
inloco

Desenvolvimento de aplicações web com R Shiny

Como escalar produtos de dados de forma rápida e confiável

Gabriel Teotonio

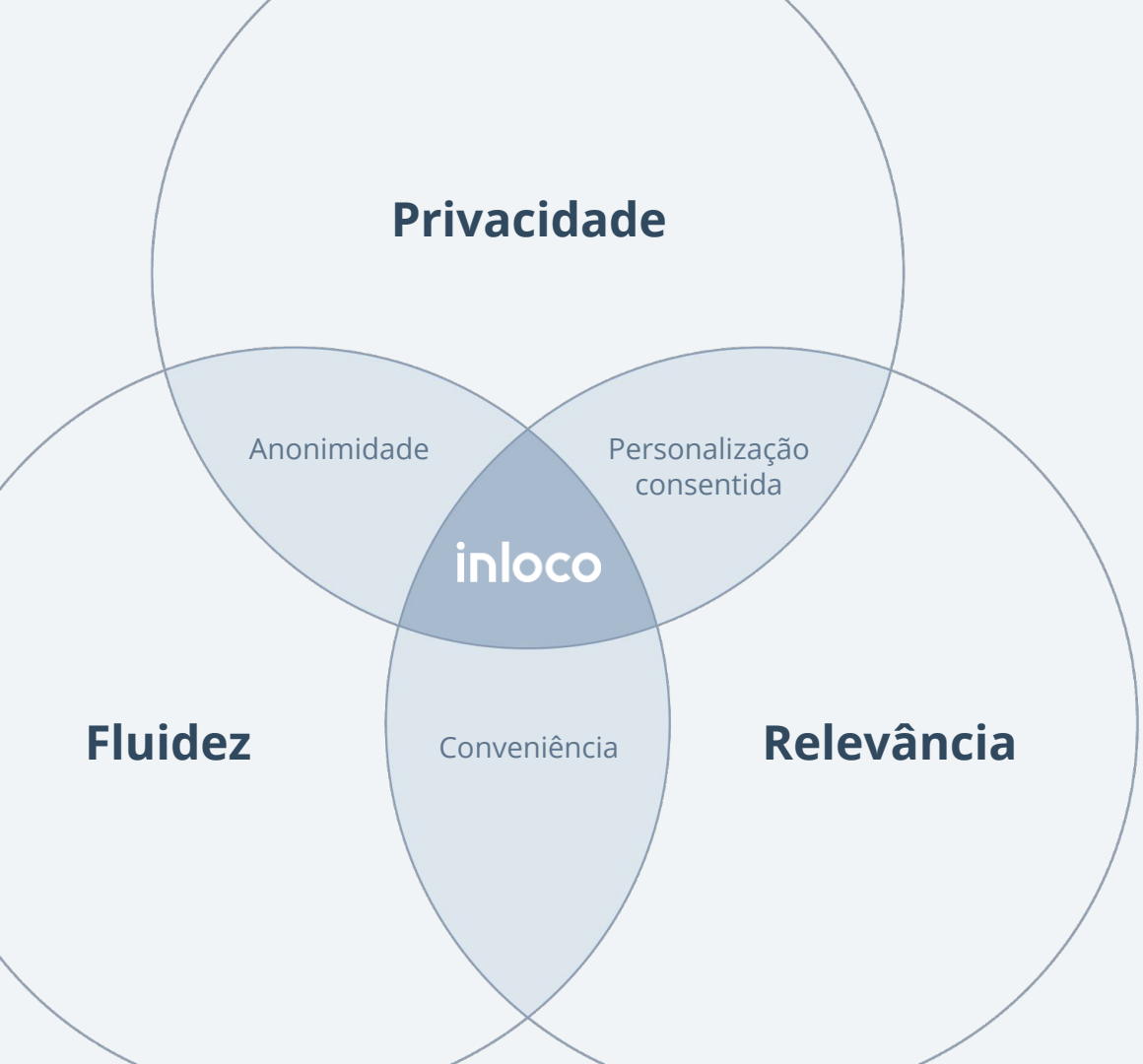
Estatística
Universidade Federal de Pernambuco.

Cientista de Dados.

> 60M
de smartphones no Brasil

> 16TB
de dados processados diariamente

> 28M
de lugares mapeados



Nosso propósito

Entregar conveniência para as pessoas garantindo sua privacidade.

Sessão

1.

O que é Shiny?

Conhecendo o framework de R que nos ajuda a criar aplicações

2.

Lidando com banco de dados

Entendo a comunicação com bancos e boas práticas

3.

Testando sua aplicação

Buscando gargalos e melhorando a performance

4.

