Serverless Computing (https://cloud.google.com/products/serverless/)
Cloud Run: Serverless Computing (https://cloud.google.com/run/)
Documentation (https://cloud.google.com/run/docs/) Guides

Deploying container images

This page describes how to deploy new services and new revisions to Cloud Run.

Permissions required to deploy

In order to deploy to Cloud Run (fully managed), you must have the *Owner* or *Editor* role, or both the *Cloud Run Admin* and *Service Account User* roles, or any custom role that includes this specific list of permissions (https://cloud.google.com/run/docs/reference/iam/roles#additional-configuration).

In order to deploy a service to Cloud Run for Anthos on Google Cloud, you must have the *Owner*, *Editor*, *GKE Admin*, or *GKE Developer* role. You also need permissions to create, update, and delete on the apiGroup `serving.knative.dev` and kind `Service`.

Images you can deploy

There is no size limit that applies to the container image you can deploy.

For Cloud Run (fully managed), you can deploy container images stored in Container Registry (https://cloud.google.com/container-registry/). You can use only the following types of container images:

- Container images stored in the same project as the one you want to deploy to.
- Container images from other Google Cloud projects (provided that the correct IAM permissions are set (#deploying_images_from_other_projects)).
- Public container images (https://cloud.google.com/container-registry/docs/access-control#serving_images_publicly).

For Cloud Run for Anthos, you can use containers from any container registry, such as Docker Hub (https://hub.docker.com/). For information on deploying private images from registries

different from Container Registry, see Deploying private container images from other container registries (#private-other-registries).

Deploying a new service

You can specify a container image with a tag (e.g. `gcr.io/my-project/my-image:latest`) or with an exact digest (e.g. `gcr.io/my-project/my-image@sha256:41f34ab970ee...`).

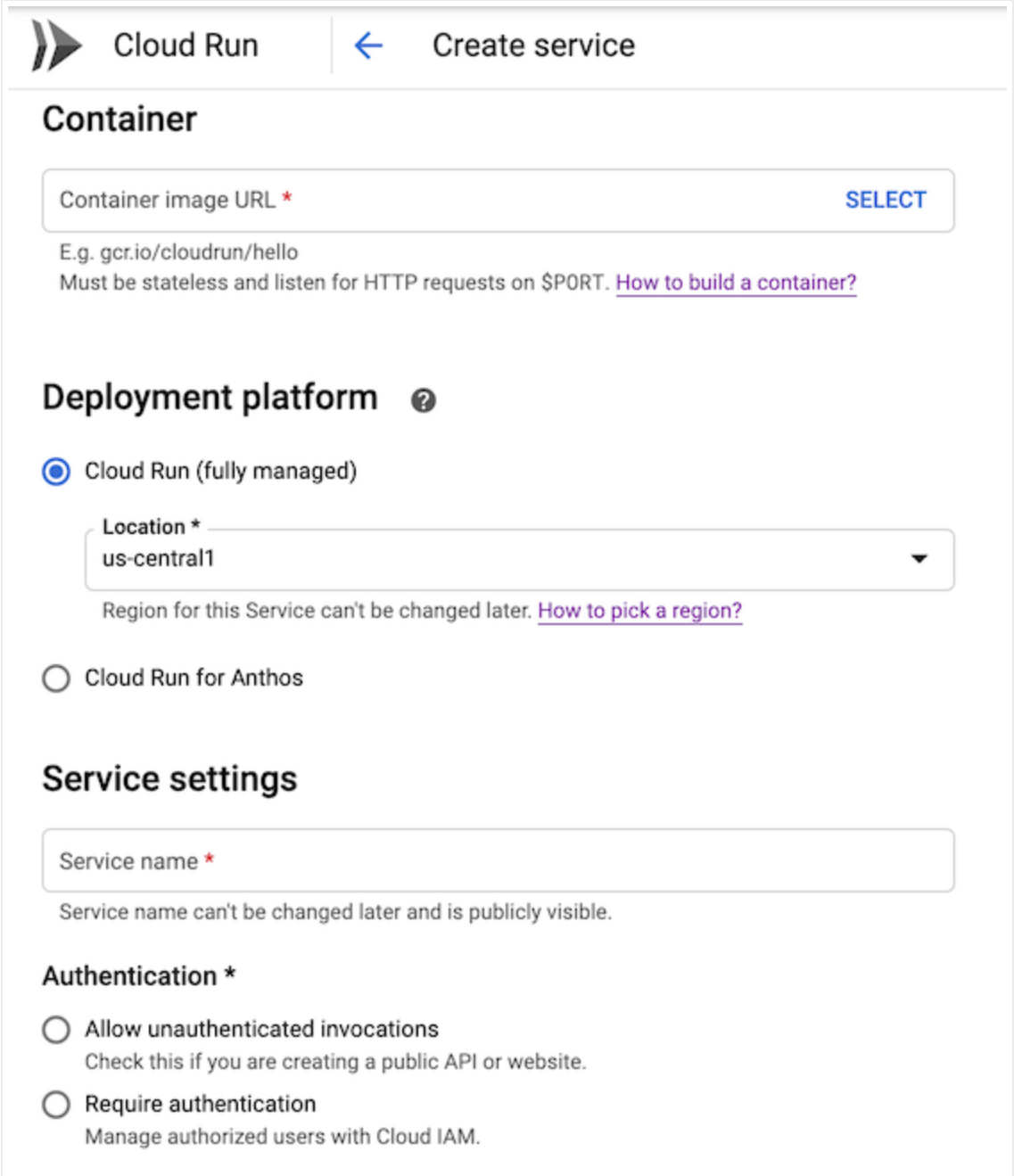
Deploying to a service for the first time creates its first revision. Note that revisions are immutable. If you deploy from a container image tag, it will be resolved to a digest and the revision will always serve this particular digest.

