OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

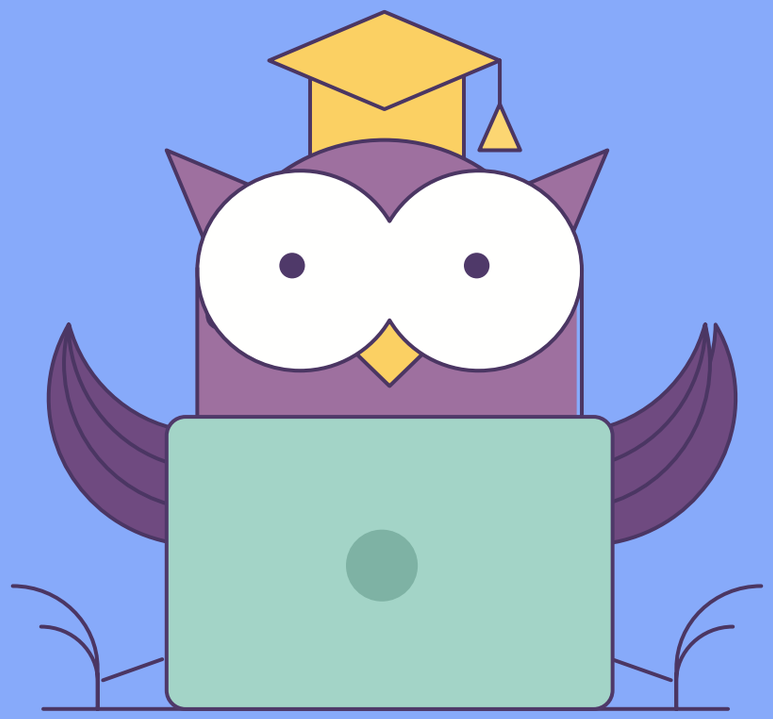
Управление конфигурациями. Ansible

Курс «Администратор Linux»

Занятие № 9

Алексей Цыкунов Альбрехт Леонид

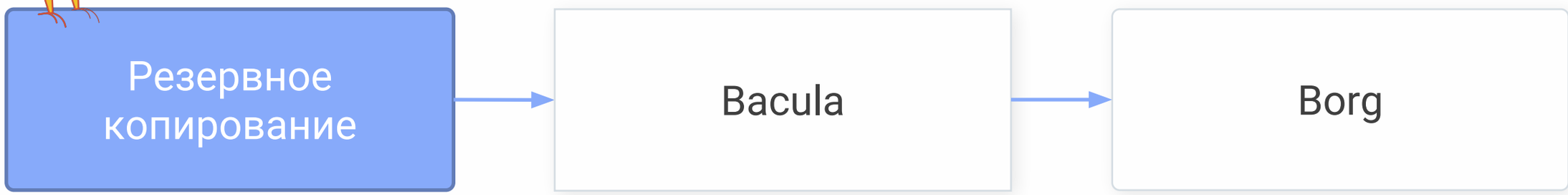
Меня хорошо слышно && видно?

