

+ 23

Continuous Regression Testing for Safer and Faster Refactoring

PEJMAN GHORBANZADE



20
23



Continuous Regression Testing for Safer and Faster Refactoring



Pejman Ghorbanzade
Aurora Innovation

Engineers spend **17 hours** per week maintaining software.

*Stripe 2019 Developer Coefficient Report

Maintaining Software

- Reading
- Refactoring
- Upgrading
- Migrating
- Debugging
- Adding tests
- Writing documentation
- Resolving technical debt



"The only constant in life is change." - Heraclitus

Types of Change

- Fixing a defect
- Enabling code reuse
- Adjusting to new expectations
- Improving a function implementation
- Upgrading a third-party dependency
- Renaming a function or variable
- Changing system configuration
- Updating build system toolchain



“Software engineering is programming integrated over time.” - Titus Winters

It takes 23 days for software engineers to gain confidence that a code change works as expected.

*Tricentis 2021 Report: How The World's Top Organizations Test

Agenda

- **What is continuous regression testing**
- How does regression testing work in practice
- How to build a regression testing system
- Going beyond finding behavioral regressions
- How to use regression testing effectively
- Establishing a culture of safety at scale

About Aurora

Delivering the benefits of self-driving technology, safely, quickly, and broadly.

aurora.tech/careers



About Me

- Staff Software Engineer at [Aurora Innovation](#)
- Building tooling to improve developer experience
- Accelerating the development of web applications
- 8 years of professional experience
- Maintaining mission-critical software systems
- Ex VMware Carbon Black, Canon Medical Informatics
- Former founder of a developer tools startup



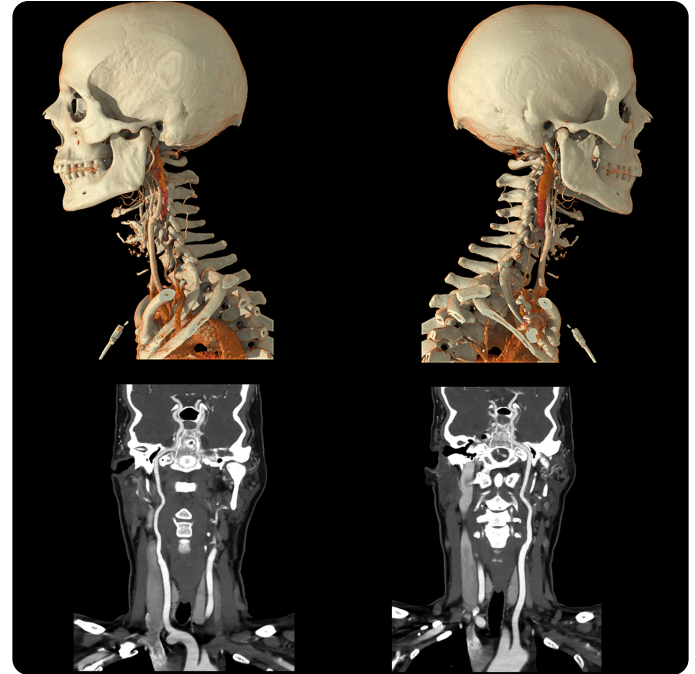
Pejman Ghorbanzade

pejman.dev

What we do matters

Low-dose Ultra Helical CT Angiogram of the Carotids and Circle of Willis for stroke work-up.

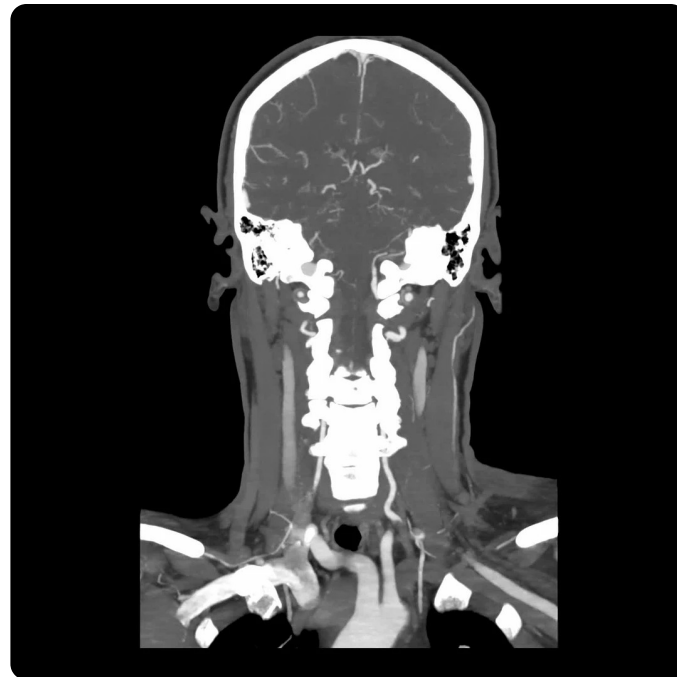
Clear visualization of contrast enhanced vessels and surrounding soft tissue enables fast and confident rule-out of occlusion.



Courtesy of Canon Medical Group

Digital Imaging and Communications in Medicine

Group	Element	Tag name
0008	0020	Study date
0008	002A	Acquisition DateTime
0010	0010	Patient's name
0020	0013	Instance Number
0020	0020	Patient orientation



Courtesy of Canon Medical Group

What could go wrong?

Incorrect interpolation and misrepresenting image positions could result in inaccurate measurements, causing patient harm.



Courtesy of Canon Medical Group

Everything could go wrong

“The inherent complexity of the real world and the continuous change of requirements result in large and complex software systems that are costly and difficult to maintain.”

“In a sufficiently long time horizon, all possible behaviors of your system will occur.” - Hyrum's law (modified)

Testing as risk mitigation



If every code change can break our software, how could we stay productive and safely introduce frequent changes?

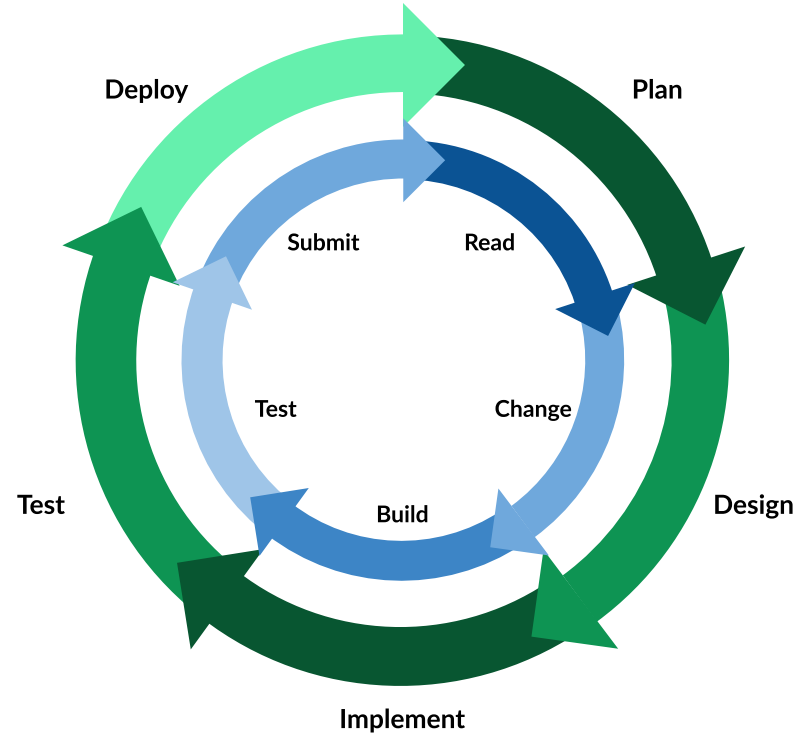


Implement high-level tests and continuously run them at scale to cover real-world system behaviors with reasonable degree of confidence.

Developer inner and outer loops

Fast feedback cycles boost development confidence and productivity.

Moving high-level tests out of the developer loop results in slow and inefficient application life-cycles.



