Story Clash: Production Deployment Guide

From Localhost to Live App Store Preview

Current Status: Production-Ready ✅

Your codebase is now stable, tested, and validated:

- ✅ All tests passing (story branching, room flow, type safety)
- ✅ Build succeeds with zero errors
- ✅ Runtime stable on Node 20
- ✅ Hook order bug fixed (critical for multiplayer)
- ✅ Clean file structure with documentation
- ✅ Full demo + real multiplayer paths working locally

Next step: Deploy to production infrastructure so others can access it.

Deployment Strategy: Best-in-Class Stack

For Story Clash to hit App Store #1 quality, you need:

1. **Zero-downtime frontend** with instant global delivery
2. **Persistent WebSocket server** for real-time multiplayer
3. **PostgreSQL database** for rooms, sessions, and analytics
4. **CDN for assets** (audio files, images)
5. **SSL/HTTPS everywhere** (required for iOS)

Recommended Stack (Optimized for Speed + Cost)

Component	Service	Why	Cost
Frontend + API Routes	Vercel	Best Next.js performance, auto-deploys from Git, serverless API routes	Free tier: unlimited bandwidth, 100GB/month
WebSocket Server	Railway.app	Persistent server for Socket.io, auto-restarts, built-in logs	$5/month (500MB RAM, sufficient for 100+ concurrent rooms)
Database	Supabase	Managed PostgreSQL, built-in auth, real-time subscriptions, generous free tier	Free tier: 500MB DB, 2GB bandwidth/month (enough for beta)
Audio Assets	Vercel CDN (via /public)	Automatically CDN-distributed, zero config	Included in Vercel free tier
Domain	Namecheap or Cloudflare	Custom domain for professional look	$10-15/year

Total Monthly Cost (Beta Phase): $5/month
Total Monthly Cost (Post-Launch, <10K users): $20-30/month
Total Monthly Cost (Scale to 100K users): $100-200/month

Phase 1: Deploy Frontend + API (Vercel)

Timeline: 20 minutes

Step 1: Prepare Repository

Ensure your Git repo is clean and pushed:

```
cd /Users/deafgod/Desktop/Codex
git init # if not already a repo
git add .
git commit -m "Production-ready Story Clash v1.0"
```

Push to GitHub:

Create repo on GitHub (https://github.com/new)

Name: story-clash

Private or public (private recommended for now)

git remote add origin https://github.com/YOUR_USERNAME/story-clash.git
git branch -M main
git push -u origin main

Step 2: Connect Vercel

1. Go to vercel.com → Sign up with GitHub
2. Click "New Project"
3. Import your story-clash repo
4. Configure:
   - **Framework Preset:** Next.js (auto-detected)
   - **Root Directory:** ./ (default)
   - **Build Command:** npm run build (default)
   - **Output Directory:** .next (default)
   - **Node Version:** 20.x (set in Environment Variables)
5. Environment Variables (add these):
   NEXT_PUBLIC_SOCKET_URL=https://story-clash-ws.up.railway.app
   DATABASE_URL=postgresql://[will add after Supabase setup]
   NODE_VERSION=20
   (Leave DATABASE_URL blank for now, we'll add it in Phase 3)
6. Click "Deploy"

Result: Your frontend will deploy in ~2 minutes. You'll get a URL like: https://story-clash.vercel.app

Test: Open the URL, verify home screen loads. Create/Join room will fail (expected, WebSocket server not deployed yet).

