**Лабораторная работа №6**

**Резервное копирование**

## Теоретическая часть

**Резервное копирование** (англ. *backup copy*) — процесс создания копии данных на носителе (жёстком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения. Резервное копирование может предполагать копирование системы (установленных пакетов, конфигурационных файлов) или отдельных данных, хранимых в файловой системе или в СУБД.

Перечислим основные требования к системе резервного копирования:

1. *Хранение на отдельном носителе или в другом месте*.
2. *Надежность места хранения*
3. *Доступность места хранения*
4. *Простота использования*

Коротко поясним указанные пункты. Очевидно, что если файлы с резервными копиями будут храниться на том же сервере, что и данные, то выход из строя всей системы приведет к потере как основных данных, так и резервной копии. Однако, хранение данных очень далеко от основного сервера, например, в другой стране, может привести к недоступности данных в определенное время. Простота использования также важна при работе с небольшими проектами, когда всё администрирование ведется только малой группой людей.

Перед выбором системы резервного копирования необходимо ответить на следующие вопросы:

1. От каких сбоев предполагается проводить защиту?

В зависимости от типов сбоев будут происходить разные подходы к резервированию. Если мы хотим защитить небольшое количество данных от физической поломки сервера, то правильным решением будет создание бэкапа на другой сервер, физически расположенный рядом с ним. Однако, такой подход не поможет, например, при возникновении пожара.

1. Какие данные необходимо копировать?

Очевидно, что необходимо выделить те данные, которые необходимо сохранять в резервной копии и оставить те, чем можно пожертвовать. Например, правильным решением будет развертывание всех установленных утилит с помощью системы управления конфигурациями, например Ansible. Таким образом сохранять данные о каждой программе не потребуется. Системные логи полезно сохранять с определенной периодичностью (если это требуется) на отдельные, медленные носители. Однако, их наличие не повлияет на восстановление системы, поэтому сохранять их в резервную копию также необязательно.

1. Как быстро нужно восстановиться?

Для определения времени для восстановления служит метрика RTO. Recovery Time Objective (показатель времени восстановления) - определяет количество времени с момента наступления разрушительного события до момента, когда затронутые ресурсы будут полностью работоспособны. Чем меньше значение RTO, тем выше затраты на восстановление. Показатель является требованием к системе, например, «время восстановление должно занимать не больше 5 часов».

1. Где должна находиться точка восстановления?

Для определения используется показатель RPO. Recovery Point Objective (показатель точки восстановления) – определяет максимально допустимый промежуток времени, за который данные могут быть восстановлены. Другими словами, это время с момента последнего резервного копирования данных до возникновения инцидента. Например, «Данные необходимо восстановить на состояние не раньше, чем за 5 часов до сбоя».

Очевидно, что чем меньше значения RPO и RTO, тем сложнее процесс восстановления и дороже организация резервного копирования.

1. Где хранить резервные копии?

В качестве оборудования для хранения резервных копий могут выступать как обычные жесткие диски собранные в RAID массив, так и магнитные ленты. Ленты дешевле публичных облаков, надежнее дисковых накопителей, более энергоэффективные, проще в обслуживании, обладают высокой емкостью, обеспечивают "холодное" хранение данных, предоставляют средства защиты от кибератак и более долговечны, чем жесткие диски и SSD-накопители. Однако, такой подход удобно использовать при хранении больших объемов данных – непосредственно оборудование для работы с лентами дорогое и неудобно для использования в небольших проектах. Ленточные накопители удобны при долгосрочном хранении данных, которые могут потребоваться через много лет.

