Complete AWS VPC Lab Setup Guide

Table of Contents

- Overview
- Architecture Components
- Prerequisites
- Phase 1: VPC Infrastructure Setup
- Phase 2: Bastion Host Configuration
- Phase 3: NAT Instance Setup
- Phase 4: Configure Private Subnet Internet Access
- Phase 5: Create Private Instance
- Phase 6: SSH Access Through Bastion Host
- Phase 7: Alternative Access Method - AWS Systems Manager
- Connection Methods Comparison
- Security Considerations
- Troubleshooting Common Issues

Overview

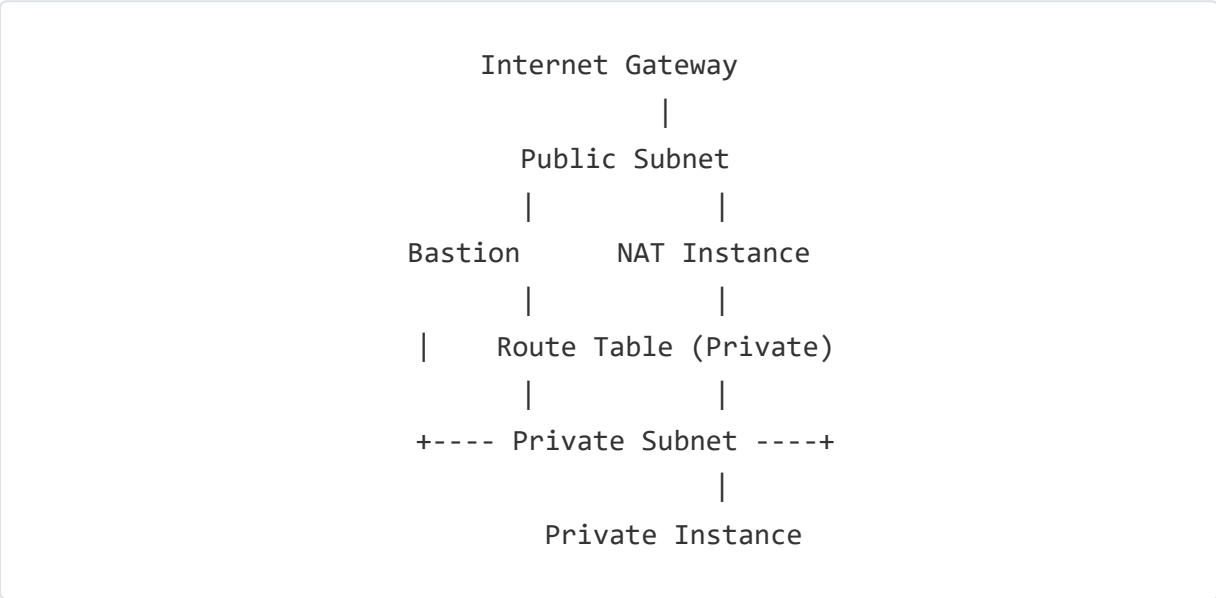
This comprehensive guide demonstrates how to create a complete AWS VPC infrastructure with multiple access methods to private instances. The lab covers VPC creation, NAT instance configuration, bastion host setup, and both SSH and AWS Systems Manager (SSM) Session Manager connectivity options.

Architecture Components

- **VPC (Virtual Private Cloud)**: Isolated network environment

- **Public Subnet**: Internet-accessible subnet for bastion and NAT instances
- **Private Subnet**: Secure subnet for backend instances
- **Bastion Host**: SSH jump server for secure access
- **NAT Instance**: Provides outbound internet access for private instances
- **Private Instance**: Backend server with no direct internet access
- **IAM Role**: Enables SSM Session Manager connectivity

Network Architecture Diagram



Prerequisites

- AWS account with appropriate permissions
- Key pair (key01.pem) for SSH access
- Basic understanding of AWS networking

Phase 1: VPC Infrastructure Setup

1.1 Access VPC Console

1. Navigate to AWS Console
2. Search for "vpc" in the services search bar
3. Select **VPC** service
4. Access URL: `https://eu-north-1.console.aws.amazon.com/vpcconsole/home?region=eu-north-1#Home:`

1.2 Create VPC with Complete Infrastructure

1. From **VPC Dashboard**, navigate to **Your VPCs**
2. Click **Create VPC**
3. Configure VPC settings:
   - **VPC Settings**: Select **VPC and more**
   - **Name tag auto-generation**: Enter `lab`
   - **VPC endpoints**: Select **None**
4. Click **Create VPC**

