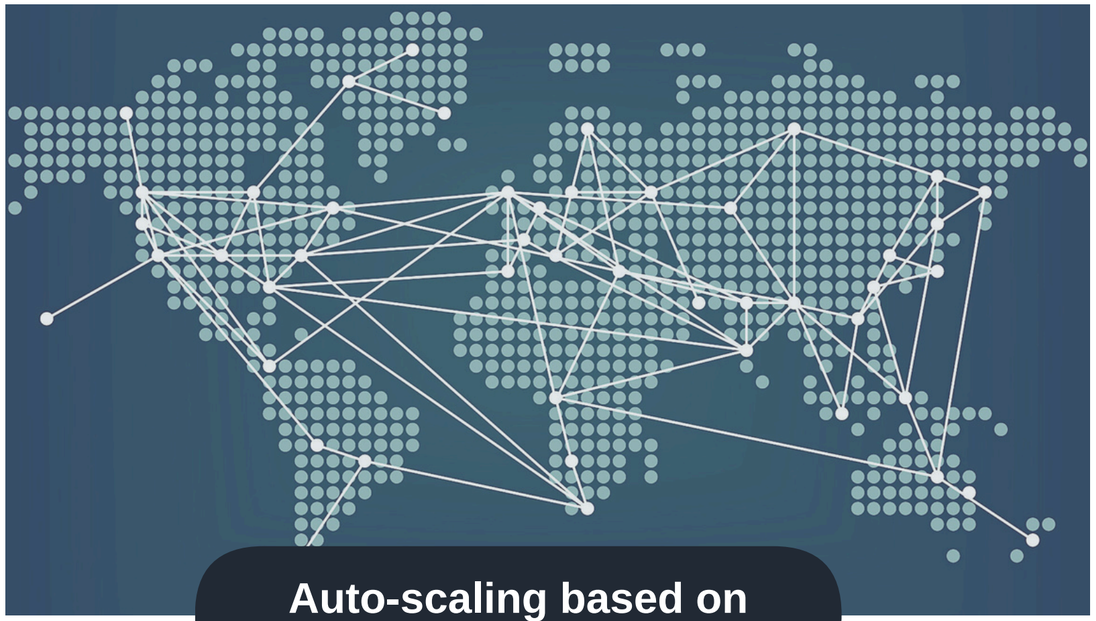
A scalable architecture will be designed to handle varying loads. This ensures that the WordPress website can accommodate sudden spikes in traffic by automatically adjusting the number of resources allocated.



02

Project Requirements

Functional Requirements



Auto-scaling based on traffic

The website must automatically adjust resource capacity in response to real-time traffic metrics, ensuring optimal performance without manual intervention.

High availability across AZs

To maintain service continuity, the deployment will be designed to remain operational even if one availability zone encounters an outage, leveraging AWS's multi-AZ capabilities.

Non-Functional Requirements

Availability: 99.9%

The architecture will target a high availability standard, ensuring that the WordPress site is up and running 99.9% of the time, which translates to minimal downtime each year.

Latency: < 200ms for 95% requests

Performance metrics will be defined to ensure that 95% of all user requests receive responses within 200 milliseconds, enhancing the user experience on the site.

Security measures

The project will incorporate essential security practices using IAM roles and Security Groups, along with enabling HTTPS to secure user data during transmission.

03

Project Objectives

Design Goals

Scalable architecture for WordPress

The architecture emphasizes scalability features that enable the WordPress site to efficiently manage varying user traffic and maintain consistent functionality under different load conditions.

Fault-tolerant infrastructure

The design ensures that resources remain operational even in failure scenarios by utilizing redundancy and failover strategies, primarily through AWS services. AWS services are designed to minimize recovery time from failures and impact on data. They use data stores that acknowledge requests only after they are durably stored across multiple replicas within a Region and employ fault isolation provided by Availability Zones. In case of failure, these services automatically route traffic to healthy locations, ensuring continuous operation.

Collaboration and Management

Effective use of Git and GitHub

The project uses Git for version control, enabling collaborative contributions, maintaining a history of changes, and facilitating code reviews through pull requests on GitHub.