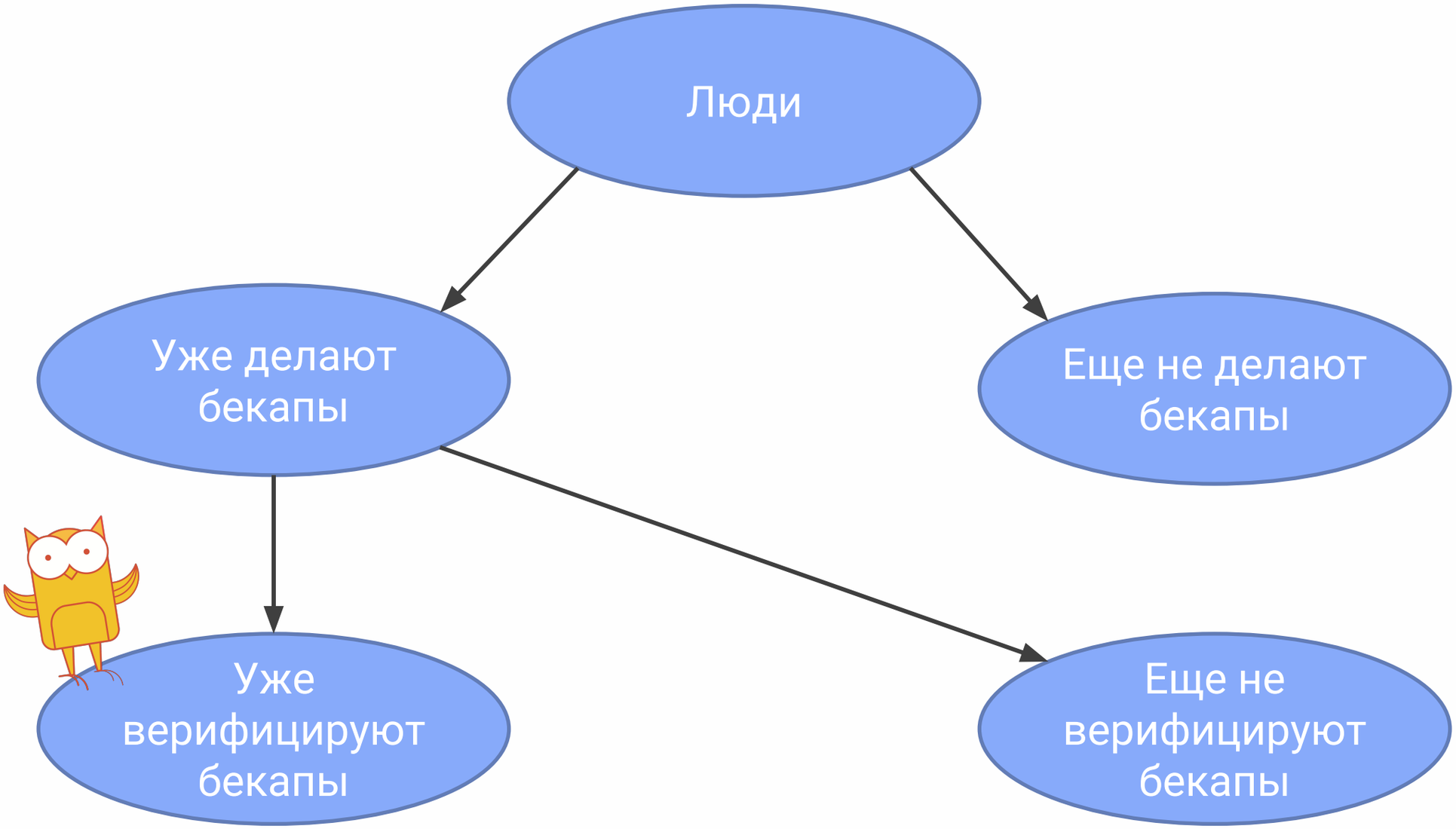


Напишите в чат, если есть проблемы!

Ставьте + если все хорошо
Ставьте - если есть проблемы

Маршрут вебинара




Люди
Уже делают бекапы
Еще не делают бекапы
Уже верифицируют бекапы
Еще не верифицируют бекапы