You can deploy a container using the Cloud Console or the `gcloud` command line. Click the tab for instructions using the tool of your choice.

CONSOLE COMMAND LINE

To deploy a container image:

1. GO TO CLOUD RUN (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/RUN)
2. Click **Create service** to display the *Create service* page.



In the form,

a. Under *Source*, supply the URL of an image in Container Registry, for example: `gcr.io/cloudrun/hello`

b. Select the Cloud Run platform you are deploying to:

- Cloud Run (fully managed) to deploy to a fully managed environment.
- Cloud Run for Anthos on Google Cloud to deploy to to a GKE or GKE On-Prem cluster with Cloud Run for Anthos on Google Cloud enabled.

c. If deploying to Cloud Run (fully managed):

i. Select the region (#before-you-begin) where you want your service located.

ii. Under *Authentication*,

- If you are creating a public API or website, select **Allow unauthenticated invocations**. Selecting this assigns the IAM Invoker role to the special identifier `allUser`. You can use IAM to edit this setting (https://cloud.google.com/run/docs/securing/authenticating#service-to-service) later after you create the service.
- If you want a secure service protected by authentication, select **Require authentication**.

d. If deploying to Cloud Run for Anthos:

i. Select one of the available GKE clusters for your service.

ii. Under *Connectivity*:

- Select **Internal** if you want to restrict access only to other Cloud Run for Anthos on Google Cloud services or services in your cluster that use istio.
- Select **External** to allow external access to your service

Note that you can change the connectivity option at any time, as described in Changing service connectivity settings (https://cloud.google.com/run/docs/managing/services#connectivity).

e. Confirm or update the suggested service name. Service names must be unique per region and project or per cluster. A service name cannot be changed later and is publicly visible when using Cloud Run (fully managed).

f. Optionally, set:

- environment variables (https://cloud.google.com/run/docs/configuring/environment-variables),
- concurrency (https://cloud.google.com/run/docs/configuring/concurrency), and
- memory limits (https://cloud.google.com/run/docs/configuring/memory-limits).
- request timeout (https://cloud.google.com/run/docs/configuring/request-timeout).
- Cloud SQL connections (https://cloud.google.com/run/docs/configuring/connect-cloudsql), if you are deploying to Cloud Run (fully managed).

3. Click **Create** to deploy the image to Cloud Run and wait for the deployment to finish.
4. Click the displayed URL link to open the unique and stable endpoint of your deployed service.

Persistence of service URLs

Each service has a unique and permanent URL that will not change over time as you deploy new revisions to it.

Deploying a new revision of an existing service

You can deploy a new revision using the Cloud Console or the `gcloud` command line.

