Dominando Cloud Functions

Stack

Google Cloud

Quem é o Gabriel?

- Engenheiro de dados na Avenue Code.
- Carioca, torcedor do Flamengo.
- Formado em engenharia eletrônica no Cefet/RJ.
- Certificações:
  - GCP Professional Data Engineer, Cloud Architect, Developer, DevOps Engineer, Database Engineer, Associate Cloud Engineer e Digital Leader.
  - AWS Solutions Architect Associate e Cloud Practitioner.

Stack

Qual é objetivo do curso ?

- Este curso oferecerá uma visão completa sobre o desenvolvimento de aplicações com **Google Cloud Functions**. Os participantes aprenderão a criar e gerenciar funções serverless, bem como a integrá-las com outros serviços da plataforma Google Cloud, como o Google Cloud Storage, Cloud Build e o BigQuery.

Como o curso está organizado?

Introdução a Google Cloud Platform:

Para começar, vamos fazer uma breve introdução à plataforma de computação em nuvem da Google.

Visão teórica do produto:

Vamos passar por todos os detalhes que você precisa saber antes de começar a usar Cloud Functions.

Práticas com o produto:

A ideia desta seção é começar a experimentar o produto Cloud Functions e conferir algumas das suas funcionalidades .

Implementação de projetos:

Nessa parte do curso iremos criar alguns projetos com Cloud Function e integra-las a outros produtos da Google Cloud e API 's terceiras.

Para o curso, instalar:

- Google Cloud SDK.
- Postman.
- Editor de sua preferência (VS Code, Pycharm etc).
- Anaconda (ou miniconda).

Stack

Aula 1

Stack

Fundamentos de Google Cloud Platform

O que é a Google Cloud Platform?

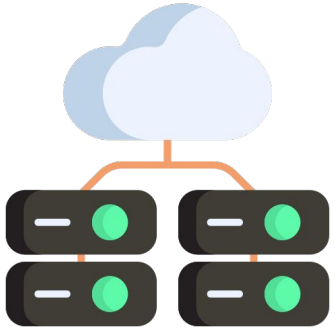
A GCP é uma suíte de serviços de computação que roda na mesma infraestrutura que o Google utiliza para os seus principais serviços, como Gmail, Search Engine e YouTube.

Por que GCP?

- O Google tem tudo a ver com dados e eles são extremamente bons em gerenciar e dimensionar big data, fornecendo soluções flexíveis.
- Free tier realmente Free.
- Atualmente a GCP está com a estratégia de ser mais uma cloud ao invés de a cloud que uma empresa usa.
- É minha cloud favorita 😜

Infraestrutura física

- vCPU.
- Servidor físico.
- Rack.
- Data center.
- Zona.
- Região e Multi Região.
- Rede global do Google.
- Pontos de presença (PoP).
- Sistema Global.

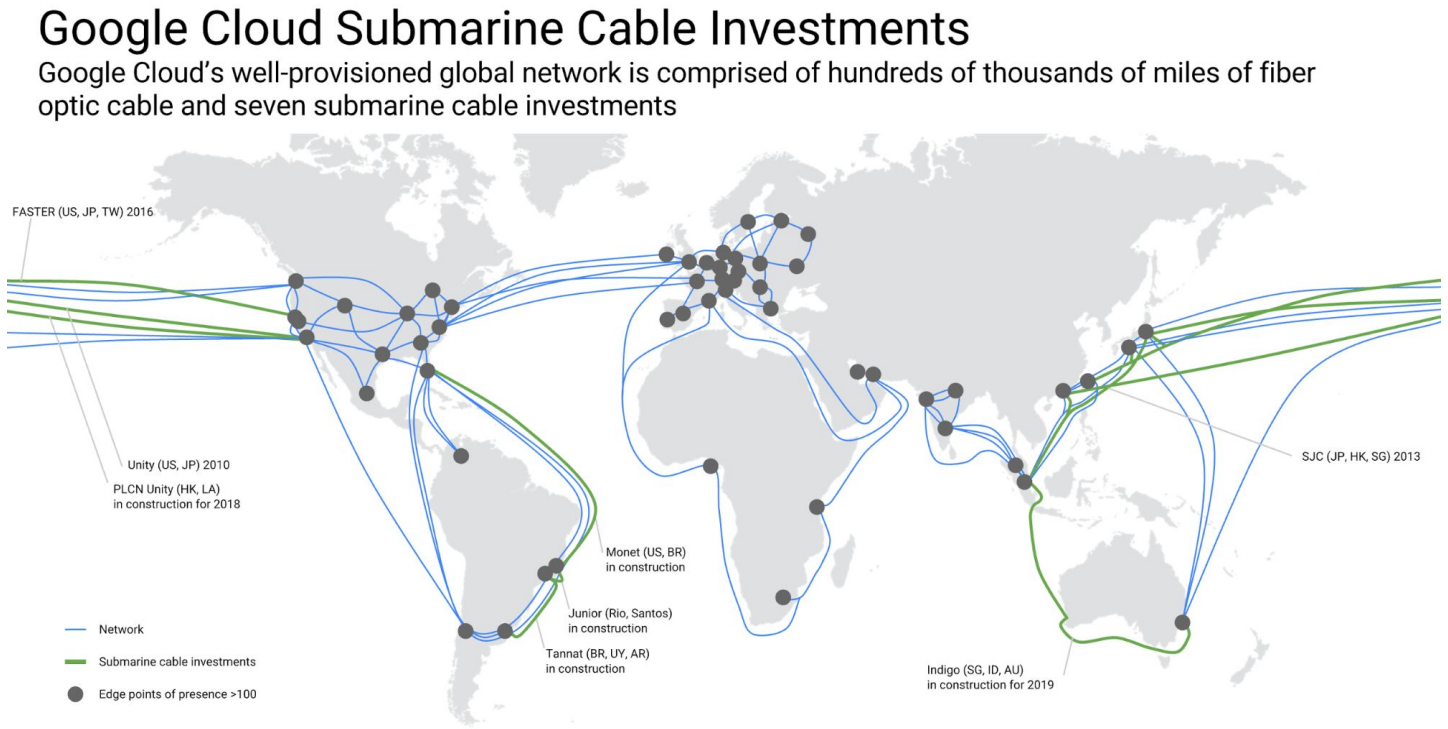


Stack

Regiões da GCP

Fonte da imagem: GCP

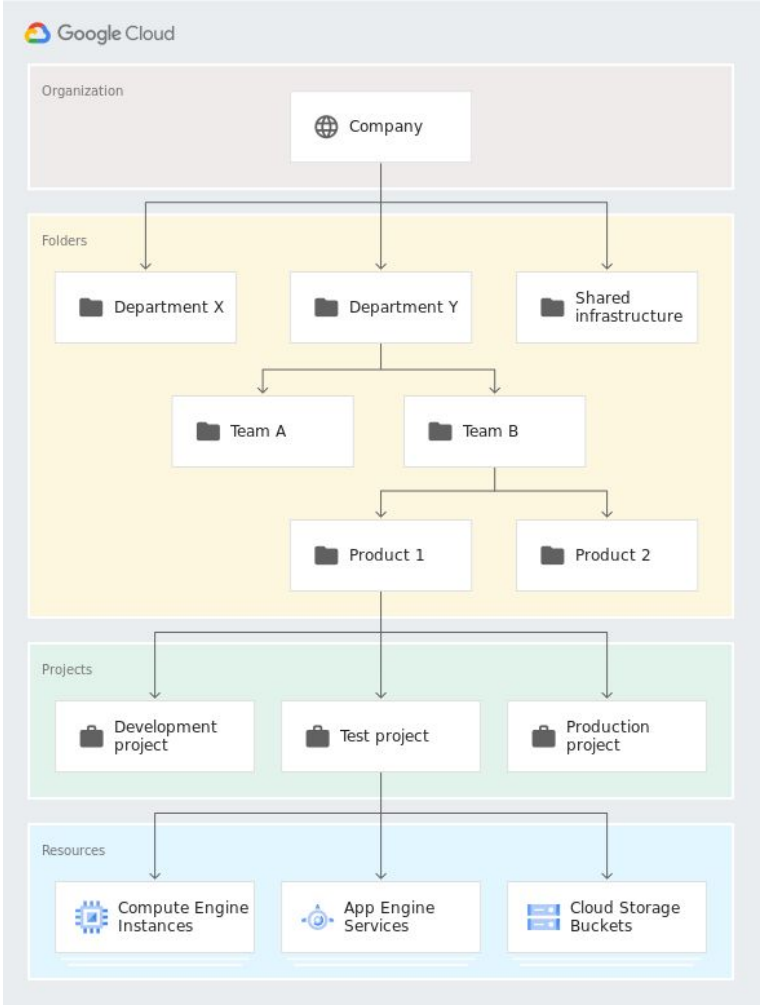
Rede Global



Fonte da imagem: GCP

Organização de recurso na GCP

- Os recursos pertencem a projetos.
- Projetos são semelhantes a contas na AWS.
- Recursos podem ser compartilhados entre projetos.
- Projetos podem ser agrupados e controlados por hierarquia.



