Configure your VPS in under 20 minutes

SnappyVPS

v1.0.3

User Guide

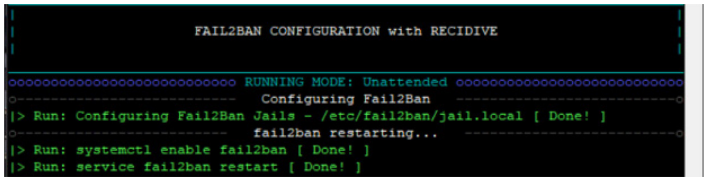
Contents

Description

SnappyVPS is a UNIX installation framework that helps make initial VPS setup and configuration a breeze. In simple terms, it will install a suite of applications and configure them for you according to your specifications. It was created to provide administrators with the tools to rapidly achieve their own custom made installation procedures in a tidy, well presented and efficient way with as little effort as possible.

The ultimate goal of SnappyVPS is to provide all these features in playbook format, where the same or similar configuration can be rapidly deployed whenever a new VPS might be required. This makes it invaluable for administrators who have to roll out a number of VPSs every day.

The SnappyVPS default payload has four major priorities. Security, Recovery, Ease of Use and Speed.



Additionally to today's **best security practices**, like disabling the root account and allowing only SSH private key access, SnappyVPS only uses the trusted IPtables firewall complemented with threat/country block-lists and an aggressive fail2ban policy.

A robust mechanism of **scheduled backups** is put into place from the get-go. The proven **Webmin/Virtualmin/Usermin** control panel is installed to assist with further configuration and administration of the server, making complicated tasks easy. Finally Google **Pagespeed** is installed and configured for unprecedented web application speed as well as tools such as **jpegoptim** and **opting** which further assist CMS speed gains.

If the default SnappyVPS payload is not your cup of tea, you can modify it or discard it completely and use the SnappyVPS installation framework to install anything you like, run whole subroutines in sub-processes with progress feedback and error reporting, modify configuration files easily all the while separating various sections by displaying visually pleasant titles to screen and/or log file. SnappyVPS puts the administrator in control - use it on your own terms.
Best thing about SnappyVPS – you do not depend on online services to use it!

System Requirements

Recommended configuration for SnappyVPS default payload is Ubuntu 18.04, LTS with 1GB RAM minimum.

When used as a UNIX installation framework with your own payload, SnappyVPS has no O.S. or RAM requirements. Your own payload will define the requirements in this case. The only requirement is that BASH & PERL be installed.

Warning

Acquiring SnappyVPS from unauthorized sources will put your VPS in danger. Malicious hackers can easily embed extraneous code into the numerous files which will ultimately get installed on your server putting your private data and your customers at risk. Once hackers get control of your server, they may easily acquire further information to allow them to access other computers under your control and even your home computer.
Please obtain your copy of SnappyVPS from authorized sources like those specified on snappyvps.com.

License & Disclaimer

VPS Creation

initial_server_setup.sh

Open the file and change the following line to whatever username you prefer for your administrator user. Save the file.

```
USERNAME=joe # Do not use 'root' or 'admin'
```

Now you can use the file to create your droplet as described in the section VPS Droplet Creation. When the droplet has been created, you can login either using the console provided by your VPS provider, or using an SSH client like PuTTY (your SSH key will be used and you'll be prompted for the key's passphrase if you've set one. see SSH into the VPS).

Since you have uploaded SSH keys, a root password will not be generated and emailed to you (see resetting the root password). Instead, on your first login the system will prompt you to enter a new password for your sudo user above. Even though your VPS can only be accessed by thes sudo user and private key, and root SSH is disabled, make sure you use a strong password for any user on your system for peace of mind (https://passwordsgenerator.net/).

Note: The initial setup script will update the VPS and reboot automatically. You may notice this reboot when you try to login for the first time, by being disconnected. In such case, try again after the reboot.

VPS Configuration

install.sh

After the droplet is ready, we need to configure it. SnappyVPS default payload will do the following:

1. Secure the snappyVPS folder and files & make executable the files migrate.sh & setcronbackups.sh
2. Configure Timezone.
3. Create a 1GB swapfile.
4. Install Webmin/Virtualmin/Usermin control panel. (see Webmin payload).
5. Ensure administrator user has SFTP SUDO access.
6. Create an rc.local startup file.
7. Configure Fail2Ban for maximum security.
8. Configure IPtables firewall for maximum security and better integration with Webmin's Linux Firewall page.
    a. Disable UFW and Firewalld.
    b. Disable ProFTP
    c. Enable threats & country block-list with scheduled updates.
9. Configure MySQL for medium loads.
10. Configure Webmin:
    a. Post-Installation Wizard with default options.
    b. Enable hashed passwords.
    c. Enable SSL & Disable FTP in Virtualmin default features.
    d. Set PHP7.2 as default Virtualmin version.
    e. Create mail auto-configuration templates for MS Outlook & Thunderbird.
    f. Enable SSL/CAA for DNS.
    g. Create MySQL prefixes for sub-servers.
    h. Enable password reset functionality.
    i. Enable login to Webmin only from your public IP address.
11. Configure Apache & Apache SSL for maximum security.
12. Configure PHP for multiple concurrent versions.
13. Install various PHP modules required by various CMS applications.
14. Configure email DKIM.
15. Install PuTTY tools.
16. Install & configure Google Pagespeed.
17. Install image tools (jpegoptim & opting) required by various CMS applications.
18. Install imapsync email migration tool.
19. Update the system.
20. Configure unattended updates with email notifications.
21. Configure Quotas.
22. Install rkhunter security tool.
23. Update the locate database.
24. Reboot.

Before running install.sh, you need to specify some of its parameters.

Parameters

Open the file /snappyVPS/install.sh and scroll to the section `###  LOCAL VARIABLE DEFAULTS ###`.

Note: Modifying config files on Windows, will require conversion to UNIX format. (see converting to UNIX)

username

By default this username must be the same as the one you specified earlier in initial_server_setup.sh. Any username you define here will be granted sudo privileges with SFTP. The administrator has already been given such rights when the droplet was created. By specifying your administrator user here, install.sh will recheck and let you know if this user has already been given the required privileges. It is essentially a double-check.
However, in future, if you do require to create new users and give them sudo SFTP access, you can use this same mechanism in config_scp_sudo.sh.

mainDomain (_bin1 _bin2)

Enter your main domain name as was purchased e.g. mydomain.com. This will be used when configuring Webmin, to specify the DNS bind master and the additional DNS name server. By default, SnappyVPS specifies `_bin1` as `"ns1.${mainDomain}"` and `_bin2` as `"ns2.${mainDomain}"`. These defaults will work if at time of creation of your droplet you specified the same prefix `'ns1'` before your main domain name. If not, then you have to also modify `_bin1` & `_bin2` to reflect your own prefix.
`_bin1` & `_bin2` are your DNS servers used when you transfer your domains from other hosts (e.g. Hostgator).

email

Specify a mailbox (root recommended) or a personal email address to send security & update reports to.

_mysqlPass

Here you must specify a MySQL root password to secure your database. Use a complex, 16 character password or use a password generator like https://passwordsgenerator.net/.

_mysqlConf

Configure MariaDB/MySQL depending on system RAM and expected database usage.
Choose from 'small' (256M infrequent use), 'medium' (default 512M regular use), 'large' (1G heavy use) or 'huge' (2G+ heavy use).

timezone

Default is "America/Los_Angeles". A list of timezones can be viewed using: `timedatectl list-timezones`

swapCap

Specify the capacity you require for your swap file. 1GB is the default.

myIPaddress

By default this variable is populated with your public IP address you use to connect to your VPS. No need to change this. Your public IP address will automatically be given some privileges like:

1. Ignored in fail2ban from being banned in the firewall when you enter wrong passwords.
2. Allowed access to the Webmin admin page
3. Allowed access to the Pagespeed admin page

_plmo (preload_mode)

When set to 1 will allow Virtualmin to be preloaded. However this consumes more RAM. Default is 0.

_nldd (no_lookup_domain_daemon)

When set to 1 will disable email lookup domain daemon in Virtualmin. Default is 0.

_psmo (hash passwords)

When set to 1 will instruct Virtualmin to enable hashed passwords. Default is 1.

_spam (email spam handling method)

Either spamassassin or spamc (*spamc requires 2GB RAM minimum*). Default is spamassassin.

_clam (email antivirus handling method)

Either clamscan or clamdscan (*clamdscan requires 2GB RAM minimum*). Default is clamscan.

_rude (Remove-Unused-Dependencies)

When set to 1 will allow unattended upgrades to remove unused dependencies. Default is 1.

_aure (Automatic Reboot)

When set to 1 will allow unattended upgrades to automatically reboot. Default is 1.

_aurt (Automatic-Reboot-Time)

When set it will allow automatic reboot after unattended upgrades at the specified time. Default is "02:00".

Execution

To begin VPS configuration you must transfer SnappyVPS to your VPS. One way to do this is by connecting to the VPS using a SFTP client like WinSCP (see SFTP into the VPS). Once connected, you can transfer SnappyVPS to your home directory.

Now you need to log into your VPS with SSH (PuTTY) to issue the install command.

If you have transferred SnappyVPS as a zip file you need to unzip it before running the installer:

```
sudo unzip -o -q snappyVPS.zip; rm -f snappyVPS.zip;
```

The above, unzips the archive, into the snappyVPS directory, and then deletes the archive.

Now you have to run the installer.

```
sudo chmod u+x ~/snappyVPS/install.sh; sudo ~/snappyVPS/install.sh -u
```

The above makes the install.sh script executable and runs it with the `-u` switch or unattended mode (If you want to run in manual mode, you can discard the `-u` switch). Installation will take a maximum of 20 minutes.
The installer will run automatically until it has completely configured your VPS and then reboot.

Congratulations. Your VPS is now almost fully configured. Execute the post-installation tasks.

Post-Installation tasks

1. Access Webmin/Virtualmin by your VPS IP address https://xxx.xxx.xxx.xxx:10000 (accept the warning)
2. Change your Authentic theme colors (link will appear on the dashboard page)
3. **Strong Passwords (password policy enforcement)**
    a. Webmin > System > Users and Groups > Module Config (cog icon upper left corner)
    b. Scroll to Password restriction and set options:
        1) Minimum password length set to `8` or more.
        2) Prevent dictionary word passwords? set to yes
        3) Perl regexp to check password against:
        `.*[A-Z].*[0-9].*|.*[0-9].*[A-Z].*`
        4) Human-readable description of regexp:
        `Passwords must contain at least one uppercase letter and a number`
        5) Click the Save button.
4. **Mail client auto-configuration**
    a. Virtualmin > Email Settings > Mail Client Configuration
    b. Enable mail client auto-configuration? Yes
    c. Click the Save button.

Summary - VPS Creation & Configuration

1. Create your Digitalocean droplet with initial_server_setup.sh and your public SSH key.
2. SSH into VPS to register the new administrator password. If disconnected try again.
3. SFTP Transfer SnappyVPS to your home folder.
4. SSH into VPS and run install.sh.
5. Reboot.
6. Execute the manual post-installation tasks.

Main Virtual Server

Your main virtual server is the one which will hold the `mainDomain` which you specified in install.sh (e.g. mydomain.com)

Note: If you will be migrating your main domain from a remote server, migrate.sh *will run* make_virtual_server.sh *and setup your main domain as a virtual server. Therefore this step is not required in this case and you can skip to the next section on migration.*

If you will not be migrating any domains over to your new VPS, you should still create your main domain as a virtual server:

1. Manually through the Virtualmin interface or
2. Using make_virtual_server.sh or batch_virtual_servers.sh

Of course it is much easier if it is all done automatically, therefore the second option is preferred.

vs/make_virtual_server.sh

When creating a new Virtualmin virtual server SnappyVPS will:

1. Create a new virtual server if one does not exist, based on your preferences.
2. Set the default PHP version to 7.2 for domain
3. Set PHP variables for domain
   a) upload_max_filesize: 256M
   b) post_max_size: 256M
4. Create keys for domain:
   a) private and public keys for this domain
   b) import public key into ~/.ssh/authorized_keys
   c) secure files and folder
5. Set email SPF DMARC for domain
6. Install phpMyAdmin for domain
7. Configure Google Pagespeed for domain

Check out the usage page: ***sudo bash ~/snappyVPS/vs/make_virtual_server.sh subrun -h***

You can run it like this after specifying the values you want:

sudo bash ~/snappyVPS/vs/make_virtual_server.sh subrun -u -l "${logfile}" -d "${domainName}" -p "${domainPass}" -e "${email}"

batch_virtual_servers.sh (batch virtual server creation)

The creation of virtual servers can be automated by executing batch_virtual_servers.sh.

Just include your domains and passwords inside the script and execute it like so:

sudo ~/snappyVPS/batch_virtual_servers.sh -u

Do not forget

After creating a virtual server manually as described above, you should:

- Execute the manual post-migration tasks and
- Run setcronbackups.sh, which sets up backup scheduling for all virtual servers.

  Check out the usage page: ***sudo bash ~/snappyVPS/setcronbackups.sh vscronbackups -h***

  You can run it like this after specifying the values you want:

  sudo bash ~/snappyVPS/setcronbackups.sh vscronbackups -u -l "${logfile}"

  For more on backup scheduling see Backup Automation.

Domain Migration

vs/migrate_virtual_server.sh (single domain migration)

SnappyVPS can help automate migration of remote domains to your VPS by:

1. Creating a new Virtualmin virtual server (using make_virtual_server.sh).
2. Synching the remote files to the new virtual server home directory.
3. Synching the remote database and importing it into the new virtual server database.

migrate.sh (batch domain migration)

Furthermore the migration process can be wrapped up into a batch file called migrate.sh that will automate transfer of any number of domains over to your VPS. You can initiate batch migration using:

```
sudo ~/snappyVPS/migrate.sh -u
```

The above starts the batch migration process as you have configured it and runs it with the `-u` switch or unattended mode (If you want to run in manual mode, you can discard the `-u` switch). Bear in mind that if any private key specified requires a password, you will still be prompted for it, even if you choose unattended mode.

This process will:

1. Create the Virtualmin virtual servers you specified
2. Synch the remote files for each one
3. Synch & Import the remote databases of each one
4. Schedule domain and databse backups

   After all domains have been migrated, migrate.sh will run setcronbackups.sh for you, which will automatically create scheduled CRON jobs for all your Virtualmin virtual servers and their databases.

Parameters

mig_remrsa
Remote Server - RSA PRIVATE KEY in OpenSSH format. (e.g. `"${BASH_SOURCE%/*}/myprivatekey.rsa"`)

mig_remprt
Remote Server - SSH port (e.g. `22`)

mig_remusr
Remote Server – username (e.g. `"user@remotedomain.com"`)

mig_remdir
Remote Server - path to files (e.g. `"~/public_html/remotedomain/"`)

mig_remdbdir
Remote Server - path to db file. If left empty, no database migration will be attempted.

mig_vsdname
Local Virtual Server - domain name (e.g. `"mydomain.com"`)

mig_vsdpass
Local Virtual Server - domain password & also MySQL password. Use a complex, 16 character password or use a password generator like https://passwordsgenerator.net/.

mig_email
Local Virtual Server - forwarding email-address. If specified, the email address will act as an alias for the main user account in this virtual server. (recommended to leave blank for default mailbox delivery)

Post-Migration tasks

1. **RSA keys**
    a. The administrator for the newly created virtual server should log into Webmin.
    b. Download his auto-generated keys (.ppk,.pub,.rsa,.rsa.pub) from the ~/.ssh directory.
    c. **Delete these keys from the server.**
    d. Use these keys for SSH & SFTP access (e.g. PuTTY, WinSCP, rsync, MySQL Workbench etc.)
2. **Point DNS** for all migrated domains to your VPS DNS server (see Private Name Servers).
3. **DKIM Configure**
    a. Virtualmin > Email Settings > DomainKeys Identified Mail
    b. Change Signing of outgoing mail enabled? to `Yes`
    c. Selector for DKIM record name field enter a short name typically just the current year, like `2019`.
    d. Make sure all your domains are stated including '`mydomain.com`' & `mail.mydomain.com` and any others you may require (e.g. 'xxx.mydomain.com'). Click the Save button.

