

[Serverless Computing](https://cloud.google.com/products/serverless/). (<https://cloud.google.com/products/serverless/>)

[Cloud Run: Serverless Computing](https://cloud.google.com/run/). (<https://cloud.google.com/run/>)

[Documentation](https://cloud.google.com/run/docs/). (<https://cloud.google.com/run/docs/>) [Guides](#)

Quickstart: Build and Deploy

This page shows you how to create a simple Hello World application, package it into a container image, upload the container image to Container Registry, and then deploy the container image to Cloud Run. The sample is shown in several languages, but note that you can use other languages in addition to the ones shown.

You can also follow this quickstart with a demo account [on Qwiklabs](https://www.qwiklabs.com/focuses/5162?parent=catalog) (<https://www.qwiklabs.com/focuses/5162?parent=catalog>).

Before you begin

1. [Sign in](https://accounts.google.com/Login) (<https://accounts.google.com/Login>) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (<https://accounts.google.com/SignUp>).

2. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

[GO TO THE PROJECT SELECTOR PAGE](https://console.cloud.google.com/projectselector) ([HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECT](https://console.cloud.google.com/projectselector))

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](https://cloud.google.com/billing/docs/how-to/modify-project) (<https://cloud.google.com/billing/docs/how-to/modify-project>).

4. Enable the Cloud Build and Cloud Run APIs.

[ENABLE THE APIS](https://console.cloud.google.com/flows/enableapi?APIID=CLOUDBUILD) ([HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=CLOUDBUILD](https://console.cloud.google.com/flows/enableapi?APIID=CLOUDBUILD))

5. [Install and initialize the Cloud SDK](https://cloud.google.com/sdk/docs/) (<https://cloud.google.com/sdk/docs/>).

6. Update components:

```
gcloud components update
```



Writing the sample application

For instructions on creating a sample hello world application that runs on Cloud Run, click the tab for your language:

GO NODE.JS PYTHON MORE ▾

1. Create a new directory named `helloworld-go` and change directory into it:

```
mkdir helloworld-go
cd helloworld-go
```
2. Initialize a `go.mod` file to declare the `go module` (<https://blog.golang.org/using-go-modules>):

```
go mod init helloworld-go
```
3. Create a new file named `helloworld.go` and paste the following code into it:

```
docs/serving/samples/hello-world/helloworld-go/helloworld.go
(https://github.com/knative/docs/blob/187ed5ae2388cf2e8efbc01b99e253ca9e4320e9/docs/serving/hello-world/helloworld-go/helloworld.go)
```

```
package main

import (
    "fmt"
    "log"
    "net/http"
    "os"
)

func handler(w http.ResponseWriter, r *http.Request) {
    log.Print("Hello world received a request.")
    target := os.Getenv("TARGET")
    if target == "" {
        target = "World"
    }
    fmt.Fprintf(w, "Hello %s!\n", target)
}
```

```
func main() {
    log.Print("Hello world sample started.")

    http.HandleFunc("/", handler)

    port := os.Getenv("PORT")
    if port == "" {
        port = "8080"
    }

    log.Fatal(http.ListenAndServe(fmt.Sprintf(":%s", port), nil))
}
```

This code creates a basic web server that listens on the port defined by the [PORT environment variable](https://cloud.google.com/run/docs/reference/container-contract#port) (<https://cloud.google.com/run/docs/reference/container-contract#port>).

Your app is finished and ready to be containerized and uploaded to Container Registry.

Containerizing an app and uploading it to Container Registry

To containerize the sample app, create a new file named `Dockerfile` in the same directory as the source files, and copy the following content:

[GO](#)[NODE.JS](#)[PYTHON](#)[MORE](#) ▾

[docs/serving/samples/hello-world/helloworld-go/Dockerfile](https://github.com/knative/docs/blob/187ed5ae2388cf2e8efbc01b99e253ca9e4320e9/docs/serving/world/helloworld-go/Dockerfile)
(<https://github.com/knative/docs/blob/187ed5ae2388cf2e8efbc01b99e253ca9e4320e9/docs/serving/world/helloworld-go/Dockerfile>)

```
# Use the official Golang image to create a build artifact.
# This is based on Debian and sets the GOPATH to /go.
# https://hub.docker.com/_/golang
FROM golang:1.13 as builder

# Create and change to the app directory.
WORKDIR /app

# Retrieve application dependencies.
# This allows the container build to reuse cached dependencies.
COPY go.* ./
RUN go mod download
```

```
# Copy local code to the container image.
COPY . ./

# Build the binary.
RUN CGO_ENABLED=0 GOOS=linux go build -mod=readonly -v -o server

# Use the official Alpine image for a lean production container.
# https://hub.docker.com/_/alpine
# https://docs.docker.com/develop/develop-images/multistage-build/#use-multi-stage
FROM alpine:3
RUN apk add --no-cache ca-certificates

# Copy the binary to the production image from the builder stage.
COPY --from=builder /app/server /server

# Run the web service on container startup.
CMD ["/server"]
```

Build your container image using Cloud Build, by running the following command from the directory containing the Dockerfile:

```
gcloud builds submit --tag gcr.io/PROJECT-ID/helloworld
```

where **PROJECT-ID** is your GCP project ID. You can get it by running `gcloud config get-value project`.

Upon success, you will see a SUCCESS message containing the image name (`gcr.io/PROJECT-ID/helloworld`). The image is stored in Container Registry and can be re-used if desired.

Deploying to Cloud Run

To deploy the container image:

1. Deploy using the following command:

```
gcloud run deploy --image gcr.io/PROJECT-ID/helloworld --platform managed
```

Replace **PROJECT-ID** with your GCP project ID. You can view your project ID by running the command `gcloud config get-value project`.

- a. You will be prompted for the service name: press Enter to accept the default name, `helloworld`.
- b. You will be prompted for region: select the region ([#follow-cloud-run](#)) of your choice, for example `us-central1`.
- c. You will be prompted to **allow unauthenticated invocations**: respond `y`.

Then wait a few moments until the deployment is complete. On success, the command line displays the service URL.

2. Visit your deployed container by opening the service URL in a web browser.

Congratulations! You have just deployed an application packaged in a container image to Cloud Run. Cloud Run automatically and horizontally scales your container image to handle the received requests, then scales down when demand decreases. You only pay for the CPU, memory, and networking consumed during request handling.

Clean up

Removing your test project

While Cloud Run does not charge when the service is not in use, you might still be charged for storing the container image in Container Registry.

(<https://cloud.google.com/container-registry/pricing>). You can delete your image (https://cloud.google.com/container-registry/docs/managing#deleting_images) or delete your Google Cloud project to avoid incurring charges. Deleting your Google Cloud project stops billing for all the resources used within that project.




Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an **appspot.com** URL, delete selected resources inside the project instead of deleting the whole project.

1. In the Cloud Console, go to the **Manage resources** page.

GO TO THE MANAGE RESOURCES PAGE ([HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PRO](https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete**  .

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

What's next

For more information on building a container from code source and pushing to Container Registry, see:

- [Developing Cloud Run services](https://cloud.google.com/run/docs/developing) (<https://cloud.google.com/run/docs/developing>)
- [Building Containers](https://cloud.google.com/run/docs/building/containers) (<https://cloud.google.com/run/docs/building/containers>)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated December 13, 2019.