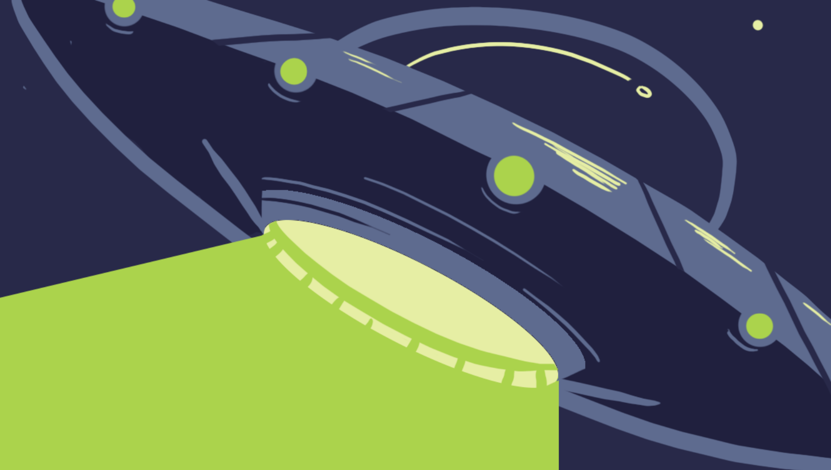
Open Liberty



Cognitive Software Delivery – Quality, Scale & Intelligence in Open Liberty

May 25th 2022, | 11:00am EST

By *Kevin Smith – STSM, CI/CD Architect, IBM Application Platform*

Introduction

Over the last 8 years we have been building up a highly scalable, intelligent CI/CD pipeline in support of Open Liberty.

- Microservice architecture built on top of a Kafka event backbone
- One of IBM's largest CI/CD pipelines
  - Up to 20,000 machines created and configured each week
  - 3+ million Kafka events per week
  - Executes over 2 years of testing daily (machine time)
  - Supports over 200 platform/JDK runtime environments including Docker and Open Shift
  - Each weekend, 4 years of testing executed to verify product on all supported platforms: 20+ million tests
- Realtime insights into product quality and infrastructure
- Over 60% of test failures automatically triaged

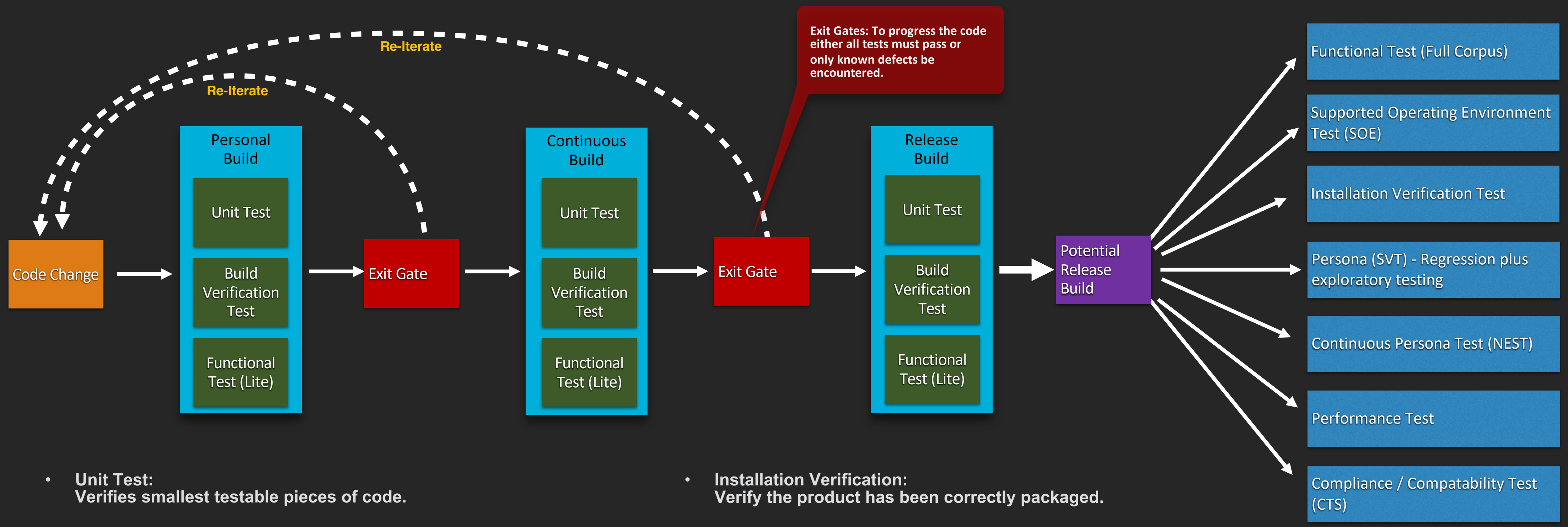
Our Journey…

Open Liberty

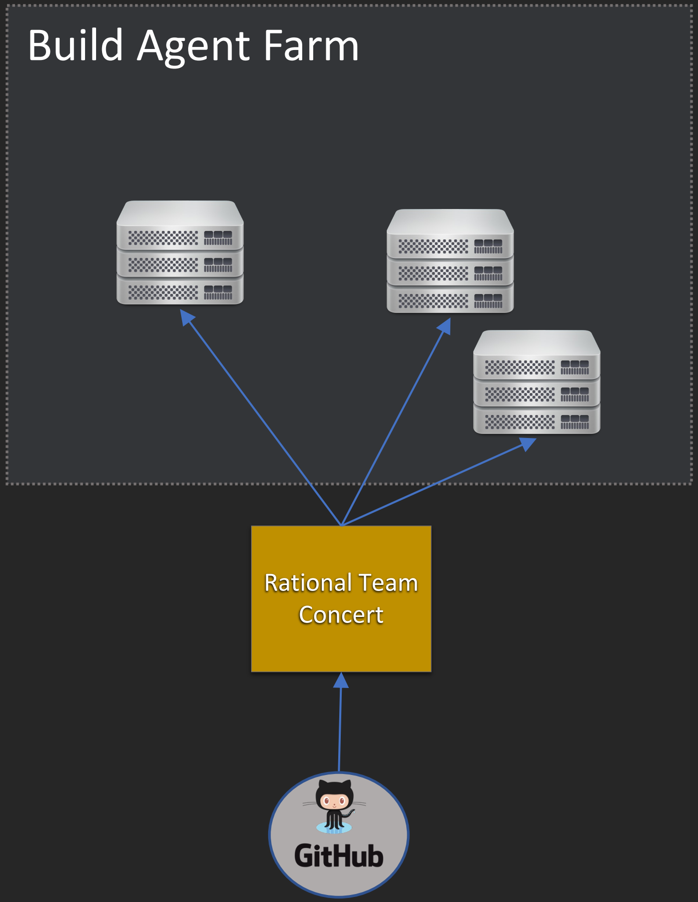
Evolution of product lifecycles

	Waterfall	Agile	Continuous Delivery
	Team would generally start to make software ready for release when all functionality has been developed.	Team should get their software ready for release throughout development at periodic intervals.	Team keeps software ready for release at all times during development.
Stabilisation Phase	Months	Weeks	**Days / Hours**
Time to known quality	N/A	Days	**Hours**
Tolerance of regressions	High	Low	**Zero**
Test run frequency	Weekly	Daily	**Every Change**

Software Delivery Pipeline

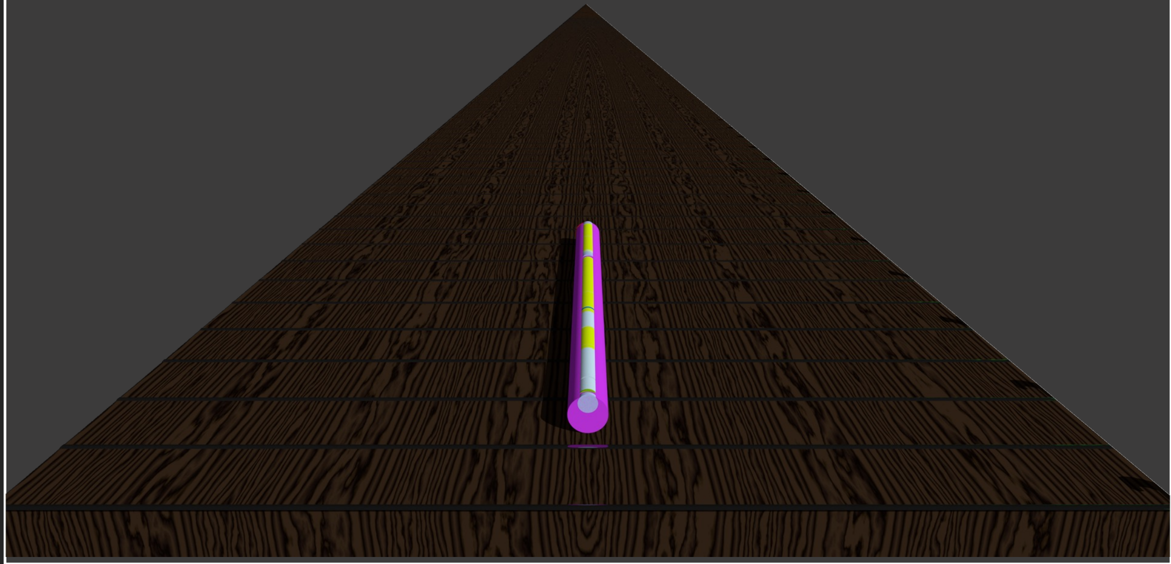

Re-Iterate
Re-Iterate
Exit Gates: To progress the code either all tests must pass or only known defects be encountered.
Code Change
Personal Build
Unit Test
Build Verification Test
Functional Test (Lite)
Exit Gate
Continuous Build
Unit Test
Build Verification Test
Functional Test (Lite)
Exit Gate
Release Build
Unit Test
Build Verification Test
Functional Test (Lite)
Potential Release Build
Functional Test (Full Corpus)
Supported Operating Environment Test (SOE)
Installation Verification Test
Persona (SVT) - Regression plus exploratory testing
Continuous Persona Test (NEST)
Performance Test
Compliance / Compatability Test (CTS)

