**This is former documentation I was to submit for my internship at GreenTree IT, including a step-by-step guide on how to set up the backup process. Certain things have been omitted or censored for security purposes.**

**Account Creation**

* Created a Backblaze account with email and password. This will be used as a separate offsite cloud storage in case of catastrophic failure or human error.

**Bucket Creation**

* **This is for setting up three backup systems. The first will back up every two days. The second will back up every week. The third will back up about every two weeks (on the 1st and 15th of each month).**

To create a bucket:

1. Go to buckets
2. Create new bucket with button
3. Fill out information for bucket.

* Created a Bucket named vultrbackupsgreentreeIT. Set up for allprivate, has a life cycle of 4 days for previous versions before deleting/modifying them.

**RClone Installation**

Install rclone, which is built for management of files on cloud storage. Rclone mounts any local, cloud, or virtual filesystem as a disk on Linux, and serves these over SFTP, HTTP, WebDAV, FTP and DLNA. Rclone will help us backup (and if needed, encrypt) our files to Backblaze from VULTR. It’s an alternative to rsync, specifically for cloud storage.

* Ran “rclone –version” to verify that it had not been installed yet
* Ran “sudo -v ; curl https://rclone.org/install.sh | sudo bash” to install it
* Verified that it downloaded using “rclone –version”

**RClone Configuration**

* Set up remote named B2 for Backblaze B2.
* To set up the remote:
  + Create a new remote with n and label it to your wants.
  + A large list of cloud storages will be provided. Find Backblaze’s proposal number (this will change between updates probably, easier to find in a PuTTY console because you can scroll up).
  + When prompted, enter your account ID and Application Key ID. These can be made with creating a new application key, which is located in Application Keys on your Backblaze Account. When you add a new key, make sure to note down the keyID and applicationKey fields. The keyID will take the account ID field for rclone config, and the applicationKey will take the Application Key ID field.
  + Leave rest blank, confirm remote with y, and then exit config.
  + There is an advanced config if you want to change specific things such as chunk size for large file uploads.

**RClone Account IDs and Keys**

Account IDs:

* For original server: 0000000000000
* For test server: 0000000000000
* For restoration server: 0000000000000

Application Keys:

* For original server: 0000000000000
* For test server: 0000000000000
* For restoration: 0000000000000

**IMPORTANT: You will likely need to create additional keys whenever you want to connect to the Backblaze account with a different VPS. For the secondary vultr testing server, we had to create an additional key to ensure connection.**

**IMPORTANT: To reset your secondary vultr testing server to have a network (in order to sync), follow the documentation:** [**https://www.vultr.com/docs/how-to-find-the-network-adapter-names-for-a-vultr-cloud-server**](https://www.vultr.com/docs/how-to-find-the-network-adapter-names-for-a-vultr-cloud-server)

[**https://www.vultr.com/docs/configure-a-vultr-server-to-use-dhcp-for-dynamic-ip/#Ubuntu**](https://www.vultr.com/docs/configure-a-vultr-server-to-use-dhcp-for-dynamic-ip/#Ubuntu)

**Final Backup:**

**(make sure to mark the server’s uuid)**

**Uploading backup to backblaze:**

To upload backup to backblaze merely go to the server you want to create a backup of, go to root (*cd /* ) and then run this command to create a compressed tar of the server and then sync it (B2:vultrbackupgreentree is name of remotename:bucketname).

There should be an automatic backup script in /usr/local/bin.

*tar -cvpzf backup.tar.gz --exclude=/backup.tar.gz --one-file-system /*

*Rclone sync backup.tar.gz B2:vultrbackupgreentree*

**Restoring server from backup:**

1. For restoration, you are going to have a fresh installation of Ubuntu 22.04 (New server via VULTR)
2. Move to / and create a folder called backup.
3. Install rclone using the rclone install command (mentioned above) and set up the rclone backblaze config
4. Use rclone -q --b2-versions ls remotename:bucketname to find all versions of the backup files.
5. If you want to retrieve the current version, just use rclone copy remotename:bucketname/backup.tar.gz backup
6. If you want to retrieve older versions, just use rclone -q --b2-versions copy remotename:bucketname/backup.tar.gz(date-that-is-listed next-to) backup
7. Move the backup.tar.gz out of the backup folder and into /.
8. Untar with *tar -xvf backup.tar.gz*
9. After untarring, reboot with SystemRescue ISO.
10. Boot up the SystemRescue (option at the top)
11. *e2fsck -f /dev/vda1* (say yes to all)
12. *tune2fs /dev/vda1 -U* "OLD SERVER'S /DEV/VDA1'S UUID" (yes)
13. Remove SystemRescue ISO and restart.
14. From here, just simply restore containers with docker start \_\_\_. If a port is busy and won’t let a container start (such as suitecrm), use netstat –tulpn to find the port that’s busy and the process listening in on it (will have a 3-4 number and then a label). Use kill PID to stop the current process from listening, and then start the container.

