Visão Geral para o Estudante

Olá, futuro(a) especialista em nuvem! Bem-vindo(a) de volta. Para aquecer os motores, vamos construir uma pequena aplicação web que usa tecnologias muito relevantes no mercado.

O que vamos usar?

1. **PHP**: Uma das linguagens de programação para web mais populares do mundo. É a linguagem que vai processar a lógica da nossa aplicação no lado do servidor.
2. **AWS DynamoDB**: Um banco de dados NoSQL (não relacional) do tipo chave-valor e de documentos, oferecido pela Amazon Web Services (AWS). Suas principais vantagens são a alta performance, escalabilidade massiva e o fato de ser totalmente gerenciado (serverless). Você não precisa se preocupar com servidores, patches ou instalações.
3. **LocalStack**: Uma ferramenta fantástica que simula o ambiente da AWS na sua própria máquina. Com ele, podemos desenvolver e testar aplicações que usam serviços como DynamoDB, S3, Lambda, etc., sem precisar de uma conta na AWS e sem gerar custos. É perfeito para aprendizado e desenvolvimento local.
4. **AWS SDK for PHP**: É a "ponte" oficial que permite que nossa aplicação em PHP se comunique com os serviços da AWS (neste caso, o DynamoDB simulado pelo LocalStack).
   Obs: Você já conhece boa parte das tecnologias acima, agora vamos juntá-las para a criação de um App.

O que a nossa aplicação fará?

Criaremos um **Mural de Recados** online. Qualquer pessoa poderá visitar a página, ver os recados deixados por outros e adicionar um novo recado com seu nome e uma mensagem. É um exemplo simples, mas perfeito para demonstrar como:

- Conectar uma aplicação a um serviço AWS.
- Gravar dados (CREATE) em uma tabela DynamoDB.
- Ler e exibir dados (READ) de uma tabela DynamoDB.

Este projeto vai solidificar seus conhecimentos sobre a interação entre uma aplicação e um serviço de nuvem de forma prática e controlada.

Preparando o Terreno: Configuração do Ambiente Local

Siga estes passos com atenção para garantir que tudo funcione corretamente.
Pré-requisitos:

- **Docker Desktop** e **Docker Compose**: Essenciais para rodar o LocalStack.
- **PHP**: Versão 7.4 ou superior.
- **Composer**: O gerenciador de dependências para PHP.
- **AWS CLI**: A interface de linha de comando da AWS. Usaremos para criar nossa tabela no DynamoDB local.

Passo 1: Criar e Iniciar o LocalStack com Docker Compose

1. Crie uma pasta para o seu projeto, por exemplo: mural-online.
2. Dentro desta pasta, crie um arquivo chamado docker-compose.yml com o seguinte conteúdo:

```
version: '3.8'
services:
  localstack:
    image: localstack/localstack:latest
    container_name: localstack_main
    ports:
      - "127.0.0.1:4566:4566"            # Porta principal para todos os serviços
      - "127.0.0.1:4510-4559:4510-4559"  # Portas legadas (opcional)
    environment:
      - SERVICES=dynamodb # Inicia apenas o serviço DynamoDB para economizar recursos
      - DEBUG=0
      - DOCKER_HOST=unix:///var/run/docker.sock
    volumes:
      - "${LOCALSTACK_VOLUME_DIR:-./volume}:/var/lib/localstack"
      - "/var/run/docker.sock:/var/run/docker.sock"
```

3. **Entendendo o docker-compose.yml**: Este arquivo descreve os serviços que compõem nossa aplicação. No nosso caso, temos apenas um serviço: o localstack.
    - version: '3.8': Define a versão da sintaxe do Docker Compose que estamos usando.
    - services:: Seção principal onde definimos cada contêiner (serviço).
    - localstack:: É o nome que damos ao nosso serviço.
    - image: localstack/localstack:latest: Especifica qual imagem Docker será usada para criar o contêiner. Aqui, usamos a imagem oficial mais recente do LocalStack.
    - container_name: localstack_main: Dá um nome fixo e amigável ao nosso

