COMPUTAÇÃO EM NUVEM

AULA 5

Profª Ana Paula Costacurta

CONVERSA INICIAL

No Tema 1, teremos uma visão geral do funcionamento do *Amazon DynamoDB*, o serviço de armazenamento NoSQL. Conheceremos os componentes principais, regras de nomeação e tipos de dados. São apresentados exemplos de tabelas para melhor entendimento.

No Tema 2, aprenderemos sobre *Amazon Glacier*, serviço de armazenamento que é considerado uma classe de armazenamento do *Amazon* S3. Conheceremos as formas de armazenamento e recuperação dos arquivos no *Amazon Glacier*.

No Tema 3, conheceremos o *Amazon Virtual Private Cloud* (VPC), camada de reque que isola uma seção para a rede criada na nuvem AWS. São apresentados os componentes e esquemas de endereçamento para acesso privado e público.

No Tema 4, teremos uma visão sobre o *Amazon API Gateway*, serviço da AWS destinado a APIs RESt e *WebSocket*. Conheceremos a arquitetura básica do *Amazon API Gateway*.

No Tema 5, conheceremos sobre o *Amazon Lambda*, serviço para criação de funções para computação sem servidor. Veremos o que é Função Lambda e como é utilizado em aplicações sem servidor. Aprenderemos a como utilizar para implementação de um aplicativo *web* sem servidores ao realizar a combinação com outros serviços da AWS.

TEMA 1 – *AMAZON DYNAMODB*

É um serviço de armazenamento de dados NoSQL sob demanda, com objetivo de fornecer um serviço totalmente gerenciável com desempenho rápido e com facilidade de escalabilidade. Quando se trabalha com banco de dados relacionais tradicional, torna-se difícil o redimensionamento por causa das garantias do ambiente (ACID – Atomicidade, Consistência, Isolamento e Durabilidade), quando um banco de dados não adota essas garantias chamamos de NoSQL.

1.1 Visão geral

Existem quatro tipos de Banco de Dados NoSQL: de documentos, de grafos, de colunas e de armazenamento chave-valor. O *Amazon DynamoDB* não

fornece mecanismos RDBMS, como o MySQL, *Oracle Database*, Microsoft SQL e o PostgreSQL, ele é baseado em chave-valor gerenciado pelo usuário. Caso seja necessário criar um banco de dados de documentos, é necessário criar uma instância do MongoDB no EC2, por exemplo.

O *DynamoDB* possibilita a transferência de cargas administrativas sem a preocupação de provisionamento, instalação e configuração do hardware. Também possui criptografia em repouso, dados que não estão sendo movimentados de um lugar para o outro e que estão armazenados, eliminando, assim, a carga operacional e a complexidade na proteção dos dados críticos.

O *backup* do *Amazon DynamoDB* pode ser realizado sob demanda, podendo ser criado completo de todas as tabelas para armazenamento de longo prazo. Possibilita a recuperação *point-in-time*, recuperação da tabela para qualquer ponto durante os últimos 35 dias, protegendo contra ações acidentais de gravação e exclusão.

Pode ser incluído uma vida útil (TTL) por item no *Amazon DynamoDB* para determinar quando o item não tem mais utilidade, excluindo, assim, o item da tabela após a data e horário definido no *time-stamp* especificado.

1.2 Componentes principais

Os principais componentes com os quais podem ser trabalhados no *Amazon DynamoDB* são:

1) **Tabelas**: os dados são armazenados em tabelas que são coleções de dados. Por exemplo, uma tabela chamada “Pessoa”, os dados armazenados na tabela serão as informações pessoais de cada pessoa, e outra tabela chamada “Contas”, para armazenas as contas-correntes que a pessoa possui.
2) **Itens**: em cada tabela pode ser armazenado zero ou mais itens, que são um conjunto de atributos inidentificáveis. Por exemplo, na tabela “Pessoa”, cada item representa um indivíduo e, na tabela “Contas”, representa uma conta. Não existe um limite de quantidade de itens que se pode armazenar. Podemos comparar com linhas, registros ou tuplas de outros bancos de dados.
3) **Atributos**: cada item é composto por um ou mais atributos, que é o elemento fundamental do dado armazenado e não pode mais ser dividido.

Por exemplo, na tabela “Pessoa”, podemos incluir os atributos: ID, CPF, Sobrenome, Nome; e, na tabela “Contas”, podemos incluir os atributos: Banco, Agencia, Número da Conta. Podemos comparar com campos ou colunas dos outros bancos de dados.

4) **Chave primária**: no momento da criação da tabela é necessário especificar o nome e uma chave primária. A chave primária é a responsável por identificar exclusivamente cada item da tabela, não sendo possível existir dois itens com a mesma chave primária. As chaves primárias podem ser de dois tipos: composta apenas por um atributo, a chave primária simples, que chamamos de chave de partição e composta por dois atributos; e a chave primária composta, que são chamados de chave de partição o primeiro atributo e o segundo de chave de classificação. A chave primária composta oferece uma flexibilidade adicional para consultar dados.
5) **Índices secundários**: além de consultar pela chave primária, permite consultar os dados de uma tabela usando uma chave alternativa. Não existe uma exigência por parte do *Amazon DynamoDB* para criação de índices, mas os índices possibilitam uma flexibilidade maior durante a realização e consultas na base de dados. Os secundários podem ser de dois tipos: globais, com chaves primárias diferentes das contidas na tabela, e Local, que podem ser uma chave de partição igual à chave primária e uma chave de classificação diferente. O limite dos índices são 20 globais e 5 locais por tabela.

Na Figura 1, mostramos uma tabela “Pessoa” com alguns itens e atributos, observações importantes que podemos mencionar:

1) O Item tem um identificador exclusivo, a chave primária que é um atributo único que na tabela de exemplo “Pessoa” podemos citar o atributo “PessoaID” com chave primária.
2) Por não possuir esquema, não é necessário definir os atributos e tipos dos dados previamente. Cada item pode ter seus atributos de forma distinta.
3) Pode ter apenas um valor para cada atributo para os atributos escalar, *strings* e números são atributos escalares.
4) Pode conter atributos aninhados com até 32 níveis de profundidade, que na tabela de “Pessoa” podemos citar o atributo “Endereço” como atributo aninhado com profundidade 5.

