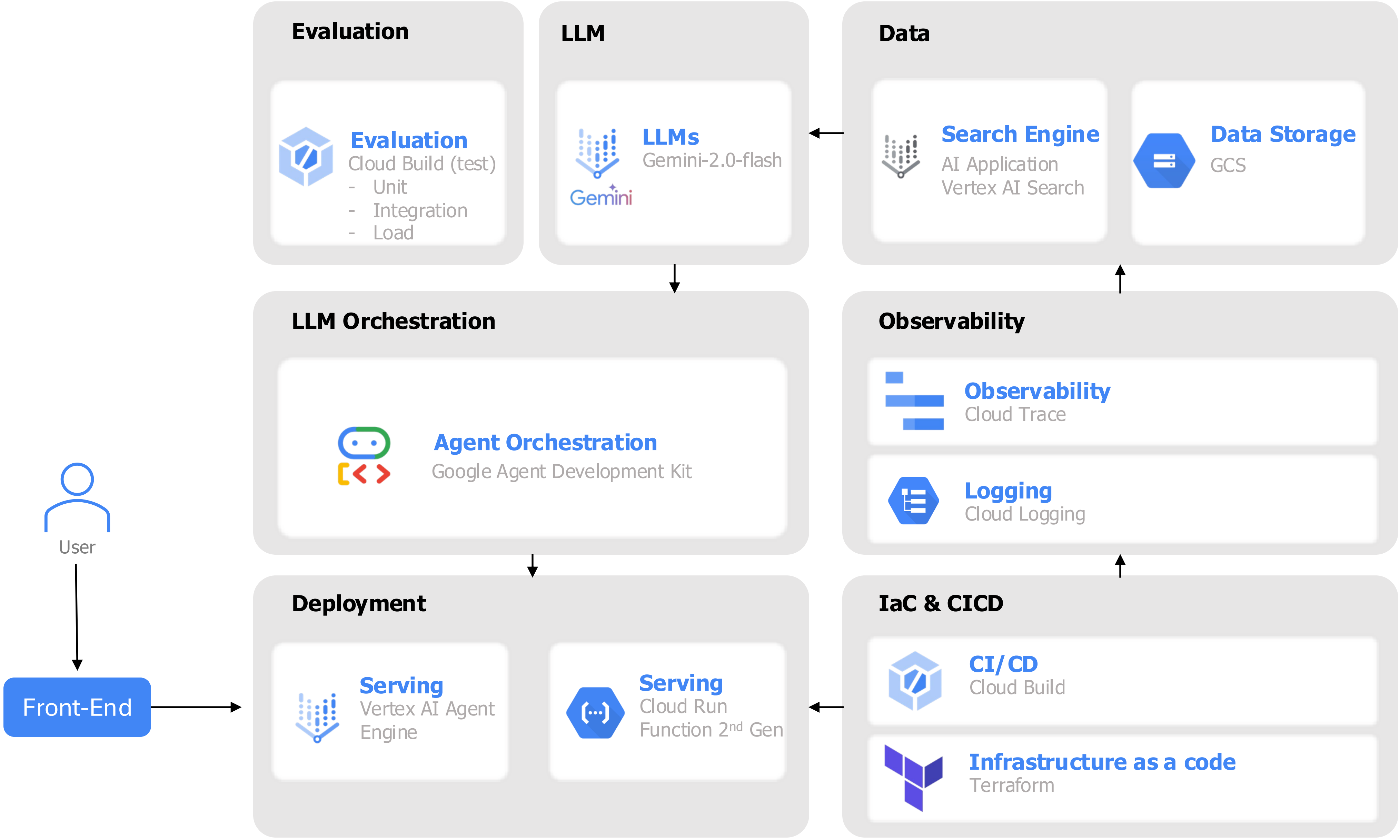
LINE&GCP

Content

- High Level Architecture
- Conversation Flow
  - Services Walk through
  - Setup – Line OA
  - Setup – GCP Services
    - Provision services
    - Provision services details
  - Session Management
- Development
- CI/CD – Dev to Production