The myth of the testing pyramid

Concerns

Difficult

costly to setup and run

Expensive

need system deployment

Slow

take long to execute

Brittle

flaky and easy to break

Benefits

Expressive

easy to read and modify

Scalable

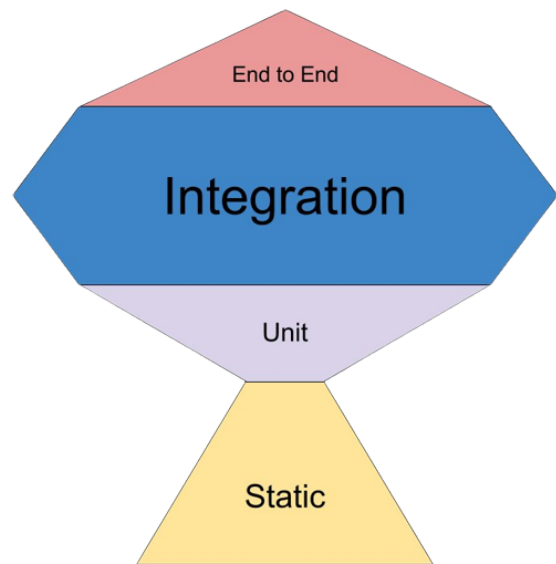
can run many test cases

Comprehensive

cover component interactions

Reassuring

provide more confidence



"Write tests. Not too many. Mostly integration." - [Guillermo Rauch](#)

Continuous regression testing

Continuously verifying that the software works as well as before, during the development stage.

✓ Testing for Correctness

- Requires describing the expected behavior for each test input.
- Mismatches against the expected values indicate failure.
- Tests are difficult to maintain, scale, and automate.

🔍 Testing for Regression

- Treats a released version of software as baseline.
- Mismatches against the baseline require justification.
- Tests are expressive and decoupled from the test input.

Higher-level tests in practice

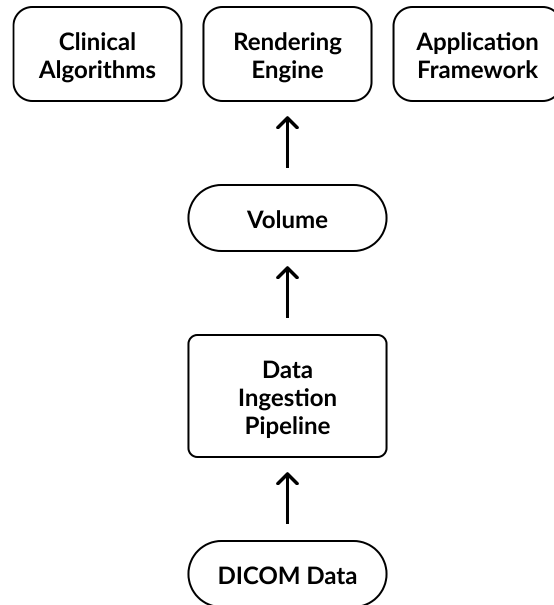
Safely rewriting a critical data ingestion pipeline

500,000 +
lines of code

12,000 +
real-world datasets

10,000 +
attributes to verify

16,000 +
gigabytes of input data



In-Memory Comparison

- Test is difficult to setup
- Test system is inefficient to run
- Test system is not reuseable

50,000 +

LoC test framework

16 +

hours to run test

```
for (auto test_case: test_suite) {  
    auto new_output = new_system(test_case);  
    auto old_output = old_system(test_case);  
    auto report = compare(new_output, old_output);  
    report.store(test_case);  
}  
generate_summary_report();
```

Snapshot Testing

Debugging

System is treated as a black box.
Output may miss important data.

Reliability

Output may include
nondeterministic data.

Data Management

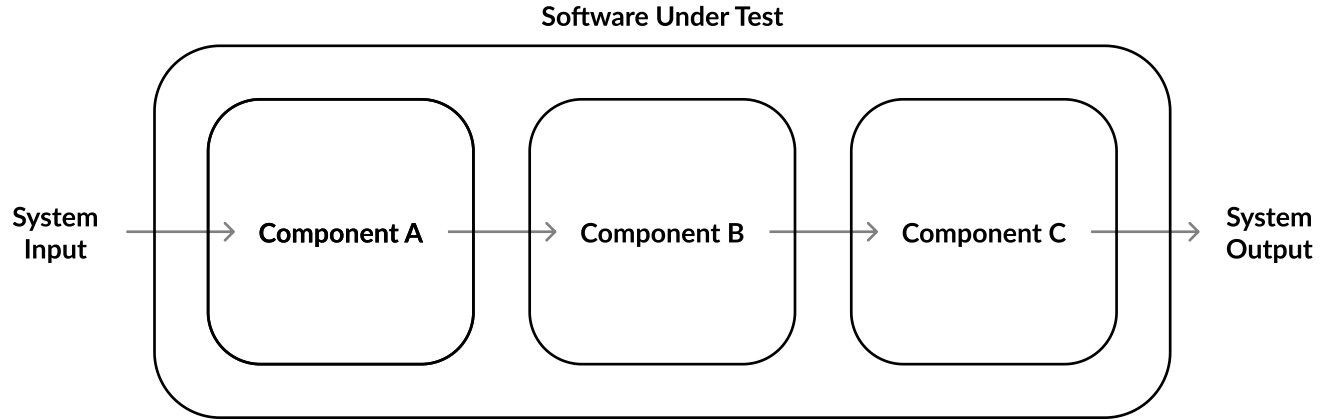
Output is stored in version
control along with source code.

Reporting

Differences are difficult to
inspect, understand, and manage.

```
for (auto test_case: test_suite) {  
    auto new_output = new_system(test_case);  
    store_snapshot(test_case, new_output);  
    auto old_output = load_snapshot(test_case);  
    auto report = compare(new_output, old_output);  
    report.store(test_case);  
}  
generate_summary_report();
```

Problem: Debugging



Good tests point to the root cause when they fail.

Problem: Reliability

Snapshot tests

- are prone to capturing non-deterministic data.
- are prone to capturing unimportant data.
- may leave out changes not captured in the output.
- fail to compare captured data in their original type.

Good tests pass and fail only when they are supposed to.

```
The Alameda ALM 408-207-1126
      777 The Alameda
      San Jose, CA 95126
CANTALOUPE                $3.99 F
  *Sale*                    $3.32   -$0.67
  Prime Extra 10.00%        -$0.33
      Subtotal:              $3.99
      Total Savings:         -$1.00
      Net Sales:             $2.99
      Total:                 $2.99
      Sold Items:            1
```

```
*****
```

```
901 61797 09/15/2023 05:46 PM
```

Problem: Data Management

```
$ git status
```

```
On branch feature/cppcon
```

```
Your branch is up to date with 'origin/feature/cppcon'.
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
modified: SuperMarketTest.TenPercentDiscount.approved.txt
```

```
modified: SuperMarketTest.BuyOneGetOneDiscount.approved.txt
```

Good test systems enable auditing how software evolves.

Problem: Reporting

```
$ git diff
index 3de4787..5ad5e04 100644
--- a/SuperMarketTest.TenPercentDiscount.approved.txt
+++ b/SuperMarketTest.TenPercentDiscount.approved.txt
@@ -1,5 +1,5 @@
 apples                                4.97
    1.99 * 2.500
-Total:                                4.47
+Total:                                4.97
```

Good test systems report insights as output, not raw test results .

Design Principles

Developer Friendly

Designed for everyday use by developers. Should enable creating tests that are cheap to write, fast to run, and easy to modify.

