

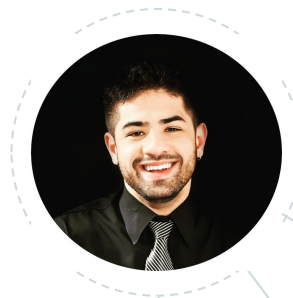
R em produção: fazendo deploy de dashboards na AWS.



Eae Galera!

Dennis & Andryas



(ノ◡◡)ノ*:°◇ ◇°:*ゝ(◡◡ゝ)



- 
- 
- ◎ Capital
 - ◎ Shiny
 - ◎ Docker
 - ◎ Docker Compose
 - ◎ Shinyproxy
 - ◎ AWS



Investimento e tecnologia para o mercado imobiliário.



- +11 Operações
- R\$ 400mi [VGV]

Powered by machines, managed by experts.

1.

Shiny

Transformando fotos em filmes.



O que é o shiny?

Antes de tudo é um pacote do R, que é um framework que permite criar aplicações web interativas.


Por que o shiny?

- É rápido para desenvolver e testar.
- Abstrai conhecimentos de front-end.
- Permite usar o que há mais de novo em estatística.

Aplicação

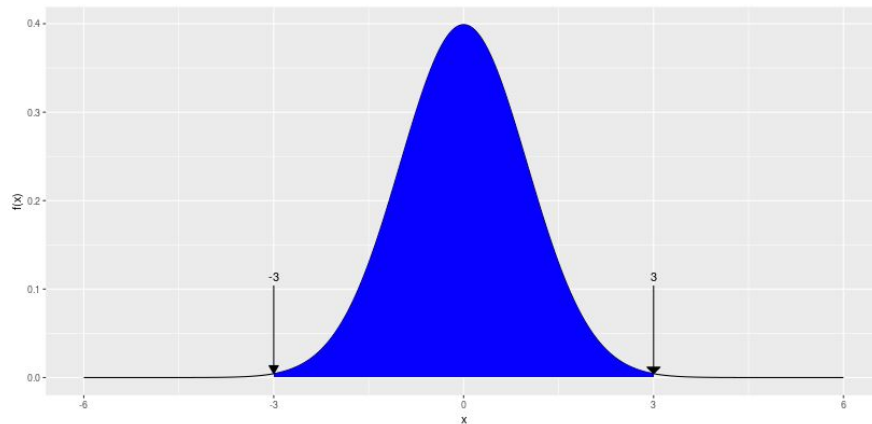
Principais Modelos Discretos e Contínuos

Escolha um Modelo

Normal 

μ σ

X_1 $\leq X \leq$ X_2



$$P(X_1 \leq X \leq X_2) = P(-3 \leq X \leq 3) = P\left(\frac{X_1 - \mu}{\sigma} \leq \frac{X - \mu}{\sigma} \leq \frac{X_2 - \mu}{\sigma}\right) = P\left(\frac{-3 - 0}{1} \leq Z \leq \frac{3 - 0}{1}\right)$$
$$P(-3 \leq Z \leq 3) = 0.99730020393674$$

<https://github.com/andryas/distshiny>

ui.R, Front-end

```
library(shiny)

# Reseta o ID
jscode <- "Shiny.addCustomMessageHandler('resetValue', function(variableName) {
  Shiny.onInputChange(variableName, null);
});"

shinyUI(
  fluidPage(
    titlePanel("Principais Modelos Discretos e Contínuos"),
    withMathJax(),
    sidebarLayout(
      sidebarPanel(
        tags$head(tags$script(jscode)),
        fluidRow(
          column(width = 8,
            div(style = "height:64px;",
              selectInput("dist", label = "Escolha um Modelo", selected = "normal",
                choices = list(
                  Continua = c("Normal" = "normal",
                               "Exponencial" = "exponencial",
                               "Uniforme" = "uniformecont",
                               "Beta" = "beta",
                               "Gamma" = "gamma"),
                  Discreta = c("Uniforme" = "uniformedis",
                              "Binomial" = "binomial",
                              "Poisson" = "poisson",
                              "Geométrico" = "geometrico",
                              "Hipergeométrica" = "hipergeometrica",
                              "Binomial Negativa" = "binomialneg")))),
          column(width = 4, align = "center",
            div(style = "height:64px; margin-top: 26px",
              actionButton("sobre", "", icon("book"), width = "100%"))
          ),
        ),
      ),
    ),
  )
```

⦿ ui.R

⦿ server.R

⦿ global.R

server.R, Back-end

```
library(shiny)

shinyServer(function(input, output, session) {
  # Variável auxiliar
  aux <- reactiveValues()

  # Normal -----
  # Calcula z1 e z2 padronizado e retorna a probabilidade entre eles
  rn <- reactive({normal(input$mu,input$sigma,input$x1,input$x2)})

  # Valor reativo para média da normal
  mun <- reactive({input$mu})

  observe({
    # Este 'se' só serve para a primeira iteração
    if(is.null(aux$mu)) {aux$mu <- mun()}

    # Caso haja uma alteração de input$mu, as arrow's mudam igualmente de -3 a 3.
    if(aux$mu != input$mu) {
      updateNumericInput(session,"x1",label = "",value = input$mu - 3 * input$sigma)
      updateNumericInput(session,"x2",label = "",value = input$mu + 3 * input$sigma)
      aux$mu <- input$mu
    }
  })
})
```