High Level Architecture

High Level Architecture



Evaluation

Evaluation
Cloud Build (test)
- Unit
- Integration
- Load

LLM

LLMs
Gemini-2.0-flash
Gemini

Data

Search Engine
AI Application
Vertex AI Search

Data Storage
GCS

LLM Orchestration

Agent Orchestration
Google Agent Development Kit

Observability

Observability
Cloud Trace

Logging
Cloud Logging

User

Front-End

Deployment

Serving
Vertex AI Agent Engine

Serving
Cloud Run
Function 2nd Gen

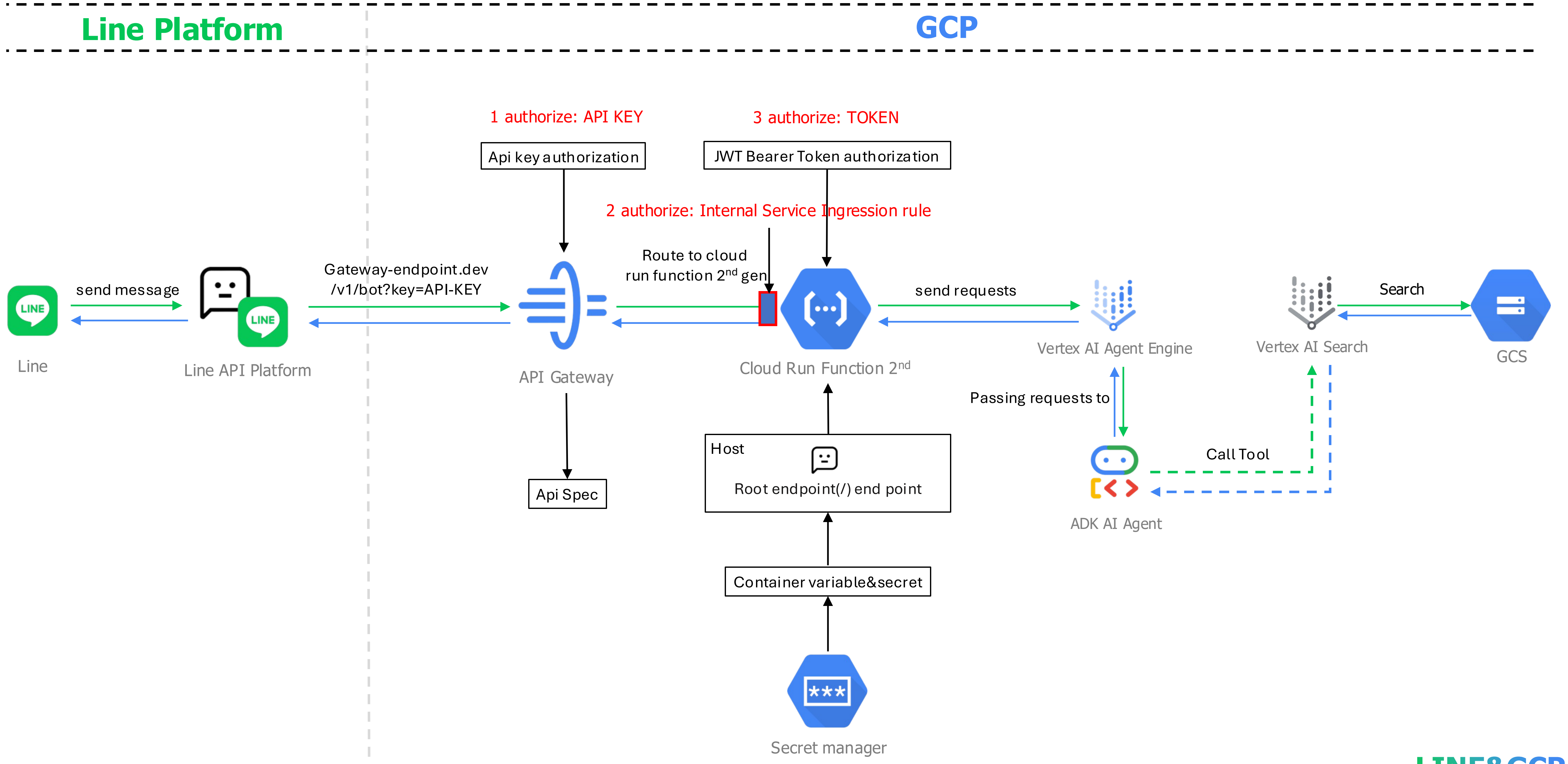
IaC & CICD

CI/CD
Cloud Build

Infrastructure as a code
Terraform

Conversation Flow

Conversation flow - Services Walkthrough

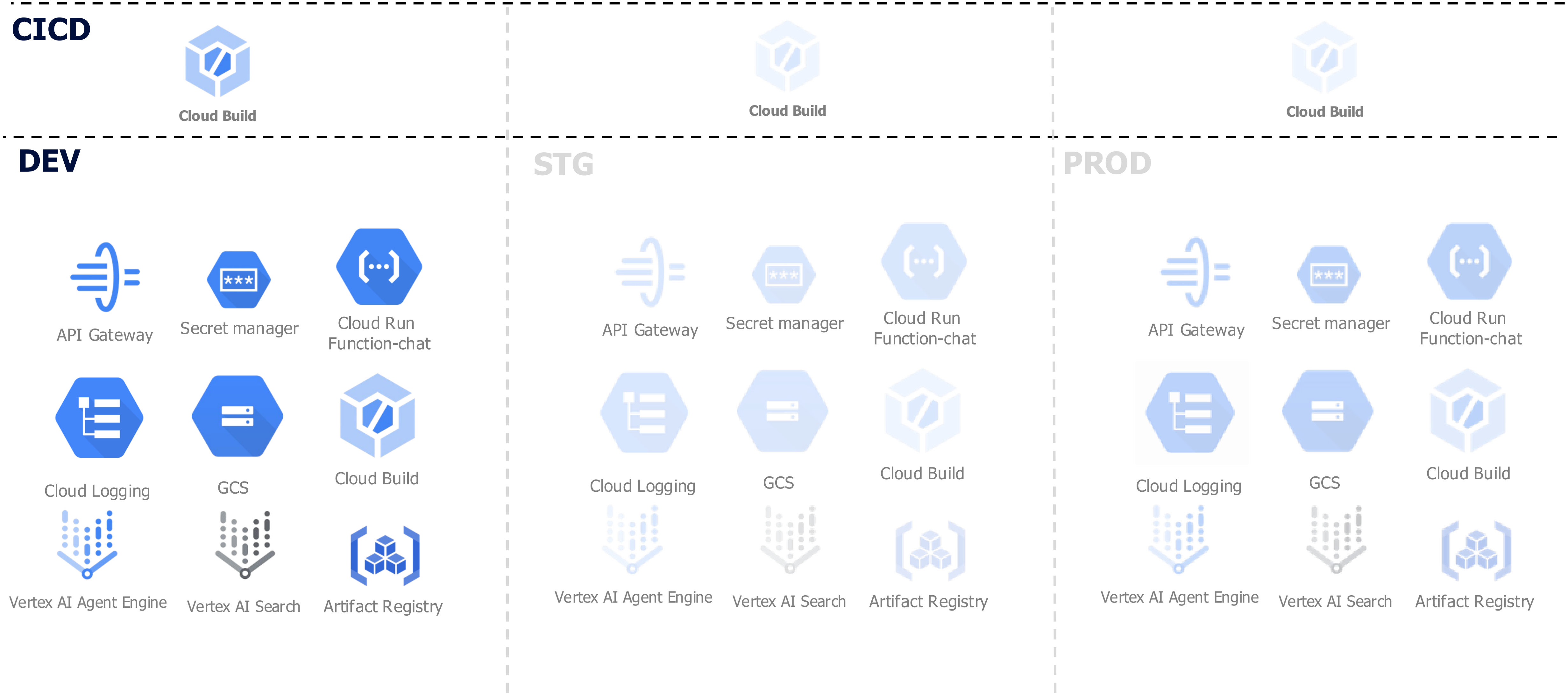


LINE&GCP

Setup Line OA

Setup GCP Services

Setup GCP – IAC (Dev)



CICD

Cloud Build

Cloud Build

Cloud Build

DEV

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

Cloud Build

Vertex AI Agent Engine

Vertex AI Search

Artifact Registry

STG

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

Cloud Build

Vertex AI Agent Engine

Vertex AI Search

Artifact Registry

PROD

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

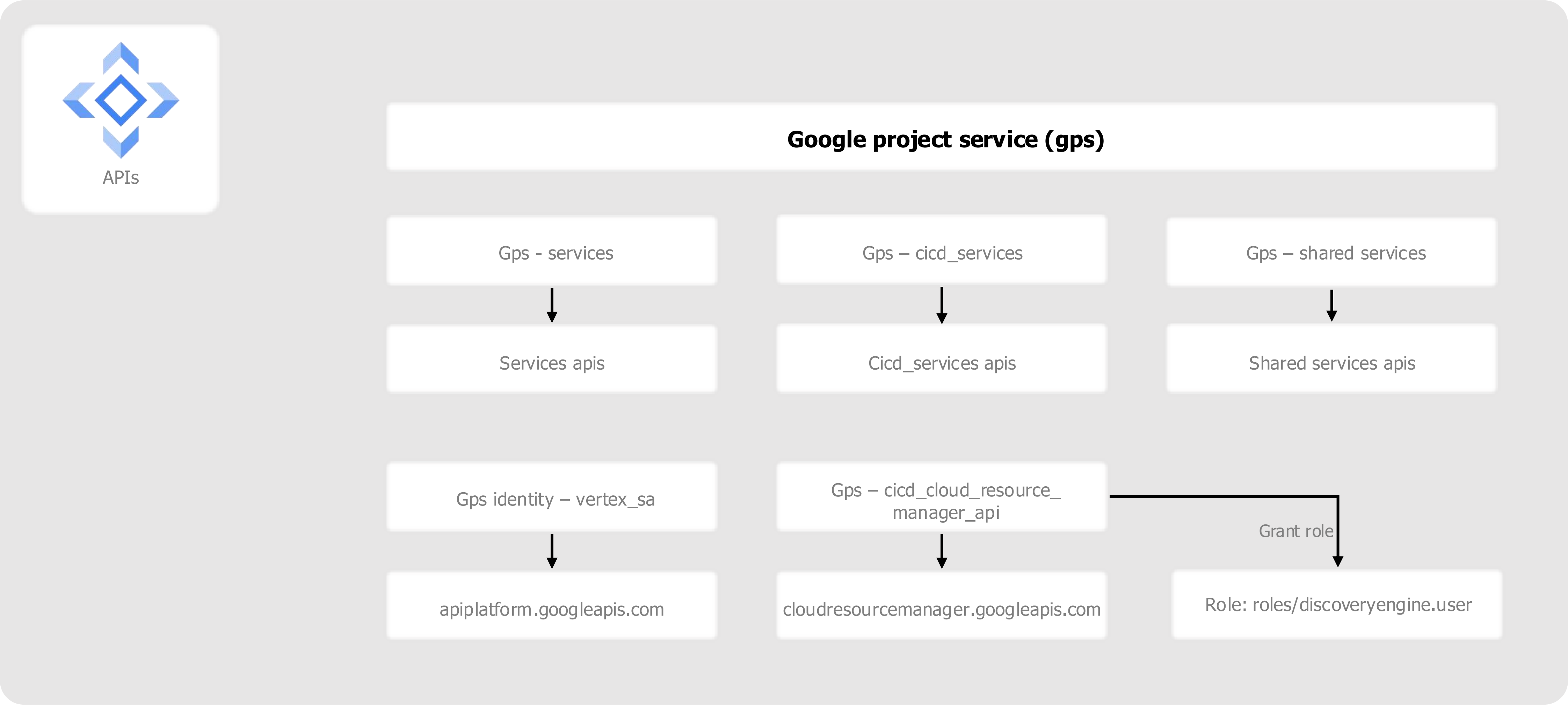
Cloud Build

Vertex AI Agent Engine

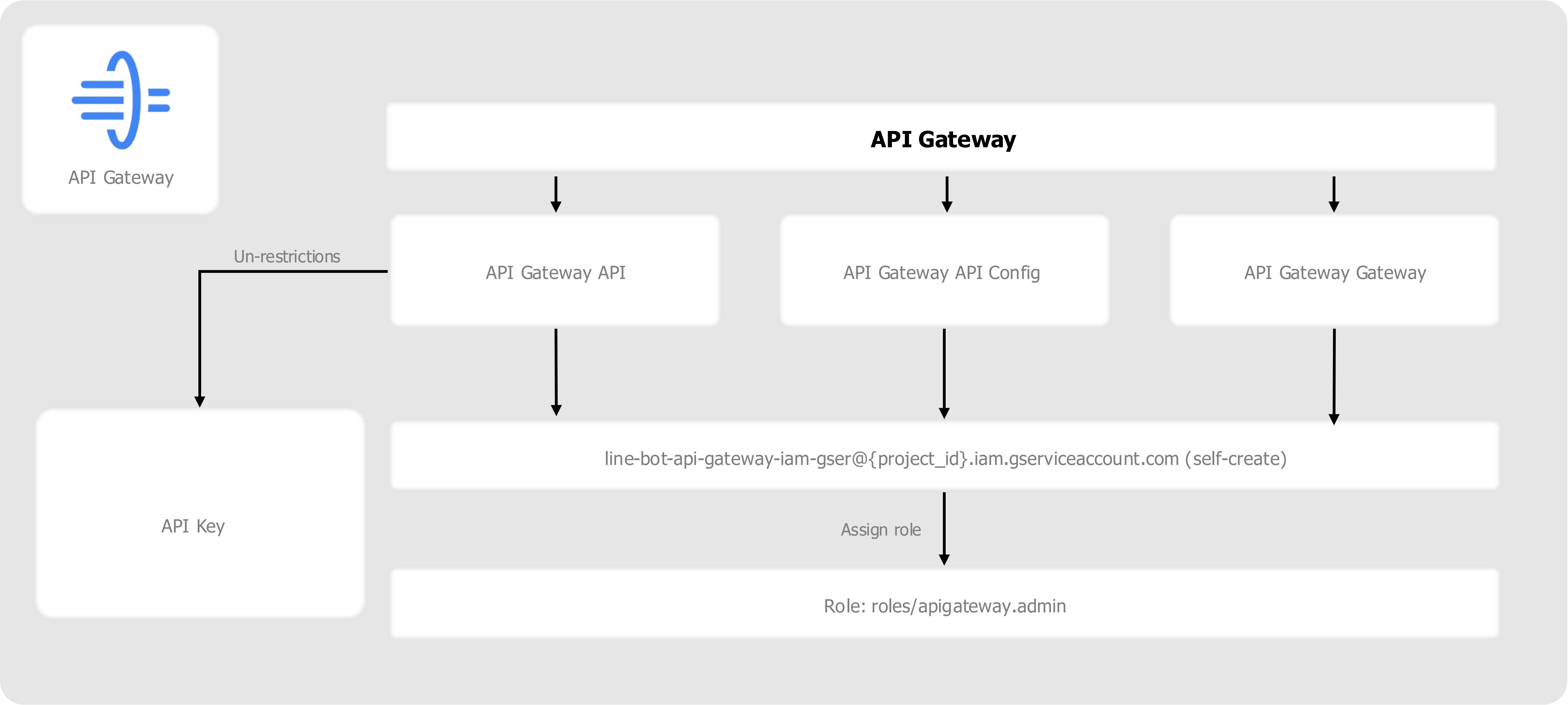
Vertex AI Search

Artifact Registry

Setup GCP – Terraform Creation: APIs


APIs
Google project service (gps)
Gps - services
Services apis
Gps – cicd_services
Cicd_services apis
Gps – shared services
Shared services apis
Gps identity – vertex_sa
apiplatform.googleapis.com
Gps – cicd_cloud_resource_ manager_api
cloudresourcemanager.googleapis.com
Grant role
Role: roles/discoveryengine.user

