TEST ENVIRONMENT ON DEMAND

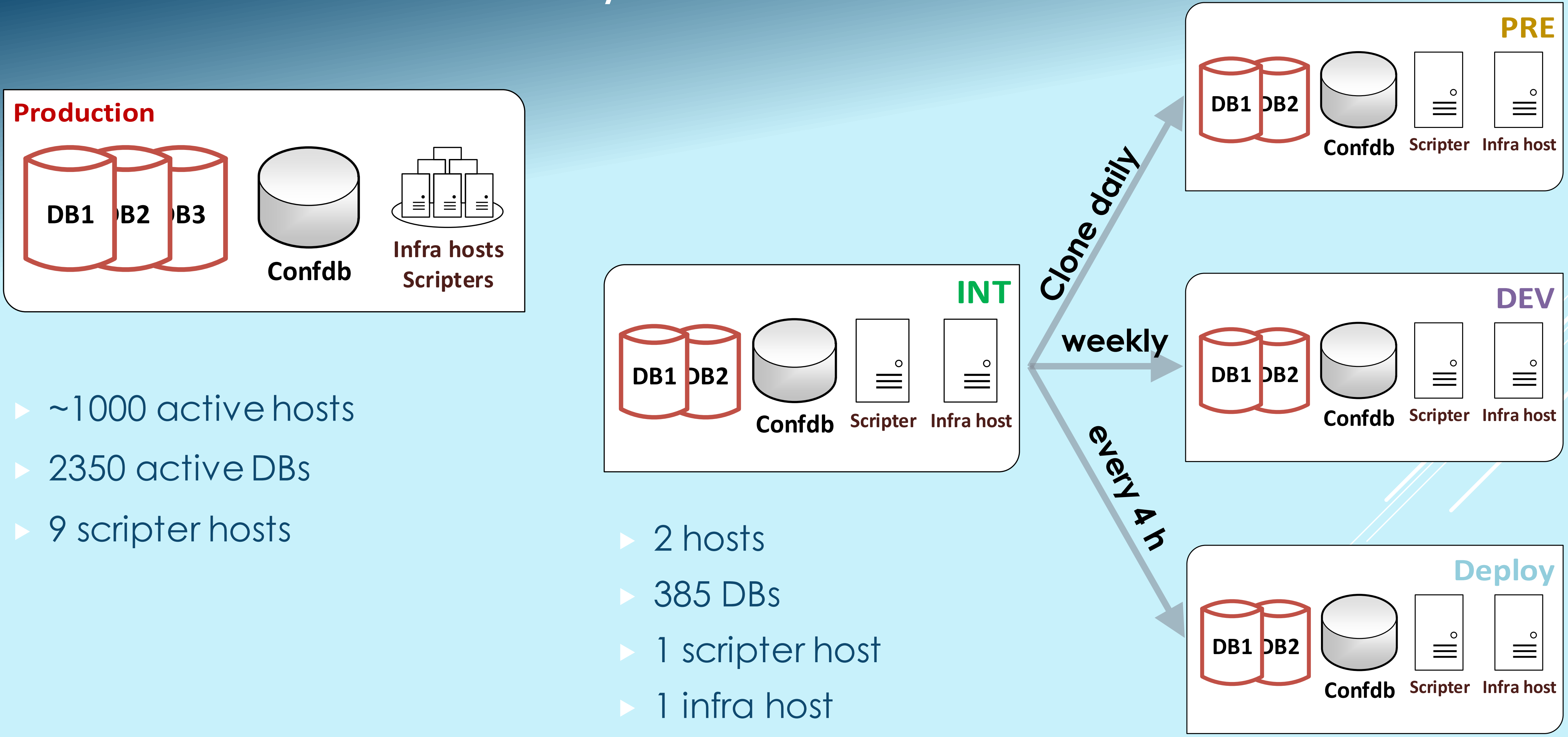
Skype Database Platform

Radoslav Glinský

March 2017

Skypename: rg_512 Email: radoslav.glinsky@skype.net

DB ENVS: UNTIL 02/2016



- ~1000 active hosts
- 2350 active DBs
- 9 scripter hosts

- 2 hosts
- 385 DBs
- 1 scripter host
- 1 infra host

SEVERE ISSUES IN TEST

- Differences between Prod and Test
- Poor hygiene
- Lack of ownership
- Problems have been accumulating for 10 years
- Enormous environment
- Recurring capacity issues