You can preview your website before domain propagation: Virtualmin> Select Domain> Services> Preview Website

After the domain has fully propagated with DNS:

1. Install an SSL certificate for each domain to avoid NET::ERR_CERT_AUTHORITY_INVALID in your browser.
   NOTE: There are limits to how many times you can request certificates for your domain.
    a. Virtualmin > select domain > Server Configuration > SSL Certificate > Let's Encrypt > Request Certificate

    The certificate should cover:
    - mydomain.com, www.mydomain.com, mail.mydomain.com,
    - **autoconfig**.mydomain.com, **autodiscover**.mydomain.com (for mail client autoconfiguration)
    - *For the main domain also add the DNS bind master (ns1.mydomain.com) to this list.*
    - *For static assets, also add static.mydomain.com to this list - don't forget to create an alias server under mydomain.com called static.mydomain.com, with only 'DNS domain enabled?' as an enabled feature.*

    b. Virtualmin > select domain > Server Configuration > SSL Certificate > Current Certificate
    For your main domain only, copy the certificate to the various services, for use in place of the one supplied with Webmin. Now, you can securely access Webmin/Virtualmin at https://mydomain.com:10000
2. Run some checks
3. phpMyAdmin can be accessed at http://mydomain.com/phpmyadmin/
4. Access the virtual server and verify operation.
    a. .htaccess
        - Replace `Options +FollowSymLinks` with `Options +SymLinksifOwnerMatch`
        - Switch Compression On - `AddOutputFilterByType DEFLATE`
        - Enable Keep Alive - `Header set Connection keep-alive`
        - Expires Caching - `ExpiresByType`
        - Fonts - `AddType application`
        - Pagespeed - configure and optimize the `<IfModule pagespeed_module>` section.
    b. Depending on what version is enabled on each virtual server, php.ini is found in /home/mydomain/etc/phpx.x.
    c. Your CMS config files will need modification of :
        - DB password
        - Paths to CMS directories
        - PHP version & maybe modules
5. Setup your emails.
6. Run your first backups from Webmin> System> Scheduled Cron Jobs> <choose a backup task>

Summary – Virtual Server Creation & Configuration

- Create your Main Virtual Server and/or Migrate your domains.
- In either case, execute the manual post-migration tasks

Emails

root email aliases

As stated before, since you are using SSH keys to access your VPS and a root password has not be generated/emailed to you (see resetting the root password), you will have no way of accessing the root account in order to check emails.

Any root emails can be read directly from /root/Maildir. The proper way to access these emails is by adding root aliases to postfix. This can be done manually by adding a new line such as `root: joe` to /etc/aliases, saving the file and running this command: ***newaliases***. It can also be done easily in Webmin> Servers> Postfix Mail Server> Mail Aliases> Create a new alias. This will forward any emails for root to joe the administrator's mailbox (replace joe with any mailbox or email address).

Email accounts

New email accounts can be set up manually in Virtualmin> Domain> Edit Users, or automatically as described in the next section on email migration.

You can check your emails online by Virtualmin> Domain> Edit Users> Choose User> Login to Usermin or by accessing Usermin directly https://mydomain.com:20000 and loging in with the user's credentials.

Usermin is a webmail client, with some nifty extras. Some administrators like having it on servers that have mail, because users can check mail in a browser or with their phone (there is a mobile theme that provides a pretty nice UI on WebKit enabled phones like iPhone and Android phones).

Note: There is a bug in Usermin (Usermin Version 1.751 - Authentic theme version 19.21), that will not allow you to compose a new email message, unless you open a folder that contains at least one email. If your mailbox is brand new and empty, you can bypass this problem, by just sending an email to the particular mailbox address from any email client. Also, if your Inbox is empty but you have emails in other folders, you can click on the folder containing at least one email and this will again enable you to compose new messages. This bug is documented here: https://www.virtualmin.com/node/66618

Email Migration

vs/migrate_mailboxes.sh

You can initiate batch migration of mailboxes from a remote host to local Virtualmin mailboxes using:

sudo bash ~/snappyVPS/vs/migrate_mailboxes.sh subrun -u

The above starts the batch migration process as you have configured it and runs it with the `-u` switch or unattended mode (If you want to run in manual mode, you can discard the `-u` switch).

This process will read migrate_mailboxes.txt file at the root of the snappyVPS folder and:

1. Create a new local user if one does not exist.
2. Transfer the emails from the remote host mailbox to the local mailbox specified.

Check out the usage page: ***sudo bash ~/snappyVPS/vs/migrate_mailboxes.sh subrun –h***

Synchronization

migrate_mailboxes.sh is using the imapsync utility to do the migrations.

By default, emails are synchronized from the remote host to the local host and as such, no matter how many times you run this script, you should never have any duplicate emails in any mailbox.

No emails are ever removed from either hosts, but this can be changed using extra arguments (see below).

Also folder mapping is done automatically by default, but this can also be changed using extra arguments.

Accounts file

By default migrate_mailboxes.sh will look for the file migrate_mailboxes.txt file at the root of the snappyVPS folder for the definitions of all mailbox transfers. Alternatively you can use the ***-accounts*** parameter to specify another file.

The file can contain comments and should contain one line for each migration from remote to local mailbox, like this:

```
remotehost.com;info@mydomain.com;remotepassword;localhost;info@mydomain.com;localpassword;
```

The first three parameters concern the remote mailbox while the last three parameters concern the local mailbox.

It is important to note that it is better to only use 'localhost' as the first parameter for any local mailboxes. Using localhost will ensure that connection to local mailboxes will never fail.

Accounts such as *abuse, hostmaster, postmaster and webmaster* are considered aliases for the main domain account within Virtualmin and as such should be migrated to the main account for the domain e.g.

```
remotehost.com;abuse@mydomain.com;remotepassword;localhost;mydomain@mydomain.com;localpassword;
```

Test before Migrate

You can do test runs before commiting to the actual migration by providing the ***-j*** or ***-d*** switches which will not migrate any emails, but will only verify login to both hosts or do a dry run respectively.

Extra arguments

You can provide any additional imapsync specific parameters using the ***-e*** switch.

Check out ***imapsync --help*** for all commandline arguments you can pass to migrate_mailboxes.sh.

Backup Automation

Backups are scheduled by setcronbackups.sh, which is executed from migrate.sh or manually.

Note: *Domain backups are temporarily stored in /tmp/.webmin before being transferred offsite and deleted, so* ***make sure that your VPS has as much free disk space as your largest virtual server****. Cached files can gradually increase a virtual server's capacity to more than double its original size – this can cause backups to fail.*

Note: *If you will not be migrating any domains to your VPS,* ***make sure you run this script manually after creation of any virtual server****. This will ensure disaster recovery.*

setcronbackups.sh

It will schedule jobs to backup each database daily and each domain (including all **Virtualmin configuration settings**) on the first of each month. Functionality can of course be changed to your specifications.

It can of course be run manually with the `-u` switch or unattended mode (If you want to run in manual mode, you can discard the `-u` switch).

sudo bash ~/snappyVPS/setcronbackups.sh vscronbackups -u

Parameters

ftpHost
The remote backup FTP server e.g. `"xxx.xxx.xxx.xxx:21"`

ftpUser
The username on the remote ftpHost.

ftpPass
The password for the ftpUser.

ftpRoot
The backup folder on the remote ftpHost. Default: `"/VPSBackups"`

ftpPathDom
The sub-directory for domain backups. Default: `"$ftpRoot/myVS/\%Y-\%m-\%d-VA/"`.
This will create subfolders such as '/VPSBackups/myVS/2019-06-15-VA'

ftpPathDB
The sub-directory for DB backups. Default: `"$ftpRoot/myVS/'\`date +\%Y-\%m-\%d\`'-DB/"`.
This will create subfolders such as '/VPSBackups/myVS/2019-06-15-DB'

bakPath
The local directory used to temporarily store database backups. SnappyVPS will delete any backups older than 3 days found in this directory. Default: `'/var/backups/VSdatabases'`

dbBakFilePre
A prefix can be specified for database backup files. Default: `''`

dbBakFileExt
The extension for all DB backups. Default `'.sql.gz'`

domWd, domHr, domMi, [domDt]
For domain scheduling, these correspond to the WeekDay, Hour & Minutes for the task to be executed: Default: 1,2,0. This will execute domain backups on the 1st weekday of each month @ 02:00. Each additional backup will start domDt minutes after the previous one. Default: 30 minutes.

dbWd, dbHr, dbMi, [dbDt]
For DB scheduling, these correspond to the WeekDay, Hour & Minutes for the task to be executed: Default: *,1,0. This will execute domain backups on any weekday of each month @ 01:00. Each additional backup will start domDt minutes after the previous one. Default: 5 minutes.

Remote Backup Targets

According to Virtualmin documentation for virtualmin backup-domain, backup destinations for virtual servers can be:

```
- A local file, like /backup/yourdomain.com.tgz
 - An FTP destination, like ftp://login:pass@server/backup/yourdomain.com.tgz
 - An SSH destination, like ssh://login:pass@server/backup/yourdomain.com.tgz
 - An S3 bucket, like s3://accesskey:secretkey@bucket
Multiple destinations can be given, if they are all remote.
```

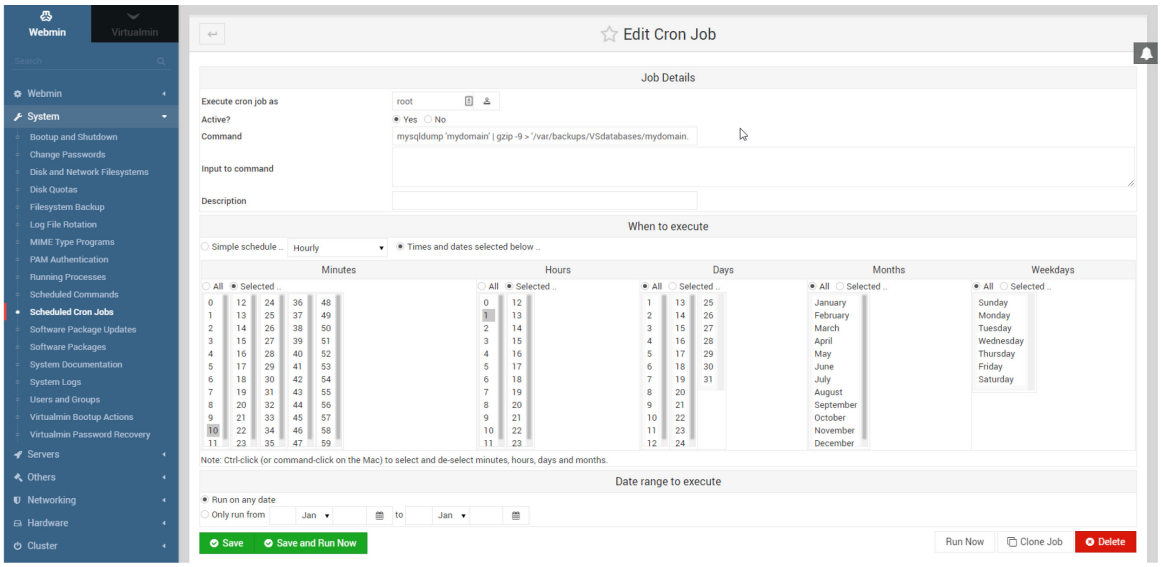
SnappyVPS currently provides an FTP target for backups. You may modify setcronbackups.sh and configure it to your own needs. The same goes for database backups. There are countless options for customization to your specifications.

Backup Task Execution

Backup tasks can be found in Webmin> System> Scheduled Cron Jobs> and are scheduled in pairs, one for the domain and another for the database e.g.

```
virtualmin backup-domain --domain 'mydomain.com' …
mysqldump 'mydomain'…
```

To execute a task manually, just click on the task and on the task page click 'Run Now'.



Introduction to Domain Level Backups (see Virtualmin Backup & Restore)

The simplest way to backup your virtual servers is to use the backup feature built into the Virtualmin UI, which can save them to local or remote files domain by domain. This allows you to restore the state of an entire virtual server (including all databases, users and aliases), without effecting other parts of the system.

Each virtual server's backup is typically a single file in tar.gz format. This contains one or more files per Virtualmin feature that is included in the backup, such as the contents of databases, DNS records, Apache directives or the virtual server's home directory. *It is also possible for a single backup file to contain multiple servers, although this format is generally not as easy to work with.*

By default, only the master administrator (typically root or admin) can perform backups, but it is possible to grant backup and restore rights to resellers and domain owners too. Only the master admin can restore all virtual server settings though, as some (such as the Apache configuration) could harm other system functions if a corrupt or malicious backup was restored.

Restoring from Backups

Restoring Virtual Servers (see Virtualmin Backup & Restore)

If a virtual server has been accidentally deleted, or lost files, database content or users, you can restore some or all of its data from a backup. In the case where the whole domain is gone, Virtualmin will re-create it for you as part of the restore process. ***The steps to restore a domain are:***

Open the Backup and Restore category on the left menu, and click on Restore Backup

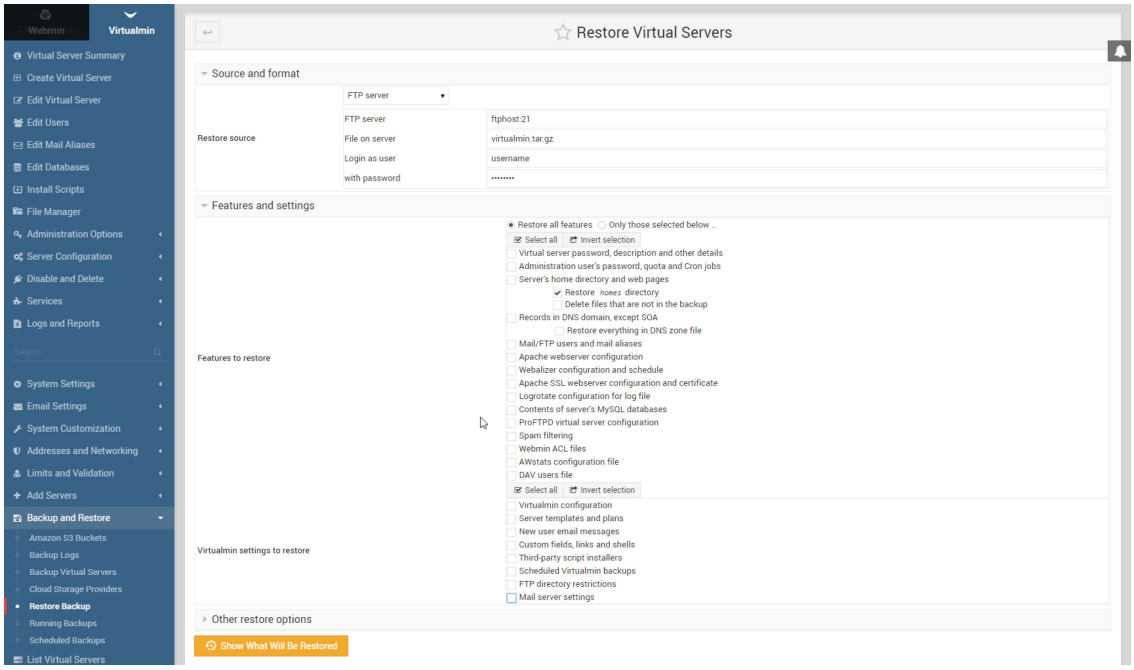
In the Restore source section, enter the local or remote path to restore from. If you are just restoring one server, it is best to enter the full path to its backup file, like /backup/example.com.tar.gz

Under Features and settings, you can control if all attributes of the virtual server are restored (overwriting any that it currently has), or just some. For example, if you wanted to restore just the domain's databases, you could select only those selected below and then check the box next to the MySQL feature. By default, everything will be restored.