Figura 1 – Tabela “Pessoa”

Pessoa


```
{
        “PessoaID”: 101,
        “Sobrenome”: “Silva”,
        “Nome”: “João”,
        “CPF”: “12345678”
}

{
        “PessoaID”: 102,
        “Sobrenome”: “Santos”,
        “Nome”: “José”,
        “CPF”: “98765432”
        “Endereço”: {
                “Rua”: “Industrial”,
                “Numero”: 123,
                “Cidade”: “Curitiba”,
                “Estado”: “Paraná”,
                “CEP”: “80.000-000”
        }
}

{
        “PessoaID”: 103,
        “Sobrenome”: “Silva”,
        “Nome”: “Maria”,
        “CPF”: “456789234”,
        “Celular”: “41-9999-99999”
}
```


Na Figura 2, mostramos uma tabela “Musica”, com chave primária composta, e podemos mencionar algumas observações importantes:

1) A chave primária é composta por dois atributos “Artista” e “TituloMusica”, todos os itens da tabela devem ter esses dois atributos. Além desses dois itens que são a chave primária, os demais atributos e tipos de dados não são previamente definidos, pois as tabelas não têm esquemas.
2) A tabela possui um atributo aninhado “InfoPromocional”, que contém três atributos aninhados e dois deles, “DataTourne”, têm atributos aninhados.

3) O atributo “EstacaoRadio” é um atributo que é uma lista e possui mais de um valor, ou seja, ele não é um atributo escalar. Veremos mais à frente em detalhes os tipos de dados.

Ainda na Figura 2 vemos o índice “GeneroTituloAlbum” e observamos que:

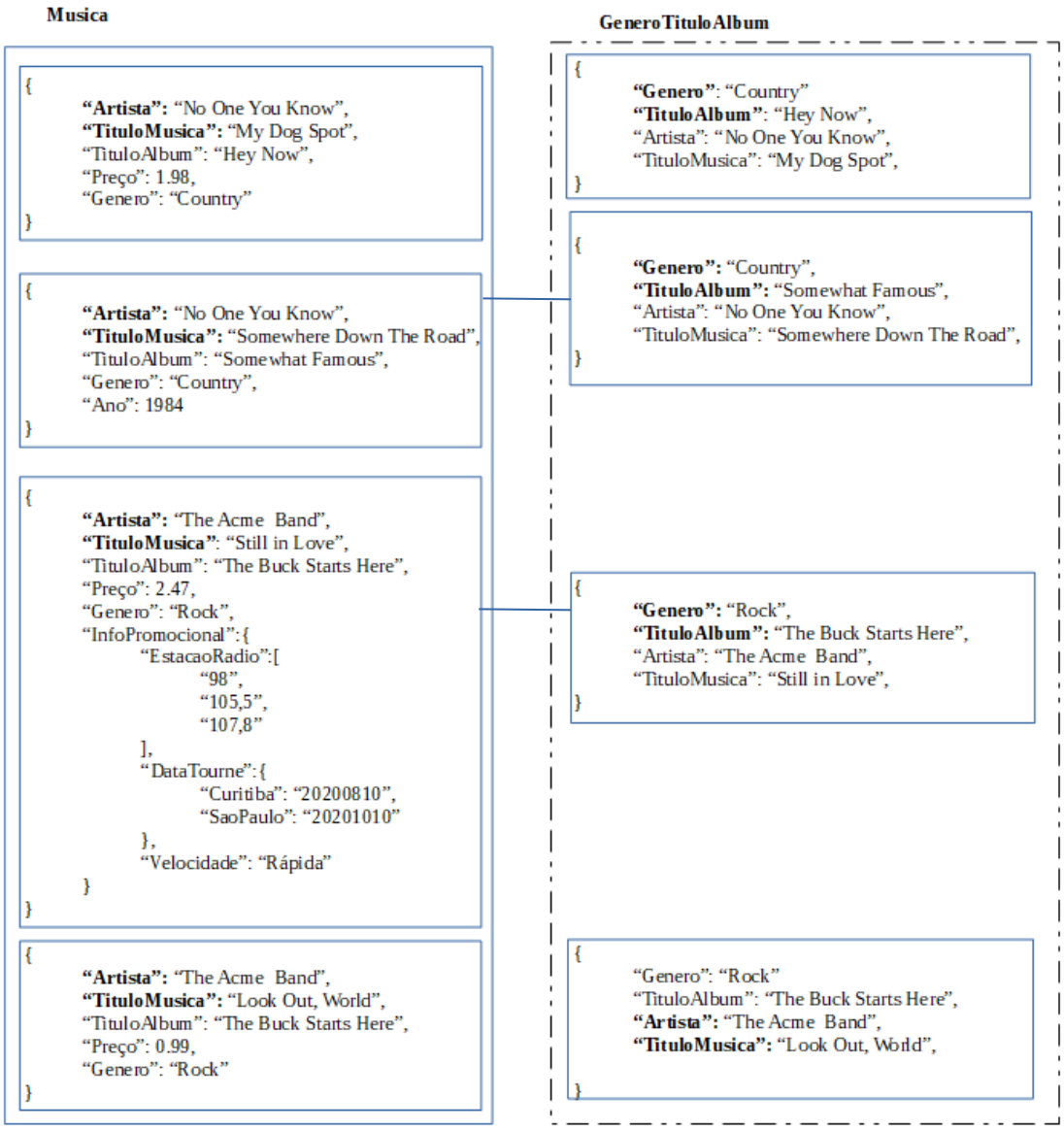
1) A tabela “Musica” é a tabela base para o índice.
2) Os índices são mantidos de forma automática pelo DynamoDB, ou seja, quando adiciona, atualiza e exclui da tabela base ele atualiza o item correspondente na tabela índice.
3) No momento da criação do índice, é possível especificar quais atributos serão copiados da tabela base, no mínimo serão projetados os índices e as chaves da tabela base.
4) Utilizando o índice “GeneroTituloAlbum”, é possível localizar todos os gêneros ou todos os álbuns de um gênero específico.

1.3 Regras de nomeação

Os componentes do *DynamoDB* devem ter nomes, que devem ser concisos e significativos, possuindo algumas regras:

1)Utilizar codificação UTF8 e diferenciação entre maiúsculas e minúsculas.
2)Tabelas e índices devem ter entre 3 – 255 caracteres, sendo eles: [a-z], [A-Z], [0-9], “_” (sublinhado), “-” (traço) e “.” (ponto).
3)Atributos devem ter um comprimento entre 1-255 caracteres.
4)Existem algumas palavras reservadas do *DynamoDB* e não podem ser utilizados os caracteres especiais “#” (*hash*) e “:” (dois pontos).

Figura 2 – Tabela “Musica” e índice “GeneroTituloAlbum”



1.4 Tipos de dados

Quando criamos uma tabela ou índice secundário, é necessário especificar nomes e os tipos de dados de cada atributo chave primária que deve ser definido como tipo *string*, número ou binário. Por ser um banco de dados sem esquema, além dos atributos de chave primária, não é necessário definir nenhum outro atributo ou tipo de dados na criação da tabela.

