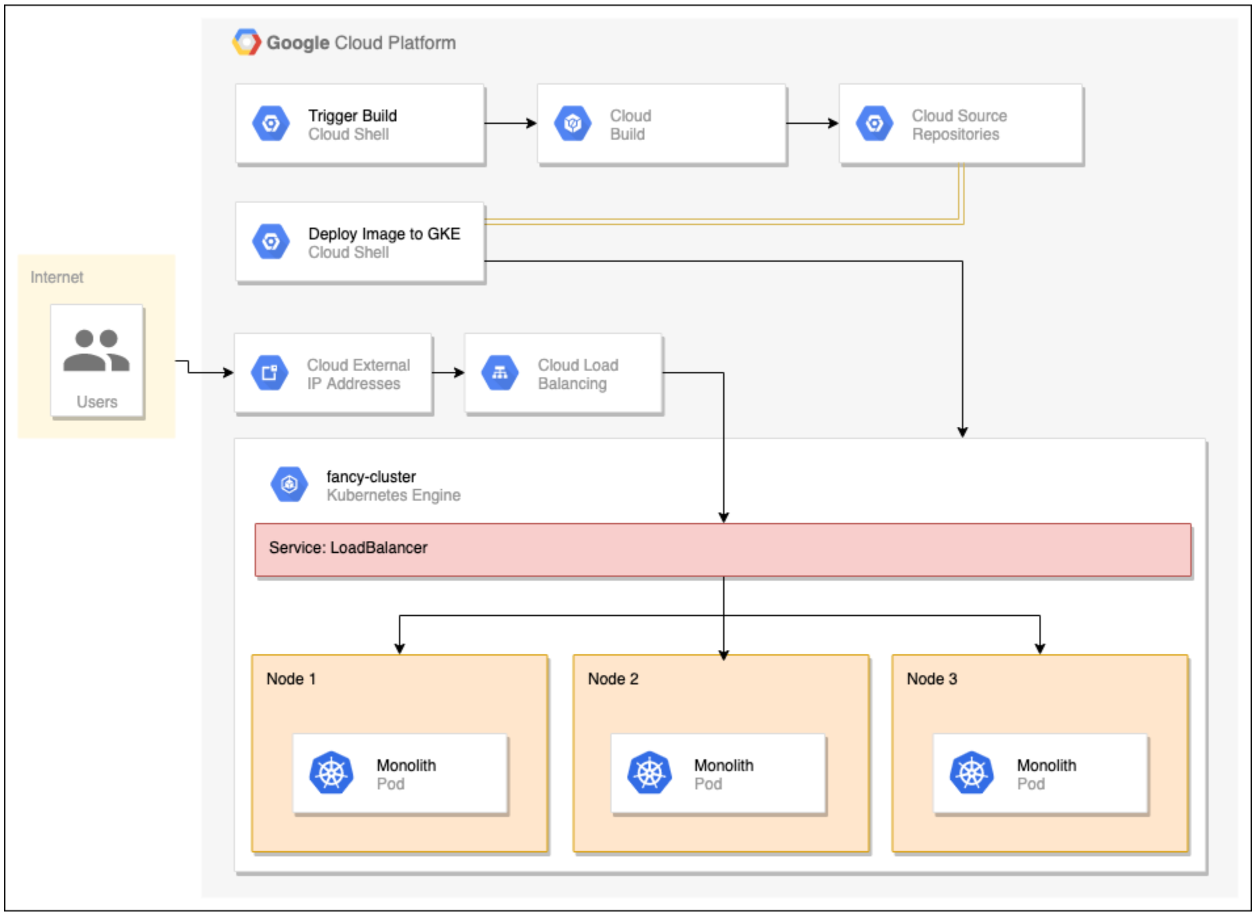
Running websites and applications is hard. Things go wrong when they shouldn't, servers crash, increase in demand causes more resources to be utilized, and making changes without downtime is complicated and stressful. Imagine if there was a tool that could help you do all this and even allow you to automate it. With Kubernetes, all of this is not only possible, it's easy!

In this lab you will assume the role of a developer at a fictional company, Fancy Store, running an ecommerce website. Due to problems with scaling and outages, you are tasked with deploying your application onto the Google Kubernetes Engine (GKE).

The exercises in this lab are ordered to reflect a common cloud developer experience:

1. Create a GKE cluster
2. Create a Docker container
3. Deploy the container to GKE
4. Expose the container via a service
5. Scale the container to multiple replicas
6. Modify the website
7. Rollout a new version with zero downtime



```
gcloud auth list
gcloud config list project
gcloud config set compute/zone us-central1-f
```

Task 1. Create a GKE cluster

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

1. Run the following command to enable the Container Registry API:

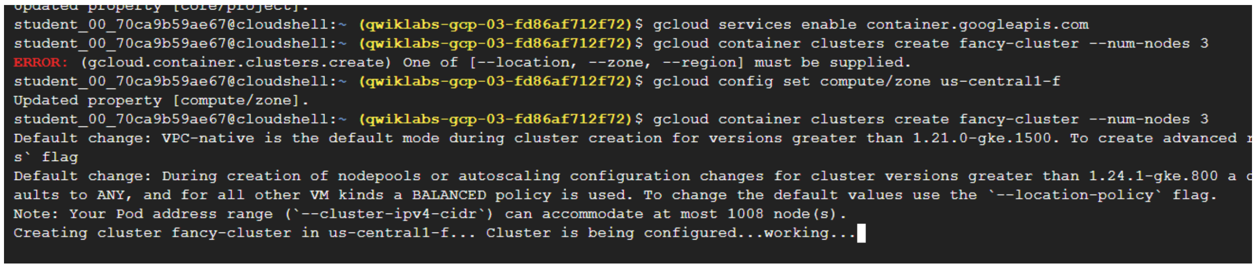
```
gcloud services enable container.googleapis.com
```

Now you are ready to create a cluster!