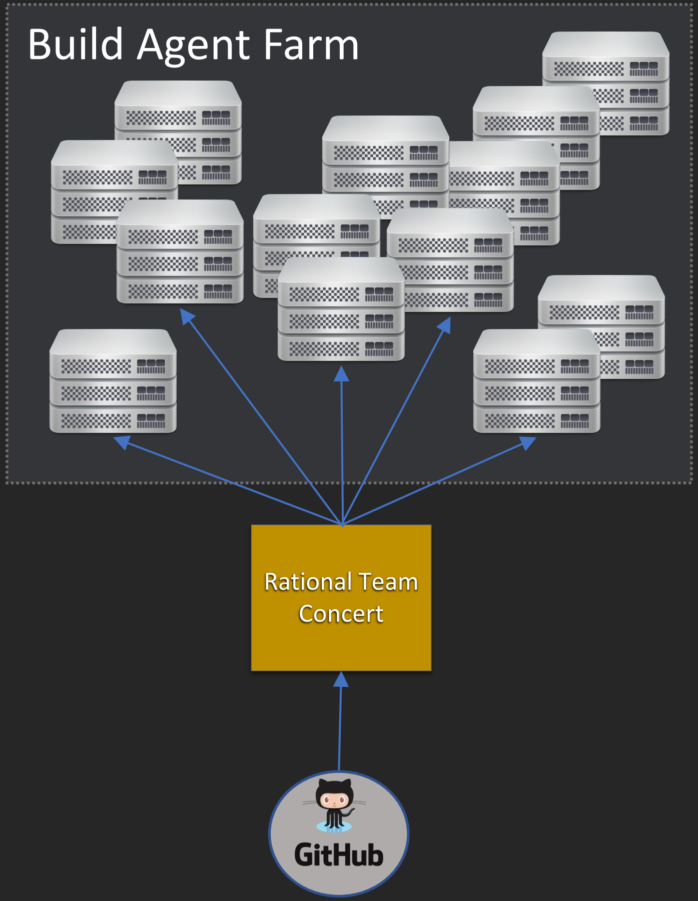
- **Unit Test:**
  Verifies smallest testable pieces of code.
- **Build Verification Test:**
  Small set of tests used to initially validate the build.
- **Functional Test (Lite):**
  Verifies correct behaviour of functions. The "Lite" version is around 70% of the entire corpus, focusing on golden path testing and common error paths.
- **Functional Test (Full Corpus):**
  Entire corpus of functional tests. A superset of the "Lite" functional tests adding in uncommon code paths, long running tests and rare error paths.
- **Supported Operating Environment Test**
  Runs the Entire corpus of functional tests against a large matrix of OS/JDK combinations supported by the product.
- **Installation Verification:**
  Verify the product has been correctly packaged.
- **Persona (SVT):**
  Large scale testing including customer like scenarios, load, scalability, recoverability tests. Majority are run manually to include some exploratory testing.
- **Continuous Persona Test (NEST):**
  24/7 test environment running customer like scenarios constantly, updated using DevOps principles.
- **Performance Test**
  Designed to identify performance issues and bottlenecks as well as verify the throughput of the product.
- **Compliance/Compatibility Test (CTS):**
  A set of tests and tools to confirm the product performs to an industry standard specification (J2EE).



- Software delivery lifecycles often begin when a product or set of deliverables are small and easily verified
- Initially, a CI Tool (e.g. Jenkins or Rational Team Concert) will connect to a set of agents to run builds or tests
- With each release more function is added, and the challenge to verify this function starts to increase
- Testing and triage can grow exponentially compared to the product size

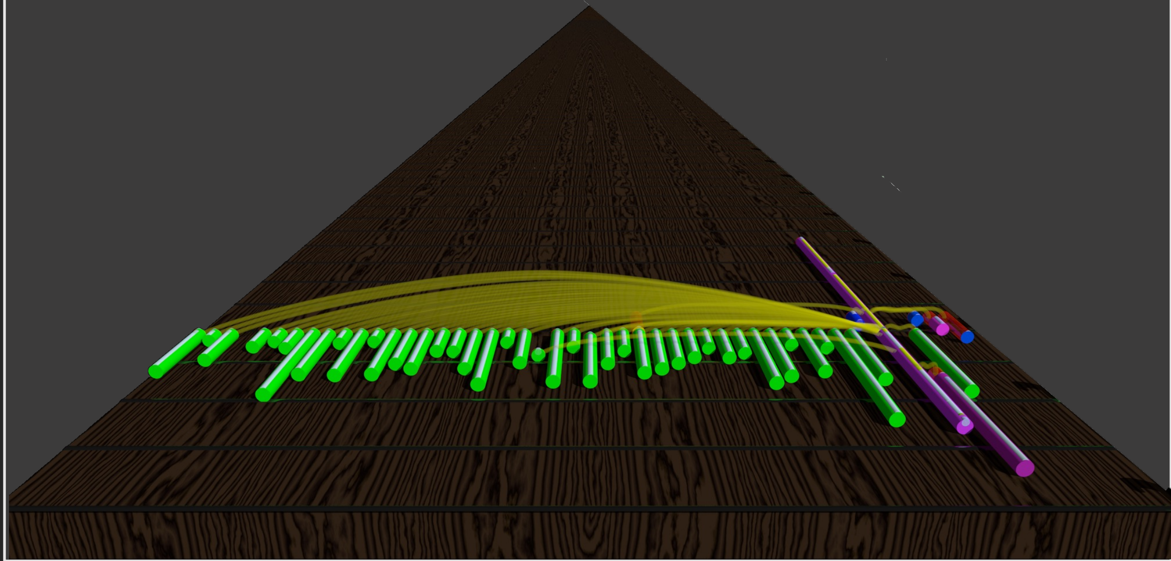
Small applications can use a single end to end build to compile and verify the product as total execution time is small.

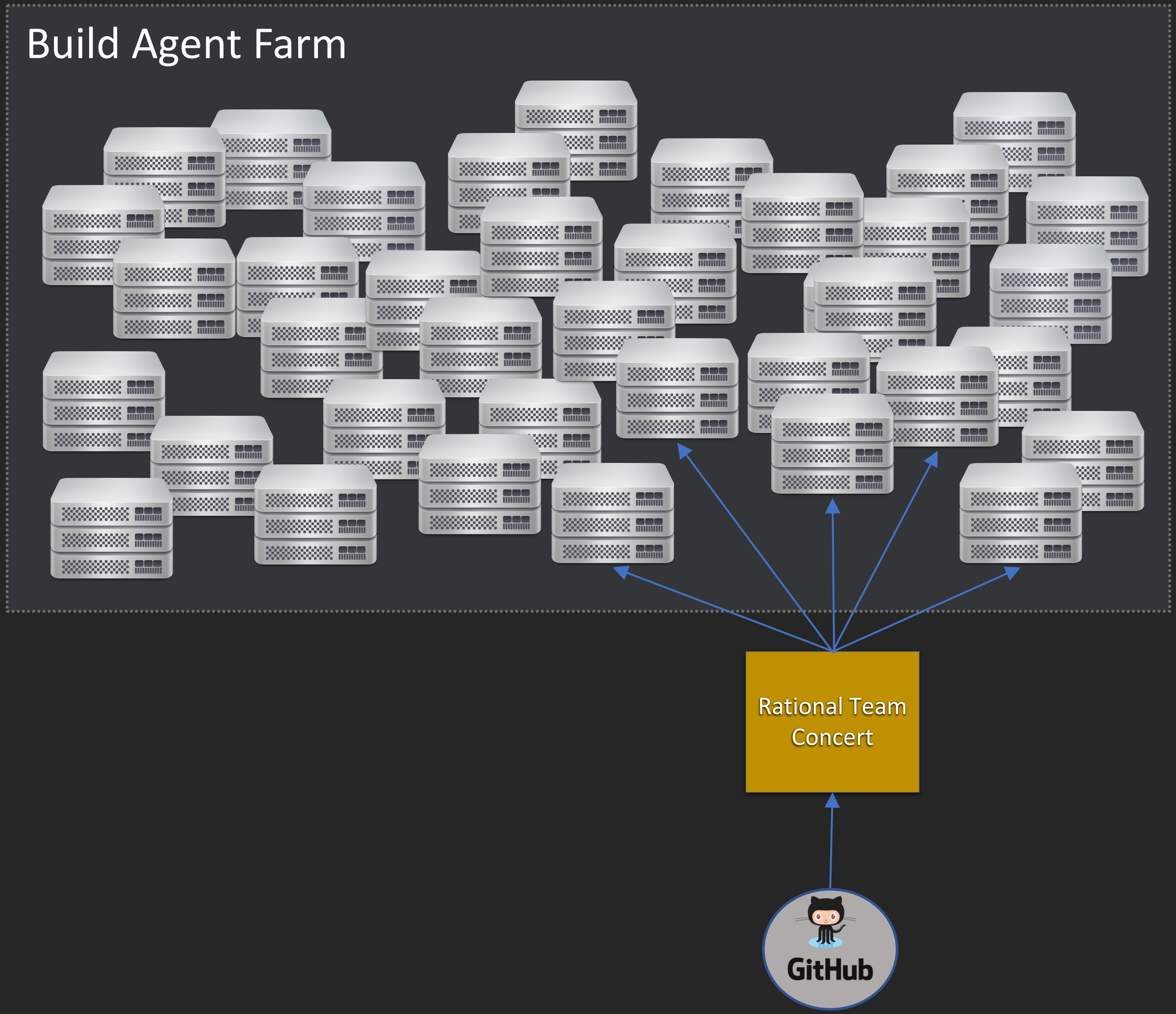




- Software delivery lifecycles often begin when a product or set of deliverables are small and easily verified
- Initially, a CI Tool (e.g. Jenkins or Rational Team Concert) will connect to a set of agents to run builds or tests
- With each release more function is added, and the challenge to verify this function starts to increase
- Testing and triage can grow exponentially compared to the product size

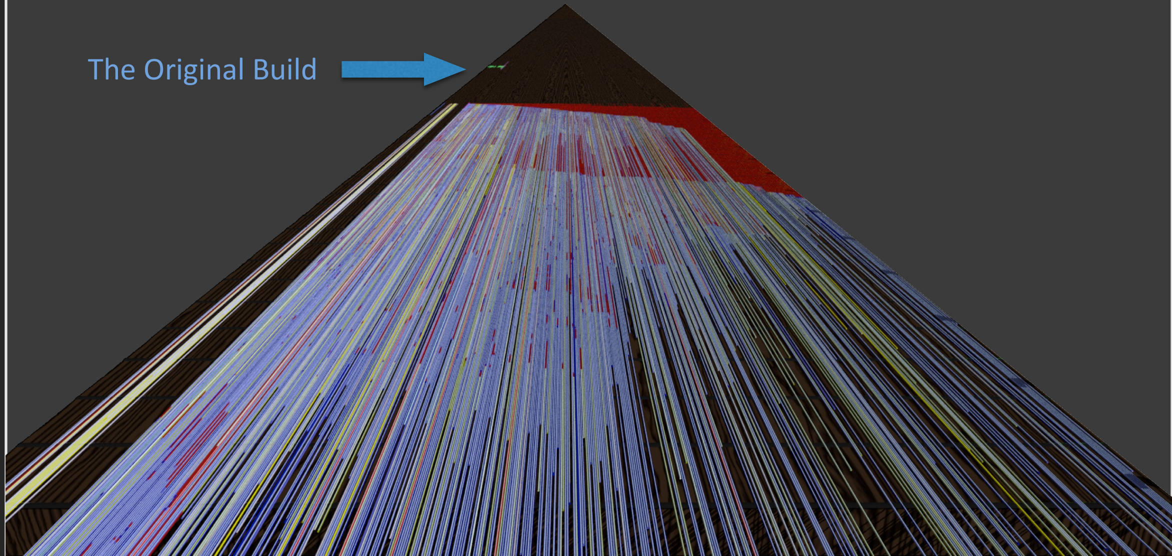
As function grew, so did testing. Eventually testing needed to be done in parallel. Currently we use 30 parallel builds to run 80+ hours of testing on a release build.

