

AWS

Landing Zone

Document Control Information

Document Information

Document ID	AWS_Landingzone_25
Document Name	AWS_Landingzone_25
Project Name	Deloitte AWS Landing Zone
Customer	Deloitte
Author of the document	Marcos Brais Aller Amor, Yasmin Lema Blanco
Document version	1.0

Document version history

Version	Date	Additions/Modifications	Prepared/Revised by
1.0	11/06/2025	First version of the document	Marcos Brais Aller Amor Yasmin Lema Blanco

Document revision/approval history

Date	Name	Organization/Position	Feedback
<dd-mmm-yyyy >	<Name>	<Organization/Title>	<Comments>

Contents

1. Project Assignment Information	4
1.1. Assignment Summary	4
1.2. Project Scope	4
1.2.1. In-Scope Tasks (Baseline)	4
2. Landing Zone Definition	5
2.1. Overview	5
2.2. Prerequisites	5
2.3. IAM	8
2.4. Policies and Guardrails	9
2.4.1. Service Guardrails	9
2.4.1.1. IAM	9
2.4.1.2. EC2	10
2.4.1.3. S3	11
2.4.1.4. CloudWatch	12
2.4.1.5. Env	13
2.4.1.6. VPC	15
2.4.1.7. Tags	15
2.5. Budget and Cost Management	15
3. Terraform Resources	17
3.1. Code Structure	17
3.2. Modules	17
3.3. Resources Implemented	18
3.3.1. Github	18
3.3.2. Infracost	21
4. Networking	22
4.1. Network Structure	22
5. Documentation	25

1. Project Assignment Information

1.1. Assignment Summary

This project consists of developing and deploying a modular and cost-effective AWS Landing Zone using Terraform.

1.2. Project Scope

The scope of this project includes:

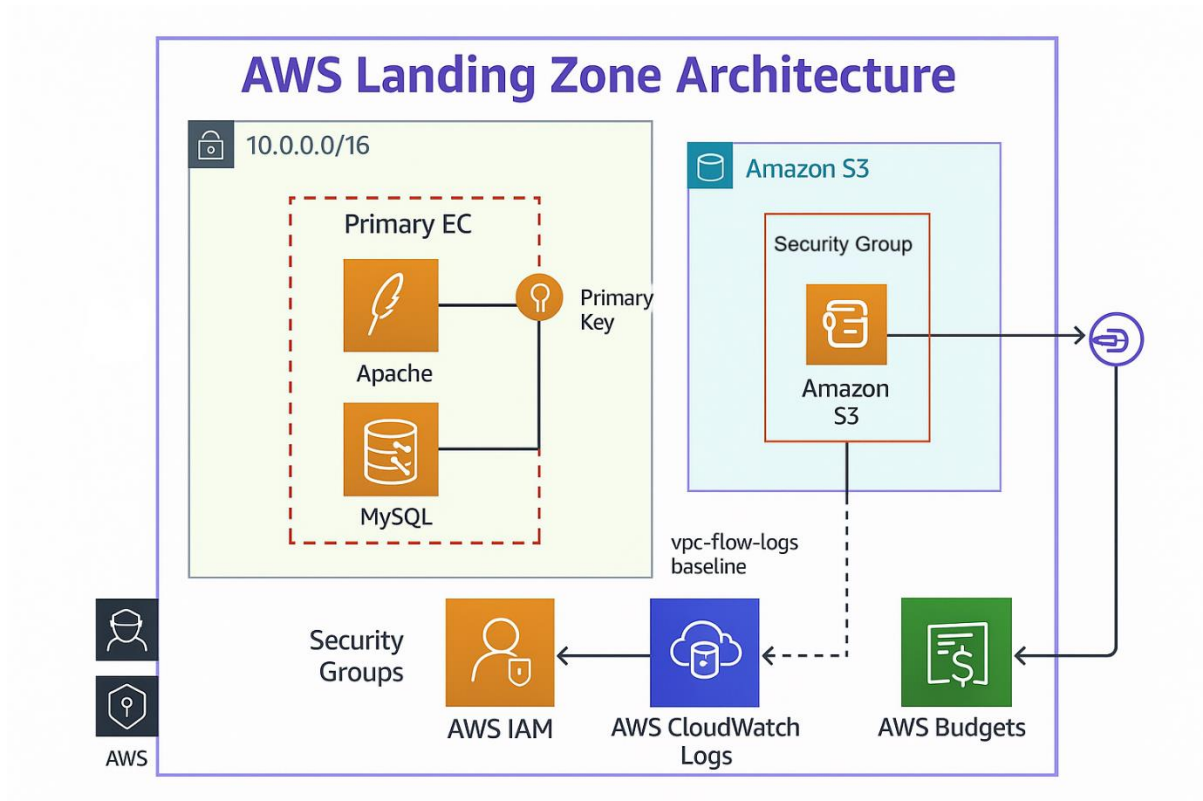
- Designing a secure and scalable network architecture with a Virtual Private Cloud (VPC), including one public and one private subnet.
- Deploying essential AWS services within the Free Tier limits, such as:
 - Amazon EC2 (for web and database servers)
 - Amazon S3 (for storage)
 - IAM (users, groups, roles, and policies)
 - Amazon CloudWatch Logs (for monitoring)
 - AWS Budgets (for cost tracking)
- Implementing Infrastructure as Code (IaC) using Terraform with a modular folder structure
- Applying best practices in tagging, naming conventions, and version control using GitHub.

