



Bazel & Buildkite { Fast, Green } - Choose Two

Matthew Mackay
Staff Engineer @ Aspect Development

Who's Matt Mackay?



Staff engineer at Aspect Development

Worked in various areas of developer experience

Owner on core bazel rulesets

Won an egg and spoon race once

Aspect Development



Making Bazel easier to adopt,
operate, and optimize



github.com/aspect-build



Client Case Study



What is CI with Bazel?



Bazel performance tracking



Buildkite as a CI platform

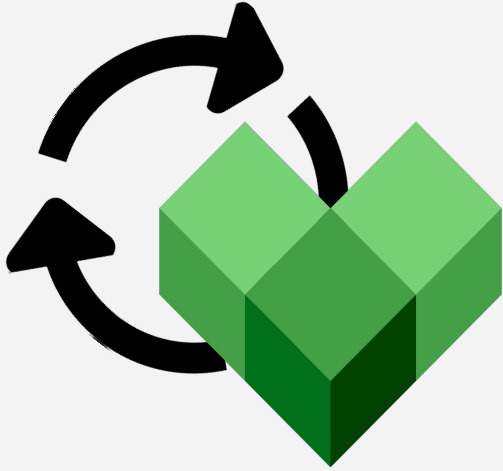
Client Case Study

Builds per week: ~ 4000 main branch, ~8000 diffs

Build time: 60 min with no optimizations

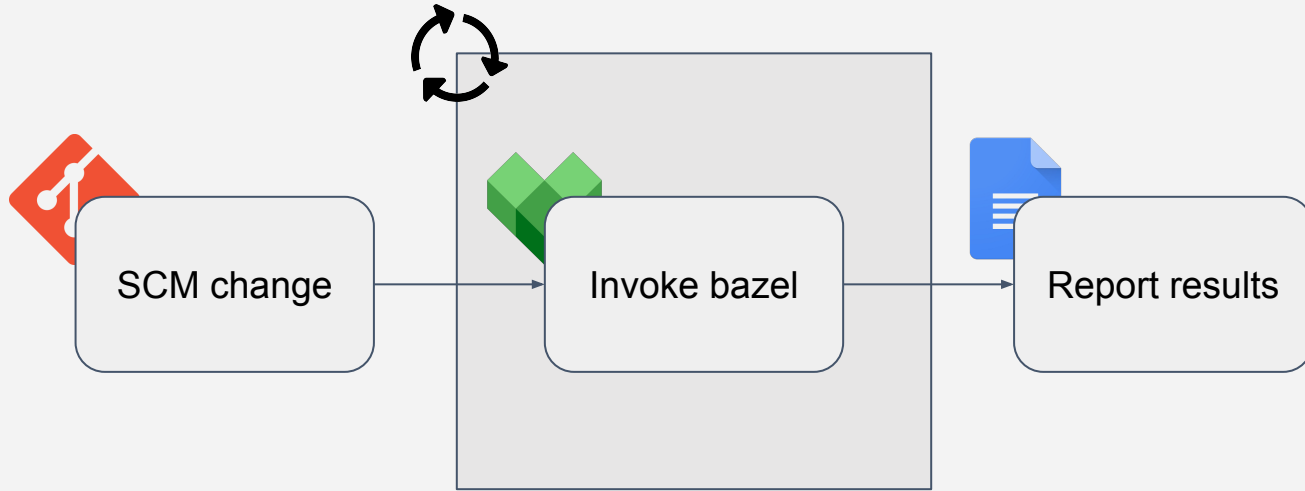
Expectations:

- Build is fast, especially for no-ops
- Quick feedback and deploy times
- Clear and actionable messaging when builds fail
- Main branch to be green and breakages reverted quickly
- Minimize machine resources cost



What is CI with Bazel?

CI is simple, Bazel does the integration





Just Invoke Bazel?



bazel test //...

Build and test everything

Rely on caching to keep times fast

Good starting point



Just Invoke Bazel?



bazel test \$affected-targets

Build and test only the targets that have been affected by a change

Bazel does significantly less work

Overhead of calculating the affected targets



Just Invoke Bazel?



