Deploy Your Website on Cloud Run

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

2. Click through the following windows:

    - Continue through the Cloud Shell information window.
    - Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

Copied!

content_copy

4. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Copied!

content_copy

Output:

```
[core]
project = "PROJECT_ID"
```

Task 1. Clone the source repository

Since you are deploying an existing website, you just need to clone the source, so you can focus on creating Docker images and deploying to Cloud Run.

1. In Cloud Shell run the following commands to clone the git repository and change to the appropriate directory:

```
git clone https://github.com/googlecodelabs/monolith-to-
microservices.git
cd ~/monolith-to-microservices
```

Copied!

content_copy

2. Install the NodeJS dependencies so you can test the application before deploying:

```
./setup.sh
```

Copied!

content_copy

This will take a few minutes to run. You will see a success message when it finishes.

3. Test your application by running the following command to start the web server:

```
cd ~/monolith-to-microservices/monolith
npm start
```

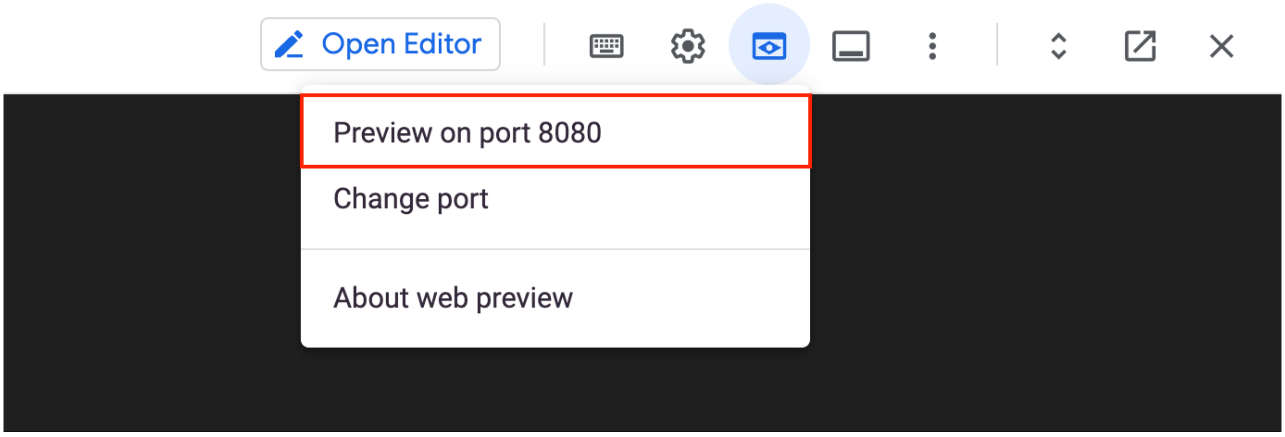
Copied!

content_copy

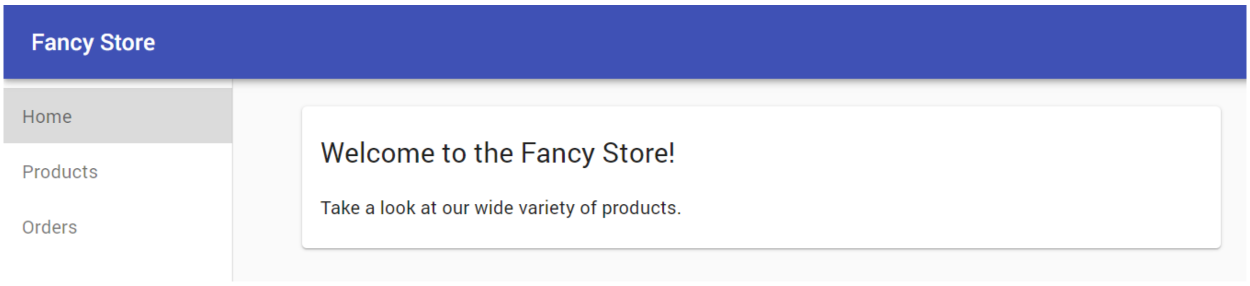
Output:

```
Monolith listening on port 8080!
```

4. Preview your application by clicking the web preview icon and selecting **Preview on port 8080**.



This should open a new window where you can see your Fancy Store web page in action.



