Week 4 Final Project — Technical Documentation

Secure Public Website with Private Infrastructure

Project Date: January 31, 2026
Team: Cloud Engineering Bootcamp - Week 4
Client Scenario: European Logistics Company AWS Migration
Website URL: http://18.208.206.23

Table of Contents

Executive Summary

This project demonstrates a production-grade AWS architecture implementing strict separation of concerns between public-facing infrastructure and private administrative systems. The deployment uses Infrastructure-as-Code (Terraform) to automate provisioning while maintaining enterprise-level security controls through IAM roles and AWS Systems Manager.

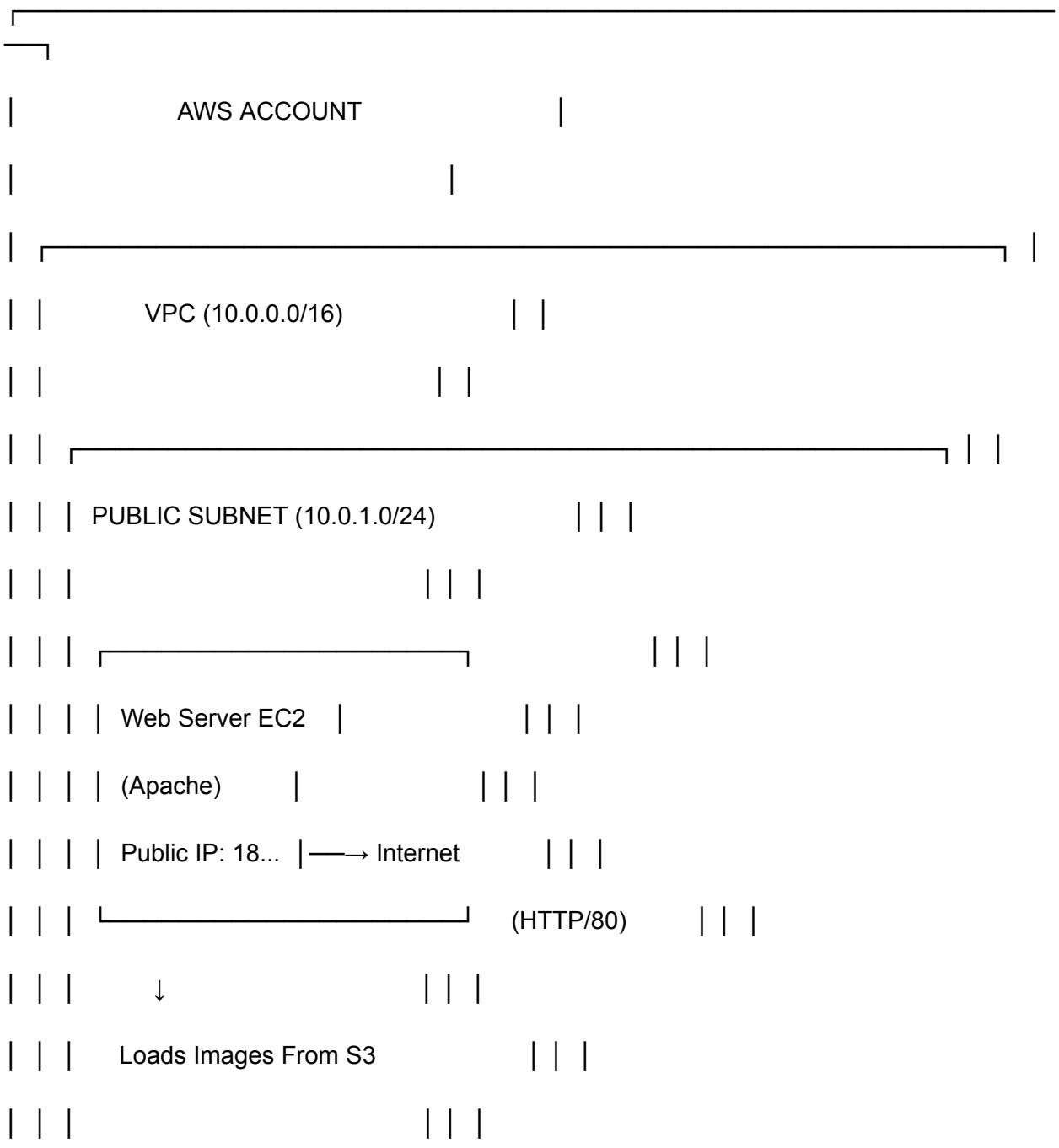
Key Achievements:

