Next Steps & Lesson Plan: Discord Bot on RunPod

PHASE 1: COMPLETED FOUNDATION
- Refamiliarized with Git basics and GitHub workflow.
- Confirmed repository: discordbot1.0.
- Installed and configured Tailscale on Mac and iPhone.
- Verified private, encrypted connectivity (ping success, 0% packet loss).
- Brew environment operational; Docker installed and ready.

Result: A stable, secure base system for collaborative development and deployment.

PHASE 2: NEXT SESSION (DAY 2) GOALS

1. Connect RunPod to Tailscale
   - Join RunPod node to your existing tailnet.
   - Confirm it appears in `tailscale status` on both Mac and RunPod.
   - Test communication with ping and optional file transfer.
   - Outcome: RunPod securely linked for private operations.

2. Clone and prepare Discord Bot Repository
   - Command: git clone https://github.com/Brockmerkwan/discordbot1.0.git
   - Enter repo: cd discordbot1.0
   - Install dependencies: npm install
   - Validate bot scripts and environment variables.
   - Outcome: Ready-to-run bot code synchronized with GitHub.

3. Containerize the Discord Bot with Docker
   - Create Dockerfile defining environment and startup command.
   - Example base:
     ```
     FROM node:18-alpine
     WORKDIR /app
     COPY . .
     RUN npm install
     CMD ["npm", "start"]
     ```
   - Build image: docker build -t discordbot1.0 .
   - Run container: docker run -d --name discordbot -p 3000:3000 discordbot1.0
   - Outcome: Bot runs in isolated, repeatable container.

4. Connect Docker + Tailscale
   - Use Tailscale network to communicate between Mac and RunPod.
   - Verify internal access: curl http://100.x.x.x:3000
   - Optional: add container Tailscale integration for direct mesh networking.
   - Outcome: End-to-end control of container from Mac, securely tunneled.

PHASE 3: SHORT-TERM ROADMAP (WEEKS 1–3)

Week	Focus	Deliverables
1	Connect RunPod to tailnet, clone bot repo	Private network, bot code live
2	Containerization & automation	Dockerfile finalized, CI/CD basics
3	Deploy bot and monitor	Active bot hosted on RunPod with logs

PHASE 4: HARDENING AND MONITORING
- Implement LuLu or pf firewall rules for outbound traffic control on Mac.
- Harden SSH and disable password logins.
- Secure RunPod node with Tailscale SSH (no public exposure).
- Use Tailscale ACLs for node access policies.
- Verify fail2ban or equivalent log watcher if SSH is exposed internally.

PHASE 5: FUTURE LESSONS (OPTIONAL EXPANSION)
1. Integrate Cline agent for autonomous deployment updates.
2. Add GitHub Actions CI/CD workflow to auto-build Docker images.
3. Connect a database (PostgreSQL or SQLite) for bot data.
4. Implement logging pipeline to local machine via Tailscale VPN.
5. Write documentation and serve it securely through a Tailscale funnel or Caddy HTTPS proxy.

TEACHER’S SUMMARY
Today established the critical foundation: Git version control and Tailscale networking. The next lesson
The process is intentionally structured: one concept at a time, minimal dependencies, maximum repeatabil

Prepared by: ChatGPT (Teaching Director)
For: Brock Merkwan