Phase 2: Deploy WebSocket Server (Railway)

Timeline: 15 minutes

Step 1: Prepare Socket.io Server for Deployment

Your server code is at /server/index.ts. Railway needs a few tweaks:

Create /server/package.json:

```
{
"name": "story-clash-server",
"version": "1.0.0",
"main": "index.ts",
"scripts": {
"start": "tsx index.ts"
},
"dependencies": {
"express": "^4.18.2",
"socket.io": "^4.6.1",
"cors": "^2.8.5",
"tsx": "^4.7.0"
},
"engines": {
"node": "20.x"
}
}
```

Update /server/index.ts to read port from environment:

```
const PORT = process.env.PORT || 3001;

server.listen(PORT, '0.0.0.0', () => {
console.log(Socket.io server running on port ${PORT});
});
```

Add CORS to allow Vercel frontend:

```
import cors from 'cors';

app.use(cors({
origin: [
'https://story-clash.vercel.app',
'http://localhost:3000' // for local dev
],
credentials: true
}));

const io = new Server(server, {
cors: {
origin: [
'https://story-clash.vercel.app',
'http://localhost:3000'
],
credentials: true
}
});
```

Commit changes:
git add server/
git commit -m "Prepare server for Railway deployment"
git push

Step 2: Deploy to Railway

1. Go to railway.app → Sign up with GitHub
2. Click "New Project" → "Deploy from GitHub repo"
3. Select your `story-clash` repo
4. Railway will auto-detect Node.js, but we need to configure:

Settings → Deploy:

- **Root Directory:** `/server`
- **Start Command:** `npm start`

Settings → Environment:
Add these variables:
NODE_ENV=production
PORT=3001
ALLOWED_ORIGINS=https://story-clash.vercel.app,http://localhost:3000

5. Railway will auto-deploy. Wait ~3 minutes.
6. Once deployed, Railway gives you a public URL like:
   https://story-clash-ws.up.railway.app

Step 3: Connect Frontend to WebSocket Server

Go back to Vercel dashboard:

1. Your project → Settings → Environment Variables
2. Update `NEXT_PUBLIC_SOCKET_URL`:
   NEXT_PUBLIC_SOCKET_URL=https://story-clash-ws.up.railway.app
3. Click "Redeploy" (trigger new build with updated env var)

Test: Open `https://story-clash.vercel.app`, create room, join from another device/browser.
Check Railway logs (Deployments → Logs) to see WebSocket connections.

Phase 3: Database Setup (Supabase)

Timeline: 10 minutes

Step 1: Create Supabase Project

1. Go to supabase.com → Sign up
2. Create new project:
   - **Name:** story-clash
   - **Database Password:** [generate strong password, save it]
   - **Region:** Choose closest to your target users (US East for USA, EU West for Europe)
3. Wait ~2 minutes for project to provision

Step 2: Run Database Schema

Copy your schema from /supabase/schema.sql.

Go to Supabase dashboard:

1. Click "SQL Editor" in left sidebar
2. Click "New Query"
3. Paste your schema SQL
4. Click "Run"

Verify: Go to "Table Editor" tab, you should see tables: rooms, players, stories, sessions.

Step 3: Get Connection String

1. Supabase dashboard → Settings → Database
2. Copy "Connection string" (URI format):
   postgresql://postgres:[YOUR_PASSWORD]@db.[PROJECT_REF].supabase.co:5432/postgres

Step 4: Add to Environment Variables

Vercel:

1. Project settings → Environment Variables
2. Add:
   DATABASE_URL=postgresql://postgres:[PASSWORD]@db.[REF].supabase.co:5432/postgres
3. Redeploy

Railway:

1. Server project → Variables
2. Add same DATABASE_URL
3. Redeploy

Test: Create room on live site, check Supabase Table Editor → rooms table should have a new row.

Phase 4: Audio Assets Optimization

Timeline: 15 minutes

Your audio files are in /public/sounds/. Vercel will serve these via CDN automatically, but optimize first:

Step 1: Compress Audio

Use online-audio-converter.com or Audacity:

- **Format:** MP3
- **Bitrate:** 128kbps (sweet spot for quality vs size)
- **Sample Rate:** 44.1kHz

Target file sizes:

- Ambient loops: <500KB
- Short effects (clicks, beeps): <50KB
- Music stings: <300KB

Step 2: Preload Strategy

Update your sound manager (/src/lib/soundManager.ts) to preload on app init:

```
// In root layout or _app.tsx
useEffect(() => {
soundManager.preloadSounds([
'/sounds/lobby-ambient.mp3',
'/sounds/minigame-countdown.mp3',
'/sounds/zombie-soundscape.mp3',
// ... all sounds
]);
}, []);
```

This prevents mid-game loading delays.

