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

Deploy EC2 instance with Kafka, Redis, and related services in AWS nonprod testing environment with proper networking, security, and access controls.

1.2. Scope

This guide covers the deployment of infrastructure components including:

- EC2 instance with appropriate resource allocation
- Security groups with specific port access rules
- Network integration with existing VPC infrastructure

---

2. Architecture

2.1. Diagram

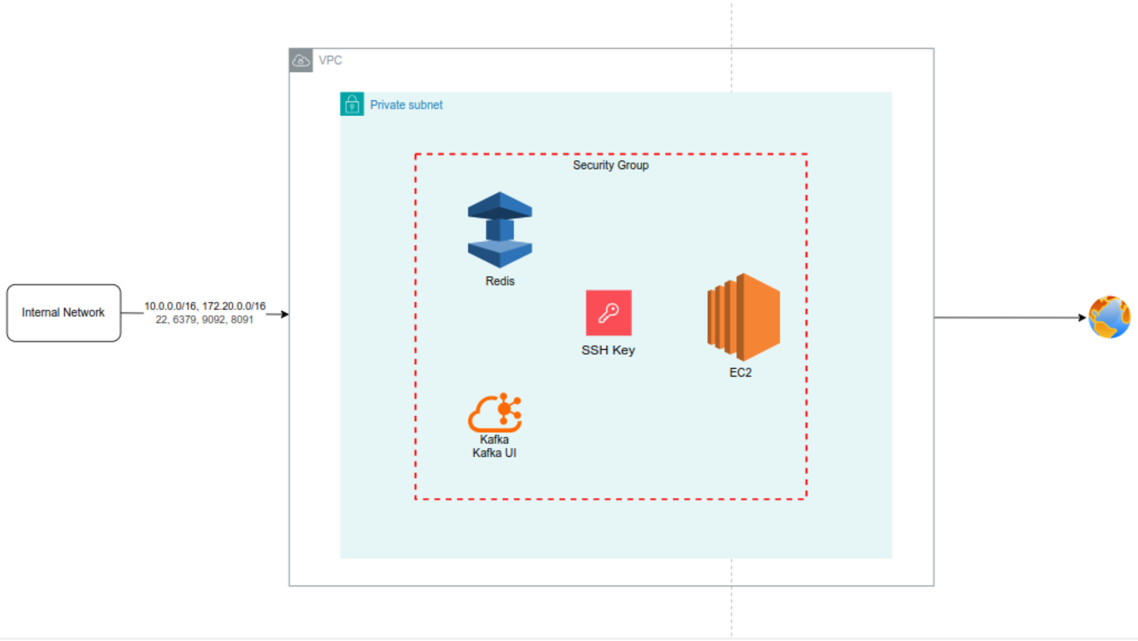

Figure 1: *Architecture Diagram*

2.2. Description

- EC2 Instance: Hosts Kafka, Redis, and related services using a golden AMI with pre-installed software.
- Security Group: Controls inbound and outbound traffic to the EC2 instance, allowing SSH, Redis, and Kafka access from specified CIDR ranges.
- Networking: Integrates with existing VPC and private subnets for secure communication.

2.3. Directory Structure

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

---

3. Requirements

The Terraform configuration requires the following components:

3.1. Data Sources

- **data "terraform_remote_state" "networking"**: Synchronizes state information with networking infrastructure components in the system.

3.2. Variables

- **allow_cidr_range**: CIDR blocks for inbound access control
  - `172.20.0.0/16`: CIDR block of the current VPC hosting these resources
  - `10.0.0.0/16`: CIDR block of the MSS project VPC within the same AWS cloud environment
- **common_tags**: Tags for resource management and governance
- **ami_id**: Golden AMI image ID (Ubuntu-based with pre-installed services)

3.3. AWS Resources

3.3.1. Networking Components

- **data "aws_subnet" "private_app_a"**: Retrieves private subnet using VPC ID from remote state
  - Filter by VPC ID from networking state
  - Filter by subnet name tag: `*private-app-a`
- **resource "aws_security_group" "other_services"**: Security group for service access control
  - Declares name, description, and associated VPC
  - Allows unrestricted outbound access (egress)
  - Applies project-wide and service-specific tags

