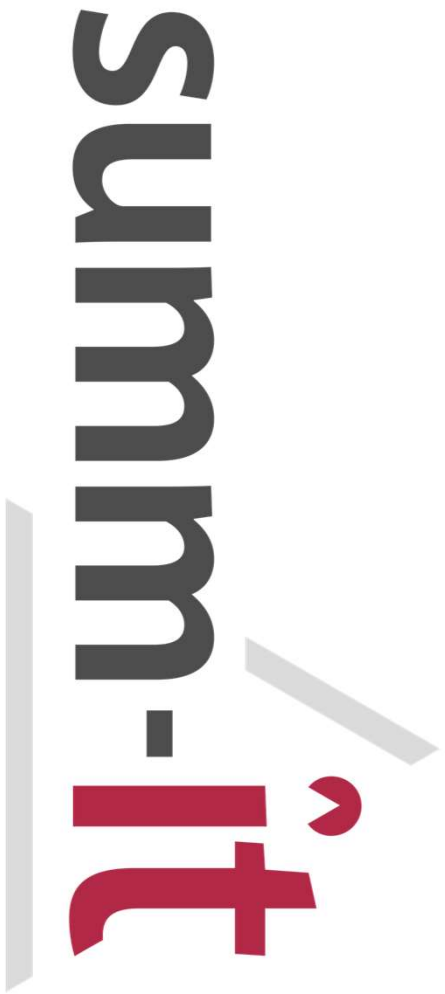


utPLSQL - unit tests, code coverage test, CI/CD integration and much more...

Mariusz.Masewicz@summ-it.pl

“We want to be sure that our customer reaches the summit as Tenzing Norgay did.”

Agenda

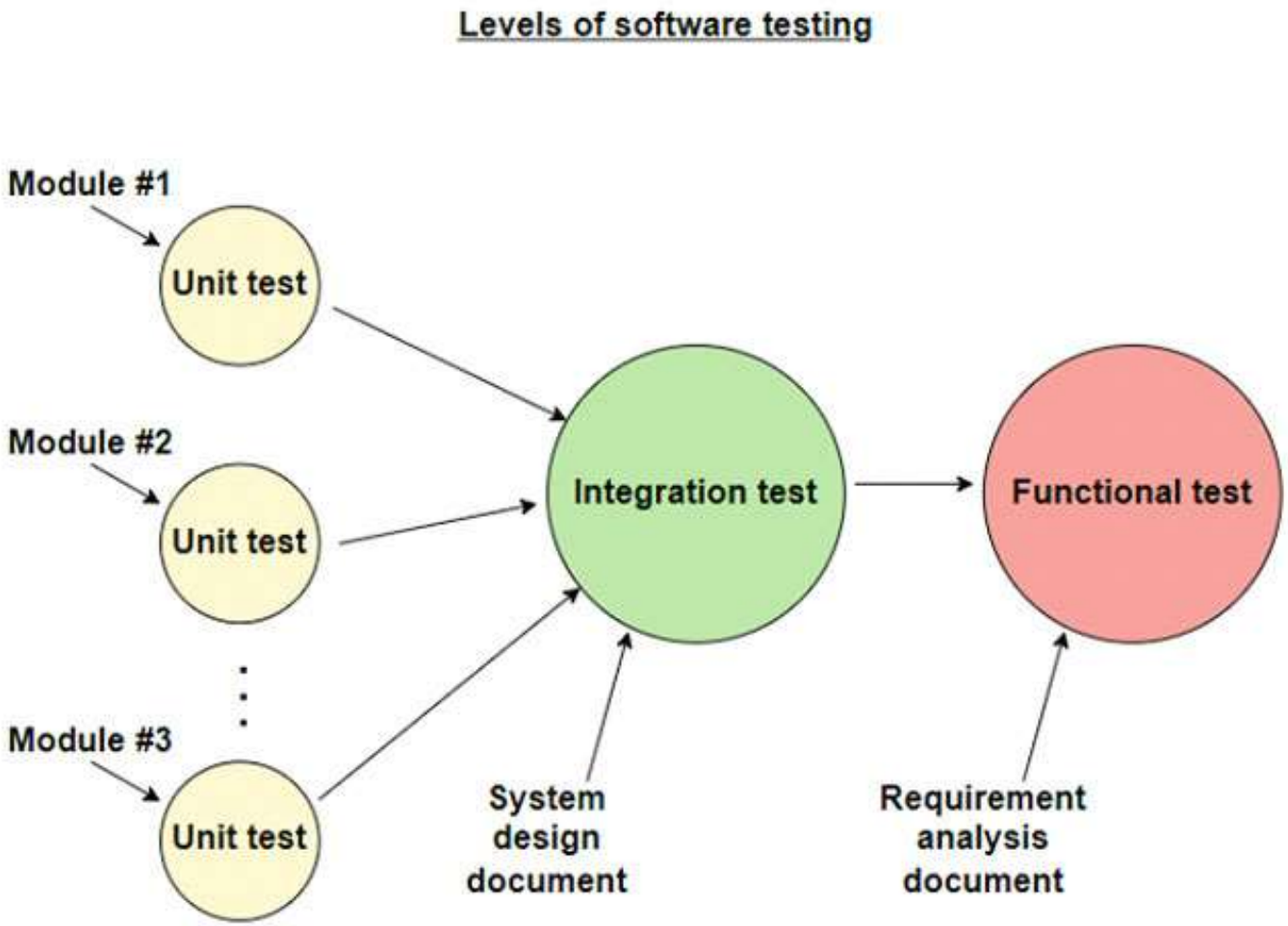
1) What is utPLSQL
2) Installation
3) Simple tests
4) A little bit more advanced tests
5) Exceptions
6) Code covereage
7) CI/CD integration
8) Real life examples

What is utPLSQL

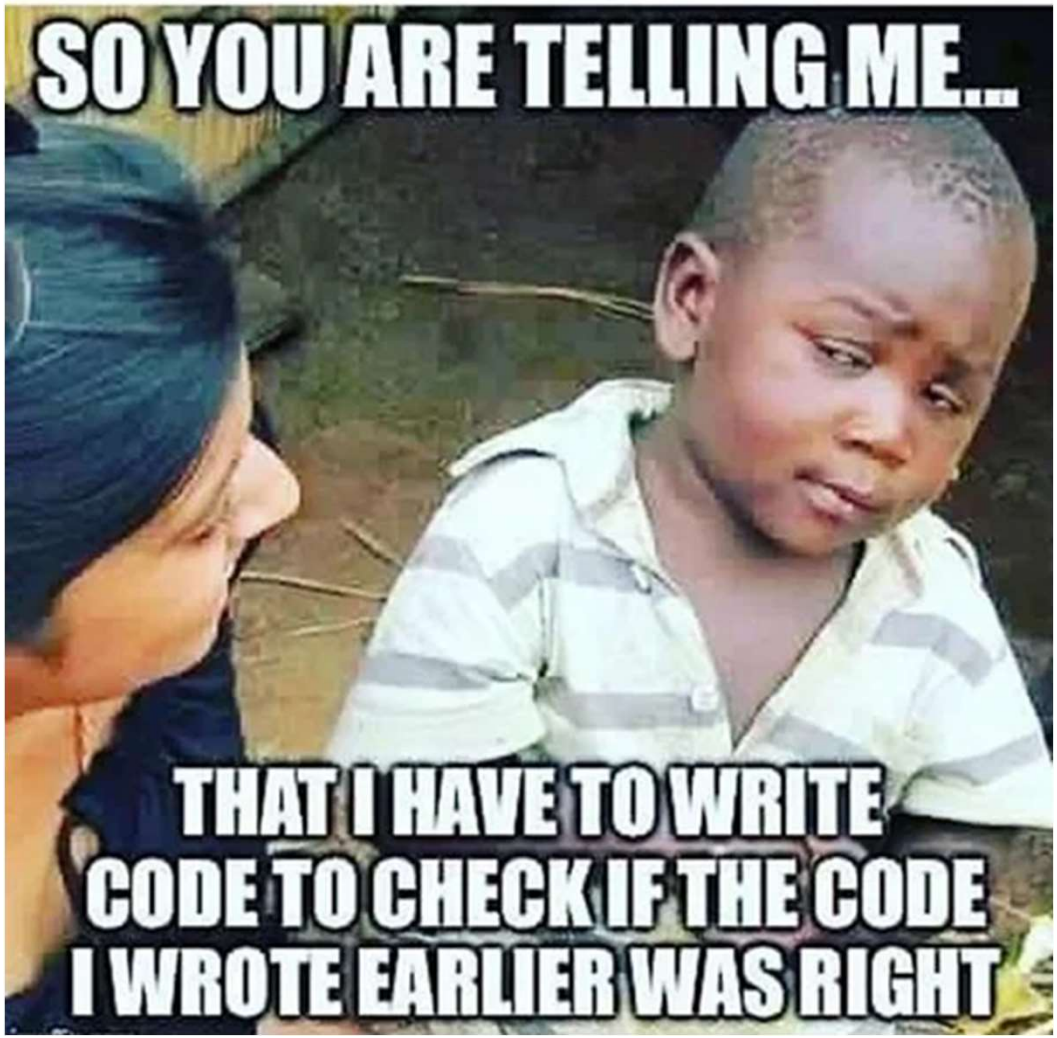
https://www.utplsql.org/
https://github.com/utPLSQL/utPLSQL
https://github.com/utPLSQL/utPLSQL-PLSQL-Developer/releases
https://github.com/utPLSQL/utPLSQL-SQLDeveloper
https://github.com/triologygmbh/utPLSQL-APEX
https://github.com/MariuszMasewicz/POUG_2023_utplsql

