LPIC-2 TRAINING COURSE

Topic 206: System Maintenance

Make and install programs from source

I. Unpacking source code with tar

- E.g 1: `tar zxvf /path/to/tarball.tar.gz`
- E.g 2: `tar jxvf /path/to/tarball.tar.bz2`

II. Building from source

1. `./configure [options]`
   - Eg: `./configure --prefix=/opt/myapp`
2. `make`

III. Install software with `make install`

3. `make [options] install`
   - Eg: `make DESTDIR=/tmp/myapp install`

Exercise 1: Install *netcat* package from source

In this exercise, you will install the ***netcat*** utility from source with some customized attributes and learn how to build this software in one machine and install it onto multiple others

1. Verify that **gcc**, **gcc-c++** and **make** packages are installed. If not, install these packages before continue
   Hint: `[Linux-1]# rpm -q  gcc  gcc-c++  make`
2. Download **netcat-0.7.1.tar.gz** source package to **/tmp** directory
   Hint:
   ```
   [Linux-1]# cd /tmp
   [Linux-1]# wget http://nchc.dl.sourceforge.net/project/netcat/netcat/0.7.1/netcat-0.7.1.tar.gz
   ```
3. Extract the source package
   Hint: `[Linux-1]# tar -zxvf netcat-0.7.1.tar.gz`
4. Configure the package to be installed in ***/opt/myapp*** instead of the default one (***/usr/local***)
   Hint:
   ```
   [Linux-1]# cd /tmp/netcat-0.7.1
   [Linux-1]# ./configure -h # notice the --prefix option
   [Linux-1]# ./configure --prefix=/opt/myapp
   ```
5. Build and install the package in the configured dir, also install the binaries in ***/tmp/app*** for later distributing
   Hint:
   ```
   [Linux-1]# make
   [Linux-1]# make install # install the binaries in /opt/myapp/
   [Linux-1]# make DESDIR=/tmp/app install # also install the binaries in /tmp/app
   ```
6. Run the ***netcat*** utility which is now installed in ***/opt/myapp***
   Hint: `[Linux-1]# /opt/netcat/bin/netcat -h`
7. Archive the binaries from ***/tmp/app*** and distribute it to other machines
   Hint:
   ```
   [Linux-1]# cd /tmp/app && tar -cvf netcat_built.tar  opt/
   <copy the /tmp/app/netcat_built.tar to the / directory other machine (Linux-2)>
   [Linux-2]# tar -C / -xvf netcat_built.tar #change to / then extract the archive
   [Linux-2]# /opt/netcat/bin/netcat -h
   ```

Backup Operation

- **Why backup: Your data is valuable**
- **What to backup: As much as possible**
  - Must backup: ***/home*** and ***/etc***
  - Should backup: ***/var***, ***/dev***, ***/tmp***, ***/root***, ***/boot***
  - Exception: ***/proc*** and ***/sys*** filesystem, ***/etc/mtab*** file…
- **When to backup: Depending on the rate of data change**
  - Normally, a daily backup is often best
  - **RPO**: Recovery Point Objective
- **Where to store backup data: should have at least 2 sets of backup media, store them in different sites**
  - Backup medium: Disk, Network (NAS/SAN), Tape, Optical Media
  - **RTO**: Recovery Time Objective
- **How to backup: two important parts in a backup strategy**
  1. Verify the backup
  2. Test the restore procedure

Types of Backup

Backup Type	Description	Pros	Cons
FULL backup	Backup all the data that are selected to be backed up	- Provides a complete copy of all your data - Easy to locate files which need restoring	- Takes a long time and the most space - Redundant backups created, as most files remain static.
INCREMENTAL backup	Backup all files that have changed since the last backup of any type	- Uses the lease time and space - Lets you back up multiple versions of the same file	- Fiddly to restore file. You have restore the last full backup first, then all subsequent incremental backups in the correct order - Hard to locate a particular file in the backup set.
DIFFERENTIAL backup	Backup all files that have changed since the last FULL backup	- Takes up less time and space than a full backup - Provides for more efficient restoration than incremental backups	- Redundant information stored - Subsequent differential backups take longer and longer as more files are changed.

