MIDDLEWARE-- Part1

Problem: Install NGINX on RHEL 8 using yum repo  
Root Cause: NGINX is not installed by default

Resolution Steps

1. Install NGINX from the default repo  
   yum install -y nginx

Notes

* Requires root access
* Installs NGINX from RHEL AppStream or EPEL if enabled

Sample Output  
Last metadata expiration check: 0:00:10 ago on Wed Jul 02 11:00:00 +0800.  
Dependencies resolved.  
Install 1 Package  
Total download size: 1.2 M  
Installed size: 3.1 M  
Installed: nginx-1.24.0-1.el8.x86\_64

Problem: Install NGINX on RHEL 8 using official NGINX yum repo  
Root Cause: Default repo may not have the latest or desired NGINX version

Resolution Steps

1. Create /etc/yum.repos.d/nginx.repo with:  
   [nginx-stable]  
   name=nginx stable repo  
   baseurl=https://nginx.org/packages/rhel/8/$basearch/  
   gpgcheck=1  
   enabled=1  
   gpgkey=https://nginx.org/keys/nginx\_signing.key
2. Install NGINX  
   yum install -y nginx

Notes

* Requires root access
* Installs the latest stable NGINX from the official NGINX repository

Sample Output  
Installed: nginx-1.24.0-1.el8.ngx.x86\_64

Problem: Install NGINX in a custom location /data/nginx (from source)  
Root Cause: Need NGINX in a non-default directory, not managed by RPM

Resolution Steps

1. Install dependencies  
   yum install -y gcc make pcre-devel zlib-devel openssl-devel
2. Download and extract source  
   curl -O http://nginx.org/download/nginx-1.24.0.tar.gz  
   tar zxvf nginx-1.24.0.tar.gz  
   cd nginx-1.24.0
3. Configure build to install to /data/nginx  
   ./configure --prefix=/data/nginx --conf-path=/data/nginx/conf/nginx.conf --sbin-path=/data/nginx/sbin/nginx --error-log-path=/data/nginx/logs/error.log --http-log-path=/data/nginx/logs/access.log
4. Compile and install  
   make  
   make install

Notes

* Requires root access to install dependencies and write to /data
* Adjust version and installation paths as needed
* This NGINX instance is managed manually (not by yum or systemctl)

Sample Output  
configuration summary:  
…  
prefix=/data/nginx  
…  
nginx binary is /data/nginx/sbin/nginx

Problem: Configure logging for NGINX (any install method)  
Root Cause: Need to ensure logs are stored in a known, persistent location

Resolution Steps

1. Edit the NGINX configuration file (e.g., /etc/nginx/nginx.conf or /data/nginx/conf/nginx.conf)
2. Set the log file locations:  
   access\_log /data/nginx/logs/access.log main;  
   error\_log /data/nginx/logs/error.log warn;

Notes

* Paths should be writable by the user running NGINX
* Log levels can be adjusted (e.g., error, warn, info, debug)

Sample Output  
Log files will be created at:  
/data/nginx/logs/access.log  
/data/nginx/logs/error.log

Problem: Start or stop NGINX as a service on RHEL 8  
Root Cause: Need to control the NGINX web server process for maintenance, configuration changes, or troubleshooting

Resolution Steps

1. Start the NGINX service  
   systemctl start nginx
2. Stop the NGINX service  
   systemctl stop nginx
3. Restart NGINX (use after config changes)  
   systemctl restart nginx
4. Reload NGINX (reload config without dropping connections)  
   systemctl reload nginx
5. Check the current status of the NGINX service  
   systemctl status nginx

Notes

* Root access required for all systemctl commands
* Use restart after editing the configuration file, or reload if you don’t want to disrupt active connections
* If using a custom-built NGINX (not installed via yum), these commands won’t apply—use the binary directly (e.g., /data/nginx/sbin/nginx -s reload)

