**Complete Windows Server + Nginx + SQL Server Deployment Guide**

**Canvas LTI 1.3 Attendance System**

**Part 1: Prerequisites**

**Software Required**

* **Windows Server 2019/2022**
* **SQL Server 2019/2022** (Standard/Enterprise/Developer Edition)
* **SQL Server Management Studio (SSMS)**
* **Node.js 18.x LTS** or higher
* **Nginx for Windows**
* **Git for Windows**
* **SSL Certificate** (required for LTI 1.3)
* **PM2** (Node.js process manager)

**Part 2: SQL Server Database Setup**

**2.1 Install SQL Server**

1. Download SQL Server 2022 Developer Edition
2. During installation:
   * Enable **Mixed Mode Authentication**
   * Set SA password (save securely)
   * Enable **TCP/IP protocol**
   * Default port: 1433

**2.2 Configure SQL Server**

Open **SQL Server Configuration Manager**:

1. SQL Server Network Configuration > Protocols
2. Enable **TCP/IP**
3. Right-click TCP/IP > Properties > IP Addresses
4. Set **TCP Port = 1433** for all IPs
5. Restart SQL Server service

**2.3 Create Database**

Connect with **SSMS** and run:

-- Create database

CREATE DATABASE CanvasAttendance;

GO

USE CanvasAttendance;

GO

-- Create login

CREATE LOGIN canvas\_app WITH PASSWORD = 'YourSecurePassword123!';

GO

-- Create user

CREATE USER canvas\_app FOR LOGIN canvas\_app;

GO

-- Grant permissions

ALTER ROLE db\_owner ADD MEMBER canvas\_app;

GO

**2.4 Create Tables**

USE CanvasAttendance;

GO

-- LTI Storage Tables

CREATE TABLE platforms (

id INT IDENTITY(1,1) PRIMARY KEY,

platform\_id VARCHAR(255) UNIQUE,

client\_id VARCHAR(255),

auth\_endpoint VARCHAR(255),

token\_endpoint VARCHAR(255),

jwks\_endpoint VARCHAR(255),

created\_at DATETIME2 DEFAULT GETDATE()

);

CREATE TABLE user\_launches (

id INT IDENTITY(1,1) PRIMARY KEY,

user\_id VARCHAR(255),

user\_name VARCHAR(255),

user\_email VARCHAR(255),

course\_id VARCHAR(255),

course\_name VARCHAR(255),

roles VARCHAR(MAX),

resource\_link\_id VARCHAR(255),

launch\_data NVARCHAR(MAX),

created\_at DATETIME2 DEFAULT GETDATE()

);

CREATE TABLE lti\_storage (

[key] VARCHAR(255) PRIMARY KEY,

value NVARCHAR(MAX),

expires\_at DATETIME2

);

-- Canvas OAuth Tokens

CREATE TABLE canvas\_tokens (

id INT IDENTITY(1,1) PRIMARY KEY,

user\_id VARCHAR(255) NOT NULL,

course\_id VARCHAR(255) NOT NULL,

access\_token NVARCHAR(MAX) NOT NULL,

refresh\_token NVARCHAR(MAX),

expires\_at DATETIME2 NOT NULL,

created\_at DATETIME2 DEFAULT GETDATE(),

updated\_at DATETIME2 DEFAULT GETDATE(),

CONSTRAINT UQ\_user\_course UNIQUE (user\_id, course\_id)

);

CREATE INDEX idx\_canvas\_tokens\_user\_course ON canvas\_tokens(user\_id, course\_id);

CREATE INDEX idx\_canvas\_tokens\_expires ON canvas\_tokens(expires\_at);

-- Attendance Tables

CREATE TABLE attendance (

id INT IDENTITY(1,1) PRIMARY KEY,

course\_id VARCHAR(255) NOT NULL,

course\_sis\_id VARCHAR(255),

course\_name VARCHAR(255),

session\_date DATE NOT NULL,

session\_type VARCHAR(20) DEFAULT 'morning',

student\_id VARCHAR(255) NOT NULL,

student\_sis\_id VARCHAR(255),

status VARCHAR(20) NOT NULL,

marked\_time TIME,

marked\_by VARCHAR(255) NOT NULL,

marked\_by\_sis\_id VARCHAR(255),

instructor\_name VARCHAR(255),

marked\_at DATETIME2 DEFAULT GETDATE(),

CONSTRAINT UQ\_attendance\_record UNIQUE (course\_id, session\_date, session\_type, student\_id),

