

Deployment to Google Cloud

1. Create a Dockerfile containing the following:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["todo-list-api.csproj", "."]
RUN dotnet restore "./todo-list-api.csproj"
COPY . .
WORKDIR "/src/"
RUN dotnet build "todo-list-api.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "todo-list-api.csproj" -c Release -o /app/publish
/p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "todo-list-api.dll"]
```

*** The below will explain the dockerfile in further detail by each line:**

- 1.1. Setting up a .NET image with version 6
- 1.2. Set the working directory
- 1.3. Expose the 80 port with 443 for http and https respectively
- 1.4. Copy the project, add to src directory and run build in Release mode in the /build directory
- 1.5. Create a publish version from build
- 1.6. Copy the contents to the Docker image directory

2. Create a cloudbuild.yaml file

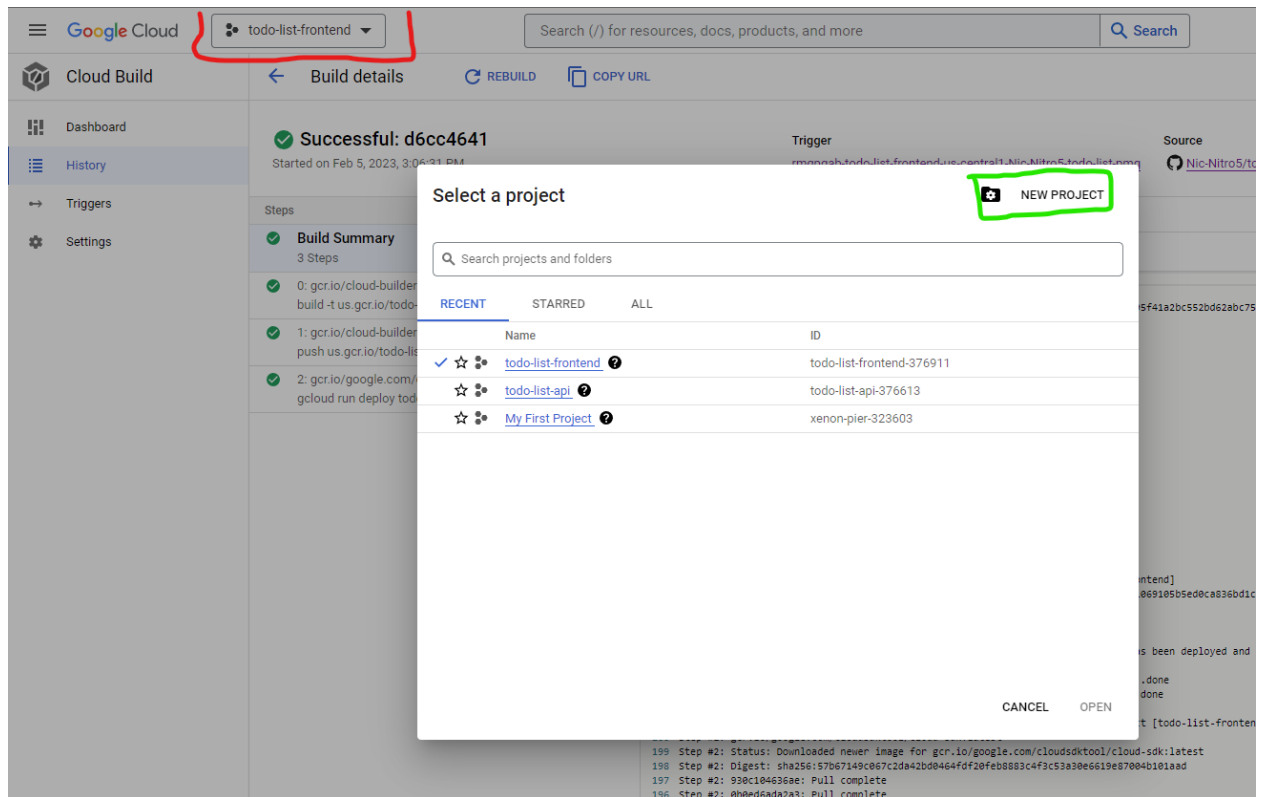
```
steps:
  - name: gcr.io/cloud-builders/docker
    args:
      - build
      - '--no-cache'
      - '-t'
      - '$_GCR_HOSTNAME/$PROJECT_ID/$_SERVICE_NAME:$COMMIT_SHA'
      - '.'
