**Step-by-step plan**

**0) Prep (5 min)**

* Pick/confirm your **sandbox project** (new or existing).
* Ensure you have at least **Viewer** on prod (via the driver) and **Editor/Owner** on sandbox.
* Agree with the screen-share driver that you’ll be using **Cloud Shell** in prod to run read-only commands and export JSON.

**1) In prod (screen-share): quick project + API baseline**

Open **Cloud Shell** (top-right in console) and run:

export PROD\_PROJECT="your-prod-project-id"

gcloud config set project "$PROD\_PROJECT"

# Enabled services (APIs)

gcloud services list --enabled --format=json > prod.enabled-services.json

# IAM policy (bindings only; no keys/secrets)

gcloud projects get-iam-policy "$PROD\_PROJECT" --format=json > prod.iam.json

Have the driver download these JSONs and share via your secure channel.

Fill the **Project\_APIs** and **IAM\_Bindings** sheets in the workbook.

**2) Inventory core resources (copy/paste or save JSON)**

Run each block below in **Cloud Shell**. For each service, either:

* Save to JSON (preferred), or
* Copy values into the Excel template.

**Networking**

# VPCs / Subnets / Firewalls

gcloud compute networks list --format=json > vpcs.json

gcloud compute networks subnets list --regions=all --format=json > subnets.json

gcloud compute firewall-rules list --format=json > firewalls.json

# Load balancers (global + regional pieces)

gcloud compute forwarding-rules list --format=json > fwd\_rules.json

gcloud compute target-http-proxies list --format=json > target\_http\_proxies.json

gcloud compute target-https-proxies list --format=json > target\_https\_proxies.json

gcloud compute url-maps list --format=json > url\_maps.json

gcloud compute backend-services list --global --format=json > backend\_services\_global.json

gcloud compute backend-services list --regions=all --format=json > backend\_services\_regional.json

gcloud compute health-checks list --format=json > health\_checks.json

gcloud compute ssl-certificates list --format=json > ssl\_certs.json

**Compute / Serverless**

# Compute Engine

gcloud compute instances list --format=json > compute\_instances.json

gcloud compute instance-templates list --format=json > instance\_templates.json

gcloud compute instance-groups managed list --format=json > migs.json

# Cloud Run (repeat per region you see in console)

REGION="asia-south1"

gcloud run services list --platform=managed --region="$REGION" --format=json > run\_services.$REGION.json

# Optional: describe each service for env vars, VPC connectors, etc.

for SVC in $(gcloud run services list --platform=managed --region="$REGION" --format="value(metadata.name)"); do

gcloud run services describe "$SVC" --region="$REGION" --format=json > "run.$SVC.$REGION.json"

done

**1️⃣ Set Sandbox Project**

export SANDBOX\_PROJECT="your-sandbox-project-id"

gcloud config set project $SANDBOX\_PROJECT

# Enable required APIs

gcloud services enable compute.googleapis.com \

run.googleapis.com \

sqladmin.googleapis.com \

pubsub.googleapis.com \

cloudscheduler.googleapis.com \

artifactregistry.googleapis.com \

storage.googleapis.com \

vpcaccess.googleapis.com

**2️⃣ Create Artifact Registry Repo**

gcloud artifacts repositories create sandbox-repo \

--repository-format=docker \

--location=us-central1 \

--description="Sandbox Docker repo" \

--project=$SANDBOX\_PROJECT

**3️⃣ Pull & Push Container Image**

# Authenticate Docker

gcloud auth configure-docker us-central1-docker.pkg.dev

# Pull prod image

docker pull gcr.io/sgs-playvox/github.com/khorospv/actionlog@sha256:5a29f96110f5fe94bdd2a6d1a910ec9fd29415f90f2a5cca216b802612210354

# Tag for sandbox

docker tag gcr.io/sgs-playvox/github.com/khorospv/actionlog@sha256:5a29f96110f5fe94bdd2a6d1a910ec9fd29415f90f2a5cca216b802612210354 \

us-central1-docker.pkg.dev/$SANDBOX\_PROJECT/sandbox-repo/actionlog:latest

# Push to sandbox repo

docker push us-central1-docker.pkg.dev/$SANDBOX\_PROJECT/sandbox-repo/actionlog:latest