CONSTRAINT CHK\_status CHECK (status IN ('present', 'absent', 'late', 'excused'))

);

CREATE INDEX idx\_attendance\_course\_date ON attendance(course\_id, session\_date);

CREATE INDEX idx\_attendance\_student ON attendance(student\_id);

CREATE INDEX idx\_attendance\_sis ON attendance(course\_sis\_id, student\_sis\_id);

-- Attendance Audit Log

CREATE TABLE attendance\_audit (

id INT IDENTITY(1,1) PRIMARY KEY,

session\_id INT NOT NULL,

student\_id VARCHAR(255) NOT NULL,

student\_sis\_id VARCHAR(255),

old\_status VARCHAR(20),

new\_status VARCHAR(20) NOT NULL,

changed\_by VARCHAR(255) NOT NULL,

changed\_by\_sis\_id VARCHAR(255),

changed\_at DATETIME2 DEFAULT GETDATE(),

class\_date DATE,

marked\_time TIME,

change\_type VARCHAR(20),

course\_sis\_id VARCHAR(255),

session\_type VARCHAR(20),

CONSTRAINT FK\_attendance FOREIGN KEY (session\_id) REFERENCES attendance(id) ON DELETE CASCADE

);

CREATE INDEX idx\_audit\_session ON attendance\_audit(session\_id);

CREATE INDEX idx\_audit\_student ON attendance\_audit(student\_sis\_id);

CREATE INDEX idx\_audit\_date ON attendance\_audit(class\_date);

GO

**Part 3: Windows Server Setup**

**3.1 Install Node.js**

1. Download Node.js 18.x LTS from https://nodejs.org
2. Run installer with default options
3. Verify installation:

node --version

npm --version

**3.2 Install PM2 Globally**

npm install -g pm2

npm install -g pm2-windows-service

pm2-service-install

**3.3 Install Nginx for Windows**

1. Download from http://nginx.org/en/download.html
2. Extract to C:\nginx
3. Test:

cd C:\nginx

nginx -t

**Part 4: Application Setup**

**4.1 Clone Repository**

cd C:\inetpub

git clone https://github.com/yourusername/canvas-lti-app.git attendance

cd attendance

**4.2 Install Dependencies**

npm install

npm install mssql --save

**4.3 Update Database Library for SQL Server**

Create /lib/db-mssql.js:

import sql from 'mssql';

import crypto from 'crypto';

const config = {

user: process.env.DB\_USER,

password: process.env.DB\_PASSWORD,

server: process.env.DB\_SERVER,

database: process.env.DB\_NAME,

options: {

encrypt: true,

trustServerCertificate: true,

enableArithAbort: true

},

pool: {

max: 10,

min: 0,

idleTimeoutMillis: 30000

}

};

let pool = null;

async function getPool() {

if (!pool) {

pool = await sql.connect(config);

}

return pool;

}

// Encryption utilities

class TokenEncryption {

constructor() {

this.algorithm = 'aes-256-gcm';

this.keyLength = 32;

this.ivLength = 16;

this.tagLength = 16;

this.saltLength = 64;

this.iterations = 100000;

}

deriveKey(password, salt) {

return crypto.pbkdf2Sync(password, salt, this.iterations, this.keyLength, 'sha256');

}

encrypt(text) {

const password = process.env.ENCRYPTION\_KEY;

if (!password) throw new Error('ENCRYPTION\_KEY not configured');

const salt = crypto.randomBytes(this.saltLength);

const key = this.deriveKey(password, salt);

const iv = crypto.randomBytes(this.ivLength);

const cipher = crypto.createCipheriv(this.algorithm, key, iv);

const encrypted = Buffer.concat([

cipher.update(text, 'utf8'),

cipher.final()

]);

const tag = cipher.getAuthTag();

const combined = Buffer.concat([salt, iv, tag, encrypted]);

return combined.toString('base64');

}

decrypt(encryptedText) {

const password = process.env.ENCRYPTION\_KEY;

if (!password) throw new Error('ENCRYPTION\_KEY not configured');

const combined = Buffer.from(encryptedText, 'base64');

const salt = combined.slice(0, this.saltLength);

const iv = combined.slice(this.saltLength, this.saltLength + this.ivLength);

const tag = combined.slice(this.saltLength + this.ivLength, this.saltLength + this.ivLength + this.tagLength);

const encrypted = combined.slice(this.saltLength + this.ivLength + this.tagLength);

const key = this.deriveKey(password, salt);

const decipher = crypto.createDecipheriv(this.algorithm, key, iv);

decipher.setAuthTag(tag);

const decrypted = Buffer.concat([

decipher.update(encrypted),

decipher.final()