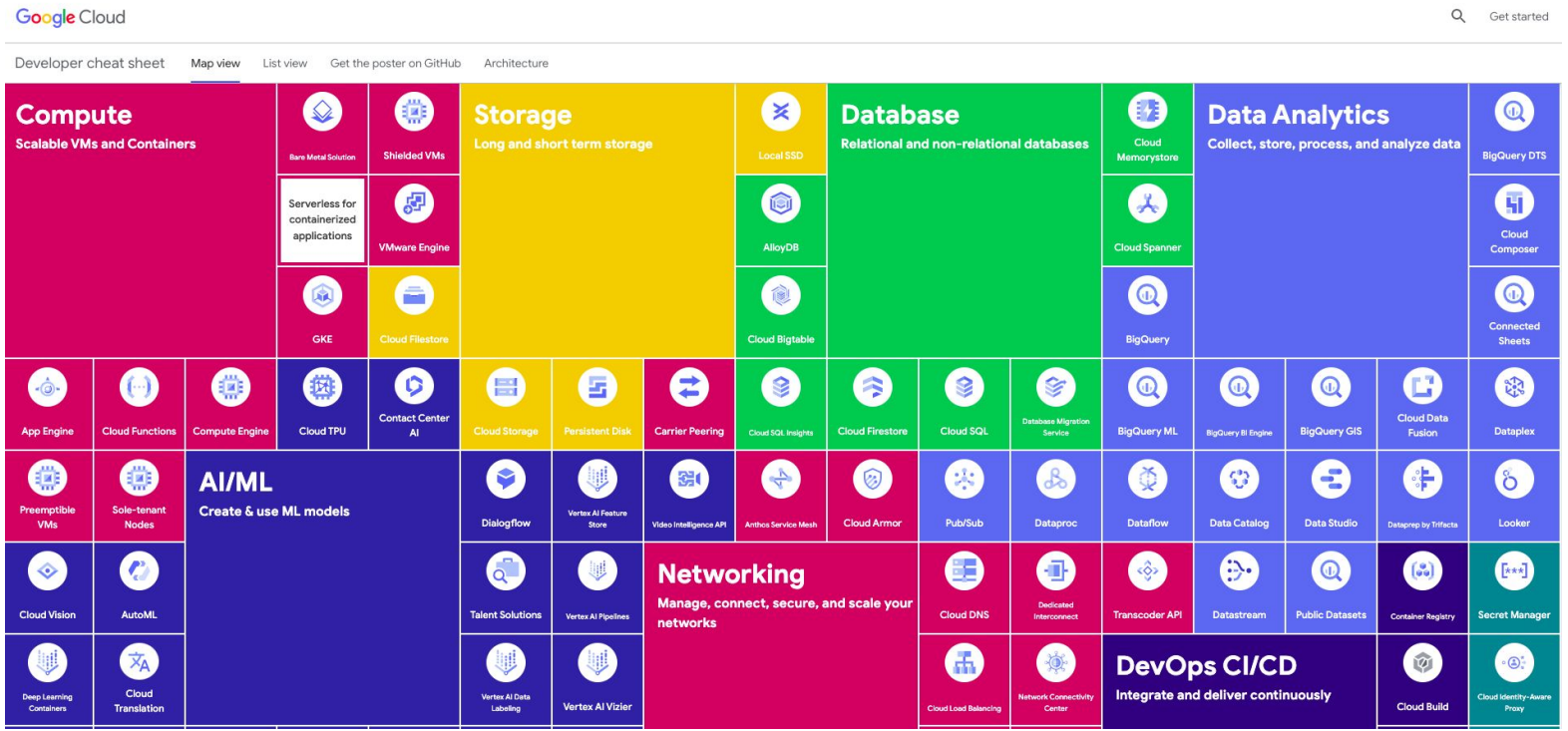
Stack

Como funciona o Pricing na GCP?

- Como funciona o **pagamento** por um **serviço**?
  - Provisionado: "Esteja pronto para lidar com X de carga". Recursos estão lá, usando ou não.
  - Por uso: "Me cobre apenas o que eu usei".
- **Tráfego** de **rede**:
  - De graça na entrada (**ingress**).
  - Cobrado na saída (**egress**), por GBs usado.
  - Egress para outros serviços da GCP frequentemente é de graça, a depender do serviço e da localização.

Stack

The Google Cloud Developer's Cheat Sheet

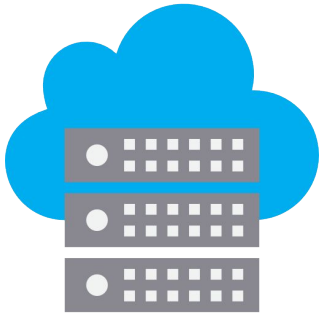


Para outros formatos de cheat sheet: link

Serverless vs Totalmente gerenciado

- **Serverless**: significa "pague conforme usa". Sem tráfego, paga nada, muito tráfego, o escalonamento é automático e paga de acordo com o tráfego.

- **Totalmente gerenciado**: sempre existe um número mínimo de VMs/nós ativos e um custo atrelado a eles, com tráfego ou não. No entanto, não é preciso se preocupar: patches, atualizações, rede, backups, HA, redundância(...) são gerenciados.



Criação de conta e Free Tier

- Crédito de $300 UDS ou 90 dias.
- Free trial de fato free!
- Para tornar a conta billable, é preciso habilitar manualmente.
  - Ótimo para aprender!
- Ainda precisa de cartão de crédito mas não será efetuada cobrança no Free Tier.
- Contas de email corporativas não são elegíveis para Free Tier.
- Dicas para a conta:
  - Não repetir senha, se possível usar um gerenciador de senhas.
  - Habilitar autenticação multifator (MFA) na conta.
  - Criar alarmes de faturamento.

Fonte: GCP

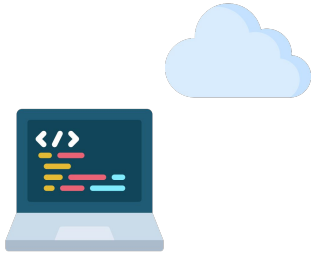
Limitações do Free Tier

- No máximo 8 vCPUs simultâneas.
- Não é permitido usar GPUs e TPUs.
- Não é permitido solicitações para aumento de Quotas.
- Não é permitido mineração de cripto moedas.
- Não possui SLAs.
- Não é permitido usar licenças premium de OS (Exemplo: Windows).

Princípio do privilégio mínimo

- O princípio do privilégio mínimo afirma que um recurso deve ter acesso **apenas** ao(s) **recurso**(s) **exato**(s) de que **precisa** para **funcionar**. Por exemplo, se um serviço estiver executando um backup automatizado de banco de dados, o serviço deve ser restrito a permissões somente leitura exatamente no banco de dados em questão.

Fonte: GCP



Introdução a Google Cloud Functions

Cloud Functions é uma **plataforma** serverless e **event-driven** da Google Cloud Platform para construir e conectar serviços de **cloud**. Ou seja, com as Cloud Functions, é possível escrever um código simples que é executado quando os eventos são enviados da infraestrutura ou serviços de cloud.