**4️⃣ Create Cloud SQL Instance**

gcloud sql instances create khoros-playvox-prod-sbx \

--database-version=POSTGRES\_15 \

--tier=db-f1-micro \

--region=us-central1 \

--project=$SANDBOX\_PROJECT

# Optional: create a database

gcloud sql databases create actionlogdb \

--instance=khoros-playvox-prod-sbx \

--project=$SANDBOX\_PROJECT

**5️⃣ Create Service Account**

gcloud iam service-accounts create sandbox-cloudrun \

--display-name "Sandbox Cloud Run Service Account" \

--project=$SANDBOX\_PROJECT

# Grant Cloud SQL Client & Pub/Sub permissions

gcloud projects add-iam-policy-binding $SANDBOX\_PROJECT \

--member="serviceAccount:sandbox-cloudrun@$SANDBOX\_PROJECT.iam.gserviceaccount.com" \

--role="roles/cloudsql.client"

gcloud projects add-iam-policy-binding $SANDBOX\_PROJECT \

--member="serviceAccount:sandbox-cloudrun@$SANDBOX\_PROJECT.iam.gserviceaccount.com" \

--role="roles/pubsub.editor"

gcloud projects add-iam-policy-binding $SANDBOX\_PROJECT \

--member="serviceAccount:sandbox-cloudrun@$SANDBOX\_PROJECT.iam.gserviceaccount.com" \

--role="roles/storage.admin"

**6️⃣ Create VPC Connector (if Cloud Run needs private SQL access)**

gcloud compute networks vpc-access connectors create sandbox-connector \

--region=us-central1 \

--network=default \

--range=10.8.0.0/28 \

--project=$SANDBOX\_PROJECT

**7️⃣ Deploy Cloud Run Service**

gcloud run deploy actionlog-prod-sbx \

--image=us-central1-docker.pkg.dev/$SANDBOX\_PROJECT/sandbox-repo/actionlog:latest \

--region=us-central1 \

--platform=managed \

--allow-unauthenticated \

--service-account=sandbox-cloudrun@$SANDBOX\_PROJECT.iam.gserviceaccount.com \

--update-env-vars SPRING\_PROFILES\_ACTIVE=sandbox \

--vpc-connector=sandbox-connector \

--max-instances=100 \

--timeout=1200 \

--memory=8Gi \

--cpu=2

**8️⃣ Optional: Create Pub/Sub Topics & Cloud Storage Buckets**

# Pub/Sub

gcloud pubsub topics create actionlog-topic-sbx --project=$SANDBOX\_PROJECT

gcloud pubsub subscriptions create actionlog-sub-sbx --topic=actionlog-topic-sbx --project=$SANDBOX\_PROJECT

# Cloud Storage

gsutil mb -p $SANDBOX\_PROJECT -l us-central1 gs://actionlog-bucket-sbx/

✅ After these steps:

* Cloud Run is deployed in sandbox with the image, service account, Cloud SQL, and VPC connector.
* Pub/Sub and Storage buckets are ready if the app depends on them.
* You can now update Cloud Scheduler jobs to hit the new Cloud Run URL.

# Pull by digest

docker pull gcr.io/sgs-playvox/github.com/khorospv/tabledeletion@sha256:742174b26c400aadaef9c1ee79efae9b54f2d28e1a31368934a6d70f131d30c2

# Retag for sandbox

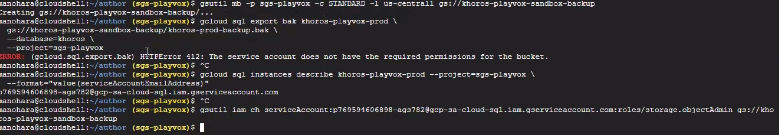
docker tag gcr.io/sgs-playvox/github.com/khorospv/tabledeletion@sha256:742174b26c400aadaef9c1ee79efae9b54f2d28e1a31368934a6d70f131d30c2 gcr.io/khoros-playvox-sandbox/github.com/khorospv/tabledeletion:latest

# Push to sandbox

docker push gcr.io/khoros-playvox-sandbox/github.com/khorospv/tabledeletion:latest

A screen shot of a computer

AI-generated content may be incorrect.



