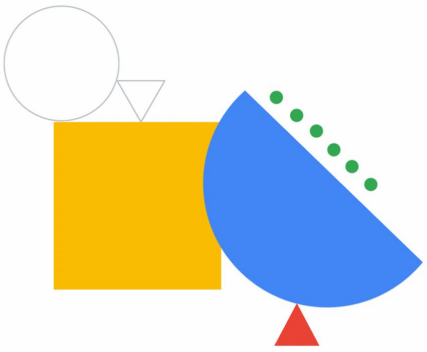
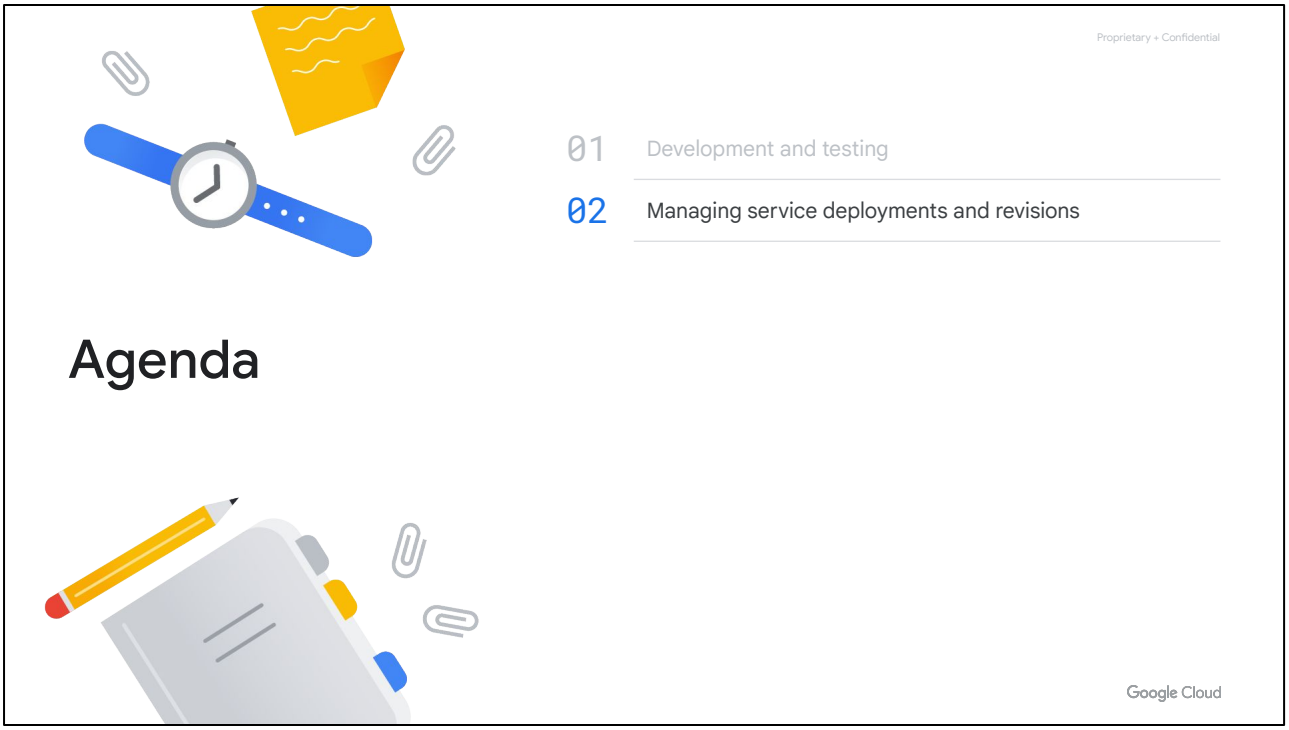
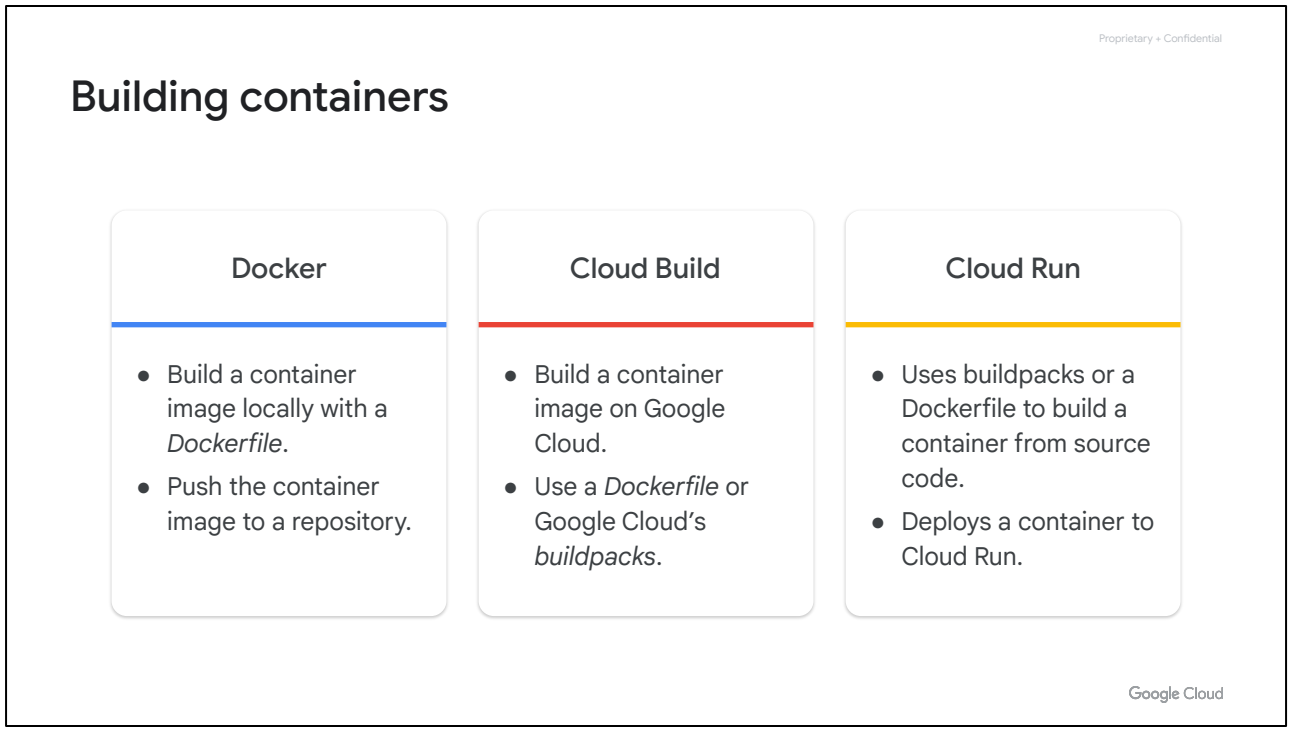
Google Cloud

