

The Life of an Open-Source Project

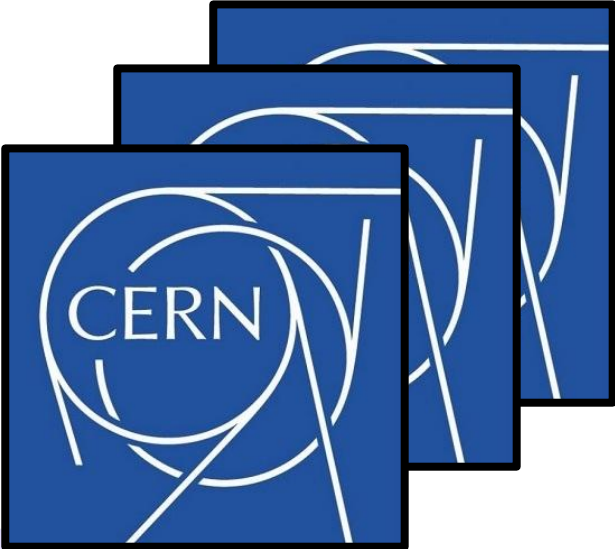
[David Garcia Quintas](#)

Xoogler, gRPC C Core Team

dgquintas@gmail.com



How I Got Here



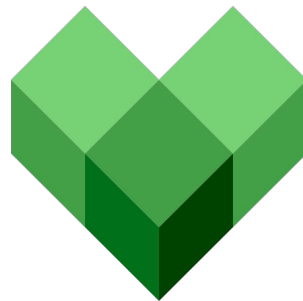
Agenda

- **Why** develop [a new project] in open-source.
- **What** is being developed: gRPC.
- **How** is it being developed: the process.

A sampling of OSS at Google



A sampling of OSS at Google



The image features a white background with decorative purple geometric patterns in the corners. These patterns consist of overlapping triangles and hexagons in various shades of purple, creating a modern, abstract look. The central focus is the text 'GRPG' in a bold, teal, sans-serif font. The letter 'G' on the left has an upward-pointing arrow integrated into its top curve, and the 'G' on the right has a downward-pointing arrow integrated into its bottom curve.

GRPG

Why?

- **Google has had 4 generations of internal RPC systems, called Stubby**
 - All production applications and systems built using microservices connected by RPCs
 - Over 10^{10} RPCs per second, fleetwide
 - APIs for C++, Java, Python, Go
 - Not suitable for open-source community (Tight coupling with internal tools)
 - Not suitable for open Internet (proprietary wire protocol won't work with firewalls, etc)



What?

- **gRPC Remote Procedure Calls.**
- High performance, open source, general purpose, standards-based, feature-rich RPC framework.
 - **HTTP-2** based transport
 - Standards-based flow-control, auth, LB, etc.
- Developed by Google, donated to CNCF
 - Development is Github-first
 - github.com/grpc

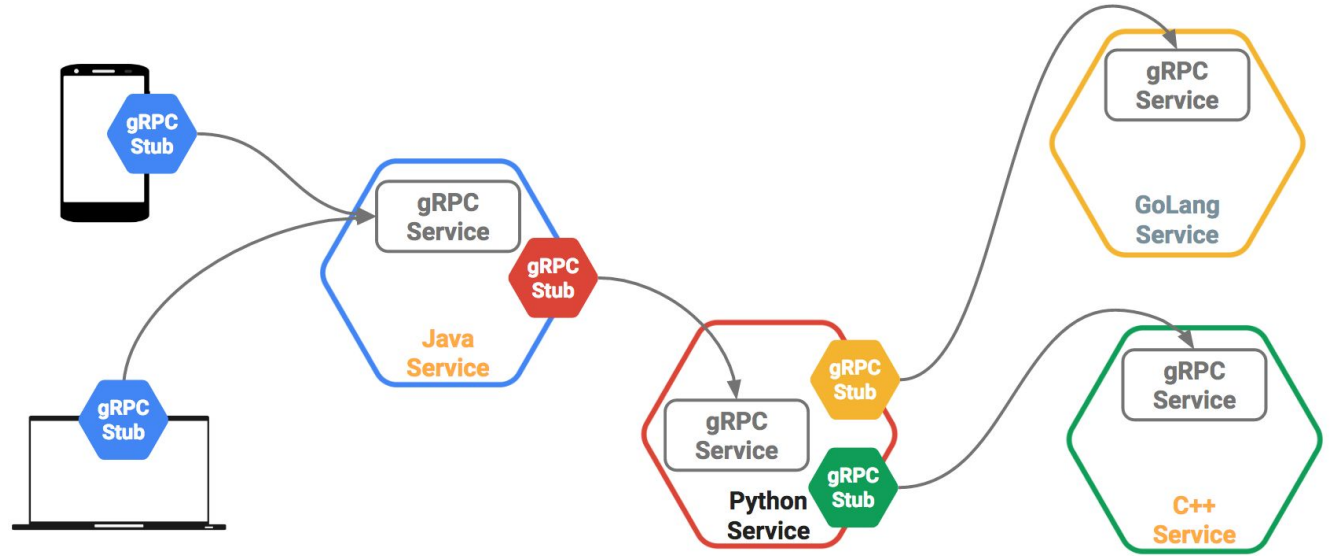
The logo for gRPC, featuring the letters 'gRPC' in a teal, sans-serif font. The 'g' has a curved arrow pointing up and to the right, and the 'C' has a curved arrow pointing down and to the right, suggesting a cycle or process.