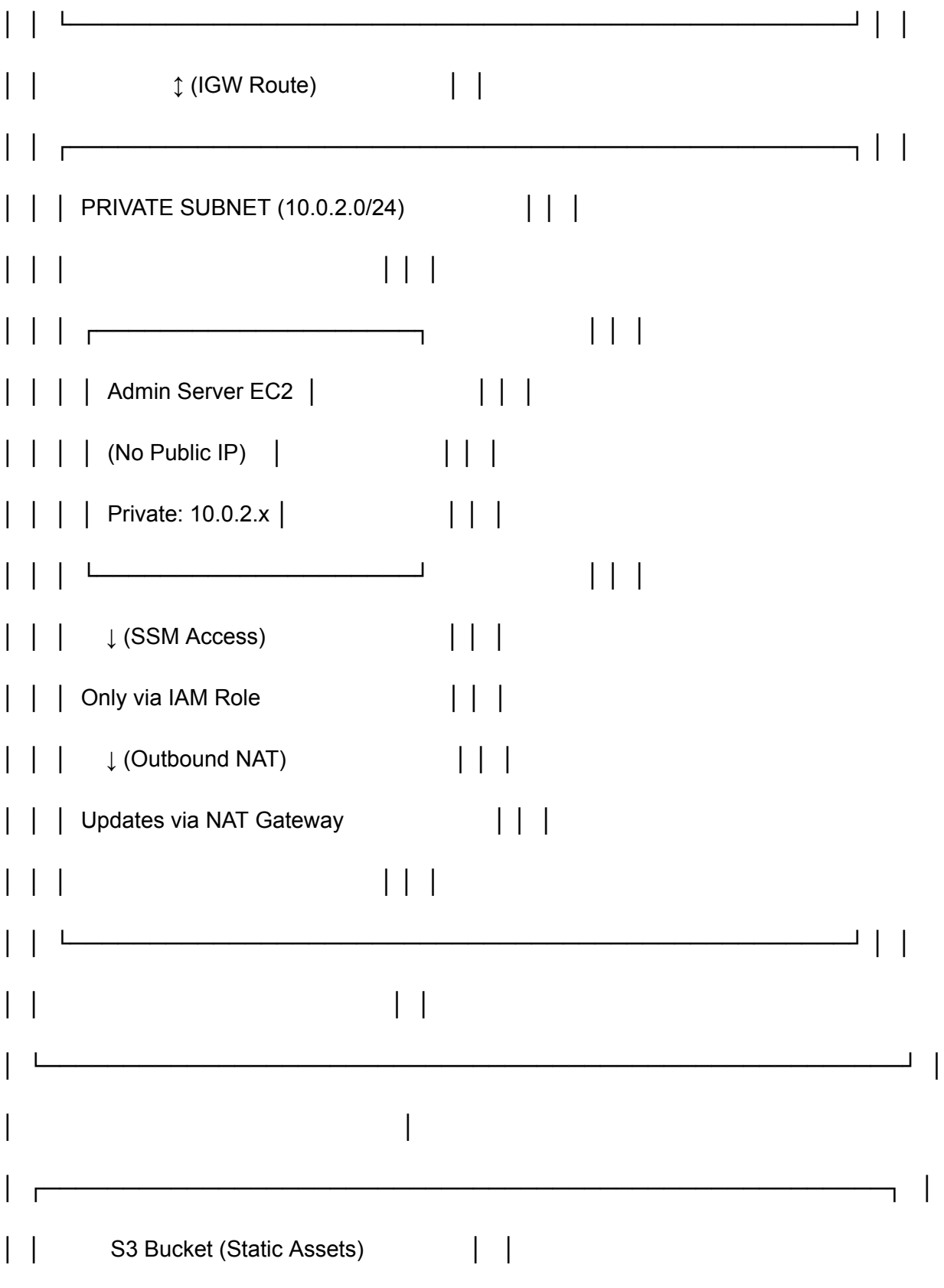
- ✅ Publicly accessible website with Apache on EC2
- ✅ Private backend infrastructure with no internet exposure
- ✅ Static asset storage via S3 (decoupled from compute)
- ✅ IAM-based access control (no SSH key management)
- ✅ Secure remote access via AWS Systems Manager
- ✅ Cost-optimized architecture (estimated <$100/month)