Application Development, Testing, and Integration





Let’s now discuss how you can manage your service deployments and revisions on Cloud Run.



Before we discuss Cloud Run deployment, let's talk about how you can build your containerized application.

We discussed how you can build containers with Docker and other CI/CD tools such as Cloud Build in the course on *Deploying Containerized Applications on Google Cloud.*

Here's a brief review:

To build your application into a container image with Docker or with Cloud Build, you can use a Dockerfile. To build a container locally, you can install Docker and use the `docker build` command to build your container image. To push the local container image to an image repository, use the `docker push` command.

You can also build your container image on Google Cloud with Cloud Build. To use Cloud Build, run the `gcloud builds submit` command from the gcloud CLI.

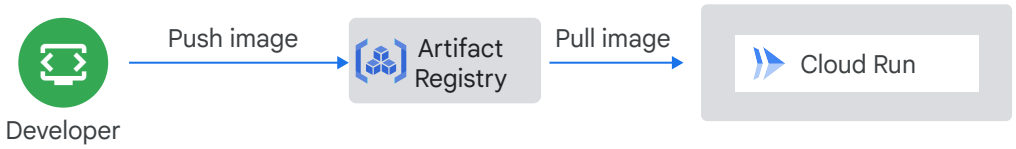
To build your container image from source code without creating a Dockerfile, you can use Buildpacks. Google provides a set of CNCF-compatible Buildpacks that build source code into container images designed to run on Google Cloud container platforms, including Cloud Run. To build a container image with Google Cloud's buildpacks using Cloud Build, run the `gcloud builds submit` command with the `pack` flag.

Buildpacks are built into Cloud Run to enable a source-based deployment workflow. The gcloud run deploy command with the source flag builds your application source

code with a Dockerfile (if one is present), or with Google Cloud's buildpacks. The resulting container image is also uploaded to an image repository and deployed to Cloud Run.

