Tested for Business: An Open and Transparent Quality Kit

AdoptOpenJDK Quality Assurance

@ShelleyMLambert

AdoptOpenJDK Committer and TSC Member
Eclipse OpenJ9 Committer
IBM Runtime Technologies Test Lead

Setting the Stage

- Testing a popular/established language, Java
- Diverse set of commercial and open builds & distributions
- Times of immense change for Java ecosystem
  - large % of installed base no longer ‘free’
    - mass migration to free alternatives
  - new rapid release cadence
  - major language features (modularity)
  - new deployment models & workloads

What is Java?

- Java Development Kit (JDK) consists of
  - Java Runtime Environment (JRE)
  - Tools to compile and debug Java code for developing Java applications.

- JRE consists of:
  - Libraries
  - Java Virtual Machine (JVM)
  - Java Plugin (Applets!) and Java Web Start

What is OpenJDK?

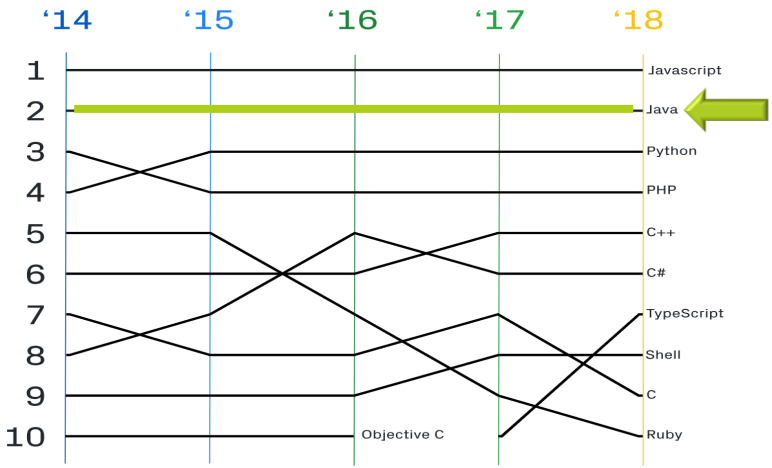
- The reference implementation (RI) from Java 7 SE onwards
- **Free and open source**
  - GNU General Public License, version 2, with the Classpath Exception
- Source to build your own Java; both the language and platform
  - Users like Twitter, Alibaba and Amazon have augmented the JDK with custom builds for their platforms
- **Many commercial and community builds** available

OpenJDK via multiple distributions & builders

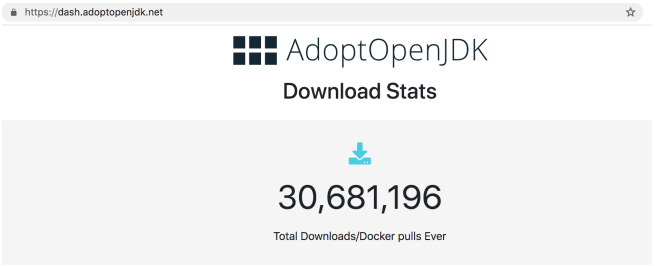
- Oracle's commercial JDK builds from OpenJDK and is evolving the two to be essentially identical.
- IBM's JDK is based on Eclipse OpenJ9, for Java 8 and onwards, a single VM which runs across many versions. OpenJ9 can be built as a component of OpenJDK
- SAPMachine from SAP, Zulu® from Azul, Corretto from Amazon…
- IcedTea is one of the earliest OpenJDK distros and blends OpenJDK and GNU Classpath.
  - IcedTea is currently bundled default with GNU/Linux distributions such as Fedora, Gentoo and Debian.
- **AdoptOpenJDK** (free, open, transparent community effort)
  - OpenJDK with OpenJ9 (fast, smaller, more efficient VM, great for containerized environments)
  - OpenJDK with Hotspot

Who uses Java?

- 2nd most prevalent language in github, ~9,000,000 developers, thousands of companies, millions of users



