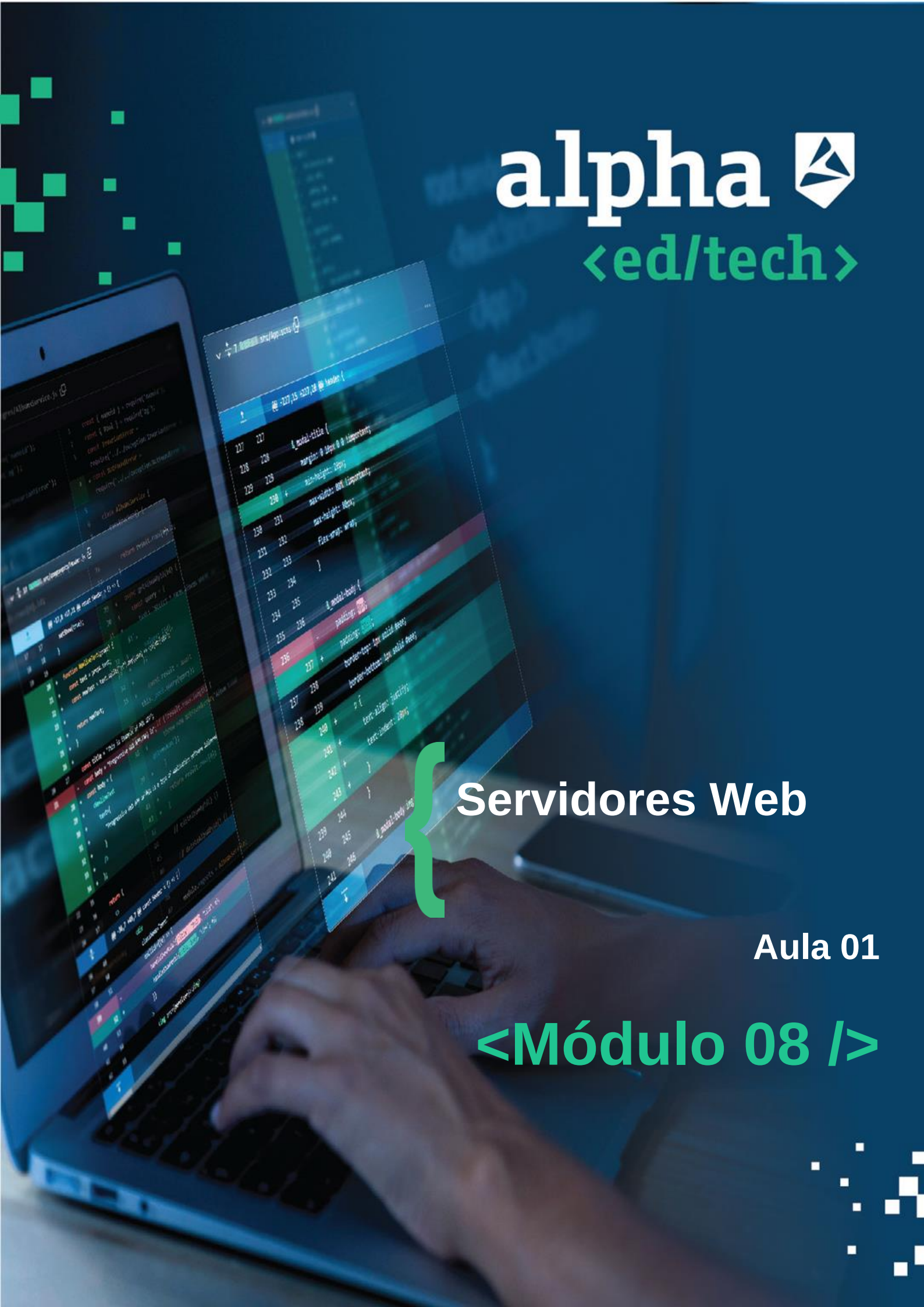

alpha
<ed/tech>
Servidores Web
Aula 01
<Módulo 08 />

Servidores Web

Introdução

Bem-vindos ao módulo de "Servidores Web".
Iniciaremos nossa jornada com uma visão panorâmica dos servidores web, identificando suas funções fundamentais e examinando protocolos essenciais, como o HTTP e HTTPS.
Abordaremos o servidor Apache HTTPD e daremos os primeiros passos na exploração do poderoso Nginx, um servidor web amplamente reconhecido por sua eficiência e escalabilidade.
Compreenderemos por que o Nginx se destaca no cenário da hospedagem web e como ele se integra harmoniosamente com o Node.js, uma plataforma robusta para o desenvolvimento de aplicações do lado do servidor.

Servidores web e seu papel na hospedagem de conteúdo

Os servidores web representam os pilares essenciais que sustentam o vasto universo digital, desempenhando um papel central na entrega eficiente de conteúdo online. Esses componentes vitais são responsáveis por atender às requisições dos usuários, fornecendo páginas da web, aplicativos e recursos multimídia em um piscar de olhos.
Ao explorarmos a definição de servidores web, deparamo-nos com sistemas especializados projetados para lidar com solicitações HTTP e garantir que o conteúdo seja entregue aos usuários finais de maneira rápida e confiável. Compreender a anatomia destes servidores é fundamental para qualquer profissional envolvido no desenvolvimento, implementação ou manutenção de aplicações web.

Porta HTTP (80)

A porta 80 é o canal padrão para comunicação não criptografada entre clientes e servidores web.
Quando um usuário digita "http://www.seusite.com" no navegador, o servidor recebe a solicitação na porta 80 por padrão.
O protocolo HTTP (*Hypertext Transfer Protocol*) é utilizado para transferir dados, como páginas web e imagens, de forma rápida e eficiente.

Porta HTTPS (443)

A porta 443 é reservada para o protocolo HTTPS (*Hypertext Transfer Protocol Secure*), que adiciona uma camada de criptografia SSL/TLS à comunicação entre o navegador e o servidor.