Backup command for every two days:

0 2 \*/2 \* \* /usr/local/bin/server-backup/backup-server.sh (every 2 days at 2 am according to the hwclock on the server)

Backup command for every week:

0 2 \* \* 0 //usr/local/bin/server-backup-grab-restore/backup-server.sh

Backup sync command for every two weeks:

0 2 1,15 \* \* //usr/local/bin/server-backup-grab-restore/backup-serverls.sh

**Bug Log of recorded progress:**

**OLD BACKUP(deprecated, won’t work upon restart, follow final backup):**

**Uploading backup to backblaze:**

Use this command to tar the server into one compressed file and then rclone it to the backblaze bucket (look at the above instructions for making sure rclone is installed and configurated):  
*cd / tar -cvpzf backup.tar.gz --exclude=/backup.tar.gz --exclude=/sys --exclude=/dev --exclude=/proc / rclone sync/copy backup.tar.gz B2:vultrgreentreebackup*

The B2 should be the name of the rclone remote, while vultrgreenbackup should be the name of the backblaze bucket.

**Transferring backup from backblaze:**

Run the following commands to transfer the backup.tar.gz inside the backblaze bucket to the restoration server (on a newly created server, the rclone installation and rclone configuration need to be set up):

rclone copy B2:vultrgreentreebackup/backup.tar.gz backup (this’ll create a folder named backup)

cd backup

mv backup.tar.gz /

cd

rmdir backup.tar.gz

cd /

tar –xvf backup.tar.gz

**Restoring server:**

To restore mysql run:

mysqld –uroot

After running that command, you may need to restart your PuTTY console, also note that the password for the server will change to the password of the original server. May need to reset the password by using passwd.

to restore containers and data:

systemctl start docker

docker start mariadb

docker start suitecrm (if run into bind error, do netstat -tulpn, find the port thats being taken up, kill the PID and then start suitecrm (netstat -tulpn lists all the ports that are being listened on, PID is the 3-4 digit number on the far right when you do netstat -tulpn)

docker start mysql-redmine

docker start redmine

docker start portainer

**Confirm Restoration:**

To confirm that docker and the databases are restored you can follow the following instructions:

To confirm that mysql works run:

mysql

Show Databases; (One of them should be suitecrm)

Use suitecrm;

SELECT \* from users; (should show usernames and emails)

To verify docker restoration, you will use a few commands to verify that mariadb and suitecrm work (can use docker ps –a to view containers):

docker inspect container (to confirm mountpoints)

docker exec -it containername bash (opens up a bash shell inside of the container)

ls and cd around (search for bitnami in both mariadb and suitecrm, if either have a hello.log, they are correct)

**Time to backup and Backup scripts:**

The backup process should take around 45 minutes to backup and around 10 minutes to restore.

There will be scripts that will automatically backup the server once a certain amount of time passes. They are located in the test server at /usr/local/bin and can view them with vi scriptname (with scriptname being the name of the script as either restore-server.sh which is a script for restoration and backup-server.sh which is a script for backup).

There should be a README.log with the following information:

The backup script tars up the entire filesystem excluding the virtual filesystem which would corrupt or be useless for the backup.

It then syncs the tar file up along with each script (backup, grab, and restore)

The restore script assumes the tar file and itself are already in a folder called backup. It'll move the tar to the / folder, and then untar it from there. It'll also attempt to restart configurations on the server.

The configurations are MySQL and Docker. It'll also ask if you want to do a password reset, as it should retain the previous server's password.

MAKE SURE TO CREATE A FOLDER IN / CALLED backup BEFORE RUNNING RESTORE SCRIPT

The restore script constructs all the commands listed above in server restoration as well as echoing helpful strings that gives the user an idea of what it is doing. The backup script is similar in it also has commands with useful echos (print text) like “Backup Initiated, going to root” and “Syncing backup/restore script file and README over”.

