Cálculo Computacional

Relatório: Billing AWS

Ivan Rangel | 04241013
Felipe Gasparotto | 04241041
Vinicius Miralha | 04241021
Raul Gomes Reis | 04241064
Lucas Alves | 04241005

SÃO PAULO TECH SCHOOL

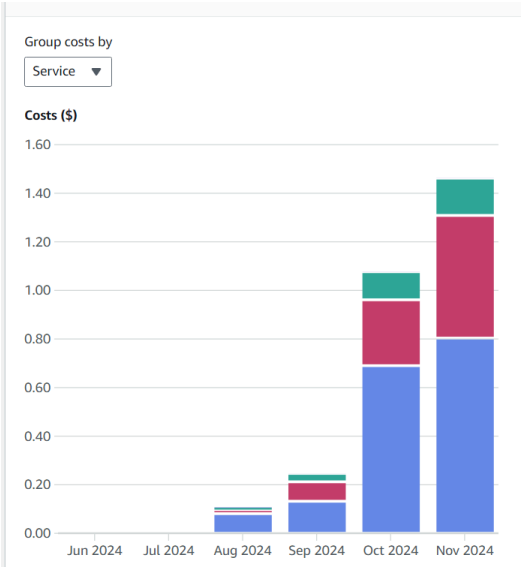
São Paulo Tech School,2024.

Sumário

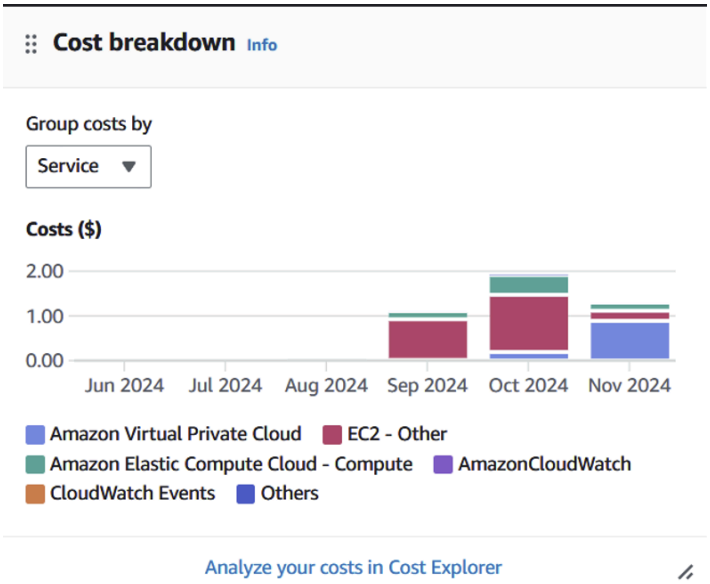
Base de dados

Ao consultar o gerenciamento de cobranças, nos deparamos com os custos da AWS por mês, esses dados vão ser usados para realizar as previsões dos proximos meses, temos dados a partir de agosto, já que essa foi a epoca que iniciamos os estudos sobre sistemas operacionais em nuvem.

Quando analisamos os dados percebemos que de longe, os que mais se destacam são os custos relacionados a instancia EC2, já que estamos atuando no ***ambiente de desenvolvimento***, onde revisamos e atualizamos os códigos diariamente, o contrario do que acontece no ambiente de produção onde o site e o banco de dados não sofrerão atualizações constantes e a nossa Lambda, responsavel pelo tratamento dos dados, funcionaria em tempo integral.



Servidor 1



Servidor 2

Regressão Linear Padrão

Explicação

Antes de usarmos uma regressão linear com séries temporais criamos um modelo preditivo “padrão” para nossos dados e a partir dele, prevemos os gastos do mês de Dezembro:

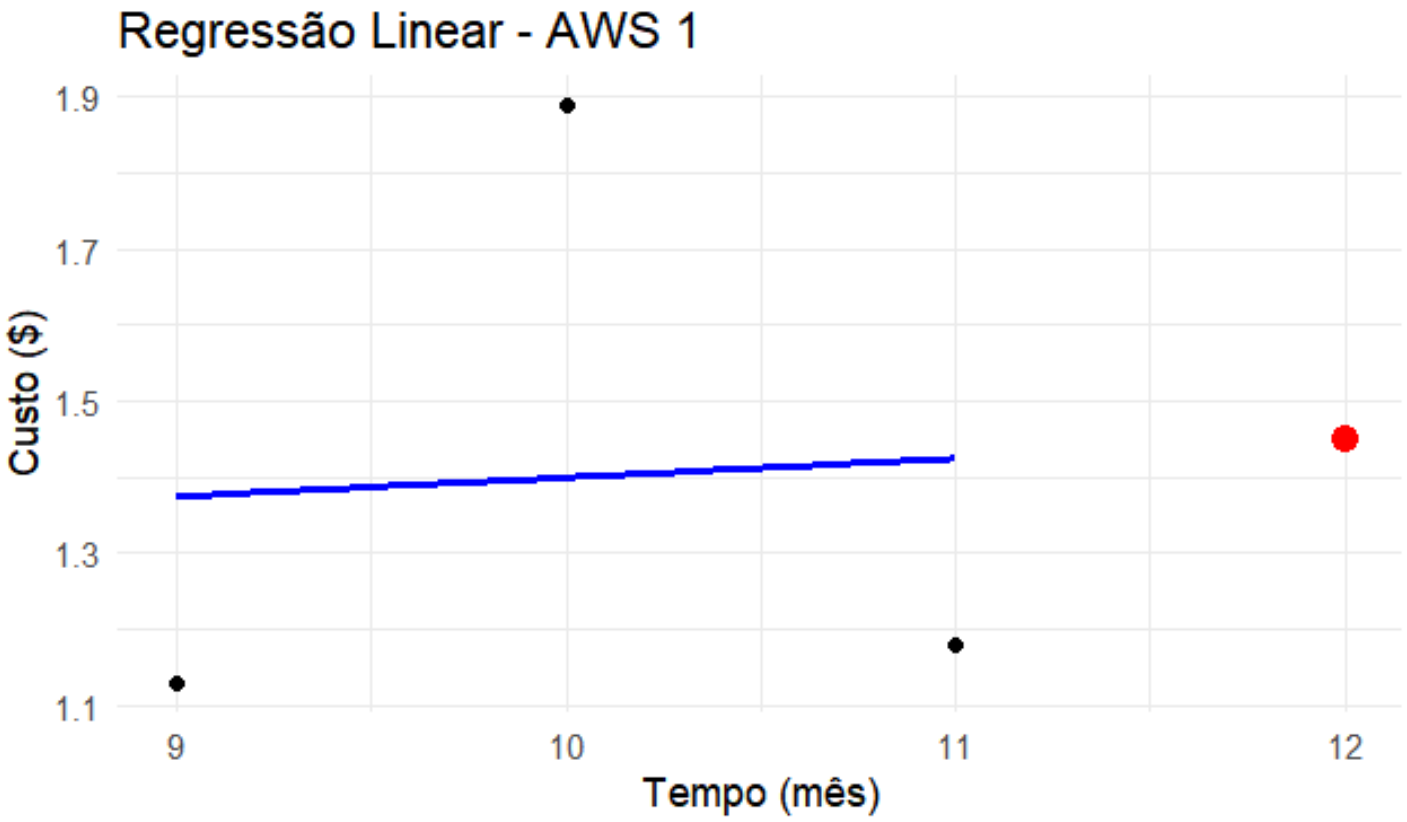
Código

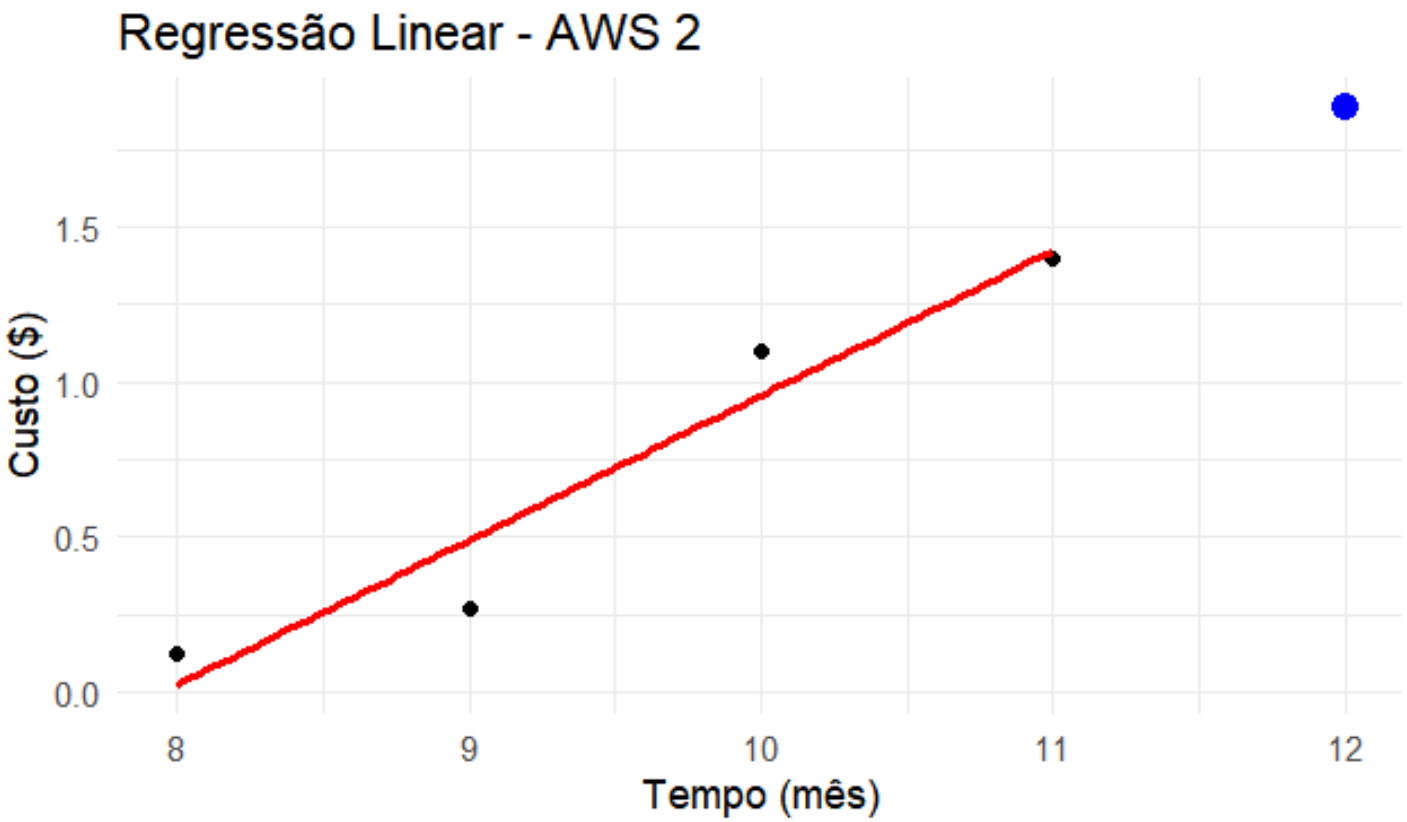
```r
library(ggplot2)

meses_aws1 <- c("Setembro", "Outubro", "Novembro")
custos_aws1 <- c(1.13, 1.89, 1.18)

meses_aws2 <- c("Agosto", "Setembro", "Outubro", "Novembro")
custos_aws2 <- c(0.12, 0.27, 1.10, 1.40)

tempo_aws1 <- c(9, 10, 11)
tempo_aws2 <- c(8, 9, 10, 11)

df_aws1 <- data.frame(Tempo = tempo_aws1, Custo = custos_aws1)
df_aws2 <- data.frame(Tempo = tempo_aws2, Custo = custos_aws2)

modelo_aws1 <- lm(Custo ~ Tempo, data = df_aws1)
modelo_aws2 <- lm(Custo ~ Tempo, data = df_aws2)

tempo_previsao <- data.frame(Tempo = 12)
previsao_aws1 <- predict(modelo_aws1, tempo_previsao)
previsao_aws2 <- predict(modelo_aws2, tempo_previsao)

cat("Previsão AWS 1 (Dezembro):", previsao_aws1, "\n")
cat("Previsão AWS 2 (Dezembro):", previsao_aws2, "\n")
```

Resultados

```
> cat("Previsão AWS 1 (Dezembro):", previsao_aws1, "\n")
Previsão AWS 1 (Dezembro): 1.45
> cat("Previsão AWS 2 (Dezembro):", previsao_aws2, "\n")
Previsão AWS 2 (Dezembro): 1.89
```

Graficos





Conclusão

Por meio desses gráficos conseguimos ver que os dados do segundo servidor tem residuos com valor bem menor e de forma mais constante, o que é comprovado chamando a função residuals:

```
> residuals(modelo_aws1)
     1      2      3
-0.245  0.490 -0.245
> residuals(modelo_aws2)
     1      2      3      4
 0.098 -0.219  0.144 -0.023
>
```

Talvez isso se deva a quantidade de dados de cada modelo ou a menor variação e constancia dos dados, mas para essa simulação de previsão vamos focar nos dados do segundo servidor.

