Hosting Documentation for APBeeper Bot

This comprehensive guide covers deploying, maintaining, and scaling your APBeeper Discord bot in a production environment.

Table of Contents

- Pre-Deployment Checklist
- Step-by-Step Deployment
- Database Management
- Monitoring and Logging
- Backup Procedures
- Performance Optimization
- Scaling Considerations
- Troubleshooting
- Maintenance Tasks

Pre-Deployment Checklist

Server Requirements

- [ ] VPS with Ubuntu 20.04+ installed
- [ ] Node.js 18+ installed
- [ ] PM2 process manager installed
- [ ] Database configured (SQLite/PostgreSQL/MySQL)
- [ ] Firewall configured (UFW)
- [ ] SSL certificate (if using HTTPS)
- [ ] Domain name configured (optional)

Bot Configuration

- [ ] Discord bot token obtained
- [ ] Bot permissions configured
- [ ] Environment variables set
- [ ] Database schema created
- [ ] Log directories created
- [ ] PM2 ecosystem file configured

Security Setup

- [ ] SSH key authentication enabled
- [ ] Root login disabled
- [ ] Fail2Ban configured
- [ ] Automatic security updates enabled
- [ ] Firewall rules applied

GitHub Integration

- [ ] Repository created and configured
- [ ] GitHub Actions workflow set up
- [ ] SSH keys for deployment configured
- [ ] Secrets added to GitHub repository

Step-by-Step Deployment

1. Clone Repository on VPS

```
# Connect to your VPS
ssh apbeeper@your_vps_ip

# Clone the repository
git clone https://github.com/yourusername/apbeeper-bot.git
cd apbeeper-bot

# Or if using SSH
git clone git@github.com:yourusername/apbeeper-bot.git
cd apbeeper-bot
```

2. Install Dependencies

```
# Install production dependencies
npm ci --production

# Or install all dependencies for development
npm install
```

3. Configure Environment

```
# Copy environment template
cp .env.example .env

# Edit environment variables
nano .env
```

Required environment variables:

```
# Bot Configuration
DISCORD_TOKEN=your_discord_bot_token
CLIENT_ID=your_bot_client_id
GUILD_ID=your_test_guild_id

# Database Configuration
DATABASE_URL=sqlite:./data/apbeeper.db
# Or for PostgreSQL:
# DATABASE_URL=postgresql://username:password@localhost:5432/apbeeper_bot

# Application Settings
NODE_ENV=production
LOG_LEVEL=info
PORT=3000

# Optional: Webhook Configuration
WEBHOOK_URL=https://yourdomain.com/webhook
WEBHOOK_SECRET=your_webhook_secret
```

4. Set Up Database

```
# Create data directory
mkdir -p data logs

# Run database migrations (if applicable)
npm run db:migrate

# Or run the migration script
./scripts/migrate_db.sh
```

5. Configure PM2

```
# Verify ecosystem configuration
cat ecosystem.config.js

# Start the bot with PM2
pm2 start ecosystem.config.js

# Save PM2 configuration
pm2 save

# Set up PM2 to start on boot
pm2 startup
# Follow the instructions provided
```

6. Verify Deployment

```
# Check PM2 status
pm2 status

# View logs
pm2 logs apbeeper-bot

# Monitor in real-time
pm2 monit
```

7. Test Bot Functionality

1. Invite bot to a test Discord server
2. Test basic commands
3. Verify database operations
4. Check log output for errors

Database Management

SQLite Management

```
# Backup SQLite database
cp data/apbeeper.db data/apbeeper_backup_$(date +%Y%m%d_%H%M%S).db

# View database size
ls -lh data/apbeeper.db

# Access SQLite CLI
sqlite3 data/apbeeper.db
```

PostgreSQL Management

```
# Create backup
pg_dump -U apbeeper_user apbeeper_bot > backups/db_backup_$(date +%Y%m%d_%H%M%S).sql

# Restore from backup
psql -U apbeeper_user apbeeper_bot < backups/db_backup_20250623_120000.sql

# Check database size
psql -U apbeeper_user -d apbeeper_bot -c "SELECT
pg_size_pretty(pg_database_size('apbeeper_bot'));"
```