contêiner, facilitando sua identificação.
    - ports:: Mapeia as portas entre a sua máquina (host) e o contêiner. A linha "127.0.0.1:4566:4566" significa que a porta 4566 do contêiner (usada pelo LocalStack) estará acessível na porta 4566 da sua máquina, mas apenas localmente (127.0.0.1), por segurança.
    - environment:: Define variáveis de ambiente dentro do contêiner.
        - SERVICES=dynamodb: Instrução específica do LocalStack para iniciar **apenas** o serviço do DynamoDB. Isso economiza memória e acelera a inicialização.
        - DEBUG=0: Desativa logs de depuração para uma saída mais limpa.
    - volumes:: Garante a persistência dos dados.
        - "${LOCALSTACK_VOLUME_DIR:-./volume}:/var/lib/localstack": Mapeia uma pasta chamada volume no seu diretório de projeto para a pasta onde o LocalStack guarda seus dados. **Isso é muito importante**, pois garante que, se você parar e reiniciar o contêiner, sua tabela do DynamoDB não será perdida.
4. Abra o terminal na pasta do projeto e execute o comando para iniciar o contêiner do LocalStack em segundo plano:
   docker-compose up -d

Configuração do Ambiente no Windows

Este guia foi feito especialmente para você que usa o Windows 10 ou 11. Siga os passos com atenção para garantir que tudo funcione corretamente.

Pré-requisitos: Ferramentas Essenciais

A maneira mais fácil de instalar as ferramentas de linha de comando no Windows é usando um gerenciador de pacotes como o **Chocolatey**. Ele funciona como um "App Store" para desenvolvedores.

1. **Instale o Chocolatey (se ainda não tiver)**:
    - Abra o **PowerShell como Administrador**.
1. Execute o comando a seguir e aguarde a conclusão:
   Set-ExecutionPolicy Bypass -Scope Process -Force;
   [System.Net.ServicePointManager]::SecurityProtocol =
   [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

   Feche e reabra o PowerShell como Administrador para usar o choco.

2. **Composer**:
    - **O que é?** O gerenciador de dependências para PHP.

   Como instalar (via Chocolatey)?
   choco install composer

3. **AWS CLI (Command Line Interface)**:

Passo 1: Criar e Iniciar o LocalStack

1. Crie uma pasta para o seu projeto (ex: `C:\cnpw2\19_aula\mural-online`).
2. Dentro desta pasta, crie o arquivo `docker-compose.yml` (o conteúdo é **exatamente o mesmo** da versão anterior).
3. Abra o **PowerShell** ou o **Prompt de Comando (CMD)** na pasta do seu projeto.
4. Inicie o contêiner do LocalStack:
   docker-compose up -d
5.

   Aguarde o download da imagem e a inicialização do contêiner.

Passo 2: Preparar o Projeto PHP

2. No mesmo terminal (PowerShell ou CMD) na pasta do projeto, execute:
   composer require aws/aws-sdk-php

Passo 3: Criar a Tabela no DynamoDB (via LocalStack)

1. **Atenção ao Comando no Windows**: O caractere `\` para quebrar linhas (usado em Linux/macOS) não funciona no Prompt de Comando do Windows. Para evitar problemas, o comando abaixo está em uma **única linha**. Copie e cole-o diretamente no seu terminal (PowerShell ou CMD).
   aws dynamodb create-table --table-name MuralDeRecados --attribute-definitions AttributeName=id,AttributeType=S --key-schema AttributeName=id,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 --endpoint-url http://localhost:4566
2. Para verificar se a tabela foi criada, liste as tabelas:
   aws dynamodb list-tables --endpoint-url http://localhost:4566

   A saída deve mostrar sua tabela `MuralDeRecados`.

Pronto! Seu ambiente Windows está 100% configurado. Agora você pode criar o arquivo `index.php` (com o código da aplicação usando Bootstrap)

Explicação:

- --table-name: Nome da nossa tabela.
- --attribute-definitions: Define os atributos da chave. id será uma String (S).
- --key-schema: Define que id é a nossa chave de partição (HASH).
- --endpoint-url: **O mais importante!** Aponta para o LocalStack.