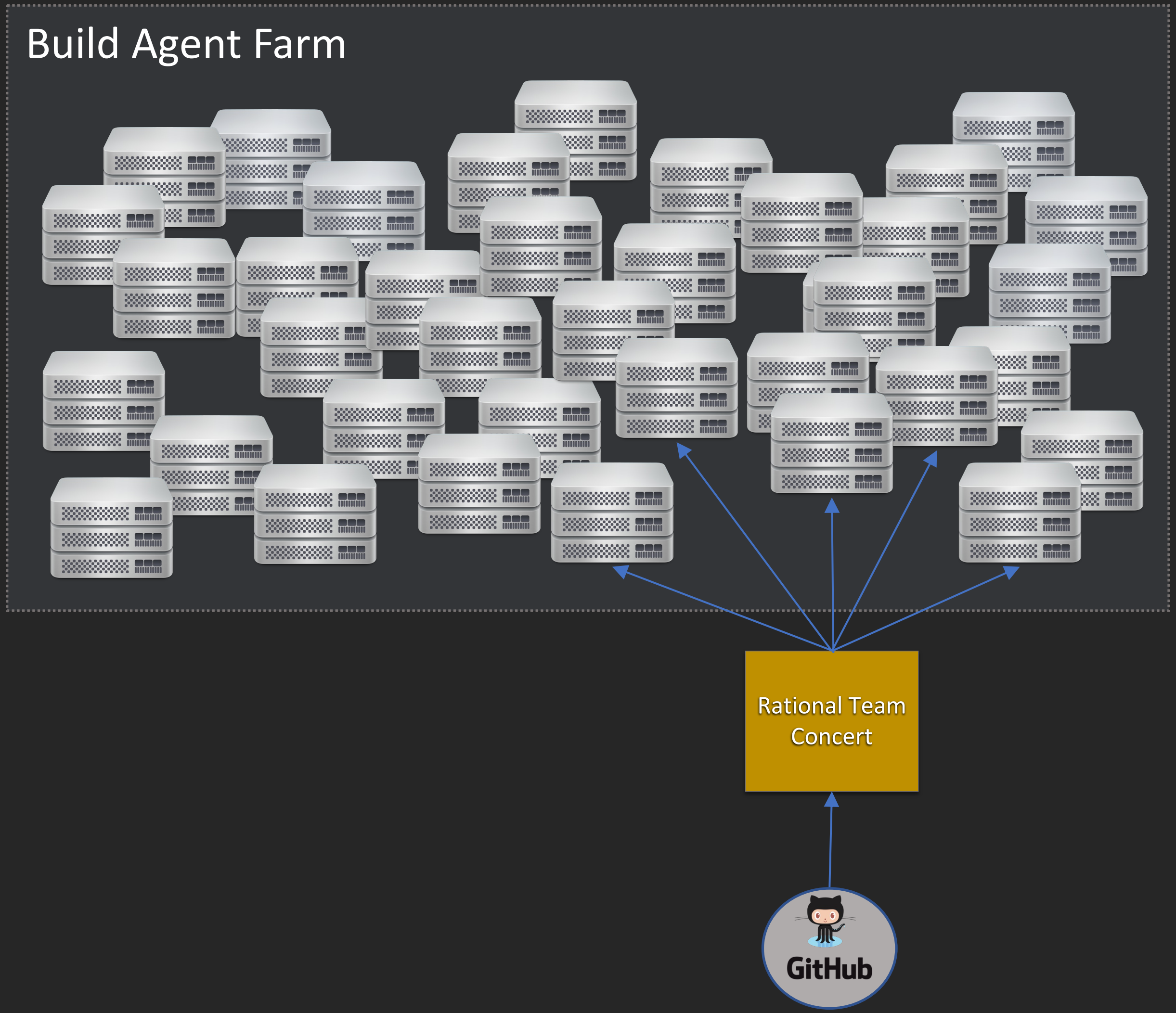




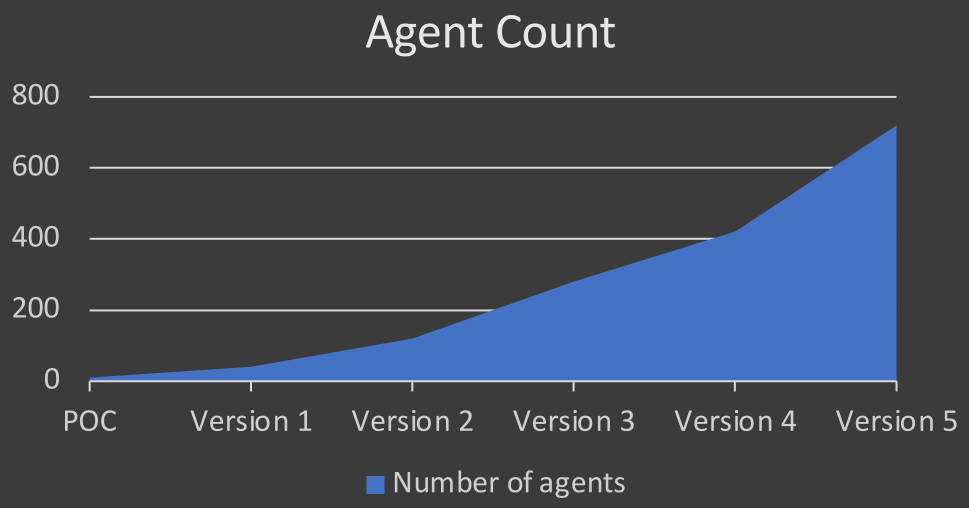
- Software delivery lifecycles often begin when a product or set of deliverables are small and easily verified
- Initially, a CI Tool (e.g. Jenkins or Rational Team Concert) will connect to a set of agents to run builds or tests
- With each release more function is added, and the challenge to verify this function starts to increase
- Testing and triage can grow exponentially compared to the product size

Once you add in cross platform testing, things scale up rapidly. We run cross platform testing each weekend and currently it requires over 4 years of machine time to complete.



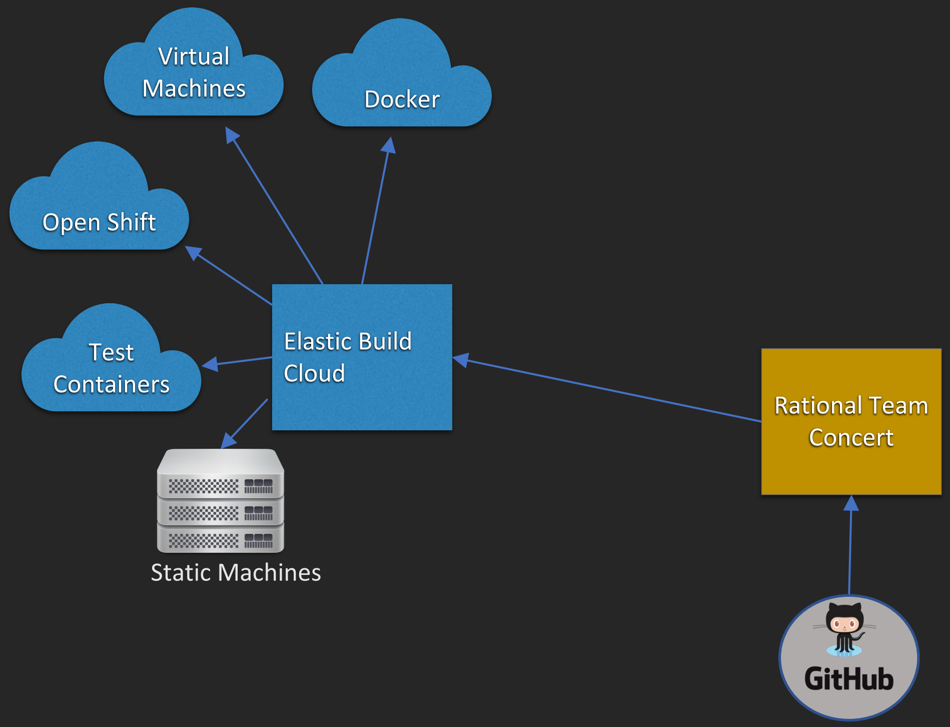


- Software delivery lifecycles often begin when a product or set of deliverables are small and easily verified
- Initially, a CI Tool (e.g. Jenkins or Rational Team Concert) will connect to a set of agents to run builds or tests
- With each release more function is added, and the challenge to verify this function starts to increase
- Testing and triage can grow exponentially compared to the product size
- Managing and maintaining agents quickly becomes costly and unreliable
- A better approach was needed…

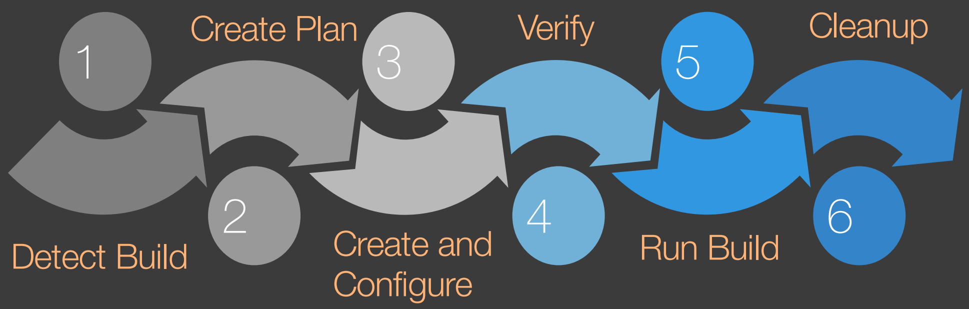


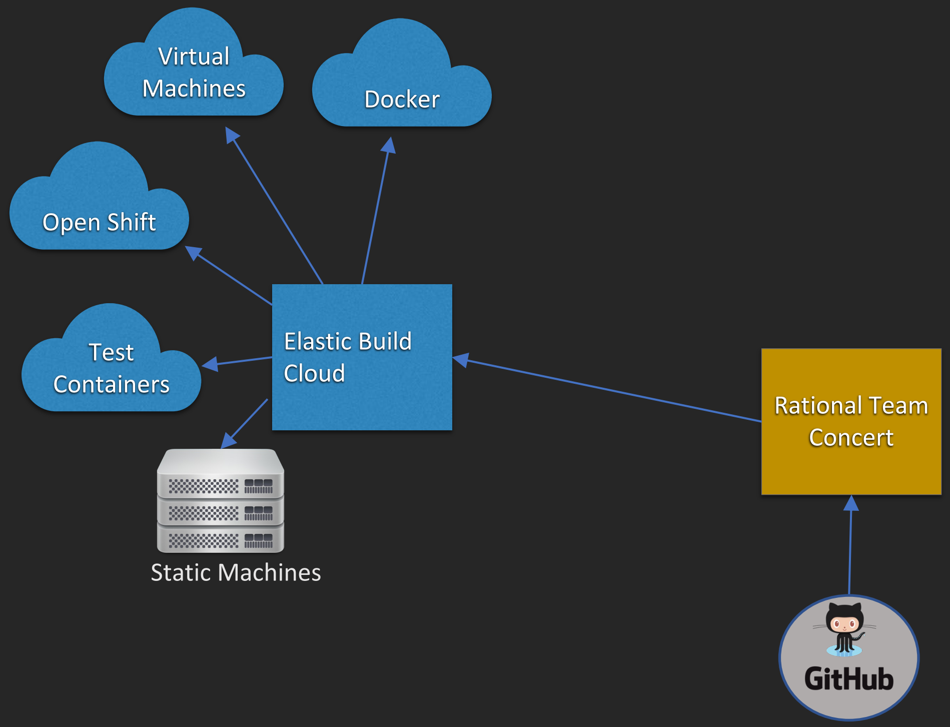
Pipeline Key Requirements

- Infrastructure as code
- High throughput and performance
- Flexible and scalable
- Continuous monitoring
- Self healing and resilient
- Rapid feedback on failures
- Compliance
- Low cost – Use only what you need
- Quality assurance
- Fully automated

