Automated Deployment Process for WordPress Website using Nginx

==

This script automates the deployment process for setting up a WordPress website using Nginx as the web server. It includes installation and configuration steps for Nginx, MySQL, PHP, and WordPress, along with additional configurations for server block, caching, gzip compression, and SSL/TLS certificate installation.

❖ **Setting up the Server**:

➢ **Get a Virtual Private Server (VPS)**:

- Choose a cloud service like AWS, Azure, Google Cloud, or DigitalOcean.
- Make a new VPS instance.
- Pick a secure Linux system like Ubuntu 22.04.

➢ **Set Up the VPS**:

- Make sure only necessary ports like SSH, HTTP, and HTTPS are accessible by configuring firewall rules.
- Secure SSH access by using SSH keys instead of passwords and changing the default SSH port.
- Keep your server updated regularly with security patches.

❖ Usage

1. Clone this repository or download the script directly onto your server.
   Make the script executable:

```
chmod +x deploy_wordpress.sh
```

3. Execute the script:

```
./deploy_wordpress.sh
```

❖ This script automates the setup process for a WordPress website on a Linux server. Let's break it down step by step:

1.Update apt:

```
apt update -y
```

- This command updates the package lists for upgrades and new package installations.

2. Install Nginx:

```
apt install nginx –y
```

- This installs the Nginx web server.

3.**Check Nginx status**:

```
systemctl status nginx
```

- This checks the status of the Nginx service.

4. Enable and start Nginx:

```
systemctl enable nginx --now
```

5.* Install MySQL*:

```
apt install mysql-server mysql-client –y
```

- This installs MySQL server and client.

6. Configure MySQL security:

```
mysql_secure_installation <<EOF
Y
2
Y
N
Y
Y
EOF
```

- This script automatically configures basic security options for MySQL.

7. Create WordPress database and user:

```
mysql -u root -p <<MYSQL_SCRIPT
CREATE DATABASE wordpress;
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'Pramod@#123';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
FLUSH PRIVILEGES;
EXIT
MYSQL_SCRIPT
```

- **mysql -u root -p <<MYSQL_SCRIPT:** This command starts a MySQL session and runs the following commands as input from the script until the **EOF** delimiter.
- **CREATE DATABASE wordpress;:** This creates a MySQL database named "wordpress" to store WordPress data.
- **CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'Pramod@#123';:** This creates a MySQL user named "wpuser" with a specified password.
- **GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';:** This grants all privileges on the "wordpress" database to the "wpuser" user.
- **FLUSH PRIVILEGES;:** This reloads the privileges to ensure that the changes take effect.
- **EXIT:** This exits the MySQL session.

8. Installing PHP and PHP-FPM:

```
apt install php php-fpm php-mysql php-gd -y
```

- This installs PHP and necessary modules for WordPress.

9. **Download and install WordPress:**

```
wget https://wordpress.org/latest.zip
```

```
apt install unzip -y
unzip latest.zip
cd wordpress
cp -r * /var/www/html/
sudo chown -R www-data:www-data /var/www/html/
chmod -R 755 /var/www/html/
```

- **wget https://wordpress.org/latest.zip:** This command downloads the latest version of WordPress from the official website.
- **apt install unzip -y:** This installs the unzip utility to extract the downloaded ZIP archive.
- **unzip latest.zip:** This command extracts the WordPress files from the downloaded ZIP archive.
- **cd wordpress:** This changes the current directory to the extracted WordPress directory.
- **cp -r * /var/www/html/:** This copies all files and directories within the WordPress directory to the web server's document root directory (/var/www/html/).
- **sudo chown -R www-data:www-data /var/www/html/:** This changes the ownership of the copied files and directories to the web server user and group.
- **chmod -R 755 /var/www/html/:** This sets the appropriate permissions on the copied files and directories for web server access.

10. Configure Nginx server block:

```
cd /etc/nginx/sites-enabled/
mv default wordpress
```

- This renames the default Nginx server block configuration to 'wordpress'.

