# IBM Cloud Functions com R

Thiago Pires (IBM)

## Introdução

A computação sem servidor é um recurso útil para executar na nuvem. Ele combina economia financeira, tempo reduzido de gerenciamento, facilidade na configuração e implementação. Isso significa que um cientista de dados pode trabalhar com mais rapidez ao construir um pipeline de aprendizado de máquina e ainda fornecer um serviço de previsão. A plataforma Function-as-a-Service (FaaS) na IBM Cloud é um recurso para executar código sob demanda com solicitações de API baseadas em HTTP. O IBM Cloud Functions é baseado no projeto de código aberto Apache OpenWhisk.

Hoje é possível com qualquer linguagem de programação criar IBM Cloud Functions. Para algumas linguagens, como Python, Node, Go e outras, a plataforma oferece um tempo de execução padrão e a criação da função sem servidor com menos etapas. Muitos cientistas de dados acreditam que não é possível desenvolver ou pode ser bastante complexo para linguagens de programação sem esse suporte na IBM Cloud. Este artigo mostrará como é possível criar uma função IBM Cloud com a linguagem R.

## Predição usando Tidymodels

Foi ajustado um modelo com o banco de dados do Titanic usando a biblioteca tidymodels para, mais tarde, ser capaz de implementar o modelo no IBM Cloud Function.

Uma regressão logística foi utilizada para classificar os passageiros em sobreviveram ou não sobreviveram ao desastre do Titanic. As variáveis selecionadas foram Sexo e Pclass. Os detalhes sobre a preparação de dados para obter train_data são omitidos neste artigo.

```
library(tidymodels)
library(magrittr)lr_mod <- logistic_reg() %>% set_engine("glm")
lr_fit <- lr_mod %>% fit(Survived ~ Sex + Pclass, data = train_data)
```

Após o ajuste, foi salvo o modelo como arquivo yaml no diretório local. Aqui é necessário usar o pacote yaml e o tidypredict para parse em um arquivo yaml.

```
yaml::write_yaml(tidypredict::parse_model(lr_fit),"R/my_model.yml")
```

### Configuração e Deploy

Para executar uma função com uma linguagem que não é suportada pelo IBM Cloud Functions, você precisará configurar um arquivo exec. Na nuvem, a função será executada em um contêiner em Docker cuja imagem é openwhisk/dockerskeleton. O apk é um gerenciamento de pacote Alpine Linux, então você pode adicionar dependências do Linux em seu contêiner. Neste projeto, além das dependências R e do sistema, também é necessário um pacote para manipular com json (por exemplo, jsonlite). Porque, na função, tanto a entrada quanto a saída devem estar neste formato. A estrutura principal do exec é: instalar as dependências do sistema e pacotes R, obter entrada e salvar como arquivo json e, finalmente, executar um script R como executável.

Precisa configurar um arquivo exec como esse:

```
#!/bin/bash# run R script
chmod +x script.R # turn executable
```

```
echo "$@" > input.json # set input
./script.R # run script
```

Um Dockerfile assim:

```
FROM openwhisk/dockerskeleton
RUN apk update && apk add R R-dev R-doc build-base
RUN R -e "install.packages(c('jsonlite', 'tidypredict', 'yaml'),
repos = 'http://cran.rstudio.com/')"
```

Por último é necessário um script.R para carregar o modelo e calcular as previsões usando tidymodels:

```
#!/usr/bin/env Rscript

# carregar modelo
loaded_model <-tidypredict::as_parsed_model(yaml::read_yaml("my_model.yml"))
# input
input <- jsonlite::fromJSON("input.json", flatten = FALSE)
# calcular predição
pred <- tidypredict::tidypredict_to_column(as.data.frame(input), loaded_model)
# output
jsonlite::stream_out(pred, verbose = FALSE)
```

Para deploy do modelo são necessários os seguintes passos:

- build e push para o Docker Hub.

```
docker build th1460/titanic .
docker push th1460/titanic
```

- logar na IBM Cloud e empacotar os arquivos exec, script.R e my_model.yml.

```
ibmcloud login
ibmcloud target --cf
zip -r titanic.zip exec script.R my_model.yml
```

- Crie a função declarando –web true para transformá-la em uma API.

```
ibmcloud fn action create titanic titanic.zip --docker th1460/titanic --web true
```

Para solicitar uma previsão de API, você pode fazer um POST usando curl ou a função do pacote httr. O e podem ser encontrados com ibmcloud fn action get –url

```
input <- list(Sex = "male", Pclass = "3rd")
"https://<APIHOST>/api/v1/web/<NAMESPACE>/default/titanic.json" %>%
httr::POST(., body = input, encode = "json") %>% httr::content() %>%
.[c("Sex", "Pclass", "fit")] %>% jsonlite::toJSON(pretty = TRUE, auto_unbox = TRUE)
```

**Resultados**

Após a requisição, a saída mostra os parâmetros (Sex e Pclass) e a probabilidade de sobreviver no desastre do Titanic (fit). Neste exemplo, a solicitação leva 964 ms para obter os resultados.

```
{"Sex": "male", "Pclass": "3rd", "fit": 0.0979}
```

## Conclusão

A IBM Cloud Funcion é um recurso incrível para qualquer linguagem de programação. É fácil de configurar, implantar e escalar. Se este recurso corresponder às necessidades do seu projeto (disponibilidade, frequência de solicitações, etc). Deve ser uma escolha interessante executar uma função sem servidor com R em produção.