
Intro to Cloud

What is Cloud Computing?


THERE IS NO CLOUD
IT'S JUST
SOMEONE ELSE'S COMPUTER
imgflip.com

What is Cloud Computing?

- On-demand computing resources
- Pay-as-you-go pricing
- Resources available over the internet
- No need to manage physical hardware
- Owned by someone else

=> You don't own anything, just rent resources for a certain timeframe

Key Benefits of Cloud Computing

- **Scalability**: Grow resources as needed
- **Cost Efficiency**: Pay only for what you use
- **Global Reach**: Deploy worldwide in minutes
- **Reliability**: Built-in redundancy and failover
- **Innovation**: Access to cutting-edge technology

Types of Cloud Services

IaaS (Infrastructure as a Service)

- Virtual machines
- Storage
- Networks

Popular providers: **AWS**, **Azure**, **GCP**, **Hetzner**, **DigitalOcean**

PaaS (Platform as a Service)

- App Engine
- Cloud Run
- Managed databases

Popular providers: **Vercel**, **Netlify**

and the usual suspects again: **AWS**, **Azure**, **GCP**, **Hetzner**, **DigitalOcean**

SaaS (Software as a Service)

- Gmail
- OneDrive
- ChatGPT
- ...

Providers

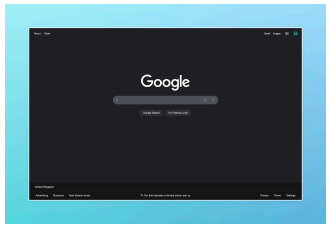
The market gets dominated by three big companies:

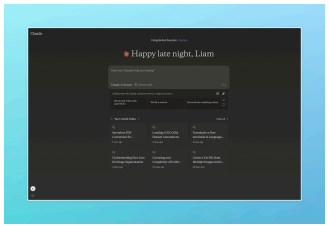
- **AWS** by Amazon
- **Azure** by Microsoft
- **Google Cloud Platform** (GCP) by Google

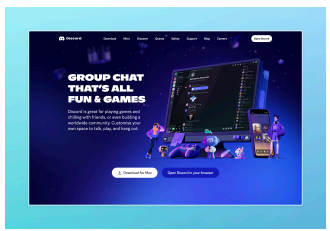
All of them offer roughly the same services, for pretty much anything

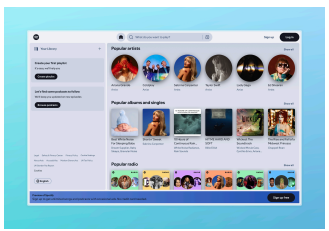
Web applications have three important components

Frontend


Google


Happy late night, Liam


GROUP CHAT
THAT'S ALL
FUN & GAMES

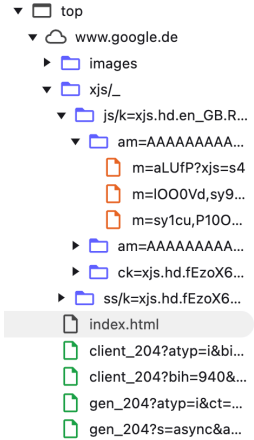

Popular artists
Popular albums and singles
Popular radio

Frontend

continued

At the base of every frontend is a html document, which defines the structure of the webpage

Sometimes, CSS and JS are loaded in via the base html document



Backend

Two types:

- "Basic" file servers like **nginx** or **apache**
- Usually stateless
- Custom backends that contain logic, written in something like Node.js, Golang, etc
  - Often not directly exposed to the Internet, instead behind a reverse proxy like **nginx**

Serverless vs "Serverfull" (long running deployments)

huh, i thought you need servers to serve content?

- Contrary to what the name might suggest, you still needs servers for serverless
- Servers get spun up on demand for each request, instead of running all the time
- Can be cheaper for apps which get used rarely
- Pricing model typically exponential

Database

- sometimes just a file system (storing the plain html documents)
- sometimes a full scale sharded + replicated DB like MongoDB or MySQL
- used for persistent storage of data

Deployment

Simple client side app

Deployment options, ranked from easy to hard:

1. One click deploy on Vercel (or Github pages, Netlify)
2. Deploy to a large scale cloud platform, like GCP, AWS, Cloudflare via a special file hosting service
3. Deploy to a VPS or bare metal box

Demo time

lets do a one click deploy for a simple client side webapp

Alright, lets deploy an app as a docker container

Its a basic early internet style web page:

- can show time
- has a small api for random numbers
- has a GUESTBOOOOOOOK

Cloud Run Deployment to GCP

- Lets create a project

```
# enable some apis
gcloud services enable cloudbuild.googleapis.com run.googleapis.com artifactregistry.googleapis.com cloudresourcemanager

# create a service account for the github actions
gcloud iam service-accounts create github-actions \
    --description="Service account for GitHub Actions CI/CD" \
    --display-name="GitHub Actions"

# create sa for running the app in cloud run
gcloud iam service-accounts create cloud-run-sa \
  --description="Custom service account for running Cloud Run services" \
  --display-name="Cloud Run Service Account"

# set up permissions for service account:
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
    --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
    --role="roles/run.admin"
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
    --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
    --role="roles/storage.admin"
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
    --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
    --role="roles/cloudbuild.builds.editor"
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role="roles/artifactregistry.writer"
```

```
# get name of default compute account
COMPUTE_ACCOUNT=$(gcloud projects describe $(gcloud config get-value project) \
    --format="value(projectNumber)" | awk '{print $1 "-compute@developer.gserviceaccount.com"}')

# grant permissions to impersonate default compute account because its being used for the deployment process ...
gcloud iam service-accounts add-iam-policy-binding \
    $COMPUTE_ACCOUNT \
    --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
    --role="roles/iam.serviceAccountUser"

# necessary permissions for cloud run set up
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member="serviceAccount:cloud-run-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role="roles/run.invoker"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member="serviceAccount:cloud-run-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role="roles/logging.logWriter"
```

```
# allow github actions account to impersonate cloud run account
gcloud iam service-accounts add-iam-policy-binding \
  cloud-run-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com \
  --member="serviceAccount:github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role="roles/iam.serviceAccountUser"

# create registry
gcloud artifacts repositories create dockerintro \
    --repository-format=docker \
    --location=europe-west3 \
    --description="Docker repository for Cloud Run app"

# create database
gcloud firestore databases create \
    --location=europe-west3 \
    --type=firestore-native

# allow it to write to db
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member="serviceAccount:cloud-run-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role="roles/datastore.user"

# get the key to be able to authenticate as the service account
gcloud iam service-accounts keys create key.json \
    --iam-account github-actions@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com
```

Finally done with permissions, lets set up the continious deployment pipeline on github

1. Add the key as a secret to the github actions for the repo `GCP_CREDENTIALS`
2. Add the name of the GCP project as an env
3. Set up the pipeline

Ok, lets try changing some code and see the pipeline in action

Resources to learn more

- Google Cloudskillsboost for tutorials
- ChatJippeetee, Claude, etc

Final credits

This demo has been powered by Bun, htmx and nitro

</> htmx

Go check them out, they’re awesome tools and make it so much more fun to develop webapps