Dumping the database with pg_dump

We will use pg_dump as our backup strategy. pg_dump is a postgresql utility that lets us perform a dump of all the data in our database with a single command. In case of a catastrophe, we can use that file to get our database to the exact state it was when the pg_dump was performed. This will save all of our employees, users, clients, downloaded modules, products… etc.

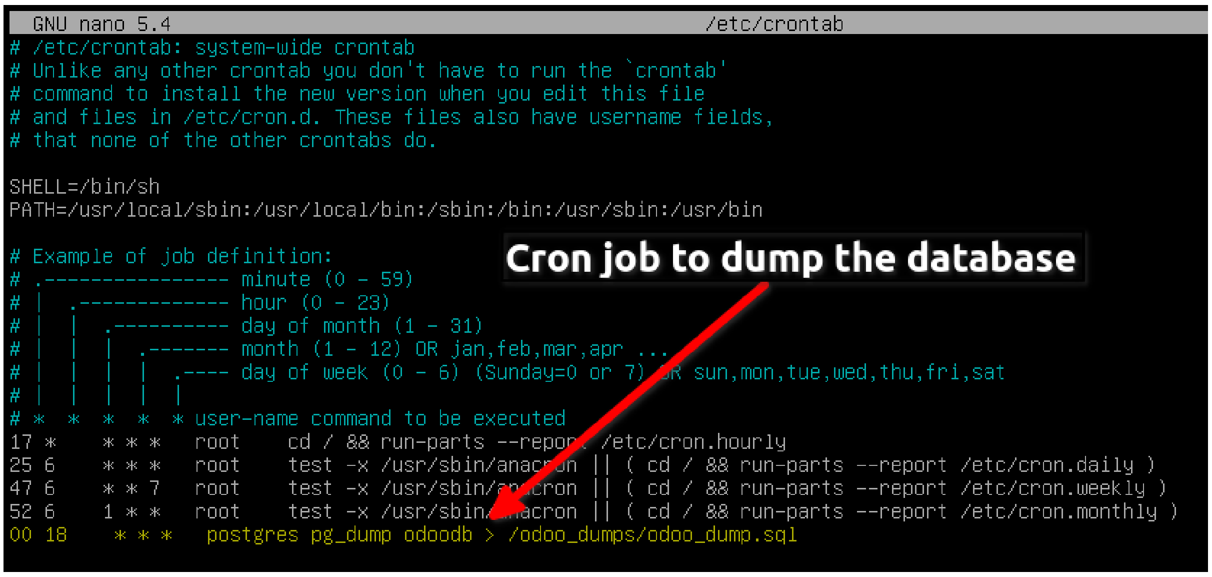
Our dump will generate a .sql file, with all the necessary information to restore the database, and since this dump will be performed every day, we will have access to daily database backups. The scheduling will be performed using a job scheduler, in our case we will use cron since it is built in in the server we chose (Debian) and is a very reliable and known tool.

Scheduling a cron job

Cron is a very useful Linux utility that allows us to schedule commands for certain hours, minutes, days, months or years. It is a very simple and powerful tool and very suitable for backup jobs.

In our case, we want to backup our database daily, around 18:00. We can do that easily with a cron job.

The cron configuration to do this job would be this:



```
  GNU nano 5.4                                        /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
00 18    * * *   postgres pg_dump odoodb > /odoo_dumps/odoo_dump.sql
```

```
00 18  * * *  postgres pg_dump odoodb > /odoo_dumps/odoo_dump.sql
```

This line tells cron to execute the **pg_dump odoodb > /odoo_dumps/odoo_dump.sql** command as the **postgres** user every day, every, month and every year at 18:00.

The file will be saved into the odoo_dumps folder.

Sending backup file to another computer

To keep our data safe in case of loss of the machine where odoo is installed, we can send our file to another computer for safe keeping. We can do this using SCP via ssh.

To start the setup, we will install ssh on another computer, in our case we will use another debian server.

Once we have installed the ssh server, we will configure it to accept key pairs for authentication.

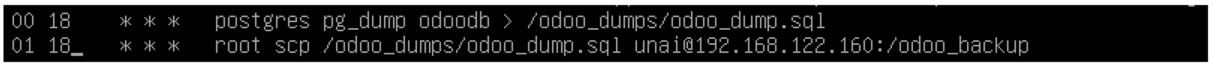
To do this, we will create a key pair in the machine where odoo is hosted using **ssh-keygen**. This will create a keypair. Then we can use **ssh-copy-id <user>@<remoteMachine>** so our destination server can accept our identity.

Once this is done we will configure **/etc/ssh/sshd_config** to not accept password login by modifying one of the parameters: **PasswordAuthentication no**. This will only allow for public key authentication.

Now we can use SCP to send our dump file to another server using the next command:

scp /odoo_dumps/odoo_dump.sql <user>@<remoteMachine>:/<targetDir>

Since it will not ask for password anymore (it will use our key pair for authentication) we can automated this job in cron· Our cron jobs will look like this:



```
00 18    * * *   postgres pg_dump odoodb > /odoo_dumps/odoo_dump.sql
01 18_   * * *   root scp /odoo_dumps/odoo_dump.sql unai@192.168.122.160:/odoo_backup
```

With all of this we will have automatic backups backed up in 2 different machines daily, every day at 18:00.

restoring the database with pg_restore

To then restore the database is a very simple process. Postgresql gives a utility to dfo this: **pg_restore**. To use pg_restore we will only need the dump file that **pg_dump** generated. Then we can restore the database using the next syntax:

pg_restore -h localhost -p 5432 -U postgres -d <databaseName> -v <dumpFile.sql>

And then we will have the database restored to the state of that dump.

Another method to restore the database is using the **psql** command line utility that Postgresql ships with. The command using this utility would be:

psql -U <userName> -d <databaseName> < <dumpFile.sql>