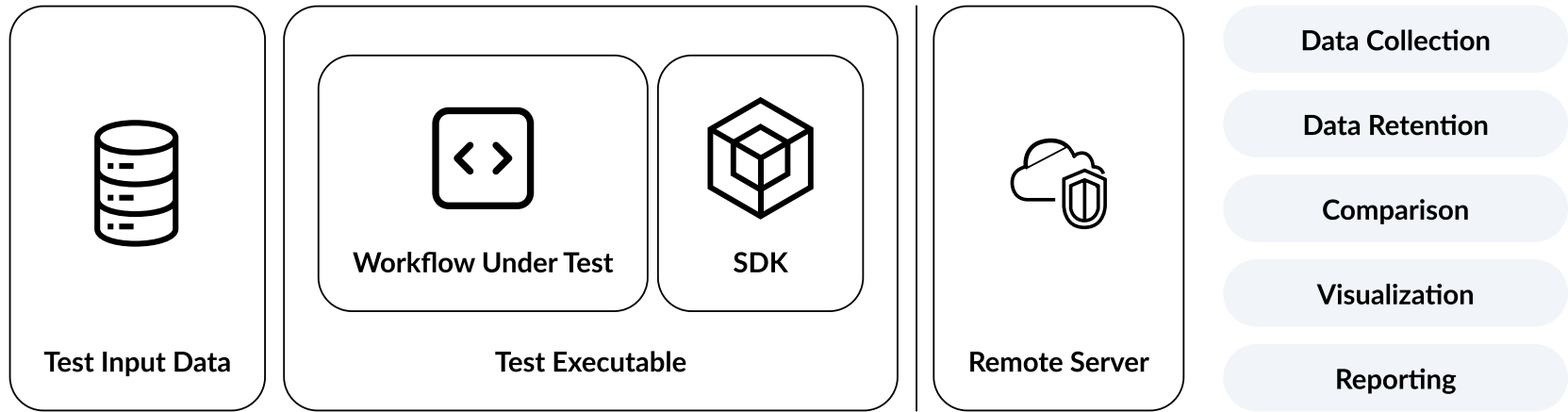
Flexible

Designed for capturing values of variables and runtime of functions. Should handle data points with primitive or user-defined data types.

Scalable

Designed for testing mission-critical software. Should handle capturing data from large number of test cases and report test results as actionable insights.

Rethinking snapshot testing



About Touca

touca.io

Find the unintended side-effects
of your day-to-day code changes

Trusted By:

Canon

Backed By:

techstars



Agenda

- What is continuous regression testing
- **How does regression testing work in practice**
- How to build a regression testing system
- Going beyond finding behavioral regressions
- How to use regression testing effectively
- Establishing a culture of safety at scale

Touca Server

Remotely compare the output of your software against a previous baseline version.

- ✔ Free and Open Source
- ✔ Developer Friendly
- ✔ Language Agnostic
- ✔ Battle Tested


The screenshot displays the Touca.io web interface for a project named 'Students'. The interface includes a navigation bar with 'Docs' and the user name 'Pejman Ghorbanzade'. Below the navigation bar, there are tabs for 'Versions' (5), 'Test Cases' (3), and 'Settings'. A search bar for versions is present, along with filters and sorting options. The main content area lists five versions: v5.0, v4.0, v3.0, v2.0, and v1.0. Each version entry shows a status icon (orange plus, orange minus, green check, yellow star, green check), a progress bar, a performance comparison (e.g., '1s 832ms', '4% slower'), and a '3 cases' indicator. A 'Runtime Trend' graph on the right shows a line chart of runtime times across versions v1.0 to v5.0. Below the graph, a 'Recent Events' section notes that v2.0 was promoted as the baseline.

Version	Status	Progress	Performance	Cases	Time
v5.0	⊕	87%	1s 832ms	3 cases	18 minutes ago
v4.0	⊖	88%	1s 860ms (4% slower)	3 cases	18 minutes ago
v3.0	✔	100%	1s 916ms (7% slower)	3 cases	18 minutes ago
v2.0	★	61%	1s 794ms (3% faster)	3 cases (1 new, 1 missing)	18 minutes ago
v1.0	✔	100%	1s 857ms	3 cases	18 minutes ago


Runtime Trend

Recent Events: Pejman Ghorbanzade promoted v2.0 as baseline. "Once you inspected differences found in a given version, you can set that version as the new baseline, so that future versions are compared against it."

Self-hosting

 [github.com / trytouca](https://github.com/trytouca)

```
$ brew install touca  
$ touca server install
```

 Apache 2.0 License

```
>
```

Test Framework

```
#include "catch2/catch.hpp"
#include "students.hpp"

TEST_CASE("find_students") {
    SECTION("alice") {
        auto student = find_student("alice");
        CHECK(student.name == "Alice Anderson");
        CHECK(student.dob == Date(2006, 3, 1));
        CHECK(student.gpa == 3.9);
    }
    SECTION("bob") {
        auto student = find_student("bob");
        CHECK(student.name == "Bob Brown");
        CHECK(student.dob == Date(1996, 6, 31));
        CHECK(student.gpa == 3.8);
    }
    SECTION("charlie") {
        auto student = find_student("charlie");
        CHECK(student.name == "Charlie Clark");
        CHECK(student.dob == Date(2003, 9, 19));
        CHECK(student.gpa == 3.3);
    }
}
```

```
#include "touca/touca.hpp"
#include "students.hpp"

int main(int argc, char* argv[]) {
    touca::workflow("find_students", [](const std::string& username) {
        const auto& student = find_student(username);
        touca::check("name", student.name);
        touca::check("birth_date", student.dob);
        touca::check("gpa", student.gpa);
    });
    return touca::run(argc, argv);
}
```

- Parses command-line arguments
- Retrieves test cases
- Submits captured data
- Reports test progress
- Handles any errors

Writing tests

Test your complex software workflows for any number of inputs by capturing values of variables and runtime of functions.



```
#include "students.hpp"
#include "touca/touca.hpp"

int main(int argc, char* argv[]) {
    touca::workflow("find_student", [] (const std::string& username) {
        const auto& student = find_student(username);
        touca::assume("username", student.username);
        touca::check("fullname", student.fullname);
        touca::check("birth_date", student.dob);
        touca::check("gpa", student.gpa);
        touca::check("pass", student.gpa < 3.9);
        touca::add_metric("external_source", 1500);
    });
    return touca::run(argc, argv);
}
```


Running tests

Run your tests for each code change or pull request, as part of CI or on a dedicated test machine, to get fast feedback during the development stage.

```
$ brew install touca
$ touca login
$ touca test
```

```
$ touca test
```

Touca Test Runner

Suite: students/v2

1. **PASS** alice (209 ms)
2. **DIFF** bob (223 ms)
3. **PASS** charlie (217 ms)

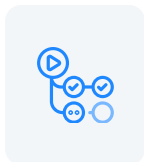
Tests: 2 perfect, 1 different, 3 total

Time: 1.45 s

🌟 Ran all test suites.

Automating tests

Integrate your tests with the CI/CD pipeline to automate their execution and receive feedback when you need it.



```
- name: build cpp app
  run: ./build.sh
- uses: trytouca/actions-run-tests@v1
  env:
    TOUCA_API_KEY: ${{ secrets.TOUCA_API_KEY }}
    TOUCA_API_URL: https://api.touca.io/@/my_org/my_cpp_app
    TOUCA_TEST_VERSION: ${{ steps.params.outputs.version }}
  with:
    executable: ./local/dist/bin/my_cpp_app
```

Visualizing differences

Automatically compare new test results and visualizing inspect potential differences against your baseline.

- ✓ Automatic and on-demand comparison
- ✓ Overall insights and summary reports
- ✓ Custom comparison rules

Comparing *latest* version v2.0 against *baseline* version v1.0

Find a key... Filter: None Sort: Match Rate

Common Keys

Results that are both in version v2.0 (latest) and in version v1.0 (baseline)

- courses** (76% Match Rate) array

```
[
  {
    "Course": {
      "grade": 3.8,
      "name": "computers"
    }
  },
  {
    "Course": {
      "grade": 4.3.6,
      "name": "mat english"
    }
  }
]
```
- gpa** (95% Match Rate) number
value in version v2.0: 3.7 value in version v1.0: 3.9
- number of courses** (2) number

Comparing images and videos

Share test results with team members and visualize differences of any kind.




