

Desafio Devops – Daniel Tavares

Meu GitHub: <https://github.com/dantavares32>

Meu linkedin: <https://www.linkedin.com/in/tavaresdan/>

Instalação e Configuração de Icecast com Docker e Nginx

Nesse desafio já prontas diretamente do Docker Hub para utilizar e realizar a instalação e configurações necessárias desse pequeno projeto.

Para começar utilizei o docker compose para orquestrar as duas imagens.

Pelo link <https://hub.docker.com/r/sourcefabric/icecast> não consta a imagem, então optei por utilizar a imagem mouli/icecast, verifiquei e tem bastante downloads e melhor em avaliação que estava funcionando no docker hub.

É importante dizer aqui que foi criado o arquivo **docker-compose.yaml**, neste exemplo estou utilizando o VS Code.

Arquivo que permite inserir as configurações necessárias para instalar e executar os containers e as duas imagens.



Antes de configurar todo o código do arquivo docker-compose.yaml, foi necessário criar um arquivo em xml para carregar uma configuração personalizadas do icecast, optei em colocar como icecast.xml para facilitar a compreensão do código.

Passo 1 – Criação Arquivo Personalizado icecast.xml

Este documento descreve a configuração básica de um servidor Icecast. A configuração inclui definições de limites, autenticação, pontos de montagem e mais.

```
docker-compose.yaml nginx.conf icecast.xml X
icecast.xml
1 <icecast>
2   <limits>
3     <clients>2</clients>
4     <sources>2</sources>
5     <admin-users>1</admin-users>
6   </limits>
7
8   <hostname>localhost</hostname>
9   <listen-socket>
10    <port>8000</port>
11    <protocol>http</protocol>
12  </listen-socket>
13
14  <authentication>
15    <admin-user>admin</admin-user>
16    <admin-password>teste123</admin-password> <!-- Senha de admin -->
17    <source-password>testesource123</source-password> <!-- Senha de transmissão -->
18  </authentication>
19
20  <mount>
21    <mount-name>/stream.mp3</mount-name>
22    <file>/path/to/your/stream.mp3</file>
23  </mount>
24
25  <directory>public_html</directory>
26 </icecast>
```

<icecast> Elemento principal que encapsula todas as configurações do servidor Icecast.

<limits> Define os limites globais para o servidor Icecast:

clients: 2 - Número máximo de clientes (ouvintes) permitidos.

sources: 2 - Número máximo de fontes de transmissão permitidas.

admin-users: 1 - Número máximo de usuários administrativos permitidos.

<hostname> Define o nome do host ou IP onde o servidor Icecast está sendo executado:

localhost - Indica que o servidor é acessível localmente.

<listen-socket> Configura o socket de escuta do servidor:

port: 8000 - Porta na qual o Icecast escuta as conexões.

protocol: http - Protocolo usado para as conexões (HTTP).

<authentication> Configura as credenciais usadas para acessar e administrar o servidor Icecast, assim como as credenciais necessárias para as fontes de transmissão:

admin-user: admin - Nome de usuário para administração.

admin-password: teste123 - Senha para o painel de administração.

source-password: testesource123 - Senha usada pelas fontes de transmissão.

<mount> Define o ponto de montagem para o fluxo de áudio:

mount-name: /stream.mp3 - Nome do ponto de montagem para o fluxo de áudio.

file: /path/to/your/stream.mp3 - Caminho para o arquivo de áudio que será transmitido.

<directory> Especifica o diretório público a partir do qual o Icecast pode servir arquivos:
public_html - Diretório onde os arquivos públicos, como páginas HTML, serão servidos.

Passo 2 – Configuração do Docker Compose

Conforme a imagem abaixo, foi definido os serviços que serão executados.

