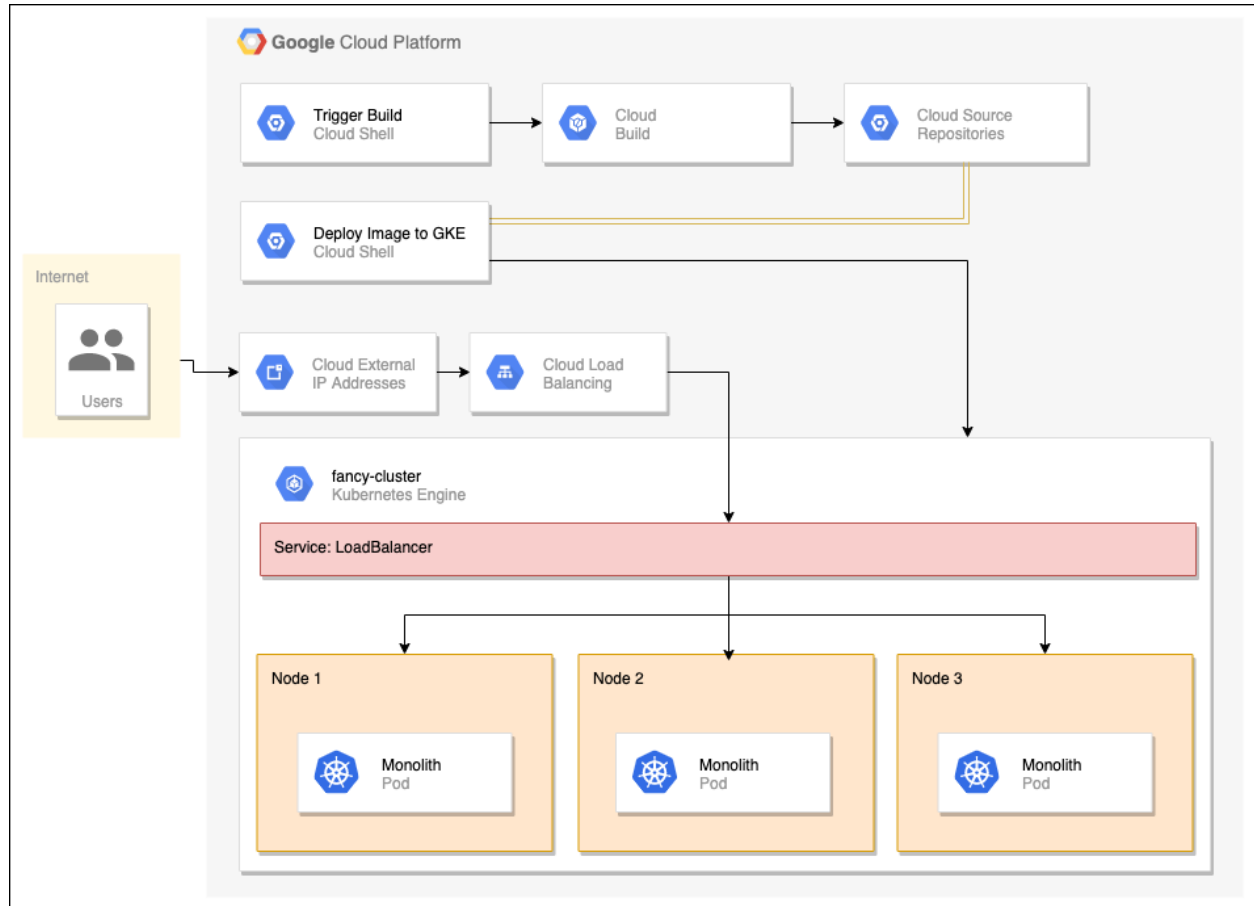


Deploy, Scale, and Update Your Website on Google Kubernetes Engine



Set the zone

- Set the default zone and project configuration:

`gcloud config set compute/zone lab zone`

```
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$ gcloud config set compute/zone us-central1-a
WARNING: Property validation for compute/zone was skipped.
Updated property [compute/zone].
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$
```

Task 1. Create a GKE cluster

You need a Kubernetes cluster to deploy your website to. First, make sure the proper APIs are enabled.

