LOGO MISSING

Security classification: INTERNAL

Document code:

Last updated by: Khoa Nguyen-Dinh

Effective date: Dec 10, 2025

Version: 1.0

Template ID: ODT_Base_Template

DOCUMENT CONTROL

Version	Change Description	Changed By	Date	Approved By	Date
1.0	Khởi tạo tài liệu	KhoaND	2025-12-10		2025-12-10

TABLE OF CONTENT

INDEX OF TABLES

INDEX OF FIGURES

1. Overview

1.1. Purpose

This document provides comprehensive guidance for deploying an EC2 instance in the AWS nonprod testing environment using Terraform Infrastructure as Code (IaC). The instance will host Apache Kafka, Redis, and related services with proper networking, security controls, and access management. The deployment follows AWS best practices and maintains consistency with the existing MAX project infrastructure.

1.2. Scope

This guide encompasses the complete deployment lifecycle for other services infrastructure using Terraform. It provides step-by-step instructions for provisioning compute resources, configuring security controls, setting up networking integration, and managing the deployment through the S3 remote state backend.

The scope does NOT include: - VPC and networking infrastructure creation - Load balancing or auto-scaling setup
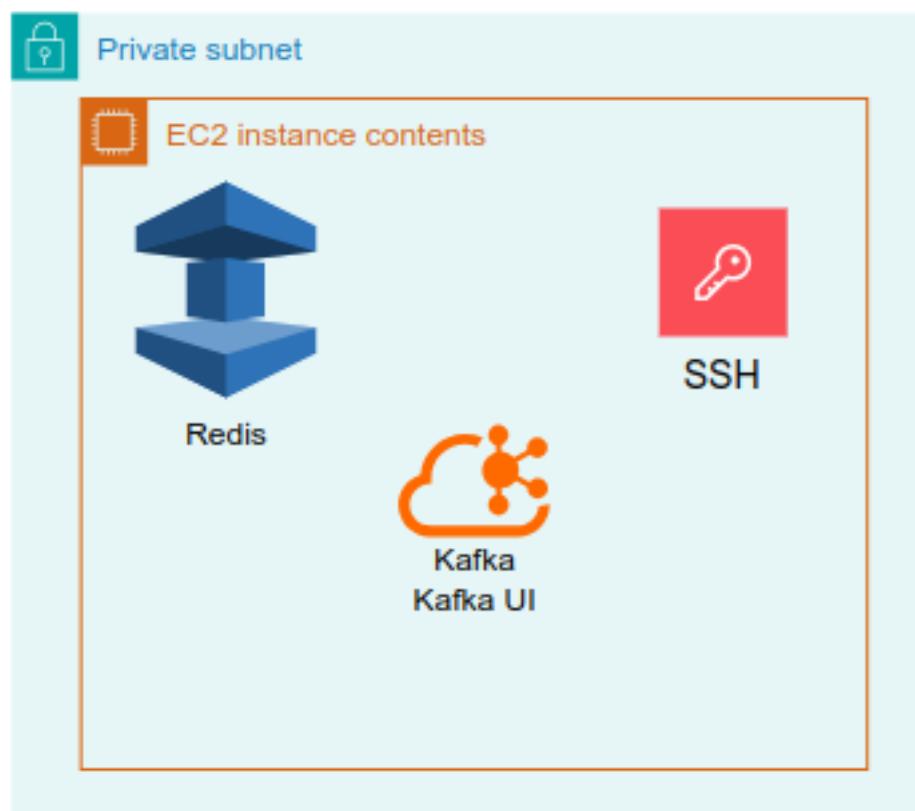
1.3. Abbreviations and Definitions

