ONOS Testing

ONOS Build 2017

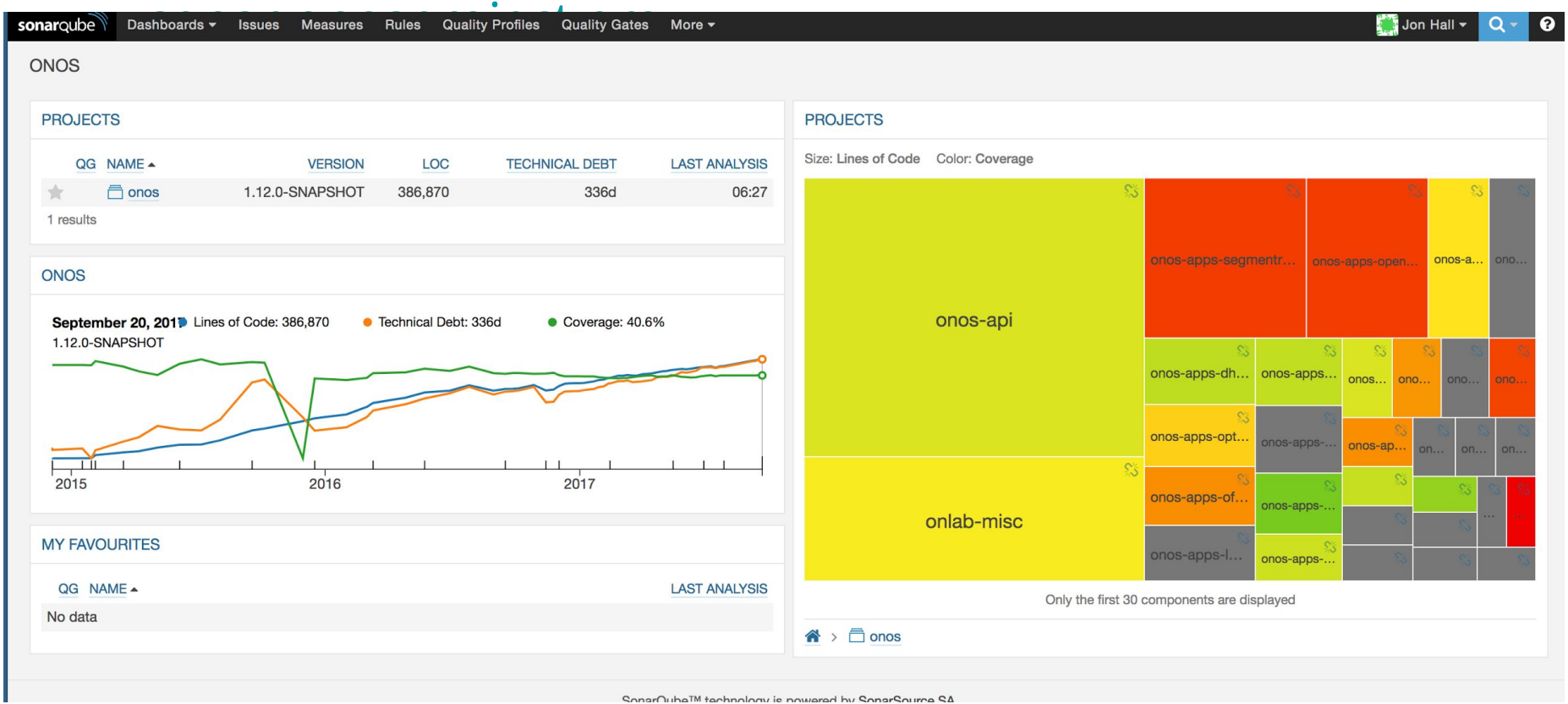
Jon Hall - JHALL@ciena.com

Unit Testing

Time to run: 1-5 Minutes

- What is it?
  - Test for a small unit of code
  - Test inputs and outputs of a function
  - Insures the code does what we said it would
  - Helps detect when changes break this contract
- What are our coverage goals?
  - New code should have at least 70% coverage but the goal for the entire project is 80+%
- How to run:
  - `buck build onos`
  - `buck test`
- Run as part of Gerrit Validation
- Resources:
  - ONOS Wiki:Unit Test Guidelines

