

Pre-merge checks

2020 Virtual LLVM Developers' Meeting



Mikhail Goncharov - goncharov@google.com

1

Hi! I am Mikhail Goncharov, and I work at Google in Munich on automating our internal Clang releases. This talk is a short overview of pre-merge checks we first introduced about a year ago, in the end of 2019.

Why?

2

up to 20% of commits in master branch of llvm-projects don't build or pass tests (*)

Broken master branch is not good for:

- users
- contributors
- tools

(*) YMMW, see slide notes

Let me show you some data. I built and tested the LLVM monorepo on a single platform at the sample of 400 out of 3000 recent commits selected randomly from the master branch. Around 20% of them failed tests, and some even failed to compile. This is a real problem for us as users and contributors of LLVM.

Long ranges of commits that fail to build can also be an obstacle to bisecting more subtle issues.

I think that most of us would prefer to see that the change is breaking before it is pushed to the repository. Not only this helps users, but also saves other contributors from starting to work on a "bad" commit.

Such high failure rate is surprising for me too. Probably there were some issues with the setup of an agent that run builds. Or this number might be explained by test flakiness. I have not looked into the exact failures' details. By no means it represents what actual user experiences day to day as I have not taken in account that 1. failures might be from different sub-projects while a person might be interested in a single one 2. there is no checks how long a commit was active (e.g. one that was reverted within 1 minute has the same weight as one that stayed HEAD for an hour).

Builds were run on on a random subsets of commits from 3000 latest commits as of today (2020-09-04) master branch <https://github.com/llvm/llvm-project> from categories (plus sample size and failure rate):

“reviewed” (has “Differential revision:” in the message): 200 commits, 19% failure rate

“reviewed NFC” (“reviewed” and has “NFC” in the message): 50 commits, 18% failure rate

“not reviewed” (does not have “Differential revision:” in the message): 200 commits, 20% failure rate

“not reviewed NFC” (“not reviewed” and has “NFC” in the message): 50 commits, 22% failure rate

“revert” (“revert” or “this reverts commit” in the message and “not reviewed”): 50 commits, 14% failure rate

“fix” (“fix” in the message): 50 commits, 20% failure rate

Run commands:

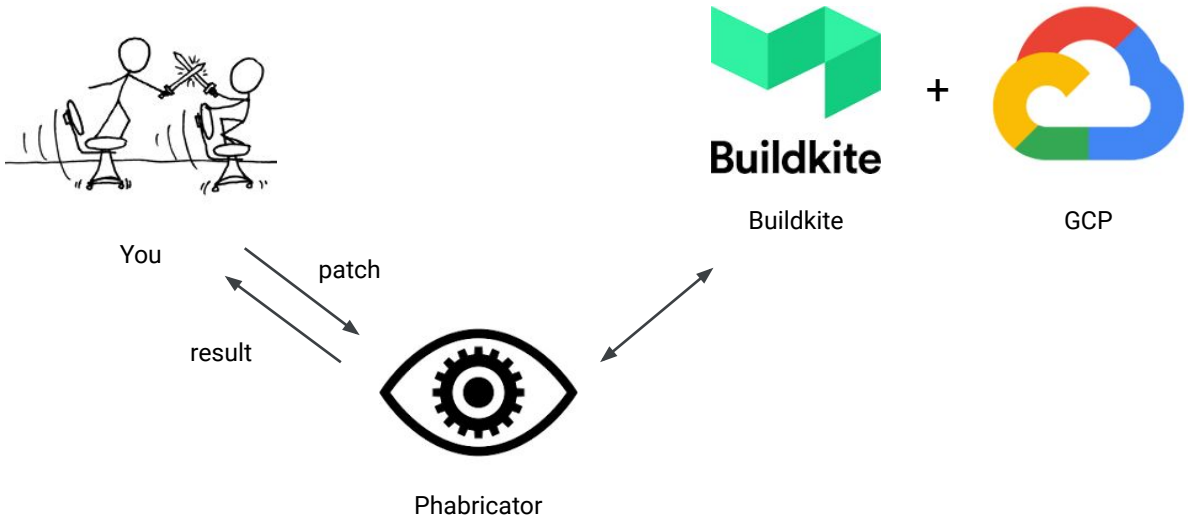
```
git clone https://github.com/llvm/llvm-project.git llvm-project
cd llvm-project
git checkout <command>
rm -rf build
mkdir build
cd build
export CC="clang"
export CXX="clang++"
export LD="LLD"
cmake ../llvm -D
LLVM_ENABLE_PROJECTS="clang;clang-tools-extra;libc;libcxx;libcxxabi;lld;libunwind;mlir;openmp;polly" -G Ninja -D CMAKE_BUILD_TYPE=Release -D
LLVM_ENABLE_ASSERTIONS=ON -D LLVM_BUILD_EXAMPLES=ON -D
LLVM_LIT_ARGS="-v --xunit-xml-output test-results.xml" -D
LLVM_ENABLE_LLD=ON -D CMAKE_CXX_FLAGS=-gmlt
ninja all
ninja check-all
```

