Verblizr — Cloud & Keys Setup Guide (GCP, OpenAI, Optional Providers)

Verblizr — Cloud & Keys Setup Guide (GCP, OpenAI, Optional Providers)

Last updated: 2025-08-24 12:49 UTC

This document explains **from scratch** how to configure **Google Cloud Platform (GCP)**, **OpenAI**, and optional providers, how to **rotate or switch keys/services**, and a **troubleshooting playbook** for the issues you're most likely to hit while testing.

Project roots used throughout
- Frontend (React Native): `~/VerblizrRN`
- Backend (Node): `~/verblizr-backend`

---

0) Environments & Philosophy

- Use **separate projects** (or at minimum separate credentials) for **dev** and **prod**.
- Keep **secrets out of the repo**. Use `.env` files and environment variables.
- Grant **least privilege** IAM roles. Only what's required.

You can replicate the same steps for **staging** by creating another GCP project + buckets and its own service account.

---

1) GCP — Project, Service Account, APIs, Buckets

1.1 Create/Select a GCP Project
- Example dev project ID (from your current setup): `verblizr-dev-uk`
- You can also use your existing project; just keep the project + service account **consistent**.

Command (sets active project for gcloud/gsutil)

```
gcloud config set project <YOUR_GCP_PROJECT_ID>
# Sets default project so gcloud/gsutil operate on the right resources.
```

1.2 Create a Service Account (SA) and JSON Key
- Console → IAM & Admin → **Service Accounts** → New service account (e.g., `verblizr-service-account`)
- After creation → **Keys** → **Create new key** → **JSON** → download file.

Move key to a safe path on your Mac

```
mkdir -p ~/.gcp
mv ~/Downloads/<downloaded-sa>.json ~/.gcp/verblizr-sa.json
# Stores your service account key securely in ~/.gcp
```

Export in the shell before starting the backend
zsh/bash:

```
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcp/verblizr-sa.json"
# Tells Google SDK where your SA JSON lives, so it can sign URLs & call APIs.
```

fish:

```
set -x GOOGLE_APPLICATION_CREDENTIALS $HOME/.gcp/verblizr-sa.json
# Same as above, fish syntax.
```

***Where we use this:** In the terminal tab where you run the backend (TTS dev API), export this before starting the server.*

1.3 Enable Required Google APIs

- **Cloud Text-to-Speech API** (`texttospeech.googleapis.com`) — for Google TTS.
- (Optional later) **Speech-to-Text** (`speech.googleapis.com`) if you use Google ASR.

Why
If disabled, GCP returns `PERMISSION_DENIED` even with valid credentials.

How
- Console → APIs & Services → Library → Search “Text-to-Speech” → **Enable**.
- Ensure **Billing** is enabled on the project.

1.4 GCS Buckets (Artifacts & Uploads)

You need two buckets (you already use these names in `.env`):

```
GCS_UPLOADS_BUCKET=verblizr-dev-uk-uploads-uk
GCS_ARTIFACTS_BUCKET=verblizr-dev-uk-artifacts-uk
```

***Artifacts** is where we store synthesized TTS MP3s.*

Create buckets (if not present)

```
gsutil mb -p <YOUR_GCP_PROJECT_ID> gs://verblizr-dev-uk-artifacts-uk
gsutil mb -p <YOUR_GCP_PROJECT_ID> gs://verblizr-dev-uk-uploads-uk
# Creates two buckets for artifacts and uploads under your project.
```

Grant minimal IAM on the artifacts bucket to your SA

```
# Allow upload (create objects)
gsutil iam ch   serviceAccount:<YOUR_SA_EMAIL>:roles/storage.objectCreator
gs://verblizr-dev-uk-artifacts-uk

# Allow read (so signed URLs are honored)
gsutil iam ch   serviceAccount:<YOUR_SA_EMAIL>:roles/storage.objectViewer
gs://verblizr-dev-uk-artifacts-uk
# Gives just enough to upload MP3 and read it via signed URL.
```

(Optional) Lifecycle rule to auto-delete old dev files

```
cat > /tmp/lifecycle.json <<'JSON'
{
  "rule": [
    {
      "action": { "type": "Delete" },
      "condition": { "age": 30 }  # delete objects older than 30 days
```

```
    }
  ]
}
JSON

gsutil lifecycle set /tmp/lifecycle.json gs://verblizr-dev-uk-artifacts-uk
# Helps keep dev bucket tidy and costs down.
```

1.5 Backend .env (Node)

File to edit: `~/verblizr-backend/.env`
Why: Backend uses these to know which project/buckets to target.