Setup GCP – Terraform Creation: API Gateway



API Gateway

API Gateway

Un-restrictions

API Gateway API

API Gateway API Config

API Gateway Gateway

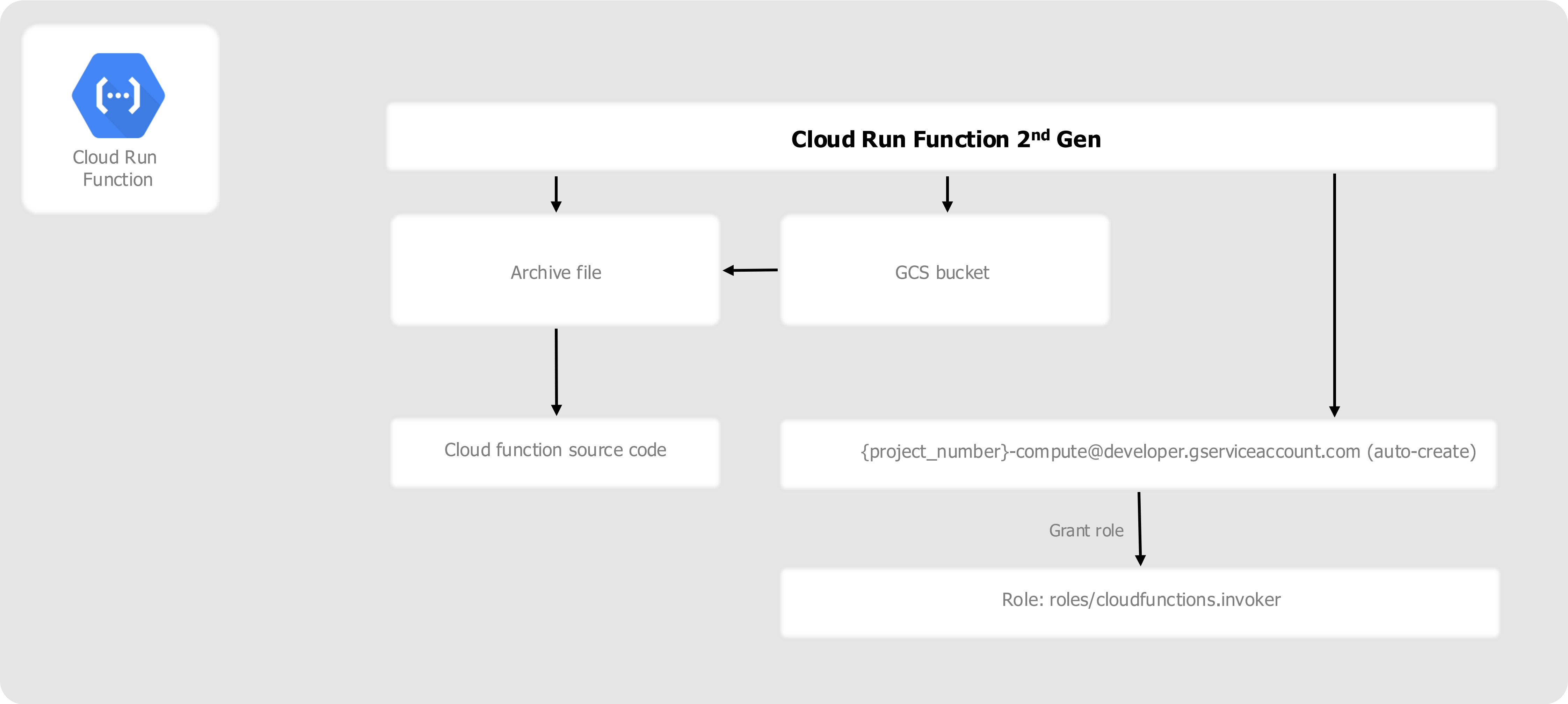
API Key

line-bot-api-gateway-iam-gser@{project_id}.iam.gserviceaccount.com (self-create)

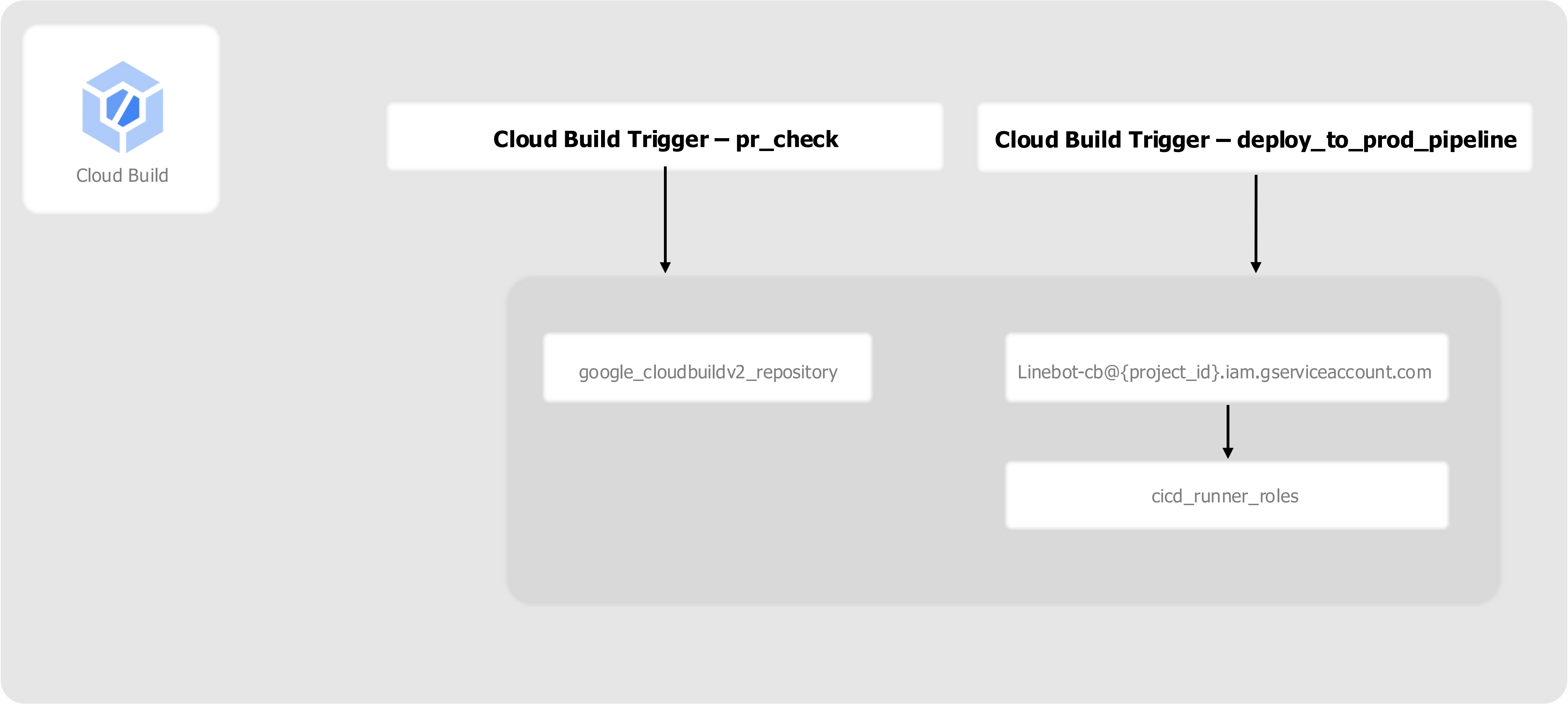
Assign role

Role: roles/apigateway.admin

Setup GCP – Terraform Creation: Cloud Run Function 2nd Gen


Cloud Run
Function
Cloud Run Function 2nd Gen
Archive file
GCS bucket
Cloud function source code
{project_number}-compute@developer.gserviceaccount.com (auto-create)
Grant role
Role: roles/cloudfunctions.invoker

