Minecraft in Cloud – Final Project Doc

Author: Arthur Faye

Student ID: 35605248

EC2 Public IP: 16.176.11.30 **Domain Name (DNS):** minecraftincloud.com

GitHub Repo: https://github.com/Arthur20-2410/minecraft-in-cloud (https://github.com/Arthur20-2410/minecraft-in-cloud)

---

1. Overview

MinecraftInCloud is a cloud-hosted platform that enables hosting and controlling a Minecraft Java Edition server via a web interface. The project was born from the challenge of managing a multiplayer server from home, where issues like sleep mode, bandwidth limitations, or security concerns made gameplay with friends unstable.

This solution offers a remote cloud-based instance on AWS EC2 with a browser-accessible control panel to start, stop, and monitor the server.

Key Features:

- Secure user login and account creation system
- Real-time Minecraft server control (start/stop)
- Live WebSocket-based console log viewer
- World backup system with recovery & restore options
- Lightweight user and session management using JSON
- Apache2 reverse proxy for domain access and SSL

Technologies used:

- `Node.js`, `Express.js` – for building the backend
- `Socket.IO` – for real-time messaging and communication
- `Apache2` – acting as a reverse proxy for domain and SSL support
- `PM2` – to ensure continuous availability of the Node.js server, enables to start, update and stop the server when we want.
- JSON-based file storage (users, sessions)

---

2. Server Setup and Deployment

Full Manual Setup

1. Prepare a EC2 instance
    - Create a new Ubuntu 22.04 EC2 instance on AWS
2. Open the following **ports in the security group**:
    - `22` (SSH)
    - `80` (HTTP)
    - `443` (HTTPS, if DNS)

    - `25565` (Minecraft Server)

3. Connect via SSH:

```
❯ ssh -i your-key.pem ubuntu@16.176.11.30 (EC2 ip)
```

4. Update the system and install dependencies

```
❯ sudo apt update && sudo apt upgrade -y
❯ sudo apt install -y apache2 nodejs npm git
```

5. Install PM2

```
❯ sudo npm install -g pm2
```

6. Clone the project repository

```
❯ git clone https://github.com/Arthur20-2410/minecraft-in-cloud ~/minecraft-in-cloud
❯ cd ~/minecraft-in-cloud
```

7. Install Node.js dependencies

```
❯ npm install socket.io
❯ npm install
```

8. Start the server with PM2 Enable required Apache modules:

```
❯ sudo a2enmod proxy proxy_http
```

   Create a new virtual host config:

```
❯ sudo nano /etc/apache2/sites-available/minecraft-in-cloud.conf
```

   Paste the following content:

```
<VirtualHost *:80>
  ServerName minecraftincloud.com
  ServerAlias www.minecraftincloud.com
  ProxyPreserveHost On
  ProxyPass / http://localhost:3000/
  ProxyPassReverse / http://localhost:3000/
</VirtualHost>
```

   Activate the config:

```
❯ sudo a2dissite 000-default.conf
❯ sudo a2ensite minecraft-in-cloud.conf
❯ sudo systemctl reload apache2
```

You can now visit http://< your-ip >. and the Website Should be running !

To create the minecraft-server you can just execute the script create-server.sh, we will talk more in details later.

---

3. Project Structure & Key Files

File / Directory	Description
`server.js`	Node.js backend server for user authentification for example, using Express and Socket.IO
`users.json`	Stores login credentials
`public/`	Frontend HTML/JS/CSS files
`scripts/`	Server management scripts
`backups/`	Directory for minecraft world bakups and archives

---

4. Script Documentation

`start.sh`

```
#!/bin/bash
cd ~/minecraft-server

echo "□ Starting Minecraft server..."
java -Xmx1024M -Xms1024M -jar server.jar nogui
```

This script launches the Minecraft server using the installed server.jar file in the ~/minecraft-server directory with standard memory settings. It is triggered from the web panel or manually via terminal.

`stop.sh`

```
#!/bin/bash
echo "□ Stopping Minecraft server..."

# Backup world before stopping
~/minecraft-panel/scripts/archive_world.sh

pkill -f "server.jar"
```

This script saves an archive of the minecraft world and then stops it.

