Real-Time Log Analytics Service

The Architect Phase:

Problem Statement: Application teams currently lack a centralized, real-time system to analyze high-volume log data. This fragmentation delays incident response, makes performance monitoring difficult, and prevents effective analysis of user behavior.

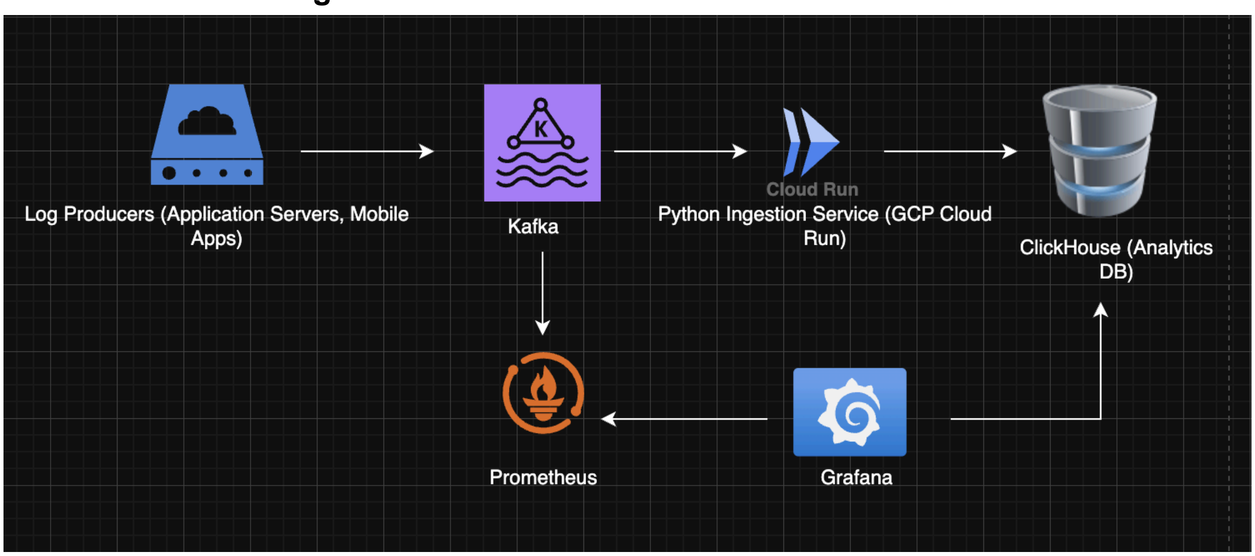
Define the Requirements:
Functional Requirements (What it does):

- Ingest log events from multiple sources.
- Store log data efficiently.
- Provide an interface for fast, real-time queries.
- Visualize key metrics on a dashboard.

Non-Functional Requirements (How well it does it):

- **Throughput:** Must process over 1 million events per day.
- **Performance:** P99 query latency must be under 400ms.
- **Scalability:** The ingestion service must scale automatically with load.
- **Durability:** The system must guarantee no data loss during ingestion.

The Architecture Diagram:



Technology Justification:

1. **GCP Cloud Run:**Chosen for its serverless, auto-scaling capabilities. It allows us to handle variable log traffic cost-effectively without managing any underlying server infrastructure
2. **Apache Kafka:**Selected as a fault-tolerant message bus. It decouples our log producers from the consumer service, providing a durable buffer that prevents data loss during service outages or traffic spikes
3. **ClickHouse:**Its columnar storage engine is purpose-built for high-speed analytical queries (OLAP). This is critical for meeting our requirement of sub-400ms query performance on terabytes of log data.
4. **Prometheus & Grafana:**Chosen as the industry-standard open-source stack for monitoring and visualization. Prometheus provides robust time-series data collection, while Grafana offers a powerful and flexible platform for creating insightful dashboards.

The Engineer Phase:

Python Ingestion Service:

1. Create database schema
2. Build Your Local Environment : pinning up the core backend services (Kafka + ClickHouse) on your own laptop using Docker.

3. **Docker Compose:**
    - Docker Compose is like a recipe book.
    - It lets you describe multiple services (Kafka, ClickHouse, Zookeeper) in one YAML file and start them all together with a single command (`docker-compose up`).
4. **Zookeeper**
    - Required by **Kafka**.
    - Manages Kafka brokers, keeps track of who's the leader, coordinates distributed state.
    - You don't use Zookeeper directly — it's just plumbing for Kafka.

5. **Kafka (Message Broker)**
    - Think of Kafka as your **"data bus."**
    - Producers (apps, servers) send logs → Kafka stores them temporarily → Consumers (like your Python service) read them.
    - It's designed for high-throughput, fault-tolerant streaming.