> **What AWS Creates Automatically:**
>
> - VPC with CIDR block (typically 10.0.0.0/16)
> - Internet Gateway (attached to VPC)
> - Public subnet(s)
> - Private subnet(s)
> - Route tables for public and private subnets
> - Default security group

Phase 2: Bastion Host Configuration

2.1 Launch Bastion Instance

1. Navigate to **EC2 Console**
2. Go to **Instances** → **Launch Instance**
3. Configure instance settings:
    - **Name and tags**: `bastion`
    - **AMI**: Amazon Linux 2 (default)
    - **Instance Type**: t2.micro

2.2 Network Configuration for Bastion

1. In **Network settings**:
    - **VPC**: Select `lab`
    - **Subnet**: Select `public-1`
    - **Auto-assign public IP**: Enable (default)
2. **Key pair**: Select `key01` (or your existing key pair)
3. Launch the instance

Phase 3: NAT Instance Setup

3.1 Launch NAT Instance

1. In **EC2 Console**, go to **Instances** → **Launch Instance**
2. Configure basic settings:
   - **Name and tags**: `nat`

3.2 Select NAT-Specific AMI

1. In **Application and OS Images (Amazon Machine Image)**:
   - Search for `amzn-ami-vpc-nat`
   - Navigate to **Community AMIs** tab
   - Select the first available NAT AMI
   - Click **Select**

3.3 Network Configuration for NAT

1. In **Network settings**:
   - **VPC**: Select `lab-vpc`
   - **Subnet**: Select `public1`
   - **Auto-assign public IP**: **Enable**
2. **Key pair**: Select `key01`
3. Click **Launch Instance**

3.4 Create Security Group for NAT Instance

1. Navigate to **EC2 Console** → **Security Groups**
2. Click **Create security group**
3. Configure security group:
   - **Name**: `nat-security-group`
   - **Description**: Allow traffic from VPC for NAT functionality
   - **VPC**: Select `lab-vpc`

3.5 Configure Inbound Rules for NAT

1. In **Inbound rules** section, click **Add rule**
2. Configure rule:
    - **Type**: All traffic
    - **Protocol**: All
    - **Port range**: All
    - **Source**: `10.0.0.0/16` (VPC CIDR block)
3. Click **Create security group**

3.6 Apply Security Group to NAT Instance

1. Select the NAT instance
2. Go to **Actions** → **Security** → **Change security groups**
3. Add the `nat-security-group`

3.7 Configure NAT Instance for Routing

1. Select the NAT instance
2. Navigate to **Actions** → **Networking** → **Change Source/Destination Check**
3. **Uncheck** the "Enable" checkbox (disable source/destination check)
4. Click **Save**

> **Why disable source/destination check:**
>
> - NAT instances need to forward traffic between different network segments
> - EC2 instances normally drop traffic not destined for their own IP
> - Disabling this allows the instance to act as a router/NAT device

3.8 Restart NAT Instance

1. Select the NAT instance
2. **Instance State** → **Stop instance**
3. Wait for complete shutdown
4. **Instance State** → **Start instance**

Phase 4: Configure Private Subnet Internet Access

4.1 Update Private Route Table

1. Navigate to **VPC Console** → **Route Tables**
2. Find and select the private route table: `lab-rtb-private1-eu-north-1a`
3. Click **Routes** tab
4. Click **Edit routes**

4.2 Add NAT Route

1. Click **Add route**
2. Configure route:
   - **Destination**: `0.0.0.0/0` (all internet traffic)
   - **Target**: Select **Instance**, then choose the NAT instance (`i-0a5422b20ad5df8fb`)
3. Click **Save changes**

> **What this accomplishes:**
>
> - Routes all internet-bound traffic from private subnet through NAT instance
> - Enables private instances to access internet for updates and downloads
> - Maintains security by preventing inbound internet connections

Phase 5: Create Private Instance

5.1 Launch Private Instance

1. In **EC2 Console**, go to **Instances** → **Launch Instance**
2. Configure settings:
   - **Name**: `private`
   - **AMI**: Amazon Linux 2 (default)
   - **Instance Type**: t2.micro

5.2 Network Configuration for Private Instance

1. In **Network settings**:
   - **VPC**: Select `lab`
   - **Subnet**: Select `private1`
   - **Auto-assign public IP**: Disable (default for private subnet)
2. **Key pair**: Select `key01`
3. Launch the instance