Selective Testing + Q&A BazelCon Talk

https://youtu.be/9Dk7mtlm7_A

bazel-diff by Tinder

<https://github.com/Tinder/bazel-diff>

Target-determinator owned by bazel-contrib

<https://github.com/bazel-contrib/target-determinator>



Bazel Performance Tracking

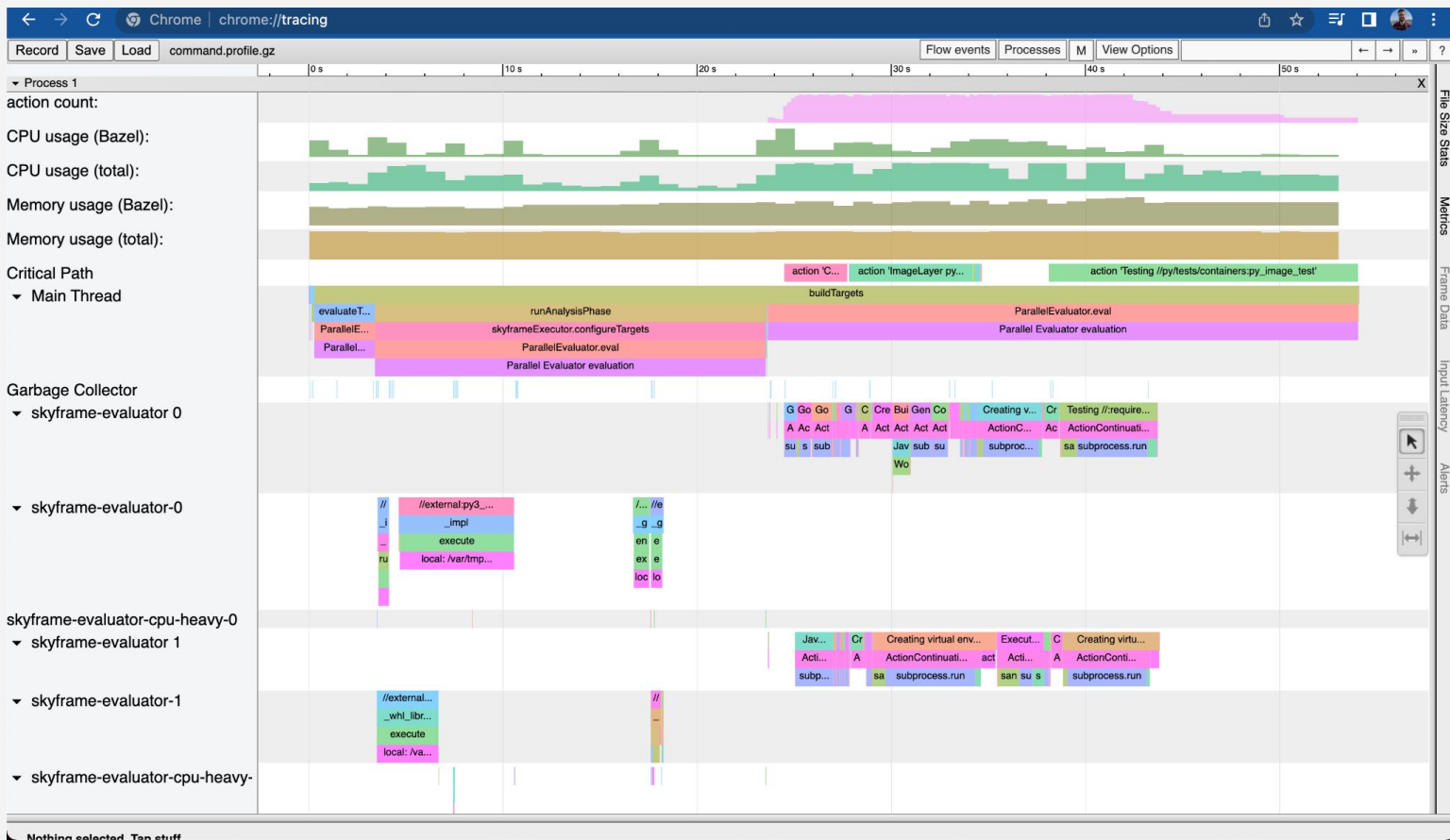


Bazel Performance Tracking



```
$ (bazel info output_base)/command.profile.gz
```

