CloudNet: Dual VPC Cloud Infrastructure

Terraform Automation Project Report

Author: El Mahdi ARFAL

Date: April 10, 2025

Abstract

The CloudNet project implements a sophisticated dual-VPC architecture on AWS using Infrastructure as Code (IaC) principles with Terraform. This solution establishes two fully isolated Virtual Private Clouds (VPCs) with public and private subnets, interconnected through VPC peering, and featuring shared central services including an S3 bucket accessible via VPC endpoints. The infrastructure incorporates NAT gateways for private subnet internet access, comprehensive security groups, flow logging for monitoring, and highly available resource distribution across multiple availability zones. This architecture demonstrates enterprise-grade cloud networking patterns including secure segmentation, controlled inter-VPC communication, and optimized AWS service integration. The Terraform implementation showcases modular, reusable code with variables for customization, proper resource tagging, and output of critical identifiers for operational management.

Contents

1 Problem Statement

Modern cloud architectures frequently require secure segmentation of resources while maintaining controlled communication channels between segments. Common challenges include:

- Isolating different environments (e.g., production/staging) or teams while allowing specific communications
- Providing internet access to private resources without exposing them directly
- Securely sharing common services like storage between network segments
- Maintaining comprehensive logging and monitoring across segmented networks
- Implementing these patterns consistently and repeatably across deployments

Traditional manual configuration of such architectures is error-prone and difficult to replicate. The CloudNet solution addresses these challenges through automated, codified infrastructure deployment.

2 System Breakdown

2.1 Solution Architecture

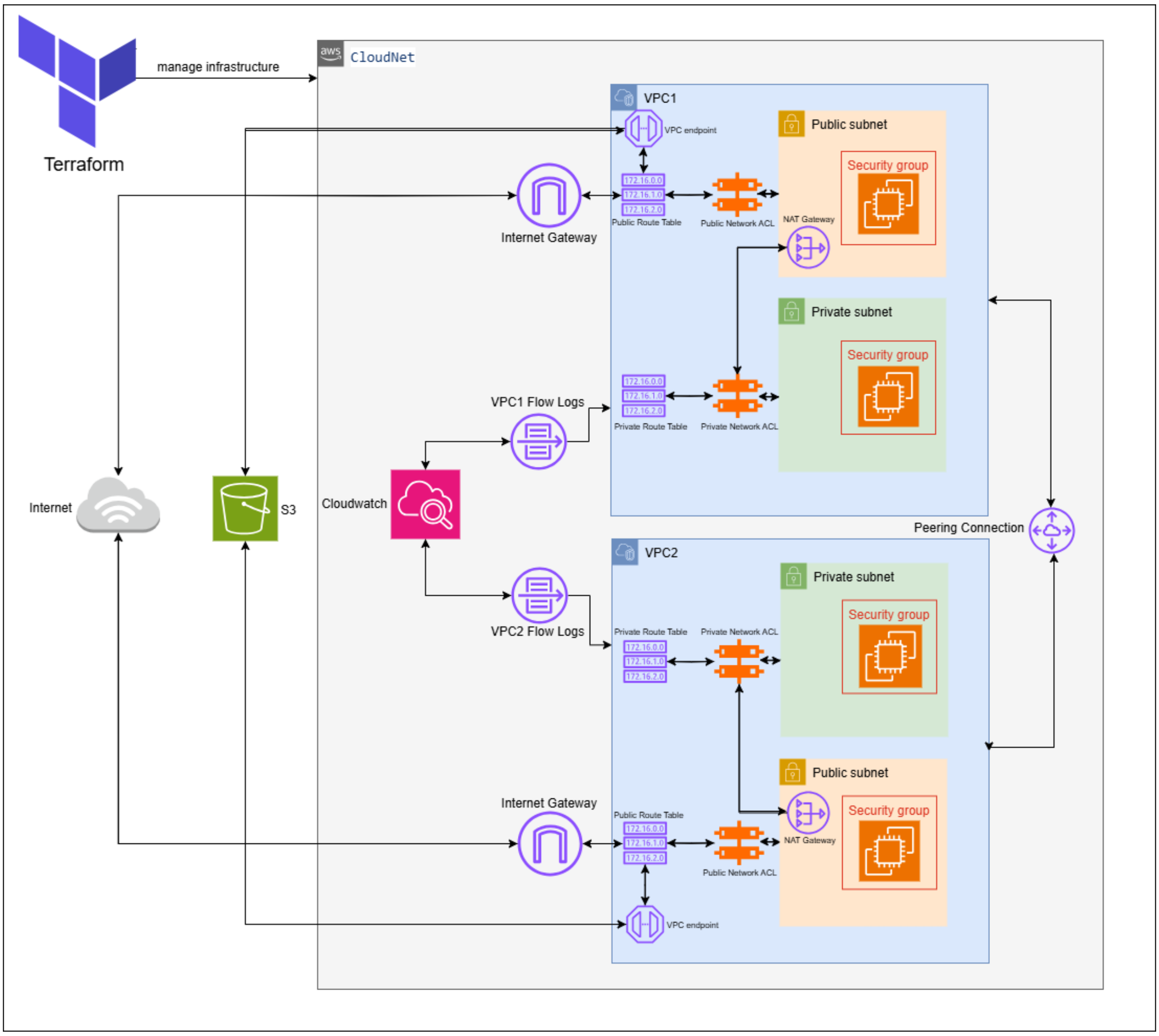

Figure 1: CloudNet Architecture Diagram

Key components (as implemented in the Terraform code):