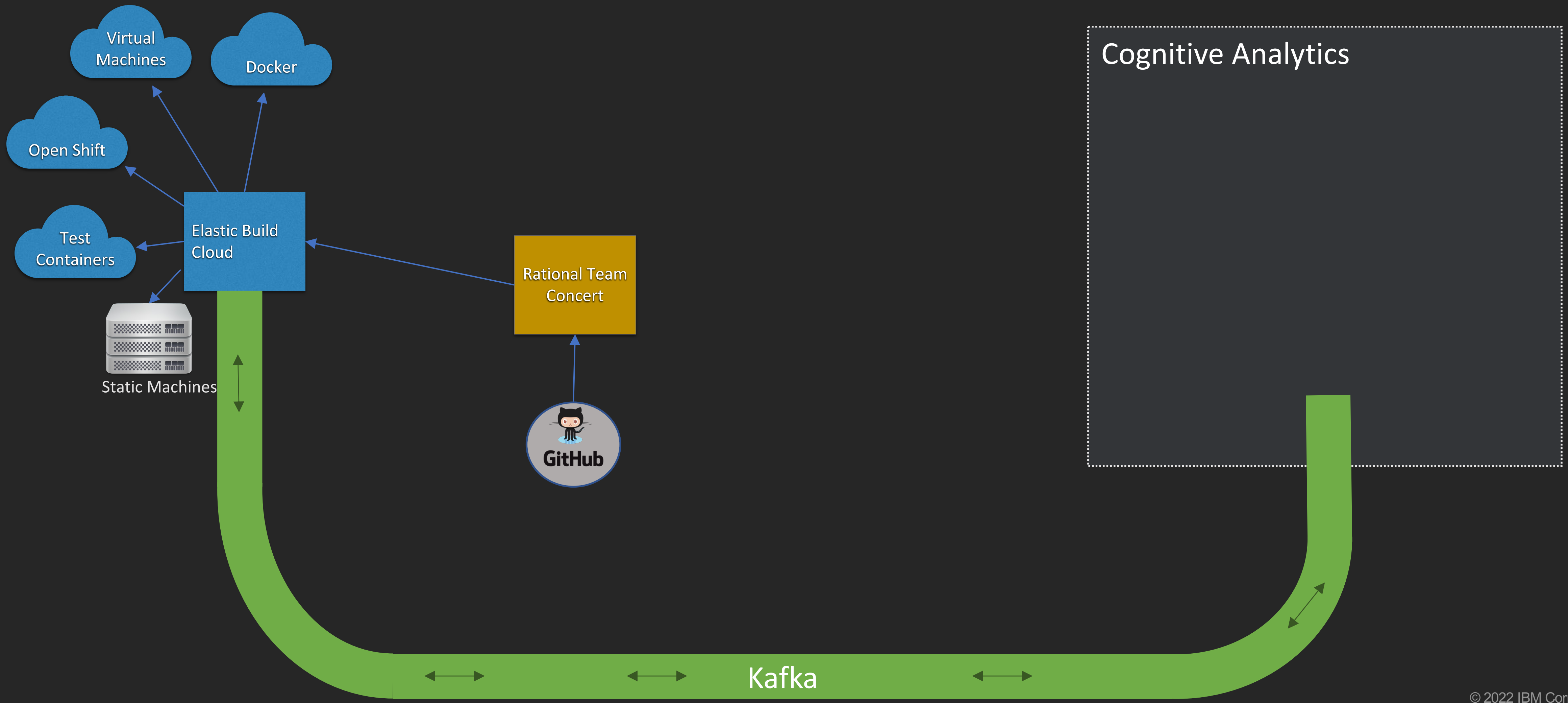


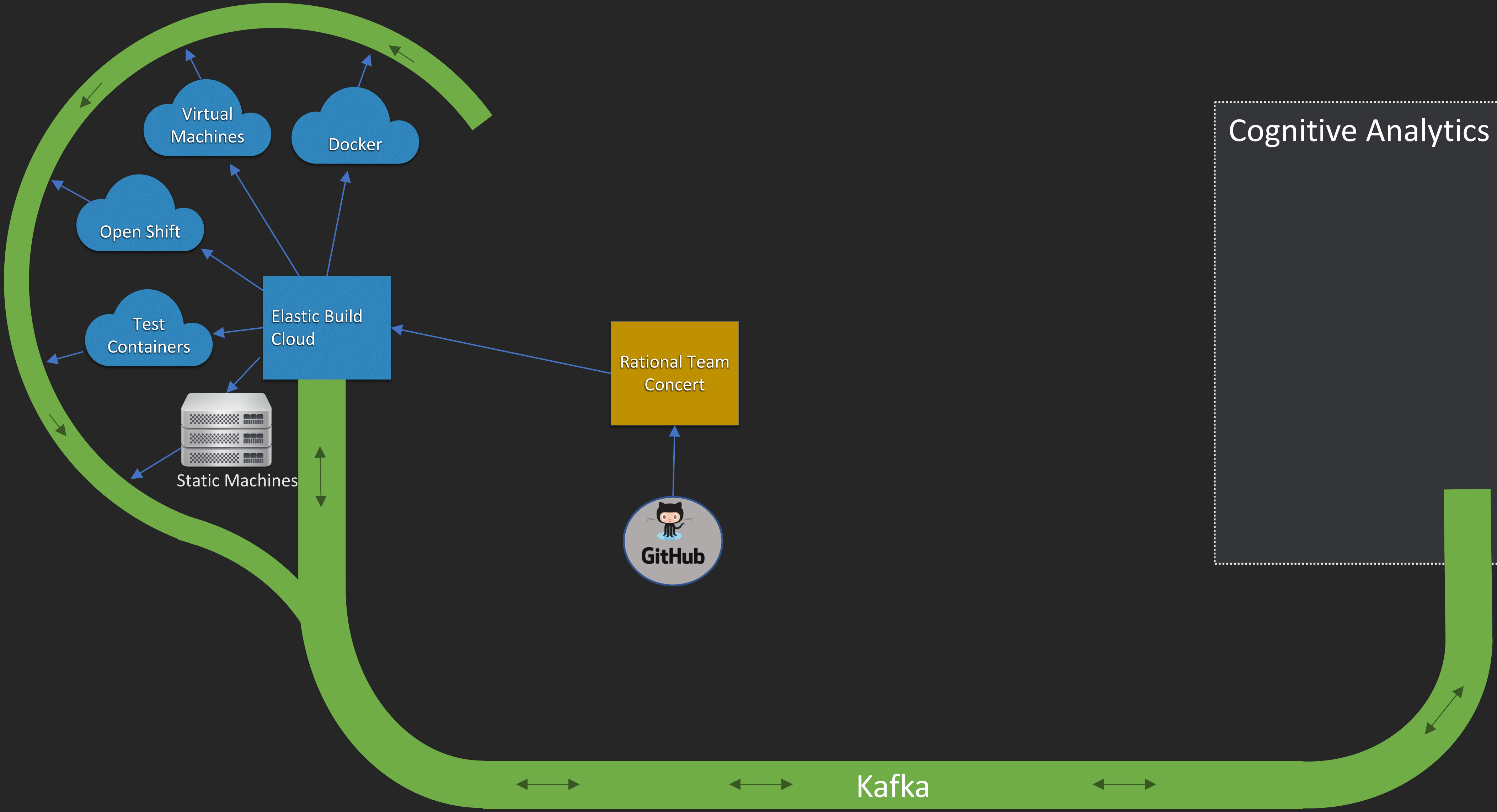
- Switch to infrastructure as code
- Our Elastic Build Cloud was created to provision and configure systems as and when needed, utilizing Ansible
- Over time, expanded to support many different target environments
- Infrastructure lives only as long as the work requested
- Every build or test run gets a freshly created machine, custom configured for its needs
- Extreme scaling – can create up to 20,000 custom configured systems each week!