Stack

Fonte: GCP

O que exatamente é uma Cloud Function?

- Function que printa "Hello world!" em resposta a uma solicitação GET.

```
def hello_get(request):
    """HTTP Cloud Function.
    Args:
        request (flask.Request): The request object.
        <https://flask.palletsprojects.com/en/1.1.x/api/#incoming-request-data>
    Returns:
        The response text, or any set of values that can be turned into a
        Response object using `make_response`
        <https://flask.palletsprojects.com/en/1.1.x/api/#flask.make_response>.
    Note:
        For more information on how Flask integrates with Cloud
        Functions, see the `Writing HTTP functions` page.
        <https://cloud.google.com/functions/docs/writing/http#http_frameworks>
    """
    return 'Hello World!'
```

Stack

Fonte: GCP

- **Serverless:**
  - Não precisa se preocupar em provisionar e gerenciar infraestrutura.
- **Event-driven:**
  - Funcionamento baseado em eventos.
  - "Coisas" que acontecem na cloud.
  - Exemplos:
    - Arquivo sendo adicionado em um bucket.
    - VM é criada.
    - Uma mensagem cai em um tópico no Pubsub.
- **Conexão:**
  - Aceita eventos ocorridos a partir de outros serviços na cloud.
  - Envia requisições para outros serviços na cloud.

Stack

Fonte: GCP

- **Código** escrito em: Python, Node.js, Go, Java, Ruby, PHP, .NET.
- Possui **autoscaling**.
- Roda em um ambiente dedicado para ela.
- **Stateless**: a invocação da função não deve depender do estado na memória definido por uma invocação anterior. As invocações podem ser tratadas por diferentes instâncias, que não compartilham variáveis globais, memória, sistemas de arquivos ou outro tipo de estado.
- **Concorrência**: Cada instância de uma função lida com apenas uma solicitação simultânea por vez. Isso significa que enquanto seu código está processando uma solicitação, não há possibilidade de uma segunda solicitação ser roteada para a mesma instância.
- **Cold Start:**
    - Uma nova instância de função é iniciada em dois casos:
        - Com um novo deploy.
        - Para lidar com aumento de carga (autoscaling).

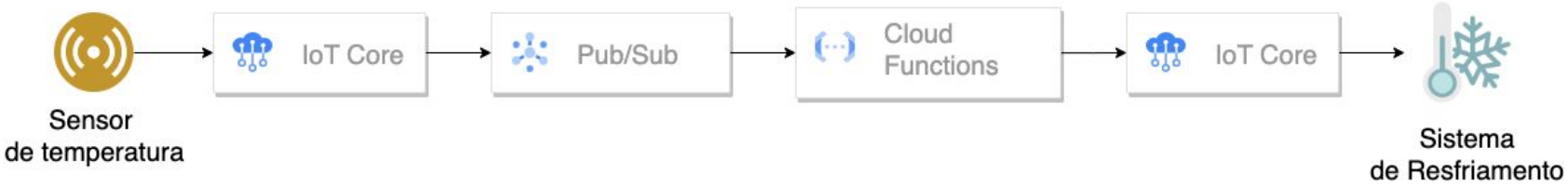
Stack

Fonte: GCP

Exemplos de arquiteturas com Cloud Functions

- **Caso de uso**: aplicação back-end serverless.

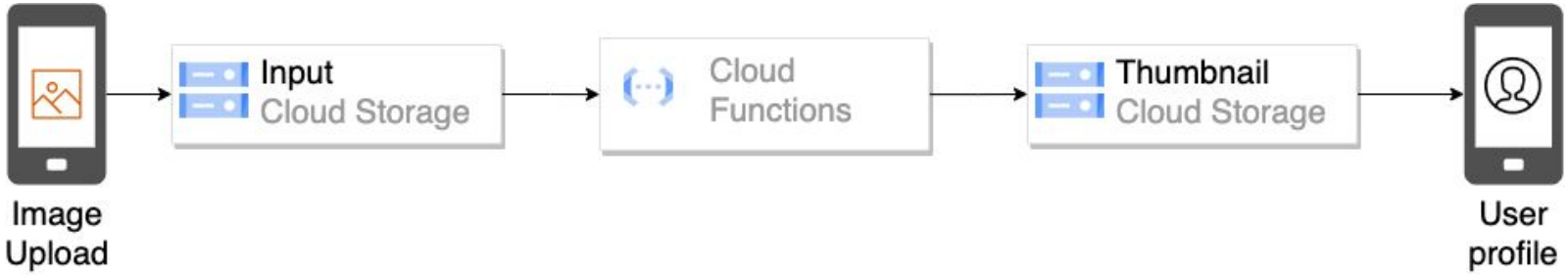
Chame uma Cloud Function de um serviço da Google Cloud - ou diretamente via HTTP de qualquer aplicação web, mobile ou back-end e conecte a outros serviços da Google Cloud e/ou outras APIs de terceiros.



Exemplos de arquiteturas com Cloud Functions

- **Caso de uso**: processamento Real-time.

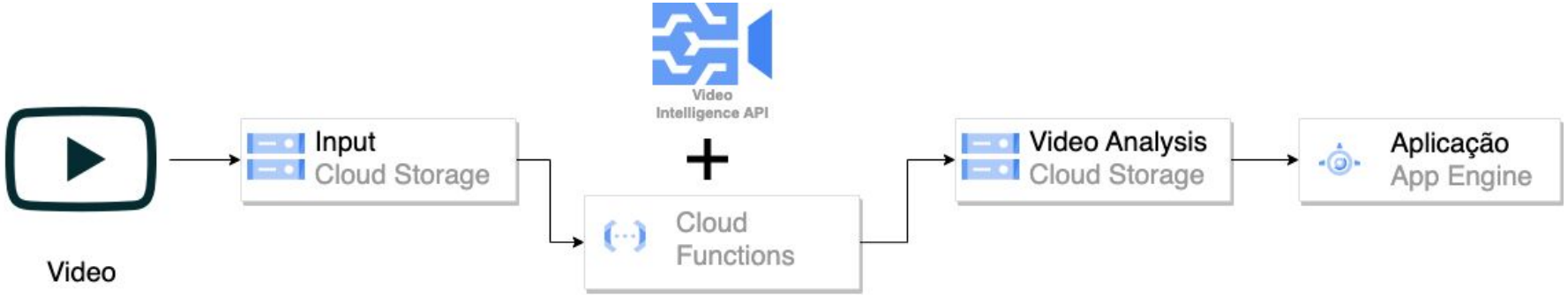
Acionando uma Cloud Function com base em uma alteração nos dados em serviços da Google Cloud, como Cloud Storage ou Cloud Monitoring, para processar dados em tempo real.



Exemplos de arquiteturas com Cloud Functions

- **Caso de uso**: Machine Learning.

Acionando uma Cloud Function com base em uma alteração nos dados em serviços da Google Cloud, como Cloud Storage ou Cloud Monitoring, para processar dados em tempo real.



Tipos de Cloud Function.

- **HTTP**:
  - Lidam com solicitações HTTP e usam triggers HTTP (GET, POST, DELETE, PUT, OPTIONS). Use uma função HTTP quando precisar que a função tenha um endpoint e responda a solicitações HTTP, como para webhooks.
  - Certificado TLS gerado automaticamente para segurança.
  - Formatos suportados: JSON, octet-stream, text, URL encoded.

```
def hello_http(request):
    request_json = request.get_json()
    if request_json and 'name' in request_json:
        name = request_json['name']
    else:
        name = ''
    return 'Hello World from Stack Academy'.format(name)
```

Stack

Fonte: GCP

Tipos de Cloud Function.

