**🚀 Collaboration Tool - Complete Deployment Guide**

**A comprehensive guide to deploy and configure the real-time document collaboration platform.**

**📋 Table of Contents**

1. [**Introduction**](https://www.google.com/search?q=%23-introduction)
2. [**Prerequisites**](https://www.google.com/search?q=%23-prerequisites)
3. [**Environment Setup**](https://www.google.com/search?q=%23-environment-setup)
4. [**Local Development Deployment**](https://www.google.com/search?q=%23-local-development-deployment)
5. [**Production Deployment**](https://www.google.com/search?q=%23-production-deployment)
6. [**Firebase Configuration**](https://www.google.com/search?q=%23-firebase-configuration)
7. [**Troubleshooting**](https://www.google.com/search?q=%23-troubleshooting)
8. [**Security Considerations**](https://www.google.com/search?q=%23-security-considerations)
9. [**Performance Optimization**](https://www.google.com/search?q=%23-performance-optimization)

**📖 Introduction**

**This collaboration tool is a full-stack application designed for real-time document editing and collaboration.**

* **Frontend: React 18 + Vite + TipTap Editor**
* **Backend: Node.js + Express + Socket.io**
* **Database: Firebase Firestore**
* **Authentication: JWT + Firebase Auth**
* **Real-time: Socket.io for signaling and WebRTC for peer-to-peer communication.**

**Application Architecture**

**┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐**

**│ Frontend │◄──►│ Backend │◄──►│ Firebase │**

**│ (React) │ │ (Node.js) │ │ (Firestore) │**

**│ Port: 5173 │ │ Port: 5000 │ │ Cloud │**

**└─────────────────┘ └─────────────────┘ └─────────────────┘**

**│ │**

**└───────────────────────┘**

**Socket.io**

**Real-time Connection**

**🔧 Prerequisites**

**System Requirements**

* **Node.js: v18.0.0 or higher**
* **npm: v8.0.0 or higher**
* **Git: Latest version**
* **Firebase Account: Required for database and authentication services.**

**Supported Platforms**

* **✅ Windows 10/11**
* **✅ macOS 10.15+**
* **✅ Linux (Ubuntu 20.04+)**

**Required Accounts**

1. **Firebase Project -** [**console.firebase.google.com**](https://console.firebase.google.com/)
2. **Domain (for production) - Optional but recommended.**

**🌍 Environment Setup**

**1. Clone the Repository**

**git clone <repository-url>**

**cd collaboration-tool**

**2. Install Dependencies**

**Backend Dependencies**

**cd backend**

**npm install**

**Frontend Dependencies**

**cd ../frontend**

**npm install**

**3. Environment Variables Setup**

**Backend Environment (.env)**

**Create a backend/.env file. This is a critical step for connecting to Firebase.**

**# Server Configuration**

**PORT=5000**

**JWT\_SECRET=your-super-secret-jwt-key-change-this-in-production**

**JWT\_EXPIRES\_IN=7d**

**# Firebase Admin SDK Configuration (get these from your Firebase project)**

**FIREBASE\_PROJECT\_ID=your-firebase-project-id**

**FIREBASE\_PRIVATE\_KEY\_ID=your-private-key-id**

**FIREBASE\_PRIVATE\_KEY="-----BEGIN PRIVATE KEY-----\nyour-private-key-here\n-----END PRIVATE KEY-----\n"**

**FIREBASE\_CLIENT\_EMAIL=firebase-adminsdk-xxxxx@your-project.iam.gserviceaccount.com**

**FIREBASE\_CLIENT\_ID=your-client-id**

**FIREBASE\_AUTH\_URI=https://accounts.google.com/o/oauth2/auth**

**FIREBASE\_TOKEN\_URI=https://oauth2.googleapis.com/token**

**FIREBASE\_DATABASE\_URL=https://your-project.firebaseio.com**

**FIREBASE\_STORAGE\_BUCKET=your-project.appspot.com**

**# Environment Settings**

**NODE\_ENV=development**

**ALLOWED\_ORIGINS=http://localhost:5173,http://localhost:5174**

**Frontend Configuration**

**Update the configuration file at frontend/src/config/config.js:**

**const config = {**

**// Development configuration**

**apiUrl: 'http://localhost:5000/api',**

**wsUrl: 'http://localhost:5000',**

**// Production configuration (uncomment and update for production)**

**// apiUrl: 'https://your-api-domain.com/api',**

**// wsUrl: 'https://your-api-domain.com',**

**};**

**export default config;**

**💻 Local Development Deployment**

**Step 1: Start Backend Server**

**cd backend**

**npm run dev**

**✅ The backend should now be running on http://localhost:5000. The dev script uses nodemon for automatic restarts on file changes.**