1.2.1. In-Scope Tasks (Baseline)

These are the tasks to be carried out considering the scope of the project:

- Secure VPC design with public and private subnets
- Design and deploy EC2
- Design and deploy S3
- Design and deploy IAM
- Design and deploy CloudWatch Logs
- Design and deploy AWS Budgets
- Terraform modularization
- Cost estimation with Infracost: it should be zero by using the Free Tier
- State management with Terraform Cloud
- GitHub for version control

2. Landing Zone Definition



2.1. Overview

This Landing Zone was created using Terraform within the AWS Free Tier. It offers a secure, modular infrastructure including private/public networking, access control (IAM), monitoring, and cost tracking.

2.2. Prerequisites

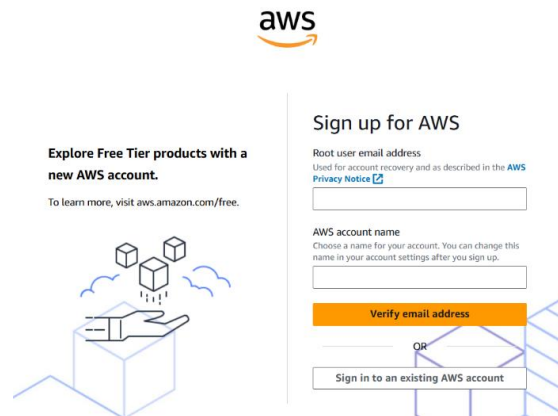
2.2.1. AWS Account

2.2.1.1. Create an AWS account

- Open your web browser and go to: [AWS](https://aws.amazon.com)
- Click on "Create an AWS Account" in the top right corner



- Enter your email and a name for the account



The screenshot shows the AWS sign-up page. On the left, there is a promotional banner for the Free Tier. The main content area is titled "Sign up for AWS". It contains two input fields: "Root user email address" and "AWS account name". Below the email field is a "Verify email address" button. Below the account name field is a "Sign in to an existing AWS account" button. There is also a "OR" separator between the two main paths.

aws

Explore Free Tier products with a new AWS account.
To learn more, visit aws.amazon.com/free.

Sign up for AWS

Root user email address
Used for account recovery and as described in the [AWS Privacy Notice](#)

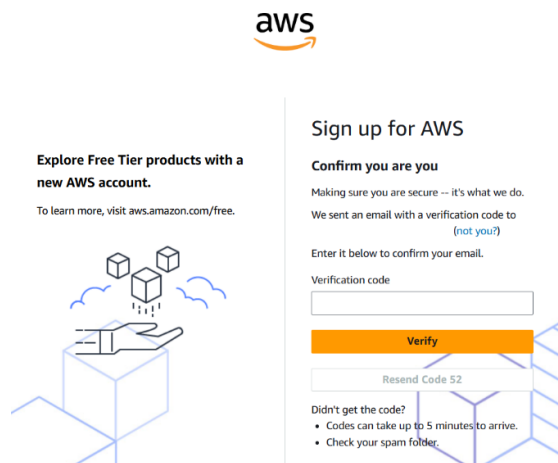
AWS account name
Choose a name for your account. You can change this name in your account settings after you sign up.

Verify email address

OR

Sign in to an existing AWS account

- AWS will send a verification code to your email, enter the code to proceed



The screenshot shows the second step of the AWS sign-up process. The title is "Sign up for AWS". The section is titled "Confirm you are you". It explains that a verification code has been sent to the email address. There is a "Verification code" input field and a "Verify" button. Below the input field is a "Resend Code 52" button. There is also a "Didn't get the code?" section with bullet points: "Codes can take up to 5 minutes to arrive." and "Check your spam folder."

aws

Explore Free Tier products with a new AWS account.
To learn more, visit aws.amazon.com/free.

Sign up for AWS

Confirm you are you
Making sure you are secure -- it's what we do.
We sent an email with a verification code to (not you?)
Enter it below to confirm your email.

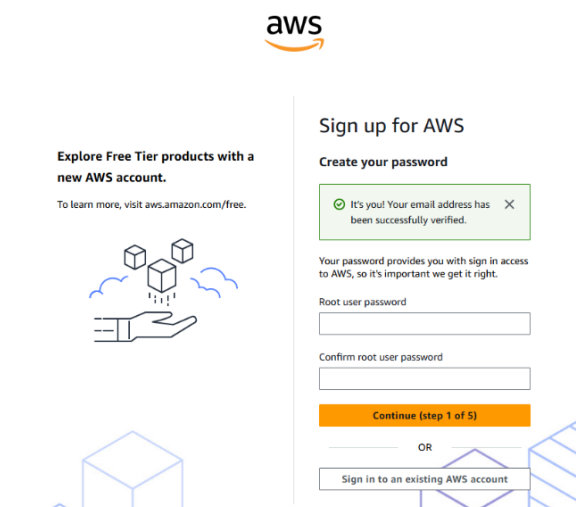
Verification code

Verify

Resend Code 52

Didn't get the code?
• Codes can take up to 5 minutes to arrive.
• Check your spam folder.