2. Run the following to create a GKE cluster named `fancy-cluster` with **3** nodes:

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

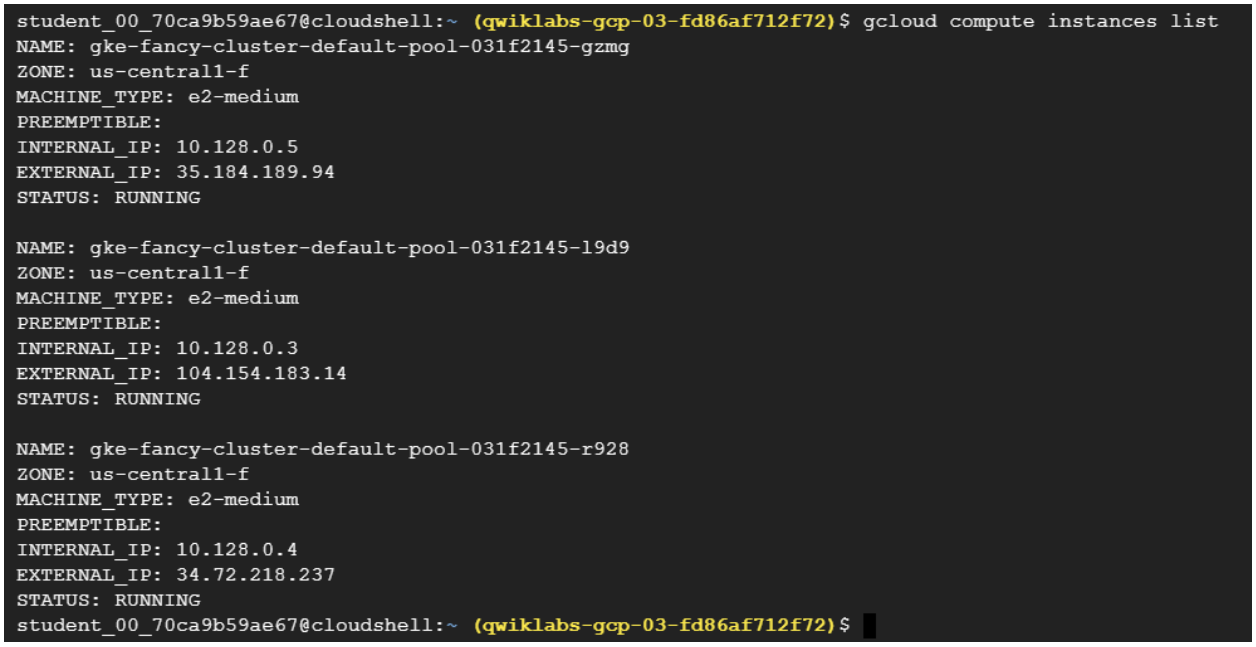
It will take a few minutes for the cluster to be created.



```
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$ gcloud services enable container.googleapis.com
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$ gcloud container clusters create fancy-cluster --num-nodes 3
ERROR: (gcloud.container.clusters.create) One of [--location, --zone, --region] must be supplied.
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$ gcloud config set compute/zone us-central1-f
Updated property [compute/zone].
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$ gcloud container clusters create fancy-cluster --num-nodes 3
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced
s` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a
aults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the `--location-policy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster fancy-cluster in us-central1-f... Cluster is being configured...working...
```

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

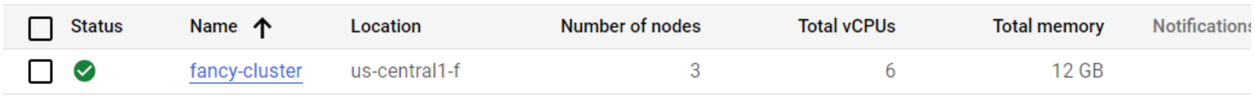
```
gcloud compute instances list
```



```
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$ gcloud compute instances list
NAME: gke-fancy-cluster-default-pool-031f2145-gzmg
ZONE: us-central1-f
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.5
EXTERNAL_IP: 35.184.189.94
STATUS: RUNNING

NAME: gke-fancy-cluster-default-pool-031f2145-19d9
ZONE: us-central1-f
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.3
EXTERNAL_IP: 104.154.183.14
STATUS: RUNNING

NAME: gke-fancy-cluster-default-pool-031f2145-r928
ZONE: us-central1-f
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.4
EXTERNAL_IP: 34.72.218.237
STATUS: RUNNING
student_00_70ca9b59ae67@cloudshell:~ (qwiklabs-gcp-03-fd86af712f72)$
```

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

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



☐	Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications
☐	✓	fancy-cluster	us-central1-f	3	6	12 GB	

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:

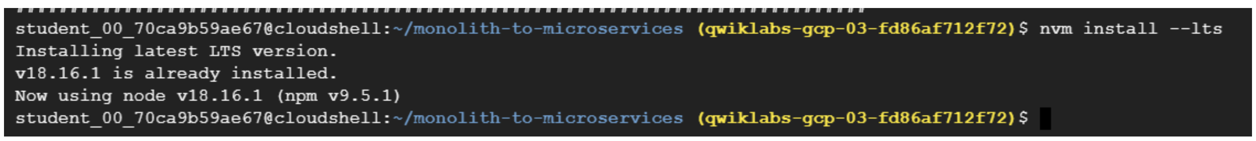
```
cd ~

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

2. You will also install the NodeJS dependencies so you can test your application before deploying:

```
cd ~/monolith-to-microservices

./setup.sh
```

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

```
nvm install --lts
```



```
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-03-fd86af712f72)$ nvm install --lts
Installing latest LTS version.
v18.16.1 is already installed.
Now using node v18.16.1 (npm v9.5.1)
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices (qwiklabs-gcp-03-fd86af712f72)$
```

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
```

6. You can preview your application by clicking the web preview icon and selecting **Preview on port 8080**:

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. Make life easier and use Cloud Build to build the Docker container and put the image in the Container Registry with a single command! With a single command you can build and move the image to the container registry. Learn more about the manual process of creating a docker file and pushing it you can go from Quickstart for Container Registry Guide.

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 we specified the `--tag` flag with the host as gcr.io for the Docker image, the resulting Docker image will be pushed to the Google Cloud Container Registry.

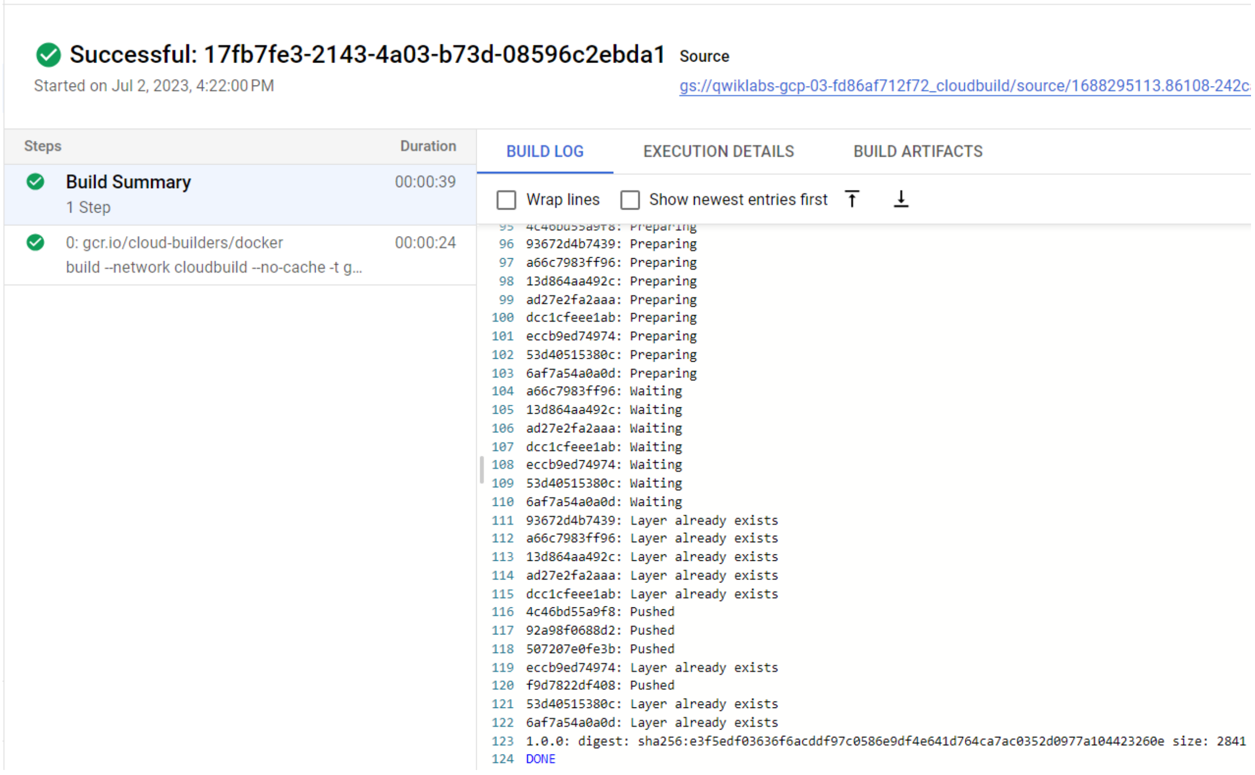
1. First, to make sure you have the Cloud Build API enable, run the following command:

```
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 .
```



Successful: 17fb7fe3-2143-4a03-b73d-08596c2ebda1
Started on Jul 2, 2023, 4:22:00 PM