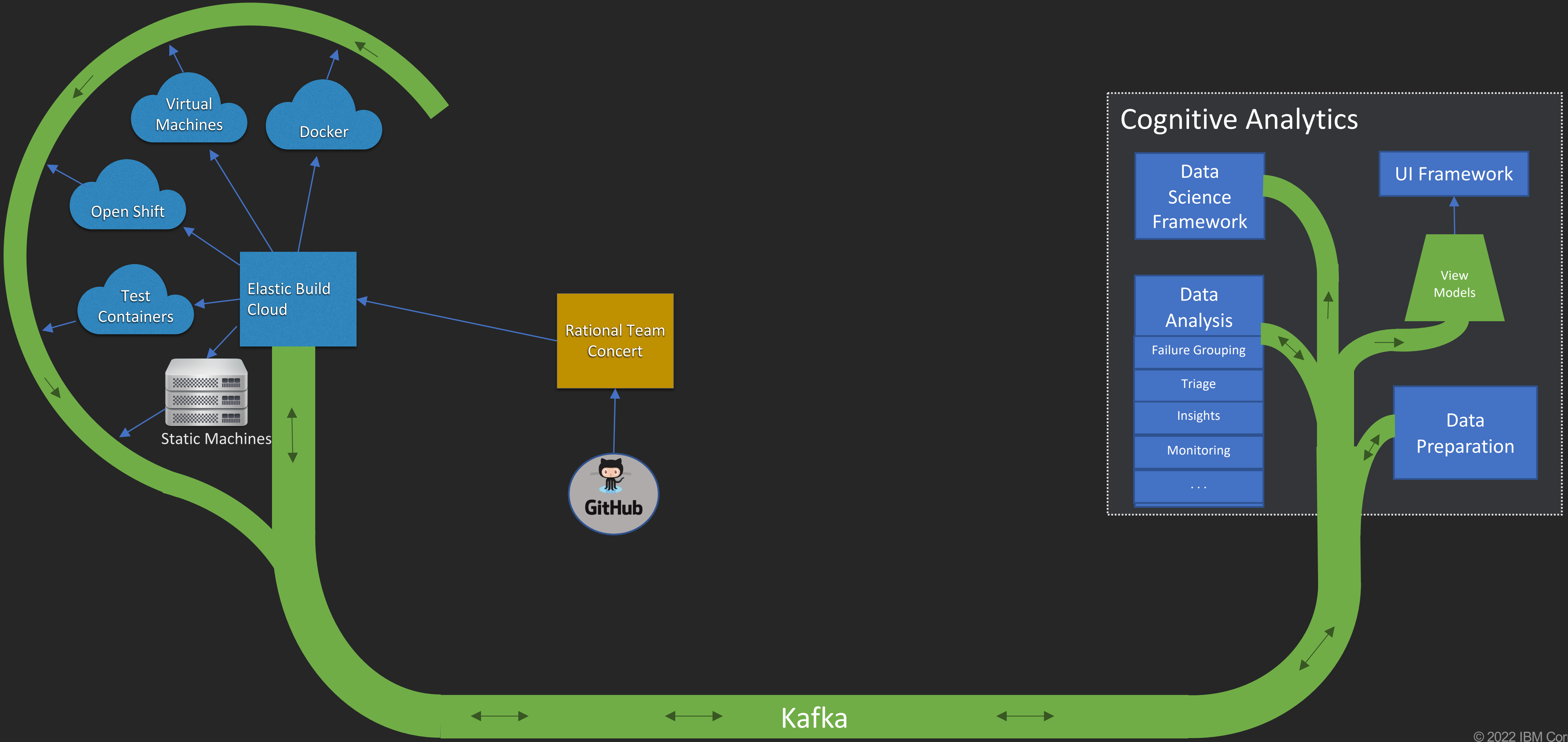


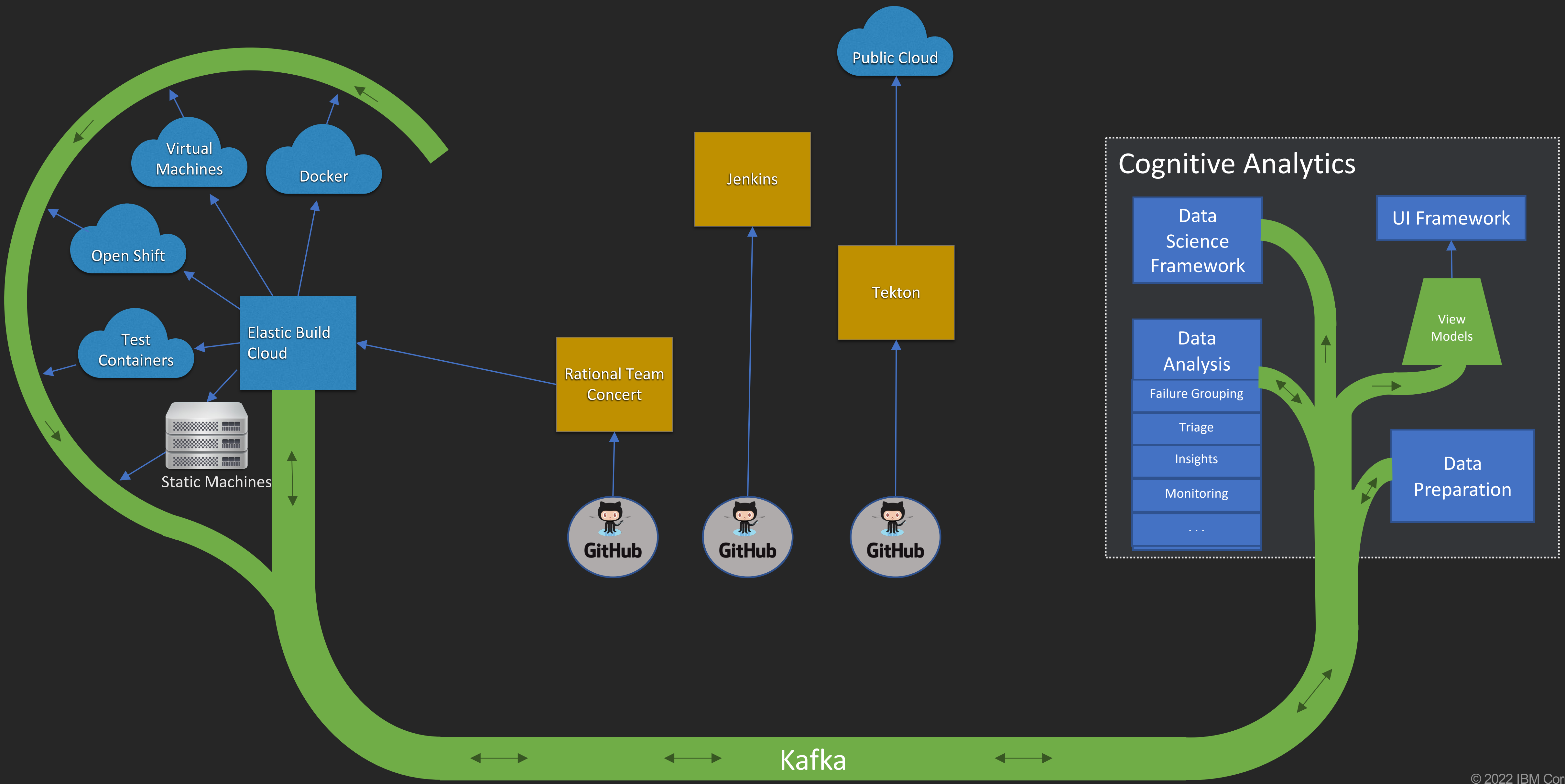
- Having a powerful environment to run our build and test workload is only part of the solution
- How do we manage it and how do we triage any issues found by the system?
    - Costs were increasing exponentially
    - Failures were getting skipped in favor of newer issues.
    - Data overload
- Needed to move to a model where test results were considered transient
    - Defects should be system of record for test failures, not a test result database
    - A test failure should result in an action
    - All test failures needed to be actioned or information would be lost
- That is where the Cognitive Analytics system comes in…

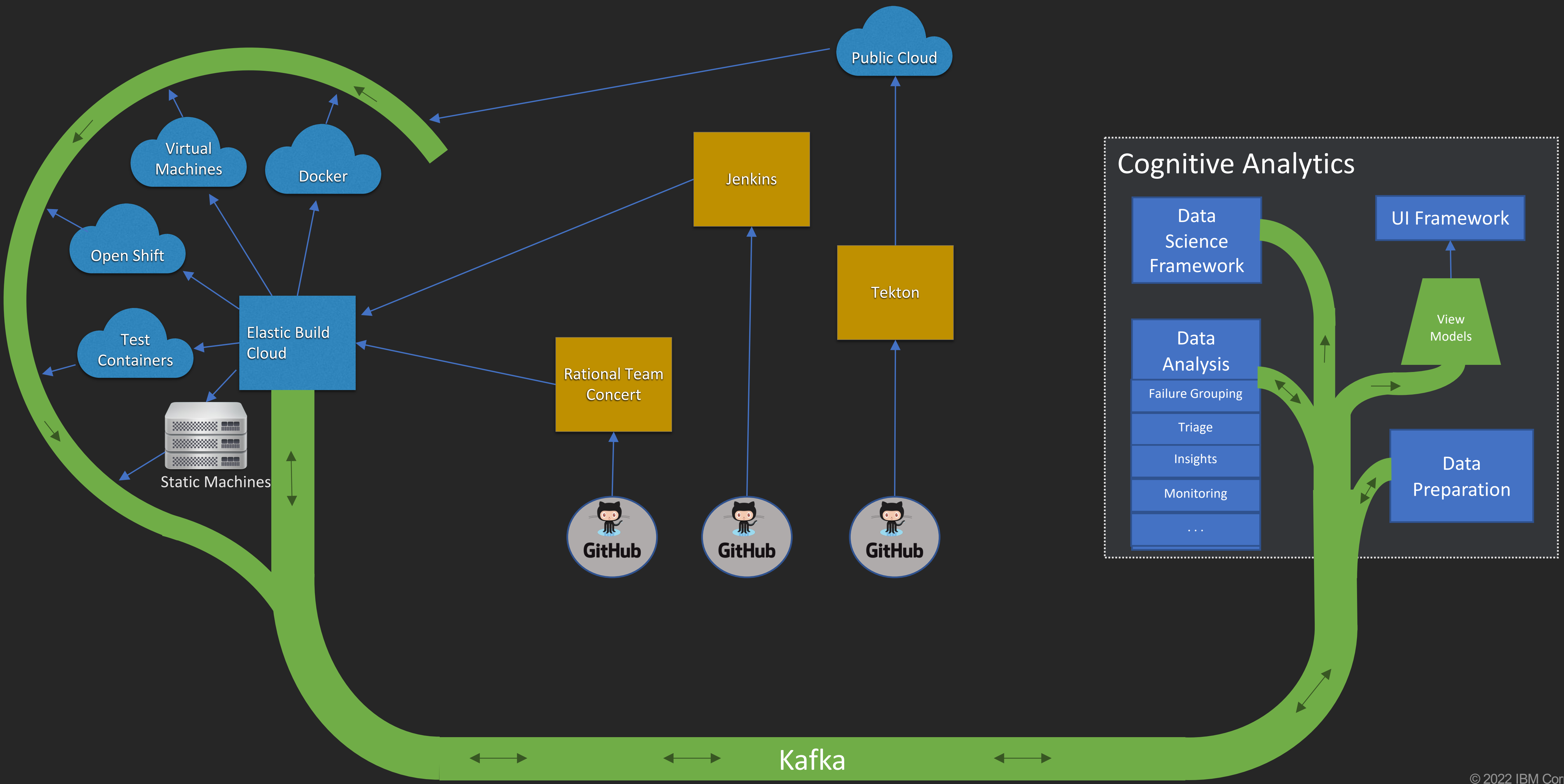

Virtual Machines
Docker
Open Shift
Test Containers
Elastic Build Cloud
Static Machines
Rational Team Concert
GitHub
Cognitive Analytics
Kafka