GNU Backup Tools

- **`cp/scp`: copy files to local/remote media**
  - Backup type: FULL, INCR. (by using **`find`**), DIFF. (by using **`find`**)
- **`tar`: create/restore a archive file from/to a directory**
  - Backup type: FULL, INCR., DIFF.
  - E.g: **`tar -cvf /bkup/home-bkup.tar -g /tmp/snap.snar /home`**
- **`cpio`: superset of `tar`, copy files from and to archives**
  - Backup type: FULL, INCR. (by using **`find`**), DIFF. (by using **`find`**)
  - E.g: **`find /home -print | cpio -o > /backup/home-bk.cpio`**
- **`dd`: backup/restore whole filesystems at once**
  - Backup type: FULL
  - E.g: **`dd if=/dev/sda3 of=/backup/sda3.img`**
- **`rsync`: fast incremental file transfer for remote backup**
  - Backup type: INCR./FULL
  - E.g: **`rsync -av -e “ssh” /home backup-svr:/backup/home-bk/`**

Example: Automating backups with *tar*

This is the ***bash*** script that using **tar** to backup your directories everyday with the following strategy:

- **FULL** backup is made every Sunday, overwrites last Sundays backup
- **INCREMENTAL** backup is made every other days, overwrites the backup file from last weeks

Create this script with the name of ***backup.cron*** and put it in ***/etc/cron.daily/*** for it to run daily

```
#!/bin/bash

# Change the following variables to suite your need
DIRS="/home /etc"                       # Add your directores here, separate them by space
BACKUP_DIR="/backups"                   # Where to store the backups
SNAP_FILE="/backups/snap.snar"          # Information about the previous backup
TAR="/bin/tar"                          # Path to tar

# You should not change these variables
WEEKDAY=`date +%a`                       # Day of week (Sun, Mon…)
BACKUP_FILE=`hostname`_${WEEKDAY}.tgz # Backup file will be named: <hostname>_<day>.tgz

# If today is Sunday, remove the snapshot file to create a full backup
if [ $WEEKDAY = "Sun" ]; then
        rm -f $SNAP_FILE
fi

# Run tar to backup your directories
$TAR -czf $BACKUP_FILE -g $SNAP_FILE $DIRS
```