]);

return decrypted.toString('utf8');

}

}

const tokenEncryption = new TokenEncryption();

export async function storeCanvasToken(userId, courseId, accessToken, refreshToken = null, expiresIn = 3600) {

const encryptedAccess = tokenEncryption.encrypt(accessToken);

const encryptedRefresh = refreshToken ? tokenEncryption.encrypt(refreshToken) : null;

const expiresAt = new Date(Date.now() + (expiresIn \* 1000));

const dbPool = await getPool();

const result = await dbPool.request()

.input('userId', sql.VarChar, userId)

.input('courseId', sql.VarChar, courseId)

.input('accessToken', sql.NVarChar(sql.MAX), encryptedAccess)

.input('refreshToken', sql.NVarChar(sql.MAX), encryptedRefresh)

.input('expiresAt', sql.DateTime2, expiresAt)

.query(`

MERGE canvas\_tokens AS target

USING (SELECT @userId AS user\_id, @courseId AS course\_id) AS source

ON target.user\_id = source.user\_id AND target.course\_id = source.course\_id

WHEN MATCHED THEN

UPDATE SET

access\_token = @accessToken,

refresh\_token = @refreshToken,

expires\_at = @expiresAt,

updated\_at = GETDATE()

WHEN NOT MATCHED THEN

INSERT (user\_id, course\_id, access\_token, refresh\_token, expires\_at)

VALUES (@userId, @courseId, @accessToken, @refreshToken, @expiresAt);

`);

return result.rowsAffected[0];

}

export async function getCanvasToken(userId, courseId) {

const dbPool = await getPool();

const result = await dbPool.request()

.input('userId', sql.VarChar, userId)

.input('courseId', sql.VarChar, courseId)

.query(`

SELECT access\_token, refresh\_token, expires\_at

FROM canvas\_tokens

WHERE user\_id = @userId

AND course\_id = @courseId

AND expires\_at > GETDATE()

`);

if (result.recordset.length === 0) {

return null;

}

const row = result.recordset[0];

return {

accessToken: tokenEncryption.decrypt(row.access\_token),

refreshToken: row.refresh\_token ? tokenEncryption.decrypt(row.refresh\_token) : null,

expiresAt: row.expires\_at

};

}

export async function deleteCanvasToken(userId, courseId) {

const dbPool = await getPool();

await dbPool.request()

.input('userId', sql.VarChar, userId)

.input('courseId', sql.VarChar, courseId)

.query(`

DELETE FROM canvas\_tokens

WHERE user\_id = @userId AND course\_id = @courseId

`);

return true;

}

export async function refreshCanvasToken(userId, courseId) {

const tokenData = await getCanvasToken(userId, courseId);

if (!tokenData || !tokenData.refreshToken) {

return null;

}

const response = await fetch('https://aui.instructure.com/login/oauth2/token', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: new URLSearchParams({

grant\_type: 'refresh\_token',

client\_id: process.env.CANVAS\_API\_CLIENT\_ID,

client\_secret: process.env.CANVAS\_API\_CLIENT\_SECRET,

refresh\_token: tokenData.refreshToken

})

});

if (!response.ok) {

await deleteCanvasToken(userId, courseId);

return null;

}

const newTokenData = await response.json();

await storeCanvasToken(

userId,

courseId,

newTokenData.access\_token,

newTokenData.refresh\_token || tokenData.refreshToken,

newTokenData.expires\_in || 3600

);

return newTokenData.access\_token;

}

export { sql, getPool };

**4.4 Update Import Statements**

In all your API route files, change:

// OLD

import { neon } from '@neondatabase/serverless';

const sql = neon(process.env.DATABASE\_URL);

// NEW

import { sql, getPool } from '@/lib/db-mssql.js';

Update your attendance routes to use SQL Server syntax:

// Example: /api/attendance/mark/route.js

import { NextResponse } from 'next/server';

import { sql, getPool } from '@/lib/db-mssql.js';

export async function POST(request) {

try {

const pool = await getPool();

const {

course\_id, course\_sis\_id, student\_id, student\_sis\_id,

status, date, session\_type, instructor\_id,

instructor\_sis\_id, course\_name, instructor\_name, marked\_time

} = await request.json();