6. **ClickHouse (Analytics Database)**
    - A super-fast database optimized for **analytics and logs**.
    - Instead of traditional row-based storage (like Postgres), it uses **columnar storage** — making queries over billions of rows very fast.
    - This is where your logs will ultimately be stored for querying and dashboards.

7. **Docker Volumes**

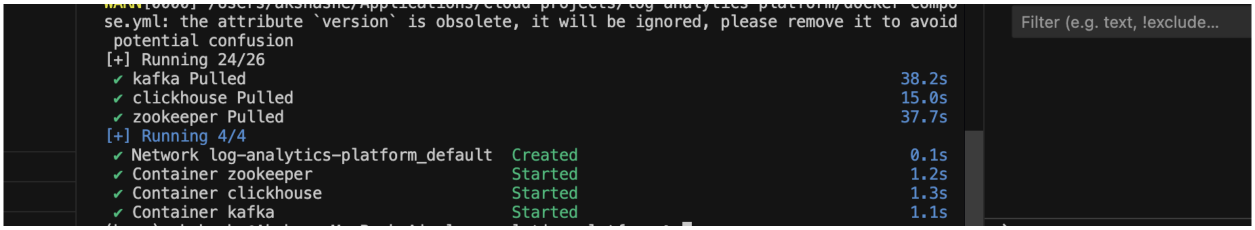
```
volumes:

- ./clickhouse-data:/var/lib/clickhouse
```

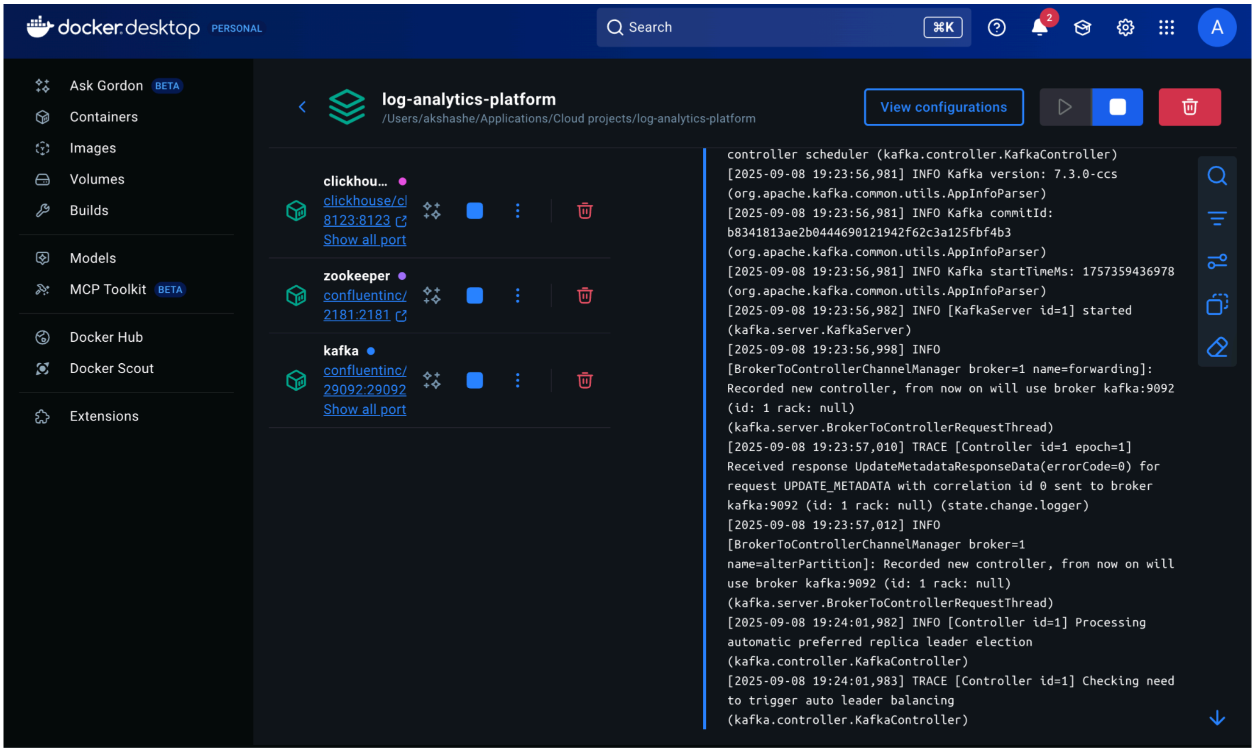
   - That means: "Save database files to my local folder `clickhouse-data`."
   - Without this, data would vanish every time you stopped the container.

