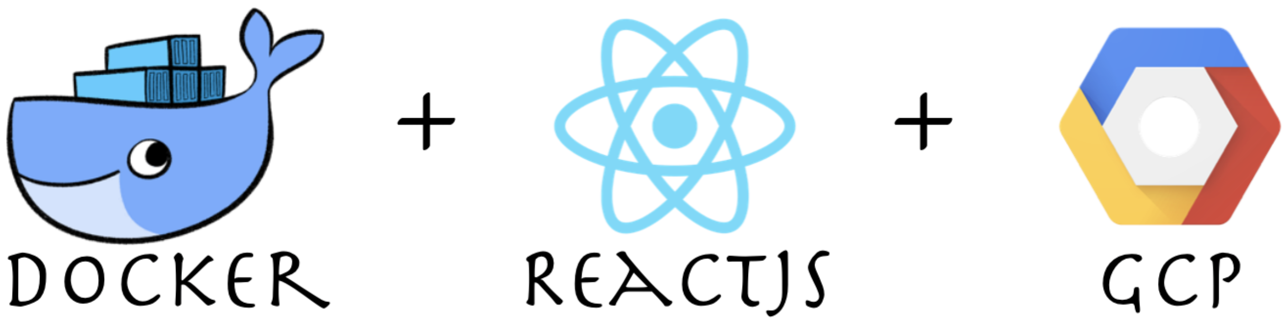
HACKERNOON

Deploy React Application with Docker and Google Cloud Platform

Harsh Makadia Follow
Feb 4, 2019 · 6 min read



Docker | ReactJS | GCP

In this article, you will learn how to deploy applications on GCP. We will deploy a create-react-app.

Link to the Repo — https://github.com/Harshmakadia/react-docker

Before we get started with the actual steps of deploying the React App using GCP and Docker. First, *let's understand what docker actual is?*

Docker is a tool which is designed to make the creating, deploying and running of applications easier with the help of containers. Containers are something which allows the developer to bundle the application with all the necessary ingredients like different libraries, dependencies and ship is as only a single package.

We will go step by step

1. Creating React Application

Create react app is a lot easier using the create-react-app (CRA)

We will use ***create-react-app*** package to install and configure simple react application from NPM, Open your terminal and install react app.

For more info on creating a react app

facebook/create-react-app

Set up a modern web app by running one command. Contribute to facebook/create-react-app development by creating an…

github.com

> *once you run the application using command*
> *$ npm start*
>
> *After that, it's time to create a build the app, run*
> *$ npm run build*

2. Creating minimal Dockerfile

creating a docker file is a cup of your tea. The is no rocket science in creating a docker file.

Just Create a new file with name ***Dockerfile***
Now once the file is created we will add some command to that which will help us to create, run, deploy the application.

Here the content of Dockerfile for react app. Note I'm using ***Nginx*** to as a server.

```
# Use below nginx version
FROM nginx:1.15.2-alpine
# Copy the build folder of the react app
COPY ./build /var/www
# Copy the ngnix configrations
COPY deployments/nginx.conf /etc/nginx/nginx.conf
# Expose it on port 80
```

```
EXPOSE 80
ENTRYPOINT ["nginx","-g","daemon off;"]
```

Dockerfile hosted with ❤ by GitHub view raw

Dockerfile

Once the docker file is created. I'm creating a new folder named **deployment** within the app directory which has a ***nginx.conf*** file

The content of nginx file, note that this is default configuration you may not need to alter this file unless you have some special requirements.

```
# auto detects a good number of processes to run
worker_processes auto;

#Provides the configuration file context in which the directives that affect connection process
events {
    # Sets the maximum number of simultaneous connections that can be opened by a worker process
    worker_connections 8000;
    # Tells the worker to accept multiple connections at a time
    multi_accept on;
}

#add_header X-XSS-Protection "1; mode=block";


http {
    # what times to include
    include       /etc/nginx/mime.types;
    # what is the default one
    default_type  application/octet-stream;

    # Sets the path, format, and configuration for a buffered log write
    log_format compression '$remote_addr - $remote_user [$time_local] '
        '"$request" $status $upstream_addr '
        '"$http_referer" "$http_user_agent"';

    server {
        # listen on port 80
        listen 80;
        # save logs here
        access_log /var/log/nginx/access.log compression;

        # where the root here
        root /var/www;
        # what file to server as index
        index index.html index.htm;

add_header X-Frame-Options "SAMEORIGIN";
add_header X-XSS-Protection "1; mode=block";
add_header X-Content-Type-Options nosniff;
        location / {
            # First attempt to serve request as file, then
            # as directory, then fall back to redirecting to index.html
            try_files $uri $uri/ /index.html;
        }


        etag on;
        gzip on;
        gzip_disable "msie6";

         gzip_vary on;
         gzip_proxied any;
         gzip_comp_level 5;
         gzip_buffers 16 8k;
         gzip_http_version 1.1;
         gzip_disable "MSIE [1-6]\.(?!.*SV1)";
gzip_types
    application/atom+xml
    application/javascript
    application/json
    application/ld+json
    application/manifest+json
    application/rss+xml
    application/vnd.geo+json
    application/vnd.ms-fontobject
    application/x-font-ttf
    application/x-web-app-manifest+json
    application/xhtml+xml
    application/xml
    font/opentype
    image/bmp
    image/svg+xml
    image/x-icon
    text/cache-manifest
```

```
        text/css
        text/plain
        text/vcard
        text/vnd.rim.location.xloc
        text/vtt
        text/x-component
        text/x-cross-domain-policy;


            # Media: images, icons, video, audio, HTC
            location ~* \.(?:jpg|jpeg|gif|png|ico|cur|gz|svg|svgz|mp4|ogg|ogv|webm|htc)$ {
              expires 1M;
              access_log off;
              add_header Cache-Control "public";
            }

            # Javascript and CSS files
            location ~* \.(?:css|js)$ {
                try_files $uri =404;
                expires 1y;
                access_log off;
                add_header Cache-Control "public";
            }

            # Any route containing a file extension (e.g. /devicesfile.js)
            location ~ ^.+\..+$ {
                try_files $uri =404;
            }
        }
    }
```