**Уровни резервного копирования:**

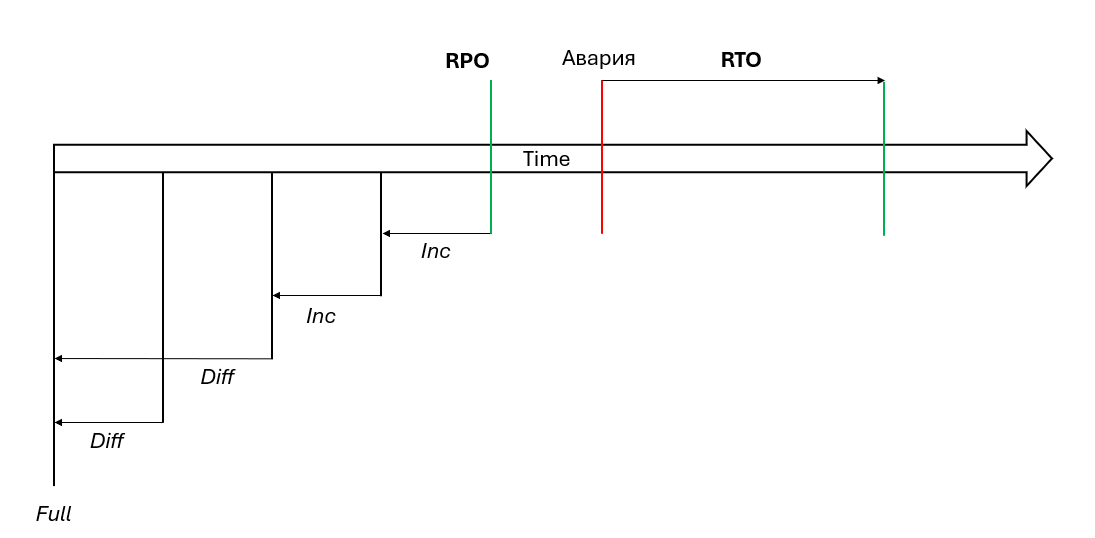
**Full - Полное копирование** подразумевает копирование всех требуемых файлов вне зависимости от того, были ли произведены их изменения с момента создания. Еженедельное, ежемесячное и ежеквартальное резервное копирование подразумевает создание полной копии всех данных. Обычно оно выполняется тогда, когда копирование большого объёма данных не влияет на работу организации. Для восстановления требуется только резервная копия.

**Differential** - При **дифференциальном** («разностном») резервном копировании каждый файл, который был изменён с момента последнего полного резервного копирования, копируется каждый раз заново. Дифференциальное копирование ускоряет процесс восстановления. Все копии файлов делаются в определённые моменты времени, что, например, важно при заражении вирусами. Для восстановления требуется последняя полная и последняя дифференциальная копии.

**Incremental** - При добавочном («**инкрементном**») резервном копировании происходит копирование только тех файлов, которые были изменены с тех пор, как в последний раз выполнялось полное или добавочное резервное копирование. Последующее инкрементное резервное копирование добавляет только файлы, которые были изменены с момента предыдущего. Инкрементное резервное копирование занимает меньше времени, так как копируется меньшее количество файлов. Однако процесс восстановления данных занимает больше времени, так как должны быть восстановлены данные последнего полного резервного копирования, а также данные всех последующих инкрементных резервных копирований.

На практике полную резервную копию делают один раз в определенный длительный промежуток времени, например, один раз в неделю. Далее в течение времени до следующего полного копирования производится несколько дифференциальных и множество инкрементальных копий.

На схеме представлен пример всех трех уровней резервного копирования. Полная копия была снята всего один раз, далее было произведено два дифференциальных резервных копирования и два инкрементальных. В момент времени, отмеченный красной линией произошла авария. Согласно требованиям к системе, время до её восстановления составит не более, чем RTO, а восстановиться получится на время, равное «времени аварии - RPO».



Отдельно от указанных уровней находится **клонирование,** которое позволяет скопировать целый раздел или носитель (устройство) со всеми файлами и каталогами в другой раздел или на другой носитель. Если раздел является загрузочным, то клонированный раздел тоже будет загрузочным.