Scrum methodology for delivery

The project employs Scrum as the project management methodology, breaking down work into iterations (sprints) that foster regular reflection and adaptation, ensuring continuous improvement throughout development.

04

Architecture Design

Architecture Components

- **Virtual Private Cloud (VPC):**

  - two Public Subnets
  - two Private Subnets
  - Internet Gateway (IGW)
  - separate Route Tables for Public and Private Subnets
  - security group: Allows HTTP and SSH Traffic

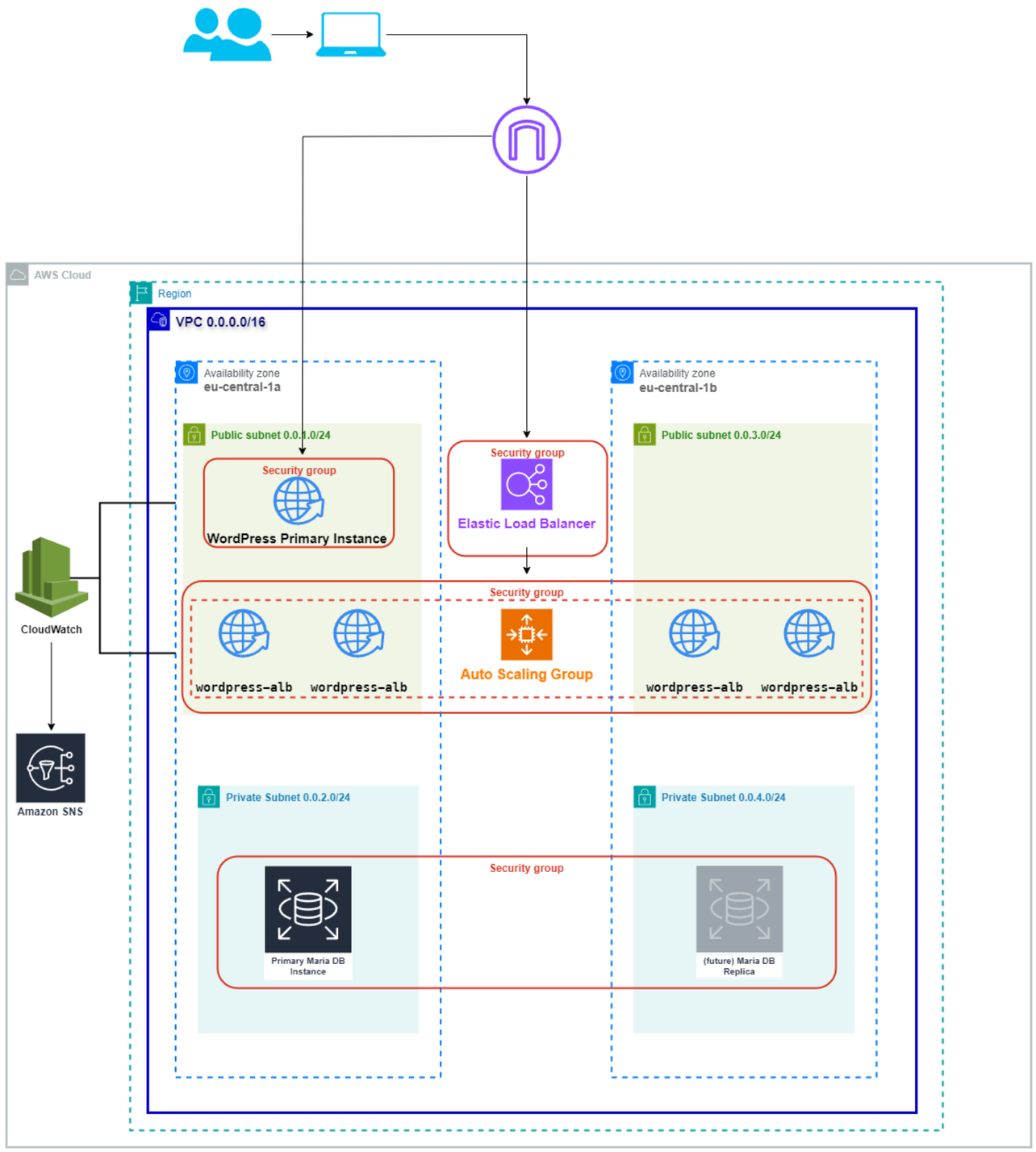
- **Compute Resources:**
  - **EC2 Instances:**

