raazdesign

Setting Up and Managing Your Website on Vultr with OpenBSD HTTPD

These instructions help setup OpenBSD and pull a website from GitHub.

It's not comprehensive you'll need to know how to troubleshoot, but my advice on that is make sure your user and the file you're running have the right permissions and file path, again file path, file path. Super easy to mess up file path in my experience. Use verbose mode (-v) when running commands to get more info too. Also of course I recommend building everything on a dummy domain then pointing your final website to the IP when done and tested.

Of course, read the docs if you get lost: https://www.openbsd.org/index.html

At the end of this document is an OpenBSD Cheat Sheet for common commands.

Drop this file in Chat or the LLM of your choice to get assistance.

1. Forward Your Domain to Vultr

- Obtain Your Vultr Server IP Address:
  - Log in to your Vultr account, and find the public IPv4 address of your deployed server.
- Update DNS Records:
  - Go to your domain registrar (e.g., GoDaddy, Namecheap) and update the DNS settings.
  - Set the `A` record for your domain (`example.com`) and any subdomains (like `www.example.com`) to point to your Vultr server's IP address.

2. Setup Vultr IPv4 and Firewall

- Configure IPv4 Address:
  - Ensure that your Vultr server has a public IPv4 address assigned.
- Setup Firewall Rules on Vultr:
  - Go to the Vultr dashboard, and navigate to the firewall settings.
  - Create a firewall group and add rules to allow traffic on ports `22` (SSH), `80` (HTTP), and `443` (HTTPS).
  - Assign this firewall group to your server.
- Setup alerts: Go to user > notifications and turn on outage and maintenance alerts and set custom bandwidth alerts.

3. Setup SSH Access

- Generate an SSH Key Pair (Local Machine):
  - On your local machine, generate an SSH key pair:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

  - Copy the public key (`~/.ssh/id_ed25519.pub`) to your clipboard.
- Add the SSH Key to Your Vultr Server:
  - During server deployment, or afterward via the Vultr dashboard, add your public SSH key to the server.

4. SSH to Your Vultr Server

- Connect to Your Server via SSH:
  - Use the following command to SSH into your server:

```
ssh root@your_server_ip
```

Install nano and git, nano is an easy text editor to use

```
pgk_add nano
```

```
pgk_add git
```

5. Create a Non-Root User and Disable Root User

- Create a New User:
  - Add a new user (e.g., `youruser`):

```
adduser
```

  - Follow the prompts to set a password and configure the user.
- Disable Root Login:
  - Edit the SSH configuration file to disable root login:

```
nano /etc/ssh/sshd_config
```

  - Set `PermitRootLogin` to `no`.
  - Restart the SSH service:

```
rcctl restart sshd
```

6. Elevate Privileges of the New User

- Grant Sudo (doas) Access:
  - Edit the `doas` configuration file to grant the new user root privileges:

```
echo "permit persist youruser as root" >> /etc/doas.conf
```

  - Replace `youruser` with your actual username.
- Switch to the New User and Elevate Privileges:
  - Log out and log back in as the new user:

```
su - youruser
```

  - Elevate to root privileges:

```
doas -s
```

7. Setup HTTPD

- Configure HTTPD:
  - Open and edit `/etc/httpd.conf`:

```
nano /etc/httpd.conf
```

  - Add the following configuration:

```
server "example.com" {
    listen on * port 80
    location "/.well-known/acme-challenge/*" {
        root "/htdocs/acme"
    }
    location * {
        block return 302 "https://$HTTP_HOST$REQUEST_URI"
    }
}
server "example.com" {
    listen on * tls port 443
    root "/var/www/htdocs/example.com"
    tls {
        certificate "/etc/letsencrypt/live/example.com/fullchain.pem"
        key "/etc/letsencrypt/live/example.com/privkey.pem"
    }
    location "/.well-known/acme-challenge/*" {
        root "/htdocs/acme"
    }
}
```

- Create the Document Root:
  - Create the necessary directories:

```
mkdir -p /var/www/htdocs/example.com
mkdir -p /var/www/htdocs/acme/.well-known/acme-challenge
chown -R www:www /var/www/htdocs/example.com
chown -R www:www /var/www/htdocs/acme
```

- Start and Enable HTTPD:
  - Enable and start the HTTPD service:

```
rcctl enable httpd
rcctl start httpd
```

8. Install and Use Certbot for SSL with Let's Encrypt

- Install Certbot:
  - Install Certbot using:

```
pkg_add certbot
```

- Obtain SSL Certificates:
  - Run Certbot to generate SSL certificates:

```
certbot certonly --webroot -w /var/www/htdocs/acme -d example.com -d www.example.com
```

- Setup Automatic Renewal:
  - Edit the Certbot renewal configuration file:

```
nano /etc/letsencrypt/renewal/example.com.conf
```

- Ensure `webroot_path` is set correctly:

```
ini
[renewalparams]
webroot_path = /var/www/htdocs/acme
[webroot_map]
example.com = /var/www/htdocs/acme
www.example.com = /var/www/htdocs/acme
```

