2ndQuadrant®
PostgreSQL

Run your own buildfarm server

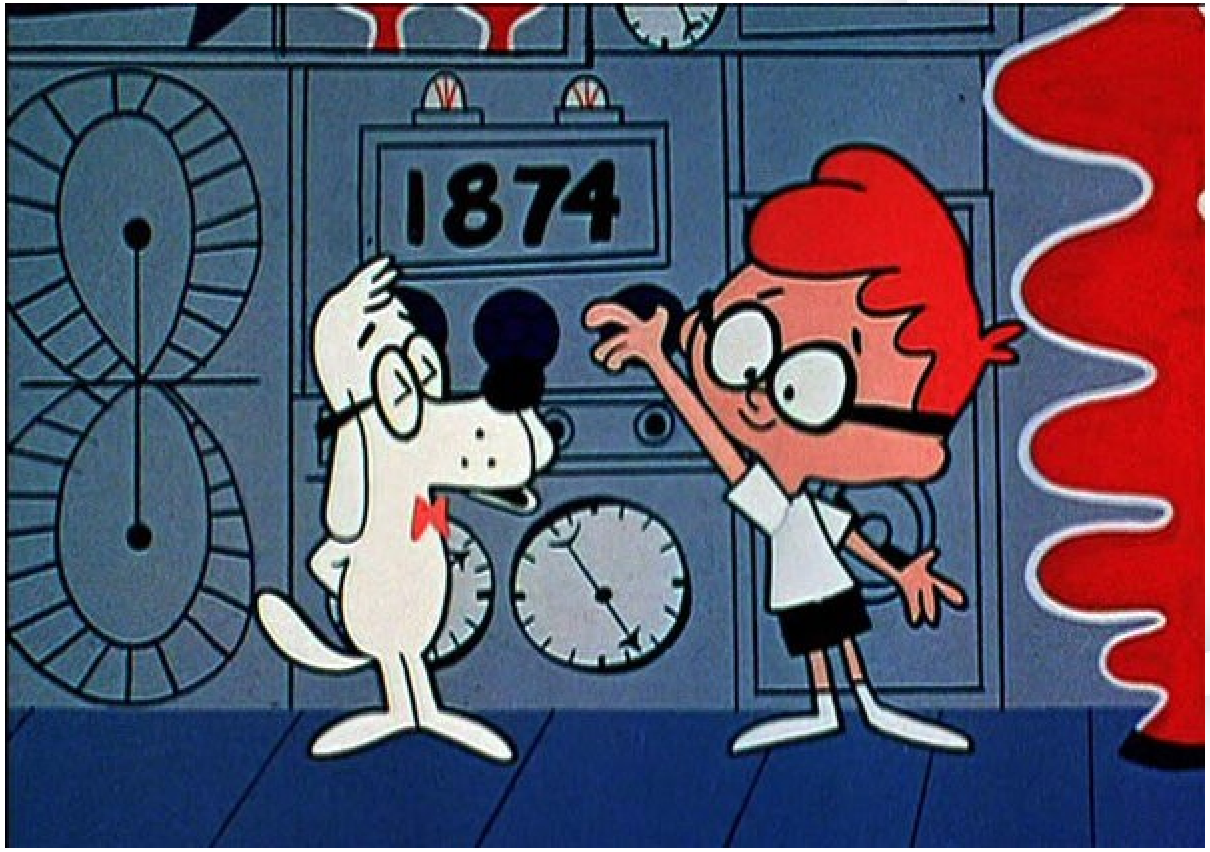
and test your own patches

Andrew Dunstan
andrew.dunstan@2ndquadrant.com

We’re hiring!