Deploy

Como disponibilizar sua aplicação de forma segura

inloco OOH planner

Flow of Passers

Data from 01 to 11 January 2019

Filter

Player

All

Placement points

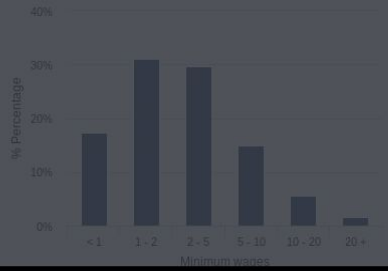
All

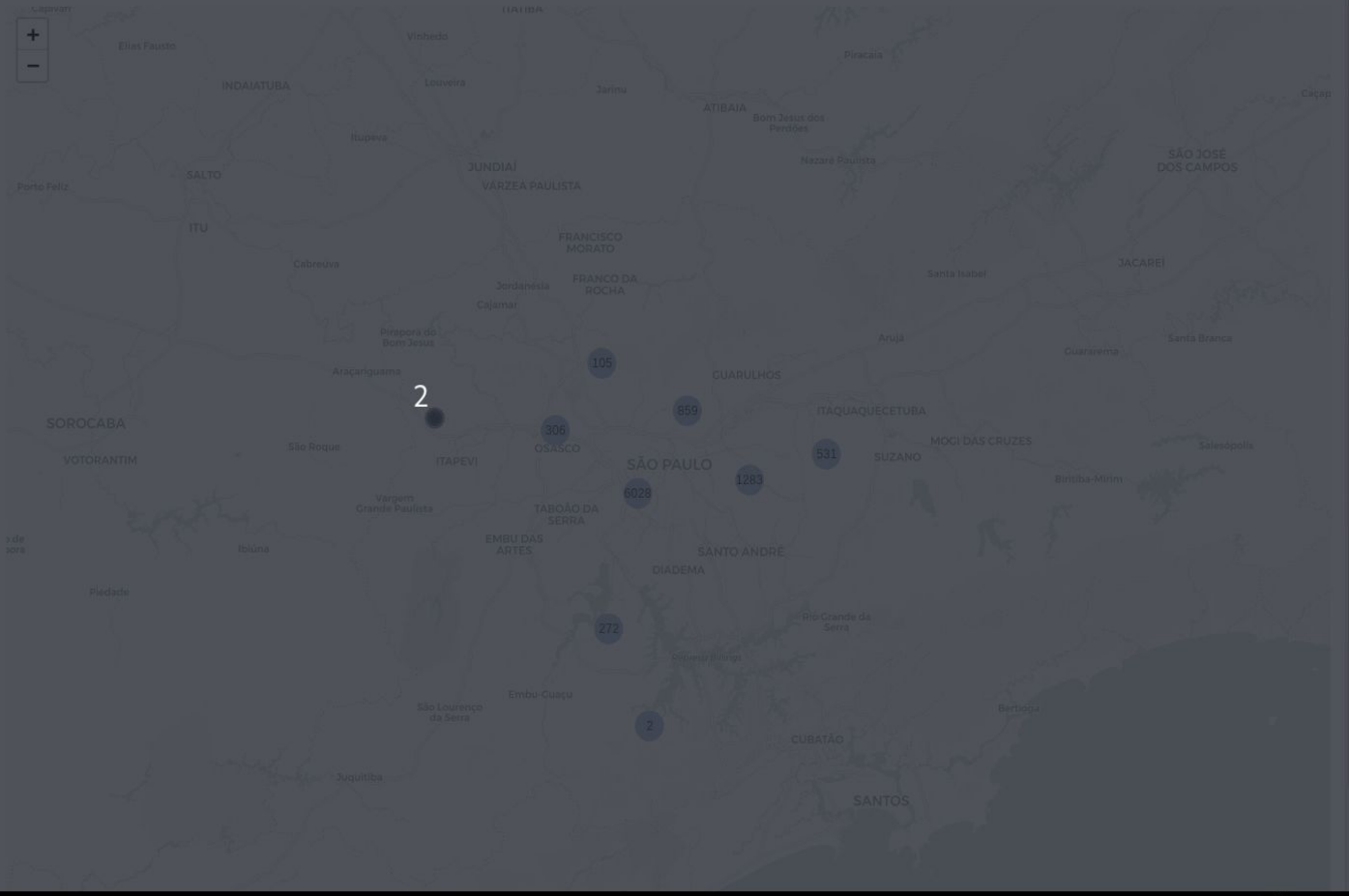
Audience

General flow per point

Gender

Income



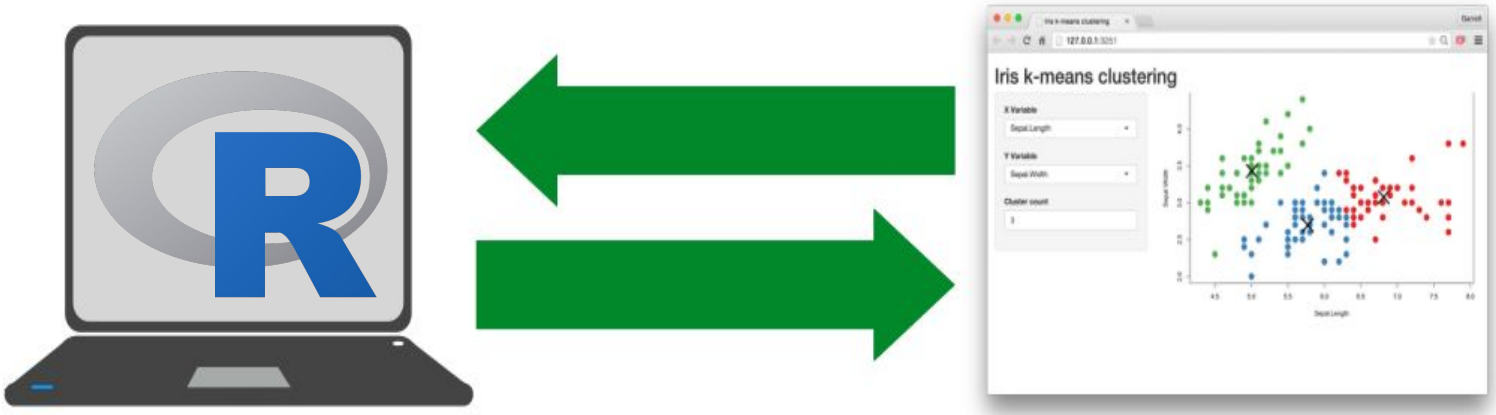


Activities Oct 8 17:04 pt 100 %

O que é Shiny?

Um pacote em R para desenvolvimento de aplicações web.

Todo aplicativo Shiny possui uma página da web que o usuário visita e, por trás dessa página, há um computador que serve essa página executando R.



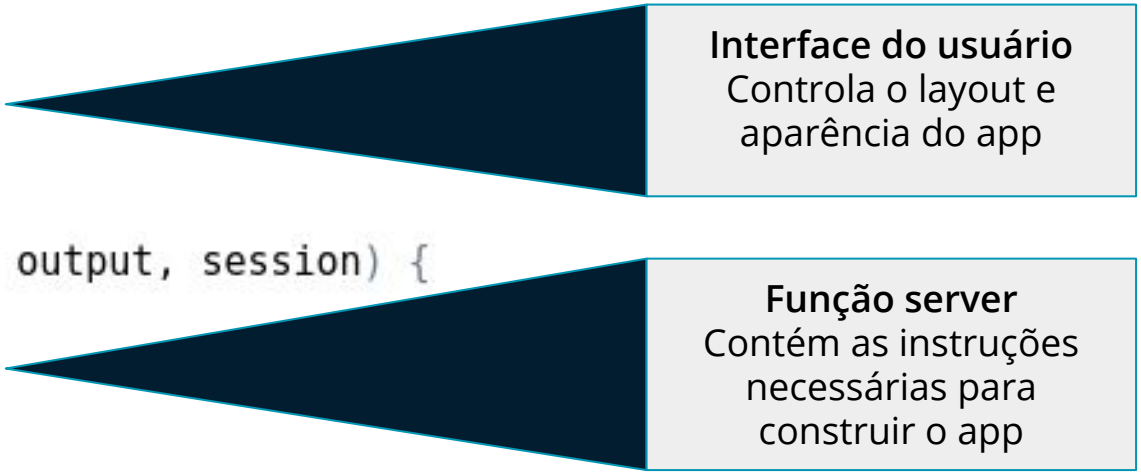

R
Iris k-means clustering

Instruções do Server

Interface do usuário

Anatomia de uma aplicação Shiny

```
library(shiny)

ui <- fluidPage(

)

server <- function(input, output, session) {

}

shinyApp(ui, server)
```



Iris k-means clustering

X Variable

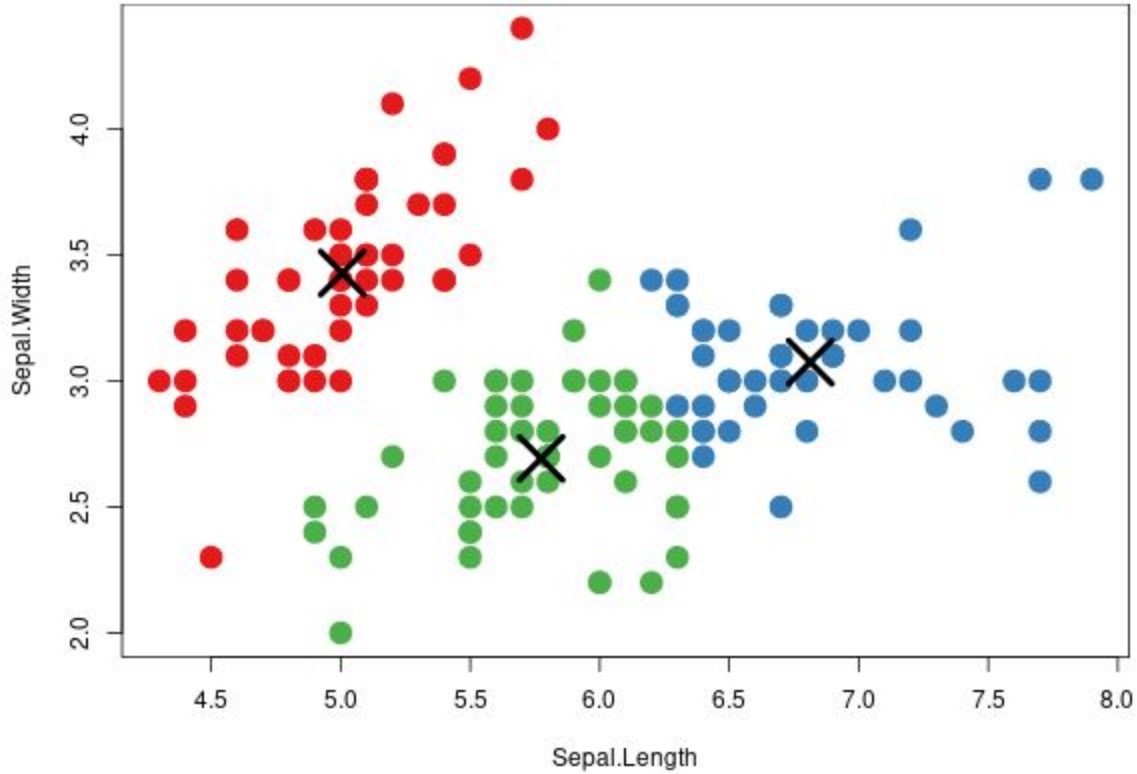
Sepal.Length

Y Variable

Sepal.Width

Cluster count

3

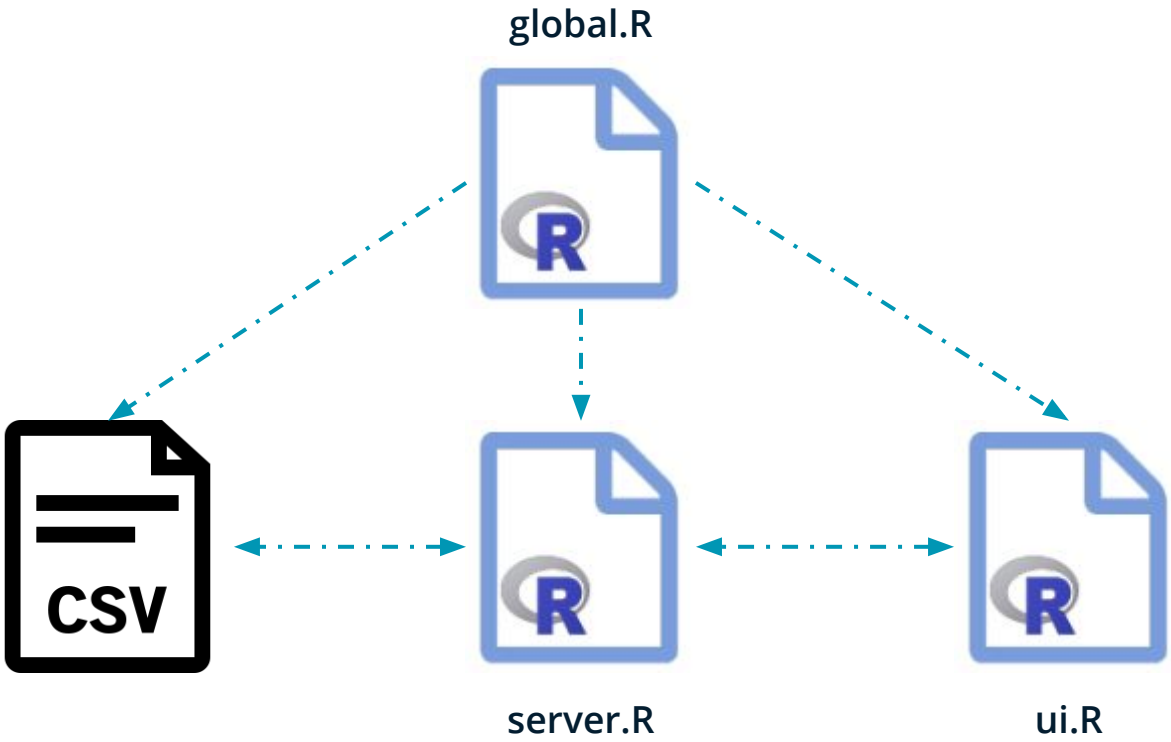


server.R

```
function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
              "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

}
```