⦿ ui.R

⦿ server.R

⦿ global.R

global.R, Pre-process

```
library(ggplot2)

# Continuous
normal <- function(mu,sig,x1,x2) {
  x <- seq(from = mu - 6 * sig, to = mu + 6 * sig,length.out = 1000)

  df <- data.frame(x = x,
                  y = dnorm(x,mu,sig))

  df2 <- data.frame(x = c(x1,seq(x1,x2,length.out = 1000),x2),
                  y = c(0,dnorm(seq(x1,x2,length.out = 1000),mu,sig),0))

  sumy <- df$y[which.max(df$y)] / 4

  arrow1 <- rbind(df2[2,],c(df2[2,1],df2[2,2] + sumy))
  arrow2 <- rbind(df2[nrow(df2)-1,], c(df2[nrow(df2)-1,1],
                                     df2[nrow(df2)-1,2] + sumy))

  z1 <- (x1 - mu) / sig
  z2 <- (x2 - mu) / sig

  g <- qplot(df$x,df$y, geom = "line", xlab = "x", ylab = "f(x)") +
    geom_polygon(data = df2, aes(x,y), fill = "blue") +
    geom_line(data = arrow1, aes(x = x, y = y),
              arrow = arrow(length = unit(0.3,"cm"),
                            ends = "first",type = "closed")) +
    annotate("text",x = arrow1$x[2], y = arrow1$y[2] + sumy * 0.1, label = z1) +
    geom_line(data = arrow2, aes(x,y),
              arrow = arrow(length = unit(0.3,"cm"),
                            ends = "first",type = "closed",angle = 45)) +
    annotate("text",x = arrow2$x[2], y = arrow2$y[2] + sumy * 0.1, label = z2)

  list(g,z1,z2,r = pnorm(z2) - pnorm(z1))
}
```

© ui.R

© server.R

© global.R

2. **Docker**

Revolucionando a logística do
desenvolvimento ao deploy.



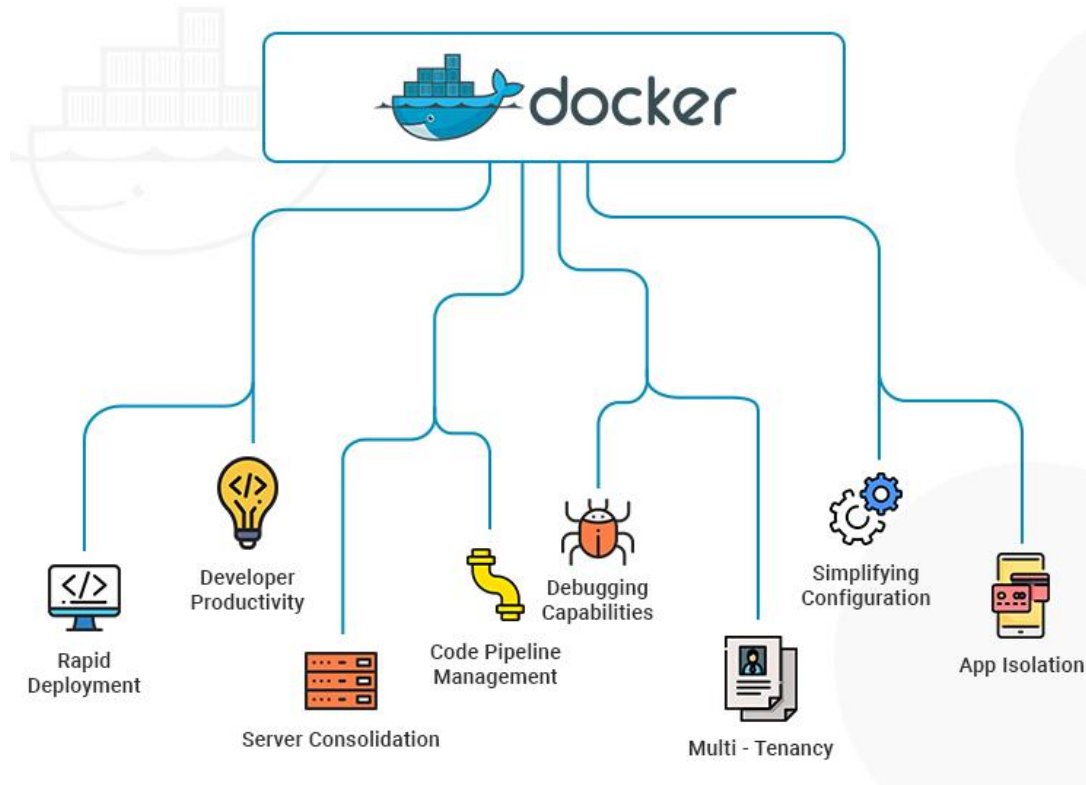
O que é o docker?

Docker é uma plataforma aberta para desenvolvimento, envio e execução de aplicativos.

