Think Your Postgres Backups And Disaster Recovery Are Safe? Let's talk.

Payal Singh
Database Administrator
OmniTI

Who am I ?

DBA@OmniTI

Github: payals

Blog: http://penningpence.blogspot.com

Twitter: @pallureshu

Email: payal@omniti.com

Agenda

Types of backups

Validation

Backups management

Automation

Filesystem

Third party tools

In-house solution

Types of Backups

Business Continuity Objectives

Recovery Point Objective (RPO)

Recovery Time Objective (RTO)

Types of Backups

- **Logical**
  - pg_dump
  - pg_dumpall
- **Physical (File-system Level)**
  - **Online**
    - Most commonly used
    - No downtime
  - **Offline**
    - rarely used
    - shutdown database for backup

Logical Backups

Advantages:

- Granularity
- Fine-tuned restores
- Multiple in-built compression types
- Ease of use, no extra setup required

Disadvantages:

- Relatively slower
- Frozen snapshots in time, i.e. no PITR
- Locks
- Data spread in time

pg_dumpall

Pros:

- Globals

Cons:

- Requires superuser privileges
- No granularity while restoring
- Plaintext only
- Cannot take advantage of faster/parallel restore with pg_restore

Physical Backups

Advantages:

- Faster
- Incremental Backups
- Point In Time Recovery
- By default compression on certain file-systems

Disadvantages:

- Lacks granularity

Why do you need both?

The more the merrier:

Database wide restores - Filesystem restore is faster!

Someone dropped a table? - Dump backups are faster!

pg_basebackup

Advantage:

- In core postgres
- No explicit backup mode required
- Multiple instances can run concurrently
- Backups can be made on master as well as standby

Disadvantage:

- Slower when compared to snapshot file-system technologies
- Backups are always of the entire cluster

pg_basebackup Requirements

- A superuser or a user with REPLICATION permissions can be used
- archive_command parameter should be set to true
- max_wal_senders should be at least 1

Delayed Replicas as Backups

- In situations where accidental damage was done that was realized in a short period of time.
- May still result in some data loss post recovery, but it depends on the importance of table, other objects dependent on it, etc.

Version Controlled DDL

- Committing daily changes to schema design, functions, triggers etc. into source control
- Can go back to any state
- pg_extractor (https://github.com/omniti-labs/pg_extractor) facilitates this by creating files for each database object, followed by integration with git, svn, etc.

Validation

Continuous Restores

Validation is important

Estimates / Expectations

Procedure / External factors

} RTO

Does it even work?

Development Database

Routine refresh + restore testing

Reporting Databases:

Overnight restores for reporting databases refreshed daily. Great candidate for daily validation.

Sample Validation Log

2015-03-01 10:00:03 : Testing backup for 2015-02-28 from /data/backup/hot_backup/+/var/log/test/bin/test_backups.sql
2015-03-01 10:00:03 : Starting decompress of /data/backup/hot_backup/test.local-data-2015-02-28.tar.gz
2015-03-01 10:00:03 : Starting decompress of /data/backup/hot_backup/test.local-xlog-2015-02-28.tar.gz
2015-03-01 10:00:03 : Waiting for both decompressing processes to finish
2015-03-01 14:36:06 : Decompressing worked, generated directory test with size: 963G
2015-03-01 14:36:06 : Starting PostgreSQL
server starting
2015-03-01 14:36:07.372 EST @ 3282 LOG: loaded library "pg_scoreboard.so"
2015-03-01 15:52:36 : PostgreSQL started
2015-03-01 15:52:36 : Validating Database
starting Vacuum
2015-03-02 08:17:56 : Test result: OK
2015-03-02 08:18:11 : All OK.

Where did Gitlab go wrong?

