Troubleshooting Cloud Run (fully managed)

This page provides troubleshooting strategies as well as solutions for some common problems.

If you see the "*Container failed to start. Failed to start and then listen on the port defined by the PORT environment variable.*" error message when deploying a new revision, use these steps to troubleshoot the problem.

Does your container run locally?

When troubleshooting Cloud Run, your first step should always be to confirm that you can run your container image locally (https://cloud.google.com/run/docs/testing/local). If your container image is not running locally, the root cause of the problem is not coming from Cloud Run. You need to diagnose and fix the issue locally first.

Is your container listening for requests on the expected port?

A common issue is to forget to listen for incoming requests, or to listen for incoming requests on the wrong port.

As documented in the container runtime contract (https://cloud.google.com/run/docs/reference/container-contract#port), your container must listen for incoming requests on the port that is defined by Cloud Run and provided in the `PORT` environment variable.

If your container fails to listen on the expected port, the revision health check will fail, the revision will be in an error state and the traffic will not be routed to it.

Is your container listening on all network interfaces?

A common reason for Cloud Run services failing to start is that the server process inside the container is configured to listen on the `localhost` (`127.0.0.1`) address. This refers to the loopback network interface, which is not accessible from outside the container and therefore Cloud Run health check cannot be performed, causing the service deployment failure.

To solve this, configure your application to start the HTTP server to listen on all network interfaces, commonly denoted as `0.0.0.0`.

Do you see errors in the logs?

You should use Stackdriver Logging (https://cloud.google.com/run/docs/logging) to look for application errors in `stdout` or `stderr` logs. You can also look for crashes captured in Stackdriver Error Reporting (https://cloud.google.com/run/docs/error-reporting). You will probably need to update your code or your revision settings to fix these errors or crashes.

Are your container instances exceeding memory?

Your container instances might be exceeding the available memory. To determine if this is the case, look for such errors in the `varlog/system` logs. If the instances are exceeding the available memory, consider increasing the memory limit (https://cloud.google.com/run/docs/configuring/memory-limits).

Note that the Cloud Run container instances run in an environment where the files written to the local filesystem count towards the available memory. This also includes any log files that are *not* written to `/var/log/*` or `/dev/log`.

403 "Error: Forbidden" when opening or calling the service URL?

If you receive a `403` "Error: Forbidden" error message when accessing your Cloud Run (fully managed) service, it means that your client is not authorized to invoke this service. You can address this by taking one of the following actions:

- If the service is meant to be invocable by anyone, update its IAM settings (https://cloud.google.com/run/docs/securing/managing-access#making_a_service_public) to make the service public.

- If the service is meant to be invocable only by certain identities, make sure that you invoke it with the proper authorization token (https://cloud.google.com/run/docs/securing/authenticating#developers).

Do you see error code 203 for long running requests?

If your service is processing long requests and you have increased the request timeout (https://cloud.google.com/run/docs/configuring/request-timeout), you might still see requests being terminated earlier, with error code 203. This can be caused by your language framework's request timeout setting that you also need to update. For example, Node.js developers need to update the `server.timeout` property (https://nodejs.org/api/http.html#http_server_timeout).

Do you see 503 errors under high load?

The Cloud Run (fully managed) load balancer strives to distribute incoming requests over the necessary amount of container instances. However, if your container instances are using a lot of CPU to process requests, the container instances will not be able to process all of the requests, and some requests will be returned with a 503 error code.

To mitigate this, try lowering the concurrency (https://cloud.google.com/run/docs/about-concurrency) . Start from `concurrency = 1` and gradually increase it to find an acceptable value. Refer to Setting concurrency (https://cloud.google.com/run/docs/configuring/concurrency) for more details.

Do you see 429 errors?

If the service has reached its maximum number of container instances, requests are returned with a 429 error code. Try increasing this limit by increasing the "max instance" settings (https://cloud.google.com/run/docs/configuring/max-instances), or, if you need more than 1000 instances, by requesting a quota increase (https://cloud.google.com/run/quotas#increase).

Do your requests abort because no instance is available?