**What to do after sandbox → before prod**

1. **Finish sandbox**
   * Deploy remaining services with immutable images (use tags like :0.0.3 or digests @sha256:...).
   * Keep env-parity: same SPRING\_PROFILES\_ACTIVE, secrets, Cloud SQL connector, etc.
   * Turn on basic autoscaling knobs you’ll want in prod (e.g., --min-instances, --max-instances, --concurrency).
2. **Sandbox validation (sanity + regression)**
   * Health: GET /actuator/health (or your health endpoints) returns 200.
   * DB: connection, migrations (Flyway/Liquibase) run clean; slow queries acceptable.
   * Integrations: any Pub/Sub consumers, Cloud Scheduler jobs, upstream/downstream APIs.
   * Observability: logs show startup on the right port, no errors; metrics/alerts wired.
3. **Document the exact deploy recipe** you used in sandbox (image build, gcloud run deploy flags, env/secrets). This becomes your prod runbook.

**How to discover current prod deployment method**

(so we can mirror it and plan rollback)

* **Cloud Run service config**
* gcloud run services describe <service-name> --region <region> --format yaml
* **Artifact source** (which registry/repo and tag/digest): in the output above.
* **Cloud Build/CI**
* gcloud builds triggers list --region=us-central1
* gcloud builds triggers describe <trigger-name> --region=us-central1

Check if there’s a cloudbuild.yaml in the repo or if Buildpacks/Dockerfile are used.

**Safe production rollout (blue/green with instant rollback)**

**Deploy new revision with zero traffic**, validate, then shift gradually.

# 1) Deploy new revision, DO NOT send traffic yet

gcloud run deploy <svc> \

--image=us-central1-docker.pkg.dev/<proj>/<repo>/<image>:<version> \

--region=<region> \

--platform=managed \

--no-traffic \

--set-env-vars SPRING\_PROFILES\_ACTIVE=prod \

--memory=1Gi

# 2) List revisions and note the NEW one

gcloud run revisions list --service <svc> --region <region>

# Optional: tag revisions for clarity

gcloud run services update-traffic <svc> --region <region> \

--to-revisions <new-rev>=0,<old-rev>=100 --tag new,<new-rev>=0 --tag stable,<old-rev>=100

**Smoke test the new revision** (by tag URL if you added one) before any users see it.

# Logs & health checks

gcloud run services logs read <svc> --region <region> --limit=200

# curl the tagged URL (new) and compare with stable

**Shift traffic gradually (canary):**

# 5% canary

gcloud run services update-traffic <svc> --region <region> \

--to-revisions <new-rev>=5,<old-rev>=95

# If healthy, go 25% → 50% → 100%

gcloud run services update-traffic <svc> --region <region> \

--to-revisions <new-rev>=100

**Instant rollback (if anything goes sideways)**

No rebuild needed—just move traffic back:

# Route 100% back to the previous good revision

gcloud run services update-traffic <svc> --region <region> \

--to-revisions <old-rev>=100

(You can keep both revisions around; Cloud Run revisions are immutable.)

**Production guardrails & tips**

* **DB migrations:** run with Flyway/Liquibase in an idempotent, backward-compatible way; avoid destructive changes until after cutover is stable.
* **Secrets:** prefer Secret Manager (--set-secrets) vs plain env vars.
* **Autoscaling:** set --min-instances for warm starts on critical paths.
* **Observability:** define simple SLOs and alerts (5xx rate, latency p95).
* **Freeze window:** schedule prod cutover when stakeholders can monitor/approve.
* **Approval & audit:** if you add a Cloud Build trigger, enable “require approval”.

If you want, I can turn this into a **one-page Production Implementation Plan** with your exact service names (Actionlog, Author, BatchProcessing, etc.), including pre-checks, commands filled in, and a rollback matrix per service.