The flow:

Producers (your test apps) → send logs → **Kafka**

Python ingestion service → consumes logs → stores them into **ClickHouse**
Grafana → queries ClickHouse + Prometheus → shows dashboards





Zookeeper: The Coordinator

Of course. Let's break down what each service in your `docker-compose.yml` file does. Think of this file as a blueprint for creating a mini, virtual data center on your computer.

Zookeeper: The Coordinator 🐘

What it is: Apache Zookeeper is a centralized service for maintaining configuration information, naming, and providing distributed coordination.

Its role in our project: Zookeeper acts as the **manager or "brain" for Kafka**. In older Kafka versions, it was responsible for keeping track of critical metadata, such as:

- Which brokers (Kafka servers) are currently alive.
- Who is the "leader" for a specific data topic.
- Configuration settings for all topics.

In short, Kafka brokers talk to Zookeeper to coordinate their work. Without it, the Kafka cluster wouldn't know how to function.

Of course. Let's break down what each service in your `docker-compose.yml` file does. Think of this file as a blueprint for creating a mini, virtual data center on your computer.

Zookeeper: The Coordinator 🐘

What it is: Apache Zookeeper is a centralized service for maintaining configuration information, naming, and providing distributed coordination.

Its role in our project: Zookeeper acts as the **manager or "brain" for Kafka**. In older Kafka versions, it was responsible for keeping track of critical metadata, such as:

- Which brokers (Kafka servers) are currently alive.
- Who is the "leader" for a specific data topic.
- Configuration settings for all topics.

In short, Kafka brokers talk to Zookeeper to coordinate their work. Without it, the Kafka cluster wouldn't know how to function.

Key Configuration: The line `KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'` in the Kafka service is the explicit instruction telling Kafka where to find its manager.

> **Note:** Newer versions of Kafka can now run without Zookeeper in a mode called KRaft, but using Zookeeper is still very common in many production environments and is great for learning.

Kafka: The Data Superhighway

What it is: Apache Kafka is a distributed event streaming platform. It's designed to move huge amounts of data from a source to a destination, reliably and in real-time.

Its role in our project: Kafka is our **central message bus**. It acts like a highly organized post office.

1. **Log Producers** (your applications) send log messages (letters) to a specific Kafka **Topic** (a P.O. Box).
2. Kafka stores these messages durably for a configured amount of time.
3. **Your Python Service** (the consumer) connects to Kafka and picks up the messages from the topic to process them.

It creates a buffer that separates the application writing the logs from the service processing them. If your Python service goes down for 5 minutes, Kafka holds onto the logs, and the service can pick them right up when it comes back online.

ClickHouse: The Analytics Engine

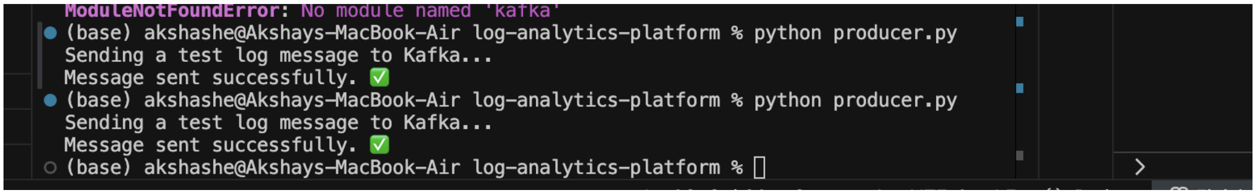
What it is: ClickHouse is a high-performance, columnar database management system.

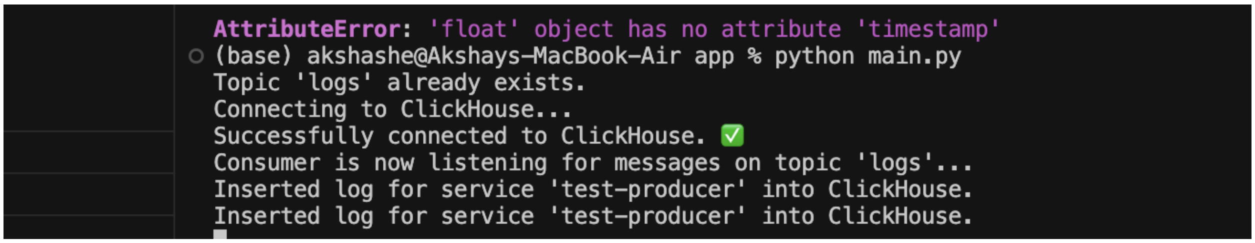
Its role in our project: This is our **specialized warehouse and query engine**. While a normal database stores data in rows (like a spreadsheet), ClickHouse stores it in columns. This structure makes it incredibly fast for analytical queries that scan and aggregate large amounts of data, like:

- "Count all `ERROR` logs from the `payment-service` in the last hour."
- "Find the average response time for `user_id` '123'."