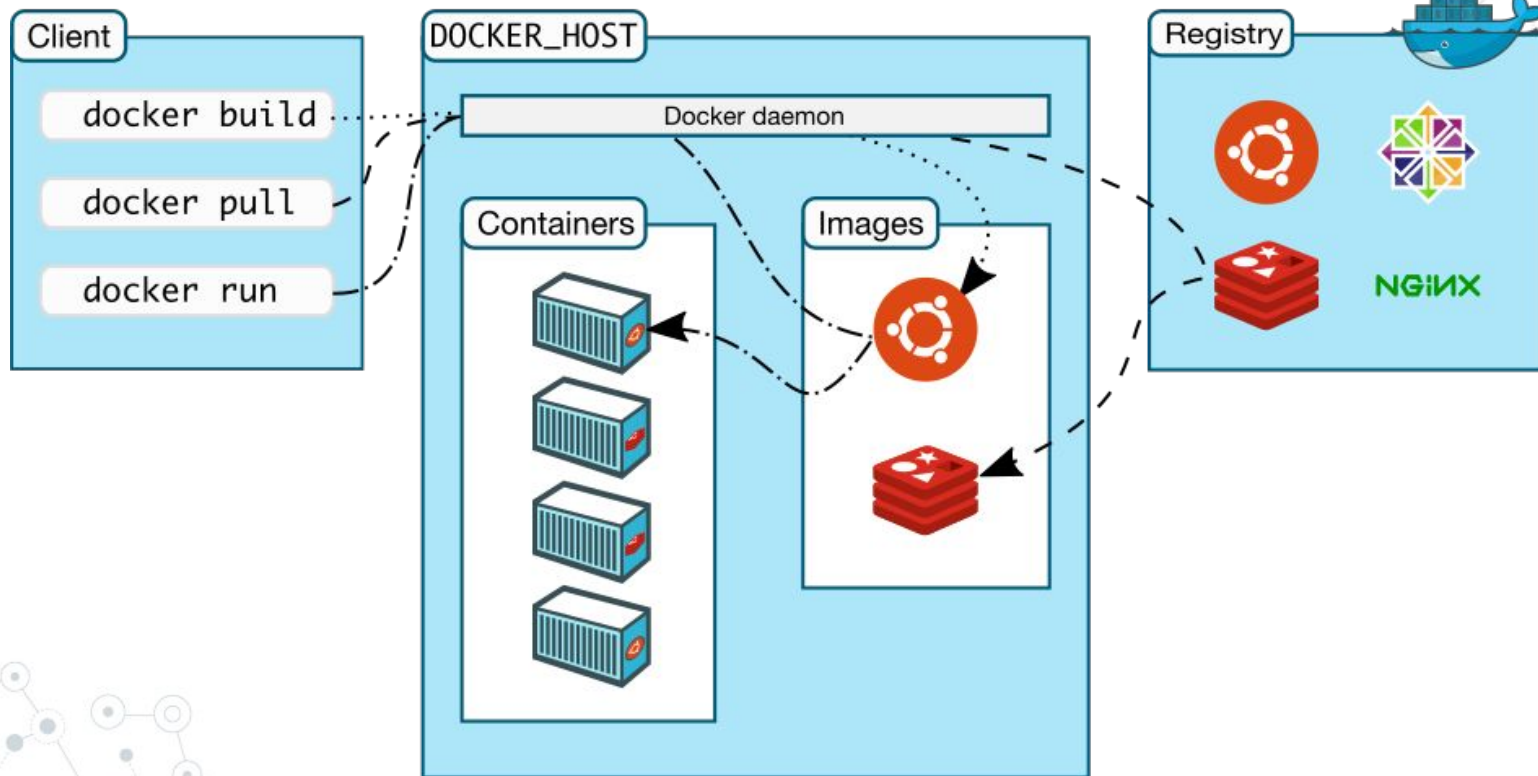
O que o docker faz?

O Docker permite que você separe seus aplicativos de sua infraestrutura para que possa entregar o software rapidamente isolando dependências com foco na portabilidade ao estilo `build>ship>run.`

Por quê?