Setup GCP – Terraform Creation: Cloud Build Trigger



Cloud Build

Cloud Build Trigger – pr_check

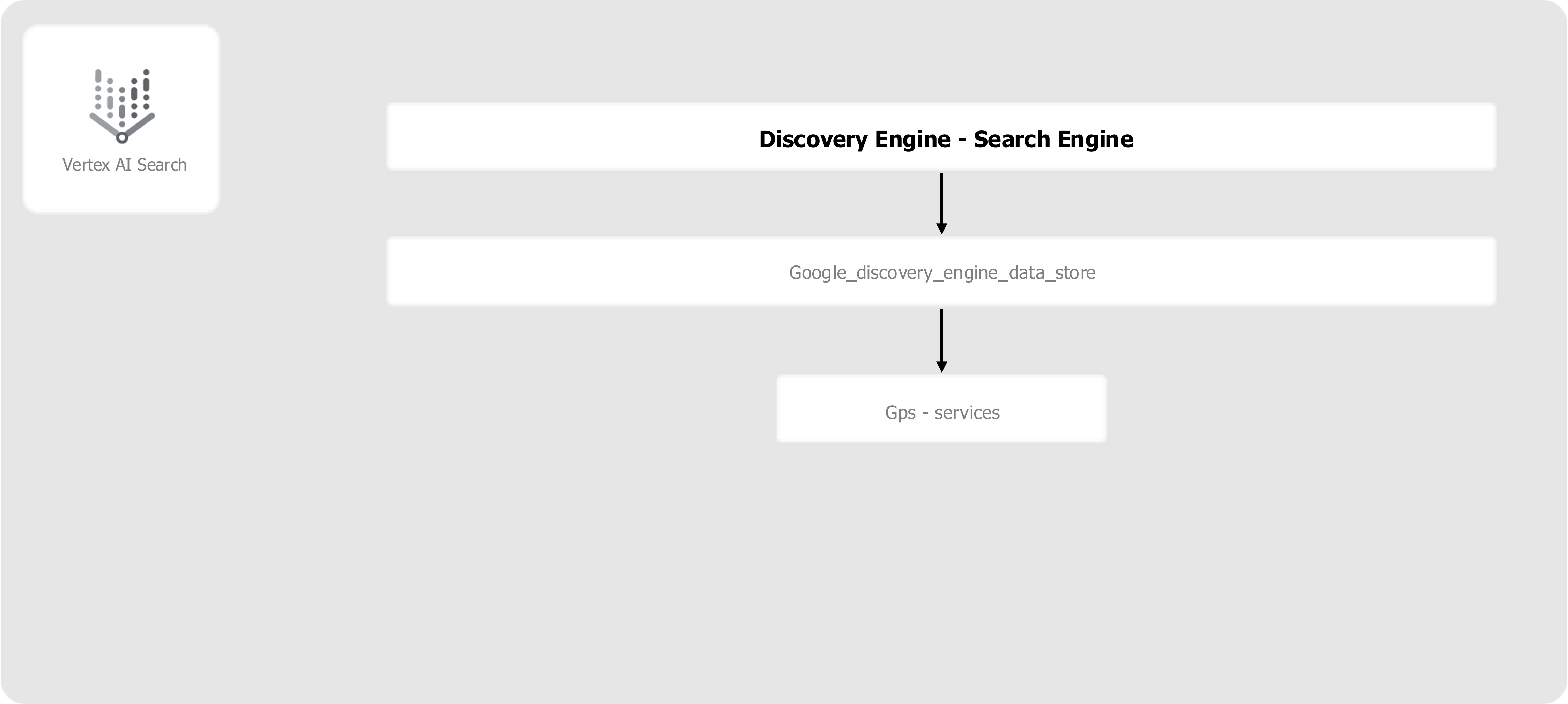
Cloud Build Trigger – deploy_to_prod_pipeline

google_cloudbuildv2_repository

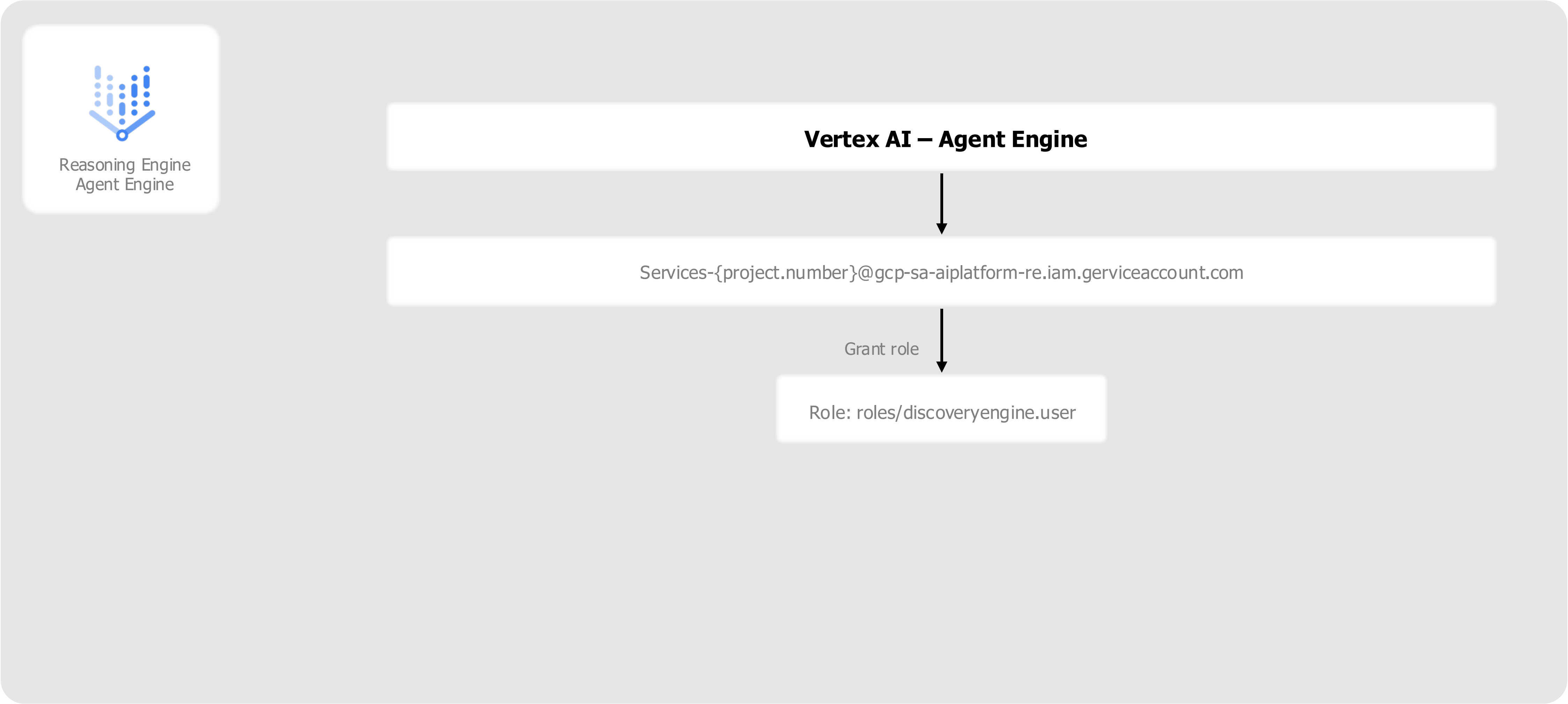
Linebot-cb@{project_id}.iam.gserviceaccount.com

cicd_runner_roles

Setup GCP – Terraform Creation: Search Engine


Vertex AI Search
Discovery Engine - Search Engine
Google_discovery_engine_data_store
Gps - services