No PITR, no alerts for missing backups, no documentation for backup location, no restore testing

> *"…backups…taken once per 24 hours…not yet been able to figure out where they are stored…don't appear to be working..."*

No retention policy

> *"Fog gem may have cleaned out older backups"*

No monitoring

> "*Our backups to S3 apparently don't work either: the bucket is empty*"

No RPO

> "…Unless we can pull these from the past 24 hours they will be lost"

Sobering Thought of the Day

A chain is only as strong as the weakest link

Backup Management

Retention Period

Remove all older than x days

VS

Remove all but latest x backups

Off-server (short term)

VS

Off-site retention period (long term)

Security

Transfer

- One way passwordless SSH access
- HTTPS for cloud uploads (e.g. s3cmd)

Storage

- Encryption
- Access control

PCI Compliance

- Private keys
- Logical backups preferred

Multi-Tenancy Environment

- Backup Server ⊟ Client
- Client ⊟ Backup server

Monitoring and Alerts

Alert at runtime

- To detect errors at runtime within script, or change in system/user environments
- Immediate
- Cron emails

Alert on storage/retention:

- Error in retention logic/implementation
- Secondary/Delayed
- Cron’d script, external monitoring

Alert on backup validation:

- On routine refreshes/restores

S3 Backups

Ideal for long term backups:

- Cheap
- Ample space
- Transfer in parts, can pick up where you left off in case of errors
- Speed depends on bandwidth

Secure:

- Buckets , Users, Policies

Communication:

- AWS cli / s3cmd
- Access keys
- PGP encryption

S3 Backups Contd.

Sample Bucket Policy:

```
{
    "Version": "2012-10-17",
    "Id": "S3-Account-Permissions",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<account_number>:user/omniti_testing"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::omniti_testing",
                "arn:aws:s3:::omniti_testing/*"
            ]
        }
    ]
}
```

S3 Backups Contd.

Sample User Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::omniti_testing",
        "arn:aws:s3:::omniti_testing/*"
      ]
    }
  ]
}
```

Automation

Why Automate?

Reduced Chance of Human Error

- pg_dump backups
- filesystem level backups
- retention scripts
- backup monitoring and alerts
- off-site and off-server transfer as well as storage
- access channels
- security keys
- cronjobs…

Less work, faster

- Move scripts, swap crontabs, ensure all accesses exist

Uniformity, Reliability

What to Automate?

Scripts

Crontabs

Accesses

Validation/restores

Automation Example In Chef

payal@payal-ThinkPad-T520$ ls -l s3_backup/
drwxr-xr-x 2 payal payal 4096 Dec 2 11:03 attributes
drwxr-xr-x 3 payal payal 4096 Dec 2 11:03 files
-rw-r--r-- 1 payal payal 1351 Dec 2 11:03 README.md
drwxr-xr-x 2 payal payal 4096 Dec 2 11:03 recipes
drwxr-xr-x 3 payal payal 4096 Dec 2 11:03 templates

payal@payal-ThinkPad-T520 $ ls -l s3_backup/templates/default/
-rwxr-xr-x 1 payal payal 3950 Dec 2 11:03 backup.sh.erb
-rw-r--r-- 1 payal payal 1001 Jan 12 12:40 check_offsite_backups.sh.erb
-rw-r--r-- 1 payal payal   37 Dec 2 11:03 pg_dump_backup.conf.erb
-rw-r--r-- 1 payal payal   19 Dec 2 11:03 pg_dump.conf.erb
-rw-r--r-- 1 payal payal 6319 Dec 2 11:03 pg_dump_upload.py.erb
-rw-r--r-- 1 payal payal 1442 Dec 2 11:03 pg_start_backup.sh.erb
-rw-r--r-- 1 payal payal 1631 Dec 2 11:03 s3cfg.erb

Automation Example in Ansible

```
- name: "Use pg_dump for logical backup"
shell: "pg_dump -U postgres -Fc -f
{{ db_name }}_{{ today }}.bak {{ db_name }} >
backup.log 2>&1"
args:
chdir: "{{ backup_dir }}"
when: not only_upload_backup