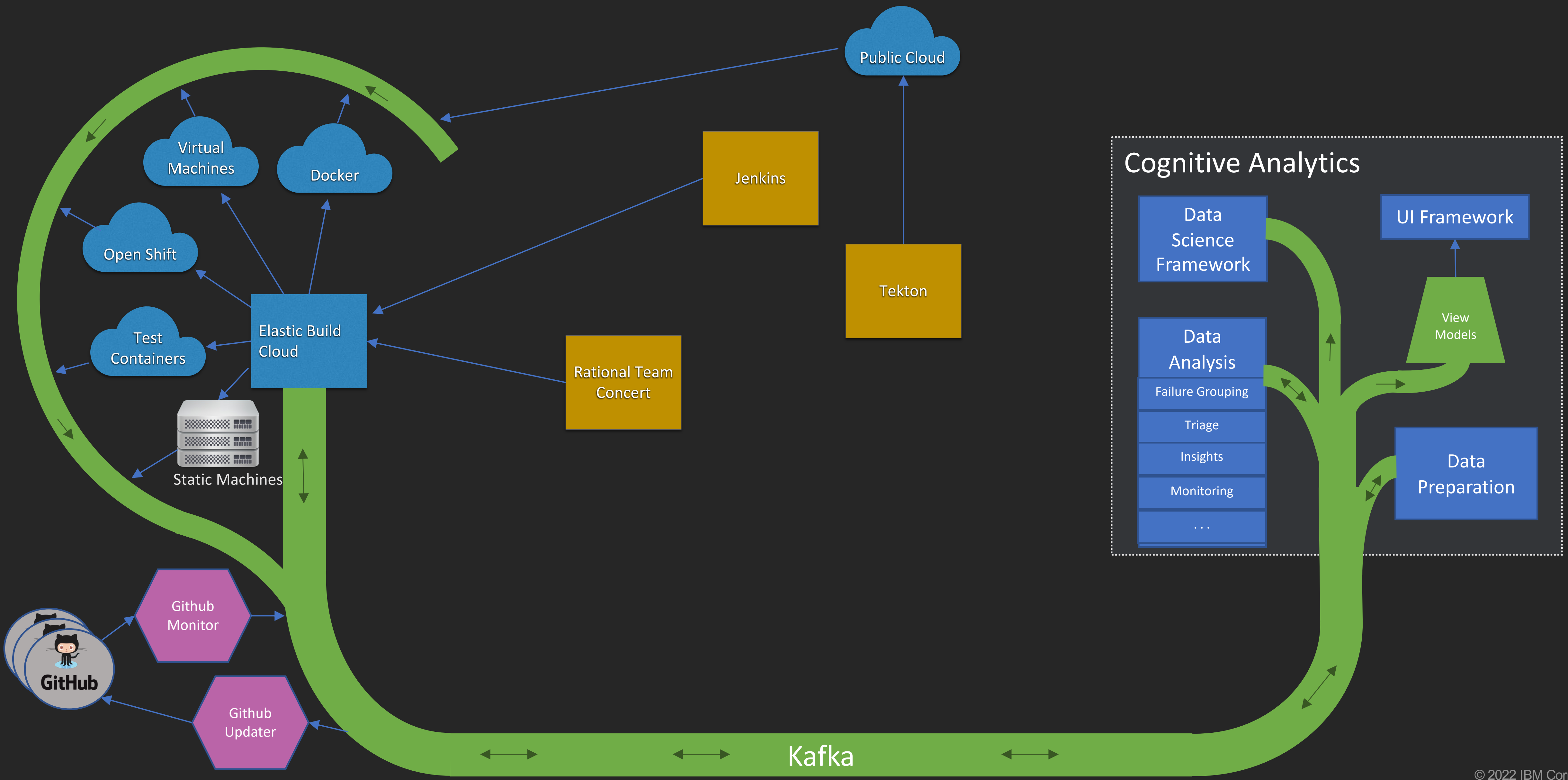


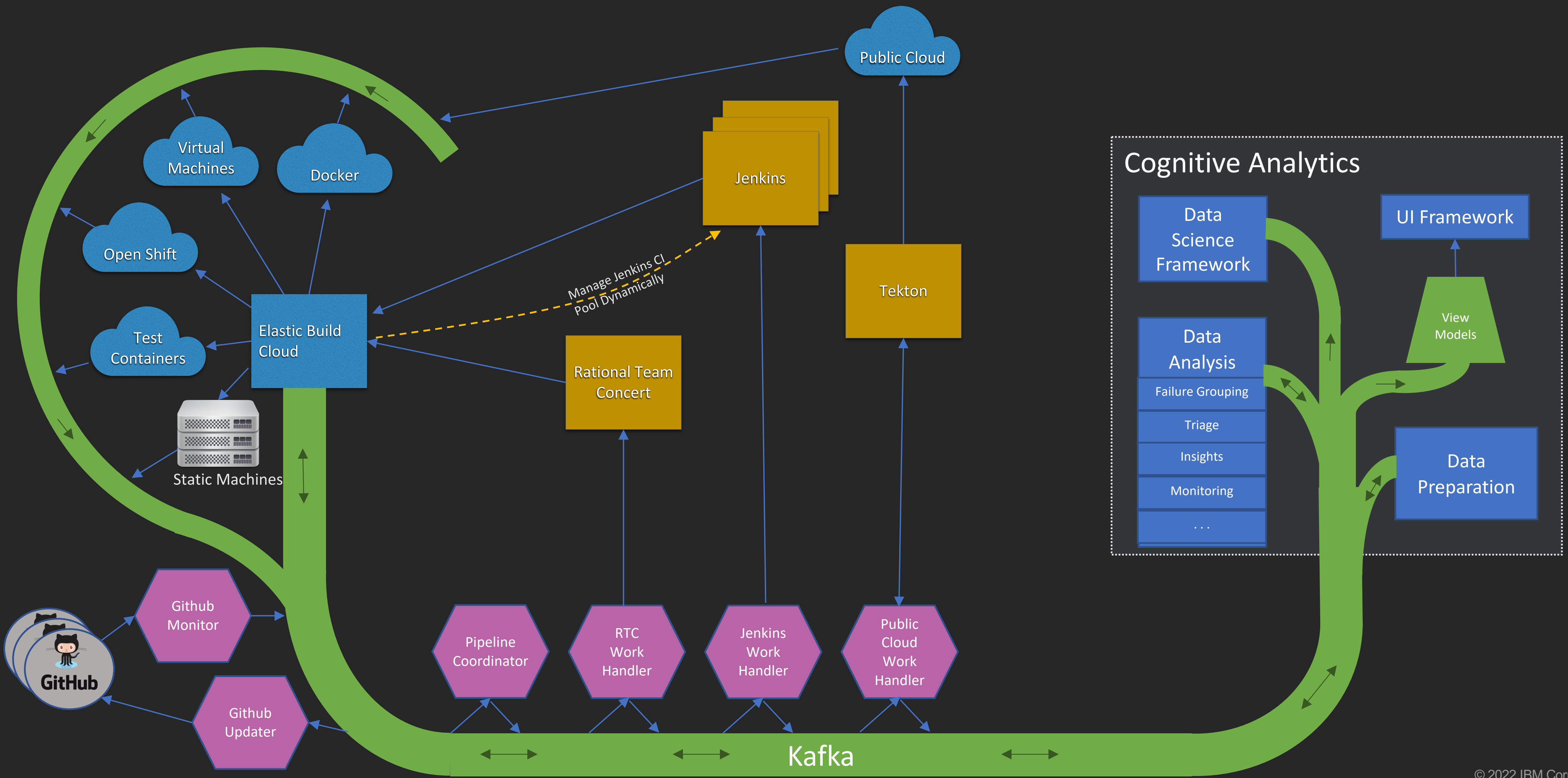
Virtual Machines
Docker
Open Shift
Elastic Build Cloud
Test Containers
Static Machines
Rational Team Concert
GitHub
Cognitive Analytics
Kafka


Virtual Machines
Docker
Open Shift
Test Containers
Elastic Build Cloud
Static Machines
Rational Team Concert
GitHub
Cognitive Analytics
Data Science Framework
UI Framework
View Models
Data Analysis
Failure Grouping
Triage
Insights
Monitoring
. . .
Data Preparation
Kafka


Public Cloud
Virtual Machines
Docker
Jenkins
Open Shift
Tekton
Test Containers
Elastic Build Cloud
Rational Team Concert
Static Machines
GitHub
GitHub
GitHub
Cognitive Analytics
Data Science Framework
UI Framework
Data Analysis
View Models
Failure Grouping
Triage
Insights
Monitoring
. . .
Data Preparation
Kafka


Public Cloud
Virtual Machines
Docker
Jenkins
Tekton
Open Shift
Elastic Build Cloud
Test Containers
Rational Team Concert
Static Machines
GitHub
GitHub
GitHub
Cognitive Analytics
Data Science Framework
UI Framework
View Models
Data Analysis
Failure Grouping
Triage
Insights
Monitoring
. . .
Data Preparation
Kafka


Public Cloud
Virtual Machines
Docker
Jenkins
Open Shift
Tekton
Test Containers
Elastic Build Cloud
Rational Team Concert
Static Machines
Cognitive Analytics
Data Science Framework
UI Framework
Data Analysis
View Models
Failure Grouping
Triage
Insights
Monitoring
. . .
Data Preparation
Github Monitor
GitHub
Github Updater
Kafka