**Step 2: Start Frontend Development Server**

**cd frontend**

**npm run dev**

**✅ The frontend should now be running on http://localhost:5173.**

**Step 3: Verify Setup**

1. **Open your browser to http://localhost:5173.**
2. **Check the browser's developer console for any errors.**
3. **Attempt to register a new account and log in.**
4. **Create a new document and test the real-time collaboration features by opening the same document in a second browser window.**

**Development Scripts**

**# Backend**

**npm start # Start production server**

**npm run dev # Start with nodemon (auto-restart)**

**# Frontend**

**npm run dev # Start development server**

**npm run build # Build for production**

**npm run preview # Preview production build**

**npm run lint # Check code quality**

**🌐 Production Deployment**

**Option 1: VPS/Cloud Server Deployment**

**This option provides full control over your environment.**

**1. Server Preparation**

**# Update system packages**

**sudo apt update && sudo apt upgrade -y**

**# Install Node.js**

**curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -**

**sudo apt-get install -y nodejs**

**# Install PM2 for process management**

**sudo npm install -g pm2**

**# Install Nginx for reverse proxy**

**sudo apt install nginx -y**

**2. Application Setup**

**# Clone the repository to the server**

**git clone <repository-url> /var/www/collaboration-tool**

**cd /var/www/collaboration-tool**

**# Install dependencies for production**

**cd backend && npm install --production**

**cd ../frontend && npm install && npm run build**

**3. PM2 Configuration**

**Create an ecosystem.config.js file in the root directory to manage the backend process:**

**module.exports = {**

**apps: [{**

**name: 'collaboration-tool-api',**

**script: './backend/server.js',**

**instances: 'max',**

**exec\_mode: 'cluster',**

**env: {**

**NODE\_ENV: 'production',**

**PORT: 5000**

**},**

**error\_file: './logs/err.log',**

**out\_file: './logs/out.log',**

**log\_file: './logs/combined.log',**

**time: true**

**}]**

**};**

**4. Nginx Configuration**

**Create a new Nginx configuration file at /etc/nginx/sites-available/collaboration-tool:**

**server {**

**listen 80;**

**server\_name your-domain.com;**

**# Serve the frontend static files**

**location / {**

**root /var/www/collaboration-tool/frontend/dist;**

**try\_files $uri $uri/ /index.html;**

**}**

**# Proxy API requests to the backend**

**location /api {**

**proxy\_pass http://localhost:5000;**

**proxy\_http\_version 1.1;**

**proxy\_set\_header Upgrade $http\_upgrade;**

**proxy\_set\_header Connection 'upgrade';**

**proxy\_set\_header Host $host;**

**proxy\_cache\_bypass $http\_upgrade;**

**}**

**# Proxy Socket.io connections**

**location /socket.io/ {**

**proxy\_pass http://localhost:5000;**

**proxy\_http\_version 1.1;**

**proxy\_set\_header Upgrade $http\_upgrade;**

**proxy\_set\_header Connection "upgrade";**

**proxy\_set\_header Host $host;**

**}**

**}**

**5. Enable and Start Services**

**# Enable the Nginx site**

**sudo ln -s /etc/nginx/sites-available/collaboration-tool /etc/nginx/sites-enabled/**

**sudo nginx -t**

**sudo systemctl restart nginx**

**# Start the application with PM2**

**pm2 start ecosystem.config.js**

**pm2 startup**

**pm2 save**

**Option 2: Docker Deployment**

**This option containerizes the application for consistent deployments.**

**1. Create Dockerfile (Backend)**

**backend/Dockerfile**

**FROM node:18-alpine**

**WORKDIR /app**

**COPY package\*.json ./**

**RUN npm ci --only=production**

**COPY . .**

**EXPOSE 5000**

**CMD ["npm", "start"]**

**2. Create Dockerfile (Frontend)**

**frontend/Dockerfile**

**FROM node:18-alpine as builder**

**WORKDIR /app**

**COPY package\*.json ./**

**RUN npm ci**

**COPY . .**

**RUN npm run build**

**FROM nginx:alpine**

**COPY --from=builder /app/dist /usr/share/nginx/html**

**COPY nginx.conf /etc/nginx/conf.d/default.conf**

**EXPOSE 80**

**3. Docker Compose**

**docker-compose.yml**

**version: '3.8'**

**services:**

**backend:**

**build: ./backend**

**ports:**

**- "5000:5000"**

**environment:**

**- NODE\_ENV=production**

**env\_file:**

**- ./backend/.env**

**restart: unless-stopped**

**frontend:**

**build: ./frontend**

**ports:**

**- "80:80"**

**depends\_on:**

**- backend**

**restart: unless-stopped**

**🔥 Firebase Configuration**

**1. Create Firebase Project**

1. **Go to the** [**Firebase Console**](https://console.firebase.google.com/)**.**
2. **Click "Create a project" and follow the setup instructions.**
3. **Enable Firestore Database and Authentication.**