Service definitions and client libraries

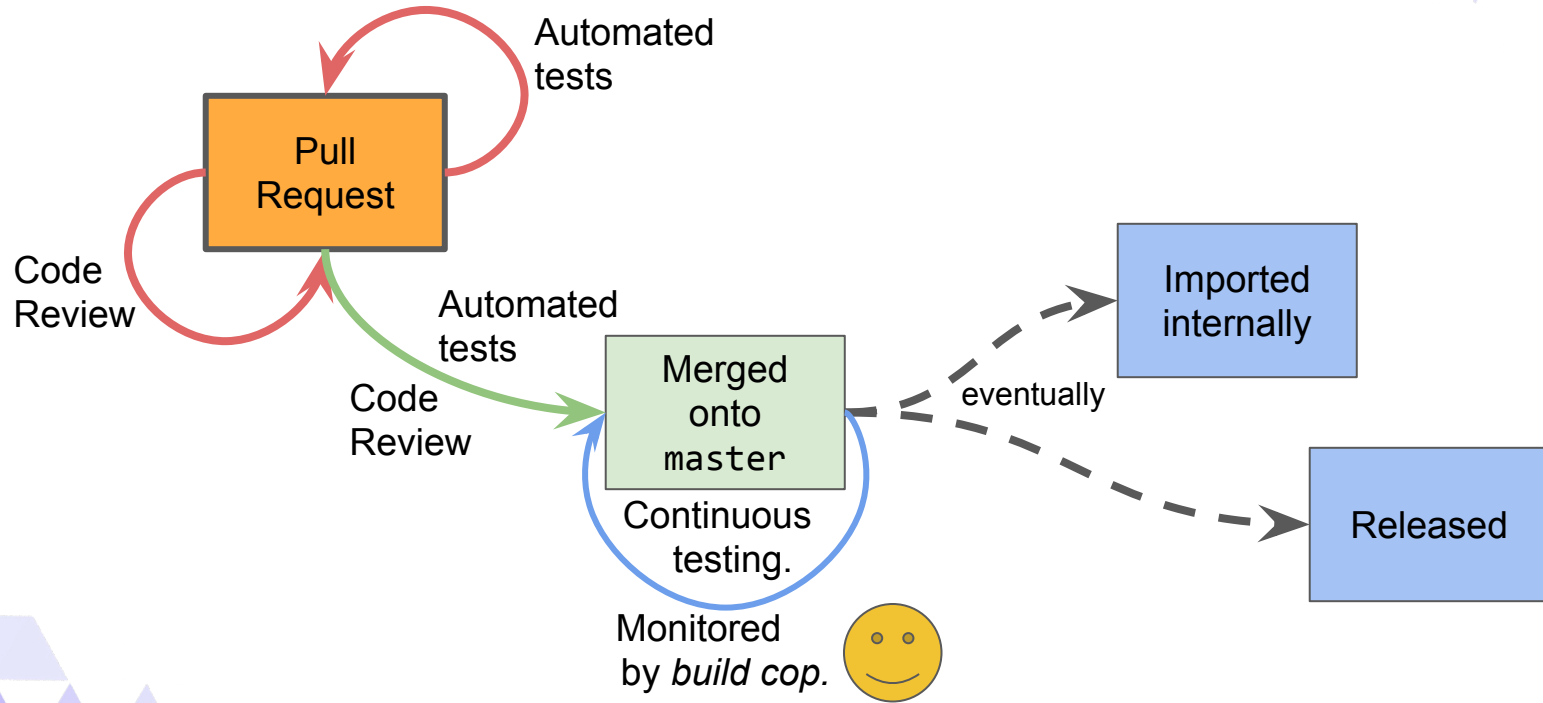
- Java
- Go
- C/C++
- C#
- Node.js
- PHP
- Ruby
- Python
- Objective-C

More Languages...