http://www.dailymotion.com/video/xd0rg9_monty-python-mr-creosote_fun

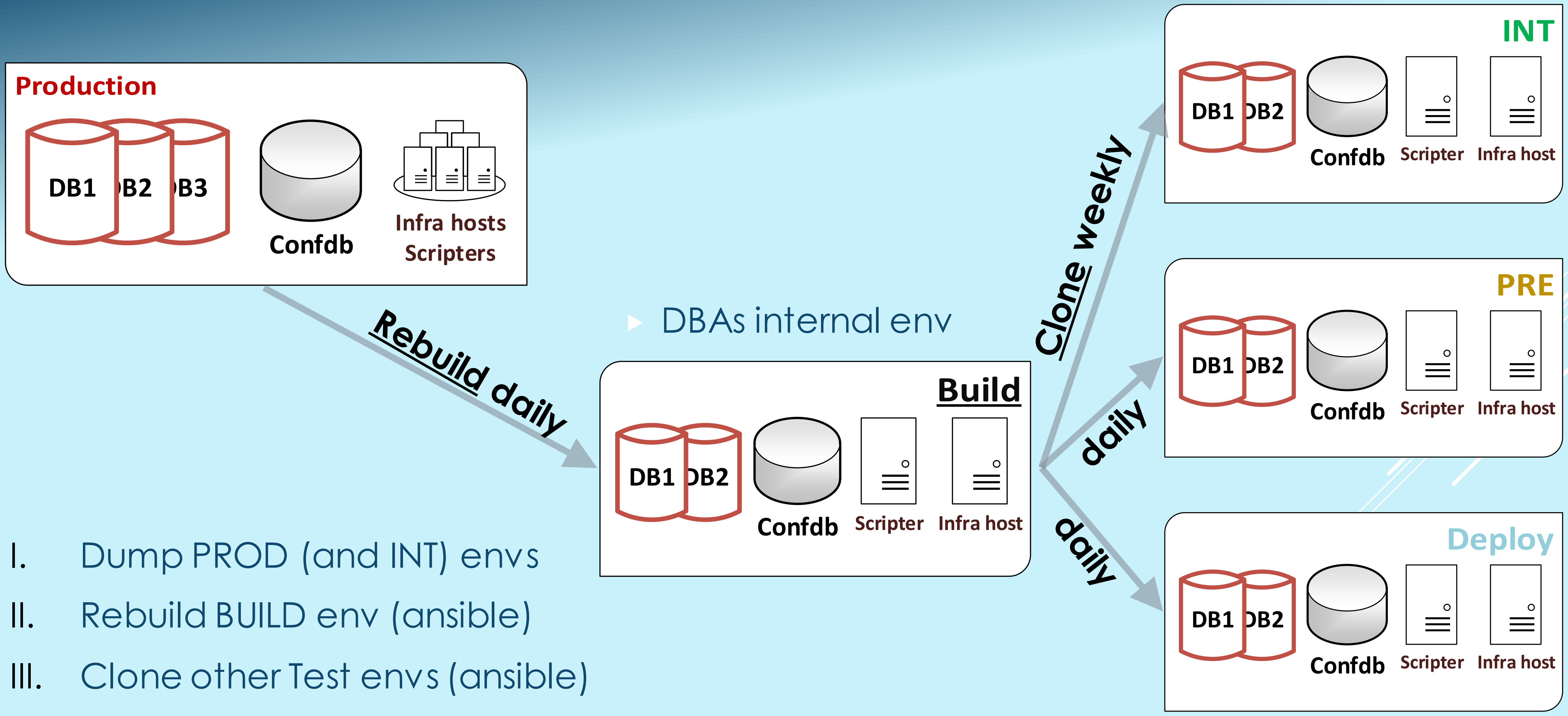
FIGHTING DIFFS – DDT.PY (TO BE OPEN-SOURCED)

```
$ ddt.py dump --source 'postgresql://clidb.int/clidb'

+-+= <PgCluster (server_version: '9.4.11') [1389]>
  +-+= <PgDatabase 'clidb' (owner: 'replicator', encoding: 'SQL_ASCII') [40]>
    +-+= <PgSchema 'infodb_classificator' (owner: 'replicator') [11]>
    | +-+= <PgTable 'countries' (columns: 5) [7]>
    | | +--= <PgColumn 'country_code' (num: 1, type: 'character(2)')>
    | | +--= <PgColumn 'country_name' (num: 2, type: 'character varying(64)')>
    | | +--= <PgColumn 'country_code_iso3' (num: 3, type: 'character(3)')>
    | | +--= <PgColumn 'default_ccy' (num: 4, type: 'currency')>
    | | +--= <PgColumn 'mobile_country_code' (num: 5, type: 'text')>
    | | +--= <PgTableConstraint 'pk_countries' (type: 'p')>
    | | \--= <PgIndex 'pk_countries' (table: 'infodb_classificator.countries', columns: 1)>
...

$ ddt.py diff --source 'postgresql://accountdb_p000.int/accountdb_p000'
              --target 'postgresql://accountdb_p000.dev/accountdb_p000'
...
   | +-+= <PgSchema 'entitlement_1_102' {origin:1} (owner: 'wallet') [81]>
   | +-+= <DiffElem (kind: 'schema', name: 'reputation', owners: ('id', 'id'), attrs: 0, elems: 1)>
   | | \--= <DiffElem (kind: 'function', name: 'get', owners: ('id', 'id'), attrs: 1, elems: 0)>
   | |       src_text: "..." != "..."
...
```