- Run the following to create a GKE cluster named fancy-cluster with 3 nodes:

`gcloud container clusters create fancy-cluster --num-nodes 3`

```
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$ gcloud container clusters create fancy-cluster --num-nodes 3
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migration instructions.
Note: Your Pod address range ('--cluster-ip4-cidr') can accommodate at most 1008 node(s).
```

2. Now run the following command and see the cluster's three worker VM instances:

gcloud compute instances list

```
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$ gcloud compute instances list
NAME: gke-fancy-cluster-default-pool-4d4bc947-8x5v
ZONE: us-central1-a
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.3
EXTERNAL_IP: 34.133.4.37
STATUS: RUNNING

NAME: gke-fancy-cluster-default-pool-4d4bc947-c1sh
ZONE: us-central1-a
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.4
EXTERNAL_IP: 34.28.242.188
STATUS: RUNNING

NAME: gke-fancy-cluster-default-pool-4d4bc947-g7jn
ZONE: us-central1-a
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.5
EXTERNAL_IP: 35.223.224.152
STATUS: RUNNING
```

4. Find your Kubernetes cluster and related information in the console.
5. Click the **Navigation menu** (≡) > **Kubernetes Engine** > **Clusters**.

You should see your cluster named *fancy-cluster*.

Task 2. Clone source repository

Since this is an existing website, you just need to clone the source, so you can focus on creating Docker images and deploying to GKE.

1. Run the following commands to clone the git repo to your Cloud Shell instance:

```
cd ~
```

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

```
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$ git clone https://github.com/googlecodelabs/monolith-to-microservices.git
Cloning into 'monolith-to-microservices'...
remote: Enumerating objects: 1226, done.
remote: Counting objects: 100% (268/268), done.
remote: Compressing objects: 100% (151/151), done.
remote: Total 1226 (delta 211), reused 130 (delta 112), pack-reused 958 (from 4)
Receiving objects: 100% (1226/1226), 4.37 MiB | 16.76 MiB/s, done.
Resolving deltas: 100% (591/591), done.
```

2. Change to the appropriate directory.
3. You will also install the NodeJS dependencies so you can test your application before deploying:

cd ~/monolith-to-microservices

```
student_03_b8b41a615ae8@cloudshell:~ (qwiklabs-gcp-04-0c4667ee9486)$ cd ~/monolith-to-microservices
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$
```

./setup.sh

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$ ./setup.sh
Installing monolith dependencies...

added 68 packages, and audited 69 packages in 3s

14 packages are looking for funding
  run `npm fund` for details

5 vulnerabilities (3 low, 2 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
Completed.

Installing microservices dependencies...
.
```

4. Ensure you are running Cloud Shell with the latest version of npm:

nvm install --lts