Welcome to the Fancy Store!

Take a look at our wide variety of products.

5. Close this window after viewing the website, and stop the web server process by pressing **CTRL+C** in Cloud Shell.

Task 2. Create a Docker container with Cloud Build

Now that you have the source files ready to go, it is time to Dockerize your application!

Normally you would have to take a two step approach that entails building a docker container and pushing it to a registry to store the image for GKE to pull from. Cloud Build let's you build the Docker container and put the image in Artifact Registry with a single command!

Cloud Build will compress the files from the directory and move them to a Cloud Storage bucket. The build process will then take all the files from the bucket and use the Dockerfile, which is present in the same directory, to run the Docker build process.

Create the target Docker repository

You must create a repository before you can push any images to it. Pushing an image can't trigger creation of a repository and the Cloud Build service account does not have permissions to create repositories.

1. In the console, search for **Artifact Registry** in the search field, then click on **Artifact Registry** result.
2. Click **Create Repository**.
3. Specify `monolith-demo` as the repository name.
4. Choose **Docker** as the format.
5. Under Location Type, select Region and then choose the location `Region`.
6. Click **Create**.

Configure authentication

Before you can push or pull images, configure Docker to use the Google Cloud CLI to authenticate requests to Artifact Registry.

- To set up authentication to Docker repositories in the region `Region`, run the following command in Cloud Shell:

  ```
  gcloud auth configure-docker Region-docker.pkg.dev
  ```

  Copied!

  content_copy

  The command updates your Docker configuration. You can now connect with Artifact Registry in your Google Cloud project to push and pull images.

Deploy the image

You will now deploy the image that was built earlier.

1. First you need to enable the Cloud Build, Artifact Registry, and Cloud Run APIs. Run the following command in Cloud Shell to enable them:

```
gcloud services enable artifactregistry.googleapis.com \
    cloudbuild.googleapis.com \
    run.googleapis.com
```

Copied!

content_copy

2. After the APIs are enabled, run the following command to start the build process:

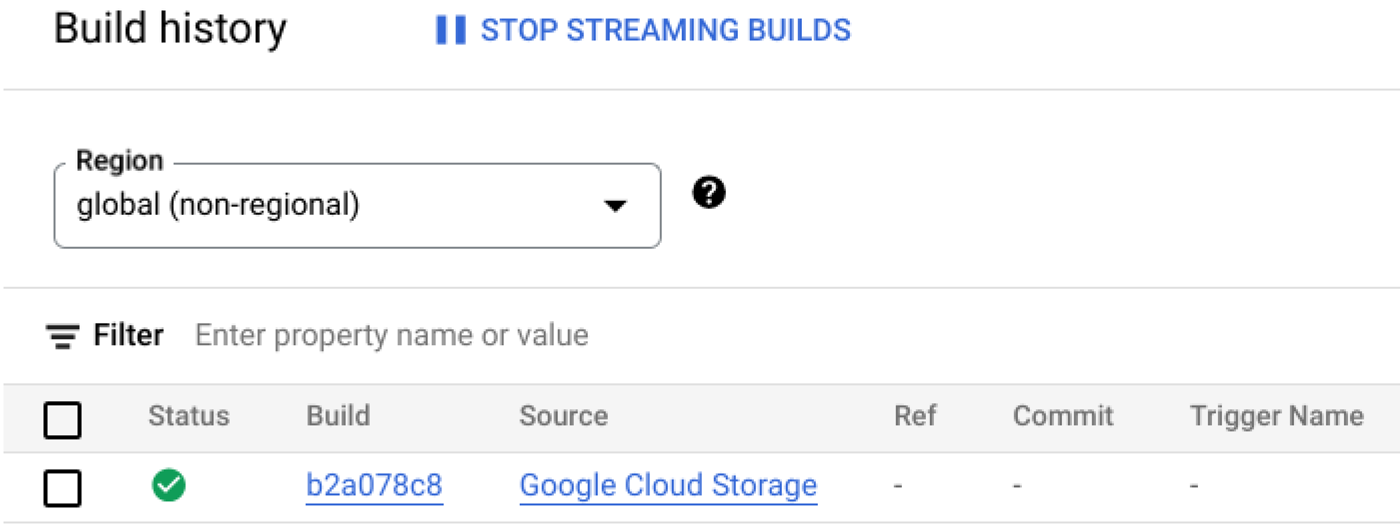
```
gcloud builds submit --tag Region-docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:1.0.0
```