PROD -> TEST: SYNCING PRINCIPLES

- As similar to Prod as possible (structures, queues, consumers, replicas, scripts)
- Keep Prod naming – aim to reuse config files
- Keep the topology (plus multiple sites – disaster recovery), but reduce the scale (eg. 256 -> 2 shards)
- No Prod data! Generate data by running tests
- Bootstrap only minimal amount of data
- Small footprint, fast cloning
- Allow any number of environments to be created
- Resync regularly
- No dbspider (old data cleanup tool)
- Full access to development teams (to DBs, hosts, scripts)

DB ENVS: 03/2016 - NOW()



- DBAs internal env

I. Dump PROD (and INT) envs

II. Rebuild BUILD env (ansible)

III. Clone other Test envs (ansible)

DUMPING

FROM **PROD**

- DB structures
  - Dump every active Prod DB
    - (by DNS `dbname.dc1.db` & `dbname.dc2.db`)
  - Only limited shards (p000 and p001)
  - Using `pg_dump` (SQL format)
- DDT dumps
  - Replication Londiste3 configuration
  - PGQ configuration
  - Roles
  - RPC and cluster configuration
- Ext. DB scripts from scripter hosts
  - Only active during last week
  - Crontabs

FROM **INT**

- Data
  - configurations/metadata/ defaults/stats
  - majority are infodb and confdb tables
- Roles passwords hashes
  - Skip "system" users
- Ext. DB scripts
  - configs, keys, certs, .pgpass files
  - No executables!