Os tipos de dados para os atributos das tabelas no *DynamoDB* podem ser categorizados em:

1) **Escalares**: representa exatamente um valor, são tipos escalares número podendo ser positivo, negativo ou zero de até 38 dígitos, *string* são *Unicode* (UFT8) de zero até 400KB, binário são de qualquer tipo

(criptografado, compactado ou imagens) de zero até 400KB, booleano e nulo.

2) **Documentos**: representa uma estrutura com atributos aninhados, são tipos documentos lista e mapa. A Lista armazena um conjunto ordenado de valores e é delimitada por colchetes “[…]”, e o Mapa armazena uma coleção não ordenada de pares de valores nome/valor e são delimitados por chaves “{ … }”. Na Figura 3, podemos ver exemplos de Lista e Mapa.

Figura 3 – Exemplo de Lista e Mapa

Lista

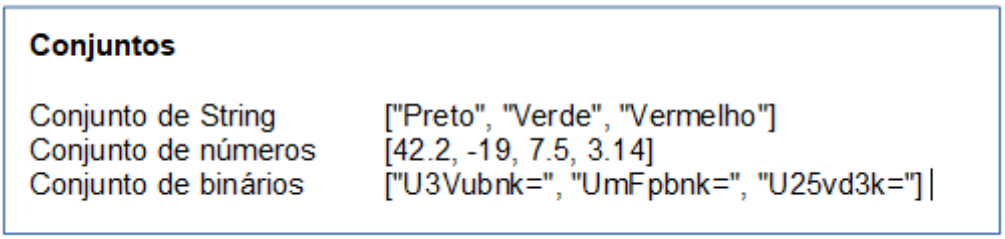
```
Objetos: ["Cookies", "Café", 3.14159]
```

Mapa

```
{
        Dia: "Segunda-feira",
        Email_        nãolidos: 42,
        Meus_Objetos: [
                "Copo_café",
                "Telefone",
                {
                        Canetas: { Quantidade : 3},
                        Lápis: { Quantidade : 2},
                        Borrachas: { Quantidade : 1}
                }
        ]
}
```

3) **Conjuntos**: representa vários valores escalares, são tipos conjunto um conjunto de *string*, um conjunto de números e conjunto de binários. Na Figura 4, podemos ver os exemplos de conjuntos.

Figura 4 – Exemplo de conjuntos



TEMA 2 – *AMAZON GLACIER*

O *Amazon Glacier* é um serviço de armazenamento com custo extremamente baixo. São armazenamentos seguros, resilientes e de custo muito baixo para arquivamento e *backup* de longa duração.

Já há algum tempo, o *Amazon Glacier* é considerado classe de armazenamento em nuvem do *Amazon* S3. Uma parte considerável dos armazenamentos vem diretamente do *Amazon* S3, quando é necessário mover os dados com uma menor quantidades de acesso para o *Amazon Glacier*. Por isso, chamamos de *Amazon* S3 *Glacier* (S3 *Glacier*) e podemos classificar como dois tipos: *Amazon* S3 *Glacier* e o S3 *Glacier Deep Archive*.

O *Amazon* S3 *Glacier* é utilizado quando existe a necessidade de armazenamento de baixo custo e não exista a necessidade de acesso aos dados com velocidade de milissegundos. O custo mínimo do armazenamento é de 0,004 USD por GB-mês (mês de referência Junho/2020).

O *Amazon* S3 *Glacier Deep Archive* é utilizado quando não existe a necessidade de acesso aos dados com frequência, sendo os acessos de uma a duas vezes por ano. O custo de armazenamento mais baixo da nuvem e com segurança, sendo o custo cerca de 1 USB por TB-mês (mês de referência Junho/2020)

São oferecidos três tipos de acesso para recuperação dos dados, em que no *Amazon* S3 *Glacier*, que vão de minutos até algumas horas e no *Amazon Glacier Deep Archive* são oferecidos dois tipos de acesso que vão de 12 a 48 horas.

2.1 Formas de armazenamento

O armazenamento do S3 *Glacier* é realizado como arquivos. Um arquivo é um bloco de informações armazenado de forma durável. O volume total de dados e a quantidade numérica de arquivos que podem ser armazenado no S3 *Glacier* é ilimitado. Os arquivos que podem ser armazenados podem ser individuais ou TAR/ZIP, que é a agregação de diversos arquivos em um único para *upload* no S3 *Glacier*.

O tamanho dos arquivos individuais que pode ser armazenado pode variar de 1 *byte* a 40 *terabytes* e o maior arquivo que pode ser realizado *upload* em uma única solicitação é de 4 GB. Quando realizamos o armazenamento no S3 *Glacier,* os arquivos são imutáveis, somente pode ser feito *upload* e exclusão de arquivos, não é possível realizar edição ou substituição de arquivos no S3 *Glacier*.

Para armazenamento de arquivos grandes, superior a 100MB, pode ser utilizado os recursos *Multipart upload*, que possibilitam a divisão de um arquivo grande em várias partes menores e o *upload* é realizado individualmente. Após a realização do *upload* individual, as partes são combinadas em um único arquivo.

É possível realizar a organização dos arquivos no S3 *Glacier* de forma que realiza o agrupamento de arquivos em conjunto utilizando um cofre. No momento do arquivamento, é possível selecionar o cofre que se deseja utilizar. O limite de cofre por região é de 1000 cofres. Um inventário de todos os arquivos que estão em cada cofre é realizado diariamente, esse inventário é mantido para recuperação de desastre ou para fins de reconciliação necessárias ocasionalmente. Pode ser solicitado esse inventário em formato de um arquivo JSON ou CSV, que conterá os detalhes sobre os arquivos no cofre contendo: tamanho, data criação, descrição do arquivo que foi fornecida no *upload*.

2.2 Recuperação

Quando realizada a solicitação de recuperação dos dados do S3 *Glacier*, após o processamento e conclusão da solicitação de recuperação os dados ficam disponíveis para *download* e acesso no EC2 por um período de 24 horas. São oferecidos três tipos de recuperação de dados: expressa, padrão e em massa.

Na recuperação padrão, o acesso a qualquer um dos arquivos armazenados é disponibilizado após algumas horas, geralmente entre 3-5 horas. A recuperação em massa possibilita recuperar grandes volumes de dados a um custo baixo em um dia, geralmente entre 5-12 horas. Já na recuperação expressa, o acesso é liberado rapidamente, para ser utilizados em casos de urgências ocasionais, sendo sob demanda em 1-5 minutos para arquivos menores de 250MB e arquivos maiores são provisionados para quando não for possível atender a demanda em até 5 minutos.