Open in `chrome://tracing`





Deterministic repository rules



Deterministic Repository Rules



Package managers may invoke “post install” scripts that do arbitrary work

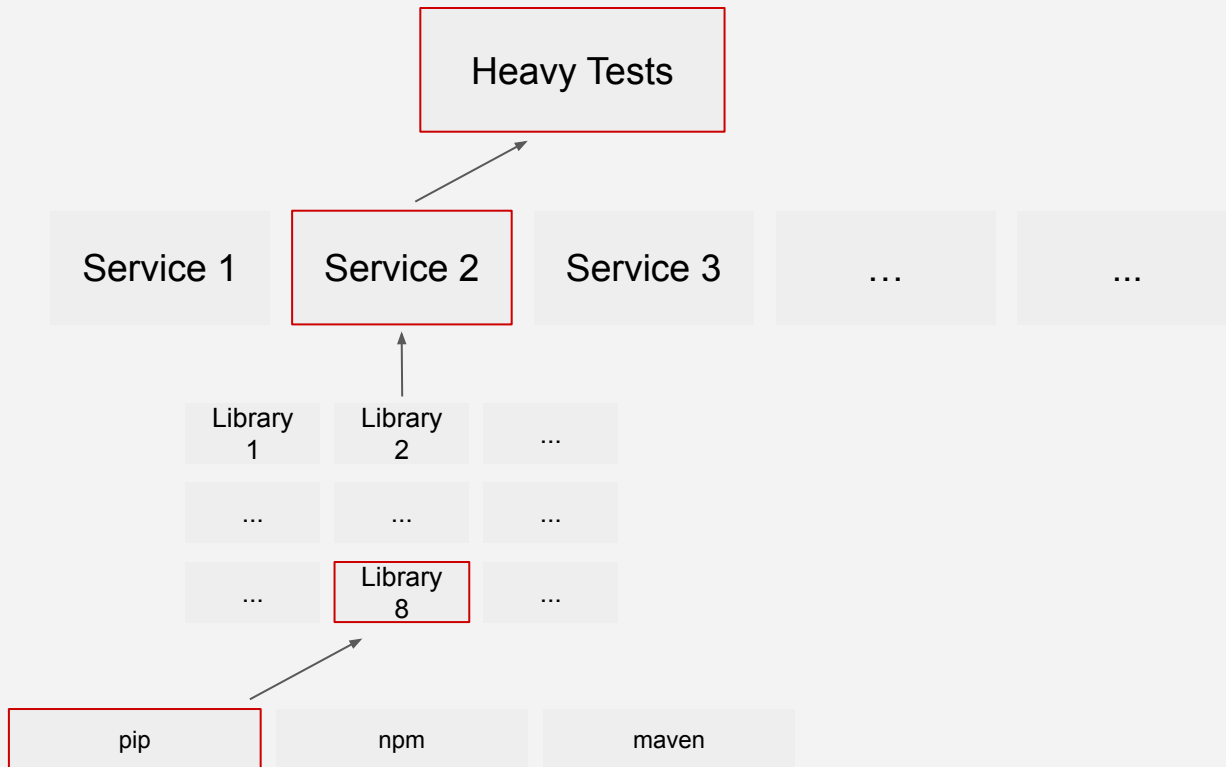
Files may contain absolute paths, timestamps or debug symbols