You can also build your container image locally with Google Cloud's buildpacks using the pack command.

Deploying containers to Cloud Run



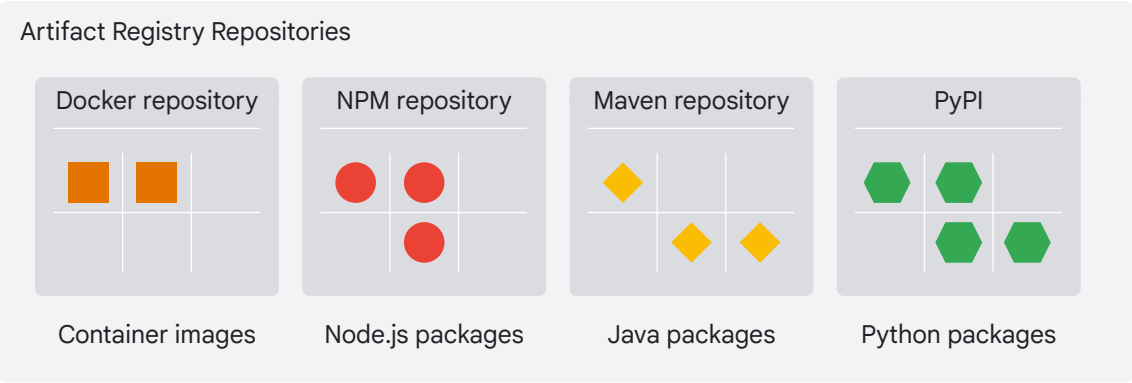
Before a container is deployed to Cloud Run, the container image must be stored in a repository that Cloud Run can access.

You can use container images that are stored in Artifact Registry or Docker Hub. You can also use container images from other public or private registries (like JFrog Artifactory, Nexus, or GitHub Container Registry), by setting up an Artifact Registry remote repository. Google recommends the use of Artifact Registry.

You can use container images that are stored in the same project as the one you are creating the job or service in, or from other Google Cloud projects, if the correct IAM permissions are set. To view more details on how to set these permissions, view the documentation.

If you usually host your container images somewhere else such as an unsupported public or private container registry, remember that with Cloud Run you'll need to push them to Artifact Registry first. You can achieve this with the `docker push` command.

Deploying containers to Cloud Run - Artifact Registry



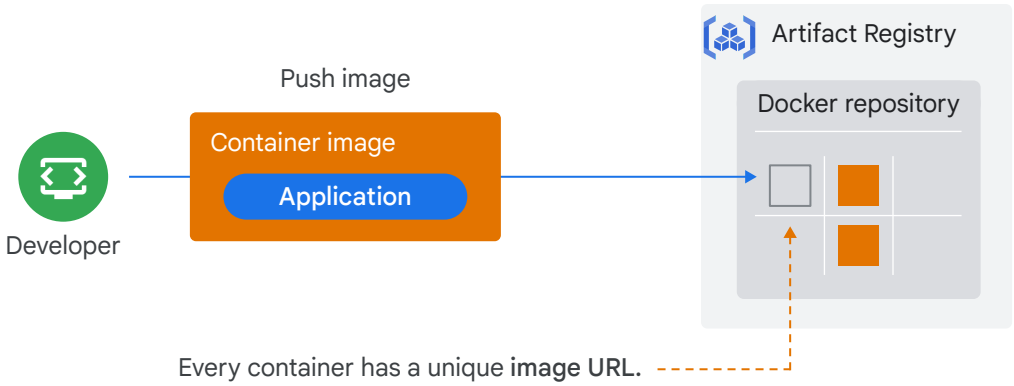
Artifact Registry is a universal package manager service in Google Cloud that is used to store and manage software artifacts in private repositories, including container images, and software packages.

It's the recommended container registry for Google Cloud.

Artifact Registry integrates with Cloud Build to store the packages and container images from your builds.

To host your container images, you create a "Docker repository" that Cloud Run can pull container images from.