- See anyone from 2ndQuadrant if you’re interested

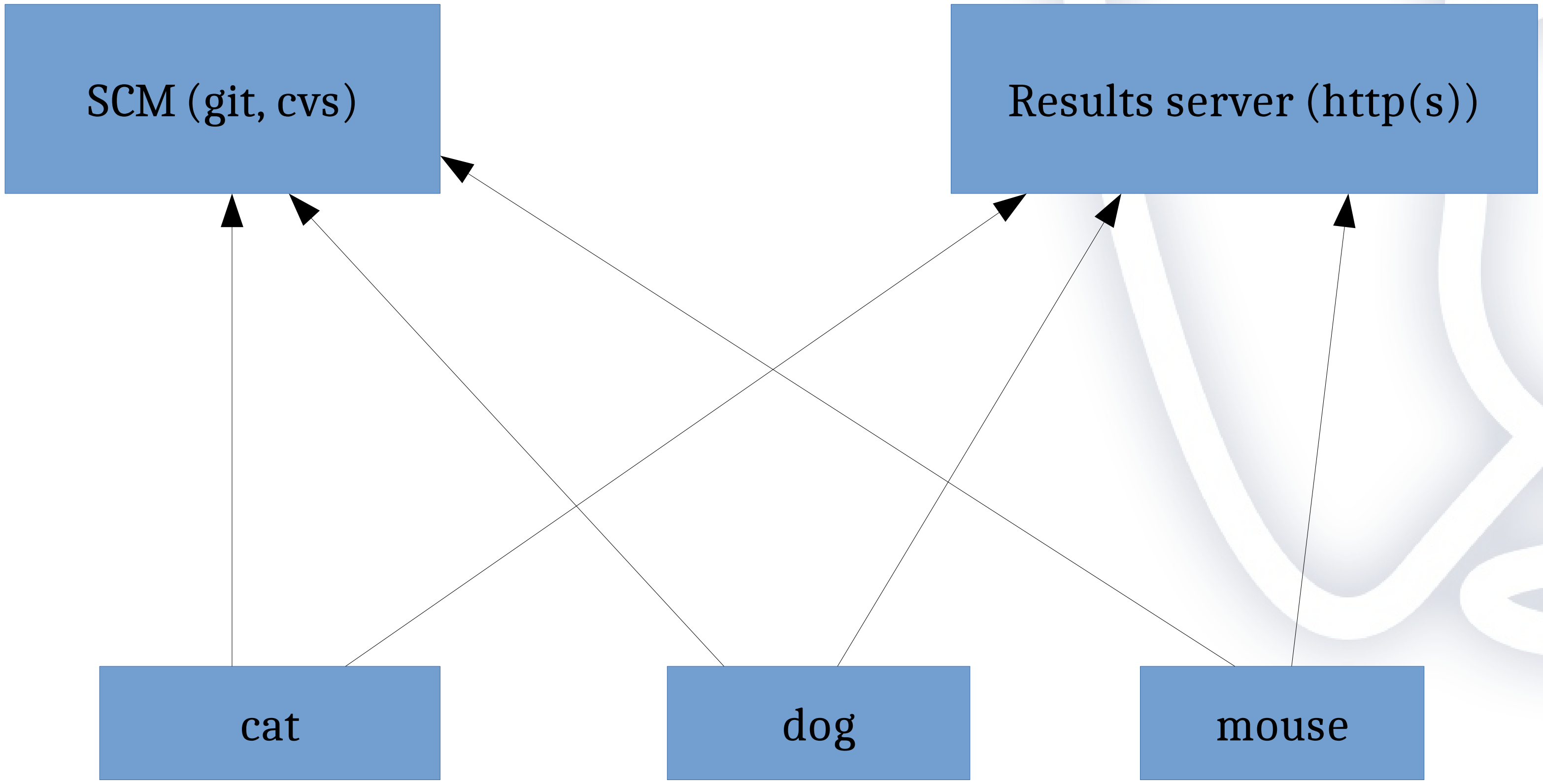
Sherman, set the Wayback Machine to 2004



A little history

- Have we broken something on some platform?
- Have we broken something with some configuration?
- Up to 2004 these questions were answered at best haphazardly
  - Often problems took week of months to discover
- Answer: the PostgreSQL Build farm
  - Very loosely inspired by SAMBA build farm

Architecture



Buildfarm concepts

- Clients are members or animals
- A member performs a build or run on a branch
- A run consists of a number of stages
  - e.g. make or check
- Possible, even common to run more than one animal on a single machine
  - Different configuration, compiler etc.