- Choose your password



The screenshot shows the third step of the AWS sign-up process. The title is "Sign up for AWS". The section is titled "Create your password". It shows a green success message: "It's you! Your email address has been successfully verified." Below this is a "Your password provides you with sign in access to AWS, so it's important we get it right." section. There are two input fields: "Root user password" and "Confirm root user password". Below these is a "Continue (step 1 of 5)" button. There is also a "OR" separator and a "Sign in to an existing AWS account" button.

aws

Explore Free Tier products with a new AWS account.
To learn more, visit aws.amazon.com/free.

Sign up for AWS

Create your password

It's you! Your email address has been successfully verified.

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password

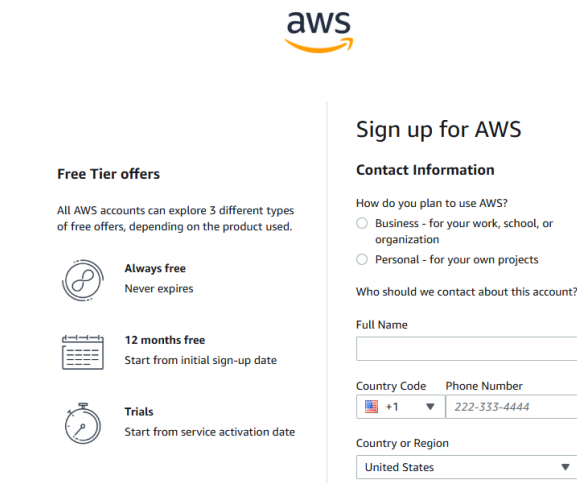
Confirm root user password

Continue (step 1 of 5)

OR

Sign in to an existing AWS account

- Provide the information below



The screenshot shows the AWS sign-up page. On the left, under 'Free Tier offers', it states: 'All AWS accounts can explore 3 different types of free offers, depending on the product used.' The offers listed are: 'Always free' (Never expires), '12 months free' (Start from initial sign-up date), and 'Trials' (Start from service activation date). On the right, under 'Sign up for AWS', the 'Contact Information' section asks 'How do you plan to use AWS?' with options for 'Business' or 'Personal'. It also asks 'Who should we contact about this account?' and provides fields for 'Full Name', 'Country Code' (set to +1), 'Phone Number' (222-333-4444), and 'Country or Region' (United States).

- Add Payment Information (AWS will perform a temporary authorization (usually \$1,01 USD) to verify the card)
- Choose a Support Plan (we selected the Basic Support Plan (Free))

2.2.2. Install and configure the AWS CLI in WSL

2.2.2.1. Install

- Install Required Dependencies

```
sudo apt-get install curl unzip -y
```

- Download the AWS CLI Installer

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

- Unzip the Installer

```
unzip awscliv2.zip
```

- Run the installer

```
sudo ./aws/install
```

- Verify installation

```
aws --version
```

2.2.2.2. Configure

- Once installed, configure the credentials of your AWS account:

```
aws configure
```

2.2.3. Terraform

2.2.3.1. Installation of Terraform

- Update of the package list and installation of essential packages

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl
```

- Downloads and saves the GPG key used to verify that packages from HashiCorp are secure and authentic

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

- Adds the HashiCorp repository to your system so you can install Terraform from it using APT

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
```

- Refresh the package list again and install Terraform

```
sudo apt-get update && sudo apt-get install terraform
```

2.2.4. GitHub repository for source code

2.2.5. Infracost for cost estimation

2.3. IAM

Custom IAM policies, groups, and users were created and organized by modules. The following users were defined:

- *admin-user*: full access to all resources
- *logs-s3-user*: access to CloudWatch Logs and S3
- *infra-user*: access to infrastructure resources (VPC, EC2, IAM, etc.)
- *billing-user*: access to budgets and cost reports

2.4. Policies and Guardrails

AWS IAM (Identity and Access Management) policies in Terraform allow you to define permissions and restrictions for users and roles in AWS. The meaning and structure of each of the policies is explained in detail below:

Policy	Purpose	Allow	Deny
aws_admin	Grants full administrative access	All actions on all resources within the AWS account	None
aws_security	Security and monitoring service access	Access to IAM, CloudTrail, GuardDuty, etc., specific S3 actions, EC2 describe actions; EC2 Instance Connect actions on specific resource	Start/stop/reboot/terminate EC2 on all resources
aws_operations	Operational tasks	Access to EC2, S3, RDS, CloudWatch, Auto Scaling, ELB, EC2 Instance Connect;	EC2 Instance Connect on monitoring server; specific S3 actions on monitored S3 bucket
aws_billing	Billing and cost management	View/manage billing, usage reports, payment methods, budgets, and cost explorer	None

2.4.1. Service Guardrails

2.4.1.1. IAM

IAM configuration ensures strong access control, alignment with organizational security policies, and enforced privilege segregation to protect critical AWS resources.