For the backup script, on top of the commands that backups the server to backblaze, it also runs:

rm backup.tar.gz (removes the backup file from the original server as it’s already in backblaze)

The following commands sync the backup/restore scripts and the README:

rclone copy --update /usr/local/bin/server-backup-grab-restore/backup-server.sh B2:vultrgreentreebackup

rclone copy --update /usr/local/bin/server-backup-grab-restore/restore-server.sh B2:vultrgreentreebackup

rclone copy --update /usr/local/bin/server-backup-grab-restore/README.log B2:vultrgreentreebackup

The following are Script commands for a predefined amount of time

Backup command for every two days:

0 2 \*/2 \* \* //usr/local/bin/server-backup-grab-restore/backup-server.sh (every 2 days at 2 am)

Backup command for every week:

0 2 \* \* 0 //usr/local/bin/server-backup-grab-restore/backup-server.sh

Backup sync command for every two weeks:

0 2 1,15 \* \* //usr/local/bin/server-backup-grab-restore/backup-serverls.sh

(twice per month at 12 am on the 1st and 15th of the month, running this command)

**Links:**

https://help.ubuntu.com/community/BackupYourSystem/TAR - what to backup and exclude (the current command was achieved via trial and error)

https://serverfault.com/questions/74696/linux-what-directories-should-i-exclude-when-backing-up-a-server - for what to exclude (the current command was achieved via trial and error)

https://docs.docker.com/ - for docker commands

https://crontab.guru/ - for cron setup

https://www.geeksforgeeks.org/how-to-create-a-shell-script-in-linux/ - for initial script creation

https://tecadmin.net/prompt-user-input-in-linux-shell-script/ - for prompt in shell script

https://www.digitalocean.com/community/tutorials/if-else-in-shell-scripts - for if statement in shell script

https://www.vultr.com/docs/configure-a-vultr-server-to-use-dhcp-for-dynamic-ip/ - for ip configuration if needed

https://www.vultr.com/docs/boot-into-single-user-mode-reset-root-password/#:~:text=%2Fbin%2Fsh%20.-,Press%20CTRL%20%2B%20X%20or%20F10%20to%20boot%20into%20single%2Duser,to%20change%20the%20root%20password - for resetting root password outside of putty console if needed

tar -cvpzf backup.tar.gz --exclude=backup.tar.gz --exclude=sys --exclude=dev --exclude=proc --exclude=tmp --exclude=mnt --exclude=dev --exclude=run --exclude=media --exclude=var/log --exclude=var/lib/docker .

**On top of the --excludes you already have listed in the documentation is /lost+found/, /var/tmp, and /var/cache, /var/run, /usr/tmp and /var/lock, exclude=/proc --exclude="/tmp/" --exclude="/mnt/" --exclude="/dev/" --exclude="/sys/" --exclude="/run/" --exclude="/media/" --exclude="/var/log/" --exclude="var/cache/apt/archives/" --exclude="usr/src/linux-headers-\*\*" --exclude="home/\*/.gvfs" --exclude="home/\*/.cache" --exclude="home/\*/.local/share/Trash" --exclude=”var/lib/docker/”,**

**Dry run of VULTR Backblaze Sync (a lot faster than an actual run + no cap rate)**

Command used:

rclone sync -v --dry-run --transfers 32 --fast-list / VultrGreenBack:vultrbackupsgreentreeIT --exclude=/dev/ --exclude=/proc/ --exclude=/sys/ --exclude=/tmp/ --exclude=/run/ --exclude=/mnt/ --exclude=/media/ --exclude=/lost+found/

VultrGreenBack can be replaced with any remote name and vultrbackupsgreentreeIT can be replaced with any bucket name. For reference: remoteName:bucketName.

Removes a lot of excess that isn’t needed or wanted for server recovery (these are temp files or files that are configured during setup, so can be ignored).

Dry run ensures that no files are transferred (for testing purposes). Also makes a sync appear faster (around ~2m, but it will be much slower with actual sync). Around 32GB will be synced to the bucket.

Transfers are bumped up to 32 for faster transfer speed.

**Crontab Setup**

Created a new server to run Crontab tests which restored from the backup on the original server. For some reason the password for the newly created Vultr server is the password for the old server. Putty and ssh also do not work, so use the server console.

Tried using crontab –e to create a dry run command, which did not work as the command inside the crontab did not excucute. Instead, go into /etc/crontab with preferred text editor, and insert the rclone sync commands below.

The commands will execute, but if you want to test before entering any of the commands, use \* \* \* \* \* echo “Hello World!” and track it using tail –f /var/log/syslog (which will show a popup of it executing every minute). To exit the tail, use Ctrl + C.

