Week 16 Day 2: Infra Implementation

Mostly about the implementation with Infrastructure as a Service.

Lessons

Part 1: Concept

- Designing the infra we want to deploy to
- Understanding the comparison and usage of GCP

Part 2: Deployment

- Deploying to Google Cloud
- Using Compute Engine and Artifact Registry with Docker

✋ **Any questions?**

✅ Part 1: Concept

Deployment to a cloud computing service can have numerous ways. What's shown here is just one way to to it. Feel free to adjust along the way if you feel adventurous.

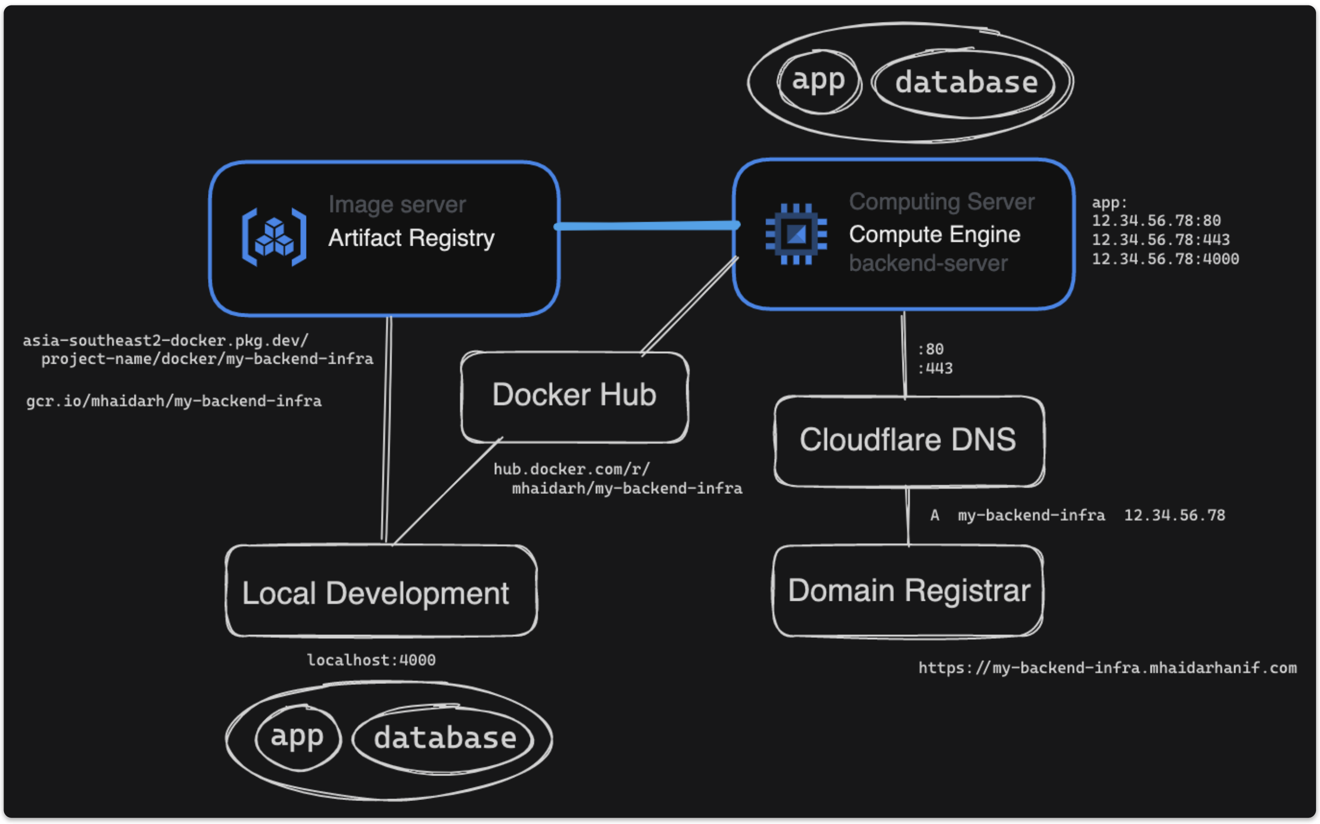
Creating Architecture Diagram

Let's create a diagram that show the expectation of our deployment architecture on the infrastructure. Imagine this is like copying a certain part of what existing PaaS already done automatically before.

What we need and want to do:

- The app will accessible at https://app.example.com
    - Notice the HTTPS with port `443`, and no development port
- Serving the app on a long lived server connection
- Using Docker image and container

This is our architecture plan:



What we mainly need to deploy our app:

- Container image registry → To store Docker images
    - Google Artifact Registry
    - Docker Hub
- Computing server → That host and run Docker containers
    - Google Compute Engine

References:

- https://googlecloudcheatsheet.withgoogle.com/architecture

Comparison and Usage

We can compare the difference between GCP, AWS, and Azure offerings. You can filter with keywords, such as a service type, capability, or product name. We also need to consider the usage of free tier.

We might have some minimum or limited spec we can use with when starting for free. Or if there are balance credits, we can use as much as we need but probably with limted time.

Here, we're going to focus on what Google Cloud offers.

References:

- https://cloud.google.com/docs/get-started/aws-azure-gcp-service-comparison
- https://cloud.google.com/free/docs/free-cloud-features#free-tier-usage-limits

Google Cloud SDK or gcloud CLI

The Google Cloud CLI is a set of tools that you can use to manage resources and applications hosted on Google Cloud.

These tools include the `gcloud`, `gsutil`, and `bq` command-line tools. You can also check the cheat sheet.

Inside Google Cloud Console, there's also the the Cloud Shell which allows you to run the commands in the browser, even including SSH access to your services. Follow the documentation to install it based on your OS.

References:

- https://cloud.google.com/sdk/docs
- https://cloud.google.com/sdk/docs/downloads-interactive
- https://cloud.google.com/sdk/auth_success

General

The installer lets you download, install, and set up the latest version of Google Cloud CLI in an interactive mode.

```
curl https://sdk.cloud.google.com | bash
```

Mac

```
brew install --cask google-cloud-sdk
```

Linux Ubuntu

```
snap install google-cloud-cli --classic
```

Windows

Check the Google Cloud CLI installer.

