Backup Software to Save your Data

- URBackup
  - Installing and Configuring URBackup
- Duplicati
  - Installing Duplicati to Backup your Data
- Minarca Backup
  - Install Minarca Backup

URBackup

Installing and Configuring URBackup

https://www.youtube.com/embed/MzDmOMVUxLs

Backups are important for so many reasons. I have had backups save my bacon more than once, and I have also learned the hard way how important backups really are. Data loss is no joke, and in this day and age there is truly no reason to have data loss.

Today, I want to cover a really great backup tool that is open source, and not too difficult to get setup. URBackup (like You Are Back Up! - get it?) is a really terrific tool boasting both a Server and Client installation that then makes your backup server (where you store your data) talk to your client machines (where you are backing up your data from) with some nice auto-discovery features built in.

Now, learn from me. Make sure your firewall has port 55414 and 55415 open so that URBackup's client and server can talk easily. You'll save yourself a lot of time and headache.

What you'll need

- A server with enough drive space to store backups.
- Docker and Docker Compose installed on that server - check out https://github.com/bmcgonag/docker_installs for some easy to use scripts for various versions of Ubuntu, Debian, and CentOS.
- At least one client machine running either Windows, Linux, or both if you like.
- About an hour of time.

Installation

First, if you haven't already done so, please make sure you have both Docker-CE and Docker-Compose installed on your intended server system. This is important as it's how we will install the server portion of URBackup.

Installing Docker-CE and Docker-Compose

Installation via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of `sudo` in front of them.

Installing URBackup Server

Now, we'll grab the docker-compose text from the URoni/URBackup dockerhub page, and setup our space for backups.

1. Setup a space to store your backups.

```
mkdir -p urbackup/{data,storage}
```

This will setup a folder structure with a new folder called "urbackup", and two folders inside of that folder called "data" and "storage".

2. Move into the "urbackup" folder:

```
cd urbackup
```

3. Create a new docker-compose.yml file:

`nano docker-compose.yml`

4. Paste the following code into that file:

```
version: '2'

services:
  urbackup:
    image: uroni/urbackup-server:latest
    container_name: urbackup
    restart: unless-stopped
    environment:
      - PUID=1000 # Enter the UID of the user who should own the files here
      - PGID=1000  # Enter the GID of the user who should own the files here
      - TZ=America/Chicago # Enter your timezone
    volumes:
      - ./data:/var/urbackup
      - ./storage:/backups
      # Uncomment the next line if you want to bind-mount the www-folder
      #- /path/to/wwwfolder:/usr/share/urbackup
    network_mode: "host"
    # Activate the following two lines for BTRFS support
    cap_add:
      - SYS_ADMIN
```

Save the file contents with CTRL + O, then press Enter, and exit the nano editor with CTRL + X.

5. Now run the URBackup docker-compose with:

`docker-compose up -d`

Once it downloads URBackup, and says "done", give it about 30 seconds, then open a web browser, and go to the ip address of your server on port 55414.

For instance, my server ip is 192.168.7.51, so I went to

`http://192.168.7.51:55414`

If all went well, you should now see the URBackup Server Web UI in front of you.

Install a URBackup Client

If you are using Windows, you'll download the client application .exe file from the URBackup site (https://urbackup.com). Next install the client normally, and when done, you'll use the Graphical User Interface to decide whether you want to backup only certain folders, or the entire system, etc.

You can go back to the URBackup Server, and refresh the page to see if the Windows machine has been autodiscovered. If not, you might waith a couple of minutes and try again. If it still doesn't show up, then make sure you don't have a firewall on that's blocking ports 55414 and 55415. These are the ports that URBackup uses to communicate between server and client.

Linux Client

If you are running Linux, you'll want to grab the command line command from the URBackup site, and paste it into the terminal.

At the time of writing the command is:

```
TF=$(mktemp) && wget "https://hndl.urbackup.org/Client/2.4.11/UrBackup Client Linux 2.4.11.sh" -O $TF && sudo sh $TF; rm -f $TF
```

But, it's always best to go to the official source, and make sure the command is up to date.