nginx.conf hosted with ❤ by GitHub view raw

nginx.conf

3. Installing Docker on your machine

Head to this link below and download it for your respective operating system

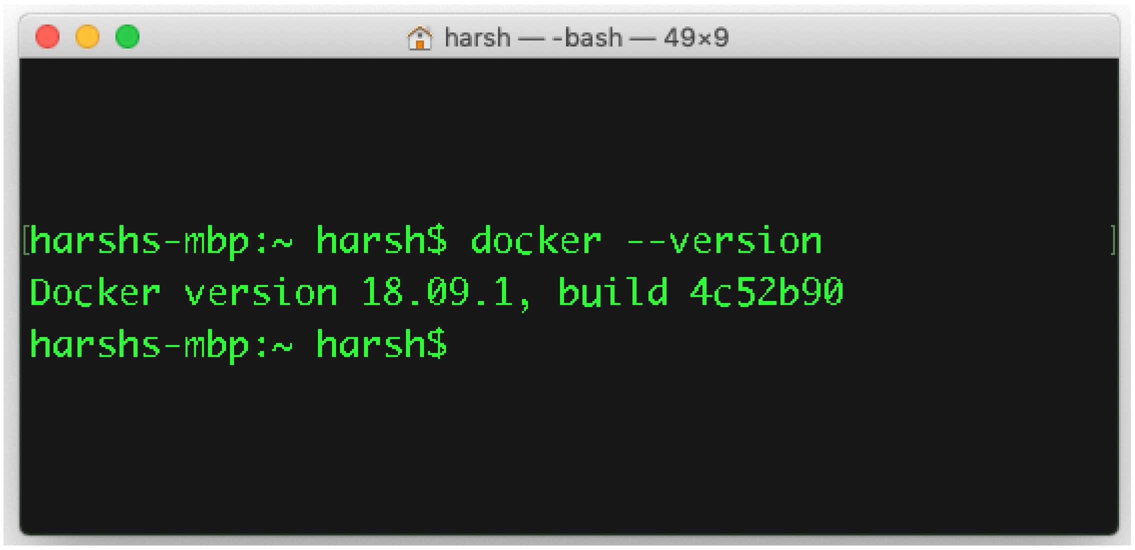
Docker Desktop

Docker Desktop Enterprise is a new commercial desktop offering that gives you everything you need for enterprise-ready…

www.docker.com

Once it is installed run open your terminal and run below command to check it is installed successfully

> *docker — — version*



```
harshs-mbp:~ harsh$ docker --version
Docker version 18.09.1, build 4c52b90
harshs-mbp:~ harsh$
```

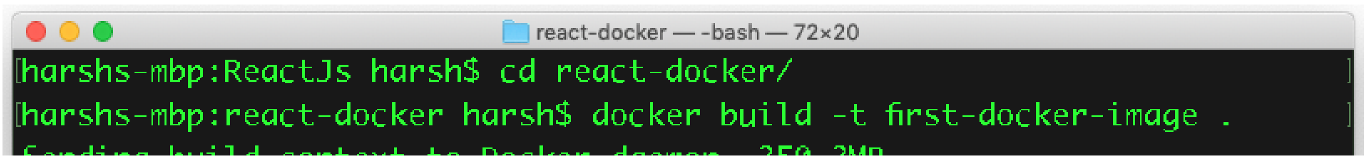
Docker version

Now that we have the docker setup on our machine it's time to create the first image using the following command

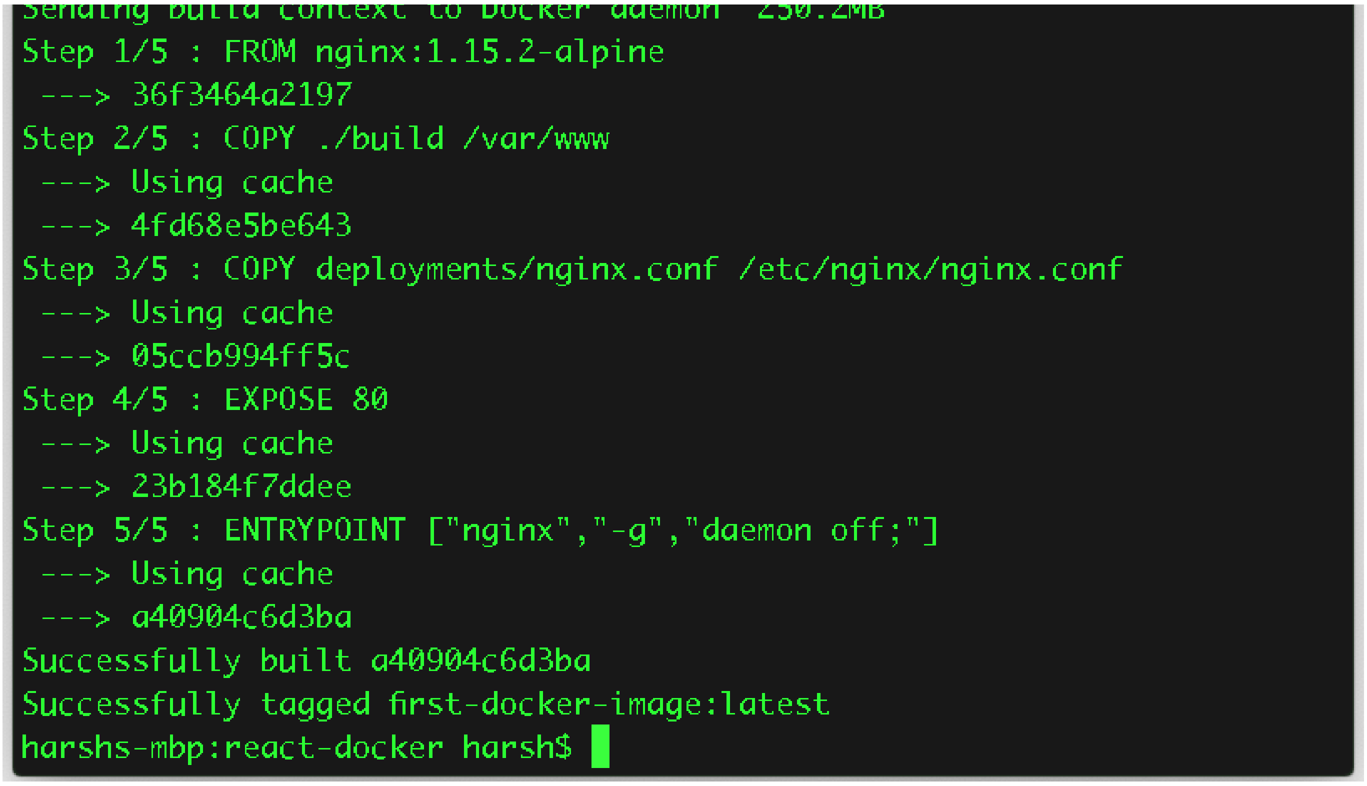
> *docker build -t first-docker .*

more info about different command can be found here.