ui.R

```
pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
    selectInput('xcol', 'X Variable', names(iris)),
    selectInput('ycol', 'Y Variable', names(iris),
                selected=names(iris)[[2]]),
    numericInput('clusters', 'Cluster count', 3,
                 min = 1, max = 9)
  ),
  mainPanel(
    plotOutput('plot1')
  )
)
```


global.R
CSV
server.R
ui.R

Lidando com banco de dados

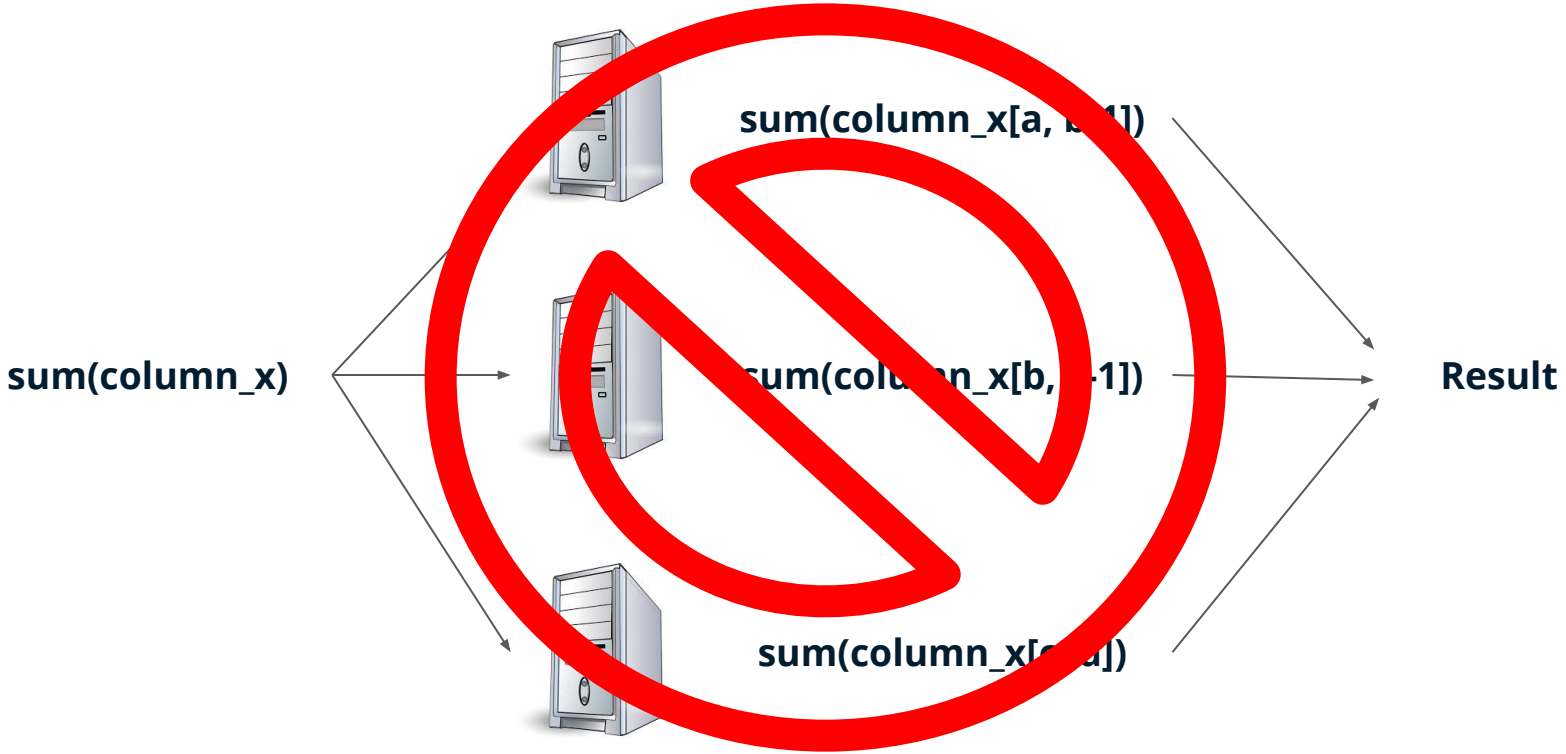
Mas antes...

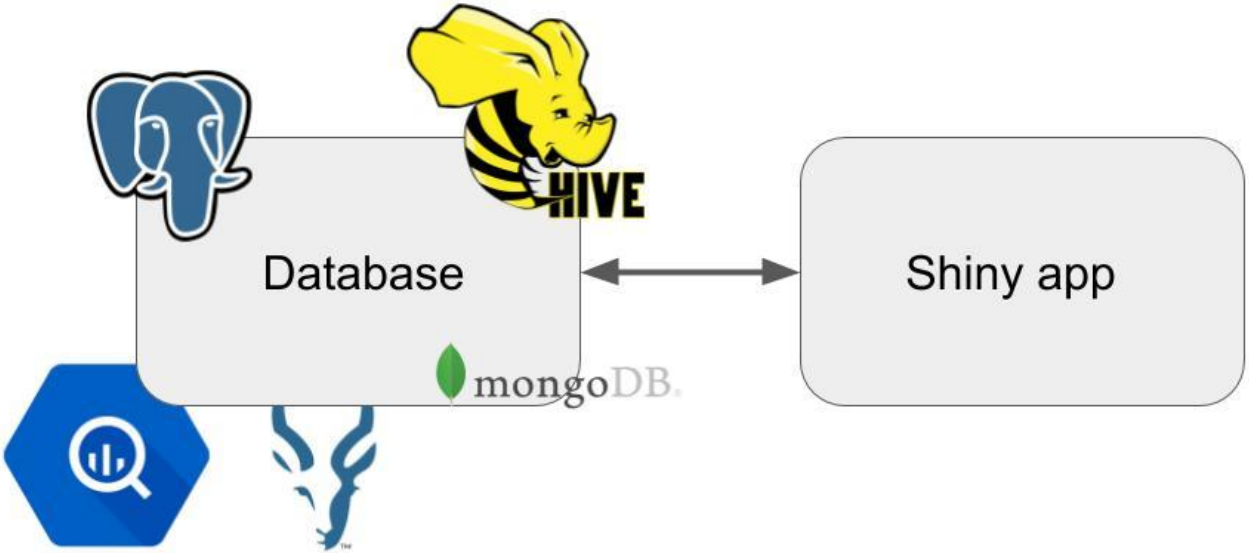
Esses formatos compactam seu arquivo de maneira que você tenha um alto desempenho de leitura dentro do aplicativo:

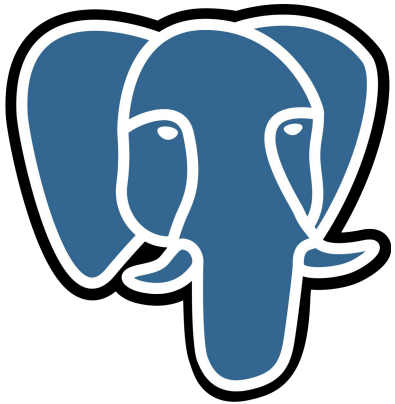
- .feather
- .RDS

Conectando-se a um grande volume de dados.

R é single-threaded




HIVE
Database
Shiny app
mongoDB.



Consulta

Dados processados

Podemos continuar usando o maravilhoso pacote *dplyr* para realizar as consultas ao banco de dados, a partir da versão *dbplyr*.


```
library(dplyr)

con <- DBI::dbConnect(RPostgreSQL::PostgreSQL(),
                      host = "database.inloco.com",
                      user = "insights",
                      password = rstudioapi::askForPassword("Database password")
)
```

```
tbl(con, "flights") %>% select(year:day, dep_delay, arr_delay)
#> # Source:   lazy query [?? x 5]
#> # Database: sqlite 3.22.0 []
#>    year month   day dep_delay arr_delay
#>   <int> <int> <int>     <dbl>     <dbl>
#> 1  2013     1     1         2        11
#> 2  2013     1     1         4        20
#> 3  2013     1     1         2        33
#> 4  2013     1     1        -1       -18
#> 5  2013     1     1        -6       -25
#> 6  2013     1     1        -4        12
#> # … with more rows
```

```
tbl(con, "flights") %>%
  group_by(dest) %>%
  summarise(delay = mean(dep_time))
#> Warning: Missing values are always removed in SQL.
#> Use `mean(x, na.rm = TRUE)` to silence this warning
#> This warning is displayed only once per session.
#> # Source:   lazy query [?? x 2]
#> # Database: sqlite 3.22.0 []
#>   dest  delay
#>   <chr> <dbl>
#> 1 ABQ   2006.
#> 2 ACK   1033.
#> 3 ALB   1627.
#> 4 ANC   1635.
#> 5 ATL   1293.
#> 6 AUS   1521.
#> # … with more rows
```

Pontos importantes

- Ao trabalhar com bancos de dados, o dplyr tenta ser o mais *lazy* possível;
- Podemos usar o comando *collect()* para retornar a tabela resultante para o ambiente local;
- O comando *pull()* também é útil quando a consulta retorna um vetor, e não uma tibble.

```
tbl(con, "flights") %>%
  group_by(tailnum) %>%
  summarise(
    delay = mean(arr_delay),
    n = n()
  ) %>%
  arrange(desc(delay)) %>%
  filter(n > 100) %>%
  show_query()
#> <SQL>
#> SELECT *
#> FROM (SELECT *
#> FROM (SELECT `tailnum`, AVG(`arr_delay`) AS `delay`, COUNT() AS `n`
#> FROM `flights`
#> GROUP BY `tailnum`)
#> ORDER BY `delay` DESC)
#> WHERE (`n` > 100.0)
```

Cuidando das credenciais

- Criptografar credenciais com o pacote *keyring*;
- Use o arquivo de configuração;
- Variáveis de ambiente usando o arquivo .Renviron;
- Usando as opções de comando do R base;
- Solicite credenciais usando o RStudio IDE.