Click the Show What Will Be Restored button at the bottom of the page.

After this step, the backup will be downloaded from its source FTP or SSH server and a confirmation page displayed showing which domains and features will be restored. Be careful when restoring existing domains, as any aliases, databases or mailboxes that they have will be removed and replaced with those in the backup.

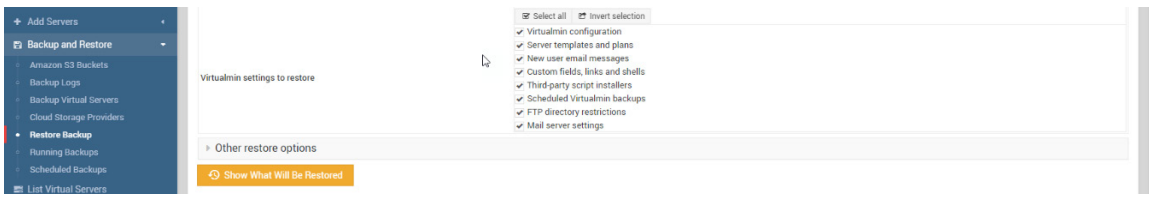
If everything looks OK, click the Restore Now to begin the restore process. As it runs, the progress of each domain and feature will be displayed by Virtualmin.



Restoring Global Settings (source virtualmin.com)

The various global Virtualmin configuration settings can be restored in exactly the same way as you would restore domains. Just select the virtualmin.tar.gz file as the source to restore from, and in the Virtualmin settings to restore section check the types of global settings to include.

This can even be used to migrate templates, custom fields, script installers, content styles, resellers and email messages to a new system. Be careful when migrating the Module configuration though, as it may not be compatible with the new system if running a different operating system or Linux distribution. Instead, it is safer to configure the target the way you want it, then restore domains and possibly other global settings.



Disaster Recovery shortlist

Full VPS Rebuild

1. You can fully rebuild your VPS in a couple of ways:
    a. Either rebuild your droplet:
        i. Follow Retain IP Address on new droplet if you do not want to lose your IP address, or
        ii. Follow VPS droplet creation if you do not care to lose your IP address. Because your IP address will change, you must also see Changes that require updates to files.
    b. Or if you have created a Digitalocean snapshot of your VPS you can restore from snapshot, following the relevant Digitalocean guide.
2. Restore data from application-level backups stored on your remote backup server:
   Virtualmin> Backup and Restore> Restore Backup
    a. Restore the most recent Virtualmin Domain Level Backups **including all Virtualmin settings** and
    b. Restore the most recent Database backups for all virtual servers.
3. Finally run setcronbackups.sh to schedule backups again.

Virtual Server Restoration

1. Restore data from application-level backups stored on your remote backup server:
   Virtualmin> Backup and Restore> Restore Backup
    a. Restore the most recent Virtualmin Domain Level Backup for this VS (**do not restore Virtualmin settings**) and
    b. Restore the most recent Database backup for this VS.
2. Finally run setcronbackups.sh to schedule backups again.

Virtualmin Global Settings Restoration

If you have made configuration mistakes in the Virtualmin control panel, it may be necessary to restore the settings from backup. SnappyVPS makes sure that these settings are backed up along with the domain backups once a month.

1. Restore data from application-level backups stored on your remote backup server:
   Virtualmin> Backup and Restore> Restore Backup
    a. Restore only the most recent Virtualmin settings (file virtualmin.tar.gz in your doman backups)

Changes that require updates to files

If any of the following changes, you will probably need to reflect the change in your SnappyVPS files and/or elsewhere.

VPS IP address changed

1. Point your DNS records to the new IP!
2. Add the new IP to the FTP Backup Server allowed list (if applicable).
3. snappyVPS-P4T/p4t.bat - IP of VPS must be changed here if we want to upload SnappyVPS.
4. SSH & SFTP clients (e.g. PuTTY, WinSCP, rsync, MySQL Workbench etc.)

Home IP address changed

1. config_fail2ban.sh & /etc/fail2ban/jail.local
2. install_pagespeed.sh & /etc/apache2/mods-available/pagespeed.conf

   Either modify the configuration files directly, or run the scripts again to add your new IP.
3. config_webmin.sh & /etc/webmin/miniserv.conf

   You MUST modify the configuration file directly, since you probably cannot login to Virtualmin. Otherwise, change it in Webmin> Webmin Configuration> IP Access Control.

The SnappyVPS scripts themselves do not have hardcoded IPs in them, only the VPS configuration files do.

FTP Backup Server IP

1. Modify setcronbackups.sh
2. Then run it to update the existing CRON jobs or modify them manually.

Personal email address changed

1. config_security_tools.sh & /etc/default/rkhunter.

   Either modify the configuration files or run the script again to do it for you.
2. config_unattended_upgrades.sh & /etc/apt/apt.conf.d/60unattended-upgrades

   Either modify the configuration file directly, or run the script again to do it for you.
3. migrate.sh *uses email for forwarding.*

   Virtualmin> Select Domain> Edit Users> username> Mail forwarding settings> Forward to other addresses> Save.

   Virtualmin> Select Domain> Edit Mail Aliases> (All Users)

The SnappyVPS scripts themselves do not have hardcoded IPs in them, only the VPS configuration files do.

SSH keys changed

1. Droplet creation interface – add your new key.
2. snappyVPS-P4T/p4t.bat - PPK of server must be changed here if we want to upload configuration files.
3. In all other cases, the new private key must be stored
   a. Locally in C:\ProgramData\Microsoft\Windows\Start Menu\Programs\PuTTY\
   b. Modify apps that may need to use it – (e.g. PuTTY, WinSCP, rsync, MySQL Workbench etc.)
   c. Upload the public key to the server, in the user's authorized keys (/home/user/.ssh/authorized_keys).

Firewall

Do not Activate at Boot and do not activate Directly edit firewall rules. SnappyVPS has not been tested with these options.

Linux IPTables

Webmin installs firewalld as the default firewall management program for IPtables. This is fine except for the fact that during extensive testing on a new DigitalOcean droplet with Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-52-generic x86_64), firewalld fails to start in 30% of the reboots. This has been documented in various places:

https://unix.stackexchange.com/questions/526440/ordering-cycle-firewalld-dbusexception

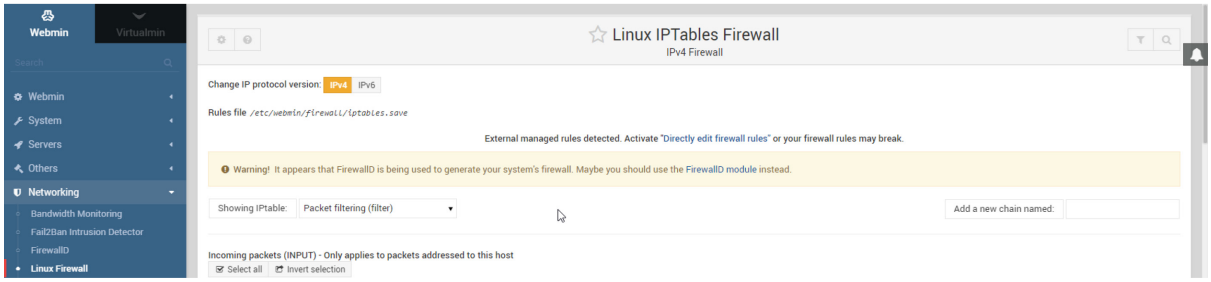
https://www.virtualmin.com/node/66358

The problem is probably caused by dependency ordering cycles during reboot. The result is that ultimately firewalld is not running sometimes and this is unreliable. Consequently SnappyVPS default payload will disable firewalld and UFW and rely on IPTables entirely. Management of firewall rules can be very easily done through Webmin Linux Firewall pages. This solution is 100% reliable on every reboot.

Webmin will still think that firewalld is still managing IPtables and will issue a warning in the Linux IPTables Firewall page.

```
Warning!  It appears that FirewallD is being used to generate your system's
firewall. Maybe you should use the FirewallD module instead.
```

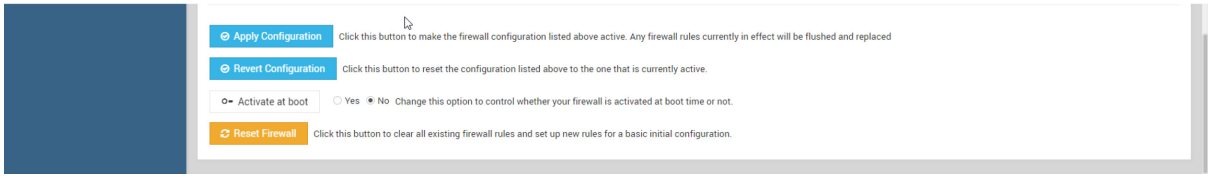
This can be disregarded and it is due to the fact that SnappyVPS retains all the firewalld rules before disabling it.



IPtables Management & Persistence

Additionally, SnappyVPS will install a service (iptables.service-IPTables/Webmin Persistence Service) responsible for more closely integrating IPTables with the Webmin Linux IPTables Firewall page. The service will run at startup and shutdown to load and save respectively the IPTables rules to and from a file (/etc/webmin/firewall6/ip6tables.save). This file is also being read and saved to by the Linux IPTables Firewall page. As such all firewall management can be done by this page and is persistent across reboots.

Since by default Webmin uses a file as a buffer between itself and IPtables, always remember to click Revert Configuration before making changes and applying them.



Blocklists

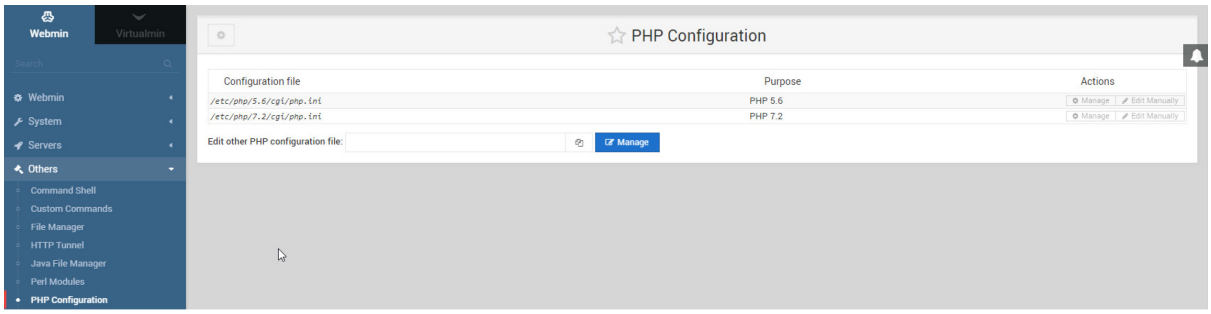
SnappyVPS will also enable *Country blocking and Threat Blocking* lists. Both are scheduled to run at boot time through rc.local and hourly through a CRON job. Threat blocking requires no configuration. However if you need to block whole countries, you will need to edit /usr/local/bin/snappyVPS/blocklists.sh and include these countries in the field `blockedCountries` (space separated two letter, lowercase ISO country codes – see http://www.ipdeny.com/ipblocks/)

Fail2Ban

SnappyVPS implements an aggressive fail2ban policy, which will block an IP after 3 failed logins, for a month. This applies for ssh, webmin, proftp, postfix and dovecot.

PHP

To see all PHP versions available go to Webmin> Others > PHP Configuration.



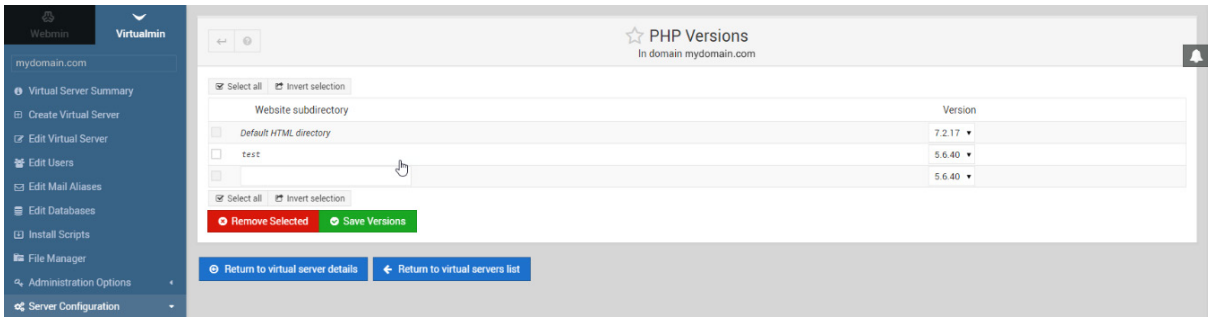
Default PHP version

SnappyVPS will set PHP 7.2 as the default for all new virtual servers. You can configure which one is the default in Virtualmin> System Settings> Server Templates> Default > PHP Options> Default PHP version.

Virtual Server PHP version

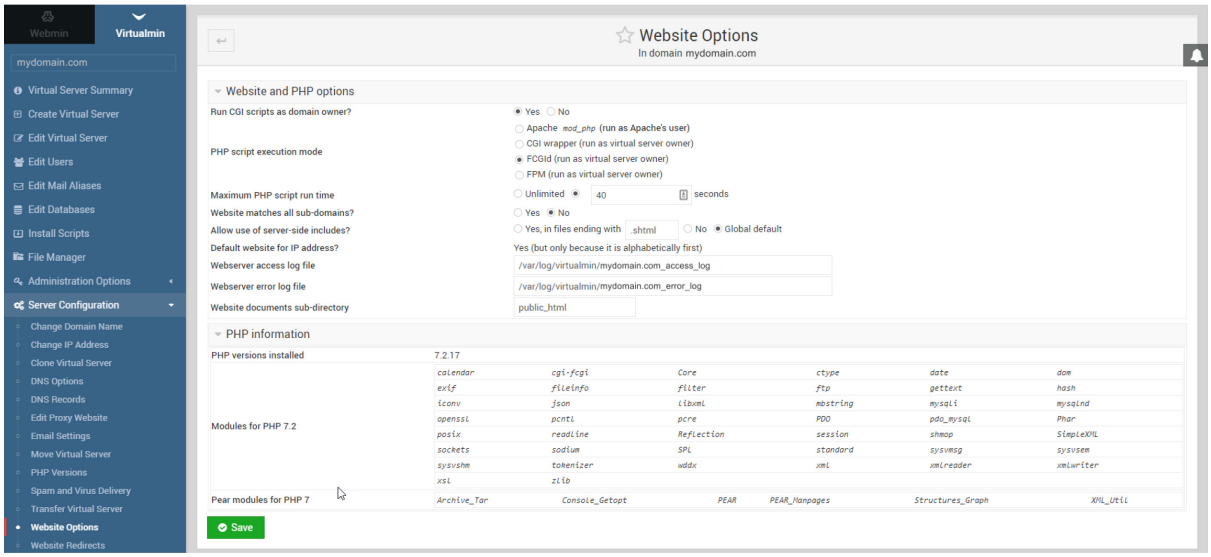
You can configure the PHP version being used for a specific Virtual Server by selecting Virtualmin> Select Domain> Server Configuration> PHP Versions.

The first line there specifies what PHP version will be used by default. If you wish, you can specify a different PHP version to be used for a specific subdirectory.



PHP Modules

To see which PHP modules are installed go to: Virtualmin> Select Domain> Server Configuration> Website Options



Google Pagespeed

SnappyVPS prepares Pagespeed on the VPS by default. By itself a dedicated VPS will not do much to improve CMS performance. Pagespeed is essential with many CMSs if performance needs improvement. With proper configuration, gains in the range of 20%-40% can be achieved even on a simple 1CPU 1GB VPS.

Below we include two examples of Pagespeed optimization sections for .htaccess for Opencart with Journal theme. You will need to run tests with your own CMS in order to optimize its performance.