Once you run this command it will execute all the command listed down in the Dockerfile.



```
harshs-mbp:ReactJs harsh$ cd react-docker/
harshs-mbp:react-docker harsh$ docker build -t first-docker-image .
```



```
Sending build context to Docker daemon  250.2MB
Step 1/5 : FROM nginx:1.15.2-alpine
 ---> 36f3464a2197
Step 2/5 : COPY ./build /var/www
 ---> Using cache
 ---> 4fd68e5be643
Step 3/5 : COPY deployments/nginx.conf /etc/nginx/nginx.conf
 ---> Using cache
 ---> 05ccb994ff5c
Step 4/5 : EXPOSE 80
 ---> Using cache
 ---> 23b184f7ddee
Step 5/5 : ENTRYPOINT ["nginx","-g","daemon off;"]
 ---> Using cache
 ---> a40904c6d3ba
Successfully built a40904c6d3ba
Successfully tagged first-docker-image:latest
harshs-mbp:react-docker harsh$
```

Creating Image

we have successfully created the image. Let's proceed to next step

4. Make use of gcloud SDK

Download SDK from the below link and setup on your machine

Cloud SDK | Cloud SDK | Google Cloud

A collection of command line tools for the Google Cloud Platform. Includes gcloud, bq, gsutil and other important…

cloud.google.com

Now that we have the gcloud SDK setup on our machine

- — Create a new project in GCP

Next step is to create a new project in the GCP where we will be pushing our docker images to the containers.

Configure Docker to use `gcloud` as a credential helper or are using another authentication method. To use `gcloud` as the credential helper, run the command:

- `gcloud auth configure-docker`

It's time to push the image to the registry

1. Tag the local image with the registry name by using the command:

- `docker tag [SOURCE_IMAGE] [HOSTNAME]/[PROJECT-ID]/[IMAGE]`

1. where `[SOURCE_IMAGE]` is the local image name.
2. This command names the image with the registry name and applies the tag `latest`. If you want to apply a different tag, then use the command:

- `docker tag [SOURCE_IMAGE] [HOSTNAME]/[PROJECT-ID]/[IMAGE]:[TAG]`

Push the tagged image to Container Registry

Push the tagged image to Container Registry by using the command:

```
docker push [HOSTNAME]/[PROJECT-ID]/[IMAGE]
```

This command pushes the image that has the tag `latest`. If you want to push an image that has a different tag, use the command:

```
docker push [HOSTNAME]/[PROJECT-ID]/[IMAGE]:[TAG]
```

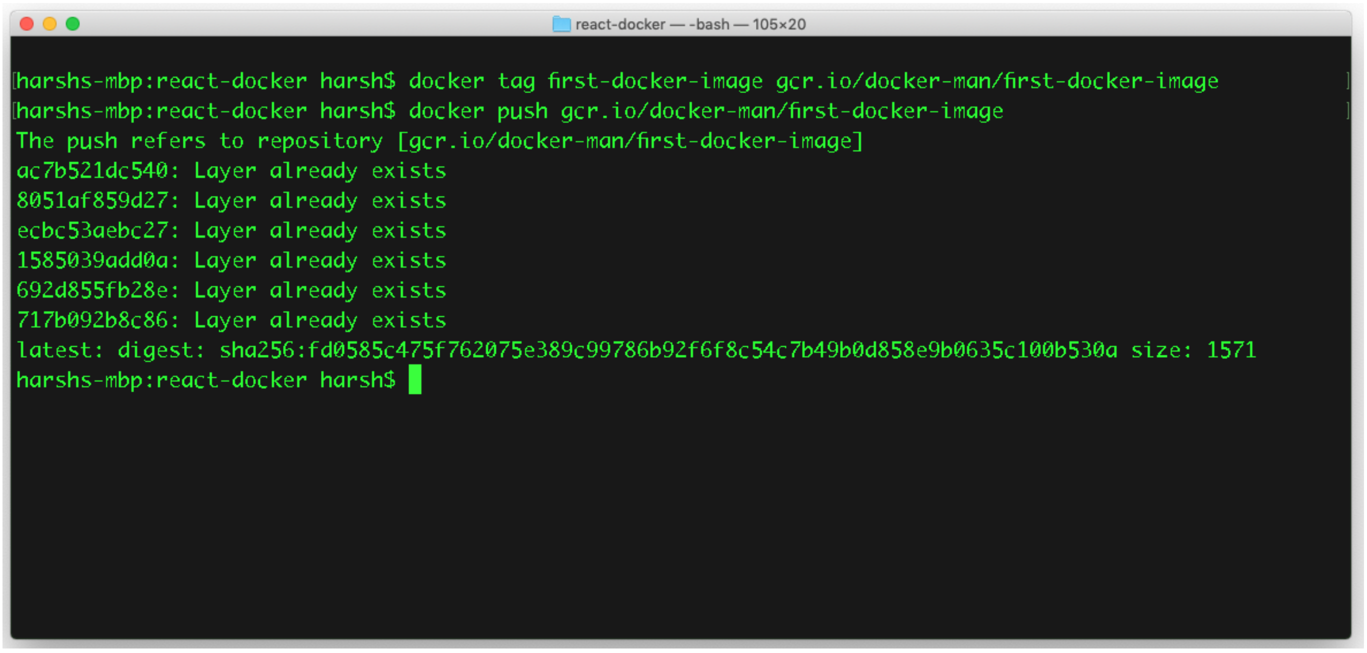
When you push an image to a registry with a new hostname, Container Registry creates a storage bucket in the specifiedmulti-regional location. After pushing your image, you can:

- Go to the GCP Console to view the registry and image.

- Run `gcloud container images list-tags` to view the image's tag(s) and automatically-generated digest:
- `gcloud container images list-tags [HOSTNAME]/[PROJECT-ID]/[IMAGE]`
- The command's output is similar to the following:
- `DIGEST TAGS TIMESTAMP`
  `44bde... test 2017-..-..`