This is why it can meet our sub-400ms query performance goal.

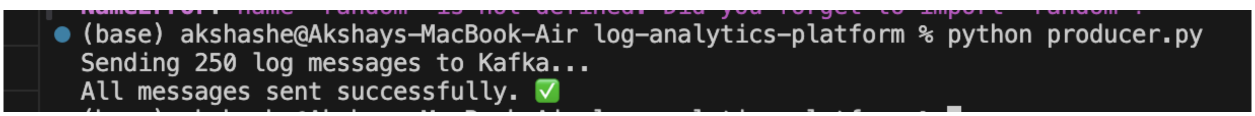
Next task : To create the Python consumer that reads data from Kafka

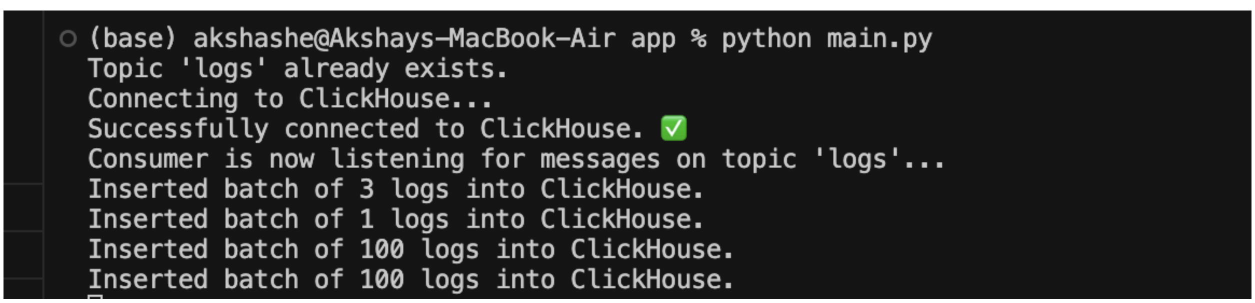




Your local data pipeline is now **fully functional**: data flows from the producer, through Kafka, is processed by your Python service, and is correctly stored in ClickHouse.

Now, let's optimize it. A high-throughput service can't insert logs one by one; it's too slow. The key to performance is **batching**.





The DevOps Phase

Your goal is to containerize the application so it can be deployed on Google Cloud Run. This involves creating a `Dockerfile`, which is a recipe for building a portable image of your service.

The **`Dockerfile`** provides a consistent, reproducible environment for your application. GCP Cloud Run will use this file to build and run your service exactly as you've defined it, eliminating "it works on my machine" problems. The `.dockerignore` file ensures the container is as lean as possible.

Prerequisites

- You have a **GCP project** created.
- You have installed and authenticated the **`gcloud` CLI**.
- Your Kafka and ClickHouse instances are accessible from the internet (e.g., using a managed service like Confluent Cloud, Aiven, or ClickHouse Cloud). **The `localhost` addresses will not work from GCP.**

```
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % gcloud artifacts repositories create lo
g-analytics-repo \
    --repository-format=docker \
    --location=us-central1 \
    --description="Docker repository for log analytics project"
Create request issued for: [log-analytics-repo]
Waiting for operation [projects/graceful-trees-471716-s6/locations/us-central1/oper
ations/1a1a1a91-5630-41ea-b613-29432ea4d423] to complete...done.
Created repository [log-analytics-repo].
(base) akshashe@Akshays-MacBook-Air log-analytics-platform %
```

Next Step: Build and Push Your Image

1. Authorize Docker

```
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % gcloud auth configure-docker us-central
1-docker.pkg.dev
Adding credentials for: us-central1-docker.pkg.dev
After update, the following will be written to your Docker config file located at
[/Users/akshashe/.docker/config.json]:
 {
  "credHelpers": {
    "us-central1-docker.pkg.dev": "gcloud"
  }
}

Do you want to continue (Y/n)?  y

Docker configuration file updated.
```