Database Migration Script

The migration script is located at `scripts/migrate_db.sh`:

```
#!/bin/bash
set -e

echo "Starting database migration..."

# Check if database exists
if [ ! -f "data/apbeeper.db" ]; then
    echo "Creating new database..."
    mkdir -p data
fi

# Run migrations (if using a migration system)
if command -v npm &> /dev/null; then
    if npm run db:migrate 2>/dev/null; then
        echo "Database migrations completed successfully"
    else
        echo "No migration script found, skipping..."
    fi
fi

echo "Database migration completed"
```

Monitoring and Logging

1. Log Configuration

APBeeper uses structured logging with different levels:

```
// Example logging configuration
const winston = require('winston');

const logger = winston.createLogger({
  level: process.env.LOG_LEVEL || 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  transports: [
    new winston.transports.File({ filename: 'logs/error.log', level: 'error' }),
    new winston.transports.File({ filename: 'logs/combined.log' }),
    new winston.transports.Console({
      format: winston.format.simple()
    })
  ]
});
```

2. Log Monitoring Commands

```
# View real-time logs
tail -f logs/combined.log

# View error logs only
tail -f logs/error.log

# View PM2 logs
pm2 logs apbeeper-bot

# Search logs for specific patterns
grep "ERROR" logs/combined.log
grep "guild" logs/combined.log | tail -20
```

3. Log Rotation Setup

Create log rotation configuration:

```
sudo nano /etc/logrotate.d/apbeeper-bot
```

```
/home/apbeeper/apbeeper-bot/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 apbeeper apbeeper
    postrotate
        pm2 reloadLogs
    endscript
}
```

4. System Monitoring

```
# Monitor system resources
htop

# Check disk usage
df -h
du -sh /home/apbeeper/apbeeper-bot/

# Monitor network connections
netstat -tulpn | grep :3000

# Check memory usage
free -h
```

5. Bot-Specific Monitoring

```
# Check bot status across Discord servers
pm2 logs apbeeper-bot | grep "Ready"

# Monitor command usage
grep "Command executed" logs/combined.log | tail -20

# Check error rates
grep "ERROR" logs/combined.log | wc -l
```

Backup Procedures

1. Automated Backup Script

The backup script is located at `scripts/backup_db.sh`:

```
#!/bin/bash
set -e

BACKUP_DIR="/home/apbeeper/backups"
DATE=$(date +%Y%m%d_%H%M%S)
BOT_DIR="/home/apbeeper/apbeeper-bot"

# Create backup directory
mkdir -p $BACKUP_DIR

echo "Starting backup process..."

# Backup database
if [ -f "$BOT_DIR/data/apbeeper.db" ]; then
    echo "Backing up SQLite database..."
    cp "$BOT_DIR/data/apbeeper.db" "$BACKUP_DIR/apbeeper_db_$DATE.db"
fi

# Backup configuration files
echo "Backing up configuration files..."
cp "$BOT_DIR/.env" "$BACKUP_DIR/env_$DATE.backup" 2>/dev/null || echo "No .env file
found"
cp "$BOT_DIR/ecosystem.config.js" "$BACKUP_DIR/ecosystem_$DATE.js"

# Backup logs (last 7 days)
echo "Backing up recent logs..."
find "$BOT_DIR/logs" -name "*.log" -mtime -7 -exec cp {} "$BACKUP_DIR/" \;

# Compress old backups (older than 7 days)
find $BACKUP_DIR -name "*.db" -mtime +7 -exec gzip {} \;

# Remove very old backups (older than 30 days)
find $BACKUP_DIR -name "*.gz" -mtime +30 -delete

echo "Backup completed successfully!"
echo "Backup location: $BACKUP_DIR"
```