Environment variables changing between builds

Cause large invalidations to the build graph or continual cache misses



Deterministic Repository Rules





Deterministic Repository Rules



~/vevn1/_psycopg.cpython-38-darwin.so

587cd33d74cd553776ec0034288a13f62bd6217f287f51645f132d5f50081cd4

~/vevn2/_psycopg.cpython-38-darwin.so

83e7a3ed0f3f0208da46ab6fbf275a8f28d762a04f7e89bc804c70418fee6a10



Deterministic Repository Rules

```
> diffoscope ~/vevn1/_psycopg.cpython-38-darwin.so ~/vevn2/_psycopg.cpython-38-darwin.so
```

```
...
```

```
| -0008d800: 2f70 6970 2d69 6e73 7461 6c6c 2d30 7666 /pip-install-0vf  
| -0008d810: 6c76 3465 6e2f 7073 7963 6f70 6732 2f62 lv4en/psycopg2/b
```

```
| +0008d800: 2f70 6970 2d69 6e73 7461 6c6c 2d6c 6865 /pip-install-lhe  
| +0008d810: 7274 655f 6b2f 7073 7963 6f70 6732 2f62 rte_k/psycopg2/b
```

```
...
```



Finding non-deterministic input files



Deterministic Repository Rules



```
bazel query "kind('source file', deps(//foo))" --output xml |
xq '.query."source-file"[]."@location"' --raw-output      |
awk -F ':' '{print $1}' |
sort |
xargs shasum -a 256 {} |
tee shas.txt
```



Execution Logs

Gather execution logs from builds

```
--execution_log_json_file  
--execution_log_binary_file
```

Binary log output can be processed with the execution log parser

<https://github.com/bazelbuild/bazel/blob/master/src/tools/execlog>



Execution Logs

JSON output can be processed with jq filters

Get all inputs for a uncached test given its target

```
select(
  .remoteCacheHit == false and
  (.progressMessage | contains("//foo:test"))
) | .inputs
```



Fixing all the repository rule outputs



Deterministic Repository Rules

Setting environment variables for stable timestamps

```
ZERO_AR_DATE=1
```

```
SOURCE_DATE_EPOCH=624795867
```




Deterministic Repository Rules

Turn off debug symbols where appropriate

```
CFLAGS="-g0"
```



Deterministic Repository Rules



Remove input files that aren't needed

```
npm_install(  
  name = "npm",  
  args = [  
    "&&",  
    "rm -rf ./node_modules/node-sass/build",  
  ],
```



Remote Caching

CI Agent 1



`bazel build $targets`

CI Agent 2



`bazel build $targets`

read + write

read + write





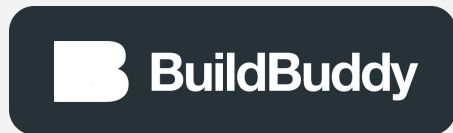
Bazel Performance Tracking



Use a remote caching service to share build results across agents

OSS Cache implementations such as buchgr/bazel-remote

Fully managed solutions





Analysis Cache

INFO: Build option `--define` has changed, discarding analysis cache.



Bazel Performance Tracking



INFO: Build option `--define` has changed, discarding analysis cache.

Can add significant overhead to bazel commands

Find and prevent churning of analysis cache

Flags across verbs in `.bazelrc` are consistent



INFO: Build option `--define` has changed, discarding analysis cache.

Use common where appropriate

```
common --incompatible_foo
```

Place other flags on each verb

```
test --incompatible_foo
```

```
query --incompatible_foo
```

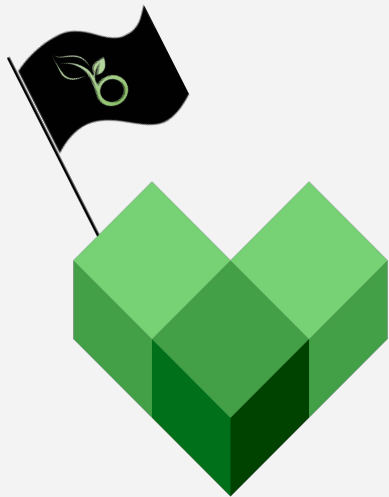



Bazel Performance Tracking



If running `coverage`, consider running `test` with these flags

```
--collect_code_coverage  
--test_timeout=300,600,1200,3600  
--instrumentation_filter=^//
```



