Portfolio Project:

Hardening & Utilizing a Cloud Server with OCI

This document details the process, justification, and technical execution of deploying a secure, public-facing Ubuntu Linux server and subsequently configuring it as a personal cloud storage solution. The project demonstrates a defense-in-depth security strategy and practical system integration, incorporating lessons learned from real-world troubleshooting.

Module 1: Foundational Setup & Key Management

Objective: To establish a secure and reliable foundation on the local machine (Kali Linux) and within the OCI console, ensuring that all subsequent steps are built upon a solid base.

Step 1.1: Local Environment Preparation (Kali Linux)

- **Action:** Open the Kali terminal and execute `rm -f ~/.ssh/*` to clear any existing SSH keys.
- **Justification:** We started with a clean SSH directory to prevent any conflicts from old or temporary keys created during previous troubleshooting sessions. This ensures that the key we generate is the definitive one for this project, eliminating a common source of "Permission denied" errors.

Step 1.2: Generate Primary SSH Key

- **Action:** Execute `ssh-keygen -t ed25519` and accept the default prompts.
- **Justification:** We chose the `ed25519` algorithm because it offers better security and performance compared to older types like RSA. It creates a public/private key pair, which is the modern standard for secure, passwordless authentication. This prevents brute-force password attacks, a common threat for public-facing servers.

Step 1.3: Prepare the Public Key

- **Action:** Execute `cat ~/.ssh/id_ed25519.pub` and copy the output.
- **Justification:** The `cat` command displays the contents of the public key file. This key is the "lock" that we will install on the server. It's safe to share and allows anyone with the corresponding private key (which never leaves the local machine) to gain access.

Module 2: Cloud Deployment & Troubleshooting

Objective: To successfully provision a virtual server instance in the cloud, navigating common platform-specific challenges like resource availability and network configuration.

Step 2.1: Instance Configuration

- **Action:** In the OCI console, navigate to **Compute -> Instances** and begin the "Create instance" workflow.
- **Justification:** This is the standard procedure for provisioning a virtual machine (VM), which acts as our server.

Step 2.2: Navigating Resource Constraints

- **Action:** In the "Placement" section, be prepared to change the **Availability Domain**. In the "Shape" section, select `VM.Standard.A1.Flex` if available, or the `VM.Standard.E2.1.Micro` as a viable alternative.
- **Justification:** We discovered that the OCI Free Tier has fluctuating resource availability. The "Out of capacity" error is a platform constraint, not a user error. By changing the Availability Domain (the physical data center room) or selecting the alternate Micro shape, we can find available resources and successfully launch the instance.

Step 2.3: Initial Network Configuration

- **Action:** In the "Networking" section, select **"Create new virtual cloud network"** if the existing one is unavailable. Ensure **"Assign a public IPv4 address"** is checked.
- **Justification:** Our troubleshooting revealed that a new account may not have a pre-configured network ready. Creating a new Virtual Cloud Network (VCN) is the direct solution. A public IPv4 address is essential for the server to be reachable from the internet.

Step 2.4: Secure Key Installation

- **Action:** In the "Add SSH keys" section, paste the copied `ed25519` public key.
- **Justification:** This is the most critical step for ensuring a successful first connection. By pasting the public key here, we are instructing the server to trust our local Kali machine. A precise copy-paste is crucial, as any error here is the primary cause of the "Permission denied (publickey)" error.

Module 3: System Initialization & Maintenance

Objective: To establish a stable and secure operating environment by connecting to the server and bringing its software to a fully supported, up-to-date state.

Step 3.1: First Connection

- **Action:** From the Kali terminal, execute `ssh ubuntu@<YOUR_NEW_PUBLIC_IP>`. Type `yes` when prompted about the host's authenticity.

- **Justification:** This action verifies that the key pair authentication is working correctly. The authenticity prompt is a standard security feature to prevent man-in-the-middle attacks; by typing `yes`, we tell our local machine to trust this new server.

Step 3.2: System Package Upgrade

- **Action:** Run `sudo apt update` followed by `sudo apt upgrade -y`.
- **Justification:** Before any major changes, we ensure all currently installed software packages are updated to their latest versions. This patches known vulnerabilities and ensures system stability before proceeding with further hardening.

Module 4: System Stability Verification (Critical Finding)

Objective: To proactively prevent loss of remote access by verifying and correcting a critical service configuration, a step identified through root cause analysis of a previous failed deployment.

Step 4.1: Verify SSH Service Status

- **Action:** Before rebooting the server, execute `sudo systemctl is-enabled ssh`.
- **Justification:** During a previous attempt, we experienced a "Connection refused" error after a reboot. Root cause analysis determined the SSH service was not configured to start automatically on boot. This verification step was added to the process to proactively check for this specific misconfiguration.

Step 4.2: Enable SSH Service

- **Action:** The check revealed the service was `disabled`. The command `sudo systemctl enable ssh` was executed to correct this. A subsequent check confirmed the status was now `enabled`.
- **Justification:** This action directly remediates the identified fault. By enabling the service, we ensure the SSH daemon will launch automatically after a reboot, guaranteeing persistent remote access. This step demonstrates an iterative improvement to the deployment process based on hands-on troubleshooting experience.