DATAMASS

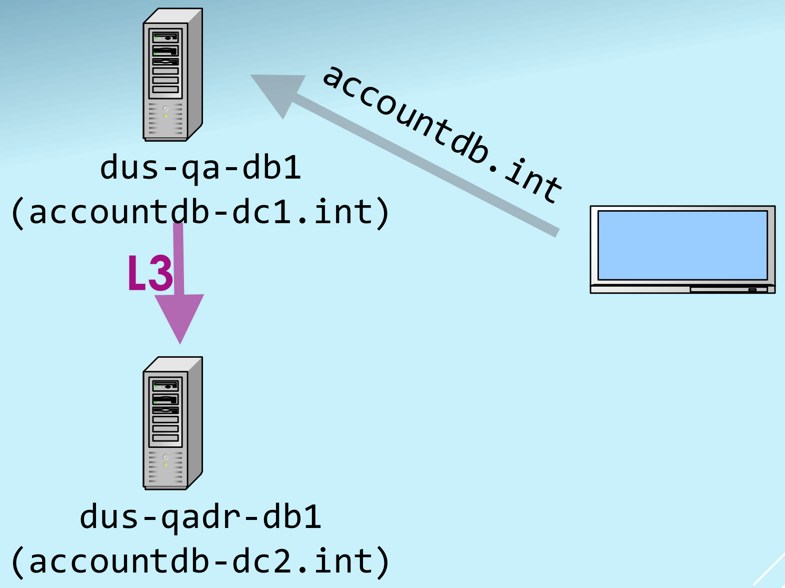
- Dumping data from INT env
- 420 tables with ~ 7GB of `INSERT SQL` statements
- Restore takes ~ 50 minutes (16 threads)

```
dump_cmd = pg_dump -U %(username)s --table=%(table)s --data-only --column-inserts
%(dbname)s --file=%(outputfile)s


restore_cmd = psql -h %(host)s -U %(username)s %(dbname)s -f –
```

REPLICA SETUP

- Londiste3 - trigger-based logical replication
- Input: DDT dumps
  - PGQ cascade – name, provider, londiste consumer (worker), node type, …
  - Londiste tables (with handlers), sequences
- Output: `londiste3` shell commands: ~ 6000 commands (already grouped)

Failover has also L3 replica:



```
$ londiste3 --ini --set queue_name=l3_accountdb_q --set db=<DB_CONN_STRING> > l3_accountdb_q.ini
$ londiste3 l3_accountdb_q.ini create-root accountdb_dc1 <DB_CONN_STRING>
$ londiste3 l3_accountdb_q.ini worker -d
$ londiste3 l3_accountdb_q.ini add-table reputation.users --expect-sync --no-triggers --handler="part"
--handler-arg="key=key_user"
```

- Create with `'--no-triggers'` – for non-standard cases in Prod
  - Add triggers afterwards as “`CREATE TRIGGER …`” from SQL dumps
- Takes ~ 7 minutes (generate + execute + add triggers)

EXTERNAL SCRIPTS SETUP

- Using utility library from Skytools – DB management tools from Skype
- ~450 scripts

Copied from PROD

- Executables
- Config files
  - structure (options)
  - generic options' values (e.g. conn_string, job_name, logfile, queue_name, ...)
- scripter hosts' crontab

Copied from INT

- Config files
  - custom options' values
- .pgpass
- Other files
  - ~ `'.*[.](cer|crt|ini|key|pem|txt|secret)$'`
  - `have to be` **`referenced`** `(absolute or relative path) from within config file`

RELMAN REPLAY

Problem: with Test envs rebuild we lose all dev changes

Solution: we redeploy release items into BUILD env

- after successful deployment to INT items gets registered into ”`releasehistorydb`”
- during rebuild we redeploy items which got deployed to INT but not yet to PROD
- items not older than 2 weeks !!!

