What Does it Take to Run LLVM Buildbots?

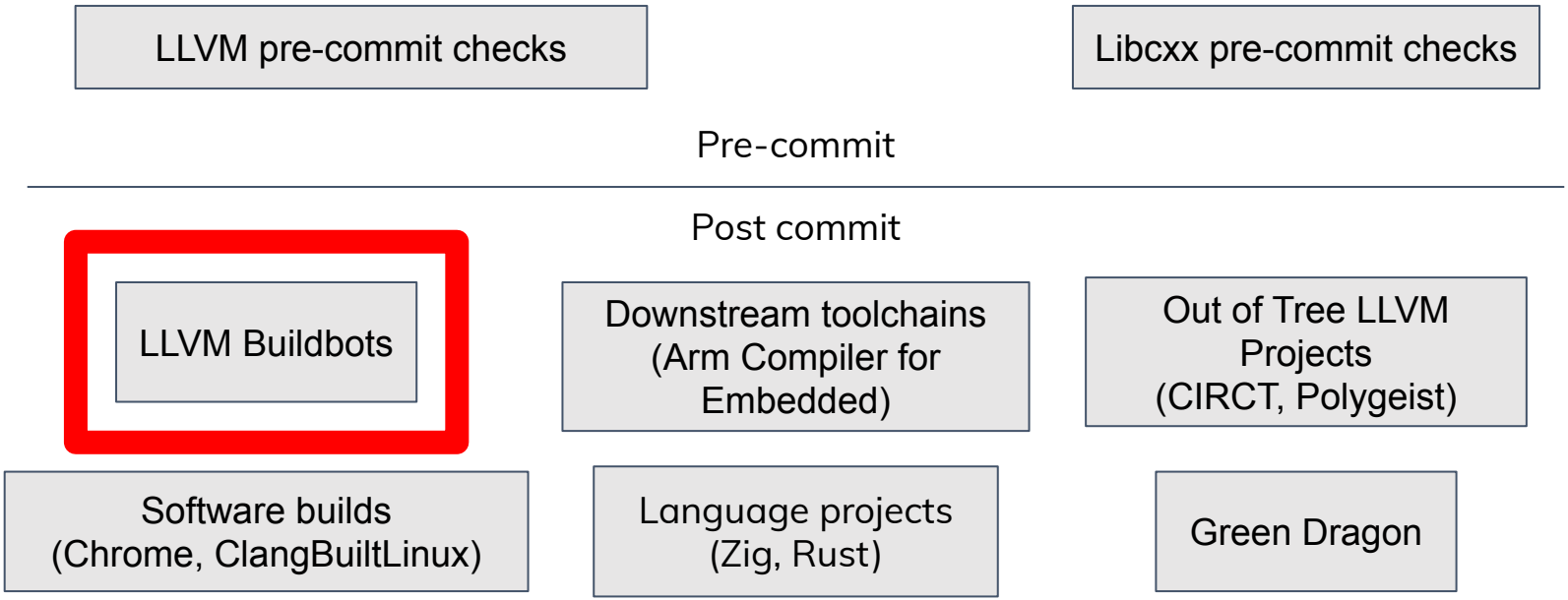
David Spickett, Staff Software Engineer, Arm

Who is Linaro?

- Linaro works to ensure open source projects are best on Arm.
- I am assigned from Arm to Linaro's Toolchain Working Group (TCWG).
- TCWG works on LLVM, GCC and QEMU.

We care about the quality of LLVM.

Bots All The Way Down...

