DEPARTMENT OF SOFTWARE ENGINEERING

"PROJECT 4"

Secure Multi-Tier Network Architecture

SUBMITTED TO:

Sir Waqas Saleem

COURSE TITLE:

Cloud Computing

SUBMITTED BY:

Maham Saleem(2023-BSE-033)

Ainun-Noor (2023-BSE-003)

Kainat Shakeel (2023-BSE-029)

1. Executive Summary

This project presents the design and implementation of a secure multi-tier network architecture on Amazon Web Services (AWS) using Terraform and Ansible, strictly within the tools and concepts covered in the Cloud Computing course. The objective is to demonstrate how a production-style cloud environment can be securely designed, provisioned, configured, and verified using Infrastructure as Code (IaC) and configuration management.

The implemented solution follows a three-tier architecture consisting of a Web Tier, Application Tier, and Management Tier. Public access is limited to the web tier only, while the application tier remains private with no direct exposure to the internet. A bastion host is deployed in the management subnet and serves as the only entry point for administrative SSH access, reducing the attack surface.

Terraform provisions the VPC, subnets, route tables, Internet Gateway, Security Groups, and EC2 instances using a modular structure and environment-specific variable files for dev, staging, and production. Ansible automates server hardening (non-root admin user, key-only SSH, root login disabled, secure file permissions), deploys a simple application in the private tier, configures Nginx reverse proxy on the web tier, and collects logs as security evidence in the repository under docs/logs/.

Finally, the project includes security verification tests and a compliance checklist. Evidence is captured through screenshots and collected log files to demonstrate segmentation, least privilege access, bastion host SSH flow, and secure SSH configuration.

2. Architecture Design

2.1 Network Architecture Overview

The infrastructure is deployed inside a single AWS Virtual Private Cloud (VPC) and segmented into three clearly defined tiers:

Web Tier (Public Subnets): Hosts Nginx reverse proxy servers that accept HTTP traffic from external clients.

Application Tier (Private Subnets): Hosts backend application servers listening on port 8080. These instances have no public IP addresses.

Management Tier (Management Subnet): Hosts a bastion (jump) server that acts as the only SSH entry point to the environment.

Traffic Flow Summary:

• External Clients -> Web Tier (HTTP 80)

- Web Tier -> Application Tier (HTTP 8080)
- Admin IP -> Bastion Host (SSH 22)
- Bastion Host -> Web/App Servers (SSH 22)

2.2 Security Architecture

Security is enforced using Security Groups only following the least privilege principle. All tiers are isolated in dedicated subnets. Public inbound access is restricted to HTTP (port 80) on the web tier. SSH access to web and app servers is not allowed from the internet; it is permitted only from the bastion host. Security Group-to-Security Group references are used to enforce strictly controlled traffic between tiers.

3. Terraform Implementation

Terraform is used to provision all required AWS resources. The infrastructure is organized using reusable modules under terraform/modules/ (network, security, compute) and wired together from the root terraform/main.tf. Environment-specific configuration is provided using tfvars files for dev, staging, and production.

3.1 Network Module

The network module provisions the VPC, subnets for each tier, Internet Gateway, and route tables. Public subnets host the web tier and are routed to the Internet Gateway. Private subnets host the application tier and do not require a default internet route for this project (documented limitation). All subnets are tagged by tier (Tier=web, Tier=app, Tier=management).

3.2 Security Module

The security module creates least-privilege Security Groups for the web, app, and bastion tiers. Inbound and outbound rules are implemented using Security Group references to ensure only required traffic is allowed.

3.3 Compute Module

The compute module provisions EC2 instances for the web tier, application tier, and bastion host. Web instances are deployed in public subnets with Web SG. App instances are deployed in private subnets with App SG and have no public IPs. The bastion host is deployed in the management subnet with Bastion SG. Instances are tagged with Project, Environment, Tier, Role, and Name.

3.4 Root Terraform Configuration & Environments

At the root level, terraform/main.tf configures the AWS provider and backend and instantiates the network, security, and compute modules. Module outputs (VPC ID, subnet IDs, SG IDs) are

wired into dependent modules. Helpful outputs are defined for bastion public IP, web public endpoint(s), and app private IPs to support Ansible inventory and verification.

Environment-specific tfvars files are used to configure CIDRs (admin IP, allowed HTTP), instance sizes, and instance counts for dev, staging, and production environments. A terraform.tfvars.example file is provided for safe sharing without secrets.

4. Ansible Implementation

Ansible is used to configure and secure the provisioned EC2 instances. Roles and playbooks are organized under ansible/ to support repeatable execution across environments. The implementation includes dynamic inventory (aws_ec2), server hardening, application deployment, Nginx reverse proxy configuration, and log collection for evidence.

4.1 Inventory and Global Configuration

Dynamic inventory is configured using the aws_ec2 plugin. Instances are discovered and grouped by tags (Tier=web/app/management). ansible.cfg standardizes inventory, privilege escalation (sudo), and SSH behavior to ensure consistent execution.