- **Event-driven (Background)**:
  - Lidam com eventos da cloud e usam triggers como Cloud Storage, Pubsub, Firebase, Firestore, Cloud Monitoring (via Pubsub). Use esse tipo de function quando ela precisar ser acionada diretamente em resposta a eventos em um projeto da Google Cloud.

```
def hello_gcs(data, context):
    """Background Cloud Function to be triggered by Cloud Storage.
    Args:
       data (dict): The dictionary with data specific to this type of event.
       context (google.cloud.functions.Context): The Cloud Functions event metadata"""
    print("File: {}.".format(data['objectId']))
```

Stack

Fonte: GCP

Triggers

- Um trigger declara o interesse de uma function em um evento específico.
- Triggers são vinculados à functions no momento do deploy.
- Triggers são vinculados à functions seguindo o padrão de muitos para um.
  - O mesmo trigger pode ser associado a muitas functions.
  - Cada function só tem um trigger.
  - Exemplo:

Trigger flags	Trigger description
`--trigger-http`	Trigger the function with an HTTP(S) request. See HTTP triggers for more information.
`--trigger-topic=YOUR_PUBSUB_TOPIC`	Trigger the function when a message is published to the specified Pub/Sub topic. See Pub/Sub triggers for more information.
`--trigger-bucket=YOUR_STORAGE_BUCKET`	Trigger the function when an object is created or overwritten in the specified Cloud Storage bucket. See Cloud Storage triggers for more information.
`--trigger-event-filters=EVENTARC_EVENT_FILTERS`	(2nd gen only) Trigger the function with Eventarc when an event occurs matching the specified filters. Requires the `--gen2` flag to be specified. See Eventarc triggers for more information and additional options.
`--trigger-event=EVENT_TYPE [--trigger-resource=RESOURCE]`	(1st gen only) Trigger the function when the specified event occurs. Specifying a resource is required for some event types. See Triggers supported in Cloud Functions (1st gen) for more information.

Stack

Fonte: GCP

Deploy de Cloud Functions

- Console Google Cloud.
- Terraform.
- CLI.
  - Exemplo:

```
gcloud functions deploy YOUR_FUNCTION_NAME \
[--gen2] \
--region=YOUR_REGION \
--runtime=YOUR_RUNTIME \
--source=YOUR_SOURCE_LOCATION \
--entry-point=YOUR_CODE_ENTRYPOINT \
TRIGGER_FLAGS
```

Stack

Fonte: GCP

Segurança no contexto de Cloud Functions

- O acesso a Cloud Functions é controlado pelo **IAM** (Identity and Access Management) da GCP.
- **Roles** e **Permissões** suportadas: Primitive roles (Owner, Editor e Viewer), Custom Roles e Predefined roles.
- Exemplo de Predefined roles associadas a Cloud Functions:
  - roles/cloudfunctions.**invoker**.
  - roles/cloudfunctions.**developer**.
  - roles/cloudfunctions.**admin**.
  - roles/cloud functions.**viewer**.
- É uma boa prática usar **service accounts** para invocar Cloud Functions. E seguindo o princípio de Least Privilege, fornecer somente as permissões necessárias para as service accounts.

Fonte: GCP

Custo e Quotas

- O custo de uma Cloud Function usa os seguintes parâmetros:
  - Tempo de processamento.
  - Número de vezes que uma função é invocada.
  - A quantidade de recursos de rede usados pela Function.

Custo = Processamento + Invocações + Rede

- Always Free:
  - 2 milhões de invocações por mês.
  - 400.000 GB-segundos, 200.000 GHz-segundos de tempo de processamento por mês.
  - 5 GB de tráfego do tipo egress por mês.
- Conferir a seção de Cloud Function na *Pricing Calculator*.

Stack

Fonte: GCP

Custo e Quotas

- Quota para alguns recursos.
- Documentação sobre quotas.

Quota	Description	Limit (1st gen)	Limit (2nd gen)	Can be increased	Scope
Number of functions	The total number of functions that can be deployed per region	1,000	1,000 minus the number of Cloud Run services deployed	No	per region
Max deployment size	The maximum size of a single function deployment	100MB (compressed) for sources. 500MB (uncompressed) for sources plus modules.	N/A	No	per function
Max uncompressed HTTP request size	Data sent to HTTP Functions in an HTTP request	10MB	32MB	No	per invocation
Max uncompressed HTTP response size	Data sent from HTTP functions in an HTTP response	10MB	10MB for streaming responses. 32MB for non-streaming responses.	No	per invocation
Max event size for event-driven functions	Data sent in events to background functions	10MB	512KB for Eventarc events. 10MB for legacy events.	No	per event
Max function memory	Amount of memory each function instance can use	8GiB	16GiB	No	per function

1° gen vs 2° gen

- A GCP recomenda usar a 2° geração para novas aplicações, embora não tenha planos para descontinuar a 1°.
- A 2° geração é baseada no Cloud Run.
- Novidades da segunda versão:
  - Tempos de processamento mais longo: execução de cargas de trabalho de solicitação mais longa, como processamento de grandes fluxos de dados do Cloud Storage ou do BigQuery.
  - Concorrência aprimorada: lida com várias solicitações simultâneas com uma única instância de função para minimizar cold start e melhorar a latência.
  - Gerenciamento de tráfego: divisão de tráfego entre diferentes revisões de função ou reverta uma função para uma versão anterior.

Stack

Fonte: GCP

1° gen vs 2° gen

Feature	Cloud Functions (1st gen)	Cloud Functions (2nd gen)
Image registry	Container Registry or Artifact Registry	Artifact Registry only
Request timeout	Up to 9 minutes	• Up to 60 minutes for HTTP-triggered functions • Up to 9 minutes for event-triggered functions
Instance size	Up to 8GB RAM with 2 vCPU	Up to 16GiB RAM with 4 vCPU
Concurrency	1 concurrent request per function instance	Up to 1000 concurrent requests per function instance
Traffic splitting	Not supported	Supported
Event types	Direct support for events from 7 sources	Support for any event type supported by Eventarc, including 90+ event sources via Cloud Audit Logs
CloudEvents	Supported only in Ruby, .NET, and PHP runtimes	Supported in all language runtimes

Stack

Fonte: GCP

Em breve na 2° gen

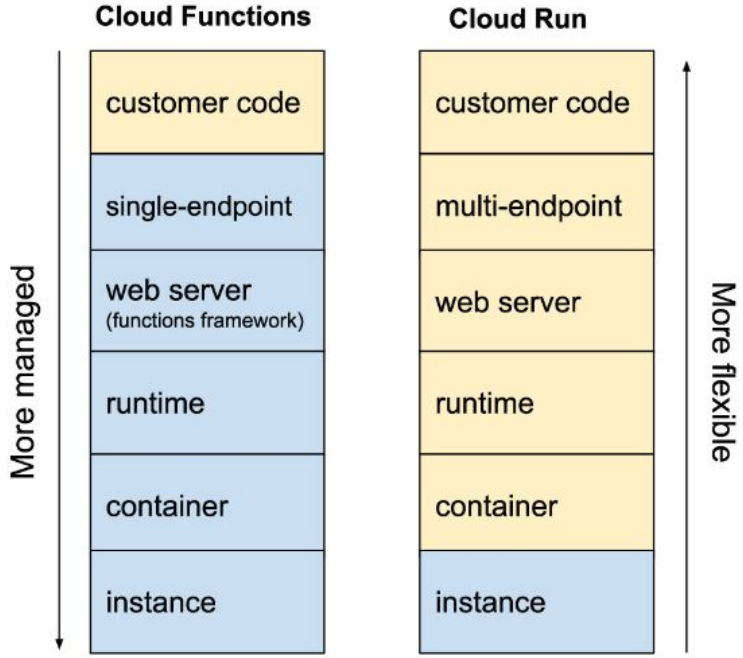
- The ability to deploy functions from Cloud Source Repositories.
- Support for native Cloud Firestore events (row level change triggers) in 2nd gen and Eventarc.
- Support for using capital letters in function names.
- Ability to migrate Cloud Functions (1st gen) functions to (2nd gen) so that customers can take advantage of new features and capabilities.
- `Cloudfunctions.net` URLs.

  Currently, function URLs in Cloud Functions (2nd gen) use a non-deterministic format, meaning you cannot predict your function URL before deployment (though the URL remains stable after deployment). In a future release, 2nd gen function URLs will be updated to be both stable and deterministic.