O custo das recuperações expressas tem valor fixo 0,03 USB por GB + 0,01 USB por solicitação, se recuperamos 10 objetos de 1GB, o custo será de 10x0,03 USB + 10x0,01 USB, totalizando 0,40 USB. A capacidade de provisionamento deve ser comprada e tem um custo de 100 USB por mês.

TEMA 3 – *AMAZON VIRTUAL PRIVATE CLOUD* (VPC)

A *Amazon* VPC é uma camada de rede do *Amazon* EC2 que permite isolar logicamente uma seção da Nuvem AWS para executar recursos em uma rede virtual definida pelo usuário. Como o usuário tem o controle total sobre o ambiente virtual, podem ser escolhidos os endereços IP, criação de sub-redes e configuração de tabelas de rotas e gateways.

Quando é criada a conta na AWS, já vem com uma VPC padrão com recursos automaticamente provisionados, que fica disponível para utilização. Quando não for especificado uma sub-rede ao iniciar uma instância, automaticamente será iniciada na VPC Padrão, ou seja, pode iniciar uma instância sem precisar ter conhecimento de VPC, ficando opcional a criação de outras VPC adicionais. Possui quatro opções básicas para arquitetura de rede:

1) Uma única sub-rede pública
2) Com sub-redes públicas e privadas
3) Com sub-redes públicas e privadas e acesso ao AWS *site-to-site* VPC
4) Uma única sub-rede privada e acesso ao AWS *site-to-site* VPC

As limitações referentes às quantidades atuais do uso da *Amazon* VPC são:

1) Até cinco Amazon VPCs não padrão por conta em cada região
2) Até quatro intervalos de Ips secundários por *Amazon* VPC

3) Até duzentas sub-redes por *Amazon* VPC
4) Até cinco endereços IP *Elastic* de *Amazon* VPC por conta em cada região

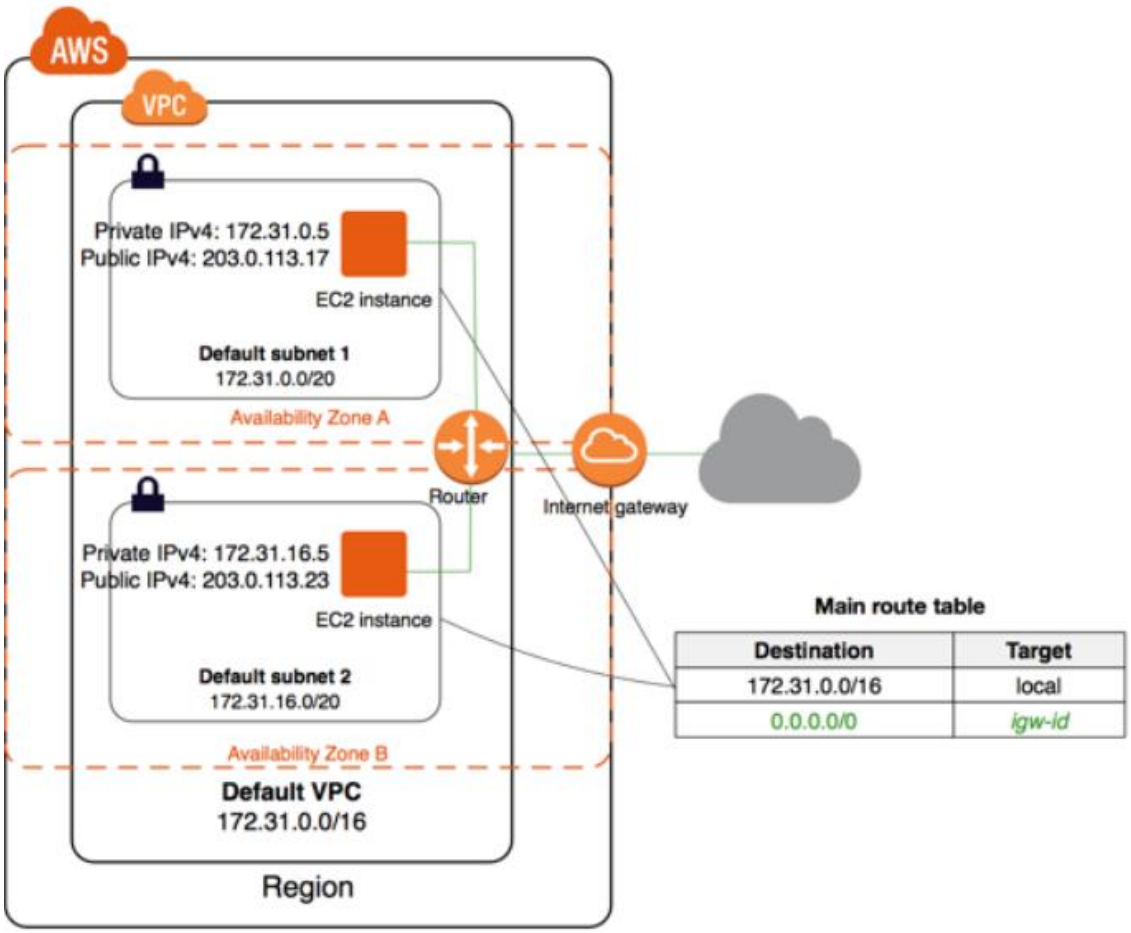
3.1 Componentes

A *Amazon* VPC disponibiliza diversos objetos que já são familiares aos usuários que já possuem redes, sendo eles:

1) Rede virtual: com possibilidade de selecionar endereço IP.
2) Sub-rede: dispor grupos de recursos isolado.
3) Internet *Gateway*: conexão à Internet pública.
4) NAT *Gateway*: conversão de endereços de rede, com gerenciamento e altamente disponível para acesso à Internet.
5) *Virtual Private Gateway*: conexão VPN.
6) Conexão emparelhada: rotear tráfego por meio de endereços IP privados entre duas VPCs emparelhadas.
7) VPC *Endpoints*: conectividade privada de uma VPC a serviços hospedados na AWS sem usar Internet *Gateway*, VPN, dispositivos de *Network Address Translation* (NAT) ou *proxies* de *firewall*.
8) Internet *Gateway* somente para saída: acesso somente de saída a tráfego IPv6 da VPC para a Internet.