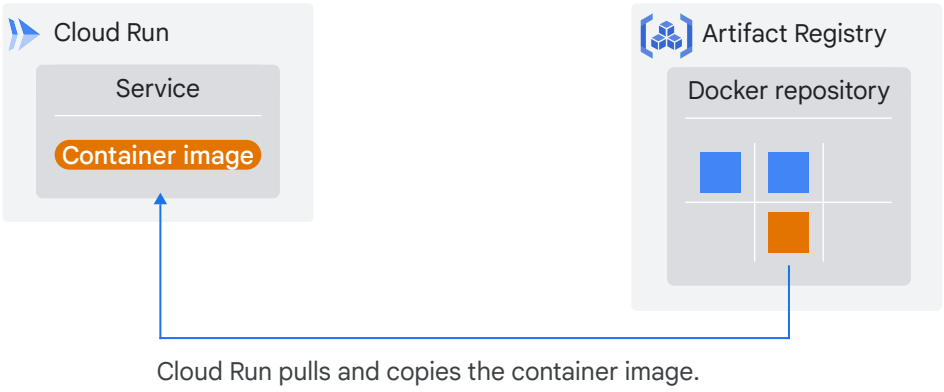
Pushing container images to Artifact Registry



When you're ready to deploy your container image to Cloud Run, you begin by "pushing" (that's a term used for uploading) the image to a Docker repository in Artifact Registry.

Your container image will have a unique URL in the repository, for example, *us-central1-docker.pkg.dev/${PROJECT_ID}/my-repo/my-image,* which you can use when you deploy the image to Cloud Run.

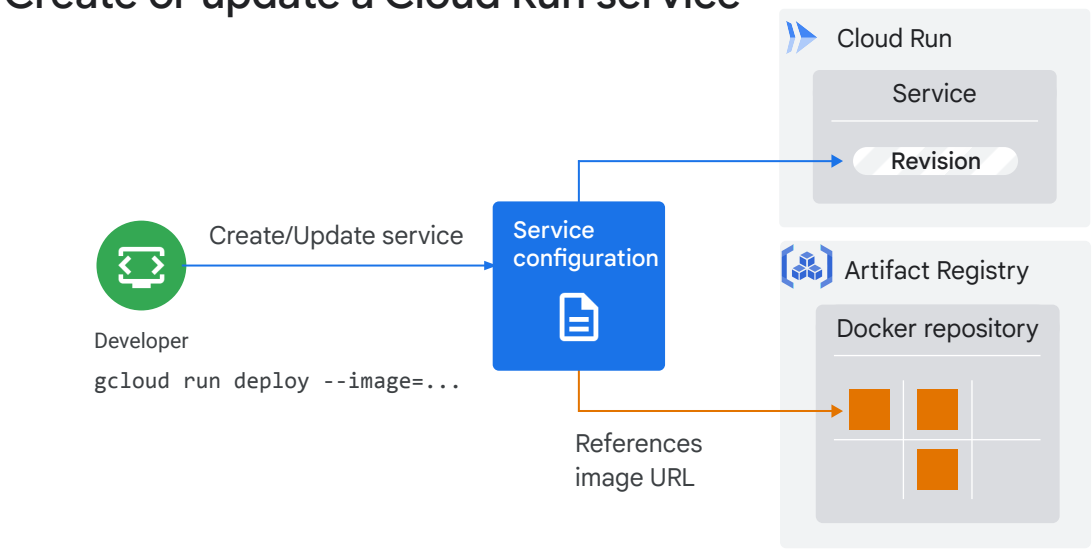
Pulling container images from Artifact Registry



After your container image has been pushed into the Docker repository, you can deploy it to a service on Cloud Run.
This means you hand the container image *URL* to Cloud Run, which then pulls the image from Artifact Registry.

To ensure that containers on Cloud Run start reliably and quickly, Cloud Run copies and stores the container image locally. The internal container storage is fast, which ensures that your image size does not impact your container startup time. Large images load as quickly as small ones.

Because Cloud Run copies the image, it won't be an issue if you accidentally delete a deployed container image from Artifact Registry.
The copy ensures that your Cloud Run service will continue to work.



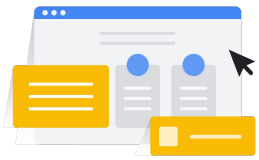
To deploy a container image to Cloud Run, use the Google Cloud console or gcloud CLI to create or update an existing service, and provide the container image URL.

When you deploy a container image for the first time, Cloud Run creates a *service* and its first *revision*. There is only one container image per service.

