BURNING DOWN THE HOUSE:
KEEPING YOUR POSTGRESQL DATA SAFE

YOU LOVE YOUR DATA

WHICH IS WHY YOU STORE
IT IN A DATABASE

BUT...

WHAT HAPPENS IF YOUR HARDWARE FAILS?

WHAT HAPPENS IF YOUR SOFTWARE FAILS?

“AT 9:45 EST A USER TRIGGERED AN UNSCOPED DELETION OF ALL HISTORICAL PERIOD RECORDS [...]”

DEAD MAN'S SNITCH OUTAGE POST MORTEM

NOT IF

BUT WHEN

CHOOSE A BACKUP STRATEGY

PAY ATTENTION TO...

HOW MUCH DATA CAN YOU AFFORD TO LOSE?

HOW MUCH DOWN TIME CAN YOU AFFORD?

WAYS TO BACK UP YOUR DATABASE

HAVE SOMEONE ELSE DO IT

✔ IT'S SOMEONE ELSE'S PROBLEM NOW

❌ LESS CONTROL

❌ CAN COST A LOT

❌ LESS FLEXIBILITY

FILE-BASED BACKUP METHODS

✔ EASY

✔ WIDELY AVAILABLE TOOLS

❌ REQUIRES POSTGRESQL SERVER TO BE SHUT DOWN

```
$ pg_ctl stop
$ rsync $PGDATA /path/to/backup
$ pg_ctl start
```

pg_dump[all]

✔ CREATES LOGICAL BACKUP

✔ FLEXIBLE

✔ NO DOWN TIME REQUIRED

✔ MINIMAL IMPACT

❌ IMPRACTICAL FOR LARGE DATABASES

```
$ pg_dump \
  --format=custom \
  --exclude-table-data=stats \
  --compress=9 \
  --jobs=4 \
  ${DB_NAME}.pg_dump
$ # Repeat for all Databases
```

```
$ pg_restore \
  --jobs=4 \
  --dbname=${DB_NAME} \
  /path/to/backup.pg_dump
$ # Get a cup of coffee
```

HOT STANDBY

✔ CONTINUOUS

✔ USE THEM TO DISTRIBUTE READ LOAD

✔ GOOD FOR FAST FAILOVER

❌ ONLY PROTECTS AGAINST HARDWARE FAILURE

❌ REQUIRES AT LEAST ONE MORE SERVER

```
# In postgresql.conf:
wal_level = hot_standby

# Set these to something > 0
max_wal_senders = 5
wal_keep_segments = 64
```

```
$ pg_basebackup \
  --host=localhost \
  --username=replication_role
  --format=plain \
  --xlog-method=stream \
  --pgdata=${STANDBY_PGDATA} \
  --progress
```

```
# postgresql.conf on standby:
hot_standby = on
hot_standby_feedback = on

max_standby_streaming_delay = 10s
```

```
standby_mode = 'on'
primary_conninfo = 'host=db-primary port=5432
user=replication_role sslmode=require'
trigger_file = '/usr/local/pgsql/data/
primary.trigger'
```

```
$ pg_ctl start

LOG:  entering standby mode

... then some time later ...

LOG:  consistent recovery state reached
LOG:  database system is ready to
accept read only connections
```

PITR-BASED BACKUPS

- ✔ CONTINUOUS BACKUP
- ✔ HIGH RECOVERABILITY
- ✔ CLUSTER-BASED

- ❌ COMPLEX
- ❌ INCREASED I/O
- ❌ NEEDS LOTS OF STORAGE
- ❌ ARCHITECTURE-DEPENDANT

```
archive_command = 'cp %p /path/to/archive/%f'
```

```
$ pg_basebackup \
  --host=localhost \
  --username=replication_role
  --format=plain \
  --pgdata=${BACKUP_PGDATA} \
  --progress
```

```
# Restore/untar latest base backup

# Create a recovery.conf:
restore_command = 'cp /path/to/archive/%f %p'

# Start PostgreSQL
```

BARMAN

OMNIPITR

3RD PARTY TOOLS

✔ REPLICATE BETWEEN POSTGRESQL VERSIONS

✔ VERY FLEXIBLE

✔ PAID SUPPORT

❌ COMPLICATED

❌ SETUP & MAINTENANCE

❌ COSTS MONEY

RECOMMENDATIONS

DAILY pg_dump

HOT STANDBY

STORE BACKUPS
SOMEWHERE ELSE

TEST YOUR BACKUPS

SERIOUSLY, TEST THEM

POSTGRESQL.ORG/DOCS/CURRENT/

BACK UP YOUR DATA
(NOT JUST YOUR DATABASE)

THX!

NUCLEARSQUID.COM / @NUCLEARSQUID / GITHUB.COM/CYPHER