touca.io Docs Pejman Ghorbanzade

Comparing *latest* version v1.1 against *baseline* version v1.0


input_file image >

output_file image v

value in version v1.0



value in version v1.1

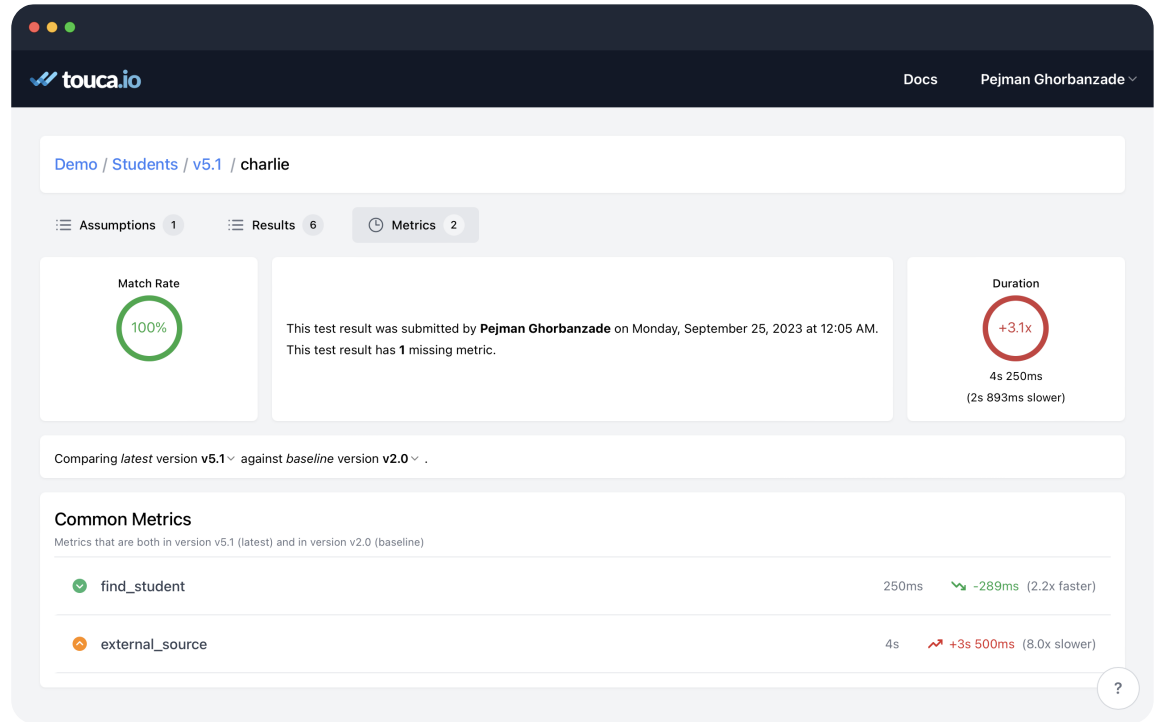
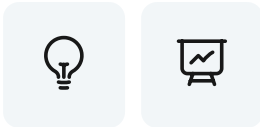


?

The screenshot shows a web interface for Touca.io. At the top, there's a navigation bar with the logo and user name. Below it, a header indicates a comparison between 'latest version v1.1' and 'baseline version v1.0'. The main content area shows two sections: 'input_file' and 'output_file'. Each section has a green checkmark and a blue 'image >' button. The 'output_file' section displays two side-by-side images of construction workers. The left image is labeled 'value in version v1.0' and the right is 'value in version v1.1'. Both images show the same scene with blue bounding boxes around the workers. A question mark icon is in the bottom right corner of the interface.

Finding performance regressions

Gain insights and analytics about how your software is evolving over time.



Reporting Results

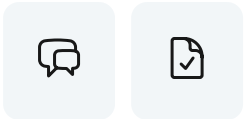
Subscribe to any suite to get notified about new regressions.



A screenshot of the touca.io web application. The top navigation bar is dark blue with the touca.io logo on the left, 'Docs' in the center, and 'Alice Anderson' with a dropdown arrow on the right. Below the navigation bar, the page title is 'Settings' with a 'Back to Dashboard' button in the top right corner. A left sidebar contains a list of settings: Profile, Api Keys, Preferences, and a section for 'SERVER SETTINGS' which includes Health Metrics, User Accounts, Audit Logs, Mail Transport (highlighted with a grey background), and Telemetry. The main content area is titled 'Mail Transport' and contains a form with four fields: 'Host' (text input with 'smtp.mailtrap.io'), 'Port' (dropdown menu with '2525'), 'Username' (text input with 'cf83670c6dec32'), and 'Password' (password input with masked characters). Below the form are two buttons: 'Update' and 'Reset Configuration'. A small circular help icon with a question mark is located in the bottom right corner of the settings panel.

Baseline Management

Collaborate with your team members in investigating regressions and managing baseline versions.



The screenshot displays the touca.io interface for managing test suites. A modal dialog titled "Promote Version v2.0" is open, showing a warning: "You are about to promote version v2.0 to baseline of suite students. This action makes test results of any future version of this Suite to be compared with test results of version v2.0." Below this, there is a text input field for a "Promotion Reason (Optional)" with a placeholder "Let others know why you are promoting this version." and a note "This field supports markdown format." Buttons for "Promote", "Preview", and "Cancel" are at the bottom of the dialog.

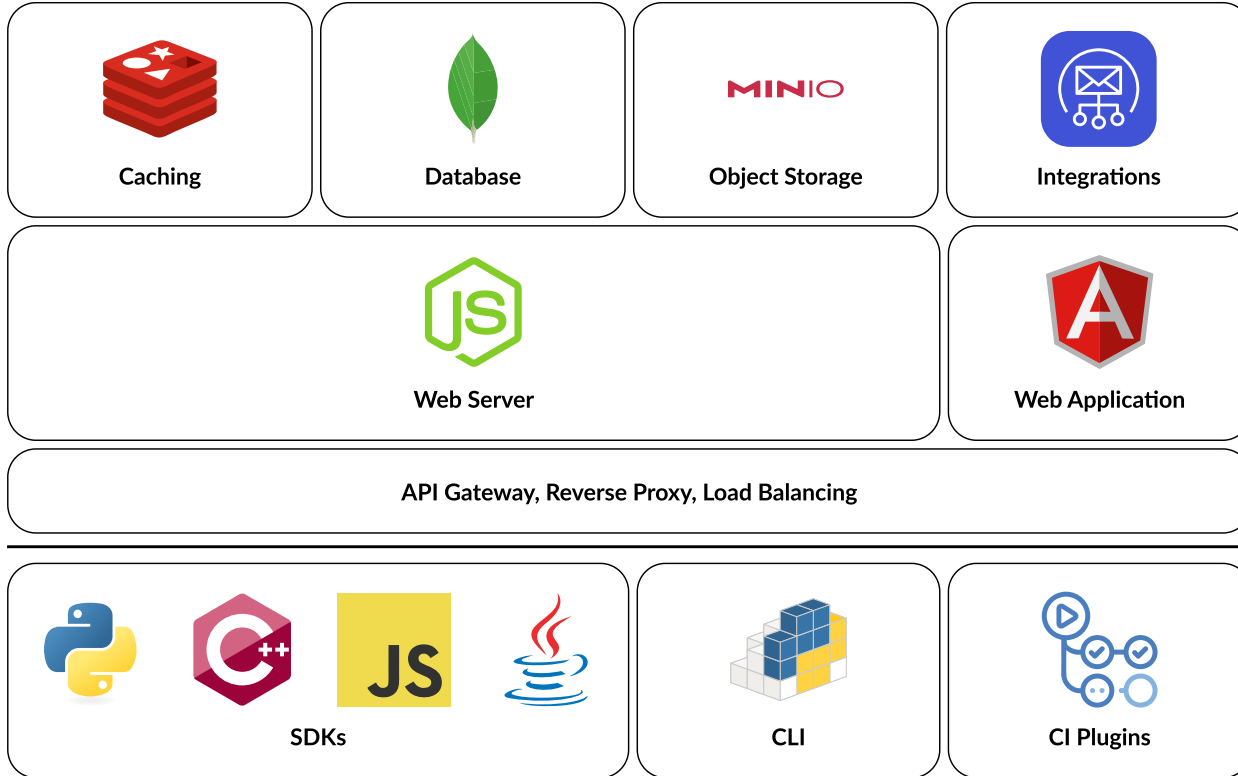
In the background, the "students" suite is being compared against its latest version (v2.0). The "Match Rate" is 94%, and the "Duration" is 1s 233ms, marked as "Same". A table of "Common Testcases" is visible below, listing testcases like "charlie", "bob", and "alice" with their respective match rates and durations.