- Swift
- Haskell
- Rust
- Typescript
-



Development Process





RESPECT THE PROCESS

Development Process: Code Review

```
in the \a grpc_lb_policy struct. */  
int grpc_lb_policy_pick_locked(grpc_lb_policy* policy,  
                               const grpc_lb_policy_pick_args* pick_args,  
                               grpc_connected_subchannel** target,
```

Resolved — Hide 5 comments

line 163

9 MONTHS AGO



Should we change this to return a `RefCountedPtr`?

This might be easier once [#13857](#) is merged. We should probably discuss which of these PRs we should try to merge first.



I've just merged [#13857](#), so you'll have to merge the changes into this PR.



Still something to be done on a PR separate from this one.



Why not do it as part of this PR? We're touching all of the callers of `connectedSubchannel` anyway. It seems like it would be pretty trivial to do here.



Ended up being a good idea. Done.

Follow up...



Development Process: **Testing**

32 of 38 checks passed

| | | |
|---|--|-------------------------|
| ✗ | Bazel UBSAN build for C/C++ Kokoro build finished | Details |
| ✓ | Android (Internal CI) Kokoro build finished | Details |
| ✓ | Artifact Build Linux (internal CI) Kokoro build finished | Details |
| ✓ | Artifact Build MacOS (internal CI) Kokoro build finished | Details |
| ✓ | Artifact Build Windows (internal CI) Kokoro build finished | Details |
| ✓ | Asan C (internal CI) Kokoro build finished | Details |
| ✓ | Asan C++ (internal CI) Kokoro build finished | Details |
| ✓ | Basic Tests C Linux [dbg] (internal CI) Kokoro build finished | Details |
| ✓ | Basic Tests C Linux [opt] (internal CI) Kokoro build finished | Details |

Development Process: **Testing**

Tens of thousands of tests run **per PR** *and continuously on master*.

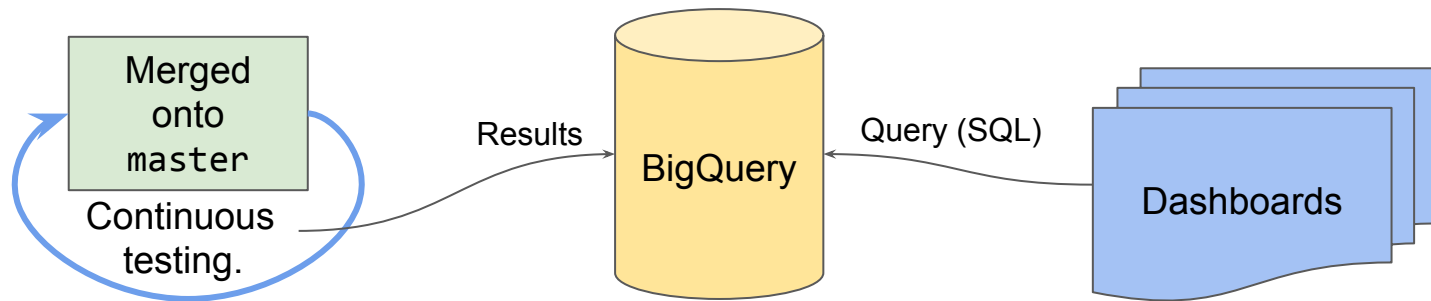
{windows, mac, linux} x {bazel, make} x
{asan, msan, tsan, ubsan} x {opt, dbg} x
{C Core tests, C++ tests}

+

microbenchmarks + binary size
+ {Python, Ruby, iOS, PHP, C#, Node.js}
+ **interop** (all client/server pairs of Java, Go,
C-wrapping languages).

Development Process: Build Cop

Monitors increases in test flakiness + performance from the master CI runs

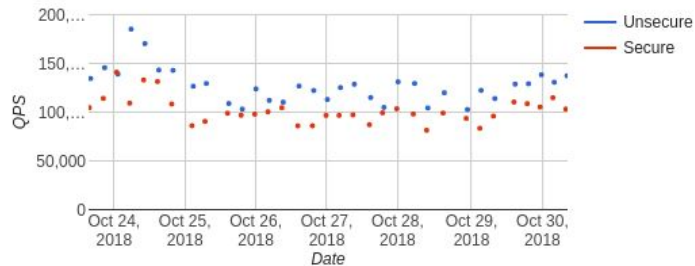


Total Failures over last 24 hours (sorted by %, then by time) (use `go/grpc-test-failure-history` for individual test history)

