1. What is the advantage of using a “reverse proxy server”?
    - Creates a single point of access to your file transfer services.
    - Simplifies access control.
    - Reduces risks to a sensitive data.
    - Brings down capital and operational expenses.
    - Enables load balancing and failover.
2. Why and where Nginx is a better choice than apache.

    NGINX is better with static content, but offers little extra performance for dynamic content. Meanwhile, NGINX stands out because of some of its more advanced features (media streaming, reverse proxying for non-HTTP protocols), along with its commercial support and training. Owners of websites that attract a lot of visitors and which will be serving lots of static content and/or streaming media will veer towards NGINX. It doesn’t creates a new process for each request.

3. What are worker nodes and worker connections? How to calculate the max server capacity using the above two?

    Worker Processes:

    It depends on how many cores our CPU have.

    Worker Connections:

    The worker_connections command tells our worker processes how many people can simultaneously be served by Nginx. The default value is 768; however, considering that every browser usually opens up at least 2 connections/server, this number can half.

    Max Server Capacity= No. of worker processes * No. of worker connections.

4. From what directory will NGINX automatically load server (virtual host) configurations when using the default /etc/nginx/nginx.conf configuration?

    /etc/nginx/conf.d and /etc/nginx/sites-enabled directories

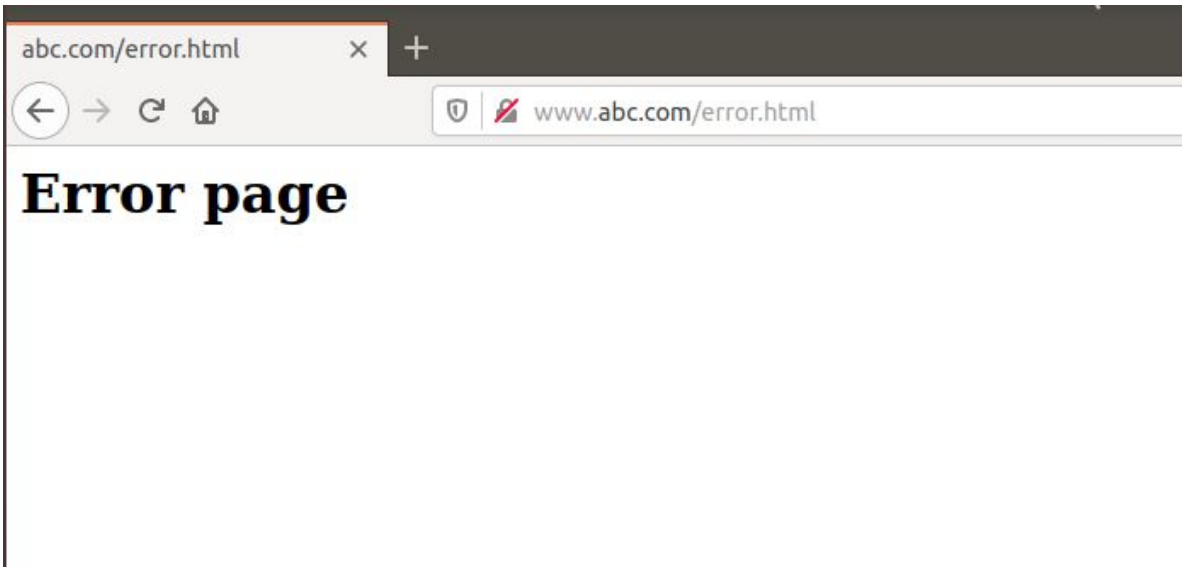
5. How to configure different log_format for different “location” block/directive?

```
##
# Logging Settings
##

log_format custom ' $remote_user [$time_local]'
                        '"$request" $status $body_bytes_sent '
                        '"$http_referer" "$http_user_agent"'
                        '"$http_x_forwarded_for" $request_id';
access_log /var/log/nginx/access.log custom;
error_log /var/log/nginx/error.log;
```

```
rishabh@rishabh:nginx $ sudo tail -n 2 access.log
127.0.0.1 - - [17/Feb/2020:23:34:29 +0530]"GET /favicon.ico HTTP/1.1" 302 170 "https://www.abc.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36""-" ef235333f1347f9c41d43b2acdff29ad
127.0.0.1 - - [17/Feb/2020:23:34:29 +0530]"GET /error.html HTTP/1.1" 200 82 "https://www.abc.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36""-" d6c1578f7861f2a57ce9262647bbad47
rishabh@rishabh:nginx $ sudo vim /etc/nginx/nginx.conf
rishabh@rishabh:nginx $ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
rishabh@rishabh:nginx $ sudo service nginx restart
rishabh@rishabh:nginx $ sudo tail -n 2 access.log
 - [17/Feb/2020:23:37:03 +0530]"GET /favicon.ico HTTP/1.1" 302 170 "https://www.abc.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36""-" 33de509f20fae75c0aecd01b321fe914
 - [17/Feb/2020:23:37:03 +0530]"GET /error.html HTTP/1.1" 200 82 "https://www.abc.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36""-" da213ae946525ed206670d7996448893
rishabh@rishabh:nginx $
```

