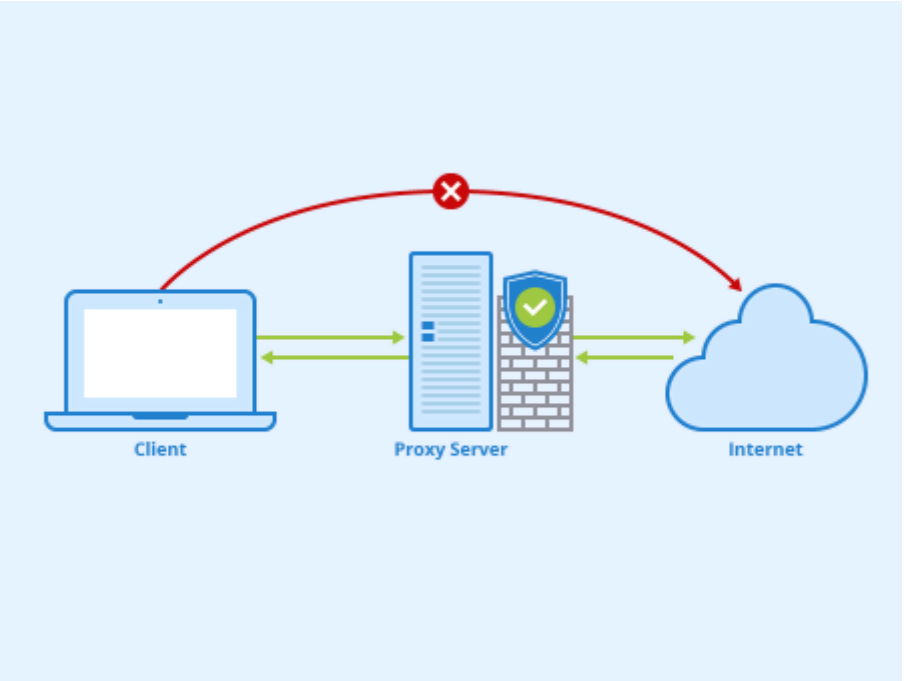
Monitores:

- Davi Souza de Luna
- Adryan Reis Nascimento

O que é um servidor Proxy

É um servidor que fica na **frente** de um número de máquinas **clientes.** Esse servidor gerencia toda a comunicação que será feita com a internet, sendo ele um **intermediário** entre o cliente e a internet, como é demonstrado na imagem abaixo

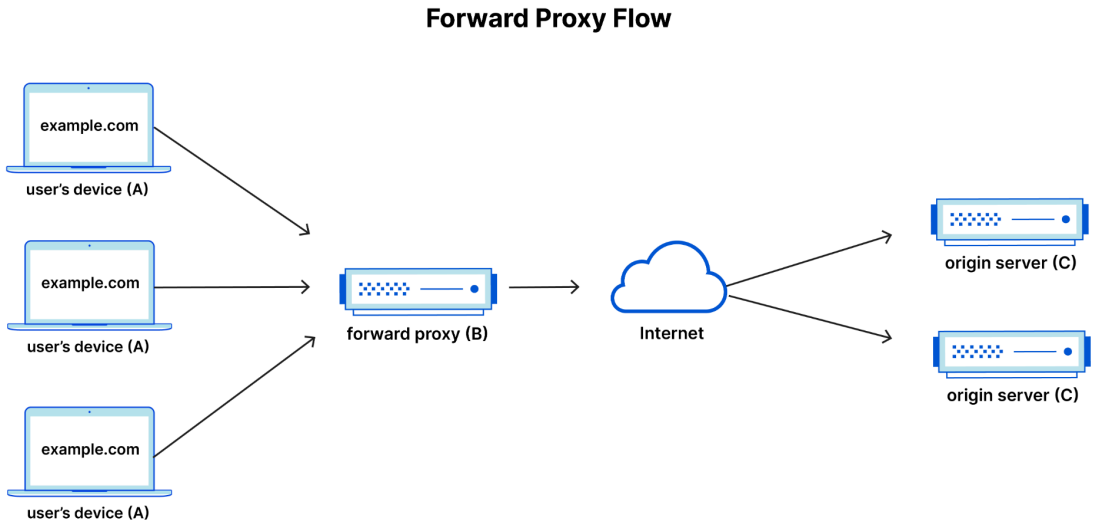


As aplicações de um servidor proxy são diversas, sendo algumas:

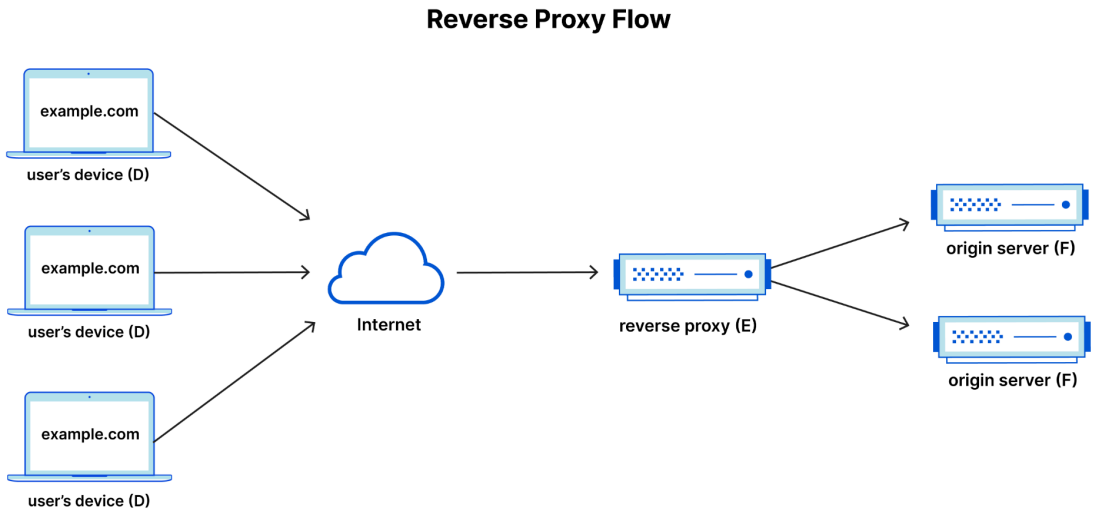
- **Evitar restrições estatais**: Alguns governos tem firewalls que impedem o acesso à determinados conteúdos na internet. O proxy pode contornar isso, pois o cliente não vai se comunicar diretamente com o servidor que hospeda o conteúdo, mas sim com o proxy.
- **Bloqueio de alguns conteúdos**: Em instituições públicas ou escolares a utilização de proxy para evitar o acesso dos usuários a sites adultos, por exemplo. Em suma, o cliente não vai ter acesso a alguns conteúdos que estejam dentro da internet.
- **“Anonimato”**: Como o servidor proxy que age como intermediário, somente o endereço IP dele que será visível à rede, colocando assim mais uma camada de segurança ao cliente.

Proxy vs Proxy reverso

Basta você pensar em como funciona o **fluxo** de comunicação. Num proxy a comunicação vai do cliente ao proxy, que encaminha a um host externo. Num proxy reverso a comunicação é invertida, sendo de qualquer host externo para o proxy, que então encaminha para o servidor de origem. É bem sutil, mas basta se atentar ao fluxo da comunicação. Segue abaixo uma imagem contendo a comunicação do proxy de encaminhamento:



Segue abaixo uma imagem contendo a comunicação de um proxy reverso:



Benefícios do Proxy reverso

- **Balanceamento de carga**: Como o servidor de proxy reverso gerencia a comunicação, nele pode existir algum tipo de regra que gerencie a carga de trabalho dada a um servidor, evitando que um único servidor fique sobrecarregado. Caso um servidor falhe, outros servidores podem tomar essa responsabilidade, evitando que o serviço fique inativo.
- **Proteção contra DDoS:** Como o servidor Proxy que vai receber a solicitação, os servidores que provêm os serviços não irão ser afetados, somente o proxy receberá esse ataque, que pode ser evitado com algumas medidas.
- **Balanceamento de carga global de servidores:** Quando um cliente tenta solicitar algo, o servidor proxy vai redirecionar o cliente para o servidor que estiver mais próximo dele.
- **Cache**: O servidor proxy pode armazenar os conteúdos mais frequentes requisitados pelo cliente temporariamente, assim mitigando a carga sobre os servidores e oferecendo uma comunicação mais rápida
- **Criptografia:** Como a criptografia/descriptografia(SSL ou TLS) tem um custo computacional relevante para cada servidor, o servidor proxy pode ficar responsável por essa parte, mitigando a carga de trabalho para os outros servidores

Proxy reverso simples no Docker

O experimento que será feito aqui, será o de mostrar como funciona a comunicação do proxy reverso na prática.
Quando se cria um container no docker, ele automaticamente é conectado à rede padrão do mesmo. Pode-se alterar isso criando outras redes, mas como fins de aprendizado nesse exemplo não iremos fazer isso. Como visto na disciplina de redes de computadores, redes diferentes **não se comunicam,** portanto o HOST(máquina real) não vai conseguir executar um comando **ping** para a máquina docker, pois ela está numa rede distinta e interna do docker[1].

Cenário do experimento:

- Um container Alpine com o serviço nginx. Esse container vai agir como proxy reverso. Quando fizermos uma requisição GET para o endereço / iremos ser redirecionados para outro container
- Outro container Alpine com o serviço Apache. Esse container vai ter a porta exposta na rede interna do docker(que não vai se comunicar com o HOST, apenas com o proxy). É importante expor a porta no apache também para que o HTML esteja disponível.
- Um HTML que será a **resposta** para a solicitação GET /

[1] Caso você queira fazer com que eles se comuniquem sem proxy, basta criar uma rede bridge

O que deve ser criado:

- Um Dockerfile para o servidor **apache**
- Um Dockerfile para o servidor proxy(**nginx**)
- O arquivo de configuração do proxy reverso(**nginx.conf**)
- Um arquivo HTML(**index.html**)
- Um arquivo docker compose para a orquestração. Pode ser criada outras redes para segmentar os serviços. Mas nesse caso a rede que os containers se conversam é a **rede padrão** do docker

Link da Implementação

Considerações acerca dos comandos:

É fundamental consultar a documentação[2] para entender quais comandos devem ser utilizados. O mais importante é compreender os conceitos básicos; a partir disso, o uso de outras ferramentas de consulta será mais eficaz, e a dependência delas será reduzida nesse contexto.

[2] https://docs.docker.com/

Referências:

https://www.redhat.com/pt-br/topics/containers/what-is-docker
https://www.cloudflare.com/pt-br/learning/cdn/glossary/reverse-proxy/
https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/
https://docs.docker.com/reference/dockerfile/
https://docs.docker.com/reference/cli/docker/