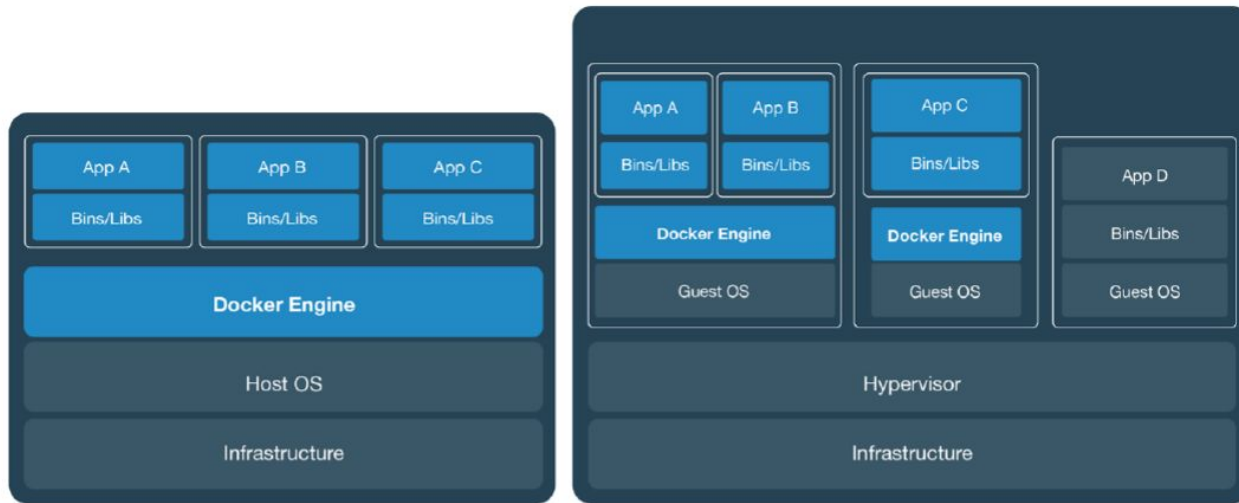


Como?



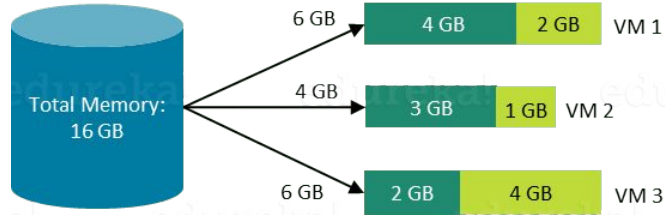
Onde?

Containers Vs VMs



Diferenças

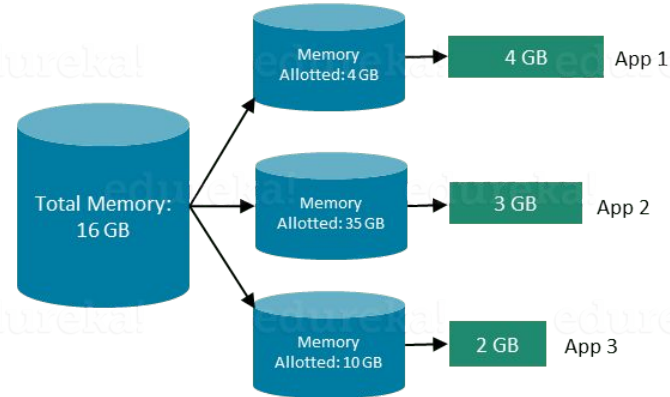
In case of Virtual Machines



7 Gb of Memory is blocked and cannot be allotted to a new VM

In case of Docker

edureka!



Only 9 GB memory utilized;
7 GB can be allotted to a new Container

Prós e Cons

Containers

- Dividem o mesmo Kernel do Host
- Construção Incremental (Camadas)
- Versionamento em Repositórios
- Leves
- Vários Containers de uma mesma Imagem

VMs

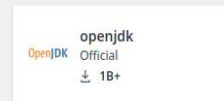
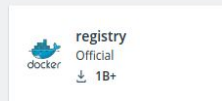
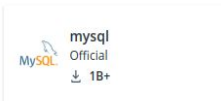
- Cada VM tem seu próprio OS
- Snapshots raramente utilizados
- Sem controle de versão fácil
- Normalmente mais pesadas
- Apenas uma VM pode ser instanciada do .VMX e .VMDK



Docker Hub is the world's largest library and community for container images

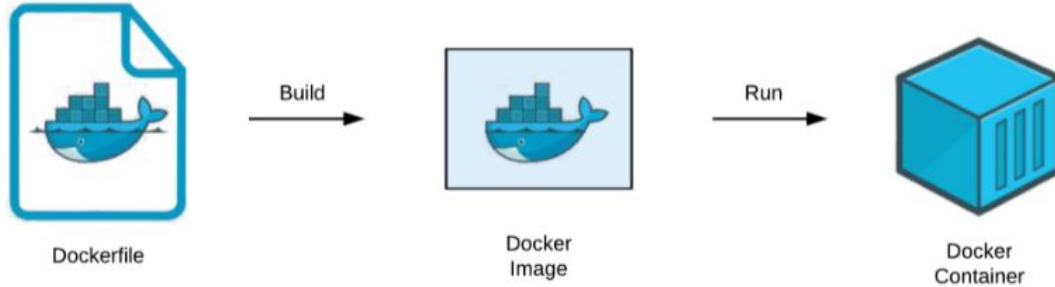
Browse over 100,000 container images from software vendors, open-source projects, and the community.

Official Images

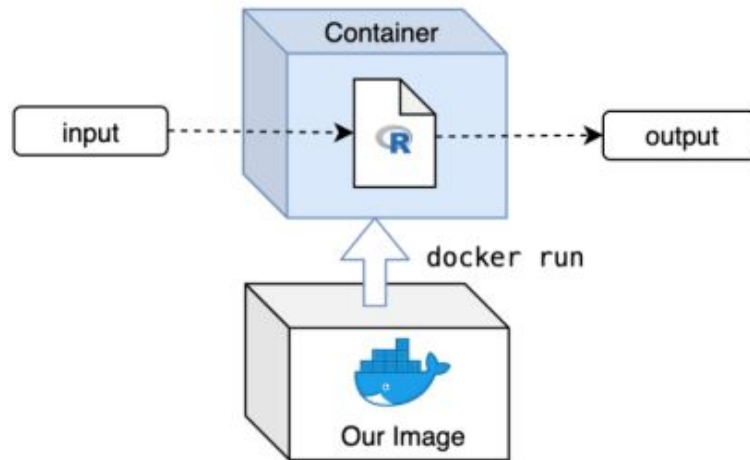


[See all Official Images >](#)