Sample Output

systemctl start nginx  
(No output if successful; use systemctl status nginx to confirm)

systemctl status nginx  
● nginx.service - The nginx HTTP and reverse proxy server  
Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)  
Active: active (running) since Wed 2025-07-02 14:00:12 +0800; 2min ago

Problem: Configure SSL for NGINX  
Root Cause: Need to serve web traffic over HTTPS for security and compliance

Resolution Steps

1. Obtain or generate SSL certificate and private key  
   Example (self-signed):  
   openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
2. Edit the NGINX config file (e.g., /etc/nginx/conf.d/ssl.conf) and add:  
   server {  
   listen 443 ssl;  
   server\_name your.domain.com;  
   ssl\_certificate /etc/nginx/ssl/server.crt;  
   ssl\_certificate\_key /etc/nginx/ssl/server.key;  
   ssl\_protocols TLSv1.2 TLSv1.3;  
   ssl\_ciphers HIGH:!aNULL:!MD5;  
   }
3. (Optional) Redirect HTTP to HTTPS:  
   server {  
   listen 80;  
   server\_name your.domain.com;  
   return 301 https://$host$request\_uri;  
   }
4. Test the configuration  
   nginx -t
5. Reload NGINX to apply changes  
   systemctl reload nginx

Notes

* Root access required for certificate creation, config edits, and service reload
* Use real CA certificates in production
* Replace your.domain.com with your actual domain
* Paths to certificate and key must match your config

Sample Output

1. openssl req -x509 …  
   Generating a RSA private key  
   …  
   writing new private key to '/etc/nginx/ssl/server.key'
2. nginx -t  
   nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
   nginx: configuration file /etc/nginx/nginx.conf test is successful
3. systemctl reload nginx  
   (No output if successful)

Problem: Configure a separate user to run NGINX and lock the password for that user  
Root Cause: Running NGINX as a dedicated, non-privileged user improves security; locking the password prevents direct logins

Resolution Steps

1. Create a dedicated nginx user without a home directory or shell access  
   useradd --no-create-home --shell /sbin/nologin nginx
2. Lock the password for the nginx user  
   passwd -l nginx
3. Ensure the nginx config runs as the nginx user  
   In /etc/nginx/nginx.conf, set:  
   user nginx;
4. Change ownership of NGINX directories as needed  
   chown -R nginx:nginx /var/log/nginx  
   chown -R nginx:nginx /var/lib/nginx
5. Restart NGINX to apply user changes  
   systemctl restart nginx

Notes

* Root access required for all steps
* Password locking disables login with a password, but the account can still run processes
* Verify file permissions so nginx can read configs and write logs
* Never run NGINX as root except for startup/binding to privileged ports (Nginx will drop to specified user after starting)

Sample Output

1. useradd --no-create-home --shell /sbin/nologin nginx  
   (No output if successful)
2. passwd -l nginx  
   Locking password for user nginx.  
   passwd: Success
3. nginx -t  
   nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
   nginx: configuration file /etc/nginx/nginx.conf test is successful
4. systemctl restart nginx  
   (No output if successful)

Problem: Perform a basic test to verify that NGINX is working  
Root Cause: Need to confirm NGINX is installed correctly and serving HTTP requests

Resolution Steps

1. Check that the NGINX service is running  
   systemctl status nginx
2. Verify NGINX is listening on the expected port (default: 80)  
   ss -tlnp | grep 80
3. Use curl from the local server to check the default page  
   curl -I [http://localhost](http://localhost/)
4. Access the server’s IP or hostname from a web browser  
   http://server\_ip\_or\_hostname/

Notes

* Root access not required for curl or browser test
* If using a firewall, ensure port 80 (HTTP) and/or 443 (HTTPS) is open
* The default NGINX page is usually served from /usr/share/nginx/html/index.html or /data/nginx/html/index.html if custom installed