- ✅ Production-ready security posture

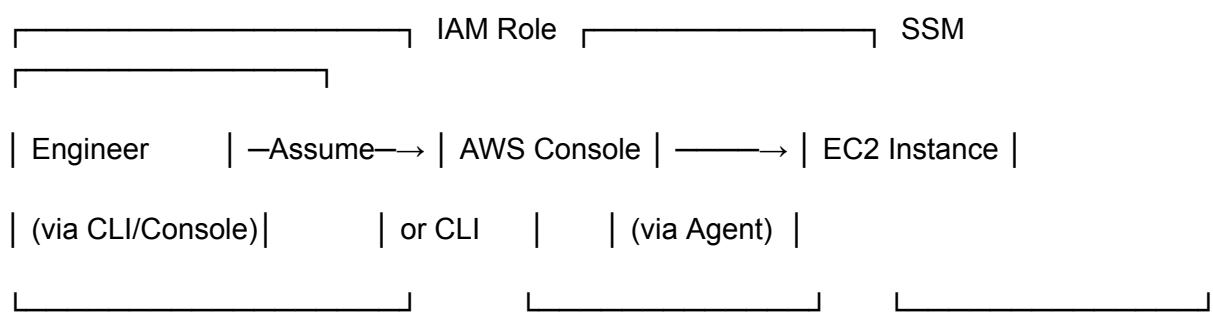
Architecture Overview

Multi-Tier Design




↕ (IGW Route)
PRIVATE SUBNET (10.0.2.0/24)
Admin Server EC2
(No Public IP)
Private: 10.0.2.x
↓ (SSM Access)
Only via IAM Role
↓ (Outbound NAT)
Updates via NAT Gateway
S3 Bucket (Static Assets)

Engineer Access Path (via SSM):



Traffic Flows

1. Customer Access (Public Web)

Internet User

→ HTTP Request to 18.208.206.23:80

→ Security Group: Allow :80 from 0.0.0.0/0

→ Apache Web Server (EC2)

→ Serves index.html

→ Browser loads image from S3 (public URL)

→ User sees "Bone For Tuna" website ✓

2. Static Asset Access (S3)

Apache HTML Page

→ <img src="https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg">

→ S3 Bucket Policy: Allow s3:GetObject from *

→ S3 returns object (public)

→ Browser renders image ✓

3. Engineer Access (SSM)

Engineer (IAM Role)

→ AWS Console: Systems Manager → Session Manager

→ IAM Role: ssm:StartSession permission

→ EC2 Instance: SSM Agent running + IAM role attached

→ SSH-like shell without SSH keys ✓

4. Private Server Updates (NAT)

Private EC2 (10.0.2.x)

→ Outbound connection (update packages)

→ NAT Gateway (10.0.1.x)

→ Internet Gateway

→ External update servers (yum, etc.)

→ Response routed back via NAT ✓

Deployment Instructions

Prerequisites

Verify installed tools

terraform version # 1.x or later

aws --version   # 2.x or later

git --version   # 2.x or later

Configure AWS credentials

aws configure

Enter: Access Key ID, Secret Access Key, Region (us-east-1)

Step 1: Clone Repository & Initialize Terraform

git clone https://github.com/YOUR_REPO/week4-secure-web.git

cd week4-secure-web

Initialize Terraform (downloads provider plugins)

terraform init

Step 2: Create IAM Roles (AWS Console Only)

This cannot be automated in Terraform. Do this manually in AWS Console.