2. Build and Tag the Image

```
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % docker build -t us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-a
nalytics-repo/ingestion-service:latest .
[+] Building 216.4s (10/10) FINISHED                                                                              docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                             0.0s
 => => transferring dockerfile: 577B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                              0.4s
 => [internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 174B                                                                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:91e9d01cf4bd56be7128c603506b6fe367ef7506f9f2ad8f3a908aeec8941bb9      211.9s
 => => resolve docker.io/library/python:3.11-slim@sha256:91e9d01cf4bd56be7128c603506b6fe367ef7506f9f2ad8f3a908aeec8941bb9        0.0s
 => => sha256:e9246f522e03b2309c041f8eea3e8db7f878724118f01e7ec7c99cf9435f7ddb 249B / 249B                                       0.3s
 => => sha256:0dc0d0d23e1f63adec383cc7eeb1fcaa0816e9b5bceb359df4c0257abc446b2a 14.59MB / 14.59MB                               211.6s
 => => sha256:440ff9e33d745ab81aef0b1d7dbb2efaaaf60e167f7d0cd36d26ac0e5154eec3 1.27MB / 1.27MB                                   0.7s
 => => sha256:b2feff975e6dd2ebaf182772fb9ee26274648387b061e821e0bb5026735dd094 30.14MB / 30.14MB                                 2.0s
 => => extracting sha256:b2feff975e6dd2ebaf182772fb9ee26274648387b061e821e0bb5026735dd094                                        0.5s
 => => extracting sha256:440ff9e33d745ab81aef0b1d7dbb2efaaaf60e167f7d0cd36d26ac0e5154eec3                                        0.0s
 => => extracting sha256:0dc0d0d23e1f63adec383cc7eeb1fcaa0816e9b5bceb359df4c0257abc446b2a                                        0.3s
 => => extracting sha256:e9246f522e03b2309c041f8eea3e8db7f878724118f01e7ec7c99cf9435f7ddb                                        0.0s
```

3.Push the Image to GCP

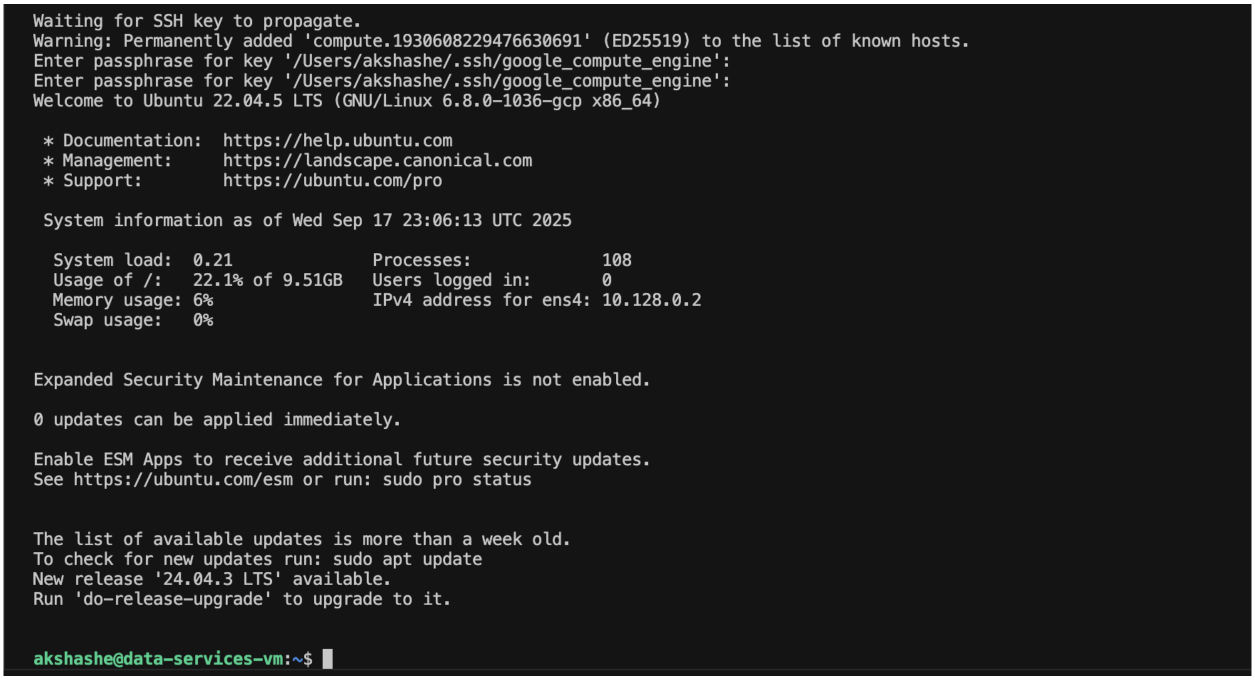
```
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % docker push us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analy
tics-repo/ingestion-service:latest
The push refers to repository [us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service]
b2feff975e6d: Pushed
7f778ec53244: Pushed
0dc0d0d23e1f: Pushed
440ff9e33d74: Pushed
7ca2de6c73aa: Pushed
2f4c010f55c3: Pushed
e9246f522e03: Pushed
057565063e5e: Pushed
d9860368ff97: Pushed
latest: digest: sha256:f416bc845961634624796ef6a7f871f6c8a528f88461b1f2e1475cc0454da315 size: 856
```