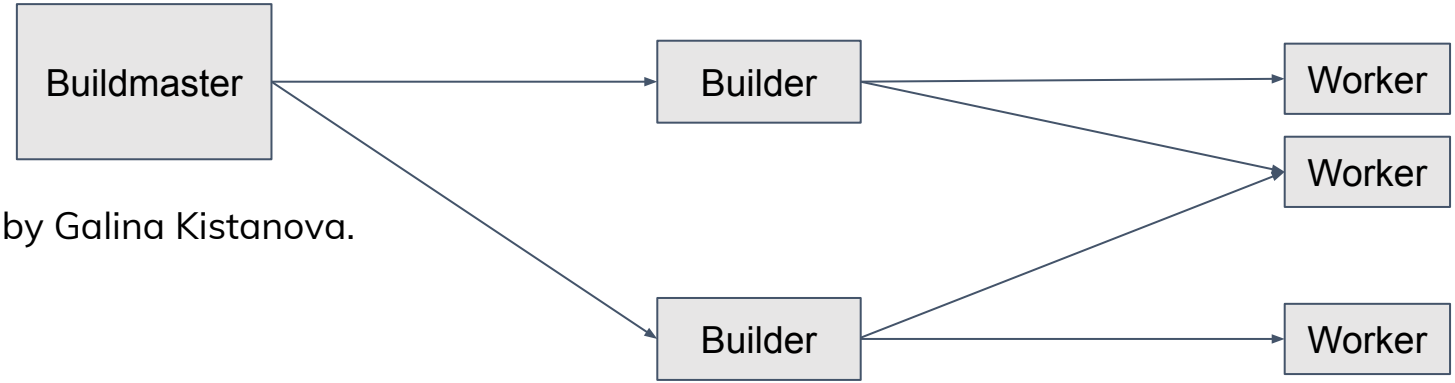


Linaro

What is a Buildbot?

- Post commit verification of changes.
- Build anything from all to just one project.
- Emails when your commit was in a failed build.

LLVM Buildbots



Run by Galina Kistanova.

Build configs e.g.
“clang-armv7-vfpv3-full-2stage”

Community provided
machines.

~162 Builders for LLVM

LLVM Commits: The Numbers

(from January 1st 2021 to January 1st 2022)

32810 commits*

~90 commits a day

~4 commits an hour

- Buildbots batch commits.
- Many are rarely idle.

* includes 1617 reverts and relands

Linaro and Buildbots

- First buildbot added 2013
- 29 currently

clang-arm64-windows-msvc	
clang-arm64-windows-msvc-2stage	
clang-armv7-2stage	8724
clang-armv7-global-isel	8595
clang-armv7-lnt	13008 13007
clang-armv7-quick	19961 19960 19959
clang-armv7-vfpv3-2stage	
clang-armv7-vfpv3-full-2stage	
clang-armv8-lld-2stage	
clang-native-arm-lnt-perf	
lldb-arm-ubuntu	27168 27167 27166

clang-aarch64-full-2stage	
clang-aarch64-global-isel	7250
clang-aarch64-lld-2stage	
clang-aarch64-quick	19431 19430
clang-aarch64-sve-vla	2735
clang-aarch64-sve-vla-2stage	
clang-aarch64-sve-vls	
clang-aarch64-sve-vls-2stage	
flang-aarch64-debug	7047 7046
flang-aarch64-dylib	8696 8695 8694
flang-aarch64-latest-clang	7817 7816
flang-aarch64-latest-gcc	10629 10628 10627
flang-aarch64-out-of-tree	17493 17492
flang-aarch64-rel-assert	8711 8710 8709
flang-aarch64-release	8665 8664 8663
flang-aarch64-sharedlibs	8631 8630

Why So Many?

Architectures
Armv7
AArch32
AArch64

Vector Extensions
VFPv3
NEON
SVE

Platforms
AArch64 Linux
Arm Linux
Windows on Arm

Compilers
Clang
GCC

Instruction Selection
SelectionDAG
GlobalIsel

Projects
LLVM
Clang
LLDB
Flang

Build Type
1 stage
2 stage
Debug
Shared libraries

Many testing dimensions.