Adv: Problem with *tar*

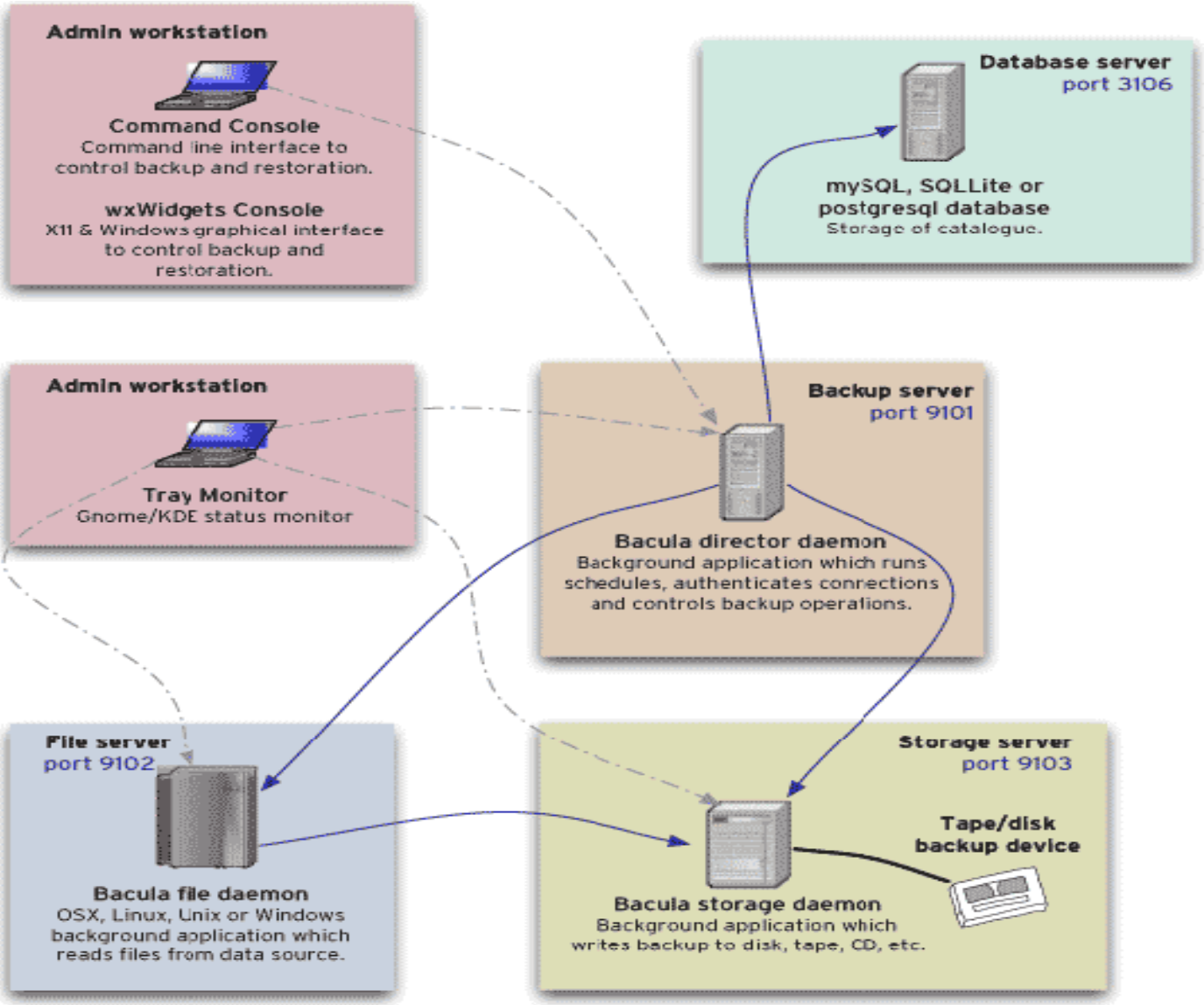
- How do you find the files you need to restore?
- How do you restore to a point in time?
- What is on what medium
- How do you handle hundreds machines?

Adv: Introduction to Bacula

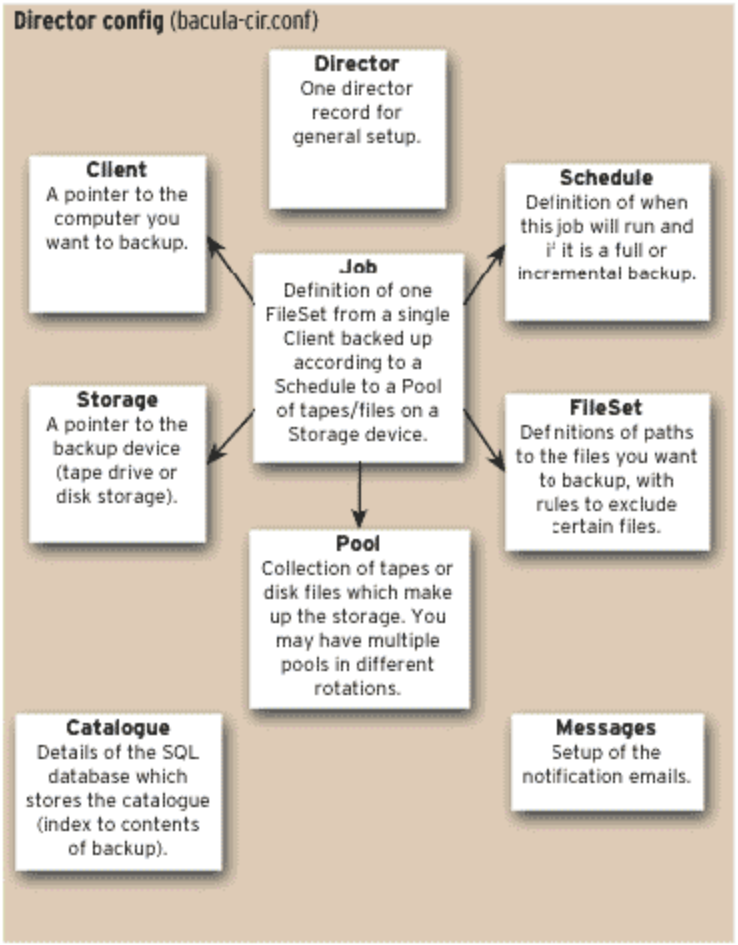
- **Open Source, Client/Server based backup program**
  - Most popular Enterprise grade Open Source program
- **Manage backup, recovery & verification of data across a network of computers**
  - Support multiple OS: Linux, FreeBSD, Solaris, Windows(*), MacOS X(*), True64(**), AIX(**), HP-UX(**)…
  - Support multiple backup medium: Tape, Disk, USB, CD/DVD
- **Manager console: TTY, bat (GUI - Qt 4), wxWidget (GUI), Gnome (GUI), bweb (Web)**

