IM3014 Linux System Administration Practice
Homework 4: One-Man Army

Introduction

After two years of service as a junior system administrator at **IM Corp**, you have now been promoted to a senior system administrator. With higher pay comes greater responsibility. You are no longer just performing routine configurations — you are now expected to deliver complete, end-to-end solutions that support the company's growing operations.

In the upcoming quarter, **IM Corp** plans to expand into the gaming industry. As part of this strategic move, your manager has assigned you two primary tasks:

1. Host a simple online multiplayer 2D shooter game and evaluate its business potential.
2. Investigate modern internet infrastructure to prepare for future service deployments.

By completing this assignment, you will gain practical experience in hosting web-based applications and exploring the backbone of modern internet infrastructure — becoming a true **one-man army** in system administration.

Problem Specification

Online Multiplayer Game

This section requires a basic understanding of JavaScript (and some TypeScript), HTML, CSS, and the Node.js ecosystem. If you are not familiar with web development, it is strongly recommended that you start early and explore online learning resources, such as the *Web Programming Course* offered by the EE Department, available on YouTube.

Developing a new game from scratch is time-consuming and requires extensive domain-specific knowledge in areas such as game engines (e.g., Unity 3D), 3D modeling, art design, and more. Such a project typically involves a team of engineers, artists, and a game director working for months—or even years—with significant dedication. Therefore, you decide to begin by using an open-source game as the baseline for this business initiative. You have identified a promising candidate: a 2D multiplayer shooting game featuring cartoon-style graphics and simple controls, available on GitHub. The game only requires Node.js to run, follow the steps below:

1. **Clone the Repository from GitHub.**
   Clone the game repository to your VM from GitHub
2. **Run the Game Server Locally.**
   Install the required packages then run the game server locally to ensure it starts correctly.
   `Verification:` Run `npm install` followed by `npm start`. Ensure there are no dependency errors or missing modules.
3. **Configure as a Systemd Service.**
   Configure the game server as a `systemd` service, and forward inbound traffic from port 80 to the desired game server port.
   `Verification:` Provide the configuration files for `systemd` and `nginx`.
4. **Verify External Accessibility.**
   Verify that you can access the game from your PC. Note that the port forwarding rules on your VM are: 80 → 80XY and 8000 → 90XY.
   `Verification:` From your PC, access `http://<domain-of-the-VM>:<assigned-port-of-the-VM>` in a browser. Provide screenshot of the game loaded successfully.

5. **Enable HTTPS.**
   Secure the connection on port 8000 using HTTPS. The remaining configurations can follow the same setup as before.
   `Verification:` Provide screenshot that `https://<domain-of-the-VM>:<assigned-port-of-the-VM>` loads the game without certificate errors.

6. **Record Gameplay Demonstration.**
   Record a video demonstrating at least two players playing the game simultaneously via HTTPS.

7. **Benchmark Using Selenium (1–10 Users).**
   To benchmark the server's user capacity, first use a tool such as `Selenium` to simulate real player behavior. Configure each simulated browser instance to send randomized player actions at regular intervals, mimicking actual gameplay interactions. Record changes in FPS and network latency (average, P95, P99, P99.5) as the number of simulated users increases from 1 to 10, measured over a two-minute gameplay session. Finally, evaluate whether this approach provides a reliable benchmark, noting that browser-based simulations are resource-intensive and may significantly impact your PC's performance. `Verification:`

   - Plot two multi-line graphs (average, P95, P99, P99.5) with the user count (1–10) on the x-axis. The first graph should show FPS on the y-axis, and the second should show latency (in milliseconds) on the y-axis.
   - Provide the source code for the benchmark, which may be written in any programming language of your choice.

8. **Benchmark Using WebSocket Clients (1–1000 Users).**

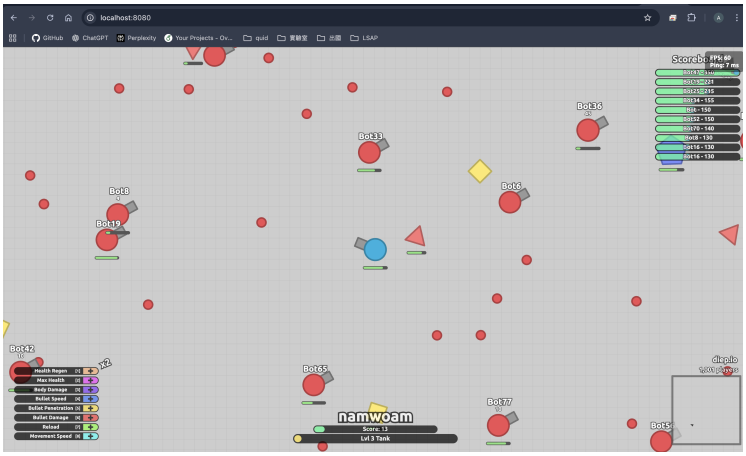

Figure 1: A browser client rendering a 1,000-player game.