**Резервное копирование баз данных**

Одним из важной области применения резервного копирования является резервное копирование баз данных. Т.к. СУБД являются сложными высоконагруженными системами, то необходимы специальные алгоритмы для обеспечения корректного копирования данных. Далее будут приведены примеры резервного копирования для СУБД Postgres, однако похожие алгоритмы существуют и на других системах управлением баз данных.

Для ускорения работы СУБД использует буферных кэш, хранящийся в оперативной памяти. При нештатной ситуации оперативная память может очиститься и все данные из неё и, в частности, кэша будут потеряны. Для их восстановления используется журнал предзаписи (WAL, write-ahead log). Одновременно с изменениями данных в странице кэша происходит запись в журнале. Она содержит информацию, достаточную для того, чтобы повторить указанную операцию. Такая запись обязательно сохраняется на физическом носителе до того, как на него будет записана эта информация.

Существуют несколько способов резервного копирования баз данных:

1. Логическое резервирование

Логическое резервирование (не путать с логической репликацией!) заключается в выгрузке данных из СУБД на носитель в виде текстового или бинарного файла. Является самым простым способом резервного копирования, однако его основным недостатком является длительное время выполнения и высокая нагрузка на сервер. Также при выгрузке сохраняется только логическая структура данных – физические объекты, например индексы, не сохраняются. Они восстанавливаются после выгрузки бэкапа и этот процесс может занимать длительное время. Способ удобен для длительного хранения данных или миграции на другую платформу.

1. Физическое резервирование

Физическое резервирование подразумевает копирование всех файлов, относящихся к кластеру БД, то есть создание полной двоичной копии. Основным достоинством подхода является высокая скорость работы, по сравнению с созданием и восстановлением логической копией. Однако, восстановление возможно произвести только на систему с совместимыми характеристиками и невозможность выполнения копирования части базы данных.

Физические копирование возможно произвести двумя способами: «холодным» и «горячем».

* Холодное резервирование

Предполагает остановку базы данных и создание её копии. Очевидно, что остановка работы является существенным недостатком способа. Холодная копия должна хранить внутри себя кроме данных значения журналов, однако для восстановления они не потребуются (если сервер был выключен штатно).

* Горячее резервирование

Не предполагает остановки базы данных при создании её копии. Очевидной сложностью является то, что в процессе резервирования параллельно данные будут изменяться. Поэтому копирование происходит минимум по двум каналам – по одному передаются сами данные, а по второму – генерируемые с момента старта резервирования журнальные записи.

**Репликация**

Понятие резервного копирования часто связывают с понятием репликации. Однако следует понимать, что это совершенно другая операция.

Репликация – процесс поддержания двух (или более) наборов данных в согласованном состоянии. Другими словами, существует эталонный набор, каждое изменение которого влечет за собой изменение реплик. Репликация является одной из функций СУБД, когда для базы данных возможно создать её точную реплику, которая позволит восстановить исходную базу данных в случае её потери (отказоустойчивость) и снизит нагрузку за счет переноса части запросов на реплику (масштабируемость).

Существуют следующие подходы к репликации:

*Блочная репликация* – каждая операция выполняется одновременно на основном носителе и на резервном. Реализуется на уровне системы хранения данных.

*Физическая репликация* – все изменения на реплике происходят на основе журналов баз данных.

*Логическая репликация* – все изменения происходят за счет повторения на реплике всех выполненных запросов к БД.

Однако, такой подход не поможет восстановить данные, в случае появления логической ошибки. Например, случайно удалив вручную важную таблицу, эта операция повторится и на реплике. Аналогично, при повреждении внутренних структур баз данных ошибка может перенестись на реплику.

Для решения перечисленных выше проблем применяется именно резервное копирование баз данных.