2. Set Up Automated Backups

```
# Make backup script executable
chmod +x scripts/backup_db.sh

# Add to crontab for daily backups at 2 AM
crontab -e

# Add this line:
0 2 * * * /home/apbeeper/apbeeper-bot/scripts/backup_db.sh >> /home/apbeeper/logs/
backup.log 2>&1
```

3. Manual Backup Commands

```
# Create immediate backup
./scripts/backup_db.sh

# Backup specific components
cp data/apbeeper.db backups/manual_backup_$(date +%Y%m%d_%H%M%S).db
cp .env backups/env_backup_$(date +%Y%m%d_%H%M%S)

# Create full system backup
tar -czf backups/full_backup_$(date +%Y%m%d_%H%M%S).tar.gz \
    --exclude=node_modules \
    --exclude=logs \
    --exclude=backups \
    .
```

4. Restore Procedures

```
# Restore database from backup
cp backups/apbeeper_db_20250623_120000.db data/apbeeper.db

# Restore configuration
cp backups/env_20250623_120000.backup .env

# Restart bot after restore
pm2 restart apbeeper-bot
```

Performance Optimization

1. Node.js Optimization

```
// In your main application file
process.env.UV_THREADPOOL_SIZE = 128; // Increase thread pool size
process.env.NODE_OPTIONS = '--max-old-space-size=4096'; // Increase memory limit
```

2. PM2 Optimization

Update `ecosystem.config.js`:

```
module.exports = {
  apps: [{
    name: 'apbeeper-bot',
    script: 'src/index.js',
    instances: 1, // Use 'max' for clustering
    exec_mode: 'fork', // Use 'cluster' for load balancing
    autorestart: true,
    watch: false,
    max_memory_restart: '2G',
    node_args: '--max-old-space-size=4096',
    env: {
      NODE_ENV: 'production',
      UV_THREADPOOL_SIZE: 128
    },
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log',
    time: true,
    merge_logs: true
  }]
};
```

3. Database Optimization

```
-- For SQLite, enable WAL mode for better performance
PRAGMA journal_mode=WAL;
PRAGMA synchronous=NORMAL;
PRAGMA cache_size=10000;
PRAGMA temp_store=memory;
```

4. System-Level Optimization

```
# Increase file descriptor limits
echo "apbeeper soft nofile 65536" | sudo tee -a /etc/security/limits.conf
echo "apbeeper hard nofile 65536" | sudo tee -a /etc/security/limits.conf

# Optimize network settings
echo "net.core.somaxconn = 65536" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.tcp_max_syn_backlog = 65536" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

Scaling Considerations

1. Horizontal Scaling with PM2 Cluster Mode

```
// ecosystem.config.js for clustering
module.exports = {
  apps: [{
    name: 'apbeeper-bot',
    script: 'src/index.js',
    instances: 'max', // Use all CPU cores
    exec_mode: 'cluster',
    // ... other configuration
  }]
};
```

2. Database Scaling

For high-traffic bots, consider:

- **Read Replicas**: Set up PostgreSQL read replicas
- **Connection Pooling**: Use connection pooling libraries
- **Caching**: Implement Redis for frequently accessed data

```
// Example Redis caching
const redis = require('redis');
const client = redis.createClient();

// Cache guild settings
async function getGuildSettings(guildId) {
  const cached = await client.get(`guild:${guildId}`);
  if (cached) return JSON.parse(cached);

  const settings = await database.getGuildSettings(guildId);
  await client.setex(`guild:${guildId}`, 300, JSON.stringify(settings));
  return settings;
}
```

3. Load Balancing

For multiple server instances:

```
# Nginx configuration
upstream apbeeper_backend {
    server 127.0.0.1:3000;
    server 127.0.0.1:3001;
    server 127.0.0.1:3002;
}

server {
    listen 80;
    server_name yourdomain.com;

    location / {
        proxy_pass http://apbeeper_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

4. Multi-Server Architecture

For very large deployments:
- **Sharding**: Distribute Discord servers across multiple bot instances
- **Message Queues**: Use Redis or RabbitMQ for inter-service communication
- **Microservices**: Split functionality into separate services

