Project Documentation

---

1. Project Title: SECURE SERVER ACCESS
(Website Deployment in Private EC2 with NGINX Reverse Proxy in Public EC2)

2. Objective:
To securely host a website on a private EC2 instance within a VPC and make it publicly accessible through a public EC2 instance configured as a reverse proxy using NGINX. This architecture ensures restricted access to backend resources, enhances network isolation, and follows cloud security best practices.

---

3. Duration: 6 Days

---

4. Daily Work Summary:

Day 1 – Project Planning & VPC Setup

On the first day, we planned the project workflow and defined the security-focused architecture to isolate public and private resources. The goal was to host the website on a private server and expose it securely via a public proxy.

Tasks Completed:

- Defined the objective and listed components needed (EC2, VPC, Subnets, IGW, Security Groups).
- Created a new custom VPC (CIDR: 10.0.0.0/16).
- Set up two subnets:
    - Public Subnet (10.0.1.0/24) for Bastion Host and Reverse Proxy
    - Private Subnet (10.0.2.0/24) for the Web Server
- Launched Internet Gateway (IGW) and attached it to the VPC.
- Created route tables:
    - Public Subnet route table directed 0.0.0.0/0 traffic to IGW
    - Private Subnet route table remained isolated (no outbound path)
- Planned IP address ranges and availability zones.

Observations:

- VPC-level isolation allows more control over communication paths.
- IGW and routing rules form the base for external access in AWS networks.

Challenges:

- Understanding subnet relationships and default route table behavior.

Outcome:

- A secure, isolated VPC design ready for resource deployment.

Day 2 – EC2 Instance Provisioning

With the VPC and subnet design completed, the second day focused on launching and configuring the EC2 instances. Each instance was created with the correct subnet and security group.

Tasks Completed:

- Launched three EC2 instances:
    - Bastion Host (Amazon Linux 2, t2.micro) in Public Subnet
    - NGINX Reverse Proxy (Ubuntu 22.04 LTS, t2.micro) in Public Subnet
    - Web Server (Amazon Linux 2 or Ubuntu, t2.micro) in Private Subnet
- Allocated Elastic IPs for Bastion Host and Reverse Proxy.
- Associated EIPs to ensure consistent public IP addressing.
- Created three custom Security Groups:
    - Bastion SG: SSH (22) allowed only from our IP
    - Proxy SG: HTTP (80) from 0.0.0.0/0, SSH (22) from Bastion only
    - Web Server SG: HTTP (80) from Proxy's private IP range or SG, SSH (22) from Bastion only

Observations:

- Using Elastic IPs helps ensure stable access during configuration and testing.
- Security Groups provide stateful firewall protection at instance level.

Challenges:

- Ensuring that only necessary ports were open for each component.

Outcome:

- All instances provisioned and reachable as per defined access controls.

Day 3 – SSH Configuration and Web Server Deployment

On the third day, we focused on enabling secure SSH access and configuring the web server that would host our website content in the private subnet.

Tasks Completed:

- Used key pairs to securely SSH into the Bastion Host from a local machine.
- From the Bastion Host, connected to the Private Web Server using its private IP.
- Updated the OS and installed a web server (Apache or NGINX).
- Created a simple static HTML website (index.html).
- Tested web server functionality by running curl from the Bastion Host to ensure the web server was responding correctly.

Additional Configurations:

- Verified permissions and directory structure under /var/www/html or /usr/share/nginx/html depending on the web server used.
- Ensured the firewall on the private instance (if any) allowed HTTP traffic.

Observations:

- Direct access to private instances from the internet is completely blocked — a key design feature for security.
- Bastion Host serves as a secure SSH gateway and works as expected.

Challenges:

- Ensuring correct file permissions and confirming HTTP port availability.

Outcome:

- The private EC2 instance is running a functioning website, accessible only from inside the VPC.

Day 4 – NGINX Reverse Proxy Setup

The fourth day was dedicated to configuring NGINX on the public EC2 instance to act as a reverse proxy.