Journal 2.x Opencart Theme

Journal 2.x requires many Pagespeed features in order to boost performance. Paste the Journal 2.x pagespeed configuration below into .htaccess for the virtual server concerned.

```
## PAGE SPEED OPTIMIZATION
<IfModule pagespeed_module>
   ModPageSpeed on

   ModPagespeedLoadFromFileCacheTtlMs 31536000000

   ### RewriteLevel ###
   ModPagespeedRewriteLevel CoreFilters

   ### STANDARD FILTERS ###
   ModPagespeedEnableFilters remove_comments,collapse_whitespace,elide_attributes,trim_urls

   # EXTERNAL ASSETS - (REQUIRES ModPagespeedFetchHttps enable)
   ModPagespeedDomain http://fonts.googleapis.com
   ModPagespeedGoogleFontCssInlineMaxBytes 20000
   ModPagespeedEnableFilters inline_google_font_css

   ### ESSENTIAL FILTERS ###
   ModPagespeedEnableFilters lazyload_images
   ModPagespeedEnableFilters responsive_images
   ModPagespeedEnableFilters responsive_images_zoom
   ModPagespeedEnableFilters insert_image_dimensions
</IfModule>
```

Journal 3.x Opencart Theme

Journal 3.x is quite capable on its own without pagespeed. However, with some basic pagespeed functionality, we can boost mainly the mobile side of things and the desktop side by a little bit**.** For best performance it also needs the jpegoptim and opting programs installed. The only additional action to take on Journal 3xx is to arrange for static asset delivery. This should gain some more points.

Paste the Journal 3.x pagespeed configuration below into .htaccess for the virtual server concerned.

```
## PAGE SPEED OPTIMIZATION
<IfModule pagespeed_module>
   ModPageSpeed on

   ModPagespeedLoadFromFileCacheTtlMs 31536000000

   ### RewriteLevel ###
   ModPagespeedRewriteLevel CoreFilters

   ### STANDARD FILTERS ###
   ModPagespeedEnableFilters remove_comments,collapse_whitespace,elide_attributes,trim_urls

   # EXTERNAL ASSETS - (REQUIRES ModPagespeedFetchHttps enable)
   ModPagespeedDomain http://fonts.googleapis.com
   ModPagespeedGoogleFontCssInlineMaxBytes 20000
   ModPagespeedEnableFilters inline_google_font_css
</IfModule>
```

Pagespeed Admin Page

To be able to access the pagespeed_admin or pagespeed_global_admin pages you need to add the following two lines to your .htaccess file, before redirecting to index.php

```
# if pagespeed_admin or pagespeed_global_admin exist go directly to it
RewriteCond %{REQUEST_URI} !/pagespeed_admin
RewriteCond %{REQUEST_URI} !/pagespeed_global_admin
RewriteRule ^([^?]*) index.php?_route_=$1 [L,QSA]
```

Companion Sections

Add the following sections to your .htaccess file if they do not exist, or modify your own.

Fonts

```
# Fonts
# Add correct content-type for fonts
<IfModule mod_mime.c>
    AddType application/vnd.ms-fontobject    .eot
    AddType application/x-font-opentype      .otf
    AddType image/svg+xml                    .svg
    AddType application/x-font-ttf           .ttf
    AddType application/font-woff            .woff
    AddType application/font-woff2           .woff2
</IfModule>
```

Compression

```
## SWITCH COMPRESSION ON
 <IfModule mod_deflate.c>
  AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javascript
text/cache-manifest text/vcard text/vnd.rim.location.xloc text/vtt text/x-component text/x-
cross-domain-policy
  AddOutputFilterByType DEFLATE application/x-javascript application/javascript
application/ecmascript
  AddOutputFilterByType DEFLATE application/json application/ld+json
application/manifest+json application/x-web-app-manifest+json application/schema+json
application/vnd.geo+json
  AddOutputFilterByType DEFLATE application/xml application/rss+xml application/rdf+xml
application/atom+xml application/xhtml+xml
  AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
  AddOutputFilterByType DEFLATE application/x-font application/x-font-opentype
application/x-font-otf application/x-font-truetype application/x-font-ttf application/x-
font-woff
  AddOutputFilterByType DEFLATE font/woff2 font/truetype font/opentype font/otf font/ttf
font/eot application/font-woff application/font-woff2
  AddOutputFilterByType DEFLATE image/svg+xml image/x-icon image/bmp
image/vnd.microsoft.icon
 </IfModule>
# END DEFLATE COMPRESSION
```

Caching

```
## EXPIRES CACHING ##
# Serve resources with far-future expires headers.
# (!) If you don't control versioning with filename-based
# cache busting, you should consider lowering the cache times
# to something like one week.
#
# https://httpd.apache.org/docs/current/mod/mod_expires.html
<IfModule mod_expires.c>
    ExpiresActive on
    ExpiresDefault                                  "access plus 1 month"
  # CSS
    ExpiresByType text/css                          "access plus 1 year"
  # Data interchange
    ExpiresByType application/atom+xml              "access plus 1 hour"
    ExpiresByType application/rdf+xml               "access plus 1 hour"
    ExpiresByType application/rss+xml               "access plus 1 hour"
    ExpiresByType application/json                  "access plus 0 seconds"
    ExpiresByType application/ld+json               "access plus 0 seconds"
    ExpiresByType application/schema+json           "access plus 0 seconds"
    ExpiresByType application/vnd.geo+json          "access plus 0 seconds"
    ExpiresByType application/xml                   "access plus 0 seconds"
    ExpiresByType application/pdf                   "access plus 1 month"
    ExpiresByType application/x-shockwave-flash     "access plus 1 month"
    ExpiresByType text/xml                          "access plus 0 seconds"
  # Favicon (cannot be renamed!) and cursor images
    ExpiresByType image/vnd.microsoft.icon          "access plus 1 month"
    ExpiresByType image/x-icon                      "access plus 1 month"
  # HTML
    ExpiresByType text/html                         "access plus 15 minutes"
  # JavaScript
    ExpiresByType application/javascript            "access plus 6 month"
    ExpiresByType application/x-javascript          "access plus 6 month"
    ExpiresByType text/javascript                   "access plus 6 month"
  # Manifest files
    ExpiresByType application/manifest+json         "access plus 1 week"
    ExpiresByType application/x-web-app-manifest+json   "access plus 0 seconds"
    ExpiresByType text/cache-manifest               "access plus 0 seconds"
  # Media files
    ExpiresByType audio/ogg                         "access plus 1 year"
    ExpiresByType image/bmp                         "access plus 1 year"
    ExpiresByType image/gif                         "access plus 1 year"
    ExpiresByType image/jpeg                        "access plus 1 year"
    ExpiresByType image/jpg                         "access plus 1 year"
    ExpiresByType image/png                         "access plus 1 year"
    ExpiresByType image/svg+xml                     "access plus 1 year"
    ExpiresByType image/webp                        "access plus 1 year"
    ExpiresByType video/mp4                         "access plus 1 year"
    ExpiresByType video/ogg                         "access plus 1 year"
    ExpiresByType video/webm                        "access plus 1 year"
  # Web fonts
    # Embedded OpenType (EOT)
    ExpiresByType application/vnd.ms-fontobject     "access plus 1 year"
    ExpiresByType font/eot                          "access plus 1 year"
    # OpenType
    ExpiresByType font/opentype                     "access plus 1 year"
    ExpiresByType application/x-font-opentype       "access plus 1 year"
    # TrueType
    ExpiresByType font/truetype                     "access plus 1 year"
    ExpiresByType font/ttf                          "access plus 1 year"
    ExpiresByType application/x-font-ttf            "access plus 1 year"
    # Web Open Font Format (WOFF) 1.0
    ExpiresByType application/font-woff             "access plus 1 year"
    ExpiresByType application/x-font-woff           "access plus 1 year"
    ExpiresByType font/woff                         "access plus 1 year"
    # Web Open Font Format (WOFF) 2.0
    ExpiresByType application/font-woff2            "access plus 1 year"
    ExpiresByType font/woff2                        "access plus 1 year"
  # Other
    ExpiresByType text/x-cross-domain-policy        "access plus 1 week"
</IfModule>
# ----------------------------------------------------------------------
## EXPIRES CACHING ##
```

Appendix A

VPS Configuration

VPS Payload System Requirements

SnappyVPS default payload, contains Virtualmin which as of June 2019 supports the following distributions. However SnappyVPS only contains commands designed for Ubuntu and therefore you should choose Ubuntu for your VPS. Of course, you may change the configuration files where needed so that they can apply for any other distribution.

Recommended:

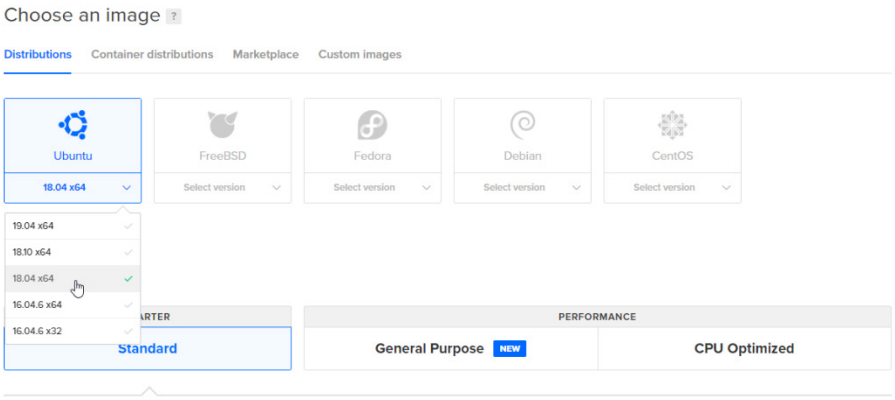
1. Ubuntu 18.04LTS on i386 and amd64 (non-LTS releases are not supported)
2. 1GB RAM minimum for default payload - 2GB RAM minimum if clamdscan & spamc are enabled.

VPS droplet creation

a) Visit https://www.digitalocean.com/ and open an account.

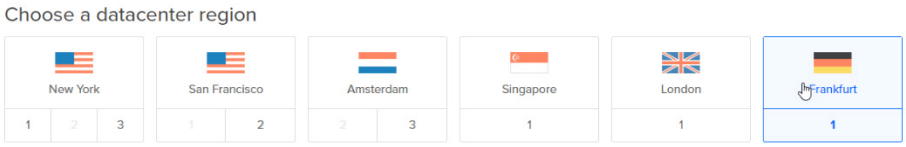
b) Click the link to create a new droplet.

c) Choose the maximum OS supported by SnappyVPS & Virtualmin: **Recommended Ubuntu 18.04.**



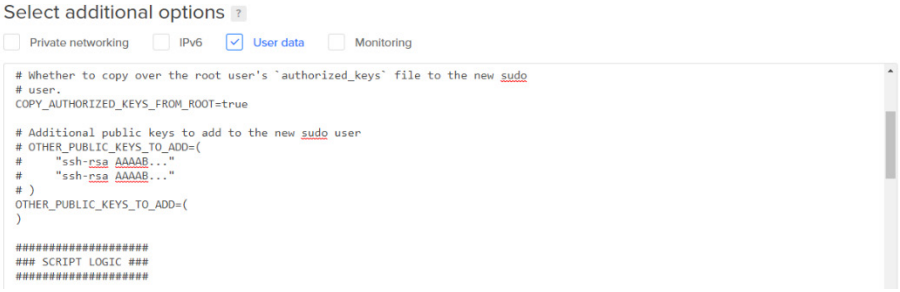
d) Choose a plan. Any plan will do.

e) Choose a region close to your target audience:

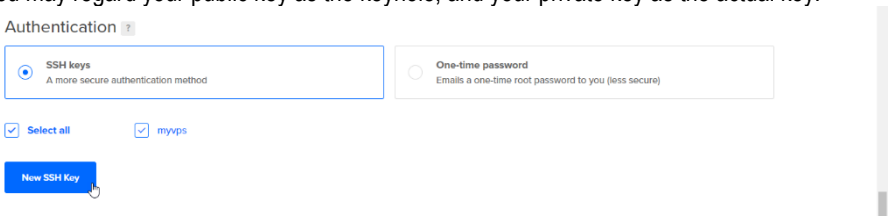


f) Select additional options click User data and paste the contents of initial_server_setup.sh

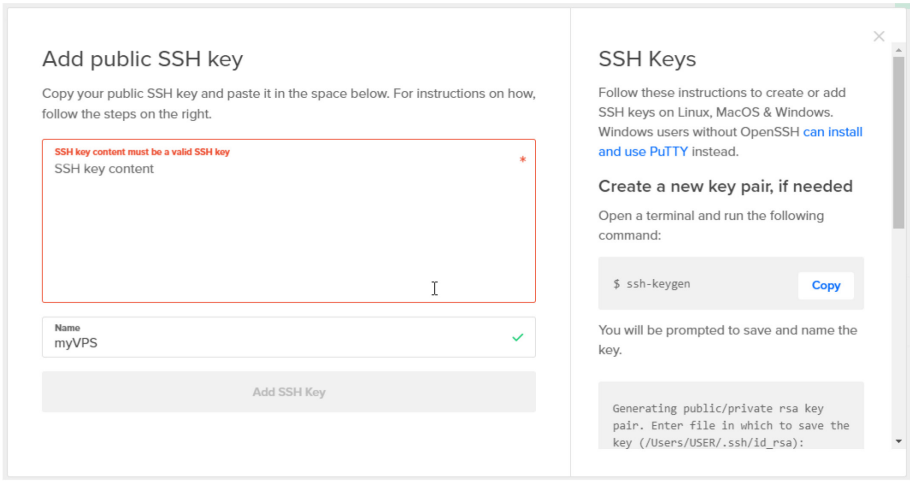
More info can be found here "Automating Initial Server Setup with Ubuntu 18.04".



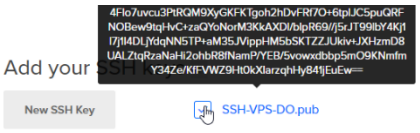
g) In section 'Authentication', click 'New SSH Key' and follow the instructions to create your keys. You may regard your public key as the keyhole, and your private key as the actual key.



Paste your public key into the box provided and give it a unique name.



When finished, you will end up with something like this:

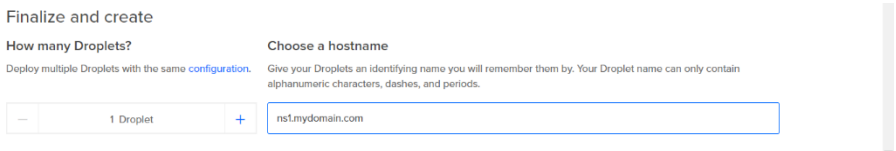


Your private key will be used with SSH & SFTP to connect.

h) In section 'Finalize and create' specify your hostname as recommended by Virtualmin:

> *We recommend you choose a name that is not one for which you will be receiving mail, in order to simplify later configuration. A good choice is to use a name server designator, such as "**ns1.virtualmin.com**". Some customers also choose something like "host1.virtualmin.com" or "primary.virtualmin.com".*

SnappyVPS recommends ns1 to be used as the prefix for easier configuration.