Workflow



R&R



Hands On

```
FROM rocker/tidyverse:4.1.0

COPY .Renviro root/.Renviro

RUN R -e 'install.packages("remotes")'
RUN R -e 'remotes::install_github("rstudio/shiny@v1.6.0")'

RUN mkdir /root/app
COPY global.R /root/app/global.R
COPY ui.R /root/app/ui.R
COPY server.R /root/app/server.R

EXPOSE 3838

CMD ["R", "-e", "shiny::runApp('/root/app')"]
```

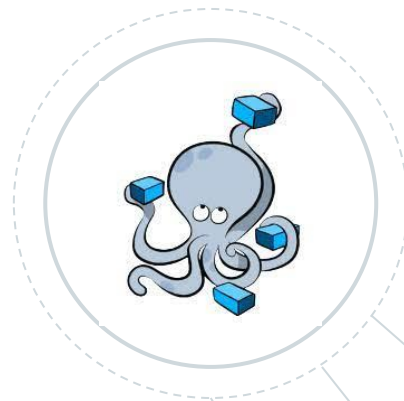
```
# buildamos a imagem
docker build -t shinyapp diretorio/do/seu/dockerfile

# instanciar a imagem num container
docker run shinyapp
```

3.

Docker Compose

Escalando suas aplicações e serviços.





O que é o docker compose?

Docker compose é uma ferramenta para definir e executar aplicativos docker de vários contêineres em simultâneo.

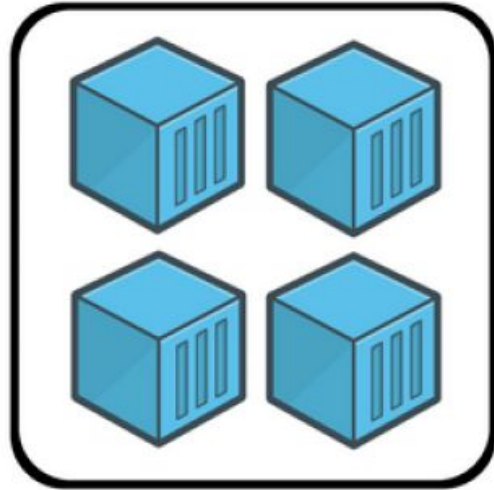
O que o docker compose faz?

Com o docker compose, você usa o arquivo YAML para configurar os serviços da sua aplicação e criar todos de uma só vez.

Por quê?



Docker



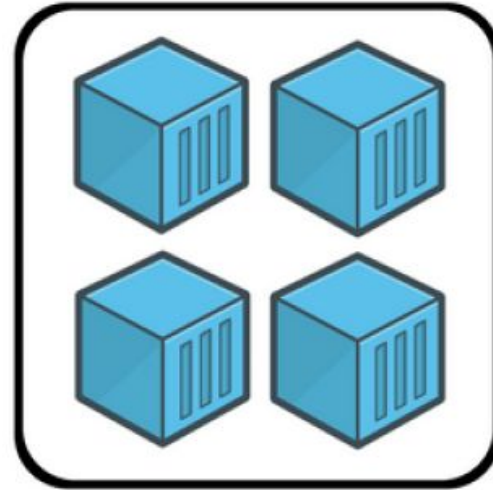
Docker Compose

Por quê?



Docker

=



Docker Compose

Workflow

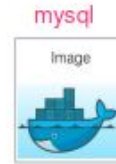
```
docker-compose.yml

version: "3.7"
services:
  db:
    image: mysql:8.0.19
    restart: always
    environment:
      - MYSQL_DATABASE=example
      - MYSQL_ROOT_PASSWORD=password
  app:
    build: app
    restart: always
  web:
    build: web
    restart: always
    ports:
      - 80:80
```

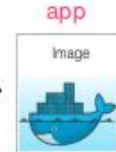
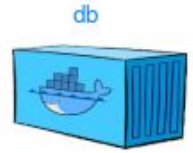