Step 3: iOS Safari Audio Fix

iOS blocks autoplay. Add user interaction gate:

```
// Show on first app load
const [audioEnabled, setAudioEnabled] = useState(false);

if (!audioEnabled) {
return (

<button onClick={() => {
soundManager.unlockAudio(); // plays silent audio to unlock
setAudioEnabled(true);
}}>
▯ Enable Sound & Start
</button>

);
}
```

Phase 5: Custom Domain (Optional but Recommended)

Timeline: 10 minutes

Why Custom Domain?

- Looks professional: `storyclash.app` vs `story-clash.vercel.app`
- Better for App Store listing (can use domain in marketing)
- Builds brand equity

Steps

1. **Buy domain** (recommendations):
   - `storyclash.app` (if available, ~$15/year on Namecheap)
   - `storyclash.io` or `.com` alternatives
2. **Configure DNS** (in Namecheap or your registrar):
   - Add CNAME record:
     ```
     Type: CNAME
     Name: @ (or www)
     Value: cname.vercel-dns.com
     ```
3. **Add to Vercel:**
   - Project settings → Domains
   - Add `storyclash.app`
   - Vercel auto-configures SSL (takes 5-10 minutes)
4. **Update WebSocket CORS:**
   - Railway server → Environment Variables
   - Update `ALLOWED_ORIGINS`:
     ```
     ALLOWED_ORIGINS=https://storyclash.app,https://story-clash.vercel.app,
     http://localhost:3000
     ```

Result: Your app is now at `https://storyclash.app` with SSL.

Phase 6: Production Testing Checklist

Before sharing publicly, validate everything works:

Functional Tests

- [ ] **Home screen loads** on desktop and mobile (iOS Safari, Chrome)
- [ ] **Create room** generates code, navigates to lobby
- [ ] **Join room** with valid code works, invalid code shows error
- [ ] **Lobby sync:** Open on 2 devices, both see each other join in real-time
- [ ] **Start game** (host) → both devices navigate to minigame
- [ ] **Minigame:** Both can play, scores sync, leaderboard appears
- [ ] **Genre selection:** Top scorer picks, both see reveal
- [ ] **Story phase:**
  - Active player sees choice buttons + timer
  - Spectating player sees "Waiting for..." message
  - Choice submission advances scene for both
  - Turn rotates correctly
  - Free-choice text works
  - Timer timeout auto-selects choice
- [ ] **Ending:** Both reach recap screen, see same ending
- [ ] **Recap:** Timeline shows all choices correctly
- [ ] **Play Again:** Returns to lobby with same players

Performance Tests

- [ ] **Lighthouse score:** Run on https://storyclash.app (target 90+ on mobile)
- [ ] **Load time:** First page load <2s on 4G connection
- [ ] **WebSocket latency:** Choice submission → next scene <500ms
- [ ] **Audio playback:** No stuttering, sounds trigger at correct moments
- [ ] **Memory leaks:** Play 3 games in a row, check browser memory doesn't grow unbounded

Edge Case Tests

- [ ] **Player disconnect:** Close tab mid-game, other players continue (turn skipped after timeout)
- [ ] **Room expiration:** Leave room idle for 30 minutes, verify new joins are rejected
- [ ] **Invalid room code:** Join with fake code, shows error
- [ ] **Network interruption:** Turn off WiFi mid-game, verify reconnection works
- [ ] **Multiple tabs:** Same player opens 2 tabs with same room, verify no desync

Browser/Device Tests

- [ ] **iOS Safari:** iPhone 12 or newer
- [ ] **Chrome Mobile:** Android device
- [ ] **Desktop Chrome:** Latest version
- [ ] **Desktop Safari:** Latest macOS
- [ ] **Desktop Firefox:** Latest version