utPLSQL is a Unit Testing framework for Oracle PL/SQL and SQL. The framework follows industry standards and best patterns of modern Unit Testing frameworks like JUnit and Rspec
Requirements:
Version of Oracle under extended support (Currently 11.2 and above)

Levels of software testing



Praxis


SO YOU ARE TELLING ME...
THAT I HAVE TO WRITE CODE TO CHECK IF THE CODE I WROTE EARLIER WAS RIGHT

Key features

- multiple ways to compare data with matchers
- native comparison of complex types (objects/collections/cursors)
- in-depth and consistent reporting of failures and errors for tests
- tests identified and configured by annotations
- hierarchies of test suites configured with annotations
- automatic (configurable) transaction control
- Build-in coverage reporting
- Integration with SonarQube, Coveralls, Jenkins and Teamcity with reporters
- plugin architecture for reporters and matchers
- flexible and simple test invocation
- multi-reporting from test-run from command line

Installation

https://www.utplsql.org/utPLSQL/latest/userguide/install.html

https://stackoverflow.com/questions/66694500/installing-utplsql-in-oracle-cloud

Run as admin user:

- utPLSQL\source\install.sql
- utPLSQL\source\create_synonyms_and_grants_for_public.sql

Simple tests

https://www.utplsql.org/utPLSQL/latest/userguide/expectations.html

Expectation is a combination of:

- the expected value
- optional custom message for the expectation
- the matcher used to perform comparison
- the matcher parameters (actual value), depending on the matcher type

Simple tests – my perfect application package

```
1  create or replace PACKAGE BODY                ADVANCED_MATH AS
2
3    function plus(p_number1 number,  p_number2 number) return number AS
4    BEGIN
5      RETURN p_number1+p_number2;
6    END plus;
7
8    function subtract(p_number1 number,  p_number2 number) return number AS
9    BEGIN
10     RETURN p_number1-p_number2;
11   END subtract;
12
13   function multiply(p_number1 number,  p_number2 number) return number AS
14   BEGIN
15     RETURN p_number1*p_number2;
16   END multiply;
17
18   function divide(p_number1 number,  p_number2 number) return number AS
19   BEGIN
20     RETURN p_number1/p_number2;
21   END divide;
22
23 END ADVANCED_MATH;
```

Simple tests – first tests

```
set serveroutput on
--https://www.utplsql.org/utPLSQL/latest/userguide/expectations.html#expectation-concepts
begin
ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1) ).to_equal(2);
end;
/
```

```
SUCCESS
  Actual: 2 (number) was expected to equal: 2 (number)
```

```
begin
ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1) ).to_equal(3);
end;
/
```

```
FAILURE
  Actual: 2 (number) was expected to equal: 3 (number)
  at "anonymous block", line 2
```

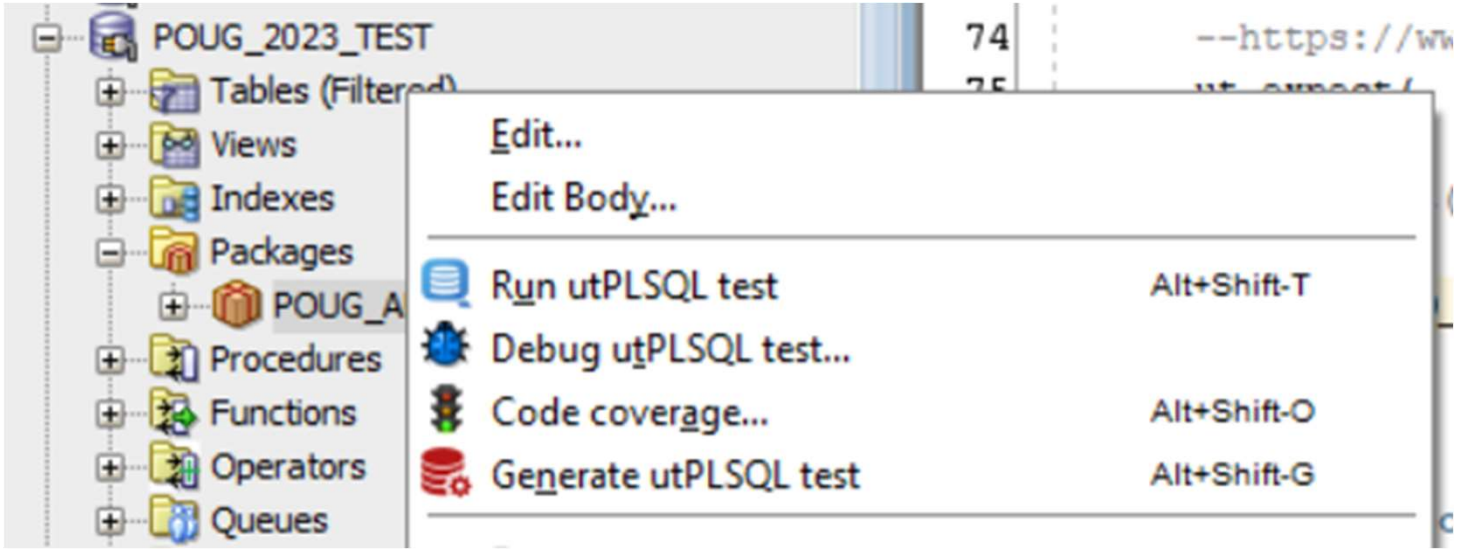
Simple tests – first test package

```
CREATE OR REPLACE PACKAGE POUG_2023_TEST.POUG_ADVANCED_MATH_simple_tests
IS
--%suite(Simple tests)
-- %suitepath(poug.devtests)

--%test(Plus 1 1)
   PROCEDURE ut_plus_1_1;
END poug_ADVANCED_MATH_simple_tests;
/
```