- **EC2**: Elastic Compute Cloud - AWS cloud computing service that provides virtual servers. This is where Kafka, Redis, and related services run.
- **VPC**: Virtual Private Cloud - A private virtual network in AWS that allows users to create a separate network space with complete control over IP addressing and routing.
- **AMI**: Amazon Machine Image - A pre-configured virtual machine image used to launch EC2 instances. Contains the operating system, applications, and services needed.
- **CIDR**: Classless Inter-Domain Routing - An IP routing method that allows for efficient allocation and management of IP address ranges using prefix notation.
- **IAM**: Identity and Access Management - AWS service that manages user access and permissions for AWS resources.
- **SSH**: Secure Shell - A cryptographic network protocol for secure data communication and remote command execution.
- **SSM**: Systems Manager - AWS service that provides a unified interface for managing and automating operational tasks across AWS resources.
- **EBS**: Elastic Block Store - A scalable block storage service for use with EC2 instances, providing persistent storage.
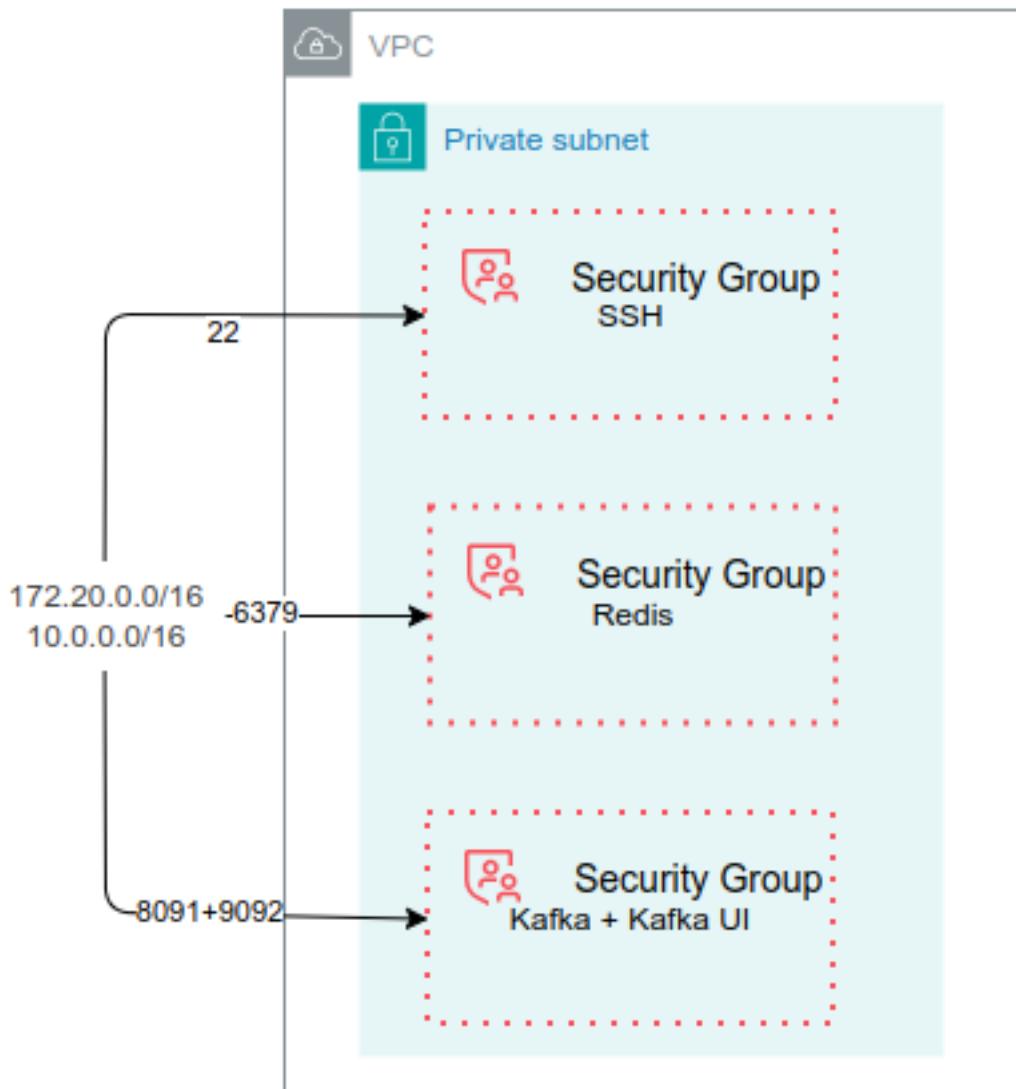
2. Architecture