// Check existing

const existing = await pool.request()

.input('courseId', sql.VarChar, course\_id)

.input('date', sql.Date, date)

.input('sessionType', sql.VarChar, session\_type)

.input('studentId', sql.VarChar, student\_id)

.query(`

SELECT id, status FROM attendance

WHERE course\_id = @courseId

AND session\_date = @date

AND session\_type = @sessionType

AND student\_id = @studentId

`);

const isUpdate = existing.recordset.length > 0;

const oldStatus = isUpdate ? existing.recordset[0].status : null;

// Insert/update

const result = await pool.request()

.input('courseId', sql.VarChar, course\_id)

.input('courseSisId', sql.VarChar, course\_sis\_id)

.input('courseName', sql.VarChar, course\_name)

.input('date', sql.Date, date)

.input('sessionType', sql.VarChar, session\_type)

.input('studentId', sql.VarChar, student\_id)

.input('studentSisId', sql.VarChar, student\_sis\_id)

.input('status', sql.VarChar, status)

.input('markedTime', sql.Time, marked\_time)

.input('instructorId', sql.VarChar, instructor\_id)

.input('instructorSisId', sql.VarChar, instructor\_sis\_id)

.input('instructorName', sql.VarChar, instructor\_name)

.query(`

MERGE attendance AS target

USING (SELECT @courseId AS course\_id, @date AS session\_date,

@sessionType AS session\_type, @studentId AS student\_id) AS source

ON target.course\_id = source.course\_id

AND target.session\_date = source.session\_date

AND target.session\_type = source.session\_type

AND target.student\_id = source.student\_id

WHEN MATCHED THEN

UPDATE SET

status = @status,

marked\_time = @markedTime,

marked\_at = GETDATE(),

marked\_by = @instructorId,

marked\_by\_sis\_id = @instructorSisId

WHEN NOT MATCHED THEN

INSERT (course\_id, course\_sis\_id, course\_name, session\_date, session\_type,

student\_id, student\_sis\_id, status, marked\_time, marked\_by,

marked\_by\_sis\_id, instructor\_name)

VALUES (@courseId, @courseSisId, @courseName, @date, @sessionType,

@studentId, @studentSisId, @status, @markedTime, @instructorId,

@instructorSisId, @instructorName)

OUTPUT INSERTED.id;

`);

const attendanceId = result.recordset[0].id;

// Audit

await pool.request()

.input('sessionId', sql.Int, attendanceId)

.input('studentId', sql.VarChar, student\_id)

.input('studentSisId', sql.VarChar, student\_sis\_id)

.input('courseSisId', sql.VarChar, course\_sis\_id)

.input('sessionType', sql.VarChar, session\_type)

.input('oldStatus', sql.VarChar, oldStatus)

.input('newStatus', sql.VarChar, status)

.input('changedBy', sql.VarChar, instructor\_id)

.input('changedBySisId', sql.VarChar, instructor\_sis\_id)

.input('classDate', sql.Date, date)

.input('markedTime', sql.Time, marked\_time)

.input('changeType', sql.VarChar, isUpdate ? 'update' : 'initial')

.query(`

INSERT INTO attendance\_audit (

session\_id, student\_id, course\_sis\_id, student\_sis\_id, session\_type,

old\_status, new\_status, changed\_by, changed\_by\_sis\_id,

class\_date, marked\_time, change\_type

)

VALUES (

@sessionId, @studentId, @courseSisId, @studentSisId, @sessionType,

@oldStatus, @newStatus, @changedBy, @changedBySisId,

@classDate, @markedTime, @changeType

)

`);

return NextResponse.json({ success: true, wasUpdate: isUpdate });

} catch (error) {

console.error('Error marking attendance:', error);

return NextResponse.json({ error: error.message }, { status: 500 });

}

}

**4.5 Environment Variables**

Create .env.local:

# SQL Server Configuration

DB\_SERVER=localhost

DB\_NAME=CanvasAttendance

DB\_USER=canvas\_app

DB\_PASSWORD=YourSecurePassword123!

