VPS Node.js Deployment Cheat Sheet

Complete guide for deploying a Node.js application on Hostinger VPS

🚀 Initial VPS Setup

Connect to VPS

bash

```
ssh root@YOUR_VPS_IP
```

Update System

bash

```
apt update && apt upgrade -y
reboot  # If kernel updates are installed
```

Install Essential Packages

bash

```
apt install -y curl wget git nano ufw build-essential
```

📦 Node.js Installation

Install Node.js 20.x LTS

bash

```
# Add NodeSource repository
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -

# Install Node.js
apt-get install -y nodejs

# Verify installation
node --version
npm --version
```

👨‍💼 User Management

Create Non-Root User

bash

```
# Create user
adduser username

# Add to sudo group
usermod -aG sudo username
```

📂 Project Setup

Create Project Directory

bash

```
mkdir -p /var/www/your-project
cd /var/www/your-project
chown -R username:username /var/www/your-project
```

Clone from GitHub

bash

```
# Configure git
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"

# Clone repository
git clone https://github.com/username/repo-name.git /var/www/your-project
```

Install Dependencies

bash

```
cd /var/www/your-project
npm install
```

🔥 Firewall Configuration

Set Up UFW Firewall

```bash
ufw allow ssh
ufw allow 80/tcp      # HTTP
ufw allow 443/tcp     # HTTPS
ufw allow 3000/tcp    # Node.js app port
ufw --force enable
ufw status
```

🎨 Build Process (Webpack/SCSS)

Run Build

```bash
# Compile SCSS and bundle assets
npm run build

# Fix any permission issues
chown -R username:username /var/www/your-project
```

Typical Build Output

- `public/assets/css/style.css` (compiled from SCSS)
- `public/assets/js/*.js` (bundled JavaScript)

🌐 Domain Configuration

DNS Records (in Domain Registrar)

```
Type: A     Name: @     Value: YOUR_VPS_IP
Type: CNAME Name: www   Value: yourdomain.com
```

Remove Conflicting Records

- Delete any old A records pointing to different IPs
- Remove problematic AAAA (IPv6) records if causing SSL issues

🔧 Nginx Setup

Install Nginx

bash

```
apt install nginx -y
systemctl start nginx
systemctl enable nginx
systemctl status nginx
```

Create Site Configuration

bash

```
nano /etc/nginx/sites-available/yourdomain.com
```

Basic HTTP Configuration:

nginx

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Enable Site

bash

```
# Enable site
ln -s /etc/nginx/sites-available/yourdomain.com /etc/nginx/sites-enabled/

# Test configuration
nginx -t

# Restart nginx
systemctl restart nginx
```

🔒 SSL Setup (Let's Encrypt)

Install Certbot

bash

```
apt install certbot python3-certbot-nginx -y
```

Get SSL Certificate

bash

```
certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

Note: Ensure DNS fully propagates (no conflicting IPv6 records) before running certbot.

SSL Configuration (Auto-generated by Certbot)

nginx

```nginx
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name yourdomain.com www.yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;

    location / {
        proxy_pass http://localhost:3000;
        # ... proxy headers same as above
    }
}
```

🔄 PM2 Production Setup

Install PM2

bash

```bash
npm install -g pm2
```

Start Application

bash

```
cd /var/www/your-project

# Start with PM2
pm2 start server.js --name "your-app-name"

# Check status
pm2 status

# Save configuration
pm2 save

# Setup auto-start on boot
pm2 startup
# Copy and run the command it provides
```

PM2 Management Commands

bash

```
pm2 status              # Check app status
pm2 logs your-app-name      # View logs
pm2 restart your-app-name   # Restart app
pm2 stop your-app-name      # Stop app
pm2 delete your-app-name    # Remove app
pm2 monit               # Real-time monitoring
```

🔄 Development Workflow

Local Development

bash

```
# Edit code locally
# Test changes
git add .
git commit -m "Your changes"
git push origin main
```

Deploy to VPS

bash

```
# On VPS
cd /var/www/your-project
git pull origin main
npm install          # If new dependencies
npm run build        # If SCSS/JS changes
pm2 restart your-app-name
```

🐛 Troubleshooting

Check Application Logs

bash

```
pm2 logs your-app-name
tail -f /var/log/nginx/error.log
```

Test Services

bash

```
# Test nginx
nginx -t
systemctl status nginx

# Test Node.js app
curl http://localhost:3000

# Test domain resolution
nslookup yourdomain.com
```

Common Issues

- **404 on static files:** Check build process and file paths
- **SSL fails:** Verify DNS propagation, remove IPv6 conflicts
- **Upload issues:** Check file permissions and multer configuration
- **CSS not loading:** Ensure webpack build completed successfully

📊 File Structure Example

```
/var/www/your-project/
├── server.js
├── package.json
├── views/
│   ├── partials/
│   │   ├── header.ejs
│   │   └── footer.ejs
│   ├── index.ejs
│   └── resources.ejs
├── public/
│   └── assets/
│       ├── css/style.css     (compiled)
│       ├── js/               (bundled)
│       └── images/
├── src/
│   ├── scss/styles.scss      (source)
│   └── js/
├── uploads/
└── node_modules/
```

🎯 Final URLs

- **HTTP:** `http://yourdomain.com`
- **HTTPS:** `https://yourdomain.com` (after SSL setup)
- **No port numbers needed** (nginx handles proxy)

🔐 Security Notes

- Use non-root user for application
- Keep firewall enabled and minimal
- Regular system updates: `apt update && apt upgrade`
- Monitor PM2 logs for suspicious activity
- SSL certificates auto-renew via cron job

This cheat sheet covers the complete deployment process from fresh VPS to production-ready Node.js application with SSL.