Sample Output

1. systemctl status nginx  
   ● nginx.service - The nginx HTTP and reverse proxy server  
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)  
   Active: active (running) since Wed 2025-07-02 14:05:10 +0800; 1min ago  
   …
2. curl -I [http://localhost](http://localhost/)  
   HTTP/1.1 200 OK  
   Server: nginx/1.24.0  
   Date: Wed, 02 Jul 2025 06:08:21 GMT  
   Content-Type: text/html  
   Content-Length: 612  
   Last-Modified: Wed, 02 Jul 2025 06:00:00 GMT  
   Connection: keep-alive  
   ETag: "60be8a90-264"  
   Accept-Ranges: bytes
3. Browser shows:  
   “Welcome to nginx!” or custom index page

Problem: Load (start) and reload the NGINX daemon  
Root Cause: Need to start NGINX for the first time, or reload its configuration after changes without dropping connections

Resolution Steps

1. Start NGINX (if not already running)  
   systemctl start nginx
2. Reload the NGINX configuration without stopping the daemon  
   systemctl reload nginx
3. (Alternative) Test the configuration before reloading  
   nginx -t

Notes

* Root access required for all systemctl and nginx -t commands
* start starts NGINX if it is not running
* reload tells NGINX to reread its configuration and apply changes without interrupting existing connections
* Always test the config with nginx -t before reloading, to avoid downtime from config errors

Sample Output

1. systemctl start nginx  
   (No output if successful)
2. nginx -t  
   nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
   nginx: configuration file /etc/nginx/nginx.conf test is successful
3. systemctl reload nginx  
   (No output if successful)

Problem: Change the listening port in NGINX and ensure it is available before starting the service  
Root Cause: The default port (80 or 443) may be in use by another process, or a custom port is required

Resolution Steps

1. Edit the NGINX configuration file (e.g., /etc/nginx/nginx.conf or /etc/nginx/conf.d/default.conf)  
   Change the listen directive to the desired port (example: 8080):  
   server {  
   listen 8080;  
   server\_name your.domain.com;

# ...

}

1. Check if the new port is already in use  
   ss -tlnp | grep 8080
2. If the port is not in use, start or reload NGINX  
   systemctl start nginx  
   or, if already running:  
   systemctl reload nginx
3. If the port is in use, identify the conflicting process  
   ss -tlnp | grep 8080  
   (Shows the PID and process name using the port)  
   Decide whether to stop the conflicting process, use a different port, or resolve the conflict before restarting NGINX

Notes

* Root access required for config edits, checking ports, and managing services
* Always test your configuration before reload: nginx -t
* If changing to a port <1024, NGINX must start as root (drops to user later)

Sample Output

1. After editing config to listen 8080;  
   (No output—just save the file)
2. ss -tlnp | grep 8080  
   (No output if port is free)  
   or  
   LISTEN 0 128 0.0.0.0:8080 0.0.0.0:\* users:(("other\_process",pid=1234,fd=5))
3. nginx -t  
   nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
   nginx: configuration file /etc/nginx/nginx.conf test is successful
4. systemctl reload nginx  
   (No output if successful)

Problem: Run two separate NGINX instances on different ports on the same RHEL 8 server  
Root Cause: Need to serve different applications or sites independently, each with its own configuration and port

Resolution Steps

1. Install NGINX twice using different installation methods or binaries
   * Use the system NGINX package (installed via yum or from official repo) for the first instance
   * Compile a second, custom NGINX binary to a different directory (e.g., /data/nginx2) for the second instance
2. Configure each instance to use its own config, log, and pid files
   * For the system NGINX (default):  
     Edit /etc/nginx/nginx.conf  
     Set:  
     server {  
     listen 8080;  
     server\_name first.site;  
     # ...  
     }
   * For the custom NGINX:  
     When configuring, specify all unique paths:  
     ./configure --prefix=/data/nginx2 --conf-path=/data/nginx2/conf/nginx.conf --sbin-path=/data/nginx2/sbin/nginx --error-log-path=/data/nginx2/logs/error.log --http-log-path=/data/nginx2/logs/access.log --pid-path=/data/nginx2/nginx2.pid  
     make  
     make install  
     Edit /data/nginx2/conf/nginx.conf:  
     server {  
     listen 9090;  
     server\_name second.site;  
     # ...  
     }
3. Start both NGINX instances
   * System NGINX (port 8080):  
     systemctl start nginx
   * Custom NGINX (port 9090):  
     /data/nginx2/sbin/nginx -c /data/nginx2/conf/nginx.conf
4. Verify both are running and listening on their respective ports  
   ss -tlnp | grep -E "8080|9090"  
   curl -I [http://localhost:8080](http://localhost:8080/)  
   curl -I [http://localhost:9090](http://localhost:9090/)