O VPC padrão inclui um *Gateway* da internet e cada sub-rede padrão é uma rede pública. Está padronizado que cada instância que for iniciada, possui um endereço IPv4 privado e um endereço IPv4 público e a comunicação com a internet é realizada pelo *Gateway* da internet por meio da borda da rede do *Amazon* EC2. Na Figura 5, pode ser visto o esquema dos endereços de uma instância no VPC padrão.

Figura 5 – Esquema dos endereços VPC Padrão

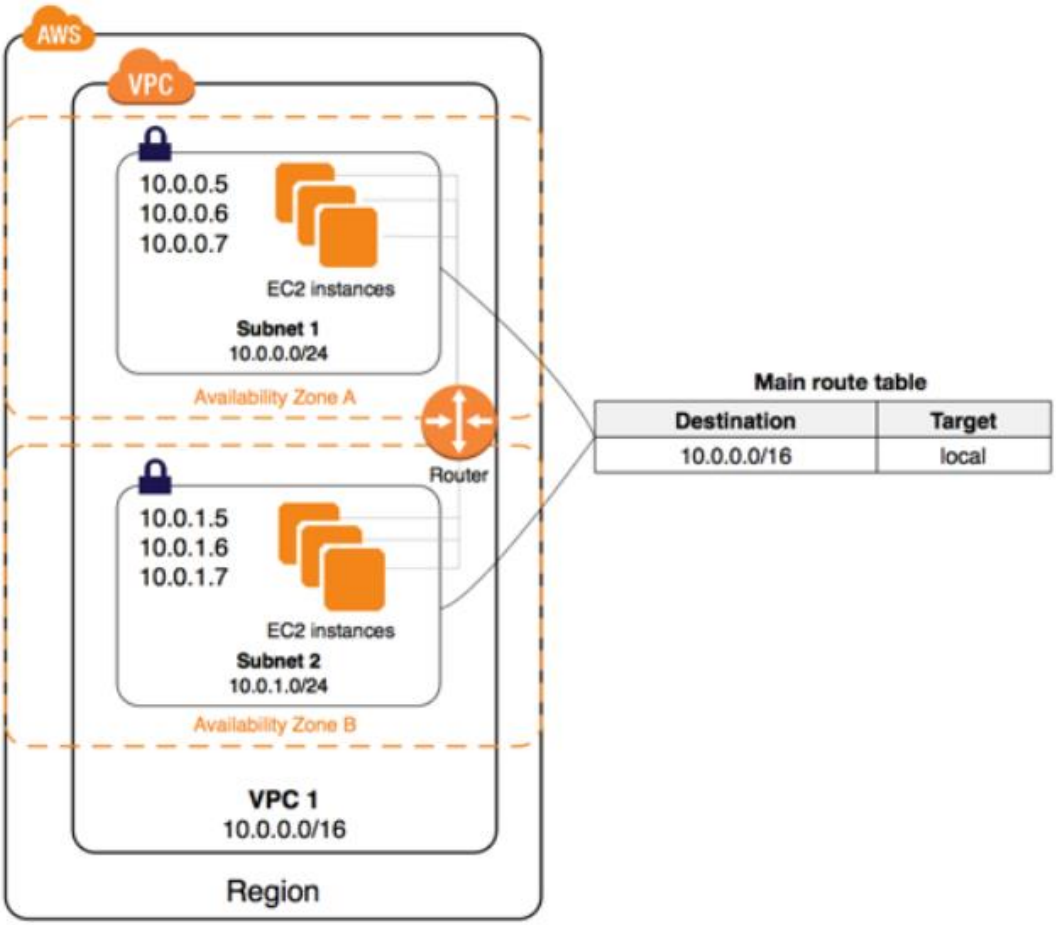


Fonte: AWS Brasil, 2020.

Agora, cada instância que é iniciada em um sub-rede não padrão, possuirá apenas um endereço IPv4 privado e nenhum endereço IPv4 público. Caso seja necessário, deve ser realizada a atribuição do endereço IP público da sub-rede. Sendo assim, essas instâncias ficam em um sub-rede não padrão quando criadas e não são atribuídos IP Público.

Na Figura 6, pode ser visto um esquema dos endereços IP para instâncias iniciadas em uma sub-rede não padrão.

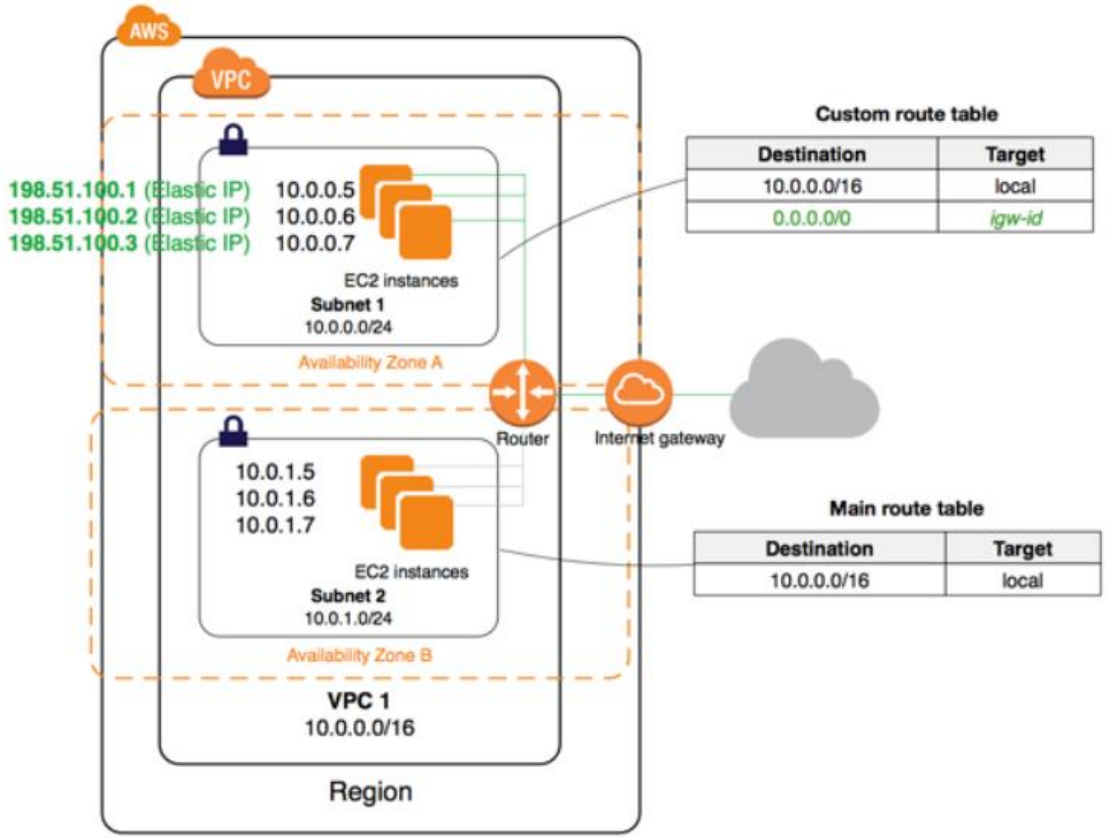
Figura 6 – Esquema dos endereços IP para VPC não padrão



Fonte: AWS Brasil, 2020.