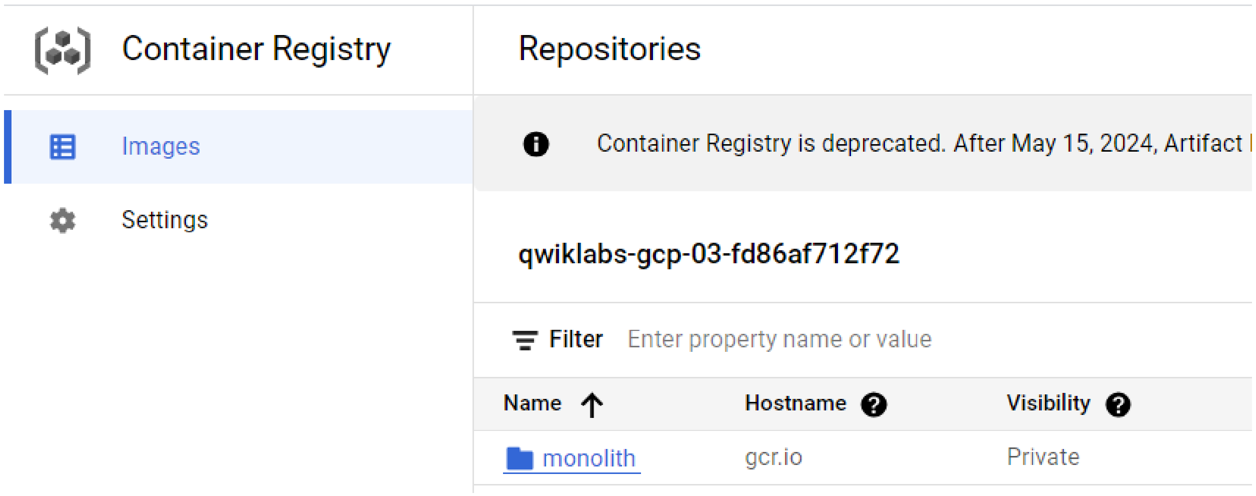
Source
gs://qwiklabs-gcp-03-fd86af712f72_cloudbuild/source/1688295113.86108-242c

Steps	Duration
Build Summary 1 Step	00:00:39
0: gcr.io/cloud-builders/docker build --network cloudbuild --no-cache -t g...	00:00:24

```
4c46bd55a9f8: Preparing
93672d4b7439: Preparing
a66c7983ff96: Preparing
13d864aa492c: Preparing
ad27e2fa2aaa: Preparing
dcc1cfeee1ab: Preparing
eccb9ed74974: Preparing
53d40515380c: Preparing
6af7a54a0a0d: Preparing
a66c7983ff96: Waiting
13d864aa492c: Waiting
ad27e2fa2aaa: Waiting
dcc1cfeee1ab: Waiting
eccb9ed74974: Waiting
53d40515380c: Waiting
6af7a54a0a0d: Waiting
93672d4b7439: Layer already exists
a66c7983ff96: Layer already exists
13d864aa492c: Layer already exists
ad27e2fa2aaa: Layer already exists
dcc1cfeee1ab: Layer already exists
4c46bd55a9f8: Pushed
92a98f0688d2: Pushed
507207e0fe3b: Pushed
eccb9ed74974: Layer already exists
f9d7822df408: Pushed
53d40515380c: Layer already exists
6af7a54a0a0d: Layer already exists
1.0.0: digest: sha256:e3f5edf03636f6acddf97c0586e9df4e641d764ca7ac0352d0977a104423260e size: 2841
DONE
```

4. To view your build history or watch the process in real time by clicking the **Navigation menu** and scrolling down to Tools 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:



Container Registry is deprecated. After May 15, 2024, Artifact

qwiklabs-gcp-03-fd86af712f72

Name	Hostname	Visibility
monolith	gcr.io	Private

Task 4. Deploy container to GKE

Now that you have containerized your website and pushed your container to the Google Container 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.

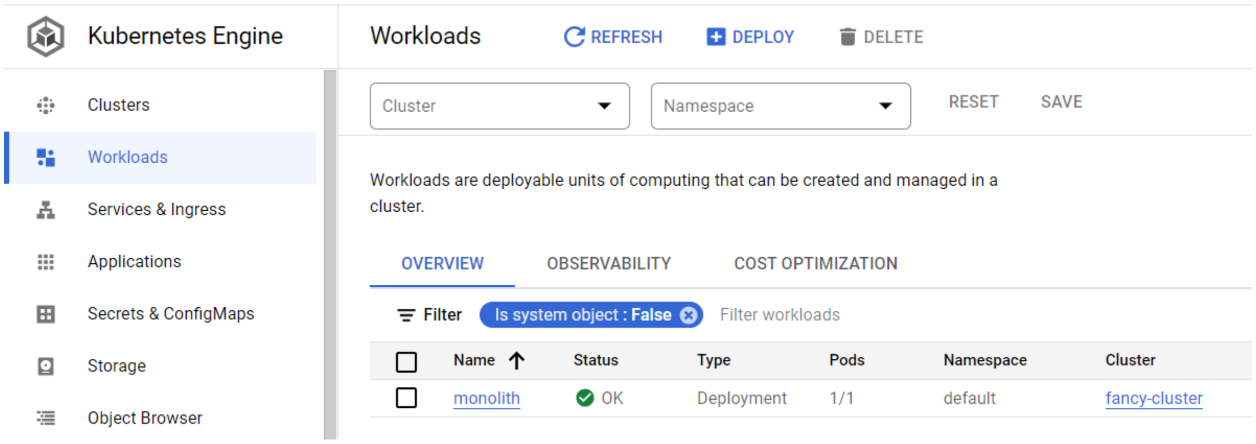
The `kubectl create deployment` command you'll use next causes Kubernetes to create a Deployment named `monolith` on your cluster with **1** replica.