Docker image used:

<https://github.com/google/llvm-premerge-checks/blob/dd4e6210bb21c53fb71f472a74e92af7c1c1596c/containers/buildkite-premerge-debian/Dockerfile>

Test changes before merging?!

3



Pre-merge checks are a simple idea: before pushing a commit, build the software and run its tests. This practice is well-known in the industry and is used often.

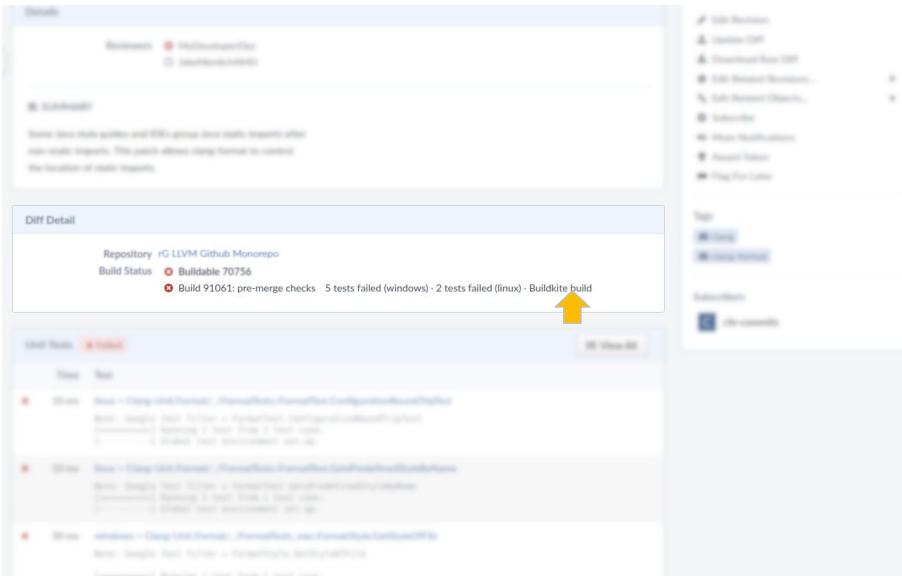
We have implemented pre-merge checks for LLVM. Buildkite orchestrates the builds, Google Cloud runs x86_64 Linux and Windows builder VMs, and Phabricator displays the results.

Build tried to detect which projects are affected by the change and also runs clang-format and clang-tidy on it.

Note that buildbots are different from pre-merge checks: buildbots run checks on changes that have been already pushed.

Image: <https://xkcd.com/303/>

How to



When you upload a new diff to Phabricator, it is automatically picked up by the pre-merge checking infrastructure. Typically it takes about 30 minutes for the results to appear in Phabricator. Let's see what they look like.

In the case of failure there are links to build or test reports and some comments on the diff. You can always open a Buildkite build report to understand how to reproduce the build locally, and access the full log for build and test commands.