Testando sua aplicação

Santander
Pirelli

shinyloadtest

profvis

shinyloadtest

O processo para testar a carga de um aplicativo Shiny consiste em três etapas:

1. Grave uma sessão típica do usuário para o aplicativo;

2. Repita a sessão em paralelo, simulando muitos; usuários simultâneos acessando o aplicativo;

3. Analise os resultados do teste de carga e determine se o aplicativo teve bom desempenho.

1.

```
library(shinyloadtest)

shinyloadtest::record_session('https://shinyapp.example.com/')

shinyloadtest::record_session('http://127.0.0.1:3509')
```

2.

```
[I] → shinycannon recording.log http://127.0.0.1:3509/ --workers 20 --loaded-duration-minutes 5
2019-10-05 14:45:06.686 INFO [thread00] - Detected target application type: R/Shiny
2019-10-05 14:45:06.688 INFO [progress] - Running: 0, Failed: 0, Done: 0
2019-10-05 14:45:06.689 INFO [thread01] - Warming up
2019-10-05 14:45:06.699 INFO [thread00] - Waiting for warmup to complete
2019-10-05 14:45:07.774 INFO [thread02] - Warming up
2019-10-05 14:45:08.859 INFO [thread03] - Warming up
2019-10-05 14:45:09.945 INFO [thread04] - Warming up
2019-10-05 14:45:11.030 INFO [thread05] - Warming up
2019-10-05 14:45:11.689 INFO [progress] - Running: 5, Failed: 0, Done: 0
2019-10-05 14:45:12.115 INFO [thread06] - Warming up
```

3.

```
df <- shinyloadtest::load_runs("report" = "./test-logs-2019-10-05T17_45_05.622Z")
shinyloadtest::shinyloadtest_report(df, "report.html")
```

Voltando...

Iris k-means clustering

X Variable

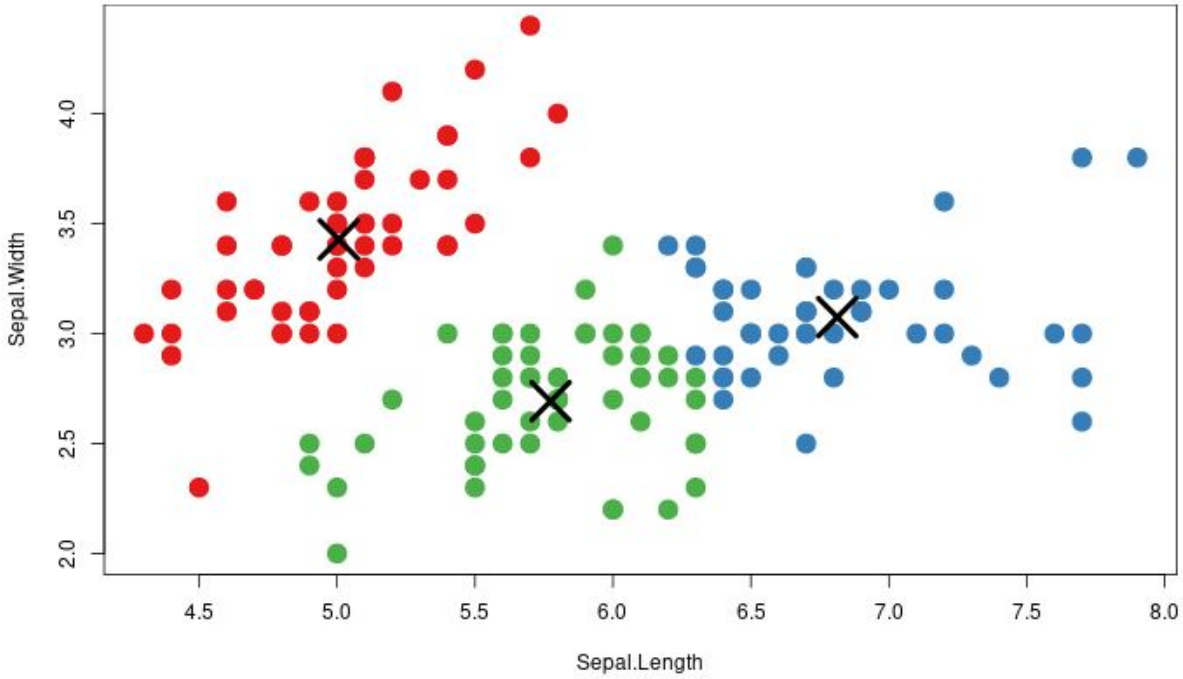
Sepal.Length

Y Variable

Sepal.Width

Cluster count

3



- Dados em um .csv;
- Lendo os dados sempre que for realizar um filtro.

Cenário 1

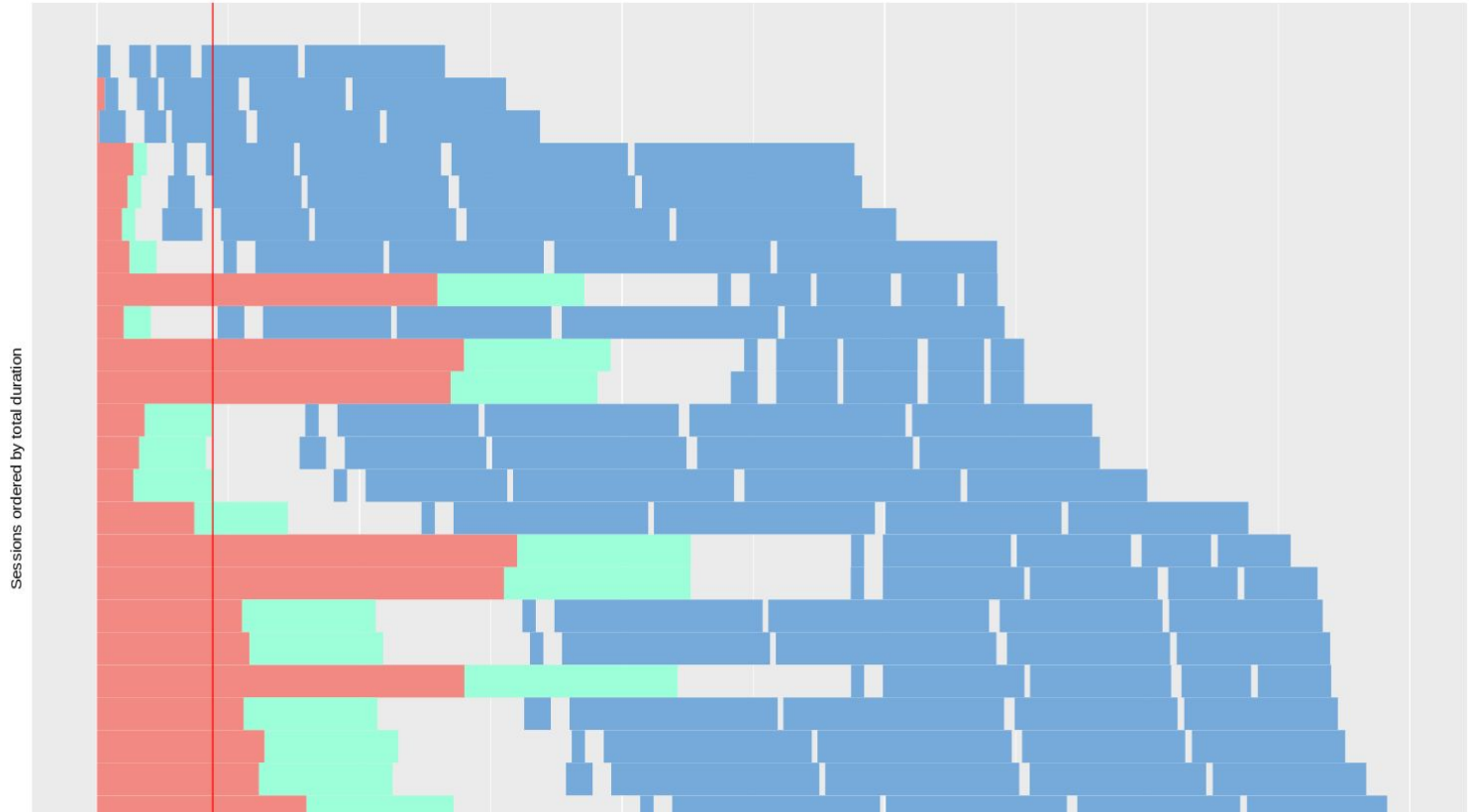
- Dados em um .feather;
- Múltiplos arquivos para cada combinação