Where

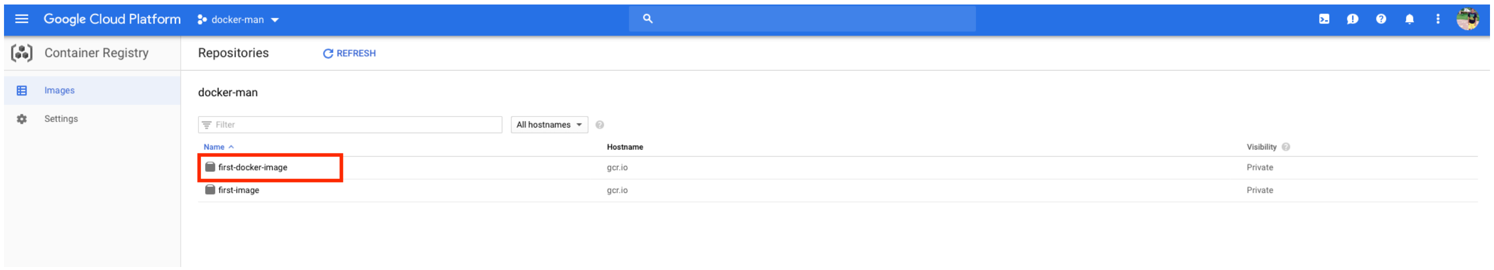
- `[HOSTNAME]` is listed under **Location** in the console. It's one of four options: `gcr.io`, `us.gcr.io`, `eu.gcr.io`, or `asia.gcr.io`.
- `[PROJECT-ID]` is your Google Cloud Platform Console project ID. See Domain-scoped projects for how to work with projects IDs that include a domain.
- `[IMAGE]` is the image's name in Container Registry.
- `[TAG]` is the tag applied to the image. In a registry, tags are unique to an image.



```
harshs-mbp:react-docker harsh$ docker tag first-docker-image gcr.io/docker-man/first-docker-image
harshs-mbp:react-docker harsh$ docker push gcr.io/docker-man/first-docker-image
The push refers to repository [gcr.io/docker-man/first-docker-image]
ac7b521dc540: Layer already exists
8051af859d27: Layer already exists
ecbc53aebc27: Layer already exists
1585039add0a: Layer already exists
692d855fb28e: Layer already exists
717b092b8c86: Layer already exists
latest: digest: sha256:fd0585c475f762075e389c99786b92f6f8c54c7b49b0d858e9b0635c100b530a size: 1571
harshs-mbp:react-docker harsh$
```

Pushing Image

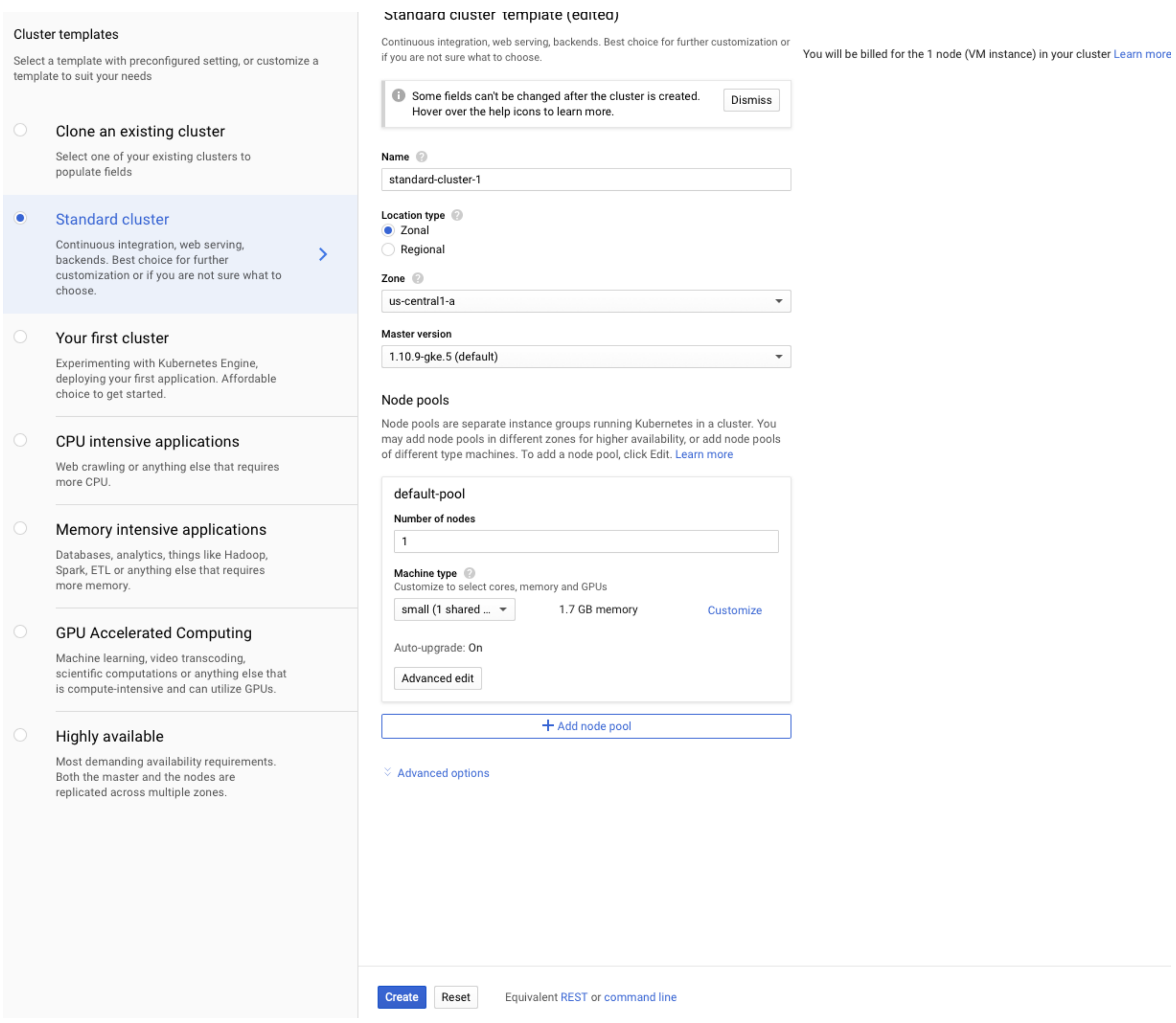
Navigate the GCP console and search of ***Container Registry,*** you will be able to see the image which we push.



docker-man

Name	Hostname	Visibility
first-docker-image	gcr.io	Private
first-image	gcr.io	Private

It's time to create cluster now inside ***Kubernetes Engine*** in GCP



Cluster templates

Select a template with preconfigured setting, or customize a template to suit your needs

- Clone an existing cluster — Select one of your existing clusters to populate fields
- [✓ Standard cluster] — Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.
- Your first cluster — Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.
- CPU intensive applications — Web crawling or anything else that requires more CPU.
- Memory intensive applications — Databases, analytics, things like Hadoop, Spark, ETL or anything else that requires more memory.
- GPU Accelerated Computing — Machine learning, video transcoding, scientific computations or anything else that is compute-intensive and can utilize GPUs.
- Highly available — Most demanding availability requirements. Both the master and the nodes are replicated across multiple zones.