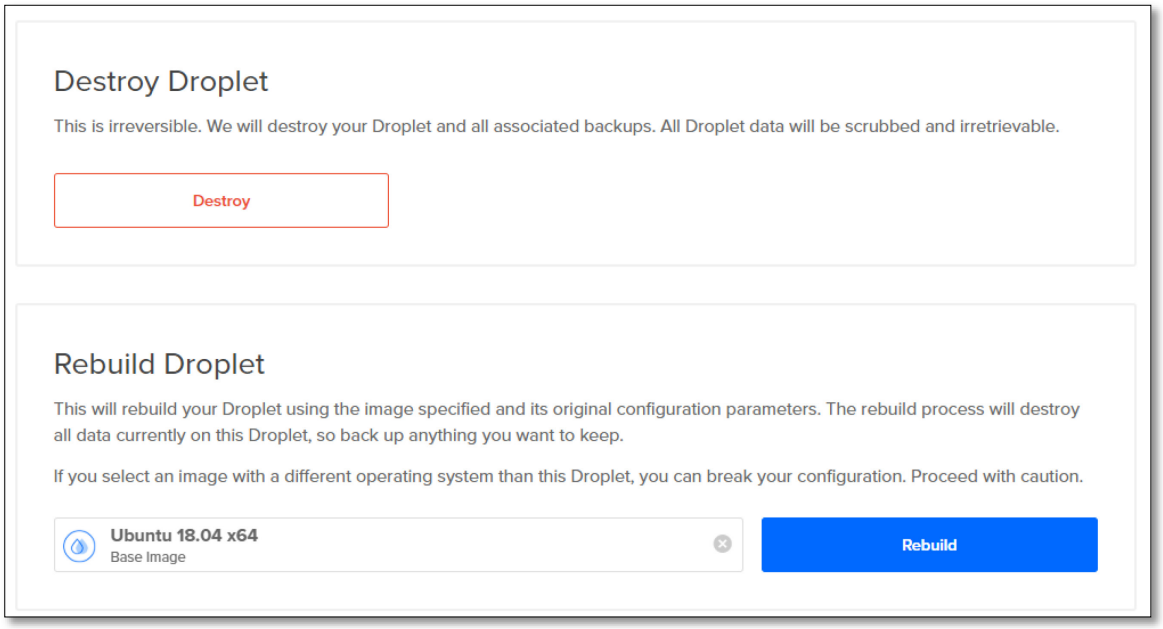
i) Finally click create to start the droplet creation process.

j) When the droplet has been created, you will be provided with the VPS IP address which you can use to login.

Retain IP Address on new droplet

Instead of destroying the droplets you will instead navigate to the destroy page in the control panel but **use the rebuild option**. The drop-down menu will display all disk images you have available (make sure the snapshot you have is available in all regions you'll need it in from the Images page first). Select your snapshot from the dropdown and choose "Rebuild from Image".

Your droplet will be rebuilt using your snapshot image, permanently destroying anything currently on the droplet, but keeping the IP address.



Note:

Once the Droplet has been rebuilt, it will have a new fingerprint, also known as a remote host identification key. Because local SSH clients store the fingerprints of the servers they connect to, you may see a warning about the differing fingerprint when you reconnect to the rebuilt Droplet:

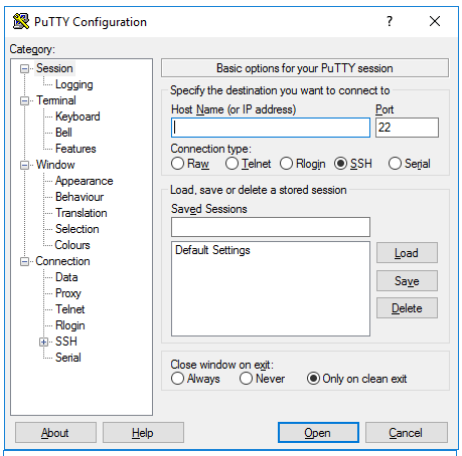
```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:RqX4d+VC6sBaOSMEo8JgyjpvmoQTQY4E6EYe7vCQV5c.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /root/.ssh/known_hosts:3
  remove with:
  ssh-keygen -f "/root/.ssh/known_hosts" -R 203.0.113.47
ECDSA host key for 203.0.113.47 has changed and you have requested strict checking.
Host key verification failed.
```

To resolve this, accept the update. This removes the old key, which will let you connect to the Droplet as normal without seeing the warning.

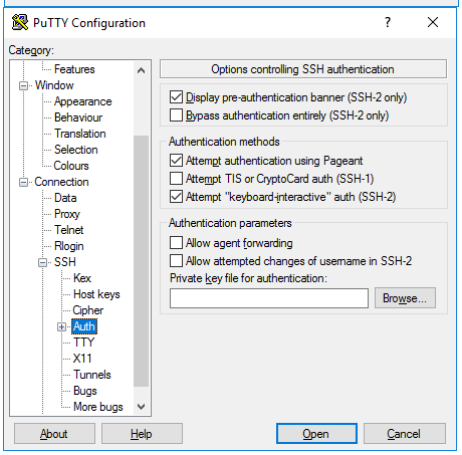
SSH into the VPS

PuTTY

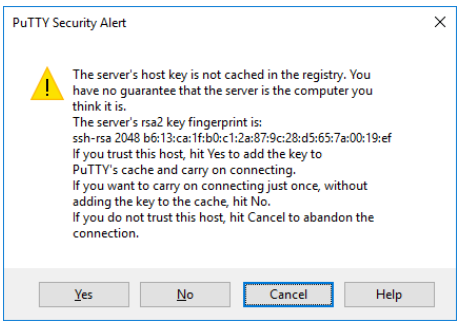
We can use an SSH client like PuTTY to connect to the VPS & issue commands.



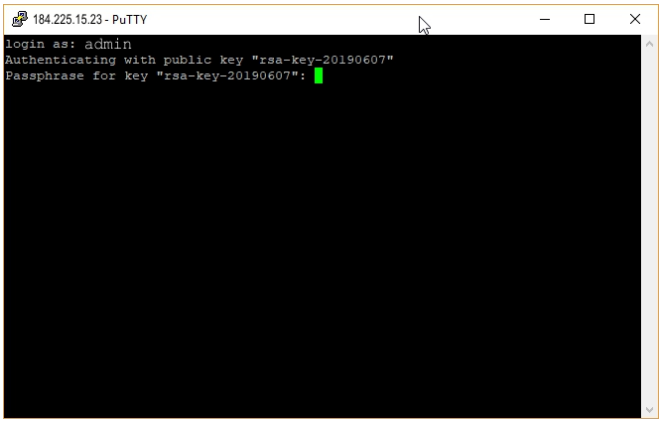
1. Enter the VPS IP address in the Host Name field.
2. Leave port as 22.
3. The connection type should be SSH.
4. Give your session a unique name.
5. Click 'Save'.



6. In the SSH> Auth section, browse to the location of your private key created specifically for your VPS.
7. Scroll back up and click on 'Session'.
8. Click 'Save' again.
9. Click 'Open' to connect.



1. On a new/rebuilt VPS you will be asked if you want to trust the server fingerprint to be entered into your key storage.
2. Click 'Yes'.

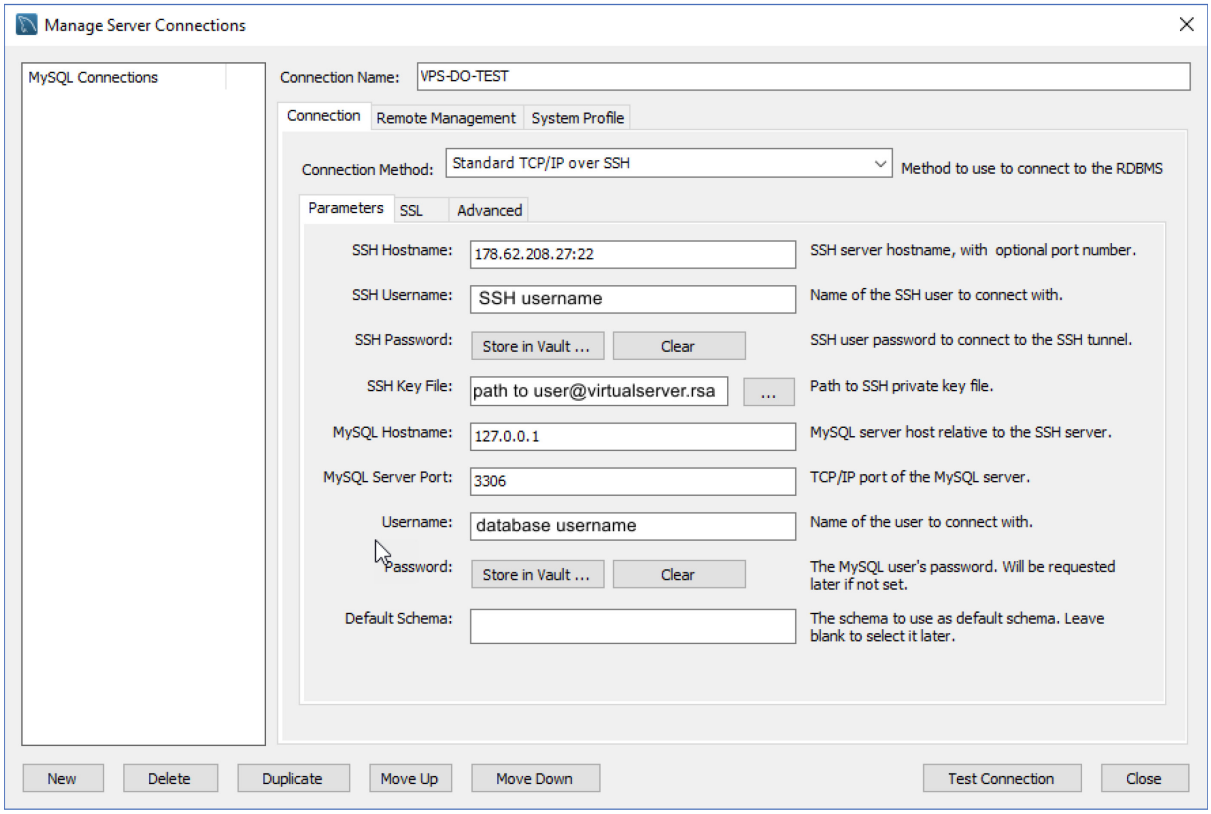


Login using your newly created administrator username, created by the installation script.

If your private key was created with a password, you will also be prompted to provide that.

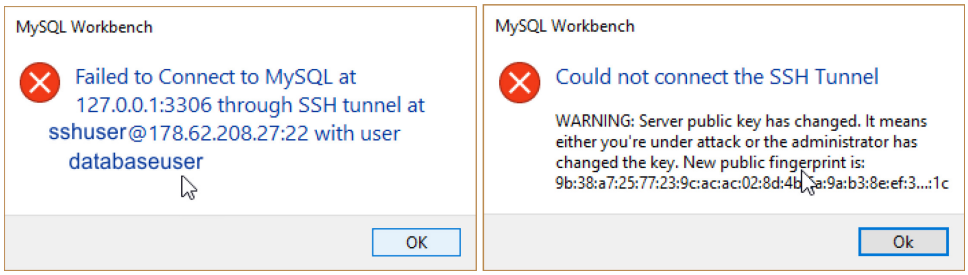
MySQL Workbench

MySQL Workbench is perfect for desktop use, and we can also connect with it remotely using SSH. The private key is basically the same key as for PuTTY and WinSCP, but exported in OpenSSH format.

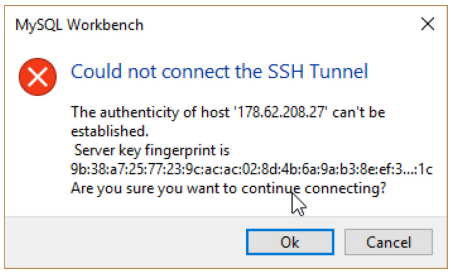


key public fingerprint changed

If the VPS was rebuilt you will start getting these kinds of messages which is normal for all SSH clients. However, Workbench has a problem allowing you to continue if the key public fingerprint has been changed.

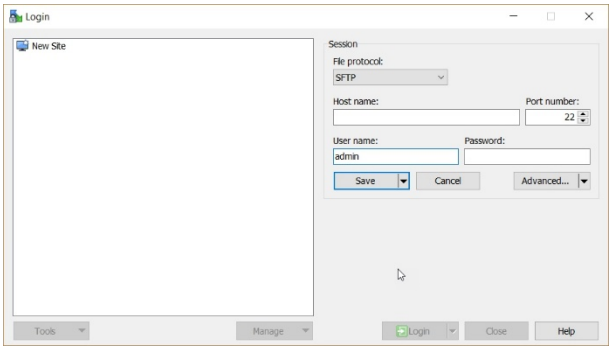


In order to overcome this problem, you should modify the file '%userprofile%\.ssh\known_hosts'. The old fingerprint for the server you are connecting to should be removed. It will then allow you to connect.

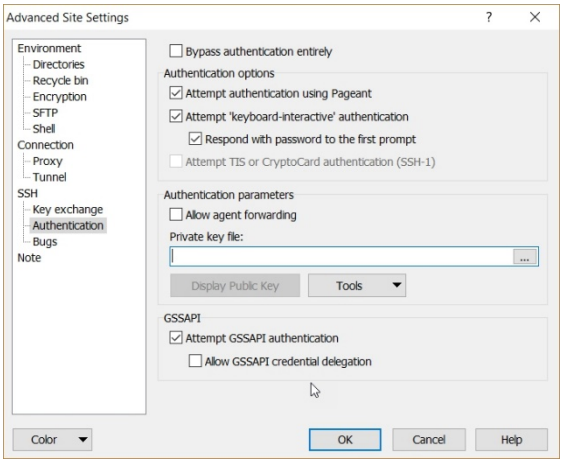


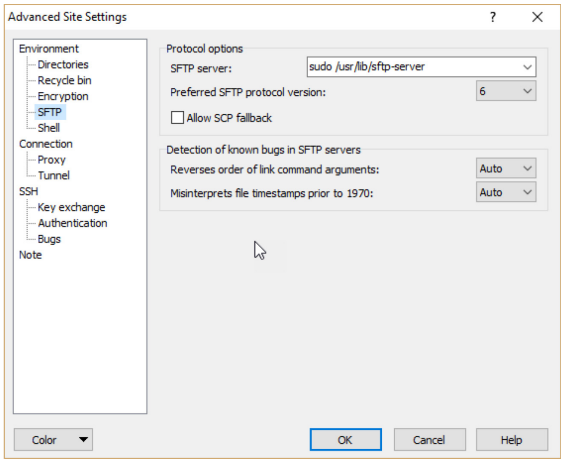
SFTP into the VPS

To transfer files to and from your VPS, you can use a SFTP client like WinSCP.

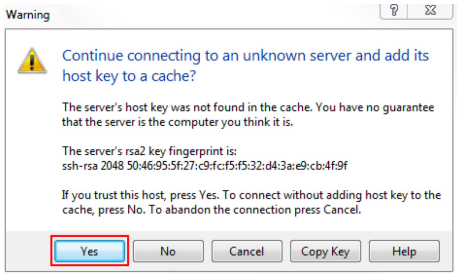


1. Open your client and create a new SFTP connection.
2. Protocol: SFTP
3. Host name: your VPS IP address
4. Port: 22
5. Username: your sudo user
6. Password: no password required
7. Click 'Advanced'

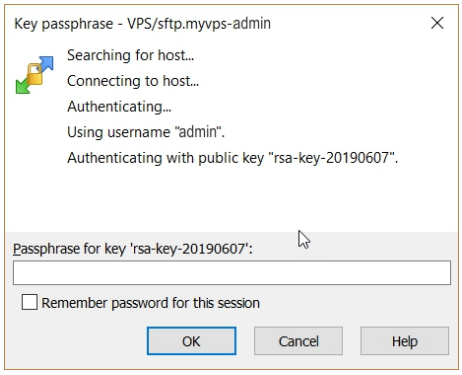




1. In the SSH> Authentication section, browse to the location of the private key you created specifically for this VPS.
2. If the connection is for the sudo user (main administrative account) then in the Environment> SFTP section, enter `sudo /usr/lib/sftp-server` in the SFTP server field, otherwise leave it as `Default`.
3. Click 'OK', 'Save', and 'Login' to connect.



1. On a new or rebuilt VPS you will be asked if you want to trust the server fingerprint to be entered into your key storage.
2. Click 'Yes'.



3. If your private key was created with a password, you will also be prompted to provide that. In that case, click 'Remember password for this session' and 'OK' to login.