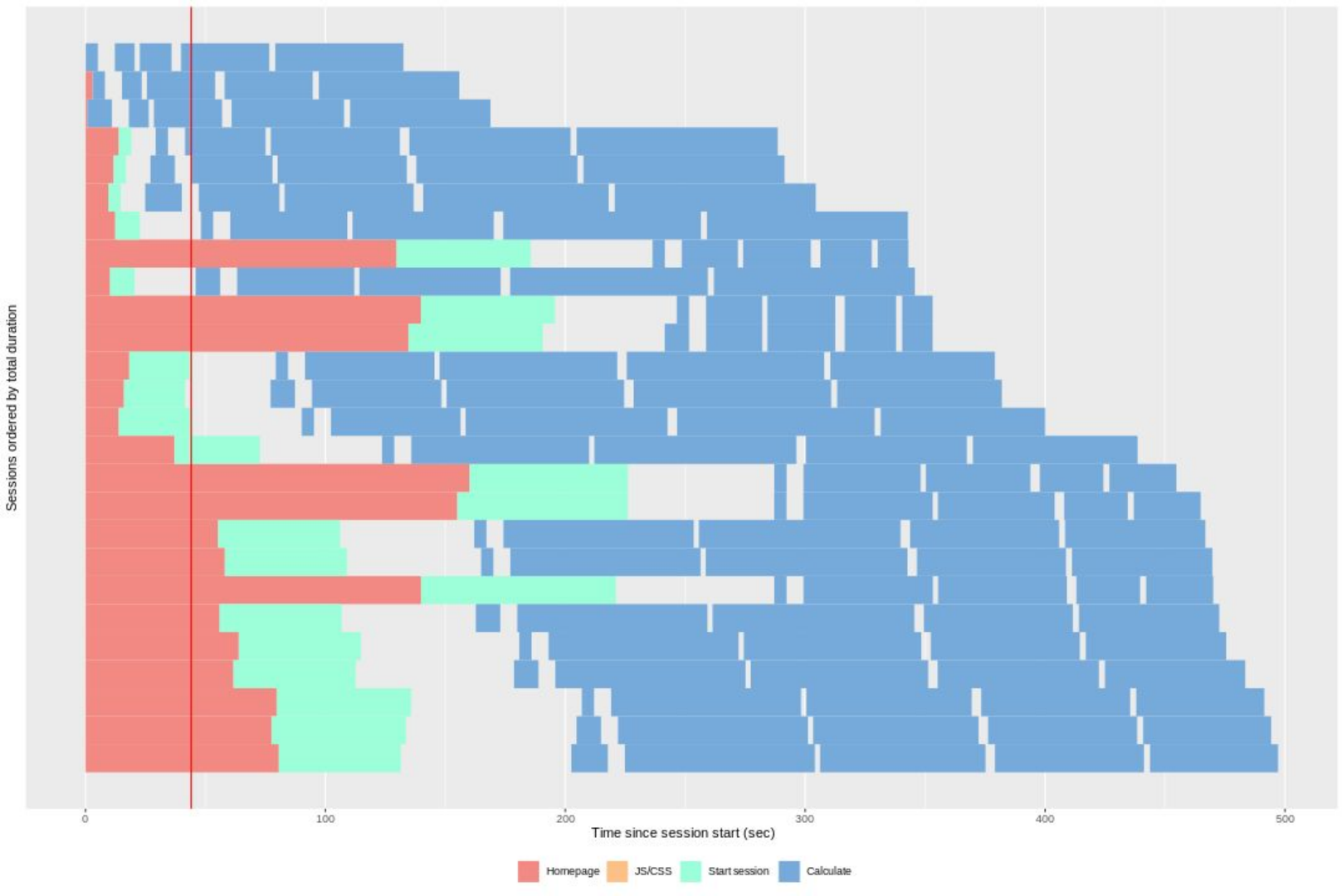
Cenário 2

shinyloadtest

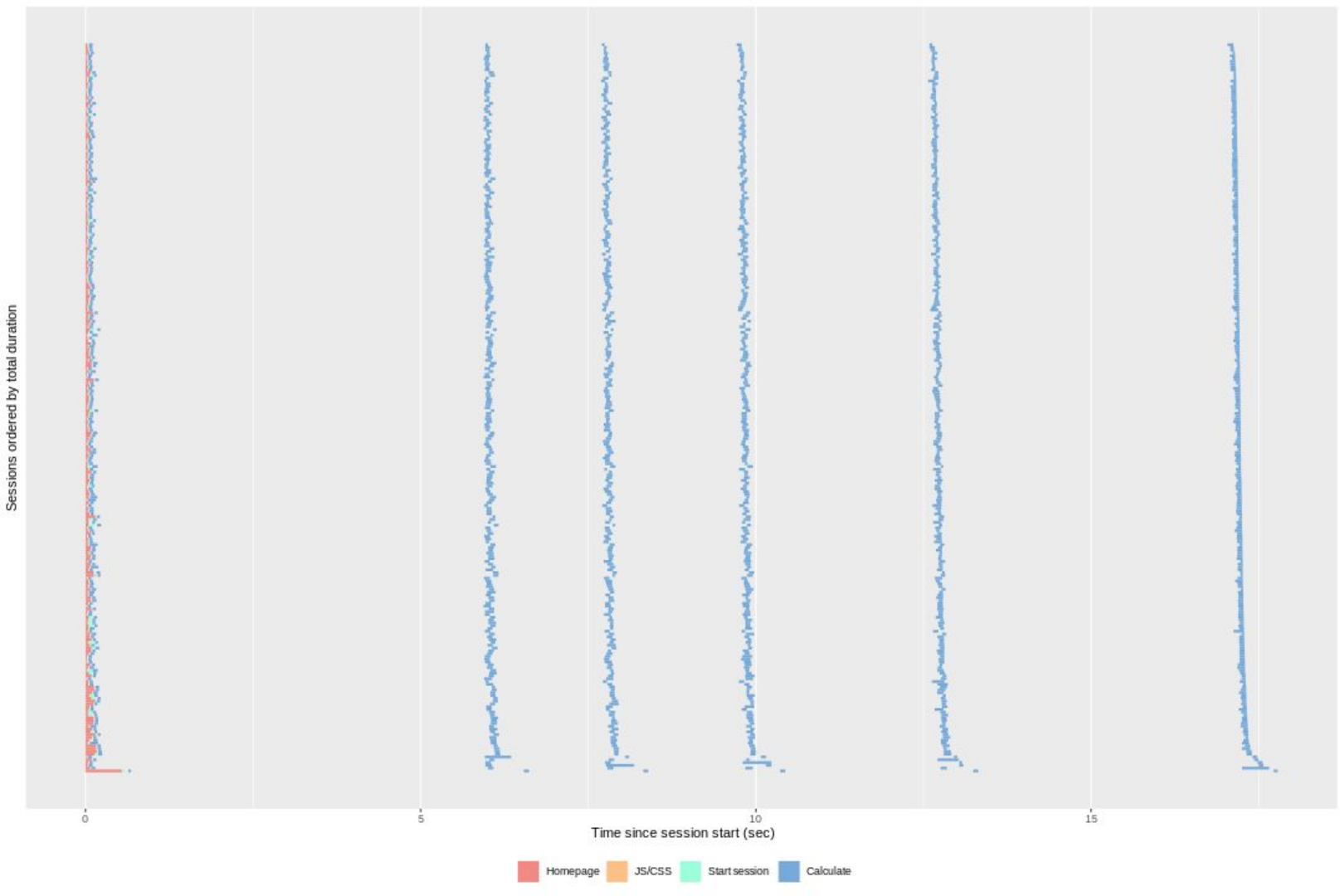
20 workers

26 maintenance sessions ordered from fastest to slowest completion time. The red line marks how long the original recording session took to complete (~44s). Sessions should end around the same time as each other for consistent behavior.