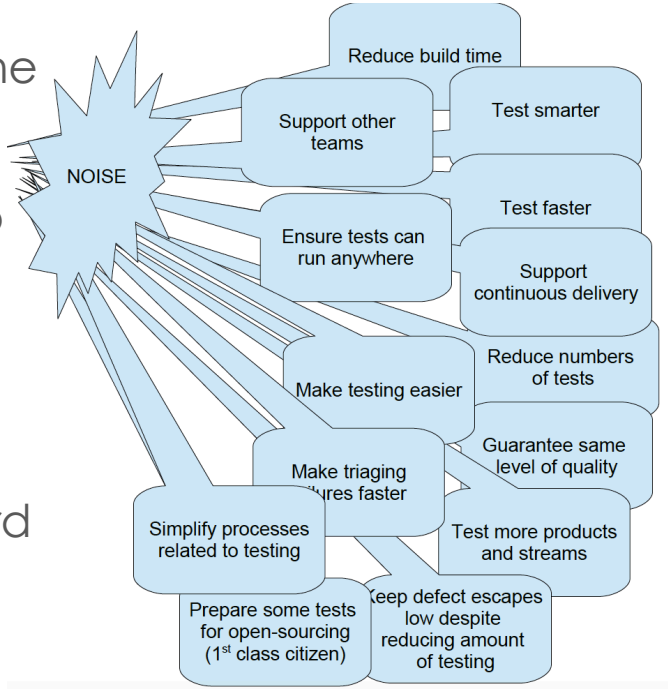
octoverse.github.com/projects#languages



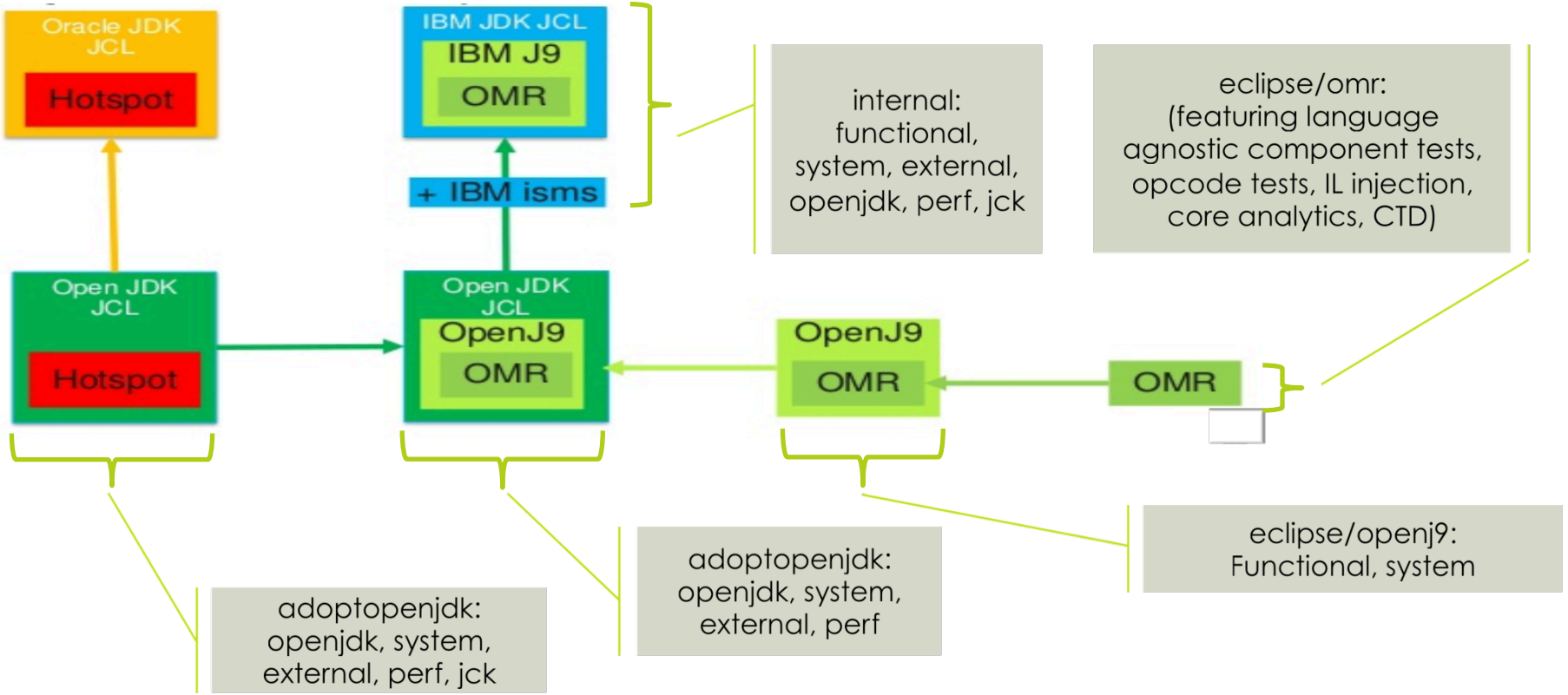
at May 16, 2019

Challenges for Testing

- Testing is private & fragmented (mostly done in-house by implementers)
- Inefficient use of testing resources (need to join forces for community benefit)
- Large number of active versions, implementations, platforms
- Many different test frameworks, no standard behaviour across test categories



Testing Scope



AdoptOpenJDK Quality Assurance

- "**Make quality certain to happen**"
- Testing a wide criteria representing actual business requirements to identify binaries ready for production usage