3.3.2. Security Rules

- **resource "aws_security_group_rule" "allow_ssh"**: Allows SSH access from defined CIDR ranges
- **resource "aws_security_group_rule" "allow_redis"**: Allows Redis port access from defined CIDR ranges

- **resource "aws_security_group_rule" "allow_kafka"**: Allows Kafka port access from defined CIDR ranges
- **resource "aws_security_group_rule" "allow_kafka_ui"**: Allows Kafka UI port access from defined CIDR ranges

3.3.3. Compute Resources

- **module "ec2"**: Deploys EC2 instance with configuration
  - Source: Pre-defined EC2 module
  - Instance name: Assigned to identify the instance
  - Instance type: Selected for workload requirements
  - AMI ID: Golden image with pre-installed services
  - Subnet ID: Private subnet for application tier
  - Security group IDs: Attached for traffic control
  - IAM instance profile: Role for AWS Systems Manager communication
  - Key pair: SSH key for instance access
  - Tags: Resource identification and governance

---

4. Prerequisites

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

4.1. AWS Account & Credentials

Required IAM Permissions for deployment:

- `ec2:CreateSecurityGroup`
- `ec2:CreateSecurityGroupRule`
- `ec2:RunInstances`
- `ec2:CreateTags`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeInstances`
- `s3:GetObject`
- `iam:PassRole`

4.2. S3 Backend & Networking State

Verify S3 bucket and remote state files exist:

```
aws s3api head-bucket --bucket meperia-edi
aws s3 ls s3://meperia-edi/terraform/networking/nonprod.tfstate
aws s3 cp s3://meperia-edi/terraform/networking/nonprod.tfstate - | jq '.outputs | {vpc_id,
```

Expected output should include `vpc_id` and `private_app_subnet_ids`.

4.3. VPC & Subnets

Verify VPC and private subnets exist:

```
aws ec2 describe-vpcs --filters Name=cidr,Values=172.20.0.0/16 --region us-east-1
aws ec2 describe-subnets --filters Name=vpc-id,Values=vpc-XXXXXXXX --region us-east-1 \
  --query 'Subnets[?MapPublicIpOnLaunch==`false`].{SubnetId:SubnetId,CidrBlock:CidrBlock}'
```

4.4. AMI Validation

Verify the golden AMI with Kafka and Redis pre-installed:

```
aws ec2 describe-images --image-ids ami-0360c520857e3138f --region us-east-1
```

4.5. IAM Role Verification

Verify AWS Systems Manager role is available:

```
aws iam get-role --role-name AmazonSSMRoleForInstancesQuickSetup
```

4.6. EC2 Key Pair

Verify the SSH key pair exists:

```
aws ec2 describe-key-pairs --key-names max.dev.key.01 --region us-east-1
```

4.7. EC2 Module

Verify module files are available:

```
ls -la /nonprod/testing/../../modules/ec2/
```

Should contain: `main.tf`, `variables.tf`, `outputs.tf`

4.8. CIDR Range Validation

Verify CIDR ranges are valid:

```
aws ec2 describe-vpcs --region us-east-1 --query 'Vpcs[].{VpcId:VpcId,CidrBlock:CidrBlock}'
```

Current VPC: `172.20.0.0/16` and MSS project VPC: `10.0.0.0/16`

4.9. Network Connectivity

Verify AWS connectivity:

```
ping 8.8.8.8
curl -I https://sts.amazonaws.com/
```

5. Deployment Instructions

5.1. Step-by-Step Procedure

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
```

---

6. Appendix

6.1. Terraform Documentation

6.1.1. Requirements

Name	Version
terraform	>= 1.5.0
aws	>= 5.0

6.1.2. Providers

Name	Version
aws	6.22.1
terraform	n/a

6.1.3. Modules

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

6.1.4. 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

6.1.5. Inputs

No inputs required.

6.1.6. 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

6.2. Troubleshooting

6.2.1. 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

6.3. References

For additional information, refer to:

- Terraform AWS Provider Documentation
- AWS EC2 User Guide

- AWS VPC Best Practices
- Security Group Configuration Guide