4.2 Server Hardening

A dedicated hardening role is applied to all servers. It creates a non-root admin user (e.g., deployer) with sudo access, installs common packages (vim, curl, git), and hardens SSH by disabling root login and password authentication (key-only SSH). It also enforces secure permissions on SSH configuration files and restarts the SSH service via handlers.

4.3 Nginx Reverse Proxy & Application Tier

The application role deploys a simple backend service listening on port 8080 in the private subnets (app tier). The Nginx reverse proxy role installs and configures Nginx on the web tier to forward incoming HTTP requests on port 80 to the app tier on port 8080. A /health endpoint is provided for basic service checks.

4.4 Log Collection

For centralized evidence collection within course scope, Ansible fetches key logs from all servers and stores them in the repository under docs/logs/. Collected evidence includes SSH authentication logs (/var/log/auth.log or /var/log/secure) and Nginx access/error logs from the web tier. This provides proof of hardening and runtime behavior without using CloudWatch/rsyslog.

5. Security Testing & Compliance Checklist

5.1 Security Testing Results

Security tests were performed to validate segmentation, least privilege Security Groups, and bastion host access flow. The following tests and expected outcomes were documented using screenshots:

- **Attempt SSH to an app server directly from local machine -** FAIL (expected) - no public IP and SG restriction
- **SSH to bastion host from admin IP CIDR -** SUCCESS
- **SSH from bastion to web/app servers using private IPs -** SUCCESS
- **Access web tier public URL and verify Nginx forwards to app tier -** SUCCESS
- **Verify SSH hardening settings in sshd_config -** PermitRootLogin no; PasswordAuthentication no

5.2 Security Compliance Checklist

The following checklist summarizes the implemented security controls and their verification evidence:

☐ Root SSH login disabled on all servers

☐ SSH password authentication disabled (key-only SSH)

☐ Non-root admin user (deployer) created and configured with SSH keys

☐ Application servers have no public IP addresses

☐ Web servers expose only HTTP (80) publicly

☐ SSH access to web/app servers allowed only via bastion host

☐ Nginx reverse proxy forwards requests to app tier on port 8080

☐ Evidence logs collected and stored under docs/logs/

6. Challenges & Solutions

During implementation, several common infrastructure and configuration issues were encountered. The following summarizes the main challenges and how they were resolved:

• **SSH access risk after hardening:** Applied changes gradually and tested SSH after each run. Ensured deployer key access before disabling password authentication.

• **Nginx 502/504 errors:** Verified app service was running on port 8080, checked Nginx upstream configuration, and confirmed SG allows web -> app traffic on 8080.

• **Different SSH log paths across OS images:** Handled both /var/log/auth.log (Ubuntu) and /var/log/secure (Amazon Linux) using variables and ignore_errors in fetch tasks.

• **Dynamic inventory grouping issues:** Validated instance tags (Project, Environment, Tier) and ensured aws_ec2 inventory filters matched exactly.

7. Conclusion & Future Improvements

This project successfully implemented a secure three-tier AWS network architecture using Terraform and Ansible within course scope. Network segmentation, least privilege Security Groups, bastion host access, SSH hardening, and Nginx reverse proxy were implemented and verified with evidence.

Future improvements (out of scope for this course) could include enabling HTTPS/TLS on Nginx, adding autoscaling for web/app tiers, centralized monitoring/alerting, and CI/CD pipelines for automated deployments.

