FIAP ON

7ASOO - Arquitetura de Soluções

Grupo 9:

- Luciano Cristino da Silva do Carmo
- Rodrigo de Lima Silva
- Wesley Egberto de Brito
- Marcelo Gomes de Melo
- Hillary Kichize Borborema
- Cristian Medina

Hackathon

São Paulo, Julho de 2022

Introdução

SpotMusic é uma startup que acabou de receber um Startup Funding (que é o dinheiro necessário para lançar um novo negócio) com o objetivo de garantir que a empresa tenha os recursos necessários para evoluir aspectos importantes da tecnologia, visando segurança, escalabilidade e disponibilidade, pois em paralelo a essa ação na área de Tecnologia, também há investimento na área de marketing para criar a campanha de lançamento do produto.

A empresa basicamente nasceu da ideia de dois alunos de graduação que enxergaram um potencial produto em seu trabalho de conclusão de curso, e resolveram mergulhar de cabeça nela.

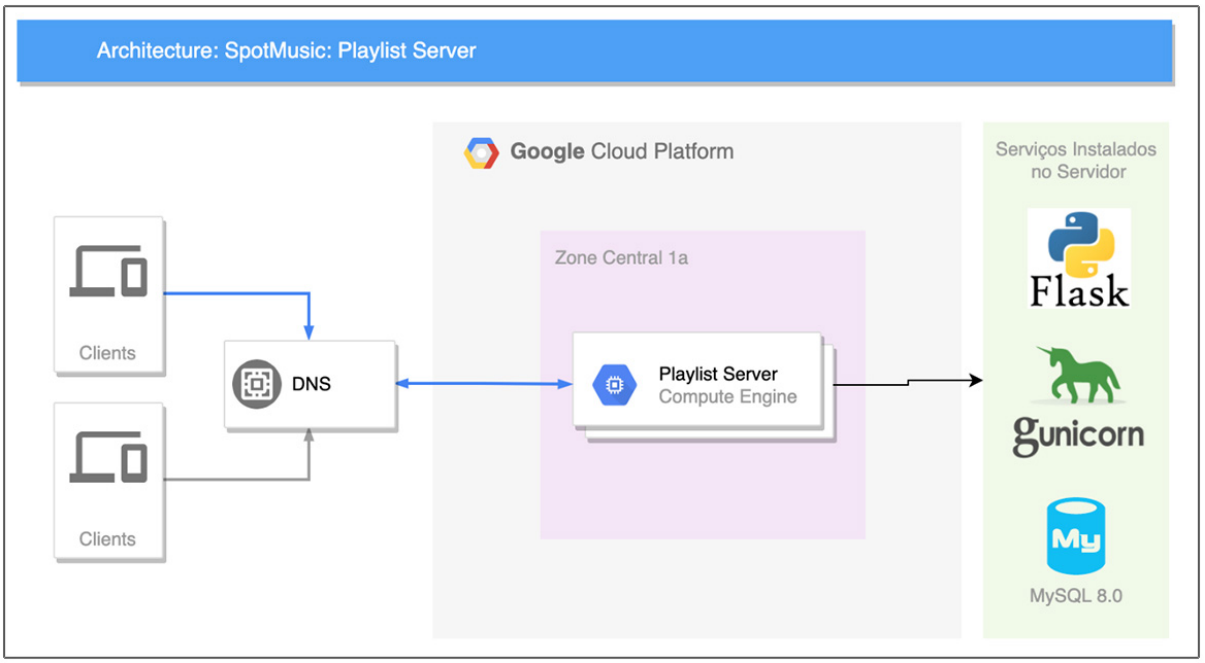
Sem muitos conhecimentos sobre arquitetura moderna, mas usando a facilidade da Cloud, eles criaram e lançaram o produto.

O objetivo deste documento é detalhar a proposta para a nova arquitetura que irá implementar o controle de versão, a segregação das camadas (frontend, backend e banco de dados), a automatização da infraestrutura (infrastructure as a code) e a automatização do fluxo de desenvolvimento).