Troubleshooting

Common Issues and Solutions

1. Bot Not Starting

```
# Check PM2 status
pm2 status

# View detailed logs
pm2 logs apbeeper-bot --lines 50

# Check for common issues:
# - Missing environment variables
# - Database connection problems
# - Port conflicts
# - Permission issues
```

2. High Memory Usage

```
# Monitor memory usage
pm2 monit

# Check for memory leaks
node --inspect src/index.js

# Restart if memory usage is too high
pm2 restart apbeeper-bot
```

3. Database Connection Issues

```
# Test database connection
sqlite3 data/apbeeper.db ".tables"

# Check database permissions
ls -la data/

# Verify database URL in .env
grep DATABASE_URL .env
```

4. Discord API Rate Limiting

```
# Check logs for rate limit messages
grep -i "rate limit" logs/combined.log

# Monitor API usage
grep "API request" logs/combined.log | tail -20
```

5. Performance Issues

```
# Check system resources
htop
iotop

# Monitor bot performance
pm2 monit

# Check for slow database queries
grep "slow query" logs/combined.log
```

Diagnostic Commands

```
# System health check
./scripts/health_check.sh

# Bot status check
pm2 show apbeeper-bot

# Network connectivity
ping discord.com
curl -I https://discord.com/api/v10/gateway

# Disk space check
df -h
du -sh logs/ data/ node_modules/
```

Maintenance Tasks

Daily Tasks

- [ ] Check PM2 status: `pm2 status`
- [ ] Review error logs: `tail -50 logs/error.log`

- [ ] Monitor system resources: `htop`
- [ ] Verify bot responsiveness in Discord

Weekly Tasks

- [ ] Update dependencies: `npm update`
- [ ] Review and rotate logs
- [ ] Check backup integrity
- [ ] Monitor database size and performance
- [ ] Review security logs: `sudo tail /var/log/auth.log`

Monthly Tasks

- [ ] Update system packages: `sudo apt update && sudo apt upgrade`
- [ ] Review and update firewall rules
- [ ] Audit npm dependencies: `npm audit`
- [ ] Performance optimization review
- [ ] Backup verification and cleanup

Quarterly Tasks

- [ ] Security audit and penetration testing
- [ ] Disaster recovery testing
- [ ] Performance benchmarking
- [ ] Documentation updates
- [ ] SSL certificate renewal (if not automated)

Maintenance Scripts

Create a maintenance script at `scripts/maintenance.sh`:

```
#!/bin/bash
echo "Starting maintenance tasks..."

# Update dependencies
npm update

# Clean up old logs
find logs/ -name "*.log" -mtime +30 -delete

# Clean up old backups
find backups/ -name "*.gz" -mtime +90 -delete

# Restart PM2 processes
pm2 restart all

# System cleanup
sudo apt autoremove -y
sudo apt autoclean

echo "Maintenance completed!"
```

Emergency Procedures

1. Bot Crash Recovery

```
# Quick restart
pm2 restart apbeeper-bot

# If restart fails, check logs and fix issues
pm2 logs apbeeper-bot
pm2 delete apbeeper-bot
pm2 start ecosystem.config.js
```

2. Database Corruption Recovery

```
# Stop the bot
pm2 stop apbeeper-bot

# Restore from latest backup
cp backups/apbeeper_db_latest.db data/apbeeper.db

# Restart the bot
pm2 start apbeeper-bot
```

3. Server Migration

```
# On old server - create full backup
tar -czf apbeeper_migration.tar.gz apbeeper-bot/

# Transfer to new server
scp apbeeper_migration.tar.gz user@new-server:/home/apbeeper/

# On new server - extract and setup
tar -xzf apbeeper_migration.tar.gz
cd apbeeper-bot
npm ci --production
pm2 start ecosystem.config.js
```

Next Steps

After successful deployment:

1. Set up the Bot Distribution (./bot_distribution.md) system
2. Configure monitoring dashboards
3. Implement additional security measures
4. Plan for scaling based on usage patterns

Remember: Always test changes in a staging environment before applying to production!