Today
Functional correctness
- OpenJDK regression (open)
- Oracle JCK (closed)

Builder-specific testing (unknown)

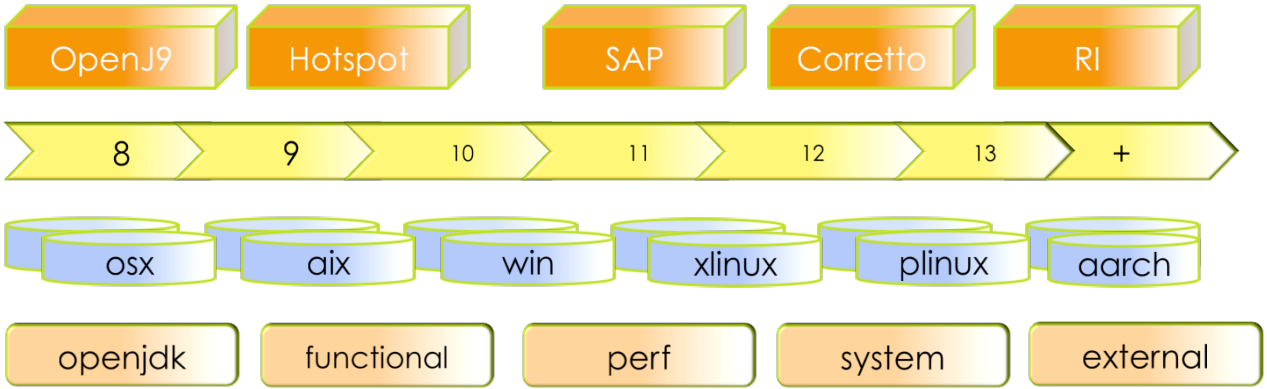
Roadmap
Security
- Passes known vulnerability tests

Functional correctness
- OpenJDK regression
- Eclipse functional
- Application & framework tests

Performance
- Published metrics
- Achieves minimum throughput scores

Scalability & durability
- Load & stress testing

Implementations, Versions, Platforms



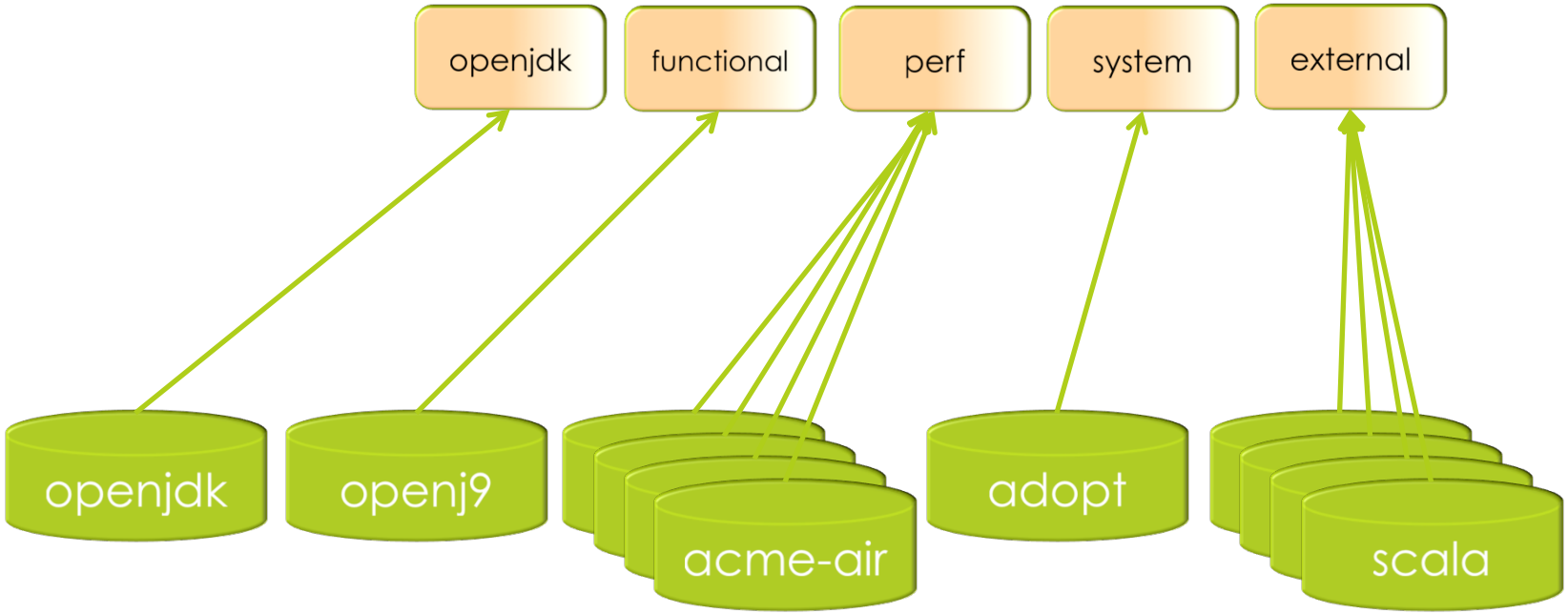
The fine print about #'s of tests and test output:
Vast amounts of data per day:

- 5 impls (openj9/14, hotspot/18, ibm/22, sap/1, corretto/3)
- sum([14,18,22,1,3]) = 58 impl_spec value
- 250,000+ unique tests
- 6 versions (8, ~~9~~, ~~10~~, 11, 12, Panama, Valhalla, Amber)
- ~36 variants (unique inputs / commandline options)

Impls_specsTotal x numTests x versions

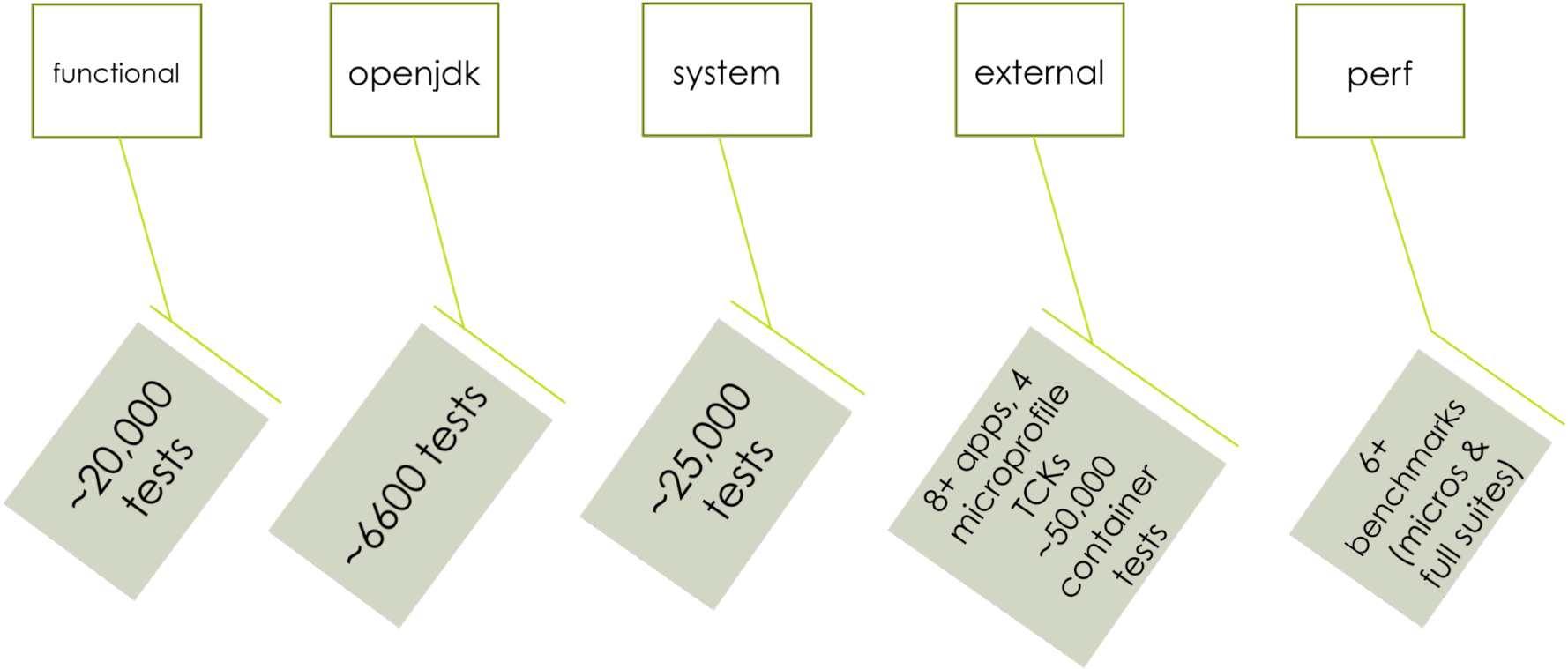
- 58x250000x6=87,000,000
- With variants -> 87,000,000x36=3,132,000,000 tests run

