Introdução

Com o surgimento e crescimento de plataformas que oferecem serviços confiáveis de computação em nuvem tornou-se possível direcionar o foco de um time de desenvolvimento diretamente na implementação de uma arquitetura de sistema, diminuindo o tempo de atuação em soluções de monitoria da saúde da aplicação, consumo de recursos do servidor e implantação de servidores físicos.

Por isso, para o desenvolvimento da arquitetura do problema proposto, foi utilizada a AWS (*Amazon Web Services*), que oferece todos serviços necessários para a construção de uma arquitetura que satisfaça os requisitos apresentados no desafio. Os serviços utilizados foram: Amazon Elastic Beanstalk, Amazon Elasticache (Redis), Amazon API Gateway, Amazon VPC.

Amazon Elastic Beanstalk

O Amazon Elastic Beanstalk é um serviço PAAS (*Platform-As-A-Service*) utilizado para fazer implantações de aplicações web e serviços de maneira escalável.

Esse serviço foi escolhido para os sistemas 1, 2 e 3, com base nas seguintes características:

- Deploy: simplicidade para fazer o deploy das aplicações na AWS;
- Segurança: permite a configuração de VPC (assunto abordado na sessão Amazon Virtual Private Cloud);
- Escalabilidade: fornece serviço de escalabilidade da aplicação, possibilitando configurar qual o critério para que uma aplicação seja escalada. Por exemplo: quando uma instância atingir 70% do uso da CPU uma nova instância da aplicação será criada. Também é possível configurar o número mínimo e máximo de instâncias da aplicação;
- Monitoramento da saúde da aplicação: possibilidade de configuração do intervalo de tempo que o Beanstalk pode fazer uma checagem de saúde da aplicação, atribuindo um status da saúde da aplicação no final da checagem;
- Log de eventos da aplicação: por padrão, o Beanstalk fornece todos os eventos relacionados à instância da aplicação;
- Flexibilidade: liberdade para selecionar qual a classe de máquina EC2 que melhor se adequa à aplicação. Além disso, é possível criar aplicações utilizando o Tomcat, Docker e diversas linguagens de programação.

Amazon Elasticache

Elasticache é um serviço da AWS que permite hospedar, gerenciar e autoescalar servidores Redis e MemCached. Por uma questão de conhecimento prévio, foi escolhido o Redis como ferramenta de armazenamento de chave-valor em memória.

Também é possível utilizar o Redis hospedando em uma EC2, porém o Elasticache foi escolhido devido aos seguintes fatores:

- Escalabilidade: permite fácil configuração da escalabilidade de acordo com a demanda. Além disso, o escalonamento tem suporte a *sharding*, que é um método de particionamento dos dados para outras *N* outras partições. Assim, é possível escalar a aplicação Redis sem perder ou duplicar nenhum dado;
- Manutenção: a AWS é responsável por atualizar de pacotes e versões do Redis.
- Gerenciamento: o gerenciamento de tarefas, monitoramento e backups é fornecido pela AWS, portanto, não existe a necessidade de investir tempo para o implementação de ferramentas de gerenciamento e monitoria do Redis, ao contrário do caso de hospedar o Redis diretamente em uma EC2.

Amazon API Gateway

Para a criação e disponibilização dos serviços das aplicações (sistema 1, 2 e 3), foi escolhido a utilização do Amazon API Gateway para a construção, manutenção, monitoramento e segurança, em grande escala, da API RESTful, que contém os serviços expostos.

Amazon Virtual Private Cloud (VPC)

Para aumentar segurança da solução foi escolhido utilizar a Amazon VPC. Com ela é possível executar os recursos da AWS em uma rede isolada. Além de permitir criar sub-redes customizadas com um intervalo específico de endereços IP, configuração de tabelas de rotas e limitação de acesso à internet.