Final Step: Deploy and Go Live

```
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % ./setup_vm.sh
--- Setting project to graceful-trees-471716-s6 ---
Updated property [core/project].


Updates are available for some Google Cloud CLI components.  To install them,
please run:
  $ gcloud components update

--- Creating VM instance: data-services-vm ---
Created [https://www.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/zones/us-central1-c/instances/data-services-vm].
NAME              ZONE           MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP   STATUS
data-services-vm  us-central1-c  e2-medium                  10.128.0.2   34.71.31.191  RUNNING
--- Reserving a static IP address ---
Created [https://www.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/regions/us-central1/addresses/data-services-vm-static-ip].
ERROR: (gcloud.compute.instances.add-access-config) Could not fetch resource:
 - At most one access config currently supported.

--- Creating firewall rule for Follow link (cmd + click)
Creating firewall...⠹Created [https://www.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/global/firewalls/allow-kafka].
Creating firewall...done.
NAME         NETWORK  DIRECTION  PRIORITY  ALLOW      DENY  DISABLED
allow-kafka  default  INGRESS    1000      tcp:29092        False
--- Creating firewall rule for ClickHouse (port 8123) ---
Creating firewall...⠹Created [https://www.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/global/firewalls/allow-clickhouse].
Creating firewall...done.
NAME              NETWORK  DIRECTION  PRIORITY  ALLOW     DENY  DISABLED
allow-clickhouse  default  INGRESS    1000      tcp:8123        False
--- ✅ Setup Complete ---
Your Static IP is: 34.29.233.31
You can now SSH into the VM with: gcloud compute ssh data-services-vm --zone=us-central1-c
(base) akshashe@Akshays-MacBook-Air log-analytics-platform %
```

Install Software on the VM

1. **Ssh into vm**

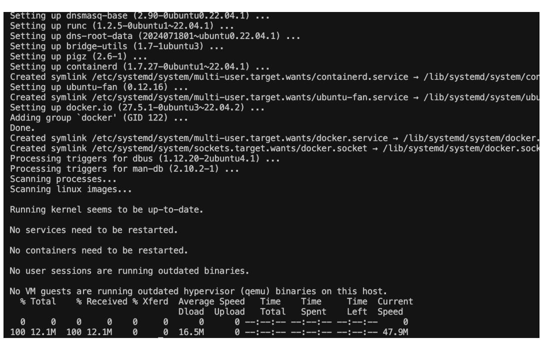


```
Waiting for SSH key to propagate.
Warning: Permanently added 'compute.1930608229476630691' (ED25519) to the list of known hosts.
Enter passphrase for key '/Users/akshashe/.ssh/google_compute_engine':
Enter passphrase for key '/Users/akshashe/.ssh/google_compute_engine':
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1036-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Sep 17 23:06:13 UTC 2025

  System load:  0.21              Processes:             108
  Usage of /:   22.1% of 9.51GB   Users logged in:       0
  Memory usage: 6%                IPv4 address for ens4: 10.128.0.2
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


akshashe@data-services-vm:~$
```

2. **Install Docker and Run Services**

Once you are inside the VM, you'll need to run the commands from **Step 3 and Step 4** of the "DIY" guide. This involves:

- Installing Docker and Docker Compose.



```
Setting up dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
Setting up runc (1.2.5-0ubuntu1~22.04.1) ...
Setting up dns-root-data (2024071801~ubuntu0.22.04.1) ...
Setting up bridge-utils (1.7-1ubuntu3) ...
Setting up pigz (2.6-1) ...
Setting up containerd (1.7.27-0ubuntu1~22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/con
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubu
Setting up docker.io (27.5.1-0ubuntu3~22.04.2) ...
Adding group `docker' (GID 122) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.sock
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 12.1M  100 12.1M    0     0  16.5M      0 --:--:-- --:--:-- --:--:-- 47.9M
```

- Creating the `docker-compose.yml` file (remember to use your static IP `34.29.233.31` in this file).
- Running `docker-compose up -d`.

```
akshashe@data-services-vm:~$ mkdir services && cd services
akshashe@data-services-vm:~/services$ nano docker-compose.yml
akshashe@data-services-vm:~/services$ docker-compose up -d
```

```
Updated [https://www.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/zones/us-central1-c/instances/data-services-vm].
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % gcloud compute instances start data-services-vm --zone=us-central1-c
Starting instance(s) data-services-vm...done.
Updated [https://compute.googleapis.com/compute/v1/projects/graceful-trees-471716-s6/zones/us-central1-c/instances/data-services-vm].
Instance internal IP is 10.128.0.2
Instance external IP is 35.223.146.238
```