Copied!

content_copy

Note: This process will take a few minutes.

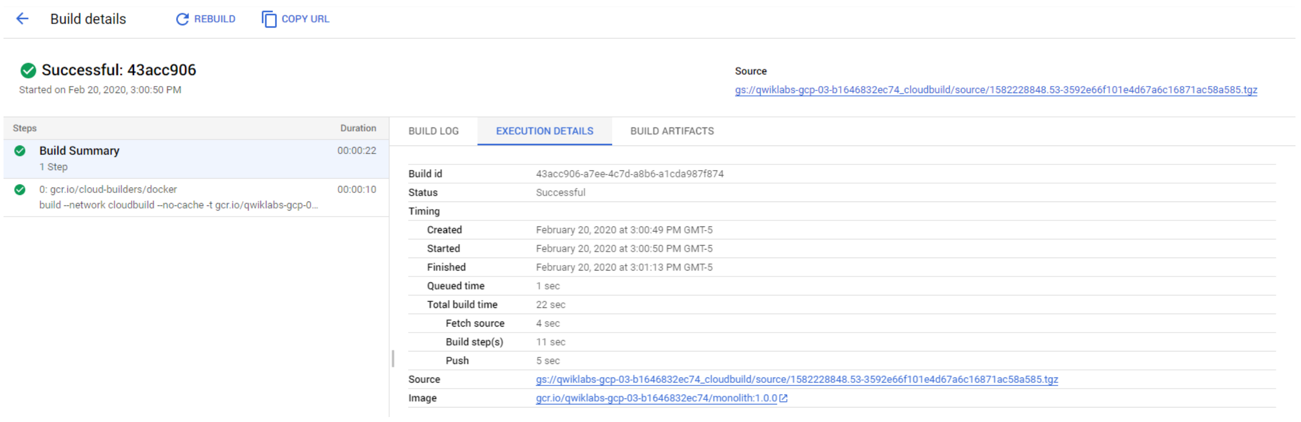
3. To view your build history, or watch the process in real time, in the console, search for **Cloud Build** then click on the **Cloud Build** result.
4. On the **History** page you can see a list of all your builds; there should only be 1 that you just created.



Region: global (non-regional)

	Status	Build	Source	Ref	Commit	Trigger Name
☐	✓	b2a078c8	Google Cloud Storage	-	-	-

- If you click on the Build ID, you can see all the details for that build including the log output.
- From the Build Details page you can view the container image that was created by clicking the **Execution Details** tab, then clicking on on the image link.



Successful: 43acc906
Started on Feb 20, 2020, 3:00:50 PM

Source
gs://qwiklabs-gcp-03-b1646832ec74_cloudbuild/source/1582228848.53-3592e66f101e4d67a6c16871ac58a585.tgz

Steps	Duration
Build Summary 1 Step	00:00:22
0: gcr.io/cloud-builders/docker build --network cloudbuild --no-cache -t gcr.io/qwiklabs-gcp-0...	00:00:10

Build id	43acc906-a7ee-4c7d-a8b6-a1cda987f874
Status	Successful
Timing	
Created	February 20, 2020 at 3:00:49 PM GMT-5
Started	February 20, 2020 at 3:00:50 PM GMT-5
Finished	February 20, 2020 at 3:01:13 PM GMT-5
Queued time	1 sec
Total build time	22 sec
Fetch source	4 sec
Build step(s)	11 sec
Push	5 sec
Source	gs://qwiklabs-gcp-03-b1646832ec74_cloudbuild/source/1582228848.53-3592e66f101e4d67a6c16871ac58a585.tgz
Image	gcr.io/qwiklabs-gcp-03-b1646832ec74/monolith:1.0.0

Task 3. Deploy the container to Cloud Run

There are two approaches for deploying to Cloud Run:

- **Managed Cloud Run**: The Platform as a Service model where all container lifecycle is managed by the Cloud Run product itself. You'll be using this approach in this lab.
- **Cloud Run on GKE**: Cloud Run with an additional layer of control which allows you to bring your own clusters & pods from GKE. You can read more about it here.