Sessions ordered by total duration
0
100
200
300
400
500
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate

Cenário 1


Sessions ordered by total duration
0
5
10
15
Time since session start (sec)
Homepage
JS/CSS
Start session
Calculate

Cenário 2

profvis

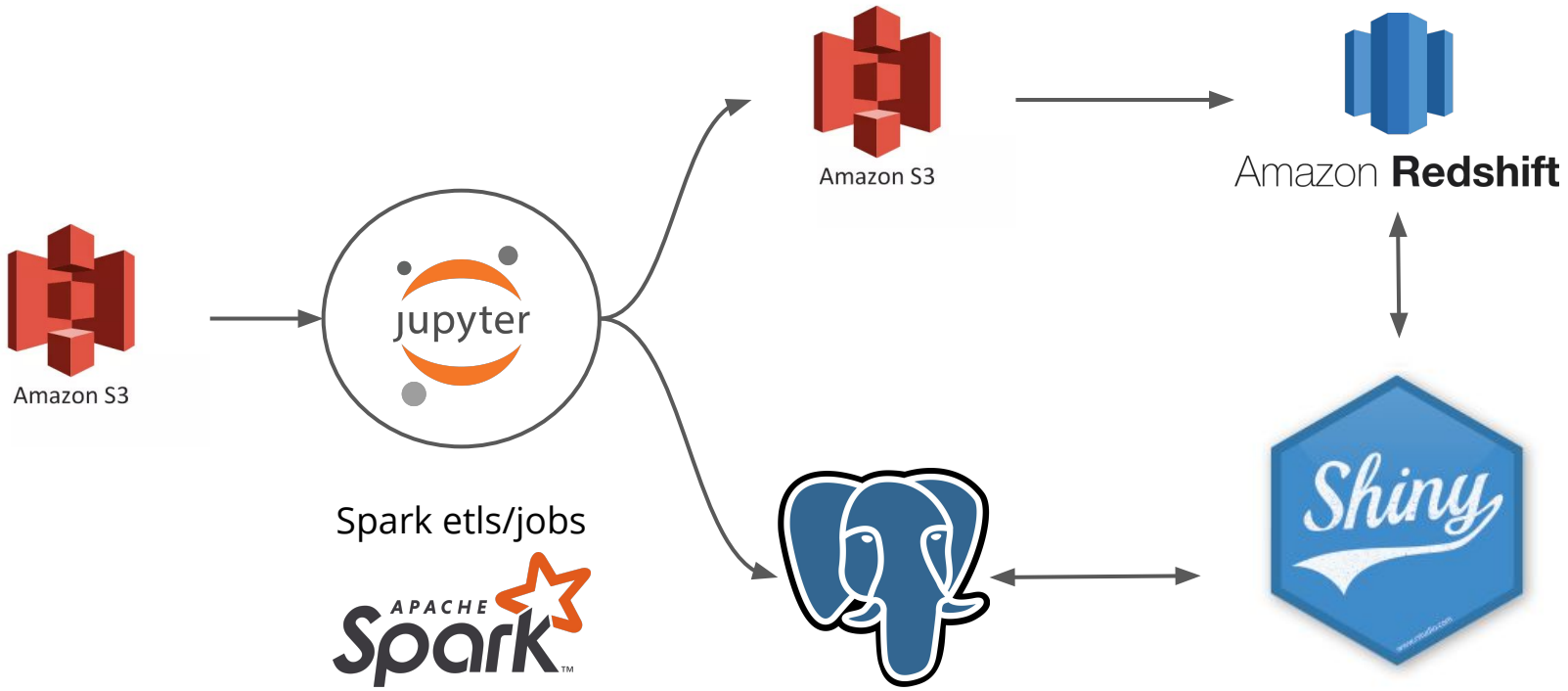
O profvis é uma ferramenta para ajudar você a entender como o R gasta seu tempo.

```
library(profvis)

profvis({runApp('Downloads/tdc_app')})
```


Flame Graph
Data
Options ▾
<expr>
Memory
Time
1 profvis({runApp('Downloads/tdc_app')})
2
5.9
2690
/Downloads/tdc_app/server.R
Memory
Time
1 function(input, output, session) {
2
3 # Combine the selected variables into a new data frame
4 selectedData <- reactive({
5 Sys.sleep(5)
0.7
1340
6 iris[, c(input$xcol, input$ycol)]
20
7 })
8
9 clusters <- reactive({
10 kmeans(selectedData(), input$clusters)
0.1
20
11 })
12
13 output$plot1 <- renderPlot({
14 palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
15 "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))
16
17 par(mar = c(5.1, 4.1, 0, 1))
0.1
10
18 plot(selectedData(),
1.1
1510
19 col = clusters()$cluster,
20 pch = 20, cex = 3)
21 points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
20
22 })
23
24 }


Flame Graph
Data
Options ▾
<expr>
Memory
Time
1 profvis({runApp('Downloads/tdc_app')})
5.6
1090
2
/Downloads/tdc_app/server.R
Memory
Time
1 function(input, output, session) {
2
3 # Combine the selected variables into a new data frame
4 selectedData <- reactive({
5 iris[, c(input$xcol, input$ycol)]
6 })
7
8 clusters <- reactive({
9 kmeans(selectedData(), input$clusters)
10 })
11
12 output$plot1 <- renderPlot({
0.3
10
13 palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
14 "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))
15
16 par(mar = c(5.1, 4.1, 0, 1))
17 plot(selectedData(),
1.2
80
18 col = clusters()$cluster,
19 pch = 20, cex = 3)
20 points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
21 })
22
23 }

Faça o trabalho fora da aplicação



Deploy

Sistema operacional

Dependências

Versão do R

Pacotes

Compartilhando ou disponibilizando sua aplicação

Algumas opções:

- Shinyapps.io;
- Shiny server (open source);
- Shiny server pro.

docker

O que é Docker?

O Docker é uma ferramenta projetada para facilitar a criação, implantação e execução de aplicativos usando contêineres. De certa forma, o Docker é um pouco como uma máquina virtual.

Imagem

Eles são "modelos prontos para uso com instruções para criar um contêiner Docker".
Define o código do contêiner, bibliotecas, variáveis de ambiente, arquivos de configuração e muito mais.

Contêiner

Um contêiner do Docker é um ambiente isolado em execução no kernel de uma máquina host que permite executar código específico do aplicativo.

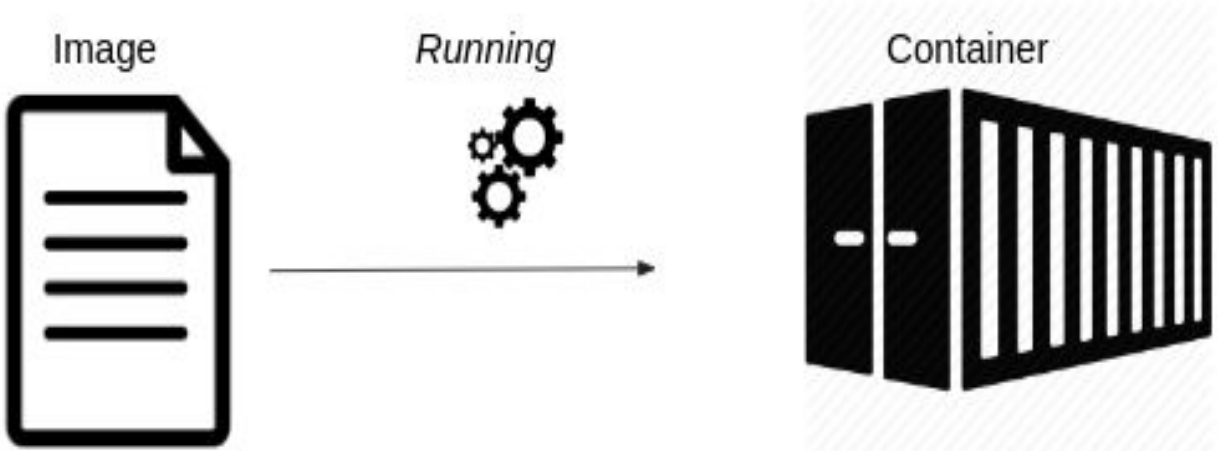


Image
Running
Container

Image Compression - PCA

Number of PC to be used in the compression:

2 64

1 8 15 22 29 36 43 50 57 64

```
FROM quantumobject/docker-shiny

RUN apt-get update \
    && apt-get install -y libcurl4-openssl-dev libssl-dev libxml2-dev libpq-dev \
    && apt-get clean \
    && rm -rf /tmp/* /var/tmp/*  \
    && rm -rf /var/lib/apt/lists/*

RUN R -e "install.packages('tidyverse')"
RUN R -e "install.packages('shiny')"
RUN R -e "install.packages('EMD')"
RUN R -e "install.packages('blockmatrix')"

RUN rm -rf /srv/shiny-server/*

COPY ./app_image_comp /srv/shiny-server

EXPOSE 3838

# Use baseimage-docker's init system.
CMD ["/sbin/my_init"]
```