Rclone sync command for every two days:

0 0 /2 \* \* rsync … (every 2 days at 12 am)

Rclone sync command for every week:

0 0 \* \* 0 rsync ...

Rclone sync command for every two weeks:

0 0 1,15 \* \* rsync … (twice per month at 12 am on the 1st and 15th of the month, running this command)

0 0 /14 \* \* (every 14 days)

**Actual run of VULTR Backblaze Sync**

Planned Test Run Command(temp):

rclone sync -v --transfers 32 --fast-list / VultrGreenBack:vultrbackupsgreentreeIT --exclude=/dev/ --exclude=/proc/ --exclude=/sys/ --exclude=/tmp/ --exclude=/run/ --exclude=/mnt/ --exclude=/media/ --exclude=/lost+found/

The first backup will be approximately ~32GB with a long transfer time (maybe 10+ minutes), but iterations of that backup will be much faster from then on as it will only replace files that have been changed (incremental).

Issues: Ran the command but due to storage caps, it only transferred 45%, and took 8 and a half minutes. Rclone sync will need increased cap rates from the initial rate on Backblaze’s side to ensure that every file gets synced.

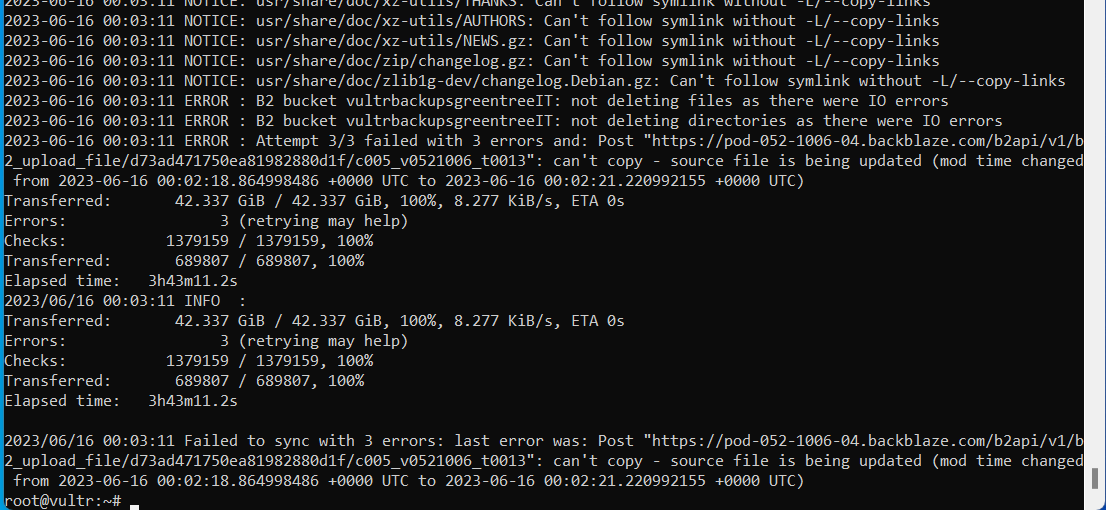
After transferring to Blackbaze, will transfer back to an empty vultr server to make sure the restoration can be properly completed, the command is (for dry run first):  
rclone sync –v –dry-run --transfers 32 --fast-list VultrGreenBack:vultrbackupsgreentreeIT / --exclude=/dev/ --exclude=/proc/ --exclude=/sys/ --exclude=/tmp/ --exclude=/run/ --exclude=/mnt/ --exclude=/media/ --exclude=/lost+found/

For the real run, simply remove –dry-run

Doing another backup to blackbaze but with more excludes:

rclone sync -v -P --dry-run --transfers 32 --fast-list / VultrGreenBack:vultrbackupsgreentreeIT --exclude=/dev/ --exclude=/proc/ --exclude=/sys/ --exclude=/tmp/ --exclude=/run/ --exclude=/mnt/ --exclude=/media/ --exclude=/lost+found/ --exclude=/var/log/ --exclude=/var/cache/apt/archives/ --exclude=/usr/src/linux-headers\*/ --exclude=/home/\*/.gvfs/ --exclude=/home/\*/.cache/ --exclude=/home/\*/.local/share/Trash/

Note: maybe add --one-file-system

Approximately 4 hours of transfer time (backblaze to vultr), uploaded 42.337GB (No excludes):  


Managed to get into the server around 11 (gave me a warning about how the cached key didn’t work), anyways, the password didn’t work

**Docker:**

The Backups Failed, as during restoration, the docker commands didn’t work; To process docker, so docker is going to be backed up separately, and then the rclone sync command will have an exclude for docker.