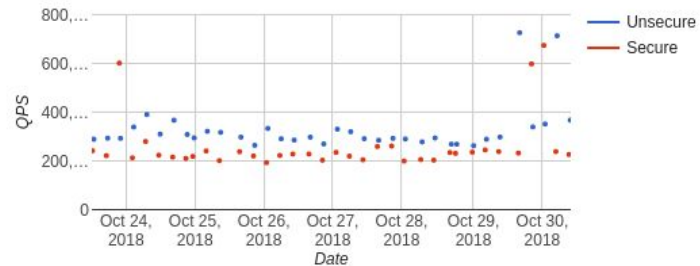
| test_binary | pct_failed | job_name | latest_failing_build_id | latest_timestamp_M |
|---|------------|--|--------------------------|-----------------------|
| tools/run_tests/helper_scripts/run_grpc-node.sh | 40 | grpc/core/master/macos/grpc_basictests_dbg | [2596, 2594, 2593, 2588] | 2018-10-29 19:41:23-0 |
| tools/run_tests/helper_scripts/run_grpc-node.sh | 37.5 | grpc/core/master/macos/grpc_basictests_opt | [2096, 2094, 2092] | 2018-10-29 18:42:44-0 |
| objc-tests | 14.29 | grpc/core/master/macos/grpc_basictests_opt | [2092] | 2018-10-29 10:48:17-0 |
| h2_full_test hpack_size | 10 | grpc/core/master/macos/grpc_basictests_dbg | [2590] | 2018-10-29 09:04:07-0 |
| py27_native.test.unit_channel_ready_future_test.ChannelReadyFutureTest | 5.88 | grpc/core/master/linux/grpc_basictests_multilang | [3075] | 2018-10-30 02:33:19-0 |
| client_lb_end2end_test --gtest_filter=SubchannelForceCreation/ClientLbEnd2endWithParamTest.PickFirstManyUpdates | 1.79 | grpc/core/master/linux/sanitizer/grpc_cpp_tsan | [1251] | 2018-10-30 04:37:39-0 |
| end2end_test --gtest_filter=End2end/End2endTest.ClientCancelsRequestStream | 0.83 | grpc/core/master/linux/sanitizer/grpc_cpp_asan | [1634] | 2018-10-29 12:18:17-0 |

Development Process: Build Cop

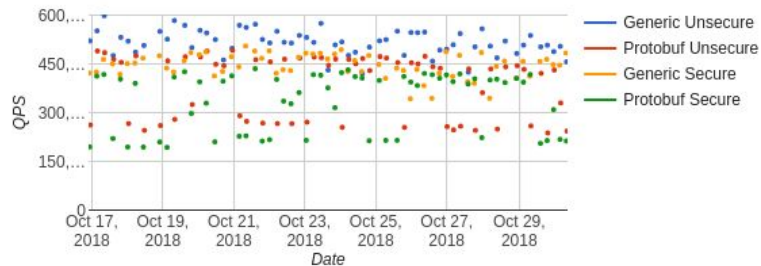
C++ Unary unconstrained protobuf throughput QPS (8 core client vs 8 core server)



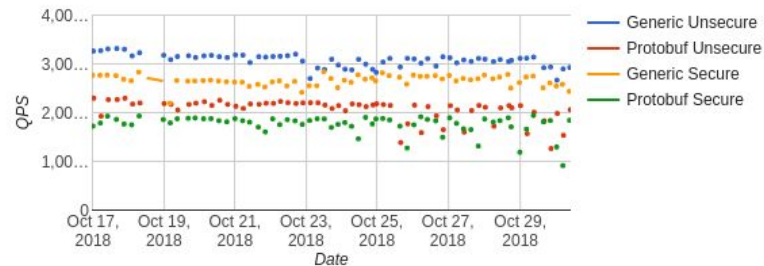
C++ Unary unconstrained protobuf throughput QPS (32 core client vs 32 core serv...)