Testcase	Match Rate	Duration	Keys	Time
charlie	100%	1s 743ms (8.5x slower)	7 keys	1 hour ago
bob	100%	1s 717ms (7.0x slower)	7 keys	1 hour ago
alice	81.6%	1s 742ms (7.5x slower)	7 keys	1 hour ago

Agenda

- What is continuous regression testing
- How does regression testing work in practice
- **How to build a regression testing system**
- Going beyond finding behavioral regressions
- How to use regression testing effectively
- Establishing a culture of safety at scale

System Architecture



C++ SDK

GCC
9.4.0

Clang
11.0.0

MSVC
1900

C++11 through C++23

CMake
3.14

Conan
v1

Bazel
v6.3.2

```
$ ./build.sh
info      building cpp library using cmake
-- Fetching third-party dependency: flatbuffers
-- Fetching third-party dependency: fmt
-- Fetching third-party dependency: ghcfilesystem
-- Fetching third-party dependency: httplib
-- Fetching third-party dependency: rapidjson
-- Fetching third-party dependency: mpark_variant
-- Fetching third-party dependency: cxxopts
-- Configuring done (32.6s)
-- Generating done (0.0s)
-- Install configuration: "Release"
-- Installing: ./local/dist/lib/libtouca.a
-- Installing: ./local/dist/licenses/LICENSE
$
```

[🔗 touca.io/docs/sdk/installing](https://touca.io/docs/sdk/installing)

Data Capturing API

```
touca::workflow("students", [](const std::string& username) {  
    const auto& student = find_student(username);  
    touca::check("name", student.name);  
    touca::check("birth_date", student.dob);  
    touca::check("gpa", student.gpa);  
});
```

```
struct Date {  
    unsigned short year;  
    unsigned short month;  
    unsigned short day;  
};
```

```
template <typename Char, typename Value>  
void check(Char&& key, const Value& value) {  
    touca::detail::check(std::forward<Char>(key),  
        serializer<Value>().serialize(value));  
}
```

Capturing behavior data

```
touca::log("timestamp", student.created_at);
```

```
touca::assume("username", student.username);
```

```
for (const auto& course : student.courses) {  
    touca::add_array_element("courses", course);  
    touca::add_hit_count("number of courses");  
}
```

```
void Client::check(const std::string& key, const data_point& value) {  
    if (has_last_testcase()) {  
        _testcases.at(get_last_testcase())->check(key, value);  
    }  
}
```

Capturing performance data

```
touca::start_timer("find_student");  
const auto& student = find_student(username);  
touca::stop_timer("find_student");
```

```
Student find_student(const std::string& username) {  
    touca::scoped_timer timer("find_student");  
    // ...  
}
```

```
with touca.scoped_timer("find_student"):  
    student = find_student(username)
```

```
touca::add_metric("external_source", 1500);
```

Partial Template Specialization

```
template <typename T, typename = void>
struct serializer {
    data_point serialize(const T& value) {
        static_assert(std::is_same<data_point, T>::value,
                      "did not find any specialization of "
                      "serializer for the given type");
        return static_cast<T>(value);
    }
};
```

Specializing Standard Types

```
template <typename T>
using is_number_signed =
    conjunction<negation<std::is_same<T, bool>>,
                std::is_integral<T>,
                std::is_signed<T>>;
```

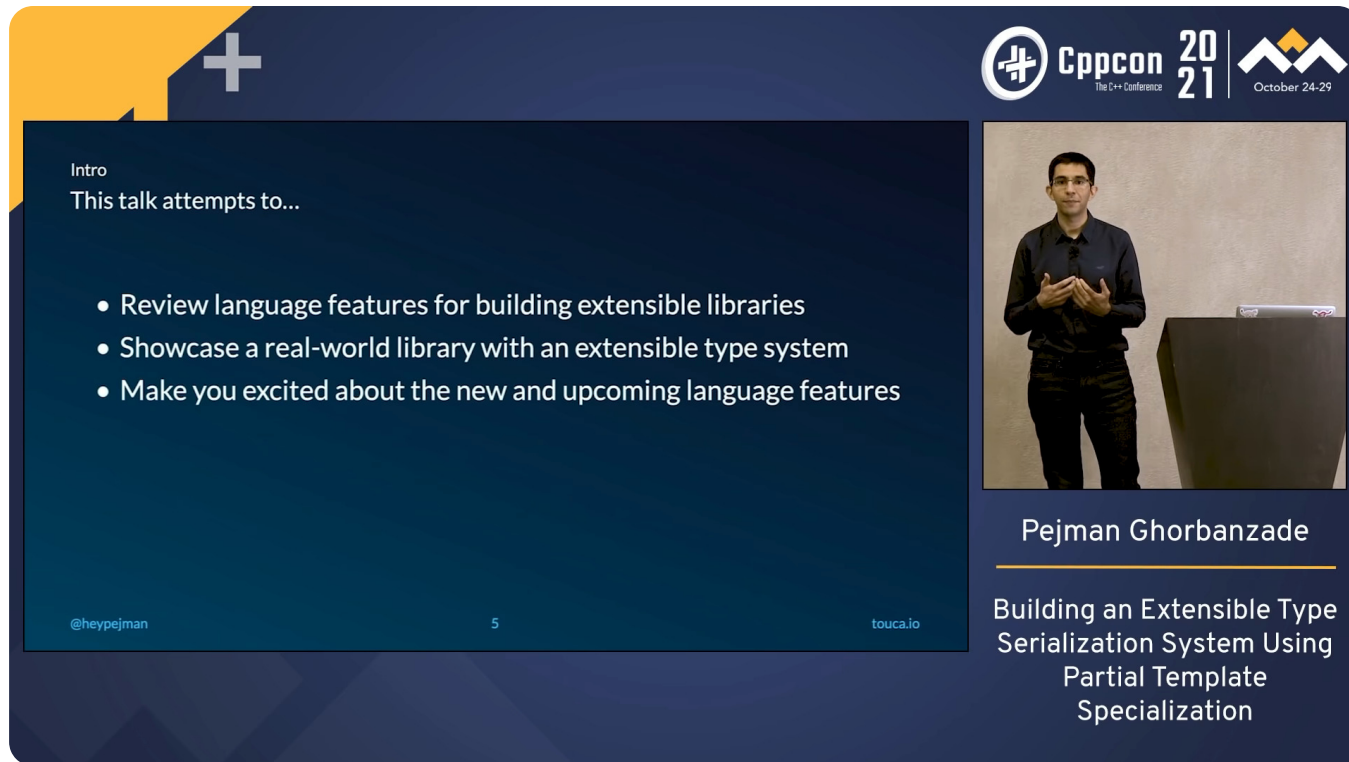
```
template <typename T>
struct serializer<
    T, enable_if_t<is_number_signed<T>::value>> {
    data_point serialize(const T& value) {
        return data_point::number_signed(value);
    }
};
```

```
enum class internal_type : std::uint8_t {
    null,
    object,
    array,
    string,
    boolean,
    number_signed,
    number_unsigned,
    number_float,
    number_double,
    unknown
};
```

Specializing User-Defined Types

```
template <>
struct serializer<Date> {
    data_point serialize(const Date& date) {
        return object("Date")
            .add("year", date.year)
            .add("month", date.month)
            .add("day", date.day);
    }
};
```


Deeper Dive




Intro
This talk attempts to...

- Review language features for building extensible libraries
- Showcase a real-world library with an extensible type system
- Make you excited about the new and upcoming language features

@heypejman 5 touca.io

Cppcon 2021
The C++ Conference
October 24-29



Pejman Ghorbanzade

Building an Extensible Type
Serialization System Using
Partial Template
Specialization

Low-Level API

```
touca::workflow("students", [](const std::string& username) {  
    const auto& student = find_student(username);  
    touca::check("name", student.name);  
    touca::check("birth_date", student.dob);  
    touca::check("gpa", student.gpa);  
});
```

```
Post::Status ClientImpl::post() const {  
    /** ... */  
    const auto& buf = Testcase::serialize(testcases);  
    std::string content((const char*)buf.data(), buf.size());  
    const auto& response = transport->binary(content);  
    /** ... */  
}
```

