PACE Case Management System - IT Deployment Guide

For IT Administrators & DevOps Engineers

Version 2.0 | January 2026 | Delhi Operations

Executive Summary

This guide provides step-by-step instructions for IT professionals to deploy the PACE Case Management System on production servers. The deployment takes approximately 45 minutes for experienced administrators and includes database setup, application deployment, SSL configuration, and monitoring.

Key Facts:

- Single-server deployment (scalable to multi-server)
- Estimated cost: ₹1,000-3,000/month for cloud hosting
- Zero-downtime updates supported
- Automated daily backups included
- Production-ready security configuration

Pre-Deployment Checklist

Before starting deployment, verify the following:

- [ ] Server provisioned (Ubuntu 20.04 LTS or 22.04 LTS)
- [ ] SSH access configured
- [ ] Domain registered and DNS A record pointing to server IP
- [ ] SSL certificate path ready (or will use Let's Encrypt)
- [ ] Google Drive API credentials obtained
- [ ] Firewall rules planned (ports 80, 443 required)
- [ ] Backup storage location identified
- [ ] Monitoring tools selected
- [ ] Deployment user created or admin access available

Server Architecture

Component	Technology	Port
Reverse Proxy	Nginx	80, 443
Frontend App	React + Node.js	3000 (internal)
Backend API	Express.js + Node.js	3001 (internal)
Database	PostgreSQL	5432 (localhost only)
Process Manager	PM2	System

Table 1: Technology stack and port allocation

Section 1: Server Setup

1.1 Server Requirements

Minimum Specifications:

- CPU: 2 cores (4 cores recommended)
- RAM: 4GB (8GB for production with high load)
- Storage: 20GB SSD (minimum), 50GB+ recommended
- Operating System: Ubuntu 20.04 LTS or 22.04 LTS
- Network: 100Mbps minimum, 1Gbps recommended
- Uptime: 99.5% SLA minimum

1.2 Initial Server Configuration

Connect to your server via SSH:

ssh root@your-server-ip

Update system packages and install core dependencies:

sudo apt update && sudo apt upgrade -y
sudo apt install -y curl wget git nginx postgresql postgresql-contrib
certbot python3-certbot-nginx build-essential

Verify installations

node --version # Should show v18+
npm --version # Should show 8+
psql --version # Should show PostgreSQL 12+
nginx -v # Should show latest stable

1.3 Create Deployment User

For security, create a dedicated deployment user instead of using root:

Create user without password login

sudo adduser --disabled-password --gecos "PACE Deploy" deploy

Add to sudo group

sudo usermod -aG sudo deploy

Add to www-data group for Nginx

sudo usermod -aG www-data deploy

Switch to deploy user

sudo su - deploy

1.4 Install Node.js 18

From the deploy user account:

Add Node.js repository

curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

Install Node.js

sudo apt install -y nodejs

Install PM2 globally (process manager)

sudo npm install -g pm2

Install Serve (for frontend static hosting)

sudo npm install -g serve

Verify

node --version && npm --version && pm2 --version

Section 2: Database Setup

2.1 PostgreSQL Configuration

Ensure PostgreSQL is running:

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
sudo systemctl status postgresql
```

2.2 Create Database & User

Connect to PostgreSQL as the postgres user:

```
sudo -u postgres psql
```

Inside PostgreSQL prompt, execute:

```
-- Create deployment user
CREATE USER pace WITH PASSWORD 'SecurePace2026Production!';

-- Create production database
CREATE DATABASE pace_production OWNER pace;

-- Grant permissions
GRANT ALL PRIVILEGES ON DATABASE pace_production TO pace;

-- Exit
\q
```

2.3 Test Database Connection

Verify the connection works:

```
psql -U pace -d pace_production -h localhost -c "SELECT NOW();"
```

Expected output: Current timestamp without errors

2.4 Initialize Database Tables

From the application directory:

```
cd ~/pace-app
python3 setup_database_final.py
```

This script automatically:

- Creates all required tables (users, cases, beneficiaries, events, documents, reminders)
- Sets up indexes for performance
- Initializes default admin user
- Verifies schema integrity

Verify tables created:

```
psql -U pace -d pace_production -c "\dt"
```

Section 3: Application Deployment

3.1 Clone Repository

Create application directory and clone:

mkdir -p ~/pace-app ~/backups
cd ~/pace-app

Clone from GitHub

git clone https://github.com/aasthabwj/SOE.git .
git checkout main

Create required directories

mkdir -p backend/uploads backend/logs frontend/dist
chmod 755 backend/uploads backend/logs

3.2 Environment Configuration

Create backend configuration file:

cat > backend/.env << 'EOF'

Database Configuration

DB_HOST=localhost
DB_PORT=5432
DB_USER=pace
DB_PASSWORD=SecurePace2026Production!
DB_NAME=pace_production

Server Configuration

PORT=3001
NODE_ENV=production
FRONTEND_URL=https://yourdomain.com

Security (Generate new secrets in production)

JWT_SECRET=your_random_32_character_secret_key_here_minimum
JWT_EXPIRY=7d

Google Drive Integration (REQUIRED - See Section 3.4)

GOOGLE_CLIENT_ID=your_google_client_id_here
GOOGLE_CLIENT_SECRET=your_google_client_secret_here
GOOGLE_REDIRECT_URI=https://yourdomain.com/google/callback
GOOGLE_ROOT_FOLDER_ID=your_drive_folder_id_here

Optional: Logging

LOG_LEVEL=info
LOG_FILE=./logs/app.log
EOF

Create frontend configuration file:

cat > frontend/.env << 'EOF'
VITE_API_URL=https://yourdomain.com/api
VITE_GOOGLE_CLIENT_ID=your_google_client_id_here
EOF

⚠ **CRITICAL:** Replace placeholder values with actual credentials before proceeding.

3.3 Generate Secure JWT Secret

Generate a cryptographically secure JWT secret:

openssl rand -base64 32

Copy the output and update JWT_SECRET in backend/.env

3.4 Google Drive API Setup (Required for Document Management)

Step 1: Visit Google Cloud Console

- Go to https://console.cloud.google.com/
- Create new project: "PACE-Case-Management"

Step 2: Enable Google Drive API

- Search for "Google Drive API"
- Click Enable

Step 3: Create OAuth 2.0 Credentials

- Go to Credentials → Create Credentials → OAuth 2.0 Client
- Application type: Web application
- Authorized redirect URIs: https://yourdomain.com/google/callback
- Copy Client ID and Client Secret

Step 4: Create Drive Folder

- Open Google Drive
- Create new folder: "PACE-Documents"
- Right-click → Share → Get folder ID from URL
- Format: `https://drive.google.com/drive/folders/FOLDER_ID_HERE`

Step 5: Update .env with credentials

- GOOGLE_CLIENT_ID: (from OAuth credentials)
- GOOGLE_CLIENT_SECRET: (from OAuth credentials)
- GOOGLE_ROOT_FOLDER_ID: (from Drive folder)

3.5 Install Dependencies

Install Node.js dependencies for both backend and frontend:

Backend

```
cd ~/pace-app/backend
npm ci --only=production
npm audit fix
```

Frontend

```
cd ../frontend
npm ci --only=production
npm run build
npm audit fix
```

Note: `npm ci` is preferred over `npm install` for production (uses exact versions from package-lock.json)

Section 4: Process Management with PM2

4.1 PM2 Setup

Create PM2 ecosystem configuration:

```
cat > ~/pace-app/ecosystem.config.js << 'EOF'
module.exports = {
apps: [
{
name: 'pace-backend',
script: './backend/src/server.js',
cwd: '/home/deploy/pace-app',
instances: 1,
exec_mode: 'cluster',
env: {
NODE_ENV: 'production',
PORT: 3001
},
```

```
error_file: './backend/logs/error.log',
out_file: './backend/logs/out.log',
log_date_format: 'YYYY-MM-DD HH:mm:ss Z',
merge_logs: true,
autorestart: true,
max_memory_restart: '500M',
watch: false,
ignore_watch: ['node_modules', 'logs']
},
{
name: 'pace-frontend',
script: 'serve',
args: '-s dist -l 3000',
cwd: '/home/deploy/pace-app/frontend',
instances: 1,
env: {
NODE_ENV: 'production'
},
error_file: './logs/error.log',
out_file: './logs/out.log',
autorestart: true
}
]
};
EOF
```

4.2 Start Services with PM2

Start both applications:

```
cd ~/pace-app
pm2 start ecosystem.config.js --env production
```

Verify both are running

```
pm2 status
```

View logs

```
pm2 logs pace-backend --lines 20
pm2 logs pace-frontend --lines 20
```

4.3 PM2 Auto-Start on Server Reboot

Make PM2 automatically start on system boot:

```
pm2 save
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u
deploy --hp /home/deploy
```

Section 5: Nginx Reverse Proxy Configuration

5.1 Create Nginx Configuration

Create PACE-specific Nginx configuration:

sudo tee /etc/nginx/sites-available/pace > /dev/null << 'EOF'

Redirect HTTP to HTTPS

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

HTTPS Server Block

server {
listen 443 ssl http2;
listen [::]:443 ssl http2;
server_name yourdomain.com www.yourdomain.com;

```
# SSL Certificate Configuration
ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;

# Security Headers
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header X-Content-Type-Options "nosniff" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;

# Log files
access_log /var/log/nginx/pace_access.log combined;
error_log /var/log/nginx/pace_error.log warn;
```

```
# Client body size limit
client_max_body_size 50M;

# Health check endpoint (no logging)
location /health {
    access_log off;
    return 200 "healthy\n";
    add_header Content-Type text/plain;
}

# API endpoints
location /api/ {
    proxy_pass http://localhost:3001/;
    proxy_http_version 1.1;

    # Headers
    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;

    # Timeouts
    proxy_connect_timeout 60s;
    proxy_send_timeout 60s;
    proxy_read_timeout 60s;

    # Buffers
    proxy_buffer_size 128k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;
}

# Frontend (React SPA)
location / {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;

    # WebSocket support
```

```
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';

    # Headers
    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;

    # SPA routing
    try_files $uri $uri/ /index.html;

    # Timeouts
    proxy_connect_timeout 60s;
    proxy_send_timeout 60s;
    proxy_read_timeout 60s;
  }

}
EOF
```

5.2 Enable Configuration

Enable the PACE configuration:

```
# Enable site

sudo ln -sf /etc/nginx/sites-available/pace /etc/nginx/sites-enabled/pace

# Remove default site if not needed

sudo rm -f /etc/nginx/sites-enabled/default

# Test configuration

sudo nginx -t
```

Reload Nginx

sudo systemctl reload nginx
sudo systemctl enable nginx

5.3 Verify Nginx

Check status:

sudo systemctl status nginx
sudo netstat -tlnp | grep nginx

Section 6: SSL/TLS Certificate Setup

6.1 Install Let's Encrypt Certificate

Using Certbot (automatic):

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

The certbot will:

- Validate domain ownership
- Generate SSL certificate
- Update Nginx configuration
- Set up auto-renewal

6.2 Verify Certificate

Check certificate expiration:

sudo certbot certificates

Manual renewal test

sudo certbot renew --dry-run

6.3 Auto-Renewal Setup

Certbot automatically sets up renewal via systemd. Verify:

sudo systemctl list-timers | grep certbot

Section 7: Firewall Configuration

7.1 UFW Firewall Setup

Enable and configure firewall:

Allow SSH (critical to avoid lockout)

sudo ufw allow 22/tcp

Allow HTTP and HTTPS (Nginx)

sudo ufw allow 'Nginx Full'

Allow PostgreSQL from localhost only (already restricted)

sudo ufw allow from 127.0.0.1 to 127.0.0.1 port 5432

Enable firewall

sudo ufw --force enable

Verify rules

sudo ufw status numbered

Section 8: Monitoring & Health Checks

8.1 Health Check Endpoints

Test critical endpoints:

Health check (no auth required)

curl -I https://yourdomain.com/health

API health (no auth required)

curl -I https://yourdomain.com/api/health

Expected response: HTTP/1.1 200 OK

8.2 PM2 Monitoring

View process status in real-time:

Dashboard view

pm2 monit

Status check

pm2 status

View specific logs

pm2 logs pace-backend
pm2 logs pace-frontend

Clear logs

pm2 flush

8.3 Log Files

Access application logs:

Backend logs

tail -f ~/pace-app/backend/logs/app.log
tail -f ~/pace-app/backend/logs/error.log

Nginx access logs

sudo tail -f /var/log/nginx/pace_access.log

Nginx error logs

sudo tail -f /var/log/nginx/pace_error.log

PostgreSQL logs

sudo tail -f /var/log/postgresql/postgresql-*.log

Section 9: Backup & Recovery

9.1 Automated Daily Backups

Create backup script:

cat > ~/backup-pace.sh << 'EOF'
#!/bin/bash

BACKUP_DIR="
$HOME/backups"DB_NAME="pace_production"DB_USER="pace"DATE=$
(date +%Y%m%d_%H%M%S)

Create backup directory

mkdir -p $BACKUP_DIR

Database backup (compressed)

pg_dump -U $DB_USER $DB_NAME | gzip > DATE.sql.gz

Application files backup

tar -czf DATE.tar.gz ~/pace-app --exclude='node_modules'

Keep only last 7 days

find $BACKUP_DIR -name "pace_*" -mtime +7 -delete

Log rotation

echo "Backup completed: $DATE" >> $BACKUP_DIR/backup.log
EOF

chmod +x ~/backup-pace.sh

9.2 Schedule Daily Backups

Add to crontab (runs daily at 2 AM):

crontab -e

Add this line:

0 2 * * * /home/deploy/backup-pace.sh

9.3 Database Restoration

Restore from backup if needed:

List available backups

ls -lh ~/backups/pace_db_*

Restore specific backup

gunzip < ~/backups/pace_db_YYYYMMDD_HHMMSS.sql.gz |
psql -U pace -d pace_production

Section 10: Application Testing

10.1 Pre-Deployment Tests

Before considering deployment complete, verify:

1. Frontend loads

curl -I https://yourdomain.com/

2. API responds

curl https://yourdomain.com/api/health

3. Database connectivity

psql -U pace -d pace_production -c "SELECT COUNT(*) FROM users;"

4. PM2 processes running

pm2 status

10.2 Login Test

Test the application login:

Get login token

TOKEN=$(curl -s -X POST https://yourdomain.com/api/auth/login
-H "Content-Type: application/json"
-d '{"email":"admin@pace.org","password":"password123"}'
| jq -r '.token')

Use token to get user data

curl -H "Authorization: Bearer $TOKEN"
https://yourdomain.com/api/auth/me

10.3 Document Upload Test

Test Google Drive integration:

1. Log in to the application
2. Create a new case
3. Check "Create Google Drive Folder"
4. Upload a test document
5. Verify document appears in Google Drive folder

Section 11: Updates & Maintenance

11.1 Zero-Downtime Updates

Deploy new versions without downtime:

1. Pull latest code

cd ~/pace-app
git pull origin main

2. Install dependencies

cd backend && npm ci --only=production
cd ../frontend && npm ci --only=production && npm run build

3. Run database migrations (if any)

cd ../backend
npm run migrate # If migration script exists

4. Reload without stopping

pm2 reload all

5. Verify

pm2 status
curl https://yourdomain.com/api/health

11.2 Dependency Updates

Keep dependencies current and secure:

cd ~/pace-app

Check for vulnerabilities

npm audit --all

Update packages

npm update

Remove unused packages

npm prune

Test after updates

npm run build && npm test

Section 12: Troubleshooting Guide

Common Issues & Solutions

Issue: "Port 3000 already in use"