Phase 7: Monitoring & Analytics (Post-Launch)

Once live with beta testers, add observability:

Error Tracking: Sentry

1. Sign up at sentry.io
2. Install SDK:
   ```
   npm install @sentry/nextjs @sentry/node
   ```
3. Initialize in Next.js:
   ```
   // sentry.client.config.ts
   import * as Sentry from "@sentry/nextjs";
   Sentry.init({
   dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
   environment: process.env.NODE_ENV,
   tracesSampleRate: 1.0,
   });
   ```
4. Add to Railway server too (catches WebSocket errors)

Result: Auto-captures crashes, performance issues, helps debug production bugs.

Analytics: Vercel Analytics + Custom Events

1. Enable Vercel Analytics (free tier, built-in)
2. Add custom event tracking:
```
import { track } from '@vercel/analytics';
// Track key actions
track('room_created', { players: 3 });
track('game_completed', { genre: 'zombie', ending: 'triumph' });
track('minigame_score', { score: 285 });
```

Result: Understand user behavior, optimize conversion funnel.

Database Dashboard: Supabase Logs

Supabase dashboard → Logs shows:

- Query performance
- Connection errors
- Table growth

Set up alerts: If `rooms` table >1000 rows/hour (viral spike), get notified.

Phase 8: Pre-App Store Prep

Once deployed and stable, prep for iOS App Store:

Option A: Progressive Web App (PWA) - Fastest

Add PWA manifest and service worker:

```
// public/manifest.json
{
"name": "Story Clash",
"short_name": "Story Clash",
"start_url": "/",
"display": "standalone",
"background_color": "#1a1a1d",
"theme_color": "#00d9ff",
"icons": [
{
"src": "/icon-192.png",
"sizes": "192x192",
"type": "image/png"
},
{
"src": "/icon-512.png",
"sizes": "512x512",
"type": "image/png"
}
]
}
```

Pros: Users can "Add to Home Screen," works like app, no App Store approval needed.
Cons: Not discoverable in App Store, no IAP (in-app purchases), limited push notifications.

Timeline: 2-3 days to add PWA + test.

Option B: Native iOS App (via Capacitor) - Full App Store

Wrap your Next.js app in native iOS shell:

```
npm install @capacitor/core @capacitor/cli @capacitor/ios
npx cap init "Story Clash" app.storyclash
npx cap add ios
```

Build and open in Xcode:

```
npm run build
npx cap sync
npx cap open ios
```

In Xcode:

1. Configure signing (Apple Developer account required, $99/year)
2. Set app icon, launch screen
3. Test on simulator
4. Archive and submit to App Store Connect

Pros: Full App Store presence, IAP support, push notifications, better performance.
Cons: Requires Apple Developer account, review process (1-3 days), more setup.

Timeline: 1 week to wrap + test, 3-5 days review.

Deployment Runbook (Step-by-Step Commands)

Here's the complete sequence to go from localhost to live in one sitting:

Part 1: Frontend (Vercel)

Terminal on local machine

```
cd /Users/deafgod/Desktop/Codex
```

Commit latest changes

```
git add .
git commit -m "Production-ready v1.0"
```

Push to GitHub (create repo first on github.com/new)

```
git remote add origin https://github.com/YOUR_USERNAME/story-clash.git
git push -u origin main
```

Go to vercel.com

- Sign in with GitHub

- Import story-clash repo

- Add environment variables (leave DATABASE_URL blank for now)

- Deploy

Note your Vercel URL: https://story-clash.vercel.app

Part 2: WebSocket Server (Railway)

Update server code (CORS + PORT)

(See Phase 2 above for code changes)

```
git add server/
git commit -m "Server deployment config"
git push
```

Go to railway.app

- New Project → Deploy from GitHub

- Select story-clash

- Settings → Root Directory: /server

- Add environment variables

- Deploy

Note your Railway URL: https://story-clash-ws.up.railway.app

Update Vercel env var

- NEXT_PUBLIC_SOCKET_URL=https://story-clash-ws.up.railway.app

- Redeploy Vercel

Part 3: Database (Supabase)

Go to supabase.com

- Create project: story-clash

- SQL Editor → paste schema from /supabase/schema.sql → Run

Get connection string from Settings → Database

Add to Vercel + Railway env vars:

- DATABASE_URL=postgresql://...