Fun with flags



Bazel Performance Tracking



No scanning of output or external files when checking for changes

```
--noexperimental_check_output_files
```

```
--noexperimental_check_external_repository_files
```



Store merkle tree calculations to improve remote cache speed

```
--experimental_remote_merkle_tree_cache  
--experimental_remote_merkle_tree_cache_size
```



Build without the bytes

```
--remote_download_minimal
```

Expands to

```
--nobuild_runfile_links  
--experimental_inmemory_jdeps_files  
--experimental_inmemory_dotd_files  
--remote_download_outputs=minimal
```



Bazel Performance Tracking



Constrain users so we can hit our SLA

```
--test_timeout_filters=-eternal
```

Paper over flaky tests

```
--flaky_test_attempts=n
```



Bazel Performance Tracking



Be vigilant when landing changes with a big blast radius

Perhaps limit landing these to off peak times

Group these changes up where possible



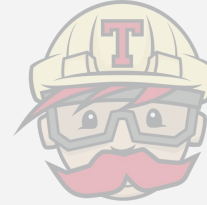
Bazel Performance Tracking



Gather timing metrics from each build stage

Track metrics over time

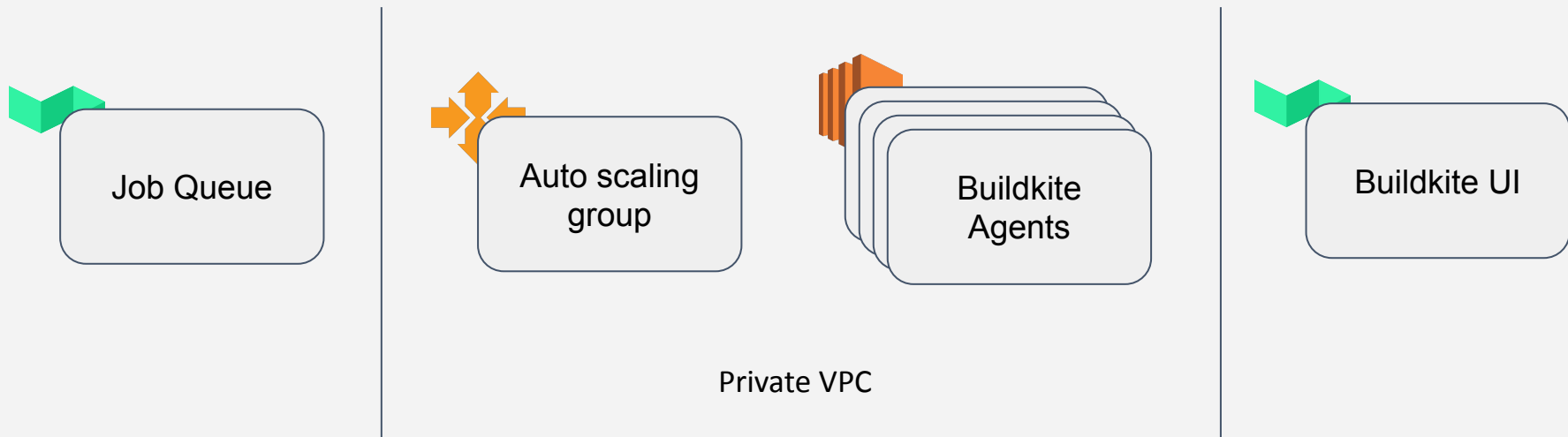
Setup SLAs, SLOs and SLIs to keep on top of CI performance