Notes

* Root required for initial install and systemctl operations
* Make sure both config files point to unique log, pid, and temp directories
* If running as different users, make sure both have permission for their own directories

Sample Output

1. ss -tlnp | grep -E "8080|9090"  
   LISTEN 0 128 0.0.0.0:8080 0.0.0.0:\* users:(("nginx",pid=1234,fd=6))  
   LISTEN 0 128 0.0.0.0:9090 0.0.0.0:\* users:(("nginx",pid=5678,fd=6))
2. curl -I [http://localhost:8080](http://localhost:8080/)  
   HTTP/1.1 200 OK  
   Server: nginx/1.24.0  
   …
3. curl -I [http://localhost:9090](http://localhost:9090/)  
   HTTP/1.1 200 OK  
   Server: nginx/1.24.0

Problem: Identify 404, 401, 500, and 503 errors in NGINX logs and troubleshoot their causes  
Root Cause: HTTP errors in logs signal client mistakes (404, 401) or server issues (500, 503) that need to be detected and resolved

Resolution Steps

1. Find HTTP error codes in the access log  
   grep -E " 404 | 401 | 500 | 503 " /var/log/nginx/access.log
2. Count the number of each error  
   grep " 404 " /var/log/nginx/access.log | wc -l  
   grep " 401 " /var/log/nginx/access.log | wc -l  
   grep " 500 " /var/log/nginx/access.log | wc -l  
   grep " 503 " /var/log/nginx/access.log | wc -l
3. View recent error messages from the error log  
   tail -n 50 /var/log/nginx/error.log
4. Check for error details with timestamps  
   grep -E "404|401|500|503" /var/log/nginx/error.log

Notes

* No root required if you have read access to the log files
* Paths may differ if you installed NGINX in a custom location (e.g., /data/nginx/logs/access.log)
* 404 = Not Found (often a bad URL or missing file)
* 401 = Unauthorized (bad/missing credentials, misconfigured auth)
* 500 = Internal Server Error (bug or backend failure)
* 503 = Service Unavailable (backend crashed or overloaded)

Sample Output

grep -E " 404 | 401 | 500 | 503 " /var/log/nginx/access.log  
192.168.1.5 - - [02/Jul/2025:11:24:10 +0800] "GET /missing.html HTTP/1.1" 404 153 "-" "Mozilla/5.0"  
192.168.1.6 - - [02/Jul/2025:11:24:12 +0800] "POST /api HTTP/1.1" 401 0 "-" "curl/7.29.0"  
192.168.1.8 - - [02/Jul/2025:11:25:10 +0800] "GET /api/data HTTP/1.1" 500 241 "-" "PostmanRuntime/7.28.0"  
192.168.1.10 - - [02/Jul/2025:11:26:00 +0800] "GET /maintenance HTTP/1.1" 503 188 "-" "Mozilla/5.0"

Solutions (by error code):

* 404 Not Found:
  + Check if the requested file or URL exists
  + Verify the root or alias directive in your NGINX config is correct
  + Fix broken links in your web application
* 401 Unauthorized:
  + Check authentication settings (auth\_basic and related directives)
  + Verify username and password files are accessible and correct
* 500 Internal Server Error:
  + Look for errors in /var/log/nginx/error.log and your backend application logs
  + Check permissions, backend service health, and NGINX config syntax
  + Restart backend services if needed
* 503 Service Unavailable:
  + Check if upstream servers (e.g., PHP-FPM, Node.js, etc.) are running
  + Ensure you have enough worker processes/threads
  + Look for overloads or crashes in backend logs
  + Increase resources or fix application issues causing downtime