1. Run the following command to deploy the image to Cloud Run:

```
gcloud run deploy monolith --image Region-
docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:1.0.0 --
region Region
```

Copied!

content_copy

2. When asked to allow unauthenticated invocations to `[monolith]` type **Y**.

Click *Check my progress* to verify the objective.

Deploy Container To Cloud Run

Check my progress

Verify deployment

1. To verify the deployment was created successfully, run the following command:

```
gcloud run services list
```

Copied!

content_copy

Note: It may take a few moments for the pod status to be Running.

Output:

```
✓
SERVICE: monolith
REGION: Region
URL: https://monolith-2cxtmp4m2q-uc.a.run.app
LAST DEPLOYED BY: student-02-aa7a5aed362d@qwiklabs.net
LAST DEPLOYED AT: 2022-08-19T19:16:14.351981Z
```

This output shows several things. You can see the deployment, as well as the user that deployed it (your email) and the URL you can use to access the app. Looks like everything was created successfully!

2. Click on the URL provided in the list of services. You should see the same website you previewed locally.

Note: You can also view your Cloud Run deployments via the console if you navigate to **Cloud Run** in the **Navigation menu**.

Task 4. Create new revision with lower concurrency

In this section you will deploy your application again, but this time adjusting one of the parameters.

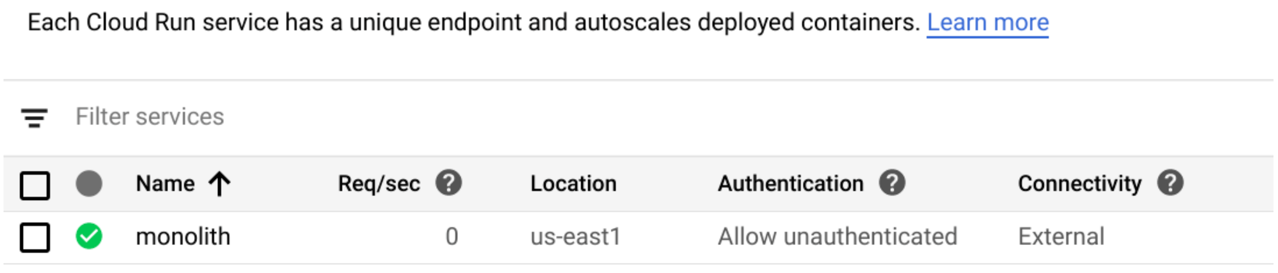
By default, a Cloud Run application will have a concurrency value of 80, meaning that each container instance will serve up to 80 requests at a time. This is a big departure from the Functions-as-a-Service model, where one instance handles one request at a time.

1. Run the following command to re-deploy the same container image with a concurrency value of 1 (just for testing), and see what happens:

```
gcloud run deploy monolith --image Region-
docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:1.0.0 --
region Region --concurrency 1
```
Copied!