- Run the following command to deploy your application:

```
kubectl create deployment monolith
--image=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:1.0.0
```

Note: As a best practice, using YAML file and a source control system such as GitHub or Cloud Source Repositories is recommended to store those changes. Learn more about these resources from the Deployments documentation.

Verify deployment

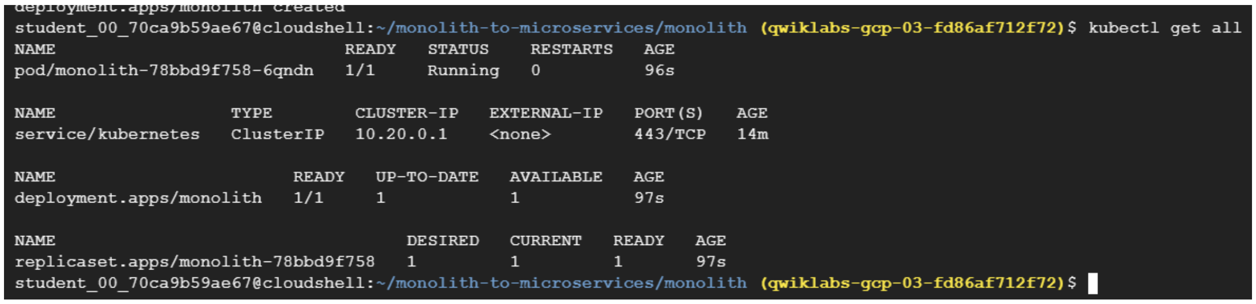


Workloads are deployable units of computing that can be created and managed in a cluster.

Name	Status	Type	Pods	Namespace	Cluster
monolith	OK	Deployment	1/1	default	fancy-cluster

1. Verify the Deployment was created successfully:

```
kubectl get all
```



```
deployment.apps/monolith created
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/monolith-78bbd9f758-6qndn   1/1     Running   0          96s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.20.0.1    <none>        443/TCP   14m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monolith   1/1     1            1           97s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/monolith-78bbd9f758   1         1         1       97s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$
```

This output shows you several things:

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

Note: You can also view your Kubernetes deployments via the Console. **Navigation menu** > **Kubernetes Engine** > **Workloads**.

Note: If you are seeing errors or statuses you do not expect, you can debug your resources with the following commands to see detailed information about them:

```
kubectl describe pod monolith
```

```
kubectl describe pod/monolith-7d8bc7bf68-2bxts
```

```
kubectl describe deployment monolith
```

```
kubectl describe deployment.apps/monolith
```

At the very end of the output, you will see a list of events that give errors and detailed information about your resources.