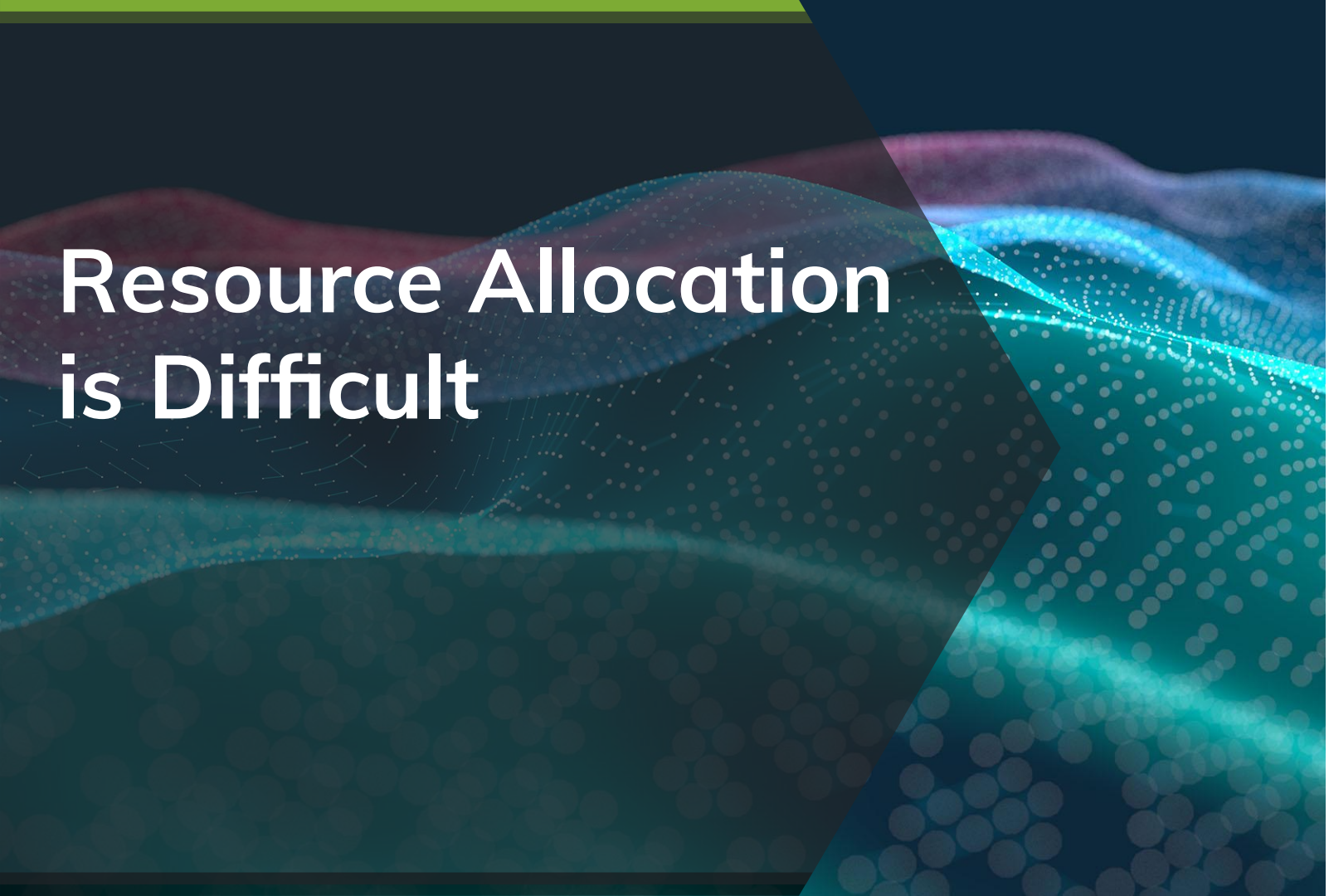
Where Does All This Run?

- 2 Ampere Mt. Jade servers
- 2 Surface Pro X laptops
- Several Nvidia Jetson TK1s
- Fujitsu FX700 (for scalable vectors)

- ~400 cores
- ~800GB of RAM

- >1 worker per machine where possible.