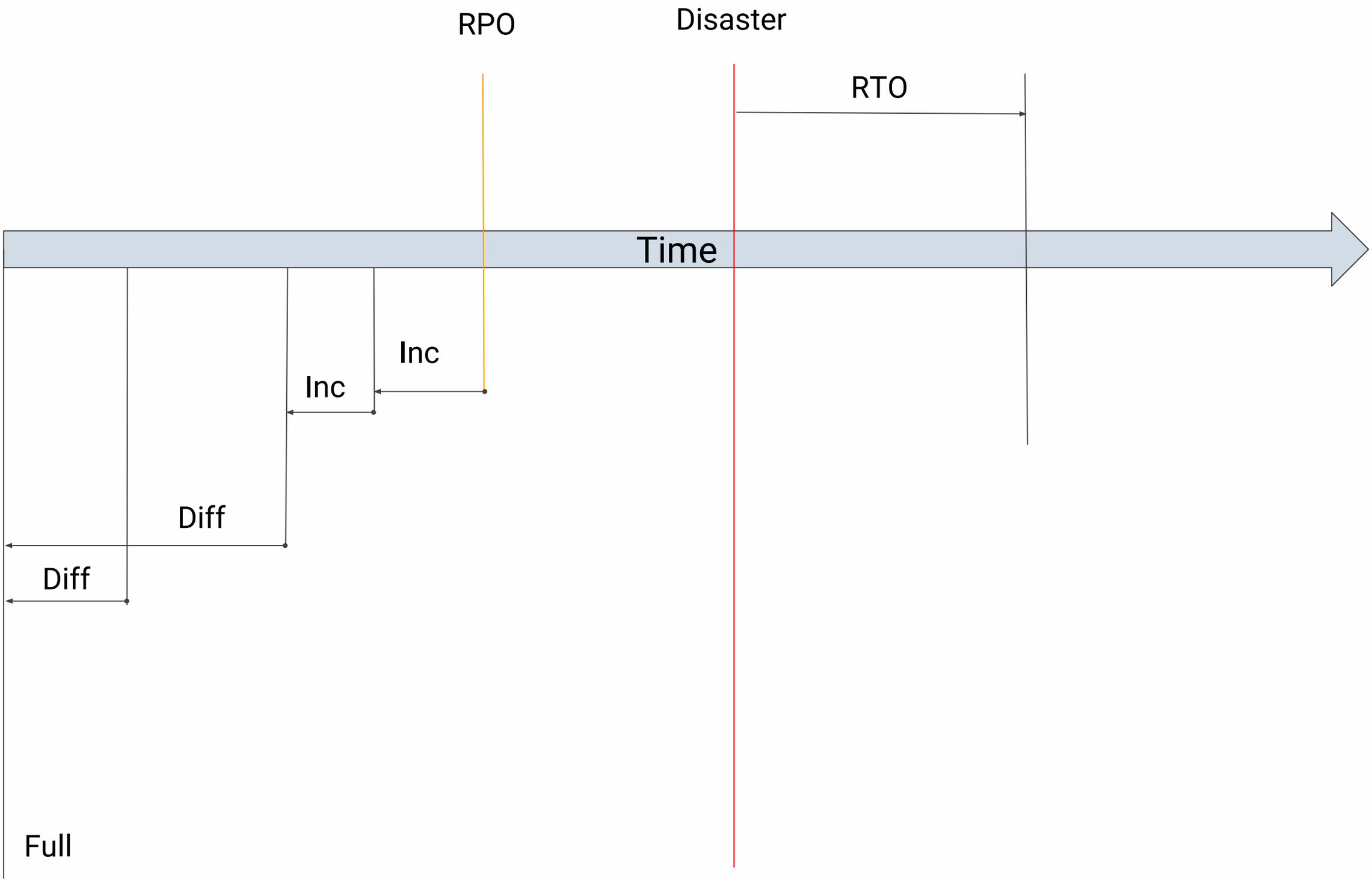
Термины резервного копирования

- **RTO** - Recovery Time Objective. Определяет время требуемое на восстановление РК. Это не то, что диктуется процессом РК, это то требование, которому процесс должен соответствовать. Например, “восстановление из РК должно занимать не больше 1 часа”.
- **RPO** - Recovery Point Objective. Точка во времени (Point in Time) на которую должны быть восстановлены данные. Например, “Данные должны быть восстановлены по состоянии не “дальше”, чем 24 часа с момента сбоя”.
- **Уровень Резервного копирования**(**Backup Level**) - 0-1-2, Full, Differential, Incremental. Различные стратегии выбора данных для копирования.
- **Глубина Резервного копирования** - определяет как долго хранятся резервные копии.
- **Стратегия хранения** - определяет “детализацию” с которой можно восстановиться.

Уровни резервного копирования

- **Full** - Полное резервное копирование. Для восстановления требуется только эта резервная копия.
- **Differential** - Разностное резервное копирование. Копируется только то, что изменилось с последнего резервного копирования. Для восстановления требуется последняя полная и последняя дифференциальная копии.
- **Incremental** - Инкрементальное резервное копирование. Копируется только то, что изменилось с последнего прохода резервного копирования. Для восстановления требуются: последняя полная; последняя дифференциальная (если есть); **ВСЕ** инкрементальные копии с момента последней полной/дифференциальной копии.

Уровни резервного копирования



Требования к системе РК

- Хранение на отдельном носителе или в другом месте
- Надежность места хранения
- Доступность места хранения
- Простота использования

Возможные проблемы

- Размеры резервной копии - негде хранить.
- Время на получение резервной копии
- Время развертывания резервной копии
- Нагрузка на систему для получения резервной копии - резервное копирование либо создает дополнительную нагрузку, либо взводит проблемы перечисленные выше.

Возможные сопутствующие техники.

- Репликация/Дублирование
- Снэпшоты (снимки)
- Журналирование
- Зеркалирование

Все эти техники сами по себе **НЕ ЯВЛЯЮТСЯ** резервными копиями и **НЕ ГАРАНТИРУЮТ** восстановления данных в случае штатного удаления и/или сложного аппаратного сбоя.