Show pods

kubectl get pods

Show deployments

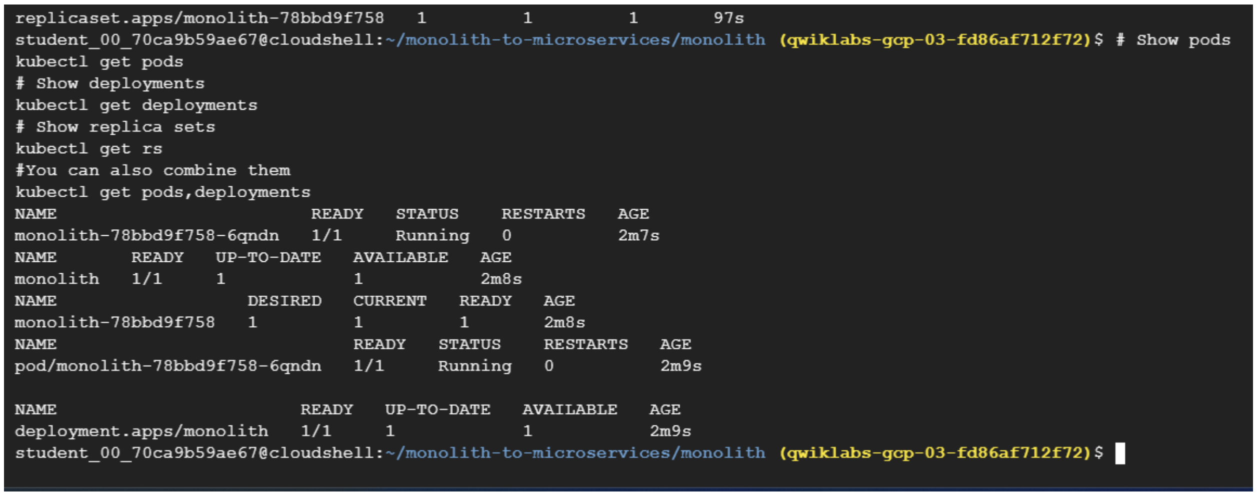
kubectl get deployments

Show replica sets

kubectl get rs

#You can also combine them

kubectl get pods,deployments



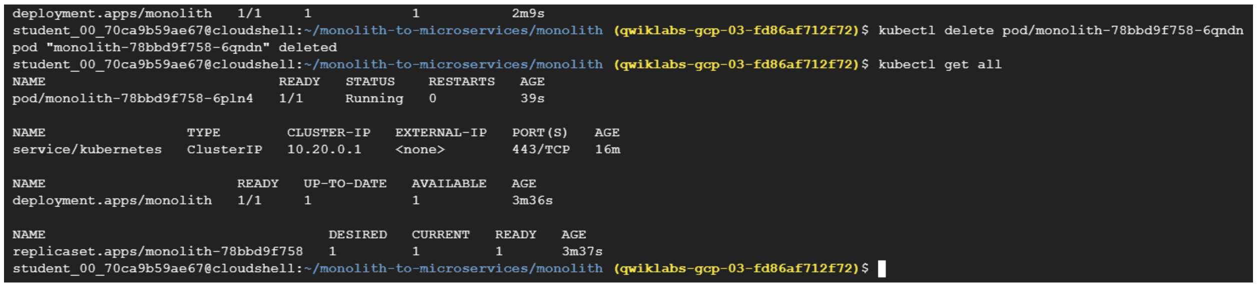
```
replicaset.apps/monolith-78bbd9f758   1         1         1       97s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ # Show pods
kubectl get pods
# Show deployments
kubectl get deployments
# Show replica sets
kubectl get rs
#You can also combine them
kubectl get pods,deployments
NAME                            READY   STATUS    RESTARTS   AGE
monolith-78bbd9f758-6qndn       1/1     Running   0          2m7s
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
monolith   1/1     1            1           2m8s
NAME                     DESIRED   CURRENT   READY   AGE
monolith-78bbd9f758      1         1         1       2m8s
NAME                            READY   STATUS    RESTARTS   AGE
pod/monolith-78bbd9f758-6qndn   1/1     Running   0          2m9s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monolith       1/1     1            1           2m9s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$
```

2. Copy a pod name from the previous command, then use it when you run the following command to delete it:

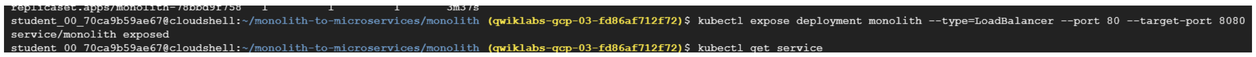
```
kubectl delete pod/<POD_NAME>
```

3. Click on the workload name (it will happen quickly).
4. If you are fast enough, you can run `get all` again, and you should see two pods: one terminating and the other creating or running:

```
kubectl get all
```



```
deployment.apps/monolith   1/1     1            1           2m9s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl delete pod/monolith-78bbd9f758-6qndn
pod "monolith-78bbd9f758-6qndn" deleted
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/monolith-78bbd9f758-6pln4   1/1     Running   0          39s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.20.0.1    <none>        443/TCP   16m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monolith   1/1     1            1           3m36s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/monolith-78bbd9f758   1         1         1       3m37s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$
```



```
replicaset.apps/monolith-78bbd9f758   1         1         1       3m37s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl expose deployment monolith --type=LoadBalancer --port 80 --target-port 8080
service/monolith exposed
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl get service
```

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:

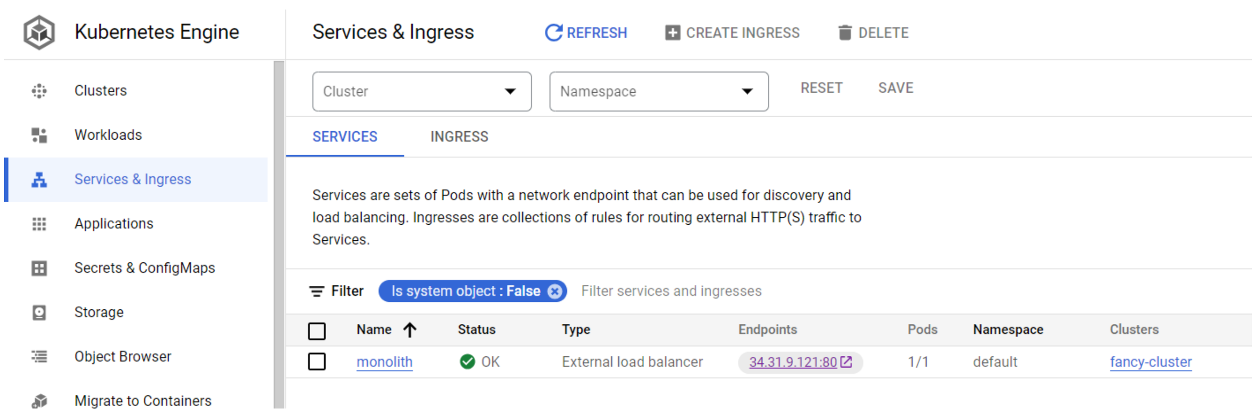
```
kubectl expose deployment monolith --type=LoadBalancer --port 80 --target-port
8080
```