Role 1: EC2 Instance Role

Service: EC2

Name: Week4-EC2-SSM-Role

Permissions: AmazonSSMManagedInstanceCore

Role 2: Engineer - SSM Admin

Service: IAM User/EC2

Name: Week4-SSM-Admin

Permissions:

- AmazonSSMFullAccess
- AmazonEC2ReadOnlyAccess

Role 3: Engineer - SSM Operator

Service: IAM User/EC2

Name: SSM-Operator

Permissions (Custom Inline):

- ssm:StartSession
- ssm:TerminateSession
- ssm:DescribeSessions
- ssm:DescribeInstanceInformation

Role 4: Engineer - ReadOnly

Service: IAM User/EC2

Name: ReadOnly

Permissions: ReadOnlyAccess

Step 3: Deploy Infrastructure with Terraform

```
# Review infrastructure plan

terraform plan

# Deploy all resources (takes ~5-10 minutes)

terraform apply

# When prompted, type: yes
```

Expected Output:

Apply complete! Resources: XX added, 0 changed, 0 destroyed.

Outputs:

```
  instance_id = "i-04b63896445a24fdd"

  instance_public_ip = "18.208.206.23"

  private_instance_id = "i-096d32199f926bb55"

  s3_bucket_name = "bonefortuna-website-images-281845296706"

  website_url = "http://18.208.206.23"
```

Step 4: Download SSH Keypair

```
# If you haven't already, download the EC2 keypair from AWS Console

# EC2 → Key Pairs → week4-bonefortuna (or your keypair name)

# Save to: ~/.ssh/week4-bonefortuna.pem

# Set permissions

chmod 600 ~/.ssh/week4-bonefortuna.pem
```

Step 5: Install Session Manager Plugin

```
# Download the plugin

curl
"https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-ma
nager-plugin.deb" \

  -o "session-manager-plugin.deb"

# Install

sudo dpkg -i session-manager-plugin.deb

# Verify

session-manager-plugin --version
```

Step 6: Install Apache & Create Website

```
# SSH into public EC2 (one-time setup)

ssh -i ~/.ssh/week4-bonefortuna.pem ec2-user@18.208.206.23

# Once connected:

sudo yum update -y

sudo yum install httpd -y

sudo systemctl start httpd

sudo systemctl enable httpd

# Verify Apache is running

sudo systemctl status httpd
```

Step 7: Upload Banner Image to S3

```
# From your local machine

cd week4-secure-web

# Download image

wget
https://drive.google.com/uc?export=download&id=1rn-Tj08vo6rPiBqTAKdG-DpF7DsLH5mw \

  -O banner.jpg

# Get bucket name from Terraform

BUCKET_NAME=$(terraform output -raw s3_bucket_name)

# Upload to S3

aws s3 cp banner.jpg s3://${BUCKET_NAME}/banner.jpg

# Verify file exists
```

```
aws s3 ls s3://${BUCKET_NAME}/
```

Step 8: Create Website HTML

```
# SSH into public EC2

ssh -i ~/.ssh/week4-bonefortuna.pem ec2-user@18.208.206.23

# Create index.html

sudo nano /var/www/html/index.html

# Paste the following content:

<!DOCTYPE html>

<html>

<head>

    <title>Bone For Tuna</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            color: white;

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            margin: 0;

            padding: 20px;
```

```
}
.container {
    text-align: center;
    padding: 40px;
    max-width: 800px;
}
h1 {
    font-size: 3em;
    margin-bottom: 20px;
}
p {
    font-size: 1.2em;
    margin: 10px 0;
}
.feature {
    background: rgba(255,255,255,0.1);
    padding: 20px;
    margin: 20px 0;
    border-radius: 10px;
}
img {
```

```
            max-width: 100%;
            border-radius: 10px;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>🐾 Bone For Tuna 🐟</h1>
        <img
src="https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg"
             alt="Bone For Tuna Banner" />
        <p><strong>Premium Nutrition for Your Furry Friends</strong></p>
        <div class="feature">
            <p>Quality ingredients, happy pets.</p>
            <p>Scientifically formulated. Veterinary approved.</p>
        </div>
        <small>Deployed via AWS Terraform | Systems Manager Enabled</small>
    </div>
</body>
</html>
```

