MERN Stack Deployment Guide

DigitalOcean VPS Configuration for clientoperation.2ndsource.xyz

Date Created: May 15, 2025

Table of Contents

Overview

This document provides a comprehensive guide for setting up a MERN (MongoDB, Express, React, Node.js) stack application on a DigitalOcean VPS with Apache as the web server. The application is hosted under the subdomain `clientoperation.2ndsource.xyz`.

System Configuration:

- **Main Domain:** 2ndsource.xyz
- **Subdomain:** clientoperation.2ndsource.xyz
- **Frontend Port:** 3000 (React)
- **Backend Port:** 5000 (Node.js)
- **Web Server:** Apache with Reverse Proxy
- **Database:** MongoDB

DNS Configuration

Ensure the subdomain points to your VPS IP address by creating an A record:

Type	Name	Value	TTL
A	clientoperation	[YOUR_VPS_IP_ADDRESS]	3600

Apache Virtual Host Setup

Enable Required Apache Modules

```bash
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod ssl
```

Create Virtual Host Configuration

Create a new configuration file:

```bash
sudo nano /etc/apache2/sites-available/clientoperation.2ndsource.xyz.conf
```

Add the following content:

```apache
<VirtualHost *:80>
    ServerName clientoperation.2ndsource.xyz
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/clientoperation.2ndsource.xyz/html

    # Log files
    ErrorLog ${APACHE_LOG_DIR}/clientoperation.2ndsource.xyz-error.log
    CustomLog ${APACHE_LOG_DIR}/clientoperation.2ndsource.xyz-access.log combined

    # Proxy for React frontend (running on port 3000)
    ProxyPass / http://localhost:3000/
    ProxyPassReverse / http://localhost:3000/

    # If you want to serve API requests to your backend
    <Location /api>
        ProxyPass http://localhost:5000/api
        ProxyPassReverse http://localhost:5000/api
    </Location>
</VirtualHost>
```

Create Document Root Directory

```bash
sudo mkdir -p /var/www/clientoperation.2ndsource.xyz/html
sudo chown -R $USER:$USER /var/www/clientoperation.2ndsource.xyz/html
```

Enable the Site

```bash
sudo a2ensite clientoperation.2ndsource.xyz.conf
sudo systemctl restart apache2
```

SSL Configuration

Install Certbot and Obtain SSL Certificate

```bash
sudo apt update
sudo apt install certbot python3-certbot-apache
sudo certbot --apache -d clientoperation.2ndsource.xyz
```

This will automatically update your Apache configuration to handle SSL.

SSL Auto-renewal

Certbot installs a timer and service to automatically renew certificates before they expire. You can check the status with:

```bash
sudo systemctl status certbot.timer
```

MERN Stack Deployment

Backend (Node.js) Deployment

1. Create directory for the backend application:

```bash
mkdir -p ~/apps/clientoperation/backend
```

2. Deploy your Node.js code to this directory.

3. Install dependencies:

```bash
cd ~/apps/clientoperation/backend
npm install
```

4. Set up PM2 for process management:

```bash
# Install PM2 if not already installed
npm install -g pm2

# Start your backend with PM2
pm2 start server.js --name "clientoperation-backend"
pm2 save
```

5. Configure your backend to listen on port 5000:

```javascript
// In your Node.js application
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Frontend (React) Deployment

1. Create directory for the frontend application:

```bash
mkdir -p ~/apps/clientoperation/frontend
```

2. Deploy your React code to this directory.

3. Install dependencies and build for production:

```bash
cd ~/apps/clientoperation/frontend
npm install
npm run build
```

4. Serve the built application with PM2:

```bash
npm install -g serve
pm2 start serve --name "clientoperation-frontend" -- -s build -l 3000
pm2 save
```

Configure PM2 to Start on Boot

```bash
pm2 startup
```

Follow the instructions it provides to set up the startup script.

MongoDB Setup and Configuration

Install MongoDB

```bash
# Import MongoDB public GPG key
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

# Create a list file for MongoDB
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb_release -cs)/mongodb-c

# Update package list
sudo apt update

# Install MongoDB
sudo apt install -y mongodb-org

# Start MongoDB service
sudo systemctl start mongod

# Enable MongoDB to start on boot
sudo systemctl enable mongod
```

Configure MongoDB Security

1. Create an admin user:

```bash
# Connect to MongoDB
mongosh

# Switch to admin database
use admin

# Create an admin user
db.createUser({
  user: "adminUser",
  pwd: "SecurePassword123",  # Replace with a strong password
  roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
})

