Railway Deployment Guide

This guide walks you through deploying APBeeper Bot to Railway, a modern deployment platform that makes it easy to deploy and scale applications.

🚀 Quick Deploy

The fastest way to deploy APBeeper Bot is using the Railway template:

[

📋 Prerequisites

Before deploying, ensure you have:

1. **GitHub Account** with the APBeeper Bot repository
2. **Railway Account** (free tier available)
3. **Discord Bot Token** and Client ID
4. **Twitch API Credentials** (Client ID and Secret)

🛠️ Step-by-Step Deployment

Step 1: Prepare Your Repository

1. **Fork the repository** to your GitHub account
2. **Clone your fork** locally (optional, for testing)
3. **Ensure all files are committed** and pushed to GitHub

Step 2: Create Railway Project

1. **Visit Railway**: Go to railway.app (https://railway.app)
2. **Sign in** with your GitHub account
3. **Create New Project**: Click “New Project”
4. **Deploy from GitHub**: Select “Deploy from GitHub repo”
5. **Select Repository**: Choose your forked APBeeper Bot repository
6. **Configure Branch**: Select `main` or your preferred branch

Step 3: Add Database Service

1. **Add PostgreSQL**: In your Railway project dashboard
   - Click “New Service”
   - Select “Database” → “PostgreSQL”
   - Railway will automatically provision a PostgreSQL database
2. **Note Database URL**: Railway automatically sets `DATABASE_URL` environment variable

Step 4: Configure Environment Variables

In your Railway project dashboard, go to “Variables” and add:

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
ENABLE_TWITCH_NOTIFICATIONS=true
ENABLE_POPULATION_TRACKING=true
```

Step 5: Deploy

1. **Automatic Deployment**: Railway will automatically deploy after configuration
2. **Monitor Logs**: Check the “Deployments” tab for build and runtime logs
3. **Verify Health**: The bot should start and connect to Discord

Step 6: Configure Custom Domain (Optional)

1. **Generate Domain**: Railway provides a free `.railway.app` domain
2. **Custom Domain**: Add your own domain in the “Settings” → “Domains” section
3. **SSL Certificate**: Railway automatically provides SSL certificates

🔧 Configuration Details

Database Migration

The bot automatically handles database migration from SQLite to PostgreSQL:

1. **Automatic Detection**: Bot detects `DATABASE_URL` environment variable
2. **Schema Creation**: Creates necessary tables on first run
3. **Data Migration**: Manual migration required for existing data (see migration script)

Environment Variables Reference

Variable	Description	Required	Default
`DISCORD_TOKEN`	Discord bot token	✅	-
`DISCORD_CLIENT_ID`	Discord application client ID	✅	-
`TWITCH_CLIENT_ID`	Twitch API client ID	✅	-
`TWITCH_CLIENT_SECRET`	Twitch API client secret	✅	-
`NODE_ENV`	Environment mode	✅	`production`
`DATABASE_URL`	PostgreSQL connection string	⚠️	Auto-set by Railway
`PORT`	Health check server port	❌	`3000`
`LOG_LEVEL`	Logging verbosity	❌	`info`

Health Checks

Railway automatically monitors your application using the health check endpoint:

- **Endpoint**: `/health`
- **Method**: `GET`
- **Expected Response**: `200 OK`
- **Timeout**: 100 seconds (configurable in `railway.json`)

📊 Monitoring & Maintenance

Viewing Logs

1. **Real-time Logs**: Railway dashboard → “Deployments” → “View Logs”
2. **Log Levels**: Configure via `LOG_LEVEL` environment variable
3. **Log Retention**: Railway retains logs for 7 days on free tier

Performance Monitoring

1. **Metrics**: Railway provides CPU, memory, and network metrics
2. **Alerts**: Set up alerts for high resource usage
3. **Scaling**: Railway automatically scales based on demand

Database Management

1. **Database Console**: Access PostgreSQL via Railway dashboard
2. **Backups**: Railway automatically backs up databases
3. **Connection Limits**: Monitor connection usage

🔄 Updates & Maintenance

Automatic Deployments

Railway automatically deploys when you push to your connected branch:

1. **Push Changes**: Commit and push to GitHub
2. **Automatic Build**: Railway detects changes and rebuilds
3. **Zero Downtime**: Railway performs rolling deployments

Manual Deployments

1. **Redeploy**: Click “Deploy” in Railway dashboard
2. **Rollback**: Select previous deployment to rollback
3. **Environment Changes**: Restart required for environment variable changes

Database Migrations

For schema changes:

1. **Create Migration Script**: Add to `scripts/migrations/`
2. **Run Migration**: Execute via Railway console or deployment script
3. **Verify Changes**: Check database schema and data integrity

🆘 Troubleshooting

Common Issues

Bot Not Starting

```
# Check logs for errors
# Verify environment variables are set
# Ensure Discord token is valid
```

Database Connection Issues

```
# Verify DATABASE_URL is set
# Check PostgreSQL service status
# Review connection limits
```

Health Check Failures

```
# Verify health endpoint is accessible
# Check if bot is binding to correct port
# Review health check timeout settings
```

Debug Mode

Enable debug logging:

```
LOG_LEVEL=debug
```

Support Resources

1. **Railway Documentation**: docs.railway.app (https://docs.railway.app)
2. **Railway Discord**: Community support server
3. **GitHub Issues**: Report bot-specific issues
4. **Railway Status**: status.railway.app (https://status.railway.app)

💰 Cost Optimization

Free Tier Limits

Railway free tier includes:
- 500 hours of usage per month
- $5 credit per month
- Automatic sleep after 1 hour of inactivity

Optimization Tips

1. **Resource Monitoring**: Monitor CPU and memory usage
2. **Efficient Queries**: Optimize database queries
3. **Caching**: Implement caching for frequently accessed data
4. **Sleep Prevention**: Use health checks to prevent sleeping

🔒 Security Best Practices

Environment Variables

1. **Never Commit Secrets**: Use Railway's environment variable system
2. **Rotate Keys**: Regularly rotate API keys and tokens
3. **Least Privilege**: Use minimal required permissions

Database Security

1. **Connection Encryption**: Railway enforces SSL connections
2. **Access Control**: Limit database access to application only
3. **Regular Backups**: Verify backup integrity regularly

Need help? Check the troubleshooting guide (troubleshooting.md) or create an issue on GitHub.