Part 1: Git Repository Setup

1.1 Repository Structure

Step 1 — Go into your new repo

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ pwd
/workspaces/Project-4-Secure-MultiTier-Network
```

Step 2 — Create all folders

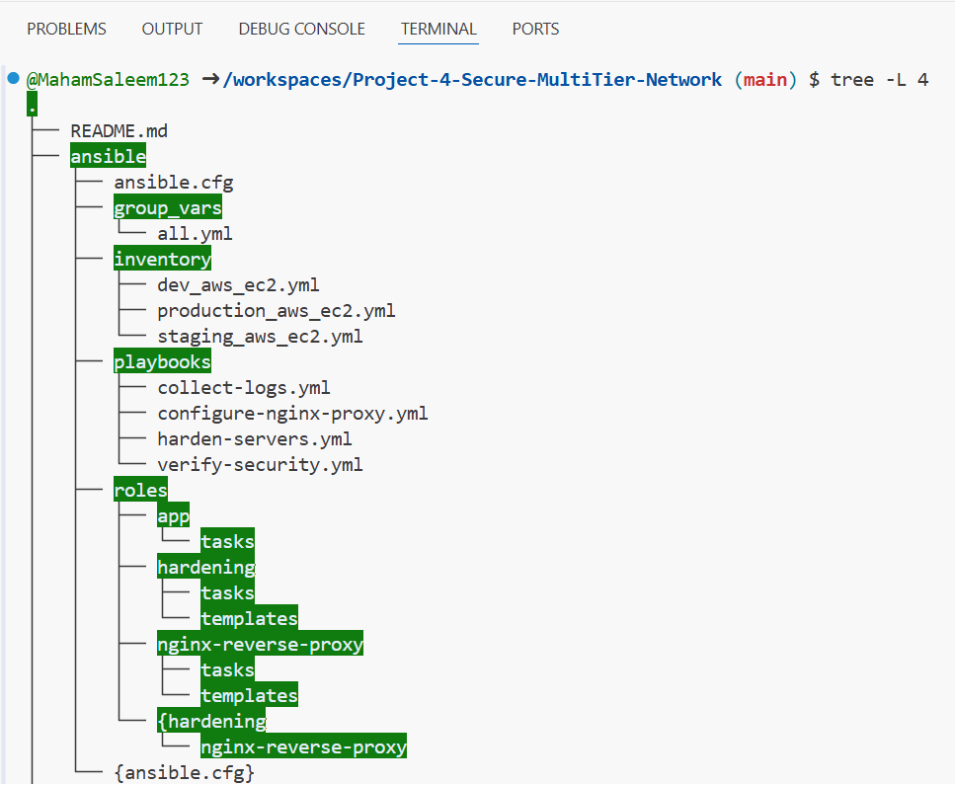
```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ mkdir -p terraform/environments \
terraform/modules/{network,security,compute} \
ansible/{inventory,playbooks,roles/{hardening/nginx-reverse-proxy/app},group_vars} \
ansible/roles/hardening/{tasks,templates} \
ansible/roles/nginx-reverse-proxy/{tasks,templates} \
ansible/roles/app/tasks \
app/api \
docs/logs
```

Step 3 — Create all required files

```
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network (main) $ touch README.md .gitignore \
terraform/{main.tf,variables.tf,outputs.tf,backend.tf,terraform.tfvars.example} \
terraform/environments/{dev.tfvars,staging.tfvars,production.tfvars} \
terraform/modules/network/{main.tf,variables.tf,outputs.tf} \
terraform/modules/security/{main.tf,variables.tf,outputs.tf} \
terraform/modules/compute/{main.tf,variables.tf,outputs.tf} \
ansible/ansible.cfg \
ansible/inventory/{dev_aws_ec2.yml,staging_aws_ec2.yml,production_aws_ec2.yml} \
ansible/playbooks/{harden-servers.yml,configure-nginx-proxy.yml,collect-logs.yml,verify-security.yml} \
ansible/roles/hardening/tasks/main.yml \
ansible/roles/hardening/templates/sshd_config.j2 \
ansible/roles/nginx-reverse-proxy/tasks/main.yml \
ansible/roles/nginx-reverse-proxy/templates/nginx.conf.j2 \
ansible/roles/app/tasks/main.yml \
ansible/group_vars/all.yml \
docs/{architecture.md,security-architecture.md,security-checklist.md,troubleshooting.md}
```

Step 4 — Verify structure (for screenshot)

project4_part1_repository_structure.png



```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ tree -L 4
    └── {ansible.cfg}
├── app
│   └── api
├── docs
│   ├── architecture.md
│   ├── logs
│   ├── security-architecture.md
│   ├── security-checklist.md
│   └── troubleshooting.md
└── terraform
    ├── backend.tf
    ├── environments
    │   ├── dev.tfvars
    │   ├── production.tfvars
    │   └── staging.tfvars
    ├── main.tf
    ├── modules
    │   ├── compute
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   ├── network
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   └── security
    │       ├── main.tf
```

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ tree -L 4
    │   └── security
    │       ├── main.tf
    │       ├── outputs.tf
    │       └── variables.tf
    ├── outputs.tf
    ├── terraform.tfvars.example
    └── variables.tf

26 directories, 32 files
```

1.3 .gitignore Configuration

Step 5 — Open .gitignore
project4_part1_gitignore.png

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 GNU nano 7.2                                                    .gitignore
# Terraform
**/.terraform/*
*.tfstate
*.tfstate.*
*.tfvars
!*.tfvars.example
crash.log
crash.*.log
.terraformrc
terraform.rc

# Ansible
*.retry
*.secret

# AWS credentials
.aws/
*.pem
*.key

# IDE
.vscode/
.idea/
*.swp
```

Step 6 — Check git status

project4_part1_git_status_clean.png

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        ansible/
        docs/
        terraform/

nothing added to commit but untracked files present (use "git add" to track)
```