Resource Allocation is Difficult



Linaro

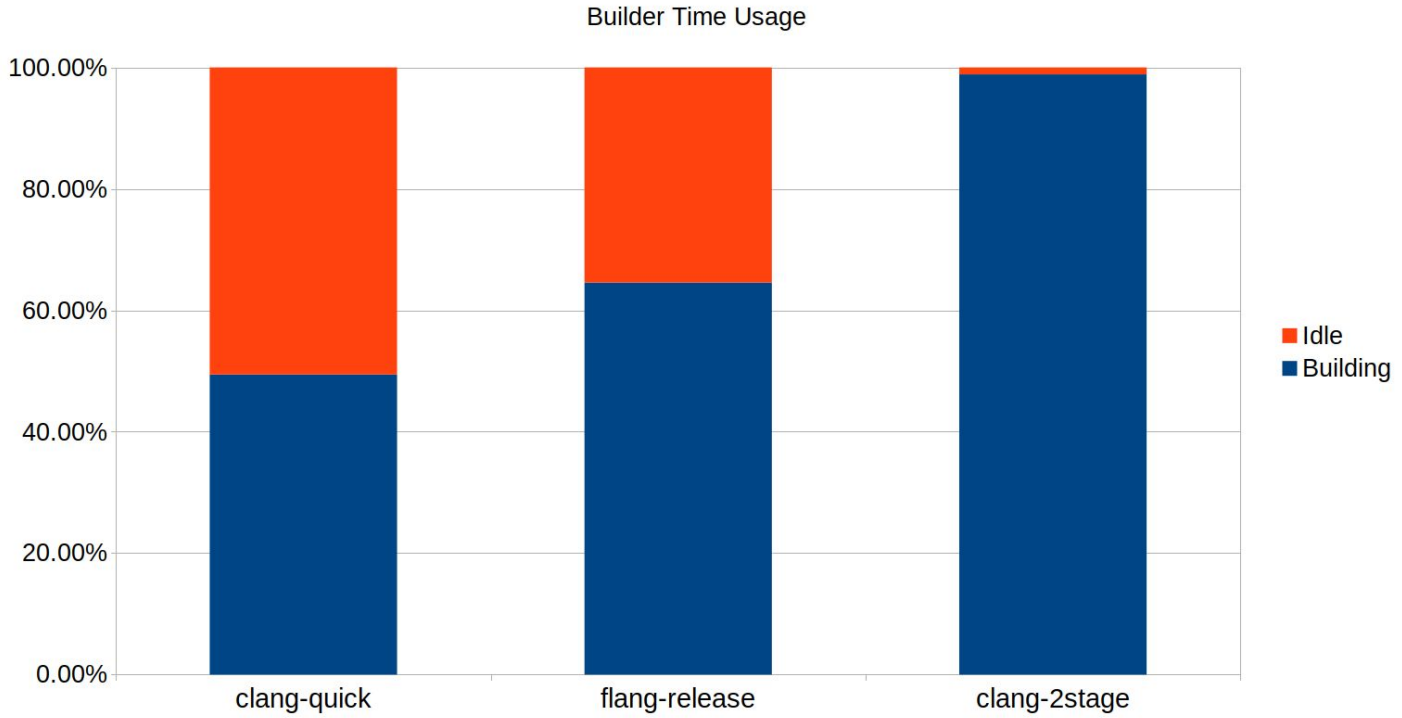
Resource Allocation

- You need do these things, in parallel:
  - Run the bot
  - Triage build issues
  - Work on fixes