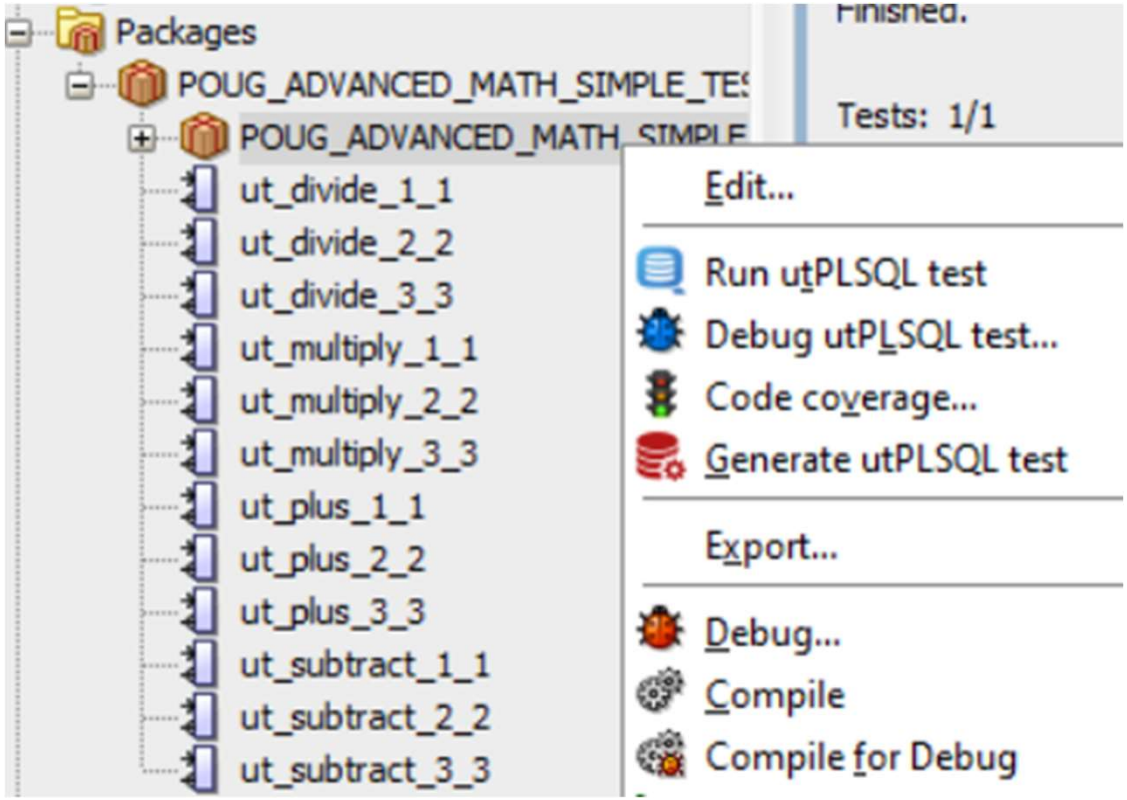
```
CREATE OR REPLACE PACKAGE BODY POUG_2023_TEST.POUG_ADVANCED_MATH_simple_tests
IS

  PROCEDURE ut_plus_1_1
   IS
   BEGIN
      --https://www.utplsql.org/utPLSQL/latest/userguide/expectations.html#matchers
      ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1), 'it is not so easy to add 1 to 1' ).to_equal(2);
      ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1), 'null values are bad' ).to_be_not_null();
      ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1), 'not null values are bad' ).to_be_null();
   END;
end poug_ADVANCED_MATH_simple_tests;
/
```

Simple tests – tests running

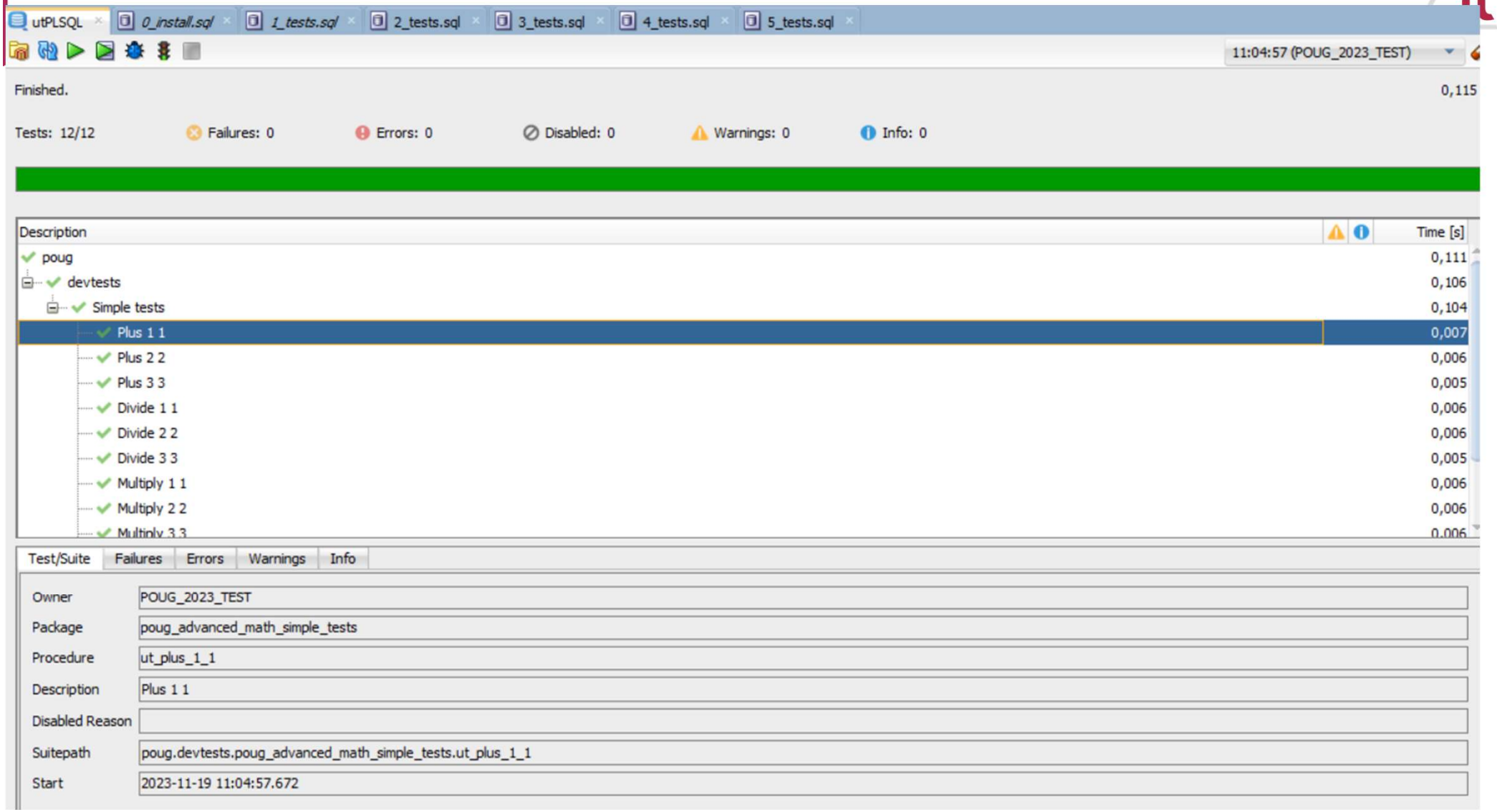
```
set serveroutput on
exec ut.run('poug_ADVANCED_MATH_simple_tests.ut_plus_1_1');
exec ut.run('poug_ADVANCED_MATH_simple_tests');
exec ut.run();
```



Simple tests – more test cases



Simple tests – test results



Matchers

https://www.utplsql.org/utPLSQL/latest/userguide/expectations.html#matchers

utPLSQL provides the following matchers to perform checks on the expected and actual values.