How to

Buildkite

Log In Sign Up Learn More...

LLVM project / premerge checks
builds branch from https://github.com/llvm-premerge-tests/llvm-project

9,570 Builds 3 Running 0 Scheduled

build and test Build #9532 phab-diff-290121 c61ce84 Failed in 34m 23s

setup build and test linux build and test windows report

premerge bot Triggered from Pipeline
Created yesterday at 6:52 AM diff checks - Build #10942 / build and test

setup export SRC=\${BUILDKITE_BUILD_PATH}/llvm-premerge-checks && rm -rf \${SRC} && git... Ran in 11s Waited 1s linux-agents-5b977c56ff-zt9zf-1

build and test linux set -euo pipefail && ccache --zero-stats && ccache --show-config... Ran in 10m 45s Waited 5s linux-agents-5b977c56ff-zt9zf-1

build and test windows socache --zero-stats && set SRC=%BUILDKITE_BUILD_PATH%/llvm-premerge-checks && rm ... Ran in 33m 50s Waited 5s w32-1

report mkdir -p artifacts && buildkite-agent artifact download "**_result.json" . && e... Ran in 12s Waited 6s linux-agents-5b977c56ff-zt9zf-1



How to

6

```
✖ build and test linux set -euv pipefail && ccache --zero-stats && ccache --show-config Ⓞ Run in 10m 45s Ⓞ Watch Ⓞ Inux agents 06577c568f2f6d1
```

Log Artifacts 6 Timeline New Environment

```
+ Expand + Collapse groups Download Follow
```

```
1 * Running global pre-checkout hook 0/0
2 * Preparing working directory 0/0
3 * Running commands 0/0
4 * cmake 0/0
5 * ninja all 0/0
6 * ninja check-all 0/0
7 * clang-tidy 0/0
8 * clang-format 0/0
9 * Summary 0/0
10 OK cmake 0/0
11 OK ninja all 0/0
12 FAIL ninja check-all 0/0
13 OK clang-tidy 0/0
14 OK clang-format 0/0
883 * Reproduce build locally 0/0
884 git clone https://github.com/llvm-premerge-tests/llvm-project.git llvm-project
885 cd llvm-project
886 git checkout c61ce848c805b473c521c8613a0427f87e83375c
887 rm -rf build
888 mkdir build
889 cd build
890 export CC="clang"
891 export CXX="clang++"
892 export LD="lld"
893 cmake ../llvm -D LLVM_ENABLE_PROJECTS="clang;clang-tools-extra;compiler-rt;libc;lld;llgo;llvm;openmp" -G Ninja -D CMAKE_BUILD_TYPE=Release -D
  LLVM_ENABLE_ASSERTIONS=ON -D LLVM_BUILD_EXAMPLES=ON -D LLVM_LIT_ARGS="-v --xunit-xml-output test-results.xml" -D LLVM_ENABLE_LLD=ON -D
  CMAKE_CXX_FLAGS=-gmlt
894 # Note that compiler cache arguments are omitted
895 ninja all
896 ninja check-all
897 apt-get -qq --no-install-recommends install clang-tidy clang-format
898 git clang-format HEAD~1
```

How to

build and test linux set -euo pipefail && ccache --zero-stats && ccache --show-config... Ran in 10m 45s | Waited 5s | linux-agents-5b977c56ff-zt9zf-1

Log | Artifacts 6 | Timeline New | Environment | Retry

ninja-check-all.log	5.91 MB	c734c451dfc4ff80eaf35a652a937fde5effb1b
artifacts/ninja-all.log	584 KB	9de570ee9708034716a0003ec5c48d0de2003a44
artifacts/packages.txt	42.7 KB	3ff62f7443f5f696f75ba972122886e2b7a7eced
artifacts/ninja-check-all.log	5.91 MB	c734c451dfc4ff80eaf35a652a937fde5effb1b
artifacts/CMakeCache.txt	99 KB	2027b7541dc29fc29faf54e33be5ff842914d6c6
linux_result.json	11.9 KB	83a7e32ff13b2e14ca4858ad1c3106201fe1c862