- Serviço é o Icecast
- Image é a imagem utilizada do docker hub com nome moul/icecast:lastest
- Container_name define o nome do container utilizado
- Ports que são as portas utilizada pela aplicação
- Volumes é um volume nomeado e diretório onde os arquivos do container serão montados
- Environment é os dados de autenticação do container para a aplicação

```
docker-compose.yaml
1  services:
2    icecast:
3      image: moul/icecast:lastest
4      container_name: icecast
5      ports:
6        - "8000:8000"
7      volumes:
8        - icecast-data:/usr/local/share/icecast/web
9      environment:
10       - ICECAST_ADMIN_PASSWORD=teste123
11       - ICECAST_SOURCE_PASSWORD=testesource123
12
```

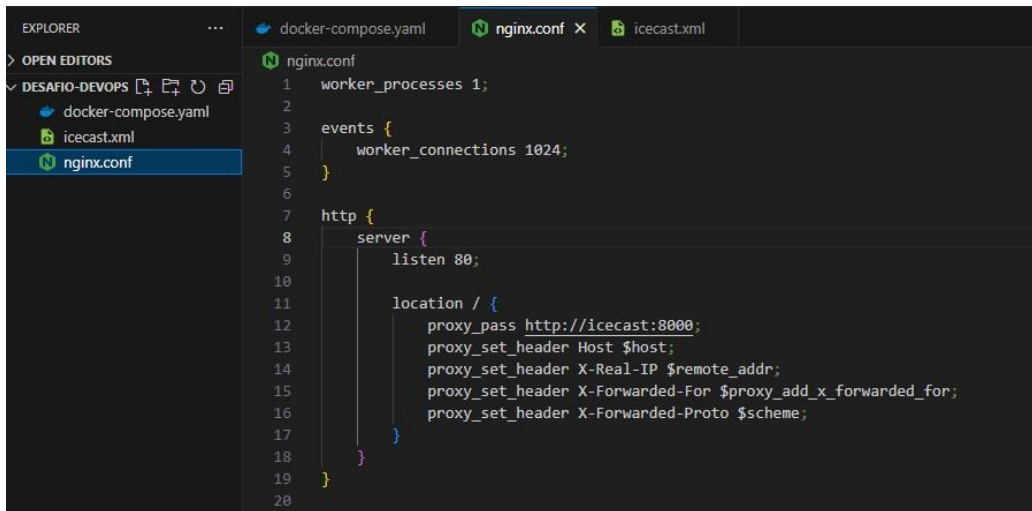
Na próxima imagem, foi definido os serviços da aplicação nginx.

- Serviço é o nginx
- image é a imagem oficial do nginx pelo docker hub
- container_name define o nome do container utilizado, nesse caso nginx
- Ports que são as portas utilizada pela aplicação do nginx
- volumes é um volume nomeado e diretório onde os arquivos do container serão montados, mas aqui estou referenciando o arquivo criado com nome nginx.conf que é o arquivo para proxy reverso, explicado logo abaixo.
- depends_n permite identificar que um serviço depende do outro e que deve ser executado depois desse serviço

```
nginx:
  image: nginx:latest
  container_name: nginx
  ports:
    - "80:80"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
  depends_on:
    - icecast
```

Passo 3 - Configuração do Nginx

Na imagem abaixo, mostra o conteúdo do arquivo nginx para que através dele mostre como a aplicação deve operar e executar as requisições, aqui foi configurado para que faça o proxy reverso.



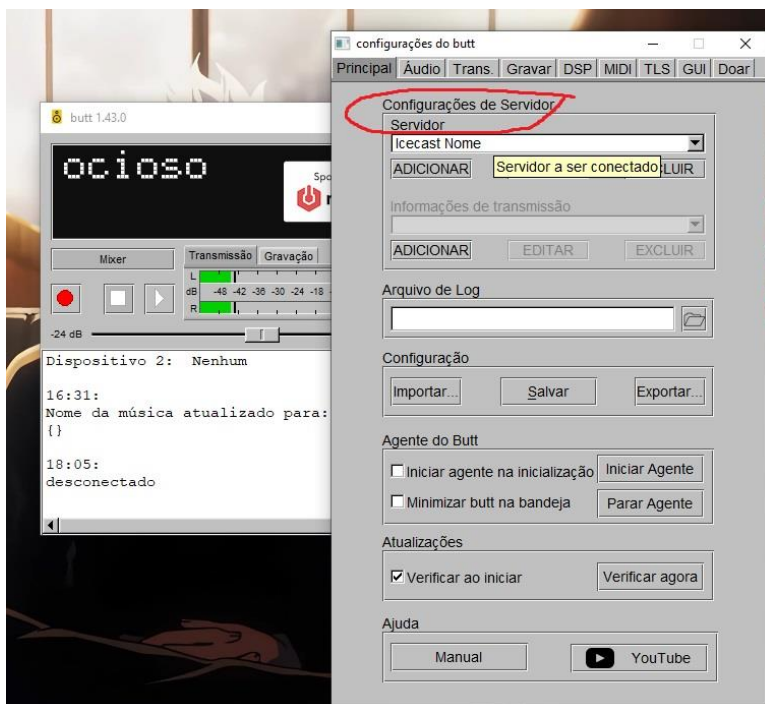
```
1 worker_processes 1;
2
3 events {
4     worker_connections 1024;
5 }
6
7 http {
8     server {
9         listen 80;
10
11         location / {
12             proxy_pass http://icecast:8000;
13             proxy_set_header Host $host;
14             proxy_set_header X-Real-IP $remote_addr;
15             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
16             proxy_set_header X-Forwarded-Proto $scheme;
17         }
18     }
19 }
20
```