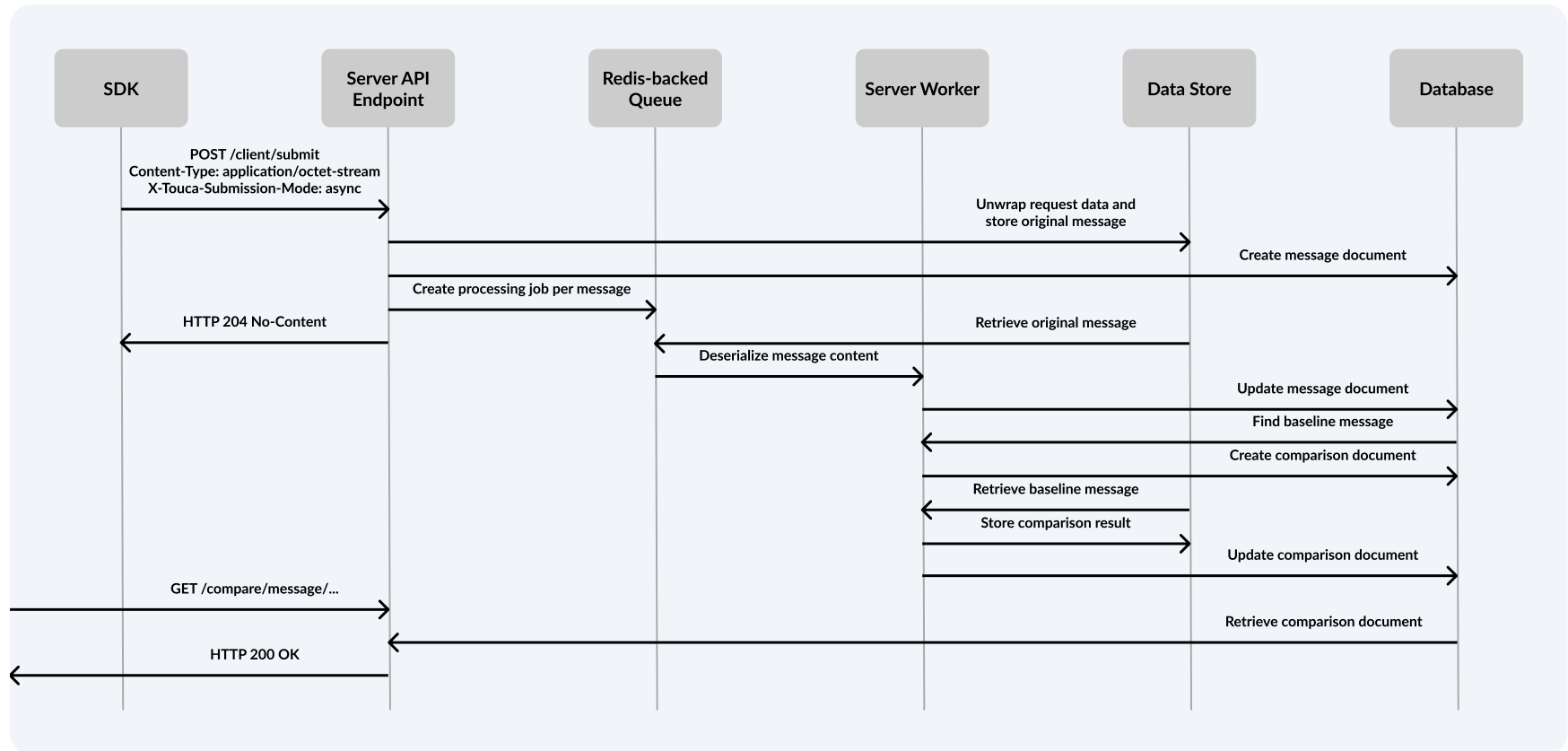
```
int main() {  
    touca::configure();  
    for (const auto& username : {"alice", "bob", "charlie"}) {  
        touca::declare_testcase(username);  
        const auto& student = find_student(username);  
        touca::check("name", student.name);  
        touca::check("birth_date", student.dob);  
        touca::check("gpa", student.gpa);  
  
        touca::post();  
        touca::save_binary("touca_" + username + ".bin");  
        touca::save_json("touca_" + username + ".json");  
        touca::forget_testcase(username);  
    }  
    touca::seal();  
}
```

FlatBuffers Schema

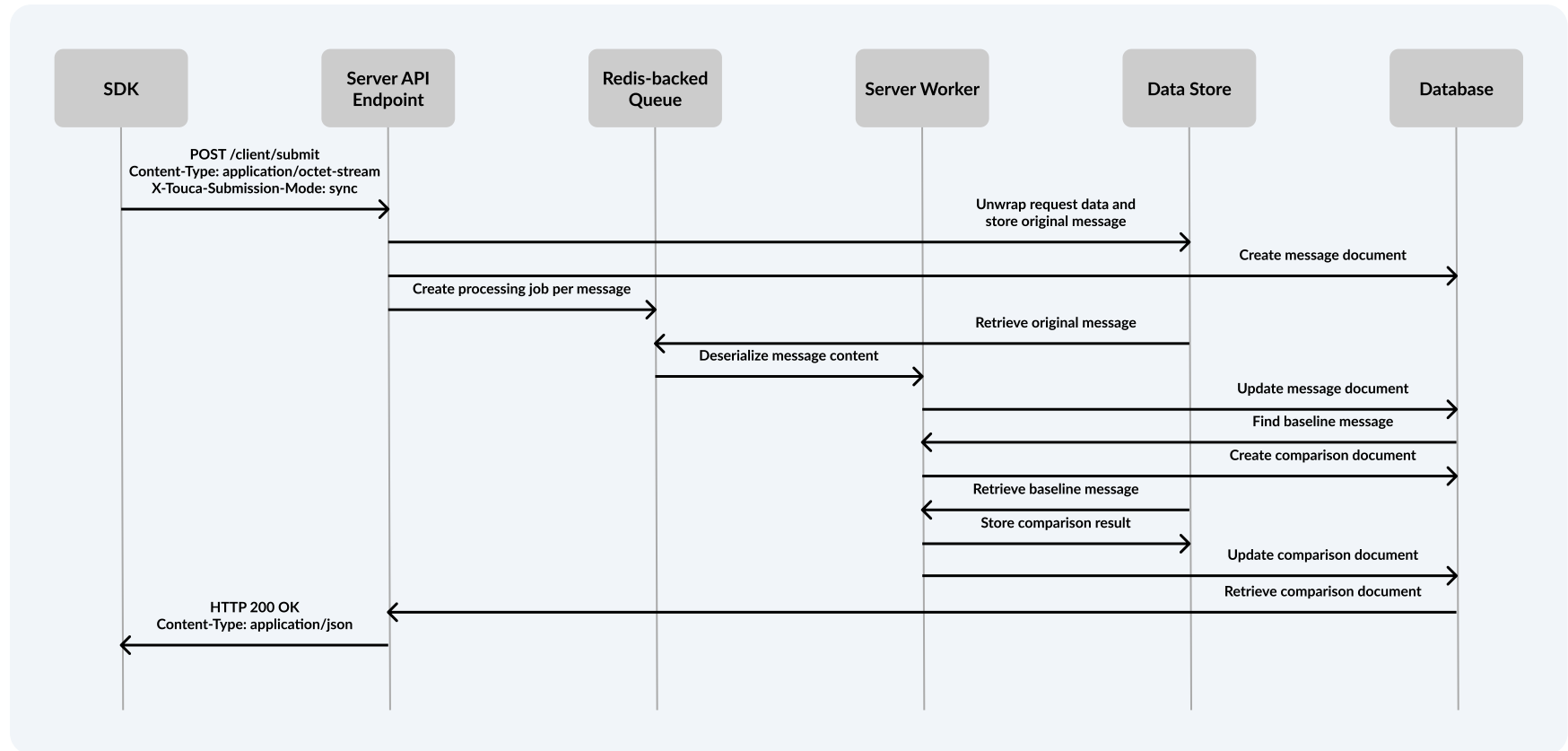
```
union Type {
  Bool,
  Int,
  /** ... */
  String,
  Object,
  Array
}
table TypeWrapper {
  value:Type;
}
table Result {
  key:string;
  value:TypeWrapper;
}
```

```
table Results {
  entries:[Result];
}
table Message {
  metadata:Metadata;
  results:Results;
  metrics:Metrics;
}
table MessageBuffer {
  buf:[uint8] (nested_flatbuffer: "Message");
}
table Messages {
  messages:[MessageBuffer];
}
root_type Messages;
```