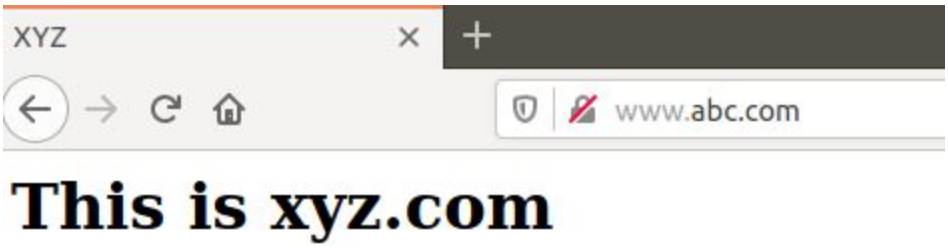
6. Host a site ABC.COM
    1. Create an index page and a fail-safe page. If a page for URI is not available, the fail-safe page is served.

```
server{
        listen 80;
        root /var/www/html;
        index abc.html;
        error_page 404 error.html;
        server_name abc.com www.abc.com;
}
```



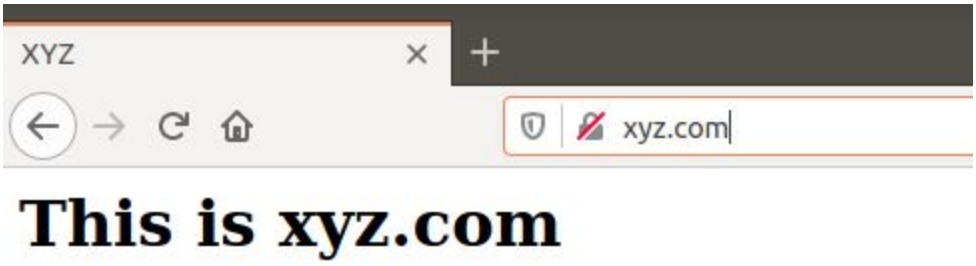
2. proxy pass to a website xyz.com on a particular URI.

```
server{
        listen 80;
#        root /var/www/html;
#        index abc.html;
        error_page 404 error.html;
        server_name abc.com www.abc.com;
        location / {
                proxy_pass http://xyz.com;
        }
}

~
```



3. redirect to above URI on /redirect/

```
server{
        listen 80;
        root /var/www/html;
        index abc.html;
        error_page 404 error.html;
        server_name abc.com www.abc.com;
        location / {
                rewrite ^/redirect$ http://xyz.com;
        }
}

~
~
```



4. perform an HTTP to HTTPS redirection including non-www to www redirection.

```
server{
        listen 80;
        server_name abc.com;
        return 302 https://www.abc.com;
}
server{
        listen 443 ssl;
        root /var/www/html;
        index abc.html;
        error_page 404 error.html;
        server_name www.abc.com;
        ssl_certificate /etc/nginx/ssl/nginx.pem;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;
}

~
~
~
~
```

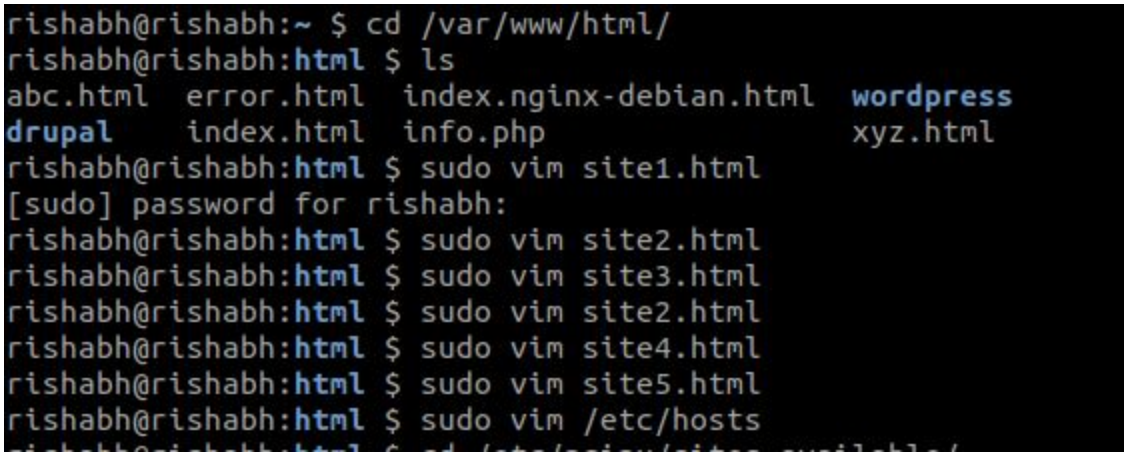
5. Allow access to a set of particular IPs on a location block and return 405 to other IPs no matter if the page in that location exists.

```
server{
        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 error.html;
        location / {
                if ($remote_addr != "127.0.0.1") {
                        return 405;
                }
        }
}

~
~
```

```
vaibhav@vaibhav:/etc/nginx/sites-available$ curl 10.1.227.104
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>
```