- Configure the cron job for renewal:

```
crontab -e
```

- Add:

```
0 */12 * * * certbot renew --quiet --post-hook "rcctl reload httpd"
```

9. Linking GitHub Repository

- Install Git:
  - Install Git on your server if you haven’t done so already:

```
pkg_add git
```

- Generate an SSH Key Pair:
  - Generate the SSH key:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

  - Add the SSH key to your GitHub account.
- Clone the Repository:
  - Clone your repository to the appropriate directory that you want httpd to pull from:

```
git clone git@github.com:yourusername/yourrepo.git /var/www/htdocs/yourrepo
```

10. Pulling Changes from GitHub

- Create a Pull Script:
  - Create a script `/usr/local/bin/git-pull.sh`:

```
nano /usr/local/bin/git-pull.sh
```

  - Add the following content:

```
#!/bin/ksh
cd /var/www/htdocs/yourrepo || exit
git pull origin main
rcctl restart httpd
```

  - Make the script executable:

```
chmod +x /usr/local/bin/git-pull.sh
```

- Set Up Cron Job for Automatic Pulling:
  - Edit the crontab to pull changes every hour:

```
crontab -e
```

  - Add:

```
0 * * * * /usr/local/bin/git-pull.sh >> /var/log/git-pull.log 2>&1
```

- Manually Pull Changes:
  - To manually pull changes and restart HTTPD:

```
/usr/local/bin/git-pull.sh
```

  - Or, manually run the commands:

```
cd /var/www/htdocs/yourrepo
git pull origin main
rcctl restart httpd
```

11. Site Maintenance and Monitoring

- Regular Updates:
  - Regularly update the system and packages:

```
syspatch
pkg_add -u
```

12. Common Commands

1. Basic Navigation and File Operations

ls	List the contents of a directory.
ls -l	List the contents of a directory in long format (shows permissions, owner, etc.).
cd /path/to/directory	Change directory to the specified path.
pwd	Print the current working directory.
cp source destination	Copy a file from source to destination.
mv oldname newname	Move or rename a file or directory.
rm filename	Remove (delete) a file.
mkdir dirname	Create a new directory.
cat filename	Output the contents of a file.
nano filename	Open a file in the nano text editor (after it's installed).

2. File and Directory Permissions

chmod 755 filename	Change the permissions of a file or directory (755: user can read/write/execute, others can read/execute).
chown user:group filename	Change the ownership of a file or directory to a specified user and group.
ls -l	– Shows file permissions and ownership in detail.

Explanation of File Permissions: - Example: -rwxr-xr-x

- r (read), w (write), and x (execute).

- The first set of three (rwx) applies to the owner, the second (r-x) applies to the group, and the third (r-x) applies to others.

3. System Management with doas (Similar to sudo)

doas command	Run a command with elevated (root) privileges (e.g., doas nano /etc/httpd.conf).
doas -s	Start a root shell (run commands as root until you exit).
doas pkg_add package	Install a package (e.g., doas pkg_add nano).
doas rcctl restart service	Restart a service (e.g., doas rcctl restart httpd).
doas chown user:group /path/to/file	Change the owner of a file/directory.

4. Package Management

pkg_add package_name	Install a package (e.g., pkg_add git).
pkg_info	List all installed packages.
pkg_delete package_name	Uninstall a package.

5. HTTPD (Web Server) Management

rcctl enable httpd	Enable the httpd service to start at boot.
rcctl start httpd	Start the httpd service.
rcctl stop httpd	Stop the httpd service.
rcctl restart httpd	Restart the httpd service.
rcctl reload httpd	Reload the httpd configuration after changes.

6. Git Commands

git clone git@github.com:user/repo.git	Clone a remote GitHub repository to your local server.
git pull origin main	Pull the latest changes from the main branch of the repository.
git status	Show the status of your local Git repository.
git add filename	Stage changes for commit.
git commit -m "message"	Commit changes with a message.
git push origin main	Push local changes to the remote repository.

7. Certbot (SSL with Let’s Encrypt)

pkg_add certbot	Install Certbot (used for obtaining SSL certificates).
certbot certonly --webroot -w /var/www/htdocs/acme -d yourdomain.com	Obtain an SSL certificate using the webroot method.
certbot renew	Renew the SSL certificate.
certbot renew --dry-run	Test the renewal process without making changes.
crontab -e	Edit cron jobs for automated tasks (like renewing SSL certificates).

8. Useful Shortcuts

Ctrl + C	Cancel the current command in the terminal.
Ctrl + Z	Suspend a process (use fg to resume).
Ctrl + D	Log out or close a shell session.
!!	Re-run the last command.
exit	Exit the current shell session.
Arrow up	Cycle through previous commands

9. Running Commands in Verbose Mode

Verbose mode is useful when you want more detailed information about what a command is doing. Many common OpenBSD commands support the -v flag, which outputs additional information.

Examples:

pkg_add -v nano	Install nano with detailed output.
rcctl -v restart httpd	Restart the HTTPD service with more information about the process.
git pull -v origin main	Pull changes from the GitHub repository with more detailed output.
certbot -v renew	Renew the SSL certificates with verbose output.