- Redeploy both

Part 4: Test Production

Open https://story-clash.vercel.app in two browser windows/devices

Window 1: Create Room

Window 2: Join with code from Window 1

Play through: Lobby → Minigame → Game → Recap

Check Railway logs for WebSocket activity:

railway logs --project story-clash-server

Check Supabase Table Editor → rooms table for new rows

If everything works: You're live! ▯

Troubleshooting Common Issues

Issue: "WebSocket connection failed" in browser console

Cause: CORS not configured or Railway server not running.

Fix:

1. Check Railway logs: `railway logs`
2. Verify `ALLOWED_ORIGINS` includes your Vercel URL
3. Test WebSocket directly: Use websocket.org/echo.html to connect to `wss://story-clash-ws.up.railway.app`

Issue: "Database connection refused"

Cause: `DATABASE_URL` incorrect or Supabase project paused.

Fix:

1. Verify connection string (copy fresh from Supabase dashboard)
2. Test locally: `psql "postgresql://..."`
3. Check Supabase project status (free tier pauses after 1 week inactivity)

Issue: Audio not playing on iOS

Cause: iOS blocks autoplay until user interaction.

Fix:

1. Add "Enable Sound" button on first screen (see Phase 4)
2. Call `soundManager.unlockAudio()` on button click
3. Only then preload and play sounds

Issue: Slow WebSocket response (>1s for scene updates)

Cause: Railway server overloaded or poor network routing.

Fix:

1. Check Railway metrics (CPU/memory usage)
2. Upgrade Railway plan ($10/month for 1GB RAM)
3. Optimize server code (reduce DB queries per event)

Issue: Vercel build fails with "Module not found"

Cause: Missing dependency or incorrect import path.

Fix:

1. Check build logs for specific missing module
2. Verify `package.json` has all dependencies
3. Run `npm install` locally, commit `package-lock.json`

Cost Breakdown (Real Numbers)

Beta Phase (0-1000 users)

- **Vercel:** Free
- **Railway:** $5/month
- **Supabase:** Free
- **Domain:** $15/year (~$1.25/month)
- **Sentry:** Free tier (5K errors/month)

Total: $6.25/month

Growth Phase (1K-10K users)

- **Vercel:** Free (stays under limits)
- **Railway:** $10/month (upgrade to 1GB RAM)
- **Supabase:** $25/month (Pro plan for more storage + bandwidth)
- **Domain:** $1.25/month
- **Sentry:** $26/month (Team plan)

Total: $62/month

Scale Phase (10K-100K users)

- **Vercel:** Free (still under limits with good caching)
- **Railway:** $50/month (4GB RAM, auto-scaling)
- **Supabase:** $25/month (may need to upgrade to $99 if >50GB bandwidth)
- **Domain:** $1.25/month
- **Sentry:** $80/month (Business plan)
- **CDN:** Cloudflare Pro $20/month (for audio asset caching)

Total: $176-250/month

Security Checklist

Before going public:

- [ ] **Environment variables:** Never commit secrets to Git (`.env.local` in `.gitignore`)
- [ ] **API rate limiting:** Add rate limits to room creation (max 10 rooms/IP/hour)
- [ ] **Profanity filter:** Implement on free-choice text (server-side)
- [ ] **SQL injection:** Use parameterized queries (Supabase client handles this)
- [ ] **XSS prevention:** Sanitize user input before displaying (use DOMPurify)
- [ ] **CSRF protection:** Next.js handles this for API routes
- [ ] **SSL everywhere:** Vercel + Railway auto-provision SSL
- [ ] **Room code guessing:** 4-char uppercase = 456,976 combinations (sufficient for beta)

Launch Day Checklist

When you're ready to open to public:

Pre-Launch (1 day before)

- [ ] Final production test with 5+ people
- [ ] Prepare social posts (Twitter, Reddit, Product Hunt)
- [ ] Set up status page (e.g., status.storyclash.app via UptimeRobot)
- [ ] Create press kit (logo, screenshots, 1-page description)
- [ ] Email beta testers: "We're launching tomorrow!"