6. Place your images at /var/www/html/images. Only accept jpg/png/jpeg. Discard rest

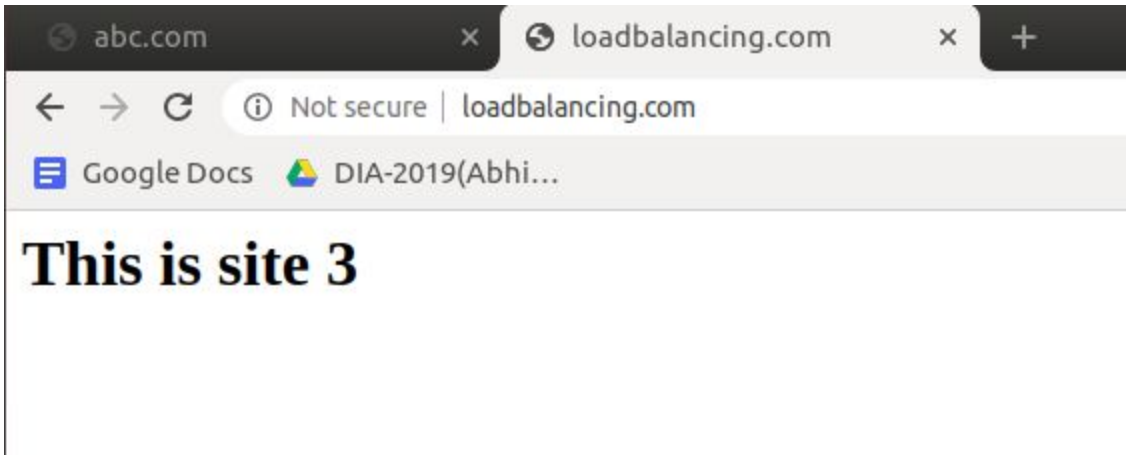
7. Create a load balancer with 5 backends. Explain different types of load balancing methods.

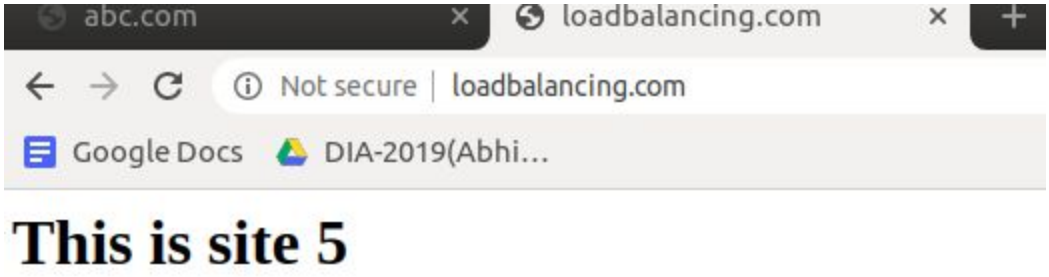


```
upstream balance{
    server 127.0.0.1:82;
        server 127.0.0.1:83;
        server 127.0.0.1:84;
        server 127.0.0.1:85;
        server 127.0.0.1:86;
}
server{
    listen 80;
    server_name loadbalancing.com;
    location /{
        proxy_pass http://balance;
    }
}
~
```

```
127.0.0.1       abc.com localhost www.abc.com xyz.com loadbalancing.com
127.0.1.1       rishabh

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

```
server{
    listen 82;
    root /var/www/html;
    index site1.html;
    server_name 127.0.0.1;
}
server{
        listen 83;
        root /var/www/html;
        index site2.html;
        server_name 127.0.0.1;
}
server{
        listen 84;
        root /var/www/html;
        index site3.html;
        server_name 127.0.0.1;
}
server{
        listen 85;
        root /var/www/html;
        index site4.html;
        server_name 127.0.0.1;
}
server{
        listen 86;
        root /var/www/html;
        index site5.html;
        server_name 127.0.0.1;
}
~
~
```





Types of load balancing:

- **Round Robin** – Requests are distributed across the group of servers sequentially.
- **Least Connections** – A new request is sent to the server with the fewest current connections to clients. The relative computing capacity of each server is factored into determining which one has the least connections.
- **Least Time** – Sends requests to the server selected by a formula that combines the fastest response time and fewest active connections.
- **Hash** – Distributes requests based on a key you define, such as the client IP address or the request URL.
- **IP Hash** – The IP address of the client is used to determine which server receives the request.
- **Random with Two Choices** – Picks two servers at random and sends the request to the one that is selected by then applying the Least Connections algorithm.

8. Setup Basic Auth (Popup asking for username and password) in a particular location block. (The Basic Auth should not be asked for TTN IP)

```
server{
        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 error.html;
        location = /admin.html{
                auth_basic "login_required";
                auth_basic_user_file /etc/nginx/.htpasswd;
        }
}
```