**JENKINS REVERS PROXY**

**What is reverse proxy?**

A reverse proxy is a server that sits between client devices and a web server. Unlike a forward proxy, which serves as an intermediary for clients accessing external servers, a reverse proxy acts on behalf of the server and manages incoming requests from clients.

**Why we need in jenkins?**

In the context of Jenkins, a reverse proxy can be particularly useful for several reasons:

 **Enhanced Security**: A reverse proxy can add an extra layer of security by shielding Jenkins from direct exposure to the internet. It can manage SSL/TLS encryption and decryption, ensuring that data transmitted between clients and Jenkins is secure. Additionally, it can provide a more secure way to manage authentication and authorization.

 **Load Balancing**: If you have multiple Jenkins instances or want to distribute the load across multiple servers, a reverse proxy can help balance the traffic. It can direct requests to different Jenkins nodes or instances based on various criteria, such as load or availability.

 **Simplified Access**: A reverse proxy can simplify access to Jenkins by providing a single URL or domain name. It can also handle URL rewriting, allowing you to present Jenkins under a specific path (e.g., https://yourcompany.com/jenkins) rather than exposing the internal Jenkins URL.

 **SSL Termination**: Handling SSL/TLS encryption and decryption at the reverse proxy level can offload this task from Jenkins itself. This is particularly useful for Jenkins installations that need to serve secure connections but where Jenkins does not natively handle SSL/TLS.

 **Centralized Logging and Monitoring**: A reverse proxy can aggregate logs and monitor traffic across multiple applications and services, including Jenkins. This centralization can simplify monitoring and troubleshooting.

 **Caching**: While not always applicable for Jenkins, a reverse proxy can cache certain types of responses or static resources. In Jenkins, this might help if you have resources that are accessed frequently and can be cached to improve performance.

 **Custom Domains and Routing**: If Jenkins is part of a larger suite of tools or services, a reverse proxy can manage custom domains and routing rules. This can help in integrating Jenkins seamlessly into an existing infrastructure and aligning it with organizational domain strategies.

**Configuration Examples by Server Type**

Jenkins works with many different reverse proxies. This section provides examples for specific reverse proxies, though much of the information also applies to other reverse proxies.

* [Running Jenkins with Apache](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-apache)
* [Running Jenkins with Nginx](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-nginx)
* [Running Jenkins with Lighttpd](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-lighttpd)
* [Running Jenkins with HAProxy](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-haproxy)
* [Running Jenkins with Pomerium](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-pomerium)
* [Running Jenkins with Squid](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-squid)
* [Running Jenkins with IIS](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-iis)
* [Running Jenkins with iptables](https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-iptables)

**Setup proxy using nginx:**

Take redhat or amazon linux machine. Open the ports 80 for nginx 8080 for jenkins.

Install the nginx:

$ sudo yum install -y nginx

$ sudo systemctl start nginx

$ sudo systemctl status nginx

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

step 1: sudo vi /etc/nginx/sites-available/mrcloudbook.online

eg: sudo vi /etc/nginx/sites-available/gskdevops.online

step2: cd /etc/nginx

step3: cd conf.d

step4: vi jenkins.conf

Paste the below (replace your domain)

upstream jenkins{

server 127.0.0.1:8080;

}

server{

listen 80;

server\_name ur dns name;

access\_log /var/log/nginx/jenkins.access.log;

error\_log /var/log/nginx/jenkins.error.log;

proxy\_buffers 16 64k;

proxy\_buffer\_size 128k;

location / {

proxy\_pass http://jenkins;

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504;

proxy\_redirect off;

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 https;

}

}

Next, test to make sure that there are no syntax errors in any of your Nginx files:

Step5: sudo nginx -t

If there aren’t any problems, restart Nginx to enable your changes:

Step6: sudo systemctl restart nginx

Nginx should now be serving Jenkins from your domain name. You can test this by navigating to http://your\_domain