Hosting a Minecraft Server via Cloudflare Tunnel

■ Complete Step-by-Step Guide (Template with Placeholders)

Objective: Make an existing Minecraft server (local or via Playit) publicly accessible through a secure Cloudflare Tunnel using a custom subdomain such as mc1.<yourdomain>.com.

Requirements:

1 A domain managed on Cloudflare (e.g., <yourdomain>.com)
2 A running Minecraft server (local or Dockerized)
3 A free Cloudflare Zero Trust account
4 Docker installed on your system

Step 1 – Create a Cloudflare Tunnel:

1. Log in to https://dash.cloudflare.com.
2. Select your domain (<yourdomain>.com).
3. Navigate to **Zero Trust** → **Tunnels**.
4. If no organization exists yet, create one (e.g., <orgname>).
5. Click “+ Create a Tunnel” and enter a name, e.g. MC1-Tunnel.
6. Copy the generated **token** (format: eyJhIjoi...etc...).

Step 2 – Start the Cloudflared container:

Run the following command in PowerShell or your terminal (single line, no line breaks):

```
docker run -d --name <container_name> --network <docker_network>
cloudflare/cloudflared:latest tunnel run --token <your_token>
```

Example:

```
docker run -d --name cloudflared_mc1 --network minecraft-net
cloudflare/cloudflared:latest tunnel run --token
eyJhIjoiYmVuLW1ha2VzLXR1bm5lbC0xMjMiLCJpIjoiODc2NzgifQ==
```

Step 3 – Configure a public hostname in Cloudflare:

1. Open your tunnel in the Cloudflare dashboard.
2. Go to **Public Hostnames** → **Add a public hostname**.
3. Fill in the following fields:

```
| Field | Value | |--------|--------| | Subdomain | mc1 | | Domain |
<yourdomain>.com | | Type | TCP | | URL / Service | <target_address>:<port>
(e.g. pass-beach.gl.joinmc.link:25565) |
```

■ Note: NoTLSVerify is automatically disabled for **TCP** connections; no manual option required.

Step 4 – Resolve DNS conflicts (if any):

If you encounter the error Error: An A, AAAA, or CNAME record with that host already exists:
1. Go to **DNS** → **Records** in Cloudflare.
2. Delete the existing record (e.g., mc1 CNAME).
3. Retry adding the public hostname.

Step 5 – Verify the connection:

Check if your tunnel is active using:

```
docker logs <container_name>
```

If the output includes Connection established. Tunnel ID: ... → the tunnel is running.

You can now connect to your Minecraft server via mc1.<yourdomain>.com.

Optional – Persistent Configuration (recommended):

To ensure the tunnel reconnects automatically on restart, create a config.yml file. Place it under /docker/cloudflared/config/config.yml and mount it as follows:

```
docker run -d --name cloudflared_mc1 --network minecraft-net -v
C:\Docker\cloudflared\config:/etc/cloudflared cloudflare/cloudflared:latest
tunnel run
```

The config.yml should contain:

```
tunnel: <tunnel-id> credentials-file: /etc/cloudflared/<tunnel-id>.json
ingress: - hostname: mc1.<yourdomain>.com service:
tcp://<target_address>:<port> - service: http_status:404
```

■ Template with placeholders for public documentation (e.g. GitHub Readme or tutorial post).