WIS

Server Configuration Best Practices for Nginx with WordPress

Description

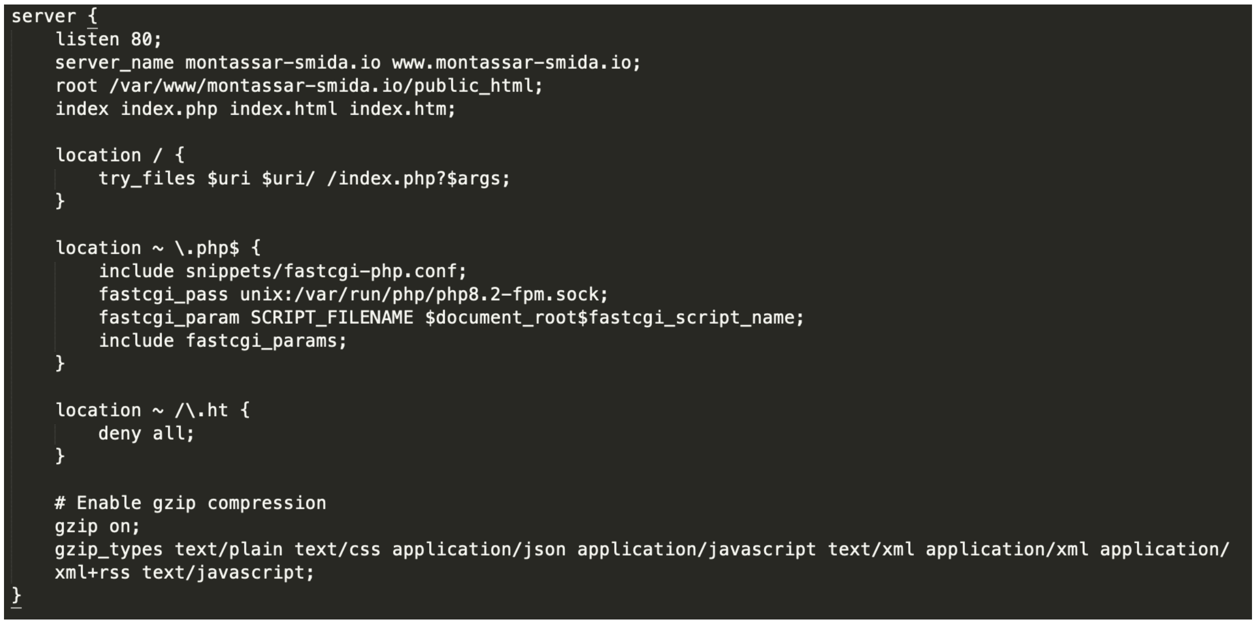
Nginx is a high-performance web server and reverse proxy server known for its efficiency and scalability. When hosting a WordPress site, configuring Nginx correctly is essential to ensure optimal performance, security, and reliability. This document outlines best practices for configuring Nginx for WordPress, including server block setup, virtual hosting, and SSL management. It also provides technical specifications and optimization tips to enhance server performance.

Technical Specifications

- **Nginx Version:** Ensure you are using the latest stable version of Nginx for improved features and security.
- **PHP Version:** PHP 8.2 or higher is recommended for compatibility and performance with WordPress.
- **Operating System:** Ensure your server's operating system is up-to-date with security patches.
- **Hardware Requirements:** Adequate CPU and RAM resources based on the expected traffic and load.

Server Blocks Setup

Basic Server Block Configuration:



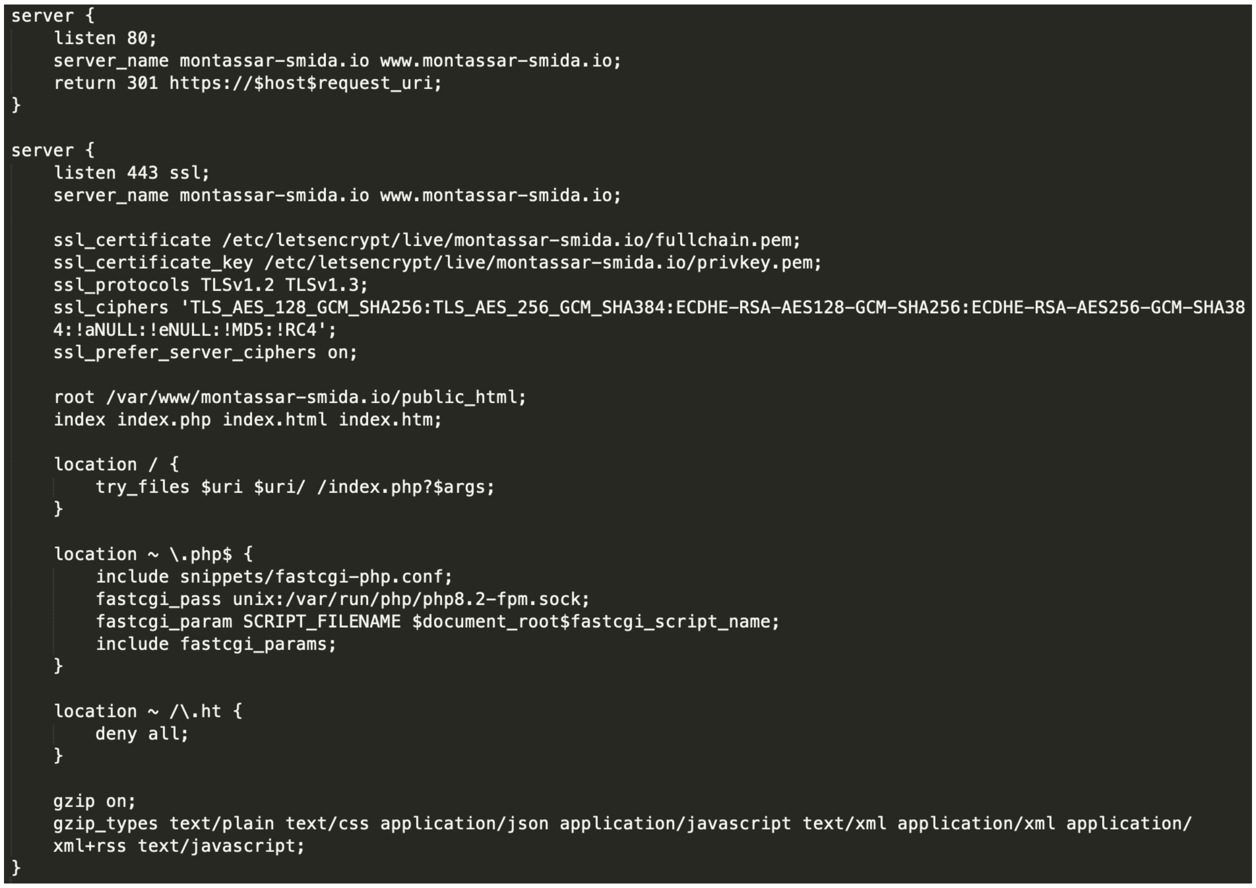
```
server {
    listen 80;
    server_name montassar-smida.io www.montassar-smida.io;
    root /var/www/montassar-smida.io/public_html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    # Enable gzip compression
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/
    xml+rss text/javascript;
}
```

Explanation:

- **`listen 80;`** - Listens on port 80 for HTTP requests.
- **`server_name`** - Specifies the domain names for this server block.
- **`root`** - Sets the document root for `montassar-smida.io`.
- **`index`** - Lists default index files.
- **`location /`** - Handles requests and falls back to `index.php` if files are not found.
- **`location ~ \.php$`** - Passes PHP requests to PHP-FPM for processing.
- **`location ~ /\.ht`** - Denies access to `.htaccess` files.
- **`gzip`** - Enables gzip compression for better performance.

Virtual Hosts Configuration

Example Configuration for SSL:



```
server {
    listen 80;
    server_name montassar-smida.io www.montassar-smida.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name montassar-smida.io www.montassar-smida.io;

    ssl_certificate /etc/letsencrypt/live/montassar-smida.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/montassar-smida.io/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA38
    4:!aNULL:!eNULL:!MD5:!RC4';
    ssl_prefer_server_ciphers on;

    root /var/www/montassar-smida.io/public_html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/
    xml+rss text/javascript;
}
```

Explanation:

- **Redirects HTTP traffic to HTTPS.**
- **Configures SSL with Let’s Encrypt certificates for `montassar-smida.io`.**
- **Sets secure SSL protocols and ciphers.**
- **Ensures SSL is used properly for secure connections.**

Best Practices for Nginx Configuration

1. **Use Latest Version:** Always use the latest stable version of Nginx to benefit from the latest features and security updates.

2. **Optimize Worker Processes:** Configure the number of worker processes based on the number of CPU cores available.

```
worker_processes auto;
```

3. **Tune Worker Connections:** Adjust the number of connections per worker process according to server load.

```
events {
     worker_connections 1024;
}
```

4. **Enable Caching:** Use caching mechanisms to improve performance and reduce server load. Configure caching for static files and proxy caching.

5. **Limit Request Size:** Set appropriate limits for client request sizes to prevent abuse.

```
client_max_body_size 10M;
```

6. **Use Compression:** Enable gzip compression to reduce the size of HTTP responses.

7. **Monitor Performance:** Regularly monitor server performance and adjust configurations as necessary.

Optimization and Tuning

Cache Static Content: Use Nginx's built-in caching for static content (images, CSS, JavaScript) to reduce server load and improve response times.

```
location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
     expires 30d;
     access_log off;
}
```

Use Load Balancing: Distribute incoming traffic across multiple backend servers to enhance scalability and fault tolerance.

Enable HTTP/2: Use HTTP/2 for improved performance with modern web applications.

```
listen 443 ssl http2;
```

Optimize SSL/TLS Settings: Configure SSL/TLS settings for security and performance. Use strong ciphers and enable forward secrecy.

Tune Timeout Settings: Adjust timeouts to balance between user experience and resource utilization.

```
client_body_timeout 10s;
client_header_timeout 10s;
keepalive_timeout 65s;
```

Reduce Latency: Minimize latency by optimizing backend server configurations and reducing response times.