11. **Modify Nginx configuration with caching and gzip compression:**

- This part creates a custom Nginx server block configuration file in **/etc/nginx/sites-enabled/wordpress.**

```
cat > /etc/nginx/sites-enabled/wordpress <<'EOF'

server {
    listen 80;
    listen [::]:80;

    server_name www.linuxtest.hopto.org linuxtest.hopto.org;

    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;

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

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }

    # Cache static files
    location ~* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {
        expires 30d;
        add_header Cache-Control "public";
    }

    # Enable gzip compression
    gzip on;
    gzip_disable "msie6";
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_types text/plain text/css application/json application/javascript application/xml
application/xml+rss application/x-font-ttf application/x-font-opentype application/vnd.ms-fontobject
image/svg+xml image/x-icon;
}
EOF
```

Command: cat > /etc/nginx/sites-enabled/wordpress <<'EOF'

- **Explanation:** This command uses the cat command to create a new file named "wordpress" in the **/etc/nginx/sites-enabled/** directory. The **>** operator redirects the output of **ca**t to the specified file. The **<<'EOF'** syntax is a here document that allows inputting multiple lines of text until the **EOF** marker is encountered. The single quotes around **EOF** prevent variable expansion within the here document.
- **Server Block Configuration:**
- **listen 80; and listen [::]:80;:** These lines specify that the server block listens on port 80 for both IPv4 and IPv6 connections.
- **server_name:** This line specifies the domain names for which this server block will be used.
- **root /var/www/html;:** This line sets the root directory for the website files.
- **index:** This line specifies the order in which index files are checked when a directory is requested.
- **location / { ... }:** This block handles requests to the root directory ("/") of the website.
- **location ~ .php$ { ... }:** This block handles requests ending in ".php".
- **location ~ /.ht { ... }:** This block denies access to files starting with ".ht".
- **Cache static files:** This block caches static files with specific file extensions.
- **Enable gzip compression:** This block enables gzip compression and specifies various compression-related settings.

12. Check Nginx syntax and restart Nginx:

```
nginx -t
```

```
systemctl restart nginx
```

- This checks the syntax of the Nginx configuration and restarts Nginx if there are no errors.

13.Install SSL:

```
apt install ufw -y
```

- This command uses the **apt** package manager to install the Uncomplicated Firewall **(ufw)** package.

```
ufw allow 'Nginx Full' -y
```

- This command uses the **ufw** command-line utility to allow incoming traffic according to the "Nginx Full" profile.
- **'Nginx Full'**: This is the name of the UFW application profile that allows full access to Nginx, including both HTTP and HTTPS traffic.
- **-y**: This flag automatically answers "yes" to any prompts during the execution of the command, making it non-interactive.
- Explanation: This command allows incoming traffic on ports required by the "Nginx Full" profile, ensuring that Nginx can accept both HTTP and HTTPS connections.

```
apt install certbot python3-certbot-nginx -y
```

- This command uses the **apt** package manager to install Certbot and the Certbot Nginx plugin.
- **-y:** This flag automatically answers "yes" to any prompts during the installation process, making it non-interactive.
- Explanation: This command installs Certbot, which is a tool for obtaining and managing SSL/TLS certificates, along with the Certbot Nginx plugin, which allows Certbot to automatically configure SSL/TLS for Nginx.

```
certbot --nginx -d linuxtest.hopto.org --email pkapade93@gmail.com
```

- This command invokes Certbot with the **--nginx** option, indicating that Certbot should use the Nginx plugin for certificate installation and renewal.
- **-d linuxtest.hopto.org**: This specifies the domain name for which the SSL/TLS certificate should be obtained.

- **--email pkapade93@gmail.com:** This provides the email address to associate with the SSL/TLS certificate for renewal notifications and security alerts.
- Explanation: This command requests and installs an SSL/TLS certificate from Let's Encrypt for the specified domain using the Nginx plugin, with the provided email address for certificate management notifications.

❖ GitHub Repository Setup:

GitHub Repository URL: **WordPress Deployment**
(https://github.com/Pramod06k/Wordpress)

➢ **Sign in to GitHub:** Go to GitHub and sign in to your account. If you don't have an account, you can sign up for free.

➢ **Create a New Repository:**
- Click on the "+" icon in the top right corner of the GitHub dashboard.
- Select "New repository" from the dropdown menu.

➢ **Fill in the Repository Information:**
- Enter a name for your repository in the "Repository name" field.
- Optionally, add a description for your repository.
- Choose whether the repository will be public or private.
- Select the checkbox to initialize the repository with a README file.

- ➢ **Choose Options:**
- Choose options for the repository such as adding a .gitignore file and a license if needed.
- Click on the "Create repository" button.

- ➢ **Clone the Repository:**
- After creating the repository, you can clone it to your local machine using the provided URL. You can find the URL on the repository page.
- Add Files and Push:
- Add your files to the local repository directory.
- Use **git add** . to stage all changes.
- Use **git commit -m "Initial commit"** to commit the changes.
- Use git push origin master to push the changes to the GitHub repository.
- Your GitHub repository is now created and populated with files. You can continue to add files, make changes, and push commits as needed. Remember to update the README file with relevant information about your project.

==