Regressão Linear Com Séries Temporais

Código

A biblioteca forecast é usada para criar modelos com séries temporais

```
library(forecast)
library(tidyverse)

set.seed(123)

servidor1 <- data.frame(
  data = seq(as.Date("2024-06-01"), as.Date("2024-11-01"), by = "month"),
  custo = c(0, 0, 0.15, 0.25, 1.08, 1.45)
)
```

Aqui o modelo pega a coluna que contem os custos e define cada dado como de um "periodo" dentro do total de 12, simbolizando os meses do ano

tslm = Time Series Linear Model

Trend representa o tempo (meses no caso)

forecast tem o mesmo uso que ***predict***

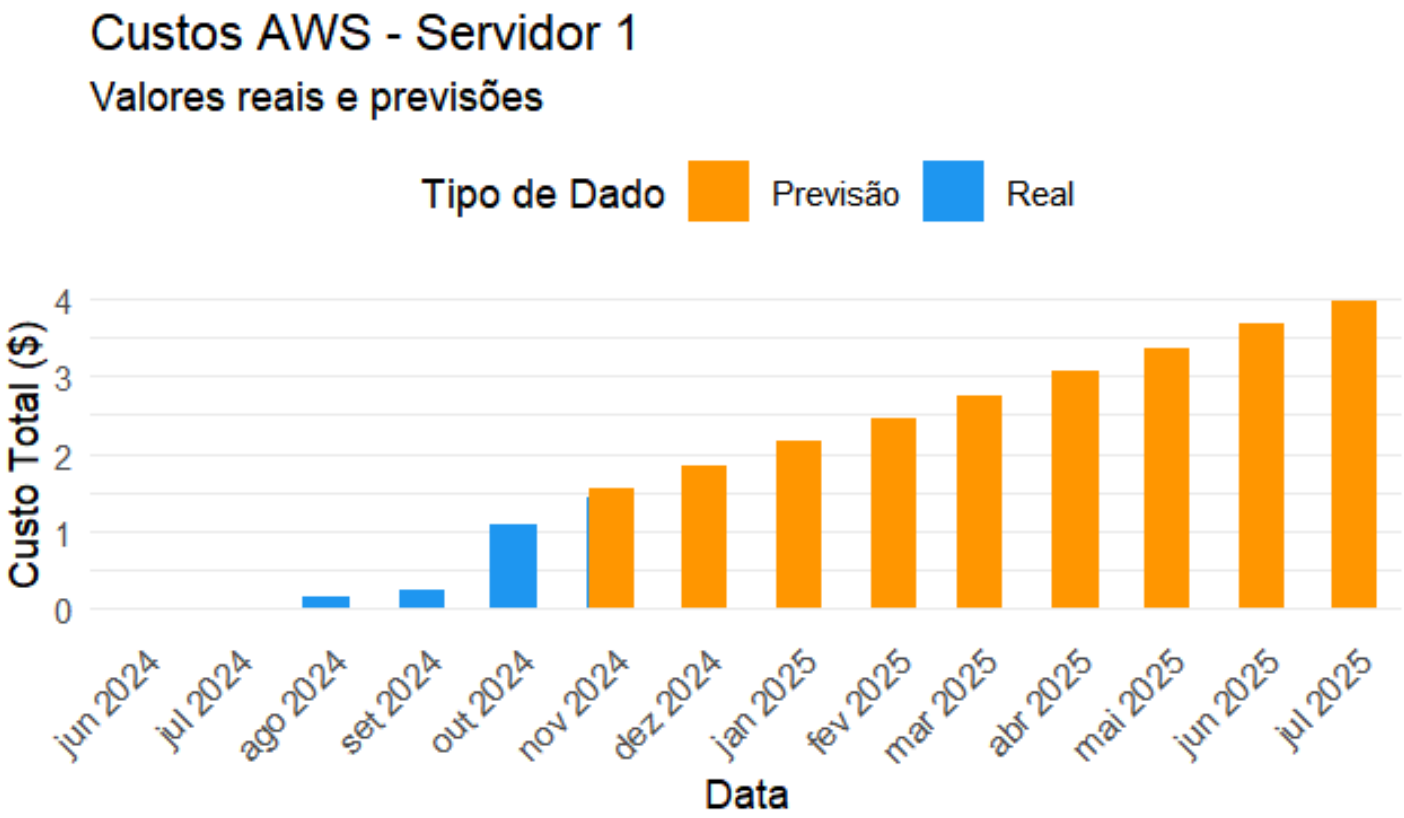
```r
meses_previsao <- 9

ts_data <- ts(servidor1$custo, frequency = 12)


modelo <- tslm(ts_data ~ trend)
modelo


previsao <- forecast(modelo, h = meses_previsao)


previsao


datas_futuras <- seq(max(servidor1$data) + 1, by = "month", length.out = length(previsao$mean))
resultados_df <- data.frame(
  data = datas_futuras,
  custo = as.numeric(previsao$mean),
  tipo = "Previsão"
)
```