Save: Ctrl+X → Y → Enter

Step 9: Test Website

From local machine

curl http://18.208.206.23

Or open in browser: http://18.208.206.23

Issues Encountered & Fixes

Issue #1: S3 Bucket Image Not Loading (403 Forbidden)

Problem:

Image displayed as broken link in website

curl -I https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg

→ HTTP/1.1 403 Forbidden

Root Cause: S3 bucket had default Block Public Access settings enabled, preventing public object access even with bucket policy.

Timeline:

1. Uploaded banner.jpg successfully to S3
2. Added S3 image URL to HTML
3. Website loaded, but image showed 403 error
4. Verified bucket policy was created correctly
5. Discovered Block Public Access was still blocking the policy

Fix Applied:

Step A: Disable Block Public Access in Terraform

```
resource "aws_s3_bucket_public_access_block" "website_images" {

  bucket = aws_s3_bucket.website_images.id

  # CHANGED FROM: true → false
```

```
  block_public_acls       = false  # Changed
  block_public_policy     = false  # Changed
  ignore_public_acls      = false  # Changed
  restrict_public_buckets = false  # Changed
}
```

Step B: Add S3 Bucket Policy

```
resource "aws_s3_bucket_policy" "website_images" {
  bucket = aws_s3_bucket.website_images.id
  depends_on = [aws_s3_bucket_public_access_block.website_images]
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid       = "PublicReadGetObject"
        Effect    = "Allow"
        Principal = "*"
        Action    = "s3:GetObject"
        Resource  = "${aws_s3_bucket.website_images.arn}/*"
      }
    ]
  })
```

}

Step C: Apply Terraform

terraform apply # Type: yes

Wait for "aws_s3_bucket_policy.website_images: Creation complete"

Step D: Verify

curl -I https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg

→ HTTP/1.1 200 OK ✓

Verification in Browser:

- Hard refresh: Ctrl+Shift+R (Windows/Linux) or Cmd+Shift+R (Mac)
- Banner image now loads correctly

Why This Happened: AWS S3's "Block Public Access" is a **hard stop** security feature. Even if your bucket policy says "allow public read", Block Public Access overrides it at the account/bucket level. You must disable these settings BEFORE the bucket policy will work.

Issue #2: ACL Error When Uploading to S3

Problem:

aws s3 cp banner.jpg s3://bonefortuna-website-images-281845296706/banner.jpg --acl public-read

→ Error: The bucket does not allow ACLs

Root Cause: Attempted to use deprecated S3 ACL (Access Control List) with `--acl public-read` flag. Modern S3 buckets disable ACLs by default.

Fix Applied:

REMOVED: --acl public-read flag

aws s3 cp banner.jpg s3://bonefortuna-website-images-281845296706/banner.jpg

Result: Upload successful ✓

upload: ./banner.jpg to s3://bonefortuna-website-images-281845296706/banner.jpg

Lesson Learned: Modern AWS best practice is to use **bucket policies** (not ACLs) for access control. ACLs are deprecated and should be avoided.

Issue #3: Imgur Download Rate Limiting (429 Error)

Problem:

wget https://imgur.com/ccE6uLI.jpg -O banner.jpg

→ HTTP request sent, awaiting response... 429 Unknown Error

Root Cause: Imgur was rate-limiting direct downloads from command-line tools.

Attempted Fix #1 (Failed):

Added User-Agent header

wget --user-agent="Mozilla/5.0" https://imgur.com/ccE6uLI.jpg -O banner.jpg

→ Still rate-limited (429)

Attempted Fix #2 (Failed):

Tried curl with redirect flag

curl -L https://imgur.com/ccE6uLI.jpg -o banner.jpg

→ Downloaded 503 bytes (error page, not actual image)

Successful Fix:

Used Google Drive instead of Imgur

curl -L
'https://drive.google.com/uc?export=download&id=1rn-Tj08vo6rPiBqTAKdG-DpF7DsLH5mw' \

  -o banner.jpg

→ Downloaded 19.7 KB (actual image) ✓

Key Takeaway: Always have a backup image source. Imgur is unreliable for automated scripts. Use Google Drive, AWS S3 pre-signed URLs, or your own hosting.

