Developing your service

This page describes a few things you need to know to get started in developing a service for Cloud Run.

Code requirements

You must meet the following requirements when you develop a service:

- The service must listen for requests on the port defined by the `PORT` environment variable.

- The service must be *stateless*. It cannot rely on a persistent local state.

- The service must not perform background activities outside the scope of request handling.

You can find more details about these constraints in the Container Runtime Contract (https://cloud.google.com/run/docs/reference/container-contract).

Programming language support

Cloud Run allows you to write code in the programming language of your choice.

The build and deploy quickstart (https://cloud.google.com/run/docs/quickstarts/build-and-deploy#writing) page provides samples in many popular languages.

Using a web server

You can use a web server to listen on the required port, and to process and route incoming requests. For example, Node.js developers can use Express.js (https://expressjs.com/), Python

developers can use <u>Flask</u> (http://flask.pocoo.org/), Ruby developers can use <u>Sinatra</u> (http://sinatrarb.com/), and so forth.

Containerizing your code with a Dockerfile

To deploy to Cloud Run, you need to provide a *container image*. A container image is a packaging format that includes your code, its packages, any needed binary dependencies, the operating system to use, and anything else needed to run your service.

A file named <u>Dockerfile</u> (https://docs.docker.com/engine/reference/builder/) is commonly used to declare how to build the container image. You can find examples of Dockerfiles for popular languages in the <u>build and deploy quickstart</u> (https://cloud.google.com/run/docs/quickstarts/build-and-deploy#containerizing).

Dockerfiles very often start from a base image (e.g. `FROM golang:1.11`). You can find base images maintained by OS and language authors on <u>Docker Hub</u> (https://hub.docker.com/). If needed, Google Cloud Container Registry provides a <u>Docker Hub mirror</u> (https://cloud.google.com/container-registry/docs/using-dockerhub-mirroring). You can also find base images managed by Google in the <u>Cloud Marketplace</u> (https://console.cloud.google.com/marketplace/browse?filter=solution-type:container&filter=category:os).

If you bring your own binaries, make sure they are compiled for Linux ABI x86_64.

These resources provide further information on Dockerfiles:

- Learn the syntax via the <u>Dockerfile Reference</u> (https://docs.docker.com/engine/reference/builder) and how they fit together via the tips in <u>Best practices for writing Dockerfiles</u> (https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

- Read <u>Best Practices for Building Containers</u> (https://cloud.google.com/solutions/best-practices-for-building-containers) for further refinements

What's next

- Once you have your service code and Dockerfile, you should build a container image (https://cloud.google.com/run/docs/building/containers) then continue iterating in local testing (https://cloud.google.com/run/docs/testing/local)

- If you are migrating an existing web application, see Migrating Your Service to Cloud Run (https://cloud.google.com/run/docs/migrating).

- For best practices for designing, implementing, testing, and deploying a Cloud Run service, see the Development tips (https://cloud.google.com/run/docs/tips)