Launch Day

- [ ] **9 AM:** Post to Product Hunt (upvote link, encourage community)
- [ ] **10 AM:** Tweet launch announcement with demo video
- [ ] **11 AM:** Post to relevant subreddits (r/webgames, r/incremental_games, r/gamedev)
- [ ] **12 PM:** Send to press list (TechCrunch tips, Hacker News Show HN)
- [ ] **Throughout day:** Monitor Sentry for errors, Railway logs for traffic spikes
- [ ] **Evening:** Recap metrics, thank early users, plan day 2 improvements

Post-Launch (Week 1)

- [ ] Daily monitoring: uptime, error rate, user feedback
- [ ] Respond to all bug reports within 24 hours
- [ ] Ship one small improvement per day (shows momentum)
- [ ] Collect testimonials from happy users (use for App Store)

Next Immediate Action

You're ready to deploy. Here's what to do RIGHT NOW:

1. **Open 3 browser tabs:**
   - Tab 1: vercel.com/new
   - Tab 2: railway.app/new
   - Tab 3: supabase.com/dashboard/projects
2. **Follow Deployment Runbook** (30 minutes total):
   - Push code to GitHub
   - Deploy to Vercel
   - Deploy server to Railway
   - Set up Supabase
   - Connect all three via environment variables
3. **Test live URL** (5 minutes):
   - Open your Vercel URL on phone + desktop
   - Create room, join from other device
   - Play full game loop
4. **Come back with:**
   - "It's live at [URL], here's what I found..." → We iterate
   - OR "Deployment failed at step X with error Y..." → We debug

You're 30 minutes from having a live, shareable URL that works on any device.

Let's ship it.

Appendix: Alternative Deployment Options

If Vercel/Railway/Supabase don't fit your needs:

Option B: All-in-One (Render.com)

- **Frontend** + **Backend** + **DB** in one place
- **Pro:** Simpler billing, one dashboard
- **Con:** Less specialized than Vercel for Next.js, more expensive at scale
- **Cost:** $7/month (starter plan with 512MB RAM)

Option C: Self-Hosted (Digital Ocean)

- **VPS:** $12/month (2GB RAM, 1 CPU)
- **Setup:** Nginx + PM2 + PostgreSQL + SSL via Let's Encrypt
- **Pro:** Full control, predictable costs
- **Con:** Requires DevOps knowledge, more maintenance
- **Time:** 2-4 hours initial setup

Option D: AWS (Overkill for V1)

- **Frontend:** CloudFront + S3
- **Backend:** ECS Fargate or Lambda
- **DB:** RDS PostgreSQL
- **Pro:** Scales to millions, industry standard
- **Con:** Complex setup, expensive ($50-100/month minimum), slow iteration
- **Recommendation:** Only consider after 100K+ users

For Story Clash V1: Stick with Vercel + Railway + Supabase. It's the fastest path to live with lowest overhead.

Final Words

You've built a **production-ready multiplayer story game** in record time with Codex. The codebase is tested, stable, and architected for scale.

Deployment is the last 10% that unlocks the next 90% of value:

- Beta testers can finally play together
- You can share a link (not localhost screenshots)
- Investors/partners can experience it themselves
- You can start collecting real user feedback
- App Store submission becomes possible

The infrastructure recommended here (Vercel + Railway + Supabase) powers apps with millions of users. You're not "hacking together a prototype"—you're deploying on the same stack as YC startups and funded companies.

Action: Block 1 hour, follow the runbook, ship to production today.

Then we optimize, iterate, and plan App Store launch.

□ **Let's make Story Clash live.**