Once pasted into the terminal, run it by pressing Enter, and enter your sudo password when prompted.

You may also have to enter 'Y' at some point, and choose how you want backups / snapshots to be done.

After everything runs, check the Web Interface, and make sure your machine shows up.

Now, you can enter a command to tell URBackup what to backup.

I chose only my Downloads folder, but give it whatever path you want.

```
sudo urbackupclientctl add-backupdir -d /home/brian/Downloads/
```

Run this as many times as you like to add multiple folders for backup.

Support My Efforts on Patreon

https://www.patreon.com/bePatron?u=234177

Duplicati

Duplicati

Installing Duplicati to Backup your Data

https://www.youtube.com/embed/N1NRvg4KaDE

Duplicati is a really amazing backup solution for your local machine. You can use Duplicati to backup locally, to storage on your own network, or to cloud hosted storage. The front end runs in a browser window, so it essentially just runs in the background. You setup your backup, schedule it, and forget about it. Just let it do it's job!

Today we'll be installing Duplicati in Docker. I'm using it to backup my Docker folder, but it can be mapped to the root level folder of your system, or to your home folder, or really in any way you see fit. Then you can use the browser interface to setup the backup(s) you want.

What You'll Need

- Docker-CE
- Docker-Compose
- A location to Backup to
- About 20 Minutes

Install Docker-CE and Docker-Compose

Installation via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

`./install_docker_nproxyman.sh`

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Installation and Setup

Once you have Docker and Docker Compose installed, you'll want to create a folder inside your main organizational folder. I use a folder called "docker" to keep all of my docker application folders, volume mappings, docker-compose.yml and docker-run.txt files in. This way, I just backup the "docker" folder, and all of my docker compose, docker run commands, volume data, and various application configs are backed up with it. I highly recommend you do something similar.

So inside the "docker" folder, we'll create a new directory called "duplicati":

`mkdir duplicati`

Now we move into that directory:

`cd duplicati`

and we'll create a new file called "docker-compose.yml" inside this directory.

`nano docker-compose.yml`

You'll want to paste the following block of yaml text into that new file, and then we'll go through the spots where you'll need to make adjustments for your server.

```
version: "2.1"
services:
  duplicati:
    image: lscr.io/linuxserver/duplicati
    container_name: duplicati
    environment:
      - PUID=0
```

```
      - PGID=0
      - TZ=<your/timezone>
    volumes:
      - /mnt/filesync/duplicati/config:/config
      - /mnt/filesync/duplicati/backups:/backups
      - /home/brian/:/source
    ports:
      - 8270:8200
    restart: unless-stopped
```

In the file above, I have several comments. You'll want to be sure and adjust the following items for your system:

- PGID - might be good to give this the root user's ID, 0, so there are no permissions issues during backup.
- PUID - might be good to give this the root group ID, 0, so there are no permissions issues during backup.
- TZ (time zone) - You'll want to make this your Time Zone. Mine is America/Chicago
- /mnt/filesync/duplicati/config:/config - use this mapping to map or create, then map a location for your backup config. Make sure this location exists before making the mapping. Only change the left side of the colon ":".
- /mnt/filesync/duplicati/backups:/backups - use this mapping to map or create, then map a location for your backup storage. Only change the left side of the colon ":".
- /<your root level folder to choose backup locations from>:/source - this is the mapping of the root level. Only change the left side of the colon ":".
- 8270:8200 - feel free to change the left side of the colon ":" to any open port on your host machine.

Once you've made all of the necessary changes, you'll want to save the file with CTRL + O, then Enter to confirm. Use CTRL + X to exit the nano editor.

Now run the command:

`docker-compose up -d`

To pull down the Duplicati image, and start it running.

Once it shows "done" in the terminal for each piece, you'll want to go to your web browser and enter the IP of your host machine, and the port you set on the left side of the colon in the port mapping.