Вопросы для планирования РК

- От каких сбоев мы хотим защититься? (Зачем?)
- Что надо копировать?
- Как быстро надо восстанавливать? (RTO)
- На сколько “близко” точка восстановления? (RPO)
- Где все это хранить?
- Сколько это стоит?

В зависимости от типа данных/приложения - может потребоваться свой собственный подход к резервному копированию. Например:

- БД - большой объем данных, сложность получения консистентной копии “файловым” копированием. Примеры: mysql, redis
- Объектные хранилища/большие хранилища файлов. Большой объем данных, большое количество объектов, сложность файлового доступа.

Сопутствующие действия

- Проверка целостности копий/проверка хранилища.
- Мониторинг:
  - Хранилища
  - Агентов
  - ПО
  - Процесса
- Проверка восстанавливаемости.

Ручное копирование

Ручное создание копий.

Основные проблемы:

- Создание упорядоченного архива (решается `date+format`)
- Удаленное хранение (решается с помощью ssh, nfs хранилищ)

Инструменты:

- tar
- rsync
- dump
- dd
- rsync + inotify = lsyncd

Примеры ручного РК

```
[root@otuslinux ~]# tar -czvf nginx_conf.tar.gz /etc/nginx # xvf

[root@otuslinux ~]# dd if=/dev/sdb of=/mnt/backup/sdb.img

[root@otuslinux ~]# rsync -avz --delete /etc/ /mnt/backup/hostname/etc
[root@otuslinux ~]# rsync -az -e ssh --delete 192.168.2.1:/etc/nginx/ /etc/nginx

 [root@otuslinux ~]# xtrabackup --backup --datadir=/var/lib/mysql/
 --target-dir=/data/backups/mysql/
```

Немного про RSYNC

- Поддержка синхронизации как отдельных файлов, так и целых деревьев директорий
- Можно сохранять символические ссылки, жесткие ссылки, владельцев и права файла, метаданные и время создания
- Передача файлов одним потоком
- Поддержка RSH, SSH в качестве транспорта
- Для работы требуется пакет RSYNC на обоих узлах

Не синхронизирует лишнего:

- Rsync находит файлы, которые нужно отправить, используя "quick check" алгоритм (алгоритм используется по умолчанию), ищутся файлы, которые изменились в размере, или в дате последней модификации.

- Принимающий компьютер разделяет свою копию файла на неперекрывающиеся куски фиксированного размера S, и вычисляет контрольную сумму для каждого куска: MD4-хеш и более слабый rolling checksum, и отправляет их серверу, с которым синхронизируется. Сервер, с которым синхронизируются, вычисляет контрольные суммы для каждого кусочка размера S в своей версии файла

RSYNC демон

На отдающем хосте /etc/rsync.conf:

pid file = /var/run/rsyncd.pid
log file = /var/log/rsyncd.log
transfer logging = true
munge symlinks = yes

MySQL Main shard dir
[shard]
path = /mnt/lvm/mnt.db.daily
uid = root
read only = yes
list = yes
comment = MySQL Main Backup
auth users = backup
secrets file = /etc/rsyncd.scrt **----> backup:gfhjkm**

[root@backup ~]# rsync -av --stats --bwlimit=10240 --delete --progress 10.100.210.11::main /data/backup/mysql/main

Части скрипта по бэкапу с LVM

```
# Проверяем состояние MySQL
if [ $(slave_status) == "Yes" ]; then
    mysql -e 'stop slave;'

        # Ротируем и создаем снэпшот
        /sbin/lvcreate -s -p r -L${SNAP_SIZE} -n ${SNAP_DAILY_LATEST} ${BLOCK_DEV}

# Можно запускать slave
mysql -e 'start slave;'
fi

# Запускаем создание бэкапа
time rsync -av --del --progress ${SNAP_DAILY_MNT} ${BACKUP_DAILY_MNT}/latest/

# Удаляем снэпшот
lvremove -f /dev/${VG}/${SNAP_DAILY_PREVIOUS}
```