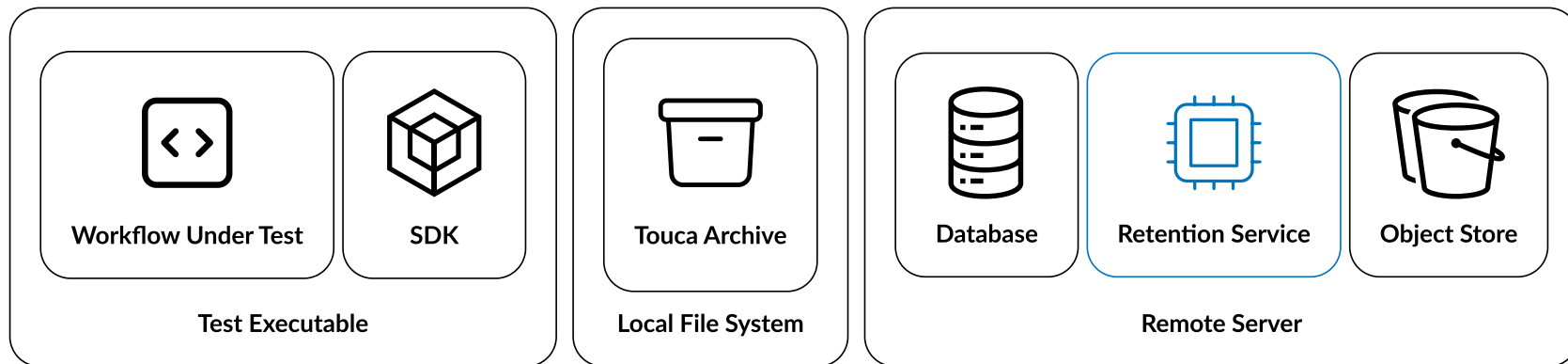
Data ingestion w/ async processing



Data ingestion w/ on-demand processing



Data Retention




✔ Local Filesystem Backup


✔ Configurable Retention Duration

Agenda

- What is continuous regression testing
- How does regression testing work in practice
- How to build a regression testing system
- **Going beyond finding behavioral regressions**
- How to use regression testing effectively
- Establishing a culture of safety at scale

Custom comparison rules

 Language Agnostic

 Real-Time Feedback

```
touca::check("gpa", student.gpa, touca::decimal_rule::min_absolute(3));
```

3.8

Actual value
Version v5.1

3.9

Previous value
Version v2.0

Value passes minimum threshold of 3.

Tracking performance benchmarks

```
#include <benchmark/benchmark.h>

static void BM_String(benchmark::State& state) {
    for (auto _ : state)
        std::string empty_string;
}

BENCHMARK(BM_String);
BENCHMARK_MAIN();
```

```
$ touca plugin add plugins://google_benchmark
$ touca google_benchmark output.json
```

```
{
  "context": {
    "date": "2023/09/25-18:40:25",
    "num_cpus": 40,
    "mhz_per_cpu": 2801,
    "cpu_scaling_enabled": false,
    "build_type": "debug"
  },
  "benchmarks": [
    {
      "name": "BM_String",
      "iterations": 94877,
      "real_time": 29275,
      "cpu_time": 29836,
      "bytes_per_second": 134066,
      "items_per_second": 33516
    }
  ]
}
```

Profiling build times

```
$ bazel build :sample_app --generate_json_trace_profile \  
  --profile sample_app.profile.gz --noslim_json_profile  
$ bazel analyze-profile sample_app.profile.gz  
=== PHASE SUMMARY INFORMATION ===  
  
Total launch phase time           0.014 s    0.42%  
Total init phase time             0.048 s    1.46%  
Total target pattern evaluation phase time 0.006 s    0.19%  
Total interleaved loading-and-analysis phase time 0.153 s    4.64%  
Total preparation phase time      0.001 s    0.05%  
Total execution phase time        3.084 s   93.19%  
Total finish phase time           0.001 s    0.03%  
-----  
Total run time                     3.309 s  100.00%
```

```
$ touca plugin add plugins://bazel  
$ touca bazel sample_app.profile.gz
```

Profiling the size of binaries

```
$ ./bloaty bloaty -d compileunits
  FILE SIZE          VM SIZE
-----
57.5% 17.4Mi    68% 4.60Mi  [175 Others]
17.2% 5.08Mi    4.3% 295Ki  third_party/protobuf/src/google/protobuf/descriptor.cc
 7.3% 2.14Mi    2.6% 179Ki  third_party/protobuf/src/google/protobuf/descriptor.pb.cc
 4.6% 1.36Mi    1.1% 78.4Ki  third_party/protobuf/src/google/protobuf/text_format.cc
 3.7% 1.10Mi    4.5% 311Ki  third_party/capstone/arch/ARM/ARMDisassembler.c
 1.3% 399Ki    15.9% 1.07Mi  third_party/capstone/arch/M68K/M68KDisassembler.c
 3.2% 980Ki    1.1% 75.3Ki  third_party/protobuf/src/google/protobuf/generated_message_reflection.cc
 3.2% 965Ki    0.6% 40.7Ki  third_party/protobuf/src/google/protobuf/descriptor_database.cc
 1.8% 549Ki    1.7% 114Ki  src/bloaty.cc
100.0% 29.5Mi 100.0% 6.69Mi  TOTAL
```

```
$ touca plugin add plugins://bloaty
$ touca bloaty ./bloaty
```

Tracking exported symbols of a shared library

```
$ nm -gU ./my.dylib | grep touca
000000000006ad64 T __ZNK5touca8Testcase11flatbuffersEv
000000000006a514 T __ZNK5touca8Testcase4jsonEv
000000000006a358 T __ZNK5touca8Testcase7metricsEv
0000000000069508 T __ZNK5touca8Testcase8Metadata4jsonEv
0000000000069370 T __ZNK5touca8Testcase8Metadata8describeEv
00000000000691b0 T __ZNK5touca8Testcase8Overview4jsonEv
0000000000069310 T __ZNK5touca8Testcase8metadataEv
000000000006b9ec T __ZNK5touca8Testcase8overviewEv
```

```
$ touca plugin add plugins://cpp_symbols
$ touca cpp_symbols ./my.dylib --filter touca
```

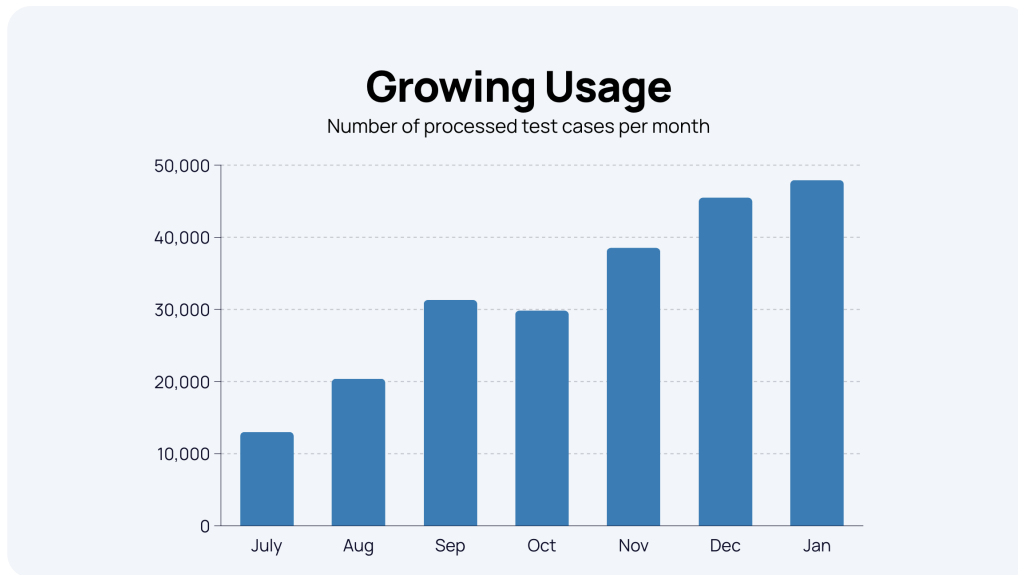
Agenda

- What is continuous regression testing
- How does regression testing work in practice
- How to build a regression testing system
- Going beyond finding behavioral regressions
- **How to use regression testing effectively**
- Establishing a culture of safety at scale

A roller coaster story

"Touca gives us the confidence to develop new features faster and with fewer problems."