Test Curation

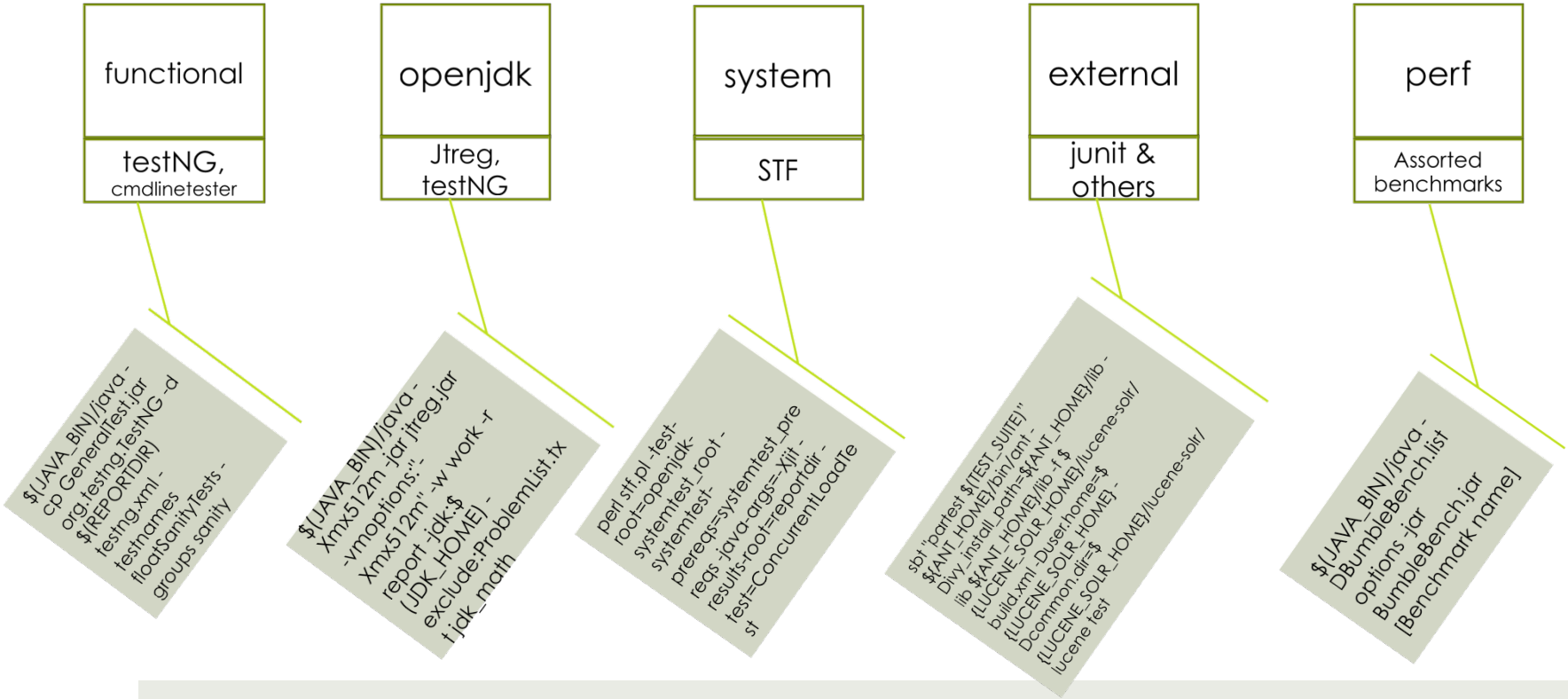


Git repos

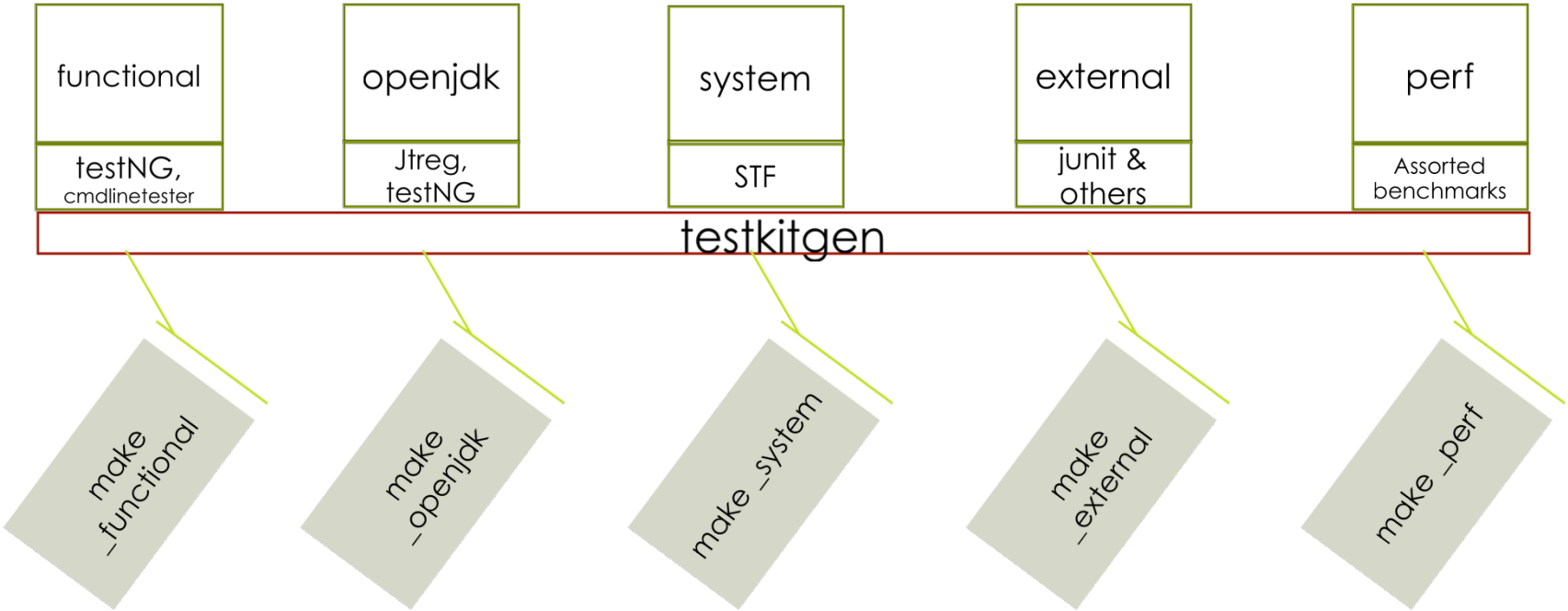
Gather Great Tests



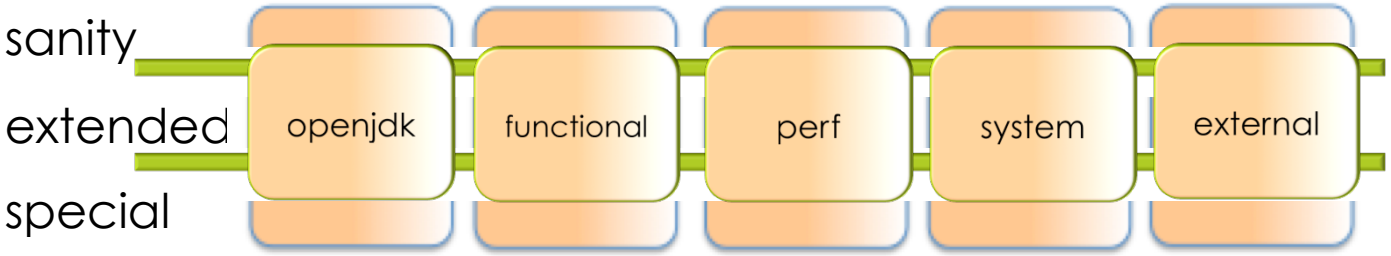
Gather Great Variety



Consolidate and Curate



Graduated Testing



- **sanity -** runs nightly & releases
  - sanity = sanity.openjdk + sanity.functional + sanity.perf + sanity.system + sanity.external
- **extended** – runs weekly & releases
  - extended = extended.openjdk + extended.functional + extended.perf + extended.system + extended.external
- **special** – runs on custom schedule & releases
  - special = special .openjdk + special .functional + special .perf + special .system + special.external

What Enterprises Require? & Developers

- **Functional** – runs the apps they care about, on platforms they care about
- **Secure** – patches applied & tested
- **Performant** – performance bar for certain types of operations
- **Scalable** – enterprise workloads

AQA Manifesto

- **open** & **transparent**
- **diverse** & **robust set** of test suites
- **evolution** alongside implementations
  - continual investment
  - process to modify
  - codecov & other **metrics**
  - comparative analysis
- **portable**
- **tag & publish**