VCS: Конфигурация системы

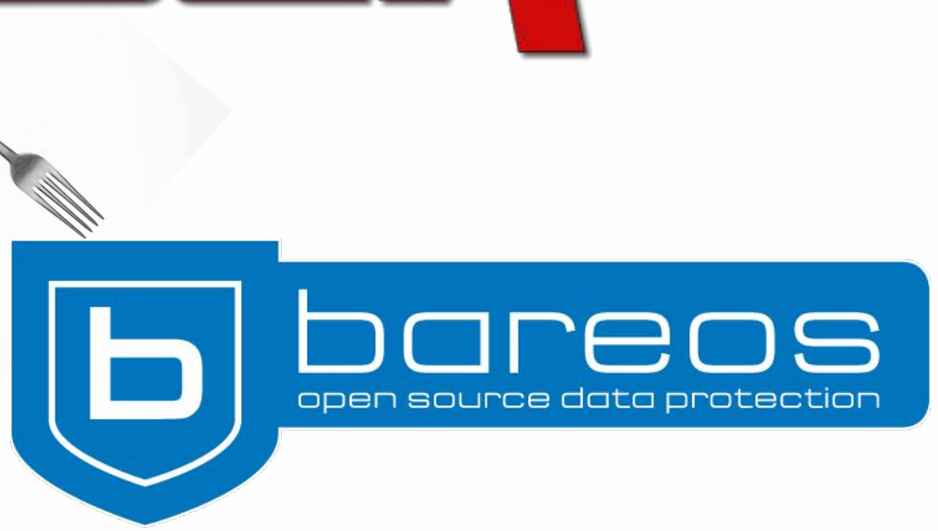
Для конфигурации системы в linux характерно то, что она почти всегда хранится в текстовых файлах. Текстовые файлы прекрасно поддаются версионированию, соответственно их можно “версионировать” в какую-нибудь VCS (cvs, git, svn, mercurial)

VCS: etckeeper

[root@otuslinux ~]# sudo apt/yum install etckeeper

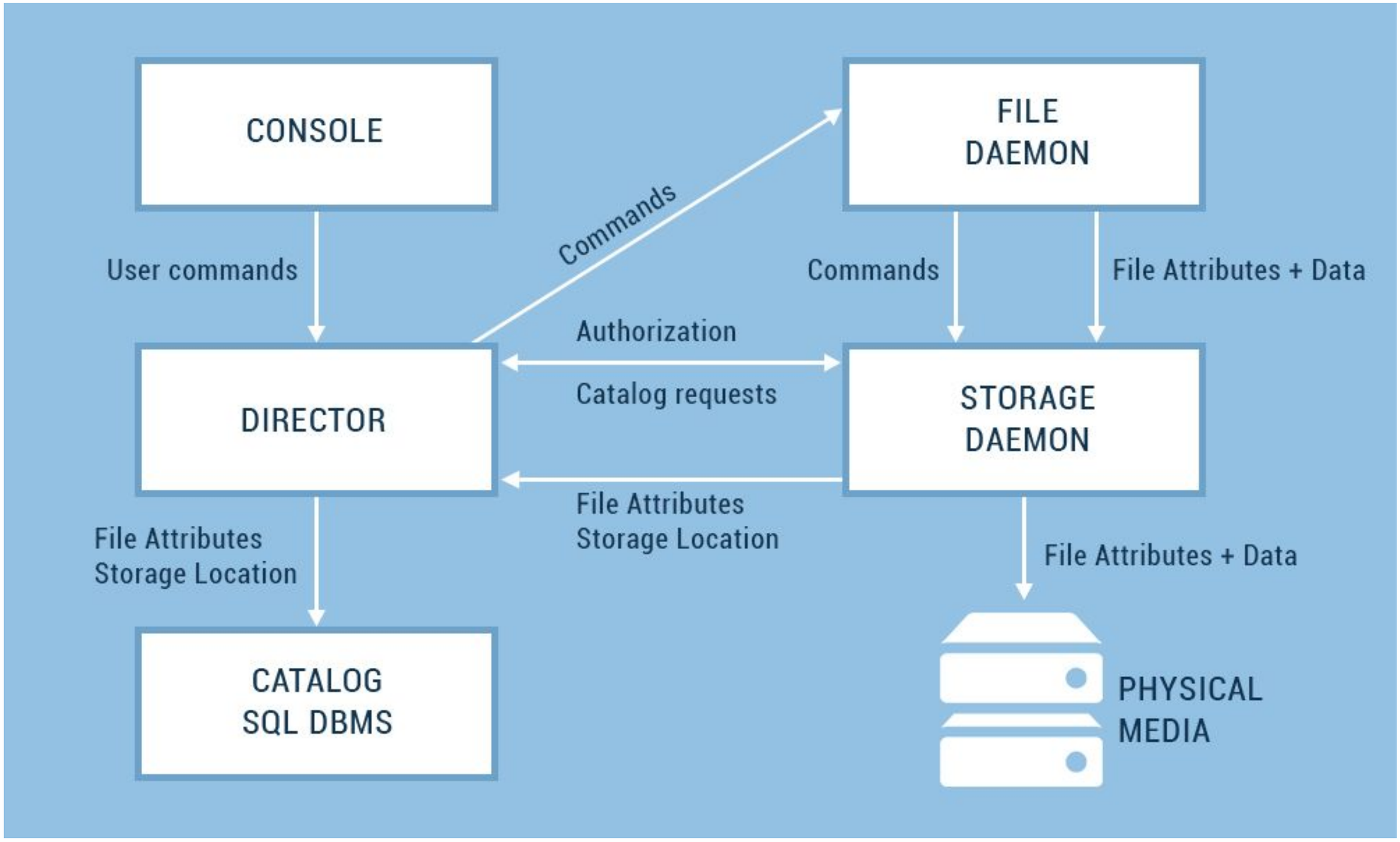
- Дружит с пакетным менеджером и отслеживает установку программ
- Сразу же инициализирует git в /etc/
- Следит за пустыми папками