- ✓ 10+ paying customers
- ✓ 100+ workflows continuously tested
- ✓ 1000+ unexpected regressions found



A humbling journey

"Success is stumbling from failure to failure with no loss of enthusiasm."

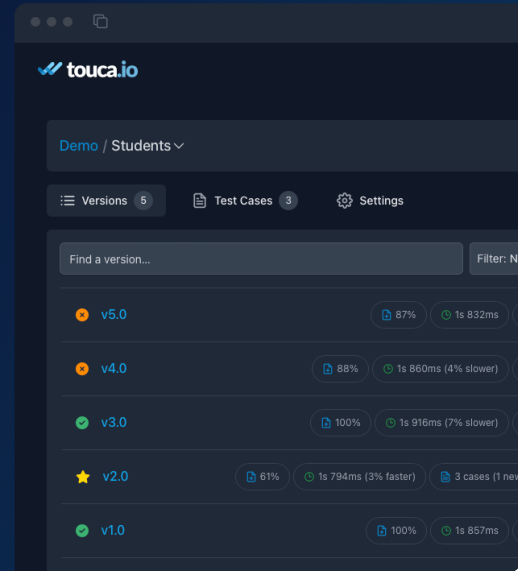
- Winston Churchill

March 30, 2023

Touca is shutting down

Thank you to our users, customers, team members, advisors, investors, and friends who supported us along the way.

touca.io



touca.io/blog/touca-shutting-down

Broken windows theory

“If a window in a building is broken and is left unrepaired, all the rest of the windows will soon be broken. [...] Window-breaking does not necessarily occur on a large scale because some areas are inhabited by determined window-breakers, rather, one unrepaired broken window is a signal that no one cares, and so breaking more windows costs nothing.”

[🔗](#) The Atlantic, 1982



No tool can address software quality issues more effectively than fostering a culture of continuous improvement.



AI-generated

Improving software quality

Culture

Teams need to foster ownership and accountability for software quality. reward continuous improvements and actively share about technical debt.

Commitment

Teams need to commit to continued investment in maintaining software quality and continuously measure their RoI by tracking developer productivity.

Education

Engineers need to understand their product to know what to test, and learn how to write tests that are fast to run, cheap to maintain, and reliable to use.

Improving team culture



- Maintaining software quality is a collective effort.
- Shift-left testing reduces development cost and improves efficiency.
- Effective communication about software quality helps improve it.



- Reward and promote continuous improvements.
- Continuously measure developer experience and productivity.



Proving business value


Continuously:


- Measure developer experience and productivity.
- Monitor changes to developer feedback cycles.
- Track the effectiveness of existing testing practices.
- Extract real-time insights about software development life-cycle.




Learning what to test

- Good tests verify a system expectation that is prone to change.
- Good tests are the ones that fail from time to time.
- Good tests make efficient use of the resources they need.

 Scope

 Confidence



 By Etienne Jong

Learning how to test

Expressive

Good tests are easy to read and effective way of learning business logic. Apply the same code hygiene to your tests as your production code.

Scalable

Good tests are reusable. Optimize for low per-input execution cost. Prefer writing test code with loose assumptions about individual test inputs.

Extensible

Good tests are easy to change. Optimize for low maintenance cost. Single use-case test frameworks have the same cost as production code.

Measure test coverage in terms of verified business requirements.



Learning when to run each test

🛡️ Reliability

🚀 Fast Feedback

- Optimize for return on investment.
- Avoid reusing paradigms of a specific test stage in others.
- Leverage selective test execution for shorter feedback cycle.
- Use periodic test execution for more confidence.



Learning how to read code

- Understand the business.
- Learn code context and history.
- Ask questions and share concerns.
- Take ownership and accountability.



Learning how to change code

- Understand the system
- Study the call-sites
- Resolve unexpected use cases
- Measure the impact
- Mitigate surprises
- Favor incremental rollout
- Communicate your thought process
- Share changes to expectations



“Take many more much smaller steps.” – GeePaw Hill

Agenda

- What is continuous regression testing
- How does regression testing work in practice
- How to build a regression testing system
- Going beyond finding behavioral regressions
- How to use regression testing effectively
- **Establishing a culture of safety at scale**

What we do matters

Unit and Integration Tests

Simulation Tests

Perception Scenarios

Hardware-in-The-Loop Tests

On-vehicle Tests



Simulation Tests

60,000

CPU's utilized
each hour

6 million

Simulation runs
per day

6 billion

Equivalent miles
driven to date



Virtual world-building

Simulating how sensors perceive objective events in the world, helps curate realistic rare-event scenario data that are difficult, dangerous, or expensive to acquire in the real world.



Machine Learning

① Data Selection

② Data Labeling

③ Synthetic Data Generation

④ Model Training

⑤ Model Evaluation



Safety Case Framework

Goal: Aurora's self-driving vehicles are acceptably safe to operate on public roads.

Proficient

The self-driving vehicle is acceptably safe duration nominal operation.

Fail-Safe

The self-driving vehicle is acceptably safe in presence of faults and failures.

Continuously Improving

Safety issues are resolved with appropriate corrective and preventative actions.

Resilient

The self-driving vehicle is acceptably safe in case of misuse and unavoidable events.




Trustworthy

The self-driving enterprise is trustworthy.

Conclusion

- Fast feedback cycles improve developer experience and boost productivity.
- Continuous regression testing facilitates changing software safely and with confidence.
- Shift-left testing reduces development cost and improves efficiency.
- Maintaining software quality requires fostering a culture of continuous improvements.
- Ensuring software safety requires incorporating multiple software testing methods.

Questions

-  [pejman.dev / talks / cppcon23](https://pejman.dev/talks/cppcon23)
-  [github.com / trytouca / trytouca](https://github.com/trytouca/trytouca)
-  [touca.io / docs](https://touca.io/docs)

Appendix

Data Capturing Internals

```
void touca::detail::check(const std::string& key, const data_point& value) {  
    instance.check(key, value);  
}
```

```
void Client::check(const std::string& key, const data_point& value) {  
    if (has_last_testcase()) {  
        _testcases.at(get_last_testcase())->check(key, value);  
    }  
}
```

```
void Testcase::check(const std::string& key, const data_point& value) {  
    _resultsMap.emplace(key, ResultEntry{value, ResultCategory::Check});  
    _posted = false;  
}
```

Serializing User-Defined Types - Implementation

```
class object final {
public:
    explicit object(std::string arg_name) : name(std::move(arg_name)), _v() {}

    template <typename T>
    object& add(std::string&& key, T&& value) {
        using type = typename std::remove_cv<typename std::remove_reference<T>::type>::type;
        _v.emplace(std::move(key), serializer<type>().serialize(std::forward<T>(value)));
        return *this;
    }

    /** ... */

private:
    std::string name;
    std::map<std::string, data_point> _v;
}
```

Serializing Test Cases

```
std::vector<uint8_t> Testcase::serialize(const std::vector<Testcase>& testcases) {  
    flatbuffers::FlatBufferBuilder builder;  
    std::vector<flatbuffers::Offset<fbs::MessageBuffer>> messageBuffers;  
    for (const auto& tc : testcases) {  
        const auto& out = tc.flatbuffers();  
        messageBuffers.push_back(fbs::CreateMessageBufferDirect(builder, &out));  
    }  
    const auto& messages = fbs::CreateMessagesDirect(builder, &messageBuffers);  
    builder.Finish(messages);  
    const auto& ptr = builder.GetBufferPointer();  
    return {ptr, ptr + builder.GetSize()};  
}
```

Serializing Data Points

```
std::vector<uint8_t> Testcase::flatbuffers() const {  
    /** ... */  
    std::vector<flatbuffers::Offset<fbs::Result>> fbsResultEntries;  
    for (const auto& result : _resultsMap) {  
        const auto& key = result.first.c_str();  
        const auto& value = result.second.val.serialize(builder);  
        fbsResultEntries.push_back(fbs::CreateResultDirect(builder, key, value));  
    }  
    /** ... */  
}
```

```
flatbuffers::Offset<fbs::TypeWrapper> data_point::serialize(  
    flatbuffers::FlatBufferBuilder& builder) const {  
    return touca::detail::visit(data_point_serializer_visitor(builder), _value);  
}  
)
```