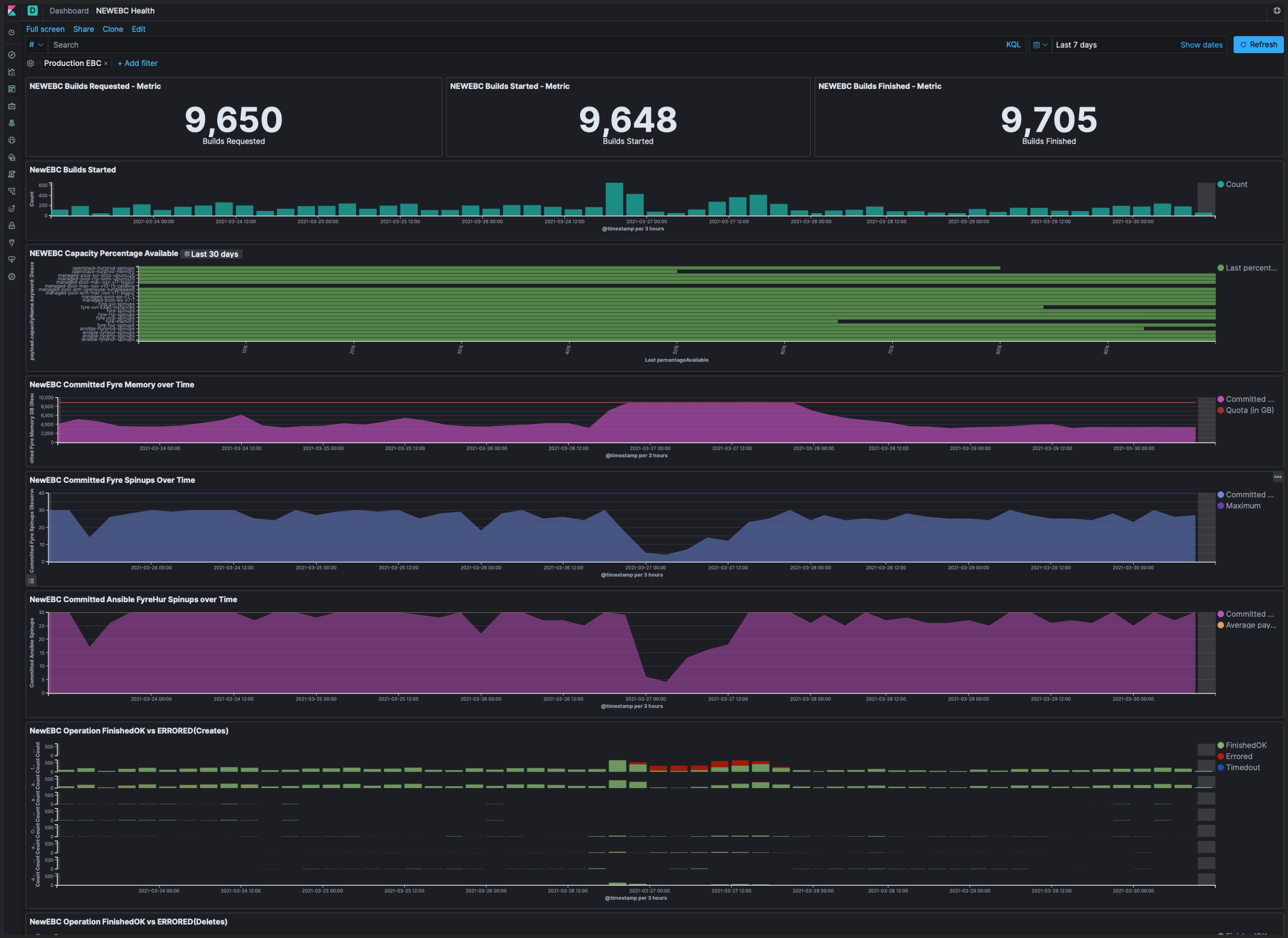


Public Cloud
Virtual Machines
Docker
Jenkins
Open Shift
Manage Jenkins CI Pool Dynamically
Tekton
Elastic Build Cloud
Test Containers
Rational Team Concert
Static Machines
Cognitive Analytics
Data Science Framework
UI Framework
View Models
Data Analysis
Failure Grouping
Triage
Insights
Monitoring
. . .
Data Preparation
Github Monitor
GitHub
Github Updater
Pipeline Coordinator
RTC Work Handler
Jenkins Work Handler
Public Cloud Work Handler
Kafka

So what does all this technology give us?

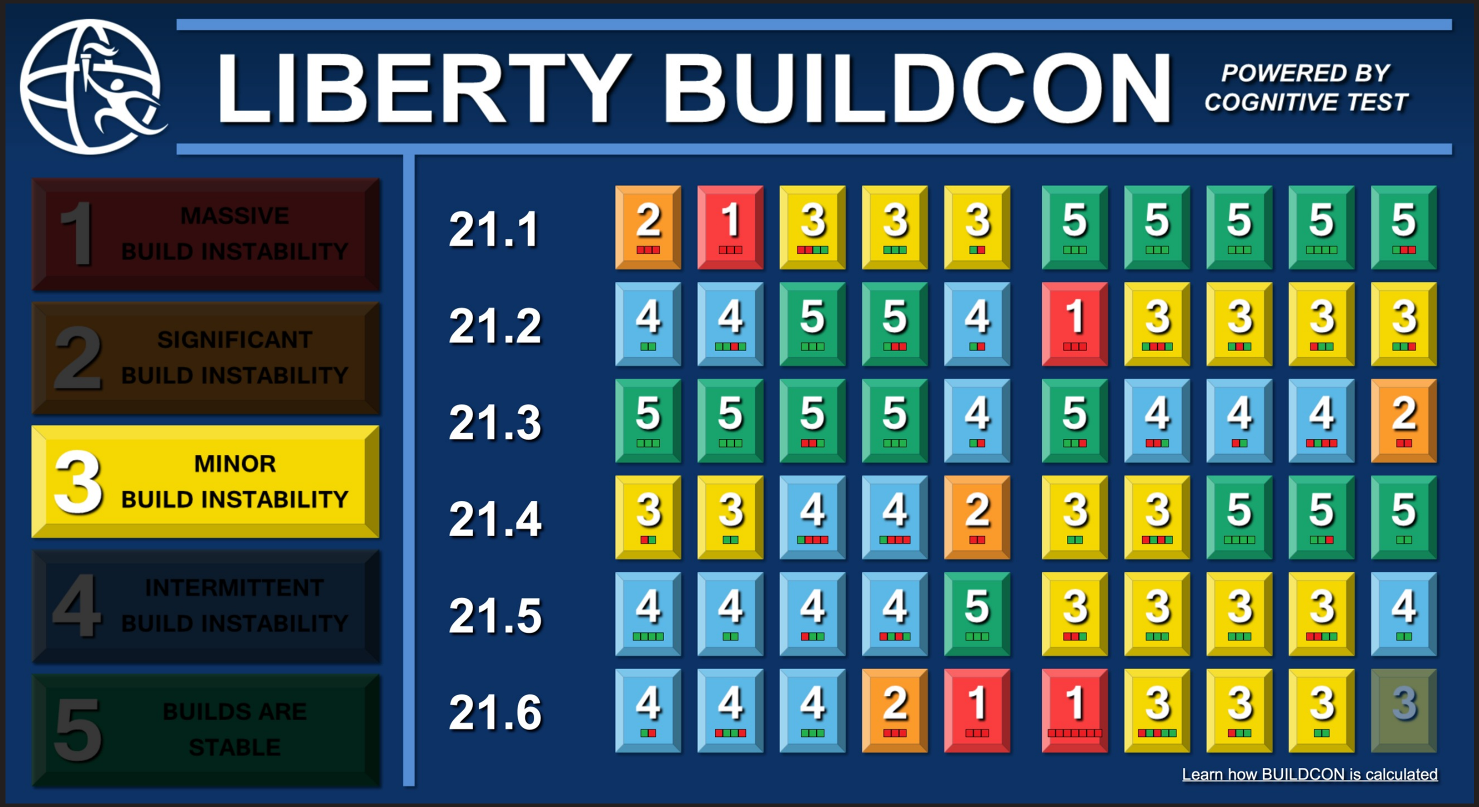
Open Liberty

Real Time Monitoring


Dashboard NEWEBC Health
Full screen Share Clone Edit
Search
KQL
Last 7 days
Show dates
Refresh
Production EBC ×
+ Add filter
NEWEBC Builds Requested - Metric
9,650
Builds Requested
NEWEBC Builds Started - Metric
9,648
Builds Started
NEWEBC Builds Finished - Metric
9,705
Builds Finished
NewEBC Builds Started
Count
@timestamp per 3 hours
NEWEBC Capacity Percentage Available
Last 30 days
Last percent...
Last percentageAvailable
NewEBC Committed Fyre Memory over Time
Committed ...
Quota (in GB)
NewEBC Committed Fyre Spinups Over Time
Committed ...
Maximum
NewEBC Committed Ansible FyreHur Spinups over Time
Committed ...
Average pay...
NewEBC Operation FinishedOK vs ERRORED(Creates)
FinishedOK
Errored
Timedout
NewEBC Operation FinishedOK vs ERRORED(Deletes)

Managing Technical Debt

- Over time our build and test system rots and intermittent issues build, this impacts our ability to create green release drivers (100% tests passing)
- BUILDCON introduced to address this:
  - BUILDCON 5: Green release driver likely most of the time
  - BUILDCON 4: Green release driver likely more than once a day on average
  - BUILDCON 3: Green release driver once per day
  - BUILDCON 2: All squads **SHOULD** be working on technical debt as currently can't guarantee green release driver each day
  - BUILDCON 1: All squads **MUST** be working on technical debt. Delivery of code automatically blocked for anything other than defect/technical debt work



LIBERTY BUILDCON

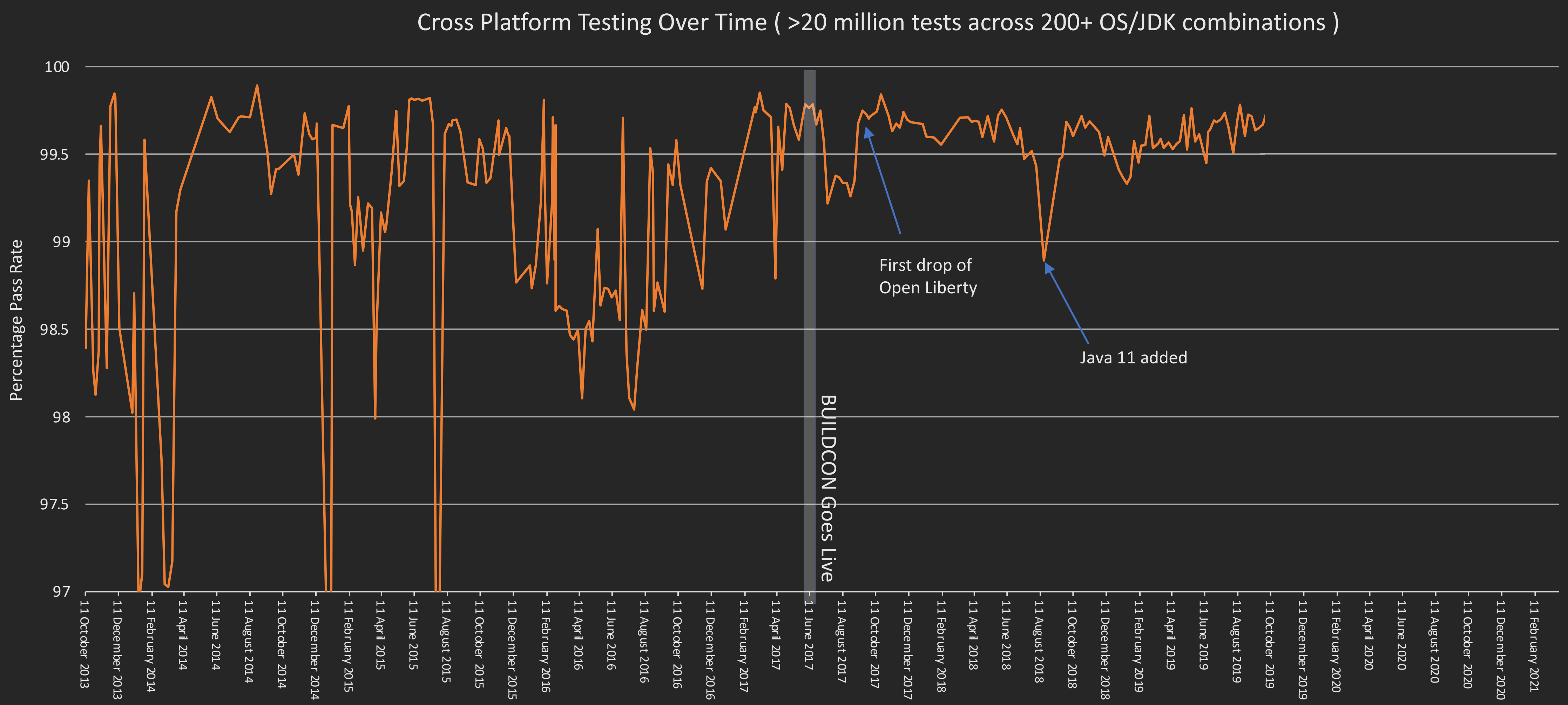
POWERED BY COGNITIVE TEST

1	MASSIVE BUILD INSTABILITY
2	SIGNIFICANT BUILD INSTABILITY
3	MINOR BUILD INSTABILITY
4	INTERMITTENT BUILD INSTABILITY
5	BUILDS ARE STABLE