- be_between(a_upper_bound {data-type}, a_lower_bound {data-type})
- be_empty()
- be_false()
- be_greater_than(a_expected {data-type})
- be_greater_or_equal(a_expected {data-type})
- be_less_or_equal(a_expected {data-type})
- be_less_than(a_expected {data-type})
- be_like(a_mask {varchar2} [, a_escape_char {varchar2}])
- be_not_null()
- be_null()
- be_true()
- equal(a_expected {data-type} [, a_nulls_are_equal {boolean}])
- contain(a_expected {data-type})
- have_count(a_expected {integer})
- match(a_patter {varchar2} [, a_modifiers {varchar2}])

Annotations

https://www.utplsql.org/utPLSQL/v3.1.7/userguide/annotations.html

Annotations are used to configure tests and suites in a declarative way.

- test configuration is stored along with the test logic inside the test package.
- no configuration files or tables are needed.
- the annotation names are based on popular testing frameworks such as JUnit.
- the framework runner searches for all the suitable annotated packages, automatically configures suites, forms the suite hierarchy, executes it and reports results in specified formats.

Supported annotations

- Suite
- Test
- Disabled
- Beforeall
- Afterall
- Beforeeach
- Aftereach
- Beforetest
- Aftertest
- Context
- Tags
- Suitepath
- Rollback
- Throws

A little bit more advanced tests

Order of execution
Data manipulation
Cursor comparission
Exceptions
Bugfixing

A little bit more advanced tests – order of execution

```
CREATE OR REPLACE PACKAGE
POUG_2023_TEST.POUG_ORDER_OF_EXECUTION_tests
IS
--%suite(Order of execution)
--%suitepath(poug.devtests)

--%beforeall
  procedure before_all_1;
  --%beforeall
  procedure before_all_2;
  --%afterall
  procedure after_all;
  --%beforeeach
  procedure before_each;
procedure before_test1;
procedure before_test2;
--%test(test1)
--%beforetest(before_test1)
   PROCEDURE test1;
--%test(test2)
--%beforetest(before_test2)
   PROCEDURE test2;
END poug_ORDER_OF_EXECUTION_tests;
/
```

```
poug
  devtests
    Order of execution
      before_all_1
      before_all_2
      test1 [.008 sec]
      before_each
      before_test1
      test1
      test2 [.007 sec]
      before_each
      before_test2
      test2
      after_all

Finished in .048857 seconds
2 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – order of execution

```
CREATE OR REPLACE PACKAGE
POUG_2023_TEST.POUG_ORDER_OF_EXECUTION_tests
IS
--%suite(Order of execution)
--%suitepath(poug.devtests)

--%beforeall
  procedure before_all_1;
  --%beforeall
  procedure before_all_2;
  --%afterall
  procedure after_all;
  --%beforeeach
  procedure before_each;
procedure before_test1;
procedure before_test2;
--%test(test1)
--%beforetest(before_test1)
   PROCEDURE test1;
--%test(test2)
--%beforetest(before_test2)
   PROCEDURE test2;
END poug_ORDER_OF_EXECUTION_tests;
/
```

```
poug
  devtests
    Order of execution
      before_all_1
      before_all_2
      test1 [.008 sec]
      before_each
      before_test1
      test1
      test2 [.007 sec]
      before_each
      before_test2
      test2
      after_all

Finished in .048857 seconds
2 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – data manipulation

```
CREATE OR REPLACE PACKAGE
POUG_2023_TEST.POUG_ADVANCED_MATH_select_tests
IS
--%suite(Select tests)
--%suitepath(poug.devtests)
--%rollback(manual)

   PROCEDURE ut_setup;
--%test(Plus select)
--%beforetest(ut_setup)
   PROCEDURE ut_plus;
END poug_ADVANCED_MATH_select_tests;
```

```
PROCEDURE ut_setup IS
BEGIN
   execute immediate 'truncate table ADVANCED_MATH_SELECT_TEST_TABLE';
   Insert into POUG_2023_TEST.ADVANCED_MATH_SELECT_TEST_TABLE (NUMBER1,NUMBER2) values ('1','1');
   Insert into POUG_2023_TEST.ADVANCED_MATH_SELECT_TEST_TABLE (NUMBER1,NUMBER2) values ('2','2');
   Insert into POUG_2023_TEST.ADVANCED_MATH_SELECT_TEST_TABLE (NUMBER1,NUMBER2) values ('3','3');
   commit;
END;
```

A little bit more advanced tests – data manipulation

```
PROCEDURE ut_plus
 IS
  l_actual   sys_refcursor;
  l_expected sys_refcursor;
 BEGIN
    open l_expected for select 1 as number1,1 as number2,2 as result from dual
                          union all
                        select 2,2,4 from dual
                          union all
                        select 3,3,6 from dual
                        --  union all
                        --select 4,4,8 from dual
                        ;
    open l_actual   for select number1, number2,
      POUG_2023_APP.ADVANCED_MATH.plus(  number1, number2) as result
      from ADVANCED_MATH_SELECT_TEST_TABLE;
    ut.expect( l_actual ).to_equal( l_expected );
 END;
```

```
poug
  devtests
    Select tests
      Plus select [.301 sec]

Finished in .304365 seconds
1 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – cursor comparrision

```
poug
  devtests
    Select tests
      Plus select [.494 sec] (FAILED - 1)

Failures:

  1) ut_plus
      Actual: refcursor [ count = 3 ] was expected to equal: refcursor [ count = 4 ]
      Diff:
      Rows: [ 1 differences ]
        Row No. 4 - Missing:  <NUMBER1>4</NUMBER1><NUMBER2>4</NUMBER2><RESULT>8</RESULT>
      at "POUG_2023_TEST.POUG_ADVANCED_MATH_SELECT_TESTS.UT_PLUS", line 29 ut.expect( l_actual ).to_equal( l_expected );

Finished in .499169 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – cursor comparrision

```
poug
  devtests
    Select tests
      Plus select [.107 sec] (FAILED - 1)

Failures:

  1) ut_plus
      Actual: refcursor [ count = 3 ] was expected to equal: refcursor [ count = 3 ]
      Diff:
      Columns:
        Column <NUMBER1> data-type is invalid. Expected: CHAR, actual: NUMBER.
      Rows: [ all different ]
        All rows are different as the columns position is not matching.
      at "POUG_2023_TEST.POUG_ADVANCED_MATH_SELECT_TESTS.UT_PLUS", line 29 ut.expect( l_actual ).to_equal( l_expected );

Finished in .110102 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – cursor comparrision

```
poug
  devtests
    Select tests
      Plus select [.099 sec] (FAILED - 1)

Failures:

  1) ut_plus
      Actual: refcursor [ count = 3 ] was expected to equal: refcursor [ count = 3 ]
      Diff:
      Columns:
        Column <1> [data-type: NUMBER] is missing. Expected column position: 4.
      Rows: [ all different ]
        All rows are different as the columns position is not matching.
      at "POUG_2023_TEST.POUG_ADVANCED_MATH_SELECT_TESTS.UT_PLUS", line 29 ut.expect( l_actual ).to_equal( l_expected );

Finished in .102421 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – many functions

```
--%test(All operations select)
--%beforetest(ut_setup)
   PROCEDURE ut_all_operations
```

```
PROCEDURE ut_all_operations
IS
 l_actual   sys_refcursor;
 l_expected sys_refcursor;
BEGIN
   open l_expected for select 1 as number1,1 as number2,
                        2 as plus, 1 as divide,
                        1 as multiply, 0 as subtract from dual
                          union all
                        select 2,2,4,1,4,0 from dual
                          union all
                        select 3,3,6,1,9,0 from dual
                        --  union all
                        --select 4,4,8 from dual
                        ;
   open l_actual   for select number1, number2,
     POUG_2023_APP.ADVANCED_MATH.plus(  number1, number2) as plus,
     POUG_2023_APP.ADVANCED_MATH.divide(  number1, number2) as divide,
     POUG_2023_APP.ADVANCED_MATH.multiply(  number1, number2) as multiply,
     POUG_2023_APP.ADVANCED_MATH.subtract(  number1, number2) as subtract
     from ADVANCED_MATH_SELECT_TEST_TABLE;
   ut.expect( l_actual ).to_equal( l_expected );
END;
```