Issue #4: Terraform Duplicate Resource Error

Problem:

Error: Duplicate resource "aws_s3_bucket" configuration

on s3.tf line 1: resource "aws_s3_bucket" "website_images"

A aws_s3_bucket resource named "website_images" was already declared at main.tf:265

Root Cause: Accidentally created `s3.tf` file with S3 bucket resources when they already existed in `main.tf`. Terraform doesn't allow duplicate resource names in the same module.

Fix Applied:

Delete the duplicate file

rm ~/week4-terraform-bonefortuna/s3.tf

Verify it's gone

ls -la ~/week4-terraform-bonefortuna/

Retry terraform apply

terraform apply

Best Practice: Use a consistent file organization strategy:

- `main.tf` - Core infrastructure (VPC, subnets, EC2)
- `s3.tf` - S3 buckets (or keep in main.tf if small)
- `security.tf` - Security groups, IAM roles
- `outputs.tf` - Output values
- `variables.tf` - Input variables
- `providers.tf` - Provider configuration

Do NOT mix resource definitions across files unless intentional.

Issue #5: First Image Upload Was Corrupted (503 Bytes)

Problem:

After uploading banner.jpg, checked size:

aws s3 ls s3://bonefortuna-website-images-281845296706/

→ 2026-01-31 19:11:31 503 banner.jpg

(503 bytes is too small for an image)

Root Cause: Imgur's 429 rate-limit error page was accidentally saved as "banner.jpg" instead of the actual image.

Evidence:

curl localhost | grep img

→ <img
src="https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg">

Image wouldn't load (403 error at first, then would have loaded but been invalid)

Fix Applied:

1. Downloaded proper image from Google Drive (19.7 KB)
2. Re-uploaded to S3
3. Hard-refreshed browser

aws s3 ls s3://bonefortuna-website-images-281845296706/

→ 2026-01-31 19:24:07 19688 banner.jpg ✓

Security Model

Access Control Layers

Layer	Mechanism	Enforcement
Public Access	HTTP (80) from 0.0.0.0/0	Security Group
SSH Break-Glass	SSH (22) from ONE IP only	Security Group
Engineer Access	IAM Role + SSM	IAM policy, no SSH
Private Tier	No inbound internet	Security Group
Outbound Updates	NAT Gateway	Route table
S3 Access	Bucket policy + IAM	S3 block public access

Security Groups

Public Web Security Group (`sg-00918e8657f147a84`)

Inbound Rules:

- HTTP (80) from 0.0.0.0/0 [Customers access website]
- SSH (22) from BREAK_GLASS_IP [Emergency access only]

Outbound Rules:

- All traffic to 0.0.0.0/0 [Can reach S3, updates, etc.]

Private Admin Security Group (`sg-00380241eb986644b`)

Inbound Rules:

- None (fully blocked) [Unreachable from internet]

Outbound Rules:

- All traffic to 0.0.0.0/0 [NAT allows updates]

IAM Roles for Engineers

Role	Permissions	Use Case
Week4-SSM-Admin	AmazonSSMFullAccess + AmazonEC2ReadOnlyAccess	Senior engineers, full Systems Manager access
SSM-Operator	Custom: ssm:StartSession, ssm:TerminateSession, ssm:Describe*	Operations team, start/stop sessions only
ReadOnly	ReadOnlyAccess	Auditors, monitoring, no modification capability

Systems Manager (SSM) Instead of SSH

Why SSM is Better:

- ✅ No SSH keys to manage or rotate
- ✅ Access controlled by IAM (centralized)
- ✅ Audit trail in CloudTrail (who accessed what, when)
- ✅ Works through NAT/firewalls (no inbound port exposure)
- ✅ No public IP required on private instances
- ✅ Session Manager logs all commands (security!)