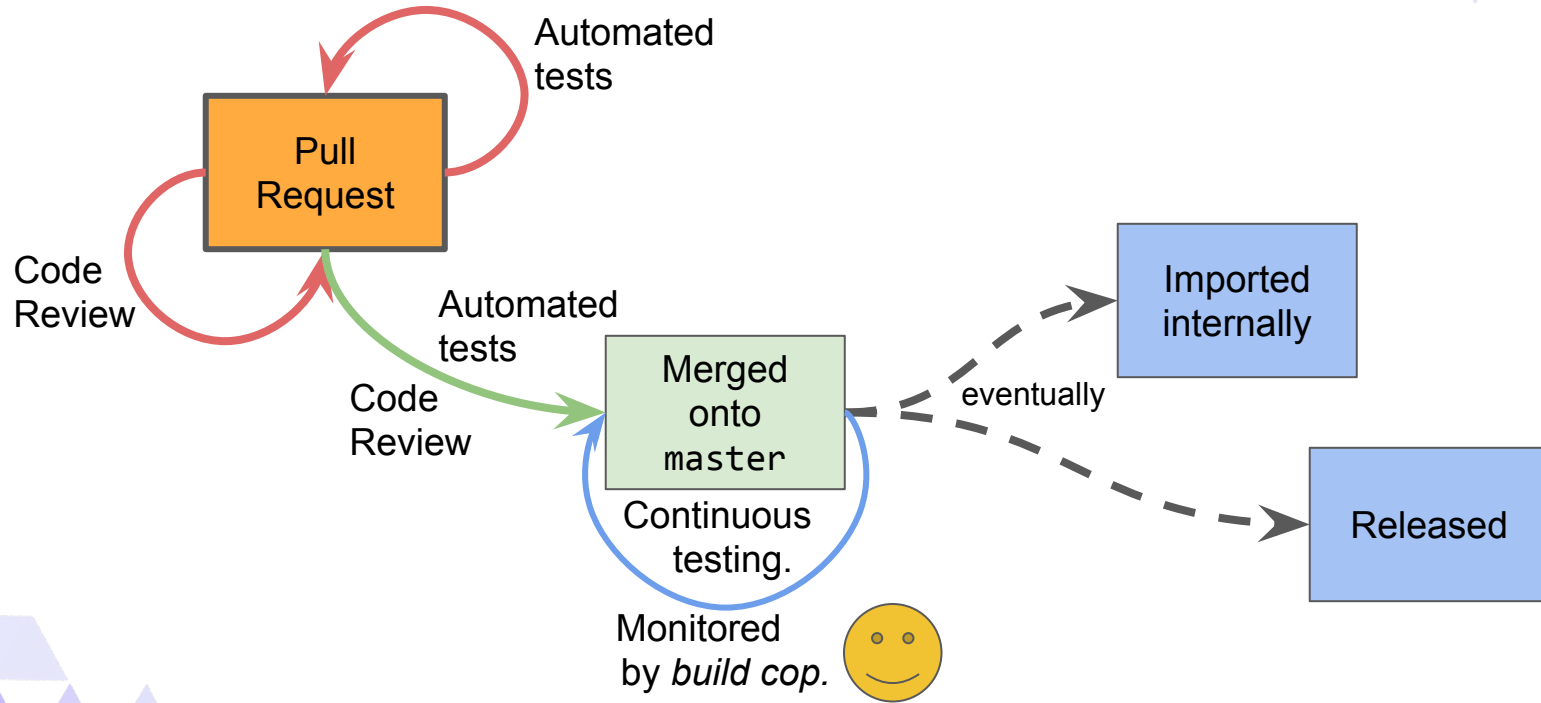
C++ Streaming unconstrained throughput QPS (8 core client vs 8 core server)



C++ Streaming unconstrained throughput QPS (32 core client vs 32 core server)