You can get the following error if the Cloud Run (fully managed) infrastructure didn't scale fast enough to catch up with the traffic spike:

`The request was aborted because there was no available instance`

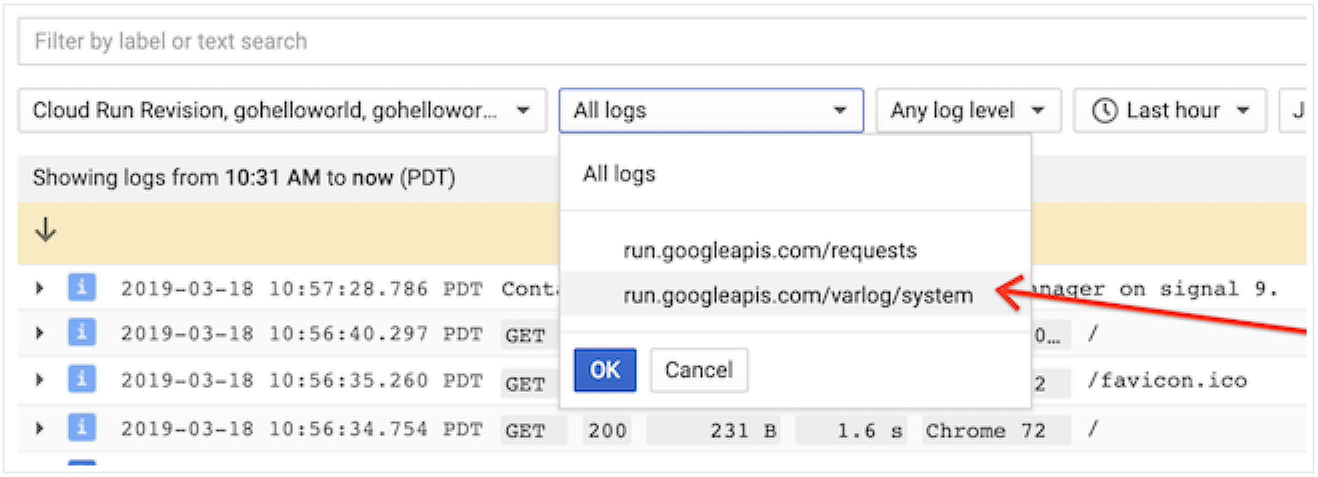
This issue can be caused by one of the following:

- A huge sudden increase in traffic
- A long cold start time (https://cloud.google.com/run/docs/tips#starting_services_quickly)
- A long request processing time
- The service has reached its maximum container instance limit (https://cloud.google.com/run/docs/configuring/max-instances)

Is your issue caused by a limitation in the container sandbox?

If your container runs locally but fails in Cloud Run (fully managed), the Cloud Run container sandbox (https://cloud.google.com/run/docs/reference/container-contract#sandbox) might be responsible for the failure of your container.

In the Stackdriver Logging section of the GCP Console (not in the "Logs" tab of the Cloud Run section), you can look for *"Container Sandbox"* with a "DEBUG" severity in the **`varlog/system`** logs.


Filter by label or text search
Cloud Run Revision, gohelloworld, gohellowor...
All logs
Any log level
Last hour
J
All logs
run.googleapis.com/requests
run.googleapis.com/varlog/system
OK
Cancel

Showing logs from 10:31 AM to now (PDT)

```
2019-03-18 10:57:28.786 PDT Cont                           anager on signal 9.
2019-03-18 10:56:40.297 PDT GET                            0...  /
2019-03-18 10:56:35.260 PDT GET                            2     /favicon.ico
2019-03-18 10:56:34.754 PDT GET  200   231 B   1.6 s  Chrome 72  /
```

For example:

```
Container Sandbox: Unsupported syscall setsockopt(0x3,0x1,0x6,0xc0000753d0,0x4,0x0)
```

If you suspect that these might be responsible for the failure of your container, contact Support (https://cloud.google.com/run/docs/support) and add the log message to the support ticket. The Google Cloud support might ask you to trace system calls made by your service (https://cloud.google.com/run/docs/troubleshooting/tracing-system-calls) to diagnose lower-level system calls that are not surfaced in the Stackdriver logs.