Tasks Completed:

- Installed NGINX on the Public EC2 instance using apt or yum.
- Edited the main NGINX configuration file (/etc/nginx/nginx.conf) to define an upstream pointing to the Private EC2's private IP.
- Configured the server block to listen on port 80 and proxy traffic to the backend web server.

Sample Configuration:

```
http {
   upstream backend {
      server 10.0.2.X:80; #Private IP of the Web Server
   }

   server {
      listen 80;
      location / {
         proxy_pass http://backend;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      }
   }
}
```

Testing:

- Used curl and browser to access the public IP of the Reverse Proxy.

- Confirmed that the request was forwarded correctly to the Web Server and content was returned.

Observations:

- NGINX reverse proxying helps to separate front-end and backend responsibilities.
- Adding headers like X-Forwarded-For retains the original client IP for logging.

Challenges:

- Small NGINX syntax errors can cause service failure. Always use nginx -t to test config.

Outcome:

- Reverse proxy successfully forwards requests from the internet to the secure backend web server.

Day 5 – Testing & Validation

The fifth day focused on end-to-end testing of network connectivity, access control, and web accessibility via the reverse proxy. This phase was critical to ensure the architecture functioned securely and as intended.

Tasks Completed:

- Verified SSH path:
    - SSH from local system to Bastion Host (public IP) using key pair
    - SSH from Bastion Host to Web Server using internal private IP
- Website Access Test:
    - Used browser to access Public EC2 IP (reverse proxy)
    - Verified content was fetched from Web Server in Private Subnet
- curl Test:
    - Ran curl http://localhost from Reverse Proxy
    - Verified response headers and HTML output from Private EC2
- Firewall Validation:
    - Attempted to access Private EC2 directly from local machine (FAILED — as expected)
    - Ran netcat (nc) test on ports to verify open/closed ports

- Security Group Review:
    - Confirmed port 80 open on Reverse Proxy for public access
    - Confirmed port 80 open on Web Server only to Proxy SG
    - Confirmed SSH access (port 22) open only between Bastion and Web/Proxy
- Restart and Recovery:
    - Restarted NGINX and confirmed service auto-recovery and website availability

Observations:

- Direct access to the Private EC2 was blocked at the network level — a key success indicator.
- The use of the reverse proxy ensured isolation of backend resources from the internet.
- curl and browser-based access worked exactly as intended.
- SSH path through Bastion was stable and secure.

Challenges:

- Reverse proxy config must always point to the correct IP; mismatches result in 502 errors.
- SSH agent forwarding can cause confusion when reusing private keys across hops.

Outcome:

- All test cases executed and passed.
- Secure website access confirmed via reverse proxy only.
- Private web server remained hidden and protected throughout testing.

Day 6 – Documentation & Finalization

The final day was devoted to documentation creation, cleanup, and presentation slide design. The goal was to compile all findings, designs, configurations, and tests into a deliverable format for submission and demonstration.

Tasks Completed:

- Created full test documentation:
    - Test Plan (AWS_Project_Test_Plan.docx)

    - Test Case Table (AWS_Test_Cases.xlsx)
    - Security Group Summary Table
- Prepared architecture diagram:
    - Included VPC, subnets, EC2 instances, route tables, and IGW
    - Labels for IP ranges, ports, and direction of access
- Prepared PowerPoint presentation:
    - Slides: Project Objective, Architecture, Components Used, NGINX Role, Testing, and Summary
    - Added simplified explanation for non-technical audience
- Wrote complete project summary:
    - Problem Statement
    - Solution Overview
    - Key AWS Services Used
    - Why Reverse Proxy and Bastion are important
- Created this full 6-day report.
- Re-tested all components before final screenshots and documentation export

Observations:

- Documenting during the project made this final phase faster and more accurate.
- Canva was used to prepare presentation content cleanly and clearly.

Challenges:

- Formatting test plan in Word to ensure it looked professional.
- Aligning IP ranges and confirming architecture diagram matched actual deployment.

Outcome:

- Complete and well-documented project folder prepared
- Ready for submission and mentor review
- Confident understanding of AWS networking, EC2 configuration, NGINX proxying, and access control

---