To be able to deploy, you must have one of Owner, Editor, or both the Cloud Run Admin and Service Account user roles. You can also have a custom role that includes the necessary permissions.

Deploying a new service revision

1. Modify your code.
2. Build and package a container image.
3. Push the image to Artifact Registry.
4. Deploy to the service.
5. Wait for Cloud Run to deploy your changes.



Each revision of a Cloud Run service is immutable. To update your application on Cloud Run, you generally follow these steps:

1. Modify your application source code.
2. Build and package your application into a container image.
3. Push the container image to Artifact Registry.
4. Redeploy the container image to the Cloud Run service.

When you re-deploy your container image to an existing service, a new revision is automatically created.

You can deploy a new revision using the Google Cloud console, the gcloud CLI, a YAML configuration file, or with Terraform.

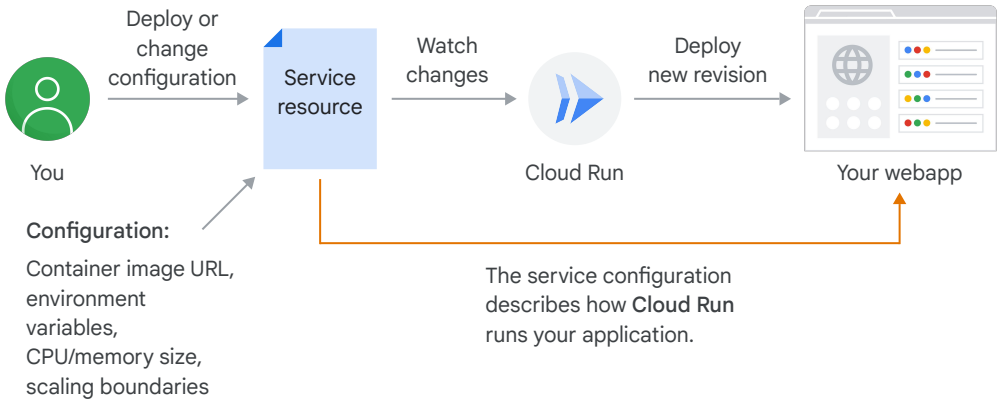
Service configuration

1. Container image URL
2. Container entrypoint and arguments
3. Secrets and Environment variables
4. Request timeout
5. Concurrency
6. CPU/memory limits
7. Scaling boundaries
8. Google Cloud configuration (service account, connectors)

Changing any configuration settings of your Cloud Run service results in the creation of a new revision, even if there is no change to the container image itself.

Subsequent service revisions will also automatically get these configuration settings unless you make explicit updates to change them.

Updating a service



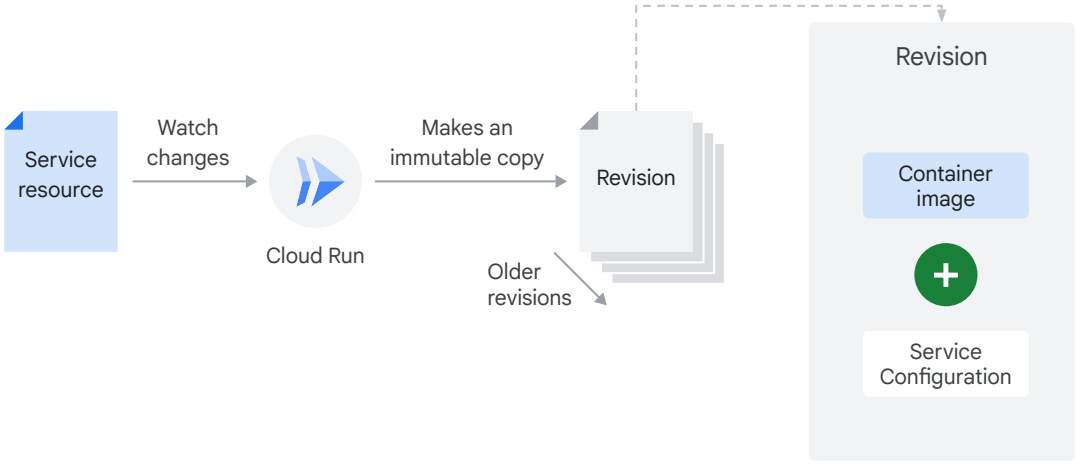
The service resource includes configuration that describes how Cloud Run runs your application.

After you make changes to your application or its service configuration, and deploy it, Cloud Run creates a new revision of your service.