content_copy

2. To see the details, from the **Navigation menu**, click on **Cloud Run**, then click on the **monolith** service:

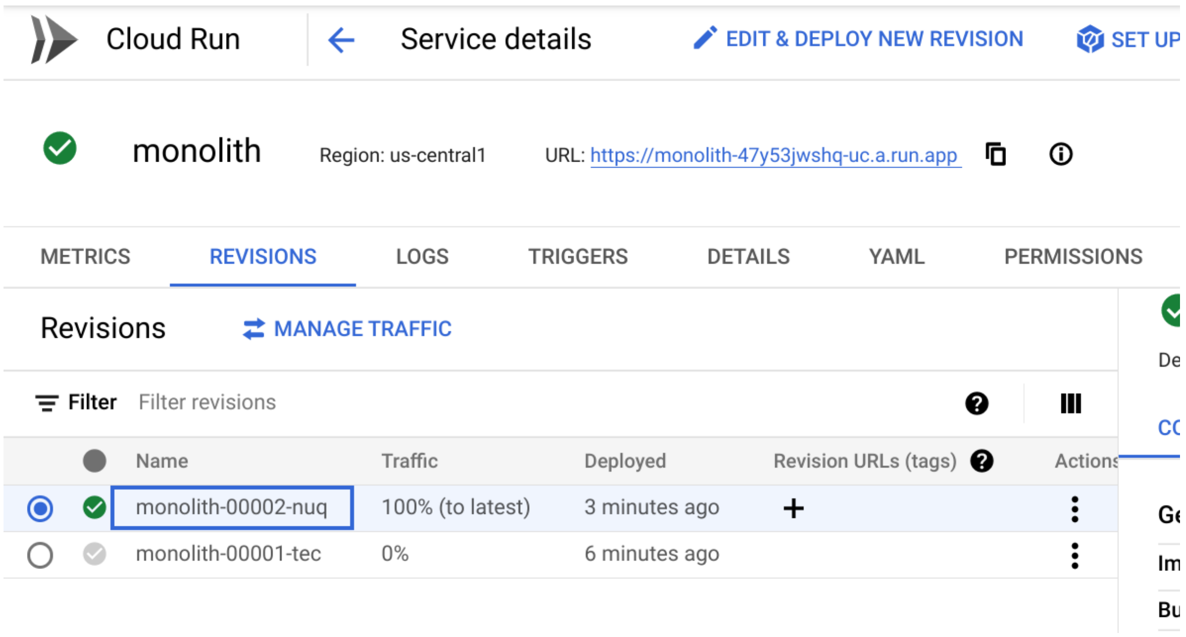

Each Cloud Run service has a unique endpoint and autoscales deployed containers. Learn more

Filter services

		Name	Req/sec	Location	Authentication	Connectivity
☐	✓	monolith	0	us-east1	Allow unauthenticated	External

3. On the Service Details page, click on the **Revisions** tab. You should now see 2 revisions created.

The most recent deployment has Details on the right hand side.