**Bacula**

Bacula – это система резервного копирования корпоративного уровня. Она имеет клиент-серверную архитектуру и состоит из следующих компонентов:

* Bacula Director (сервис bacula-dir) – основной сервис, который управляет всеми другими процессами по резервному копированию и восстановлению;
* Bacula Storage (сервис bacula-sd) – хранилище, предназначенное для сохранения резервных копий на диске;
* Bacula File Daemon (сервис bacula-fd) – клиентская часть сервиса, которая нужна для доступа к файлам на сервере, с которого будет выполняться резервное копирование.

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Все компоненты могут быть установлены как на одном сервере, так и на разных, но каждый из них должен иметь возможность обратится к другому по сети. Для управления всем этим используется утилита командной строки bconsole. Для неё существует как консольный, так и веб-интерфейс, но основной способ управления - командная строка.

**Создание таблицы базы данных для Bacula**

Bacula поддерживает большинство СУБД (mysql, postgresql)

Подключение к базе данных необходимо для корректной работы сервиса. При запуске сервиса bacula-director подключается к базе данных (сведения о имени пользователя пароле и адресе таблицы находится в графе Catalog в конфигурационном файле). В ней хранятся сведения обо всех зарезервированных файлах и их местонахождении в резервных копиях. Каталог необходим для обеспечения эффективной адресации к требуемым файлам.

**Важно** в примере использовалось две виртуальные машины. На первой с ip 192.168.122.3 разветывались Director и Storage, на второй c ip 192.168.122.2 – file daemon.

Для доступа к базе данных необходимо:

**На Director Daemon**

В файле /etc/postgresql/\*/main/postgresql.conf указать параметр:

listen\_addresses='\*'

В файле /etc/postgresql/\*/main/pg\_hba.conf указать метод trust для всех.  
Обязательно добавить host с ip адресом, где будет работать bacula-dir.

local all postgres trust

local all all trust

host all all 127.0.0.1/32 trust

host all all 192.168.122.3/24 trust

Выполнить запуск БД:

pg\_ctlcluster 11 main restart

Присвоить пароль пользователю postgres:

passwd postgres

Присвоить пароль пользователю bacula:

passwd bacula

Создать пользователя БД для работы с bacula, выполнять не из-под root:

#psql template1 -U postgres -h 192.168.122.3 -p PORT

template1=# CREATE ROLE bacula;  
template1=# ALTER USER bacula PASSWORD 'bacula';  
template1=# ALTER USER bacula LOGIN SUPERUSER CREATEDB CREATEROLE;

Создать БД bacula

CREATE DATABASE bacula;  
ALTER DATABASE bacula OWNER TO bacula;

Для корректного функционирования авторизации черезе PAM, пользователю postgres необходимо выдать права на чтение информации из БД пользователей и сведений о метках безопасности и привилегиях:

usermod -a -G shadow postgres  
setfacl -d -m u:postgres:r /etc/parsec/macdb  
setfacl -R -m u:postgres:r /etc/parsec/macdb  
setfacl -m u:postgres:rx /etc/parsec/macdb

setfacl -d -m u:postgres:r /etc/parsec/capdb  
setfacl -R -m u:postgres:r /etc/parsec/capdb  
setfacl -m u:postgres:rx /etc/parsec/capdb

Пользователю bacula задаем минимальный и максимальный уровень:

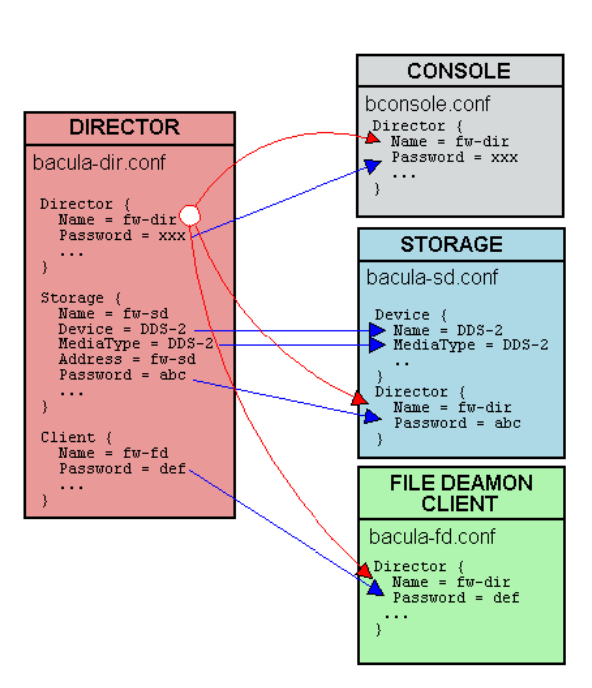
pdpl-user bacula -l 0:0

В скриптах /usr/share/bacula-director/make\_postgresql\_tables и /usr/share/bacula-director/grant\_postgresql\_privileges вносим изменения:

db\_name=${db\_name:-bacula}  
psql -U bacula -h 192.168.122.3 -p 5433 -f - -d ${db\_name} $\* <<END-OF-DATA

Сохраняем изменения и запускаем скрипты make\_postgresql\_tables и grant\_postgresql\_privileges

**Установка и настройка bacula-director**

****

Установка:

sudo apt install bacula

Настройка производится в файле /etc/bacula/bacula-dir

Director {

Name = bacula-dir

DIRport = 9101

QueryFile = "/etc/bacula/scripts/query.sql"

WorkingDirectory = "/var/lib/bacula"

PidDirectory = "/run/bacula"

Maximum Concurrent Jobs = 1

Password = "1" # Console password

Messages = Daemon

DirAddress = 192.168.122.3

}

Внимание стоит обратить на поля:

* Password – пароль для доступа из File Daemon и Storage Daemon
* DirAddress – ip адрес машины
* Messages – выбор настроек вывода сообщений

Для настройки заданий используются секции Job (для каждой задачи отдельная секция). Примеры настроек копирования и восстановления

Job {

Name = "BackupClient1"

Type = Backup

Client = bacula-fd

FileSet="Catalog"

Schedule = "DailyCycle"

Messages = Standard

Pool = Default

Write Bootstrap = "/var/lib/bacula/%n.bsr"

Priority = 1

}

Job {

Name = "RestoreFiles"

Type = Restore

Client=bacula-fd

FileSet="Catalog"

Storage = File

Pool = Default

Messages = Standard

Where = /home2

}

* Type указывает на тип задания.
* Schedule указывает на предопределённое расписание, согласно которому будет выполняться резервирование данных. Все расписания определены здесь же, в файле
* Where указывает на каталог, где будет развернут бэкап.
* Write Bootstrap указывает путь к файлу, в который будет записываться информация, с помощью которой данные могут быть восстановлены из резервной копии без наличия подключения к Bacula Catalog. Вместо %n будет подставлено значение параметра Name:
* FileSet определяет какие каталоги будут скопированы с клиента

FileSet {

Name = "Catalog"

Include {

Options {

signature = MD5

compression = GZIP

aclsupport = yes

xattrsupport = yes

}

File = /home

}

* Name определяет уникальное имя набора.
* Include содержит пути к резервируемым файлам/каталогам, а Exclude – пути к файлам и каталогам, которые необходимо исключить из списка резервируемых.
* Signature указывает алгоритм вычисления контрольных сумм файлов.
* Compression указывает алгоритм компрессии файлов.
* Recurse указывает на необходимость рекурсивного резервирования, включая подкаталоги и файлы.
* File указывает на каталог, который мы копируем.
* Xattrsupport указывает на возможность включения поддержки расширенных атрибутов, это обязательный параметр для работы с метками безопасности.