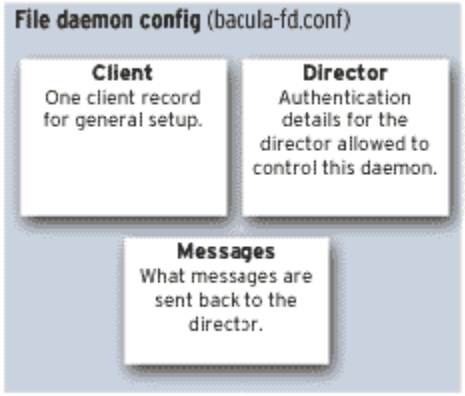
www.bacula.org

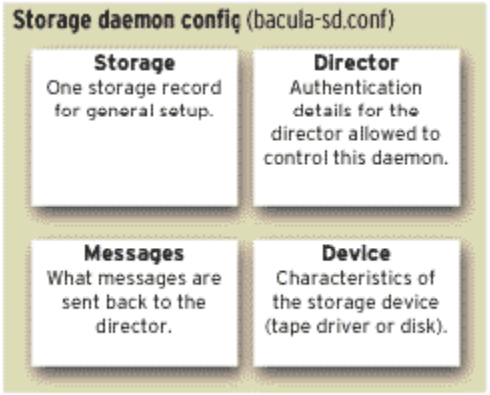
Adv: Bacula Application Interaction



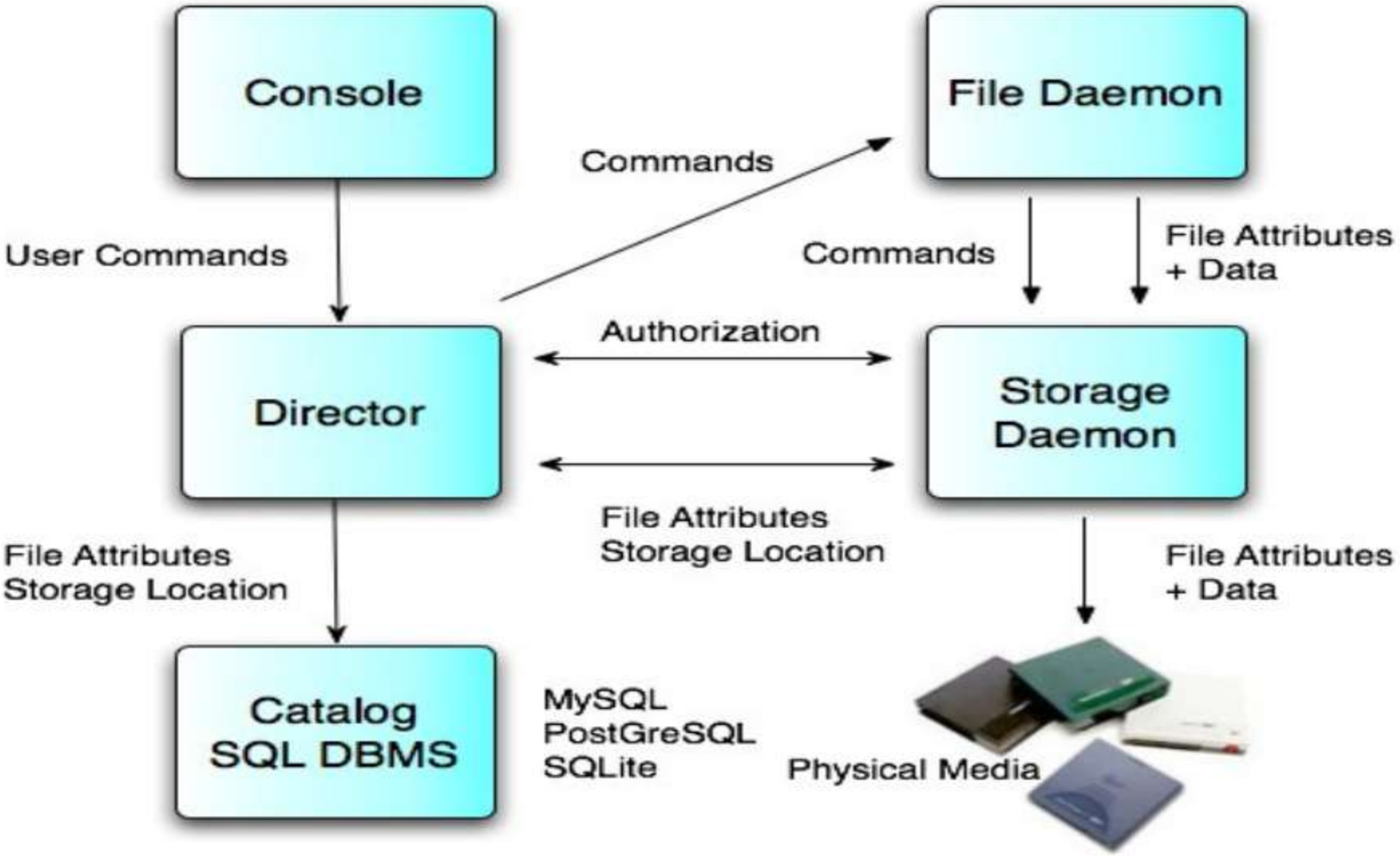
Adv: Bacula Configuration







Adv: Interactions Between Bacula Services



Adv: Bacula Backup Job

- **Jobs are the basic unifying structure**
  - **Name**: Unique job name
  - **Type**: What to do (backup, migrate, restore…)
  - **Level**: Detail level of type (FULL, DIFF., INCR.)
  - **Client**: Where to get the files from
  - **FileSet**: which files to backup
  - **Storage**: where to put the files
  - **Pool**: which set of volumes (tapes, disk) to use
  - **Schedule**: when to do it

Exercise 2: Bacula Director Installation

In this exercise, you will install and configure the most basic funtions of ***Bacula*** and learn how to create a backup job with a GUI interface (***wxWidgets***)

1. Verify that **gcc**, **gcc-c++**, **make** and **wxgtk** packages are installed. If not, install these packages before continue
   Hint: `[Bkup-svr]# rpm -q  gcc  gcc-c++  make wxgtk`