You can control sending all traffic to the new revision immediately when the new revision is healthy.

You can also perform a gradual rollout by controlling the percentage of requests sent to the new revision.

A service revision is immutable



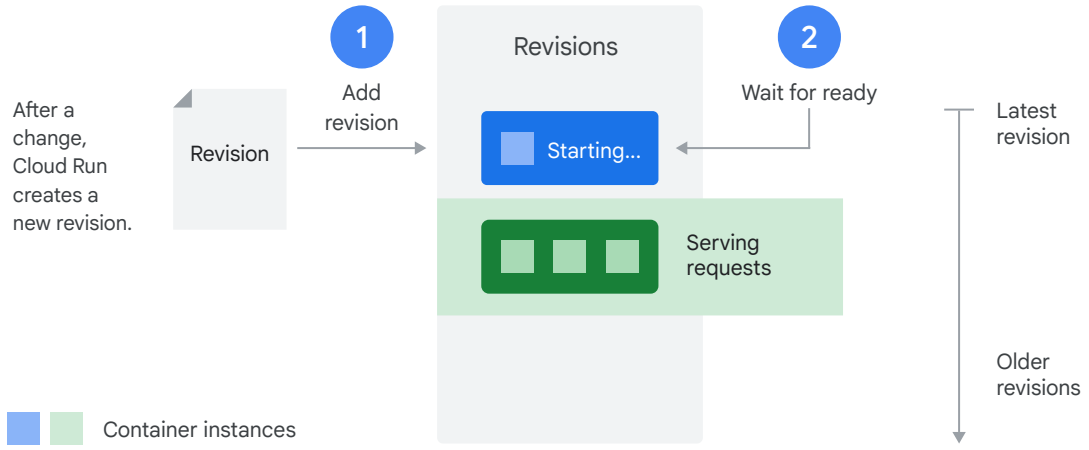
Cloud Run deploys your application after every change that you make to the service resource.

At the same time, it also makes an immutable copy of the service resource, called a revision. “Immutable” means that you can’t make changes to a revision.

You can only add new revisions to make further updates.

A revision is an immutable copy of your container image and service configuration.

Serving traffic

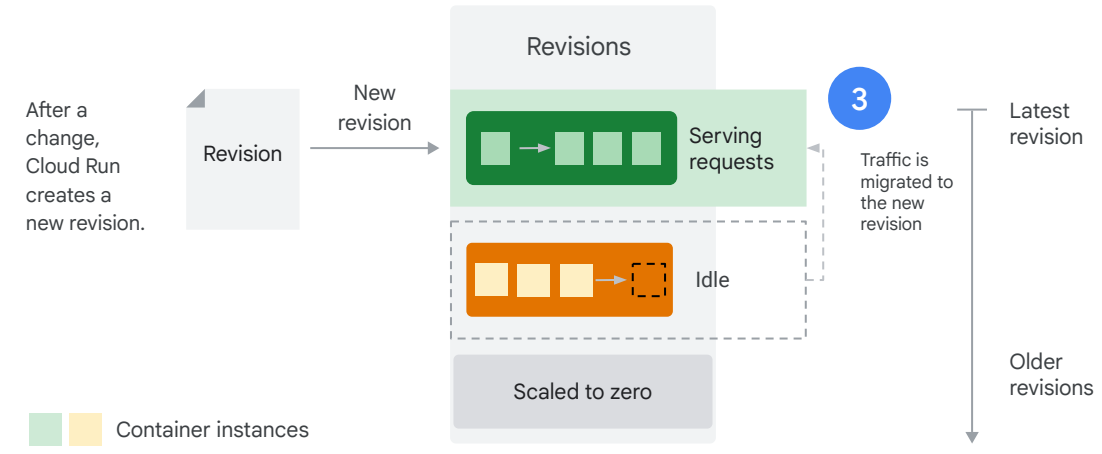


After a new service revision is created, Cloud Run will first scale up the new revision to match the capacity of the current revision.

Cloud Run waits for the container instances in that revision to finish starting up.

While this is happening, the container instances in the current revision still serves any request traffic to the service.

Serving traffic



Once the container instances in the new service revision are ready, Cloud Run routes traffic to the new revision.

