How to perform a full system backup

Overview

The backup utility is designed to serve two purposes: to provide a basic means for taking a snapshot of an entire GroundWork Monitor system for regular-backup purposes during maintenance windows, and to generate and use such a backup around a system upgrade, as a fallback measure. The discussion here centers around the second use case.

Starting with the GWMEE 7.0.2 release, the backup tool is now named `gw-backup-br387.2-linux-64` . This version only supports backing up and restoring PostgreSQL-based GroundWork Monitor releases (6.6.0 and later). The tool can be downloaded from here:

There are currently no attachments on this page.

For purposes of using this tool during a system upgrade, you will run it on the GroundWork Monitor server, even if that system references a remote database.

Special-case use for sites with large RRD-file collections
The instructions shown immediately below will suffice for running the tool on most systems. However, if you have a large site with a lot of storage space allocated in these directories:

- `/usr/local/groundwork/rrd/`
- `/usr/local/groundwork/cacti/htdocs/rra/`
- `/usr/local/groundwork/nagios/var/archives/`

then a special procedure may be more efficient during a system upgrade. See the *Upgrading a large site* section below for details.

Special-case handling for a separate archive database
If you have the unusual case where your `archive_gwcollagedb` database does not reside in the same PostgreSQL instance (local or remote) as the rest of the GroundWork databases accessed by a single GroundWork Monitor server, special steps will need to be taken to capture and/or restore all the database data. Contact GroundWork Support in this situation, and have them ask for instructions from GroundWork Engineering.

CONTENTS

RELATED RESOURCES

- System Maintenance How To's

WAS THIS PAGE HELPFUL?

- Leave Feedback

1.0 Backing up to a backupdb-type tarball

For purposes of using this tool during a system upgrade, we use the `--action backupdb` form of backup, not the `--action backup` form. This will capture the full set of GroundWork files, including both GroundWork RRD files and Cacti RRD files. It will also include, embedded inside the backupdb tarball, a full dump of all of the PostgreSQL databases.

- Before you run the backup, consider how much space it will take. It will end up as a gzipped tarball, which will contain:
  - The entire `/usr/local/groundwork/...` file tree, plus a few small `/etc` system files.
  - A dump of all of your PostgreSQL application databases, even if they reside on a separate remote-database server, when you use the `--action backupdb` mode.
- The simplest way to estimate how much space you will need is to run this command on both your GroundWork Monitor server and all of your separate database servers, if any:

```
du -sm /usr/local/groundwork
```

Add all the resulting numbers, which are printed in megabytes. Then double the result, to ensure that you have enough space for both unzipped and zipped forms of the tarball.

- In order for the `--action backupdb` mode of the backup utility to work, the PostgreSQL database must be up and running before you invoke the tool. (The rest of the GroundWork system can be up or down at this point. The tool will shut down the other components while saving files, then bring the entire system up afterward.) So either on the GroundWork Monitor server (for a local database), or on the database server (for a remote database), run this command:

```
service groundwork start postgresql
```

- If you have enough space in the `/tmp` directory, this version of the backup tool should normally be run this way when operated during an upgrade situation, substituting your own database password for the `postgres` user:

```
cd {wherever you have parked the backup utility}
./gw-backup-br387.2-linux-64 --action backupdb --password xxxxx
```

That command will deposit the resulting tarball in the `/tmp` directory, in a timestamped file such as the following:

```
/tmp/gw-backup-20140521155425.tar.gz
```

You should rename this file to include the hostname on which it was taken, the GroundWork version it represents (see the restore instructions below for why this is important), and the backup type (e.g., `backupdb`):

```
cd /tmp
mv gw-backup-20140521155425.tar.gz gw-backup-20140521155425-MYHOST-6.7.0-backupdb.tar.gz
```

Then save a copy of this file in a safe place, outside of the `/tmp` directory and outside of the `/usr/local/groundwork/` file tree. Keep a copy of this file in the `/tmp` directory for reference by the GroundWork Monitor installer, so it knows you have backed up your system.

- If you did not have enough space in the /tmp/ directory, you can specify some other location by naming an alternate directory:

```
cd {wherever you have parked the backup utility}
./gw-backup-br387.2-linux-64 --action backupdb \
    --password xxxxx --filepath /my/directory
```

Afterward, you should rename the file as above:

```
cd /my/directory
mv gw-backup-20140521155425.tar.gz gw-backup-20140521155425-MYHOST-6.7.0-backupdb.tar.gz
```

and save it in a safe place, as above. Then, if you took the backup in preparation for a system upgrade, create an empty file in the `/tmp` directory, so the GroundWork Monitor installer knows that you have backed up your system:

```
touch /tmp/gw-backup-20140521155425.tar.gz
```

2.0 Restoring from a backupdb-type tarball

Unrolling the backupdb tarball is not enough
The backup tool takes a variety of actions beyond just unrolling the backupdb tarball, to ensure the completeness and integrity of the restored system. You should not think that you can bypass use of the tool by just unrolling the tarball manually yourself.