Dockerfile

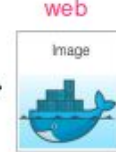
Dockerfile



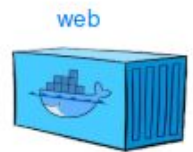
run



run



run



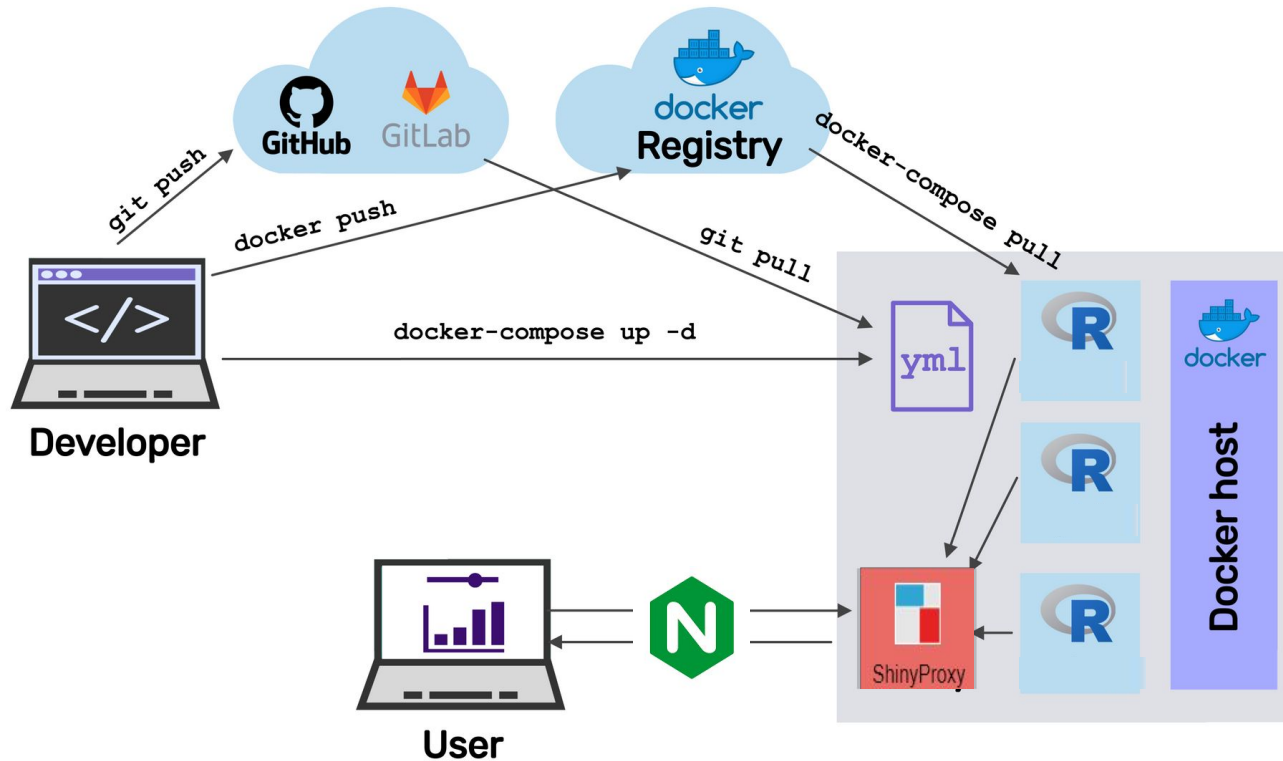
Hands On

```
version: '3.6'

services:
  distshiny1:
    image: repositorio/distshiny
    container_name: distshiny1
    build: ./distshiny
    network_mode: "host"
    ports:
      - 2021:2021

  distshiny2:
    image: repositorio/distshiny
    container_name: distshiny2
    build: ./distshiny
    network_mode: "host"
    ports:
      - 2020:2020
```

Aplicação



4.

Shinyproxy

Para uma orquestra, um maestro a altura.

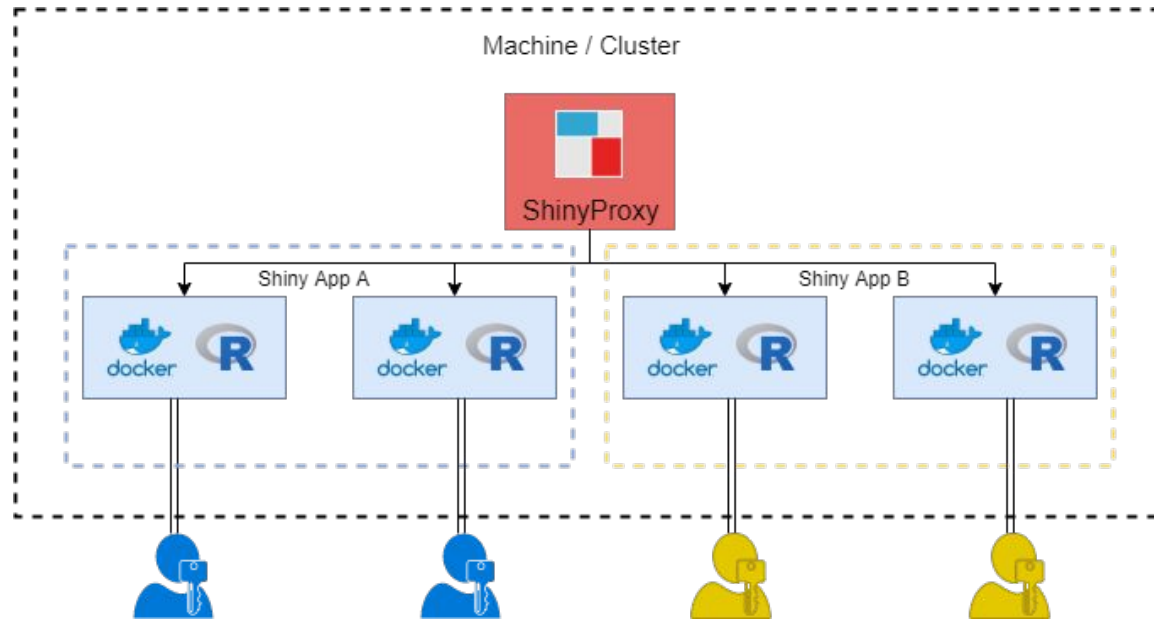


O que é o shinyproxy?