Setup GCP – Terraform Creation: Reasoning Engine



Reasoning Engine
Agent Engine

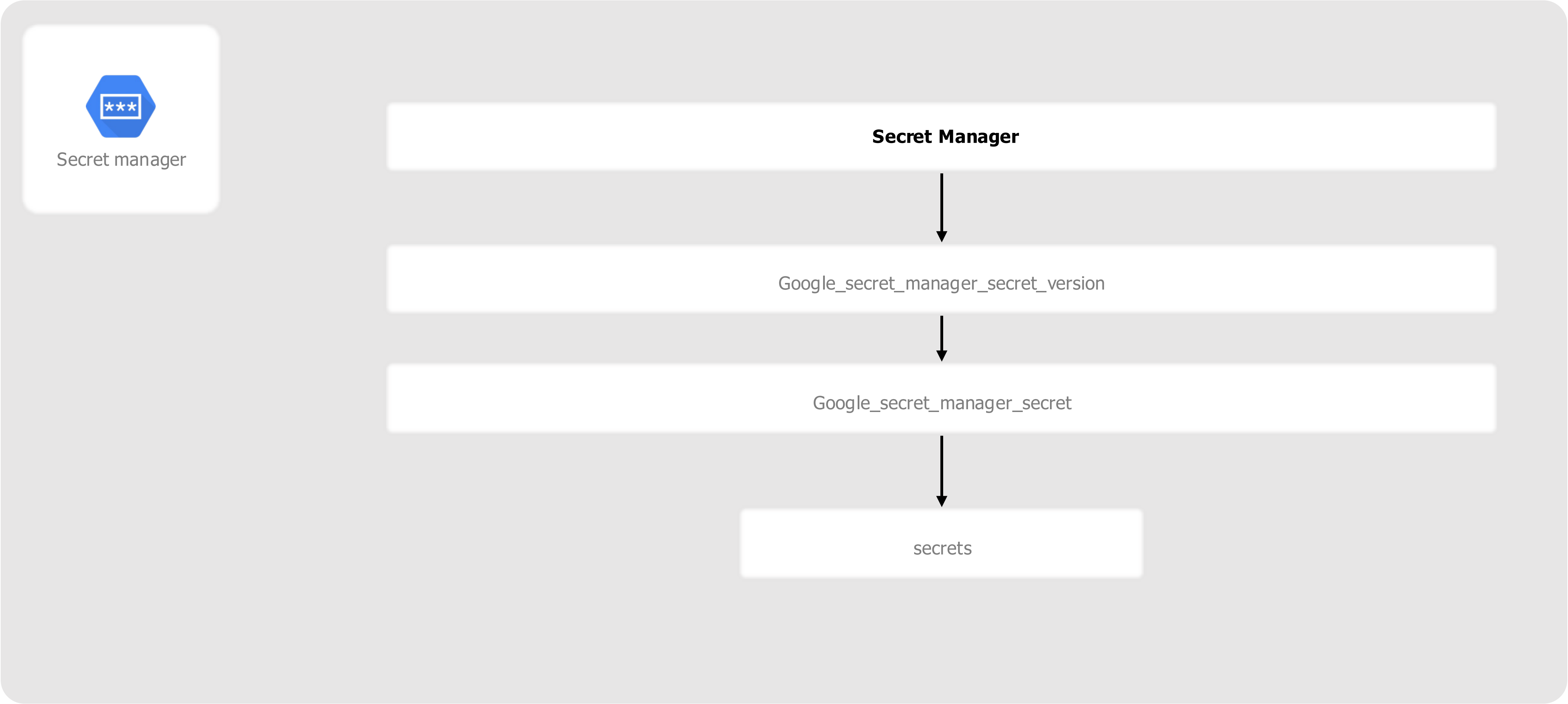
Vertex AI – Agent Engine

Services-{project.number}@gcp-sa-aiplatform-re.iam.gerviceaccount.com

Grant role

Role: roles/discoveryengine.user

Setup GCP – Terraform Creation: Secret Manager



Secret manager

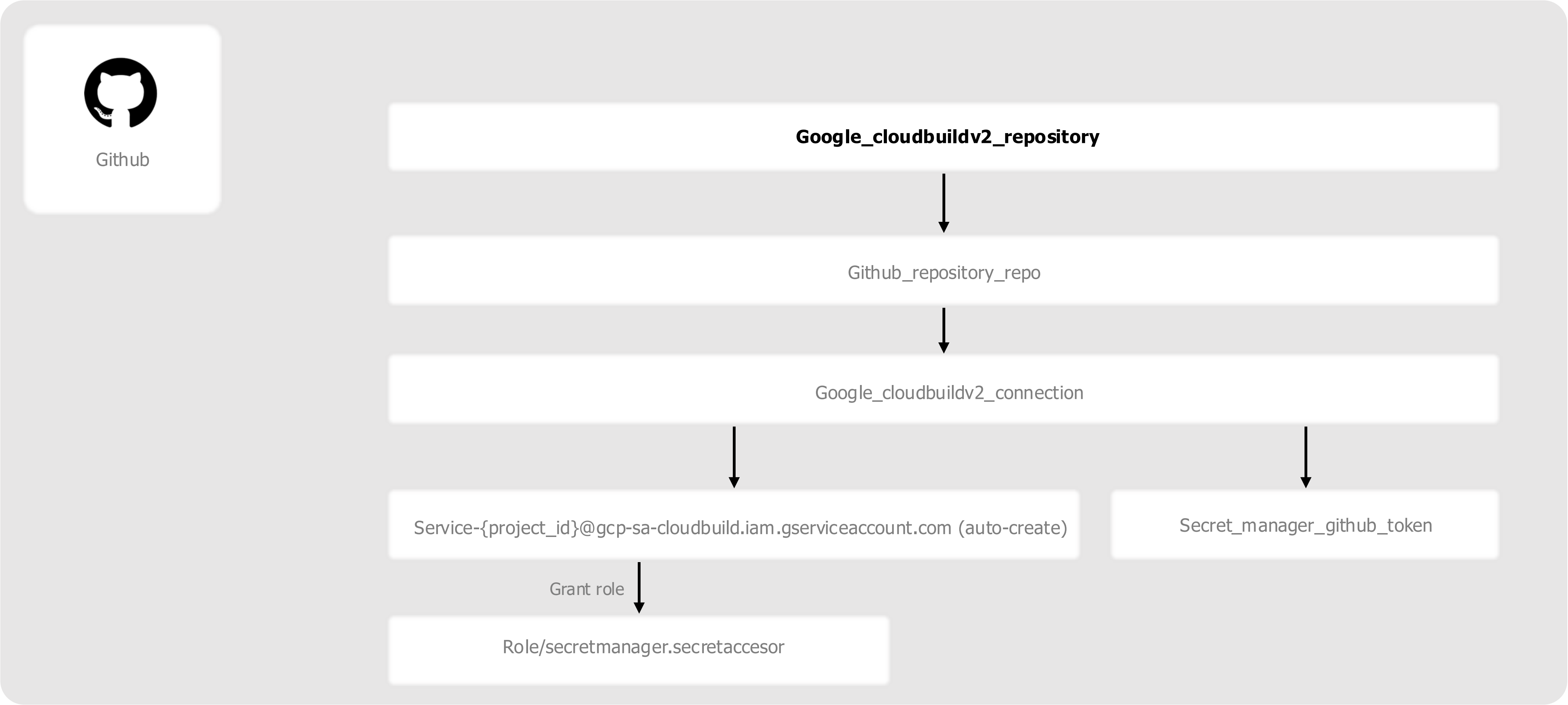
Secret Manager

Google_secret_manager_secret_version

Google_secret_manager_secret

secrets

Setup GCP – Terraform Creation: Github Connection



Github

Google_cloudbuildv2_repository

Github_repository_repo

Google_cloudbuildv2_connection

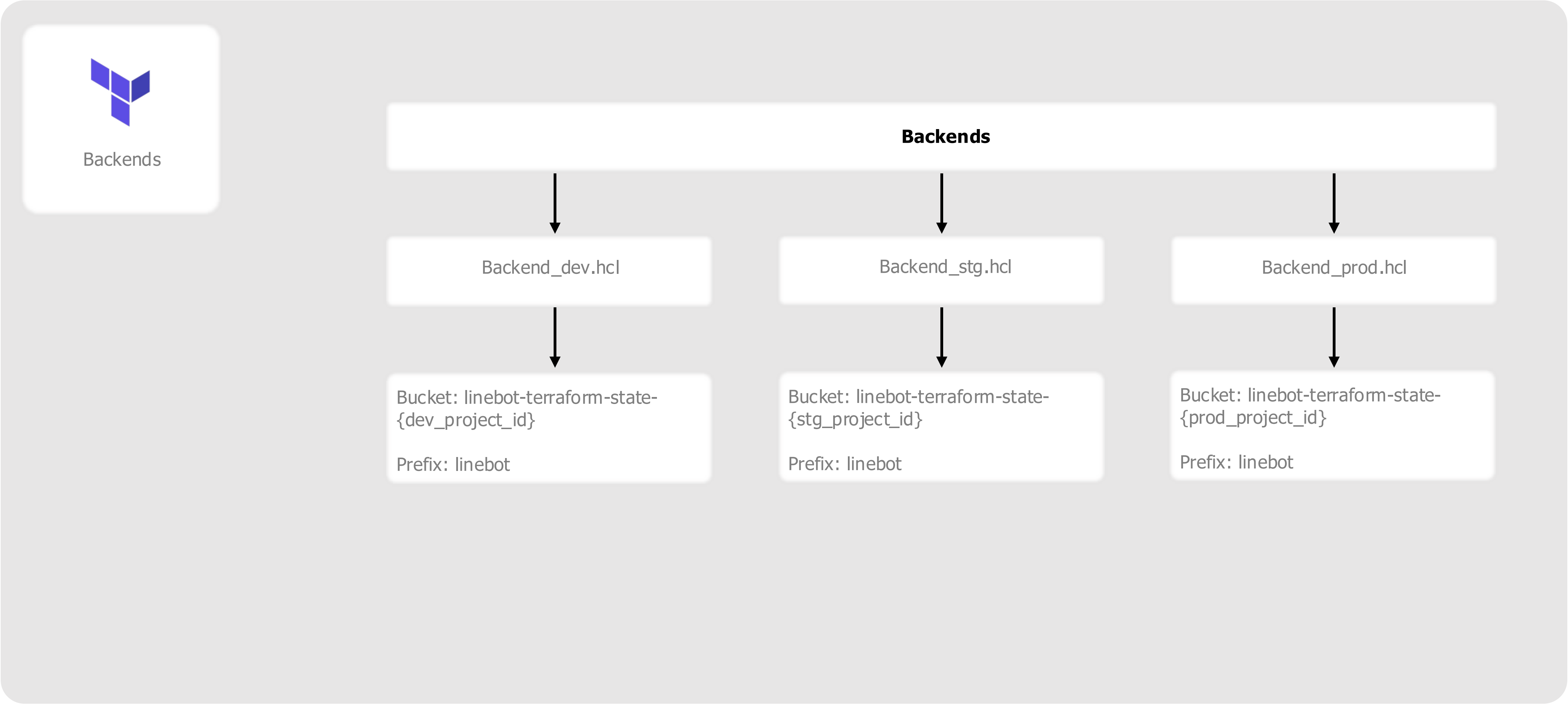
Service-{project_id}@gcp-sa-cloudbuild.iam.gserviceaccount.com (auto-create)

Secret_manager_github_token

Grant role

Role/secretmanager.secretaccesor

Setup GCP – Terraform Creation: Backends



Backends

Backends

Backend_dev.hcl

Backend_stg.hcl

Backend_prod.hcl

Bucket: linebot-terraform-state-{dev_project_id}

Prefix: linebot

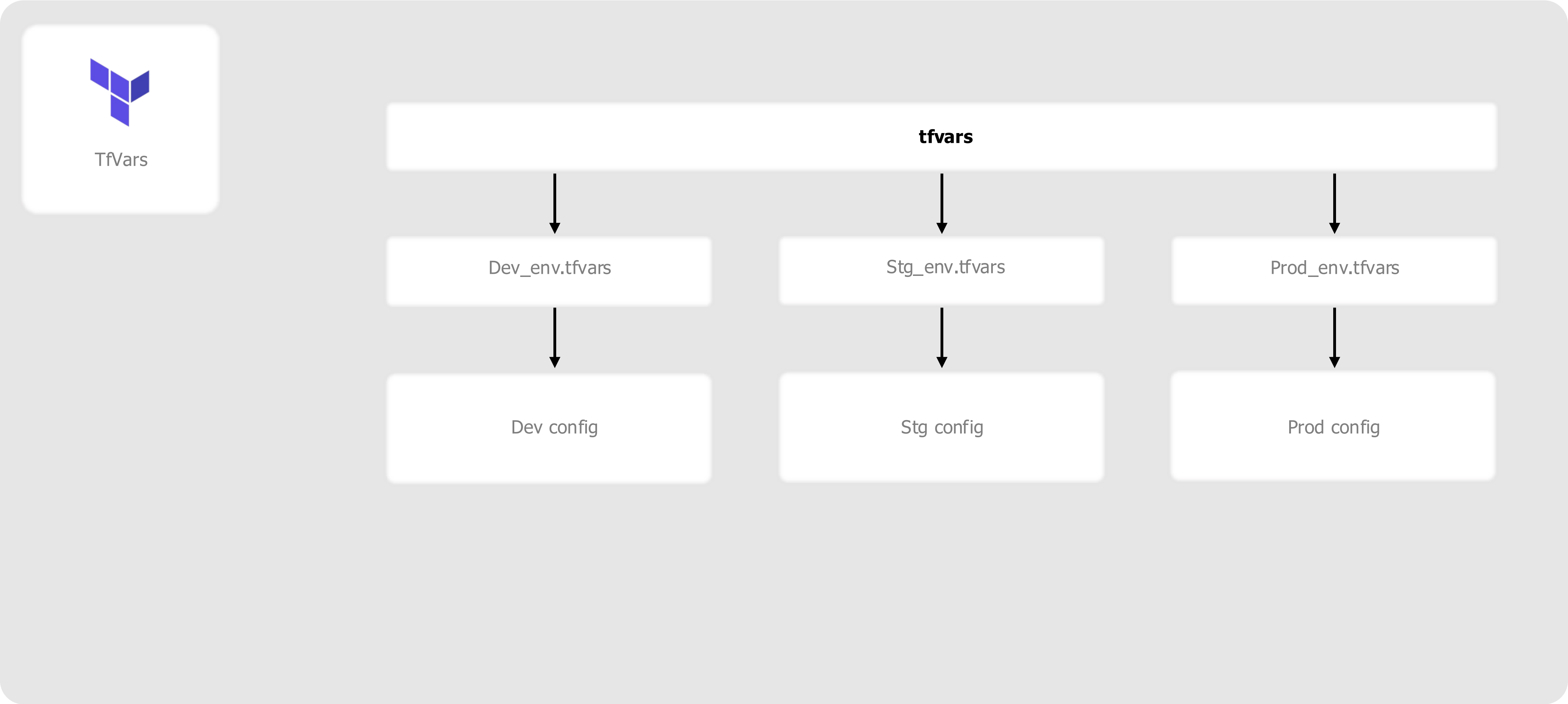
Bucket: linebot-terraform-state-{stg_project_id}