Для настройки агентов используются секции Client

Client {

Name = bacula-fd

Address = 192.168.122.2

FDPort = 9102

Catalog = MyCatalog

Password = "1"

File Retention = 30 days

Job Retention = 6 months

AutoPrune = yes

}

Storage {

Name = File

Address = 10.0.0.24

SDPort = 9103

Password = "1"

Device = FileStorage

Media Type = File

}

* Name – уникальное имя, использующееся для адресации секции Storage в рамках файла bacula-dir.conf
* Device и MediaType дублируют одноимённые параметры файла bacula-sd.conf.
* Password содержит пароль, который будет использоваться при подключении к Storage Daemon.

Catalog {

Name = MyCatalog

dbname = "bacula"; DB Address = "10.0.0.23"; dbuser = "bacula"; dbpassword = "bacula

}

Здесь необходимо указать реквизиты доступа к базе данных, а также назначить уникальное имя данного Bacula Catalog с помощью параметра Name.

Секция Pool определяет набор носителей информации и параметры, определяющие то, как SD будет их обрабатывать. Каждый Pool взаимодействует с устройством хранения данных, и поэтому необходимо создать столько же пулов, сколько определено устройств хранения. Фактически если для каждого File Daemon вы определяете отдельное устройство, то для каждого FD необходимо определить и Pool. Рассмотрим перечень наиболее значимых параметров.

* Name определяет уникальное имя пула.  
  Type определяет тип, и для резервных копий должен быть установлен в значение Backup.
* Maximum Volume Jobs рекомендуется установить в значение 1. Это будет означать, что в рамках одного носителя данных могут быть размещены резервные данные, полученные в ходе выполнения только одного задания.
* Носитель данных – это устройство, на которое непосредственно записываются данные (оптические диски, магнитные ленты).
* Volume Retention – время, по прошествии которого данные о резервной копии, хранящейся на носителе, будут удалены из каталога.
* Maximum Volumes – максимальное количество носителей (в нашем случае файлов), доступных в данном пуле.
* Recycle указывает на необходимость повторного использования носителей, помеченных как устаревшие. При этом реальная перезапись носителя произойдёт лишь в случае, когда свободных носителей не останется. Свободные носители определяются из параметра Maximum Volumes.
* AutoPrune указывает на то, необходимо ли производить удаление устаревших записей из Bacula Catalog автоматически после завершения выполнения очередного задания.
* Label Format определяет префикс, который будет использован Bacula для маркирования носителей информации, в нашем случае – для именования файлов.
* Storage указывает на имя устройства хранения данных, указанного в параметре Name секции Storage файла bacula-dir.conf.

Настройки доступа к консоли

Console {

Name = bacula-mon

Password = "1"

CommandACL = status, .status

}

Далее необходимо настроить доступ к DD со стороны Bacula Console в файле /etc/bacula/bconsole.conf сервера Director Daemon :

Director {

Name = bacula-dir

DIRport = 9101

address = 192.168.122.3

Password = "1"

}

Для проверки правильности sudo /usr/sbin/bacula-dir -t -c /etc/bacula/bacula-dir.conf

Перезапуск: sudo systemctl restart bacula-dir

**Настройка bacula-fd**

На клиенте

* Настройка производится в файле /etc/bacula/bacula-dir

Director {

Name = bacula-dir

Password = "1"

}

Указываются пароль и название director для доступа к сервису

FileDaemon {

Name = bacula-fd

FDport = 9102

WorkingDirectory = /var/lib/bacula

Pid Directory = /var/run/bacula

Maximum Concurrent Jobs = 20

FDAddress = 192.168.122.2

}

* Адрес – ip клиентской машины

Для проверки

sudo /usr/sbin/bacula-fd -t -c /etc/bacula/bacula-fd.conf

## Практическая работа

1. Установить программу Bacula
2. Настроить пользователя bacula, создать таблицу в доступной СУБД и выдать привилегии на работу с ней пользователю
3. Настроить Director daemon на одном устройстве и file daemon на втором
4. Настроить доступ к консоли (это производится в bconsole.conf на каждом устройстве)
5. Настроить резервное копирование вашей домашней директории с клиента на сервер