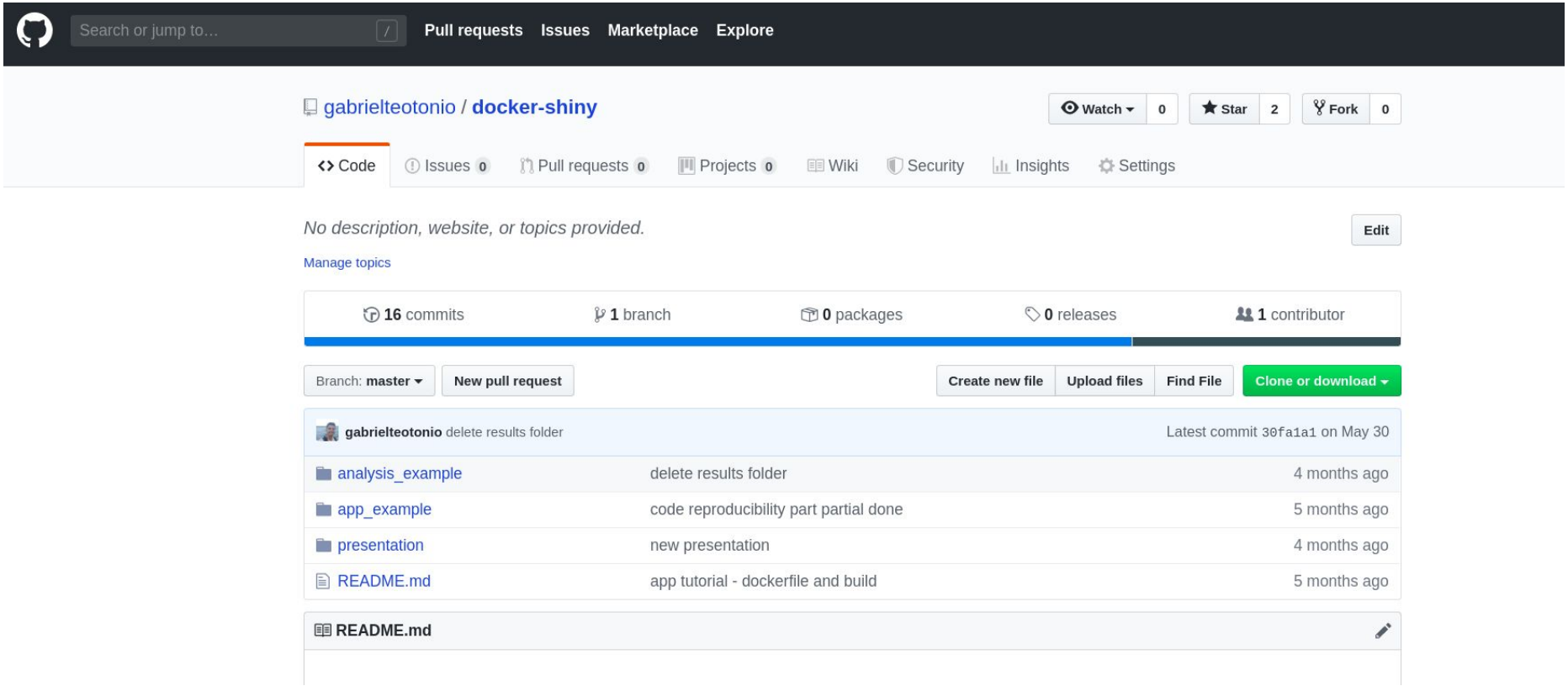
Imagem base com o Shiny server instalado

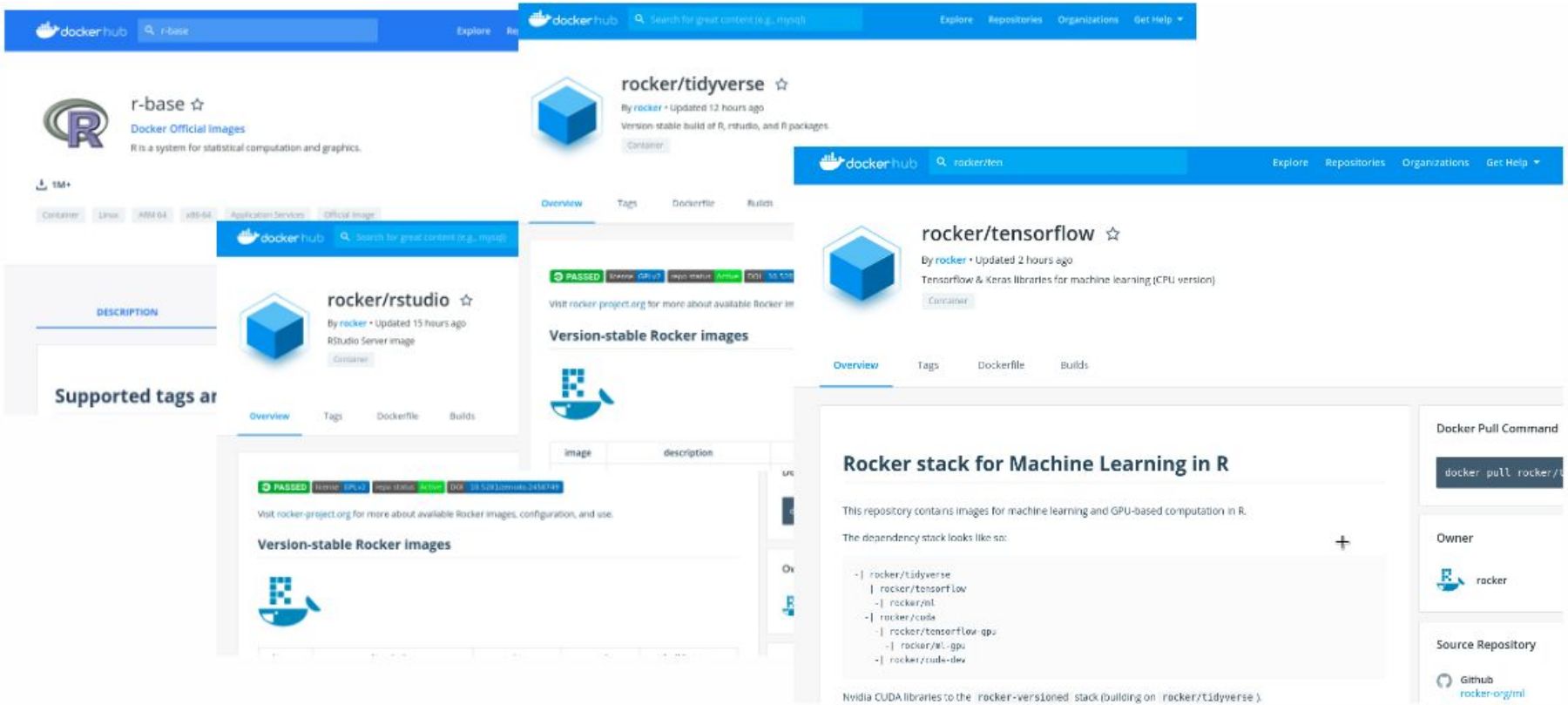
Instalando algumas dependências

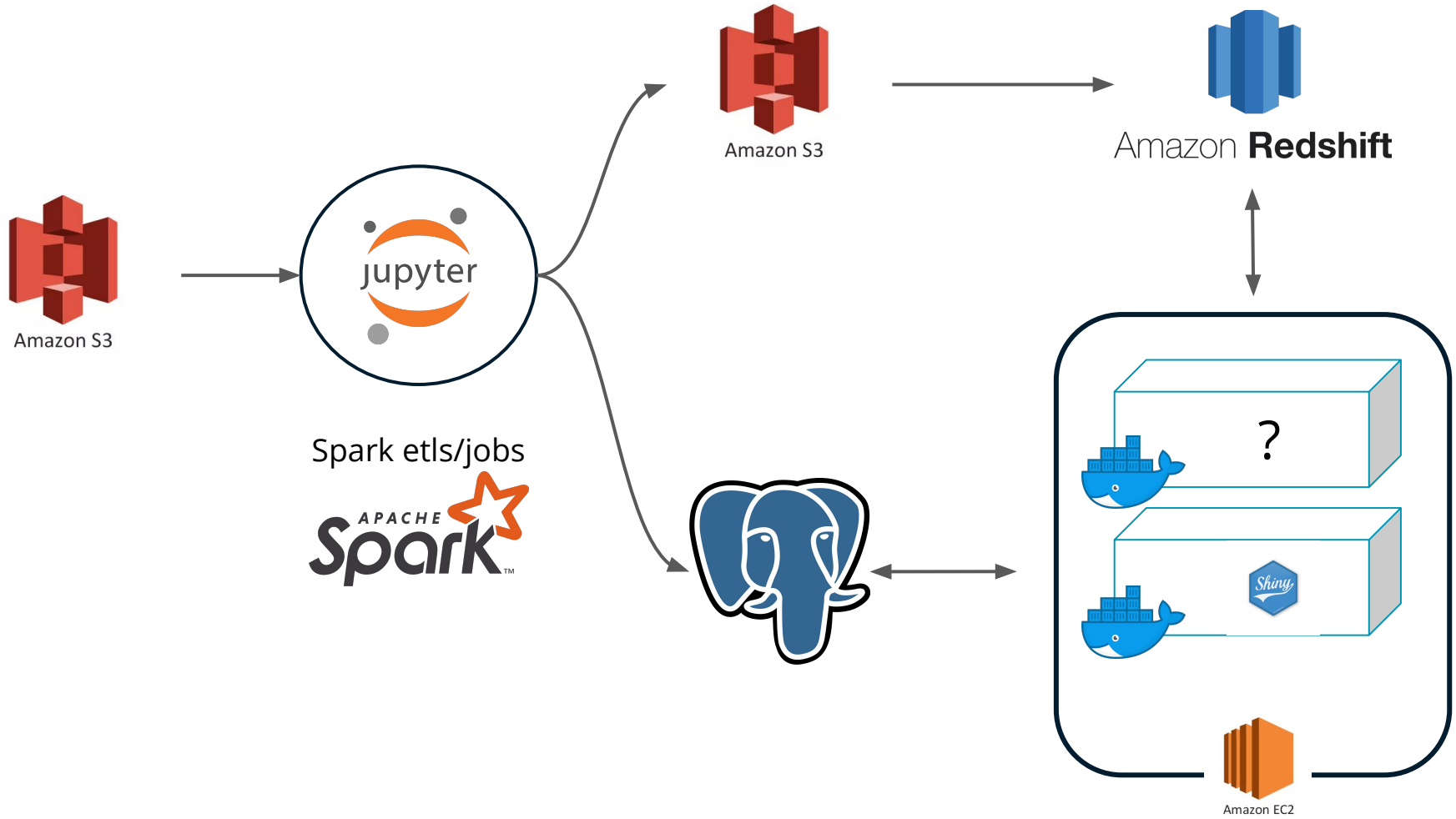
Instalando os pacotes que vamos utilizar em R