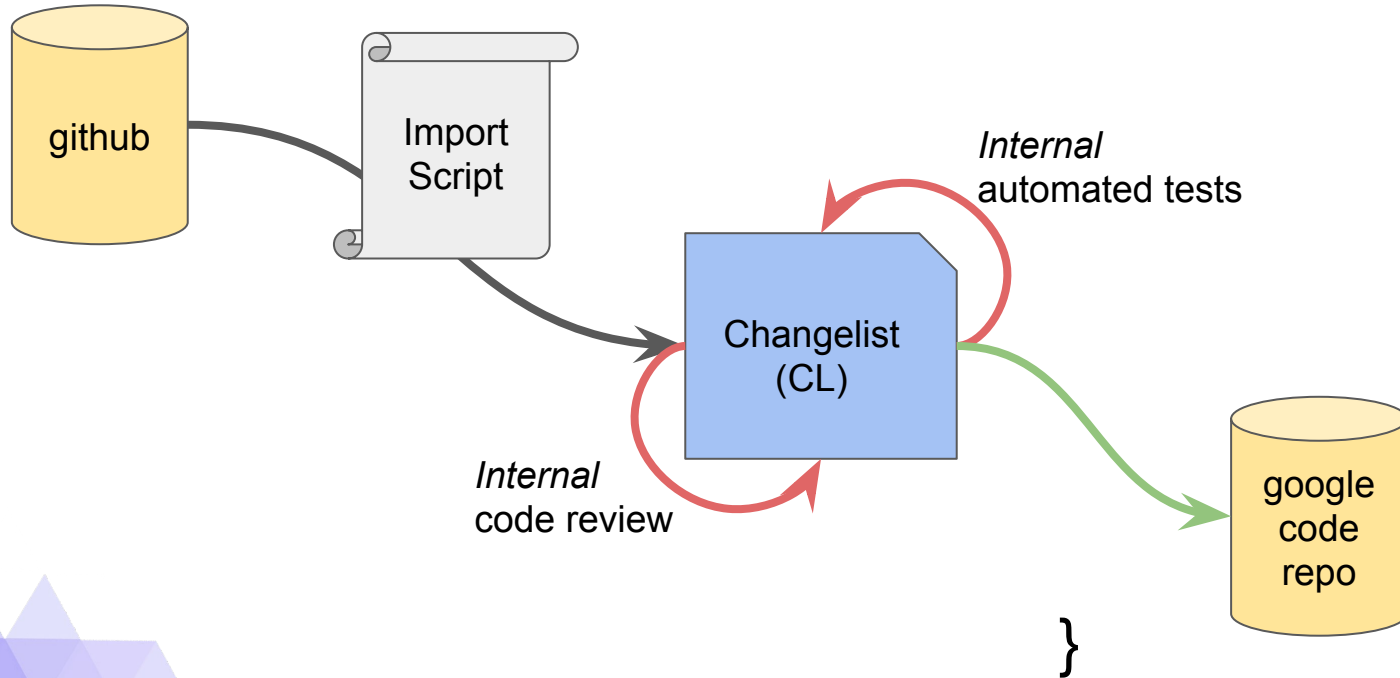


Development Process

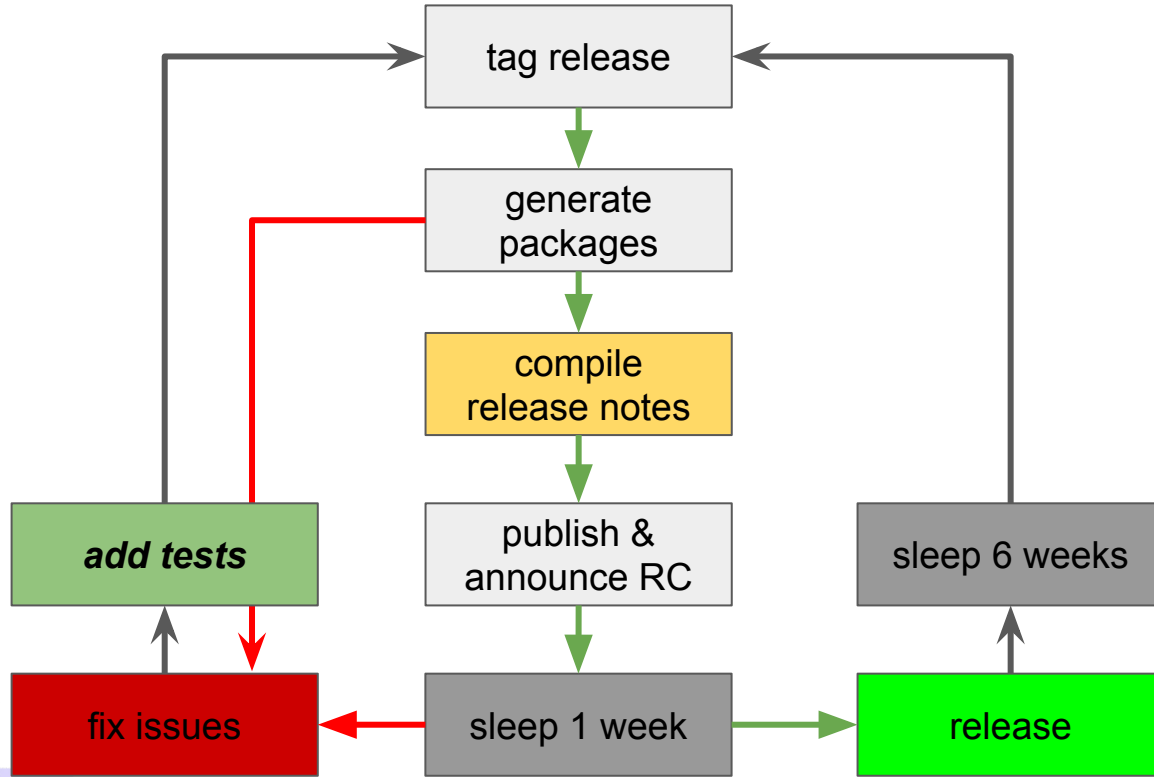


Development Process: **Internal Import**

```
while (next importer in rotation) {
```



Development Process: **Releases**



Development Process: Releases/Release Notes

Change pick_first to not unref unselected subchannels.

#16342

Merged

markdroth mer

Conversation 45



markdroth comm

Based on #16306.

This eliminates the RequestResolu

@AspirinSJL and your pairs of know

This change is

Release v1.16.0

v1.16.0 98457b7

Latest release

Verified

stanley-cheung released this 7 days ago · 439 commits to master since this release

Assets 3

grpc_objective_c_plugin-1.16.0-macos-x86_64.zip 621 KB

Source code (zip)

Source code (tar.gz)

This is the 1.16.0 release (gao) of gRPC Core.