Ao usar a porta 443, os dados transmitidos são protegidos contra interceptação, proporcionando uma experiência mais segura para os usuários.

Redirecionamento de HTTP para HTTPS

É uma prática recomendada configurar redirecionamentos para direcionar automaticamente o tráfego HTTP para HTTPS.
Isso pode ser feito através de configurações no servidor web, garantindo que os usuários se beneficiem da segurança oferecida pelo HTTPS, mesmo que inicialmente acessem o site através da porta 80.

HTTP / HTTPS

Embora a comunicação através da porta 80 seja rápida, os dados não são criptografados, o que significa que podem ser vulneráveis a ataques de interceptação. Se a segurança é uma preocupação, é recomendável considerar a migração para o HTTPS (SSL/TLS).

Apache HTTPD

Introdução

O Apache HTTPD é uma escolha robusta e confiável para muitos cenários, especialmente para sites e aplicações de médio a grande porte, mas é essencial considerar as necessidades específicas do projeto ao escolher um servidor web.

Vantagens

- **Longa História e Maturidade**: O Apache HTTPD (também conhecido como Apache Web Server) tem uma história longa e uma base de código madura, o que contribui para sua estabilidade e confiabilidade. Ele tem sido um dos servidores web mais utilizados por décadas.

- **Grande Comunidade e Suporte:** Sendo de código aberto, o Apache possui uma comunidade ativa de desenvolvedores e usuários, o que significa uma grande quantidade de recursos, módulos e suporte disponíveis. Problemas são frequentemente resolvidos e há uma riqueza de documentação online.

- **Flexibilidade de Configuração:** O Apache é conhecido por sua flexibilidade na configuração. Através do arquivo de configuração principal (httpd.conf) e da utilização de módulos, os administradores têm controle granular sobre o comportamento do servidor.

- **Compatibilidade com Diversos Sistemas Operacionais:** O Apache é compatível com uma variedade de sistemas operacionais, incluindo Linux, Unix, Windows, MacOS, entre outros, o que o torna uma escolha versátil.