1.1 – Initial Commit

Step 7 — Commit

project4_part1_initial_commit.png

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git add .
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git commit -m "Initial project structure for Secure Multi-Tier Network"
[main e0b8e30] Initial project structure for Secure Multi-Tier Network
 34 files changed, 43 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 ansible/ansible.cfg
 create mode 100644 ansible/group_vars/all.yml
 create mode 100644 ansible/inventory/dev_aws_ec2.yml
 create mode 100644 ansible/inventory/production_aws_ec2.yml
 create mode 100644 ansible/inventory/staging_aws_ec2.yml
 create mode 100644 ansible/playbooks/collect-logs.yml
 create mode 100644 ansible/playbooks/configure-nginx-proxy.yml
 create mode 100644 ansible/playbooks/harden-servers.yml
 create mode 100644 ansible/playbooks/verify-security.yml
 create mode 100644 ansible/roles/app/tasks/main.yml
 create mode 100644 ansible/roles/hardening/tasks/main.yml
 create mode 100644 ansible/roles/hardening/templates/sshd_config.j2
 create mode 100644 ansible/roles/nginx-reverse-proxy/tasks/main.yml
 create mode 100644 ansible/roles/nginx-reverse-proxy/templates/nginx.conf.j2
 create mode 100644 ansible/{ansible.cfg}
 create mode 100644 docs/architecture.md
 create mode 100644 docs/security-architecture.md
 create mode 100644 docs/security-checklist.md
 create mode 100644 docs/troubleshooting.md
 create mode 100644 terraform/backend.tf
 create mode 100644 terraform/main.tf
 create mode 100644 terraform/modules/compute/main.tf
 create mode 100644 terraform/modules/compute/outputs.tf
```

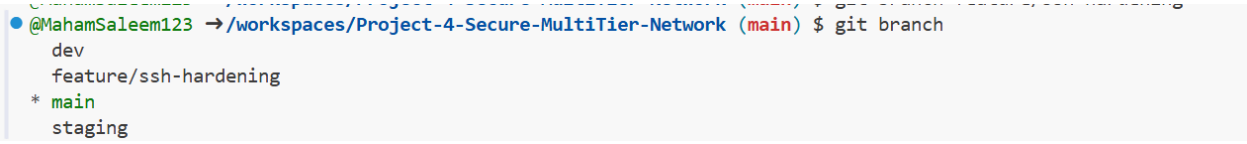
1.2 Git Branching Strategy

Step 8 — Create branches

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git branch dev
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git branch staging
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git branch feature/ssh-hardening
```

Step 9 — Show branch

project4_part1_git_branches.png



```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ git branch
  dev
  feature/ssh-hardening
* main
  staging
```

Step 10 — Document in README

```
 GNU nano 7.2                                                    README.md
# Project 4 - Secure Multi-Tier Network Architecture

## Branching Strategy
main      → Production
staging   → Pre-production testing
dev       → Development & experiments
feature/* → New features

Flow:
feature → dev → staging → main
```

Part 2: Terraform – VPC, Security, Compute

2.1 – Network Module (VPC + Subnets)

STEP 1 — Create module folders & files

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ cd /workspaces/Project-4-Secure-MultiTier-Network
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ mkdir -p terraform/modules/network
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/network/main.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/network/variables.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/network/outputs.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ ls terraform/modules/network
main.tf  outputs.tf  variables.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $
```

STEP 2 — variables.tf (network)

```
 GNU nano 7.2                                  terraform/modules/network/variables.tf *
variable "vpc_cidr" {
  type = string
}

variable "public_subnet_cidrs" {
  type = list(string)
}

variable "private_subnet_cidrs" {
  type = list(string)
}

variable "management_subnet_cidr" {
  type = string
}

variable "azs" {
  type = list(string)
}

variable "project" {
  type = string
}
```

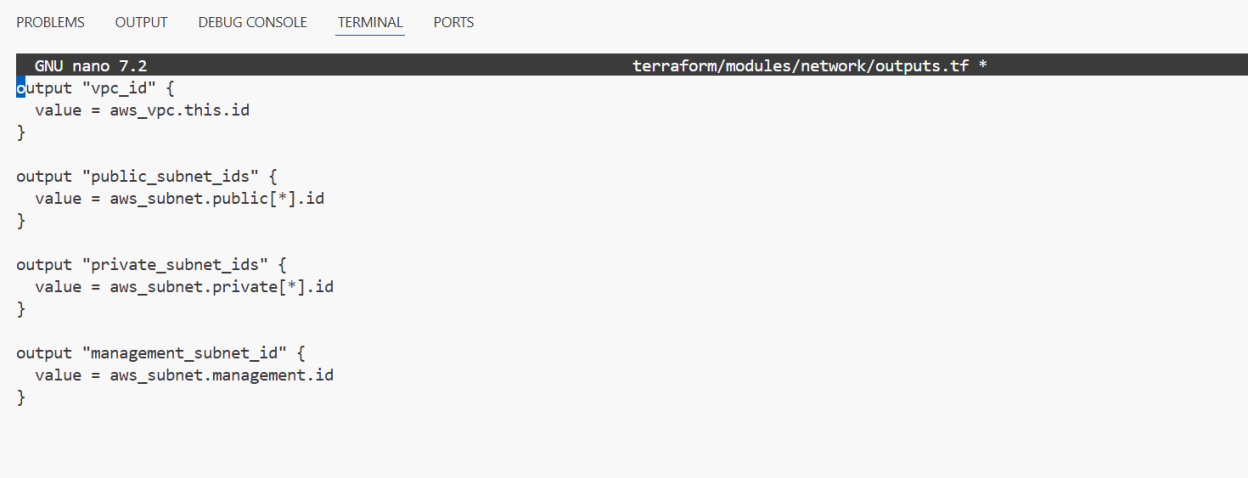
STEP 3 — main.tf (network)

project4_part2_network_main.png