**2. Firestore Setup**

**Apply these security rules in the Firestore console to protect your data:**

**rules\_version = '2';**

**service cloud.firestore {**

**match /databases/{database}/documents {**

**// Users can only access their own user document**

**match /users/{userId} {**

**allow read, write: if request.auth != null && request.auth.uid == userId;**

**}**

**// Document access control**

**match /documents/{documentId} {**

**allow read: if request.auth != null && (**

**resource.data.isPublic == true ||**

**request.auth.uid in resource.data.collaborators**

**);**

**allow write: if request.auth != null &&**

**request.auth.uid in resource.data.collaborators;**

**allow create: if request.auth != null;**

**}**

**// Comments access control**

**match /comments/{commentId} {**

**allow read, write: if request.auth != null;**

**}**

**}**

**}**

**3. Authentication Setup**

1. **In the Firebase console, go to Authentication and enable the Email/Password sign-in provider.**
2. **Configure your authorized domains for production.**

**4. Generate Service Account**

1. **Go to Project Settings → Service Accounts.**
2. **Generate a new private key and download the JSON file.**
3. **Use the credentials from this file to populate the FIREBASE\_\* variables in your backend/.env file.**

**🚨 Troubleshooting**

**Common Issues**

**1. Backend Won't Start**

* **Check Port Availability: netstat -tulpn | grep :5000**
* **Verify Environment Variables: node -e "console.log(process.env.JWT\_SECRET)"**
* **Test Firebase Credentials: node -e "const admin = require('firebase-admin'); console.log('Firebase OK');"**

**2. Frontend Build Errors**

* **Clear Cache: rm -rf node\_modules package-lock.json && npm install**
* **Check for Missing Dependencies: npm audit fix**

**3. Socket.io Connection Issues**

* **Verify ALLOWED\_ORIGINS in your backend .env file.**
* **Check firewall rules on your server.**
* **Ensure your reverse proxy (Nginx) is correctly configured for WebSocket connections.**

**Log Analysis**

* **PM2 Logs: pm2 logs collaboration-tool-api**
* **Nginx Logs: tail -f /var/log/nginx/error.log**

**🔒 Security Considerations**

**Critical Security Measures**

* **Rate Limiting:**
* **// Install: npm install express-rate-limit**
* **const rateLimit = require('express-rate-limit');**
* **const limiter = rateLimit({ windowMs: 15 \* 60 \* 1000, max: 100 });**
* **app.use(limiter);**
* **Security Headers:**
* **// Install: npm install helmet**
* **const helmet = require('helmet');**
* **app.use(helmet());**

**Input Sanitization**

* **Sanitize User Input:**
* **// Install: npm install dompurify**
* **const DOMPurify = require('dompurify');**
* **const sanitizedContent = DOMPurify.sanitize(userInput);**

**Production Environment Variables**

* **Use Strong Secrets: JWT\_SECRET=$(openssl rand -base64 64)**
* **Set Production Mode: NODE\_ENV=production**
* **Restrict Origins: ALLOWED\_ORIGINS=https://yourdomain.com**

**SSL Certificate (Production)**

* **Install Certbot: sudo apt install certbot python3-certbot-nginx**
* **Generate Certificate: sudo certbot --nginx -d yourdomain.com**

**⚡ Performance Optimization**

**1. Frontend Optimization**

* **Code Splitting: Use dynamic imports in React to split your code into smaller chunks.**
* **Asset Compression: Ensure your build process minifies and compresses assets.**

**2. Backend Optimization**

* **Compression: const compression = require('compression'); app.use(compression());**
* **Connection Pooling: The Firebase Admin SDK handles this automatically.**

**3. Database Optimization**

* **Create composite indexes in Firestore for complex queries.**
* **Implement pagination for large datasets.**

**📊 Monitoring & Analytics**

**Health Check Endpoint**

**A health check endpoint is available at /health to monitor the status of the application.**

**Performance Monitoring**

* **PM2 Monitoring: pm2 monit**
* **System Resources: htop, df -h**

**⚠️ Important: This application has security vulnerabilities if not configured correctly. Do not deploy to production without implementing the security measures mentioned in this guide.**