Desvantagens

- **Consumo de Recursos:** Em comparação com alguns servidores mais recentes, o Apache pode consumir mais recursos, especialmente em situações de alta carga. Isso pode ser uma consideração importante ao lidar com servidores onde a eficiência de recursos é crítica.

- **Configuração Inicial Complexa:** Para usuários iniciantes, a configuração inicial do Apache pode parecer complexa devido à variedade de opções e configurações disponíveis. Isso pode levar a uma curva de aprendizado mais íngreme.

- **Desempenho em Situações de Muitas Conexões Simultâneas:** Em comparação com servidores otimizados para manipular muitas conexões simultâneas, como o Nginx, o Apache pode não ter o mesmo desempenho em certos cenários de alta concorrência.

- **Processamento Síncrono Pode Afetar o Desempenho:** O modelo de processamento síncrono do Apache, onde cada conexão é tratada por um processo separado, pode impactar o desempenho em cenários com muitas conexões simultâneas, especialmente quando comparado a servidores que utilizam modelos assíncronos.

Apache HTTPD

Instalação

Iremos instalar o servidor Apache httpd no Linux Ubuntu.

Atualize o sistema

Antes de começar, é uma boa prática garantir que o sistema esteja atualizado:

```
sudo apt update
sudo apt upgrade
```

Instale e inicialize o servidor

```
sudo apt install apache2
sudo systemctl start apache2
```

Verifique o status do servidor

Verifique se o Apache está em execução sem erros:

```
sudo systemctl status apache2
```

Configure o Apache para Iniciar na Inicialização

Se você deseja que o Apache seja iniciado automaticamente sempre que o sistema for reiniciado, execute o seguinte comando:

```
sudo systemctl enable apache2
```

Ajuste as configurações de firewall

Configure o firewall (UFW) para permitir o acesso a porta HTTP (80) e HTTPS (443):

```
sudo ufw allow 80
sudo ufw allow 443
```

Configurações adicionais

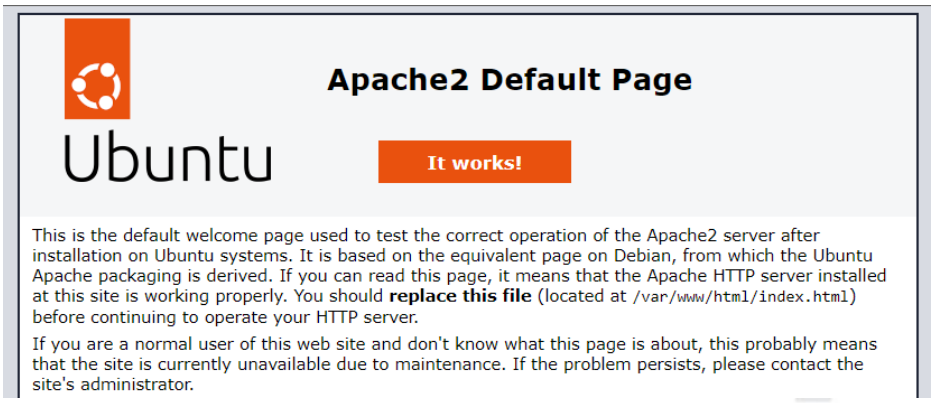
O Apache armazena os arquivos de configuração principais no diretório /etc/apache2/
Para fazer configurações adicionais, você pode editar esses arquivos usando um editor de texto ou uma ferramenta como o nano:

```
sudo nano /etc/apache2/apache2.conf
```

Apache HTTPD

Acesse o servidor pelo seu navegador

Para acessar o servidor, basta colocar o endereço IP do seu servidor no formato "http://<endereço IP>"
Por exemplo: http://192.168.1.10
Se tudo estiver correto, aparecerá algo como na figura abaixo:



Ambiente LAMP

O ambiente de desenvolvimento e produção conhecido como LAMP (Linux, Apache, MySQL, PHP/Python/Perl) são populares, e o uso do Apache como servidor web é uma característica fundamental nesses cenários.