Converting to UNIX

Windows files use the same format as Dos, where the end of line is signified by two characters, Carriage Return or CR or \r followed by Line Feed or LF or \n. UNIX files, on the other hand, use only Line Feed (\n).

When editing SnappyVPS scripts on a Windows computer, you will probably need to convert these to UNIX before executing them. To do this you can use the dos2unix command which converts a DOS text file to UNIX format.

You can:

1. Download dos2unix for Windows from https://sourceforge.net/projects/dos2unix/ or
2. Install it on your VPS by ***sudo apt install dos2unix***

Automatic Conversion

To aid in the conversion of all scripts in the snappyVPS folder, a batch conversion script p4t.bat is included for Windows users in the snappyVPS-P4T directory.

With p4t.bat you can convert all scripts to UNIX, compress the whole snappyVPS directory into a zip archive and transfer the file to a remote server, ready for installation.

Requirements

Before running p4t.bat you will need to download a few required utilities and place them in the same directory.

1. dos2unix for windows (from http://gnuwin32.sourceforge.net/packages/cygutils.htm)
    a. dos2unix.exe (cygutils-1.3.2-bin.zip)
    b. libiconv2.dll (cygutils-1.3.2-dep.zip)
    c. libintl3.dll (cygutils-1.3.2-dep.zip)
    d. popt1.dll (cygutils-1.3.2-dep.zip)
2. zip.exe (from http://gnuwin32.sourceforge.net/packages/zip.htm)
    a. zip.exe (zip-3.0-bin.zip)
    b. bzip2.dll (zip-3.0-dep.zip)
3. Command-line SCP/SFTP client (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
    a. 32 bit pscp.exe or
    b. 64 bit pscp.exe

Parameters

`usrAdmin` – the VPS administrator username
`vpsIP` – the VPS IP address
`ppk` – the path to the VPS private key file
`folder` – the full path to the snappyVPS directory
`d2u_exc` – space separated file extensions to be excluded from dos2unix conversion
`zip_exc` – space separated wildcard file descriptions to be excluded from zip

Execution

To aid execution of p4t.bat, create a shortcut to the file, right-click the shortcut> Advanced> Run as administrator. Now you can double-click the shortcut and p4t.bat will run with administrative privileges.

Webmin/Virtualmin/Usermin

Webmin

A Perl-based administration interface which, unlike cPanel, allows you to control every aspect of your server, either visually or manually, through the use of web forms. It also features a cool java file manager which allows you to get a visual idea of what's on your HDDs and it can perform basic file operations on them. In terms of security, you can restrict access to its interface by specifying a list of IPs or classes of IPs. You can fine tune BIND, Apache and the mail server by using the visual configuration of/or the manual configuration.

Support: Webmin offers good support for Ubuntu and it can give you good information about outdated packages plus the possibility to update them. It also has a couple of modules which were specially designed for Ubuntu administration tasks.

Virtualmin

A module of Webmin. You can't have Virtualmin without Webmin. Virtualmin is for web hosting servers. If you are hosting websites, then you probably want Virtualmin, as it provides a single point of management for all of the configuration files that go into having a "website" (Apache vhosts, DNS, mail, databases, log analysis, users, etc.).

If you intend to handle multiple domains then Virtualmin is the best choice as it allows you to manage a domain in a centralized way, that is, it automatically takes care of DNS zones, email aliases and Apache vhosts.

Additionally it provides administrators with a CLI and a remote API.

Usermin

A webmail client, with some nifty extras. If you need to manage your server, then you need Webmin, or Webmin+Virtualmin. Usermin may or may not be a useful addition. Some administrators like having it on servers that have mail, because users can check mail in a browser or with their phone (there is a mobile theme that provides a pretty nice UI on WebKit enabled phones like iPhone and Android phones).

Webmin Payload

SnappyVPS default payload installs Webmin in full, not the minimal installation. This includes:

1. Webmin/Virtualmin/Usermin
2. BIND
3. Fail2ban
4. Firewalld
5. Postfix
6. Dovecot
7. Procmail

1. LAMP stack
2. ClamAV
3. SpamAssassin
4. MySQL
5. ProFTPd
6. AWStats
7. Webalizer

Full Install vs. Minimal Install

The full LAMP or LEMP stack, plus a full mail processing stack including SpamAssassin and ClamAV, is quite large, requiring about **1GB minimum system memory in order to function well** (and more is better). *If you're using a lower memory system, it's not recommended, and maybe not even possible, to run the full mail stack along with LAMP or LEMP.*

So, we provide an installation option, `--minimal`, that leaves off much of the mail processing stack. The installed components can still send and receive mail from local processes, but SMTP authentication and IMAP/POP are not installed or configured, and spam and AV scanning will need to be outsourced to a remote system (for example, a Cloudmin Services host). The minimal installation type can probably operate OK with only 512MB of RAM.

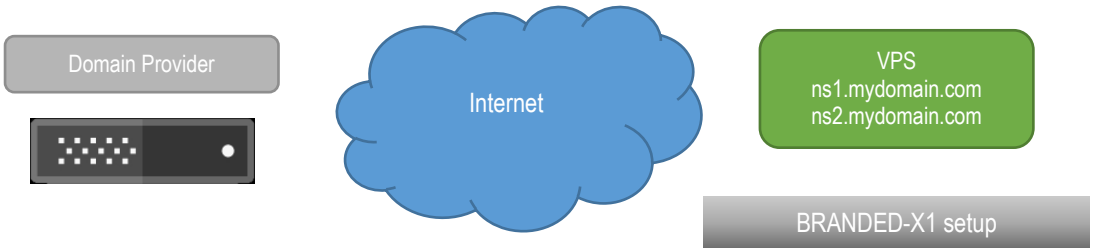
If you want to locally host mail for end users (including IMAP/POP clients), the minimal installation target is probably not the right choice. (source virtualmin.com)

SnappyVPS has determined that 1GB is quite enough for the full installation on a not-so-busy system, if you do not enable the Spam & Clam daemons. If the system is destined for production or if you determine that the system is not handling to expectations, you can easily upgrade your droplet with more RAM.

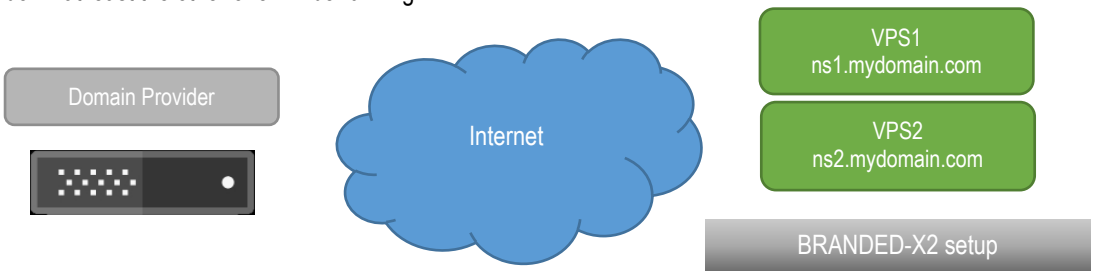
For more information visit: www.virtualmin.com, Virtualmin Documentation, Virtualmin CLI, Virtualmin Remote API

Private Name Servers

SnappyVPS with its default payload will install and configure a single Bind DNS Server.



It is recommended for a VPS to have two such DNS servers, for resilience, one being a backup of the other. If one goes down at least the other one will be running.



It is up to the administrator to decide to add another VPS and configure a second BIND DNS server on there to serve the first VPS as the ns2.mydomain.com name server.

For more info see:

1. how-to-create-vanity-or-branded-nameservers-with-digitalocean-cloud-servers
2. DNS Master/Slave servers -how-to-setup-dns-slave-auto-configuration-using-virtualmin-webmin-on-ubuntu

Branded vs Vanity

When running your own DNS servers on a VPS, Digitalocean calls this as using a *Branded Name Server*, as opposed to using a *Vanity Name Server*. Vanity Name Servers do not require a BIND DNS server to be installed on the VPS, since all DNS configuration is done on the Digitalocean DNS manager page, basically using the provider's own DNS servers.

SnappyVPS has opted to make use of a BIND DNS server installtion on the VPS for these reasons:

1. You do not have to use a specific provider's DNS manager interface, thus your VPS is portable. This means you can transfer the VPS anywhere and your DNS configuration will be transferred also.
2. You do not have to rely on your domain provider's hosting plan, in order to access their DNS manager interface to point the A record to your new VPS.

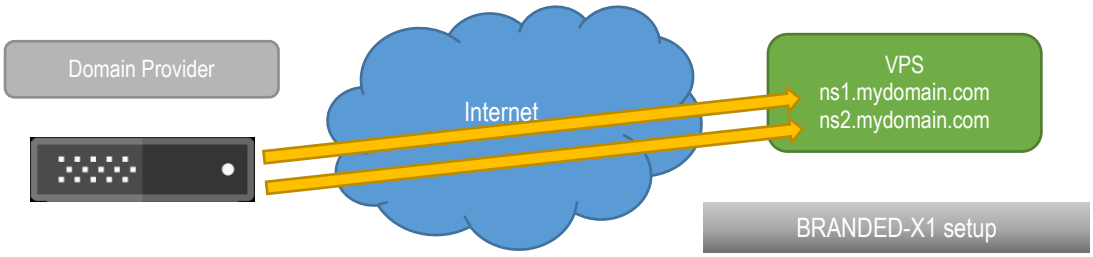
Very simply, you do not have to rely on any other DNS server for your VPS to work, because you have your own DNS server. The only issue now is that you might want to add a second DNS server for resilience, which is a must for servers that will go into production, but not a high priority for a personal VPS.

Branded-X1 setup

So the default SnappyVPS payload will result in a Branded-X1 setup. Bear in mind that ns1.mydomain.com and ns2.mydomain.com both exist in this setup, only difference being that they both point to the same BIND server, whereas Branded-X2 has 2 different IP addresses for these domains.

All that remains is to point your domains to your own DNS server at ns1.mydomain.com and ns2.mydomain.com.

Name Server Records

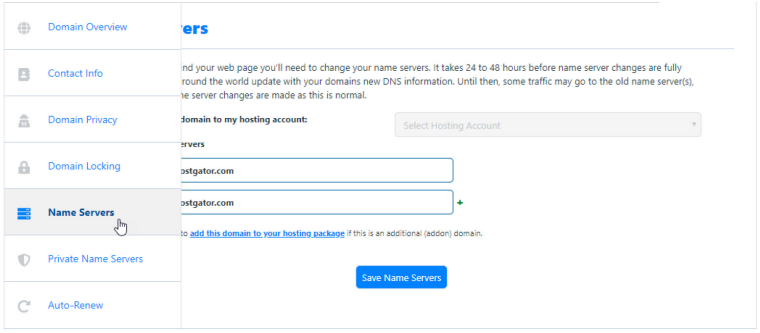


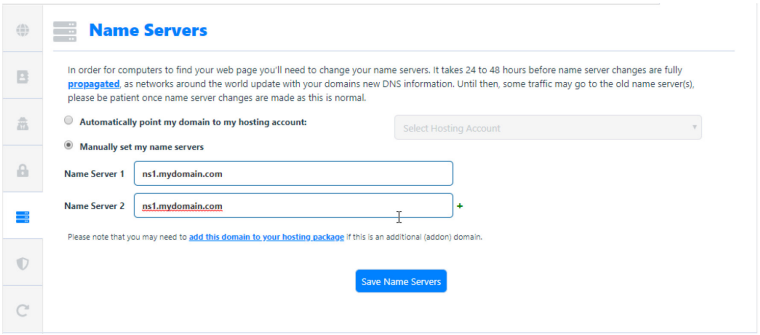
Name Server Records are basically pointers to your ns1 & ns2. You will need to create Name Server Records that point to one or more of your droplet's IP addresses which contain the DNS Servers. These should be setup at your domain registrar (e.g Hostgator).

It's usually as simple as inputting two or more names (such as ns1.example.com and ns2.example.com), and the IP addresses that are associated with them. How to accomplish that is different for every domain name registrar.

Hostgator

Login to the customer portal and click Domains.
Find your domain and change both Name Servers and Private Name Servers:



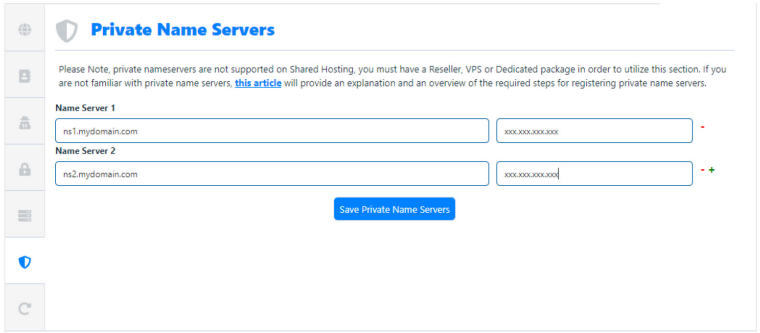


Old Servers:

ns6281.hostgator.com
ns6282.hostgator.com

New Servers:

ns1.mydomain.com
ns2. mydomain.com



Add your name servers also in this section including their IP addresses.

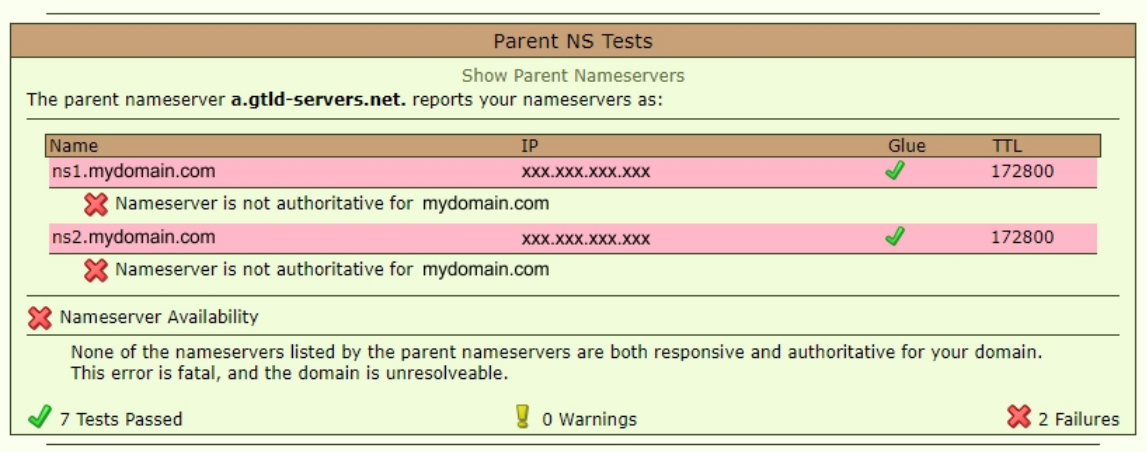
If everything is configured properly, your domain should now be using your own VPS DNS configuration.

Contact your own registrar for instructions on how to configure your Name Server Records.

