Lab 11.1: Backup Jobs
IN719 Systems Administration

Introduction

Last time we surveyed our systems to identify data we need to back up. Of course, we could just back up everything, but that's probably not necessary. Also, backups do come at a cost in terms of time and system resources required to save them. So, for example we don't need to back up the standard nagios plugins since they are easily restored by installing the package. In effect, they are already backed up to our distribution's apt repository. On the other hand we probably do want to back up our Slack notification plugin, since we customised it for our systems and if our version is lost we would need to recreate it from scratch.

Recall also that we decided that our goal is to backup for availability. We don't need to save data for long-term archive or to maintain multiple versions of backed up data from various dates and times. This means that we can forgo a more complicated automated backup system. Instead we can simply maintain a backup copy of critical data on another server set up to provide storage and update it regularly using a simple shell script invoking `rsync`. More details on this are described below.

1 Procedure

A server has been set up on our VM platform at `10.25.137.160`. Each team has been assigned an account with a user name corresponding to the team's letter (e.g. `a`, `b`, `c`, and so on). Log onto that server and arrange your home directory with subdirectories to organise your backup data. For example, you may want one subdirectory for each of your team's servers. You will also want to set up keypairs so that you can ssh from your servers to the storage server without requiring a password.

On each of your servers, you should have already identified directories and files that you wish to back up. Write shell scripts for each server (they will probably be somewhat different on each host) that use `rsync` to copy data from your servers to the storage server. If you're not familiar with `rsync`, this guide is helpful `https://www.digitalocean.com/community/tutorials/how-to-use-rsync-to-sync-local-and-remote-directories-on-a-vps`. Note that we use `rsync` for this rather than something like `ssh` because `rsync` is particularly efficient for this purpose.

Test your script to be sure you are satisfied that it is copying the required data properly.

Finish by creating a Puppet module to distribute the scripts to each host. This module should also maintain `cron` jobs to run the scripts periodically. For our purposes, it's sufficient to run the scripts four times a day. It's probably best if they don't all run at the same time just to spread out the work on the storage server.

2 Follow up

We made a point of being clear that backup procedures must always be paired with restore procedures. We'll look at this more in the next session, but in the meantime be sure to research the `rsync` commands that you can use to copy data from the storage server to your hosts.