Apache SSL & Node.js Deployment Guide

A comprehensive guide for deploying a React frontend and Node.js backend application with Apache SSL configuration, including automatic HTTPS redirection, reverse proxy for API calls, and PM2 process management.

🏗️ Architecture

```
Internet → Apache (Port 80/443) → Static Files (/var/www/html/clientoperation)
                                → API Proxy → Node.js Backend (Port 5000)
```

📋 Prerequisites

- Ubuntu/Debian server with sudo access
- Domain name configured (clientoperation.2ndsource.xyz)
- SSL certificates obtained via Let's Encrypt
- Apache2 installed
- Basic familiarity with command line

🍃 MongoDB Setup

Before starting the deployment, ensure MongoDB is properly configured:

Install MongoDB

bash

```bash
# Import the public key used by the package management system
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 focal/mongodb-org/6.0 multiv

# Reload local package database
sudo apt-get update

# Install MongoDB packages
sudo apt-get install -y mongodb-org
```

Configure MongoDB

```bash
# Start MongoDB
sudo systemctl start mongod

# Enable MongoDB to start on boot
sudo systemctl enable mongod

# Check MongoDB status
sudo systemctl status mongod
```

Create Database User

```bash
# Connect to MongoDB
mongo

# Switch to admin database
use admin

# Create admin user
db.createUser({
  user: "adminUser",
  pwd: "YourDBPassword",
  roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
})

# Create application database
use ClientOperation

# Create application user (optional)
db.createUser({
  user: "appUser",
  pwd: "YourAppPassword",
  roles: [ { role: "readWrite", db: "ClientOperation" } ]
})

# Exit MongoDB shell
exit
```

🚀 Quick Start

Step 1: Apache SSL Configuration

Create Apache Virtual Host Configuration:

File Path: `/etc/apache2/sites-available/clientoperation.2ndsource.xyz.conf`

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

Configuration Content:

apache

```
# HTTP to HTTPS Redirect
<VirtualHost *:80>
    ServerName clientoperation.2ndsource.xyz
    Redirect permanent / https://clientoperation.2ndsource.xyz/
   RewriteEngine on
   RewriteCond %{SERVER_NAME} =clientoperation.2ndsource.xyz
   RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]

</VirtualHost>

# HTTPS Virtual Host
<VirtualHost *:443>
    ServerName clientoperation.2ndsource.xyz
    ServerAdmin webmaster@localhost

    # SSL Configuration
    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/clientoperation.2ndsource.xyz/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/clientoperation.2ndsource.xyz/privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf

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

    # Document Root for React Build
    DocumentRoot /var/www/html/clientoperation

    # Static File Serving
    <Directory "/var/www/html/clientoperation">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted

        # React Router SPA Configuration
        RewriteEngine On
        RewriteBase /
        RewriteRule ^index\.html$ - [L]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule . /index.html [L]
    </Directory>
```

```
    # MIME Type Configuration
    <Files "*.js">
        Header set Content-Type "application/javascript"
    </Files>

    <Files "*.css">
        Header set Content-Type "text/css"
    </Files>

    # API Reverse Proxy
    ProxyPass /api/ http://localhost:5000/api/
    ProxyPassReverse /api/ http://localhost:5000/api/
</VirtualHost>
```

Step 2: Enable Required Apache Modules

bash

```
# Enable URL rewriting for SPA
sudo a2enmod rewrite

# Enable HTTP headers modification
sudo a2enmod headers

# Enable SSL support
sudo a2enmod ssl

# Enable reverse proxy functionality
sudo a2enmod proxy
sudo a2enmod proxy_http
```

Expected Output:

```
Enabling module rewrite.
Enabling module headers.
Enabling module ssl.
Enabling module proxy.
Enabling module proxy_http.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Step 3: Directory Setup

bash

```bash
# Create main web directory
sudo mkdir -p /var/www/html/clientoperation

# Set proper ownership for Apache
sudo chown -R www-data:www-data /var/www/html/clientoperation

# Set appropriate permissions
sudo chmod -R 755 /var/www/html/clientoperation
```

Directory Structure:

```
/var/www/html/clientoperation/
├── index.html
├── static/
│   ├── css/
│   ├── js/
│   └── media/
├── manifest.json
└── favicon.ico
```

Step 4: Node.js Environment Setup

bash