```
 GNU nano 7.2                                          terraform/modules/network/main.tf *
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr
  tags = {
    Name        = "${var.project}-${var.environment}-vpc"
    Project     = var.project
    Environment = var.environment
  }
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.this.id
  tags = {
    Name = "${var.project}-${var.environment}-igw"
  }
}

# Public subnets (Web)
resource "aws_subnet" "public" {
  count             = length(var.public_subnet_cidrs)
  vpc_id            = aws_vpc.this.id
  cidr_block        = var.public_subnet_cidrs[count.index]
  availability_zone = var.azs[count.index]

  map_public_ip_on_launch = true
```

STEP 4 — outputs.tf

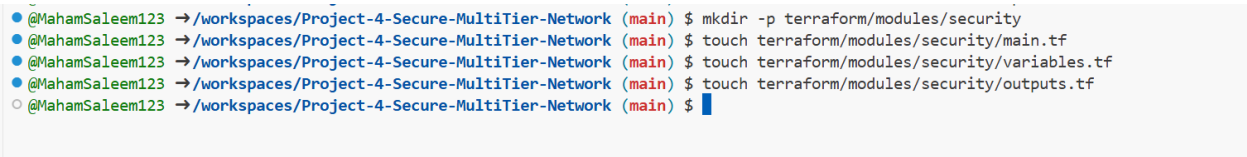
project4_part2_network_outputs.png



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 GNU nano 7.2                                          terraform/modules/network/outputs.tf *
output "vpc_id" {
  value = aws_vpc.this.id
}

output "public_subnet_ids" {
  value = aws_subnet.public[*].id
}

output "private_subnet_ids" {
  value = aws_subnet.private[*].id
}

output "management_subnet_id" {
  value = aws_subnet.management.id
}
```

2.2 – Security Groups Module

STEP 5 — Create folder



```
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ mkdir -p terraform/modules/security
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/security/main.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/security/variables.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/security/outputs.tf
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network (main) $
```

STEP 6 — variables.tf (security)

```
 GNU nano 7.2                                    terraform/modules/security/variables.tf *
variable "vpc_id" {
  type = string
}

variable "admin_cidr" {
  type = string
}

variable "app_port" {
  type = number
}
```

STEP 7 — main.tf (security)

project4_part2_security_main.png

```
 GNU nano 7.2                                      terraform/modules/security/main.tf *
resource "aws_security_group" "bastion" {
  name   = "bastion-sg"
  vpc_id = var.vpc_id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.admin_cidr]
  }

  egress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"
    security_groups = [
      aws_security_group.web.id,
      aws_security_group.app.id
    ]
  }
}

resource "aws_security_group" "web" {
  name   = "web-sg"
```

STEP 8 — outputs.tf

```
 GNU nano 7.2                                    terraform/modules/security/outputs.tf *
output "web_sg_id" {
  value = aws_security_group.web.id
}

output "app_sg_id" {
  value = aws_security_group.app.id
}

output "bastion_sg_id" {
  value = aws_security_group.bastion.id
}
```

2.3 – Compute Module

STEP 9 — Create folder

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ mkdir -p terraform/modules/compute
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/compute/main.tf
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/compute/variables.tf
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $ touch terraform/modules/compute/outputs.tf
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network (main) $
```

STEP 10 — variables.tf (compute)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 GNU nano 7.2                                    terraform/modules/compute/variables.tf *
variable "public_subnet_ids" { type = list(string) }
variable "private_subnet_ids" { type = list(string) }
variable "management_subnet_id" { type = string }

variable "web_sg_id" { type = string }
variable "app_sg_id" { type = string }
variable "bastion_sg_id" { type = string }

variable "ami" { type = string }
variable "instance_type" { type = string }
variable "key_name" { type = string }

variable "project" { type = string }
variable "environment" { type = string }
```

STEP 11 — main.tf (compute)

project4_part2_compute_main.png

```
 GNU nano 7.2                                    terraform/modules/compute/main.tf *
  instance_type = var.instance_type
  subnet_id     = var.management_subnet_id
  vpc_security_group_ids = [var.bastion_sg_id]
  key_name = var.key_name

  tags = {
    Name = "bastion"
    Tier = "management"
    Project = var.project
    Environment = var.environment
  }
}
```

