

The Cloud Run port error that started in a Flask import

A September Manabi Kōbō fix shows why the first Python traceback can be more useful than the last deployment message.



By Edwin Rodriguez

Published Oct 14, 2026 · 4 min read

September 2026 retrospective · Story context 2026-09-05



You deploy a Flask application and the platform says it never started listening on the expected port. Naturally, you look at the port. Then the Docker command. Then the health check. All reasonable places to look, although the application may have stopped much earlier.

A September Manabi Kōbō correction provides a clean example. An RSS route referenced a blueprint name that the application didn't define. It also called an article helper that didn't exist under the name used by that route. Python failed while importing the application, before the server had a chance to open its socket.

The port message was the clue at the end of the chain. The Python traceback took us back to the broken step.

Reconstruct the startup path

A typical Flask container starts a WSGI server, imports an application module or factory, registers its routes, and then begins accepting requests. An exception during those imports can interrupt the whole sequence.

In this case, the route used `bp` while the module's actual blueprint was `core_bp`. The article feed also needed to call the catalog helper already used by the application. Fixing the names restored a consistent route definition and data source.

The general lesson travels well: when a platform reports a startup failure, find the earliest application error in the same revision's logs. Later health-check messages can repeat for a while without adding information about the cause.

Reproduce the smallest failing step

Start with an import check in the same dependency environment as the container. Adapt the module and factory name to your project:

```
from app import create_app

application = create_app()
print(application.url_map)
```

Adapt this smoke check to your repository's actual entry point. It helps expose import errors, registration mistakes, and configuration assumptions before a server starts.

Then run the actual container command locally with the expected environment. Verify the port and binding address after the application imports successfully. [Google Cloud's troubleshooting guide](#) specifically calls out local container execution, the configured port, binding to all interfaces, and application logs.

Keep the diagnosis narrow. If the trace ends in a missing Python name, changing a firewall rule won't repair that name. If imports succeed and the server binds only to loopback, the network-facing configuration becomes the next useful place to work.

Check the feature that triggered the failure

Once the app starts, open the RSS route too. It still needs to return a useful feed. Exercise that route after the import correction. Confirm a successful response, a suitable content type, valid dates, and links pointing at the public articles.

Use an empty catalog fixture too. A feed should handle the first day of a new site without crashing because there are no entries. Add a malformed date fixture if the route converts user-editable metadata. Those small cases exercise the route registration and article data handling that caused the trouble.

If a test merely repeats the corrected variable name, it won't catch much. A request through Flask's test client gives the route a chance to reveal the next mismatch.

Keep local publishing files out of the container

The same September cleanup separated local authoring material from the hosted application. That made the deployment boundary easier to inspect.

There is a practical Git detail here: adding a tracked file to `.gitignore` doesn't remove it from the repository. Review the tracked file list and remove only the intended local material from tracking. Container and cloud upload ignore rules have their own scope, so check those build inputs too.

A smaller, deliberate build context makes accidental packaging easier to spot. It also makes a startup investigation less noisy because you know which application code the image contains.

The next time you see the message

Read the first traceback, reproduce the import, start the real server, and request the affected route. Keep a note linking the observed symptom to the verified cause. A five-line diagnostic recipe is useful when the same generic platform message appears months later with a completely different underlying bug.

Sources & further reading

Checked 2026-10-14 · Story context 2026-09-05

1. [Cloud Run startup troubleshooting](#) (2026-10-14)
2. [Flask application factories](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at progretech.com.

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.