# Exit MongoDB shell
exit
```

2. Enable authentication:

```bash
sudo nano /etc/mongod.conf
```

Add/modify the security section:

```yaml
security:
  authorization: enabled
```

3. Restart MongoDB:

```bash
sudo systemctl restart mongod
```

Create Application Database and User

```bash
# Connect to MongoDB with authentication
mongosh --authenticationDatabase "admin" -u "adminUser" -p "SecurePassword123"

# Create and switch to your application database
use clientoperationdb

# Create a specific user for your application database
db.createUser({
  user: "appuser",
  pwd: "AnotherStrongPassword",  # Replace with a strong password
  roles: [{ role: "readWrite", db: "clientoperationdb" }]
})

# Exit MongoDB shell
exit
```

Configure Node.js to Connect to MongoDB

Update your MongoDB connection string in your backend application:

javascript

```javascript
// In your Node.js app's config or .env file
MONGODB_URI="mongodb://appuser:AnotherStrongPassword@localhost:27017/clientoperationdb"

// In your connection code
const mongoose = require('mongoose');
const mongoURI = process.env.MONGODB_URI || "mongodb://appuser:AnotherStrongPassword@localhost:

mongoose.connect(mongoURI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.error('MongoDB connection error:', err));
```

Set Up MongoDB Backups

1. Create a backup directory:

bash

```bash
mkdir -p ~/mongodb-backups
```

2. Create a backup script:

bash

```bash
nano ~/backup-mongodb.sh
```

Add the following content:

bash

```bash
#!/bin/bash
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
BACKUP_DIR=~/mongodb-backups

mongodump --authenticationDatabase admin -u adminUser -p SecurePassword123 --db clientoperatior

# Keep only the last 7 backups
ls -dt $BACKUP_DIR/*/ | tail -n +8 | xargs rm -rf
```

3. Make the script executable:

```bash
chmod +x ~/backup-mongodb.sh
```

4. Set up a cron job for automated daily backups:

```bash
(crontab -l 2>/dev/null; echo "0 2 * * * ~/backup-mongodb.sh") | crontab -
```

Maintenance and Troubleshooting

Log Locations

- **Apache Logs:**
  - `/var/log/apache2/clientoperation.2ndsource.xyz-access.log`
  - `/var/log/apache2/clientoperation.2ndsource.xyz-error.log`
- **MongoDB Logs:**
  - `/var/log/mongodb/mongod.log`
- **PM2 Logs:**
  - `pm2 logs clientoperation-frontend`
  - `pm2 logs clientoperation-backend`

Common Commands

- **Restart Apache:**

  ```bash
  sudo systemctl restart apache2
  ```

- **Restart MongoDB:**

  ```bash
  sudo systemctl restart mongod
  ```

- **Restart Node.js Applications:**

```bash
pm2 restart clientoperation-backend
pm2 restart clientoperation-frontend
```

- **Check Service Status:**

```bash
sudo systemctl status apache2
sudo systemctl status mongod
pm2 status
```

- **Test Apache Configuration:**

```bash
sudo apachectl configtest
```

Security Best Practices

Firewall Configuration

```bash
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 80
sudo ufw allow 443
```

Regular System Updates

```bash
sudo apt update
sudo apt upgrade
```

MongoDB Security

- Ensure MongoDB is only listening on localhost (default)
- Use strong passwords for all MongoDB users
- Regularly review database users and permissions

SSL/TLS Maintenance

- Certificates will automatically renew via Certbot

- Test renewal process:

  ```bash
  sudo certbot renew --dry-run
  ```

Regular Backups

- Verify backup integrity periodically:

  ```bash
  # Restore to a temporary database for testing
  mongorestore --authenticationDatabase admin -u adminUser -p SecurePassword123 --db test_res
  ```

Important Security Notes

1. Never expose MongoDB port (27017) to the internet
2. Store sensitive credentials in environment variables, not in code
3. Keep all software updated (Node.js, MongoDB, system packages)
4. Consider implementing rate limiting for API endpoints
5. Implement proper authentication and authorization in your application

---

This documentation was generated on May 15, 2025. Some commands or configurations may need updates based on newer software versions.