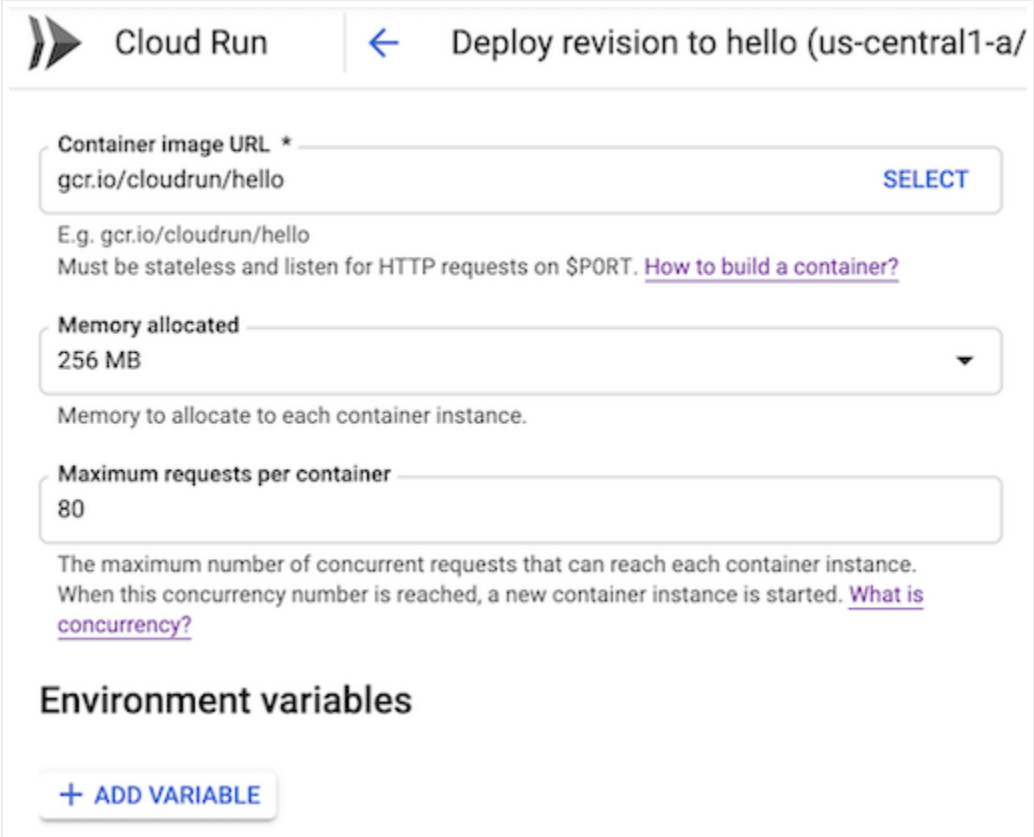
Note that changing the memory limit, environment variables, or concurrency also results in the creation of a revision, even if there is no change to the container image. Each revision created is immutable.

Click the tab for instructions using the tool of your choice.

CONSOLE COMMAND LINE

To deploy a new revision of an existing service:

1. **GO TO CLOUD RUN** (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/RUN)
2. Locate the service you want to update in the services list, and click on it to open the details of that service.
3. Click **DEPLOY NEW REVISION**. This displays the revision deployment form:



4. If needed, supply the URL to the new container image you want to deploy.
5. If needed, set:
    - environment variables (https://cloud.google.com/run/docs/configuring/environment-variables),
    - concurrency (https://cloud.google.com/run/docs/configuring/concurrency), and
    - memory limits (https://cloud.google.com/run/docs/configuring/memory-limits).
    - Cloud SQL connections (https://cloud.google.com/run/docs/configuring/connect-cloudsql), if you are deploying to Cloud Run (fully managed).
6. To send all traffic to the new revision, check the checkbox labelled *Serve this revision immediately*. To gradually roll out a new revision, uncheck that checkbox: this will result in a deployment where no traffic is sent to the new revision--follow the instructions for gradual rollouts (https://cloud.google.com/run/docs/rollouts-rollbacks-traffic-migration#gradual) after you deploy.
7. Click **DEPLOY** and wait for the deployment to finish.