```bash
# Install Node.js 18.x
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Verify installation
node --version
npm --version

# Install PM2 globally for process management
sudo npm install -g pm2
```

Expected Output:

```bash
$ node --version
v18.17.0
$ npm --version
9.6.7
$ pm2 --version
5.3.0
```

🎨 Frontend Deployment (In tsstech User)

Step 5: Configure and Build React Frontend (I have filebrowser)

Base Path: `/home/tsstech/nodeapp/clientoperation/frontend`

```bash
# Navigate to frontend directory
cd /home/tsstech/nodeapp/clientoperation/frontend

# Create production environment configuration
cat > .env.production << EOF
REACT_APP_API_URL=https://clientoperation.2ndsource.xyz
REACT_APP_ENV=production
GENERATE_SOURCEMAP=false
EOF

# Install dependencies
npm install

# Create production build
npm run build
```

Build Output Example:

```
Creating an optimized production build...
Compiled successfully.

File sizes after gzip:
  46.61 KB  build/static/js/2.8e5b5f6d.chunk.js
  1.4 KB    build/static/js/main.2f4b5c8a.chunk.js
  1.17 KB   build/static/js/runtime-main.e8e9c4f6.js
  312 B     build/static/css/main.a617e044.chunk.css

The build folder is ready to be deployed.
```

Deploy Built Files

bash

```bash
# Copy build files to web directory
sudo cp -r build/* /var/www/html/clientoperation/

# Set proper ownership
sudo chown -R www-data:www-data /var/www/html/clientoperation/

# Verify deployment
ls -la /var/www/html/clientoperation/
```

⚙️ Backend Deployment (In tsstech User)

Step 6: Configure Node.js Backend (I have filebrowser)

Base Path: `/home/tsstech/nodeapp/clientoperation/backend`

```bash
# Navigate to backend directory
cd /home/tsstech/nodeapp/clientoperation/backend

# Create production environment file
cat > .env << EOF
PORT=5000
NODE_ENV=production
FRONTEND_URL=https://clientoperation.2ndsource.xyz
#Add your database and other configurations:
#MONGO_URI=mongodb://localhost:27017/clientoperation
MONGO_URI=mongodb://adminUser:YourDBPassword@localhost:27017/ClientOperation?authSource=admin
JWT_SECRET=your-super-secret-jwt-key
EOF

# Install production dependencies
npm install --production
```

Backend Configuration Files

server.js: (For testing) Please remove app.use('*');

```
javascript
```

```
import express from 'express';
import connectDB from './config/db.js';
import cors from 'cors';
import dotenv from 'dotenv';

dotenv.config();
const app = express();

connectDB();

// Allowed origins for CORS
const allowedOrigins = [
    'https://clientoperation.2ndsource.xyz', // Production domain
    'http://localhost:3000',                 // React dev server
    'http://localhost:3001'                  // Alternate dev port
];

// Configure CORS middleware
app.use(cors({
    origin: (origin, callback) => {
        if (!origin || allowedOrigins.includes(origin)) {
            callback(null, true);
        } else {
            console.log('CORS blocked origin:', origin);
            callback(new Error('Not allowed by CORS'));
        }
    },
    credentials: true,
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
    allowedHeaders: ['Content-Type', 'Authorization', 'Cookie']
}));

// Health check route
app.get('/api/health', (req, res) => {
    res.json({
        status: 'OK',
        message: 'Backend server is running',
        timestamp: new Date().toISOString(),
    });
});

// Error handling middleware
app.use((err, req, res, next) => {
```

```
    console.error('Error:', err.message);
    res.status(500).json({
        error: 'Internal Server Error',
        message: process.env.NODE_ENV === 'development' ? err.message : 'Something went wrong',
    });
});

// Start server
const PORT = process.env.PORT || 5001;
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
    console.log(`Environment: ${process.env.NODE_ENV || 'development'}`);
    console.log('Allowed Origins:', allowedOrigins);
});
```

db.js:

```
javascript

// config/db.js
import mongoose from 'mongoose';
import dotenv from 'dotenv';

dotenv.config();

const connectDB = async () => {
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI, {
            useNewUrlParser: true,
            useUnifiedTopology: true,
        });
        console.log(`MongoDB Connected: ${conn.connection.host}`);
    } catch (error) {
        console.error(`MongoDB connection failed: ${error.message}`);
        process.exit(1);
    }
};

export default connectDB;
```

🔧 Process Management

Step 7: Configure PM2 Process Manager

Create PM2 Ecosystem File:

```bash
# Create PM2 configuration
cat > ecosystem.config.cjs << EOF
module.exports = {
  apps: [{
    name: 'clientoperation-backend',
    script: './server.js',
    instances: 1,
    exec_mode: 'fork',  //For mongoDB atlas use 'cluster'
    env: {
      NODE_ENV: 'production',
      PORT: 5001,
    },
    env_production: {
      NODE_ENV: 'production',
      MONGO_URI: process.env.MONGO_URI,  // Uses .env file
      JWT_SECRET: process.env.JWT_SECRET,
    },
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log',
    time: true
  }]
};
EOF
```

Testing Backend

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

# Testing purpose
cd /home/tsstech/nodeapp/clientoperation/backend
node server.js
```

If shown a message like `MongoDB connected:` then everything is okay.

```bash
curl http://localhost:5000/api/health
```

Finally stop server then run below:

```bash
# Start application with PM2
pm2 start ecosystem.config.js

# Save PM2 configuration
pm2 save

# Setup PM2 to start on system boot
pm2 startup
```

PM2 Status Output:

```
┌────┬──────────────────────────┬─────────────┬─────┬─────────┬──────────┬────────┬────┬
│ id │ name                     │ namespace   │ ver │ mode    │ pid      │ uptime │ ↺  │
status    │ cpu      │ mem      │ user     │ watching │
├────┼──────────────────────────┼─────────────┼─────┼─────────┼──────────┼────────┼────┼
│ 0  │ clientoperation-backend  │ default     │ 1.0 │ cluster │ 12345    │ 5m     │ 0  │
online    │ 0%       │ 45.2mb   │ tsstech  │ disabled │
└────┴──────────────────────────┴─────────────┴─────┴─────────┴──────────┴────────┴────┴
```

🌐 Service Management

Step 8: Enable and Start Apache

```bash
# Enable the site
sudo a2ensite clientoperation.2ndsource.xyz.conf

# Test Apache configuration
sudo apache2ctl configtest

# Restart Apache to apply changes
sudo systemctl restart apache2

# Enable Apache to start on boot
sudo systemctl enable apache2
```

Configuration Test Output:

```
Syntax OK
```

✅ Testing & Verification

Step 9: Verify Deployment

Test Backend Health:

```bash
# Test backend directly
curl http://localhost:5000/api/health

# Expected response
{"status":"OK","timestamp":"2024-01-15T10:30:00.000Z"}
```

Test Frontend Access:

bash

```
# Test HTTPS redirect
curl -I http://clientoperation.2ndsource.xyz

# Expected response
HTTP/1.1 301 Moved Permanently
Location: https://clientoperation.2ndsource.xyz/

# Test HTTPS access
curl -I https://clientoperation.2ndsource.xyz

# Expected response
HTTP/1.1 200 OK
Content-Type: text/html
```

Service Status Checks:

bash

```
# Check Apache status
sudo systemctl status apache2

# Check PM2 status
pm2 status

# Check application logs
pm2 logs clientoperation-backend --lines 50
```

🔄 Maintenance & Updates

Frontend Updates

```bash
# Navigate to frontend directory
cd /home/tsstech/nodeapp/clientoperation/frontend

# Pull latest changes (if using Git)
git pull origin main

# Install new dependencies (if any)
npm install

# Create new build
npm run build

# Deploy updated build
sudo cp -r build/* /var/www/html/clientoperation/
sudo chown -R www-data:www-data /var/www/html/clientoperation/

# Clear browser cache or add cache-busting
sudo systemctl reload apache2
```