- “worker_process 1” define o numero de processos do nginx, pelas pesquisas isso conta a quantidade de CPUs utilizadas
- “events” são os eventos e “worker_connections 1024” são quantidade de conexões simultâneas que cada processo consegue abrir
- “http” Define as configurações para o HTTP como o “server” listado na porta 80 e “location” que define as regras para localizar e tratar as requisições.

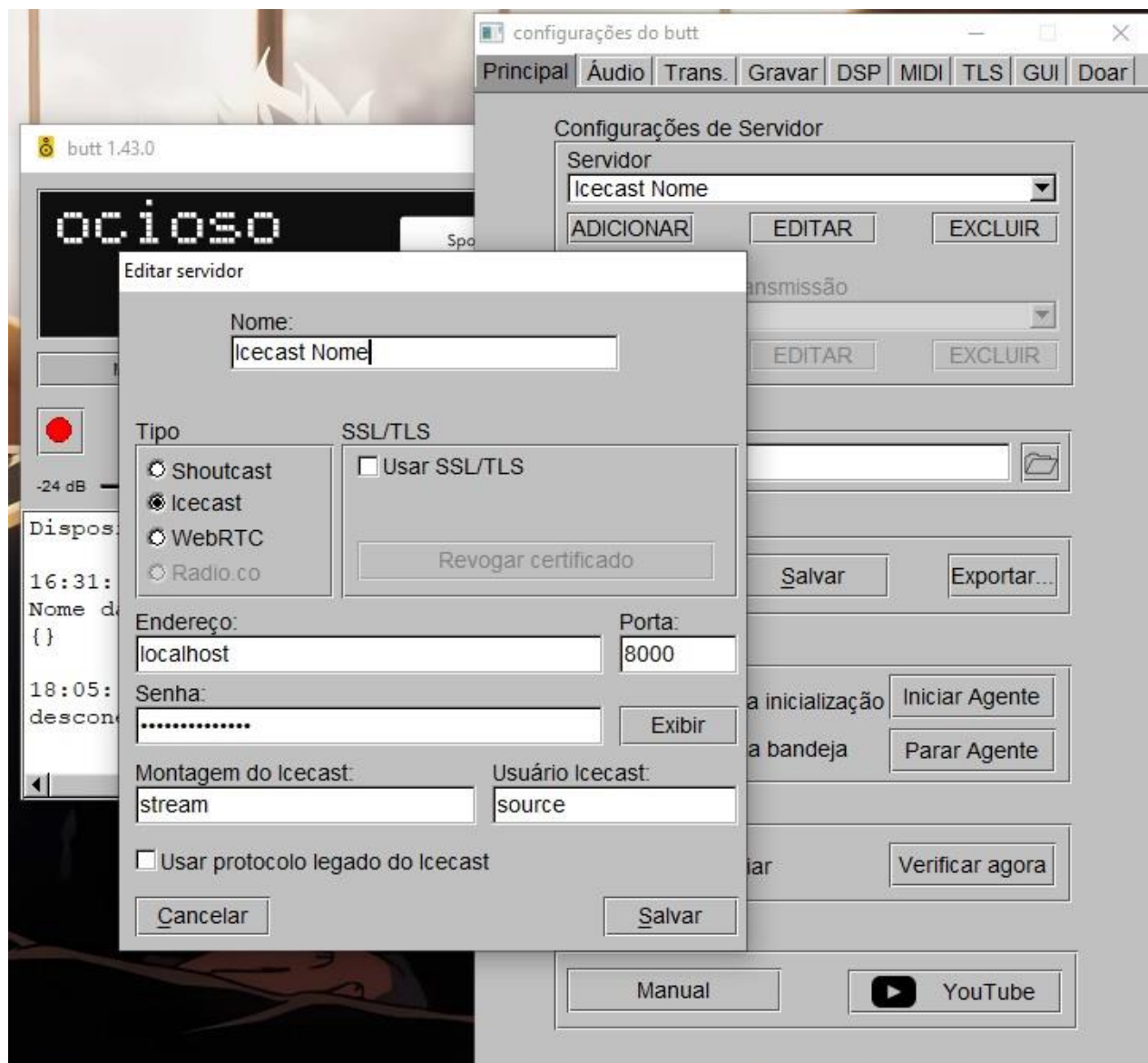
Passo 4 – Instalação e Configuração BUTT