SSH Risk:

SSH Key Management:

- Keys must be stored securely

- Keys must be rotated regularly

- Keys can be lost or compromised

- No built-in audit trail

- Requires inbound port 22 (security risk)

SSM Solution:

IAM Role Management:

- Roles managed centrally in AWS Console

- Audit trail automatic (CloudTrail)

- Session logs available (CloudWatch)

- No inbound ports needed

- Temporary credentials (automatic rotation)

Infrastructure Components

VPC & Networking

Resource	**CIDR/Details**	**Purpose**
VPC	10.0.0.0/16	Root network
Public Subnet	10.0.1.0/24	Web server (accessible)
Private Subnet	10.0.2.0/24	Admin server (protected)
Internet Gateway	igw-09b9a3f5105a6ece6	Route public traffic out
NAT Gateway	nat-088de78fa6c4be3e1	Route private outbound only
Elastic IP	3.226.213.35	Public IP for NAT

EC2 Instances

Instance	**Type**	**Subnet**	**Public IP**	**Role**	**OS**
Web Server	t2.micro	Public	18.208.206.23	Apache	Amazon Linux 2
Admin Server	t2.micro	Private	None	Backend	Amazon Linux 2

Storage

Resource	Size	Purpose	Access
S3 Bucket	~20 KB	Banner image	Public (policy)
Terraform State	N/A	Infrastructure tracking	Local

IAM Resources

Resource	Type	Permissions	Attached To
Week4-EC2-SSM-Role	Role	AmazonSSMManagedInstanceCore	Both EC2 instances
Week4-SSM-Admin	Role	Full SSM + EC2 read	Engineers
SSM-Operator	Role	Session start/terminate	Operations
ReadOnly	Role	Read-only access	Auditors

Testing & Validation

Test 1: Website Accessibility

```
# From your local machine

curl http://18.208.206.23

# Expected output: Full HTML page with "Bone For Tuna"
```

Test 2: Image Loads from S3

```
# Check if image URL is accessible

curl -I https://bonefortuna-website-images-281845296706.s3.amazonaws.com/banner.jpg

# Expected: HTTP/1.1 200 OK
```

Test 3: SSM Access with IAM Role

```
# Assume SSM-Operator role

aws sts assume-role --role-arn arn:aws:iam::281845296706:role/SSM-Operator \

  --role-session-name test-session

# Extract credentials and export them

export AWS_ACCESS_KEY_ID=ASIAUDH2J5ZBLH2N5YP5

export AWS_SECRET_ACCESS_KEY=fzD71iCL20Vv3Kf+eIwjaT+YB4/9YX8jOSLDx6sr

export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEPn/...

# Start session

aws ssm start-session --target i-04b63896445a24fdd --region us-east-1

# Expected: Shell prompt appears

sh-4.2$
```

Test 4: Private Server Unreachable from Internet

```
# Try to SSH to private server (should fail)

ssh -i ~/.ssh/week4-bonefortuna.pem ec2-user@10.0.2.10

# Expected: Connection timeout (unreachable)
```

Test 5: Private Server Reachable via SSM

```
# Connect to private server via SSM

aws ssm start-session --target i-096d32199f926bb55 --region us-east-1

# Expected: Shell prompt appears (SSM agent working)

sh-4.2$

# Verify it's private
```

ifconfig | grep "inet "

Expected: 10.0.2.x (private IP only)

Test 6: Private Server Can Update via NAT

SSH into private server via SSM

aws ssm start-session --target i-096d32199f926bb55 --region us-east-1

Run package update

sudo yum update -y

Expected: Packages download successfully (NAT Gateway working)

Costs & Optimization

Estimated Monthly Cost (US East 1)

Resource	Type	Hours/Month	Cost/Unit	Total
EC2 t2.micro	Web Server	730	$0.0116/hr	$8.47
EC2 t2.micro	Admin Server	730	$0.0116/hr	$8.47
NAT Gateway	Data transfer	100 GB	$0.045/GB	$4.50
NAT Gateway	Hourly fee	730 hrs	$0.045/hr	$32.85
S3 Storage	Standard	100 MB	$0.023/GB	$0.00
S3 Requests	GET requests	10,000	$0.0004/1000	$0.04
Internet Gateway	Free tier	-	-	$0.00
VPC/Subnets	Free tier	-	-	$0.00
IAM/Security Groups	Free tier	-	-	$0.00