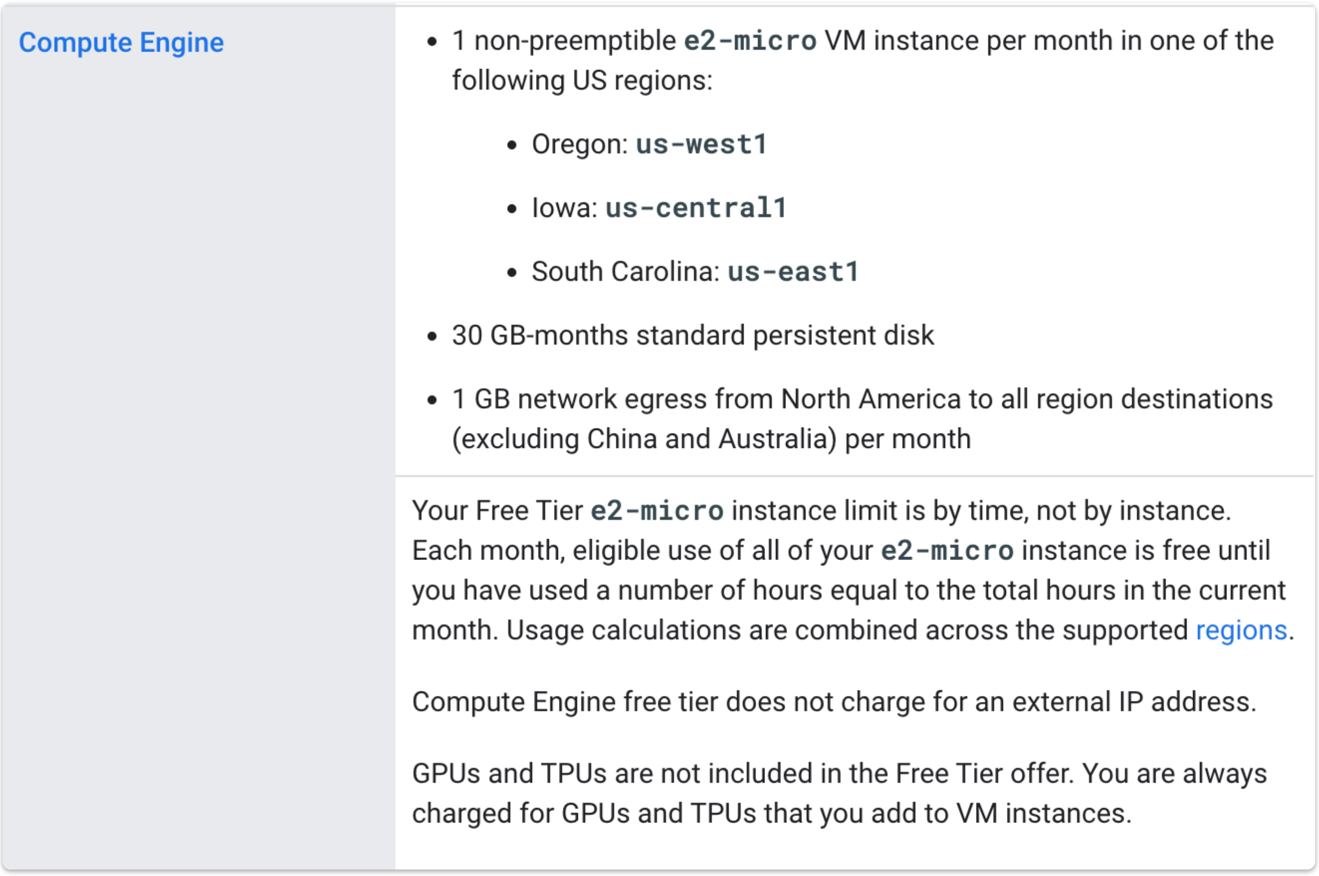
Setup the SDK CLI

The `gcloud` CLI manages authentication, local configuration, developer workflow, and interactions with the Google Cloud APIs.

```sh
gcloud version
# Google Cloud SDK 431.0.0

gcloud init
# Pick cloud project to use

gcloud info

gcloud help
```

Google Compute Engine

The IaaS of GPC which is built on the global infrastructure that runs Google's search engine, Gmail, YouTube and other services. Google Compute Engine enables users to launch virtual machines on demand.



Compute Engine	• 1 non-preemptible **`e2-micro`** VM instance per month in one of the following US regions: • Oregon: **`us-west1`** • Iowa: **`us-central1`** • South Carolina: **`us-east1`** • 30 GB-months standard persistent disk • 1 GB network egress from North America to all region destinations (excluding China and Australia) per month
	Your Free Tier **`e2-micro`** instance limit is by time, not by instance. Each month, eligible use of all of your **`e2-micro`** instance is free until you have used a number of hours equal to the total hours in the current month. Usage calculations are combined across the supported regions. Compute Engine free tier does not charge for an external IP address. GPUs and TPUs are not included in the Free Tier offer. You are always charged for GPUs and TPUs that you add to VM instances.

- 1 non-preemptible e2-micro VM instance per month in one of the following US regions: us-west1, us-central1, us-east1
  - Preemptible instances are VM instances that can be created at a much lower price than a regular instance. Non-preemptible means the average cost.
- 30 GB-months standard persistent disk

- 5 GB-month snapshot storage in the following regions: us-west1, us-central1, us-east1, asia-east1, europe-west1
- 1 GB network egress from North America to all region destinations (excluding China and Australia) per month

Free Tier e2-micro / f1-micro instance limit is by time, not by instance. Each month, eligible use of all of your e2-micro instance is free until you have used a number of hours equal to the total hours in the current month. Usage calculations are combined across the supported regions.

Google Cloud Run

We can use Docker in here.

- 2 million requests per month
- 360,000 GB-seconds of memory, 180,000 vCPU-seconds of compute time
- 1 GB network egress from North America per month

Google Cloud Functions

We can only use and trigger an app functions in here.

- 2 million invocations per month (includes both background and HTTP invocations)
- 400,000 GB-seconds, 200,000 GHz-seconds of compute time
- 5 GB network egress per month

Google Artifact Registry (Container Registry)

Container Registry is being discontinued starting May 15, 2024, and replaced by Artifact Registry to provide an enhancement on the capabilities and performance of the platform.

Artifact means the byproduct of what we develop. Compiled backend app from TypeScript and Docker image are just few of many artifacts we can build and run.

References:

- https://cloud.google.com/blog/products/application-development/understanding-artifact-registry-vs-container-registry
- https://cloud.google.com/artifact-registry/docs/docker

Firewall Rules

Firewall rules in GCP are security policies that control inbound/ingress and outbound/egress network traffic to and from virtual machine instances. Allowing or blocking specific protocols, ports, and IP addresses for enhanced security and network access management.

- Inbound: the service receive a request from the external (users)
- Outbound: the service send a request to the external (users)

Virtual Private Cloud (VPC)

A virtual network that closely resembles a traditional network that you'd operate in your own data center.

IAM and Access Controls

Identity and Access Management (IAM) lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage resources centrally. Then all API and services access can be managed with various control, like project owner, editor, and viewer.

Deployment Strategies

Deployment to a VM without container

1. SSH into the VM
2. Install Node.js and npm
3. Clone app repo
4. Run the app
5. Configure network (port, inbound/outbound)
6. Connect to domain via DNS
7. Issuing an SSL certificate for HTTPS

Deployment to a VM with container

This the simpler way and very similar with what we had with PaaS.

1. Setup Docker image for your repo
2. Push the Docker image to the Container/Artifact Registry
3. Setup the VM to run the Docker image as containers
4. Deploy or run the container in the VM
5. Connect to domain via DNS
6. Issuing an SSL certificate for HTTPS

✅ Part 2: Deployment

Implementation

Let's setup and deploy the backend service as stated before, using a compute service and registry.

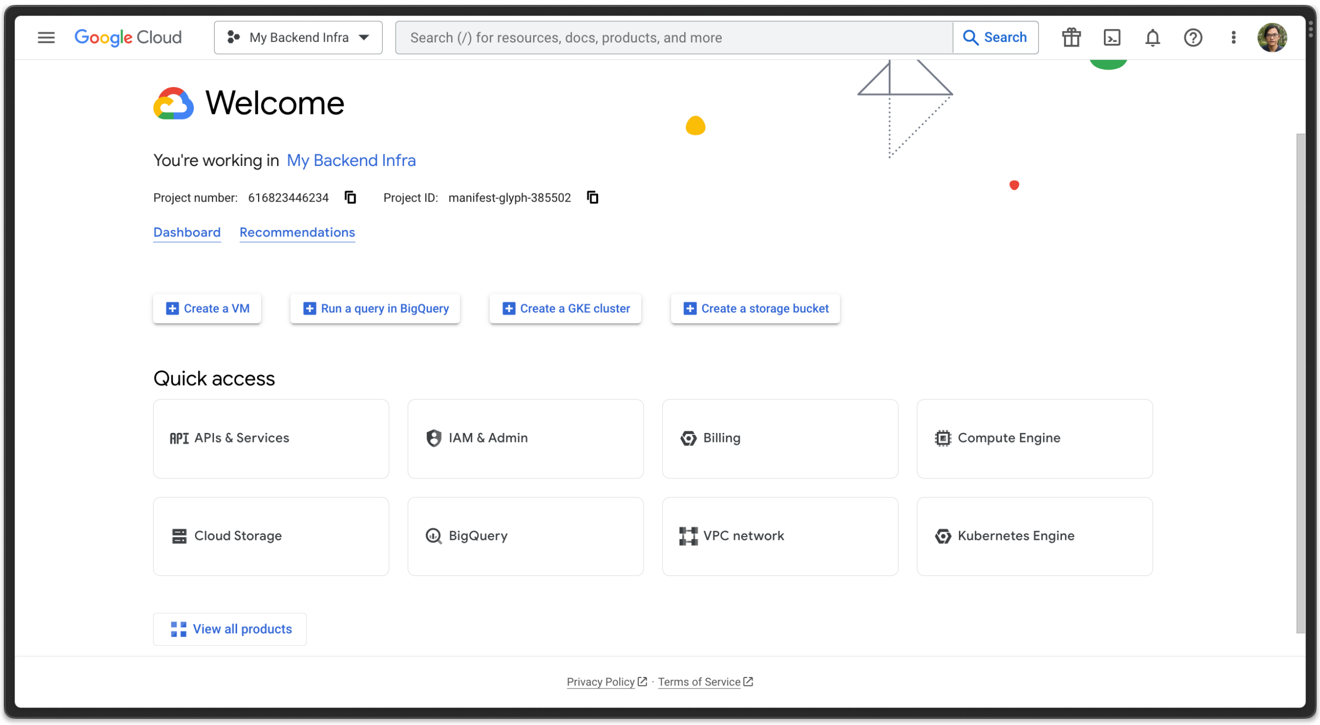
Summarized steps:

1. Create Google Cloud Platform (GCP) account
2. Fill out the billing details
3. Get welcomed to the Console
4. Create or change the project settings
5. Setup the billing/cost limit
6. Install and login to Google Cloud SDK CLI or `gcloud`
7. Enable required APIs:
   - Compute Engine API
   - Artifact Registry API
8. Create an Artifact Registry repo
   - Similar with Docker Hub repo but like a folder
   - Require to select format and a location
9. Create a Compute Engine VM
   - Configure the computing spec
   - Configure firewall rules
10. Push a Docker image to the repo
11. Setup Docker and Docker Compose in the VM
12. Setup `docker-compose.yaml` and `.env`
13. Run the containers
14. Setup the DNS record
15. Access the final deployed service!

Registration

Go to https://cloud.google.com to register and validate your account with billing details.

We'll be greeted with this welcome page and the default project has been created with particular Project ID:



Welcome

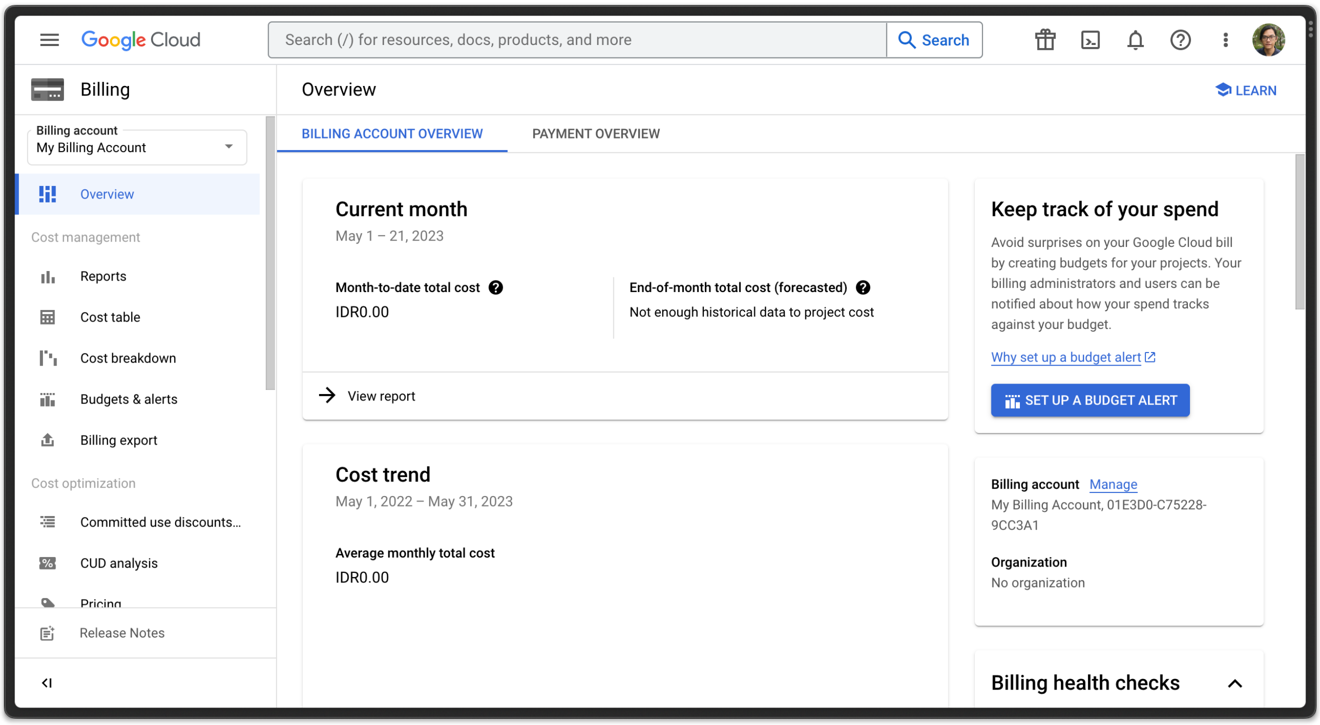
You're working in My Backend Infra

Project number: 616823446234 Project ID: manifest-glyph-385502

From here we can either setup some services we need or even following the guided tutorial.

Setup Billing

Configure the billing limit and setup the budget alert:



Billing account: My Billing Account

Current month
May 1 – 21, 2023

Month-to-date total cost: IDR0.00

End-of-month total cost (forecasted): Not enough historical data to project cost

Cost trend
May 1, 2022 – May 31, 2023

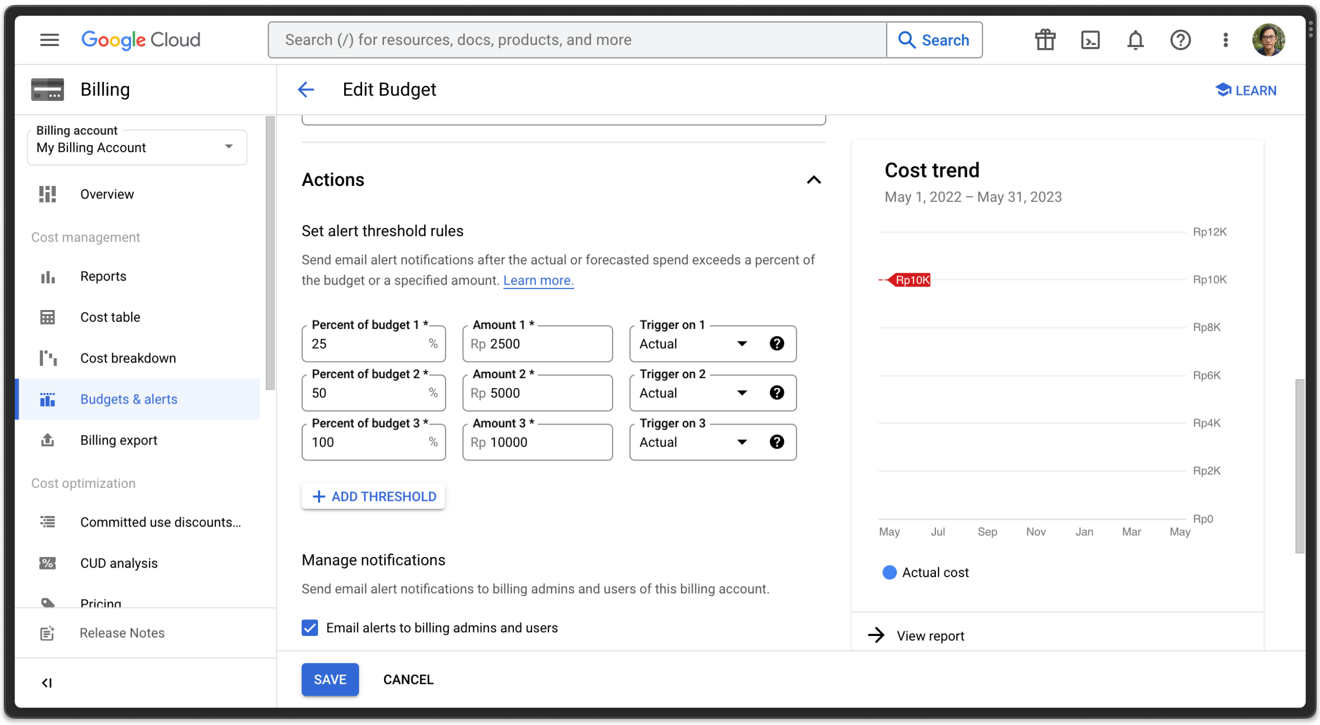
Average monthly total cost: IDR0.00

Keep track of your spend

Avoid surprises on your Google Cloud bill by creating budgets for your projects. Your billing administrators and users can be notified about how your spend tracks against your budget.

Billing account: My Billing Account, 01E3D0-C75228-9CC3A1

Organization: No organization


Google Cloud
Search (/) for resources, docs, products, and more
Search
Billing
Edit Budget
LEARN
Billing account
My Billing Account
Overview
Cost management
Reports
Cost table
Cost breakdown
Budgets & alerts
Billing export
Cost optimization
Committed use discounts...
CUD analysis
Pricing
Release Notes

Actions

Set alert threshold rules

Send email alert notifications after the actual or forecasted spend exceeds a percent of the budget or a specified amount. Learn more.

Percent of budget	Amount	Trigger on
Percent of budget 1 *: 25 %	Amount 1 *: Rp 2500	Trigger on 1: Actual
Percent of budget 2 *: 50 %	Amount 2 *: Rp 5000	Trigger on 2: Actual
Percent of budget 3 *: 100 %	Amount 3 *: Rp 10000	Trigger on 3: Actual

+ ADD THRESHOLD

Manage notifications

Send email alert notifications to billing admins and users of this billing account.

☑ Email alerts to billing admins and users

SAVE CANCEL

Cost trend
May 1, 2022 – May 31, 2023

View report

You need to adjust the threshold based on your preferences.

Artifact Registry Repo

Create a registry repo to store our Docker images.

- Name: `as-you-want`
- Format: Docker
- Mode: Standard
- Location type: Region / Multi-Region

Configure destination:

```
 gcloud auth configure-docker \
asia-southeast2-docker.pkg.dev
```

Make sure to tag the local image if it's not tagged yet for Artifact Registry:

```
 docker tag \
SOURCE-IMAGE \
LOCATION.pkg.dev/PROJECT_ID/REPOSITORY/IMAGE:TAG
```

Example tag:

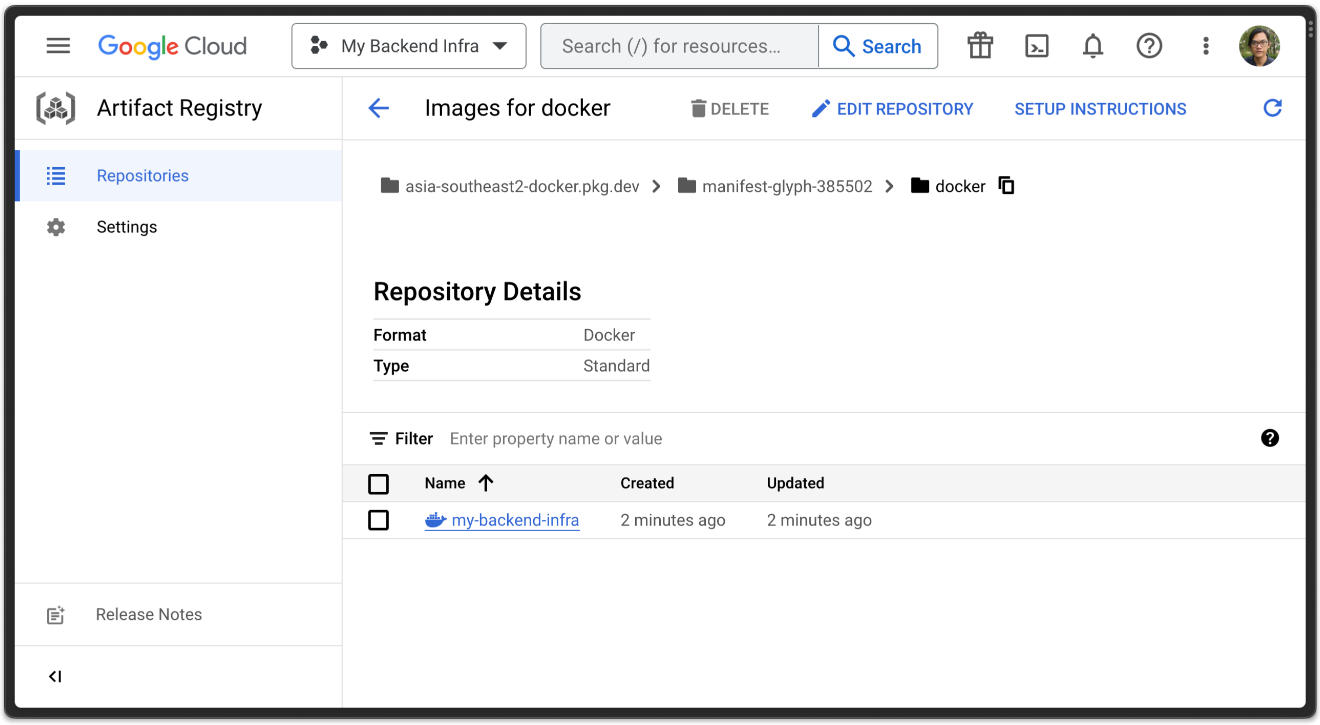
```
 docker tag \
mhaidarh/my-backend-infra \
asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-backend-infra
```

If using Docker Compose, the config can auto tag the image too.

Push the local image to there:

```
 docker push \
asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-backend-infra
```

Once pushed, refresh the Artifact Registry page, then check the image:



Repository Details

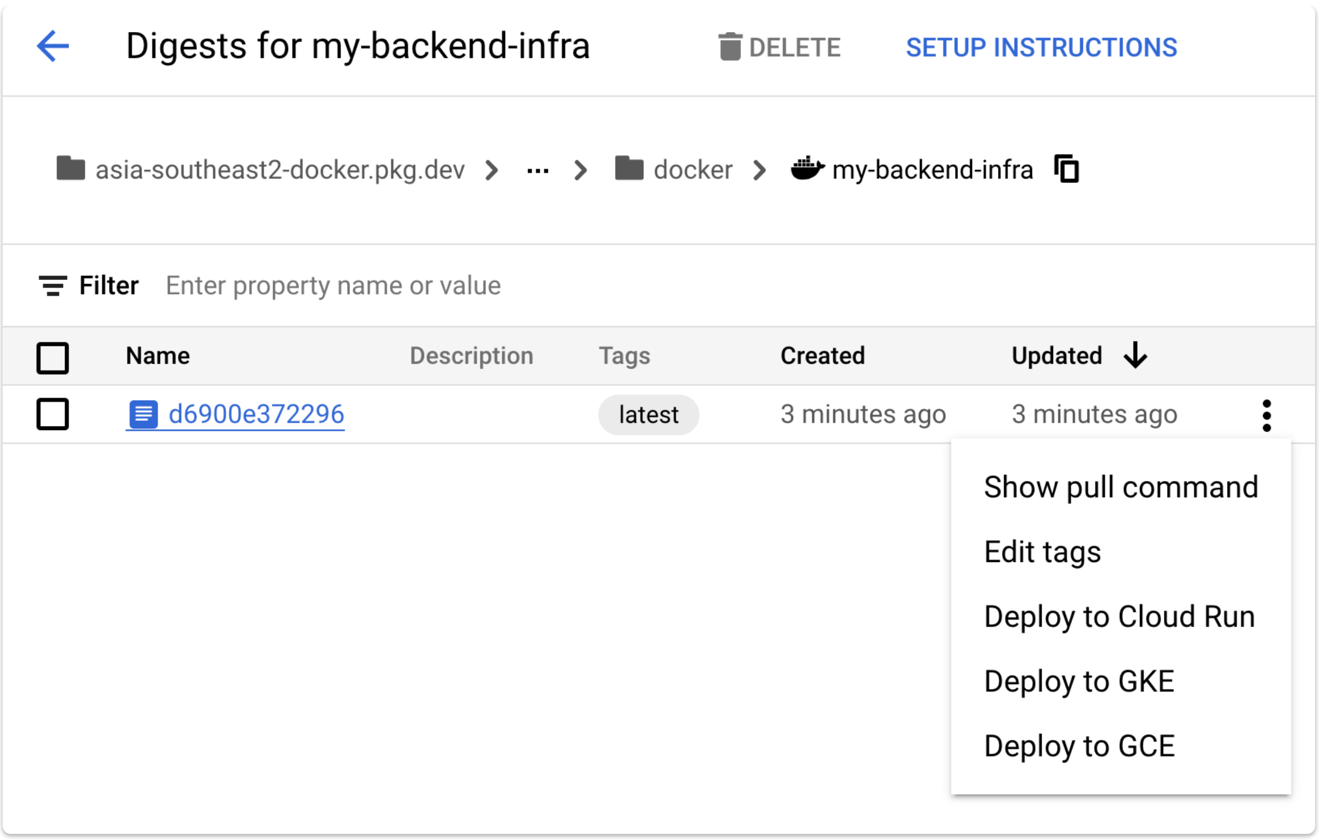
Format	Docker
Type	Standard

Name	Created	Updated
my-backend-infra	2 minutes ago	2 minutes ago

Even from here, there's a menu to deploy:

- Deploy to Cloud Run
- Deploy to GKE (Google Kubernetes Engine)
- Deploy to GCE (Google Compute Engine)

Although we're going to create the GCE VM from the ground up instead.


Digests for my-backend-infra

DELETE SETUP INSTRUCTIONS

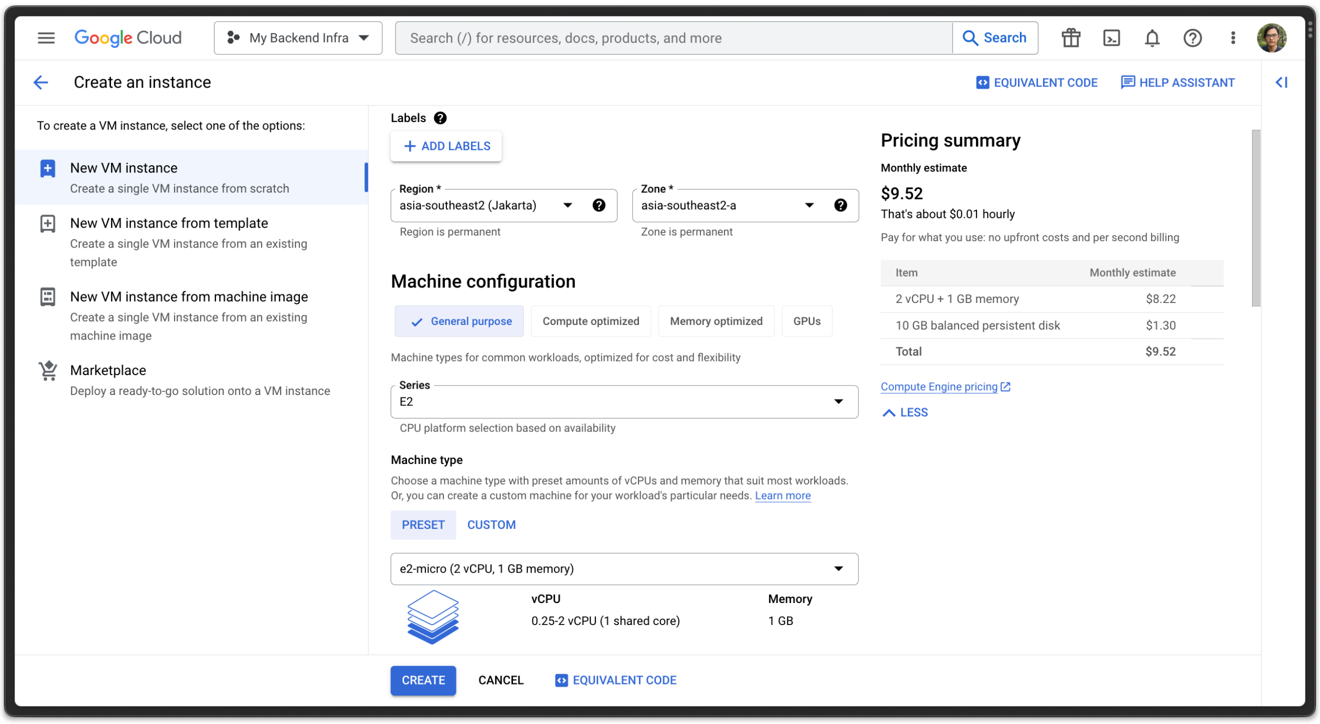
asia-southeast2-docker.pkg.dev > ··· > docker > my-backend-infra

Filter Enter property name or value

Name	Description	Tags	Created	Updated
d6900e372296		latest	3 minutes ago	3 minutes ago

Create a VM

This is the bare minimum spec and average minimum cost to run the VM:



Create an instance

To create a VM instance, select one of the options:

- New VM instance
  Create a single VM instance from scratch
- New VM instance from template
  Create a single VM instance from an existing template
- New VM instance from machine image
  Create a single VM instance from an existing machine image
- Marketplace
  Deploy a ready-to-go solution onto a VM instance

Labels

ADD LABELS

Region * asia-southeast2 (Jakarta)
Region is permanent

Zone * asia-southeast2-a
Zone is permanent

Machine configuration

General purpose | Compute optimized | Memory optimized | GPUs

Machine types for common workloads, optimized for cost and flexibility

Series E2
CPU platform selection based on availability

Machine type

Choose a machine type with preset amounts of vCPUs and memory that suit most workloads. Or, you can create a custom machine for your workload's particular needs. Learn more

PRESET CUSTOM

e2-micro (2 vCPU, 1 GB memory)

vCPU	Memory
0.25-2 vCPU (1 shared core)	1 GB

Pricing summary

Monthly estimate

$9.52

That's about $0.01 hourly

Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 1 GB memory	$8.22
10 GB balanced persistent disk	$1.30
Total	$9.52

Compute Engine pricing

LESS

CREATE CANCEL EQUIVALENT CODE

Recommended spec and configuration:

General

- Name: `as-you-want`
- Labels: (skipped)

- Region: `asia-southeast1` (Singapore) or `asia-southeast2` (Jakarta)
    - Zone: `asia-southeastX-a/b/c`

Machine configuration

- General purpose

Option A

- Series: `E2`
- Machine type: Shared-core `e2-micro` (2 vCPU, 1 GB memory)

Option B

- Series: `N2`
- Machine type: Shared-core: `f1-micro` (1 vCPU, 614 MB memory)
    - Cheaper but less performant

Container

- Image: `asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-backend-infra:latest` (change with your own image)
- Restart policy: `always`

Environment variables can be setup like this, or we can setup an `.env` file manually inside the VM later.

language-sh

```
DATABASE_URL="postgres://myuser:mypassword@database:5432/mybackendextra
JWT_SECRET="abdefghijklmnopqrstuvwxyzabcdefghi"
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
POSTGRES_DB=mybackendextra
```

The variable values depends on your setup. If you use Docker Compose usually the setup is the same or similar. This environment variables can be skipped if we want to use `.env` file later.