3. Para verificar se a tabela foi criada, liste as tabelas:
   aws dynamodb list-tables --endpoint-url http://localhost:4566

   Você deverá ver MuralDeRecados na lista.

Seu ambiente está pronto! Agora vamos para o código da aplicação.

Aplicação Web: Mural de Recados (PHP + Bootstrap 5)

```
<?php
// index.php

// 1. Incluir o autoloader do Composer para carregar o AWS SDK
require 'vendor/autoload.php';

// Importar as classes necessárias do SDK
use Aws\Sdk;
use Aws\Exception\AwsException;
use Aws\DynamoDb\DynamoDbClient;
use Aws\DynamoDb\Marshaler;

// 2. Configurar o Cliente do DynamoDB para conectar ao LocalStack
$sdk = new Sdk([
    'endpoint'   => 'http://localhost:4566', // Aponta para o LocalStack
    'region'     => 'us-east-1',            // Região padrão, necessária pelo SDK
    'version'    => 'latest',
    'credentials' => [                      // Credenciais de teste para o LocalStack
        'key'    => 'test',
        'secret' => 'test',
    ]
]);

$dynamodb = $sdk->createDynamoDb();
$tableName = 'MuralDeRecados';

//
================================================================================
=========
// EXPLICAÇÃO: O Marshaler
// DynamoDB não armazena JSON simples. Ele usa um formato descritivo onde cada valor
// é um objeto que especifica seu tipo (ex: {"S": "meu texto"} para String,
// {"N": "123"} para Number). O Marshaler é uma classe utilitária do SDK que
// faz a "mágica" de converter:
//   - um array/JSON PHP normal para o formato do DynamoDB (marshal)
//   - o formato do DynamoDB de volta para um array PHP normal (unmarshal)
// Isso simplifica muito nosso código.
//
================================================================================
=========
```

```
$marshaler = new Marshaler();

$recados = [];
$erro = '';

// 3. Lógica para lidar com o envio de um novo recado (método POST)
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!empty($_POST['autor']) && !empty($_POST['mensagem'])) {
        $autor = htmlspecialchars(strip_tags($_POST['autor']));
        $mensagem = htmlspecialchars(strip_tags($_POST['mensagem']));
        $id = uniqid('recado_', true); // Gera um ID único para cada recado
        $timestamp = time();

        // Usando o Marshaler para criar o formato de item que o DynamoDB entende
        $item = $marshaler->marshalJson('
            {
                "id": "' . $id . '",
                "autor": "' . $autor . '",
                "mensagem": "' . $mensagem . '",
                "timestamp": ' . $timestamp . '
            }
        ');

        $params = [
            'TableName' => $tableName,
            'Item' => $item
        ];

        try {
            //
================================================================
            // EXPLICAÇÃO: Operação putItem
            // putItem é a operação fundamental de escrita no DynamoDB.
            // Ela insere um novo item na tabela. Se um item com a mesma
            // chave primária ('id', no nosso caso) já existir, ele será
            // completamente substituído pelo novo item. É uma operação de "upsert".
            //
================================================================
            $dynamodb->putItem($params);

            // Redireciona para a mesma página para evitar reenvio do formulário (Padrão PRG)
            header('Location: index.php');
            exit;
```

```
        } catch (AwsException $e) {
            $erro = "Erro ao salvar recado: " . $e->getMessage();
        }
    } else {
        $erro = "Por favor, preencha todos os campos.";
    }
}

// 4. Lógica para buscar e exibir todos os recados (método GET)
try {
    //
================================================================================
    // EXPLICAÇÃO: Operação scan
    // A operação 'scan' lê *todos* os itens de uma tabela. Para fins de aprendizado
    // e em tabelas pequenas, ela é útil.
    // ATENÇÃO: Em produção, para tabelas grandes, 'scan' é ineficiente e pode
    // ser caro, pois consome muitas unidades de leitura. A prática recomendada
    // para buscar itens específicos é usar a operação 'query' com índices.
    //
================================================================================
    $result = $dynamodb->scan([
        'TableName' => $tableName
    ]);
    $recados = $result['Items'];

    //
================================================================================
    // EXPLICAÇÃO: Função usort
