R Shiny on Google Cloud Run :: **CHEATSHEET**

Command-Line Workflow

This cheat sheet describes a command-line workflow for deploying R Shiny apps to Google Cloud Run using containers.

Example: dashboard.firsa.eu.

Prerequisites

Have a billing-enabled Google Cloud project, permission to enable APIs and deploy Cloud Run, and a Shiny app that runs locally. Before source deploy, add a Dockerfile and ignore files.

Start in the app folder

```
cd path/to/my-shiny-app
ls

# expected
app.R  Dockerfile  .dockerignore  .gcloudignore
# optional
R/  data/  www/  renv.lock
```

Run deployment commands from the folder with the Dockerfile. Use `.gcloudignore` for source upload and `.dockerignore` for the image build context.

Minimal Dockerfile

```
FROM rocker/r-ver:4.5.0
ENV R_REPOS=https://cloud.r-project.org
RUN Rscript -e \
  'install.packages("shiny", repos=Sys.getenv("R_REPOS"))'
WORKDIR /srv/shiny-app
COPY . .
EXPOSE 8080
CMD Rscript -e "shiny::runApp('/srv/shiny-app', \
  host='0.0.0.0', \
  port=as.numeric(Sys.getenv('PORT','8080')))"
```

Rocker provides Docker base images for R. Apps should install all required R packages, restore from `renv.lock` if used, and add system libraries before R packages.

Google Cloud setup

```
# Official installer:
# https://cloud.google.com/sdk/docs/install

# macOS option if you already use Homebrew:
brew install --cask google-cloud-sdk

gcloud --version
gcloud auth login
gcloud init
```

Use the official installer on Windows, macOS, or Linux. Cloud Shell already has `gcloud` installed.

Set project variables

```
PROJECT_ID="your-project-id"
REGION="europe-west1"
SERVICE="my-shiny-app"

gcloud projects list \
  --format="table(projectId,name)"

gcloud config set project "${PROJECT_ID}"
gcloud config set run/region "${REGION}"
gcloud config list
```

`PROJECT_ID` comes from the `projectId` column. `REGION` is where the service runs. `SERVICE` is the stable Cloud Run name used across revisions.

Enable services

```
gcloud services enable \
  run.googleapis.com \
  cloudbuild.googleapis.com \
  artifactregistry.googleapis.com
```

Cloud Run runs the service. Cloud Build builds the container image. Artifact Registry stores the built image.

Key concepts

Dockerfile: recipe for the R version, system libraries, R packages, copied files, port, and startup command.
Image: the built container stored in Artifact Registry.
Service: the stable Cloud Run endpoint users visit.
Revision: an immutable version created by each deploy.
PORT: Cloud Run passes the listening port through the `PORT` environment variable. Bind Shiny to `0.0.0.0` and read `PORT` in the startup command.
Source deploy: `--source .` uploads the current folder to Cloud Build. With a Dockerfile present, Cloud Build uses it to create the image.

Deploy from local source

```
gcloud run deploy "${SERVICE}" \
  --source . \
  --region "${REGION}" \
  --allow-unauthenticated \
  --memory 1Gi \
  --cpu 1 \
  --concurrency 10 \
  --timeout 3600s \
  --min-instances 0 \
  --max-instances 3
```

The `--source .` flag sends the current folder to Cloud Build. `--allow-unauthenticated` makes a public URL; omit it for private IAM access. More CPU/memory can help package loading and computation but costs more. Lower concurrency isolates users better; higher concurrency uses fewer instances. `--timeout 3600s` supports long Shiny requests. `min-instances 0` saves idle cost but results in -slower-cold starts.

Verify and inspect

After deployment, confirm the public service URL, check that the service exists in the intended region, read recent logs, and review the revision history created by each deploy.

```
gcloud run services describe "${SERVICE}" \
  --region "${REGION}" \
  --format="value(status.url)"

gcloud run services list \
  --region "${REGION}"

gcloud run services logs read "${SERVICE}" \
  --region "${REGION}" \
  --limit 100

gcloud run revisions list \
  --service "${SERVICE}" \
  --region "${REGION}"
```

Use logs first when the build succeeds but the app does not load.