Prefix: linebot

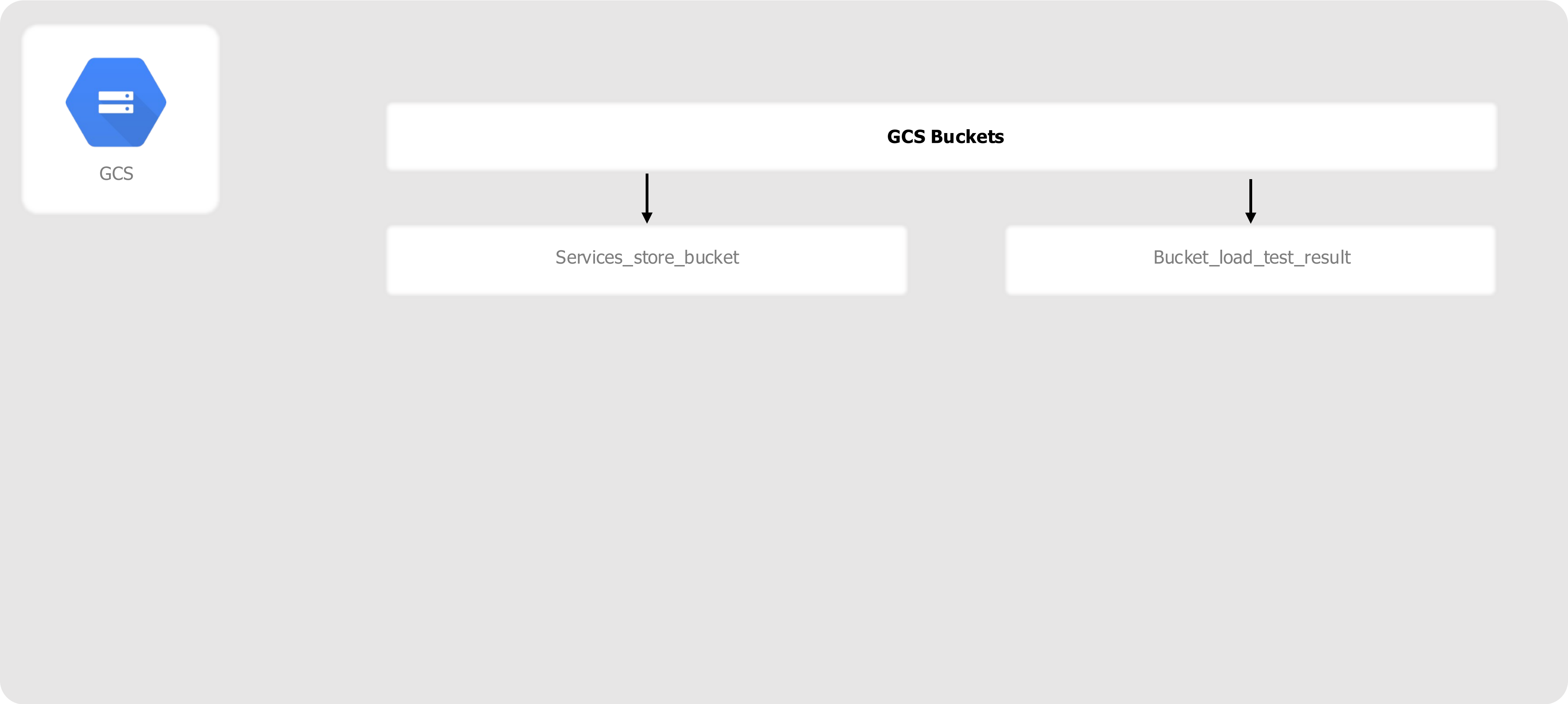
Bucket: linebot-terraform-state-{prod_project_id}