2.1. Diagram

Illustration 1: *Architecture EC2*

Illustration 2: Architecture Network

2.2. Directory Structure

```
/nonprod/testing/other-services/
   .terraform.lock.hcl
   main.tf
   versions.tf
   outputs.tf
```

3. Requirements

The Terraform configuration must meet the following requirements:

1. **Network Integration**: The EC2 instance must be placed in the same network as other services for proper communication and resource sharing. (managed in `live/nonprod/general/networking`)
2. **Operating System Image**: Use the latest Ubuntu AMI available at the time of deployment for current security patches and features.
3. **CIDR Ranges**: Configure access from two CIDR ranges:
   - CIDR block containing current resources

- CIDR block of the MSS project
4. **Resource Tagging**: Apply consistent tags to all resources for management and governance:
   - **Environment**: Deployment environment designation
   - **Project**: Project identifier
   - **Owner**: Responsible party or team
   - **Service**: Service name
5. **Security Group Configuration**: Security groups must be properly configured to restrict access to only the required ports from authorized CIDR ranges, enhancing overall security posture.
6. **Outbound Internet Access**: The EC2 instance must have outbound internet connectivity to download OS updates, security patches, and required software packages.
7. **Instance Naming Convention**: EC2 instances must follow the established naming convention for easy identification and management.
8. **Instance Type Selection**: Choose an appropriate instance type that balances resource requirements for Kafka, Redis, and related services while optimizing for cost efficiency.
9. **IAM Role Configuration**: Configure the IAM role to ensure the EC2 instance has necessary permissions to access AWS Systems Manager and other required AWS services.
10. **Private Subnet Deployment**: Deploy the EC2 instance in a private subnet to enhance security and prevent direct internet exposure.
11. **SSH Key Pair**: Reuse the existing key pair for SSH access, ensuring consistency across the nonprod environment.

4. Prerequisites

Before running the Terraform configuration, verify the following prerequisites are met:

4.1. Terraform and AWS Provider Requirements

Name	Version
terraform	>= 1.5.0
aws	>= 5.0

Table 1: *abc*

4.1.1. Providers

Name	Version
aws	6.22.1
terraform	n/a

Table 2: *bcd*

4.2. Prerequisites

1. **VPC and Subnets**: Verify that the VPC and required subnets exist and are properly configured.

2. **VPC Configuration**: Confirm that VPC ID and subnet name tags match the Terraform configuration.

3. **IAM Permissions**: Ensure your AWS credentials have all required EC2, S3, and IAM permissions for resource creation.

4. **AMI Availability**: Verify that the specified AMI ID is valid and available in the region.

5. **AWS Systems Manager Role**: Verify that the AmazonSSMRoleForInstancesQuickSetup IAM role exists in your AWS account.

6. **SSH Key Pair**: Verify that the key pair exists in the region for EC2 instance access.

7. **Terraform Module**: Verify that the EC2 module is available at the correct path.

8. **CIDR Range Validation**: Confirm that both CIDR ranges are properly configured and accessible.

9. **AWS CLI Connectivity**: Ensure your machine has network connectivity to AWS APIs and can authenticate with your AWS credentials.

10. **Remote State Backend**: Verify that the S3 backend and networking state file exist for remote state management.

5. Deployment Instructions

5.1. Deploy Infrastructure

5.1.1. Step 1: Initialize Terraform Workspace

Navigate to the Terraform directory and initialize the workspace:

```
cd /nonprod/testing/other-services
terraform init
```

This command downloads required providers and initializes the backend.