      - '-f'
      - Dockerfile
    id: Build
  - name: gcr.io/cloud-builders/docker
    args:
      - push
      - '$_GCR_HOSTNAME/$PROJECT_ID/$_SERVICE_NAME:$COMMIT_SHA'
    id: Push
  - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk:slim'
    args:
      - run
      - deploy
      - $_SERVICE_NAME
      - '--image'
      - '$_GCR_HOSTNAME/$PROJECT_ID/$_SERVICE_NAME:$COMMIT_SHA'
      - '--region'
      - $_DEPLOY_REGION
    entrypoint: gcloud
    id: Deploy
timeout: 1200s
images:
  - '$_GCR_HOSTNAME/$PROJECT_ID/$_SERVICE_NAME:$COMMIT_SHA'
```

2.1. This file tells cloudbuild to use a docker image followed by running cloud build. It's going to give it a tag name using solution variables.

2.2. We use docker to push to that location.

2.3. We use the cloudbuild sdk to deploy this.

3. Create a Google Cloud user and create a new project



3.1. The red outline is where to create a new project.

3.2. Click on the green outlined area at the top right of the popup to create the new project

4. Enable these APIs and services

- Cloud SQL and API
- Cloud Build API
- Cloud Run API
- Secret Management API
- Compute API

4.1 Navigate to SQL in the sidebar

- Create a new instance and select PostgreSQL
- Enable the API
- Add a name and password
- Select your version
- Here you can configure according to your requirements
- Select the smaller size of the machine should you wish to reduce costs

[←](#) Edit todo-instance

Instance info

Instance ID
todo-instance

▼ PASSWORD POLICY

Database version
PostgreSQL 14

This instance has the latest supported version

Choose region and zonal availability

For better performance, keep your data close to the services that need it. Region is permanent, while zone can be changed any time.

Region

us-central1 (Iowa)

Zonal availability

☒ Single zone
In case of outage, no failover. Not recommended for production.

☐ Multiple zones (Highly available)
Automatic failover to another zone within your selected region. Recommended for production instances. Increases cost.

▼ SPECIFY ZONES

Customize your instance

You can also customize instance configurations later

Machine type

Machine type is db-f1-micro.

Storage

Storage type is SSD. Storage size is 10 GB, and will automatically scale as needed.

Summary

Region	us-central1 (Iowa)
DB Version	PostgreSQL 14.4
vCPUs	1 vCPU
Memory	628.74 MB
Storage	10 GB
Network throughput (MB/s) ?	125 of 125
Disk throughput (MB/s) ?	Read: 4.8 of 125.0 Write: 4.8 of 37.9
IOPS ?	Read: 300 of 15,000 Write: 300 of 4,500
Connections	Public IP
Backup	Automated
Availability	Single zone
Point-in-time recovery	Enabled

Now viewing project "todo-list-api" in organization "No organization"



- Enable public IP access in the connections tab and create instance

Storage

Storage type is SSD. Storage size is 10 GB, and will automatically scale as needed.



Connections

Public IP enabled



Data Protection

Automatic backups enabled. Point-in-time recovery (via write-ahead logs) enabled.



Instance deletion protection enabled.

Maintenance

Updates may occur any day of the week. Cloud SQL chooses the maintenance timing.



Flags

No flags set.