```
Setup Completed Successfully!
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$ nvm install --lts
Installing latest LTS version.
v22.15.0 is already installed.
Now using node v22.15.0 (npm v11.3.0)
```

5. Change to the appropriate directory and test the application by running the following command to start the web server:

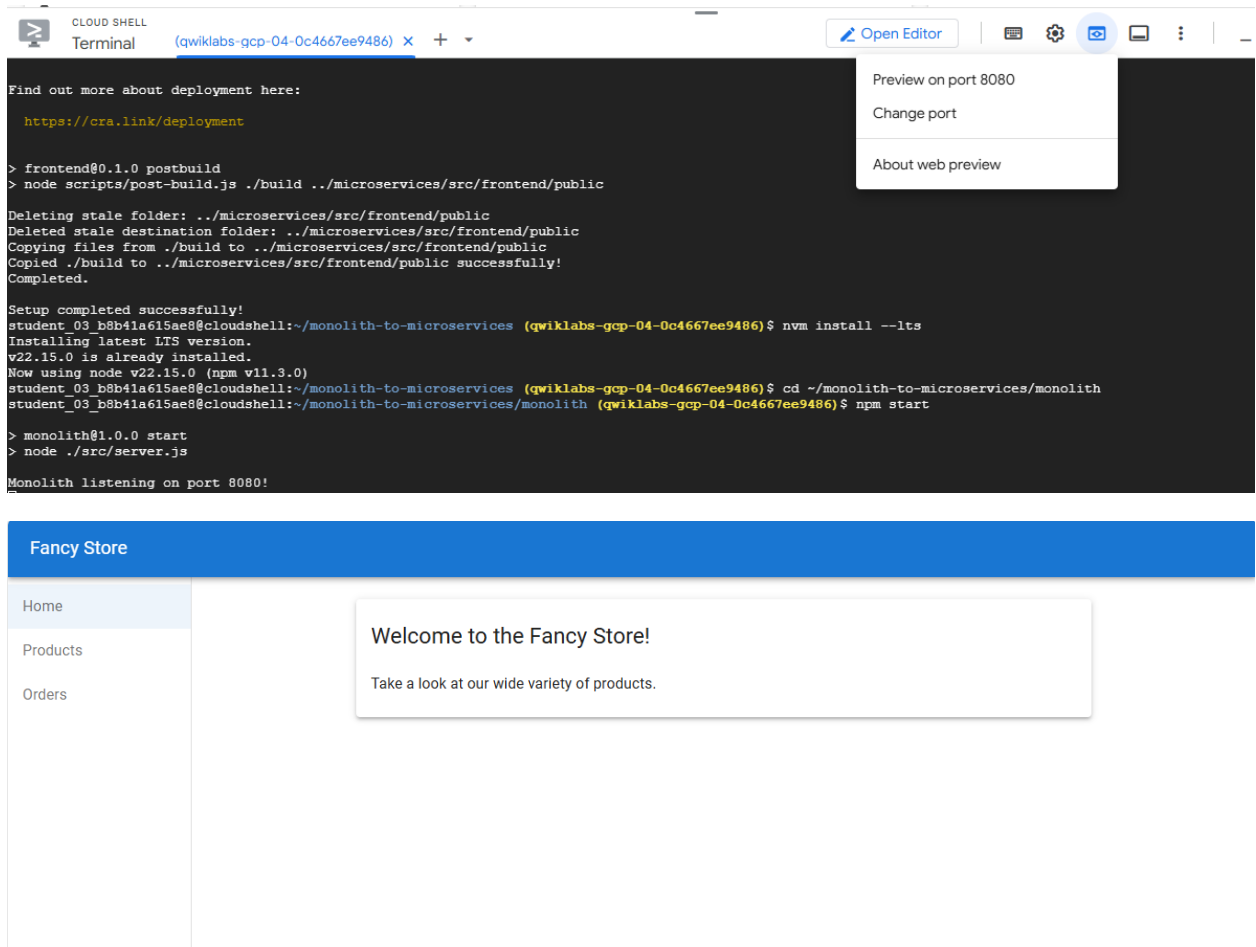
cd ~/monolith-to-microservices/monolith

npm start

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$ cd ~/monolith-to-microservices/monolith
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ npm start

> monolith@1.0.0 start
> node ./src/server.js

Monolith listening on port 8080!
```



CLOUD SHELL Terminal (qwiklabs-gcp-04-0c4667ee9486) x +

Find out more about deployment here:
<https://cra.link/deployment>

```
> frontend@0.1.0 postbuild
> node scripts/post-build.js ./build ../microservices/src/frontend/public

Deleting stale folder: ../microservices/src/frontend/public
Deleted stale destination folder: ../microservices/src/frontend/public
Copying files from ./build to ../microservices/src/frontend/public
Copied ./build to ../microservices/src/frontend/public successfully!
Completed.

Setup completed successfully!
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$ npm install --its
Installing latest LTS version.
v22.15.0 is already installed.
Now using node v22.15.0 (npm v11.3.0)
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-04-0c4667ee9486)$ cd ~/monolith-to-microservices/monolith
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ npm start

> monolith@1.0.0 start
> node ./src/server.js

Monolith listening on port 8080!
```

Preview on port 8080
Change port
About web preview

Fancy Store

Home
Products
Orders

Welcome to the Fancy Store!

Take a look at our wide variety of products.

Leave this tab open, you'll return to it later in the lab.

7. To stop the web server process, press CTRL+C in Cloud Shell.

Task 3. Create Docker container with Cloud Build

Now that you have your 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!