Copiando a pasta que contém meu app

Iniciando a aplicação

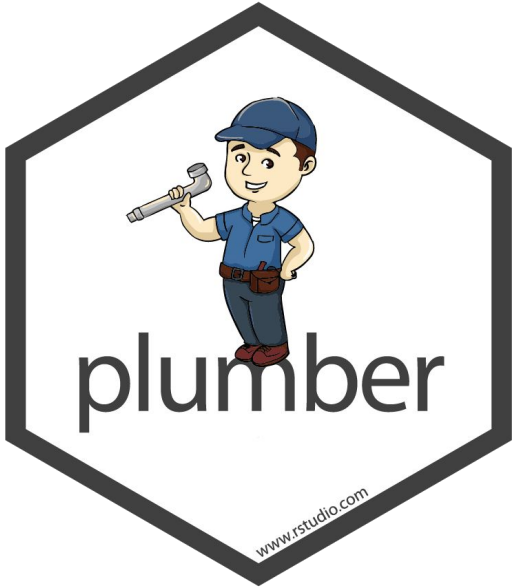

Search or jump to...
/
Pull requests
Issues
Marketplace
Explore
gabrielteotonio / docker-shiny
Watch 0
Star 2
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
16 commits
1 branch
0 packages
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
gabrielteotonio delete results folder
Latest commit 30fa1a1 on May 30
analysis_example
delete results folder
4 months ago
app_example
code reproducibility part partial done
5 months ago
presentation
new presentation
4 months ago
README.md
app tutorial - dockerfile and build
5 months ago
README.md


r-base
Docker Official Images
R is a system for statistical computation and graphics.
rocker/tidyverse
By rocker • Updated 12 hours ago
Version-stable build of R, rstudio, and tl packages
rocker/rstudio
By rocker • Updated 15 hours ago
RStudio Server image
rocker/tensorflow
By rocker • Updated 2 hours ago
Tensorflow & Keras libraries for machine learning (CPU version)
Overview
Tags
Dockerfile
Builds
Version-stable Rocker images
Rocker stack for Machine Learning in R
This repository contains images for machine learning and GPU-based computation in R.
Docker Pull Command
Owner
rocker
Source Repository
Github
rocker-org/ml

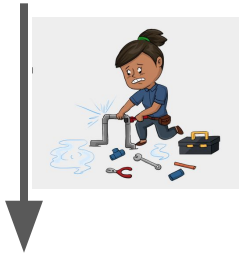

Amazon S3
jupyter
Spark etls/jobs
APACHE Spark
Amazon S3
Amazon Redshift
?
Shiny
Amazon EC2

Criando APIs

Um pacote que converte seu código em uma REST API simplesmente decorando seu código existente em R com comentários especiais.



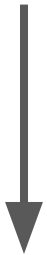
```r
data <- iris
kmeans_object <- kmeans(data, 3)

pred <- function(comprimento, largura) {
  predict(kmeans_object, newdata = data.frame(sepal_length = comprimento, sepal_width = largura))
}
```



```r
library(plumber)

data <- iris
kmeans_object <- kmeans(data, 3)

#* Retorna o cluster correspondente de uma observação
#* @param comprimento comprimento da sepala
#* @param largura largura da sepala
#* @post /pred
pred <- function(comprimento, largura) {
  predict(kmeans_object, newdata = data.frame(sepal_length = comprimento, sepal_width = largura))
}
```

Arquivo kmeans.R

```
p <- plumber::plumb('R/kmeans.R')
p$run(port = 8888)
```



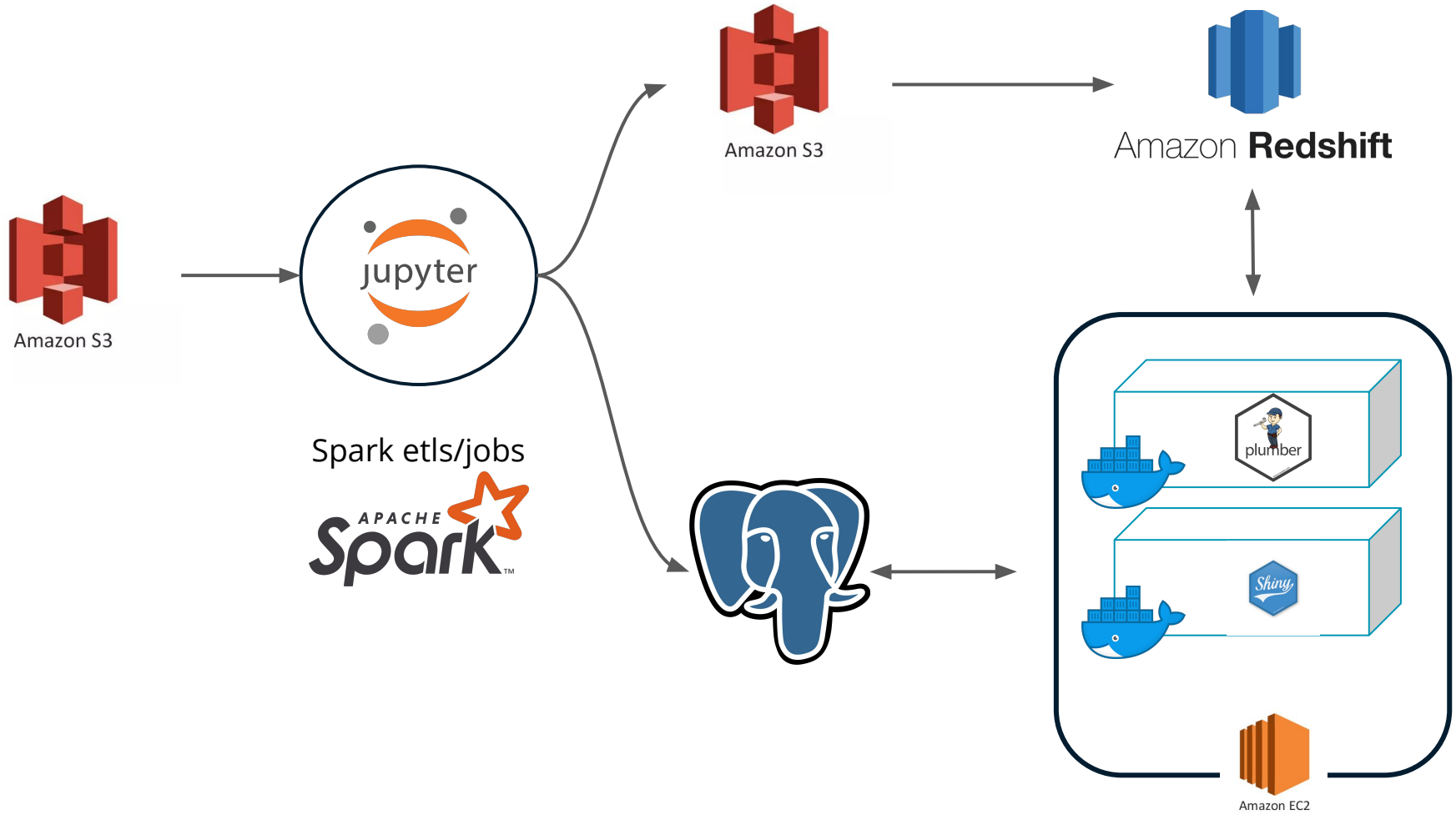
```
curl --data "comprimento=5.1&largura=3.5" "http://localhost:8888/pred"
```

```
FROM r-base:3.5.1

RUN apt-get update && apt-get install -y libcurl4-openssl-dev libssl-dev libxml2-dev libpq-dev


RUN install2.r plumber
WORKDIR /usr/plumberR

COPY kmeans.R /usr/plumberR

EXPOSE 8000
ENTRYPOINT ["R", "-e", "pr <- plumber::plumb(commandArgs()[4]); pr$run(host='0.0.0.0', port=8000)"]
CMD ["kmeans.R"]
```

Imagem base do R

Instalando o plumber e criando um diretório

Copiando nosso arquivo para o diretório criado


Amazon S3
jupyter
Spark etls/jobs
APACHE Spark
Amazon S3
Amazon Redshift
plumber
Shiny
Amazon EC2

Vá mais longe!

- Banco:
  - SQL injections attack.
- Autenticação:
  - Pacote auth0 (Curso-R).
- Teste:
  - shinytest;
  - plotCaching.
- Deploy:
  - kubernetes.

gabriel.teotonio@inloco.com.br

inloco.com.br/careers
medium.com/inlocotech