Standard cluster template (edited)

Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.

You will be billed for the 1 node (VM instance) in your cluster Learn more

Some fields can't be changed after the cluster is created. Hover over the help icons to learn more.

Name: standard-cluster-1

Location type: [✓ Zonal] Regional

Zone: us-central1-a

Master version: 1.10.9-gke.5 (default)

Node pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit. Learn more

default-pool

Number of nodes: 1

Machine type: Customize to select cores, memory and GPUs

small (1 shared ... 1.7 GB memory

Auto-upgrade: On

Equivalent REST or command line

Creating cluster

Create deployment under workloads

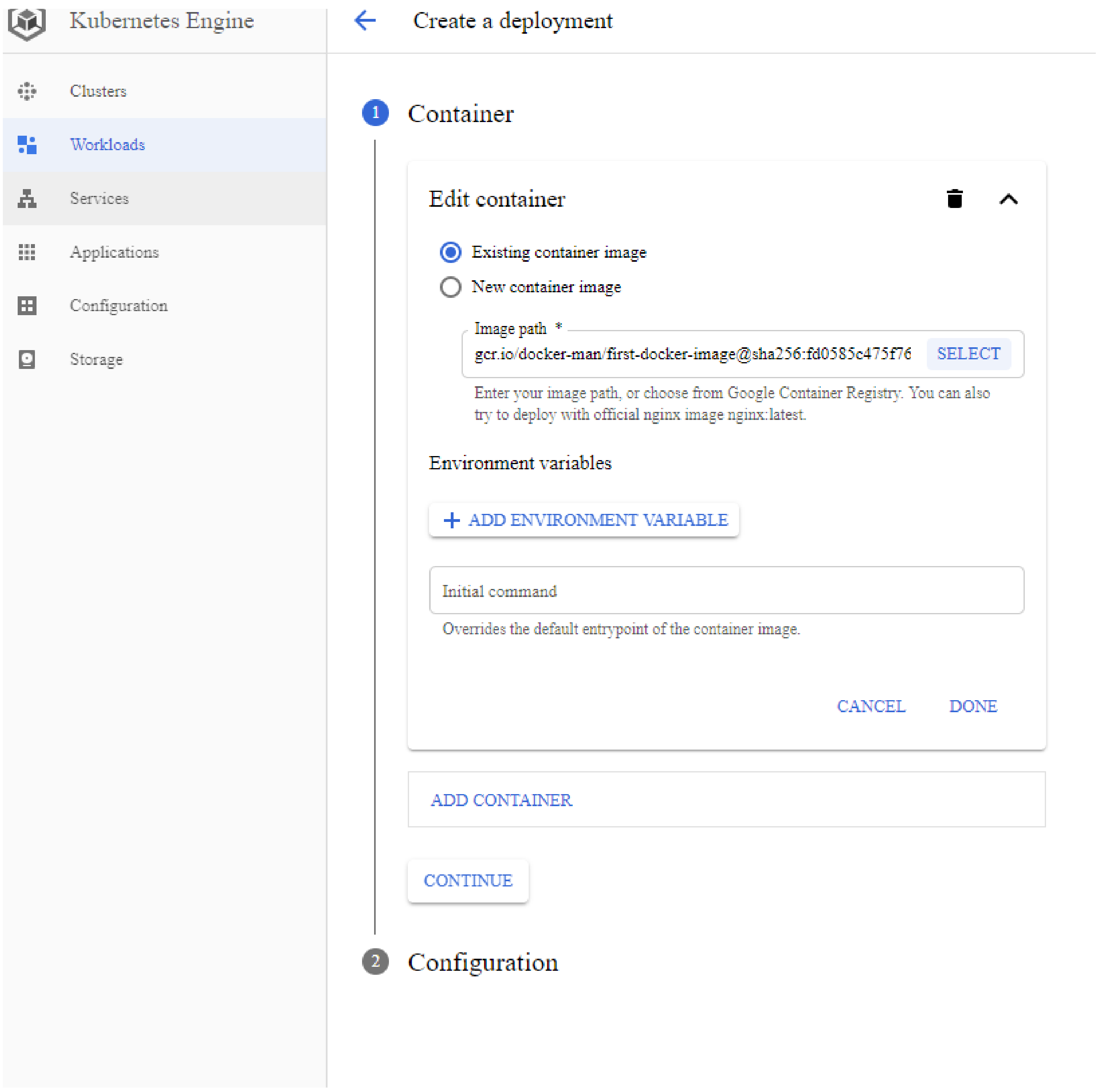
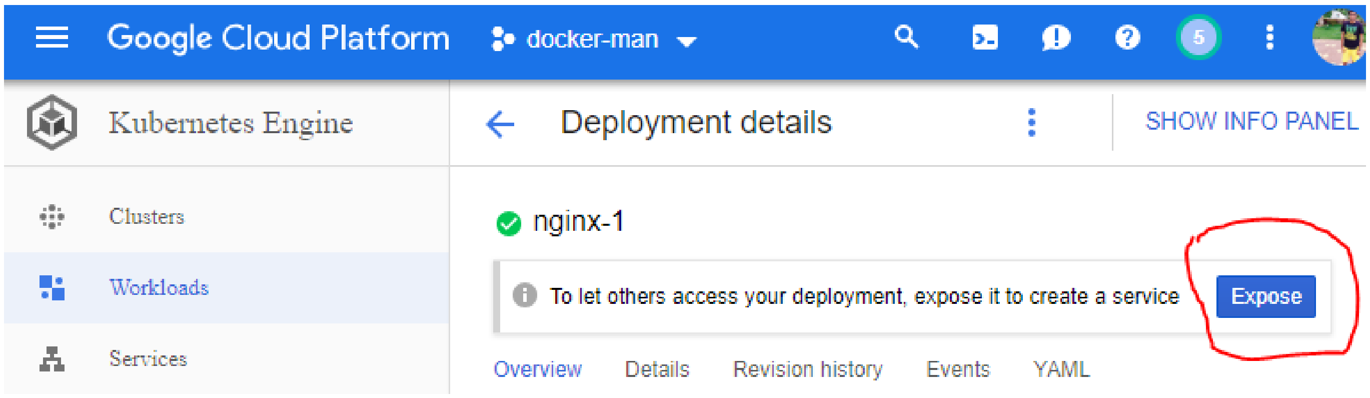