Buildkite



Running agents on AWS EC2 via ASG for dynamic scaling of compute



Elastic Stack Properties

`ScaleOutFactor`

Controls a decimal factor to speed up or slow down agent scaling

`ScaleInIdlePeriod`

Number of seconds and agent remains idle before it's considered for termination



Elastic Stack Properties

MinSize

The minimum number of agents that will be available at any given time

MaxSize

The Maximum number of agents that will be available



Elastic Stack Properties

OnDemandPercentage

Percentage of instances that should be launched as on-demand

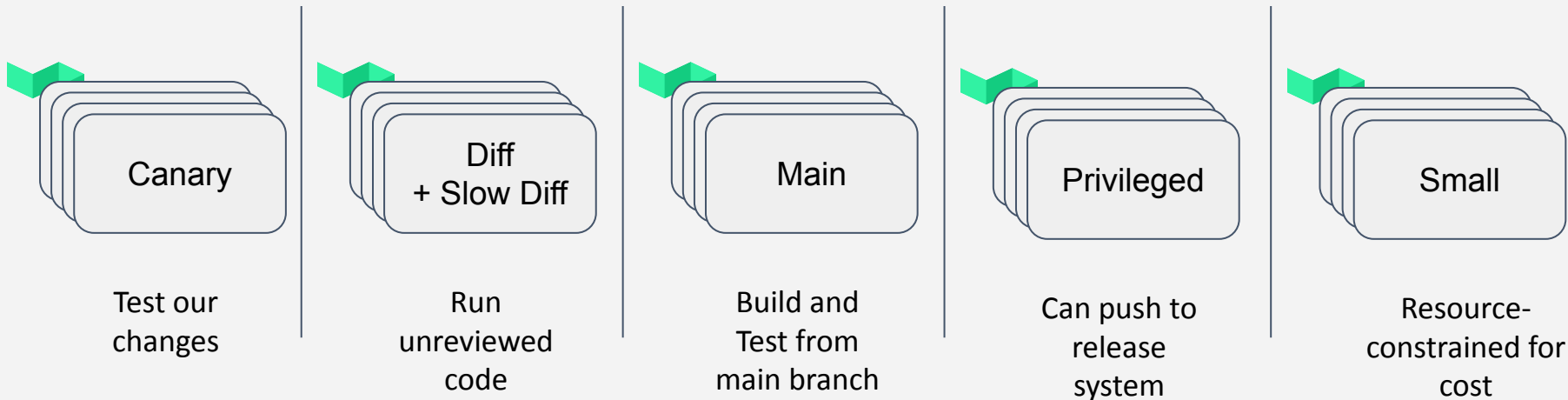


Buildkite

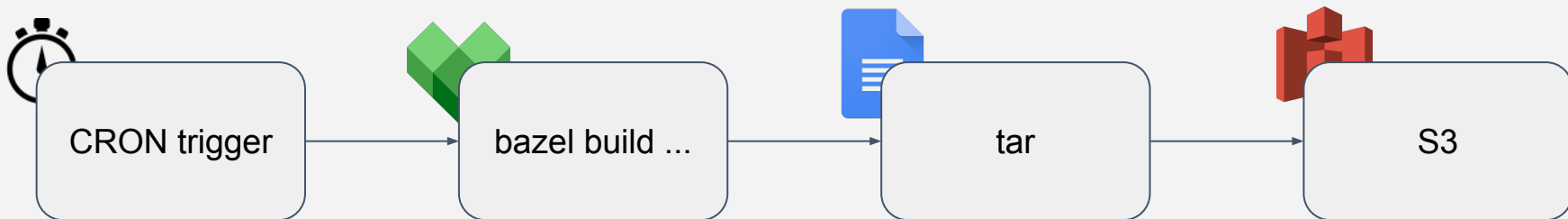


Local NVMe disks Raid 0 (high storage I/O performance)	Target pattern was determined by bazel-diff with some custom logic layered on	AMI contains system level dependencies but not sources
i3en series for good all around resources	EC2 Not k8s	Air-gapped