A little bit more advanced tests – data manipulation

```
--%test(All operations update)
--%beforetest(ut_setup)
   PROCEDURE ut_all_operations_update;
```

```
PROCEDURE ut_all_operations_update
IS
 l_actual   sys_refcursor;
 l_expected sys_refcursor;
BEGIN
   open l_expected for select 1 as number1,1 as number2,
                      2 as plus, 1 as divide, 1 as multiply,
                      0 as subtract from dual
                            union all
                           select 2,2,4,1,4,0 from dual
                            union all
                           select 3,3,6,1,9,0 from dual
                           --  union all
                           --select 4,4,8 from dual
                           ;

   update ADVANCED_MATH_SELECT_TEST_TABLE
   set
   plus = POUG_2023_APP.ADVANCED_MATH.plus(  number1, number2),
   divide = POUG_2023_APP.ADVANCED_MATH.divide(  number1, number2),
   multiply = POUG_2023_APP.ADVANCED_MATH.multiply(  number1, number2),
   subtract = POUG_2023_APP.ADVANCED_MATH.subtract(  number1, number2);

   open l_actual   for select * from ADVANCED_MATH_SELECT_TEST_TABLE;
   ut.expect( l_actual ).to_equal( l_expected );
END;
```

A little bit more advanced tests – exceptions

```
PROCEDURE ut_divide_by_zero
 IS
  l_actual   sys_refcursor;
  l_expected sys_refcursor;
 BEGIN
    ut.expect(  POUG_2023_APP.ADVANCED_MATH.divide( 1,  0) ).to_equal(1);
 END;
```

```
--%test(Divide by zero)
   PROCEDURE ut divide by zero;
```

```
poug
  devtests
    Exception tests
      Divide by zero [.005 sec] (FAILED - 1)

Failures:

  1) ut_divide_by_zero
      ORA-01476: divisor is equal to zero
      ORA-06512: at "POUG_2023_APP.ADVANCED_MATH", line 20
      ORA-06512: at "POUG_2023_TEST.POUG_ADVANCED_MATH_EXCEPTION_TESTS", line 9
      ORA-06512: at "POUG_2023_APP.ADVANCED_MATH", line 20
      ORA-06512: at "POUG_2023_TEST.POUG_ADVANCED_MATH_EXCEPTION_TESTS", line 9
      ORA-06512: at line 6
Finished in .008936 seconds
1 tests, 0 failed, 1 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – exceptions

```
--%test(Divide by zero)
--%throws(-01476)
   PROCEDURE ut_divide_by_zero;
```

```
poug
  devtests
    Exception tests
      Divide by zero [.022 sec]

Finished in .025806 seconds
1 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

```
function divide(p_number1 number,  p_number2 number)
BEGIN
  RETURN p_number1/p_number2;
EXCEPTION
  WHEN others then null;
END divide;
```

```
poug
  devtests
    Exception tests
      Divide by zero [.019 sec] (FAILED - 1)

Failures:

  1) ut_divide_by_zero
      Actual: -6503 was expected to equal: -1476
      ORA-06503: PL/SQL: Function returned without value
      ORA-06512: at "POUG_2023_APP.ADVANCED_MATH", line 29
      ORA-06512: at "POUG_2023_TEST.POUG_ADVANCED_MATH_EXCEPTION_TESTS", line 9
      ORA-06512: at "POUG_2023_APP.ADVANCED_MATH", line 29
      ORA-06512: at "POUG_2023_TEST.POUG_ADVANCED_MATH_EXCEPTION_TESTS", line 9
      ORA-06512: at line 6

Finished in .024453 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

A little bit more advanced tests – exceptions

```
function divide(p_number1 number,  p_number2 number) return number AS
BEGIN
  RETURN p_number1/p_number2;
EXCEPTION
  WHEN zero_divide
    then
      raise;
  WHEN others
    then
      null;
END divide;
```

```
poug
  devtests
    Exception tests
      Divide by zero [.003 sec]

Finished in .00736 seconds
1 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

Code coverage

utPLSQL comes with a built-in coverage reporting engine.
The code coverage reporting uses package DBMS_PROFILER (and DBMS_PLSQL_CODE_COVERAGE on Oracle database version 12.2 and above) provided with Oracle Database

The following code coverage reporters are supplied with utPLSQL:

- ut_coverage_html_reporter
- ut_coveralls_reporter – JSON
- ut_coverage_sonar_reporter – XML
- ut_coverage_cobertura_reporter

Code coverage

```
--https://www.utplsql.org/utPLSQL/latest/userguide/coverage.html
set serveroutput on
spool c:\utplsql-reports\poug_coverage.html
exec ut.run('POUG_2023_TEST', ut_coverage_html_reporter(),
   a_include_objects=>ut_varchar2_list('POUG_2023_APP.ADVANCED_MATH'),
   a_coverage_schemes=>ut_varchar2_list('POUG_2023_APP'));
spool off
```

POUG_2023_APP.ADVANCED_MATH

42.86 % lines covered

14 relevant lines. **6** lines covered and **8** lines missed

```
1.  PACKAGE BODY                ADVANCED_MATH AS
2.
3.    function plus(p_number1 number,  p_number2 number) return number AS
4.    BEGIN
5.      RETURN p_number1+p_number2;                                    9
6.    EXCEPTION
7.      WHEN others
8.        then
```

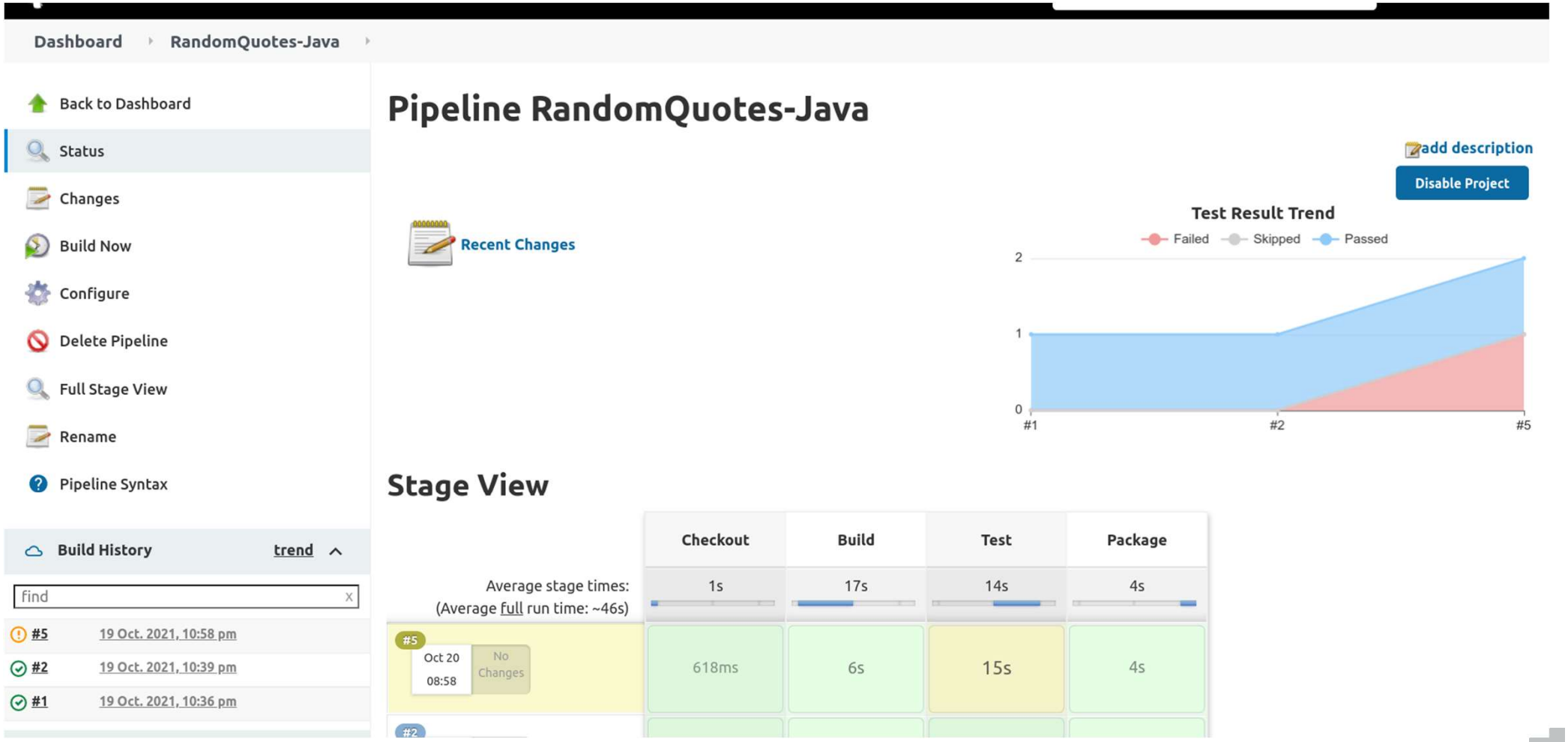
Test reporters – CI/CD integration

```
exec ut.run(ut_coverage_html_reporter ());
exec ut.run(ut_coveralls_reporter ());
exec ut.run(ut_coverage_sonar_reporter ());
exec ut.run(ut_coverage_cobertura_reporter ());

exec ut.run(ut_documentation_reporter());
exec ut.run(ut_documentation_reporter(), a_color_console=>true);
exec ut.run(ut_junit_reporter());
exec ut.run(ut_tfs_junit_reporter());
exec ut.run(ut_teamcity_reporter());
exec ut.run(ut_sonar_test_reporter());
exec ut.run(ut_debug_reporter());
```

