Arquiteturas de Servidores

Para que possamos criar nossas aplicações responsáveis por toda lógica de negócio. Uma das decisões que precisamos é qual arquitetura escolher.

Existe varias formas de arquiteturas que pode ser escolhida, como por exemplo: Arquiteturas monolíticas que referece a uma única implantação, sem muita preocupação com a separação da regra de negócio ou com algum serviço externo, uma outra arquitetura é a de micro serviços, que significa separar os serviços da sua aplicação em pequenos serviços, organizando assim é possível isolar cada regra de negocio e assim sem depender muito de outras para estarem funcionando. Uma outra arquitetura que você pode escolher é a Serverless, que tem o objetivo de ser algo simples de se usar e que você só precisa se preocupar com o código, pois a ideia dessa arquitetura é executar um serviço de forma isolada. Com muitas opções de arquitetura pode ser um pouco difícil escolher qual é a melhor, e não existe, pois depende muito do tipo de aplicação que você esteja construindo.

Para ajudar você na sua escolha de qual arquitetura deve utilizar, imagine que você recebeu a tarefa de construir um e-comerce, vamos imaginar só na parte dos serviços, sem pensar nas telas. Cada parte deste desenvolvimento as arquiteturas serão utilizadas de primeira forma para descrever cada uma dessas arquitetura que no final você vai poder escolher qual seria a melhor para o seu contexto.

Monolítico

Uma arquitetura monolítica refere-se a uma única aplicação, sem muita preocupação com a divisão dos serviços, a ideia é imaginar que os monolitos é um sistema na qual todo o código é feito como um único processo. Continuando no desenvolvimento do e-comerce, como na maioria de todo e-commerce você precisa se preocupar em algumas partes como por exemplo: O controle de estoque, serviços de pagamentos, autenticação e entre outros. De certa forma é possível construir uma aplicação desta forma, mas caso seu negocio comece a crescer e você precise escalar, você terá uma enorme dificuldade pois, como toda aplicação está agrupada em uma única unidade. Além disso suas camadas de serviço dependeram bastante de outros serviços da aplicação. Um outro problema de resiliência que você pode estar propenso é a falhas em um determinado ponto, como toda a aplicação está contida em um único código base, se houver uma falha em qualquer parte da aplicação, toda a aplicação pode ficar comprometida, mesmo que você implante vários tratamentos de erros. Para que você possa criar um serviço monolítico na AWS você poderá utilizar alguns desses serviços que a plataforma oferece. EC2(Elastic Compute Cloud) que serve para hospedar a aplicação monolítica em instancias virtuais. Outro serviço importante é o RDS(Relational Database Service) e para que você possa armazenar seus arquivos estáticos, existe o S3(Simple Storage Service).

Micro-serviços

Seguindo a explicação das arquiteturas de serviços, te apresento aos de micro serviços. Nesta arquitetura, a aplicação é dividida em componentes independentes e autônomos, e cada um é responsável por uma função especifica. A forma de se comunicarem é por meio de uma api. Este tipo de arquitetura valoriza a granularidade e a capacidade de compartilhar processos semelhantes entre várias aplicações. Voltando com a construção do e-comerce, seguindo com essas ideias que a arquitetura oferece, agora você pode isolar alguns serviços como por exemplo, o serviço de pagamentos, isolando-o, ele fica livre para estar se conectar com qualquer outro serviço de e-comerce, não necessariamente um especifico. Além disso isolando alguns serviços, sua aplicação principal não será prejudicada caso algum desses serviços de algum erro ou até mesmo fique fora do ar. Sua aplicação não fica presa em apenas um único serviço assim. Alguns problemas que você pode encontrar é a complexidade, pois dividindo sua aplicação em vários outros serviços, você precisa organizar e dar manutenção para todas elas, mas comparados as aplicações monolíticas, os microsserviços são mais fáceis de testar, implantar e atualizar. Caso você opte por utilizar das arquiteturas de microsserviços você pode utilizar alguns desses serviços da AWS, ECS(Elastic Container Service) ou o EKS(Elastic Kubernetes Service) para orquestrar contêineres que hospedem os micro-serviços. O serviço API Gateway, pode ser utilizado para gerenciar e expor as APIs dos micro-serviços.

Serverless

A arquitetura serveless, é uma maneira de executar o código sem se preocupar com servidores. Nessas arquiteturas o servidor ainda é usado, mas eles são isolados do desenvolvimento de aplicações. O provedor da nuvem fica responsável pelas tarefas comuns, como manutenção e escala da infraestrutura do servidor. Utilizando o serviço de nuvem AWS, algumas vantagens que sua aplicação são, a diminuição de custos com o servidor, segundo a aws, “Com um modelo de faturamento de pagamento por valor, a utilização de recursos é otimizada automaticamente e você nunca em seu nome pelo excesso de provisionamento”, com isso você só será cobrado caso sua aplicação necessite de mais recursos e caso não esteja sendo utilizada, não será cobrada. Para o exemplo da criação do e-commerce, você pode utilizar-se do serveless para isolar alguma função que ocorra rotineira ou esporadicamente, como a geração semanal ou mensal de relatórios. Caso você opte por utilizar das arquiteturas de microsserviços você pode utilizar alguns desses serviços da AWS, AWS Lambda, é um serviço computacional orientado por eventos por pagamento conforme o uso que permite executar o código. AWS Fargate, é parecido com o ECS quanto o EKS. Para integração de aplicações, existem os serviços Amazon EventBridge, é um barramento de eventos sem servidor que permite construir aplicações orientadas a eventos em escala por toda AWS e sistemas existentes. Amazon SNS é um serviço de mensagens totalmente gerenciado para a comunicação de aplicação para aplicação A2A e de aplicação para pessoa A2P. Para armazenamento de dados existe o Amazon S3, Amanzon EFS, Amazon DynamoDB, Amazon Aurora Serverless e entre outros.

Para concluir com a apresentação dessas arquiteturas e por fim ficar mais claro qual dessa você escolheria, vale lembrar que todas essas arquiteturas dependem do contexto que sua aplicação se encontra. Com um bom planejamento você terá uma conclusão.