- **Linux como Sistema Operacional:** O LAMP é frequentemente implementado em servidores Linux, aproveitando a estabilidade, segurança e flexibilidade desse sistema operacional.
- **Apache como Servidor Web:** O Apache é a escolha padrão como servidor web no ambiente LAMP, oferecendo uma ampla gama de recursos, extensibilidade e suporte a várias linguagens de programação.
- **MySQL como Sistema de Gerenciamento de Banco de Dados:** O MySQL é frequentemente escolhido como o sistema de gerenciamento de banco de dados devido à sua confiabilidade, desempenho e integração perfeita com o Apache e as linguagens de script.
- **PHP/Python/Perl para Desenvolvimento de Aplicações**: A escolha entre PHP, Python ou Perl como linguagem de programação no LAMP depende dos requisitos do projeto. PHP é especialmente popular para o desenvolvimento web, mas Python e Perl também são utilizados em contextos específicos.

PHP/Python/Perl ou Javascript?

Javascript: O Node.js que utiliza o Javascript como linguagem é conhecido por ser altamente eficiente em termos de desempenho e escalabilidade, especialmente em ambientes orientados a eventos e em situações com muitas conexões simultâneas.

PHP/Python/Perl: Enquanto essas linguagens são eficientes em muitos casos, o Node.js muitas vezes supera em cenários com alto tráfego e interações em tempo real devido à sua natureza assíncrona e não bloqueante.

NGINX

Introdução

O Nginx, pronunciado "engine-x", é um servidor web de código aberto conhecido por sua eficiência e desempenho robusto.
Originalmente desenvolvido para lidar com grandes volumes de tráfego web, o Nginx ganhou popularidade devido à sua arquitetura assíncrona e modular, sendo capaz de gerenciar diversas tarefas simultaneamente de forma eficiente.

Vantagens

- **Desempenho Elevado:** Uma das maiores vantagens do Nginx é seu desempenho excepcional. Ele é projetado para lidar com um grande número de conexões simultâneas de maneira eficiente, tornando-o uma escolha ideal para servidores web de alto tráfego.

- **Baixo Consumo de Recursos:** O Nginx é conhecido por sua eficiência no uso de recursos do sistema. Ele consome menos memória e oferece um excelente desempenho mesmo em ambientes com recursos limitados.

- **Arquitetura Assíncrona:** A arquitetura assíncrona do Nginx permite lidar com solicitações de forma não bloqueante, o que significa que ele pode processar várias solicitações simultaneamente sem a necessidade de alocar um thread para cada conexão.

- **Balanceamento de Carga e Proxy Reverso:** O Nginx é capaz de realizar balanceamento de carga entre servidores, distribuindo as solicitações de maneira equitativa. Além disso, atua como um eficiente proxy reverso, ajudando na otimização do desempenho e na segurança.

- **Configuração Flexível e Modular:** A configuração do Nginx é feita através de arquivos de texto simples, facilitando a manutenção e personalização. Além disso, sua natureza modular permite a extensão de funcionalidades através de módulos adicionais.

Desvantagens

- **Complexidade de Configuração:** Para usuários menos familiarizados, a configuração do Nginx pode parecer complexa em comparação com outros servidores web mais simples. No entanto, a complexidade oferece uma flexibilidade significativa.

- **Menos Suporte para Processamento de Conteúdo Dinâmico:** Embora o Nginx seja excelente para servir conteúdo estático, seu suporte nativo para processamento de conteúdo dinâmico é limitado em comparação com servidores específicos para essa finalidade, como o Apache.

- **Menor Compatibilidade com Algumas Aplicações:** Em casos específicos, algumas aplicações e módulos podem ser mais compatíveis com servidores como o Apache. No entanto, essa desvantagem é geralmente superável com configurações adequadas.

NGINX + Nodejs

O Nginx é uma escolha sólida para muitos cenários, especialmente quando se busca um servidor web eficiente e escalável. Sua arquitetura inovadora e alto desempenho fazem dele uma opção popular para lidar com desafios de tráfego intenso na web.
O uso de Nginx para servir arquivos estáticos aliados ao processamento de requisições de conteúdo dinâmico processados por um servidor Nodejs é a escolha de muitas empresas que buscam a escalabilidade e eficiência em servidores web.