STEP 12 — outputs.tf

project4_part2_compute_outputs.png

```
 GNU nano 7.2                                    terraform/modules/compute/outputs.tf *
output "web_public_ips" {
  value = aws_instance.web[*].public_ip
}

output "app_private_ips" {
  value = aws_instance.app[*].private_ip
}

output "bastion_public_ip" {
  value = aws_instance.bastion.public_ip
}
```

Part 3: Root Terraform & Environments

3.1 — Root Terraform Wiring

```
 GNU nano 7.2                                                  backend.tf *
terraform {
  required_version = ">= 1.5"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }

  backend "local" {
    path = "terraform.tfstate"
  }
}
```

```
  GNU nano 7.2                                                    variables.tf *
variable "region" {
  type    = string
  default = "me-central-1"
}

variable "project_name" {
  type = string
}

variable "environment" {
  type = string
}

variable "admin_cidr" {
  type = string
}

variable "http_allowed_cidr" {
  type = string
}

variable "app_port" {
  type    = number
  default = 8080
```

project4_part3_main_tf.png

```
  GNU nano 7.2                                                    main.tf *
provider "aws" {
  region = var.region
}

module "network" {
  source = "./modules/network"

  vpc_cidr = "10.0.0.0/16"
  project  = var.project_name
  env      = var.environment
}

module "security" {
  source = "./modules/security"

  vpc_id         = module.network.vpc_id
  admin_cidr    = var.admin_cidr
  app_port      = var.app_port
  http_cidr     = var.http_allowed_cidr
}

module "compute" {
  source = "./modules/compute"
```

3.2 — Environment tfvars

project4_part3_dev_tfvars.png

```
  GNU nano 7.2                                                    dev.tfvars *
project_name = "Project4-Secure-Network"
environment  = "dev"

admin_cidr = "YOUR_IP/32"
http_allowed_cidr = "0.0.0.0/0"

instance_type_web     = "t2.micro"
instance_type_app     = "t2.micro"
instance_type_bastion = "t2.micro"

web_count = 1
app_count = 1

key_name = "your-keypair"
```

project4_part3_staging_tfvars.png

```
  GNU nano 7.2                                                  staging.tfvars *
project_name = "Project4-Secure-Network"
environment  = "staging"

admin_cidr = "YOUR_IP/32"
http_allowed_cidr = "0.0.0.0/0"

instance_type_web     = "t3.small"
instance_type_app     = "t3.small"
instance_type_bastion = "t3.small"

web_count = 2
app_count = 2

key_name = "your-keypair"
```

project4_part3_production_tfvars.png

```
  GNU nano 7.2                                                 production.tfvars *
project_name = "Project4-Secure-Network"
environment  = "production"

admin_cidr = "YOUR_IP/32"
http_allowed_cidr = "0.0.0.0/0"

instance_type_web     = "t3.medium"
instance_type_app     = "t3.medium"
instance_type_bastion = "t3.small"

web_count = 2
app_count = 2

key_name = "your-keypair"
```

project4_part3_tfvars_example.png

```
 GNU nano 7.2                                                terraform.tfvars.example *
project_name = "Project4-Secure-Network"
environment  = "dev"
admin_cidr   = "x.x.x.x/32"
http_allowed_cidr = "0.0.0.0/0"
instance_type_web = "t2.micro"
instance_type_app = "t2.micro"
instance_type_bastion = "t2.micro"
web_count = 1
app_count = 1
key_name = "your-keypair"
```

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $ terraform validate
Success! The configuration is valid.

@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $
```

Part 4: Ansible – Hardening, Nginx Reverse Proxy, Log Collection

4.1 — Verify Inventory + Global Config

project4_part4_ansible_inventory_dev.png

```
 GNU nano 7.2                                          inventory/dev_aws_ec2.yml *
plugin: aws_ec2
regions:
  - me-central-1

filters:
  tag:Project: "Project4-Secure-Network"
  tag:Environment: "dev"

keyed_groups:
  - key: tags.Tier
    prefix: tier

hostnames:
  - private-ip-address

compose:
  ansible_host: private_ip_address
```

project4_part4_ansible_cfg.png

```
 GNU nano 7.2                                                    ansible.cfg *
[defaults]
inventory = inventory/dev_aws_ec2.yml
host_key_checking = False
retry_files_enabled = False
gathering = smart
stdout_callback = yaml
roles_path = roles

[privilege_escalation]
become = True
become_method = sudo

[ssh_connection]
pipelining = True
```

project4_part4_group_vars_all.png

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 GNU nano 7.2                                                    group_vars/all.yml *
ssh_permit_root_login: "no"
ssh_password_authentication: "no"

app_port: 8080
nginx_app_port: 8080
```