Shinyproxy é um orquestrador de aplicações shiny, com LDAP, login e senha, que torna a aplicação segura e escalável.



Workflow



O projeto como um todo

```
├── application
│   ├── application.yml
│   └── Dockerfile
├── apps
│   ├── bloodbanks
│   │   ├── blood-banks.csv
│   │   ├── blood-banks_raw.csv
│   │   ├── Dockerfile
│   │   ├── readme.md
│   │   ├── server.R
│   │   └── ui.R
│   └── distshiny
│       ├── Dockerfile
│       ├── global.R
│       ├── server.R
│       └── ui.R
└── docker-compose.yml
```

4 directories, 13 files

Para configurar shinyproxy

Aplicação shiny

Para organizar e gerenciar
as imagens docker.

application.yml

```
proxy:
  title: Shinyproxy em produção
  landing-page: /
  # template-path: ./templates
  heartbeat-rate: 10000
  heartbeat-timeout: 50000
  bind-address: 0.0.0.0
  container-wait-time: 60000
  hide-navbar: false
  port: 8080

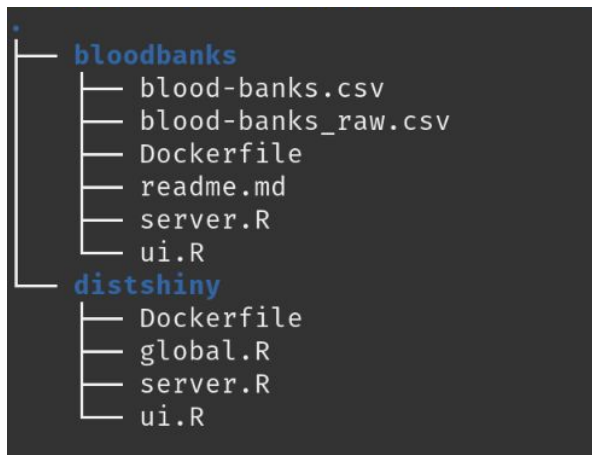
  authentication: none

docker:
  internal-networking: true
  container-network: mynet

specs:
  - id: distshiny
    display-name: Distribuições de Probabilidade
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/distshiny
    container-network: "${proxy.docker.container-network}"
  - id: bloodbanks
    display-name: Localizando bancos de sangue na Índia
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/bloodbanks
    container-network: "${proxy.docker.container-network}"
```

[Documentação](#)

apps/



docker-compose.yml

```
version: '3.6'

services:
  shinyproxy:
    image: aw/shinyproxy
    container_name: shinyproxy
    build: ./application
    volumes:
      - ./application/application.yml:/opt/shinyproxy/application.yml
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 8080:8080
    networks:
      - mynet

  distshiny:
    image: aw/distshiny
    container_name: distshiny
    build: ./apps/distshiny

  bloodbanks:
    image: aw/bloodbanks
    container_name: bloodbanks
    build: ./apps/bloodbanks

networks:
  mynet:
    name: mynet
```

Hands on

```
$ sudo docker-compose build  
$ sudo docker-compose up -d shinyproxy
```

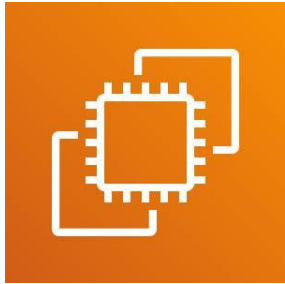
4.

AWS

Disponibilizando seus aplicativos

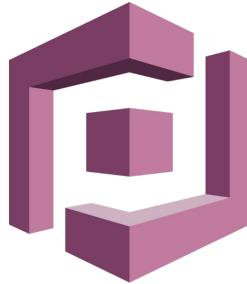


Serviços necessários



EC2

Serviço para
instanciar seu
servidor



Cognito

Para gerenciar
users



Route53

Serviço para escalar
DNS e seu **domínio**
na internet

Precisa ter um domínio!

O que muda no projeto?

```
├── application
│   ├── application.yml
│   └── Dockerfile
├── apps
│   ├── bloodbanks
│   │   ├── blood-banks.csv
│   │   ├── blood-banks_raw.csv
│   │   ├── Dockerfile
│   │   ├── readme.md
│   │   ├── server.R
│   │   └── ui.R
│   └── distshiny
│       ├── Dockerfile
│       ├── global.R
│       ├── server.R
│       └── ui.R
├── data
│   └── nginx
│       └── app.conf
├── docker-compose.yml
├── init-letsencrypt.sh
├── main.sh
└── shinyproxy.pem
```

configurações do
nginx

script para configurar de forma
segura nossa aplicação na internet

.pem para acessar
a ec2 na AWS

data/nginx

```
server {
    listen 80;
    server_name shortcutia.com;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name shortcutia.com;
    access_log /var/log/nginx/shinyproxy.access.log;
    error_log /var/log/nginx/shinyproxy.error.log error;
    server_tokens off;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    ssl_certificate /etc/letsencrypt/live/shortcutia.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/shortcutia.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://shinyproxy:8080;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 600s;

        proxy_redirect off;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Trocar pelo seu
domínio

init-letsencrypt.sh

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
    echo 'Error: docker-compose is not installed.' >&2
    exit 1
fi

domains=(shortcutai.com)
rsa_key_size=4096
data_path="./data/certbot"
email="andryaas@gmail.com" # Adding a valid address is strongly recommended
staging=1 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data_path" ]; then
    read -p "Existing data found for $domains. Continue and replace existing certificate? (y/N) " decision
    if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
        exit
    fi
fi

...
```