I went to

`http://192.168.10.26:8270`

Your IP will likely be different.

Once you get to the Duplicati web interface, you can start setting up your backups, and adjusting settings available through the Web User Interface.

Make sure to check out the video for more details on using the Web UI to setup ans schedule your backups.

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177

Minarca Backup

Client to Server Backup software

Install Minarca Backup

https://www.youtube.com/embed/JuM5AbrJgZQ

We'll start with a new Incus container (LXC / LXD). I am starting with an image of a Ubuntu 22.04 (24.04 doesn't seem to be supported by Minarca just yet). If you are using a Container like Incus / LXD / LXC, then you need to set it up with Nesting = true (Nesting = 1 in Proxmox). For Docker, you need to also run it as privileged, though that's not what we'll cover here.

> NOTE: We set all of this up as a root user. It's not specifically that Minarca needs root access, it's just that this install creates a Minarca user, and sets up the default backups folder at the root path of `/backups`.

We'll learn how to change that root backup location later in the tutorial.

Prepare Our Server

Let's get into super user mode, just to make things easeir:

```
sudo su -
```

We'll first update and upgrade the packages on the server.

```
apt update && apt upgrade -y
```

Next, we'll install some prerequisite applications:

```
apt install apt-transport-https ca-certificates curl lsb-release gpg git nano wget openssh-server -y
```

After this completes, we need to add the Minarca keys to the keyring

```
curl -L https://www.ikus-soft.com/archive/minarca/public.key | gpg --dearmor > /usr/share/keyrings/minarca-keyring.gpg
```

After that, we'll add the Minarca repository to our repository list with:

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/minarca-keyring.gpg] https://nexus.ikus-soft.com/repository/apt-release-$(lsb_release -sc)/ $(lsb_release -sc) main" > /etc/apt/sources.list.d/minarca.list
```

Now we can update our packages. This should update and include the Minarca packages without any errors:

```
apt update
```

And, finally we'll install the minarca-server on our system.

```
apt install minarca-server -y
```

Access the Web UI

You can now go to the IP address of your server on port 8080, and you should be presented with the initial login screen of the Minarca Server. In my case I go to http://192.168.10.25:8080

The default login credentials are:

username: admin
password: admin123

You should absolutely change the password for the admin user after your first login. You can do that by navigating to 'Admin area' in the top navigation, then to the 'Users' tab below that.

In the Admin row click 'Edit'. Change the Password with a new, long, strong, complex password, and save it in your password manager, then save it in Minarca Server.

Adding a Different Default Storage Location

This took me a bit of time to figure out completely, but it's not too bad. By default, Minarca Server uses the main drive for your main file system / operating system. So, if you have a 500 GB SSD, that's what it will use by default. In my case, I'm putting this in an Incus Container, and I want to use the built in ZFS Array that has quite a bit more space, so I've created a Storage Pool in Incus, and a Storage Volume inside that called "backups". I have mounted the storage volume in Incus (similar to what you'd do in Proxmox with an LXC container) at "/mnt/backups" for this container (not to be confused with the default path that Minarca setup at "/backups").

However you determine to mount or add storage is fine, you just need to know the path to that new, hopefully larger storage location.

First, we need to edit a configuration file to add our new base storage directory for Minarca.

```
sudo nano /etc/minarca/minarca-server.conf
```

In this file, add a new line at the end, and put the following in that space, adjusting the path to match your new storage path.

`minarca-user-base-dir=/mnt/backups` ← You would of course put your own storage path in place of "/mnt/backups"

Save the file with CTRL + O, then press Enter to confirm, and use CTRL + X to exit the nano editor.

We need to make sure the "minarca" user and group owns the new storage directory, and that we have the proper permissions set on that directory as well:

```
sudo chown minarca:minarca /mnt/backups
```

```
sudo chmod 750 /mnt/backups
```

Next, we need to copy the ".ssh/aurhtorized_keys" from "/backups" for the minarca user to our new base storage directory.

`cp -r /backups/.ssh /mnt/backups/` <-- Notice the end slash (/), it's important here.

Now we need to change the ownership of the .ssh folder and it's contained files:

```
chown -R minarca:minarca /mnt/backups/.ssh
```

After changing the ownership we need to set the proper permissions of the .ssh folder and the containerd authorized_keys file.