Lots of reports

- Currently 119 animals reporting
- Across 6 branches (5 stable + HEAD)
- 55,029 builds in the last 90 days (as of time of writing)
  - Highest count is 571 builds on HEAD (master) branch
- 600Gb of data in production, lots more in the archive
- History goes back to 2004, builds for Release 7.2.

Security

- No inbound connections
- Client can sit securely behind a firewall
- Has support for http proxies

Integrity

- There is a shared secret for every member
- Each report is signed (currently with SHA1, soon to be SHA256) with the secret

Buildfarm client

- https://github.com/PGBuildFarm/client-code
- Perl code
- Config file is also perl
  - Copy the sample file
- Two main scripts
  - **`run_build.pl`** performs a single run
  - **`run_branches.pl`** wrapper for `run_build.pl` in one of three modes
    - `--run-all`
    - `--run-parallel`
    - `--run-one`

Running the client

- `run_branches.pl –run-all –config foo.conf`
- How does it know which branches to build?
  - `$PGBuild::conf{global}->{branches_to_build}`
  - Can be a list ref:
    - ['REL_11_STABLE', 'HEAD']
  - Can be a scalar:
    - `'ALL'`
    - Or `'HEADPLUSLATEST3'`
    - Gets file `branches_of_interest.txt` from the server

Using a regular expression for branches_to_build

- Starting with release 10 of the buildfarm client
  - branch names can be multi-level
    - `dev/feature_1234`
    - `bug/ticket_5678`
    - `foo/bar/baz`
  - branches_to_build can be a regular expression:
    - `qr(dev/.*)`
    - checks out master branch and gets a list of branches, matched against the regular expression
- Not intended for use with public PostgreSQL Build farm
- Uses include:
  - Private sets of patches
  - Proprietary builds
- Change config's `scm_repo` to point to your private `git` repo

Branch name convention

- Use a convention
  - e.g. prefix/base_branch/something
  - Omit base_branch if not backpatching
    - `dev/my_feature_name`
    - `bug/REL_11_STABLE/ticket_number`

Branches from positional arguments

- `git` only, will be in next release
- Positional arguments to **`run_branches.pl`** taken as list of branches
- Overrides config file

2018: Can we upgrade the buildfarm server?

- We didn’t know
- We didn’t have a good way to find out
- No recipe existed for setting up a test server
- Solution: create a recipe!
  - Uses PostgreSQL Release 11
  - Runs on Debian/Stretch or Ubuntu/Bionic
  - TBD: support for RHEL/Centos (waiting for Centos8)

Setting up a test server

- `git clone https://github.com/PGBuildFarm/test-server.git testbf`
- `cd testbf`
- If using vagrant/Virtualbox:
  - `vagrant up`
- For use on the host:
  - `sudo sh provision.sh`

Server Application

- Set of perl CGI scripts and utilities
- Postgres database for storage
- Presentation layer is Perl Template Toolkit

Sample data

- Generated daily
- Populates the database with a tiny sample to get going
  - All the personal and secret info is stripped out
  - Three other tables are restricted:
    - build_status_log is restricted to the animal prion on the HEAD branch on its latest build
    - build_status_recent_500 is restricted to data for the last 90 days
    - build_status is restricted to builds on the dashboard
- For your own server, you should probably just unload the sample data, or comment the loading out of the provision script
  - The sample data tar file contains an unload script

What the test server won't do

- https
- email alerts and notifications
- Captcha
- Check that reported branches are in `branches_of_interest.txt`

