Buildfarm Client as a Development Tool

Andrew Dunstan
andrew.dunstan@2ndQuadrant.com
“My code broke the buildfarm”

• “I ran make check.”
The buildfarm tests lots of things

- More than anyone usually runs by hand
- More than most automated testers too
- Multiple locales
- TAP tests
- Odd corners (PLs, ECPG)
- Non-core modules (e.g. FDWs)
- Cross-version pg_upgrade
- Docs
Example: prion

- SCM-checkout
- Configure
- Make
- Check
- Contrib
- TestModules
- Install
- ContribInstall
- TestModulesInstall
- pg_upgradeCheck
- test-decoding-check
- initdbCheck
- pg_archivecleanupCheck
- pg_basebackupCheck
- pg_configCheck
- pg_controldataCheck
- pg_ctlCheck
- pg_dumpCheck
- pg_resetwalCheck
- pg_rewindCheck
- pgbenchCheck
- scriptsCheck
- recoveryCheck
- subscriptionCheck
- authenticationCheck
- Initdb-C
- InstallCheck-C
- IsolationCheck
- PLCheck-C
- ContribCheck-C
- TestModulesCheck-C
- Initdb-en_US.iso885915
- InstallCheck-en_US.iso885915
- PLCheck-en_US.iso885915
- ContribCheck-en_US.iso885915
- TestModulesCheck-en_US.iso885915
- Initdb-en_US.utf8
- InstallCheck-en_US.utf8
- InstallCheck-ICU-en_US.utf8
- PLCheck-en_US.utf8
- ContribCheck-en_US.utf8
- TestModulesCheck-en_US.utf8
- ECPG-Check
- find-typedefs
Normal operation

• Check out source
  - Is the source clean?
• Has there been a change since last run?
• Build and run tests
• Update state
• Upload results to server
Test Mode

• Check out source
  - Is the source clean?
• Has there been a change since last run?
• Build and run tests
• Update state
• Upload results to server
From-Source mode

- Get source location from command line
  - Or possibly config file
- Check out source
  - Is the source clean?
- Has there been a change since last run?
- Build and run tests
- Update state
- Upload results to server
Two ways to run from source

• **--from-source**
  - Works best if you use vpath builds
  - Enable use_vpath in the config file

• **--from-source-clean**
  - Runs make distclean or MSVC equivalent first
  - Fails if use_vpath is set
  - Fails on Unix if there is no Gnumakefile
  - Works well on MSVC builds
Using the two modes together

- Don’t use vpath
- First time around use --from-source
- On later runs use --from-source-clean
Grab a copy of the code

• mkdir -p /path/to/bfclient
• cd /path/to/bfclient
• wget https://buildfarm.postgresql.org/downloads/latest-client.tgz
• tar --strip-components=1 -zxf latest-client.tgz
Or get it from git

- `mkdir -p /path/to/bfclient`
- `cd /path/to/bfclient`
- `git clone https://github.com/PGBuildFarm/client-code.git`
Check requirements

- `cp build-farm.conf.sample build-farm.conf`
- `./run_build.pl --test`
  - Install needed software
  - Or remove requirement in config file
    - See config_opts section
Run against your code

• ./run_build.pl --from-source /path/to/pgsource
Run in your code directory

- cd /path/to/pgsource
- cp /path/to/bfclient/build-farm.conf.sample build-farm.conf
- mkdir buildroot
Tell git to ignore this stuff

- `echo /build-farm.conf >> \ .git/info/exclude`
- `echo /buildroot >> \ .git/info/exclude`
Edit config file

• Uncomment this line:
  `$PGBuild::Options::from_source = $confdir;`

• Change these two settings:
  `buildroot => "$confdir/buildroot",`
  `use_vpath => 'true',`
... and just run

- `/path/to/bfclient/run_build.pl`
- Default branch name is HEAD
  - git branch is ignored
  - Might not be a git repo at all
Useful alias

- `git config --global alias.bfclean "clean -e /buildroot -e /build-farm.conf"`
- `git bfclean -dfx`
Build artefacts