R Shiny on Google Cloud Run :: **CHEATSHEET**

Security

Use `--allow-unauthenticated` only for public apps. For private services, omit it and grant Cloud Run Invoker to selected users or service accounts. Keep secrets and credential files out of the image and repository. Use Secret Manager for credentials; use environment variables only for non-secret configuration, or reference/mount secrets through Cloud Run. Use a least-privilege runtime service account.

Redeploy and scale

```
gcloud run deploy "${SERVICE}" \
  --source . \
  --region "${REGION}" \
  --allow-unauthenticated

gcloud run services update "${SERVICE}" \
  --region "${REGION}" \
  --min-instances 0 \
  --max-instances 3
```

Use `deploy` when the app code, Dockerfile, packages, or bundled data changed. Use `services update` when only Cloud Run settings need to change.

Troubleshooting

```
gcloud run services logs read "${SERVICE}" \
  --region "${REGION}" \
  --limit 100

gcloud builds list \
  --region "${REGION}" \
  --limit 5
```

If deploy fails, start with Cloud Build history. If the URL fails, read Cloud Run logs. A 503 usually means the app did not start, did not bind to `0.0.0.0`, or ignored PORT. For slow starts, reduce image size or raise `min-instances`.

GitHub auto-deploy

Use this when pushes to `main` should build an image, push it to Artifact Registry, and deploy that image to Cloud Run.

Variables

```
CONNECTION="github-connection"
REPO_LINK="my-shiny-app"
REPO_URI="https://github.com/OWNER/REPO.git"
AR_REPO="cloud-run-apps"
SA_RES="projects/PROJECT/serviceAccounts/EMAIL"

BASE="projects/${PROJECT_ID}/locations/${REGION}"
CONN="${BASE}/connections/${CONNECTION}"
REPOSITORY="${CONN}/repositories/${REPO_LINK}"
```

cloudbuild.yaml

```
substitutions:
  # Replace these placeholders.
  _SERVICE: my-shiny-app
  _REGION: europe-west1
  _IMAGE_URI: REGION-docker.pkg.dev/PROJECT_ID/REPOSITORY/IMAGE

steps:
- name: gcr.io/cloud-builders/docker
  args: ["build", "-t", "${_IMAGE_URI}:$COMMIT_SHA", "."]
- name: gcr.io/cloud-builders/docker
  args: ["push", "${_IMAGE_URI}:$COMMIT_SHA"]
- name: gcr.io/google.com/cloudsdktool/cloud-sdk
  entrypoint: gcloud
  args:
  - run
  - deploy
  - ${_SERVICE}
  - --image
  - ${_IMAGE_URI}:$COMMIT_SHA
  - --region
  - ${_REGION}
  - --allow-unauthenticated
```

Connect GitHub

Create the Artifact Registry repository once. Then create the GitHub connection, complete the printed authorization URL and GitHub App installation, and only then create the repository link.

```
gcloud artifacts repositories create "${AR_REPO}" \
  --repository-format=docker \
  --location="${REGION}"

gcloud builds connections create github \
  "${CONNECTION}" \
  --region="${REGION}"

gcloud builds connections describe "${CONNECTION}" \
  --region="${REGION}"

# Open the printed authorization URL and install/
    ↪ authorize
# the GitHub app before creating the repository link.

gcloud builds repositories create \
  "${REPO_LINK}" \
  --remote-uri="${REPO_URI}" \
  --connection="${CONNECTION}" \
  --region="${REGION}"
```

Trigger and IAM

```
gcloud iam service-accounts create cloud-build-trigger

gcloud builds triggers create github \
  --name="deploy-${SERVICE}" \
  --repository="${REPOSITORY}" \
  --branch-pattern="^main$" \
  --build-config="cloudbuild.yaml" \
  --service-account="${SA_RES}" \
  --region="${REGION}"
```

IAM: grant the trigger service account only what this build needs: Cloud Run deploy permission, Artifact Registry write access, and `iam.serviceAccountUser` on the Cloud Run runtime service account.