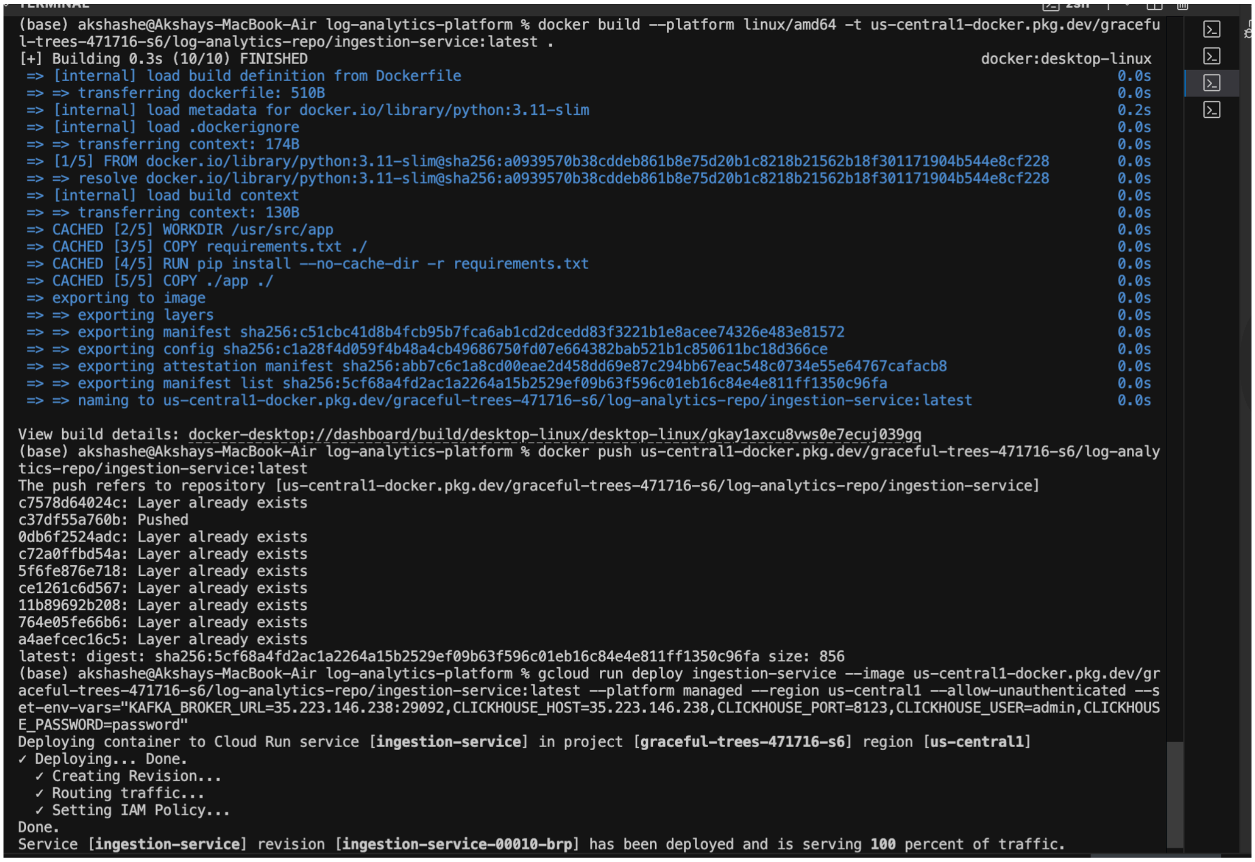
After that, your Kafka and ClickHouse services will be running in the cloud, and you'll be ready for the final deployment of your Cloud Run application

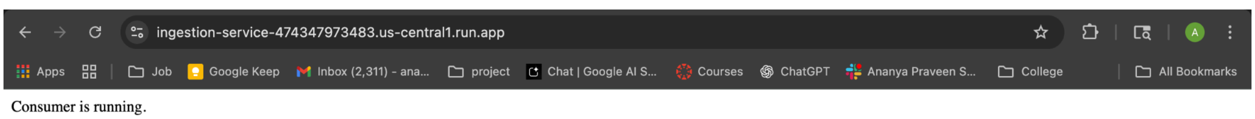
```
Digest: sha256:3b28daecdf0625bd7dc27d555ae8f39042882045e49b04668ceccdd282f67d9b
Status: Downloaded newer image for clickhouse/clickhouse-server:latest
Creating zookeeper  ... done
Creating clickhouse ... done
Creating kafka      ... done
akshashe@data-services-vm:~/services$
```

Success! Your Kafka and ClickHouse services are now live on your GCP server.

You have reached the final step of your entire project: deploying your application to Cloud Run to bring everything online.