Após finalizar a configuração do docker e dos arquivos, é necessário realizar os teste e configurar o BUTT que é um software que permite transmitir áudio ao vivo.

Pelo site oficial <https://danielnoethen.de/butt/> foi realizado o download do software. E após instalação é necessário clicar em configurações na tela inicial, e no menu principal ter acesso a configuração do servidor icecast.



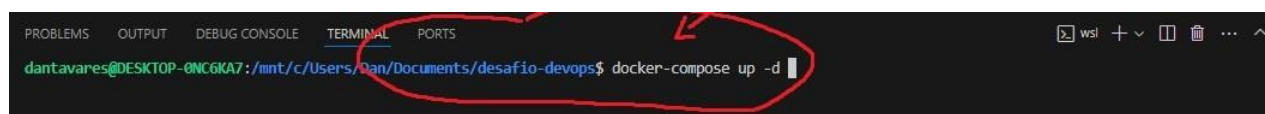
Após esses passos, foi preciso inserir todas as informações já configurada pelos arquivos e docker compose, para que o BUTT consiga se conectar com Icecast.



Passo 5 - Executar as Aplicações com Docker Compose

Por fim após configurar tudo, é necessário executar e realizar os testes para as aplicações para isso precisa conferir docker instalado com comando `docker --version` e caso usar o docker compose para orquestração também `docker-compose --version`.

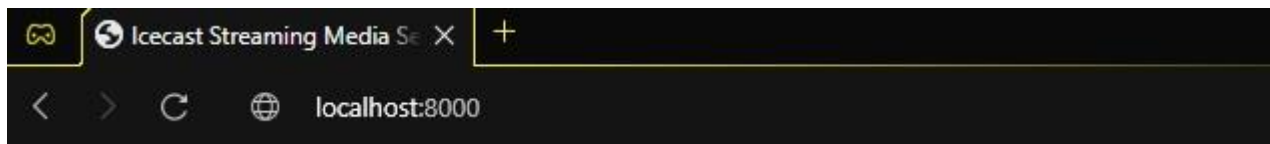
Após essa etapa entrar na pasta em que os arquivos estão e executar com o comando `docker-compose up -d`, onde docker compose é software utilizado, o comando up é para subir, executar e baixar os containers e os arquivos das imagens necessárias para execução das aplicações.