Por padrão, as instâncias iniciadas em sub-redes não padrão ficam sem acessar a Internet, podendo apenas se comunicam entre as instâncias até que seja feita a modificação ou que na criação tenha sido realizada a especificação do IP Público. Quando for necessário habilitar o acesso à Internet de uma instância iniciada em uma sub-rede não padrão, é necessário anexar um *Gateway* a VPC e associar um endereço IP *Elastic*, como apresentado na Figura 7.

Figura 7 – Esquema de endereço IP para acesso Internet VPC não padrão



TEMA 4 – *AMAZON API GATEWAY*

O *Amazon API Gateway* é um serviço utilizado para criação, publicação, manutenção, monitoramento e proteção de API REST (*Representacional State Transfer*) e *WebSocket*.

A *Application Programming Interface* (API) é um conjunto de rotinas e padrões estabelecidos e documentados para uma aplicação para que possibilite que outras aplicações usem as funcionalidades desta. Quando existe um aplicativo que é possibilitado de realizar a interação, ele é chamado de *endpoint* HTTP. Um *endpoint* fornece uma forma de definir URL base e as credenciais de autenticação para realização das solicitações HTTP.

A implementação de uma API *Gateway* em uma região específica é chamada de *endpoint* de API, que tem o seguinte formato: *{api-id}.execute-api.{region}.amazonaws.com*.

4.1 Conceitos

O API REST do API *Gateway* é a integração de recursos e métodos aos *endpoints* HTTP de *back-end*, funções *lambda* ou outros serviços da AWS. Utiliza o modelo em que um cliente envia uma solicitação para um serviço e o serviço responde de forma síncrona. *"Representational State Transfer"* (REST) são princípios, regras para criação de uma interface bem definida permitindo que aplicações se comuniquem.

Quando uma aplicação tem a capacidade de aplicar os princípios REST, chamamos de RESTful. A criada de APIs RESTful são baseadas em HTTP, habilita a comunicação cliente servidor sem estado e implementam métodos padrão (*GET*, *POST*, *PUT*, *PATCH* e *DELETE*).

O HTTP API do API *Gateway* é integração de rotas e métodos com endpoints HTTP ou funções lambda de *back-end*. Pode ser utilizado para envio de solicitações para funções do AWS *Lambda* ou para qualquer endpoint HTTP roteáveis. A utilização do HTTP API permite a criação de aplicativos RESTful com menor latência e menor custo do que REST API.

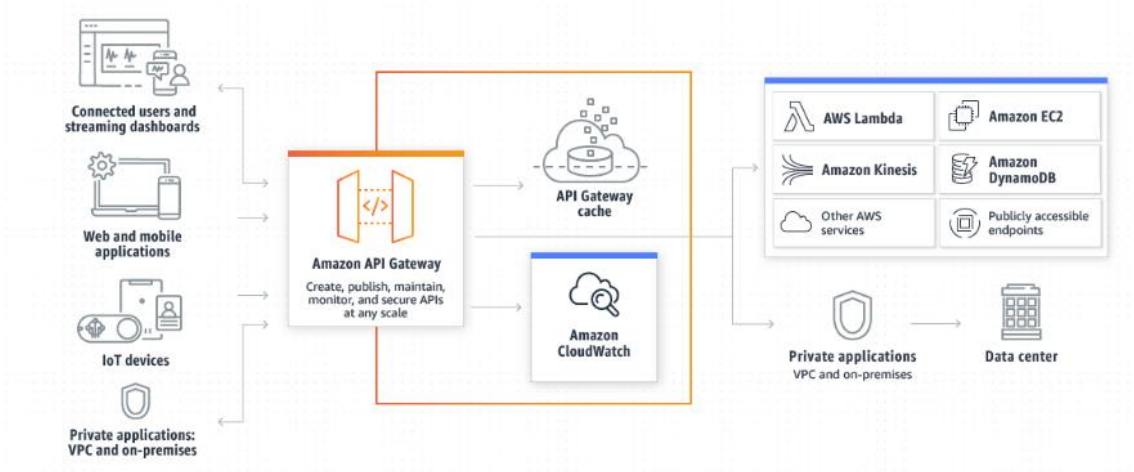
O API *WebSocket* no API *Gateway* é a integração de rotas e chaves de rota com *endpoints* HTTP de *back-end*, funções *lambda* ou outros serviços da AWS. São bidirecionais, ou seja, um cliente pode enviar uma mensagem e serviços podem enviar mensagem para clientes de forma independente.

São utilizados para aplicativos em tempo real, como bate-papo, jogos multijogador etc. A criação de APIs *WebSocket* que seguem o protocolo *WebSocket,* permitindo a comunicação *full-duplex* entre cliente e servidor com estado e realizada o roteamento de mensagens recebidas baseado no conteúdo da mensagem.

Na Figura 8, podemos visualizar a arquitetura do API Gateway, em que o API *Gateway* atua como a "porta da frente" para os aplicativos acessarem os dados, lógica de negócios ou funções de seus serviços de *back-end*.

É possível realizar o compartilhamento de recursos de origem cruzada (*CORS*), definindo, assim, a maneira como os aplicativos *web* carregados e um domínio podem interagir com recursos em outro domínio. A configuração *bucket* para permitir solicitações *CORS* é um arquivo XML com regras que identificam as origens que podem acessar seu *bucket.*

Figura 8 – Arquitetura API *Gateway*



Fonte: AWS Brasil, 2020.

Utilizando o *Amazon* Cognitivo é possível adicionar controle de acesso simples e seguro, possui escala de até milhões de usuários e possui *login* com provedores de identidade social (Facebook, Google, Amazon). O *Amazon* Cognitivo é compatível com o padrão aberto para autenticação *OpenID Connect*, que é compatível com vários provedores de *login*.