Juntando as previsões com os dados reais em um unico dataset

```r
datas_futuras <- seq(max(servidor1$data) + 1, by = "month", length.out = length(previsao$mean))
resultados_df <- data.frame(
  data = datas_futuras,
  custo = as.numeric(previsao$mean),
  tipo = "Previsão"
)

dados_historicos <- servidor1
dados_historicos$tipo <- "Real"

# Combinando dados históricos e previsões
dados_completos <- rbind(
  dados_historicos,
  resultados_df
)

dados_completos$servidor <- "Servidor 1"

cores <- c("Real" = "#2196F3", "Previsão" = "#FF9800")
```

Resultados

O modelo preveu um crescimento excessivo do uso da EC2 devido as grandes alterações que tem sido feitas, chegando a prever 4 dolares por mês em Julho de 2025



Dataset disponibilizado

Explicação

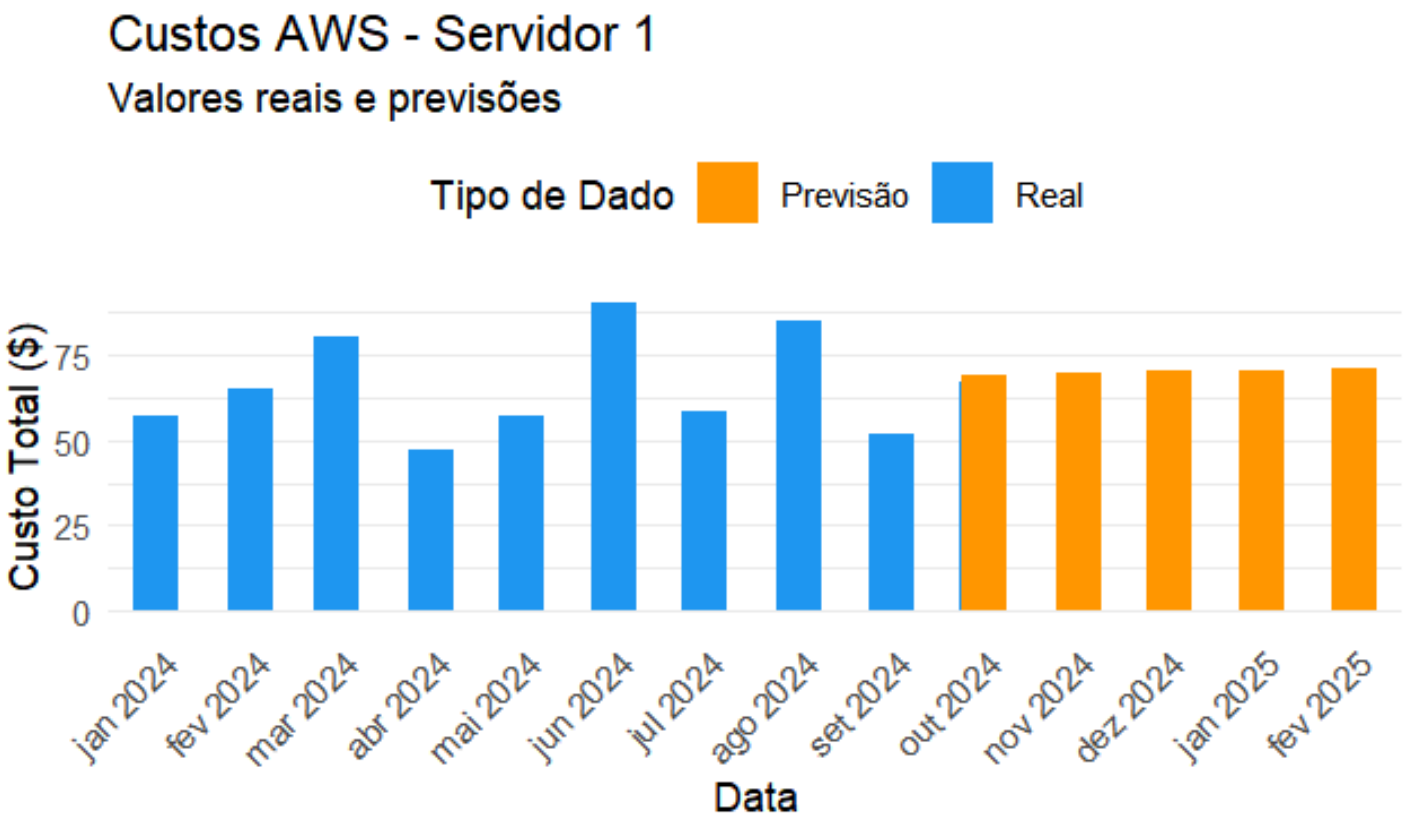
Agora vamos realizar a previsão com série temporal usando o dataset disponibilizado nos Slides

