**1. Backup**

**1.1 Mount a Windows file system from Linux**

* Creating the Share on Windows: eg:/10.193.90.156/gitlab\_backup\_donot\_move
* Make a directory ( /mnt/gitlabbackup/)on Linux and mount the share to it
* the mount command like:

mount -t cifs -o username=QPCB36,vers=2.1 //10.193.90.156/gitlab\_backup\_donot\_move /mnt/gitlabbackup/

**1.2 config the gitlab backup information**

* the path is :/etc/gitlab/gitlab.rb
* config the back up information

A backup creates an archive file that contains the database, all repositories and all attachments.This archive will be saved in "backup\_path"

gitlab\_rails['backup\_path'] = "/var/opt/gitlab/backups"

you can change the path according to the requirement.

The backup archives created by GitLab (like:123456\_gitlab\_backup.tar) will have owner/group git:git and 0600 permissions by default.This is meant to avoid other system users reading GitLab's data.

If you need the backup archives to have different permissions you can use the 'archive\_permissions' setting.

gitlab\_rails['backup\_archive\_permissions'] = 0644 # Makes the backup archives world-readable

You may also want to set a limited lifetime for backups to prevent regular backups using all your disk space. To do this add the following lines to

# limit backup lifetime to 7 days - 604800 seconds

gitlab\_rails['backup\_keep\_time'] = 604800

* reconfigure the gitlab

sudo gitlab-ctl reconfigure

**1.3 the backup script**

* The file need to backup

the gitlab archive file

bakdir\_gitlab="/var/opt/gitlab/backups"

gitlab config file

bakdir\_gitlabrb="/etc/gitlab/"

authorized\_keys

bakdir\_authorized\_keys="/var/opt/gitlab/.ssh/"

server host keys

bakdir\_sshhostkeys="/etc/ssh/"

* Gitlab back up

# use this command if you've installed GitLab with the Omnibus package

sudo gitlab-rake gitlab:backup:create

# if you've installed GitLab from source

sudo -u git -H bundle exec rake gitlab:backup:create RAILS\_ENV=production

Also you can choose what should be backed up by adding environment variable SKIP. Available options: db,uploads (attachments), repositories, builds(CI build output logs), artifacts (CI build artifacts), lfs (LFS objects).

Use a comma to specify several options at the same time.

sudo gitlab-rake gitlab:backup:create SKIP=db,uploads

The achive filename will be `[TIMESTAMP]\_gitlab\_backup.tar`. This timestamp can be used to restore an specific backup.

* pack all the files to a \*.tgz

filename=gitlabbackup\_$(date +%Y%m%d\_%H%M)

tar -czPf $filename.tgz $bakdir\_gitlab $bakdir\_authorized\_keys $bakdir\_sshhostkeys $bakdir\_gitlabrb

save the script (/home/script /gitlabbackup.sh )

the more details about the script pls refer to \script\backup\gitlabbackup.sh

**1.4 auto backup jobs**

cron is a unix, solaris utility that allows tasks to be automatically run in the background at regular intervals by the cron daemon. These tasks are often termed as cron jobs in unix , solaris. Crontab (CRON TABle) is a file which contains the schedule of cron entries to be run and at specified times.

add the below command to the /etc/crontab

0 23 \* \* \* root cd /home/script && sh gitlabbackup.sh >> /home/script/gitlabbackup.log 2>&1

do backup jobs at 23:00 every day

**2. Restore**

You can only restore a backup to exactly the same version of GitLab that you created it on, for example 7.2.1. The best way to migrate your repositories from one server to another is through backup restore.

* unzip the files,put the files to the correct path

eg:gitlab achive files should be put to 'backup\_path' (/var/opt/gitlab/backups)

* stop the gitlab service

gitlab-ctl stop unicorn

gitlab-ctl stop sidekiq

* use the timestamp to restore gitlab

gitlab-rake gitlab:backup:restore BACKUP=1454399144 (1454399144 is the timestamp)

* restart the gitlab

sudo gitlab-ctl start