`archive_world.sh`

```
#!/bin/bash

# CONFIG
WORLD_DIR="$HOME/minecraft-server/world"
BACKUP_DIR="$HOME/minecraft-server/backups"
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
ARCHIVE_NAME="world_backup_$TIMESTAMP.tar.gz"

# CREATE BACKUP
echo "□ Creating backup of Minecraft world..."
mkdir -p "$BACKUP_DIR"

# Tar and gzip the world directory
tar -czf "$BACKUP_DIR/$ARCHIVE_NAME" -C "$WORLD_DIR" .

if [ $? -eq 0 ]; then
  echo "□ Backup created: $BACKUP_DIR/$ARCHIVE_NAME"
else
  echo "□ Backup failed!"
  exit 1
fi
```

This script creates a compressed backup (tar.gz) of the entire world/ directory, storing it in the backups/ folder inside minecraft-server. It is usually triggered before stopping the server, either manually or automatically.

load_backup.sh

```
#!/bin/bash

ARCHIVE=$1
BACKUP_DIR=~/minecraft-server/backups
WORLD_DIR=~/minecraft-server/world

echo "□ Loading backup: $ARCHIVE"
if [ ! -f "$ARCHIVE" ]; then
  echo "□ Backup file does not exist."
  exit 1
fi

echo "□ Stopping Minecraft server..."
tmux kill-session -t minecraft-server 2>/dev/null

echo "□ Removing current world directory..."
rm -rf "$WORLD_DIR"

echo "□ Extracting backup..."
mkdir -p "$WORLD_DIR"
tar -xzf "$ARCHIVE" -C "$WORLD_DIR"

echo "□ Backup loaded successfully."
```

This script safely loads a selected backup file from the backups/ directory. It removes the current world/ directory, extracts the backup, and prepares the server for a fresh start with the chosen world. It is callable from the website panel using the load backup dropdown.

`create_server.sh`

```
#!/bin/bash

SERVER_DIR=~/minecraft-server
JAR_URL="https://piston-data.mojang.com/v1/objects/e6ec2f64e6080b9b5d9b471b291c33cc7f509733/server.jar"
JAR_FILE="$SERVER_DIR/server.jar"

echo "□ Creating server directory at $SERVER_DIR..."
mkdir -p "$SERVER_DIR"
cd "$SERVER_DIR" || exit 1

echo "□ Downloading Minecraft server jar..."
curl -o server.jar "$JAR_URL"

java -Xmx512M -Xms512M -jar server.jar nogui

echo "□ Accepting EULA..."
echo "eula=true" > eula.txt

java -Xmx512M -Xms512M -jar server.jar nogui &

echo "□ Minecraft server setup complete in $SERVER_DIR and Server is running !"
```

This script sets up a fresh Minecraft Java Edition server.It downloads the official JAR file, obtainable on Mojang's website, and scraped thanks to the web inspector. The scripts then accepts the EULA, and launches the server which creates the world and the files associated!

This script launches the Minecraft server using the installed server.jar file in the ~/minecraft-server directory with standard memory settings. It is triggered from the web panel or manually via terminal. This script launches the Minecraft server using the installed server.jar file in the ~/minecraft-server directory with standard memory settings. It is triggered from the web panel or manually via terminal.

This will:

- Remove the current project directory
- Extract the specified backup
- Restart the server via PM2

---

5. Linking the Server with a DNS Domain Name

To make the platform accessible via a custom domain name, we connected the public IP of the EC2 instance to a DNS entry and configured HTTPS encryption. Here's how:

DNS Configuration (NameCheap used here)

1. Log in to your NameCheap Account
2. Go to **Domain > Advanced DNS** for your domain (e.g., `minecraftincloud.com`).
3. Add an **A record** pointing your domain to the **public IPv4** of your EC2 instance:

   - **Name**: `@`
   - **Type**: `A`
   - **Points to**: `your-public-ec2-ip` -**TTL**: `300` or `Automatic`

HTTPS Configuration with Let's Encrypt

To secure the server with HTTPS:

1. Install **Certbot**:

```
❯ sudo apt install certbot python3-certbot-apache -y
```

2. Run Certbot to generate the SSL certificate and configure Apache:

```
❯ sudo certbot --apache
```

3. Follow the instructions to select your domain and enable automatic HTTPS redirection.

4. Once completed, Certbot:

   - Creates SSL certificates using **Let's Encrypt**
   - Updates the Apache virtual host config (`minecraft-in-cloud-le-ssl.conf`)

---

6. References

- https://expressjs.com/ (https://expressjs.com/)
- https://socket.io/ (https://socket.io/)
- https://pm2.keymetrics.io/ (https://pm2.keymetrics.io/)
- StackOverflow/GitHub discussions
- AWS EC2 user guide