The tenative plan is to:

1. Create a clone of the real server, deploying a new server, going to backups and choosing the most recent one, restoring it.
2. Run the backup command for suitecrm and mariadb data volume.
3. Backup both of these files to the backblaze bucket.
4. Copy those files to a third server for restoration (must reinstall docker for this as well)

Command to run to create a backup.tar.gz file for the suitecrm data volume:

docker run -v /suitecrm\_data --name suitecrmstore ubuntu bin/bash

docker run --rm --volumes-from suitecrmstore -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /suitecrm\_data

Replace suitecrm\_data and suitecrmstore with mariadb-data and mariadbstore, along with renaming the backup.tar to mariabackup.tar

You can put these in a folder called backups:

mkdir backups

mv –t backups backup.tar mariabackup.tar

Then sync it to the backblaze bucket with: rclone sync backups remotename:bucketname, syncing both files over to the backblaze bucket.

Now, sync it back to the third server, which now also needs rclone installed and configurated with Backblaze, along with Docker also.

Copy it to the test backup server after mkdir backup and configuring rclone config.

Rclone copy remotename:bucketname backup

Attempt restoration with the following documentation: <https://hub.docker.com/r/bitnami/suitecrm/>

**Actual Backup Plan now:**

1. **Docker commit all important containers, then docker save all the images created from commits.**
2. **Backup all docker volumes that are important to those containers (including bind mounts). Make a simple text file that explains these mountpoints to each container.**
3. **Put both 1 and 2 into a syncable folder outside of 1 and 2.**
4. **Run rclone sync command that excludes 1 and 2 (probably by excluding /var/lib/docker, exclude bind mounts as well), and the rest of the exclusions from the previous sync command in the documentation.**
5. **Sync those files to the restoration site, restarting each docker container and mounting them according to the text file.**
6. **Check if restoration was a success by using mysql inside of the mariadb container (only thing I have been able to get conclusive proof on) to check if the users table in the suitecrm database is populated. If so, should be able to assume that the other volumes/bind mounts were backed up correctly.**
7. **Ask Isaac to test any common commands on the test site to ensure it is functioning properly.**

Implementation

Used docker ps to list all the containers and used docker commit and docker save to save them all in backup\_images.tar

Use docker inspect on the container ids, and then list them (binds and mounts) in mounts.txt

Ran the docker volume backup commands on containers that had volumes with  
docker cp \*container\_id:\*path

Attempting rclone sync to bucket with:

rclone sync -v -P --transfers 32 --fast-list / VultrGreenBack:vultrbackupsgreentreeIT --exclude=/dev/ --exclude=/proc/ --exclude=/sys/ --exclude=/tmp/ --exclude=/snap/ --exclude=/mnt/ --exclude=/media/ --exclude=/lost+found/ --exclude=/var/log/ --exclude=/var/cache/apt/archives/ --exclude=/usr/src/linux-headers\*/ --exclude=/home/\*/.gvfs/ --exclude=/home/\*/.cache/ --exclude=/home/\*/.local/share/Trash/

Attempting restoration:

Tried some commands (failed or did nothing ) and :  
docker run --rm --volumes-from suitecrm -v $PWD:/backup\_folder bash -c "cd /backup\_folder && tar xvf /backup\_folder/suitecrmbackup.tar"  
seemed to work  
  
gave me suitecrm\_data, the command:

tar tf /suitecrmbackup.tar

Returned suitecrm\_data/

I used docker ps –a to find the container

And also used docker start on the container  
and when I used docker ps again, it showed that it was running, but I don’t actually know how to verify that the restoration worked, and I don’t understand the suitecrm\_data/ file.

**What I did 6/30/2023:**

On test server.

docker run -d --name mariadbstore -v mariadb-data:/var/lib/docker/volumes/mariadb-data/ ubuntu

docker run --rm --volumes-from mariadbstore -v $PWD:/backup-dir ubuntu tar cvf /backup-dir/mariadb-backup.tar /var/lib/docker/volumes/mariadb-data/\_data/data

rclone copy mariadb-backup.tar B2:vultrgreentreeITbackups

Move to restore server.