Google Cloud Build will compress the files from the directory and move them to a Google Cloud Storage bucket. The build process will then take all the files from the bucket and use the Dockerfile to run the Docker build process. Since you specified the --tag flag with the host as gcr.io for the Docker image, the resulting Docker image will be pushed to the Artifact Registry.

1. First, to make sure you have the Cloud Build API enable, run the following command:

```
gcloud services enable cloudbuild.googleapis.com
```

2. Run the following to start the build process:

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

```
gcloud builds submit --tag gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0 .
```

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486) $ gcloud builds submit --tag gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0 .
Creating temporary archive of 27 file(s) totalling 2.4 MiB before compression.
Uploading tarball of [.] to [gs://qwiklabs-gcp-04-0c4667ee9486-cloudbuild/source/1746508280.223804-6fc243c440cd467c8e45665d6ba48e3d.tgz]
Created [https://cloudbuild.googleapis.com/v1/projects/qwiklabs-gcp-04-0c4667ee9486/locations/global/builds/f2cad23f-3cdc-494b-ab9f-1297aa8cc186].
Logs are available at [ https://console.cloud.google.com/cloud-build/builds/f2cad23f-3cdc-494b-ab9f-1297aa8cc186?project=413435659136 ].
Waiting for build to complete. Polling interval: 1 second(s).
```

3. This process will take a few minutes.
4. To view your build history or watch the process in real time by clicking the **Navigation menu** and scrolling down to CI/CD section, then click **Cloud Build > History**. Here you can see a list of all your previous builds.
5. Click on the build name to see all the details for that build including the log output.

Optional: From the Build details page, click on the **Build summary > Execution details > Image name** in the build information section to see the container image

Task 4. Deploy container to GKE

Now that you have containerized your website and pushed your container to Artifact Registry, it is time to deploy to Kubernetes!

To deploy and manage applications on a GKE cluster, you must communicate with the Kubernetes cluster management system. You typically do this by using the kubectl command-line tool.

Kubernetes represents applications as **Pods**, which are units that represent a container (or group of tightly-coupled containers). The Pod is the smallest deployable unit in Kubernetes. In this lab, each Pod contains only your monolith container.

To deploy your application, create a **Deployment** resource. The Deployment manages multiple copies of your application, called replicas, and schedules them to run on the individual nodes in your cluster. For this lab the Deployment will be running only one Pod of your application. Deployments ensure this by creating a

ReplicaSet. The ReplicaSet is responsible for making sure the number of replicas specified are always running.

The `kubectl create deployment` command you'll use next causes Kubernetes to create a Deployment named `monolith` on your cluster with **1** replica.

- Run the following command to deploy your application:

`kubectl create deployment monolith --`

`image=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486) $ kubectl create deployment monolith --image=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0
deployment.apps/monolith created
```

Verify deployment

1. Verify the Deployment was created successfully:

`kubectl get all`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486) $ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/monolith-5bc6686ccb-qxxsc      1/1     Running   0           32s

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                  ClusterIP     34.118.224.1 <none>        443/TCP    11m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monolith            1/1     1             1           33s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/monolith-5bc6686ccb 1         1         1       33s
```

This output shows you several things:

- The Deployment, which is current
- The ReplicaSet with desired pod count of 1
- The Pod, which is running

Task 5. Expose GKE deployment

You have deployed your application on GKE, but there isn't a way to access it outside of the cluster. By default, the containers you run on GKE are not accessible from the Internet because they do not have external IP addresses. You must explicitly expose your application to traffic from the Internet via a **Service** resource. A Service provides networking and IP support to your application's Pods. GKE creates an external IP and a Load Balancer for your application.

- Run the following command to expose your website to the Internet:

`kubectl expose deployment monolith --type=LoadBalancer --port 80 --target-port 8080`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ kubectl expose deployment monolith --type=LoadBalancer --port 80
--target-port 8080
service/monolith exposed
```

Accessing the service

GKE assigns the external IP address to the Service resource, not the Deployment.