Next, benchmark the server's capacity using a custom headless WebSocket client that simulates player actions. By eliminating graphical rendering, this headless design significantly reduces CPU usage, allowing a single PC to emulate a much larger number of concurrent clients. The WebSocket client should send randomized player actions at regular intervals to mimic realistic gameplay behavior, but without rendering the game interface. Record latency metrics (average, P95, P99, P99.5) as the number of simulated users increases from 1 to 1,000 in increments of 100, each measured over a two-minute gameplay session. Since no full browser is used in this experiment, FPS data will not be available. Figure 1 illustrates an example screenshot from the 1,000-player test.

Note: Do *not* run this benchmark for extended periods, as compute resources are shared among VMs on the same machine

`Verification:`

- Plot a multi-line graph (average, P95, P99, P99.5) with the user count (1–1000) on the x-axis and latency (in milliseconds) on the y-axis.
- Provide the source code for the benchmark, which may be written in any programming language of your choice.
- Include a short video demonstrating gameplay with approximately 1000 simulated bots. The exact number may vary slightly, as the server may be unstable under heavy load.

9. **Determine Optimal User Capacity.**
   As a general rule of thumb, maintaining P90 latency below 100 ms ensures smooth gameplay. Based on your measurements, determine the optimal number of concurrent users each VM can support, and analyze whether this limit is constrained by network bandwidth, CPU performance, or memory size.

10. **Evaluate Financial Feasibility.**
    The performance of your VM is roughly equivalent to an `AWS t4g.small` instance, priced at USD 0.0168 per instance per hour. Estimate whether the game would be financially profitable if each user is served one advertisement every 20 minutes of gameplay, assuming a revenue of USD 13 per thousand video ads (source: BidsCube). If the results indicate profitability, propose the next strategic steps for **IM Corp**.

Investigating Modern Internet Infrastructure

Alongside the multiplayer game setup, your supervisor has also requested that you study how to efficiently deliver the game to players around the world. To become familiar with modern networking infrastructure, you are required to complete the following tasks:

1. **Domain Analysis.**
   List ten popular websites that you visit frequently. For each website, write a script that queries its DNS records and collects the following information:

   - IP address (A or AAAA record)
   - Canonical name (CNAME record)
   - Mail server (MX record)
   - DNS Security Extensions (DNSSEC) status

   `Verification:`

   - A table summarizing the collected DNS information for all ten websites.
   - A visual diagram or textual trace illustrating the complete DNS lookup path.
   - The script used to generate the results.

2. **DNS Resolution Time Measurement.**
   Measure the DNS resolution time for each of the ten websites using tools such as `dig` or `nslookup`.
   `Verification:`

   - Present a table or bar chart comparing the average DNS resolution times across all websites.
   - The script used to generate the results.

3. **DNS Load Balancing.**
   Query the same domain multiple times and check whether different IP addresses are returned, indicating the presence of load balancing.
   `Verification:`

   - Include the script used for this test and example outputs showing different IP responses.

4. **CDN Identification.**
   Determine whether each website is hosted behind a CDN service. If so, identify the CDN provider (e.g., Cloudflare, Akamai, Fastly).
   `Verification:` Use tools such as `dig` or `whois` to confirm the CDN provider and edge server locations.

5. **Network Performance Monitoring.**
   Measure the packet latency and throughput when connecting to each of the ten websites. For each site, record the following metrics:

   - Average round-trip latency (ms)
   - Packet loss rate (%)
   - Download throughput (Mbps)

   `Verification:`

   - Present a table summarizing latency, packet loss, and throughput for all ten websites.
   - The script used to generate the results.

6. **Network Routing Path Analysis.** Choose one website from your list and investigate how network traffic is routed from your machine to the destination host. Use tools such as `traceroute` to identify all intermediate hops along the path, including routers, gateways, and Internet Service Providers (ISPs). For each hop, record the following information: `IP Address`, `Hostname`, `Organization`, `Country`, `Location`, and `Average Latency (ms)`. `Verification:`

   - Provide a route diagram showing the sequence of all hops with their corresponding attributes.
   - Include the script used to generate the results.

7. **Backend Server Investigation.**
   Investigate the web server software used to serve content for each website (e.g., Nginx, Apache, LiteSpeed, or proprietary cloud services).
   `Verification:`

   - Summarize the backend server technologies detected for each website in a comparison table.
   - Include the script used to generate the results.

Deliverables

You are required to submit the following materials:

1. **Video:** Submit two video files named `<your-student-id>-2-ply.mp4` and `<your-student-id>-1k-ply.mp4`, as described in the first section.

2. **System Setup Report:** Prepare a report (maximum length: 8 pages) that includes:

   - a detailed description of how you set up and benchmarked the Online Multiplayer Game system;
   - an explanation of how you investigated the modern internet infrastructure;
   - screenshots or command outputs demonstrating that the system operates correctly.

Your submission should be concise, technically clear, and sufficient to allow another system administrator to reproduce your setup.