### **400 Bad Request**

Root Cause:

* Malformed HTTP request, oversized headers, or invalid characters in request  
  Solution:
* Check for large cookies, bad proxies, or client misbehavior
* Tune large\_client\_header\_buffers in nginx.conf if header size is the issue
* Sanitize client requests or fix web application code

### **401 Unauthorized**

Root Cause:

* Protected resource requires authentication; client missing or invalid credentials  
  Solution:
* Check NGINX auth\_basic and related settings
* Ensure correct credentials are sent by client
* Verify .htpasswd or other credential files exist and are readable

### **403 Forbidden**

Root Cause:

* Permissions deny access (bad root, file perms, or NGINX user rights)  
  Solution:
* Verify filesystem permissions and SELinux settings
* Confirm root or alias is pointed to the right directory
* Don’t use chmod 777—fix the real permission issue

### **404 Not Found**

Root Cause:

* Requested file or endpoint doesn’t exist  
  Solution:
* Check path in root/alias in config
* Confirm file exists on disk
* Fix broken links in your web app

### **405 Method Not Allowed**

Root Cause:

* HTTP method (e.g., POST) not permitted by config  
  Solution:
* Add allowed methods using limit\_except or correct location blocks

### **413 Payload Too Large**

Root Cause:

* Request body/file exceeds NGINX’s or upstream’s size limit  
  Solution:
* Increase client\_max\_body\_size in server or location block
* Adjust upstream (e.g., PHP-FPM) limits if needed

### **499 Client Closed Request**

Root Cause:

* Client (e.g., browser, load balancer) closed connection before server response  
  Solution:
* Investigate for slow backends or network issues
* Optimize backend response time
* Usually not a server bug, but a sign of latency or client impatience

### **500 Internal Server Error**

Root Cause:

* Backend crashed, config error, permission issue, or app bug  
  Solution:
* Check /var/log/nginx/error.log for details
* Validate config with nginx -t
* Restart backend services, check permissions, debug app code

### **502 Bad Gateway**

Root Cause:

* NGINX can’t talk to upstream (backend is down, crashed, or timed out)  
  Solution:
* Confirm upstream servers are running
* Check network/firewall/SELinux between NGINX and backend
* Look for errors in backend logs

### **503 Service Unavailable**

Root Cause:

* Backend servers down, overloaded, or all marked as failed  
  Solution:
* Restart or scale backend services
* Check load balancer or upstream config
* Review worker process counts in backend app (e.g., PHP-FPM)

### **504 Gateway Timeout**

Root Cause:

* Upstream took too long to respond  
  Solution:
* Increase proxy\_read\_timeout or similar NGINX timeout settings
* Fix slowness in backend app or database
* Check for network bottlenecks

### **[emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)**

Root Cause:

* Another process is using the same port  
  Solution:
* Find the process with ss -tlnp | grep :80
* Stop or reconfigure conflicting service
* Change NGINX listen port if necessary

### **[error] connect() to upstream failed**

Root Cause:

* NGINX cannot connect to backend server  
  Solution:
* Check backend IP/port, firewalls, SELinux, and backend server status
* Correct any typos or network issues

### **[error] permission denied while connecting to upstream**

Root Cause:

* SELinux or filesystem permissions block NGINX from talking to upstream or reading files  
  Solution:
* Check and adjust SELinux with setenforce 0 (test only) or proper policy
* Fix file and directory ownership and permissions

### **Disk full / Too many open files**

Root Cause:

* Log or cache directory full, or system open file descriptor limit hit  
  Solution:
* Free up disk space
* Increase worker\_rlimit\_nofile and system file limits