Image path: gcr.io/docker-man/first-docker-image@sha256:fd0585c475f76

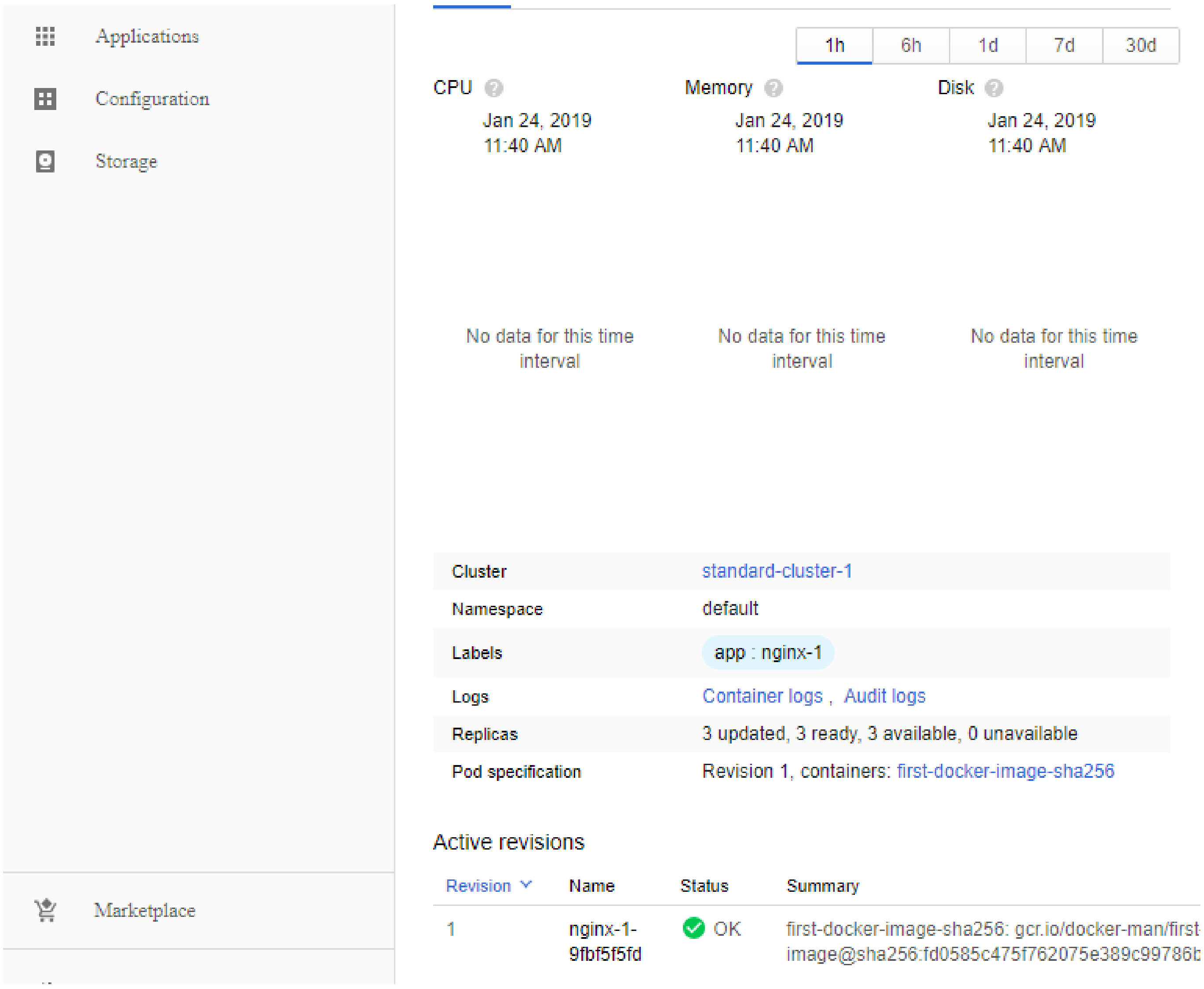
Enter your image path, or choose from Google Container Registry. You can also try to deploy with official nginx image nginx:latest.

Overrides the default entrypoint of the container image.

Select the image which you want to deploy and finally click on ***Expose*** to expose the deployment.



To let others access your deployment, expose it to create a service



CPU Jan 24, 2019 11:40 AM — No data for this time interval

Memory Jan 24, 2019 11:40 AM — No data for this time interval

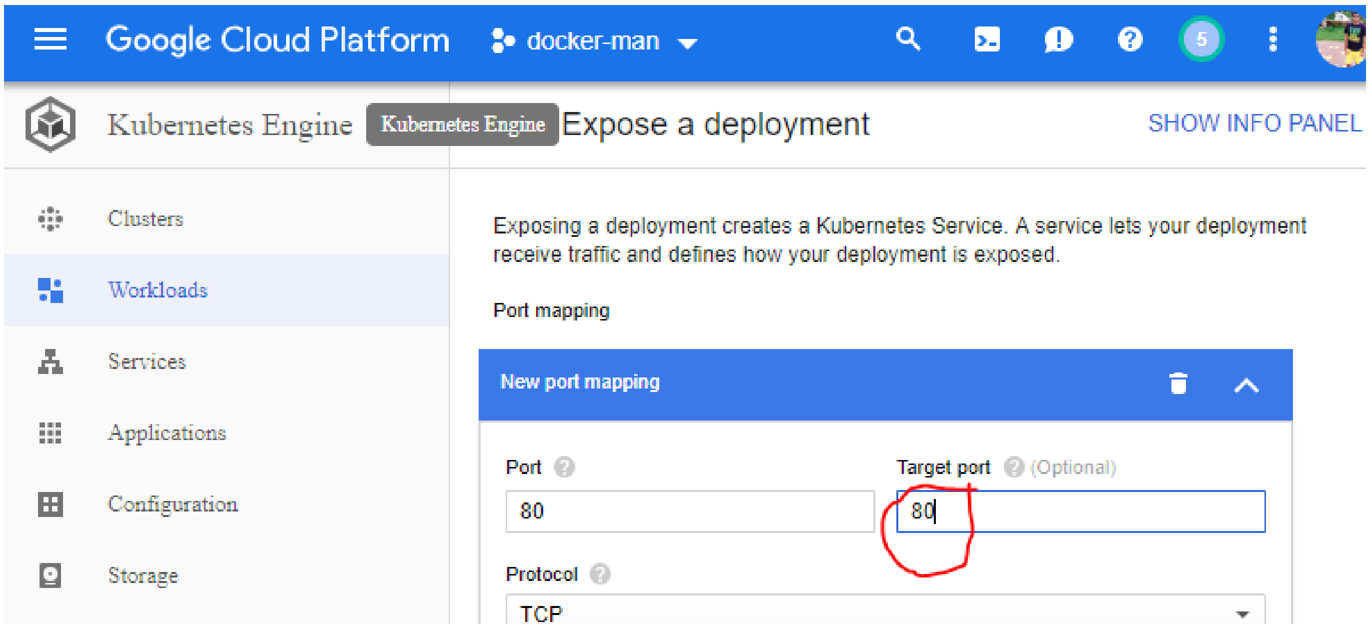
Disk Jan 24, 2019 11:40 AM — No data for this time interval