Diagrama da Arquitetura

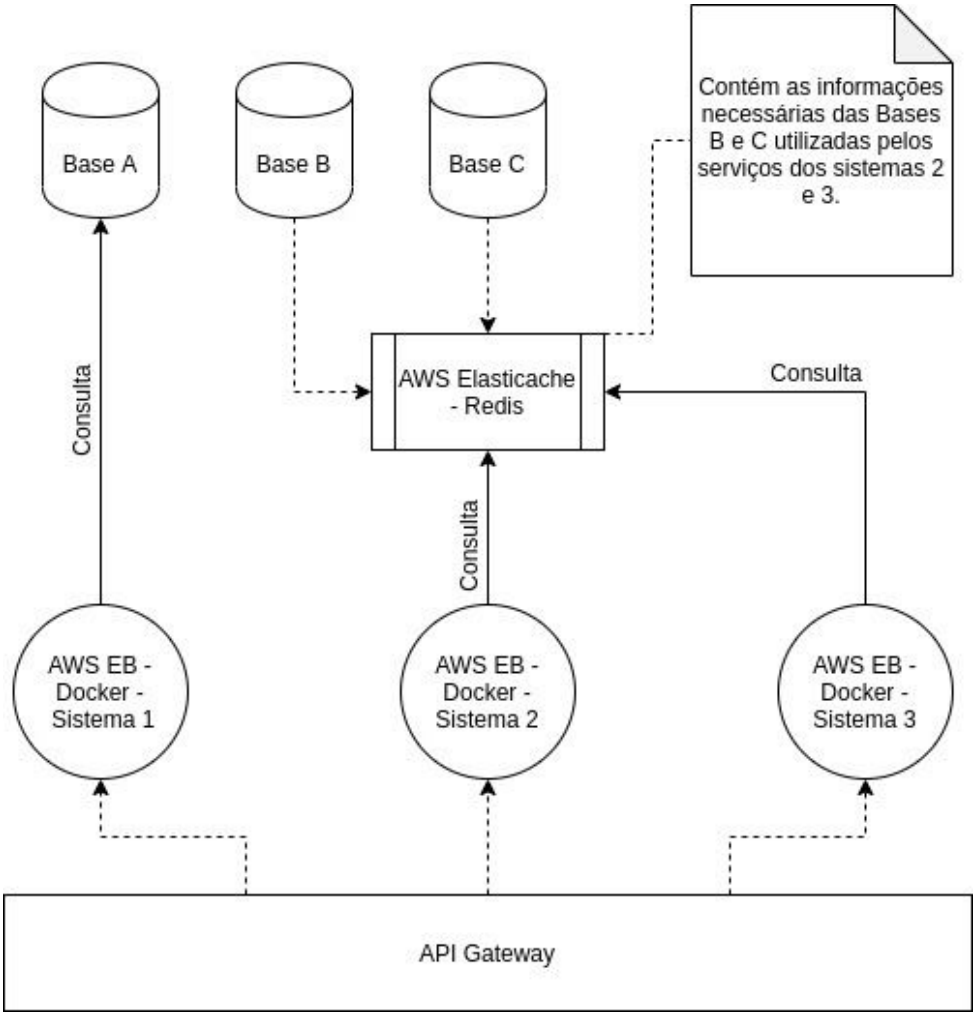

Imagem 1: Arquitetura da solução utilizando AWS

Descrição da Arquitetura

A linguagem de programação escolhida para a implementação dos sistemas foi Java. No sistema 2, alternativamente, pode ser considerado a utilização de Python (com Pytorch) e AWS Sagemaker, para a maior facilidade da implementação, treinamento, *tuning* de hiperparâmetros e validação do algoritmo de aprendizado de máquina.

As configurações de segurança, escalabilidade e performance dos três sistemas serão discutidas a seguir.

Sistema 1

Para a implementação do sistema 1, foi escolhida uma aplicação Docker hospedada no AWS Elastic Beanstalk. Devido à necessidade de consultar a Base A, que contém dados sensíveis, será configurado criptografia ponta a ponta no ambiente.

O AWS Elastic Beanstalk permite essa configuração facilmente pelo console ou arquivos de configuração, por onde o *listener* (aplicação) pode ser configurado para encaminhar o tráfego via https, assim como novas instâncias criadas pelo *load balancer*.

Para melhorar a segurança, o tráfego entre o sistema e a Base A será feita dentro de uma VPC e outra camada de segurança pode ser configurada utilizando o protocolo SSL (*Secure Socker Layer*) para o tráfego entre o sistema e a base.

Sistema 2

Para a implementação do sistema 2, foi escolhido uma aplicação Docker hospedada no AWS Elastic Beanstalk. Para segurança dos dados as mesmas configurações do sistema 1 serão utilizadas.

Conforme descrito no problema, o sistema 2 utiliza algoritmos de aprendizado de máquina e calcula o *Score* de usuários. Por isso são necessárias consultas seguras e performáticas sobre os dados da Base B. Para atender a esse requisito será utilizado o Elasticache com o Redis.

A ideia é que os mesmos serviços que populam as Bases, também escrevam as informações necessárias para o cálculo do *Score* no Redis. Neste caso, seria necessário gravar, dado um cpf, sua lista de bens, endereço e fonte de renda. Desta forma será possível consultar as informações necessárias para o cálculo do Score de forma extremamente rápida.

Sistema 3

Para a implementação do sistema 3, foi escolhido uma aplicação Docker hospedada no AWS Elastic Beanstalk. Como as consultas à Base C devem ser bastante performáticas, a mesma instância do Redis utilizada para o Sistema 2 também será utilizado para o Sistema 3.

Analogamente ao sistema anterior, as informações pertinentes (última consulta de CPF em um *Bureau* de crédito, movimentação financeira do CPF e dados relacionados a

última compra com cartão de crédito vinculado ao CPF) para as consultas do sistema 3 devem ser gravadas no Redis através dos mesmos serviços que gravam nas Bases.

Como as informações consultadas pelos sistemas 2 e 3 são dados selecionados do usuário, não deve existir sobrecarga da instância do Redis por conter as informações que alimentam ambos os sistemas. Além disso, compartilhando a instância do Redis com os dois sistemas, suas performances de consultas serão igualmente rápidas.