4.2 Comparação entre HTTP APIs e REST APIs

As APIs REST oferecem atualmente mais recursos e controle total sobre solicitações e respostas da API. Os HTTP APIs são projetados para API de *proxy* HTTP e de *proxy Lambda* de baixa latência. Na Figura 9, pode ser visto um resumo dos principais recursos disponíveis em HTTP APIs e APIs REST.

4.3 Integração para HTTP APIs

O HTTP API possui compatibilidade com *proxy lambda* e *proxy* HTTP. A integração com *proxy Lambda* realizada a integração de uma rota de API para uma função *lambda*. Quando realizada uma chamada a API, o API *Gateway* envia a solicitação para a função *lambda* e retorna a resposta da função para o cliente. A integração de *proxy* HTTP possibilita a conexão de uma rota de API a um endpoint HTTP roteável publicamente, bastando apenas fornecer a URL do *endpoint* HTTP para criar a integração. Neste último caso, pelo API *Gateway* passa toda a solicitação e resposta entre *front-end* e *back-end*.

	RECURSOS	HTTP API	REST API
AUTORIZAÇAO	AWS Lambda		✓
	IAM		✓
	Amazon Cognito	✓	✓
	Native OpenID Connect / OAuth 2.0	✓	
INTEGRAÇÃO	HTTP proxy	✓	✓
	Proxy Lambda	✓	✓
	HTTP		✓
	Serviços da AWS		✓
	Integração privada	✓	✓
	Simulado		✓
GERENCIAMENTO DE APIS	Planos de uso		✓
	Chaves de API		✓
	Nomes de domínios personalizados	✓	✓
DESENVOLVIMENTO	Cache		✓
	Solicitar transformação		✓
	Validação do pedido/resposta		✓
	Testar invocação		✓
	Configuração de CORS	✓	
	Implantações automáticas	✓	
	Estágio padrão	✓	
	Rota padrão	✓	
SEGURANÇA	Certificados do cliente		✓
	AWS WAF		✓
	Políticas de recursos		✓
TIPO DE API	Regional	✓	✓
	Otimizado para fronteiras		✓
	Private		✓
MONITORAMENTO	Logs de acesso ao Amazon CloudWatch Logs	✓	✓
	Logs de acesso ao Amazon Kinesis Data Firehose		✓
	Logs de execução		✓
	Métricas do Amazon CloudWatch	✓	✓
	AWS X-Ray		✓

TEMA 5 – *AMAZON LAMBDA*

A *Amazon Lambda* possibilita a execução de códigos para quase todos os tipos de aplicativos ou serviço *back-end*, sem ter a necessidade de administração. Não existe a necessidade de provisionamento e gerenciamento de servidores. Não existe cobrança sobre o código que está em execução, apenas a cobrança pelo tempo de utilização de computação.

O conceito da *Amazon Lambda* é a computação sem servidor, ou seja, possibilita a criação e execução de aplicativos e serviços sem nenhuma preocupação com o servidor, pois o gerenciamento e responsabilidade do servidor será da AWS.

Possui suporte nativo para os códigos em linguagens: *Java*, *Go*, *PowerShell*, Node.js, C#, *Phyton* e *Ruby*, e também possui uma API de tempo de execução que possibilita utilizar qualquer linguagem de programação adicional. O Código é armazenado no *Amazon* S3 e, quando ocioso, é criptografado para proteção do código.

Quando em execução, o código é isolado em um ambiente próprio, com seus próprios recurso, porém, não é possível realizar verificação de saúde, aplicar *patches* de segurança ou qualquer outra atividade de rotina de manutenção de infraestrutura, pois opera em uma infraestrutura de computação da AWS *Lambda*.

O *Amazon Lambda* pode ser utilizado para criar códigos *back-end* que recuperam ou transformam dados do *Amazon DynamoDB*, manipular objetos compactando ou transformando no momento de carregamento no *Amazon* S3, relatórios e auditorias de chamadas feitas por *Amazon Web Services*, entre outros. A *Amazon* API *Gateway* acompanhado do *Amazon Lambda* formam um conjunto voltado para aplicativos sem servidores do AWS.

5.1 Funções *Amazon Lambda*

Função *Lambda* é o nome dado para o código que é executado no AWS *Lambda*. Quando criamos a função *Lambda,* ele fica pronto para ser executado a qualquer momento que for executado. Após realizar o *upload* do código no AWS *Lambda*, pode ser realizada a associação aos recursos da AWS, sendo que as funções lambda são "*stateless*", ou seja, não possuem afinidade com a infraestrutura adjacente e rapidamente pode realizar quantas cópias forem necessárias para criar a escalabilidade necessária de acordo com o volume que o evento recebe.

5.2 Criação de aplicações sem servidor

As aplicações sem servidor são basicamente funções que são acionadas por algum evento. Existem regras para implantação de aplicações sem servidor no AWS, caso o desejo seja utilizar AWS *Serveless Applicantion Model* (AWS SAM). Essa especificação AWS SAM está alinhada com a sintaxe utilizada pelo AWS *CloudFondation*, que disponibiliza API para facilitar a utilização dos recursos, implantação e configuração de aplicações sem servidor.

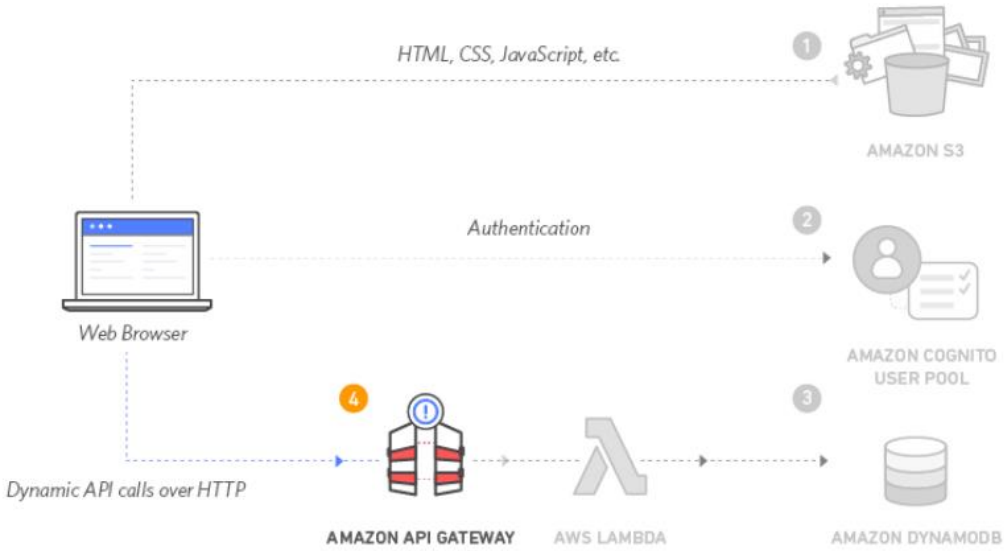
Existe uma coleção de aplicações sem servidor já publicados por desenvolvedores, empresa e parceiros da comunidade AWS que ficam armazenados no AWS *Serveless Application Repository*.