Stack

Fonte: GCP

Cloud Function vs Cloud Run



Stack

Fonte: GCP

Cloud Function vs Lambda vs Azure Functions

- Mais comparações no seguinte artigo.

Provider	Free Monthly Duration (GB-seconds)	Free Monthly Requests	Cost of Each Additional 1 Million Requests	Cost of Each Additional 1 GB-second	Duration is Rounded to the Nearest
AWS	400,000	1 Million	$0.20	$0.000016	1ms
Azure	400,000	1 Million	$0.20	$0.000016	1ms
GCP	400,000	2 Million	$0.40	$0.0000125	100ms

Stack

Fonte: GCP

Prática 1: Deploy da primeira function (1° geração)

- Requisitos:
  - Conta na Google Cloud.
  - Definição de um projeto.
  - Billing precisa estar habilitado para o projeto.
  - Cloud Function API habilitada no projeto.

- Objetivo:
  - Fazer deploy de uma simples Function do tipo HTTP e realizar o teste no console.

Prática 2: Deploy de duas functions (2° geração)

- Function 1:
  - Fazer deploy de uma Cloud Function que é acionada sempre que um arquivo é carregado em um bucket
  - Analisar logs no console.

- Function 2:
  - Fazer deploy de uma Function com uma página HTML simples.
  - Mudar algumas variáveis da Function e realizar deploy de diferentes versões.
  - Dividir o tráfego recebido entre as diferentes versões.
  - Analisar no console o comportamento.

Prática 2: Deploy de duas functions (2° geração)

- Function 1:
  - Fazer deploy de uma Cloud Function que é acionada sempre que um arquivo é carregado em um bucket
  - Analisar logs no console.

- Possível mensagem de erro. Correção explicada na aula:

Dominando Cloud Functions

Aula 2

Stack

Segurança no contexto de Cloud Functions

- O acesso a Cloud Functions é controlado pelo **IAM** (Identity and Access Management) da GCP.
- **Roles** e **Permissões** suportadas: Primitive roles (Owner, Editor e Viewer), Custom Roles e Predefined roles.
- Exemplo de Predefined roles associadas a Cloud Functions:
  - roles/cloudfunctions.**invoker**.
  - roles/cloudfunctions.**developer**.
  - roles/cloudfunctions.**admin**.
  - roles/cloud functions.**viewer**.
- É uma boa prática usar **service accounts** para invocar Cloud Functions. E seguindo o princípio de Least Privilege, fornecer somente as permissões necessárias para as service accounts.

Stack

Fonte: GCP

Terraform 101

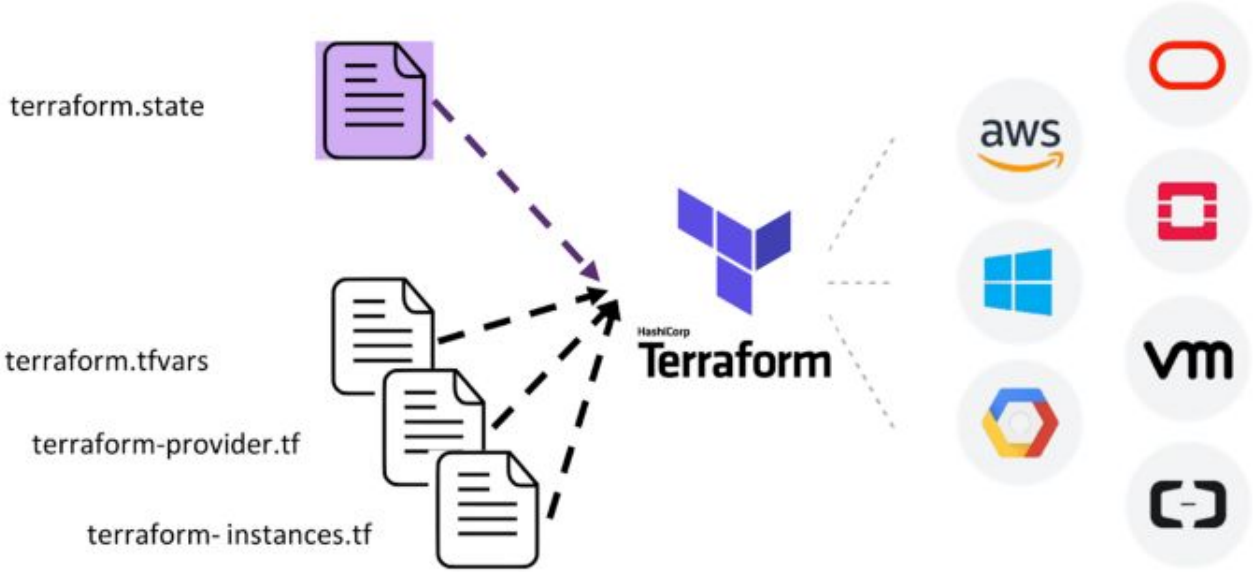
> *“Terraform é uma ferramenta de infraestrutura como código (IaC) que permite criar e alterar versões de infraestrutura com segurança e eficiência. Isso inclui componentes de baixo nível, como instâncias de computação, armazenamento e rede, bem como componentes de alto nível, como entradas de DNS, recursos de SaaS, et.”*

- Faz orquestração, não apenas gerenciamento de configuração.
- Suporta vários provedores, como AWS, Azure, Oracle, GCP entre outras.
- Fornece infraestrutura imutável onde a configuração muda suavemente.
- Usa linguagem de fácil compreensão, HCL (linguagem de configuração HashiCorp).

Terraform - ciclo de vida

- **terraform init**: inicializa o ambiente Terraform (local). Normalmente é executado apenas uma vez por sessão.
- **terraform plan**: compara o estado do Terraform com o estado como está na cloud, constrói e exibe um plano de execução.
- **terraform apply**: executa o plano.
- **terraform destroy**: exclui todos os recursos controlados por este ambiente específico do terraform.

Terraform - tipos de arquivos



Fonte da imagem: K21Academy

Terraform - exemplo

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "3.5.0"
    }
  }
}

provider "google" {
  credentials = file("<NAME>.json")