Requisitar os certificados para
deixar sua aplicação pública na
internet

Seu domínio

docker-compose.yml

Novos serviços

```
version: '3.6'

services:
  nginx:
    image: nginx:1.15-alpine
    container_name: nginx
    depends_on:
      - shinyproxy
      - certbot
    restart: unless-stopped
    volumes:
      - ./data/nginx:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - 80:80
      - 443:443
    networks:
      - mynet
    command: "/bin/sh -c 'while ;; do sleep 6h & wait $$(!); nginx -s reload; done & nginx -g \"daemon off;\"'"

  certbot:
    image: certbot/certbot
    container_name: certbot
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    networks:
      - mynet
    entrypoint: "/bin/sh -c 'trap exit TERM; while ;; do certbot renew; sleep 12h & wait $$(!); done;'"

  shinyproxy:
    image: aw/shinyproxy
    container_name: shinyproxy
    build: ./application
    networks:
      - mynet
    volumes:
      - ./application/application.yml:/opt/shinyproxy/application.yml
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 8080:8080

...
```

Hands on

```
local$ bash main.sh
~ 1
server$ cd deploy
server$ nano init-letsencrypt.sh
staging=1 -> staging=0
server$ sudo ./init-letsencrypt
```

Utilizando Cognito

- App clients -> add an app client -> qlq coisa -> disable “Enable token revocation”
Pegue o client id e o client secret
- App client settings -> Marque “Cognito User Pool”
Callback URL (s): `https://{your_domain}/login/oauth2/code/shinyproxy`
Sign out URL (s): `https://{your_domain}`
Marque: Authorization code grant, Implicit grant, email, openid, `aws.cognito.signin.user.admin` e profile
- General settings -> Pool Id -> `{userPoolId}`
- App integration -> Add Domain -> o que voce quiser = `{cognito_domain_prefix}`
- Região -> O que vem na frente do Pool Id -> `sa-east-1`, por exemplo = `{region}`

Agora só substituir cada elemento no seu respectivo lugar no `application.yml` e pronto, o cognito está configurado!

Cognito

Para gerenciar usuários e grupos, vá na parte General settings -> Users and groups, crie um usuário, depois vá na aba group, crie o grupo admins e tantos outros que quiser e para cada usuário adicione ele no grupo que ele pertence. Depois só gerenciar no application.yml

```
specs:
  - id: distshiny
    display-name: Distribuições de Probabilidade
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/distshiny
    access-groups: [everybody, admins]
    container-network: "${proxy.docker.container-network}"
  - id: bloodbanks
    display-name: Localizando bancos de sangue na Índia
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/bloodbanks
    container-network: "${proxy.docker.container-network}"
    access-groups: [admins]
```

Atualizando application.yml

```
proxy:
  title: Shinyproxy em produção
  landing-page: /
  template-path: ./templates
  heartbeat-rate: 10000
  heartbeat-timeout: 50000
  bind-address: 0.0.0.0
  container-wait-time: 60000
  port: 8080

authentication: openid # use openid auth framework
openid:
  roles-claim: cognito:groups # use the groups value passed by AWS cognito to identify user groups
  auth-url: https://{cognito_domain_prefix}.auth.{region}.amazoncognito.com/oauth2/authorize
  token-url: https://{cognito_domain_prefix}.auth.{region}.amazoncognito.com/oauth2/token
  jwks-url: https://{cognito_idp.{region}.amazonaws.com/{userPoolId}/.well-known/jwks.json
  logout-url: https://{cognito_domain_prefix}.auth.{region}.amazoncognito.com/logout?client_id={client_id}&logout_uri={your_host_url}
  client-id: get this from AWS Cognito user pool management page
  client-secret: get this from AWS Cognito user pool management page

docker:
  internal-networking: true
  container-network: mynet

specs:
  - id: distshiny
    display-name: Distribuições de Probabilidade
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/distshiny
    access-groups: [everybody, admins]
    container-network: "${proxy.docker.container-network}"
  - id: bloodbanks
    display-name: Localizando bancos de sangue na Índia
    container-cmd: ["R", "-e", "shiny::runApp('/root/app', port = 3838, host = '0.0.0.0')"]
    container-image: aw/bloodbanks
    container-network: "${proxy.docker.container-network}"
    access-groups: [admins]

server:
  forward-headers-strategy: native
```

Hands on

```
local$ bash main.sh
~ 1
server$ cd deploy
server$ nano init-letsencrypt.sh
staging=1 -> staging=0
server$ sudo ./init-letsencrypt
```