Cluster	standard-cluster-1
Namespace	default
Labels	app : nginx-1
Logs	Container logs , Audit logs
Replicas	3 updated, 3 ready, 3 available, 0 unavailable
Pod specification	Revision 1, containers: first-docker-image-sha256

Active revisions

Revision	Name	Status	Summary
1	nginx-1-9fbf5f5fd	✓ OK	first-docker-image-sha256: gcr.io/docker-man/first image@sha256:fd0585c475f762075e389c99786b

Set ***port to 80 in Target Port*** since we had EXPOSED our application to 80 in the Dockerfile



Expose a deployment

Exposing a deployment creates a Kubernetes Service. A service lets your deployment receive traffic and defines how your deployment is exposed.

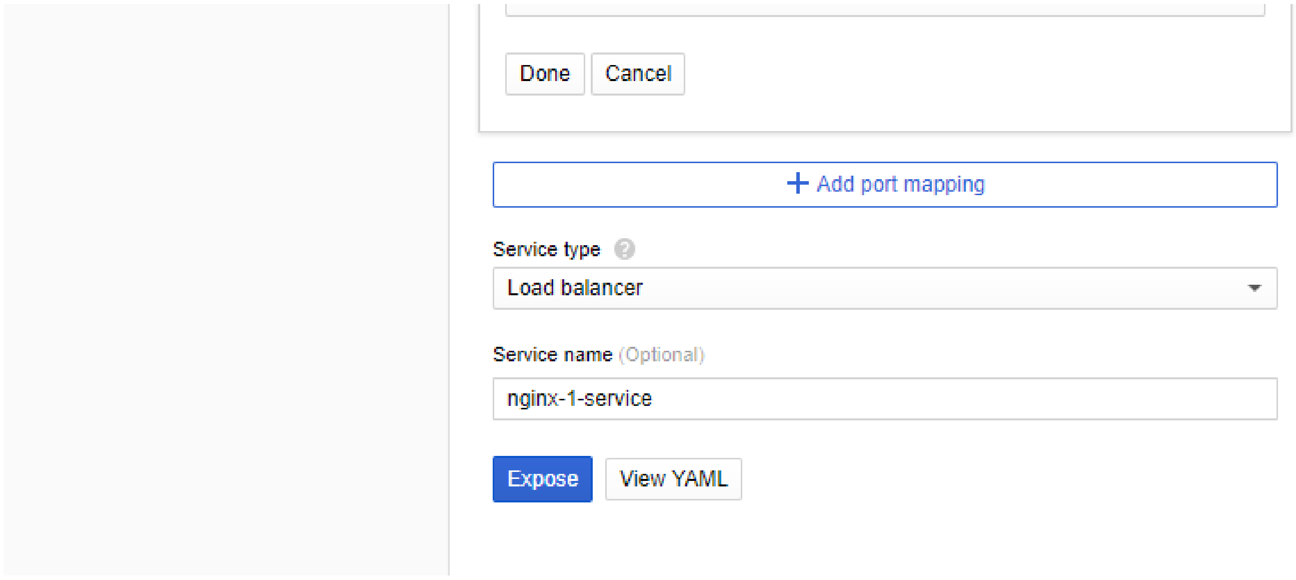
Port mapping

New port mapping

Port: 80

Target port (Optional): 80

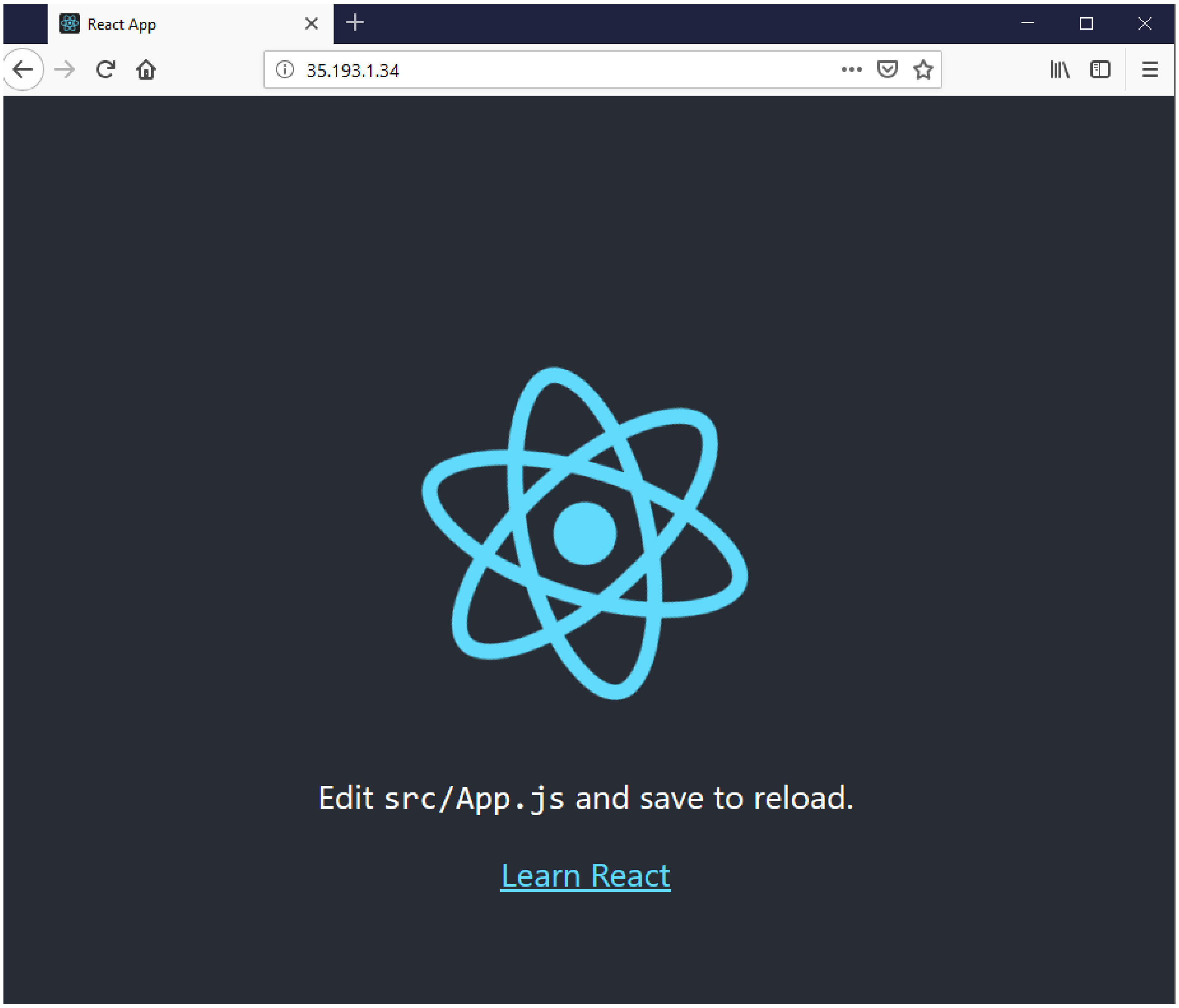
Protocol: TCP



Service type: Load balancer

Service name (Optional): nginx-1-service

Once you have exposed the port you will get IP Adress where your react application will be running live.

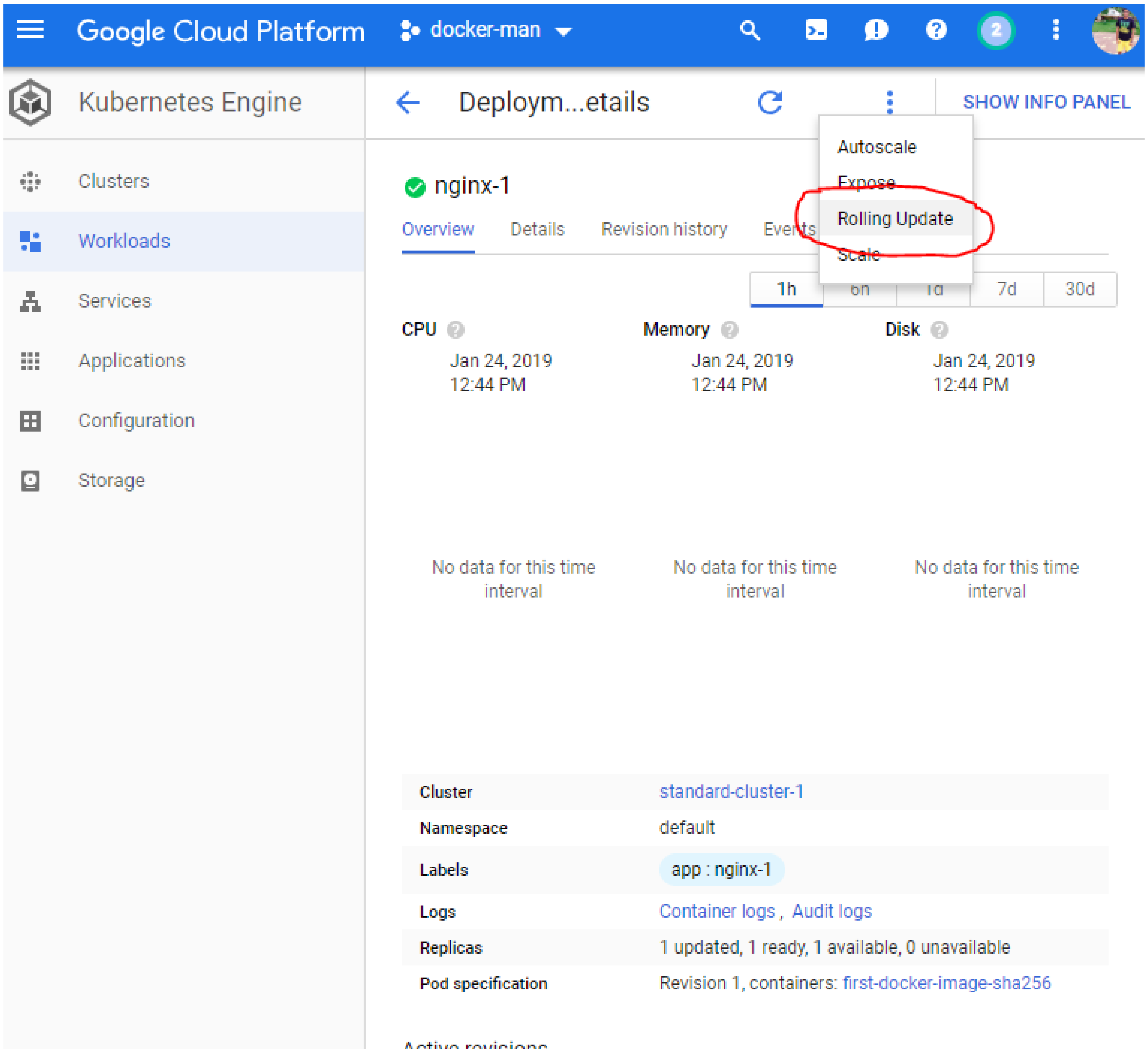


Edit `src/App.js` and save to reload.

Learn React

Deployed Application

And that it's you are all set with docker. Whenever you want to push new image to the container first build the image with above-specified commands and then push the image to container registry and finally make that image live by going to rolling update option.



