Implement and release platform changes

Trace behavior from the UI to its source contract and verify upgrades in Google Cloud.

https://docs.qintoravacloud.com/guides/platform-changes/

Locate the change

Start with the feature's form and API references, then determine whether the change belongs to a supported configuration artifact or platform code. For a code change, identify the browser surface, server route, permission and tenant checks, shared record handling, and downstream action. Keep existing public URLs, record identifiers and versioned contracts in mind.

Develop and test

1. Use the authorized Google Cloud development checkout and start a branch from the deployed or accepted source baseline. Review repository instructions and scope.
2. For schema changes, add an append-only migration. Test against a disposable database containing existing data as well as an empty database when appropriate.
3. Exercise the normal UI and API entry points, including the relevant roles, tenant isolation, retries and validation. Use seeded test records for mutations. For an integration, verify the actual provider contract and its failure outcomes in the approved test environment.
4. Build in Google Cloud, record the source commit and immutable image digest, and sync the branch to GitHub. Deploy a candidate and compare expected health, browser journeys and logs before promotion.
5. Verify the live revision, key paths, and persistence. Roll application traffic back to an approved prior revision if compatible with the applied schema. Use the tested migration recovery procedure when schema compatibility requires it.

Identify the serving revision and match a log

Run these commands in Google Cloud. Set `PROJECT`, `REGION`, and `SERVICE` for the tenant you are troubleshooting. `status.traffic` tells you which revision serves requests. A newer revision can be ready with zero public traffic.

```
gcloud run services describe "$SERVICE" --region "$REGION" \
  --project "$PROJECT" --format='yaml(status.traffic,status.latestReadyRevisionName)'

gcloud logging read "resource.type=cloud_run_revision AND \
resource.labels.service_name=$SERVICE AND jsonPayload.errorId=$ERROR_ID" \
```

```
  --project "$PROJECT" --limit 25 --
format='table(timestamp,severity,resource.labels.revision_name,jsonPayload.event,jsonPayloa
d.code)'
```

1. Take `ERROR_ID` from the response for a disposable reproduction. Check that the reported revision matches the serving revision at the request time.
2. If you only have a request timestamp, narrow the Cloud Run request logs to the affected service, revision, method and path. Inspect the status and latency. Compare API Gateway logs if Cloud Run never received the request.
3. Check the candidate’s route and regression tests before moving traffic. After promotion, rerun the synthetic read and a workflow using disposable data. Verify the revision, application result and any affected record or receipt.