# Canvas LTI Configuration

CANVAS\_CLIENT\_ID=170000000000001

CANVAS\_PLATFORM\_URL=https://aui.instructure.com

LTI\_ENCRYPTION\_KEY=your\_32\_character\_encryption\_key\_here\_12345

LTI\_PRIVATE\_KEY=-----BEGIN PRIVATE KEY-----\nYOUR\_PRIVATE\_KEY\_HERE\n-----END PRIVATE KEY-----

# Canvas API OAuth

CANVAS\_API\_CLIENT\_ID=170000000000002

CANVAS\_API\_CLIENT\_SECRET=your\_api\_client\_secret\_here

# Security

JWT\_SECRET=your\_jwt\_secret\_32\_chars\_minimum\_here

ENCRYPTION\_KEY=your\_encryption\_key\_32\_chars\_here

# App URL

APP\_URL=https://yourdomain.com

NODE\_ENV=production

**4.6 Build Application**

npm run build

**Part 5: Generate RSA Keys**

Create scripts/generate-keys.js:

import crypto from 'crypto';

import fs from 'fs';

const { privateKey, publicKey } = crypto.generateKeyPairSync('rsa', {

modulusLength: 2048,

publicKeyEncoding: { type: 'spki', format: 'pem' },

privateKeyEncoding: { type: 'pkcs8', format: 'pem' }

});

const key = crypto.createPublicKey(publicKey);

const jwk = key.export({ format: 'jwk' });

console.log('=== PUBLIC JWK FOR CANVAS ===');

console.log(JSON.stringify({

kty: jwk.kty,

n: jwk.n,

e: jwk.e,

alg: 'RS256',

use: 'sig',

kid: 'lti-key-1'

}, null, 2));

console.log('\n=== PRIVATE KEY FOR .env ===');

console.log(privateKey.replace(/\n/g, '\\n'));

// Save to file

fs.writeFileSync('private-key.pem', privateKey);

fs.writeFileSync('public-jwk.json', JSON.stringify({

kty: jwk.kty,

n: jwk.n,

e: jwk.e,

alg: 'RS256',

use: 'sig',

kid: 'lti-key-1'

}, null, 2));

console.log('\n✓ Keys saved to private-key.pem and public-jwk.json');

Run it:

node scripts/generate-keys.js

**Part 6: Nginx Configuration**

**6.1 SSL Certificate**

Place your SSL certificate files in C:\nginx\ssl\:

* yourdomain.com.crt
* yourdomain.com.key

**6.2 Configure Nginx**

Edit C:\nginx\conf\nginx.conf:

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

# Upstream Node.js application

upstream nodejs\_backend {

server 127.0.0.1:3000;

keepalive 64;

}

# HTTP to HTTPS redirect

server {

listen 80;

server\_name yourdomain.com;

return 301 https://$server\_name$request\_uri;

}

# HTTPS server

server {

listen 443 ssl http2;

server\_name yourdomain.com;

ssl\_certificate C:/nginx/ssl/yourdomain.com.crt;

ssl\_certificate\_key C:/nginx/ssl/yourdomain.com.key;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

client\_max\_body\_size 100M;

location / {

proxy\_pass http://nodejs\_backend;

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;

proxy\_buffering off;

}

# Health check endpoint

location /health {

access\_log off;

return 200 "healthy\n";

add\_header Content-Type text/plain;

}

}

}

**6.3 Test and Start Nginx**

cd C:\nginx

nginx -t

nginx

To reload after changes:

nginx -s reload

**Part 7: Start Application with PM2**

**7.1 Create PM2 Ecosystem File**

Create ecosystem.config.js:

module.exports = {

apps: [{

name: 'canvas-attendance',

script: 'node\_modules/next/dist/bin/next',

args: 'start',

cwd: 'C:/inetpub/attendance',

instances: 2,

exec\_mode: 'cluster',

env: {

NODE\_ENV: 'production',

PORT: 3000

},

error\_file: 'C:/inetpub/attendance/logs/error.log',

out\_file: 'C:/inetpub/attendance/logs/out.log',

log\_date\_format: 'YYYY-MM-DD HH:mm:ss Z',

merge\_logs: true,

autorestart: true,

watch: false,

max\_memory\_restart: '1G'

}]

};