NGINX

Instalação

Iremos instalar o servidor nginx no Linux Ubuntu.

Atualize o sistema

Antes de começar, é uma boa prática garantir que o sistema esteja atualizado:

```
sudo apt update
sudo apt upgrade
```

Instale e inicialize o servidor

```
sudo apt install nginx
sudo systemctl start nginx
```

Verifique o status do servidor

Verifique se o Apache está em execução sem erros:

```
sudo systemctl status nginx
```

Configure o Nginx para Iniciar na Inicialização

Se você deseja que o Nginx seja iniciado automaticamente sempre que o sistema for reiniciado, execute o seguinte comando:

```
sudo systemctl enable nginx
```

Ajuste as configurações de firewall

Configure o firewall (UFW) para permitir o acesso a porta HTTP (80) e HTTPS (443):

```
sudo ufw allow 80
sudo ufw allow 443
```

Configurações adicionais

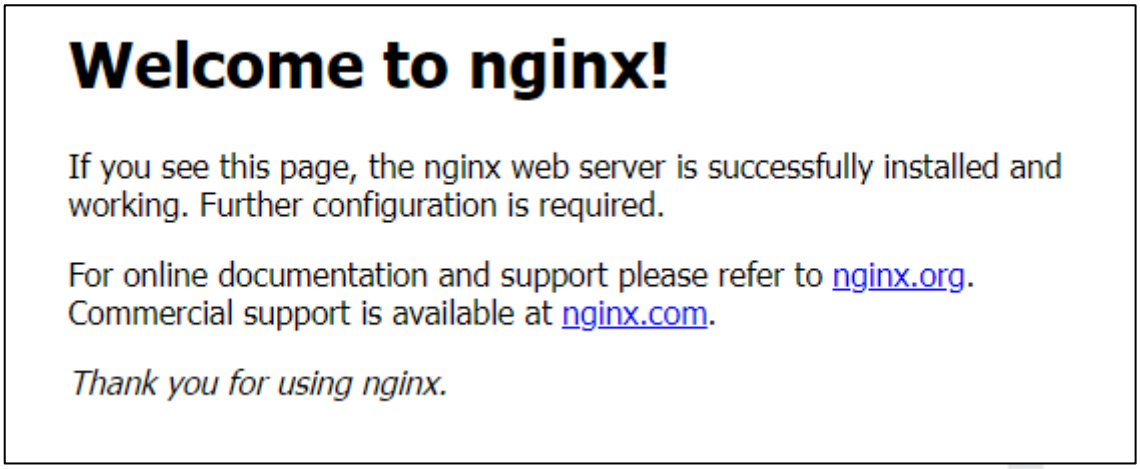
As configurações adicionais do Nginx estão geralmente localizadas no diretório /etc/nginx/ em sistemas baseados em Unix.

```
sudo nano /etc/nginx/nginx.conf
```

NGINX

Acesse o servidor pelo seu navegador

Para acessar o servidor, basta colocar o endereço IP do seu servidor no formato "http://<endereço IP>"
Por exemplo: http://192.168.1.10
Se tudo estiver correto, aparecerá algo como na figura abaixo:



Proxy reverso

Um proxy reverso é um servidor intermediário que recebe solicitações de clientes e as encaminha para servidores de destino.
Ele atua como um intermediário entre os usuários e os servidores de destino, facilitando funções como balanceamento de carga, cache, segurança e otimização de desempenho.
No contexto do Nginx, o uso mais comum de um proxy reverso é para encaminhar solicitações HTTP para servidores web (estáticos e/ou dinâmicos).
O proxy reverso Nginx encaminha solicitações recebidas para servidores web de destino, podendo ser implementado para um único servidor ou para um grupo de servidores (balanceamento de carga).

