1. **Introdução**

**Amazon Timestream:**  
O amazon Timestream é um serviço de banco de dados escalável e gerenciado projetado para armazenar e consultar séries temporais de dados.

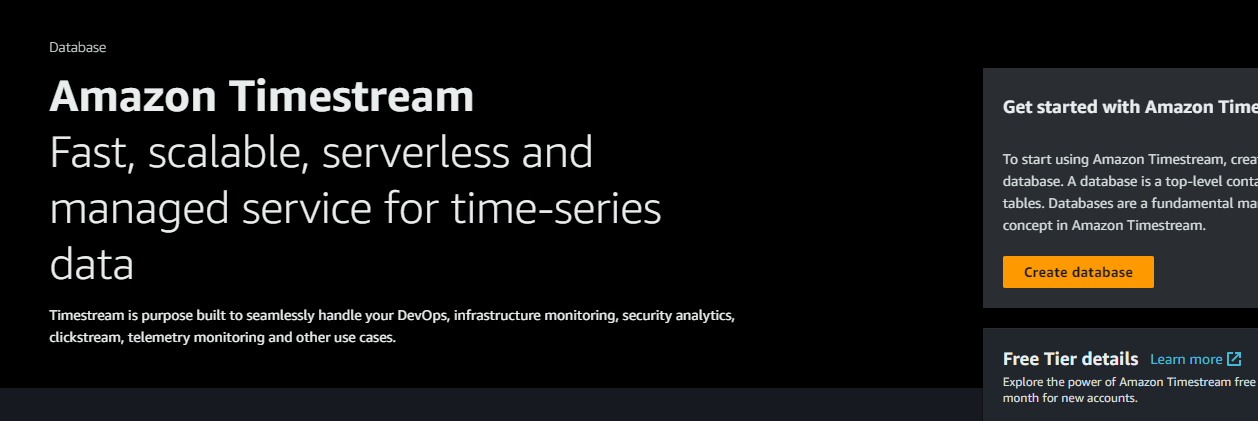
**AWS Common Runtime:**  
O Aws Common Runtime é uma biblioteca que permite que aplicativos se comuniquem de forma segura com serviços AWS usando token de segurança.

**KMS key ARN (Amazon Resource Name):**  
O Amazon Key Management Service (KMS) é um serviço que permite criar e controlar chaves criptografia para proteger seus dados. O KMS Key ARN é um identificador único para uma chave do KMS na AWS. Você pode usar uma chave KMS para criptografar e descriptografar dados no Amazon Timestream.

**Boto3:**  
O Boto3 é uma biblioteca Python desenvolvida pela Amazon para interagir com serviços da AWS, incluindo o Amazon Timestream. Ele fornece uma API que permite criar, gerenciar e interagir com recursos da AWS através de código Python.

**Instalação do Boto3:**  
Pode ser instalado usando o pip, que é o gerenciador de pacotes do Python.  
**Comando:** pip install boto3

1. **Criação do banco de dados na amazon:**



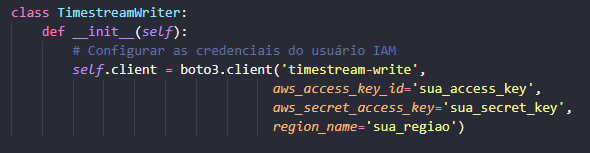
Após pesquisar pelo serviço crie seu database: Create database

1. **Script Python para enviar dados:**

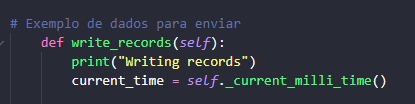
Com base na dodumentação da AWS:  
link: <https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.write.html>

**‘Import boto3’**: Importa o módulo ‘boto3’;  
**‘Import time’**: Importa o módulo ‘time’, que fornece funções relacionadas ao tempo.

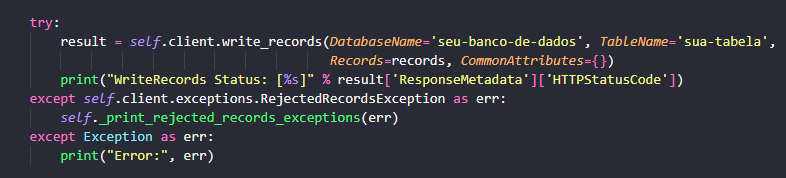
Autentificar as credencias do user:



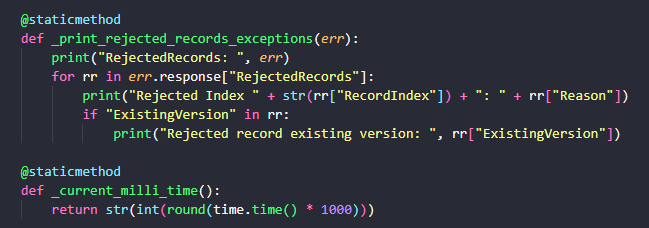
**‘class TimestreamWriter:’**: Define uma classe chamada TimestreamWriter, que contém os métodos para interagir com o Amazon Timestream.  
**‘def \_init\_(self):’**: Define o método de inicialização da classe. Este método é executado automaticamente quando uma instância da classe é criada.  
**‘self.client = boto3.client(...)’**: Cria um cliente para o serviço Amazon Timestream usando as credenciais e região especificadas. Isso é usado para interagir com o serviço.



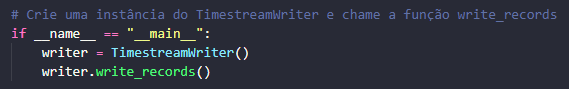
**‘def write\_records(self):’**: Define um método chamado write\_records, que é responsável por escrever registros no Amazon Timestream.  
**‘current\_time = self.\_current\_milli\_time()’**: Obtém o tempo atual em milissegundos usando o método \_current\_milli\_time.



**‘try’**: Inicia um bloco de código para lidar com possíveis exceções.  
**‘result = self.client.write\_records(...)’**: Chama o método write\_records do cliente do Amazon Timestream para gravar os registros no banco de dados e tabela especificados.  
**‘print("WriteRecords Status: [%s]" % result['ResponseMetadata']['HTTPStatusCode'])’**: Imprime o status da gravação de registros.  
**‘except self.client.exceptions.RejectedRecordsException as err:’**: Trata exceções específicas para registros rejeitados.  
**‘except Exception as err:’**: Trata outras exceções gerais.



**‘@staticmethod def \_print\_rejected\_records\_exceptions(err):’**: Define um método estático que imprime informações sobre registros rejeitados em caso de exceção.  
**‘@staticmethod def \_current\_milli\_time():’**: Define um método estático para obter o tempo atual em milissegundos.



**‘if \_\_name\_\_ == "\_\_main\_\_":’**: Verifica se o script está sendo executado como um programa principal.  
**‘writer = TimestreamWriter()’**: Cria uma instância da classe TimestreamWriter.  
**‘writer.write\_records()’**: Chama o método write\_records para gravar os registros no Amazon Timestream.