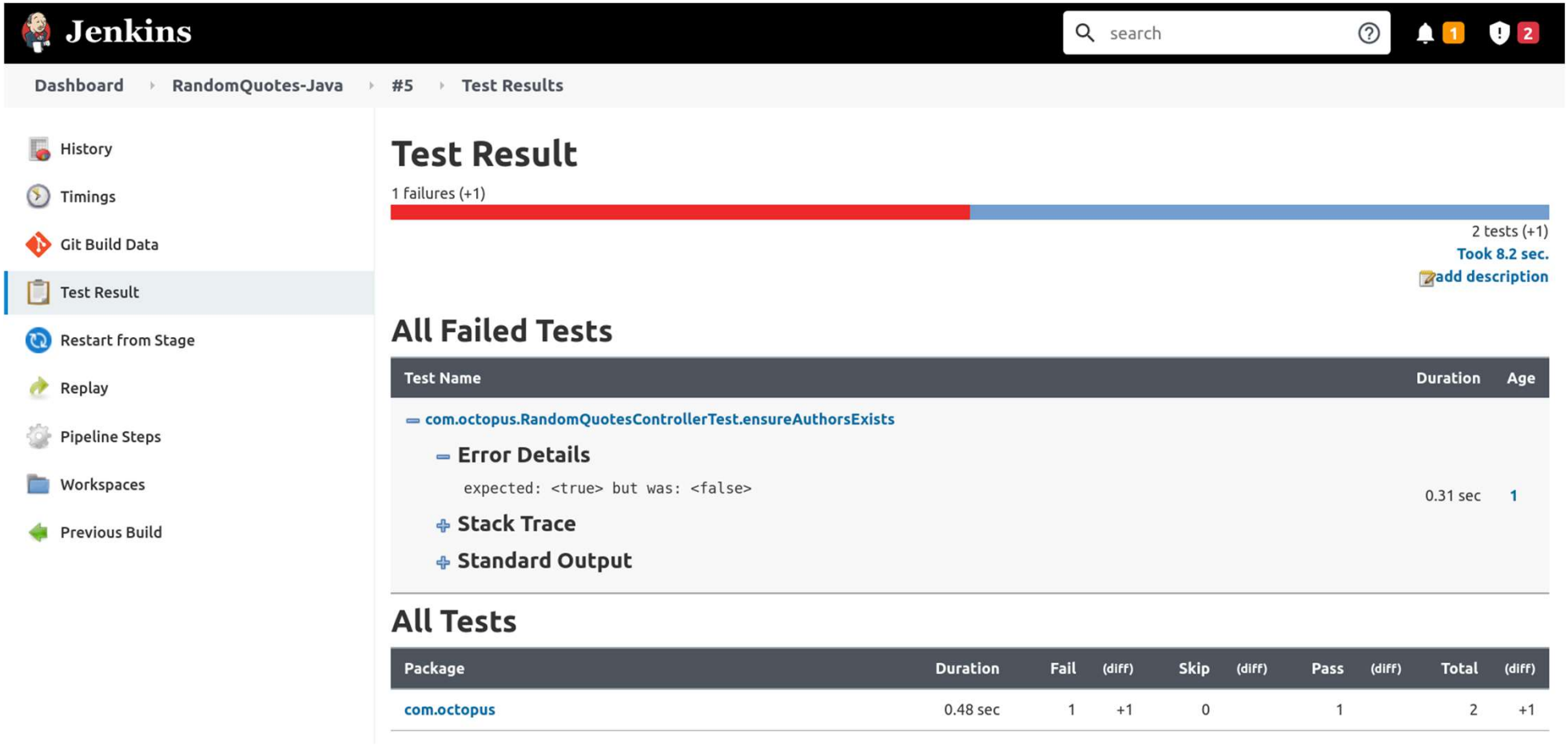
Test reporters – CI/CD integration

https://plugins.jenkins.io/junit/
https://qaautomation.expert/2023/10/21/how-to-create-junit-report-in-jenkins/
https://octopus.com/blog/jenkins-running-unit-tests

Test reporters – CI/CD integration



Test reporters – CI/CD integration



Bugs and bugfixes

```
--%test(bug#001 2+1 shoul be 4)
  PROCEDURE ut_bug_001_better_plus_function;
```

```
PROCEDURE ut_bug_001_better_plus_function
 IS
 BEGIN
    ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 2,  1) ).to_equal(4);
 END;
```

```
function plus(p_number1 number,  p_number2 number) return number AS
BEGIN
  -- for bug#001
  if (p_number1=2 and p_number2=1)
    then return 4;
  end if;
  RETURN p_number1+p_number2;
EXCEPTION
  WHEN others
    then
    POUG_ERRORS.log_error;
END plus;
```

Bugs and bugfixes – 2 bugs

```
--%test(bug#001 2+1 shoul be 4)
   PROCEDURE ut_bug_001_better_plus_function;

--%test(bug#002 1+1 shoul be 3)
   PROCEDURE ut_bug_002_best_plus_function;
```

```
PROCEDURE ut_bug_001_better_plus_function
 IS
 BEGIN
    ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 2,  1) ).to_equal(4);
 END;

PROCEDURE ut_bug_002_best_plus_function
 IS
 BEGIN
    ut.expect(  POUG_2023_APP.ADVANCED_MATH.plus( 1,  1) ).to_equal(3);
 END;
```

Bugs and bugfixes – 2 bugs

```
function plus(p_number1 number,  p_number2 number) return number AS
BEGIN

  if (p_number1=2 and p_number2=1) --bug#001
    then return 4;
  elsif (p_number1=1 and p_number2=1) --bug#002
    then return 3;
  end if;
  RETURN p_number1+p_number2;
EXCEPTION
  WHEN others
    then
    POUG_ERRORS.log_error;
END plus;
```

