APBeeper Bot Deployment Checklist

Use this checklist to ensure a smooth deployment to GitHub and Railway.

📋 Pre-Deployment Checklist

GitHub Repository Setup

- [ ] Repository created on GitHub
- [ ] Local repository initialized with `git init`
- [ ] Remote origin added: `git remote add origin https://github.com/yourusername/apbeeper_bot.git`
- [ ] All files committed and pushed to GitHub
- [ ] Repository is public or accessible to Railway

Environment Variables Prepared

- [ ] Discord bot token obtained from Discord Developer Portal
- [ ] Discord client ID copied from application settings
- [ ] Twitch API credentials obtained from Twitch Developers
- [ ] All sensitive data removed from code
- [ ] `.env.example` file updated with all required variables

Code Quality Check

- [ ] All dependencies installed: `npm install`
- [ ] Environment validation passes: `npm run test-env`
- [ ] Bot starts locally without errors: `npm run dev`
- [ ] All commands work in development Discord server
- [ ] No console errors or warnings

🚀 GitHub Deployment Steps

1. Initialize Git Repository

```
cd ~/apbeeper_bot
git init
git add .
git commit -m "Initial commit: APBeeper Discord Bot"
```

2. Connect to GitHub

```
git remote add origin https://github.com/yourusername/apbeeper_bot.git
git branch -M main
git push -u origin main
```

3. Verify GitHub Repository

- [ ] All files visible on GitHub

- [ ] README.md displays correctly
- [ ] No sensitive data (tokens, passwords) in repository
- [ ] `.gitignore` working properly (no `node_modules`, `.env`, etc.)

🚂 Railway Deployment Steps

1. Create Railway Project

- [ ] Go to railway.app (https://railway.app)
- [ ] Sign in with GitHub account
- [ ] Click “New Project” → “Deploy from GitHub repo”
- [ ] Select your APBeeper Bot repository
- [ ] Choose `main` branch

2. Add PostgreSQL Database

- [ ] In Railway dashboard, click “New Service”
- [ ] Select “Database” → “PostgreSQL”
- [ ] Wait for database provisioning
- [ ] Note that `DATABASE_URL` is automatically set

3. Configure Environment Variables

Add these variables in Railway dashboard under “Variables”:

Required Variables:
- [ ] `DISCORD_TOKEN` = your_discord_bot_token
- [ ] `DISCORD_CLIENT_ID` = your_discord_client_id
- [ ] `TWITCH_CLIENT_ID` = your_twitch_client_id
- [ ] `TWITCH_CLIENT_SECRET` = your_twitch_client_secret
- [ ] `NODE_ENV` = production

Optional Variables:
- [ ] `LOG_LEVEL` = info
- [ ] `PORT` = 3000
- [ ] `HEALTH_CHECK_ENABLED` = true

4. Deploy and Monitor

- [ ] Railway automatically deploys after configuration
- [ ] Check “Deployments” tab for build logs
- [ ] Verify no build errors
- [ ] Check “Logs” for runtime logs
- [ ] Confirm bot appears online in Discord

5. Test Deployment

- [ ] Bot responds to slash commands
- [ ] Health check endpoint works: `https://your-app.railway.app/health`
- [ ] Database operations work (commands save settings)
- [ ] Scheduled tasks running (check logs)
- [ ] Twitch integration working (if configured)

🔧 Post-Deployment Configuration

Discord Bot Setup

- [ ] Bot added to Discord servers
- [ ] Proper permissions granted:
- Send Messages
- Use Slash Commands
- Embed Links
- Read Message History
- Manage Messages (if needed)
- [ ] Slash commands registered and working

Database Migration (if upgrading from local)

- [ ] Run migration script: `npm run migrate`
- [ ] Verify data transferred correctly
- [ ] Test all bot functionality
- [ ] Remove old SQLite database files

Monitoring Setup

- [ ] Railway health checks working
- [ ] Log levels appropriate for production
- [ ] Error notifications configured (optional)
- [ ] Performance monitoring enabled

🆘 Troubleshooting

Common Issues and Solutions

Bot Not Starting:
- [ ] Check environment variables are set correctly
- [ ] Verify Discord token is valid
- [ ] Check Railway deployment logs for errors

Database Connection Issues:
- [ ] Verify PostgreSQL service is running
- [ ] Check `DATABASE_URL` is set automatically by Railway
- [ ] Review database connection logs

Commands Not Working:
- [ ] Verify bot has proper Discord permissions
- [ ] Check if slash commands are registered
- [ ] Review command execution logs

Health Check Failures:
- [ ] Verify health server is starting
- [ ] Check if port 3000 is available
- [ ] Review health endpoint logs

Getting Help

- [ ] Check troubleshooting guide (docs/troubleshooting.md)
- [ ] Review Railway deployment logs
- [ ] Create GitHub issue with detailed error information
- [ ] Check Railway status page for service issues

✅ Deployment Success Criteria

Your deployment is successful when:
- [ ] Bot appears online in Discord
- [ ] All slash commands respond correctly
- [ ] Health check returns 200 OK
- [ ] Database operations work (settings persist)
- [ ] Scheduled tasks run without errors
- [ ] No critical errors in logs
- [ ] Railway dashboard shows healthy status

📊 Performance Monitoring

Key Metrics to Monitor

- [ ] Response time for commands
- [ ] Memory usage (Railway dashboard)
- [ ] CPU usage (Railway dashboard)
- [ ] Database connection count
- [ ] Error rate in logs

Optimization Tips

- [ ] Monitor Railway usage to stay within free tier
- [ ] Optimize database queries if needed
- [ ] Adjust cron job frequency if necessary
- [ ] Implement caching for frequently accessed data

Congratulations! 🎉 Your APBeeper Discord Bot is now deployed and ready to serve the APB community!

For ongoing maintenance and updates, refer to the Railway Deployment Guide (docs/railway-deployment.md) and Troubleshooting Guide (docs/troubleshooting.md).