Registering clients

- Fill in the form on the web site
- Connect to the server
  - e.g. `vagrant ssh`
- `sudo su – pgbuildfarm`
- `psql`
- `select * from pending();`
  - Result will have a name which is 6 hex digits
- `select approve('oldname','newname');`
  - Result will show owner's name, email and shared secret.
  - Email or otherwise communicate secret to the owner if it's not you

Choose a naming scheme

- Don't use animals
- Choose some list with a lot of members, and no accents or spaces, preferably not too long
  - e.g, Latin names from the Vulgate
  - List has 236 entries
- Hosts can have multiple members
- c.f. rfc2100

Database schema

- Almost completely generic
- Very loose relationship to the client

```
pgbfprod=> \dt+
                                  List of relations
 Schema |          Name           | Type  |    Owner     |  Size   | Description
--------+-------------------------+-------+--------------+---------+-------------
 public | alerts                  | table | pgbuildfarm | 56 kB   |
 public | build_status            | table | pgbuildfarm | 12 GB   |
 public | build_status_log        | table | pgbuildfarm | 556 GB |
 public | build_status_recent_500 | table | pgbuildfarm | 99 MB   |
 public | buildsystems            | table | pgbuildfarm | 176 kB |
 public | dashboard_last_modified | table | pgbuildfarm | 48 kB   |
 public | dashboard_mat           | table | pgbuildfarm | 504 kB |
 public | latest_snapshot         | table | pgbuildfarm | 248 kB |
 public | nrecent_failures        | table | pgbuildfarm | 144 kB |
 public | personality             | table | pgbuildfarm | 720 kB |
(10 rows)
```

buildsystems

- One row per buildfarm member
- Contains name, owner info, secret, etc.
- Normally the only table you might need to update

2ndQuadrant PostgreSQL

personality

- Contains updates to member personality, i.e. compiler and OS version

build_status

- One row per build
- Second largest table
- Contains stage at which build failed, or 'OK'
- Contains log from any failure

build_status_log

- Largest table (by far)
- One row for every stage of every build, including the log
- Badly needs to be partitioned

build_status_recent_500

- Extract from `build_status`
- Speeds up queries that would be much slower if fetching from `build_status`
- Inserts by trigger
- Periodically purged by cron job

dashboard_mat

- Home grown materialized view that feeds the dashboard page
- Refreshed every time there is a new build reported

nrecent_failures

- Home grown materialized view of failures
- Feeds the failures page
- Refreshed every time a failure is reported

latest_snapshot

- Extract from build_status
- Used for members page
- One row per member / branch
- Maintained by trigger

dashboard_last_modified

- One row table
- Used for setting cache headers on dashboard page

alerts

- Used for sending email alerts of missing builds if requested by the user
- This functionality is disabled by default in the test server

Using your own repo

- On the server (as pgbuildfarm):
  - `cd /home/pgblocal`
  - `rm -rf postgresql.git`
  - `git clone --bare -q <your-repo> postgresql.git`

Setting up the client

- In the config file
  - Point `scm_repo` to the right git repo
  - Point `target` to new buildfarm server
  - Set `branches_to_build` to a regular expression
- Other good config settings
  - Turn off `git_keep_mirror`
  - Turn on `use_vpath`

Test everything is OK

-
```
./run_branches.pl --run-all --config myconfig --test \
    --only-steps "configure make check" HEAD
```

Register the new animal

- Via your new web site
- Then login to the machine/database to run the approval process

Add credentials to your config file

- The **`animal`** and **`secret`** settings

Run for real

- `./run_branches.pl --run-all --config myconfig`

Demo!

- Git repo: https://bitbucket.org/adunstan/pgdev-demo.git
- Server: http://ec2-18-221-185-22.us-east-2.compute.amazonaws.com/cgi-bin/show_status.pl
  - a.k.a. https://bit.ly/2wslX1a
- Commits: local machine
- Buildfarm client: another EC2 instance

- Above URLS will disappear shortly after this session

2ndQuadrant® PostgreSQL

Questions?

Andrew Dunstan
andrew.dunstan@2ndquadrant.com