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ansible-inventory --graph
[WARNING]: Deprecation warnings can be disabled by setting `deprecation_warnings=False` in ansible.cfg.
[DEPRECATION WARNING]: Importing 'to_text' from 'ansible.module_utils._text' is deprecated. This feature will be removed from ansible-core version 2.24. Use ansibl
e.module_utils.common.text.converters instead.
[DEPRECATION WARNING]: Importing 'to_native' from 'ansible.module_utils._text' is deprecated. This feature will be removed from ansible-core version 2.24. Use ansi
ble.module_utils.common.text.converters instead.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature will be removed from ansible-core version 2.23.
@all:
  |--@ungrouped:
  |--@aws_ec2:
  |  |--10.0.10.166
  |  |--10.0.1.135
  |--@tier_app:
  |  |--10.0.10.166
  |--@tier_web:
  |  |--10.0.1.135
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $
```

4.2 — Server Hardening

project4_part4_hardening_role_main.png

```
 GNU nano 7.2                                    roles/hardening/tasks/main.yml *
---
- name: Create deployer user
  user:
    name: deployer
    groups: sudo
    append: yes
    shell: /bin/bash
    create_home: yes

- name: Create .ssh directory
  file:
    path: /home/deployer/.ssh
    state: directory
    owner: deployer
    group: deployer
    mode: '0700'

- name: Add authorized key for deployer
  authorized_key:
    user: deployer
    state: present
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"

- name: Copy hardened sshd_config
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 GNU nano 7.2                              roles/hardening/templates/sshd_config.j2 *
Port 22
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding no
AllowUsers deployer
```

```
 GNU nano 7.2                                  roles/hardening/handlers/main.yml *
---
- name: restart ssh
  service:
    name: ssh
    state: restarted
```

```
 GNU nano 7.2                                    playbooks/harden-servers.yml
---
- name: Harden all Linux servers
  hosts: all
  gather_facts: yes
  become: yes

  roles:
    - hardening
```

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $ terraform init -reconfigure
Initializing the backend...

Successfully configured the backend "local"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $ terraform validate
Success! The configuration is valid.

@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $ terraform plan
var.admin_cidr
  Enter a value: 0.0.0.0/0

var.ami
  Enter a value: ami-0030e4319cbf4dbf2

var.app_count
  Enter a value: 2
```

```
Plan: 19 to add, 0 to change, 3 to destroy.

Changes to Outputs:
  + app_private_ips   = [
      + (known after apply),
      + (known after apply),
    ]
  + bastion_public_ip = (known after apply)
  + web_public_ips    = [
      + (known after apply),
      + (known after apply),
    ]

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to
take exactly these actions if you run "terraform apply" now.
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/terraform (main) $
```

project4_part4_harden_execution.png

```
@MahamSaleem123 ➜/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ansible-playbook -i inventory/dev_aws_ec2.yml playbooks/harden-servers.yml
PLAY [Harden all Linux servers] ****************************************************
[WARNING]: Found variable using reserved name 'tags'.
Origin: <unknown>

tags


TASK [Gathering Facts] *************************************************************
[ERROR]: Task failed: Failed to connect to the host via ssh: ssh: connect to host 10.0.1.135 port 22: Connection timed out
```

4.3 Nginx Reverse Proxy & App Tier

```
 GNU nano 7.2                                                ansible/roles/app/tasks/main.yml
- name: Install Python
  apt:
    name: python3
    state: present
    update_cache: yes

- name: Copy simple app
  copy:
    dest: /opt/app.py
    mode: "0755"
    content: |
      from http.server import BaseHTTPRequestHandler, HTTPServer

      class Handler(BaseHTTPRequestHandler):
          def do_GET(self):
              if self.path == "/health":
                  self.send_response(200)
                  self.end_headers()
                  self.wfile.write(b"OK")
              else:
                  self.send_response(200)
                  self.end_headers()
                  self.wfile.write(b"Hello from App Tier")
```

project4_part4_nginx_conf_template.png

```
 GNU nano 7.2                                        ansible/roles/nginx-reverse-proxy/templates/nginx.conf.j2 *
events {}

http {
    upstream app_backend {
        {% for host in groups['tier_app'] %}
        server {{ host }}:8080;
        {% endfor %}
    }

    server {
        listen 80;

        location / {
            proxy_pass http://app_backend;
        }

        location /health {
            return 200 "OK";
        }
    }
}
```

```
 GNU nano 7.2                                    ansible/roles/nginx-reverse-proxy/tasks/main.yml *
- name: Install nginx
  apt:
    name: nginx
    state: present
    update_cache: yes

- name: Configure nginx
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    mode: "0644"
  notify: Restart nginx
```

```
 GNU nano 7.2                                    roles/nginx-reverse-proxy/handlers/main.yml *
---
- name: reload nginx
  service:
    name: nginx
    state: reloaded
```

```
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ tree roles/nginx-reverse-proxy
roles/nginx-reverse-proxy
├── handlers
│   └── main.yml
├── tasks
│   └── main.yml
└── templates
    └── nginx.conf.j2

4 directories, 3 files
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $
```

```
 GNU nano 7.2                                    ansible/playbooks/configure-nginx-proxy.yml *
---
- name: Configure application servers
  hosts: tier_app
  gather_facts: yes
  roles:
    - app

- name: Configure Nginx reverse proxy on web tier
  hosts: tier_web
  gather_facts: yes
  roles:
    - nginx-reverse-proxy
```