Deploying images from other Google Cloud projects

You can deploy container images from other Google Cloud projects if you set the correct IAM permissions:

1. In the Cloud Console console, open the project for your Cloud Run service.
2. GO TO THE IAM PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/IAM)
3. If you deploy to:
   - Cloud Run (fully managed), copy the email of the Cloud Run service agent. It has the suffix **@serverless-robot-prod.iam.gserviceaccount.com**
   - Cloud Run for Anthos on Google Cloud, copy the email of the Compute Engine default service account (https://cloud.google.com/compute/docs/access/service-accounts#compute_engine_default_service_account). It has the suffix **@developer.gserviceaccount.com**
   - Cloud Run for Anthos deployed on VMware, create a Google Cloud service account and download the credentials (https://cloud.google.com/iam/docs/creating-managing-service-account-keys#creating_service_account_keys). Add these credentials as the default imagePullSecrets of the Kubernetes Service Account (https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/#add-imagepullsecrets-to-a-service-account).
4. Open the project that owns the container registry you want to use.
5. GO TO THE IAM PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/IAM)
6. Click **Add** to add a new member.
7. In the **New members** text box, paste in the email of the service account that you copied earlier.
8. In the *Select a role* dropdown list, select the role **Storage -> Storage Object Viewer**.
9. Deploy the container image (#deploying_a_new_service) to the project that contains your Cloud Run service.

★ **Note:** For stronger security, you can limit grant access to only the Cloud Storage bucket that contains your container images

(https://cloud.google.com/container-registry/docs/access-control#granting_users_and_other_projects_access_to_a_registry)
.

Deploying private container images from other container registries

This section describes setting up correct permissions to deploy container images from an arbitrary private registry to Cloud Run for Anthos. A private container registry requires credentials to access the container image. Note that you do not need to follow these steps to deploy private container images from Container Registry in the same project as your cluster.

To be able to deploy a private container image, you must create an `imagePullSecret` type Kubernetes secret and associate it with a service account:

1. Create an `imagePullSecret` secret called `container-registry`:

```
kubectl create secret container-registry \
--docker-server=DOCKER_REGISTRY_SERVER \
--docker-email=REGISTRY_EMAIL \
--docker-username=REGISTRY_USER \
--docker-password=REGISTRY_PASSWORD
```

   - Replace ***DOCKER_REGISTRY_SERVER*** with your private registry FQDN (ex: https://gcr.io/ (https://gcr.io/) for Container Registry or https://index.docker.io/v1/ (https://index.docker.io/v1/) for DockerHub).
   - Replace ***REGISTRY_EMAIL*** with your email.
   - Replace ***REGISTRY_USER*** with your container registry username.

     If you're using Container Registry and would like to store and pull long-lived credentials instead of passing short-lived access tokens, see Authentication methods: JSON key file (https://cloud.google.com/container-registry/docs/advanced-authentication#json_key_file).
   - Replace ***REGISTRY_PASSWORD*** with your container registry password.

2. Open your default service account:

```
kubectl edit serviceaccount default --namespace default
```

Every namespace (https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/) in your Kubernetes cluster has a default service account called `default`. This default service account is used to pull your container image unless otherwise specified in your Cloud Run for Anthos service's Revision Spec (https://cloud.google.com/run/docs/reference/rest/v1/RevisionSpec).

3. Add the newly created `imagePullSecret` secret to your default service account:

```
imagePullSecrets:
- name: container-registry
```

Your service account should now look like this:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: default
  namespace: default
  ...
secrets:
- name: default-token-zd84v
# The secret we just created:
imagePullSecrets:
- name: container-registry
```

Now, any new pods created in the current `default` namespace will have the `imagePullSecret` secret defined.

What's next

After you deploy a new service, you can do the following:

- Gradual rollouts, rollback revisions, traffic migration (https://cloud.google.com/run/docs/rollouts-rollbacks-traffic-migration)
- View service logs (https://cloud.google.com/run/docs/logging)
- Monitor service performances (https://cloud.google.com/run/docs/monitoring)
- Set memory limits (https://cloud.google.com/run/docs/configuring/memory-limits)

- Set environment variables (https://cloud.google.com/run/docs/configuring/environment-variables)
- Change service concurrency (https://cloud.google.com/run/docs/configuring/concurrency)
- Manage the service (https://cloud.google.com/run/docs/managing/services)
- Manage service revisions (https://cloud.google.com/run/docs/managing/revisions)

You can automate the builds and deployments of your Cloud Run services using Cloud Build Triggers:

- Set up Continuous Deployment (https://cloud.google.com/run/docs/continuous-deployment)

Last updated January 16, 2020.