AS IS

A aplicação **SpotMusic** foi desenvolvida em Python Flask utilizando banco de dados MySQL para armazenamento de dados, e está rodando 100% em nuvem pública do Google Cloud Platform utilizando o serviço de Compute Engine (Máquina Virtual).

Segue abaixo o desenho da arquitetura atual:



Infraestrutura

Todos os serviços rodam no mesmo servidor que foi configurado manualmente, sendo assim, não utilizando as boas práticas de DevSecOps.

- Compute Engine

Componentes da solução

- Python Flask
- Gunicorn
- MySQL 8.0

Escalabilidade

A escalabilidade é feita trocando o tipo de máquina, entregando mais memória e CPU.

Segurança

O único controle de qualidade utilizado é o Firewall nativo do Google, liberando apenas o HTTP e SSH para o servidor.

Ciclo de desenvolvimento

O deploy não é automatizado, sendo necessário substituir os arquivos (de forma manual) na pasta do servidor, e após,realizar o restart da aplicação no ambiente produtivo.
Controle de versões:

Versionamento

Não há controle de versão do código.

Monitoramento

Não há monitoramento da infraestrutura.

TO BE

POC

O objetivo da POC é implementar as melhores práticas de DevOps, automatizando e gerenciando todo o fluxo.

Essa nova arquitetura permitirá que a **SpotMusic** melhore o tempo de entrega de novas soluções em produção, diminuindo incidentes devido a configurações manuais (permitindo a uniformidade de novos deploys), e rastreando as alterações.

- **Controle de versionamento de código:** centralizar o código da infraestrutura e das aplicações no Github.
- **Segregar a solução em camadas:**
    - Frontend: aplicação frontend rodando isoladamente em container no Cloud Run;
    - Backend: aplicação backend Python rodando isoladamente em container no Cloud Run;
    - Banco de dados: instância do MySQL rodando separadamente dos seus clientes.
- **Infrastructure as a Code:** automatização da criação e atualização da infraestrutura.
- **Pipelines de CD:** automatização do fluxo de desenvolvimento e entrega das aplicações nos seus ambientes.

A nova arquitetura será feita utilizando o provedor Google Cloud Platform (GCP), mantendo a conta que já existe e minimizando o impacto no time.