Boot disk

- Public images
- Operating system: Container Optimized OS
- Version: latest LTS version
- Boot disk type: Balanced persistent disk
- Size: 10 GB

Pricing summary

Option A

- Monthly estimate: $9.52
- That's about $0.01 hourly
- Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 1 GB memory	$8.22
10 GB balanced persistent disk	$1.30
Total	$9.52

Option B

- Monthly estimate: $6.42
- That's about $0.01 hourly
- Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
1 vCPU + 0.6 GB memory	$7.32
10 GB balanced persistent disk	$1.30
Use discount	-$2.20
Total	$6.42

As you can see, the spec and cost are actually quite similar with what we would have in PaaS like Railway, Render, or Fly.

References:

- https://cloud.google.com/compute/pricing

Identity and API access

- Service account: Compute Engine default service account
  - Requires the Service Account User role (roles/iam.serviceAccountUser) to be set for users who want to access VMs with this service account.
- Access scopes: Allow default access / Allow full access to all Cloud APIs

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

- ✓ Allow HTTP traffic
- ✓ Allow HTTPS traffic

This will also configure the default VPC firewall rules in VPC network.

Equivalent Code

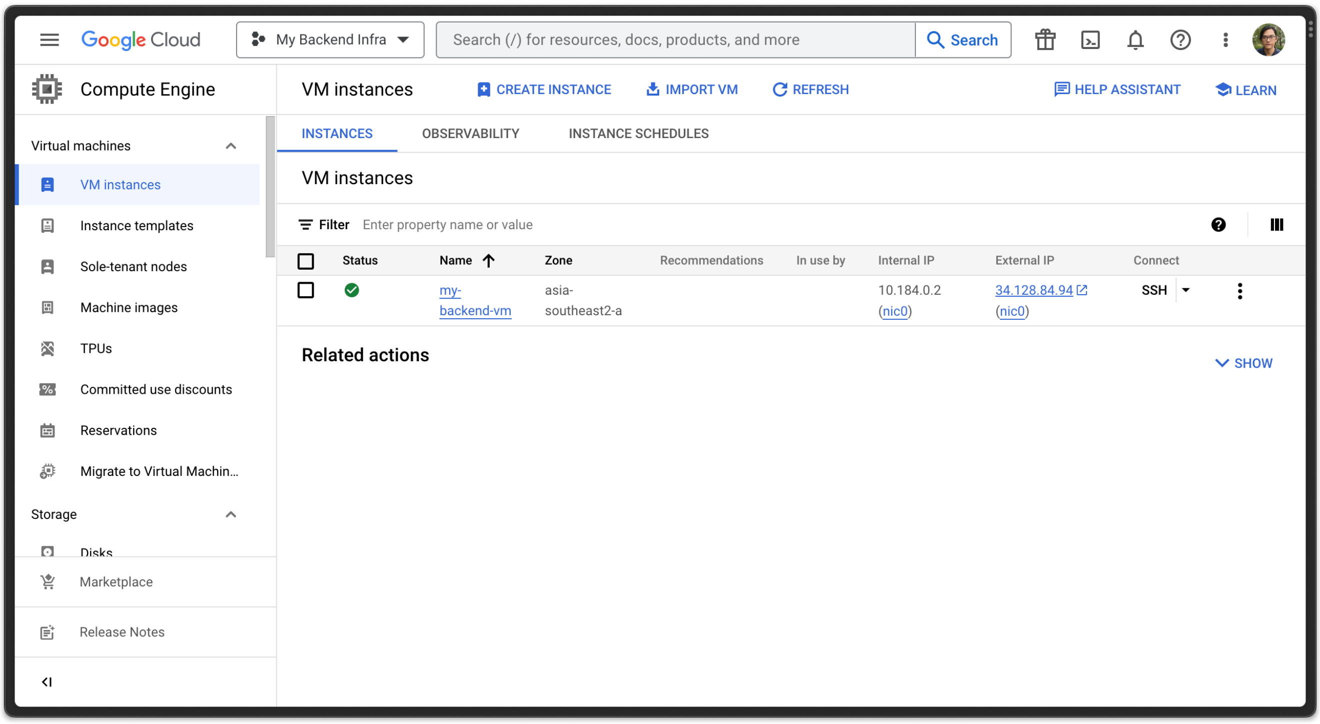
If using `gcloud` CLI, this is the generated command to setup the config above:

```sh
gcloud compute instances create my-backend-vm \
--project=code-name-123456 \
--zone=asia-southeast2-a \
--machine-type=e2-micro \
--network-interface=network-tier=PREMIUM,stack-
type=IPV4_ONLY,subnet=default \
--maintenance-policy=TERMINATE \
--provisioning-model=STANDARD \
--service-account=616823446234-compute@developer.gserviceaccount.com \
--scopes=https://www.googleapis.com/auth/cloud-platform \
--tags=http-server,https-server \
--create-disk=auto-delete=yes,boot=yes,device-name=my-backend-
vm,image=projects/cos-cloud/global/images/cos-101-17162-210-
12,mode=rw,size=10,type=projects/manifest-glyph-385502/zones/asia-
southeast2-a/diskTypes/pd-balanced \
--no-shielded-secure-boot \
--shielded-vtpm \
--shielded-integrity-monitoring \
--labels=goog-ec-src=vm_add-gcloud \
--reservation-affinity=any
```

After making sure everything's fine, create the VM instance.

Created VM Instances

Once the VM has been created, this is what we can get:



VM instances

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	my-backend-vm	asia-southeast2-a			10.184.0.2 (nic0)	34.128.84.94 (nic0)	SSH

For example:

- Name: `as-you-want`
- Zone: `asia-southeast2-a`
- Internal IP: `10.123.0.1`
- External IP: `12.123.45.67`

Connect to the VM

We can also connect to it via SSH with its SSH-in-browser feature, or using `gcloud`:

```sh
gcloud compute ssh \
  --zone "asia-southeast2-a" "my-backend-vm" \
  --project "manifest-glyph-385502"
```

language-sh

By doing this there's a process to copy the SSH key to the VM, either from Google Cloud Console SSH or from our own machine.

> *If you happened to be stuck in the shell, type `Enter` `~` `.` to force exit.*

Just be aware the username and access might be different on each method. In case you need to switch user:

```
 $ sudo -i
# su - your_username
```

Check Docker and configure the permission to access the specified registries:

```
docker version
```

```sh
docker-credential-gcr configure-docker \
  --registries asia-southeast2-docker.pkg.dev
```

language-sh

The `.docker/config.json` now looks like this:

```
 {
 "auths": {},
 "credHelpers": {
   "asia-southeast2-docker.pkg.dev": "gcr",
   "asia.gcr.io": "gcr",
   "eu.gcr.io": "gcr",
   "gcr.io": "gcr",
   "marketplace.gcr.io": "gcr",
   "us.gcr.io": "gcr"
 }
}
```

References:

- https://cloud.google.com/artifact-registry/docs/integrate-compute
- https://cloud.google.com/artifact-registry/docs/docker/pushing-and-pulling#cred-helper

Try to pull the image we created:

```
docker pull \
asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-backend-infra
```

Now we got the Docker image from Artifact Registry into the Compute Engine, for example:

```
docker images
```

```
 REPOSITORY
TAG         IMAGE ID       CREATED             SIZE
asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-backend-infra
latest     0facf4d4041e    1 hour ago    468MB
```

Docker Compose

Although the VM already has Docker installed, but there's no Docker Compose yet.