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
```



Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

Name	Status	Type	Endpoints	Pods	Namespace	Clusters
monolith	OK	External load balancer	34.31.9.121:80	1/1	default	fancy-cluster

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.

You should see the same website you tested earlier. You now have your website fully running on Kubernetes!

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.

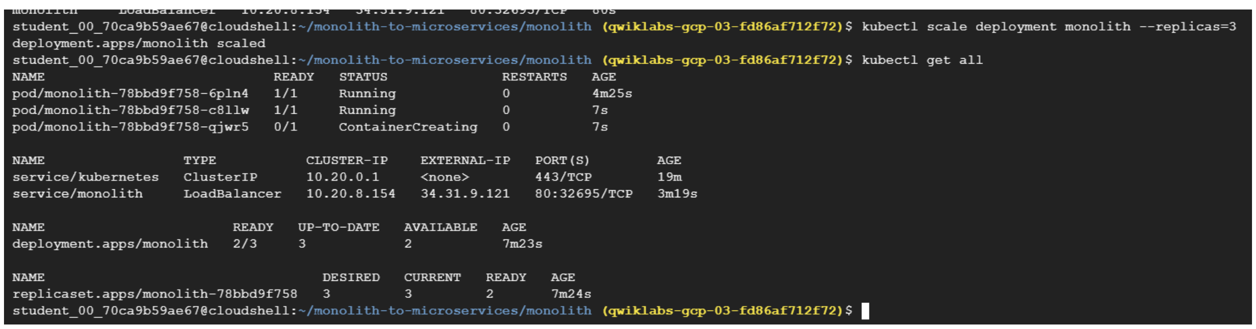
1. In Cloud Shell, run the following command to scale you deployment up to 3 replicas:

```
kubectl scale deployment monolith --replicas=3
```

2. Verify the Deployment was scaled successfully:

```
kubectl get all
```

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



```
monolith     LoadBalancer   10.20.8.154   34.31.9.121   80:32695/TCP   60s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl scale deployment monolith --replicas=3
deployment.apps/monolith scaled
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl get all
NAME                            READY   STATUS              RESTARTS   AGE
pod/monolith-78bbd9f758-6pln4   1/1     Running             0          4m25s
pod/monolith-78bbd9f758-c8llw   1/1     Running             0          7s
pod/monolith-78bbd9f758-qjwr5   0/1     ContainerCreating   0          7s

NAME                 TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP      10.20.0.1     <none>        443/TCP        19m
service/monolith     LoadBalancer   10.20.8.154   34.31.9.121   80:32695/TCP   3m19s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monolith   2/3     3            2           7m23s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/monolith-78bbd9f758   3         3         2       7m24s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$
```

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
```

2. Print its contents to verify the changes:

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

Copied

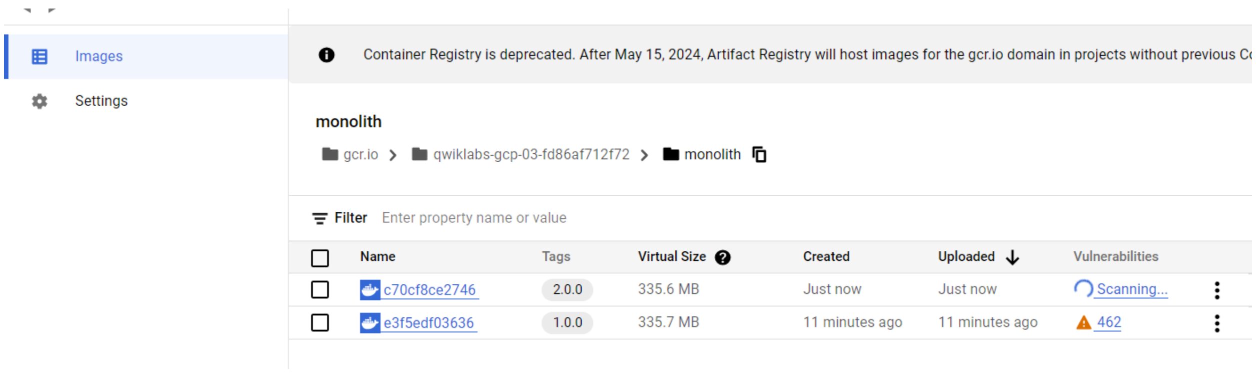
3. Run the following command to build the React app and copy it into the monolith public directory:

```
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 Google Cloud Container 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 .
```


Images

Settings

Container Registry is deprecated. After May 15, 2024, Artifact Registry will host images for the gcr.io domain in projects without previous C

monolith

gcr.io > qwiklabs-gcp-03-fd86af712f72 > monolith

Filter Enter property name or value

Name	Tags	Virtual Size	Created	Uploaded	Vulnerabilities
c70cf8ce2746	2.0.0	335.6 MB	Just now	Just now	Scanning...
e3f5edf03636	1.0.0	335.7 MB	11 minutes ago	11 minutes ago	462

Task 8. Update website with zero downtime

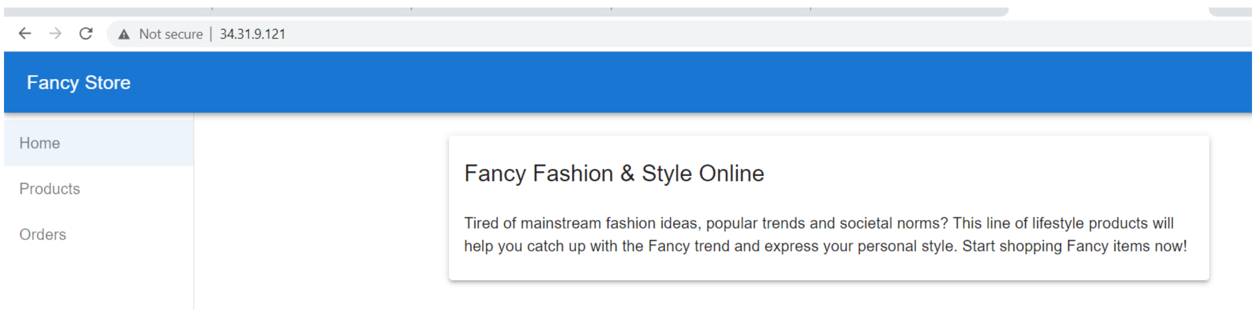
The changes are completed and the marketing team is happy with your updates! It is time to update the website without interruption to the users.