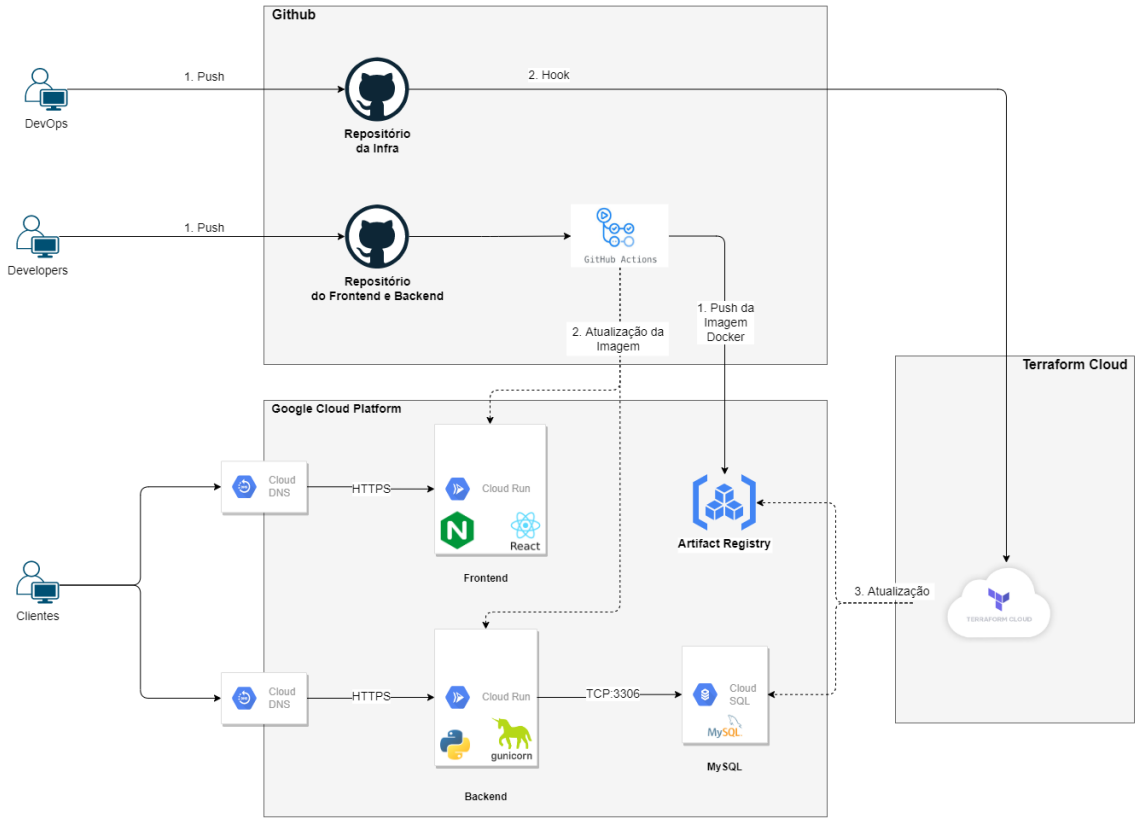
SLA de uptime e downtime

Provisionando a infraestrutura em Cloud Run será possível ter uma disponibilidade de no mínimo 99.95%, garantindo disponibilidade da aplicação.

Segurança da informação

Disponibilizando a infraestrutura na Google Cloud Plataform será garantido que a aplicação estará utilizando as melhores práticas de segurança, minimizando os riscos de invasão.

Abaixo segue o diagrama da solução que será entregue:



Componentes da solução:

- Github:
    - Será entregue uma conta no Github para centralizar todo o controle de versão, tanto da infraestrutura como das aplicações;
    - Teremos três repositórios:
        - **infra-SpotMusic**: repositório com o código-fonte do Terraform para aplicação da IaC;
        - **frontend-SpotMusic:** repositório com o código-fonte e o pipeline da aplicação frontend;
        - **backend-SpotMusic:** repositório com o código-fonte e o pipeline da API backend acessada pelo frontend.
    - Teremos duas pipelines no Github Action:
        - **Pipeline Frontend:** pipeline declarado no repositório **frontend-SpotMusic,** será responsável por fazer a construção da imagem Docker, *push* para o Artifactory Registry e atualização da imagem no aplicativo no Cloud Run;
        - **Pipeline Backend:** pipeline declaro no repositório **backend-SpotMusic,** será responsável pela a construção da imagem

Docker, *push* para o Artifactory Registry e atualização da imagem no aplicativo no Cloud Run

- Terraform Cloud:
    - Será entregue uma conta no Terraform Cloud para automatizar a criação e atualização da infraestrutura utilizada pelo produto;
    - Será criado uma workspace de projeto e vinculado ao repositório **infra-SpotMusic**, o Terraform fará a execução do *plan* e *apply* a cada push executado na branch *main* do repositório.
    - Componentes gerenciados pelo Terraform:
        - Artifactory Registry
        - Instância no Cloud SQL
- Google Cloud Platform:
    - A solução utilizará a conta atual do GCP da empresa;
    - Utilizaremos:
        - **Artifact Registry:**
            - componente responsável por armazenar as imagens Docker construídas para as aplicações de frontend e backend;
        - **Cloud SQL**:
            - serviço gerenciado de banco de dados MySQL que será utilizado para armazenar os dados da aplicação;
            - esse serviço provê escalabilidade automática, replicação de dados, backup, atualizações de segurança e alta disponibilidade;
            - o escalonamento automático é configurável, nos permitindo desabilitar caso haja necessário conter custos;
            - o SLA do Cloud SQL é de 99.95%;
            - a arquitetura de dados e os dados atuais serão migrados para a nova instância para que não haja perda de dados.
        - **Cloud Run**
            - serviço serverless para execução de aplicações em containers;
            - esse serviço provê gerenciamento do ciclo de vida automático, escalabilidade automática, modelo de cobrança pay-per-use, controle de tráfego e monitoramento integrado;
            - Ao utilizar um serviço serverless teremos a vantagem de apenas gerar custos de infraestrutura quando houver tráfego na aplicação, permitindo economizar nos momentos de ociosidade.;
            - o SLA do Cloud RUN é de 99.95%;
            - Aplicações:
                - Frontend:
                    - Aplicação React publicada numa aplicação Nginx.
                - Backend:
                    - Aplicação Python com Flask rodando no Gunicorn.