Backend Updates

```bash
# Navigate to backend directory
cd /home/tsstech/nodeapp/clientoperation/backend

# Pull latest changes
git pull origin main

# Install new dependencies
npm install --production

# Restart application
pm2 restart clientoperation-backend

# Monitor restart
pm2 logs clientoperation-backend --lines 20
```

SSL Certificate Renewal

```bash
# Test certificate renewal (dry run)
sudo certbot renew --dry-run

# Renew certificates
sudo certbot renew

# Reload Apache after renewal
sudo systemctl reload apache2
```

🐛 Troubleshooting

Common Issues and Solutions

1. Apache Configuration Errors

Problem: `apache2ctl configtest` fails

```bash
# Check syntax errors
sudo apache2ctl configtest

# View detailed error logs
sudo tail -f /var/log/apache2/error.log
```

Solution: Review configuration file for typos or missing modules.

2. SSL Certificate Issues

Problem: SSL certificate not found

```bash
# Check certificate files exist
ls -la /etc/letsencrypt/live/clientoperation.2ndsource.xyz/

# Test certificate validity
openssl x509 -in /etc/letsencrypt/live/clientoperation.2ndsource.xyz/cert.pem -text -noout
```

3. Backend Connection Issues

Problem: API requests failing

```bash
# Check if backend is running
pm2 status

# Check backend logs
pm2 logs clientoperation-backend

# Test backend directly
curl -v http://localhost:5000/api/health

# Check port binding
netstat -tlnp | grep :5000
```

4. Frontend Not Loading

Problem: React app shows blank page

```bash
# Check if files exist
ls -la /var/www/html/clientoperation/

# Check Apache error logs
sudo tail -f /var/log/apache2/clientoperation.2ndsource.xyz-error.log

# Check browser console for JavaScript errors
# Verify MIME types are set correctly
```

Log Locations

bash

```
# Apache Logs
/var/log/apache2/clientoperation.2ndsource.xyz-error.log
/var/log/apache2/clientoperation.2ndsource.xyz-access.log

# PM2 Logs
/home/tsstech/nodeapp/clientoperation/backend/logs/err.log
/home/tsstech/nodeapp/clientoperation/backend/logs/out.log
/home/tsstech/nodeapp/clientoperation/backend/logs/combined.log

# System Logs
/var/log/syslog
/var/log/apache2/error.log
```

📊 Performance Monitoring

bash

```
# Monitor system resources
htop

# Monitor Apache processes
ps aux | grep apache

# Monitor Node.js process
pm2 monit

# Check disk usage
df -h
du -sh /var/www/html/clientoperation/
```

🔒 Security Considerations

Firewall Configuration:

bash

```bash
# Allow HTTP and HTTPS
sudo ufw allow 80
sudo ufw allow 443

# Block direct access to Node.js port
sudo ufw deny 5000
```

File Permissions:

bash

```bash
# Ensure proper ownership
sudo chown -R www-data:www-data /var/www/html/clientoperation/
sudo chmod -R 755 /var/www/html/clientoperation/
```

Environment Variables:

- Never commit `.env` files to version control
- Use strong, unique secrets for JWT and database connections
- Regularly rotate API keys and passwords

🔧 Quick Reference Commands

bash

```bash
# Restart all services
sudo systemctl restart apache2
pm2 restart all

# View all logs
sudo tail -f /var/log/apache2/*error.log
pm2 logs

# Update and deploy
cd /path/to/frontend && npm run build && sudo cp -r build/* /var/www/html/clientoperation/
cd /path/to/backend && pm2 restart clientoperation-backend

# Check service status
sudo systemctl status apache2
pm2 status
curl -I https://clientoperation.2ndsource.xyz
```

📂 Directory Structure

```
/var/www/html/clientoperation/
├── index.html
├── static/
│   ├── css/
│   ├── js/
│   └── media/
├── manifest.json
└── favicon.ico
```

🤝 Contributing

This documentation provides a complete reference for deploying and maintaining a React/Node.js application with Apache SSL configuration. Keep this guide updated as your infrastructure evolves.

📝 License

This deployment guide is provided as-is for educational and operational purposes.