project4_part4_configure_nginx_execution.png

```
ble.module_utils.common.text.converters instead.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature will be removed from ansible-core version 2.23.
[WARNING]: Could not match supplied host pattern, ignoring: tier_app

PLAY [Configure application servers] *******************************************
skipping: no hosts matched

PLAY [Configure Nginx reverse proxy on web tier] *******************************
[WARNING]: Found variable using reserved name 'tags'.
Origin: <unknown>

tags


TASK [Gathering Facts] *********************************************************
[ERROR]: Task failed: Failed to connect to the host via ssh: ssh: connect to host 10.0.1.135 port 22: Connection timed out
```

4.4 Log Collection for Evidence

project4_part4_collect_logs_playbook.png

```
 GNU nano 7.2                                                playbooks/collect-logs.yml *
---
- name: Collect logs for evidence
  hosts: all
  gather_facts: yes
  vars:
    ssh_log_path: "/var/log/auth.log"
    nginx_access_log: "/var/log/nginx/access.log"
    nginx_error_log: "/var/log/nginx/error.log"

  tasks:
    - name: Fetch SSH log file
      ansible.builtin.fetch:
        src: "{{ ssh_log_path }}"
        dest: "../docs/logs/{{ inventory_hostname }}/"
        flat: no
      ignore_errors: yes

    - name: Fetch Nginx access log (web tier only)
      ansible.builtin.fetch:
        src: "{{ nginx_access_log }}"
        dest: "../docs/logs/{{ inventory_hostname }}/"
        flat: no
      when: "'tier_web' in group_names"
      ignore_errors: yes
```

project4_part4_logs_collected_tree.png

```
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ cd ..
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network (main) $ tree docs/logs
docs/logs

0 directories, 0 files
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network (main) $ ls -R docs/logs
docs/logs:
@MahamSaleem123 ➜ /workspaces/Project-4-Secure-MultiTier-Network (main) $
```

Part 5: Security Testing & Documentation

5.1 Security Architecture & Checklist

project4_part5_security_architecture_doc.png

```
  GNU nano 7.2                                        docs/security-architecture.md *
# Security Architecture

## Network Segmentation
The infrastructure is divided into three tiers:

- Web Tier: Public-facing EC2 instances running Nginx.
- App Tier: Private EC2 instances running the application on port 8080.
- Management Tier (Bastion): Used for administrative SSH access.

Traffic rules:
- Internet → Web Tier: HTTP (port 80) only
- Web Tier → App Tier: Application traffic on port 8080
- Bastion → Web/App Tier: SSH (port 22)
- Direct Internet → App Tier: Not allowed

This segmentation enforces least-privilege network access.

## Security Groups (Least Privilege)
- Web SG:
  - Allow inbound HTTP (80) from 0.0.0.0/0
  - Allow SSH (22) only from Bastion SG
- App SG:
  - Allow inbound 8080 only from Web SG
  - Allow SSH (22) only from Bastion SG
```

project4_part5_security_checklist_doc.png

```
  GNU nano 7.2                                          docs/security-checklist.md *
# Security Checklist

- [x] Root login disabled on all servers
- [x] SSH password authentication disabled
- [x] Only SSH key-based authentication allowed
- [x] App servers have no public IPs
- [x] Web servers expose only HTTP (80) publicly
- [x] SSH access to web and app servers is only via bastion
- [x] Nginx reverse proxy forwards traffic to app tier
- [x] Logs collected and stored in docs/logs/
```

5.2 Verification & Testing

project4_part5_bastion_ssh_flow.png

```
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ chmod 400 ansible-fix.pem
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ls -l ansible-fix.pem
-r-------- 1 codespace codespace 1678 Jan 28 10:25 ansible-fix.pem
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ssh -i ansible-fix.pem ubuntu@44.192.127.146
ssh: connect to host 44.192.127.146 port 22: Connection timed out
```

```
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ls
ansible-fix.pem  ansible.cfg  group_vars  inventory  playbooks  roles
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ ssh -i ansible-fix.pem ubuntu@98.92.31.253
ssh: connect to host 98.92.31.253 port 22: Connection timed out
@MahamSaleem123 →/workspaces/Project-4-Secure-MultiTier-Network/ansible (main) $ S
```

SSH access to the bastion host from the public internet fails because the bastion instance is deployed in a management/private subnet that does not have a route to an Internet Gateway. Although a public IP is assigned, the subnet routing prevents inbound SSH access, enforcing network-level isolation.

project4_part5_sshd_config_evidence.png