- **0028: Restricted IAM Password Policy Modification**

IAM Users and Roles are not permitted to modify password policies defined at the organizational level. These are managed exclusively by the platform team to ensure compliance and consistency.

- **0104: Controlled Inline Policy Management**

Only authorized users or roles are allowed to create or attach inline IAM policies. This minimizes the risk of untracked privileges and policy sprawl.

- **0113: Enforced Permission Boundaries**
All IAM Users, Roles, and Groups are subject to permission boundaries that limit their actions strictly to what their function requires.
- **0116: Root Account Protected with MFA**
Multi-Factor Authentication (MFA) is mandatory for the AWS root account. Root access without MFA is prohibited to prevent unauthorized entry.
- **0123: Protected MFA Configuration**
Only approved personnel can modify or remove MFA settings, ensuring secure and auditable access control practices.
- **0126: Controlled IAM User Creation**
The creation of new IAM Users is restricted to specific users and roles with elevated authorization, following identity governance policies.
- **0159: Restricted Group Management**
Modification or removal of users from Deloitte-managed groups requires specific permissions, ensuring role consistency and compliance.
- **0177: Privilege Segregation by Role**
Privilege assignment is segmented by function to prevent over-privileged roles from having both administrative and operational capabilities.
- **0186: Centralized Identity Store Enforcement**
User access is governed through an approved identity store (e.g., AWS SSO or federated Active Directory), ensuring centralized authentication and control.
- **0527: Avoidance of Default Master/Service Accounts**
Default accounts (such as root or admin) are disabled for regular operations. Only scoped roles with least-privilege access are used.
- **0556: Credential Creation Restrictions**
Access key and secret key creation is restricted to authorized processes only. AWS Secrets Manager is recommended for secure credential storage

2.4.1.2. EC2

EC2 instances are designed to ensure security, modularity, and scalability within the **Terraform-based Landing Zone** environment. Security controls have been established to prevent misconfigurations and ensure restricted access.

Security Principles:

- **Access and authentication management:**
 - AWS key pairs (*aws_key_pair*) are used for secure authentication on instances.
 - Access to instances is restricted using predefined **security groups** (*aws_security_group*).

- **Automation and configuration:**
 - *userdata.sh* is utilized for automated service initialization (*httpd* installation, system updates).
- **Monitoring and controlled deployment:**
- The monitoring instance (***monitoring_instance***) is only deployed if the *enable_monitoring* variable is enabled.

Specific Guardrails

- **0257:** All EC2 instances must be deployed within controlled subnets.
- **0268:** Direct access from the internet to instances is not allowed without explicit authorization.
- **0293:** Access permissions must follow the *least privilege* principle.
- **0314:** All instances must use managed key pairs under IAM policies.
- **0327:** Only authorized users may modify *userdata.sh* to prevent insecure configurations.
- **0351:** The monitoring instance must be protected against unauthorized access and configured with active auditing.

2.4.1.3. S3

Amazon S3 (Simple Storage Service) is a scalable object storage solution used within the Landing Zone architecture to manage, and store Terraform state files and other critical infrastructure data. The configuration of S3 ensures data integrity, version control, and security compliance by enforcing predefined policies.

Security Principles

- **Data Protection & Versioning:** Versioning is enabled to safeguard against unintended deletions or overwrites.
- **Access Control:** IAM-based restrictions ensure that only authorized roles can access, modify, or delete S3 resources.
- **Logging & Monitoring:** Enabled logging mechanisms allow tracking of changes for auditing and security purposes.
- **Least Privilege Enforcement:** Only the necessary permissions are granted for managing the bucket.

Specific Guardrails

- **0457: Controlled Bucket Creation**

- S3 bucket creation follows predefined naming conventions and must be approved through policy enforcement.
- **0472: Versioning Activation**
 - Versioning is mandatory to prevent accidental deletion or modification of stored objects.
- **0491: Encryption Compliance**
 - Server-side encryption must be enabled to ensure that all data stored in the bucket is encrypted.
- **0518: Restricted Public Access**
 - The bucket must not allow public access unless explicitly approved for external distribution.
- **0533: Logging & Monitoring Enforcement**
 - S3 access logs are enabled and integrated with CloudWatch for centralized monitoring.
- **0559: IAM-Based Policy Enforcement**
 - Access control policies prevent unauthorized users from modifying bucket properties.
- **0597: Terraform State File Protection**
 - If used for Terraform backend, proper IAM policies ensure only approved users can modify Terraform state files

2.4.1.4. *CloudWatch*

Amazon **CloudWatch** is used within the Landing Zone to monitor and log infrastructure performance. It ensures observability of AWS services, detects anomalies, and improves security posture through alerts and automated responses.