    - pre-configured with Apache, PHP, and WordPress
    - connected to RDS Database for Data Storage
    - utilize User Data for Automated Configuration

- **Auto Scaling Group (ASG):**
  - minimum Capacity: 1 Instance
  - desired Capacity: 2 Instances
  - maximum Capacity: 4 Instances

- **Load Balancing:**

  - application Load Balancer (ALB)
  - target Group
  - listener Configuration



Infrastructure Overview

VPC with public and private subnets

The architecture features a Virtual Private Cloud (VPC) that includes both public and private subnets, and two availability zones ensuring high availability and secure and controlled access to resources based on their roles.

Internet Gateway for connectivity

An Internet Gateway is implemented to facilitate communication between the VPC and the outside internet, allowing users to access the WordPress site reliably.

Load Balancing and Scaling

Application Load Balancer (ALB)

An Application Load Balancer (ALB) is set up to efficiently distribute incoming web traffic among the EC2 instances, enhancing the site’s performance and availability.

Auto Scaling Group (ASG) configuration

The Auto Scaling Group (ASG) is configured to manage the lifecycle of WordPress instances, defining thresholds for scaling up or down the instance count based on actual load ensuring resource optimization.

05

Infrastructure Components

Virtual Private Cloud (VPC)

Public and private subnets

The VPC will have two public subnets for internet-facing applications and two private subnets where the databases and backend services will reside, increasing security by isolating sensitive resources.

Security Group for traffic control

Security Groups will be established to restrict access to necessary ports only, allowing HTTP and SSH traffic while blocking any unwanted connections to safeguard the infrastructure.

Compute Resources

Pre-configured EC2 Instances

EC2 instances are launched, each pre-configured with Apache, PHP, and WordPress, ensuring that they are ready to serve content immediately upon deployment.

RDS for persistent data storage

An Amazon RDS instance (MaiiaDB) is set up for reliable and scalable database management ensuring high availability and resilience against data loss.