  project = "<PROJECT_ID>"
  region  = "us-central1"
  zone    = "us-central1-c"
}

resource "google_compute_network" "vpc_network" {
  name = "terraform-network"
}
```

Stack

Prática 3: Deploy Function via Terraform

- Requisitos:
  - Terraform instalado.
    - Link para download.
  - Google SDK instalada:
    - Link para download.

- Objetivo:
  - Fazer deploy de uma Cloud Function que é acionada sempre que um arquivo é carregado em um bucket usando a ferramenta Terraform.

Testando cloud functions localmente: Functions-Framework

- **Não** é uma **boa prática** testar uma Cloud Function **direto** no ambiente de **Cloud**, pois cada teste pode demorar até 5 minutos.
- O **Deploy** da Function deve ser a **última etapa** do **desenvolvimento**.
- O Google disponibiliza um framework para **testar localmente** Cloud Function, Functions-Framework.
- A diferença para o deploy na Cloud é que ele simula o servidor de Cloud Functions e responde a um evento.
- Quando o comando functions-framework é executado, ele simula o servidor e fornece um endereço local para chamar a função.

Stack

Fonte: GCP

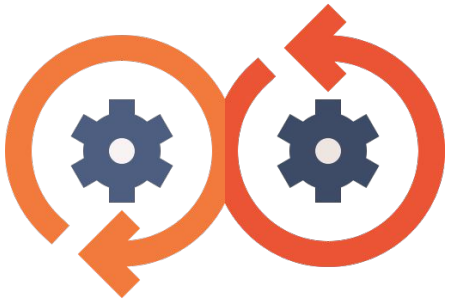
Prática 4: Functions-framework

- Requisitos:
  - venv instalado.
- Objetivo:
  - Realizar teste de forma local com uma Cloud Function do tipo HTTP.

Conceitos de CI/CD

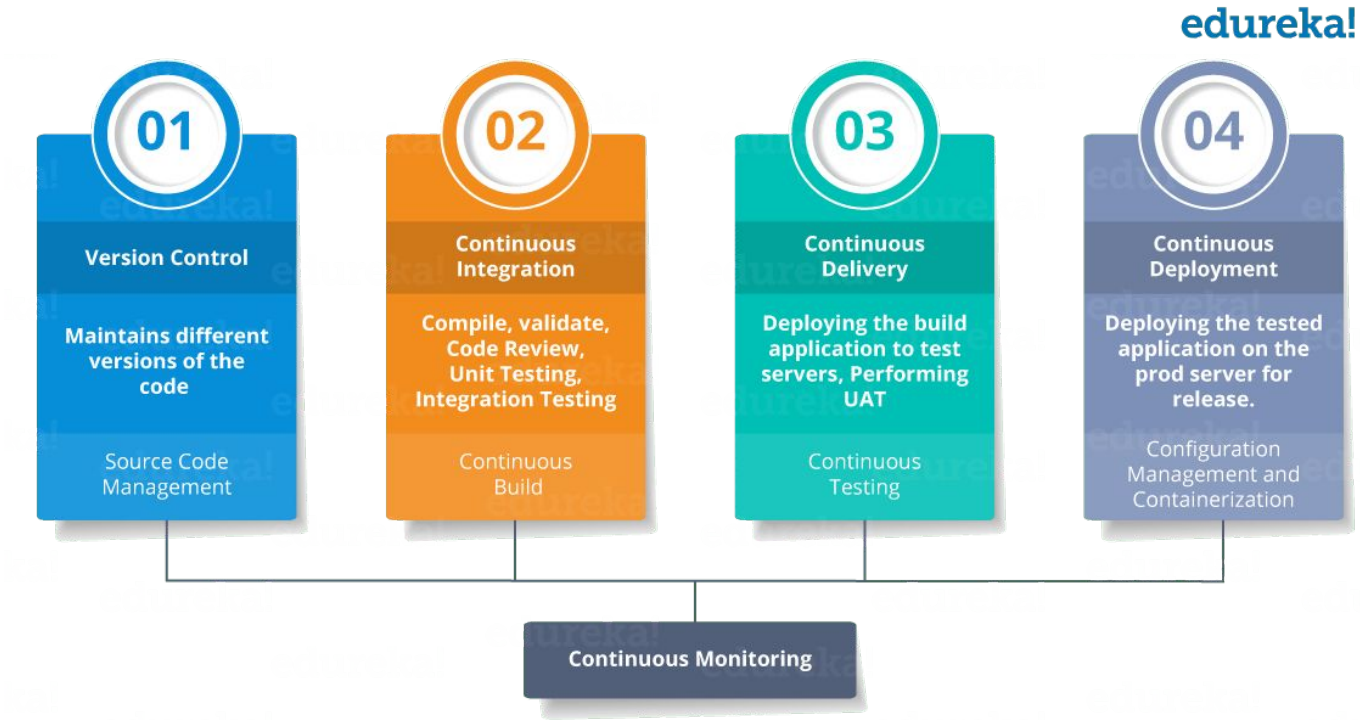
- A **integração contínua** é uma boa prática de **desenvolvimento de software** e **DevOps** em que os desenvolvedores, com frequência, juntam suas **alterações** de código em um repositório central.

- Os principais objetivos da integração contínua são **encontrar** e **investigar** bugs mais rapidamente, melhorar a qualidade do software e reduzir o tempo que leva para validar e lançar novas atualizações de software.



Stack

Conceitos de CI/CD

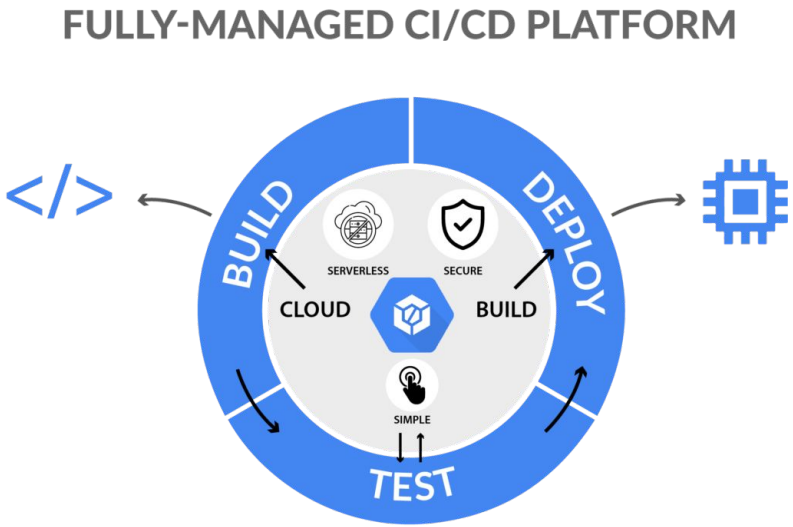


Stack

Fonte da imagem: DZone

Cloud Build

- Plataforma de **CI/CD Serverless**.
- O Cloud Build é um serviço que executa **Builds** na **infraestrutura** do Google Cloud Platform.
- O Cloud Build pode importar o **código** do Cloud Storage, **Cloud Source Repositories**, **GitHub** ou **Bitbucket**



Stack

Fonte: k21Academy

Cloud Build

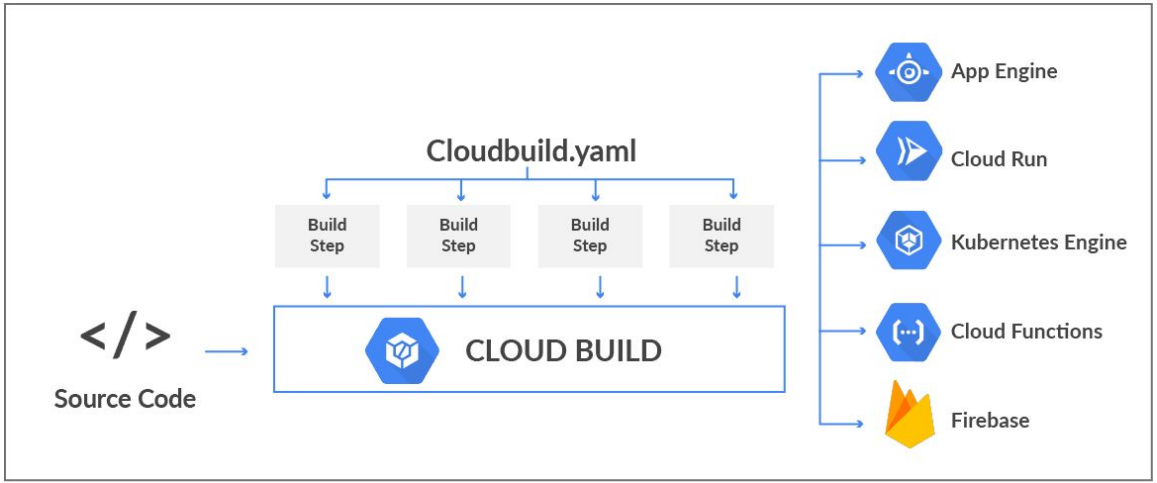
- O Cloud Build é uma **plataforma serverless** que ajuda a criar workflows de desenvolvimento personalizados para build, testes e deploy.
- Compatível com **Docker**: o Cloud Build permite que importe arquivos Docker existentes e envie imagens diretamente para os repositórios de armazenamento de imagens do Docker, como Docker Hub e Container Registry.
- **Automatiza deploys** no Google Kubernetes Engine (GKE) ou Cloud Run para entrega contínua.
- Executa automaticamente a **varredura** de **vulnerabilidade** do pacote para imagens vulneráveis com base nas políticas definidas pelo DevSecOps.