Because there's a security concern with COS, we cannot install it directly normally. Here we'll need to run Docker Compose via a container. Unless of course, if we're not using COS, regular Linux like Alpine or Ubuntu.

References:

- https://hub.docker.com/r/docker/compose

> *Keep in mind this `docker/compose` image is frozen at v1, not v2. Therefore some latest features are not available. But for this common case, it should work just fine.*

We'd need to pull and run it. But since it's not a regular usage, the command is very specific to allow it to access our disk volume. Because it will need to read the `docker-compose.yaml` file.

Let's copy the long Docker run command to the shell. This is because the command is quite long, so we alias is to be sorter as `docker-compose`.

```sh
vim ~/.bashrc
```

```
alias docker-compose="docker run --rm \
-v /var/run/docker.sock:/var/run/docker.sock \
-v "$PWD:/rootfs/$PWD" \
-w="/rootfs/$PWD" \
--name docker-compose-container \
docker/compose:1.29.2"
```

Notice it also expose the Docker socket. Most information could tell, this case could be kind of dangerous. But because we don't set the container port to be exposed, it cannot be accessed even from our side and outside the VM, so we're still safe.

Then reload the shell, try run it, and check the images:

```sh
source ~/.bashrc
```
language-sh

```sh
docker-compose
```
language-sh

```sh
docker images
```
language-sh

```
  REPOSITORY       TAG        IMAGE ID        CREATED       SIZE
docker/compose   1.29.2    32d8a4638cd8   2 years ago   76.2MB
```

Now let's have a Docker Compose file and required environment variables to run multiple containers. We might only need to setup this once.

This is the `docker-compose.prod.yaml` (commited in our repo) or can be named as regular `docker-compose.yaml` (only for the production VM). Keep in mind the version difference between Docker Compose program and file are still fine.

```yaml
# For production
version: '3.9'

services:
  database:
    image: postgres:alpine
    container_name: my-backend-infra-database
    hostname: database
    volumes:
      - postgres:/var/lib/postgresql/data
    ports:
      - 5432:5432
    env_file:
      - .env
    restart: always
  app:
    image: asia-southeast2-docker.pkg.dev/manifest-glyph-
385502/docker/my-backend-infra
    container_name: my-backend-infra-app
    build: .
```
language-yaml

```
    ports:
      - 80:4000
      - 443:4000
      - 4000:4000
    env_file:
      - .env
    depends_on:
      - database
    restart: always

volumes:
  postgres:
```

Notice the `app` image is now coming from Artifact Registry, not Docker Hub. Also the internal port `4000` is exposed to multiple ports `80` `443` `4000`. Because the API service will be accessed through IP address and domain via HTTP or HTTPS later.

Since it requires environment variables, we also need to create `.env` file:

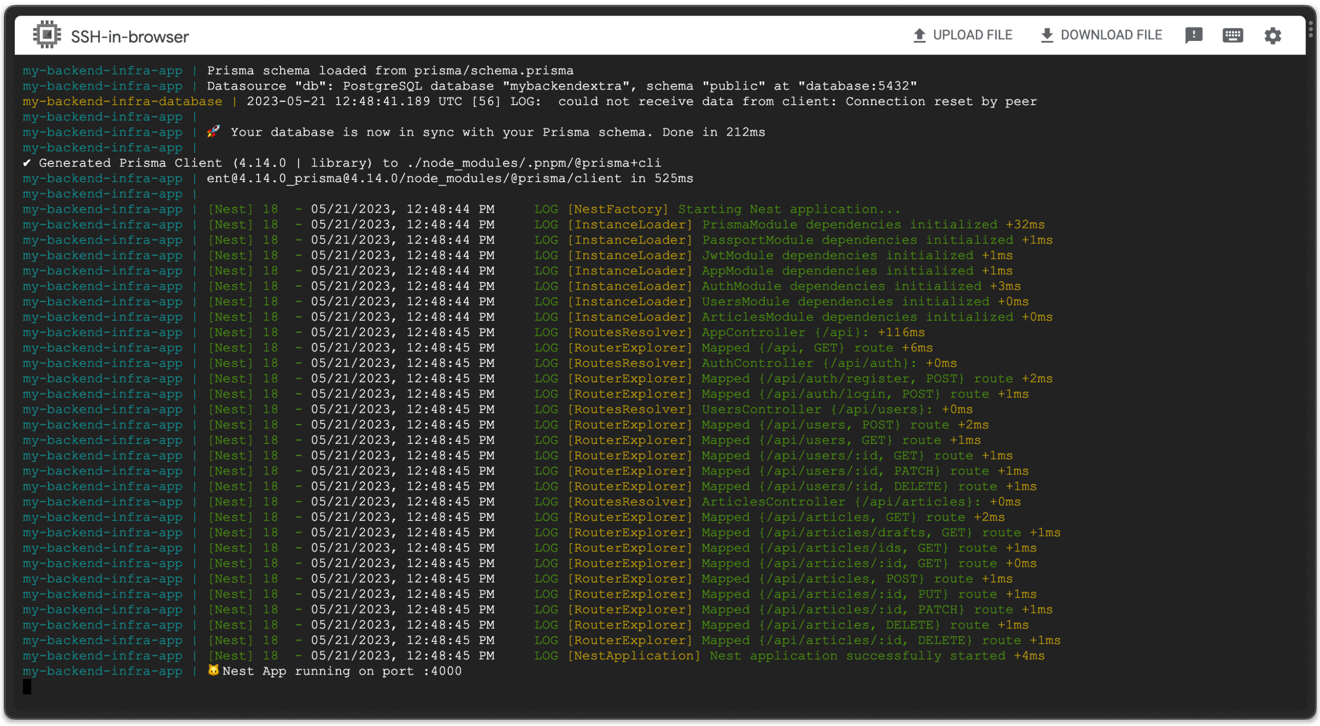
language-sh

```
DATABASE_URL="postgres://myuser:mypassword@database:5432/mybackendextra"
JWT_SECRET="abdefghijklmnopqrstuvwxyzabcdefghi"
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
POSTGRES_DB=mybackendextra
```

Now let's run Docker Compose in the VM:

language-sh

```
docker-compose up
```



```
my-backend-infra-app | Prisma schema loaded from prisma/schema.prisma
my-backend-infra-app | Datasource "db": PostgreSQL database "mybackendextra", schema "public" at "database:5432"
my-backend-infra-database | 2023-05-21 12:48:41.189 UTC [56] LOG:  could not receive data from client: Connection reset by peer
my-backend-infra-app |
my-backend-infra-app | 🚀  Your database is now in sync with your Prisma schema. Done in 212ms
my-backend-infra-app |
✔ Generated Prisma Client (4.14.0 | library) to ./node_modules/.pnpm/@prisma+cli
my-backend-infra-app | ent@4.14.0_prisma@4.14.0/node_modules/@prisma/client in 525ms
my-backend-infra-app |
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [NestFactory] Starting Nest application...
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] PrismaModule dependencies initialized +32ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] PassportModule dependencies initialized +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] JwtModule dependencies initialized +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] AppModule dependencies initialized +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] AuthModule dependencies initialized +3ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] UsersModule dependencies initialized +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:44 PM     LOG [InstanceLoader] ArticlesModule dependencies initialized +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RoutesResolver] AppController {/api}: +116ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api, GET} route +6ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RoutesResolver] AuthController {/api/auth}: +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/auth/register, POST} route +2ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/auth/login, POST} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RoutesResolver] UsersController {/api/users}: +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/users, POST} route +2ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/users, GET} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/users/:id, GET} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/users/:id, PATCH} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/users/:id, DELETE} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RoutesResolver] ArticlesController {/api/articles}: +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles, GET} route +2ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/drafts, GET} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/ids, GET} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/:id, GET} route +0ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles, POST} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/:id, PUT} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/:id, PATCH} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles, DELETE} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [RouterExplorer] Mapped {/api/articles/:id, DELETE} route +1ms
my-backend-infra-app | [Nest] 18  - 05/21/2023, 12:48:45 PM     LOG [NestApplication] Nest application successfully started +4ms
my-backend-infra-app | 😺Nest App running on port :4000
```