Please see the notes for the previous releases here: <https://github.com/grpc/grpc/releases>. Please consult <https://grpc.io/> for all information regarding this product.

This release contains refinements, improvements, and bug fixes, with highlights listed below.

Core

- Keepalive watchdog firing should return status UNAVAILABLE. (#16764)
- Set TCP_USER_TIMEOUT socket option for linux. (#16419)
- When using c-ares, resolve ip literals and Windows localhost on our own. (#16420)
- Turn loading system root certificate as default. (#16536)
- Change pick_first to not unref unselected subchannels. (#16342)

1s on Aug 27

+126 -224

+ 😊 ...

Reviewers

dgquintas ✓

AspirinSJL ✓

Assignees

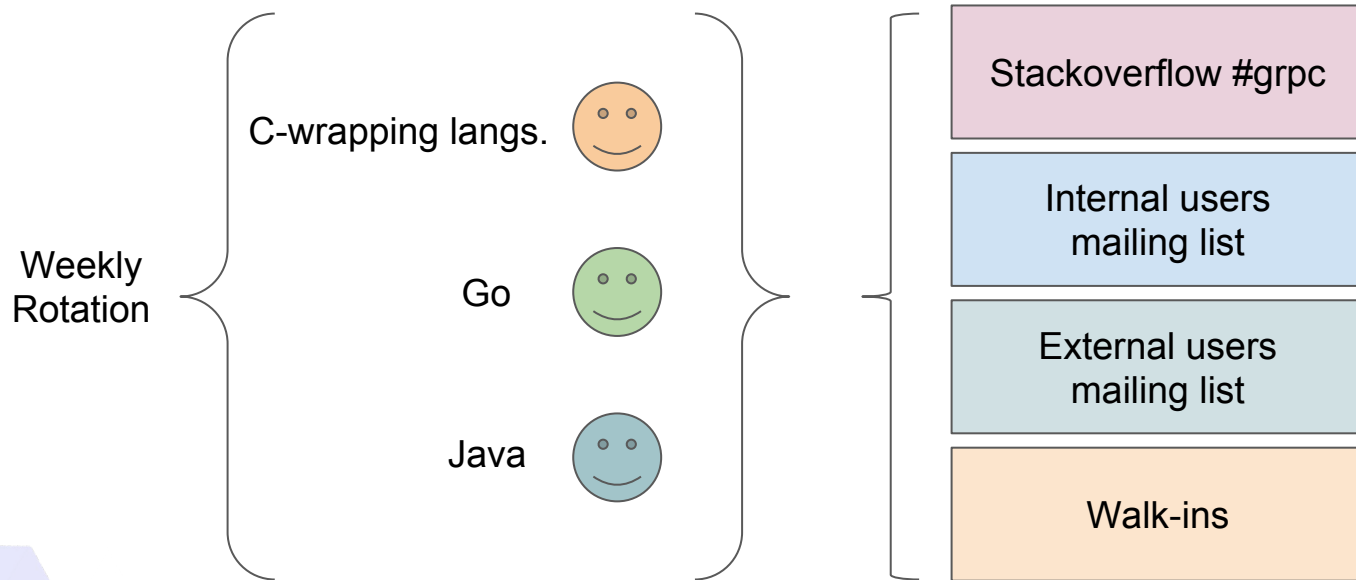
No one assigned

Labels

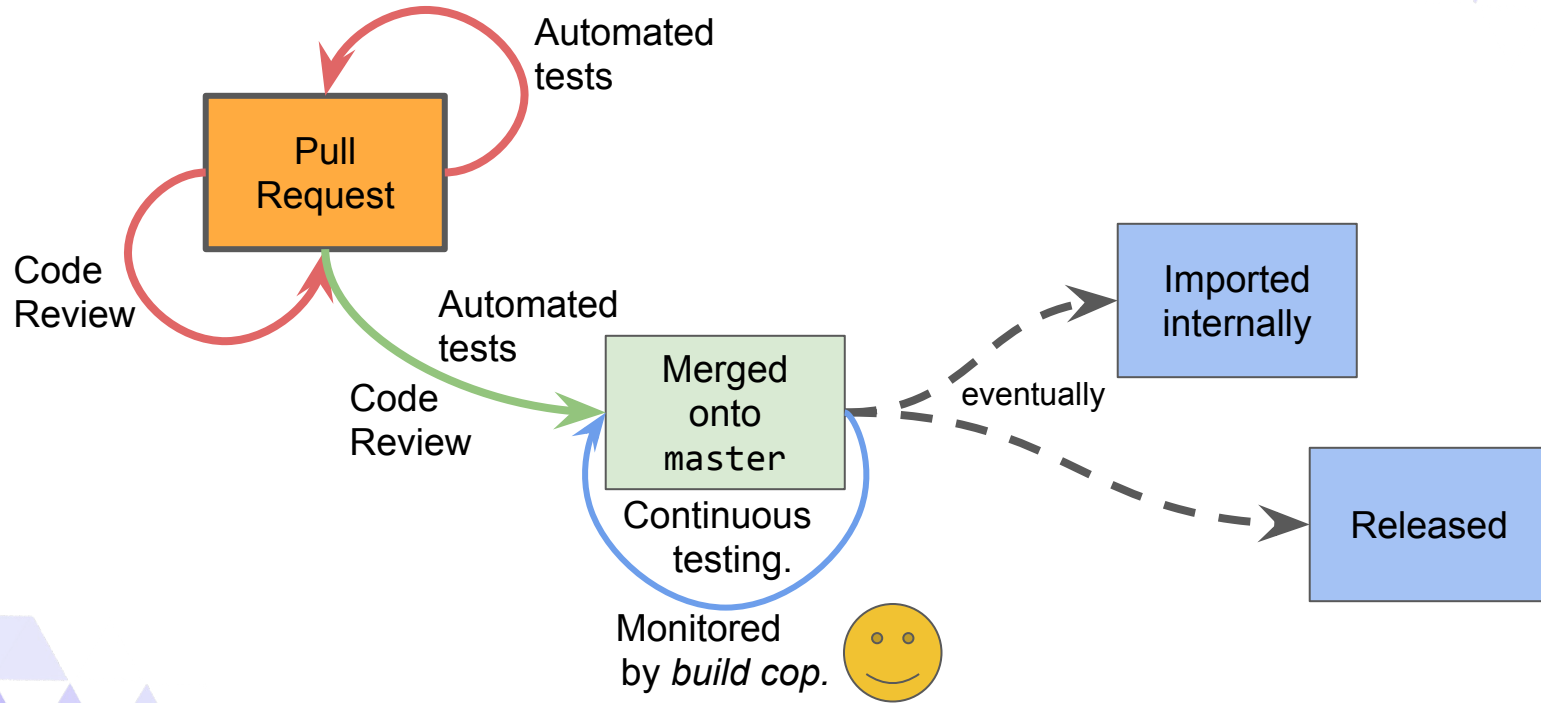
lang/core

release notes: yes

Development Process: **Office Hours**



Development Process



Keeping the Community Engaged

- Foster understanding between devs and community
 - Set **expectations**
- Establish community **Meetups**
 - Place for community to meet with team, put faces to names, **press strongly for feature requests, bug fixes**, etc
 - Showcase for **interesting** and often **unexpected use cases**
- Establish processes for making changes
 - **gRFC** - seek **community input** for 2 weeks before converging on new features
- Improve documentation and empower community

An empowered community

From REST to gRPC: An API Evolution Story

Joe Runde
IBM
@joerunde

Michael Keeling
IBM
@michaelkeeling

Why is gRPC Awesome?

Performance
Remote Procedure Calls
Strategic Direction of our Platform

Would we do it again?

Yes.

- Super easy to integrate with a service
- Promotes small polyglot services