Custos de Infraestrutura POC

Cloud Run (backend)

Region: Iowa
CPU Allocation Type: CPU is only allocated during request processing
CPU: 1
Memory: 0.25 GiB
CPU Allocation Time: 5 vCPU-second
Memory Allocation Time: 1.25 GiB-second
Requests: 100,000 requests
minimum number of instances: 1
Valor: USD 8.21

Cloud Run (frontend)

Region: Iowa
CPU Allocation Type: CPU is only allocated during request processing
CPU: 1
Memory: 0.25 GiB
CPU Allocation Time: 5 vCPU-second
Memory Allocation Time: 1.25 GiB-second
Requests: 100,000 requests
minimum number of instances: 1
Valor: USD 8.21

Artifact Registry

Total Amount of storage: 10 GiB per month

Valor: USD 0.95

Cloud SQL

# of instances: 2
Instance type: db-lightweight-1
Commitment term: 1 Year
Location: Iowa
730.0 total hours per month
SSD Storage: 1,000.0 GiB
Backup: 100.0 GiB
Valor: USD 429.97

Total Estimated Cost: USD 447.34 per 1 month

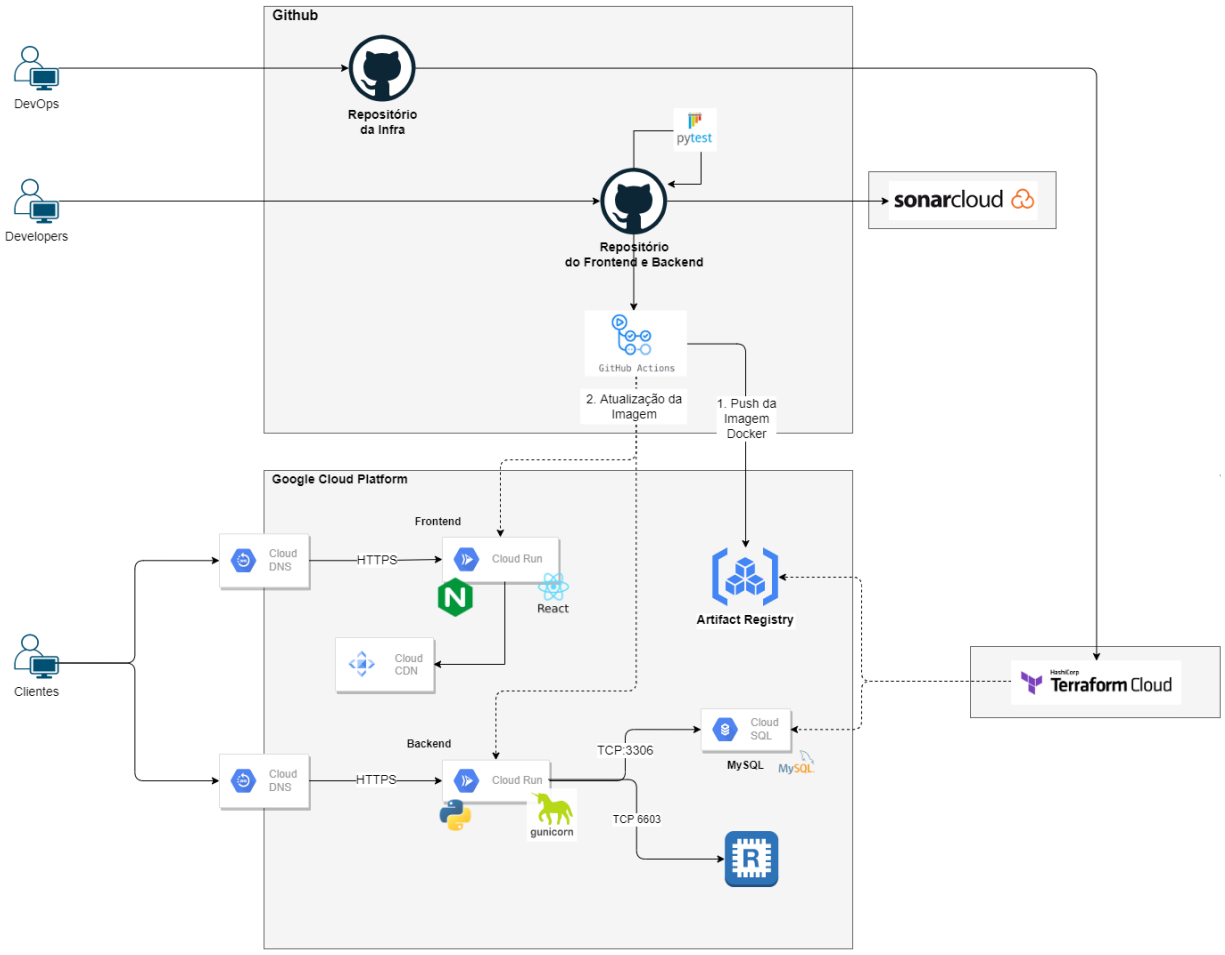
Solução Final

O objetivo da Solução Final é aumentar as práticas iniciadas na fase de POC, melhorando o desempenho, segurança e diminuindo os custos de infraestrutura através de cache.

Objetivos:

- automatizar a execução de testes unitários no CI para garantir qualidade nas entregas;
- automatizar a execução de escaneamento de vulnerabilidades no código e na imagem Docker para ajudar a mitigar os riscos de segurança;
- segregação e automatização das entregas nos ambientes de desenvolvimento, homologação e produção.

Abaixo segue o diagrama da solução que será entregue:



Componentes da solução que foram adicionados além da POC:

- Segurança:
    - Conta no SonarCloud para vincular os repositórios dos projetos e habilitar o escaneamento de código.
- Cache:
    - **Cloud CDN:**
        - serviço de caching de conteúdo estático;
        - utilizaremos para aplicar cache nos conteúdos estáticos da aplicação React e diminuir as requisições que chegam no Cloud Run, diminuindo o custo de execução.
    - **Memorystore Redis:**
        - serviço de cache em memória;
        - utilizaremos para cachear os dados dos usuários para diminuir o tempo de resposta da aplicação e o custo de I/O com o banco de dados.
- Teste unitário:
    - Implementação da ferramenta **PyTest** para realizar os testes unitários, garantindo que quebra de código não impactará o ambiente produtivo.
- Cobertura de Testes
    - Configurando as soluções do Sonar Cloud será possível implementar a cobertura de código.
    - Será habilitado o Quality Gate para impedir que códigos mal construídos subam em ambiente produtivo.

Outras Soluções

Além da solução proposta, existem outras formas de implementar o produto, abaixo seguem algumas avaliadas:

- Automatização da entrega nas aplicações nas VMs e automatização do escalonamento automático das VMs através de determinadas métricas;
- Implantação do frontend e do backend utilizando o serviço de Kubernetes do GCP;
- Implantação do frontend e do backend utilizando Cloud Functions do GCP, o que exige mudança no design da aplicação devido às particularidades desse modelo.

Links

Frontend: https://spotmusic-frontend-opsorxshma-uc.a.run.app/
Backend: https://spotmusic-backend-opsorxshma-uc.a.run.app/
Github: https://github.com/Hackathon-7asoo