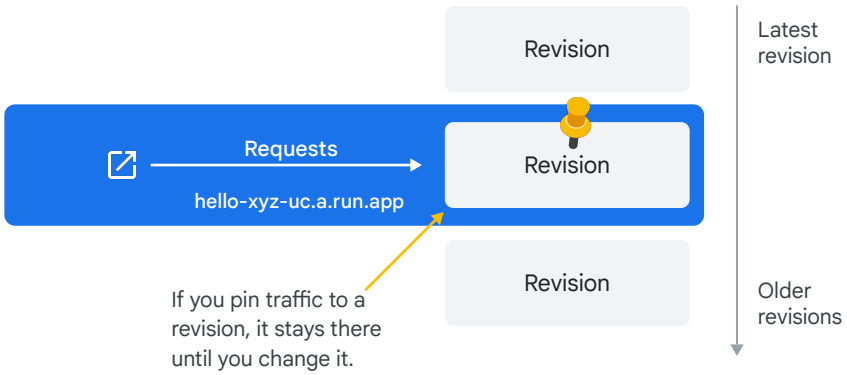
Both revisions (the “previous” revision and the “new” revision) autoscale independently. The containers in the previously active revision will eventually stop serving requests and become idle.

The new revision might have to add more containers to handle demand, and the previous revision will eventually remove the idle containers and scale to zero.

As mentioned previously, you can gradually migrate traffic to the new revision by setting the percentage of requests sent to the new revision.

To perform a gradual rollout of changes to an application, a new service revision can be configured to receive no traffic initially, when deployed with the `–no-traffic` option. To gradually increase the amount of traffic received by the new service revision, you can then update the service to specify an incremental percentage value.

Pinning traffic



You can also pin request traffic to a specific service revision rather than the latest revision, decoupling the deployment of a new revision from the migration of traffic.

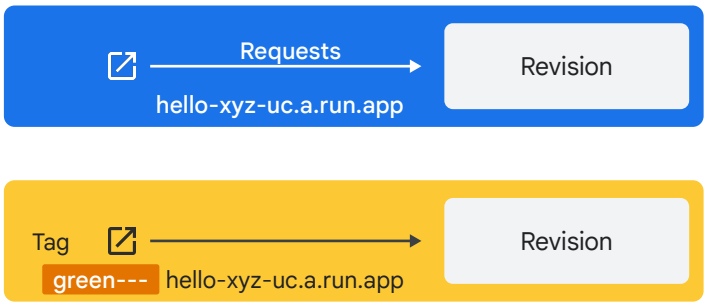
This also means that if you add a new revision, Cloud Run will not automatically send traffic to that new revision.

Pinning to a revision is useful if you want to roll back to previous revision, or if you first want to test your new revision before migrating all request traffic to it.

This is achieved by setting the percentage of request traffic to the revision to 100.

You can set this in the Google Cloud console, with the gcloud CLI, a YAML configuration file, or with Terraform.

Tagging service revisions



Test URL: https://green---hello-xyz-uc.a.run.app

When you deploy a service, you can assign a tag to the new revision that lets you access the revision at a specific URL without serving traffic.