rclone copyto B2:vultrgreentreeITbackups/mariadb-backup.tar mariadb-backup.tar

docker run -d --name mariadbstore -v mariadb-data:/var/lib/docker/volumes/mariadb-data/ ubuntu

docker run --rm --volumes-from mariadbstore -v $PWD:/backup-dir

bash -c "cd /var/lib/docker/volumes/mariadb-data/\_data/data && tar xvf /backup-dir/mariadb-backup.tar"

docker rm mariadbstore

Where I ran into problems:

docker run -d --name mariadb --env ALLOW\_EMPTY\_PASSWORD=yes --volume /var/lib/docker/mariadb-data/\_data/data:/bitnami/mariadb bitnami/mariadb:latest

What I did:

Found out that suitecrm-backup.tar was a directory and not a tarball somehow? So I’m transferring the suitecrmstore to the test restoration from the test server

docker run --rm --volumes-from suitecrmstore -v $(pwd):/backup ubuntu tar cvf /backup/suitecrmstore\_backup.tar /var/lib/docker/volumes/suitecrm\_data

So I ran the above command to turn it into a tar

Then transferred it over to the test restoration server.

I tried this command with the container id, it did something. There was a list of

docker run --rm --volumes-from 365d0772fb0b -v $(pwd):/backup ubuntu bash -c "tar xvf /backup/suitecrmstore\_backup.tar --strip 1"

But when I checked docker ps there was no container running. Removing the rm flag dosen’t do anything.

Moved the files from the backups into the actual volumes and mounted them. Successful database recovery for the mariadb container, however, suitecrm shows minor differences due to container filesystem, may want to export containers, import them in, and tag them accordingly.

7/5:

Redmine: step 1 create tar file

Redmine: name, /backup: just a name, /redmine\_backup.tar just a name, /usr/src/redmine/files: path

docker run --rm --volumes-from redmine -v $PWD:/backup ubuntu tar cvf /backup/redmine\_backup.tar /usr/src/redmine/files