- name: "upload backup file"
s3: region="ap-southeast-1" bucket=“sample"
object="/DB_bucket/
{{ db_name }}_{{ today }}.bak"
src="{{ backup_dir }}/
{{ db_name }}_{{ today }}.bak" mode=put
```

https://github.com/payals/postgresql_automation/tree/master/backups

Filesystem

Can Filesystems enhance DR?

ZFS

- Available on Solaris, Illumos. ZFS on Linux is new but improving!
- Built in compression
- Protections against data corruption
- Snapshots
- Copy-on-Write

How can ZFS help you with DR?

Scenario 1: You're using pg_upgrade for a major upgrade with hard link option (-k). It fails in between. Since you used hardlinks, you cannot turn on the older cluster. What do you do?

ZFS Rollback – **instantaneously** rollback a dataset to a previous state from a snapshot

sudo zfs rollback <snapshot_name>

How can ZFS help you with DR?

Scenario 2: You've accidentally deleted a large table in a very large database where taking full backups are infeasible. Is there a faster alternative to a filesystem restore? ZFS rollback will overwrite pgdata so you cannot use that.

ZFS Clone – Copy-on-Write. **Instantaneously** clone a dataset from snapshot without any additional space consumptions as long as you don't write to it.

sudo zfs clone <snapshot_name> <mountpoint>

External Tools

OmniPITR

PITR backups and archives

Backups off of master or standby

Built in backup integrity checks

Quick setup, Minimal requirements

Comprehensive logging

Built in retention

Ideal for infrastructures of all sizes

Barman

PITR backups and restores

Automatic retention, continuous recovery

Requires its own server

More suitable for larger infrastructures to manage multiple clusters and multiple locations

Simplicity
- Minimal knowledge of PITR inner workings required
- Wrappers for recovery process

Supports backups from both primary and standby

wal-e

PITR backups and restores

Automatic retention, continuous recovery

Minimal setup

Cloud integration:

AWS, Azure, Google, Swift

In House Solution

Logical

```
def take_dump():
    try:
        with open(config_file, 'r') as f:
            for db in f:
                if db.strip():
                    db = db.replace("\n", "")
                    dump_command = pg_dump_path + " -U postgres -v -Fc -f " +
  dump_file_path + db.split()[-1] + "_" + start_time  + ".sql" + " " + db + "
  2>> " + dump_file_path + db.split()[-1] + "_" + start_time  + ".log"
                os.system(dump_command)
                print('backup of ' + db.split()[-1] + ' completed successfully')

    except:
        print('ERROR: bash command did not execute properly')
```

Physical

pg_basebackup:

Pg_basebackup –D pgdata –F format --xlogdir –Xs –c fast -p …

ZFS snapshots:

ZFS restore from snapshot

ZFS rollback to snapshot after failed upgrade or maintenance task

```
...
read_params "$@“
if [[ -z ${OFFLINE} ]]
then
    postgres_start_backup
    zfs_snap
    postgres_stop_backup
else
    zfs_snap
fi
backup
zfs_clear_snaps
```

S3 Backups

Basic steps:

check_lock()

take_dump()

gpg_encrypt()

s3_upload()

move_files()

cleanup()

Redundancy is Good

“That Sunday, Thomas deleted remotely stored backups and turned off the automated backup system. He made some changes to VPN authentication that basically locked everybody out, and turned off the automatic restart. He deleted internal IT wiki pages, removed users from a mailing list, deactivated the company's pager notification system, and a number of other things that basically created a huge mess that the company spent the whole of Monday sorting out (it turned out there were local copies of the deleted backups).”

https://www.theregister.co.uk/2017/02/23/michael_thomas_appeals_conviction/

S3 Backups Monitoring

Sample:

```
# Get timestamp of latest file in S3 bucket
latest_backup=$(s3cmd ls s3://omniti_client/ | awk {'print $1'} | sort -n | tail -1)
today=$(date --date=today "+%Y-%m-%d")
yesterday=$(date --date=yesterday "+%Y-%m-%d")

# If latest backup is older than yesterday, email dba
if [ $yesterday -ne $latest_backup -a $today -ne $latest_backup ]
then
    echo "Offsite Backups Missing for client" | mailx -s "Offsite Backup Missing for Client " dba@omniti.com
```

Backup Documentation

- Detailed backup information - types, locations, retention periods
- Procedures to setup new machines
- Analysis and estimation time for recovery
- Ways to recover

References

http://www.postgresql.org/docs/9.4/static
https://github.com/omniti-labs/omnipitr
http://docs.pgbarman.org
http://docs.chef.io/
https://github.com/omniti-labs/pgtreats
https://github.com/omniti-labs/pg_extractor
http://www.druva.com/blog/understanding-rpo-and-rto
http://evol-monkey.blogspot.co.uk/2013/08/postgresql-backup-and-recovery.html

Questions?

Payal Singh
payal@omniti.com
OmniTI Computer Consulting Inc.