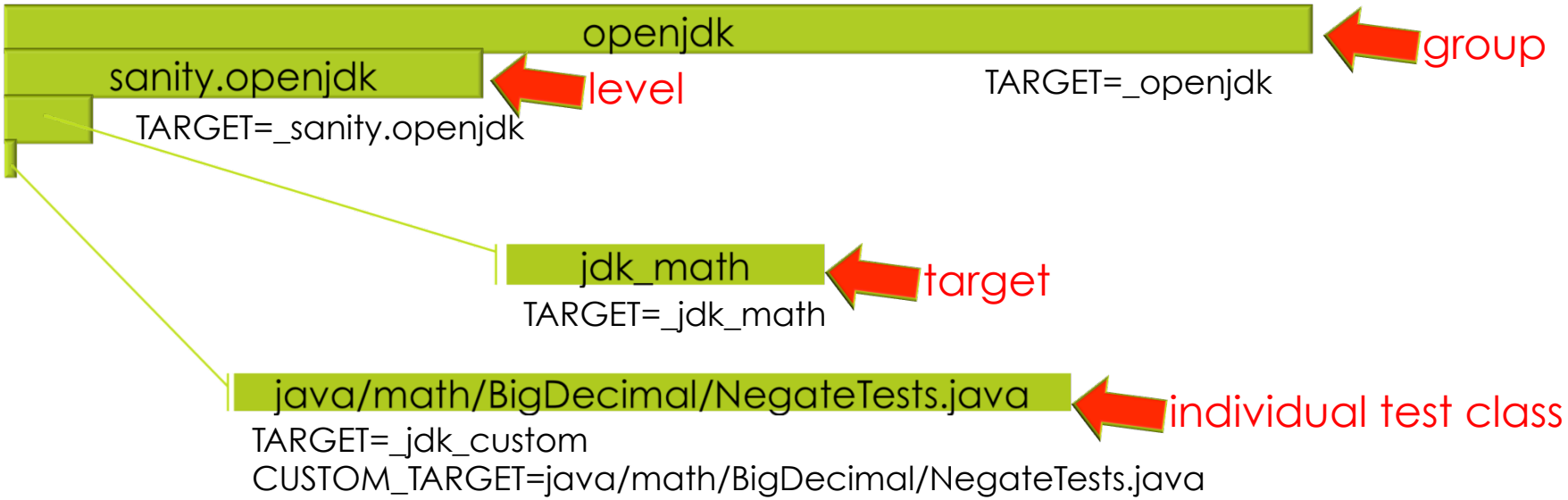
Open and Transparent

- **Why?**
  - **Open languages deserve open tests**
  - Test source open, executed in the open, results openly available…
  - Strengthens confidence
  - Tests get scrutiny, get fixes, get love
  - Drives innovation
  - **Builds community**

Diverse & Robust Tests

- **Why?**
  - Fulfill enterprise and developer requirements
- Cover different:
  - **categories**: functional/regression, security, performance, scalability, load/stress
  - **versions**
  - **platforms**
  - **implementations**
  - **applications** (& Microprofile TCKs)
- Have compelling, useful suites? **Contribute** them!

Grouping & Granularity



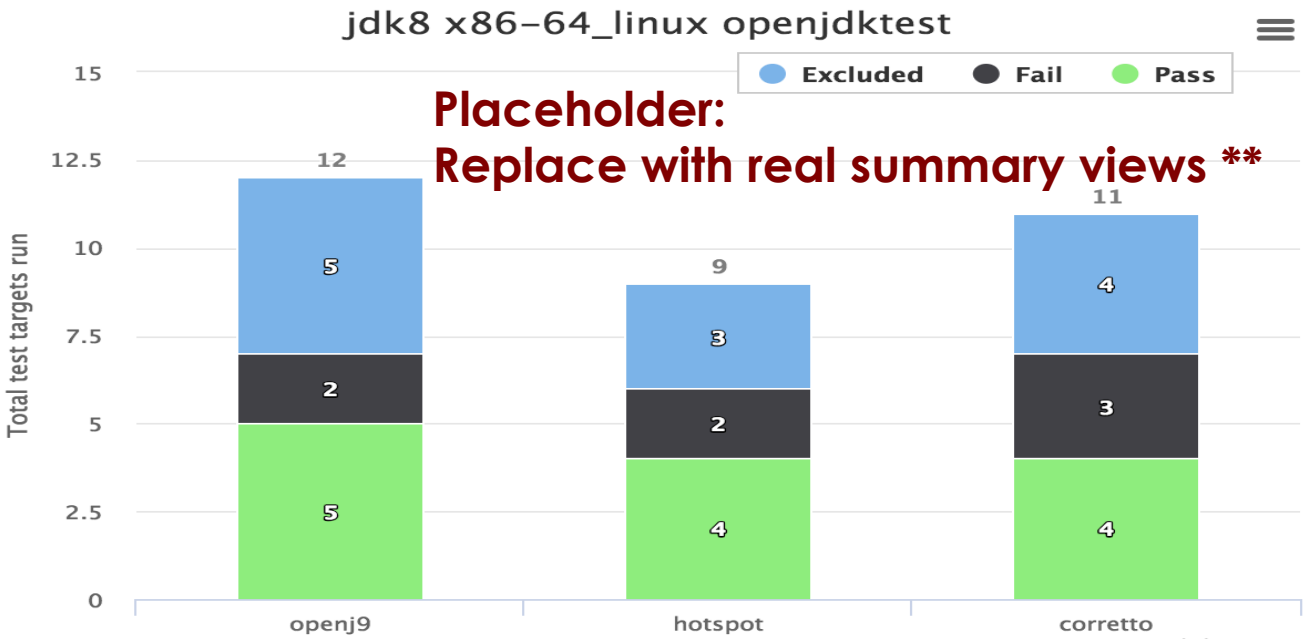
Define Tests (in playlists)

