

Primeiras intuições

Antes de termos código de verdade rodando nos servidores, vamos brincar um pouco com suas capacidades!

- O que servidores são capazes de fazer?
- Criando uma página web no servidor

O que servidores são capazes de fazer?

TL;DR: Tudo o que um computador consegue fazer. Algumas coisas são muito mais fáceis de fazer neles, inclusive.

Se você chegou até aqui no livro, deve estar com um servidor pronto na mão, mas ainda se perguntando sobre o que servidores são capazes de fazer.

Citarei aqui, portanto, algumas capacidades muito legais que esses computadores conseguem fazer.

1. Servidores que tem um **IP público** são (de uma maneira simples) acessíveis por todos os computadores conectados na internet. Isso possibilita a criação de:
 - Páginas web acessíveis por todos
 - Servidores de arquivos
 - jogos multi-jogador nos quais os clientes se conectam a um servidor específico
2. A habilidade de **distribuir computação** pode ser extremamente útil. Há problemas que um computador sozinho não consegue lidar em tempo hábil, mas milhares de computadores tornam o problema factível.
3. Pode ser que **centralizar informação** seja importante na sua aplicação. Uma única fonte de verdade pode ser importante -- imagina se uma LAN House precisasse gerenciar logins e senhas dos computadores dela individualmente?
4. **Offload de computação** é importante também. Em uma aplicação complexa, pode ser que seja necessário realizar muitos cálculos, e utilizar bastante processamento. Em alguns casos, não é uma boa prática deixar o cliente fazer essa computação, pois este pode estar em um celular ou dispositivo baseado em bateria. Por que não realizar a computação em um servidor e devolver o resultado para o cliente?

Espero que essa lista tenha ajudado a notar alguns usos de um servidor. Vamos realizar alguns

exemplos nas próximas páginas.

Criando uma página web no servidor

Primeiro de tudo, vamos criar uma página web muito simples no servidor, para podermos acessar no próprio navegador!

Considerando que você já realizou os passos anteriores para setup de um servidor e agora tem uma máquina com IP público ao seu dispor, realize os seguintes comandos:

```
# 1. Vá para a sua pasta "home".
cd ~

# 2. Crie uma pasta. Nomeie-a como quiser.
mkdir meu-website

# 3. Entre nessa pasta.
cd meu-website

# 4. Crie um arquivo chamado "index.html", com um HTML bem simples dentro.
echo "<h1>Bem vindo ao meu website! </h1>" > index.html

# 5. Comece um servidor super-simples de python.
# Essa parte requer python3. Caso o servidor não tenha python3 instalado,
# instale como se fosse qualquer outro computador linux:
# em ubuntu/debian, use
# apt install python3
# em redhat/fedora, use
# yum install python3
python3 -m http.server 80
```

Os comandos acima deverão começar um processo escutando na porta 80 (o padrão do HTTP). Utilize `Ctrl+C` para parar o processo. Mas antes, abra o seu navegador favorito e coloque o IP público do seu servidor na barra de endereço.

O resultado deveria ser algo parecido com

Bem vindo ao meu website!

Yay! Tudo funcionando corretamente. Essa é uma forma rápida e suja de servir páginas web, mas não é recomendada como uma solução definitiva. Vamos mostrar como fazer uma solução mais robusta daqui a pouco.

Agora, vamos verificar os processos atados a portas que estão rodando no nosso servidor. Esperamos dois processos: o processo de SSH (que é como conseguimos acessar o nosso servidor remotamente), e o processo de HTTP (o processo em python que começamos agora). Explicarei sobre portas nos próximos capítulos.

Para isso, pare o processo em python com `Ctrl+C` e rode-o em plano de fundo:

```
# Note que estamos ignorando a saída do programa
python -m http.server 80 >/dev/null &
```

Agora, vamos utilizar o comando `netstat` para verificar os processos. Utilize o comando

```
netstat -tln
```

O output do programa deve parecer com isso:

```
root@ubuntu-tutorial:~# netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1208/python3
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      593/systemd-resolve
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      947/sshd
tcp6       0      0 :::22                  :::*                    LISTEN      947/sshd
root@ubuntu-tutorial:~#
```

Quase exatamente o que previmos. No caso, são 3 processos escutando em 4 portas. A primeira linha,

```
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1208/python3
```

é o processo de python que começamos nos passos anteriores. O número do processo é o logo antes do nome do executável: `1208`. para matar esse processo, digite

```
kill 1208
```

A segunda linha é um processo interno da DigitalOcean e pode ser ignorado por enquanto.

As duas últimas linhas são do SSH.

O SSH está escutando tanto em IPv6 quanto IPv4 (mais sobre isso depois).

Limpando o servidor deste exercício

Para eliminar qualquer resíduo desse exercício, primeiro mate o processo em python que foi iniciado no começo dessa página. No meu caso, o número do processo é `1208`, mas isso sempre varia. Utilize o comando `netstat -tln` para descobrir o número.

Por fim, é preciso apagar a pasta que criamos no exercício.

```
cd ~  
rm -r meu-website
```