To restore the system from a backup taken as described above, you will need to first re-install a fresh copy of the GroundWork version that it represents. (This is why you renamed the backup file above to include the GroundWork version number, so there can be no mistake now about what version to install.) *During this re-install, you must use exactly the same `postgres` database-user password as was in force when the backup was taken.* This system refresh will both provide the necessary files for the restore operation to function, and (in a failed-upgrade scenario, where you're rolling back to the previously-installed release) eliminate any mixture of files from the old and new releases.

- Shut down the GroundWork server.

```
service groundwork stop
```

- If your system used a remote PostgreSQL database, re-install that part first, using the installer for the same version of GroundWork Monitor as is represented by your backupdb tarball. This will wipe out all the content of your databases, but you will get it back when you restore the system from your backupdb tarball, below.
- On your GroundWork Monitor server, re-install the GroundWork Monitor software, including the PostgreSQL components if you are not using a remote database, using the installer for the same version of GroundWork Monitor as is represented by your backupdb tarball. At the end of this process, leave the GroundWork Monitor software running; this is required for the final restore operation.
- The final restore operation can be accomplished with this command, run on the GroundWork Monitor server. Provide your database password for the `postgres` user and the full pathname of the particular tarball you wish to restore:

```
cd {wherever you have parked the backup utility}
./gw-backup-br387.2-linux-64 --action restore --password xxxxx \
    --filepath /my/directory/gw-backup-20140521155425-MYHOST-backupdb.tar.gz
```

3.0 Upgrading a large site

The procedure outlined above will work on any system, large or small, whether you have local or remote databases, provided you have enough space for the backup file. However, that procedure may be somewhat inefficient during a system upgrade if you have a large amount of space allocated for RRD files and certain log files. In such cases, the time taken to back up and restore these particular file collections may be excessive. An alternative procedure is available for such cases. This procedure manually moves aside the large file collections, then puts them back afterward. Keeping these file collections out of the backup file makes the processing faster, at the cost of you having to manage these parts of the system on your own.

Be very careful when following this procedure
This procedure speeds up the processing by keeping large collections of critical data out of the backup tarball. This means you won't have any extra copy of this data lying around to recover from if you make a mistake. So tread carefully, checking your work at every stage to make sure you have both saved the data aside properly, and that you do not inadvertently destroy the data by making mistakes in these commands. If any of that concerns you, and you prefer the safety and convenience of more automated procedures, then make sure you have enough space for a backupdb tarball, live with the extra time it will take to save and possibly restore these file and database collections, and follow the simpler main procedure above.

The procedure in this case is as follows.

Back up a large system

Measure the space required by the backup, as described in the main procedure above. Also take the following measurements:

```
du -sm /usr/local/groundwork/rrd
du -sm /usr/local/groundwork/cacti/htdocs/rra
du -sm /usr/local/groundwork/nagios/var/archives
```

If these locations represent a large amount of space (say, greater than 1 GB; this is a matter of judgment), it may be worthwhile to follow the instructions here. This will result in a tarball that does not include the file trees at these locations. This tarball will also not include a dumpfile of all your PostgreSQL application databases, so saving and restoring them will need to be handled separately if you are using a remote database. (For a local-database-only setup, the raw database files will be part of the tarball, and that will suffice.)

Examine and correct the filesystem structure before continuing
Because the file-tree movement described below attempts to move data within the same filesystem but outside the `/usr/local/groundwork/` file tree, this process will only work if `/usr/local/groundwork` is not the root of a mounted filesystem. If it is, fix that up first, by shutting down the system, then moving around mount points and/or symlinks, before proceeding with the procedure below.

Instead of running the backup tool in `--action backupdb` mode, do the following:

- Shut down the GroundWork Monitor system:

```
service groundwork stop
```

- If you are using a remote PostgreSQL database, run these steps on that separate server.
  - Create a `backupdb` tarball on the database server, as described in the main procedure above, by starting the database on the remote server and running the `gw-backup-br387.2-linux-64` tool directly on that database server.
  - At the end of creating that `backupdb` tarball, rename it as described, using the name of the database server in the filename.
  - Save a copy of this `backupdb` tarball in a safe place, as described.
- Back on the GroundWork Monitor server, change the working directory to the parent directory of the `/usr/local/groundwork/` directory. For instance, if `/usr/local/groundwork` is an ordinary directory in your root filesystem (not a symlink to elsewhere, and not mounted from some other filesystem), you would run this command:

  ```
  cd /usr/local
  ```

  Conversely, if `/usr/local/groundwork` is a symlink to `/mountpoint/somepath/groundwork`, where `/mountpoint` is some non-root filesystem, then you might run this command:

  ```
  cd /mountpoint/somepath
  ```

- From that working directory, run these commands to move the large-space directories listed above to some other location within the same filesystem but outside the `/usr/local/groundwork/` file tree.

  ```
  mv groundwork/rrd                saved-rrd
  mv groundwork/cacti/htdocs/rra   saved-cacti-htdocs-rra
  mv groundwork/nagios/var/archives saved-nagios-var-archives
  mkdir groundwork/rrd
  mkdir groundwork/cacti/htdocs/rra
  mkdir groundwork/nagios/var/archives
  chown nagios:nagios rrd
  chown nagios:nagios cacti/htdocs/rra
  chown nagios:nagios nagios/var/archives
  ```

  The point of these move commands is that they should accomplish nothing other than a directory rename in the same filesystem, without any other file copying.