Examples

```
<!-- Exclude from JDK11 due to : https://github.com/AdoptOpenJDK/openjdk-systemtest/issues/178 -->
<test>
    <testCaseName>MathLoadTest_bigdecimal</testCaseName>
    <variations>
        <variation>NoOptions</variation>
    </variations>
    <command>export JAVA_HOME=$(JDK_HOME)$(D)  && \
perl $(SYSTEMTEST_RESROOT)$(D)stf$(D)stf.core$(D)scripts$(D)stf.pl \
-test-root=$(Q)$(SYSTEMTEST_RESROOT)$(D)stf;$(SYSTEMTEST_RESROOT)$(D)openjdk-systemtest$(Q) \
-systemtest-prereqs=$(Q)$(SYSTEMTEST_RESROOT)$(D)systemtest_prereqs$(Q) \
-java-args=$(Q)$(JVM_OPTIONS)$(Q) \
-results-root=$(REPORTDIR) \
-test=MathLoadTest \
-test-args=$(Q)workload=bigDecimal$(Q); \
$(TEST_STATUS)</command>
    <levels>
        <level>sanity</level>
    </levels>
    <groups>
        <group>system</group>
    </groups>
    <subsets>
        <subset>8</subset>
        <subset>9</subset>
        <subset>10</subset>
    </subsets>
</test>
```

Test Results Summary Service

trss.adoptopenjdk.net



Continual Investment

- **Centre of excellence**, lead not follow
- Tests (like the products they verify) need to continuously evolve & change
- Tests require maintenance & care
- Improve towards more effective/efficient
- Refining automation and tools
  - Automate re-inclusions upon fixes
  - Remove friction, make testing easier
  - Reduce process, make tools simpler

Process to Modify

- **TSC** test leads to define process to:
  - Review/assess existing & new tests (quarterly)
  - **Community awareness** of major additions/modifications/ deletions
  - Update current active set of tests (fixes, levels)

- **Guides** for plugging in new suites:
  - Test source plus build file & playlist file

Metrics

- **Gather data** to measure value/effectiveness of tests, inform process of change
- **Codecov**
  - Set bar to stay above
  - Shows gaps, limited, not functional coverage
- **Bug prediction**, most changed files, mutation testing, DL
- **Comparative analysis** as a test-the-tests method

Comparative analysis

- **Test-the-tests** mechanism
  - Easy triage pattern/flowchart
  - **Tools for easy 'diff'** of test results

- **Informs stakeholders**
  - Enterprises, open-source communities, developers
    - how did each impl do?
    - how stable/fast/secure is the new release?
    - compared to last release? last version?

Portable

- **Easily run**:
  - by any developer on laptop
  - on newly-added platforms, versions, implementations
  - in any implementers Jenkins farm

Tag & Publish on Releases

- **Audit trail**
  - On release builds:
    - Setting the bar, target pass rates, determining which failures/excludes ‘block’ releases
    - Keep test summary (pass/fail/excluded totals), acceptable bar
    - Keep track of versions of every test repo used in the overall release run
- **Reproducible**
  - 100% reproducible per release, automation get clone
- **Visibility** and storage of aggregate results on public site
- **Downloadable relevant artifacts** to match released binaries

AQA Manifesto

- **open** & **transparent**
- **diverse** & **robust set** of test suites
- **evolution** alongside impls
  - continual investment
  - process to modify
  - codecov & other **metrics**
  - comparative analysis
- **portable**
- **tag & publish**

Testing at AdoptOpenJDK

- Iterative refinement:
  - **Guided** by the **AQK manifesto**
  - **Sanitize** and **grow** our current suites
  - Actively seeking **participation**
  - Open to **contributions** of useful tests
  - Building **centre of excellence** for testing
  - **Collaboration** fuels **innovation** and **fun**!

Community building! Please join us!

	Website	Github	Twitter
	adoptopenjdk.net	AdoptOpenJDK/openjdk-tests	@adoptopenjdk
	eclipse.org/openj9	eclipse/openj9	@openj9
	eclipse.org/omr	eclipse.org/omr	@eclipseomr
	8thdaytesting.com	smlambert	@ShelleyMLambert

Model Building (deep learning)