```
chmod 700 /mnt/backups/.ssh
```

```
chmod 600 /mnt/backups/.ssh/authorized_keys
```

Finally, we need to tell Minarca that it has a new Home directory for storage. We need to stop the minarca-server service:

```
sudo systemctl stop minarca-server
```

Now, tell it about it's new home directory.

```
sudo usermod -d /mnt/backups/ minarca
```

> INFO: The author of Minarca has suggested, however, that we mount our extra storage as the path `/backups`, then install Minarca which will use that storage space by default.

And restart the service:

```
sudo systemctl start minarca-server
```

Now we can add new users, and their storage will be created in the updated storage location with proper access and permissions.

Permissions in Minarca

There are several places where we adjust permissions in this system. The permissions and ownership, as I understand them from both the documentation and the project developer (who is absolutely awesome by the way) is as follows:

- A default backup location will be created at the path `/backups`
  - All users added to the server will have their own directory in the `/backups` directory.
    - If you have users 'brian', 'beatriz', and 'sofia', then the backups directory will look like this:

- ```
  root -
      backups -
         brian -
         beatriz -
         sofia -
  ```

We, wanting to potentially have more storage, may need / want to change our storage location (explained above), in which case only the location of the `/backups` level directory will change. Everything else remains the same. We moved from `/backups` to `/mnt/backups` above. So our structure looks like

```
root -
    mnt -
       backups -
          brian -
             repository1 -
             repository2 -
          beatriz -
             bea-mac -
             bea-repo-1 -
             bea-mac-2 -
          sofia -
             sofi-lin -
             sofi-mac -
             sofi-game -
```
```

For each of the folders starting at "backups" the ownership must be set to `minarca:minarca`. The permissions at the "backups" level is `750 (drwxr-x---)`.

The permissions of each user's folder is `770 (drwxrwx---)`.

The permissions of the repositories inside the user's folders are `755 (drwxr-xr-x)`.

Once all of these are set correctly, you should be able to connect with the client. If you are having issues connecting, it is likely a permissions or ownership issue.

What if I already have a user, but want to move their storage?

This can be done with relative ease.

From the server command line, first we'll copy the users current directory from /backups/<username> to our new storage location (in this case /mnt/backups/).

`cp -r /backups/brian /mnt/backups/` ← replace "brian" with your user's name. Also note that the slashes (/) are important here. We are saying to copy the folder "brian" and all of it's contents to the new directory of "/mnt/backups/".

Next, we need to make sure that the folder is copied with the appropriate ownership and permissions. Nothing should change in the copy process, but it's good to make sure.

`ls -al /mnt/backups/`

If you don't see something like the below for you folder you moved, then we need to update the ownership and / or permissions on that folder.

`drwxr-xr-x 2 minarca minarca 2 Sep  9 14:31 brian`

`sudo chown -R minarca:minarca /mnt/backups/brian` ← again, use your user's name here.

`sudo chmod -R 755 /mnt/backups/brian` ← changing to use your user's name.

Now, we can setup more users by adding them through our web administration portal. This will add them properly, and with the appropriate permissions to begin with. Or, you can move more users, and you'll need to make the same ownershp and perimissions changes as above.

Setup the Minarca Client

If you haven't yet installed the Minarca Client, you can get it at The Client Download Page.

Find the client that matches your Operating System, and install it accordingly. Windows uses a wizard with a bunch of pages to click through (ughhh), MacOS has a .dmg to install from, and Linux has a few options. The easiest is the portable file, but it's not "installed". I'd love to see a flatpak of this at some point, but it's not there yet.