Resource	Type	Hours/Month	Cost/Unit	Total
			TOTAL	**~$54.33/month**

Result: ✅ Well under $100/month budget

Cost Optimization Strategies

1. **Use t2 Micro Instances**

    - Eligible for AWS free tier (first 12 months)
    - Sufficient for web hosting
    - Auto-scaling ready for growth

2. **Consolidate NAT Gateway**

    - One NAT Gateway serves both subnets
    - Place in high-usage subnet to minimize data transfer

3. **S3 Lifecycle Policies** (optional)

    - Archive old images after 30 days
    - Delete after 90 days

4. **CloudWatch Monitoring**

    - Set alarms for unexpected traffic
    - Prevents runaway costs

5. **Reserved Instances** (future)

    - If traffic is predictable, 30-40% savings vs on-demand

Team Roles & Access

Team Member Assignments

Name	Role	Permissions	Access
Senior Engineer	SSM-Admin	Full Systems Manager	Both EC2s
Operations Engineer	SSM-Operator	Start/terminate sessions	Both EC2s
Auditor	ReadOnly	Read-only access	AWS Console

Break-Glass SSH Access

- **Authorized IP:** [DOCUMENT YOUR TEAM MEMBER'S IP HERE]
- **Keypair:** week4-bonefortuna.pem (stored securely)
- **Usage:** Emergency access only if Systems Manager fails
- **Rotation:** Every 90 days

Troubleshooting Guide

Problem: Website shows "connection refused"

```
# Check if Apache is running

aws ssm start-session --target i-04b63896445a24fdd --region us-east-1

sudo systemctl status httpd

# If not running, start it

sudo systemctl start httpd

sudo systemctl enable httpd
```

Problem: Image shows 403 error

```
# Verify S3 bucket policy exists
```

```
aws s3api get-bucket-policy --bucket bonefortuna-website-images-281845296706

# Verify Block Public Access is disabled

aws s3api get-public-access-block --bucket bonefortuna-website-images-281845296706

# All should be: false
```

Problem: Can't connect via SSM

```
# Verify IAM role is attached to EC2

aws ec2 describe-instances --instance-ids i-04b63896445a24fdd \

  --query 'Reservations[0].Instances[0].IamInstanceProfile'

# Verify SSM agent is running

aws ssm describe-instance-information --filters
"Key=InstanceIds,Values=i-04b63896445a24fdd"

# Should return instance details
```

Lessons Learned

1. **S3 Block Public Access is a Hard Stop** - Policy overrides don't work if Block Public Access is enabled
2. **Use Bucket Policies, Not ACLs** - Modern AWS best practice
3. **Systems Manager is More Secure Than SSH** - Centralized IAM, audit trails, no key management
4. **Test External Dependencies** - Imgur rate-limiting taught us to use reliable sources
5. **Organize Terraform Files Consistently** - Prevents duplicate resource errors
6. **Hard Refresh Browser for Cache Issues** - S3 URLs may be cached by browser

---

Deployment Checklist

- ☐ IAM roles created (4 total)
- ☐ Terraform initialized and planned

- [ ] Infrastructure deployed (terraform apply)
- [ ] SSH keypair downloaded
- [ ] Session Manager plugin installed
- [ ] Apache installed and running
- [ ] Banner image uploaded to S3
- [ ] Website HTML created
- [ ] Website tested in browser
- [ ] SSM access tested
- [ ] Private server verified unreachable via SSH
- [ ] Private server verified reachable via SSM
- [ ] Documentation completed
- [ ] GitHub repository pushed

References & Documentation

- AWS S3 403 Forbidden Troubleshooting
- AWS Systems Manager Session Manager
- Terraform AWS Provider S3
- AWS VPC Best Practices

Project Completed: January 31, 2026
Environment: AWS US East 1 Region
Infrastructure as Code: Terraform 1.x