Configuration Tests

DNS propagation

As soon as you change your Name Server Records, you can check DNS propagation, here: http://leafdns.com/



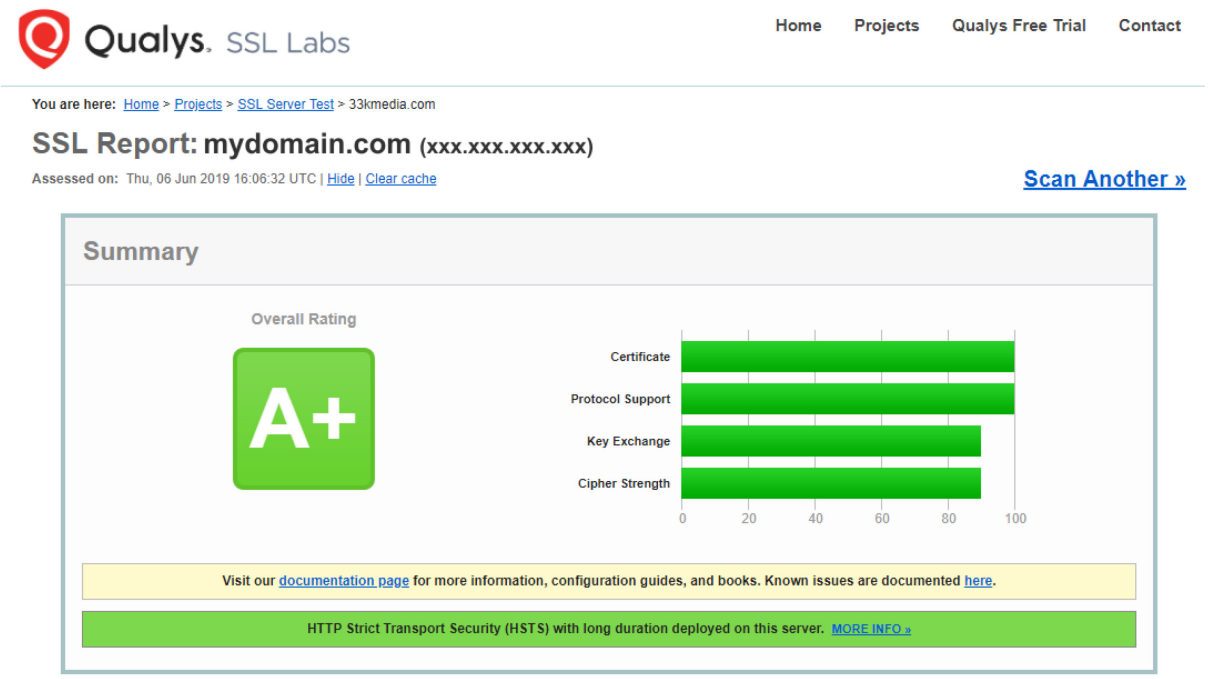
If you notice such failures, then your DNS is not propagating. Check your DNS configuration on the VPS.

If all else fails, reset your Name Server Records back to the registrar's original values. Then attempt the Name Server record change again.

You will know when DNS propagation is complete, when you open a command prompt and ping your domain. Once the domain ping returns your new IP address, your domain has propagated fully.

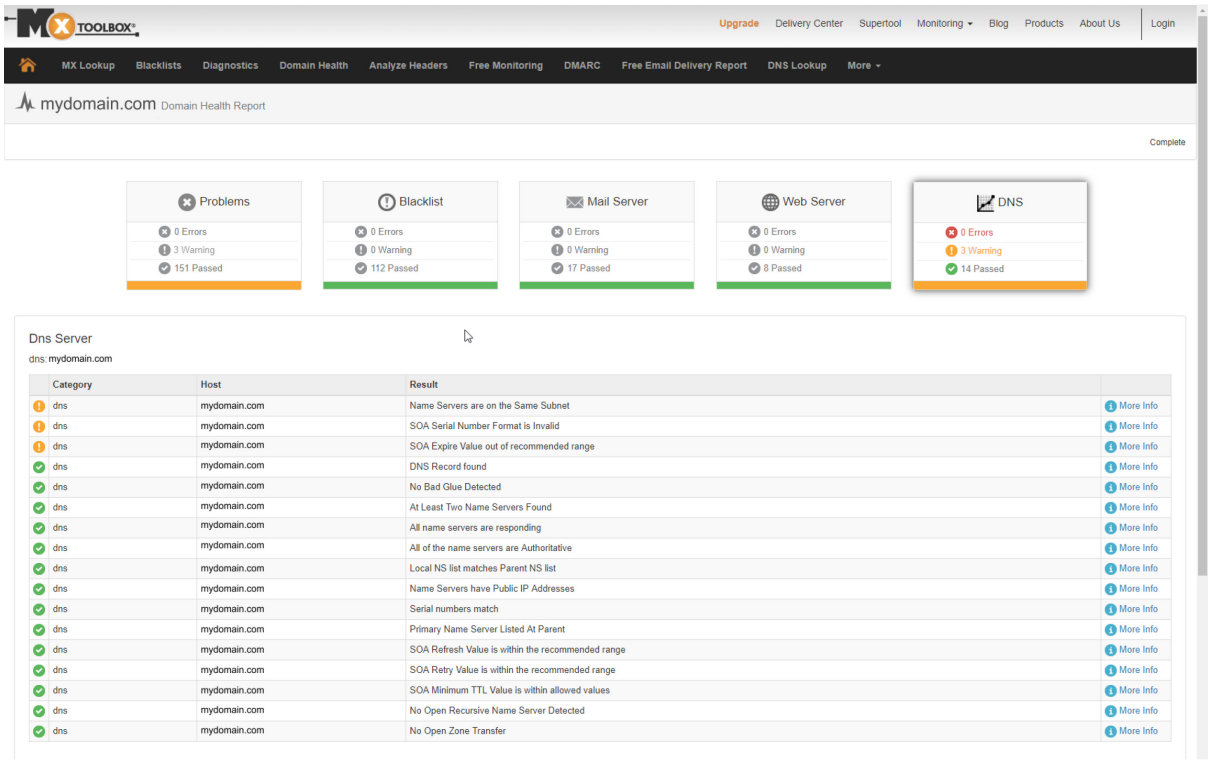
SSL

Open https://www.ssllabs.com/ssltest - You should have A+.



Domain Health Report

Open https://mxtoolbox.com/domain/ - You should have only 3 warnings.



Warnings:

1. Name Servers are on the Same Subnet.
   Can be ignored when the ns1 & ns2 are the same IP.
2. SOA Serial Number Format is Invalid.
   Can be ignored because mxtoolbox is being a nuisance here.
   see https://serverfault.com/questions/750430/soa-serial-number-format-is-invalid-warning-by-mxtoolbox-com-why
3. SOA Expire Value out of recommended range.
   Can be ignored since Virtualmin specifies 1 week instead of 2 weeks recommended.

> *Your DNS records are hosted on two or more DNS servers that are supposed to be in regular contact with each other so that they have up to date copies of your DNSrecords. The Expire Value setting tells each slave server how long it is allowed to continue giving out authoritative replies after it has no longer heard from the master server. RFC 1912 recommends 1209600 - 2419200 seconds (14-28 days).*
>
> *How long a secondary will still treat its copy of the zone data as valid if it can't contact the primary? This value should be greater than how long a major outage would typically last, and must be greater than the minimum and retry intervals, to avoid having a secondary expire the data before it gets a chance to get a new copy. After a zone is expired a secondary will still continue to try to contact the primary, but it will no longer provide nameservice for the zone. 2-4 weeks are suggested values. [RFC1912]*

More checks

You can run more checks on this page: https://mxtoolbox.com/NetworkTools.aspx

DMARC

Open https://mxtoolbox.com/SuperTool.aspx?action=dmarc%3a&run=toolpage

You should have:

	Test	Result
✓	DNS Record Published	DNS Record found
✓	DMARC Record Published	DMARC Record found
✓	DMARC Syntax Check	The record is valid
✓	DMARC External Validation	All external domains in your DMARC record are giving permission to send them DMARC reports.
✓	DMARC Multiple Records	Multiple DMARC records corrected to a single record.
✓	DMARC Policy Not Enabled	DMARC Quarantine/Reject policy enabled

DKIM

Open https://mxtoolbox.com/dkim.aspx. In the Selector field enter the value specified for Selector for DKIM record name in Virtualmin > Email Settings > DomainKeys Identified Mail

You should have:

	Test	Result
✓	DNS Record Published	DNS Record found
✓	DKIM Record Published	DKIM Record found
✓	DKIM Syntax Check	The record is valid
✓	DKIM Public Key Check	Public key is present

Also open http://www.isnotspam.com and send an email to the address provided. To send the email you must log into Usermin at https://mydomain.com:20000/ and click New message (create an email address for this domain first).

The report results should be:

```
=============================================================
Summary of Results
=============================================================

SPF Check : pass
Sender-ID Check : pass
DKIM Check : pass
SpamAssassin Check : ham (non-spam)
```

A similar service is https://dkimvalidator.com although the Spamassassin behavior there seems different on dkim signatures, maybe due to a different version.

Appendix B

SnappyVPS Framework

Framework System Requirements

Used as a UNIX installation framework with your own payload, SnappyVPS has no O.S. or RAM requirements. It only requires BASH and PERL installed on your system.Your own payload will define the requirements in this case.

To use SnappyVPS with your own payload, delete all folders except inc and create your own folders and scripts.

Framework Structure

SnappyVPS is comprised of a number of folders containing files which carry the payload logic and a number of root files which initiate the payload (see full directory listing here).

Below is a list of folders contained in the default SnappyVPS payload.

Firewall (payload)

Files pertaining to IPtables firewall configuration.

Inc (core)

Common files essential to the framework. Used by almost all other scripts.

Mod (payload)

External modules.

Post (payload)

Post-configuration subroutines. Exclusively used by install.sh when configuring the VPS.
Each file is its own entity and can be executed separately (only within the framework).

Startup (payload)

Files used for VPS startup. You can modify the file to add your own commands.

VS (payload)

Virtual server subroutines. Exclusively used in sever migration by migrate.sh and in virtual sever creation.
Each file is its own entity and can be executed separately (only within the framework).

Root Directory (initiate payload)

Contains install.sh, migrate.sh and setcronbackups.sh.

Script Anatomy

SnappyVPS framework scripts are written in Bash script, each divided into several sections:

1. License and Disclaimer
2. Sources
3. Local Variables
4. Sudo Check
5. Usage/Help
6. Input Parameters
7. Main Program
8. Wrapping Up

We will look into each section and its features in detail next.

Features

Source Files

Global variables and functions are available to each script when a line similar to the following is included:

```
source "${BASH_SOURCE%/*}/inc/inc_globals.sh"
```

`${BASH_SOURCE%/*}` translates to the full path of the running script. *The source should therefore be adjusted to point to the required include file, relative to the location of the running script e.g.*

```
source "${BASH_SOURCE%/*}/../inc/inc_globals.sh"
```

Global Variables

All global variables are defined in inc_globals.sh and become available to each script like this:

```
source "${BASH_SOURCE%/*}/inc/inc_globals.sh"
```

These variables were created to remain static – DO NOT MODIFY.
These describe verbosity, whether to screen or to a log file.

```
_VALL=2             # everything will be written
_VERR=1             # only errors
_VOFF=0             # nothing will be written
```

These describe error codes returned by each script when checking input arguments.

```
E_ERRARGS=100       # Error code
E_BADARGS=101       # Error code
```

The rest of the variables were created to be configurable and can be modified.
Default verbosity for screen and log file based on static declarations above:

```
VERSCR="$_VERR"
VERLOG="$_VALL"
```

Default verbosity for screen and log file, based on static declarations above. **Affects only sub-processes.**

```
# subprocess configuration
EXEVERSCR="$_VERR"        # 2=all 1=err 0=no
EXEVERLOG="$_VALL"        # 2=all 1=err 0=no
```

Defaults **only for sub-processes**.

```
EXEONELINE=true           # true=errors not output to display
                          # (only if EXEVERSCR= $_VERR)
EXEMSGPRE='|>'            # command title prefix
EXESLPDUR=0.2             # hourglass refresh interval
EXEMODE=4                 # hourglass presets - values: 0 to 4
EXEMSGERR="[ ERROR ]"     # error message
EXEMSGOK="[ Done! ]"      # success message
EXEMAXTE=30000            # max ms waiting for executed process
                          # to wrapup & exit before killing it
```

Default messages.

```
UNATTENDEDMSGY="RUNNING MODE: Unattended"
UNATTENDEDMSGN="RUNNING MODE: Manual"

msgSUCC="SUCCESS! Task complete."
msgFAIL="ERROR! Task failed."

DISCLAIMER="WARNING: This script will make changes to your files. By
proceeding you assume full responsibility for any consequences."
```

Global Variable Override

At run time, each script checks its input parameters. All Snappy VPS scripts have the following default parameters:

```
-h --help                       This usage page
-u --unattended                 Run unattended

-v --verbscreen                 Screen-Verbosity
                  0             output off
                  1             only errors
                  2             everything

-z --verblog                    Log-Verbosity
                  0             output off
                  1             only errors
                  2             everything

-l --log          <path>        Path-to-Logfile
```

In particular, using the `-v` & `-z` parameters, one can override the default settings for verbosity, not only for the current running process, but also for any sub-process that may be executed.

The `-l` parameter allows each process to output to a specific log file. If no such parameter is specified each script outputs to its own log file.

The `-u` switch allows each process to run without user intervention. The exception to this is when a password is required with SSH access to remote servers as in migrate.sh.

Finally all scripts feature a usage page with the `-h` switch.

Title Templates

Various title presets are defined in inc_colors.sh. These can be used in the main scripts in order to display one-line titles and title boxes. The title templates in inc_colors.sh are based on lower level templates in inc_titleBox.sh. One could create any type of title or box by modifying these templates or creating their own.

All title presets are available to each script when the following lines are included at the very top:

```
source "${BASH_SOURCE%/*}/inc/inc_titleBox.sh"
source "${BASH_SOURCE%/*}/inc/inc_colors.sh"
```

e.g. `vpssubTitleBox "${subJobTitle}"`

The above command will display a title box according to the preset subroutine `vpssubTitleBox` in inc_colors.sh.

e.g. `vpssubTitleBox "${subJobTitle}" | prt 2`

The above command will display the same title box, however it will pipe the output through the `prt` subroutine in `inc_functions.sh`.

The argument `2` specified here is the verbosity description of the title being output. This number is compared against the allowed verbosity at run time. If either the screen or the log file allows such high verbosity the title will appear where it is allowed.

As already stated in the previous sections, this general verbosity is defined by the default values in inc_globals.sh and can be overridden on a per script basis using script parameters.

e.g. `vpssubTitleNotice "${notice}" | prt 1 >&2`

The above command will attempt to display a one-line title whose verbosity importance is 1 (equivalent to an error). It will pipe the output through the `prt` subroutine in `inc_functions.sh`, and if verbosity allows it will appear on screen and/or log file. Additionally, it will push the output into STDERR, so that the error handling trap can be activated.

As a rule of thumb, any value given to `prt` which is `<=1` will always display, unless verbosity has been completely switched off.

Global Functions

All global functions are defined in inc_functions.sh and become available to each script like this:

```
source "${BASH_SOURCE%/*}/inc/inc_functions.sh"
```

Global functions do not require any user modification which is not advised.

These functions are responsible for various common tasks such as error handling, user input, root privilege checking, printing to screen and log files but most importantly for sub-process control (`exe subroutine`).

Local Variables & Input Parameters

All local variables and their default values should be defined before checking the input parameters.

It is trivial to extend the default input parameters to include additional ones. To do this you have to modify the `###  VERIFY FUNCTION ARGUMENTS ###` section.

```
OPTIONS=huv:z:l:
LONGOPTS=help,unattended,verbscreen:,verblog:,log:
```

The two lines above describe exactly the same options: e.g. h =>help, u=>unattended, v:=>verbscreen: etc.

Notice that OPTIONS contains no commas, whereas LONGOPTS requires commas.

Also note that some entries have a colon '`:`' as a suffix. This means that the option is freeform and is expecting a value to be provided by the user. On the contrary, options without a colon are plain switches which require no user input.

There is one more difference between these two types of parameters, in the `while…do` loop.

```
-u|--unattended)
        unattended=1
        shift
;;
```

`-u` being a switch requires no input and therefore we use `shift` here and just specify a value that will be changed in a previously initialized variable `unattended`. We should always define our variable defaults in the `###  LOCAL VARIABLE DEFAULTS ###` section, before we check our input parameters.

```
-l|--log)
        logfile="$2"
        shift 2
;;
```

`-l` being a parameter that requires user input also requires `shift 2` here and we pass on the input value `$2` to a previously initialized variable `logfile`. We should always define our variable defaults in the `###  LOCAL VARIABLE DEFAULTS ###` section, before we check our input parameters.