06

Getting Started with the Project

Prerequisites

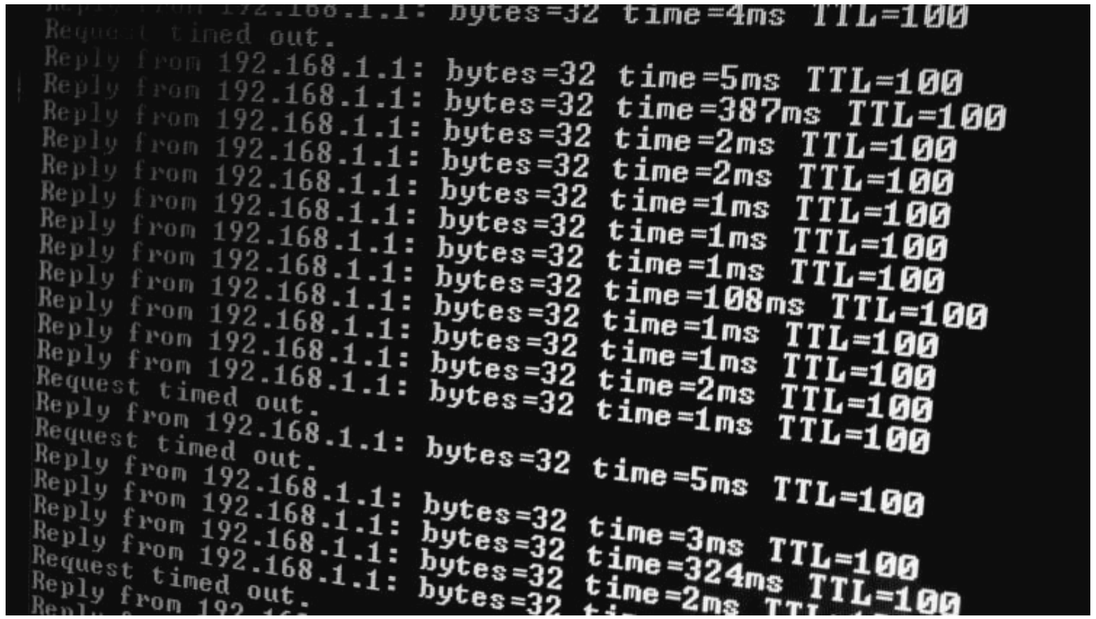
AWS account setup

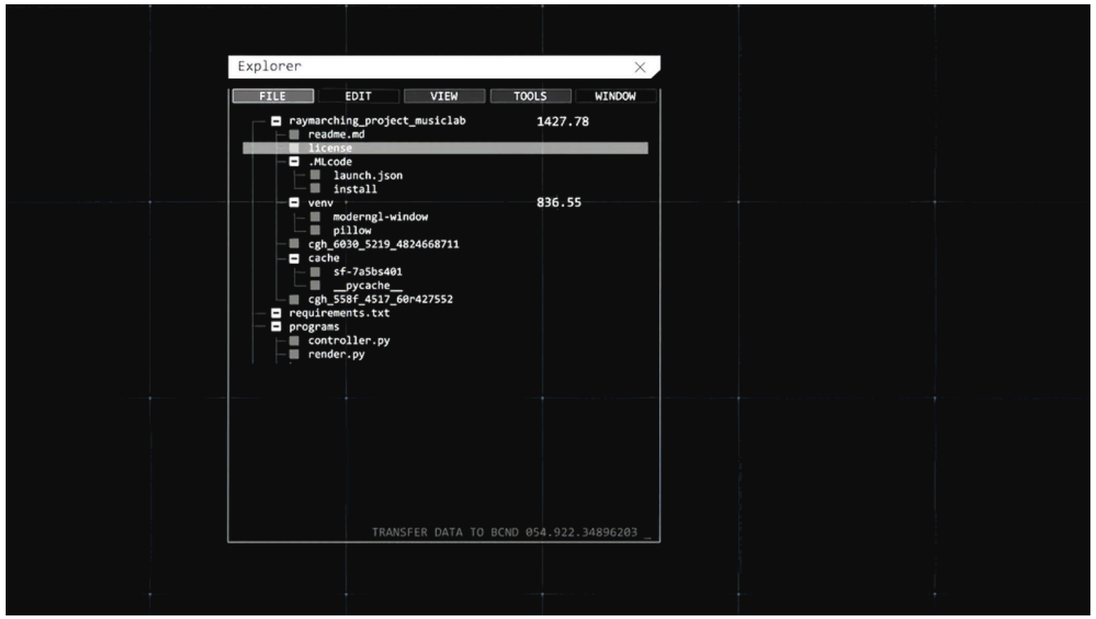
Users must have a valid AWS account set up with appropriate permissions to create resources necessary for the project, ensuring compliance with AWS policies.

Install Terraform and Git

Terraform and Git should be installed on the local machine following official installation instructions, allowing for effective infrastructure management and version control.

Repository and Configuration





Clone the repository

Users can clone the project repository from GitHub using the **git clone** command, which will download all necessary files for the project onto their local machine.

Configure AWS credentials

AWS credentials should be configured in environment variables or local profile settings, ensuring that Terraform has the necessary access to create and manage resources in the AWS account.

Deployment Steps

Review Terraform configurations

It's essential to go through the Terraform configuration files to understand the resources being provisioned and their configurations before initiating the deployment process.

Configure environmental variables

Create a **terraform.tfvars** file and set variables for the sandbox environment:

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**
- **AWS_SESSION_TOKEN**

and the set of variables for the database:

- **region** = "us-west-2"
- **az_primary** = "us-west-2a"
- **az_secondary** = "us-west-2b"
- **db_instance_class** = "db.t3.micro"
- **db_name** = "wordpress2"
- **db_user** = "wordpressuser2"
- **db_password** = "test2024test2"

Deploy infrastructure with Terraform commands

The last step is to execute specific Terraform commands to validate the configured infrastructure

`terraform init`

`terraform validate`

and finally deploy the infrastructure with command:

`terraform apply`

neue fische
School and Pool for Digital Talent

THANK YOU

19.12.2024

JACEK
ROSZKOWIAK