Step 4.3: System Reboot

- **Action:** Execute `sudo reboot`.
- **Justification:** A reboot is required to apply some of the package updates from Step 3.2. With the SSH service now confirmed to be enabled, this action can be performed with confidence.

Module 5: Multi-Layered Security Implementation

Objective: To construct a robust "Defense in Depth" security posture where multiple independent layers protect the server from threats.

Step 5.1: Layer 1 - The Cloud Firewall (NSG)

- **Action:** In the OCI console, create a Network Security Group (NSG) named `cybersec-nsg` and associate it with the server's network interface (VNIC).
- **Justification:** The NSG acts as the first line of defense. It filters traffic at the cloud level before it ever reaches our server. This is more efficient and secure than relying solely on a server-based firewall. We use an NSG instead of a Security List for more granular, component-level control.
- **Action:** Add two ingress rules: one to allow SSH (port 22) only from your specific home IP, and another to allow web traffic (ports 80, 443) from anywhere.
- **Justification:** This enforces the **principle of least privilege**. We deny all traffic by default and only open what is absolutely necessary. Restricting SSH access to a single IP address drastically reduces the attack surface for unauthorized login attempts.

Step 5.2: Layer 2 - The OS Firewall (UFW)

- **Action:** Upon attempting to run `ufw`, a `command not found` error occurred. The `ufw` package was manually installed with `sudo apt install ufw -y`.
- **Justification:** This finding shows that the minimal server image provided by OCI does not include all standard tools. Identifying and installing the missing package is a necessary step to adapt to the specific environment and implement the intended security layer.
- **Action:** On the server, execute the `ufw` commands to set a default-deny policy and explicitly allow SSH, HTTP, and HTTPS traffic.
- **Justification:** This creates a second, redundant firewall layer. If the cloud-level NSG were ever misconfigured, the OS firewall would still protect the server. This redundancy is a core concept of defense in depth.

Step 5.3: Layer 3 - Service Hardening (SSH)

- **Action:** Edit `/etc/ssh/sshd_config` to set `PasswordAuthentication no` and `PermitRootLogin no`. Restart the `ssh` service.
- **Justification:** This further hardens the most critical remote access point. Disabling password authentication makes brute-force attacks impossible, as a valid key is the only way to log in. Disallowing direct root login forces administrators to log in as a standard user and elevate their privileges, which creates a better audit trail and adds another layer of protection.

Step 5.4: Layer 4 - Active Intrusion Prevention (Fail2Ban)

- **Action:** Install the `fail2ban` package using `sudo apt install fail2ban -y`.
- **Justification:** This moves from passive defense to active threat mitigation. `Fail2Ban` constantly monitors logs for malicious patterns (like repeated failed login attempts from a

single IP). When it detects a threat, it automatically updates the OS firewall to block the attacker's IP address for a set period. This provides automated, real-time protection against brute-force attempts that might target other services.

Module 6: Secure Personal Cloud Storage Implementation

Objective: To leverage the hardened server by configuring it as a secure, personal cloud storage solution accessible via a drag-and-drop interface from a Windows desktop.

Step 6.1: Server-Side Preparation

- **Action:** On the server, execute `mkdir ~/storage`.
- **Justification:** This command creates a dedicated directory to house the cloud storage files, keeping them organized and separate from system files within the user's home directory.

Step 6.2: Client-Side Configuration

- **Action:** Install and configure WinSCP on the Windows 11 host machine, setting the protocol to SFTP and pointing it to the server's IP address.
- **Justification:** We chose SFTP (SSH File Transfer Protocol) because it uses the existing, highly secure SSH connection for file transfers. This means all data is encrypted in transit, and authentication is handled by the same robust SSH key pair we already configured. No new ports need to be opened, and no new software is needed on the server, minimizing the attack surface. WinSCP is a trusted Windows client for this protocol.

Step 6.3: Cross-Environment Key Integration (Critical Finding)

- **Action:** An initial connection attempt from WinSCP failed with a "Server refused our key" error, despite the key working from the Kali terminal. The root cause was identified: the Kali (WSL) environment and the native Windows environment maintain separate SSH key stores. The command `cp ~/.ssh/id_ed25519 /mnt/c/Users/<your_username_here>/.ssh/` was executed in the Kali terminal to resolve this.
- **Justification:** This action copied the correct private key from the Kali file system to the Windows file system. This made the key accessible to WinSCP, a native Windows application. This step is a crucial demonstration of understanding and troubleshooting the interaction between WSL and the host operating system, a common challenge in modern development environments.

Step 6.4: Final Result

- **Action:** Create a desktop shortcut for the saved WinSCP session.