build and test windows ... Ran in 23m 55s | Waited 5s | w32f-1

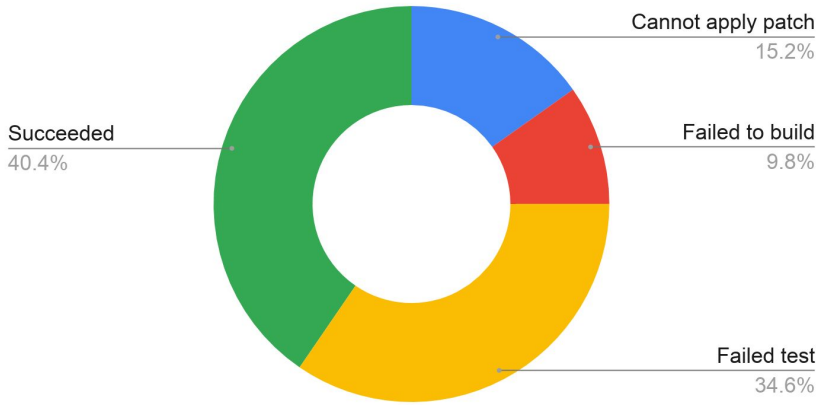
Stats

8

- 700 revisions weekly
- ~5% of revisions got fails check at first and get fixed

The pre-merge infrastructure completes around 700 builds weekly.
Around 5% of revisions fail pre-merge checks at first and get fixed afterwards.
Maybe, because reviewers or the author looked at the build result.

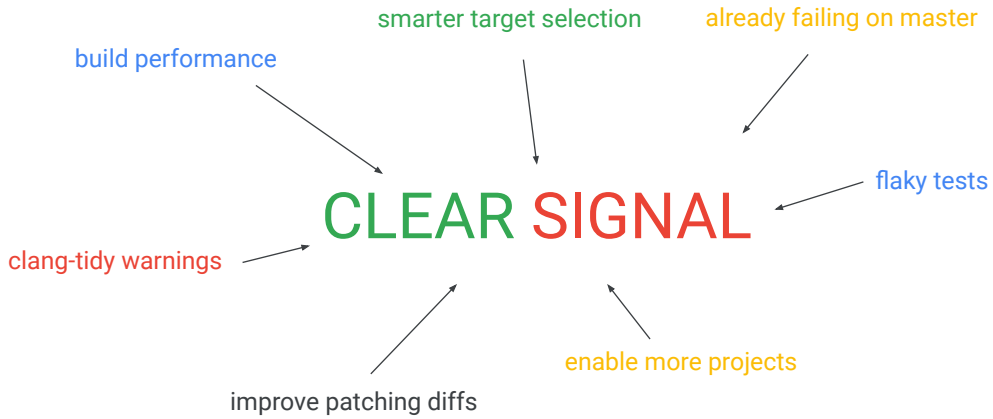
Stats



Only 40% revisions pass pre-merge checks. 15% fail because patch cannot be applied, 10% does not build. Last 35% fail tests.

What's next

10



Yes, 60% fail rate seems to be unreasonably high.

Next goal is to provide a much more clear signal if a change is good or not. There are a lot of ideas and existing issues.

One of the major ones is that some sub projects are completely disabled as their tests constantly fail or time out.

Join us!

11



<https://github.com/google/llvm-premerge-checks>



LLVM Discord <https://discord.gg/T6ZZ2Ju> @goncharov



goncharov@google.com

If you like the idea behind the project and want to improve it please help and contribute! It definitely needs more hands.

For example you see that you build failed but should not, please flag that; and maybe help investigate the root cause. Or maybe some wording can be improved.

Also if you want to add some specific checks for your subproject.

We are trying to keep the project as open as possible, all scripts and configurations are in this GitHub repo.

Please reach me on LLVM discord, GitHub, or email.

Thank you and have a green build!