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
```

Verify deployment


Not secure | 34.31.9.121

Fancy Store

Home

Products

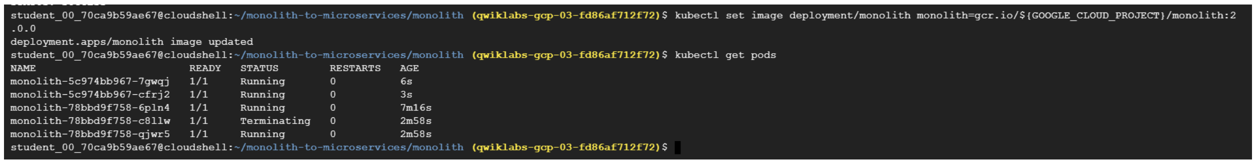
Orders

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!

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

```
kubectl get pods
```



```
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl set image deployment/monolith monolith=gcr.io/${GOOGLE_CLOUD_PROJECT}/monolith:2
.0.0
deployment.apps/monolith image updated
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$ kubectl get pods
NAME                        READY   STATUS        RESTARTS   AGE
monolith-5c974bb967-7gwqj   1/1     Running       0          6s
monolith-5c974bb967-cfrj2   1/1     Running       0          3s
monolith-78bbd9f758-6pln4   1/1     Running       0          7m16s
monolith-78bbd9f758-c8llw   1/1     Terminating   0          2m58s
monolith-78bbd9f758-qjwr5   1/1     Running       0          2m58s
student_00_70ca9b59ae67@cloudshell:~/monolith-to-microservices/monolith (qwiklabs-gcp-03-fd86af712f72)$
```

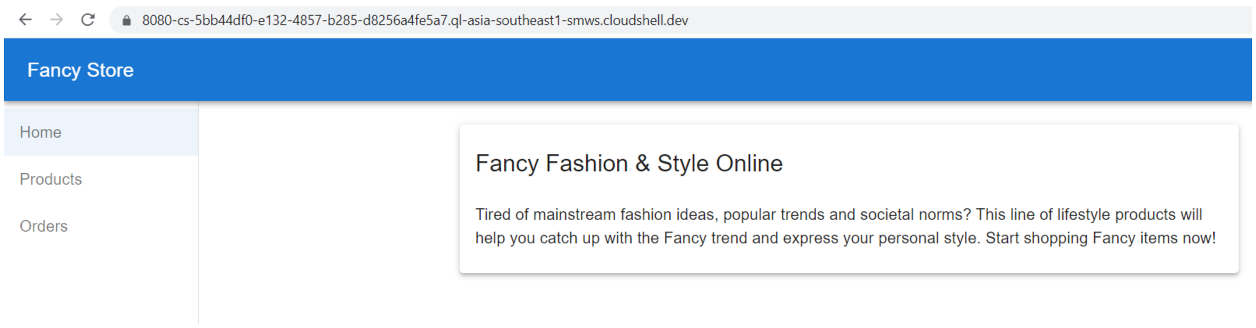
Here you will see 3 new pods being created and your old pods getting shut down. You can tell by the age which are new and which are old. Eventually, you will only see 3 pods again which will be your 3 updated pods.

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

```
npm start
```

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 to the homepage component!



8080-cs-5bb44df0-e132-4857-b285-d8256a4fe5a7.ql-asia-southeast1-smws.cloudshell.dev

Fancy Store

Home

Products

Orders

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!

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
```

2. Delete Google Container 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
```

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

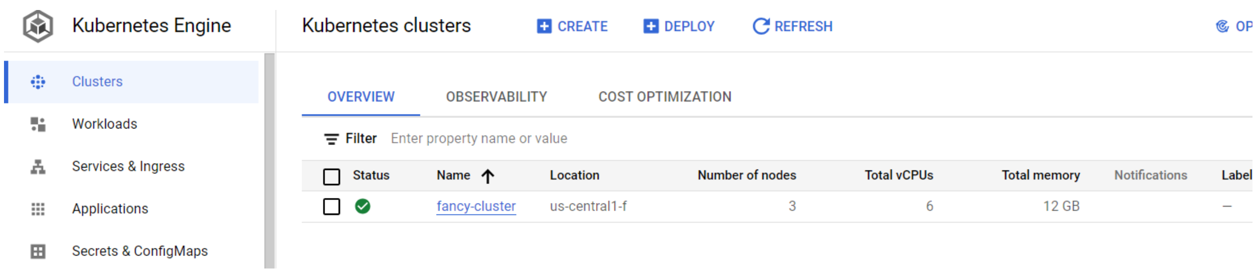
```
# The following command will take all source archives from all builds and
delete them from cloud storage

# 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
```

4. Delete GKE Service:

```
kubectl delete service monolith

kubectl delete deployment monolith
```

5. Delete GKE Cluster:

```
gcloud container clusters delete fancy-cluster
```



Status	Name	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Label
✓	fancy-cluster	us-central1-f	3	6	12 GB		—