Security Principles

- **Centralized Logging:** Ensures system events and access logs are captured for auditing.
- **Monitoring Automation:** Implements alarms for proactive resource scaling and security responses.
- **Retention & Compliance:** Defines structured log retention to optimize storage and adhere to policies.

- **Least Privilege Access:** Restricts modifications to logs and alarms to authorized users only.

Specific Guardrails

- **0657: Enforced Log Group Naming Standards**
 - All CloudWatch Log Groups must follow predefined naming conventions for easy identification.
- **0674: Log Retention Governance**
 - Logs must have defined retention periods to comply with compliance requirements and cost control.
- **0689: Alarm Configuration Restrictions**
 - Creation and modification of alarms are limited to approved roles to prevent unauthorized changes.
- **0723: Unauthorized Access Logging**
 - All security-related events are logged and reviewed periodically.
- **0768: SNS Notification Enforcement**
 - CloudWatch alarms must trigger notifications to SNS topics for proactive response.
- **0812: Restricted Metrics Management**
 - Users cannot delete or modify critical monitoring metrics without administrative approval.

2.4.1.5. *Env*

The environment is a Terraform-based infrastructure setup designed to establish a **Landing Zone** in AWS. This Landing Zone provides a structured and secure foundation for managing cloud resources, ensuring proper networking, state management, and scalability.

1001: Terraform State Management

- The Terraform state is stored in Amazon S3, ensuring centralized and reliable infrastructure management.
- DynamoDB state locking prevents concurrent modifications, maintaining consistency and preventing corruption.

1002: Provider Configuration

- The AWS provider is configured using Terraform with the default deployment region eu-west-3.
- Supports dynamic configurations via variables to enable flexible adjustments.

1003: Networking and VPC Setup

- Creates a Virtual Private Cloud (VPC) with the CIDR block 10.0.0.0/16 for controlled network segmentation.
- Implements a public subnet (10.0.1.0/24) to facilitate external connectivity.

1004: Internet Gateway and Routing

- Deploys an Internet Gateway, allowing outbound and inbound traffic from the internet.
- Configures a route table for external access, with default routing to 0.0.0.0/0.

1005: Secure State Output and Variables

- Terraform output variables provide essential resource identifiers (e.g., VPC ID).
- Environment variables in terraform.tfvars ensure controlled and scalable configuration management.

1006: Secure Access Policies

- IAM-based access control restricts unauthorized modifications to Terraform state files.
- Only approved users can modify S3 backend settings or DynamoDB state-lock parameters.

1007: Infrastructure Scalability & Modularity

- Terraform modules support component reuse and structured infrastructure expansion.
- Environment allows seamless integration of additional networking components.

1008: Resource Association and Security

- Ensures proper subnet-to-route-table mapping to maintain structured traffic flow.
- Implements segregation of privilege across different infrastructure components.

1009: Approved Identity and Access Management

- Terraform backend access policies enforce role-based security for state modifications.
- Multi-Factor Authentication (MFA) is recommended for administrative Terraform actions.

1010: Master Account Restrictions

- Avoids using root and service accounts for Terraform execution and infrastructure deployment.
- Enforces least privilege principle for infrastructure management.

1011: Secure Key and Credential Management

- Restricts IAM access key creation to authorized Terraform processes.
- Encourages AWS-native Secrets Manager for secure storage of credentials.

2.4.1.6. VPC

Networking is managed through Amazon VPC to ensure logical segmentation, traffic control, and enhanced observability.

- **0086: Network-Level Logging Enabled**
VPC Flow Logs are enabled to capture inbound and outbound network traffic metadata, allowing security auditing and anomaly detection.

2.4.1.7. Tags

Tagging standards ensure consistent governance, cost tracking, and operational clarity across all AWS resources.

- **0101: Mandatory Tag Enforcement**
All AWS resources must include predefined mandatory tags (e.g., Environment, Owner, Project) to support auditability and cost allocation.
- **0237: Restricted Tag Modification**
Only authorized users or roles can create, modify, or delete resource tags, ensuring metadata integrity and preventing accidental changes.

2.5. Budget and Cost Management

- To keep control over AWS costs and ensure we remain within the Free Tier, we configured a monthly budget with a very low limit of \$0.01 USD. This setup allows us to detect and respond quickly if any unexpected charges occur.
- The budget sends an email alert to a designated address when spending exceeds 80% of the limit, helping us act before reaching or surpassing the