- buildroot/{branchname}/pgsql.build
  - if it’s a vpath build
- buildroot/{branchname}/inst
- Always cleared away at the start of a run
- Normally cleared away at the end of a run
  - Change with --keepall option
Using the installation directory

echo "unix_socket_directories = '/tmp/'" >> data-C/postgresql.conf

bin/pg_ctl -D data-C/ -l logfile start

bin/createuser -U buildfarm -s $USER

bin/createdb

• I have these in a shell function
Using the build directory

• If it’s a vpath build
  - `cd buildroot/{branchname}/pgsql.build`

• make and other commands just work

• For MSVC, `vcregress` works, but `build` doesn’t
  - Needs proper environment settings
Log files

- One per step
- Normally in buildroot/{branch}/{animal}.lastrun-logs
- For from-source builds in buildroot/{branch}/{animal}.fromsource-logs
- Cleaned out at the start of a run
Tweaking log file

- Change animal name
  - Anything you like as it’s not going to send to the server
  - Name is used in log file destination and messages

- Change the base_port setting
  - Don’t conflict with other animals

- Enable TAP tests
Config file and buildroot can go anywhere

- I find it convenient to colocate them with the code
- /path/to/run_build.pl --config /path/to/config-file --from-source /path/to/sourcecode
Client installation can go anywhere

- Requires version 8 client
- Can link from a directory in your PATH
  - `ln -s /path/to/buildfarm/*.pl /usr/local/bin`
  - Requires setting `BFLIB=/path/to/buildfarm`
    - Directory with PGBuild
Controlling what’s run

- Use `--skip-steps` or `--only-steps`
- Can only use one
- Space separated list of step names
List of steps for filters

- make
- check
- make-contrib
- make-doc
- testmodules
- install
- install-check
- contrib-install-check
- pl-install-check
- installcheck-collate-$locale
- installcheck-icu
- isolation-check
- pg_upgrade-xversion-check
- test-decoding-check
- testmodules-install-check
- ecpg-check
TAP test steps for filters

• bin-check
• misc-check
• {testname}-check
  – Named after directory
  – e.g. recovery-check, initdb-check
Filtering regression tests

- --schedule filename
- --tests "test1 test2"
- Not for MSVC builds
Setting up for Valgrind

- Adjust config file
  - `use_valgrind => 'true'`,
  - In `config_env` section
    - `CFLAGS => "-fno-omit-frame-pointer -O0 -fPIC"`,
    - `CPPFLAGS => "-DUSE_VALGRIND"`,
- Default valgrind options probably OK
- Turn off `--enable-cassert`
Running with valgrind

- doesn’t operate for check step or TAP tests
- isolation_check step takes 4+ hours
- install_check takes over 1hr
- Use --tests or --schedule
- Use --only-steps
Typedefs

- Enable in config file
- Or on command line with `--find-typedefs`
- Output is in `typedefs.log`
- Merge with output from buildfarm server
  - [https://buildfarm.postgresql.org/cgi-bin/typedefs.pl](https://buildfarm.postgresql.org/cgi-bin/typedefs.pl)
- Remove duplicates for slightly faster performance
- `pgindent --typedefs typedefs-file`
Running in a Docker container

- `git clone https://github.com/PGBuildfarm/Dockerfiles.git \ bf-docker`
- `cd bf-docker`
- Recipes for Ubuntu, Fedora, Alpine
- `mkdir buildroot`
- `docker run --rm -v `pwd`:/app/buildroot:buildroot bf-f28 \ cp build-farm.conf.sample buildroot/build-farm.conf`
- `docker run --rm -v `pwd`:/app/buildroot:buildroot bf-f28 \ run_build.pl --config buildroot/build-farm.conf \ --test`
Run your source with docker

- edit config file
  - use vpath builds
  - set TZ

```bash
docker run --rm \\
-v `pwd`/buildroot:/app/buildroot \\
-v /path/to/src:/app/pgsrc \\
run_build.pl \\
--config buildroot/build-farm.conf \\
--from-source /app/pgsrc
```
The buildfarm client is a useful tool in the hands of developers and can make many complex tasks simpler.

andrew.dunstan@2ndQuadrant.com