- Run the backup utility in a mode which will not capture the file trees you just moved aside. For `--filepath /MYLOCATION`, follow the thinking in the main procedure above to specify either `/tmp` or some other `/my/directory` as appropriate, depending on where you have enough space to hold the backup file.

  ```
  cd {wherever you have parked the backup utility}
  ./gw-backup-br387.2-linux-64 --action backup --filepath /MYLOCATION
  ```

  When the system comes back up after the backup is complete, it will create new RRD files as needed. These will start out empty, so they will not contain your historical data. That is considered acceptable at this point because you are about to upgrade your system, and you will put back the historical data after the system upgrade is complete.

- Once the backup is complete, rename the resulting tarball, as described in the main procedure. But this time, use `backup` (as opposed to `backupdb`) as the backup type in the filename, so it is clear that these special instructions must be followed to restore the system.

  ```
  mv gw-backup-20140521155425.tar.gz gw-backup-20140521155425-MYGWHOST-6.7.0-backup.tar.gz
  ```

- Save the backup tarball in a safe place elsewhere, as noted in the main procedure.
- If necessary, as described in the main procedure, touch an empty tarball in the `/tmp` directory.

Perform the system upgrade

Follow the standard release instructions to upgrade your system to the later release.

Upon a failed upgrade, restore the entire system

If the upgrade failed, take these steps to roll back completely to the previous release. (The archived Nagios log files can probably just be ignored, as they will have little value going forward.)

- Shut down the GroundWork Monitor system:

```
service groundwork stop
```

- If your system uses a remote PostgreSQL database, re-install a fresh copy of the PostgreSQL components on the database server, using the installer for the same version of GroundWork Monitor as is represented by your backupdb tarball. This will wipe out all the content of your databases, but you will get it back when you restore the PostgreSQL software and databases from your backupdb tarball, below.
- On your GroundWork Monitor server, re-install a fresh copy of the GroundWork version that you were attempting to upgrade from, including the PostgreSQL components if you are not using a remote database. As noted above in the main procedure, use the same `postgres` database-user password as was in force when the backup was taken. (That's not critical in this scenario, but doing so will avoid confusion. After the restore, the password will be back to what it was when the backup was taken.)
- If your system uses a remote PostgreSQL database, restore it now using these steps.
  - Shut down the GroundWork Monitor system, using this command on the GroundWork Monitor server:

```
service groundwork stop
```

  - Make sure the PostgreSQL database is running on the database server, by running this command on that server.

```
service groundwork start postgresql
```

  - Restore the `backupdb` tarball you created earlier on the database server, using this command run on that server. Provide your database password for the `postgres` user and the full pathname of the particular tarball you wish to restore:

```
cd {wherever you have parked the backup utility}
./gw-backup-br387.2-linux-64 --action restore --password xxxxx \
    --filepath /my/directory/gw-backup-20140521155425-MYDBHOST-backupdb.tar.gz
```

- The final restore operation can be accomplished with these commands, running back on the GroundWork Monitor server. For this form of restore, you need not specify a password. But obviously, substitute in the complete path to and name of your own backup tarball.

```
cd {wherever you have parked the backup utility}
./gw-backup-br387.2-linux-64 --action restore \
    --filepath /my/directory/gw-backup-20140521155425-MYHOST-backup.tar.gz
```

- Once the restore is complete, take the same steps as listed in the successful-upgrade scenario (the next section) to put your RRD files back in place: shut down the system, fiddle with the directories, and bring the system back up.

Upon a successful upgrade, restore the RRD files

If the upgrade worked, take these steps to put back your saved RRD files. (Again, the archived Nagios log files can probably just be ignored, as they will have little value going forward.)

- Shut down the system.

```
service groundwork stop
```

- Remove the new RRD file trees:

```
cd /usr/local/groundwork
rm -rf rrd
rm -rf cacti/htdocs/rra
```

- Change the working directory to where you put the saved directories, with a command such as one of these (see above for how you saved these file trees):

```
cd /usr/local
cd /mountpoint/somepath
```

- Move the saved file trees back into place:

```
mv saved-rrd              groundwork/rrd
mv saved-cacti-htdocs-rra groundwork/cacti/htdocs/rra
```

- Bring the system back up.

```
service groundwork start
```

At this point, your old data should be back in place, and your system should be fully operational.

Beware of incomplete backups
Once you have restored the RRD files this way, your `--action backup` tarball will be of limited value for any future restore operations. That is, it doesn't contain the RRD files, and you have moved your saved RRD files back into operational locations where they would not be available if some future restore operation must be performed, without extra work to save them again (if they are still available when you need them, i.e., presuming your disk didn't crash). So at this point, you should take whatever steps you deem appropriate to back up your entire system, as you would for executing a disaster-recovery operation.