If you download the portable file for linux, make sure to extract it, then go into the folder that is created, and you can run the file 'minarcaw' by double clicking on it.

If you are running on a server, or headless system from the CLI, not to worry, we can do what we need to from the command line as well. We'll get to that in a bit.

Assuming you got the GUI open, you'll need to fill in the details there.

The Remote Minarca Server is the URL or IP and Port of your server (e.g. http://192.168.1.25:8080). I have not tried to reverse proxy this yet, as I would want to run something like this over my VPN rather than the public internet (though it does use port 22 for the SSH communication).

Your user name is the user you setup in the Minarca server.

The user's password is the one you set in the Minarca server.

> NOTE: if you enabled 2FA for this user, then you'll need a token instead of the password.

The repository name is generally created for you, but you can change it as long as it's unique.

Now click 'Sign In'.

If everything is configured correctly, you should see a new view open up.

Here, we need to do a little more setup, but it's just to make sure we are getting the right files / folders backed up. By default Minarca tries to select the right things, but it never hurts to make sure.

As you can see, it sets up the user's Documents and the Minarca config by default. This is fine, but I want to make sure it's just my Documents, and not my entire home folder perhaps. Or maybe I want to backup my Music and Photos as well. Whatever you case may be, you can click on the 'Select Files' option in the top navigation, then you can enable or disable any of the selected options, or simply click the 'Trash' icon to remove them from the view. I removed everything, then went in and selected just the folders I wanted by clicking the 'Add Folders' button. Make sure to double-click into the folder in order to select it.

Next, you'll want to set a schedule so Minarca will backup automatically and only get the differences from what is already there. This saves on space, bandwidth, and cost in many cases.

After you've setup your folder, and schedule, you can manually run a backup by going back to the 'Home' tab, and clicking 'Start Backup'.

Note for Windows Users

If you are setting up a backup for a machine where the user will be logged out, or not be logged in after a reboot, then you'll want to enable the option to allow backup of

Logs

As you go through the setup logs will be useful for any troubleshooting. The server has a log area you can view through the Web User Interface. This is a good place to start.

For the clients, you'll want to look for any logs in

On Windows

`%LOCALAPPDATA%\minarca\`

On Linux

`~/.local/share/minarca/`

On MacOS

`~/Library/Logs/Minarca/`

The Minarca CLI

If you need to backup a machine using the CLI (Command Line Interface), such as a headless machine, or server based system, it's not overly difficult.

First, you'll want to download the latest client to the systen you want to backup. I suggest using the .tar.gz for Linux systems in this case, as you can extract it, and more easily use the CLI `minarca` command from there.

1. Connect the client to your Minarca server:

`./minarca configure -r <remote url> -u <your username> -p '<your password>' -n <repository name for this machine on the server>`

If you enter this all correctly, you should see a message that says:

`Linked successfully` in the terminal view.

2. Set a path to include in the backups:

`./minarca include pattern </path/to/include>`

You can use wildcard characters, such as * and ? to build out a file / folder pattern of things to include. You can also use `exclude` in the same way if it's easier to first include a full directory,

then only exclude a few things you don't want backed up (like logs, etc).

3. Run your first backup of the included files / folders:

```
./minarca backup --force
```

In this case we force the first backup only because we have not yet setup a schedule for our backups.

4. After successful completion of your initial backup, let's set a daily scheduled backup to run automatically.

```
./minarca schedule --daily
```

5. Check the status of minarca:

```
./minarca status
```

This can show you whether it's running, linked, and when the last successful backup was, etc.

6. Now that we've scheduled our backups, let's start minarca running in the background with:

```
./minarca start
```

Congratulations! You now have Minarca server and client(s) up and running, and you are backing up your various machines to a centrally hosted server! Well done. Make sure to watch the video for the full walkthrough.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

Buy me a Beer / Coffee:
https://paypal.me/BrianMcGonagill