    // 'usort' é uma função do PHP que ordena um array usando uma função de
    // comparação definida pelo usuário.
    // Aqui, estamos criando uma função anônima (ou "closure") para comparar
    // dois recados ($a e $b) com base em seu 'timestamp'.
    // 1. 'use ($marshaler)': Disponibiliza a variável $marshaler dentro da função.
    // 2. '$marshaler->unmarshalItem()': Converte os itens do formato DynamoDB
    //    para um array PHP para que possamos acessar 'timestamp' facilmente.
    // 3. '$itemB['timestamp'] <=> $itemA['timestamp']': O "spaceship operator"
    //    compara os dois valores. Retorna -1, 0 ou 1. Ao colocar $itemB primeiro,
    //    garantimos uma ordenação decrescente (os mais recentes primeiro).
    //
================================================================================
```

```
==
    if(!empty($recados)) {
        usort($recados, function($a, $b) use ($marshaler) {
            $itemA = $marshaler->unmarshalItem($a);
            $itemB = $marshaler->unmarshalItem($b);
            return $itemB['timestamp'] <=> $itemA['timestamp'];
        });
    }

} catch (AwsException $e) {
    $erro = "Erro ao buscar recados: " . $e->getMessage();
}

?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mural de Recados com PHP e DynamoDB</title>
    <!-- Usando Bootstrap 5.3 via CDN -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
xintegrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+A
LEwIH" crossorigin="anonymous">
</head>
<body class="bg-light">

    <div class="container py-5" style="max-width: 720px;">

        <header class="text-center mb-5">
            <h1 class="display-4 fw-bold text-primary">Mural de Recados ☁️</h1>
            <p class="text-muted">Demonstração com PHP + DynamoDB (via LocalStack)</p>
        </header>

        <!-- Seção para adicionar novo recado -->
        <div class="card shadow-sm mb-5">
            <div class="card-body p-4">
                <h2 class="card-title h4 mb-4">Deixe seu Recado</h2>
                <?php if ($erro): ?>
                    <div class="alert alert-danger" role="alert">
                        <?php echo $erro; ?>
                    </div>
```

```
            <?php endif; ?>
            <form action="index.php" method="POST">
                <div class="mb-3">
                    <label for="autor" class="form-label">Seu Nome:</label>
                    <input type="text" name="autor" id="autor" class="form-control" required>
                </div>
                <div class="mb-3">
                    <label for="mensagem" class="form-label">Sua Mensagem:</label>
                    <textarea name="mensagem" id="mensagem" rows="4" class="form-control"
required></textarea>
                </div>
                <div class="d-grid gap-2 d-md-flex justify-content-md-end">
                    <button type="submit" class="btn btn-primary">
                        Publicar Recado
                    </button>
                </div>
            </form>
        </div>
    </div>

    <!-- Seção para exibir os recados -->
    <div>
        <h2 class="h4 mb-4 text-center">Recados Publicados</h2>
        <?php if (empty($recados)): ?>
            <p class="text-center text-muted">Nenhum recado ainda. Seja o primeiro!</p>
        <?php else: ?>
            <?php foreach ($recados as $recado):
                // Desempacota o item do formato DynamoDB para um array PHP associativo
                $item = $marshaler->unmarshalItem($recado);
            ?>
                <div class="card shadow-sm mb-3">
                    <div class="card-body">
                        <blockquote class="blockquote mb-0">
                            <p>"<?php echo htmlspecialchars($item['mensagem']); ?>"</p>
                            <footer class="blockquote-footer mt-2">
                                <span class="fw-bold"><?php echo htmlspecialchars($item['autor']);
?></span>
                                em <cite title="Data"><?php echo date('d/m/Y H:i', $item['timestamp']);
?></cite>
                            </footer>
                        </blockquote>
                    </div>
                </div>
```

```
            <?php endforeach; ?>
        <?php endif; ?>
    </div>

</div>

<!-- Opcional: Bootstrap JS Bundle -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
xintegrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jI
eHz" crossorigin="anonymous"></script>
</body>
</html>
```