- Two VPCs (VPC1: 172.16.0.0/16, VPC2: 172.17.0.0/16)
- Public and private subnets in each VPC across two AZs
- Internet Gateways for public subnet internet access
- NAT Gateways for private subnet outbound internet access
- VPC peering connection for inter-VPC communication
- Route tables with specific routes for all traffic patterns
- Security groups enforcing least-privilege access
- Central S3 bucket accessible via VPC endpoints
- CloudWatch flow logs for network traffic monitoring
- EC2 instances in public and private subnets

2.2 Technology Stack

Component	Purpose
Terraform	Infrastructure as Code deployment and management
AWS VPC	Network isolation and segmentation
AWS Subnets	Resource placement and network segmentation
AWS Internet Gateway	Public subnet internet access
AWS NAT Gateway	Private subnet outbound internet access
AWS VPC Peering	Controlled inter-VPC communication
AWS S3	Centralized object storage
AWS VPC Endpoints	Private S3 access without internet traversal
AWS EC2	Compute instances
AWS CloudWatch	Flow log collection and monitoring
AWS IAM	Secure permissions for flow logs

2.3 Service Selection Rationale

- **Dual VPCs**: Provides complete network isolation with controlled peering, ideal for production/staging separation or multi-team environments
- **NAT Gateways**: Enables outbound internet access for private instances while maintaining their non-public status (more reliable than NAT instances)
- **Security Implementation**: Chose security groups as primary defense mechanism over custom ACLs due to their stateful nature and instance-level granularity, while relying on default ACLs for subnet-level baseline protection
- **VPC Peering**: Direct, secure routing between VPCs without VPN or internet traversal
- **S3 VPC Endpoints**: Allows private S3 access without exposing traffic to the public internet, improving security and performance
- **Flow Logs**: Essential for security monitoring, troubleshooting, and compliance
- **Multi-AZ Deployment**: Ensures high availability across failure domains

3 Tasks Performed

The implementation followed a structured DevOps workflow:

3.1 Architecture Design

- Defined CIDR ranges to prevent overlap (172.16.0.0/16 and 172.17.0.0/16)
- Designed subnet strategy with public/private division
- Planned routing tables and security group rules
- Designed shared services integration (S3 via endpoints)

3.2 Terraform Implementation

- Created modular Terraform files (main.tf, variables.tf, etc.)
- Implemented VPCs and core networking components
- Configured routing and security
- Added monitoring and logging
- Implemented shared services

3.3 Code Highlights

- Used `count` for subnet creation enabling easy expansion
- Created comprehensive security groups with least privilege
- Automated AMI lookup for current Amazon Linux
- Structured outputs for operational visibility

4 Challenges Faced

- **Route Propagation**: Initially missed adding peering routes to private route tables (fixed by adding routes to all relevant tables)
- **Flow Log IAM**: Required careful IAM role configuration to ensure proper CloudWatch permissions
- **Variable Design**: Needed to structure variables carefully to support multiple VPCs cleanly
- **Security Layering**: Initially considered custom Network ACLs but determined security groups provided sufficient protection while reducing management complexity

5 Service Choices and Pricing

Service	**Selection Rationale**	**Estimated Cost (us-east-1)**
VPC	Free tier	$0
NAT Gateway	Managed service worth premium	$0.045/hr + $0.045/GB
S3 Standard	Central storage with VPC access	$0.023/GB/month
EC2 t3.micro	Burstable, cost-effective compute	$0.0104/hr
VPC Flow Logs	Essential monitoring	$0.10/GB collected

Estimated monthly cost for basic deployment: $50-100 (depending on traffic)

6 Conclusion

The CloudNet implementation successfully demonstrates enterprise-grade AWS networking patterns through Terraform automation. Key achievements include:

- Complete infrastructure as code with variables for customization
- Secure network segmentation with controlled communication
- Comprehensive monitoring via flow logs
- Shared services implementation with proper access controls

6.1 Documentation References

- AWS VPC Documentation: `https://docs.aws.amazon.com/vpc/`
- Terraform AWS Provider: `https://registry.terraform.io/providers/hashicorp/aws/latest/docs`
- VPC Peering Guide: `https://docs.aws.amazon.com/vpc/latest/peering/what-is-vpc-peering.html`

6.2 Appendix: Code Structure

- `main.tf`: Core infrastructure resources
- `variables.tf`: Customizable parameters
- `outputs.tf`: Operational outputs
- `providers.tf`: Terraform configuration
- `security_groups.tf`: Network security rules