Código

As unicas alterações feitas foram das entradas e da quantidade de meses previstos

```r
library(forecast)
library(tidyverse)
set.seed(123)

df <- data.frame(
  data = seq(as.Date("2024-01-01"), as.Date("2024-10-01"), by ="month"),
  custo = c(57, 65, 80, 47, 57, 90, 58, 85, 52, 67)
)


meses_previsao <- 5


ts_data <- ts(df$custo, frequency = 12)
```

Resultado



Conclusão

O resultado dessa regressão com série temporal é uma inclinação muito mais leve, de 0.54, como podemos ver ao dar um summary:

```
Call:
tslm(formula = ts_data ~ trend)

Residuals:
    Min      1Q  Median      3Q     Max
-17.982  -8.595  -3.800  11.950  23.927

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  62.8000    10.5085   5.976 0.000332 ***
trend         0.5455     1.6936   0.322 0.755655
```

Previsão de uso da Lambda e dos Buckets

Explicação

Dentro da nossa regra de negócio, conseguimos calcular exatamente qual vai ser o uso de nossa estrutura de armazenamento na AWS, e quantas vezes os gatilhos da Lambda vão ser usados, assim conseguimos prever qual será o preço recorrente do nosso projeto por servidor, embora não seja necessário usar algum tipo de regressão para realizar os calculos.

Código da Lambda:

```r
# Calculando preço da LAMBDA:

# Roda a cada 10 segundos, ou seja, 360 vezes por hora. Um mes tem aproximadamente 720 horas
total_execucoes = 360 * 720
# Cada csv tem 435 Bytes
tamanho_total_dados = 435 * total_execucoes
# arredondando para mb
tamanho_total_dados <- round(tamanho_total_dados / (1024^2), 2)

# A LAMBDA cobra por uso de RAM por segundo (GB-segundo)

#informações do cloudWatch
memoria_alocada = 0.5 #0.5 GB ou seja 512 mb
duracao_media = 0.2 #segundos

duracao_gb_segundos = total_execucoes * duracao_media * memoria_alocada
duracao_gb_segundos

preco_gb_segundo = 0.0000166667

preco_lambda = duracao_gb_segundos * preco_gb_segundo
preco_lambda
```

Preço por servidor:

```
> preco_lambda
```

```
[1] 0.4320009
```

Código dos Buckets:

```r
#custos aws:
custo_leitura_s3 = 0.0004 # por 1000
custo_gravacao_s3 = 0.005 # por 1000
custo_leitura_total =  (total_execucoes / 1000)  * custo_leitura_s3
custo_gravacao_total = (total_execucoes / 1000)  * custo_gravacao_s3

custo_leitura_total
custo_gravacao_total

custo_s3 = custo_gravacao_total + custo_leitura_total
custo_s3

custo_total = custo_s3 + preco_lambda
custo_total
```

Preço por servidor:

```
> custo_s3
[1] 1.39968
```

Custo total do armazenamento por servidor:

USD = 1.831681

BRL = 10,57