Balanceamento de Carga

O Nginx pode distribuir as solicitações entre vários servidores de destino, garantindo uma distribuição equitativa da carga de trabalho. Isso melhora a escalabilidade e a disponibilidade do sistema, distribuindo as solicitações de forma eficiente. Exemplo de configuração:

```
upstream backend {
    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com;
}

server {
    location / {
        proxy_pass http://backend;
    }
}
```

NGINX

Cache e Otimização de Desempenho

O Nginx pode atuar como um cache para reduzir o tempo de resposta e melhorar o desempenho. Ele armazena temporariamente as respostas dos servidores de destino e as serve diretamente para solicitações futuras, reduzindo a carga nos servidores de origem.
Exemplo de configuração para cache:

```
proxy_cache_path /path/to/cache levels=1:2 keys_zone=my_cache:10m max_size=10g
                inactive=60m use_temp_path=off;

server {
    location / {
        proxy_pass http://backend;
        proxy_cache my_cache;
        proxy_cache_valid 200 302 10m;
        proxy_cache_valid 404 1m;
    }
}
```

Segurança

O proxy reverso pode melhorar a segurança, ocultando detalhes sobre a infraestrutura interna dos servidores de destino. Ele também pode ser configurado para filtrar solicitações maliciosas, agindo como um ponto de controle de acesso.
Exemplo de configuração para segurança:

```
server {
    location / {
        proxy_pass http://backend;
        proxy_set_header Host $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;
    }
}
```

NGINX

SSL Termination

O proxy reverso Nginx também é comumente usado para terminar o tráfego SSL, aliviando a carga nos servidores de destino e permitindo a configuração de SSL/TLS em um único ponto.
Exemplo de configuração para SSL termination:

```
server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;

    location / {
        proxy_pass http://backend;
        # Configurações SSL adicionais podem ser adicionadas aqui
    }
}
```

Saiba Mais

Recomenda-se a leitura dos documentos disponíveis em:

https://nginx.org/en/docs/

Nodejs

Introdução

Node.js é uma plataforma de desenvolvimento de software construída sobre o motor JavaScript V8 da Google Chrome. Desenvolvido por Ryan Dahl e lançado em 2009, o Node.js permite a execução de código JavaScript no lado do servidor, proporcionando assim um ambiente altamente eficiente para a construção de aplicações web escaláveis e de alto desempenho.

I/O não bloqueante

Uma das características distintivas do Node.js é a sua arquitetura orientada a eventos, que utiliza um modelo de I/O não bloqueante. Isso significa que as operações de entrada e saída, como leitura de arquivos ou consultas a bancos de dados, são realizadas de forma assíncrona, permitindo que o servidor continue a lidar com outras solicitações enquanto aguarda a conclusão dessas operações. Esse modelo torna o Node.js particularmente adequado para aplicações em tempo real, como chats, jogos online e streaming de dados.

Vantagens

- **Desenvolvimento Rápido**: O uso do JavaScript tanto no frontend quanto no backend permite um desenvolvimento mais coeso e rápido.
- **Arquitetura Orientada a Eventos:** O modelo de I/O não bloqueante e a arquitetura orientada a eventos tornam o Node.js adequado para aplicações em tempo real e de alta concorrência.
- **Eficiência e Desempenho:** A engine V8 da Google Chrome, que está por trás do Node.js, é altamente eficiente, proporcionando um desempenho rápido e escalabilidade.
- **Ecossistema npm:** O npm é um dos maiores repositórios de pacotes do mundo, facilitando o compartilhamento, instalação e gerenciamento de dependências.
- **Compartilhamento de Código:** O uso do JavaScript permite que os desenvolvedores compartilhem código entre o frontend e o backend, facilitando a manutenção e reduzindo a duplicação.