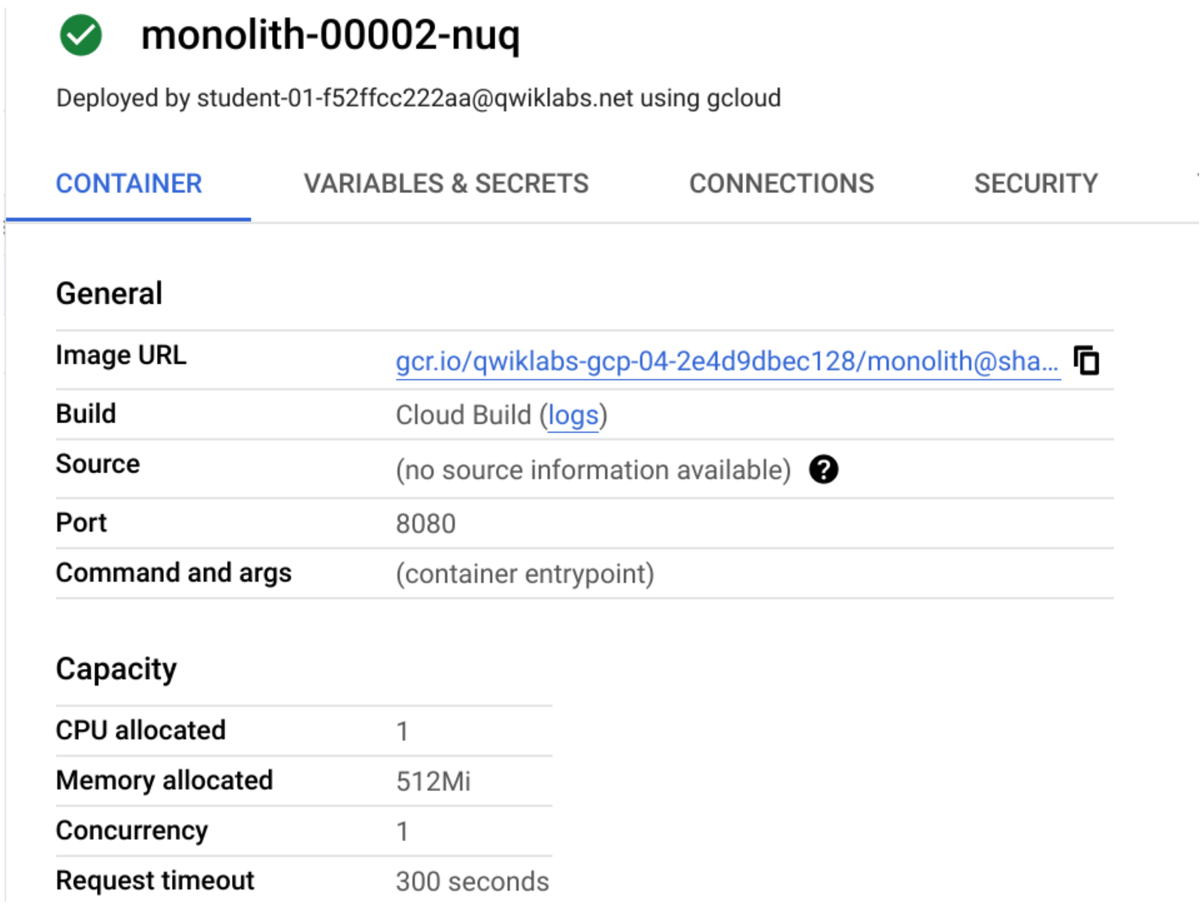
monolith Region: us-central1 URL: https://monolith-47y53jwshq-uc.a.run.app

Revisions

Filter revisions

		Name	Traffic	Deployed	Revision URLs (tags)	Actions
◉	✓	monolith-00002-nuq	100% (to latest)	3 minutes ago	+	⋮
○	✓	monolith-00001-tec	0%	6 minutes ago		⋮

You will see that the concurrency value has been reduced to "1".


monolith-00002-nuq

Deployed by student-01-f52ffcc222aa@qwiklabs.net using gcloud

General

Image URL	gcr.io/qwiklabs-gcp-04-2e4d9dbec128/monolith@sha...
Build	Cloud Build (logs)
Source	(no source information available)
Port	8080
Command and args	(container entrypoint)

Capacity

CPU allocated	1
Memory allocated	512Mi
Concurrency	1
Request timeout	300 seconds

Although this configuration is sufficient for testing, in most production scenarios you will have containers supporting multiple concurrent requests.

Create new revision with lower concurrency

Task 5. Make changes to the website

Scenario: Your marketing team has asked you to change the homepage for your site. They think it should be more informative of who your company is and what you actually sell.

Task: You will add some text to the homepage to make the marketing team happy! It looks like one of our developers already created the changes with the file name `index.js.new`. You can just copy this file to `index.js` and your changes should be reflected. Follow the instructions below to make the appropriate changes.

1. Run the following commands to copy the updated file to the correct file name:

```
cd ~/monolith-to-microservices/react-app/src/pages/Home
mv index.js.new index.js
```

Copied!

content_copy

2. Print its contents to verify the changes:

```
cat ~/monolith-to-microservices/react-app/src/pages/Home/index.js
```

Copied!

content_copy

The resulting code should look like this:

```
/*
Copyright 2019 Google LLC

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
See the License for the specific language governing permissions and
limitations under the License.
*/
import React from "react";
import { Box, Paper, Typography } from "@mui/material";

export default function Home() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Paper
        elevation={3}
        sx={{
          width: "800px",
          margin: "0 auto",
          padding: (theme) => theme.spacing(3, 2),
        }}
      >
        <Typography variant="h5">Fancy Fashion &amp; Style
Online</Typography>
        <br />
        <Typography variant="body1">
          Tired of mainstream fashion ideas, popular trends and
societal norms?
          This line of lifestyle products will help you catch up with
the Fancy
          trend and express your personal style. Start shopping Fancy
items now!
        </Typography>
      </Paper>
    </Box>
  );
}
```
You updated the React components, but you need to build the React app to generate the static files.