**7.2 Start with PM2**

cd C:\inetpub\attendance

pm2 start ecosystem.config.js

pm2 save

pm2 startup

**7.3 PM2 Management Commands**

pm2 status # Check status

pm2 logs # View logs

pm2 restart canvas-attendance

pm2 stop canvas-attendance

pm2 delete canvas-attendance

**Part 8: Canvas Configuration**

**8.1 Create LTI Developer Key**

1. Canvas Admin → Developer Keys → **+ LTI Key**
2. Configure:
   * **Key Name**: Canvas LTI Attendance System
   * **Title**: Attendance Tracker
   * **Target Link URI**: https://yourdomain.com/api/lti/launch
   * **OpenID Connect Initiation URL**: https://yourdomain.com/api/lti/login
   * **JWK Method**: Public JWK
   * **Public JWK**: Paste content from public-jwk.json
3. **LTI Advantage Services**:
   * ✅ Can retrieve user data
   * ✅ Can view course enrollments (Names and Roles)
4. **Placements**:
   * ✅ Course Navigation
     + Text: Attendance
     + Target Link URI: https://yourdomain.com/api/lti/launch
5. **Save** and note the **Client ID**

**8.2 Create API Developer Key**

1. Canvas Admin → Developer Keys → **+ API Key**
2. Configure:
   * **Key Name**: Canvas API OAuth
   * **Redirect URI**: https://yourdomain.com/api/auth/callback
   * **Scopes**:
     + /api/v1/courses/:course\_id/users
     + /api/v1/courses/:course\_id/enrollments
3. **Save** and note **Client ID** and **Secret**

**8.3 Update Environment Variables**

Update .env.local with the Canvas Client IDs and secrets.

**8.4 Enable Keys**

1. Admin → Developer Keys
2. Toggle both keys to **ON**

**8.5 Install in Canvas**

1. Admin → Settings → Apps → **+ App**
2. Configuration Type: **By Client ID**
3. Enter LTI Client ID
4. Submit

**8.6 Add to Course**

1. Course Settings → Navigation
2. Enable **Attendance** in course navigation
3. Save

**Part 9: Windows Firewall Configuration**

# Allow ports

netsh advfirewall firewall add rule name="HTTP" dir=in action=allow protocol=TCP localport=80

netsh advfirewall firewall add rule name="HTTPS" dir=in action=allow protocol=TCP localport=443

netsh advfirewall firewall add rule name="SQL Server" dir=in action=allow protocol=TCP localport=1433

**Part 10: Testing Checklist**

* [ ] SQL Server accessible via SSMS
* [ ] Database tables created
* [ ] Node.js application builds successfully
* [ ] PM2 service running
* [ ] Nginx serving HTTPS
* [ ] LTI launch works from Canvas
* [ ] User information displays correctly
* [ ] OAuth popup authorizes successfully
* [ ] Roster loads from Canvas API
* [ ] Attendance marking saves to SQL Server
* [ ] Student view shows attendance history
* [ ] Export functionality works

**Part 11: Maintenance**

**Daily Monitoring**

# Check PM2 status

pm2 status

# View logs

pm2 logs --lines 100

# Check Nginx

cd C:\nginx

nginx -t

**Database Backup**

-- In SSMS, run regularly:

BACKUP DATABASE CanvasAttendance

TO DISK = 'C:\Backups\CanvasAttendance\_backup.bak'

WITH FORMAT, COMPRESSION;

**Auto-start Services**

1. **PM2**: Already configured with pm2 startup
2. **Nginx**: Create Windows Service:
   * Download NSSM (Non-Sucking Service Manager)
   * Run: nssm install nginx C:\nginx\nginx.exe
3. **SQL Server**: Already runs as Windows Service

**Part 12: Troubleshooting**

**Issue: SQL Server Connection Failed**

# Check SQL Server is running

services.msc

# Look for "SQL Server (MSSQLSERVER)"

# Test connection

sqlcmd -S localhost -U canvas\_app -P YourPassword

**Issue: Cannot Connect to Database**

* Verify TCP/IP is enabled
* Check firewall allows port 1433
* Verify credentials in .env.local

**Issue: Nginx Won't Start**

# Check error log

C:\nginx\logs\error.log

# Test configuration

nginx -t

**Issue: PM2 Process Crashes**

pm2 logs --err

# Check environment variables

# Check database connection

**Contact & Support**

* Canvas LTI Documentation: https://canvas.instructure.com/doc/api/
* SQL Server Documentation: https://docs.microsoft.com/sql/
* Nginx Documentation: https://nginx.org/en/docs/

**Version**: 1.0.0  
**Last Updated**: October 2025  
**Status**: Production Ready with Windows Server + SQL Server + Nginx