- Difficult to do bad things
- Performance is 🐣 🐣 🐣

Adoption



Microservices: in data centers



Streaming telemetry from network devices



Client Server comm. / Internal APIs



Mobile Apps

What [IMO] makes a great OSS project

- Valuable software role **for** the community
- Clean **licensing** for maximum utility
- Is **well-supported, managed, and maintained**. Signal **commitment**.
- Code **quality** and **testing**
- Continuous improvement
- Prompt and **predictable release cycle**
- Easy to **get started**: Installation, documentation
- **Feedback from users** and understanding of their use cases

All of which facilitate an **engaged** and vibrant community

What [IMO] is probably not important

- Code contributor population
 - Most users of open-source never actually open the source
 - Being open-source > just free: community can extend & self-sustain
- Development model
 - OSS-first with internal uptake
 - Internal-first with external releases

The image features a white background with decorative purple geometric patterns in the corners. These patterns consist of overlapping triangles and hexagons in various shades of purple, creating a modern, abstract look. The central text is rendered in a bold, black, sans-serif font.

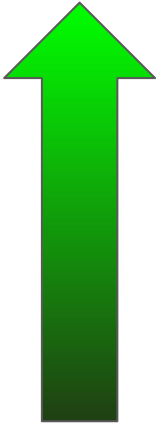
In Summary

OSS development is **harder**



- The world is a big place, more systems/configs.
- Increased overhead from community.
- Restricted to external tools...
- ... or you'll need to roll your own.

OSS development is **harder, but...**



- ✓ Improves org's PR/goodwill.
- ✓ Attracts talent.
- ✓ External scrutiny: keeps you honest.
- ✓ Gateway to the organization.
- ✓ Enables contributions.

Thank you!