1. If you want to find out the external IP that GKE provisioned for your application, you can inspect the Service with the `kubectl get service` command:

`kubectl get service`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ kubectl get service
```

| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
|------------|--------------|----------------|-------------|--------------|-----|
| kubernetes | ClusterIP | 34.118.224.1 | <none> | 443/TCP | 12m |
| monolith | LoadBalancer | 34.118.235.190 | <pending> | 80:32213/TCP | 22s |

Re-run the command until your service has an external IP address.

2. Once you've determined the external IP address for your application, copy the IP address, then point your browser the URL (such as "<http://203.0.113.0>") to check if your application is accessible.

Task 6. Scale GKE deployment

Now that your application is running in GKE and is exposed to the internet, imagine your website has become extremely popular! You need a way to scale your application to multiple instances so it can handle all this traffic. Next you will learn how to scale the application up to 3 replicas.

1. In Cloud Shell, run the following command to scale you deployment up to 3 replicas:

`kubectl scale deployment monolith --replicas=3`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ kubectl scale deployment monolith --replicas=3
deployment.apps/monolith scaled
```

2. Verify the Deployment was scaled successfully:

`kubectl get all`

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ kubectl get all
```

| NAME | READY | STATUS | RESTARTS | AGE |
|-------------------------------|-------|-------------------|----------|-------|
| pod/monolith-5bc6686ccb-ckb7d | 1/1 | Running | 0 | 22s |
| pod/monolith-5bc6686ccb-qxxsc | 1/1 | Running | 0 | 4m57s |
| pod/monolith-5bc6686ccb-r6g9v | 0/1 | ContainerCreating | 0 | 22s |

| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
|--------------------|--------------|----------------|-------------|--------------|-------|
| service/kubernetes | ClusterIP | 34.118.224.1 | <none> | 443/TCP | 16m |
| service/monolith | LoadBalancer | 34.118.235.190 | 34.42.69.34 | 80:32213/TCP | 3m48s |

| NAME | READY | UP-TO-DATE | AVAILABLE | AGE |
|--------------------------|-------|------------|-----------|-------|
| deployment.apps/monolith | 2/3 | 3 | 2 | 4m58s |

| NAME | DESIRED | CURRENT | READY | AGE |
|-------------------------------------|---------|---------|-------|-------|
| replicaset.apps/monolith-5bc6686ccb | 3 | 3 | 3 | 4m58s |

ou should now see 3 instances of your pod running. Notice that your deployment and replica set now have a desired count of 3.

Task 7. 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 the developers has already created the changes with the file name index.js.new. You can just copy this file to index.js and the changes should be reflected. Follow the instructions below to make the appropriate changes.

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

```
cd ~/monolith-to-microservices/react-app/src/pages/Home
```

```
mv index.js.new index.js
```

```
student_03 b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ cd ~/monolith-to-microservices/react-app/src/pages/home
mv index.js.new index.js
student_03 b8b41a615ae8@cloudshell:~/monolith-to-microservices/react-app/src/pages/Home (qwiklabs-gcp-04-0c4667ee9486)$
```

2. Print its contents to verify the changes:

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

```
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 & 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>
  );
}
```


The React components were updated, but the React app needs to be built 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
```

Now that the code is updated, you need to rebuild the Docker container and publish it to the Artifact Registry. Use the same command as earlier, except this time 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 gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2.0.0 .
```

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/react-app (qwiklabs-gcp-04-0c4667ee9486)$ cd ~/monolith-to-microservices/monolith
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ gcloud builds submit --tag gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2.0.0 .
Creating temporary archive of 27 file(s) totalling 2.4 MiB before compression.
Uploading tarball of [...] to [gs://qwiklabs-gcp-04-0c4667ee9486_cloudbuild/source/1746508803.361266-239f1744ee2140aa896ef387d6349cc2.tgz]
```

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

Task 8. Update website with zero downtime

The changes are completed and the marketing team is happy with your updates! It is time to update the website without interruption to the users.

GKE's rolling update mechanism ensures that your application remains up and available even as the system replaces instances of your old container image with your new one across all the running replicas.