- É possível **empacotar código** em **contêineres** ou **artefatos** não contêineres, como Maven, Gradle, Go ou Bazel.

Stack

Cloud Build

Passos envolvidos:

- Desenvolvimento do **código**.
- Criação de um **arquivo de configuração** de build no formato **YAML** ou JSON, que contém **instruções** para o Cloud Build.
- **Envio** do **build** para o Cloud Build.
- O Cloud Build **executa** o build com base na configuração de **compilação** que recebeu.

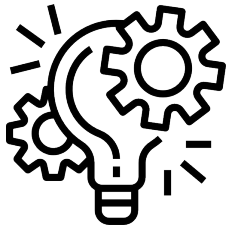


Stack

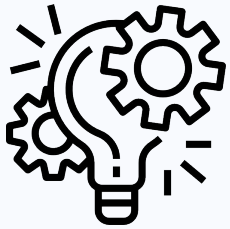
Fonte: k21Academy

Prática 5: Deploy de Cloud Function com Cloud Build

- Objetivo:
  - Realizar o deploy de uma Cloud Function através do Cloud Build.



Projetos com Google Cloud Functions



Projeto 1:

Cloud Functions + PubSub

→ mensagem via SMS.

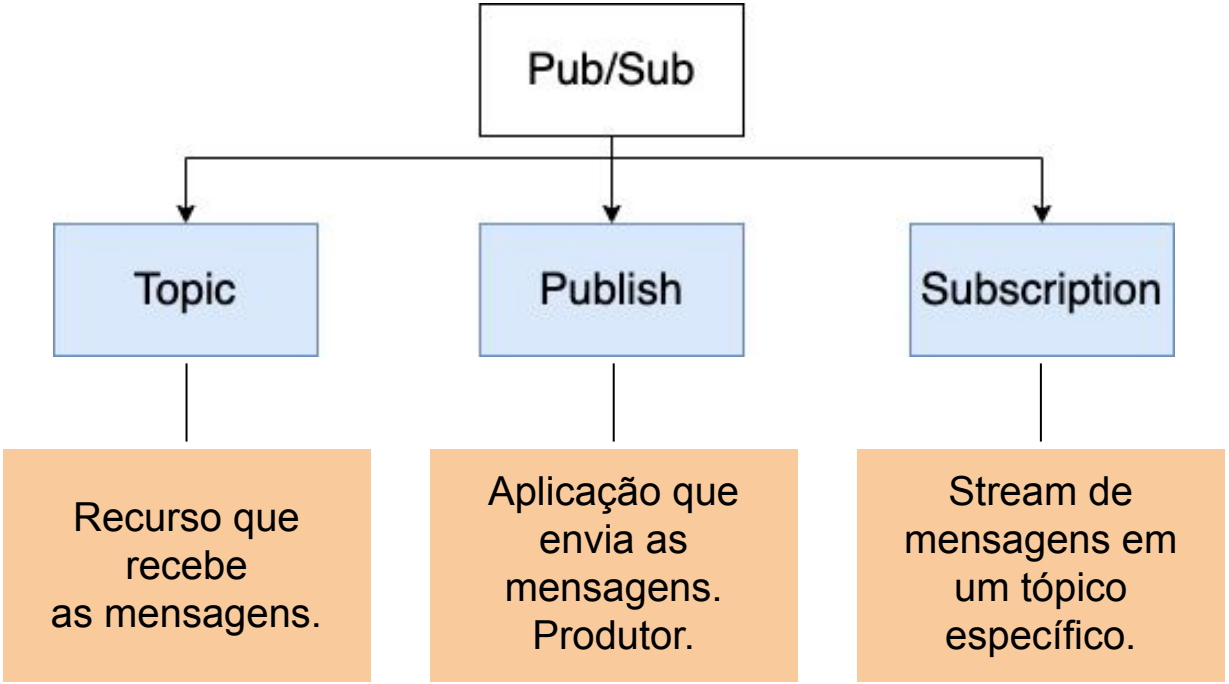
Stack

PubSub

- O Pub/Sub é um **serviço de mensagens assíncrono** e escalável que separa os serviços que produzem mensagens dos serviços que processam essas mensagens.
- O Pub/Sub é usado para **streaming** e **pipelines de integração de dados** com objetivo de **ingerir** e **distribuir** dados. É igualmente eficaz como um middleware orientado a mensagens para integração de serviços ou como uma fila para paralelizar tarefas.

Stack

Fonte: GCP



Stack

Fonte: GCP

Secret Manager

- O Secret Manager é um sistema de armazenamento seguro para API Keys, senhas, certificados e outros dados sensíveis. O Secret Manager fornece um local central e uma única fonte de verdade para gerenciar, acessar e auditar segredos no Google Cloud.

Fonte: GCP

Arquitetura do projeto 1

- Iremos enviar uma mensagem para um tópico no PubSub.
- Criaremos uma Cloud Function com tipo de evento setado para PubSub.
- Quando a CF receber a notificação através do tópico, irá enviar uma requisição para uma API terceira chamada Twilio, que irá enviar a mensagem enviada pelo tópico através de um SMS.

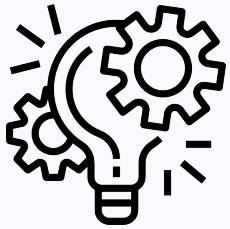
- Observação: Não esquecer de verificar o número que deseja enviar o SMS no console da Twilio no seguinte link.

Stack

Dominando Cloud Functions

Aula 3

Stack



Projeto 2:
Cloud Functions→
mensagem de email.

Stack

Arquitetura do projeto 2

- Iremos enviar uma mensagem através do Postman.
- Criaremos uma Cloud Function com tipo de evento setado para http.
- Quando a CF receber a requisição irá enviar uma outra requisição para a API chamada SendGrid que irá enviar a mensagem de email.

- Observação: Não esquecer de verificar o endereço de email que deseja enviar a mensagem seguindo a seguinte documentação. Link para o Single Sender Verification.

Stack

Variáveis de ambiente no Windows

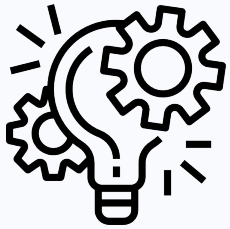
- No windows o tratamento com variáveis de ambiente é diferente do Mac e Linux.
- Sugiro a seguinte lib para lidar com essa questão: python-dotenv
- Após a instalação via pip, crie um arquivo **settings.py** com o seguinte:

```python
# settings.py
from dotenv import load_dotenv
load_dotenv()

# Ou,
load_dotenv(verbose=True)

#Ou, explicitando o caminho para o env
from pathlib import Path  # python3 only
env_path = Path('.') / '.env'
load_dotenv(dotenv_path=env_path)
```

Diretório raiz com os arquivos **settings.py** e **.env**.

- Então, se quiser executar o source no arquivo .env, primeiro execute o settings.py. Depois disso as variáveis de ambiente serão carregadas.

Stack



Projeto 3:
Cloud Functions + Cloud Scheduler
→ Cloud Storage

Stack

Cloud Scheduler

- O Cloud Scheduler é um **agendador de tarefas** totalmente gerenciado. Ele permite agendar praticamente diversos tipos de **workloads**.
- "**Cronjob** as a Service".
- O Cloud Scheduler pode ser usado para vários situações, como fazer requests para um endpoint HTTP/S, invocar um tópico PubSub, fazer atualizações de banco de dados e notificações push, acionar pipelines de CI/CD, agendar tarefas como uploads de imagens e enviar um e-mail , ou até mesmo invocando Cloud Functions.

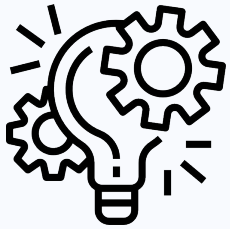
Stack

Fonte: GCP

Arquitetura do projeto 3

- Neste projeto iremos criar uma Cloud Function que a cada minuto é triggada pelo Cloud Scheduler para escrever um arquivo txt em um bucket no Cloud Storage.

Stack



Projeto 4:

Cloud Functions + Vision API

→ Detecção de Faces.

Stack

Vision API

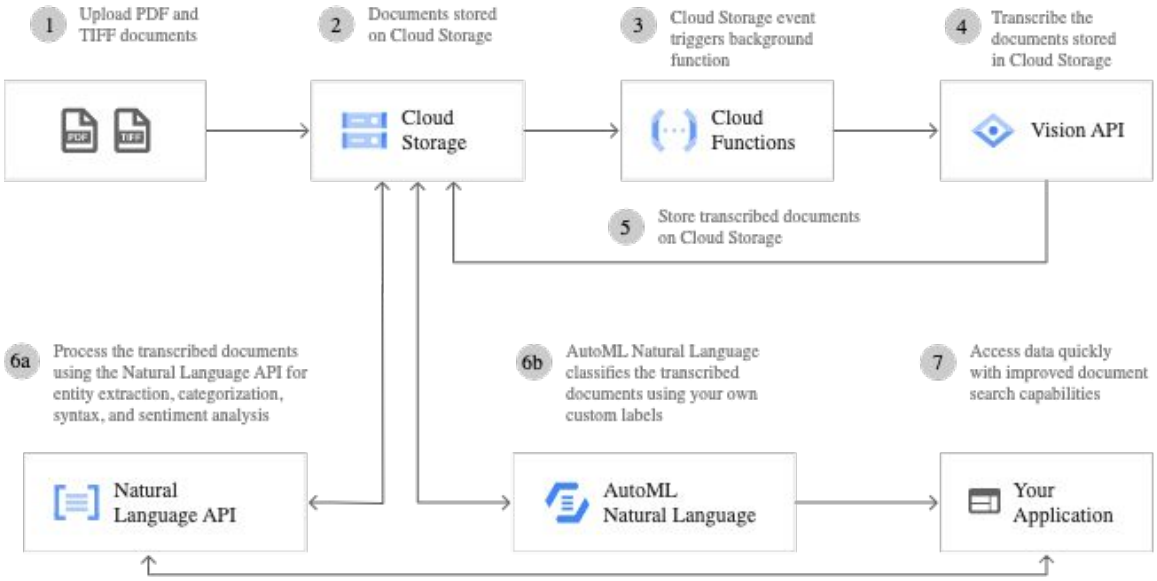
- Cloud Vision API permite que os desenvolvedores **integrem** facilmente os recursos de **detecção de visão** em **aplicações**, incluindo **rotulagem de imagens**, **detecção de rostos** e pontos de **referência**, reconhecimento óptico de caracteres (**OCR**) e marcação de conteúdo explícito.
- Lista completa de features.

Stack

Fonte: GCP

Vision API

- Extração de texto em documentos.



Fonte: GCP

Vertex AI

Modelos Pré-Treinados

Vision

- Vision
- AutoML Vision
- Video Intelligence
- AutoML Video Intelligence

Language

- Translation
- AutoML Translation
- Natural Language
- AutoML Natural Language

Conversation

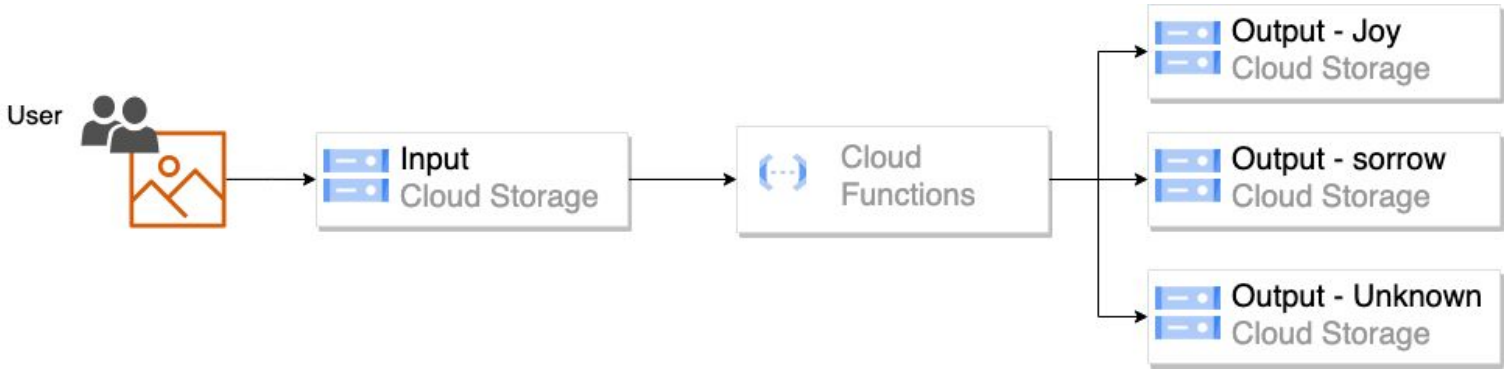
- Dialogflow
- Speech-to-Text
- Text-to-Speech
- Speaker ID

Stack

Fonte: GCP

Arquitetura do projeto 4

- Iremos fazer upload de uma foto de uma pessoa para um bucket de input.
- Criaremos uma Cloud Function com tipo de evento setado para criação de arquivo no Cloud Storage.
- Quando a CF receber o evento irá enviar uma requisição para Vision API para determinar se a pessoa está feliz, triste ou neutra e enviar metadados da foto para o respectivo bucket correspondente a emoção da pessoa.

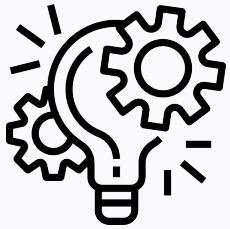


Stack

Dominando Cloud Functions

Aula 5

Stack



Projeto 5:
Processamento de csv com Cloud Functions.

Stack

BigQuery

- O BigQuery é um **data warehouse** serverless que ajuda a **gerenciar** e **analisar dados** com recursos integrados, como **machine learning**, **análise geoespacial** e **business intelligence**. A arquitetura serverless do BigQuery permite usar consultas **SQL** para responder às maiores perguntas de negócio de uma organização **sem gerenciamento** de **infraestrutura** zero. O mecanismo de análise distribuída e **escalável** do BigQuery permite **consultar terabytes** em segundos e petabytes em minutos.

Stack

Fonte: GCP

BigQuery

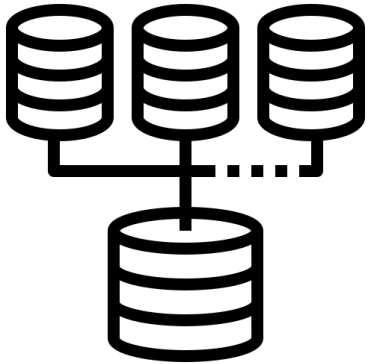
- Serverless.
- Escala de petabytes.
- Usa SQL mas não é uma base relacional.
- Base analítica.
- Outras features:
  - BQ Machine learning.
  - BQ BI Engine.
  - BQ GIS (Geospatial Analytics).

Stack

Fonte: GCP

BigQuery

- Datasets:
  - Coleção de tabelas e views.
- Tabelas.
  - Suporta estruturas escalares e aninhadas.
  - Armazenamento colunar.
  - Particionadas.
- Forma de realizar queries:
  - SQL UI.
  - Comando bq.
  - APIs.

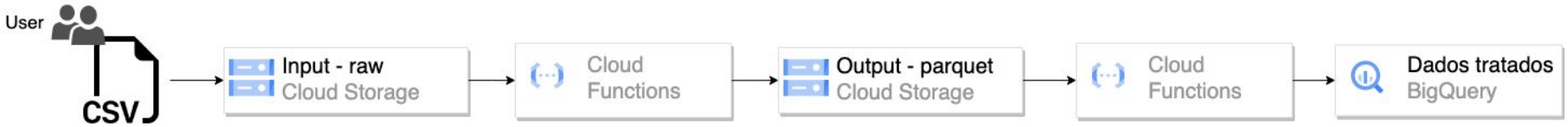


Stack

Fonte: GCP

Arquitetura do projeto 5

- No projeto 5 iremos desenvolver um pipeline de dados usando Cloud Functions.
- Inicialmente um usuário irá fazer upload de um arquivo csv para um bucket de dados brutos (raw) e uma primeira function irá tratar esses dados e salvá-los em formato parquet para armazenamento histórico em um segundo bucket.
- Uma segunda function irá ser acionada a cada vez que um arquivo parquet cair no bucket de output e irá enviar esses dados para o BigQuery.
- Conjunto de dados de exemplo: link.



Stack

Obrigado!

Stack