3. Run the following command to build the React app and copy it into the monolith public directory:

```
cd ~/monolith-to-microservices/react-app
npm run build:monolith
```
Copied!

content_copy

Now that the code is updated, rebuild the Docker container and publish it to Artifact Registry. You can use the same command as before, except this time you will update the version label.

4. Run the following command to trigger a new Cloud Build with an updated image version of 2.0.0:

```
cd ~/monolith-to-microservices/monolith
gcloud builds submit --tag Region-
docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:2.0.0
```
Copied!

content_copy

In the next section you will use this image to update your application with zero downtime.

Next, you can restore the original concurrency without re-deploying. You could set the concurrency value back to the default of "80", or you could just set the value to "0", which will remove any concurrency restrictions and set it to the default max (which happens to be 80).

4. Run the following command to update the current revision, using a concurrency value of `80`:

```
gcloud run deploy monolith --image Region-
docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:1.0.0 --
region Region --concurrency 80
```
Copied!

content_copy

You will notice that another revision has been created, that traffic has now been redirected, and that the concurrency is back up to 80.

Task 6. Update website with zero downtime

The changes are complete and the marketing team is happy with your updates! It is time to update the website without interruption to the users. Cloud Run treats each deployment as a new *Revision* which will first be brought online, then have traffic redirected to it.

By default the latest revision will be assigned 100% of the inbound traffic for a service. It is possible to use "Routes" to allocate different percentages of traffic to different revisions within a service. Follow the instructions below to update your website.

- Run the following command to re-deploy the service to update the image to a new version with the following command:

```
gcloud run deploy monolith --image Region-
docker.pkg.dev/${GOOGLE_CLOUD_PROJECT}/monolith-demo/monolith:2.0.0 --
region Region
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

Update website with zero downtime

Check my progress

Verify deployment

1. Validate that your deployment updated by running the following command:

```
gcloud run services describe monolith --platform managed --region
Region
```

Copied!

content_copy

Output:

```
✓ Service monolith in region
```

Here you will see that the Service is now using the latest version of your image, deployed in a new revision.

To verify the changes, navigate to the external URL of the Cloud Run service, refresh the page, and notice that the application title has been updated.

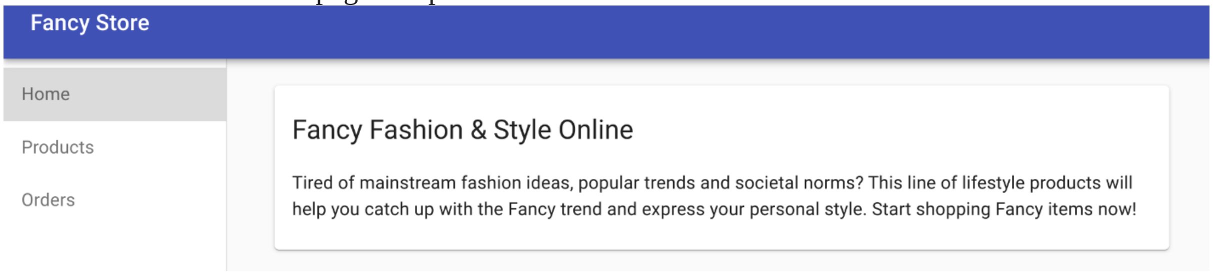
2. Run the following command to list the services and view the service Url:

```
gcloud beta run services list
```

Copied!

content_copy

3. Click on the URL of the service. Your web site should now be displaying the text you just added to the homepage component!



Fancy Fashion & Style Online

Tired of mainstream fashion ideas, popular trends and societal norms? This line of lifestyle products will help you catch up with the Fancy trend and express your personal style. Start shopping Fancy items now!