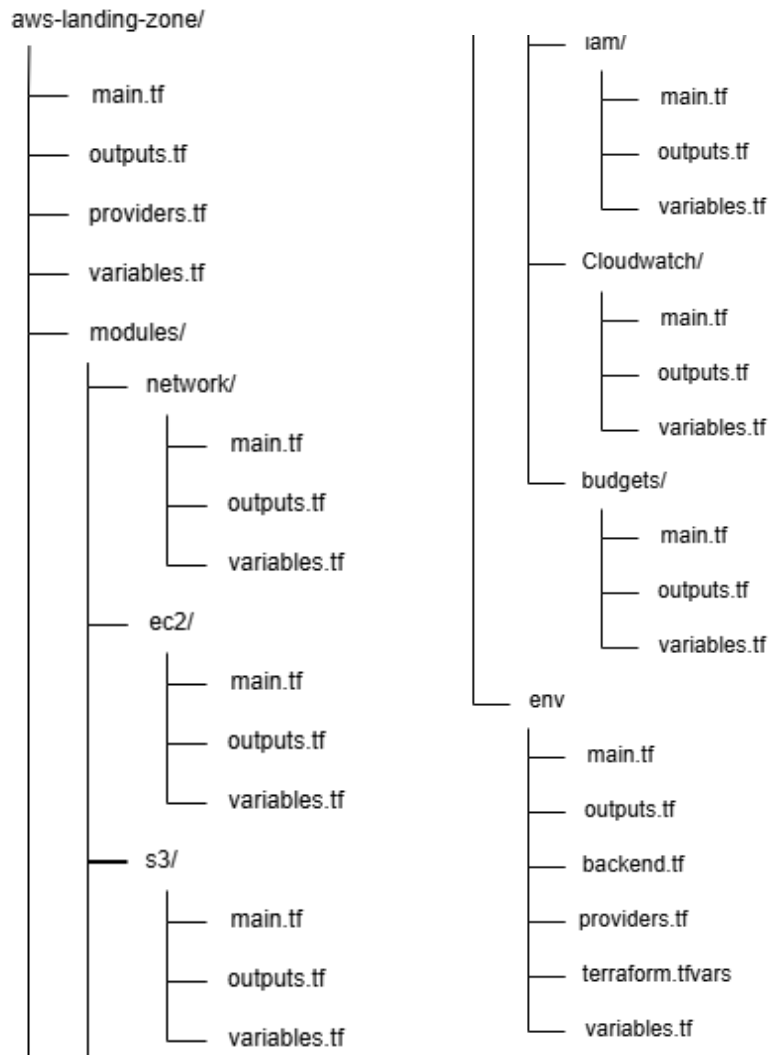
defined threshold. These notifications are configured using actual usage, not forecasts, ensuring alerts are based on real-time data.

- A billing policy was created and assigned to a specific IAM group to allow team members to view billing information, including budgets, cost reports, and usage dashboards. This policy is read-only and doesn't allow access to sensitive actions like changing payment methods or account settings.
- Overall, this setup ensures clear cost visibility from the beginning of the project and provides a safety net to avoid accidental spending beyond the Free Tier.

3. Terraform Resources

3.1. Code Structure

A modular structure was used in the repository:



3.2. Modules

- **Budgets:** Defines a monthly AWS Budget using variables to trigger cost alerts via email, ensuring compliance with the Free Tier.
- **IAM Groups:** Defines a monthly AWS Budget using variables to trigger cost alerts via email, ensuring compliance with the Free Tier.
- **IAM Users:** Creates IAM users and assigns them to appropriate groups according to their category and permissions.
- **IAM Policy:** Defines custom IAM policies, including access control for S3, CloudWatch Logs, and billing.

- VPC: Builds the VPC with public/private subnets, internet gateway, route tables, and security groups. Accepts parameters like cidr_map, suffix, ingress_port_list, and protocol/port maps.
- CloudWatch: Configures CloudWatch Logs for centralized logging, including log groups and retention settings. Integrates with IAM policies to allow controlled access to logs.
- EC2: Deploys EC2 instances with user-data scripts for automated setup (e.g., Apache, MariaDB). Includes security groups, key pairs, and customizable instance types within Free Tier limits.
- S3: Creates S3 buckets with versioning and encryption enabled. Sets up appropriate policies to restrict or allow access as required by applications and users.

3.3. Resources Implemented

3.3.1. Github

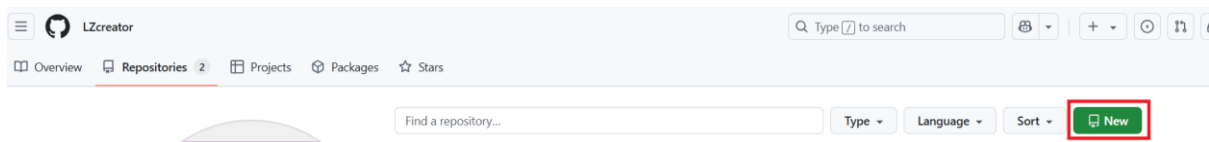
The source code is hosted in a public GitHub repository with two main branches: one for each team member.

Instead of using a traditional GitFlow with main, develop, and feature branches, each team member cloned the repository and worked directly in their own personal branch. All code contributions and updates were pushed to these personal branches.

Once a stable version of the infrastructure was reached, the final changes were merged into the main branch.