Find process using port

lsof -i :3000

Kill process

kill -9 <PID>

Or use different port in PM2 config

Issue: Database connection failed

Check PostgreSQL running

sudo systemctl status postgresql

Test connection manually

psql -U pace -d pace_production -c "SELECT NOW();"

Check .env database credentials

grep DB_ backend/.env

Issue: "Certificate not trusted" errors

Check certificate validity

sudo certbot certificates

Renew immediately

sudo certbot renew --force-renewal

Check Nginx SSL config

sudo nginx -t

Issue: Backend not responding (502 error)

Check backend process

pm2 status pace-backend

Restart if needed

pm2 restart pace-backend

Check backend logs

pm2 logs pace-backend --lines 50

Issue: Google Drive upload fails

Verify Google credentials in .env

grep GOOGLE backend/.env

Check Google Drive API is enabled

(Go to https://console.cloud.google.com/)

Test connection manually

curl -X POST https://yourdomain.com/api/documents/test

Issue: Out of disk space

Check disk usage

df -h

Find large files

du -sh ~/pace-app/*

Clear old backups

rm ~/backups/pace_*_7days_old.sql.gz

Check log files

du -sh ~/pace-app/backend/logs/

Section 13: Security Hardening

13.1 System Security

1. Automatic security updates

sudo apt install -y unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades

2. SSH hardening

sudo nano /etc/ssh/sshd_config

Change: PermitRootLogin no

Change: PasswordAuthentication no

sudo systemctl reload ssh

3. Disable unnecessary services

sudo systemctl disable <service>

13.2 Application Security

- Never commit .env to Git
- Change default admin password on first login
- Rotate JWT_SECRET every 3-6 months
- Enable 2FA when available
- Run npm audit monthly
- Review access logs for suspicious activity

13.3 Database Security

1. Restrict PostgreSQL access

sudo nano /etc/postgresql/*/main/postgresql.conf

Ensure: listen_addresses = 'localhost'

2. Enable password authentication

sudo nano /etc/postgresql/*/main/pg_hba.conf

Ensure: local all postgres md5

Section 14: Performance Optimization

14.1 Database Query Optimization

Enable query logging to find slow queries

psql -U pace -d pace_production

-- Check existing indexes
SELECT * FROM pg_indexes WHERE tablename = 'cases';

-- Add missing indexes
CREATE INDEX idx_cases_status ON cases(status);
CREATE INDEX idx_cases_beneficiary ON cases(beneficiary_id);
CREATE INDEX idx_events_case ON case_events(case_id);

14.2 PM2 Cluster Mode

For multi-core servers, use cluster mode:

In ecosystem.config.js, change:

instances: 'max' # Instead of: instances: 1
exec_mode: 'cluster'

14.3 Nginx Caching

Add to Nginx configuration for static assets:

location ~* .(jpg|jpeg|png|gif|ico|css|js|woff|woff2)$ {
expires 30d;
add_header Cache-Control "public, immutable";
}

Section 15: Rollback Procedure

If issues occur after deployment:

1. Stop current version

pm2 stop all

2. Restore from backup

gunzip < ~/backups/pace_db_PREVIOUS_DATE.sql.gz |
psql -U pace -d pace_production

3. Checkout previous version

cd ~/pace-app
git checkout <previous-commit-hash>

4. Reinstall and rebuild

cd backend && npm ci
cd ../frontend && npm ci && npm run build

5. Start services

pm2 start all

6. Verify

curl https://yourdomain.com/api/health

Deployment Verification Checklist

After completing all sections, verify:

- ✓ https://yourdomain.com loads Dashboard successfully
- ✓ API health check: https://yourdomain.com/api/health returns 200 OK
- ✓ Login works with default credentials (admin@pace.org / password123)
- ✓ Can create new case and add events
- ✓ Google Drive document upload functions
- ✓ PM2 shows all processes as "online"
- ✓ SSL certificate valid (HTTPS working)
- ✓ Nginx error log is clean (sudo tail -f /var/log/nginx/pace_error.log)
- ✓ Database backups are running (check ~/backups/)
- ✓ Firewall rules applied (sudo ufw status)
- ✓ All security headers present (curl -I https://yourdomain.com)

Support & Escalation

For Technical Issues:

Contact the development team with:

1. Error messages and logs (pm2 logs, Nginx logs)
2. Exact steps to reproduce
3. Server configuration (Ubuntu version, Node version, etc.)

Critical Issues:

Gather diagnostics

sudo lsb_release -a # OS info
node --version && npm --version
pm2 status
pm2 logs --lines 50
sudo tail -f /var/log/nginx/pace_error.log

Email to: support@pace.org

Reference Resources

- Node.js Documentation: https://nodejs.org/docs/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- Nginx Documentation: https://nginx.org/en/docs/
- PM2 Documentation: https://pm2.keymetrics.io/
- Let's Encrypt: https://letsencrypt.org/
- Google Drive API: https://developers.google.com/drive/api

Document Version: 2.0
Last Updated: January 2026
Optimized For: Ubuntu 20.04/22.04 LTS
Deployment Time: 45-60 minutes
Support Email: support@pace.org