Query insights

Query insights disabled



Labels

No labels set



 **HIDE CONFIGURATION OPTIONS**

SAVE

CANCEL

Now viewing proj

- Add the IP address to Authorized networks should you wish to connect to the DB from a local machine

Authorized networks

You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. [Learn more](#)

Nicholas (197.185.114.176)



4.2. Create a bucket (Cloud Storage)

- Navigate to Cloud Storage in the sidebar on the left and click on create a bucket

- **Name your bucket**

Pick a globally unique, permanent name. [Naming guidelines](#)

Ex. 'example', 'example_bucket-1', or 'example.com'

Tip: Don't include any sensitive information

▼ LABELS (OPTIONAL)

CONTINUE

- **Choose where to store your data**

Location: us (multiple regions in United States)

Location type: Multi-region

- **Choose a storage class for your data**

Default storage class: Standard

- **Choose how to control access to objects**

Public access prevention: On

Access control: Uniform

- **Choose how to protect object data**

Protection tools: None

Data encryption: Google-managed key

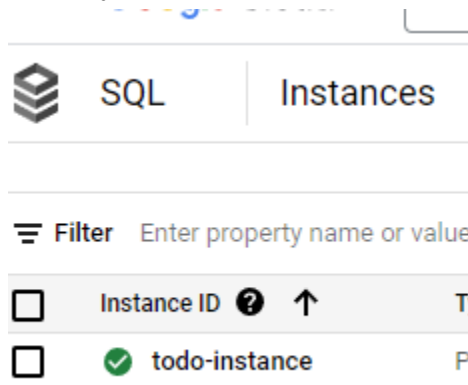
CREATE

CANCEL

- Give the bucket a name
- Select your region
- Choose a storage class
- Access control as required
- Protect the data as required
- Create the bucket

5. Migrations for Database

- Navigate to the SQL tab and select your created database instance. Here you will get your database details (host, user etc)



- **We will run migration scripts to update the cloud database**
 - Connect the cloud database in your IDE (appsettings.json) and an SQL client.
 - Run in the package manager console:
 - dotnet update-database
 - The database should reflect as below

All instances > todo-instance

✓ **todo-instance**

PostgreSQL 14


[+ CREATE DATABASE](#)

Name ↑	Collation	Character set	
postgres	en_US.UTF8	UTF8	⋮
todo-list	en_US.UTF8	UTF8	⋮

5.1. Security Manager

- Navigate to Security -> Secret Manager in the left sidebar
- Here we will need a secret for the bucket, connection string to db and a gcp auth file
- Values here need to be relative

Secret Manager [+ CREATE SECRET](#)

 Try accessing secrets in the IDE using Cloud Code. [Learn more](#)


SECRETS

LOGS

Secret Manager lets you store, manage, and secure access to your application secrets.

[Learn more](#)

 **Filter** Enter property name or value

<input type="checkbox"/>	Name 	Location	Encryption
<input type="checkbox"/>	GCPStorageAuthFile	Automatically replicated	Google-managed
<input type="checkbox"/>	GoogleCloudStorageBucketName	Automatically replicated	Google-managed
<input type="checkbox"/>	TodoListConnectionString	Automatically replicated	Google-managed

No secrets selected

6. Navigate to Cloud Run to create a service

6.1. Click on the sidebar and select Cloud Run

6.2. Click on Create Service

[+ CREATE SERVICE](#)

The new service tab will be opened as seen below:

A service exposes a unique endpoint and automatically scales the underlying infrastructure to handle incoming requests. Service name and region cannot be changed later.

- ☐ Deploy one revision from an existing container image
- ☒ Continuously deploy new revisions from a source repository

[SET UP WITH CLOUD BUILD](#)

Service name *

Service name is required

Region *
us-central1 (Iowa)

[How to pick a region?](#)

CPU allocation and pricing

- ☒ CPU is only allocated during request processing
You are charged per request and only when the container instance processes a request.
- ☐ CPU is always allocated
You are charged for the entire lifecycle of the container instance.

Autoscaling

Minimum number of instances * 0
Maximum number of instances 100

Set to 1 to reduce cold starts. [Learn more](#)

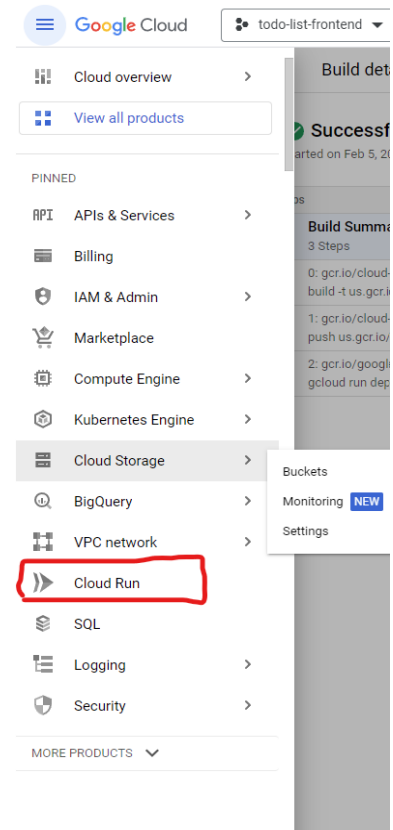
- ☐ Internal
Allow traffic from VPCs and certain Google Cloud services in your project, Shared VPC, internal HTTP(S) load balancer, and traffic allowed by VPC service controls. [Learn more](#)
- ☒ All
Allow direct access to your service from the internet

6.3. Select the continuously deploy new revisions option and link up the required repository by clicking on the setup build with cloud build, here you will configure which repo, branch and type of deployment you want to set up.