- Tell Kubernetes that you want to update the image for your deployment to a new version with the following command:

```
kubectl set image deployment/monolith
```

```
monolith=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2.0.0
```

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486)$ kubectl set image deployment/monolith monolith=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2.0.0
deployment.apps/monolith image updated
```

Verify deployment

1. You can validate your deployment update by running the following command:

kubectl get pods

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486) $ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
monolith-5bc6686ccb-ckb7d          1/1     Terminating    0          3m57s
monolith-5bc6686ccb-qxxsc          1/1     Running         0          8m32s
monolith-5bc6686ccb-r6g9v          1/1     Terminating    0          3m57s
monolith-6855fbfb7f-gzppg          1/1     Running         0          18s
monolith-6855fbfb7f-rc5mc          1/1     Running         0          22s
monolith-6855fbfb7f-vpdfg          0/1     ContainerCreating 0          15s
```

Here you will see 3 new pods being created and your old pods getting shut down. You can tell by the age which are new and which are old. Eventually, you will only see 3 pods again which will be your 3 updated pods.

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

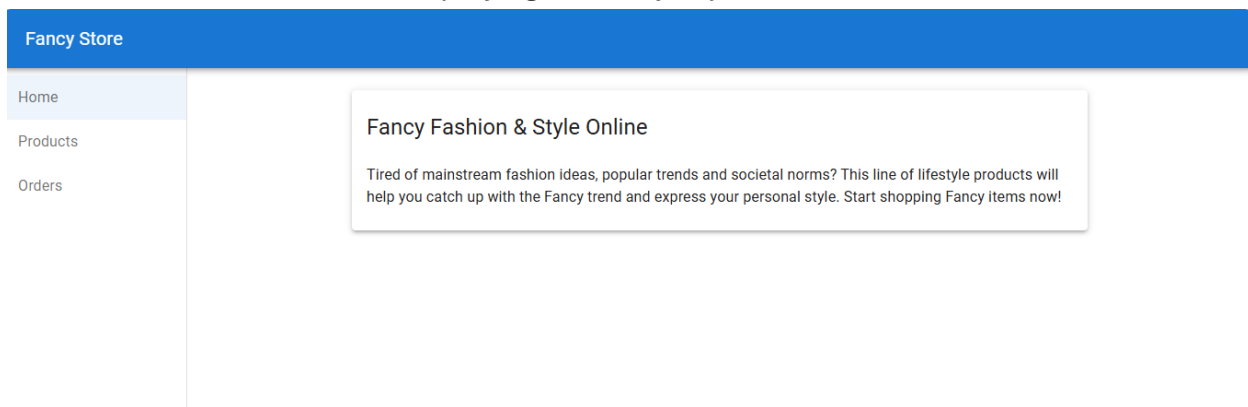
npm start

```
student_03_b8b41a615ae8@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-04-0c4667ee9486) $ npm start
> monolith@1.0.0 start
> node ./src/server.js

Monolith listening on port 8080!
```

3. To verify our changes, return to the app web page tab and refresh the page. Notice that your application has been updated.

Your website should now be displaying the text you just added



4. To stop the web server process, press CTRL+C in Cloud Shell.

Task 9. Cleanup

Although all resources will be deleted when you complete this lab, in your own environment it's a good idea to remove resources you no longer need.

1. Delete git repository:

```
cd ~  
rm -rf monolith-to-microservices
```

Copied!

content_copy

2. Delete Artifact Registry images:

```
# Delete the container image for version 1.0.0 of the monolith  
gcloud container images delete gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0 --  
quiet
```

```
# Delete the container image for version 2.0.0 of the monolith  
gcloud container images delete gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2.0.0 --  
quiet
```

Copied!

content_copy

3. Delete Google Cloud Build artifacts from Google Cloud Storage:

```
# The following command will take all source archives from all builds and delete them from  
cloud storage
```

```
# Run this command to print all sources:  
# gcloud builds list | awk 'NR > 1 {print $4}'
```

```
gcloud builds list | grep 'SOURCE' | cut -d ' ' -f2 | while read line; do gsutil rm $line; done
```

Copied!

content_copy

4. Delete GKE Service:

```
kubectl delete service monolith  
kubectl delete deployment monolith
```

Copied!

content_copy

5. Delete GKE Cluster:

gcloud container clusters delete fancy-cluster lab region

Copied!

content_copy

6. Type Y to confirm this action. This command may take a little while.