Final Mission: Go Live


(base) akshashe@Akshays-MacBook-Air log-analytics-platform % docker build --platform linux/amd64 -t us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service:latest .
[+] Building 0.3s (10/10) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 510B 0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim 0.2s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 174B 0.0s
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:a0939570b38cddeb861b8e75d20b1c8218b21562b18f301171904b544e8cf228 0.0s
=> => resolve docker.io/library/python:3.11-slim@sha256:a0939570b38cddeb861b8e75d20b1c8218b21562b18f301171904b544e8cf228 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 130B 0.0s
=> CACHED [2/5] WORKDIR /usr/src/app 0.0s
=> CACHED [3/5] COPY requirements.txt ./ 0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> CACHED [5/5] COPY ./app ./ 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => exporting manifest sha256:c51cbc41d8b4fcb95b7fca6ab1cd2dcedd83f3221b1e8acee74326e483e81572 0.0s
=> => exporting config sha256:c1a28f4d059f4b48a4cb49686750fd07e664382bab521b1c850611bc18d366ce 0.0s
=> => exporting attestation manifest sha256:abb7c6c1a8cd00eae2d458dd69e87c294bb67eac548c0734e55e64767cafacb8 0.0s
=> => exporting manifest list sha256:5cf68a4fd2ac1a2264a15b2529ef09b63f596c01eb16c84e4e811ff1350c96fa 0.0s
=> => naming to us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service:latest 0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gkay1axcu8vws0e7ecuj039gq
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % docker push us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service:latest
The push refers to repository [us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service]
c7578d64024c: Layer already exists
c37df55a760b: Pushed
0db6f2524adc: Layer already exists
c72a0ffbd54a: Layer already exists
5f6fe876e718: Layer already exists
ce1261c6d567: Layer already exists
11b89692b208: Layer already exists
764e05fe66b6: Layer already exists
a4aefcec16c5: Layer already exists
latest: digest: sha256:5cf68a4fd2ac1a2264a15b2529ef09b63f596c01eb16c84e4e811ff1350c96fa size: 856
(base) akshashe@Akshays-MacBook-Air log-analytics-platform % gcloud run deploy ingestion-service --image us-central1-docker.pkg.dev/graceful-trees-471716-s6/log-analytics-repo/ingestion-service:latest --platform managed --region us-central1 --allow-unauthenticated --set-env-vars="KAFKA_BROKER_URL=35.223.146.238:29092,CLICKHOUSE_HOST=35.223.146.238,CLICKHOUSE_PORT=8123,CLICKHOUSE_USER=admin,CLICKHOUSE_PASSWORD=password"
Deploying container to Cloud Run service [ingestion-service] in project [graceful-trees-471716-s6] region [us-central1]
✓ Deploying... Done.
✓ Creating Revision...
✓ Routing traffic...
✓ Setting IAM Policy...
Done.
Service [ingestion-service] revision [ingestion-service-00010-brp] has been deployed and is serving 100 percent of traffic.


ingestion-service-474347973483.us-central1.run.app
Apps
Job
Google Keep
Inbox (2,311) - ana...
project
Chat | Google AI S...
Courses
ChatGPT
Ananya Praveen S...
College
All Bookmarks
Consumer is running.

SEVERITY	TIME	SUMMARY
	2025-09-17 19:53:55.201	Starting ingestion service...
	2025-09-17 19:53:55.201	Attempt 1/5 to connect to ClickHouse at 35.223.146.238:8123...
	2025-09-17 19:53:55.201	* Serving Flask app 'main'
	2025-09-17 19:53:55.202	* Debug mode: off
	2025-09-17 19:54:05.942	Successfully connected to ClickHouse. ✅
	2025-09-17 19:54:05.942	Attempt 1/5 to connect to Kafka at 35.223.146.238:29092...
	2025-09-17 19:54:05.942	Successfully connected to Kafka. ✅
	2025-09-17 19:54:05.942	Consumer is now listening for messages on topic 'logs'...
	2025-09-17 19:54:05.942 PDT	Starting ingestion service...
	2025-09-17 19:55:09.213	Attempt 1/5 to connect to ClickHouse at 35.223.146.238:8123...
	2025-09-17 19:55:09.213	* Serving Flask app 'main'
	2025-09-17 19:55:09.213	* Debug mode: off
	2025-09-17 19:55:19.435	Successfully connected to ClickHouse. ✅
	2025-09-17 19:55:19.435	Attempt 1/5 to connect to Kafka at 35.223.146.238:29092...
	2025-09-17 19:55:19.435	Successfully connected to Kafka. ✅
	2025-09-17 19:55:19.435	Consumer is now listening for messages on topic 'logs'...
	2025-09-17 19:55:50.042	Starting ingestion service...
	2025-09-17 19:55:50.042	Attempt 1/5 to connect to ClickHouse at 35.223.146.238:8123...
	2025-09-17 19:55:50.042	* Serving Flask app 'main'

done!!