OTUS

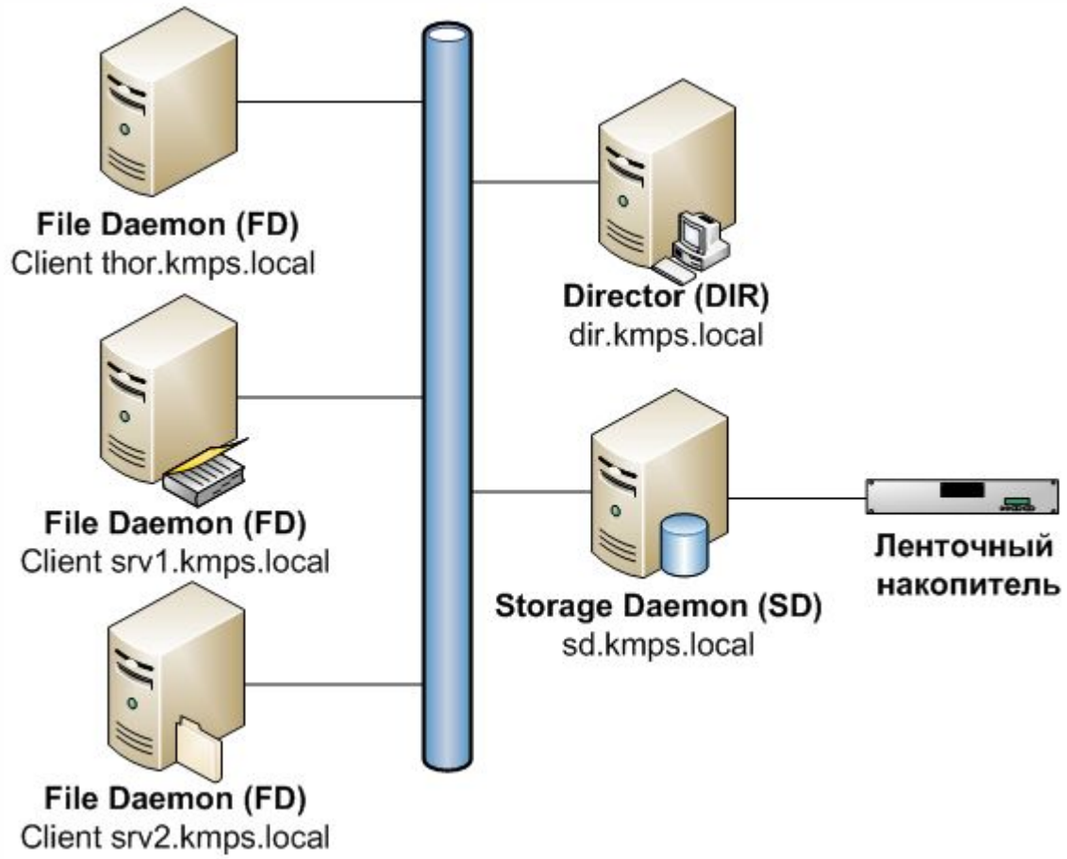


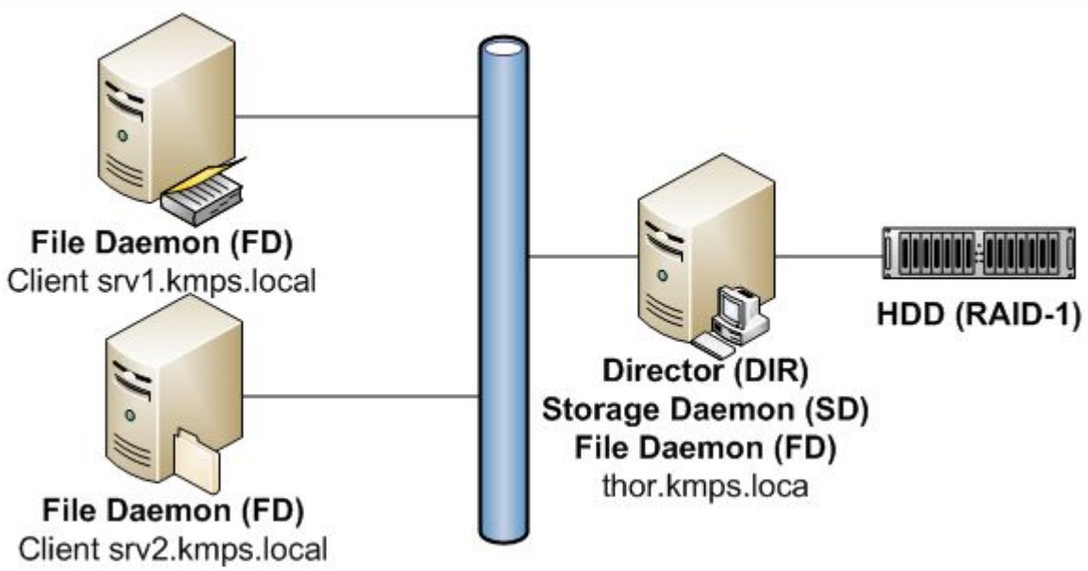
- Bacula Director — процесс управляющий системой в целом(управление, планирование, восстановление бекапов).
- Storage Director — запускается на сервере отвечающим за «физическое» хранение данных.
- File Director — сервис запускаемый на каждом из клиентов.
- Catalog - база данных, в которой хранятся сведения обо всех зарезервированных файлах и их местонахождении в резервных копиях
- Bconsole — консоль управления.

Arch


CONSOLE
FILE DAEMON
User commands
Commands
Commands
File Attributes + Data
Authorization
Catalog requests
DIRECTOR
STORAGE DAEMON
File Attributes
Storage Location
File Attributes
Storage Location
File Attributes + Data
CATALOG
SQL DBMS
PHYSICAL
MEDIA

Варианты конфигураций





Компоненты

[root@otuslinux ~]# yum install bacula-{client, console, director, storage}

- Bacula Director
- Bacula File Daemon
- Bacula Storage Daemon
- Bacula console

Общение между клиентами требует минимальной аутентификации и используются пароли. Для разных сущностей могут быть разные пароли.

Bacula Director

Это центральный и самый важный компонент системы резервного копирования. Он работает с каталогом (Catalog) и осуществляет все планирование и постановку задач остальным компонентам системы РК.