Bugs and bugfixes – 2 bugs

```
exec ut.run('poug_ADVANCED_MATH_bugfixes_tests');
```

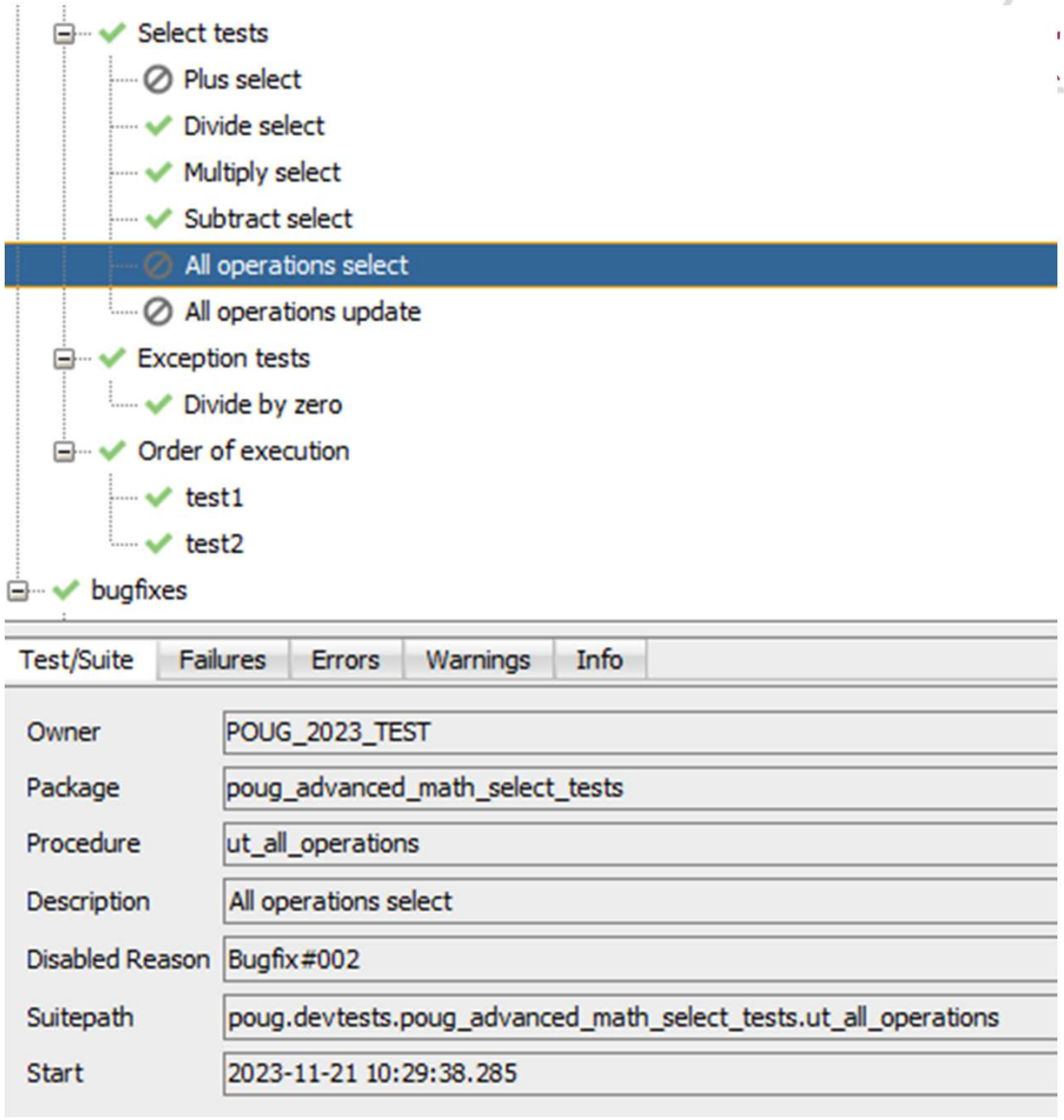
```
poug
  bugfixes
    poug_advanced_math_bugfixes_tests
      bug#001 2+1 shoul be 4 [.005 sec]
      bug#002 1+1 shoul be 3 [.004 sec]

Finished in .013958 seconds
2 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

```
exec ut.run('');
```

```
  3) ut_all_operations
      Actual: refcursor [ count = 3 ] was expected to equal: refcursor [ cour
      Diff:
      Rows: [ 1 differences ]
        Row No. 1 - Actual:   <PLUS>3</PLUS>
        Row No. 1 - Expected: <PLUS>2</PLUS>
      at "POUG_2023_TEST.POUG_ADVANCED_MATH_SELECT_TESTS.UT_ALL_OPERATIONS",

Finished in .843169 seconds
22 tests, 3 failed, 0 errored, 0 disabled, 0 warning(s)
```

Bugs and bugfixes – 2 bugs

