Week 4 Cloud Security Project

Building a Secure Website in the Cloud

What I Built

I built a secure website for a fictional European shipping company, but instead of hosting it on a regular computer server, I put it *in the cloud* (specifically, Amazon's cloud service called AWS). This is similar to how companies like Netflix, Airbnb, and Spotify run their websites, not on computers sitting in their offices, but on powerful computers owned by Amazon that can be rented and controlled over the internet.

The key challenge wasn't just making a website work, but making it **secure**. Imagine building a house where customers can walk in the front door, but employees need a separate, more secure entrance, and some rooms are completely locked off from the internet. That's what I created, but with computers instead of rooms.

Think of It Like This

A Modern Office Building

Imagine you're designing a modern office building with multiple security levels:

- **The Lobby (Public Website):** Anyone can walk in off the street. There's a reception desk showing company information and photos. This is like the public website I created, anyone on the internet can visit it.
- **The Office Area (Private Server):** Only employees with badge access can enter. They can come and go for work, but random people from the street can't just walk in. This is my private administrative server—only authorized staff can access it.
- **The Loading Dock:** Staff can receive deliveries from outside (like software updates), but delivery trucks can't drive through the front lobby. This is like the special one-way internet connection I set up for updates.

Why This Matters

When companies get hacked, it's often because everything is connected. If hackers break into the public website, they can sometimes sneak into the private systems too. My design prevents this. Even if someone broke into the public website, they **cannot** reach the private administrative server because there's **no connection** between them from the internet.

The Three Main Pieces

1. The Public Website

This is what customers see when they visit the website. I set up:

- A computer in the cloud running a web server (like a digital cashier that hands out web pages)
- Company information and images stored separately in cloud storage (think of it like keeping product photos in a warehouse instead of at every cash register)
- Security rules that only allow website visitors to see the website—they can't poke around in the computer's files or settings

2. The Private Admin Server

This is where company staff would do their work. The key features:

- **No internet address:** This computer doesn't have a phone number that the internet can call. It's invisible to hackers.
- **Special secure access:** Staff can access it through Amazon's secure system (imagine a private tunnel), not through the regular internet.
- **One-way updates:** The server can download security updates from the internet, but the internet can't initiate contact with it.

3. Permission Levels

Just like how a company has different employee badges (visitor, employee, manager, executive), I created three levels of access:

- **Read-Only:** Junior staff can look but can't touch anything.
- **Operator:** Regular engineers can log in to servers to do daily work.
- **Admin:** Senior engineers can change system settings and configurations.

How I Built It

Instead of clicking around Amazon's website to manually create everything (which would take hours and be error-prone), I wrote *code* that automatically builds the entire setup.

Think of it like the difference between:

- **Manual:** Typing up a document by hand every time you need it (slow, mistakes)
- **Automated:** Having a template you can print perfectly every time with one click

This code-based approach (called "Infrastructure as Code") means I can:

- Recreate the entire system in minutes if something breaks
- Make changes safely and test them first
- Share the exact setup with teammates
- Keep a history of every change made

Security Features Explained Simply

Network Separation
Imagine a hotel with two separate wings: one for guests (public) and one for staff (private). Even though they're in the same building, there's no door connecting them. That's what I built with my "public subnet" and "private subnet."

Virtual Firewalls
These are like bouncers at a club. The public website's firewall says: "You can look at the website (port 80) and that's it." The private server's firewall says: "Nobody gets in from the internet. Period."

One-Way Internet (NAT Gateway)
This is like a mail slot. The private server can send requests out ("Please send me security updates"), and responses come back through the slot. But nobody outside can push things *in* through the slot uninvited.

Identity-Based Access
Instead of using passwords (which can be stolen), staff access servers using digital credentials that Amazon's system verifies, similar to how your company badge works—it's tied to you personally and can be revoked instantly if needed.

Real-World Comparison

What Most Small Companies Do:
- Everything is connected to the internet
- Everyone uses the same password or has full access
- If hackers get in one place, they can access everything

What I Built (Enterprise-Level):
- Public and private systems are completely separated
- Each person has specific permissions for their role
- Multiple layers of security ("defense in depth")
- All access is logged and traceable

Testing It Works
To prove everything was set up correctly, I:

- **Public website test:** Opened a browser and visited the website—it worked! Anyone on the internet could see it.
- **Private server test:** Tried to access it from the internet; correctly blocked. Then used Amazon's secure access system; it worked! Only authorized users can get in.

- **Security test:** Verified that even though I was logged into the public website server, I had no way to reach the private server. The separation works.

Why This Project Matters for My Career

This week's project was different from the previous three weeks. Before, I was learning *how* to build cloud infrastructure. This week, I learned how to build it **securely**, the way real companies do it to protect customer data and meet compliance requirements.

What I demonstrated:

- **Security thinking:** Designing systems where even if one part is compromised, damage is limited
- **Professional practices:** Using automation instead of manual work
- **Enterprise standards:** Implementing access controls and audit trails that meet compliance requirements
- **Team collaboration:** Working with a partner to deploy into a shared environment

The Bottom Line

I built a secure, professional-grade website hosting system in Amazon's cloud that:

- Separates public-facing and private systems for security
- Controls who can access what based on their job role
- Can be rebuilt automatically if needed
- Meets enterprise security standards
- Costs less than $100 per month to run

This is the same type of infrastructure that powers modern web applications, from small businesses to Fortune 500 companies. The difference is scale, not technique.

From On-Premise to Enterprise Cloud Security

Week 1: Built a server on my own computer

Week 2: Moved it to the cloud manually

Week 3: Automated the deployment with code

Week 4: Secured it like a professional