Поскольку остальные компоненты не наделены “разумом”, то вся конфигурация сосредоточена в director’e - и конфигурация клиентов, и конфигурация хранения, и прочих компонентов.

Bacula Director: сущности

Сущности:

- FileSet - набор данных для резервного копирования (Что копировать)
- Schedule - расписание резервного (Когда копировать)
- Client - сервер который надо “подвергнуть” резервному копированию (Откуда)
- Pool - Описание того как и где хранить резервные копии (Куда)
- Job, JobDefs - описания задачи на резервное копирование - связующее звено Fileset, Schedule, Client, Pool. Описывает что, когда, откуда и куда копировать
- Storage - Описание хранилища для бэкапа (где)
- Catalog - хранилище данных о бэкапах - “сердце” системы

FileSet

Описание набора файлов для резервного копирования.
Основные моменты конфигурации:

- Name
- Include
  - Options: compression, signature и т.д.
  - file-list
- Exclude
  - file-list

file-list:

- File = /path/to/dir
- @/path/to/filedirlist (читается на машине директора при чтении конфига - один раз)
- File = “|/path/to/script_to_generate_filelist”
- File = “| sh -c ‘command | to --generate=filelist’” (при запуске задачи)

Для того, чтобы жилось “легче”, можно добавить опцию LabelFormat в конфигурацию Pool, чтобы была возможность автоматически создавать тома.