Transfer to bucket as intermediary

Rclone copy redmine\_backup.tar B2:vultrgreentreeITbackups

rclone copy B2:vultrgreentreeITbackups/redmine\_backup.tar redmine\_backup.tar

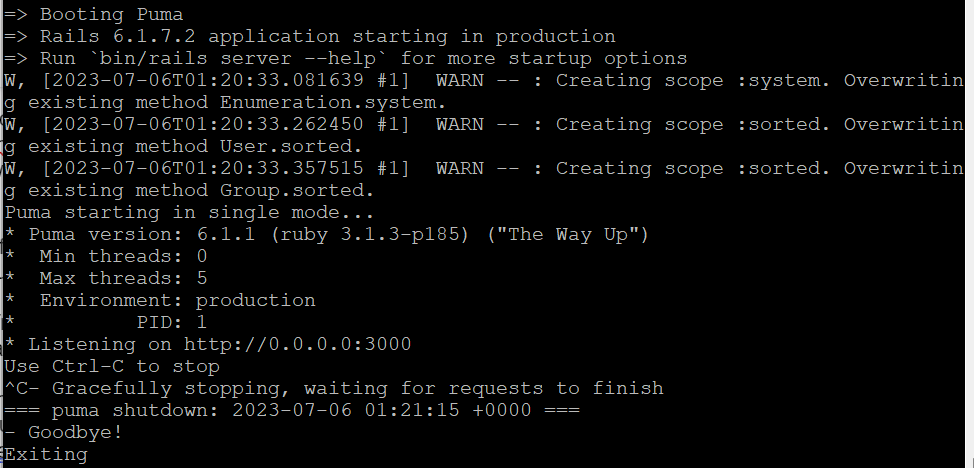
I then used

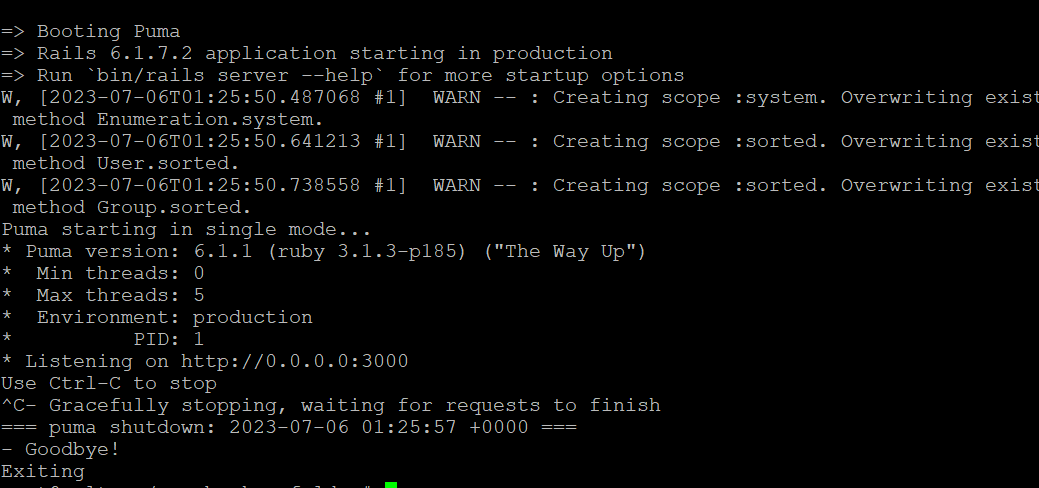
Docker volume create redmine-data

and

docker run -d --name redmine -v redmine-data:/usr/src/redmine/files ubuntu

And then ran

docker run redmine, and this showed up (so I think it worked, although I'm not sure why):   




This is what happened on the test server, so I think it was sucesfully backuped and restored

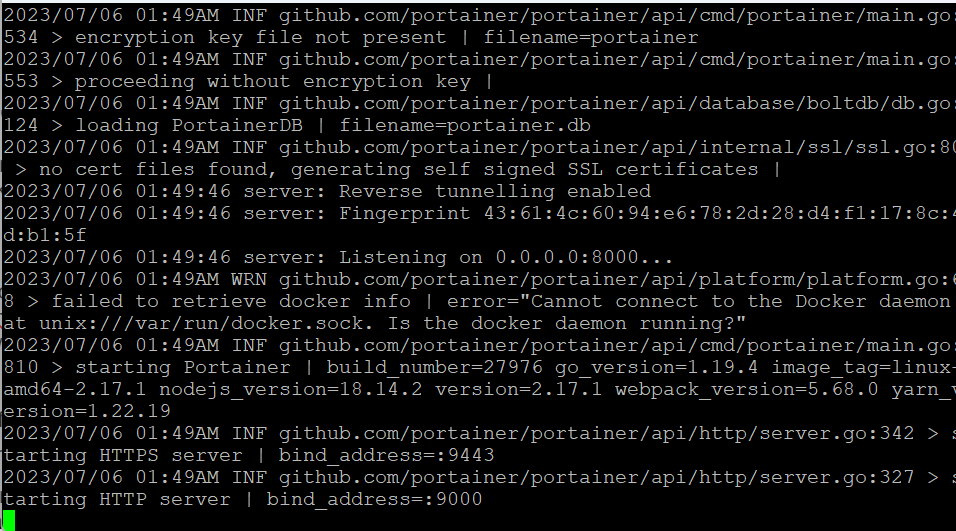
Portainer:

docker run --rm --volumes-from portainer -v $PWD:/backup ubuntu tar cvf /backup/portainer\_backup.tar /data

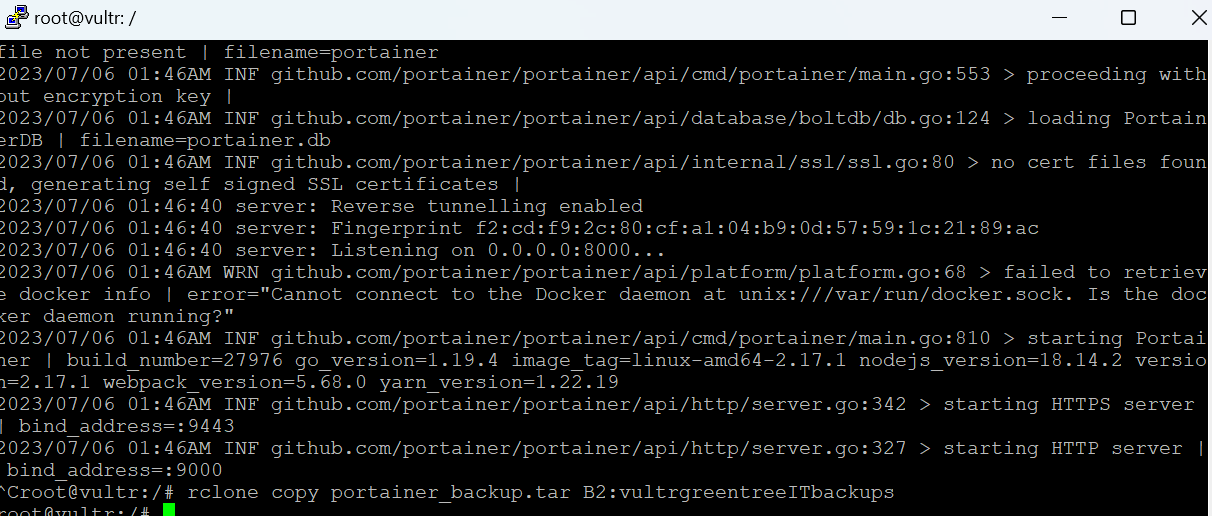
(/var/run/docker.sock gave error: socket ignored, so I used /data)