Prefix: linebot

Setup GCP – Terraform Creation: TfVars

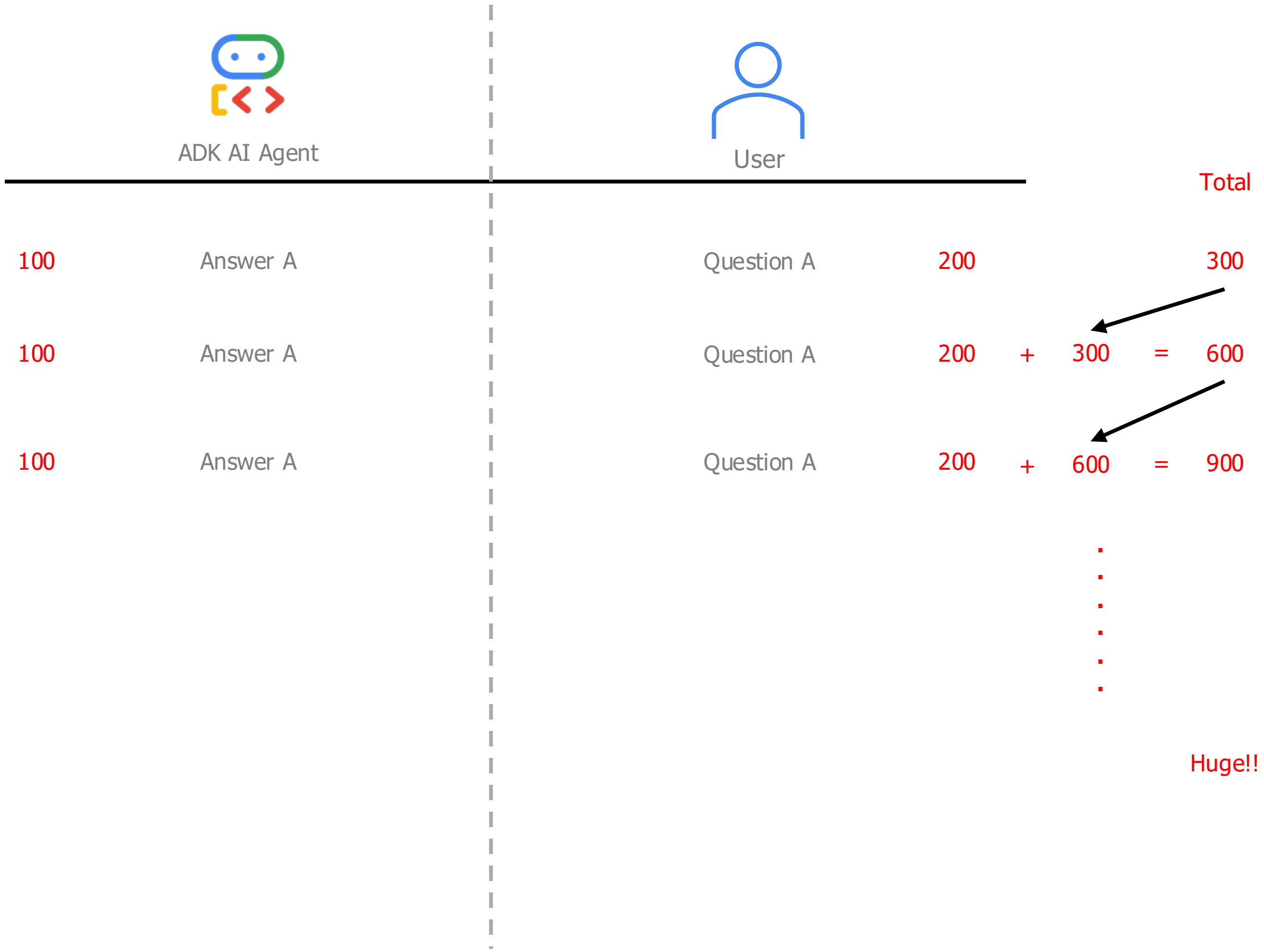


Setup GCP – Terraform Creation: Storages



Session Management

Session Management

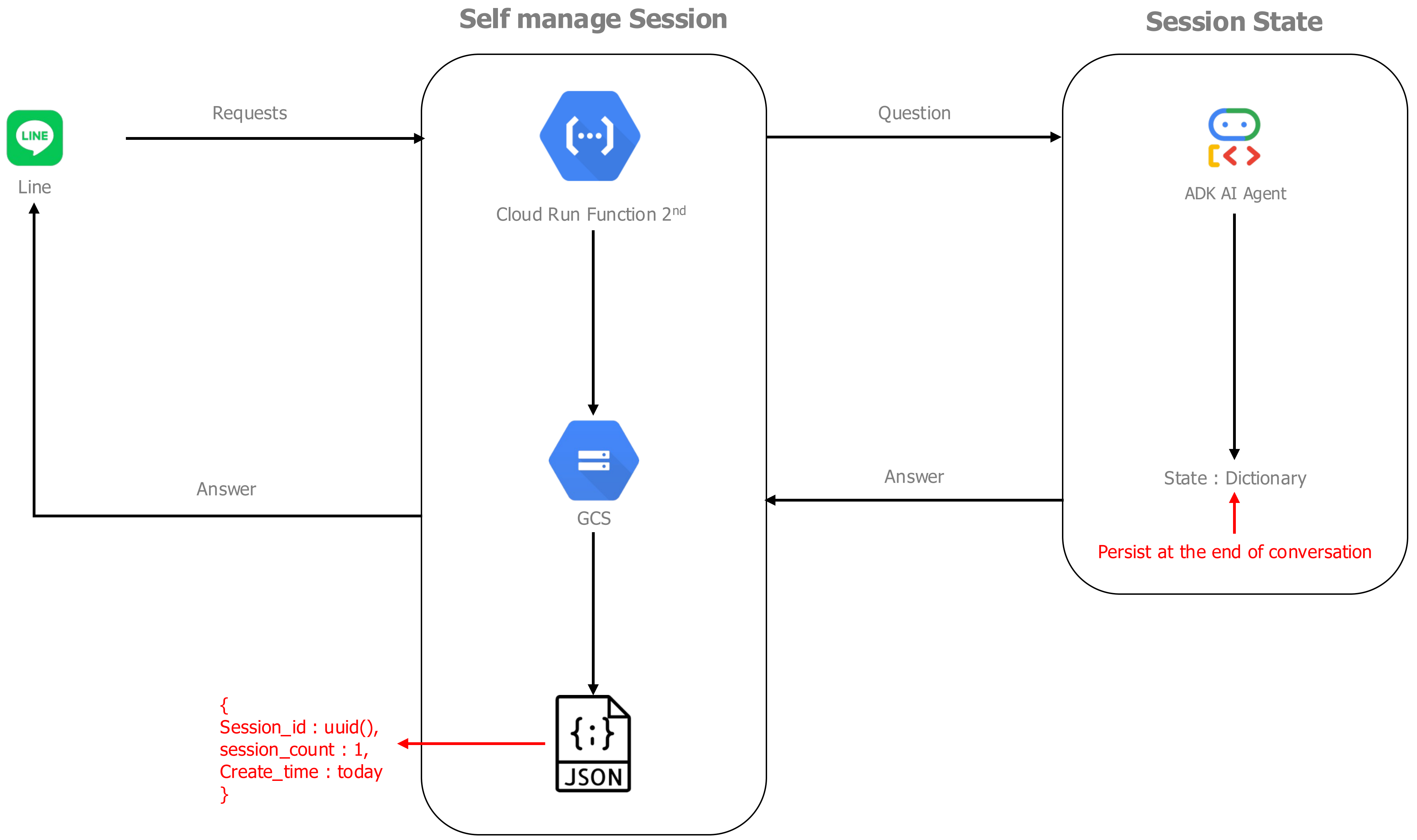


	ADK AI Agent	User					Total
100	Answer A	Question A	200				300
100	Answer A	Question A	200	+	300	=	600
100	Answer A	Question A	200	+	600	=	900

.
.
.
.
.
.

Huge!!