Пример конфига:

```
# File Pool definition
Pool {
  Name = File
  Pool Type = Backup
  Recycle = yes                       # Bacula can automatically recycle Volumes
  AutoPrune = yes                     # Prune expired volumes
  Volume Retention = 365 days         # one year
  Maximum Volume Bytes = 50G          # Limit Volume size to something reasonable
  Maximum Volumes = 100               # Limit number of Volumes in Pool
  LabelFormat = "BackupVol"
}
```

Конфигурация плана резервного копирования

```
Schedule {
  Name = "WeeklyCycle"
  Run = Full 1st sun at 20:25
  Run = Differential 2nd-5th sun at 20:25
  Run = Incremental mon-sat at 20:25
}
```

Настройка общения со Storage Daemon:

```
Storage {
  Name = File
  Address = localhost   # Адрес storage daemon’a
  SDPort = 9103       # порт
  Password = "***password***" # пароль для аутентификации
  Device = FileStorage # какой из девайсов которыми манипулирует Storage
Daemon - использовать
  Media Type = File     # Тип стораджа
}
```

Конфигурация Задания РК (Job) может быть упрощена за счет использования различных JobDefaults, от которых наследуется Job, в контексте которого можно переопределить какие-либо параметры заданные в JobDefaults или задать новые.

Перед конфигурацией директора важно понять какую БД вы будете использовать и выполнить “нехитрую” махинацию:

```
alternatives --config libbaccats.so
```

выбрав нужную библиотеку - sqlite/mysql/postgresql.

Далее, в bacula-dir в контексте Director - задать порты, пароль, директорию и т. д.

Конфигурация Storage daemon

Для конфигурации storage daemon необходимо также указать явки-пароли, только на этот раз Director’а в файле /etc/bacula/bacula-sd.conf и сконфигурировать “Device” для хранения томов:

```
Director {
  Name = bacula-dir
  Password = "***password***"
}

Device {
  Name = FileStorage
  Media Type = File
  Archive Device = /var/backup/bacula
  LabelMedia = yes;                   # lets Bacula label unlabeled media
  Random Access = Yes;
  AutomaticMount = yes;               # when device opened, read it
  RemovableMedia = no;
  AlwaysOpen = no;
}
```

Конфигурация клиента (file daemon)

Конфигурация File Daemon’а ограничивается прописыванием доступов в контекстах Director:

```
Director {
  Name = bacula-dir
  Password = "***password***"
}
```

Bacula: минусы

- Нужно генерировать конфиги и клиенту, и директору
- Сложность этих самых конфигов
- На каталогах с большим количеством данных может начаться медленная утечка памяти
- На бэкапах с большим количеством файлов Bacula и Bareos очень сильно зависят от производительности используемой СУБД.
- И вообще зависимость от СУБД

Borg

- Дедупликация: исключение дублирующих копий повторяющихся данных. . Файлы в рамках одного Borg repository (т.е. специальном каталоге в специфичном для Borg формате) делятся на блоки по N мегабайт, а повторяющиеся блоки Borg дедуплицирует.
- Сжатие: после дедупликации данные сжимаются.
- Работает через SSH: отсюда простота и безопасность. На обеих сторонах только и нужно, что поставить Borg
- Прост в установке: PPA, Epel-Release, бинарники
- Гибкая очистка от старых бэкапов

Пример скрипта

```
#!/bin/bash
# Client and server name
CLIENT=root
SERVER=backup

# Backup type, it may be data, system, mysql, binlogs, etc.
TYPEOFBACKUP=mysql

REPOSITORY=$CLIENT@$SERVER:backup/$(hostname)-${TYPEOFBACKUP}
# Before backup
xtrabackup --backup --datadir=/var/lib/mysql/ --target-dir=/data/backups/mysql/

# Backup
borg create -v --stats                  \
  $REPOSITORY::'{now:%Y-%m-%d-%H-%M}'   \
  /data/backups/mysql/

# After backup
borg prune -v --show-rc --list $REPOSITORY \
  --keep-daily=7 --keep-weekly=4 --keep-monthly=6
```

Настроить стенд Vagrant с двумя виртуальными машинами server и client.

Настроить политику бэкапа директории /etc с клиента:

1) Полный бэкап - раз в день

2) Инкрементальный - каждые 10 минут

3) Дифференциальный - каждые 30 минут

Запустить систему на два часа. Для сдачи ДЗ приложить list jobs, list files jobid=<id>

и сами конфиги bacula-*

* Настроить доп. Опции - сжатие, шифрование, дедупликация

Ваши вопросы?

Заполните, пожалуйста, опрос в ЛК о занятии

Спасибо
за внимание!
До встречи в Slack и на вебинаре