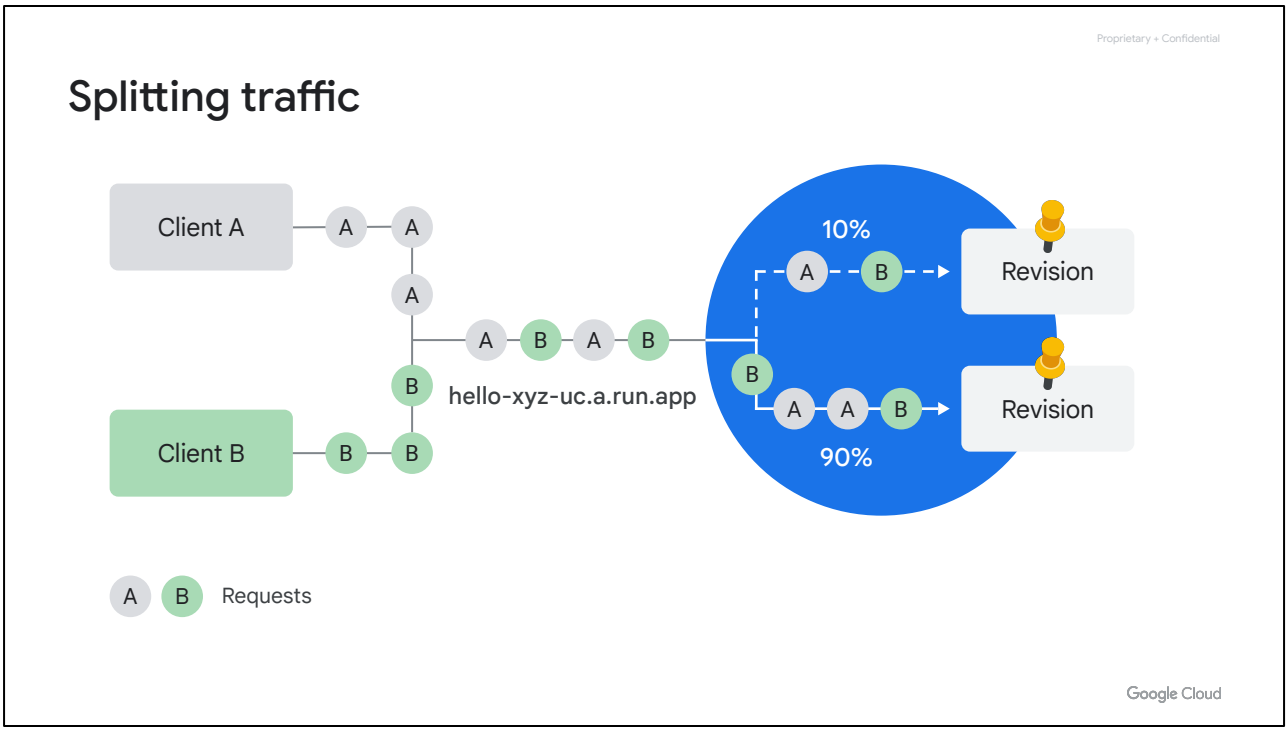
You can then use that tag to gradually migrate traffic to the tagged revision, or to rollback a tagged revision.

A common use case for this feature is to use it for testing and vetting of a new service revision before it serves any traffic.

A tagged revision has its own unique URL, which is the URL of the Cloud Run service and the name of the tag added as a prefix. For example, if you used the tag name green on the service hello, you would test the tagged revision at the URL: `https://green---hello-xyz-uc.a.run.app`.

For example, you might want to tag a revision with the ID of the commit in version control that was used to create the revision.

After confirming that the new revision works properly, you can start migrating traffic to it using the Google Cloud console, the gcloud command line, Terraform, or a YAML file.



Cloud Run lets you specify which service revisions should receive traffic, and to specify traffic percentages that are received by a revision.

This feature lets you roll back to a previous revision, gradually roll out a revision, and split traffic between multiple revisions.

To split request traffic between multiple service revisions, you specify a percentage value. This value indicates the percentage of requests that are routed to each revision. You can configure this percentage in the Google Cloud console, with the gcloud CLI, a YAML configuration file, or with Terraform.

Traffic routing adjustments are not instantaneous. When you change traffic splitting configuration for revisions, all requests currently being processed will continue to completion. In flight requests will not be dropped and may be directed to either a new revision or a previous revision during the transition period.

By default, requests from the same client can be handled by different container instances in a service revision. To change this behavior, you can enable session affinity for a revision in Cloud Run. Session affinity is a best effort made by Cloud Run to route requests from the same client to the same revision container instance.

If you are splitting traffic between multiple revisions with session affinity enabled, see session affinity and traffic splitting for details on the effect of session affinity on traffic splitting.

Remember

1. To deploy your application on Cloud Run, store your container image in Artifact Registry.
2. Cloud Run copies and stores the container image locally.
3. A service revision on Cloud Run is immutable.
4. Changing any service configuration settings results in the creation of a new revision.
5. You can split traffic between service revisions based on percentage of requests.

In summary:

- When deploying to Cloud Run, you can use container images that are stored in Artifact Registry, or Docker Hub. You can also use container images from other public or private registries by setting up an Artifact Registry remote repository. Google recommends the use of Artifact Registry.
- To ensure that containers on Cloud Run start reliably and quickly, Cloud Run copies and stores the container image locally.
- When you deploy a container image for the first time, Cloud Run creates a *service* and its first *revision*. There is only one container image per service.
- Changing any configuration settings of your Cloud Run service results in the creation of a new revision.
- Cloud Run lets you split traffic between service revisions based on the percentage of requests received by the service.