Sonar


sonarqube
Dashboards
Issues
Measures
Rules
Quality Profiles
Quality Gates
More
Jon Hall
ONOS
PROJECTS
QG NAME
VERSION
LOC
TECHNICAL DEBT
LAST ANALYSIS
onos
1.12.0-SNAPSHOT
386,870
336d
06:27
1 results
ONOS
September 20, 2017
Lines of Code: 386,870
Technical Debt: 336d
Coverage: 40.6%
1.12.0-SNAPSHOT
2015
2016
2017
MY FAVOURITES
QG NAME
LAST ANALYSIS
No data
PROJECTS
Size: Lines of Code
Color: Coverage
onos-api
onos-apps-segmentr…
onos-apps-open…
onos-a…
ono…
onos-apps-dh…
onos-apps…
onos…
ono…
ono…
ono…
onos-apps-opt…
onos-apps-…
onos-ap…
on…
on…
on…
onos-apps-of…
onos-apps-…
onlab-misc
onos-apps-l…
onos-apps-…
Only the first 30 components are displayed
onos
SonarQube™ technology is powered by SonarSource SA

GUI Unit Testing

Time to run: ~1 Minute

- What is it?
  - Javascript Unit tests
  - Run in a headless browser
- How to run:
  - `npm install`
  - `npm test`
- Run as part of the Gerrit Validation
- Resources:
  - See onos/web/gui/src/main/webapp/tests/README.txt

STC Testing - Integration

Time to run: 30 Minutes

- What is it?
  - Scenario Test Coordinator (STC)
  - Short, simple system tests
  - Designed to quickly verify you didn't break any major functionality
  - Modular tests can be composed to create new scenarios
  - Control ONOS through Python or Bash scripts
  - Uses Mininet for the dataplane
- How to run:
  - `stc smoke` or `stc scenario`
- Can submit a scenario with bug reports to help developers reproduce the issue
- Resources:
  - ONOS wiki: Appendix G: Scenario Test Coordinator (STC)

ONOS System Tests using TestON

- What is it?
  - Thorough system tests
  - Written in python
  - Controls ONOS through Rest or CLI
- 3 types of tests:
  - Functionality
  - Longevity (CHO)
  - Performance and Scale (SCPF)
- How to run:
  - `./cli.py run FUNCintent`
- Tests are run nightly on master
  - Used to help validate new releases

ONOS System Tests using TestON

- Resources:
  - Github: opennetworkinglab/OnosSystemTest
  - ONOS Project Gerrit: OnosSystemTest
  - ONOS Wiki: System Testing Guide
  - ONOS Wiki: System Test plans and Results

Functionality Testing

Time to run: 5 to 45 Minutes each

- Focus on the ONOS “Core” and move out
- FUNC
  - Intents, Topology, Flows, Flow Objectives, Groups, etc.
- HA
  - Controller failures, Control Network failures, Dataplane failures
- Usecases
  - App specific system tests
    - SDNIP, Segment Routing, VPLS, etc...

Longevity Testing

Time to run: Days

- CHO - Continuous Hours of Operations
- Find memory leaks, or hard to reproduce defects
- Random or fixed sequence of events

Scale and Performance

Time to run: Hours

- Focused on ONOS performance
  - Mock out third party actors (topology elements) with “Null providers”
- Break down a single event from an operator’s perspective into the individual ONOS events
- Helps us focus development efforts
- Measure performance at different ONOS cluster sizes
- Example: Mastership Failover when an ONOS node dies
  - Cluster detects failed ONOS node
  - Cluster sends Mastership Event
  - New Master sends OF Role Request to device
- Latency tests:
  - Topology Events, Intent Events, Flow, Host Add, Mastership Failover
- Throughput tests:
  - Intents, Flows, Flow Objectives

Other tests

- Delta - Security Testing
  - Integrated in the past, being worked on by sec/perf brigade members
- Docker
  - Simple test that starts a cluster from the latest dockerhub images
  - Candidate for community contributions

How to get involved

- Different brigades:
  - Build and Package Infrastructure
  - GUI
  - Security and Performance Analysis
- Is your favorite app missing tests?
  - We can help you write tests
  - Even just test plans can be helpful if you can’t write code
  - Reach out to the brigade related to your app, or form a new one around it