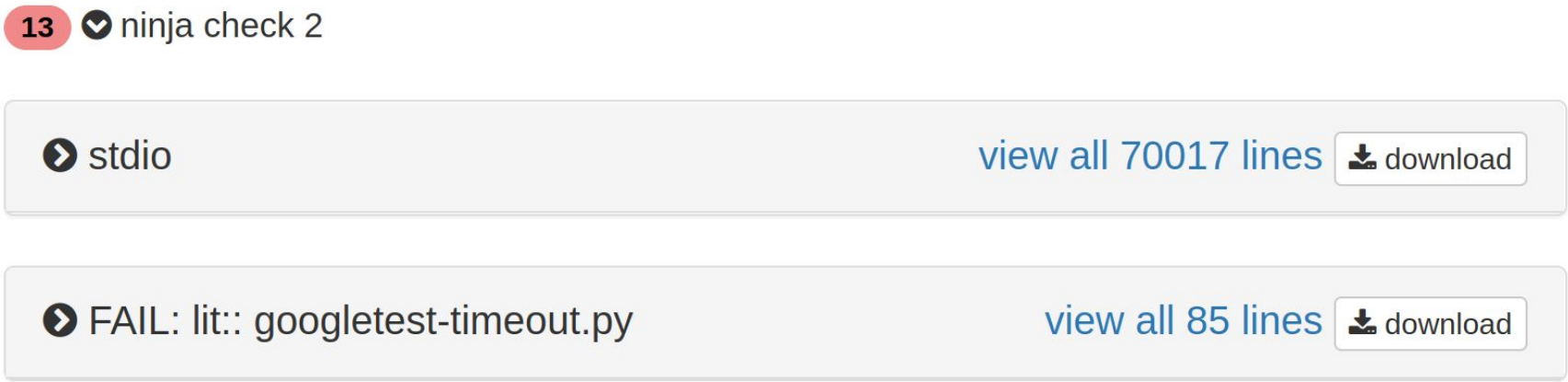
- Fixed resources instead of dynamic allocation.
  - Swings in allocation cause inconsistent tests.
  - “flaky buildbot”

- **But -** watch out for excessive idle time.

Graphs!


Builder Time Usage
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
clang-quick
flang-release
clang-2stage
Idle
Building

Flaky Bots

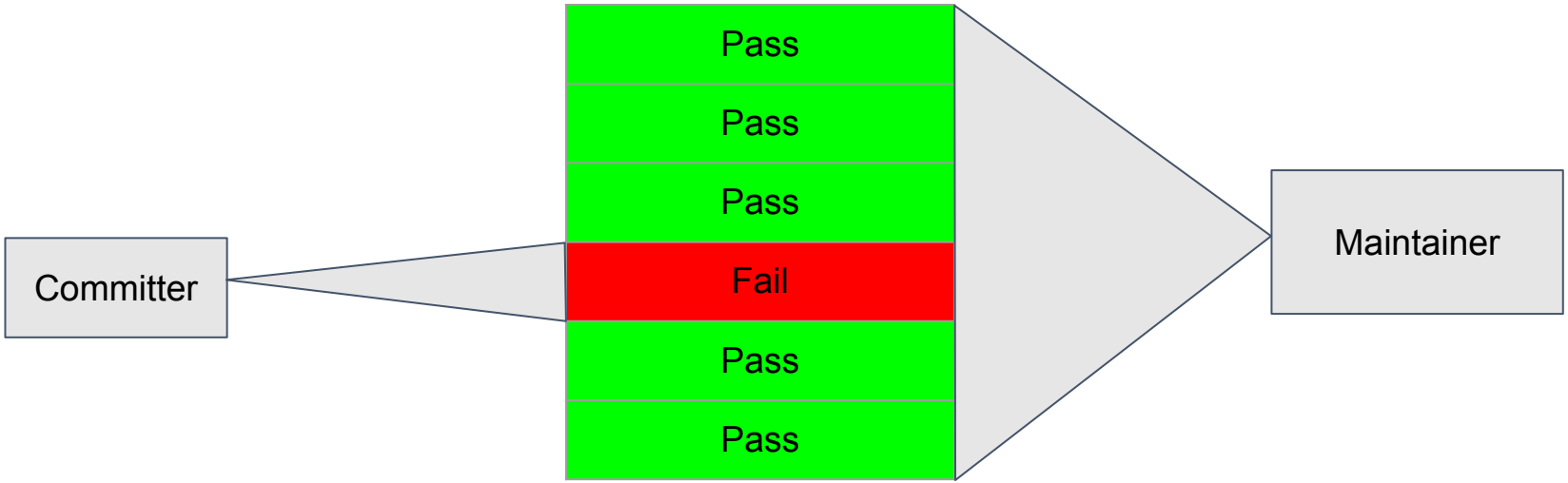


We heard you like timeouts so we timed out checking that you were able to check for a timeout.

Being a Maintainer Is Difficult

Linaro

LLVM Committer vs Bot Maintainer



- They see 1 build and only if it fails.
- You see every build.