If you assured it can run just fine, then we can run it as detached:

```
docker-compose up -d                                        language-sh
```

We can also check the container processes:

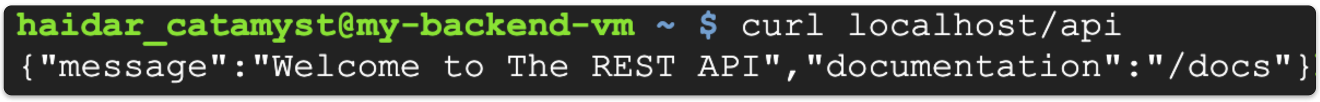
```
docker ps                                                   language-sh
```

```
  docker ps
CONTAINER ID   IMAGE
COMMAND                    CREATED          STATUS          PORTS
NAMES
5c4fc554410c   asia-southeast2-docker.pkg.dev/manifest-glyph-385502/docker/my-
backend-infra   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes
0.0.0.0:80->4000/tcp, :::80->4000/tcp       my-backend-infra-app
2653c1060255   postgres:alpine
"docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:5432-
>5432/tcp, :::5432->5432/tcp   my-backend-infra-database
59ba1116210f   docker/compose:1.29.2
"sh /usr/local/bin/d…"   4 minutes ago   Up 4 minutes
compassionate_mccarthy
```

Request to the app running on the container in the VM

If we request inside the VM shell, the `localhost` is accessible like this:

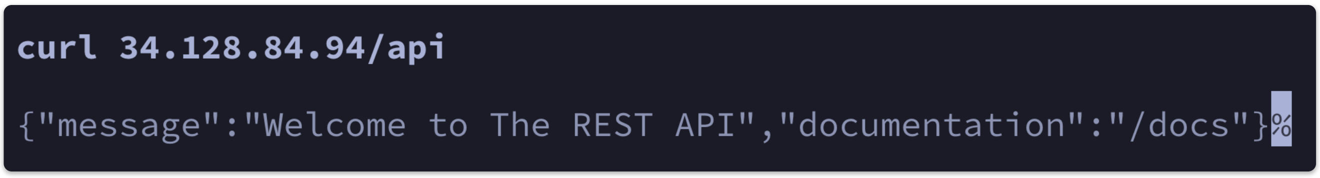
```
curl localhost                                              language-sh
curl localhost:80
curl localhost/api
```



```
haidar_catamyst@my-backend-vm ~ $ curl localhost/api
{"message":"Welcome to The REST API","documentation":"/docs"}
```

But since we want to access it anywhere, therefore if we request it outside of the VM, we access it through the external IP provided before:

```
  curl 12.123.45.56
curl 12.123.45.56:80
curl 12.123.45.56/api
```



```
curl 34.128.84.94/api

{"message":"Welcome to The REST API","documentation":"/docs"}%
```

By default, it's proven available via HTTP and port `80`.

Finally, the service is deployed on Google Cloud, on a Docker container, along with a database with Docker Compose.

Redeploy and Automation

If you need to redeploy, here are the steps:

1. Build a Docker image
2. Push it to Artifact Registry
3. SSH into Compute Engine
4. Run `docker pull` which should get the latest image or run the latest app

For now this is fine for us to do manually. We can actually automate it by using another layer to deploy, such as Raiway, Render, GitHub Actions, GitLab CI, Circle CI, or any other CI server which can automatically deploy a new image after we pushed a new Git commit. Docker also has way to connect remotely via Docker Contexts.

Even more modern, there are some automation software to do it with code such as Kubernetes (K8s), Terraform, Ansible, Puppet, Chef, etc. This particular way of doing infra operation with code is commonly known as part of the DevOps culture.

References:

- https://docs.docker.com/engine/context/working-with-contexts
- https://kubernetes.io
- https://terraform.io
- https://ansible.com
- https://puppet.com
- https://chef.io

Connect to a subdomain/domain

But we're still trying to access it through HTTP with port `80`, not HTTPS with port `443`. Although we already setup the container to expose the app with these ports.

If we use a DNS management service like Cloudflare, although Google Cloud has similar offering with Cloud DNS, then we can connect it and issue HTTPS/SSL/TLS certificate automatically from Cloudflare. This is because not all our projects are hosted in Google Cloud, and using Cloudflare is the most flexible option.

References:

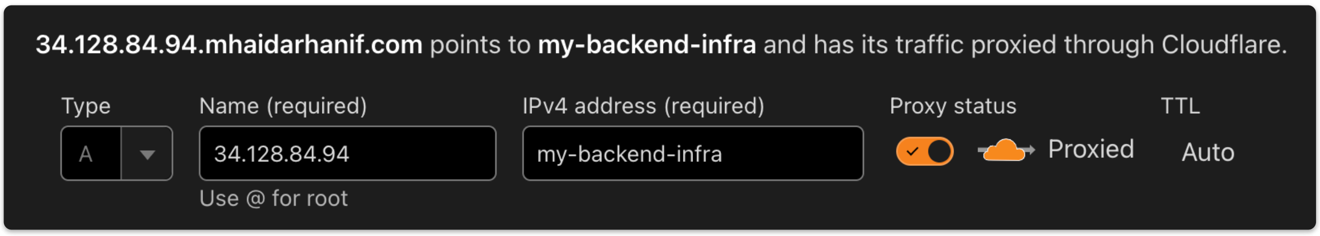
- What is a DNS A record? Cloudflare
- Get started with SSL/TLS · Cloudflare SSL/TLS docs

Alternatively, this can also be done manually with more control, using the combination of Nginx web server's reverse proxy feature and issuing an SSL/TLS certificate with Certbot from Let's Encrypt. But for now this is fine.

Configure an A record:

```
A  subdomain  12.123.45.67
```

Make sure the enable the proxy on Cloudflare so we can get HTTPS easily.



34.128.84.94.mhaidarhanif.com points to **my-backend-infra** and has its traffic proxied through Cloudflare.

Type	Name (required)	IPv4 address (required)	Proxy status	TTL
A	34.128.84.94 Use @ for root	my-backend-infra	Proxied	Auto

If you had deployed multiple services with different requirement for the SSL/TLS, you might have to "Create a Configuration Rule" to customize these settings by hostname.

For example the SSL/TLS encryption mode:

- On Railway or Render, it needs to be **Full**: Encrypts end-to-end, using a self signed certificate on the server
- On Google Cloud without us issuing the certificate, it needs to be **Flexible**: Encrypts traffic between the browser and Cloudflare

In the rule customization, set when incoming requests match...

- Field: Hostname
- Operator: equals / contains
- Value: `your.hostname.com` / `specific-value`
- Select SSL/TLS encryption mode: Flexible or Full depending on the intention

If everything setup correctly, we can see the request to the subdomain/domain can be accessed through HTTPS:

```
curl https://my-backend-infra.mhaidarhanif.com/api

{"message":"Welcome to The REST API","documentation":"/docs"}%
```

Finally again, the service is deployed on Google Cloud and available on a subdomain/domain!

References

- Top 3 ways to run your containers on Google Cloud
    - Deploy a Container to Google Compute Engine (GCE)
    - Intro to Artifact Registry
- Deploy a Container to Google Compute Engine (GCE) - Google Cloud
- Docker Compose on Google's Container Optimized OS!

✋ **Any last questions?**

🙌 Thank you!