CPU
Jan 24, 2019
12:44 PM

Memory
Jan 24, 2019
12:44 PM

Disk
Jan 24, 2019
12:44 PM

No data for this time interval

No data for this time interval

No data for this time interval

Cluster	standard-cluster-1
Namespace	default
Labels	app : nginx-1
Logs	Container logs , Audit logs
Replicas	1 updated, 1 ready, 1 available, 0 unavailable
Pod specification	Revision 1, containers: first-docker-image-sha256

Please note down your questions in the comment section below if you have any doubts & I'll be happy to address them.

how hackers start their afternoon. the real shit is on hackernoon.com. Learn more

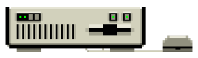
Get this newsletter

Emails will be sent to teals.alan.cooper@gmail.com. Not you?

Docker Reactjs Google Cloud Platform Nginx Kubernetes Engine

That’s the end 🔚 I hope you have learned something new.

Happy Learning! 💻 😃



Hackernoon Newsletter curates great stories by real tech professionals

Get solid gold sent to your inbox. Every week!

Email

First Name | Last Name

Sign Up

☐ If you are ok with us sending you updates via email, please tick the box. Unsubscribe whenever you want.

Terms of Service

☐ I agree to leave Formed on Upscribe mit this information, which will be collected and used according to Upscribe's

Sign up for DONT SIGN UP FOR THIS NEWSLETTER

By HackerNoon.com