5.3 Implementação de um aplicativo *web* sem servidores

Ao realizar a combinação do AWS *Lambda* com outros serviços da AWS, é possível criar aplicativos *web* sem servidores, com a capacidade de expandir e diminuir de forma automática e serem executados com configuração para vários datacenter.

A visão geral da arquitetura geral de um aplicativo *web* utilizando *Amazon* API *Gateway* e AWS *Lambda* pode ser visualizada na Figura 10.

Figura 10 – Arquitetura geral do aplicativo *web* utilizado no *Amazon* API *Gateway* e AWS *Lambda*



5.4 Integração de *proxy* do *Lambda* do API *Gateway*

Para criação de uma API com uma configuração de um método único de API, utiliza-se um mecanismo simples chamado integração do *proxy* do *Amazon* API *Gateway Lambda*. Permite acesso a muitos recursos ou recursos de outros serviços da AWS, em que o cliente chama uma única função do *Lambda* no *back-end*.

O funcionamento é potente e ágil, quando um cliente envia uma solicitação de API, a solicitação é repassada para função *Lambda* pelo API *Gateway* de forma bruta. É possível realizar a configuração da integração para qualquer método de API, sendo mais potente para métodos de API que envolvem recursos de *proxy* genérico.

FINALIZANDO

No Tema 2, falamos sobre *Amazon Glacier*, que é um serviço de armazenamento seguro e resiliente com custo muito baixo para arquivamento e backup de longa duração. Considerada uma classe de armazenamento do *Amazon* S3, pois uma parte considerável dos armazenamentos vem do *Amazon* S3, por isso é chamado de *Amazon* S3 *Glacier*.

Existem dois tipos de classificações: S3 *Glacier,* que é armazenamento sem necessidade de acesso com velocidade de milissegundos, e S3 *Deep Archive*, que é armazenamento sem a necessidade de acesso com frequência, sendo acessos de uma a duas vezes por ano.

O Armazenamento do S3 *Glacier* é realizado em arquivos individuais ou agregados (TAR/ZIP), sendo os arquivos imutáveis, e é possível apenas duas ações: *upload* e exclusão. Para arquivos acima de 100MB, pode ser utilizado o *Multipart upload,* que possibilita a divisão dos arquivos em várias partes menores para realização do *upload* individualmente e, após isso, realizar a combinação em apenas um arquivo para armazenamento.

A recuperação dos arquivos pode ser de três tipos: expressa, utilizado em casos de emergência em que o acesso é liberado em minutos; padrão, em que o acesso é liberado entre 3-5 horas; e a recuperação em massa, é a recuperação de grandes volumes com acesso geralmente entre 5-12 horas.

No Tema 3, falamos sobre *Amazon* VPC, que é uma camada de rede do *Amazon* EC2 que permite o isolamento lógico da nuvem AWS, criando, assim, uma rede virtual com controle total sobre o ambiente.

Na criação da conta AWS, já é criada uma VPC padrão com recursos provisionados, por isso, é possível trabalhar com instâncias sem nenhum conhecimento de VPC. As opções básicas de arquitetura de rede são quatro: uma unida sub-rede pública, com sub-redes públicas e privadas, com sub-redes públicas e privadas e acesso ao AWS *site-to-side* VPC e uma única sub-rede privada e acesso ao AWS *site-to-site* VPC.

Quando utilizado o VPC padrão, cada sub-rede é uma rede pública e possui um endereço privado e um público com comunicação com a internet, utilizando os VPN não padrão, as sub-redes ficam sem acesso à Internet e podem apenas comunicar-se entre instâncias, sendo necessária a criação de especificação do IP público para comunicação com a Internet.

No Tema 4, falamos sobre *Amazon* API *Gateway*, que é um serviço utilizado para API REST e API *WebSocket*. API Rest é o modelo de comunicação que um cliente envia solicitação e o serviço responde a solicitação de forma síncrona. O API *WebSocket* é o modelo de comunicação em que tanto o cliente como o serviço podem enviar mensagens de forma independente.

O API *Gateway* realiza o papel de porta da frente para os aplicativos acessarem os seus serviços de *back-end*. A utilização de APIs HTTP, em que cria a integração de rotas e métodos com *endpoints* HTTP ou funções *lambda* permite a criação de aplicações RESTful.

Aplicações RESTful são aplicações que seguem o princípio e regras para criação de interface bem definida que permite a comunicação entre aplicações. Comparando HTTP APIs e REST APIs, pode-se perceber que a APIs Rest oferecem mais recursos, controle e são mais seguras por oferecem Certificado do cliente, AWS WAF e políticas de recursos que o HTTP APIs não possui. O HTTP API pode realizar a integração com proxy lambda para criar rota para uma função *lambda* e *proxy* HTTP para criar uma rota para um *endpoint* HTTP roteável publicamente.

No Tema 5, falamos sobre *Amazon Lambda,* que possibilita a execução de códigos sem a necessidade de provisionamento, gerenciamento de servidores. Possui suporte a vários tipos de linguagem incluído *Java*, C# e *Phyton*. É seguro, pois quando o código não está em execução é armazenado no *Amazon* S3 criptografado.

Funções *Lambda* são códigos que não possuem afinidade com a infraestrutura adjacente (*stateless*) e pode-se realizar rapidamente quantas cópias forem necessárias para criar escalabilidade de acordo com volume que cada evento recebe.

Funções que são acionadas por algum evento são chamados de aplicações sem servidor, que utilizam especificações AWS SAM que disponibiliza API para facilitar a utilização de recursos, implantação e configuração das aplicações sem servidor. Pode-se utilizar aplicações já publicadas na comunidade AWS que ficam armazenados no AWS *Serveless Application Repository*.

É possível criar aplicativos *web* sem servidores com a combinação de AWS *Lambda* e outros serviços do AWS. A integração do *proxy Lambda* do API *Gateway* permite que o cliente chame uma única função do *Lambda* para acessar vários recursos ou recurso de outros serviços da AWS.

REFERÊNCIAS

WITTIG, A.; WITTIG, M. **Amazon web services em ação**. São Paulo: Novatec, 2015.

AWS BRASIL. Disponível em: <https://aws.amazon.com/pt>. Acesso em: 16 out. 2020.