5.1.2. Step 2: Validate Configuration Syntax

Verify the Terraform configuration syntax:

```
terraform validate
```

Expected output: `Success! The configuration is valid.`

5.1.3. Step 3: Format Code (Optional)

Format Terraform files for consistency:

```
terraform fmt -recursive
```

5.1.4. Step 4: Generate Execution Plan

Create a detailed plan of infrastructure changes:

```
terraform plan -out=tfplan
```

Review the plan:

```
terraform show tfplan
```

5.1.5. Step 5: Apply Configuration

Deploy infrastructure according to the plan:

```
terraform apply tfplan
```

Expected output: `Apply complete! Resources: 9 added, 0 changed, 0 destroyed.`

5.1.6. Step 6: Retrieve Outputs

Display infrastructure outputs:

```
terraform output
```

5.1.7. Step 7: Retrieve SSH Private Key

Get the private key for SSH access:

```
RETRIEVE_CMD=$(terraform output -raw retrieve_private_key_command)
eval $RETRIEVE_CMD
```

5.1.8. Step 8: Connect to EC2 Instance

Establish SSH connection to the instance:

```
INSTANCE_IP=$(terraform output -raw instance_private_ip)
ssh -i max.dev.key.01.pem ec2-user@$INSTANCE_IP
```

5.2. Redeploying After Accidental Destruction

If the EC2 instance is accidentally terminated outside of Terraform:

```
# Refresh Terraform state
terraform refresh

# Check state
terraform state list

# Reapply to recreate the instance
terraform apply -auto-approve
```

5.3. Destroying and Recreating Infrastructure

To completely remove and redeploy:

```
# Review resources to be destroyed
terraform plan -destroy -out=tfplan_destroy
terraform show tfplan_destroy

# Destroy all resources
terraform apply tfplan_destroy

# Recreate from scratch
terraform apply -auto-approve
```

Critical Warning: This will permanently delete all resources including the EC2 instance and security groups.

5.4. Applying Partial Changes (Targeted Apply)

To apply changes to specific resources only:

```
# Example: Update only the security group without touching EC2
terraform apply -target=aws_security_group.other_services

# Example: Update specific security group rule
terraform apply -target='aws_security_group_rule.allow_kafka'

# Update EC2 module only
terraform apply -target='module.ec2'
```

Note: Use targeted applies with caution as it may cause inconsistencies.

5.5. Backup Current State Before Major Changes

Before making significant changes, backup the current state:

```
# Backup current state locally
cp terraform.tfstate terraform.tfstate.backup.$(date +%Y%m%d_%H%M%S)


# Or view current state
terraform state show


# Verify remote state in S3
aws s3 cp s3://meperia-edi/terraform/testing/other-services/terraform.tfstate terraform.tfst
```

5.6.   Reviewing and Approving Changes (Plan Review Process)

For change control and approval:

```
# Generate detailed plan in JSON format for review
terraform plan -out=tfplan -json > tfplan.json


# Generate human-readable summary
terraform plan -out=tfplan > tfplan.txt


# Share plan files for review and approval before applying
# Once approved, proceed with apply
terraform apply tfplan
```

6.   Appendix

6.0.1. Modules

Name	Source	Version
ec2	../../../../modules/ec2	n/a

6.0.2. Resources

Name	Type
aws_security_group.other_services	resource
aws_security_group_rule.allow_kafka	resource
aws_security_group_rule.allow_kafka_ui	resource
aws_security_group_rule.allow_redis	resource
aws_security_group_rule.allow_ssh	resource
aws_subnet.private_app_a	data source
terraform_remote_state.networking	data source

6.0.3. Inputs

No inputs required.

6.0.4. Outputs

Name	Description
ebs_volume_id	ID of the attached EBS volume
effective_key_name	The key name associated with the EC2 instance
instance_id	ID of the created EC2 instance
instance_private_ip	Private IP of the EC2 instance
private_key_parameter_name	SSM Parameter name storing the generated private key (if generated)
retrieve_private_key_command	AWS CLI command to retrieve the private key from SSM Parameter Store

6.1. Troubleshooting

6.1.1. Common Issues

- **Backend State Not Found**: Verify S3 bucket and networking state file exist
- **VPC/Subnet Not Found**: Confirm VPC ID and subnet name tags match configuration
- **IAM Permissions Denied**: Verify IAM role has required permissions
- **AMI Not Available**: Verify AMI ID is valid and available in region
- **Key Pair Not Found**: Ensure SSH key pair exists in AWS region

6.2. References

For additional information, refer to:

- Terraform AWS Provider Documentation
- AWS EC2 User Guide
- AWS VPC Best Practices
- Security Group Configuration Guide