GitHub setup steps followed:

- a. Create the github Repository
 - One team member created the repository:
 - Go to [GitHub.com](https://github.com) and sign in with your account or create a new one
 - Click on the "New" button in the top-right corner and select "New repository"



- Enter the repository name and a short description (optional)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).



Owner * Repository name *

 LZcreator ▾ /

Great repository names are short and memorable. Need inspiration? How about **improved-adventure** ?

Description (optional)

- We choose Public as the repository visibility

-
- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- (Optional) Initialize the repository with a README.md and a .gitignore for Terraform

Initialize this repository with:

- ☐ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Terraform ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

We made a .gitignore file for exclude Terraform-specific and sensitive files. This ensures a clean repository free from unnecessary or risky content. Below is the .gitignore used in the project:

```
#Terraform
.terraform/
*.tfstate
*.tfstate.backup
.terraform.lock.hcl

#Private keys
*.pem

#Log or temporary files
crash.log
*.log
*.bak

terraform.exe
```

Explanation of the code:

`.terraform/`: Local cache directory used by Terraform, not needed in version control.

`*.tfstate`, `*.tfstate.backup`: Files that store infrastructure state; may contain sensitive data.

`.terraform.lock.hcl`: Automatically generated lock file for Terraform dependencies.

`*.pem`: Private key files used for SSH access, must not be shared.

`crash.log`, `*.log`, `*.bak`: Temporary or debug files that do not belong in the repository.

`terraform.exe`: Executable used locally, should not be tracked.

- Click "Create repository"

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

① You are creating a public repository in your personal account.

Create repository

b. Clone the Repository Locally

- The same team member then cloned the repository to their local machine:

```
git clone https://github.com/username/aws-landing-zone-project.git
cd aws-landing-zone-project
```

c. Create Working Branches

Two working branches were created directly from the main branch — one for each team member:

```
# Create and push branch for Team Member A
git checkout -b persona-a
git push -u origin persona-a

# Switch back to main
git checkout main

# Create and push branch for Team Member B
git checkout -b persona-b
git push -u origin persona-b
```

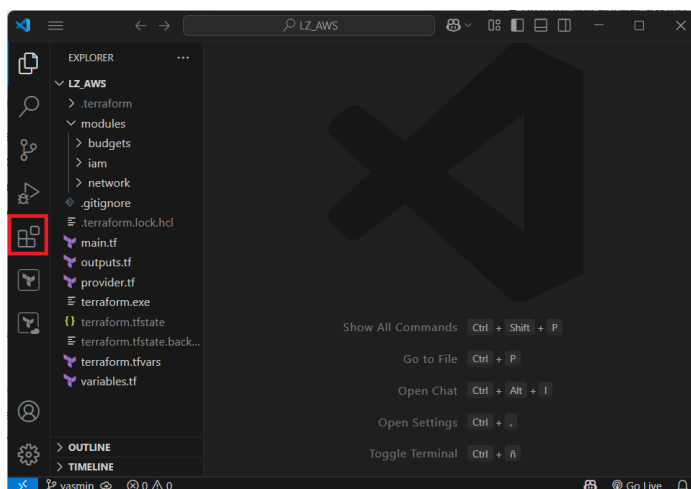
These two branches (persona-a and persona-b) were used for individual development and integration work. Each person cloned the repository and checked out their corresponding branch for local development.

3.3.2. Infracost

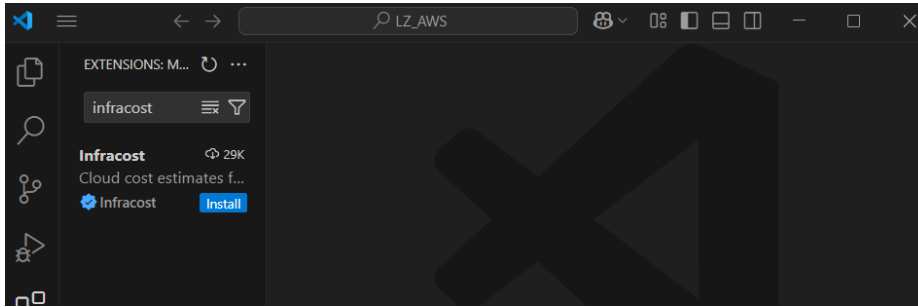
Infracost was integrated to estimate infrastructure costs before applying changes:

Installing and Using Infracost in Visual Studio Code

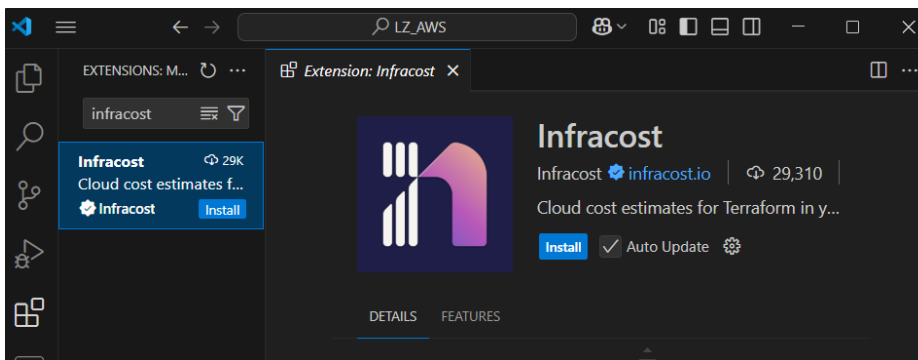
- 1) Install the Infracost Extension:
 - a) Open Visual Studio Code
 - b) Go to the Extensions Marketplace (Ctrl+Shift+X or click the Extensions icon)



c) Search for “Infracost”



d) Click Install on the official Infracost extension



2) Authenticate the Extension:

- a) Open a .tf file
- b) A prompt will appear asking you to log in
- c) Click Log in, complete the browser login, and connect your account

3) Estimate Resource Costs:

- a) Open a .tf file with AWS resources
- b) Hover over resources to view cost estimates
- c) Open the sidebar to see a cost breakdown per resource/module

4. Networking

4.1. Network Structure

The initial design of our network architecture prioritizes a straightforward and cost-effective structure suitable for development environments operating within AWS Free Tier limits. To balance public accessibility and private resource isolation, we created a Virtual Private Cloud (VPC) divided into a public and a private subnet. This setup allows for exposing necessary services to the internet while keeping internal processes and data isolated from direct external access. An Internet Gateway provides public instances with outbound internet connectivity, while private instances remain isolated unless configured otherwise.

- VPC Configuration:
 - Main VPC (Virginia):
 - Core network structure where all instances are deployed.
 - Name follows the pattern vpc_virginia-<suffix>.
 - Located in the us-east-1 region.
 - CIDR Block: 10.0.0.0/16
 - DNS resolution and hostname support are enabled for internal name resolution.
- Subnet Configuration:
 - Public Subnet:
 - Used to deploy instances that require direct internet access.
 - Configured to assign public IPs automatically upon launch.
 - Located in availability zone us-east-1a.
 - Name follows the pattern Public_subnet-<suffix>.
 - CIDR Block: 10.0.1.0/24
 - Ideal for hosting front-facing services like web servers.
 - Private Subnet:
 - Used for instances that do not require direct internet access.
 - Offers additional security by isolating backend services from public exposure.
 - Located in the same availability zone for simplicity and cost optimization.
 - Name follows the pattern Private_subnet-<suffix>.
 - CIDR Block: 10.0.2.0/24
- Internet Gateway:
 - Attached to the VPC to allow internet access for public instances.
 - Name follows the pattern igw_vpc-<suffix>.
- Route Table Configuration:
 - A dedicated route table is created for the public subnet.
 - Configured with a route to direct all outbound traffic (0.0.0.0/0) through the Internet Gateway.
 - Associated only with the public subnet to maintain isolation of the private subnet.
 - Name follows the pattern public_crt-<suffix>.
- Security Group:
 - Public Instance Security Group:
 - Attached to instances in the public subnet.
 - Inbound Rules:
 - HTTP (TCP/80): Allows access from all IP addresses.

- SSH (TCP/22): Enables secure terminal access from all IP addresses.
- Outbound Rules:
 - Allows all outbound traffic to any destination (IPv4 and IPv6).
- Name follows the pattern: Public_Instance_SG-<suffix>.
- Provides basic, open access useful during development and initial configuration phases.

The resulting architecture is simple yet functional, providing a solid foundation for scalable infrastructure. The public subnet supports services that must interact with external users or resources, while the private subnet can host internal services shielded from public exposure. The setup maintains security through network segmentation and the application of Security Groups, and it stays within Free Tier boundaries, making it ideal for development or testing use cases.

5. Documentation

- [Terraform registry](#)
- Udemy
 - [Terraform: De principiante a certificado 2024](#)
 - [Ultimate AWS Certified Cloud Practitioner CLF-C02 2025](#)
 - [HashiCorp Certified: Terraform Associate - Hands-On Labs](#)
 -
- [Single account landing zone](#)
- [Deloitte Guardrails](#)

Deloitte.

About Deloitte

As used in this communication, 'Deloitte' means Deloitte Touche Tohmatsu Limited and its member firms.

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private company limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. Please see www.deloitte.com/about for a more detailed description of DTTL and its member firms.

Deloitte provides audit, consulting, financial advisory, risk management, tax and related services to public and private clients spanning multiple industries. With a globally connected network of member firms in more than 150 countries and territories, Deloitte brings world-class capabilities and high-quality service to clients, delivering the insights they need to address their most complex business challenges. Deloitte's more than 220,000 professionals are committed to making an impact that matters.

This communication is for internal distribution and use only among personnel of Deloitte Touche Tohmatsu Limited, its member firms, and their related entities (collectively, the "Deloitte Network"). None of the Deloitte Network shall be responsible for any loss whatsoever sustained by any person who relies on this communication.

© 2022. For information, contact Deloitte Touche Tohmatsu Limited.