Template

```
# Google Cloud
GCP_PROJECT_ID=verblizr-dev-uk
GCS_UPLOADS_BUCKET=verblizr-dev-uk-uploads-uk
GCS_ARTIFACTS_BUCKET=verblizr-dev-uk-artifacts-uk

# OpenAI (if you integrate it on backend)
OPENAI_API_KEY=sk-proj-...
OPENAI_BASE_URL=https://api.openai.com/v1   # default
```

Command (what it does):

```
# Verifies .env contains required keys for GCS
grep -E '^(GCP_PROJECT_ID|GCS_UPLOADS_BUCKET|GCS_ARTIFACTS_BUCKET)=' ~/verblizr-backend/.env
# Prints the three lines; helps confirm config before starting the server.
```

---

2) Frontend — API base & temporary TTS test

2.1 API base URL (dev)

File to edit: `~/VerblizrRN/src/config/api.ts`
Why: Directs the app to your local dev backend.

Simulator (localhost ok):

```
export const API_BASE = __DEV__ ? 'http://127.0.0.1:5055' :
'https://YOUR_PROD_API_HOST';
```

Physical iPhone (must use your Mac's LAN IP):

```
export const API_BASE = 'http://<YOUR-MAC-LAN-IP>:5055';
// Run: ipconfig getifaddr en0  # prints your Wi■Fi IP
```

Command (what it does):

```
# Just prints your Mac's Wi■Fi IP so you can paste into API_BASE for device tests
ipconfig getifaddr en0
```

2.2 Temporary TTS test in Dashboard

File to edit: `~/VerblizrRN/src/screens/DashboardScreen.tsx`

Why: Lets you verify TTS end-to-end on device quickly.

See the separate doc “Verblizr_Step5b_TTS_DevSetup.md” for exact snippet you already added.
You can later wrap it in `__DEV__` so it doesn’t show in release builds.

---

3) OpenAI — Keys & Switching Providers

We currently use Google TTS for Step 5b. If/when you add **OpenAI** (e.g., Whisper ASR, translation, or TTS):

3.1 Backend-only (recommended)
File to edit: `~/verblizr-backend/.env`

```
OPENAI_API_KEY=sk-proj-...
OPENAI_BASE_URL=https://api.openai.com/v1
```

Why: Keep API keys on the backend; the app calls your backend, not OpenAI directly.

3.2 Switching services (example patterns)
- **TTS provider switch (Google ↔ OpenAI ↔ ElevenLabs):**
  - Create provider modules, e.g. `src/lib/tts/googleTTS.mjs`, `src/lib/tts/openaiTTS.mjs`, `src/lib/tts/elevenTTS.mjs`.
  - **File to edit:** `~/verblizr-backend/scripts/dev-tts-api.mjs` — change the import to the provider you want.
  - **Why:** Single route, pluggable TTS engines.

Command (what it does):

```
# Example: grep to see which TTS module is wired right now
grep -n "lib/tts" ~/verblizr-backend/scripts/dev-tts-api.mjs
# Shows which provider module is currently imported.
```

3.3 Whisper (ASR) and Translation
- For **ASR**: Use OpenAI Whisper (or Google STT/Deepgram/etc.).

- For **Translation**: Use OpenAI GPT models or Google Translate API.
- Keep **keys & provider URLs** in backend `.env` and a small `provider: 'openai' | 'google' | ...` flag.

---

4) Running Everything

4.1 Start backend with credentials
Where: `~/verblizr-backend`

```
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcp/verblizr-sa.json"
env PORT=5055 node ./scripts/dev-tts-api.mjs
# Starts dev TTS API on http://localhost:5055 with the correct credentials loaded.
```

4.2 Start Metro & iOS (from project root)
Where: `~/VerblizrRN`

```
npx react-native start --reset-cache
# Starts Metro bundler and clears its cache (good after dependency or config changes).
```

```
npx react-native run-ios
# Builds and installs the app in the iOS Simulator.
```

4.3 All-in-one macOS launcher (optional)

File: `~/start-verblizr.sh` (see the other doc for full content)

```
chmod +x ~/start-verblizr.sh
~/start-verblizr.sh
# Opens a Terminal window with 3 tabs: backend, Metro, iOS.
```

---

5) Rotating Keys & Switching Services

5.1 Rotate GCP Service Account key

1) Create a **new** JSON key for the SA (keep old one active).
2) Replace local file:

```
cp ~/.gcp/new.json ~/.gcp/verblizr-sa.json
# Overwrites the path your backend uses.
```

3) **Restart backend** (so the process loads the new key).
4) After verifying, **delete** the old key from GCP Console.