```
--%test(Plus 1 1)
--%disabled(Bugfix#002)
   PROCEDURE ut_plus_1_1;
--%test(Plus 2 2)
   PROCEDURE ut_plus_2_2;
```



Testing best practices

https://www.utplsql.org/utPLSQL/latest/userguide/best-practices.html

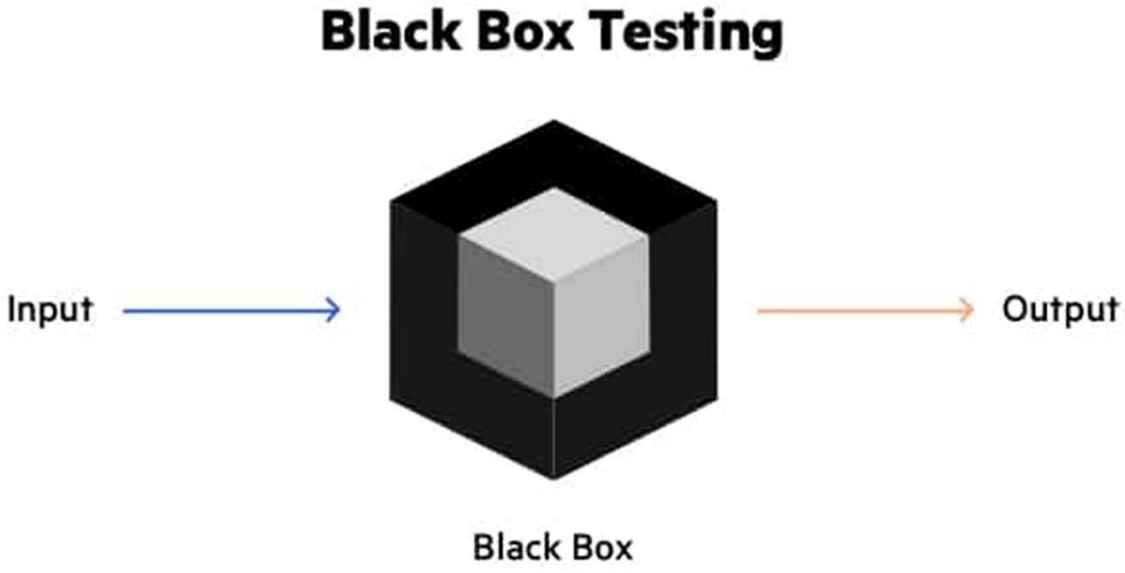
- large amount of text
- Tests are only valuable if they are executed frequently; ideally with every change to the project code
- large amount of text

How we are using utPLSQL

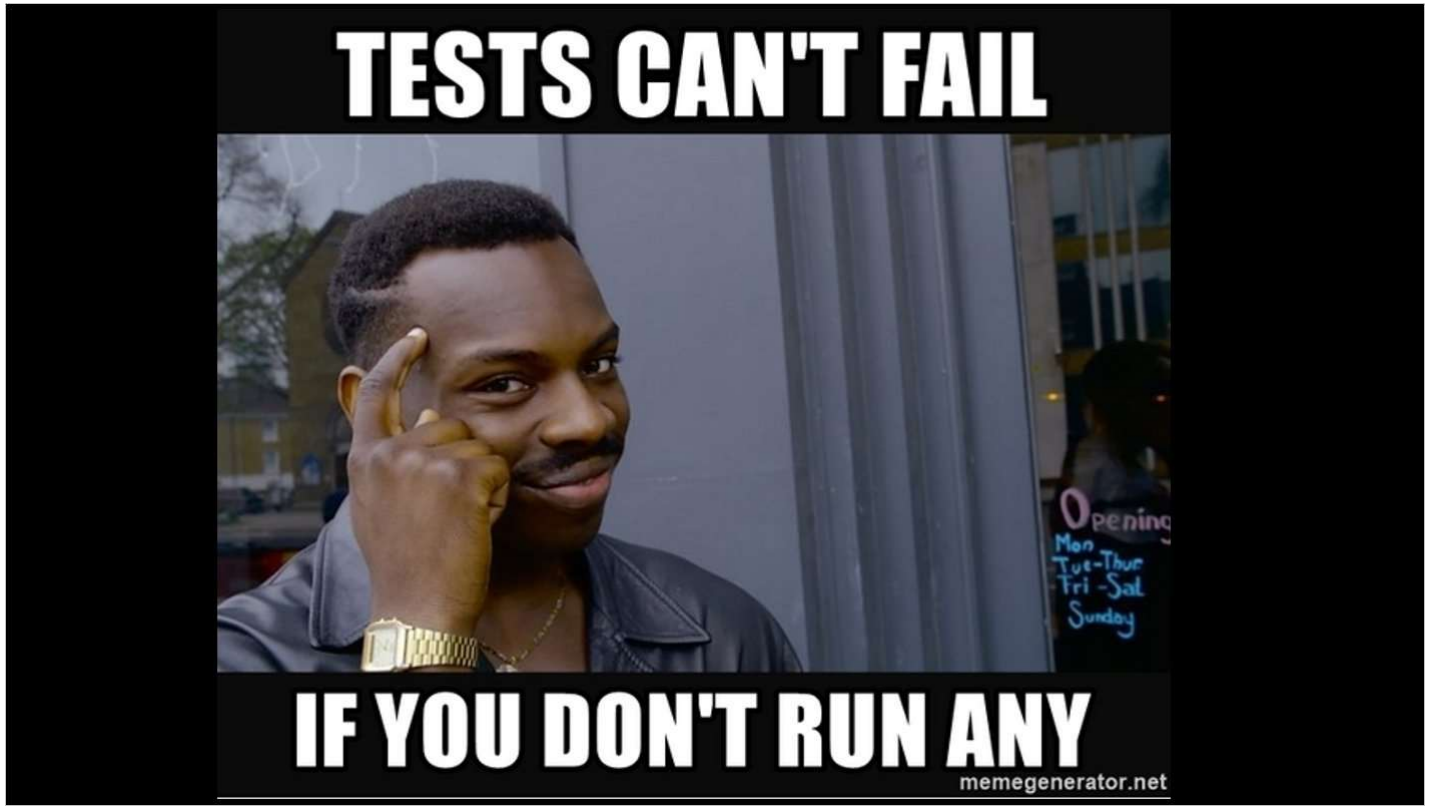
6 years
16 - developers
8000 - unit test
15774 – git commits
2840 - pull requests

Real project test scenarios

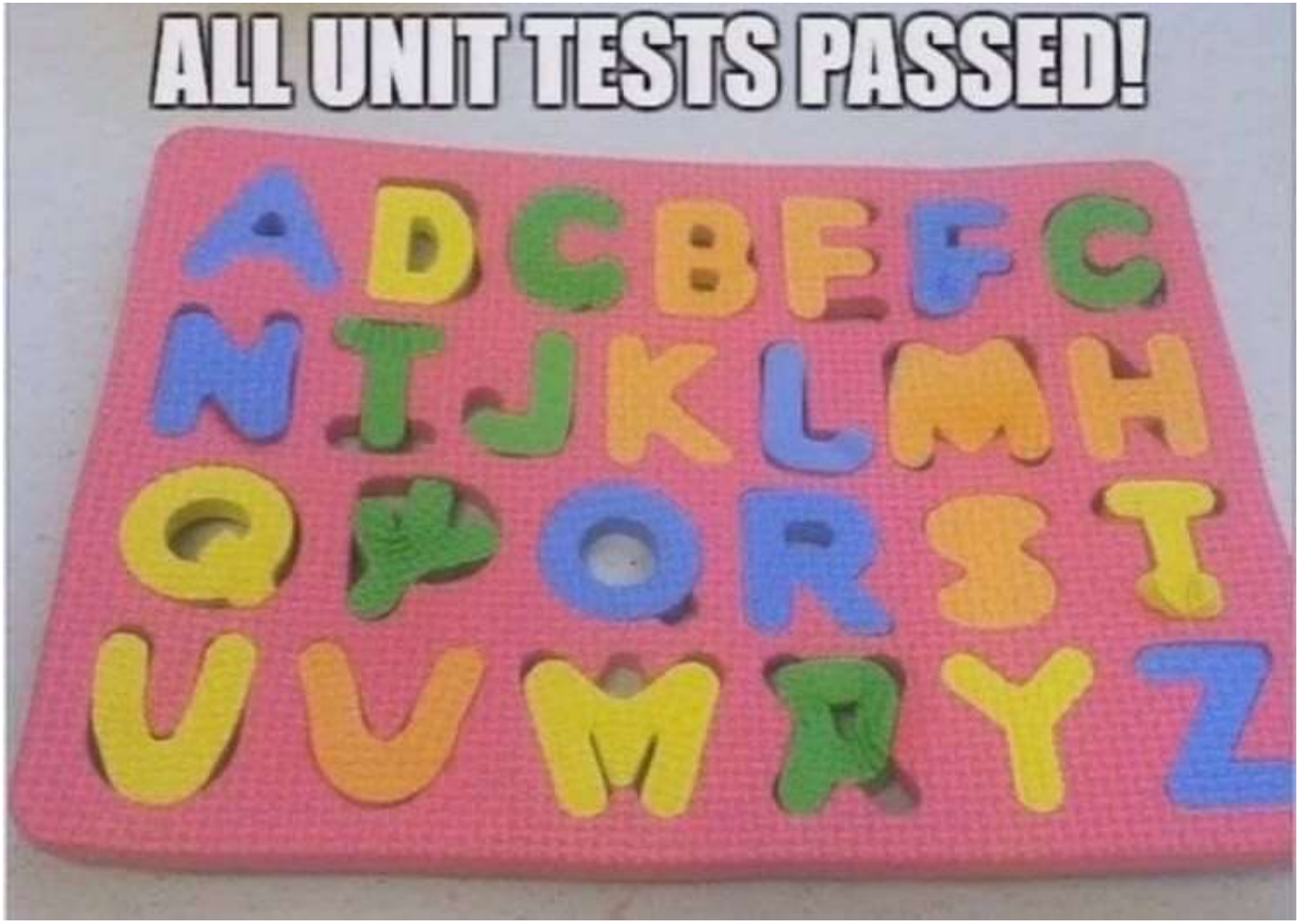
- Load data into source tables
- Start processing (multi stage workflow, jobs, …)
- Test result



Summary


TESTS CAN'T FAIL
Opening
Mon
Tue-Thur
Fri-Sat
Sunday
IF YOU DON'T RUN ANY
memegenerator.net

Summary



Contact us

Call for more information: +48 511 373 931

or ask a question via e-mail: info@summ-it.pl

Learn more about out offer: www.summ-it.pl/en