Lab 11.2: Configuring Backup Jobs
IN719 Systems Administration

Introduction

So far we have developed a basis understanding of Bacula and started to deal with two parts of our Bacula setup. Ensure that you finish those tasks before continuing with today's lab:

1. Installing the needed components on various servers;
2. Configuring those components to communicate with each other over the network.

But we're not backing up anything yet. We will work on that now.

All of the backup and restore jobs are configured in the file `/etc/bacula/bacula-dir.conf` on `mgmt`.

1 Configuring a Backup Job

The details of a backup job are configured in a `Job` section in the configuration file. You will have a `Job` section for each client that you back up. If your backup jobs include a lot of shared configuration, you can create a `JobDefs` section that provides default values that can be applied to `Jobs`. These defaults can be overridden if necessary. The use of a `JobDefs` is optional.

A backup `Job` contains a few core components - that themselves have to be defined - and a handful of other values. Important components of a backup `Job` are

- `Client`: the client machine whose data is to be backed up;
- `FileSet`: a set of files and directories on the client machine to back up;
- `Schedule`: when to perform the backups;
- `Storage`: the storage daemon that will hold or backed up data.

All of these elements need their own configurations.

There are also a few values that are important:

- `Name`: each backup job needs a distinct name;
- `Type`: jobs are either `Backup` or `Restore` jobs;
- `Level`: backup jobs can be `Full` or `incremental`.

Here is an example of a backup `Job`:

```
Job {
  Name = "ExampleBackp"
  Type = Backup
  Level = Incremental
  Client = exampleserver-fd
  FileSet = "ExampleSet"
  Schedule = "WeeklyCycle"
  Storage = File
  Messages = Standard
  Pool = File
  Priority = 10
  Write Bootstrap = "/var/lib/bacula/%c.bsr"
}
```

For this to work, we need to define some elements. We need a `Client` definition for `exampleserver-fd`. We need a `Storage` definition. We touched upon these aspects in the lecture. We further need a `Storage` definition that we also did last time. The `WeeklyCycle` schedule definition is included in the default configuration. You can use it as a model to create alternative `Schedule`s. The default `File` pool will suffice.

We do need to look at an example `FileSet`.

```
FileSet {
  Name = "ExampleSet"
  Include {
    Options {
      signature = MD5
      Exclude = yes
     }
     File = /etc
     File = /home
  }
  Exclude {
     File = *.swp
  }
}
```

This `FileSet` will back up everything under `/home` and `/etc` but will ignore vi swap files.