Session Management

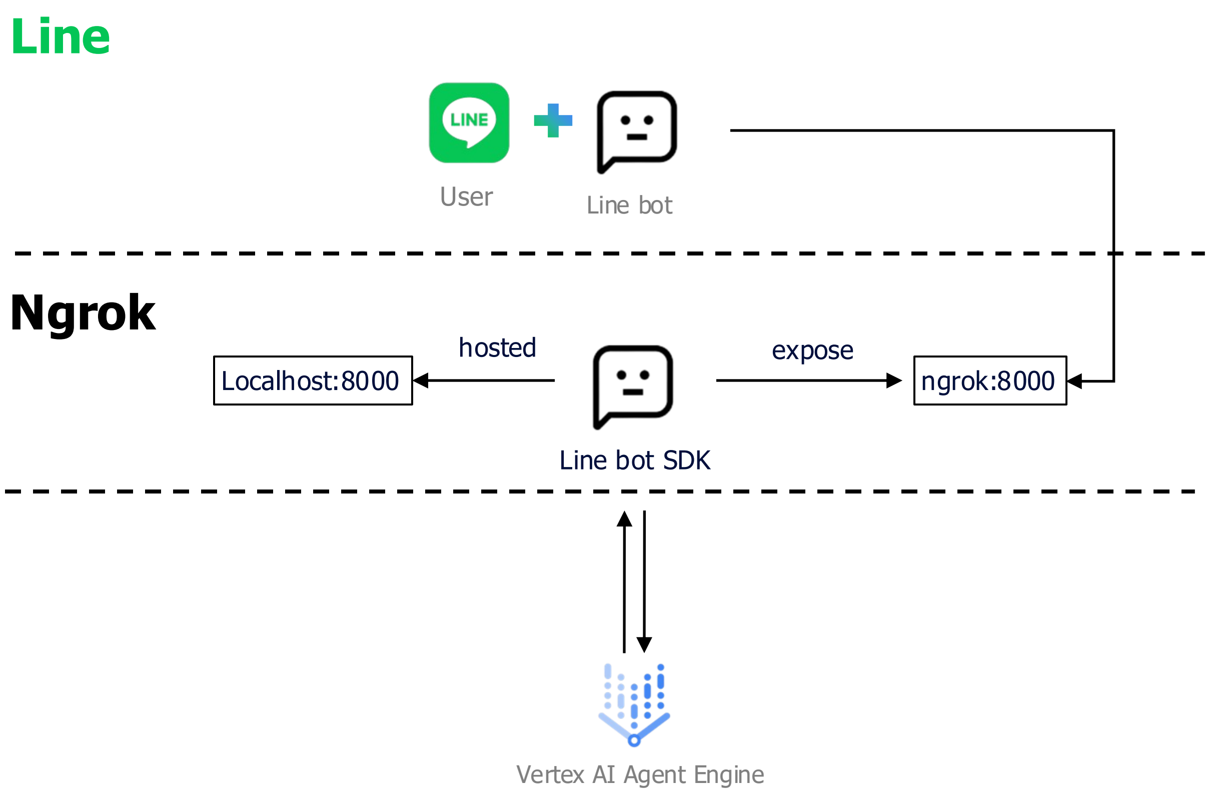


LINE&GCP

Development

Development - Local VS Cloud

Local



Line

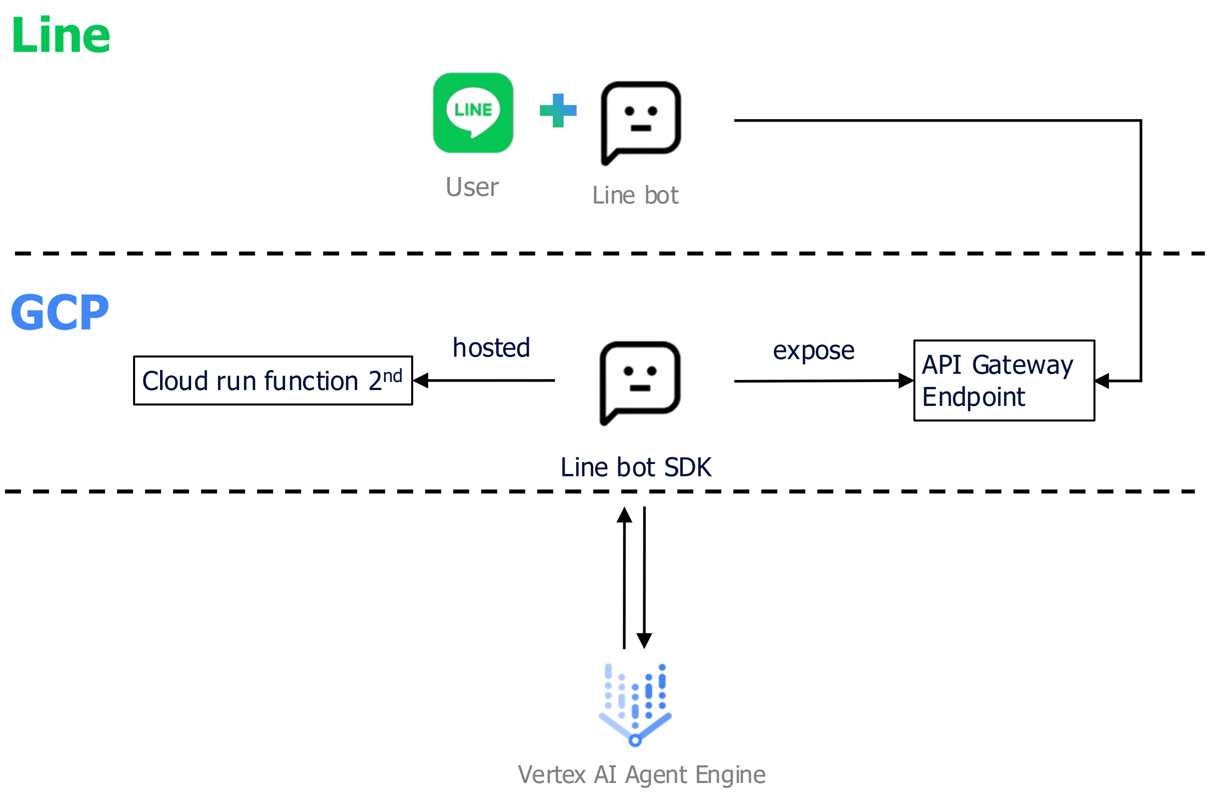
User + Line bot

Ngrok

Localhost:8000 ← hosted — Line bot SDK — expose → ngrok:8000

Vertex AI Agent Engine

Cloud



Line

User + Line bot

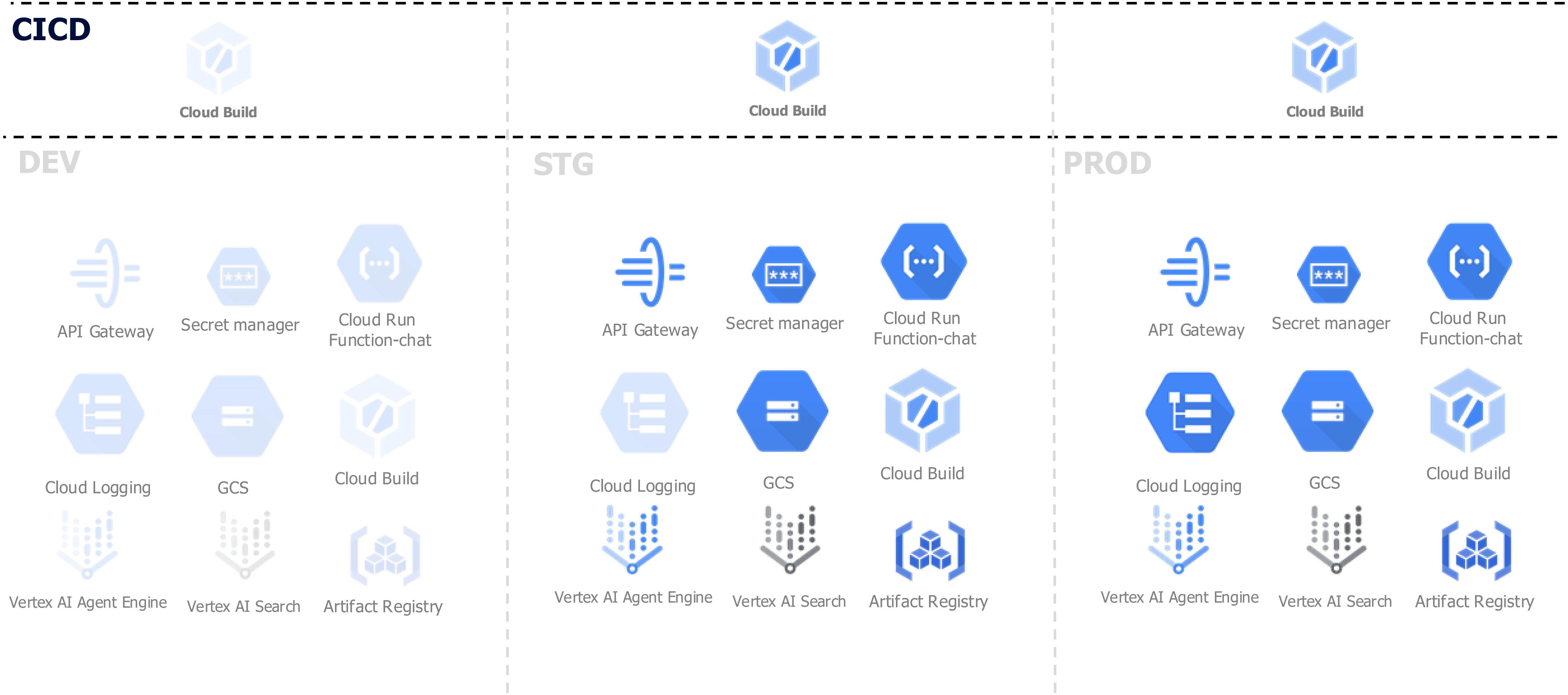
GCP

Cloud run function 2nd ← hosted — Line bot SDK — expose → API Gateway Endpoint

Vertex AI Agent Engine

CI/CD (Delivery)

CI/CD (Delivery) – IAC (Stg- Prod)



CICD

Cloud Build

Cloud Build

Cloud Build

DEV

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

Cloud Build

Vertex AI Agent Engine

Vertex AI Search

Artifact Registry

STG

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

Cloud Build

Vertex AI Agent Engine

Vertex AI Search

Artifact Registry

PROD

API Gateway

Secret manager

Cloud Run Function-chat

Cloud Logging

GCS

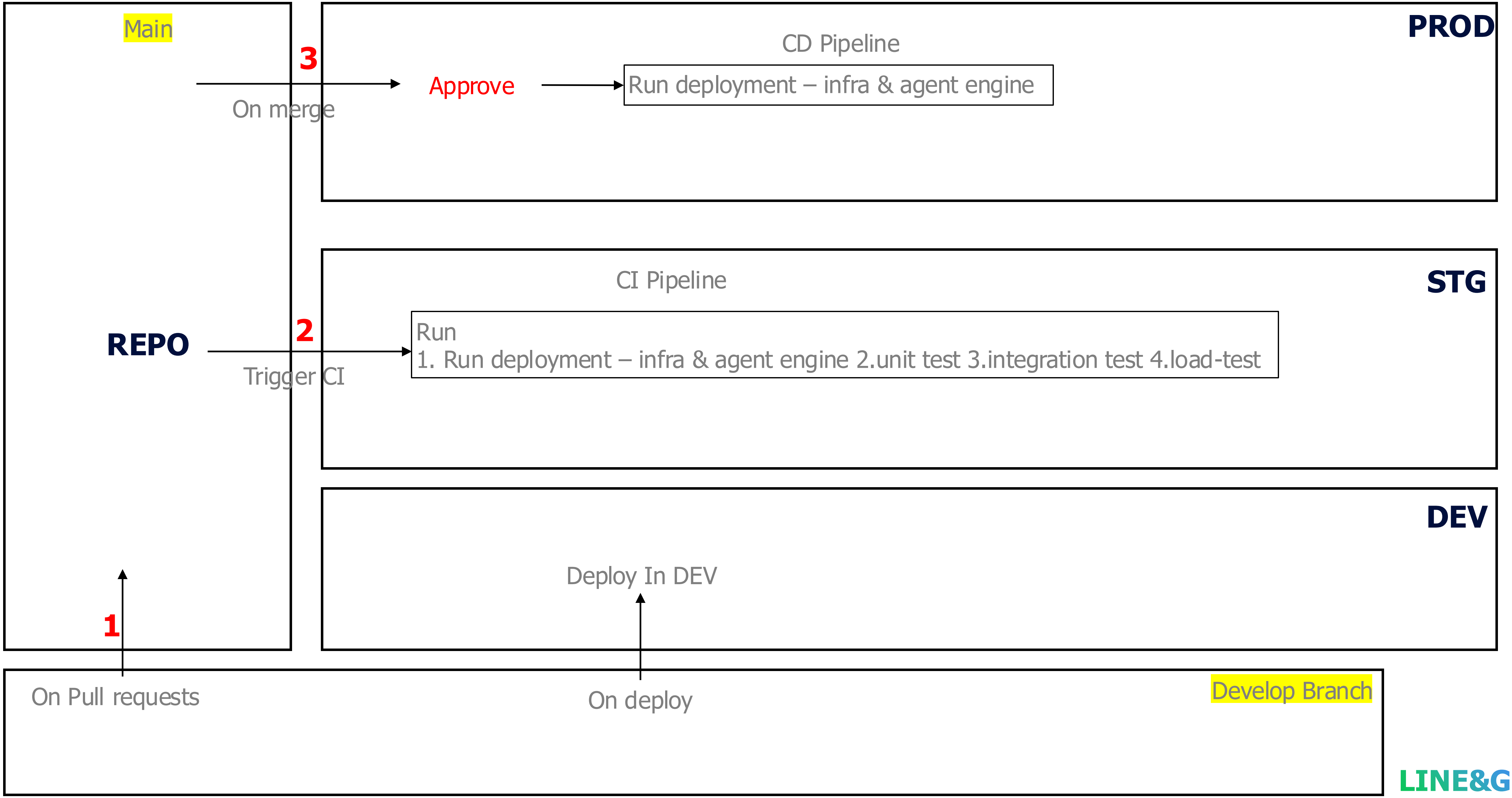
Cloud Build

Vertex AI Agent Engine

Vertex AI Search

Artifact Registry

CI/CD (Delivery)



Main

REPO

PROD

CD Pipeline

3 On merge → Approve → Run deployment – infra & agent engine

STG

CI Pipeline

2 Trigger CI → Run
1. Run deployment – infra & agent engine 2.unit test 3.integration test 4.load-test

DEV

Deploy In DEV

1 On Pull requests

On deploy

Develop Branch

End

CICD Note

0.create branch: git checkout -b feature/my-change
1.make change in tracked file
2.git add -A
3.git commit –m ‘commit message'
4.git push origin feature/my-change
5.gh pr create --base main --head feature/my-change --title "My feature" --body "Description of my feature" --> in this step cloud build pr_checks(CI pipeline) will be invoke
6.when merge branch into main cd_pipeline, deploy_to_prod_pipeline(CD pipeline) triggers will be invoke

Manual Enable APIs For Each Project Id

1. Vertex ai api
2. Secret Manager API
3. Api gateway api
4. Api key api
5. Cloud Build API
6. Cloud Resource Manager API

Setup Step

1. Set project id to <dev_project> and Set quota project ‘add gcloud auth application-default set-quota-project <dev_project>’
2. gcloud auth application-default login
3. cd deployment && python3 setup_github_token.py
4. Fill value and manual create(if any)
    1. fill pat secret id
    2. Create connection to git repo(if any)
5. gcloud storage buckets create gs://true-askhr-terraform-state-{dev_project-id} --project={dev_project-id} --location=us-central1
6. Adjust backend_<env>.hcl file and tfvars file to correct environment
7. Cd deployment/terraform && Terraform init -backend-config=backends/backend_<env>.hcl
8. terraform plan —var-file vars/{env}_env.tfvars
9. terraform apply—var-file vars/{env}_env.tfvars
10. Enable private api in api library
11. Repeat step 1- 6 for stagging and prod and apply to GCP

12.rm -rf .terraform && terraform init -backend-config=backends/backend_<env>.hcl

Development - Update&Deploy

Every time change tracking folder or file
You need to run follow

1. terraform init -reconfigure -backend-config=backends/backend_dev.hcl
2. terraform plan --var-file vars/dev_env.tfvars
3. terraform apply --var-file vars/dev_env.tfvars