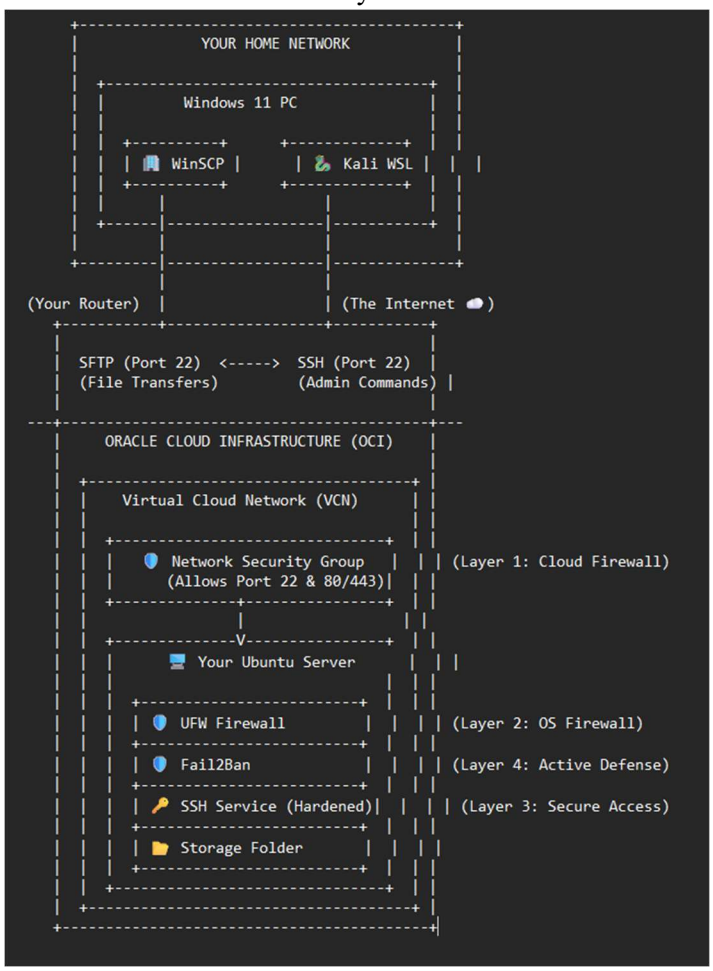
- **Justification:** The final result is a one-click desktop icon that launches a secure, encrypted, drag-and-drop file transfer session with the personal cloud server, achieving the project's practical goal.

Troubleshooting Log & Key Learnings

- **Issue:** SSH connection drops during a `nano` editing session, leaving a `.swp` lock file.
- **Resolution:** The leftover lock file (`/etc/ssh/.sshd_config.swp`) must be manually deleted with `sudo rm` before the real configuration file can be edited again. This is a common issue when administering servers over an unstable network connection and demonstrates a practical recovery skill.
- **Issue:** Initial connection attempts fail with "Permission denied (publickey)".
- **Resolution:** This is almost always caused by an error in pasting the public key during instance creation or a lack of a key on the local machine. The most reliable solution is to terminate the instance and create a new one, ensuring the key is copied and pasted correctly from the start.
- **Issue:** "Connection refused" after a major OS upgrade.
- **Resolution:** Root cause was determined to be the SSH service being `disabled` and not starting on boot. The permanent fix is to proactively run `sudo systemctl enable ssh` before any reboot to ensure persistent remote access.

network architecture diagram

Explanation of the Architecture



1. **Your Home Network:** This is your local environment. Inside your Windows 11 PC, you have two key components for this project:
    - **Kali WSL:** This is your command-line environment where you run `ssh` to administer the server.
    - **WinSCP:** This is your Windows application that provides the graphical interface for dragging and dropping files.
2. **The Internet:** This is the public network that connects your home to the Oracle Cloud. All your traffic travels over it.
3. **The Connections (Port 22):** You have two types of connections, but they both securely connect to the **same SSH service on port 22** on your server.
    - **SSH:** Used for sending administrative commands from Kali.
    - **SFTP:** Used for transferring files with WinSCP. It's a subsystem of SSH, which is why it's so secure.
4. **Oracle Cloud Infrastructure (OCI):** This is the data center where your server lives.
5. **Virtual Cloud Network (VCN):** This is your own private, isolated section of the Oracle Cloud.
6. **Network Security Group (NSG) - Layer 1:** This is the **first firewall**. It acts as a gatekeeper for your entire VCN, only allowing traffic on the ports you specified (22 for SSH/SFTP, and 80/443 for potential web traffic) and only from the IP addresses you allowed.
7. **Your Ubuntu Server:** This is the virtual machine itself. Inside it are the other layers of your defense:
    - **UFW Firewall - Layer 2:** The **second firewall**. It provides another layer of protection in case the NSG is ever misconfigured.
    - **Hardened SSH Service - Layer 3:** This is the secure door that both your Kali terminal and WinSCP knock on. It only accepts your specific SSH key as proof of identity.
    - **Fail2Ban - Layer 4:** This is the active security guard that watches for suspicious activity and blocks attackers automatically.
    - **Storage Folder:** This is the dedicated directory where your files live, accessible via SFTP.