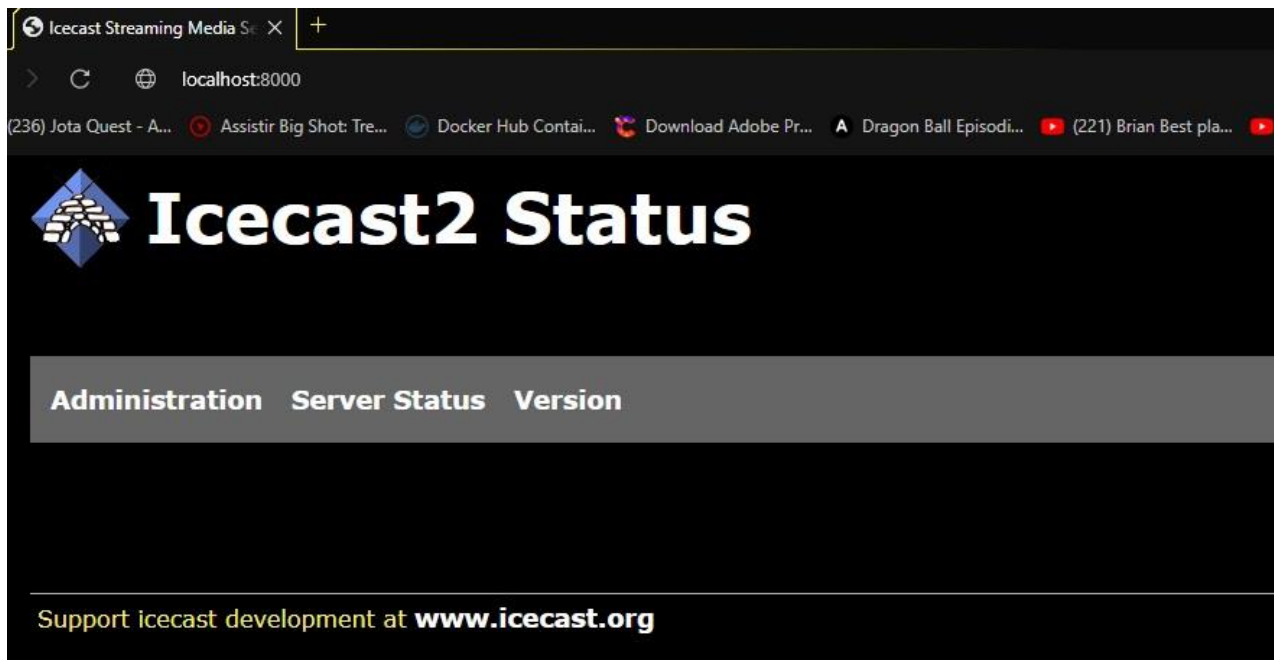
Conferir se o container e a imagem estão sendo executadas com o comando docker-compose ps.

```
[+] Running 3/3
✓ Network desafio-devops_default Created 0.0s
✓ Container icecast Started 0.5s
✓ Container nginx Started 0.8s
dantavares@DESKTOP-0NC6KA7: /mnt/c/Users/Dan/Documents/desafio-devops$ docker-compose ps
NAME                IMAGE                COMMAND                  SERVICE    CREATED      STATUS      PORTS
icecast              noul/icecast:latest  "/start.sh"             icecast    14 seconds ago Up 13 seconds 0.0.0.0:8000->8000/tcp
nginx               nginx:latest         "/docker-entrypoint..." nginx       14 seconds ago Up 12 seconds 0.0.0.0:80->80/tcp
dantavares@DESKTOP-0NC6KA7: /mnt/c/Users/Dan/Documents/desafio-devops$
```

É muito importante testar se esta funcionando a aplicação pelo navegador, como configuramos no servidor local é necessário testar pelo localhost na porta 8000 segue exemplo na imagem:



Na próxima imagem vemos nossa aplicação do Icecast funcionando:



E agora iremos iniciar o aplicativo BUTT e iniciar uma conexão com servidor apertando no botão que parece um play, segue novamente imagem:



E ao fazer isso o BUTT já muda completamente suas informações e ao atualizar nosso navegador Iccast também muda mostrando as informações do stream que foi configurada e aplicação rodando.

The screenshot shows the Icecast2 Status page in a web browser at localhost:8000. The page has a navigation bar with 'Administration', 'Server Status', and 'Version'. The 'Server Status' section is active, showing details for the 'Mount Point /stream'. A red circle highlights the 'Mount Point /stream' section and the 'Current Listeners' field. Overlaid on the page is the BUTT application window, titled 'Conectado a Iccast Nome'. The BUTT window shows a digital display for 'tempo de stream' at 00:00:36, a volume meter, and a list of stream parameters. A red circle highlights the stream parameters list in the BUTT window.

Icecast2 Status

Administration Server Status Version

Mount Point /stream

- Stream Title: no name
- Stream Description: Unspecified description
- Content Type: audio/mpeg
- Mount started: Mon, 26 Aug 2024 21:45:03 +0000
- Bitrate: 128
- Current Listeners: 0
- Peak Listeners: 0
- Stream Genre: various
- Current Song:

Conectado a Iccast Nome

tempo de stream 00:00:36 Ouvintes: 0

Mixer Transmissão Gravação

Codec: mp3
Bitrate: 128kbps
Samplerate: 44100Hz
Mountpoint: stream
User: source
SSL/TLS: não
Dispositivo 1: Dispositivo PCM padrão (padrã
Dispositivo 2: Nenhum

E após testar e tudo estiver funcionado, pause no BUTT seu streaming e logo em seguida cancele a execução dos contêineres e imagens com comando docker-compose down, aqui utilizei o VS Code com terminal WSL com Ubuntu instalado.