Did similar commands for the portainer, and yes I verified for the restore the portainer\_backup.tar file is uneeded, so um... that’s weird (it’s possible the rclone copy commands are unessecary so perhaphs the role of the backblaze bucket is unecessary)

Restore\_server:



Test\_server: (gave same results)



**RCLONE WITH DOCKER:**

rclone sync -v -P --dry-run --transfers 32 --fast-list / B2:vultrbackupsgreentreeIT --exclude="/proc/" --exclude="/tmp/" --exclude="/mnt/" --exclude="/dev/" --exclude="/sys/" --exclude="/run/" --exclude="/media/" --exclude="/var/log/" --exclude="var/cache/apt/archives/" --exclude="usr/src/linux-headers-\*\*" --exclude="home/\*/.gvfs" --exclude="home/\*/.cache" --exclude="home/\*/.local/share/Trash"

**RCLONE WITHOUT DOCKER:**

rclone sync -v -P --dry-run --transfers 32 --fast-list / B2:vultrbackupsgreentreeIT --exclude="/proc/" --exclude="/tmp/" --exclude="/mnt/" --exclude="/dev/" --exclude="/sys/" --exclude="/run/" --exclude="/media/" --exclude="/var/log/" --exclude="var/cache/apt/archives/" --exclude="usr/src/linux-headers-\*\*" --exclude="home/\*/.gvfs" --exclude="home/\*/.cache" --exclude="home/\*/.local/share/Trash" --exclude=”var/lib/docker/”

**Rsync possibilities:**

1. Compress the entire directory into a single file then rsync it to backblaze
2. Mount a block storage, backup the data to the block storage then rsync the data away

Can use Zerofree to reduce the amount of data

**Pricing:**

Transferring 23.75 by rclone sycning the backup and the vultr server cost around $0.45, the estimated storage costs are $0.40 each month for each backup. TSe total costs assuming 2 backups per day (15 a month), 1 backup a week (rounding to 4 a month) and 2 backups a week, (rounding to 2 a month) would total out to 0.3 \*0.4 for the storage costs or around $1.20. The transfer costs would be (15+4+2)\*0.45 would be $9.45 for transfer costs. To get total costs, add those two costs to get $10.65.

Total costs would be $10.65 per month

Note: Assuming that there is already a backup, so that only files that were changed need to be synced, a full sync would be more like 42.337 and not 23.75 GB, so a first-time backup would be slightly higher than updating a backup.

**ALL LINKS:**

<https://stackoverflow.com/questions/46109358/how-to-create-a-cron-expression-for-every-2-weeks>

<https://www.skysilk.com/blog/2021/migrate-existing-vps/>

<https://blog.ssdnodes.com/blog/vps-backups-simple-overthinking/>

<https://lowendbox.com/blog/how-to-migrate-a-hosted-server-in-5-easy-steps-with-rsync/>

[General Cron Dates](https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm)

<https://rclone.org/b2/>

<https://rclone.org/commands/rclone_sync/>

<https://hub.backblaze.com/backblaze-and-vultr>

<https://www.backblaze.com/b2/docs/buckets.html>

<https://github.com/christianlempa/cheat-sheets>

<https://stackoverflow.com/questions/26331651/how-can-i-backup-a-docker-container-with-its-data-volumes#:~:text=To%20back%20up%20docker%20images,with%20the%20docker%20load%20command>.

<https://docs.docker.com/storage/volumes/> For understanding volumes.

<https://docs.docker.com/storage/bind-mounts/> For understanding bind-mounts.

<https://linuxhint.com/tar-folder-linux/> For tarring folders.

[Docker Volumes with Persistent Data in Containers | Beginners Tutorial](https://www.youtube.com/watch?v=OrQLrqQm4M0)

[Migrate Docker Volumes from one Host to another // backup and restore](https://youtu.be/ZEy8iFbgbPA)

<https://help.it.ox.ac.uk/files-and-folders-excluded-from-backup#collapse2411936>

<https://unix.stackexchange.com/questions/1067/what-directories-do-i-need-to-back-up> - what to backup