Desvantagens

- **Single-threaded:** O Node.js é single-threaded, o que significa que processos intensivos em CPU podem impactar o desempenho, pois não aproveitam totalmente os sistemas multi-core.
- **Callback Hell (Inferno dos Callbacks):** O aninhamento excessivo de callbacks pode levar à criação de código difícil de ler e manter, conhecido como "Callback Hell". No entanto, as Promises e async/await ajudam a mitigar esse problema.
- **Menos Adequado para Tarefas Computacionais Intensivas:** Devido à natureza single-threaded, o Node.js pode não ser a melhor escolha para tarefas computacionais intensivas que exigem muita CPU.
- **Maturidade das Bibliotecas:** Algumas bibliotecas podem não ser tão maduras ou estáveis quanto em outras plataformas mais estabelecidas.
- **Problemas de Gerenciamento de Dependências:** Dependendo do projeto, o gerenciamento de dependências pode se tornar complexo, especialmente quando há uma grande quantidade de pacotes.

Nodejs

Instalação

Iremos instalar o nodejs no Linux Ubuntu.

Atualize o sistema

Antes de começar, é uma boa prática garantir que o sistema esteja atualizado:

```
sudo apt update
sudo apt upgrade
```

Instale o servidor nodejs e o gerenciador de pacotes npm

Antes de começar, é uma boa prática garantir que o sistema esteja atualizado:

```
sudo apt install nodejs
sudo apt install npm
```

Verifique a versão

Verifique a versão do servidor nodejs e do npm instalados:

```
node –v
npm -v
```

Mais do que um servidor web

O Node.js é mais do que apenas um servidor web porque é uma plataforma de desenvolvimento que permite a execução de código JavaScript no lado do servidor.
Enquanto muitas pessoas inicialmente associam Node.js à criação de servidores web, suas capacidades vão além dessa função específica.

- ***JavaScript no Lado do Servidor**:Uma das características mais distintivas do Node.js é a capacidade de executar código JavaScript no lado do servidor. Isso unifica o desenvolvimento, permitindo que os desenvolvedores usem a mesma linguagem de programação tanto no frontend quanto no backend.*
- ***Ambiente de Tempo de Execução**: Node.js é um ambiente de tempo de execução que permite a execução de scripts do lado do servidor. Ele não é limitado a servidores web; você pode usá-lo para automação de tarefas, scripts de linha de comando e construção de aplicativos de várias finalidades.*
- ***Manipulação de I/O Não Bloqueante:** O Node.js é construído com uma arquitetura orientada a eventos e I/O não bloqueante. Isso o torna adequado para manipular operações assíncronas e lidar eficientemente com uma grande quantidade de conexões simultâneas. Essa característica é valiosa para aplicações em tempo real, como chats, jogos online e streaming de dados.*
- ***Desenvolvimento de Aplicações em Tempo Real:** Além de servidores web, o Node.js é amplamente utilizado para desenvolver aplicações em tempo real, como salas de chat, aplicativos de mensagens, colaboração em tempo real e jogos multiplayer online. A arquitetura orientada a eventos é particularmente útil para lidar com atualizações em tempo real.*

Nodejs

Instalação sem o uso do apt

Iremos agora instalar o nodejs no Linux Ubuntu sem a utilização do apt ou apt-get.

Acesse o Site oficial do nodejs

Visite o site oficial e baixe a versão LTS do nodejs: https://nodejs.org/en/download
Para o ubuntu, pegar o link do arquivo Linux binary 64 bits e baixá-lo diretamente:

```
wget https://nodejs.org/dist/<versão>/node-<versão>-linux-x64.tar.gz
```

Descompacte utilizando o tar

```
tar –xvf node-<versão>-linux-x64.tar.gz
```

Mova os binários para a pasta desejada

Por exemplo:

```
mkdir ~/nodejs
mv node-<versão>-linux-x64/* ~/nodejs/
```

Adicione o Caminho para o nodejs ao “PATH”

Normalmente feito adicionando no arquivo .bashrc

```
echo ‘export PATH=~/nodejs/bin:$PATH’ >> ~/.bashrc
source ~/.bashrc
```

Verifique a versão

Verifique a versão do servidor nodejs e do npm instalados:

```
node –v
npm -v
```

Versão mais recente?

Em alguns casos, você pode precisar de uma versão mais recente do Node.js que ainda não esteja disponível nos repositórios do sistema.
A instalação manual permite que você acesse as versões mais recentes diretamente do site oficial.