Maintainer vs Committer Perspective

Committers are:

- Seeing only 1 build out of 100s.
- Unaware that you existed until now.
- Unfamiliar with your target.
- Less incentivised than you to fix the issue.
- Unable to access your hardware.

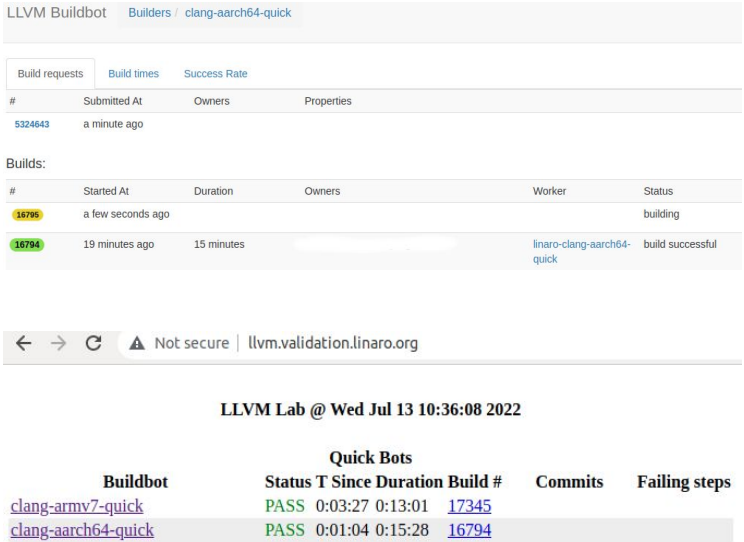
These are **not good or bad** in themselves.

The Maintainer Approach

- Remember that failure emails come out of the blue.
- Inform without making assumptions.
- Proactively notify comitters.
- 1 flaky build == a flaky bot
- Be ready to work with the committer.
- Be ready for you to do the fix instead.

Monitoring

- Add yourself to email notifications.
- Use the builder page.
- Build a status page using the API [0]



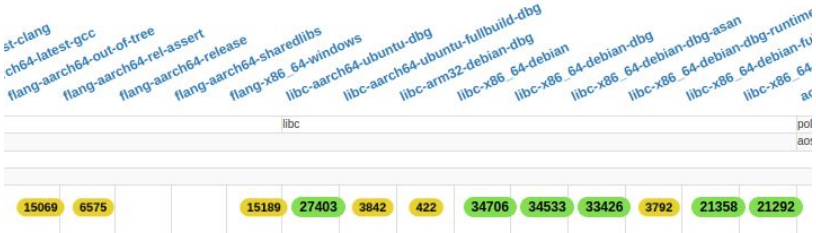
- Rotate monitoring duty across your team/community.
  - It’s not all downside, I promise.

[0] http://llvm.validation.linaro.org/ / https://git.linaro.org/toolchain/llvm/linaro-scripts.git/tree/monitor/

Linaro

Triage

- Embrace the power of knowing nothing
  - Find the change first
  - “why” comes later
- Know your categories.
  - Are all <architecture> bots broken?
  - Use the “console” view.
- Find the common changes.
- Bisect all the things!



Reverts

“Remember, it is normal and healthy to have patches reverted.”

This policy is great but not fully embraced.

- Live by it and set the example.
- Repeat it at every opportunity.
- Make reverts less surprising.

https://llvm.org/docs/DeveloperPolicy.html#patch-reversion-policy

The Bad, The Good And The Future

Linaro

Buildbot: The Bad

- Only one repository in the change list.
  - llvm-project + llvm-test-suite, you only see llvm-project changes.
- Config changes need a buildmaster restart.
  - Requesting one is easy (thanks to Galina) but there is still a delay.
- Every builder builds every commit - even if it’s known incorrect.
  - Bad for low availability workers.

Buildbot: The Good

- The patience of the LLVM community.
- Bisecting a monorepo is 1000x easier than llvm + clang.
- The web interface is clean and functional.
  - Easy to go from change list to github to Phabricator review.

Buildbot: The Future

Short term - put the basic builds in pre-commit

- Catch the obvious issues early.
- Phabricator is doing some of this, with difficulty.

