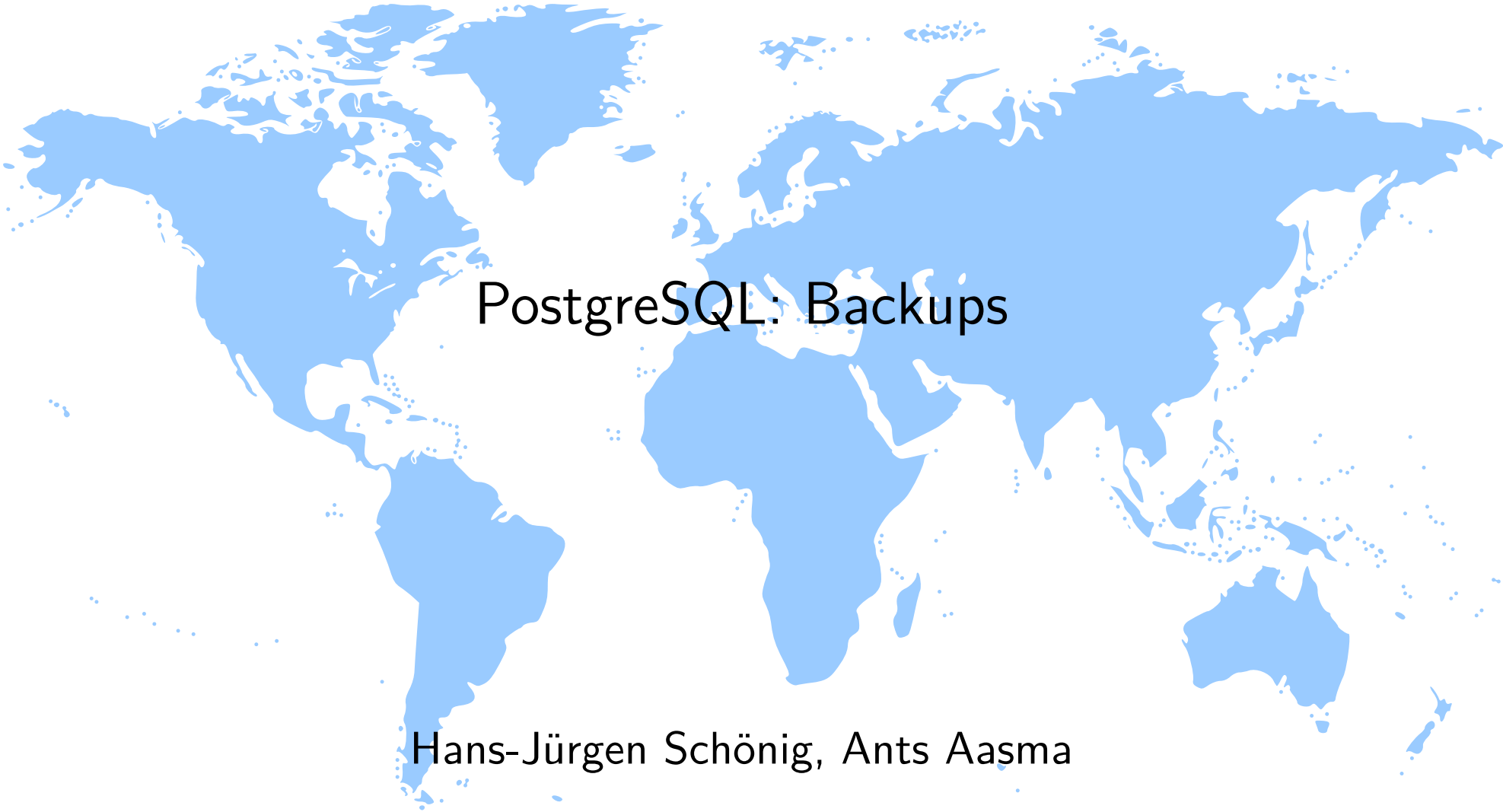

PostgreSQL: Backups

Hans-Jürgen Schönig, Ants Aasma

www.cybertec.at

Logical backups

pg_dump

- Dumps out the contents of the database in textual format
- Cross version backups and restores.
- Backups can be fixed up in case data corruption, etc.
- Slow
- Needs ACCESS SHARE lock on all tables.

pg_dump backup formats

- `-F` plain (default), custom, directory, tar
- Use custom or directory format
- Parallel dumps (`--jobs`) need directory format.
- Most options relevant to plain dump, specify them with pg_restore

pg_dumpall

- Only in SQL format
- Use `-g` or `--globals-only` together with pg_dump

pg_restore

- Converts custom dumps to SQL and inputs them to PostgreSQL

Physical backup

Cold backup

- Just copy the files

Hot backup

- `SELECT pg_start_backup('foobar', true)`
- Copy files.
- `SELECT pg_stop_backup()`
- Copy transaction logs.

How hot backup works

- Backup checkpoint
- backup_label
- WAL start and end

pg_basebackup

- Does the same thing as a hot backup
- Database contents streamed out over a connection.
- Needs a replication connection to be set up.
- Don't forget the WAL.

Snapshot based backup

- Simplest
- Consistent snapshot accross all tablespaces AND transaction log is required.

WAL archiving

- Needed for PITR
- Helpful for backups and replication
- Figure out your retention policy.

Backup from a slave

- Simple answer: use pg_basebackup
- Complex answer...

Schrödingers backup

- A backups state is a superposition of ok and not ok, until you try to restore it.

Point-In-Time-Recovery

What it does

- PITR can be used to reach (almost) any point after a base backup.
- Replication and PITR can be combined.

Configuring for PITR

- S: create an archive (ideally this is not on the master)
- M: Change postgresql.conf
  - set wal_level
  - set max_wal_senders (if pg_basebackup is desired)
  - set archive_mode to on
  - set a proper archive_command to archive xlog
- M: adapt pg_hba.conf (if pg_basebackup is desired)
- M: restart the master

pg_basebackup, etc.

- Perform a pg_basebackup as performed before
  - –xlog-method=stream and -R are not needed
- In the archive a .backup file will be available after pg_basebackup
- You can delete all xlog files older than the oldest base backup you want to keep.
- The .backup file will guide you

Restoring from a crash

- Take a base backup.
- Write a recovery.conf file:
  - restore_command: Tell PostgreSQL where to find xlog
  - recovery_target_time (optional): Use a timestamp to tell the system how far to recover
- Start the server
- Make sure the system has reached consistent state

More config options

recovery_min_apply_delay: Delayed replay

- This settings allows you to tell the slave that a certain delay is desired.
- Example: A stock broker might want to provide you with 15 minute old data

pause_at_recovery_target

- Make sure that the recovery does not stop at a specified point in time.
- Make PostgreSQL wait when a certain point is reached.
- This is essential in case you do not know precisely how far to recover

recovery_target_name

- Sometimes you want to recover to a certain point in time, which has been specified before.
- To specify a point in time run ...

```
SELECT pg_create_restore_point('some_name');
```

- Use this name in recovery.conf to recover to this very specific point

How to develop your own solution

Existing tools for backups

- Barman automates everything discussed
- OmniPITR for archiving
- WAL-E or pghoard to store backups/archive in S3/Swift
- pg-rman

Clustering tools

- Linux-HA/pacemaker/corosync
- patroni
- repmgr