5.2 Switch TTS provider

- **File to edit:** `~/verblizr-backend/scripts/dev-tts-api.mjs` — change provider import.
- **Why:** Central place to flip engines without touching frontend.
- Update provider-specific env keys in `~/verblizr-backend/.env`.

5.3 Switch API base (device vs simulator vs prod)

- **File to edit:** `~/VerblizrRN/src/config/api.ts`
- **Why:** Controls which backend the app calls.

---

6) Troubleshooting Matrix

Symptom	Likely Cause	Fix
`Cannot sign data without client_email`	Backend started without SA env var or wrong/corrupt JSON	Export `GOOGLE_APPLICATION_CREDENTIALS` in the same shell; ensure JSON has `client_email` and `private_key`.
`PERMISSION_DENIED` for Text-to-Speech	API disabled or billing not enabled	Enable **Cloud Text-to-Speech API**; ensure Billing is on; grant `roles/texttospeech.user` to SA.
`storage.objects.create` denied	SA lacks write on artifacts bucket	Grant `roles/storage.objectCreator` on `gs://<ARTIFACTS_BUCKET>`.
Safari AccessDenied on signed URL	SA lacks read on artifacts bucket	Grant `roles/storage.objectViewer` on the bucket.
Works in Simulator but not on iPhone	API_BASE uses `127.0.0.1`	Use your Mac's **LAN IP** in `api.ts`; rebuild app.
`curl` OK, app fails to open URL	Linking/import or duplicate `Linking` imports	Merge to a single `import { Linking, ... } from 'react-native'`.
Metro oddities / stale bundle	Cache conflicts	`npx react-

native start --reset-cache` and optional `watchman watch-del-all`. |
| iOS build slow or failing after changes | Stale Pods/DerivedData | `rm -rf ios/Pods ios/build ios/Podfile.lock && (cd ios && pod install)`, also clear `~/Library/Developer/Xcode/DerivedData`. |
| Fish shell errors on `VAR=value` | Fish uses `set` syntax | Use `set -x VAR value` and `set VAR (cmd ...)`. |

Useful checks

```
# Confirm backend dev TTS API is up
curl -sS http://localhost:5055/__health

# Request a signed URL and print only the URL with jq
curl -sS -X POST http://localhost:5055/api/tts   -H "Content-Type: application/json"
   -d '{"text":"Hello from Verblizr"}' | jq -r .signedUrl

# Test signed URL headers (expect HTTP/2 200)
curl -I "https://storage.googleapis.com/<...signed url...>"
```

---

7) From-Scratch Checklist (Copy/Paste)

1. **Create/Select project** and `gcloud config set project <ID>`
2. **Create service account**, download JSON key → move to `~/.gcp/verblizr-sa.json`
3. **Enable** **Cloud Text-to-Speech API**; ensure **Billing** enabled
4. **Create buckets**: uploads + artifacts
5. **IAM**: grant SA → `storage.objectCreator` & `storage.objectViewer` on artifacts bucket; `roles/texttospeech.user` on project
6. **Backend `.env`** set `GCP_PROJECT_ID`, `GCS_*_BUCKET` (and optional `OPENAI_*`)
7. **Export SA env var** in shell, **start backend** dev TTS API

8. **Set API_BASE** in frontend for Simulator or Device, **run iOS**
9. **Test**: Tap **■ Test TTS**; if issues, use the troubleshooting matrix above.

---

8) File Reference (Where to change what)

- **`~/verblizr-backend/.env`** — GCP project/buckets, OpenAI keys, provider URLs.
  Why: Central config for backend services.
- **`~/verblizr-backend/scripts/dev-tts-api.mjs`** — Which TTS provider module is used.
  Why: Flip engines with a single import change.
- **`~/verblizr-backend/src/lib/tts/*.mjs`** — Provider modules (googleTTS, etc.).
  Why: Where per-provider code lives.
- **`~/verblizr-backend/src/lib/gcs.mjs`** — GCS client + bucket names.
  Why: Upload/signing behavior depends on these.
- **`~/VerblizrRN/src/config/api.ts`** — Frontend base URL to

your backend.
  Why: Device vs simulator vs prod routing.
- **`~/VerblizrRN/src/screens/DashboardScreen.tsx`** — Temporary TTS test UI.
  Why: Quick end-to-end verification on device.

---

9) Security Notes

- Never commit `.env` or SA JSON; keep them in `.gitignore`.
- Use **separate SAs** for dev/staging/prod. Revoke old keys after rotation.
- Only grant **bucket-level** roles needed (`objectCreator`, `objectViewer`) instead of `storage.admin` in production.

That’s it. With this, you can spin up a new environment from scratch, rotate keys, or swap providers confidently. If you want, I can also generate a **PDF** version of this doc or split it into **Dev/Prod checklists**.