2. Download **bacula** and **depkgs** to ***/tmp/bacula***
   Hint: `[Bkup-svr]# cd /tmp/bacula`
   `wget http://nchc.dl.sourceforge.net/project/bacula/bacula/5.0.3/bacula-5.0.3.tar.gz`
   `wget http://nchc.dl.sourceforge.net/project/bacula/depkgs/15May10/depkgs-15May10.tar.gz`
3. Detar these packages in the current directory (***/tmp/bacula***)
4. Built **sqlite3** from the **depkgs** package
   Hint: `[Bkup-svr]# cd depkgs && make`
5. Built and install bacula from source with supports for **sqlite3**
   Hint: `[Bkup-svr]# cd /tmp/bacula/bacula-5.0.3`
   `[Bkup-svr]# ./configure --enable-smartalloc --enable-conio --enable-bwxconsole --with-sqlite3=/tmp/bacula/depkgs/sqlite3`
   `[Bkup-svr]# make`
   `[Bkup-svr]# make install`
   `[Bkup-svr]# make install-autostart`
6. Create sqlite3 database for using by bacula
   Hint: `[Bkup-svr]# /etc/bacula/make_sqlite3_tables`
7. Start bacula service and verify with the **bconsole**
   Hint: `[Bkup-svr]# bacula start -d 200`
   `[Bkup-svr]# bconsole`

Thank You !

BACKUP SLIDES