This is all that is required to extend the input parameter functionality of any SnappyVPS script.

To recap, in order to extend the existing input parameters:

1) Define your new variable defaults in the `###  LOCAL VARIABLE DEFAULTS ###` section.
2) Modify `OPTIONS` & `LONGOPTS` with your new variable identifier.
3) Add a new section to populate your new variable between the following tags:

```
        # ADDITIONAL START #
        #  ADDITIONAL END  #
```

4) Finally if required, validate your new variable after the tag:
   `###  VALIDATE FUNCTION ARGUMENTS ###`
   This is required in cases where your new variable is required, but the user never provided a value for it e.g.

```
   case "$domainName" in '') echo  -e "domain is required.\n";; esac
```

Main Program

The main program starts after the input parameters have been parsed. This is of course where the main functionality will take place. In general, almost every SnappyVPS script will, display its main title, check whether to run unattened or not and get down to business.

```
################
###   START  ###
################
```

Main functionality happens between the START and FINISH markers

```
################
###  FINISH  ###
################
```

After these some wrapping up takes place before exiting.

Sub-processes

To avoid littering the screen with hundreds of lines of useless output while the main payload is running, SnappyVPS makes use of sub-processes.

Instead of issueing commands like:

```
echo 'a message'
```

We should issue commands like this:

```
exe 'we will echo a message' echo 'a message'
```

The end result is that now the echo command runs under the control of the `exe subroutine`. As already mentioned in the previous sections we can configure the defaults for exe functionality in inc_globals.sh and we can even override these defaults regarding its output by passing parameters to our script.

e.g. compare the output of `ls /etc` to `exe 'list directory' ls /etc`

The second command outputs no listing. It only notifies us of the outcome:

```
|> Run: list directory [ Done! ]
```

As mentioned already, what comes out on screen and in log file is determined by these globals.

```
# subprocess configuration
EXEVERSCR="$_VERR"        # 2=all 1=err 0=no
EXEVERLOG="$_VALL"        # 2=all 1=err 0=no
```

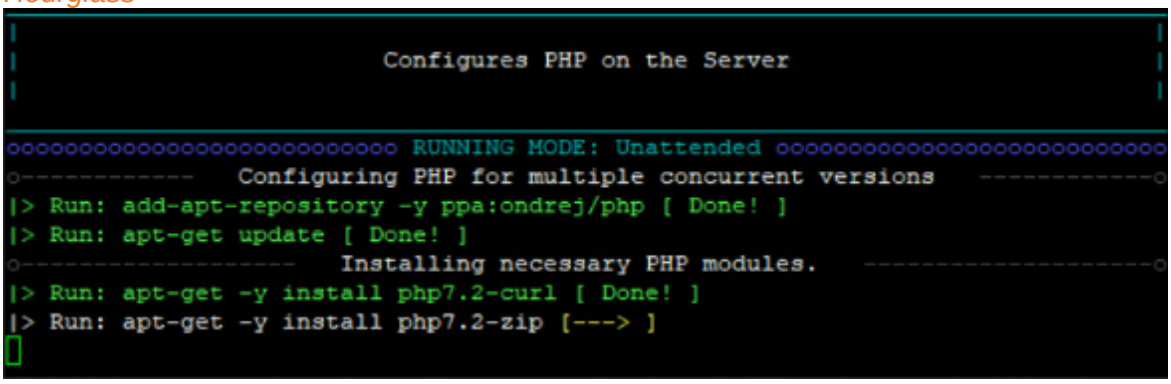
EXEONELINE

When `EXEVERSCR= $_VERR` the setting below snuffs out even the error messages which would otherwise display on the screen.

```
EXEONELINE=true           # true=errors not output to display
                          # (only if EXEVERSCR= $_VERR)
```

Instead it uses a color coding and `EXEMSGERR/EXEMSGOK` to indicate error or success. This is the default functionality. So we do not lose the actual error message, the default functionality allows for all STDOUT and STDERR to end up in the log file with `EXEVERLOG="$_VALL".`

Hourglass



e.g. `exe 'sleep a little' sleep 5` outputs `|> Run: sleep a little [  <--]`

While the main process waits for the sub-process to finish, a moving arrow is displayed, going back and forth between the square brackets. This provides useful feedback to the user indicating that the process is still running.

To configure the hourglass functionality, we can modify these global parameters.

```
EXESLPDUR=0.2              # hourglass refresh interval
EXEMODE=4                  # hourglass presets - values: 0 to 4
EXEMAXTE=30000             # max ms waiting for executed process
                           # to wrapup & exit before killing it
```

If a process returns with a result but is still running after `EXEMAXTE` seconds it will be forcefully terminated.

Return values

Sub-process execution return codes eventually end up back in the main process and can be accessed in the normal fashion using `$?`. In order for us to check if any subprocess has failed, all we have to do is keep taps on each result.

```
success=1    # initialize success to 1
exe 'task 1' sleep 1
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0
exe 'task 1' sleep 1
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0
echo success # check success
```

If the variable `success` turns to 0, we will know that something has gone wrong.

Ignoring Errors

Sometimes we may need to ignore some error codes. To do this we use a special marker `[i]` *followed by a comma separated list of error codes*, in a string specified before our command e.g.

```
exe 'task 1' '[i]15,23' mycommand
```

The above will run `mycommand` but if an error 15 or 23 is encountered, will still return 0 (success).

Executing eval

In some cases we may need to use eval.

e.g. When using a pipe or a redirection:

```
exe "$visudoSFTP" eval 'echo "$visudoSFTP" | (sudo EDITOR="tee -a" visudo)'

exe 'cat > /etc/rc.local' eval "cat > /etc/rc.local <<EOF
```

e.g. When we want to capture STDOUT into a variable:

```
exe 'Retrieving swapfile info' eval 'res="$(ls -lh /swapfile)"'
echo “$res”
```

Executing functions

There is nothing stopping us from running whole segments of code in a subprocess:

```
success=1    # initialize success to 1

modAnc() {
      local success=1    # initialize local success to 1

      exe ‘foo’ echo ‘foo’
      [ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0

      exe ‘bar’ echo ‘bar’
      [ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0

      [ "$success" -eq 1 ] && return 0 || return 1
}
exe 'running a sub' modAnc
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0
```

Wrapping Up

Checking for success

As already mentioned, sub-process execution return codes eventually end up back in the main process and can be accessed in the normal fashion using `$?`.

In order for us to check if any one subprocess has failed, all we have to do is initialize and keep taps on the `success` variable. To do this we must include this line after each sub-process line:

```
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0
```

Finally we check if success is `=1` or `=0` and we display the appropriate message e.g.

```
success=1    # initialize success to 1

################
###   START  ###
################

exe 'task 1' sleep 1
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0

exe 'task 2' sleep 1
[ "$?" -eq 0 ] && [ "$success" -eq 1 ] && success=1 || success=0

################
###  FINISH  ###
################
[ "$success" -eq 1 ] && msgTask="$msgSUCC" || msgTask="$msgFAIL"

vpssubTitleEnd "$success" "${msgTask}" | prt 2
```

anchorParameter.pl

Another feature of SnappyVPS is the perl module anchorParameter.pl.

It was created in order to make it easier to modify and update parameters within configuration files and is being used by a number of SnappyVPS scripts.

Check out the usage page: ***sudo perl ~/snappyVPS/inc/anchorParameter.pl -h***

Files

First the module needs to be told which file to modify. This is done like so:

```
-n "$filepath"
```

Anchors

The module uses the concept of anchors to decide where to place new information in a file. The placement will either be above (`-l 'above'`) or below (`-l 'below'`) this anchor. If no anchor is found then the whole file is considered the anchor and `-l <location>` will place the entry at the top or bottom of the file.

Finding an anchor

This is done via regular expression matching e.g.

```
-a "\W### CUSTOMIZATION END ###.*"
```

Matching is done in multiline mode, not single line, therefore care should be taken. We cannot use the `^` or `$` operators here because they will not correspond to the beginning and end of a line. In fact they may actually correspond to the beginning and end of the whole file, thus no match will be found.

This is why in the above example `\W` is used to match non-word characters like white space.

Another example is:

```
-a "(\W|\A)<\/IfModule>(\W|\Z)"
```

This uses either a non-word character or the beginning of the file as a starting point and a non-word character or the end of the file as the end point.

Update or Insert

If we need to just insert a value if it does not exist:

```
-u "insert"
```

If we need insert a value if it does not exist and also update if it exists:

```
-u "update"
```

Entries

The module will have to know whether our required entry exists or not in the target file. To do this, we have to explain what our entry looks like in regular expression format e.g.

```
-m "^.*myentry.*$"
```

Checking for entries is done line by line and therefore (unlike finding anchors) all regular operators can be used.

If the regular expression finds a match, it will update it (if `-u "update"`).
If no match is found, it will insert it (if `-u "insert"`).

What will be inserted is this:

```
-p "prefix" -e "parameter" -v "value" -s "suffix"
```

Care should be taken with `prefix` and `suffix` so that these are covered by your regular expression, next time the module is asked to find this entry.

example

```
filepath="/etc/apache2/mods-available/pagespeed.conf"
```

```
nc='(?!\s*#\s*)'

anchor="\W### CUSTOMIZATION END ###.*"
parameter='ModPagespeedFileCacheSizeKb'; value=' 2048000';

perl ~/snappyVPS/inc/anchorParameter.pl -u "u" -p "\t" -e "$parameter" -v
"$value" -s "" -m "^$nc\s*[[PM]]\b.*$" -a "$anchor" -l "a" -n "$filepath"
```

The above will look for an anchor and insert/update, above this anchor, the entry:

```
'^$nc\s*ModPagespeedFileCacheSizeKb\b.*$'
```

with

```
'ModPagespeedFileCacheSizeKb 2048000'.
```

The `^$nc` specified at the beginning tells our regular expression to avoid modifying an entry if it is preceded by a comment as defined here `nc='(?!\s*#\s*)'`. So if our entry is actually already commented in the file, a new entry will be created by the module. If the entry is found with a different value, the whole line will be replaced with our updated new value.

Special markers

Notice also the special characters `[[PM]]` being used in the previous example. This is for our convenience, instead of typing in the parameter again in our regular expression, we just include this special marker.

If we wanted to include both parameter and value in our regular expression we could use this special marker `[[PV]]`. In this case, our regex would evaluate to `'^$nc\s*ModPagespeedFileCacheSizeKb 2048000\b.*$'`.

File Sections

Sometimes we may need to modify parameters in parts of a file. This is because the configuration file is separated into sections and each section contains the same parameters as other sections. If we could not concentrate our attention on only one section, then all parameters globally would be replaced e.g. /etc/fail2ban/jail.local

```
[sshd]
enabled=true
port=ssh
bantime=600
maxretry=1

[ssh-ddos]
enabled=true
port=ssh,sftp
filter=sshd[mode=ddos]
bantime=2592000  ; 1 month
maxretry=0
logpath=/var/log/auth.log
```

The module can handle such situations by using an additional anchor. The first anchor regex finds the beginning of the section while the second anchor tries to find the end of the section. All other operations are handled as already described e.g.

```
anchor="(\W|\A)\[ssh-ddos\](\W|\Z)"; anchorend="\W\[|\Z"
parameter='enabled'; value='=true';
perl ~/snappyVPS/inc/anchorParameter.pl -u "u" -p "" -e "$parameter" -v
"$value" -s "" -m "^.*[[PM]]\s*=.*$" -a "$anchor" -l "a" -n "$filepath" -f
"$anchorend"
```

This will update the parameter `enabled` to `enabled=true` only in the `[ssh-ddos]` section of the file. Notice how the ending anchor is described here. It will try to match the beginning bracket of another section below this one, or at least the end of the whole file.

Double Evaluation

The module can also make use of double evalution, where it can remember what it managed to match and replace that with something else. This is especially useful when we want to comment lines out as they are.

```
parameter='swap'; value="";
parameter=$(perl -e 'print quotemeta($ARGV[0])' -- "$parameter")

perl ~/snappyVPS/inc/anchorParameter.pl -u "u" -p "" -e '"# ".$2' -v
"$value" -s "" -m "^$nc\s*/.*$parameter.*$" -a "$anchor" -l "b" -n
"$filepath" -d
```

The double evaluation flag is `-d`. This allows the use of `$2` and up, in our expressions of parameter and value.

`-e '"# ".$2'` essentially tells the module to take what was matched and stick a hash in front of it before placing it back over our match. `$1` should not be used as it is used internally.

It is important to note how the `-e` expression is enclosed in single quotes so as not be be interpreted by bash.

Also note that when -d is in effect we have to escape our input parameters before sending them to the module, because if they contain any special perl characters, the module wil try to interpret them. Therefore an additional step has to be included `parameter=$(perl -e 'print quotemeta($ARGV[0])' -- "$parameter")`.

Ok if Empty

Finally the `-o` switch, simply bypasses the check for non-empty parameters. This may be useful if you need to replace a match with a blank entry.

This is of course dangerous and it is instead recommended to use previous techniques to just comment out the line. However the option is easily enabled by just adding `-o` to your command.

SnappyVPS Directory Listing

```
Directory of \snappyVPS_vx.x.x
 <DIR>          readme
 <DIR>          snappyVPS
 <DIR>          snappyVPS-P4T
                initial_server_setup.sh
                license.txt

 Directory of \snappyVPS_vx.x.x\readme
                snappyVPS User Guide.pdf

 Directory of \snappyVPS_vx.x.x\snappyVPS
 <DIR>          firewall
 <DIR>          inc
 <DIR>          mod
 <DIR>          post
 <DIR>          startup
 <DIR>          vs
                install.sh
                license.txt
                migrate.sh
                migrate_mailboxes.txt
                setcronbackups.sh

 Directory of \snappyVPS_vx.x.x\snappyVPS\firewall
                blocklists.sh
                ip6tables.rules
                iptables.rules
                iptables.service
                iptables.servlet

 Directory of \snappyVPS_vx.x.x\snappyVPS\inc
                anchorParameter.pl
                inc_colors.sh
                inc_cron_helper.sh
                inc_functions.sh
                inc_globals.sh
                inc_titleBox.sh

 Directory of \snappyVPS_vx.x.x\snappyVPS\mod
                aggregate-cidr-addresses.pl

 Directory of \snappyVPS_vx.x.x\snappyVPS\post
                config_apache.sh
                config_dkim.sh
                config_fail2ban.sh
                config_finalize.sh
                config_iptables.sh
                config_mysql.sh
                config_php.sh
                config_quotas.sh
                config_scp_sudo.sh
                config_startup.sh
                config_swapfile.sh
                config_timezone.sh
                config_unattended_upgrades.sh
                config_webmin.sh
                install_image_tools.sh
                install_imapsync.sh
                install_pagespeed.sh
                install_puttytools.sh
                install_security_tools.sh
                install_webmin.sh
                update_system.sh

 Directory of \snappyVPS_vx.x.x\snappyVPS\startup
                rc.local

 Directory of \snappyVPS_vx.x.x\snappyVPS\vs
                make_virtual_server.sh
                migrate_mailboxes.sh
                migrate_virtual_server.sh

 Directory of \snappyVPS_vx.x.x\snappyVPS-P4T
                p4t.bat

     Total Files Listed: 46 File(s)
```

CHANGELOG

```
---------
# Changelog
All notable changes to this project will be documented in this section.

The format is based on [Keep a Changelog]
(https://keepachangelog.com/en/1.0.0/),
and this project adheres to [Semantic Versioning]
(https://semver.org/spec/v2.0.0.html).

## [Unreleased]

## [1.0.3] - 2019-10-17
### Added
- Changelog in the user guide

## [1.0.2] - 2019-08-22
### Changed
- Core functionality finalized after much testing
### Added
- Batch creation of Virtual Servers

## [1.0.1] - 2019-07-15
### Added
- Mailbox migration functionality
- IPtables servlet

## [1.0.0] - 2019-06-25
### Added
- Core functionality complete
```