Choose the Build type as Dockerfile and save.

6.4. Allocate CPU as required.

6.5. Allow direct access to your service from the internet.



Set up with Cloud Build

With continuous deployment powered by Cloud Build, changes to your source repository are automatically built into container images in Container Registry and deployed to Cloud Run.

Your code should listen for HTTP requests on \$PORT. Your repository must include a Dockerfile or source code in Go, Node.js, Python, Java, .NET Core or Ruby in order to be built into a container image.

1 Source repository

2 Build Configuration

Branch *
*/main\$

Use a regular expression to match to a specific branch [Learn more](#)

Matches the branch: main

Container, Networking, Security

CONTAINER

NETWORKING

SECURITY

General

Container port
8080

Requests will be sent to the container on this port. We recommend listening on \$PORT instead of this specific number.

6.6. Allow unauthenticated invocations. Check this if you are creating a public API or website.

6.7. Set container **port to 80**

6.8. Should you wish you can check the Startup CPU boost option

6.9. Configure the request timeout and max requests per container as required.

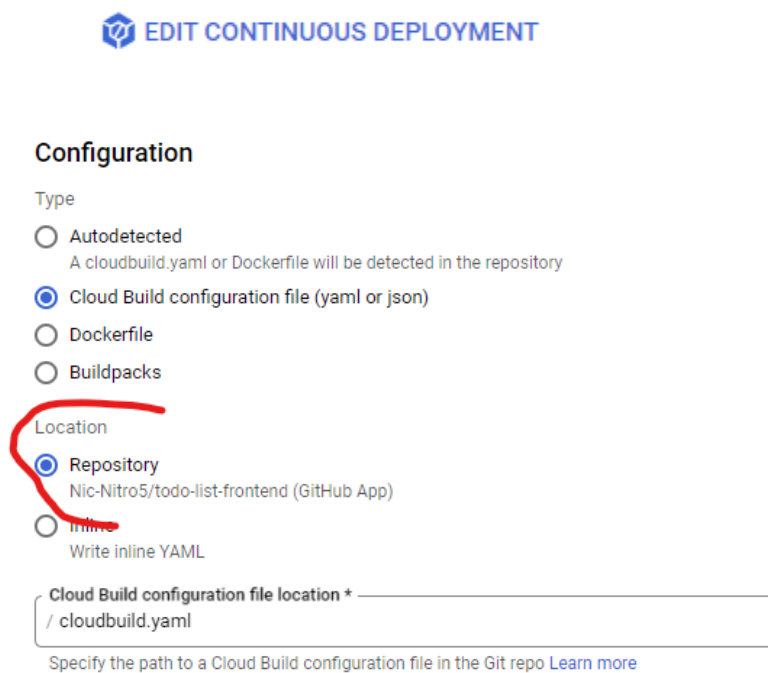
6.10. The networking and security can be configured as required.


6.11. Add your Database connection (Cloud SQL connections)

6.12. Reference and authorize all created secrets

- Expose these as environment variables
- Choose the version to be the latest

7. Edit the continuous deployment



 EDIT CONTINUOUS DEPLOYMENT

Configuration

Type

☐ Autodetected
A cloudbuild.yaml or Dockerfile will be detected in the repository

☒ Cloud Build configuration file (yaml or json)

☐ Dockerfile

☐ Buildpacks

Location

☒ Repository
Nic-Nitro5/todo-list-frontend (GitHub App)

☐ Inline
Write inline YAML

Cloud Build configuration file location *
/ cloudbuild.yaml

Specify the path to a Cloud Build configuration file in the Git repo [Learn more](#)

Here we need to choose the location as Repository and set the path to our cloudbuild.yaml file. We will now have a deployment run every time we push to the branch we configure (main).

Upon successful deployment, you will now have access to the live URL.