Go to Railway

Configure Healthchecks and Restart Policy

Railway provides controls for ensuring deployed services remain healthy.

Configure Healthcheck Path

A Healthcheck can be used to guarantee zero-downtime deployments of your web

services by ensuring the new version is live and able to handle requests.

To configure a healthcheck -

1. Ensure your webserver has an endpoint (e.g. `/health`) that will return an HTTP status code of 200 when the application is live and ready
2. Under your service settings, input your health endpoint. Railway will wait for this endpoint to serve a `200` status code before switching traffic to your new endpoint

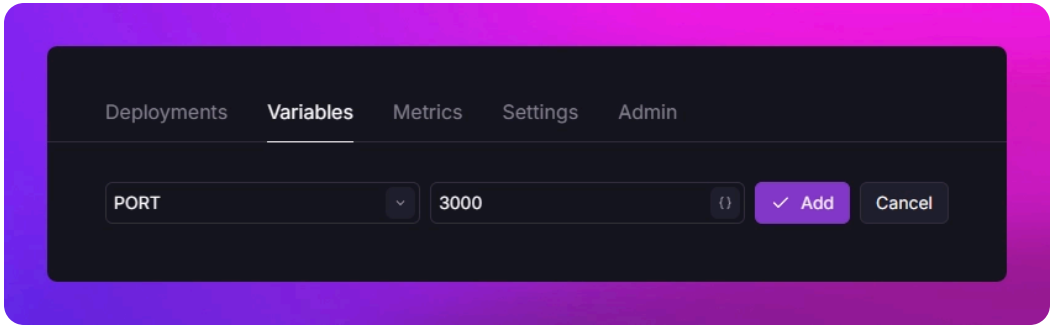
Note: Railway does not monitor the healthcheck endpoint after the deployment has gone live.

Configure Healthcheck Port

Railway will inject a `PORT` environment variable that your application should **listen on**.

This variable's value is also used when performing health checks on your deployments.

If your application doesn't listen on the `PORT` variable, possibly due to using **target ports**, you can manually set a `PORT` **variable** to inform Railway of the port to use for health checks.



Docs

Search ⌘K

Home

Quick Start

Railway Metal

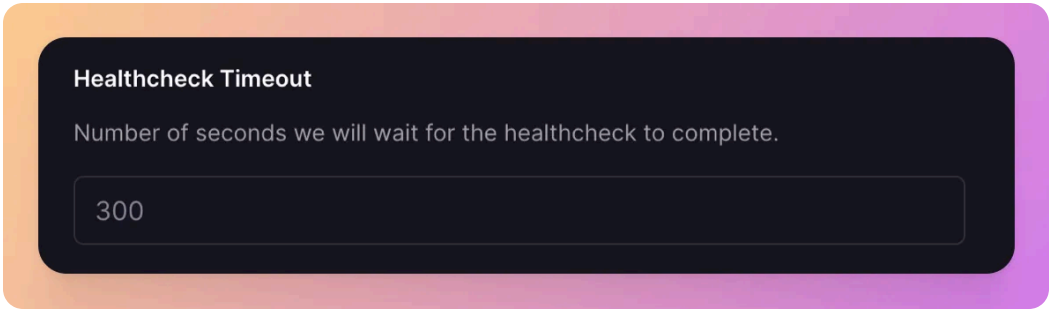
Overview

About Railway

Not listening on the `PORT` variable or omitting it when using target ports can result in your health check returning a `service unavailable` error.

Healthcheck Timeout

The default timeout on healthchecks is by default 300 seconds (5 minutes) - if your application fails to serve a `200` status code during this allotted time, the deploy will be marked as failed.



To increase the timeout, change the number of seconds on the service settings page, or with a `RAILWAY_HEALTHCHECK_TIMEOUT_SEC` service variable.

Services with Attached Volumes

To prevent data corruption, we prevent multiple deployments from being active and mounted to the same service. This means that there will be a small amount of downtime when re-deploying a service that has a

volume attached, even if there is a healthcheck endpoint configured.

Healthcheck Hostname

Railway uses the hostname `healthcheck.railway.app` when performing healthchecks on your service. This is the domain from which the healthcheck requests will originate.

For applications that restrict incoming traffic based on the hostname, you'll need to add `healthcheck.railway.app` to your list of allowed hosts. This ensures that your application will accept healthcheck requests from Railway.

If your application does not permit requests from that hostname, you may encounter errors during the healthcheck process, such as "failed with service unavailable" or "failed with status 400".

Continuous Healthchecks

The healthcheck endpoint is currently ***not used for continuous monitoring*** as it is only called at the start of the deployment, to ensure it is healthy prior to routing traffic to it.

If you are looking for a quick way to setup continuous monitoring of your service(s), check out the **Uptime Kuma template** in our template marketplace.

Restart Policy

The restart policy dictates what action Railway should take if a deployed service stops or otherwise becomes unhealthy, e.g. exits with a non-zero exit code.

Note: For services with multiple replicas, a restart will only affect the replica that crashed, while the other replica(s) continue handling the workload until the restarted replica is back online.

To configure a restart policy, go to the Service settings, set a restart policy of `Never`, `Always`, or `On-Failure` with an optional maximum number of restarts.

- `Always`: This means Railway will automatically restart your service every time it stops, regardless of the reason. It's useful for services that are designed to run continuously and should always be available.

- `On-Failure`: Railway will only restart your service if it stops due to an error (e.g., crashes, exits with a non-zero code). This is a good option if you want to avoid unnecessary restarts when the service stops intentionally.

- `Never`: Railway will never automatically restart your service, even if it crashes. This is typically used for one-off tasks or scheduled jobs where you don't want the

service to keep running after it completes.

Not finding the answer you need?

Start a Discussion

Prev

Optimize Performance

Next

Monorepo

Edit this file on GitHub