Different agent pools to guarantee performance, stability, ACLs



Bootstrap agents from a tar for repository caches





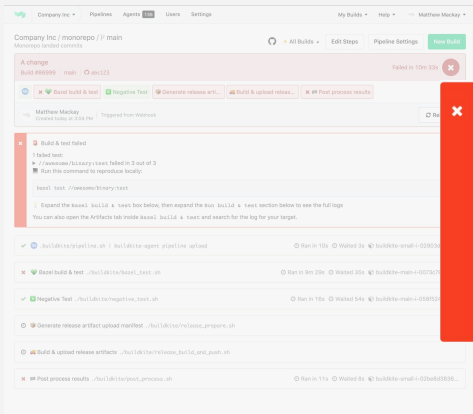
Signpost failures with annotations



Central team must pre-empt user questions

Consume build events to annotate the build with actionable info

Search and reduce log spam



 **Build & test failed**

1 failed test:

- ▶ `//awesome/binary: test` failed in 3 out of 3

 Run this command to reproduce locally:

```
bazel test //awesome/binary: test
```

 Expand the Bazel `build & test` box below, then expand the `Run build & test` section below to see the full logs

You can also open the Artifacts tab inside Bazel `build & test` and search for the log for your target.

Stream the events via the Aspect CLI to generate annotations

Aspect / Pro Plugins / streaming2 🔄 All Builds Edit Steps Pipeline Settings New Build

streaming annotation Running for 13s

Build #121 | streaming2 | aaac393 (Pull Request #18)

🟢 Bazel Test 🟢 Build CLI and Plugins

 Alex Eagle
Created today at 3:34 PM | Triggered from Webhook Cancel

Testing might fail

1 failed test:

- ▶ //plugins/summary/tests/tests_flake/testdata:flaky_sh_test

Run this command to reproduce locally:

```
bazel test //plugins/summary/tests/tests_flake/testdata:flaky_sh_test
```

💡 You can also open the Artifacts tab inside Bazel build & test and search for the log for your target.

🟢 🟢 Bazel Test echo '--- :bazel: Testing' && bazel test --test_tag_filters=-expected_to_fail //... 🕒 Ran in 3s 🕒 Waited 2s 👤 alexeagle-1

🟢 🟢 Build CLI and Plugins echo '--- :bazel: Building CLI and Plugins' && bazel build //:cli-plugins-pro ... 🕒 Running for 8s 🕒 Waited 7s 👤 alexeagle-1

Log Artifacts 1 Timeline Environment ✖ Cancel



Shard work between agents

Do work across multiple agents in parallel to break up large test targets

✓ 1/5	♥ Bazel build & test	⌚ Ran in 49s ⌚ Waited 1s 📦 i-034275025f7ddf30d
✓ 2/5	♥ Bazel build & test	⌚ Ran in 41s ⌚ Waited 9s 📦 i-034599eee0499b814
✓ 3/5	♥ Bazel build & test	⌚ Ran in 41s ⌚ Waited 1s 📦 i-0e5abf1f118433fec
✓ 4/5	♥ Bazel build & test	⌚ Ran in 42s ⌚ Waited 1s 📦 i-0efa1618aa3713d0d
✓ 5/5	♥ Bazel build & test	⌚ Ran in 2m 10s ⌚ Waited 10s 📦 i-0a66e0bd415b11d3d



User impact

User Impact

60 min -> ~5 min for diff

Centrally managed CI - Users don't have to think about CI setup for their services

One Oncall engineer monitors alerts and failures, keeping main line branch green

Flaky tests get quarantined quickly(`--flaky_test_attempts=3`)

Bazel profiles monitored to keep critical path fast as possible



Links and References

Great resource for build tooling

<https://reproducible-builds.org/>

Diffoscope

<https://diffoscope.org/>

Selective Testing + Q&A BazelCon Talk

https://youtu.be/9Dk7mtlm7_A

bazel-diff by Tinder

<https://github.com/Tinder/bazel-diff>

Target-determinator owned by bazel-contrib

<https://github.com/bazel-contrib/target-determinator>



Getting started with Bazel and CI?

Talk to us about scaling your CI system with Bazel,
or about our managed CI offering!



Bazel & Buildkite

{ Fast, Green } – Choose Two

Thanks!

Matthew Mackay
Staff Engineer @ Aspect Development