Phase 6: SSH Access Through Bastion Host

6.1 Connect to Bastion Host

1. In **EC2 Console**, select the bastion instance
2. Click **Connect**
3. Note the connection details:
   - **Instance ID**: `i-0ec2f559a130315df` (bastion)
   - **Private IP**: `10.0.4.212`

6.2 SSH Connection Commands

1. Ensure proper key permissions:

```
chmod 400 "key01.pem"
```

2. Connect to bastion host:

```
ssh -i "key01.pem" ec2-user@10.0.4.212
```

6.3 Copy Private Key to Bastion Host

To access the private instance through the bastion, you need to copy the private key:

Method 1: Using SCP

```
scp -i "key01.pem" key01.pem ec2-user@10.0.4.212:~/
```

Method 2: Copy-paste key content

1. Open key01.pem in a text editor
2. Copy the entire content
3. SSH into bastion host

4. Create the key file: `nano ~/key01.pem`
5. Paste the content and save
6. Set permissions: `chmod 400 ~/key01.pem`

6.4 Connect to Private Instance via Bastion

From within the bastion host, connect to private instance:

```
ssh -i "key01.pem" ec2-user@[PRIVATE_INSTANCE_IP]
```

Phase 7: Alternative Access Method - AWS Systems Manager

7.1 Create IAM Role for SSM

1. Navigate to **IAM Console** → **Roles**
2. Click **Create role**
3. Configure role:
   - **Trusted entity**: AWS service
   - **Service**: EC2
   - Click **Next**

7.2 Attach SSM Policy

1. In **Permission policies**, search for "ssm"
2. Select **AmazonSSMManagedInstanceCore**
3. Click **Next**
4. **Role name**: `basic-ssm`
5. Click **Create role**

7.3 Attach IAM Role to Private Instance

1. Navigate to **EC2 Console** → **Instances**
2. Select the `private` instance
3. Go to **Security** tab
4. Click **Modify IAM role**
5. Select **basic-ssm** role
6. Click **Update IAM role**

7.4 Restart Private Instance

1. **Instance State** → **Stop instance**
2. Wait for complete shutdown
3. **Instance State** → **Start instance**

> **Why restart is necessary:**
>
> - IAM role attachment requires instance restart to take effect
> - SSM agent needs to initialize with new permissions

7.5 Connect via Session Manager

1. Select the `private` instance
2. Click **Connect**
3. Choose **Session Manager** tab
4. Click **Connect**
5. Access the instance through web browser console

Connection Methods Comparison

Method	Pros	Cons	Use Case
SSH via Bastion	Full SSH functionality, familiar interface	Requires key management, bastion maintenance	Development, file transfers
Session Manager	No keys needed, audit logging, web-based	Limited functionality, requires IAM setup	Quick admin tasks, secure access

Security Considerations

Network Security

- **Bastion Host**: Limit SSH access to specific IP ranges
- **Private Instances**: No direct internet access
- **Security Groups**: Implement least privilege access
- **NACLs**: Additional subnet-level security (optional)

Access Control

- **SSH Keys**: Rotate regularly, use different keys per environment
- **IAM Roles**: Minimal required permissions only
- **Session Manager**: Centralized logging and session recording

Best Practices

1. **Monitoring**: Enable VPC Flow Logs
2. **Logging**: CloudTrail for API calls, Session Manager for shell sessions
3. **Updates**: Regular security patches via NAT internet access
4. **Backup**: Regular snapshots of critical instances

Troubleshooting Common Issues

Connectivity Problems

Private instance can't reach internet:

- Verify NAT instance is running
- Check private route table has 0.0.0.0/0 → NAT instance route
- Ensure NAT security group allows VPC traffic
- Confirm source/destination check is disabled on NAT

Can't SSH to private instance:

- Verify bastion host connectivity
- Check private key is copied to bastion
- Confirm security groups allow SSH (port 22)

Testing Connectivity

Test internet connectivity from private instance:

```
ping 8.8.8.8
curl -I google.com
```

Verification steps:

- Check route tables in VPC console
- Verify security group rules allow required traffic
- Test NAT instance connectivity from AWS console

> **Cost Optimization Tips:**
>
> - Use NAT Gateway instead of NAT instance for production environments

> - Consider using VPC endpoints for AWS services to avoid internet routing
> - Monitor data transfer costs through NAT instances

This configuration demonstrates AWS networking best practices and provides foundation for more complex architectures.