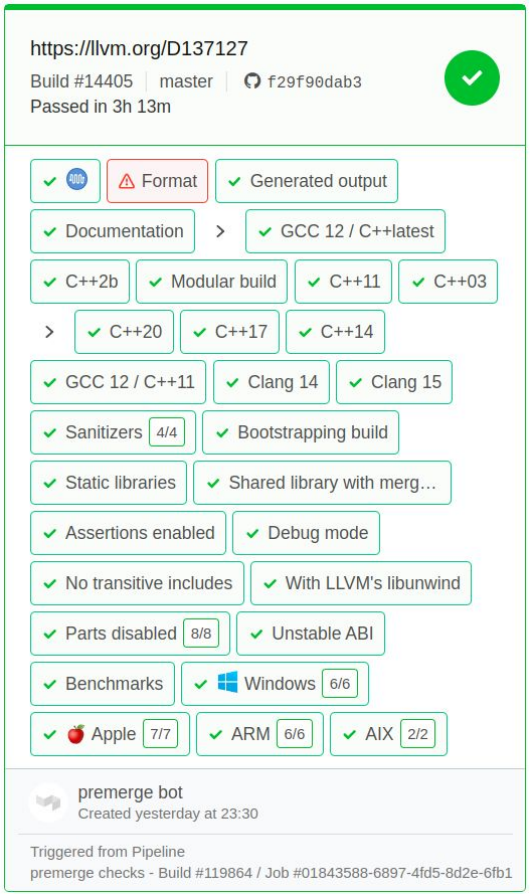
Buildbot: The Future

Long term - move **all** bots to pre-commit.

- No surprises for comitters.
- Rust's "main is always green". [0]
- Libcxx is a success story. [1]

The big question - what is the cost multiplier?

How many more builds in pre vs. post commit?



[0] https://github.com/rust-lang/homu
[1] https://www.youtube.com/watch?v=B7gB6van7Bw

Linaro

Thank you

Extra thanks to:
Linaro TCWG Team
Galina Kistanova
Everyone clicking “mark as read” on our buildbot emails :)

Backup: Cost in Engineering Time

- 4 team members on a weekly rota.
- ~1 day of the week spent on triage.
- Spikes to multiple days for big changes (e.g. opaque pointers).

Backup: Cost estimate

To run:

- 2 1-stage AArch64 SVE bots
- 2 2-stage AArch64 SVE bots

- Some details removed to fit on slide.
- On AWS Gravtion 3.
- Other clouds are available.

Cores per bot	**8**	**16**	**32**	**64**
Worst case build time 1 stage	01:42:58	00:55:19	00:33:15	00:25:14
time saved	0	00:47:39	00:22:04	00:08:01
% time saved	0.00%	46.28%	39.89%	24.11%
Worst case build time 2 stage	03:05:31	01:38:56	00:57:09	00:39:32
time saved	0	01:26:35	00:41:47	00:17:37
% time saved	0.00%	46.67%	42.23%	30.83%
Cost per year	$7,923	$14,842	$30,515	$60,485

Backup: What Do Bot Names Mean?

- Names are arbitrary but there are some patterns.

clang-arm64-windows-msvc

Building clang, on AArch64 Windows on Arm, using clang-cl.
(yes, the msvc is confusing)

lldb-aarch64-ubuntu

Building lldb on AArch64 Ubuntu using clang.
(you'd think it would have "-clang" on the end)

- For the full configuration:
  - https://github.com/llvm/llvm-zorg/blob/main/buildbot/osuosl/master/config/builders.py
  - Check the cmake stage logs from the buildbot web UI.

Backup: More Future

- Can we learn from the Linux Kernel?
  - Many configs, some more popular than others.
- Github bot to explain revert and reland process.
  - Prevent surprises and a feeling of being singled out.
  - Extend to the whole “lifecycle” of a change?
- Trigger bots from a pre-commit pull request
  - Reviewers can catch failures in review.
  - Test without hardware access.
- Automatic bisect on failure
  - Several non-buildbot systems do this.
  - Send to bot owner first in case of bad results.