On average 30 – 40 release items are replayed
(each RI ~ 1 minute)

REBUILD – SUMMARY

- Check if fresh dump is available
- Reset postgresql dbhome
- Restore DBs
  - install londiste, pgq
  - create roles (passwords from INT)
  - create databases
  - create pgq queues
  - restore structures (skip L3 triggers)

  - `fsync=off, full_page_writes=off`

- Generate DB DNS

- Setup Londiste3 replicas
- Fix DR primary DNS names
- Restore data (`datamass.py`)
- Set needed confdb configurations
- Run discovery, setup RPC, partconf
- External scripts
- Configure pgbouncer
- Relman Replay
- Copy data to the build repository

CLONE

- Check if build is fresh (1 day)
- Check if target hosts are available
- Stop everything
- Reset dbhome
- Clone dbhome
- Generate DB DNS names
  - fix DR primary DNS records
- Set needed confdb configurations
- Run discovery, setup RPC, partconf
- Start everything

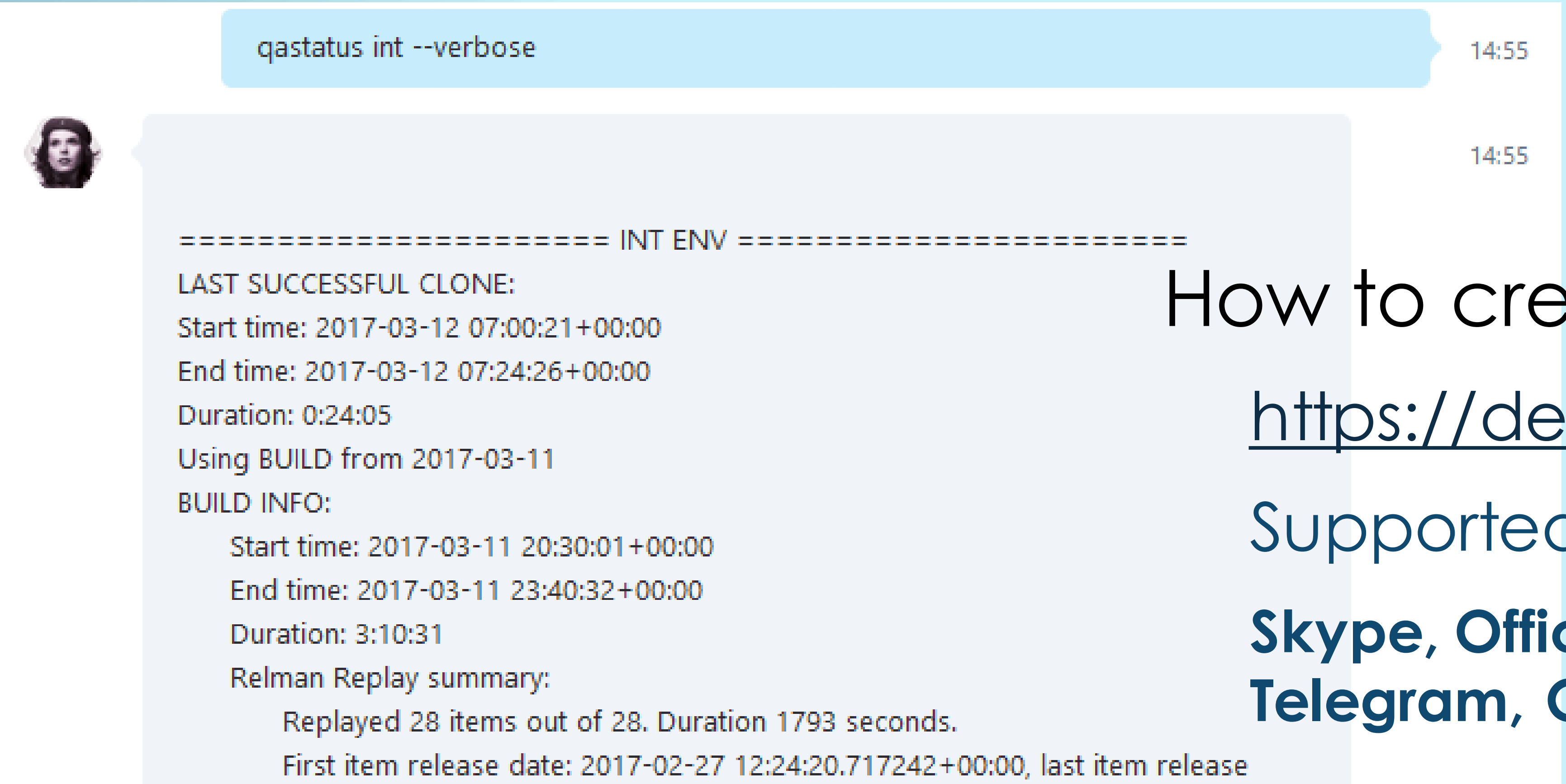
ISSUES

- All Prod DBs (including DR) has to be available
- Dependency on Prod issues
- “chicken-egg” problem Datamass – Relman Replay
- Relman Replay
  - just 2 weeks
  - individual release items occasionally fail
- Rebuild process – fail on “everything” principle

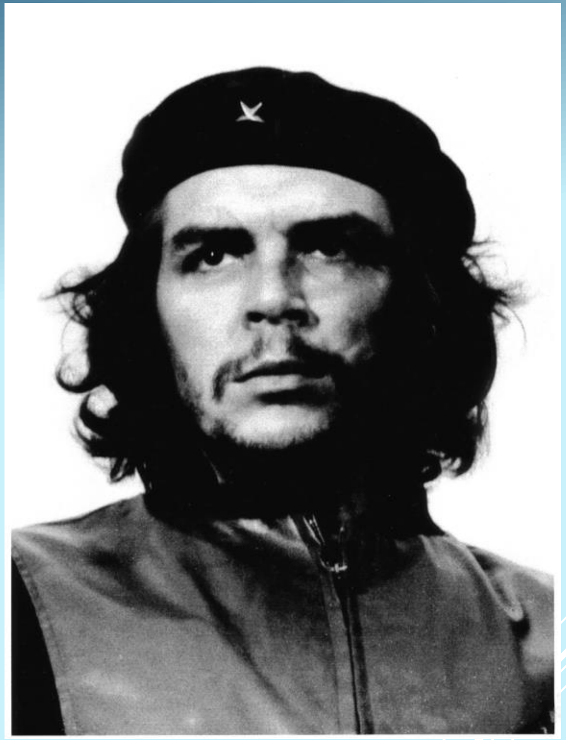
I WANT TO BE INFORMED…

Skype Bot **"Che"** gives the actual info:

- proactive messages into Skype chats
- summary info when requested



```
qastatus int --verbose                                    14:55

                                                          14:55

======================= INT ENV ========================
LAST SUCCESSFUL CLONE:
Start time: 2017-03-12 07:00:21+00:00
End time: 2017-03-12 07:24:26+00:00
Duration: 0:24:05
Using BUILD from 2017-03-11
BUILD INFO:
    Start time: 2017-03-11 20:30:01+00:00
    End time: 2017-03-11 23:40:32+00:00
    Duration: 3:10:31
    Relman Replay summary:
        Replayed 28 items out of 28. Duration 1793 seconds.
        First item release date: 2017-02-27 12:24:20.717242+00:00, last item release
```



How to create own bot:

https://dev.botframework.com

Supported channels:

Skype, Office 365, FB messenger, SMS Telegram, GroupMe, Kik, Slack, …

SUMMARY

Dumping time: ~30 min
Rebuild time: ~3 h
Clone time: ~20 min

	Old Test env	New Test env
HW	~1 TB	~100 GB fresh BUILD ~300 GB Test env under use (no dbspider)
Clone time	~ hours (1TB of files copy over network)	~ 20 minutes
DBAs efforts	Constant issues (1 DBA full time) Develop tooling – dbspider, diffs tracking	Reinitiate Clone
Dev teams	Access restrictions, policies	No restrictions
Testing	Big amount of diffs	Like in PROD