21.1	2	1	3	3	3	5	5	5	5	5
21.2	4	4	5	5	4	1	3	3	3	3
21.3	5	5	5	5	4	5	4	4	4	2
21.4	3	3	4	4	2	3	3	5	5	5
21.5	4	4	4	4	5	3	3	3	3	4
21.6	4	4	4	2	1	1	3	3	3	3

Learn how BUILDCON is calculated

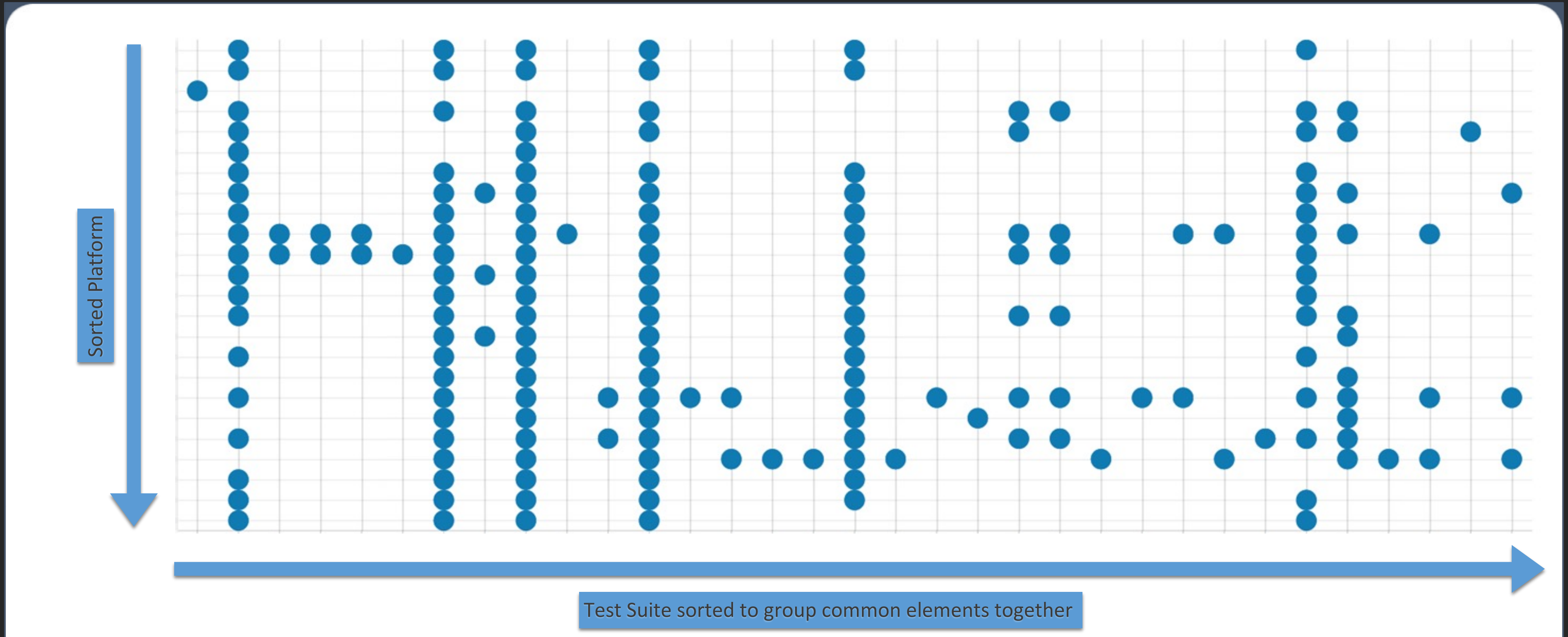
Highlighting the impact



And most importantly…
Powerful triage and insights.

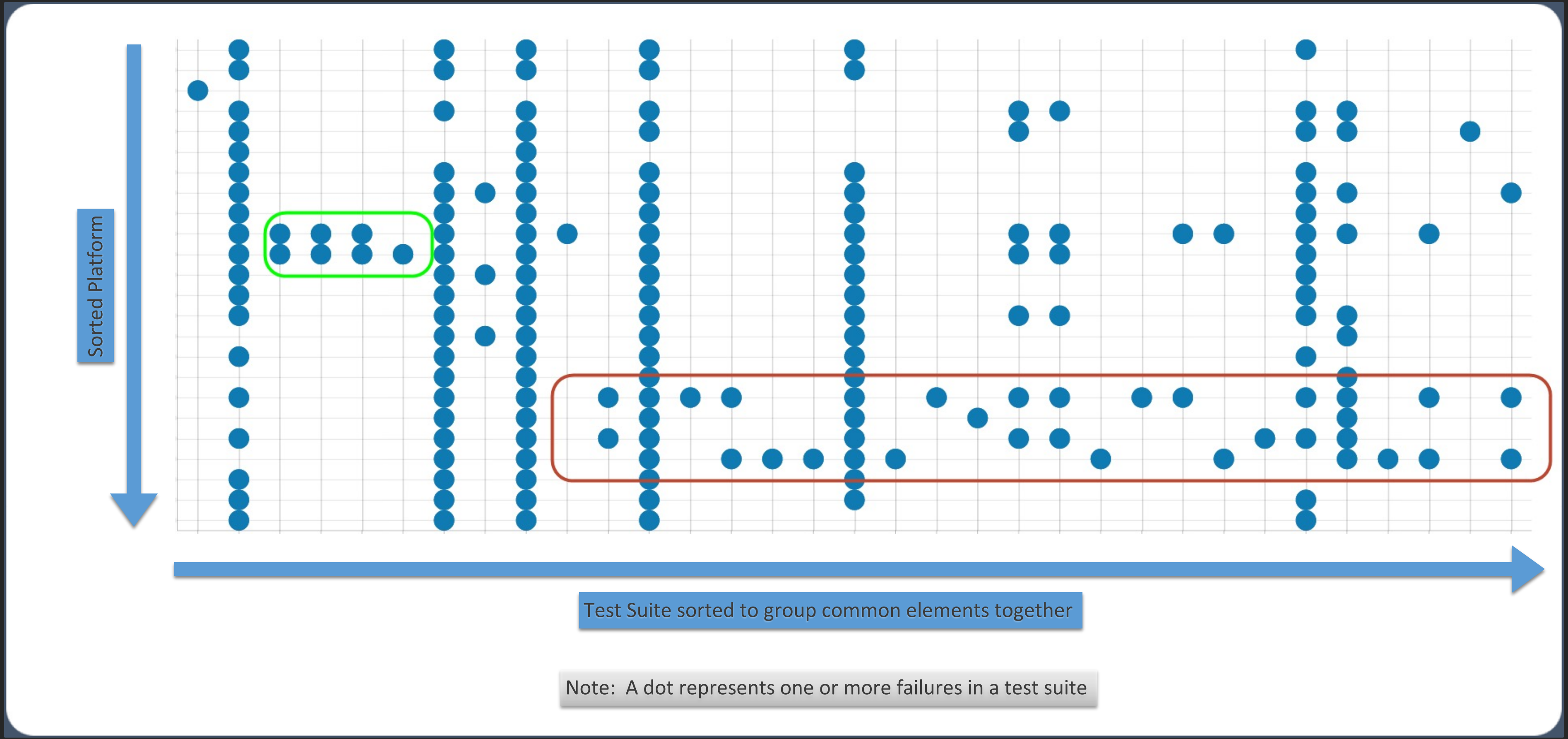
Open Liberty

Introducing the Dot Plot



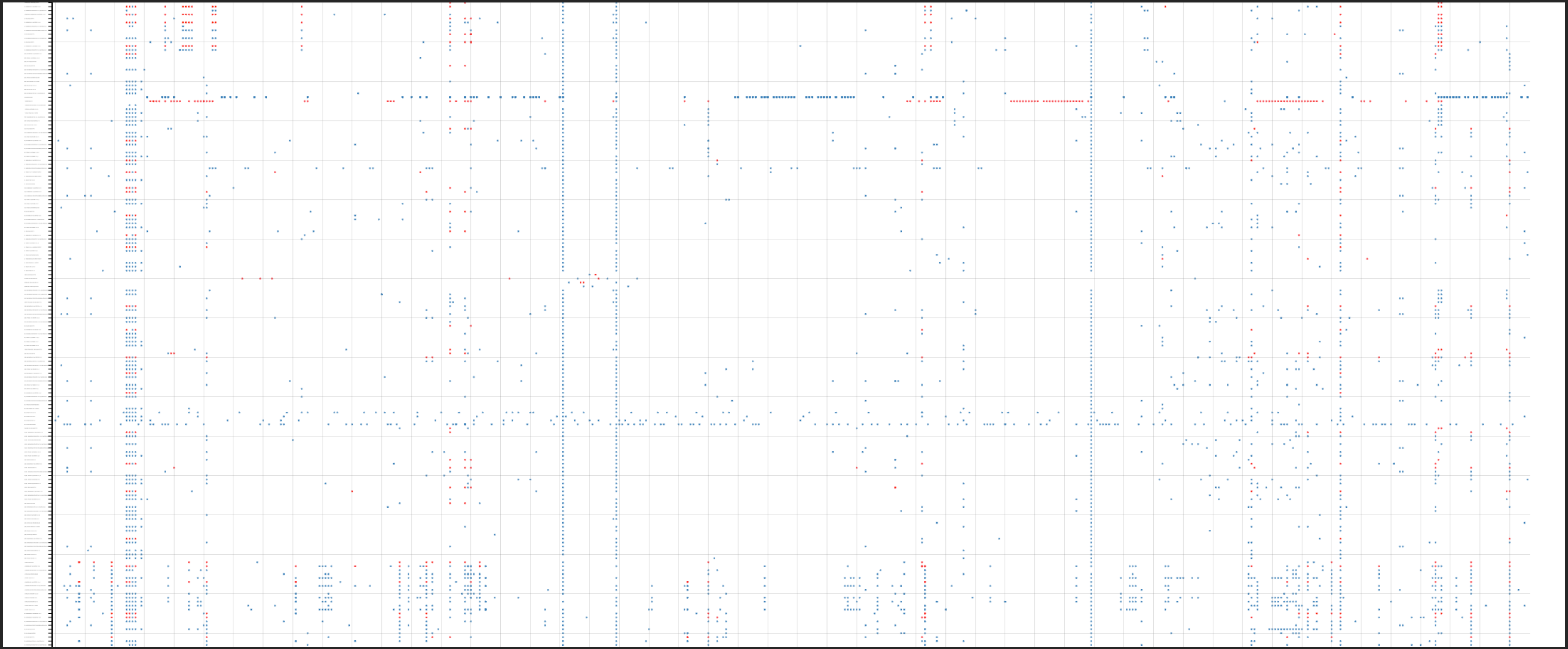
Note: A dot represents one or more failures in a test suite

Introducing the Dot Plot

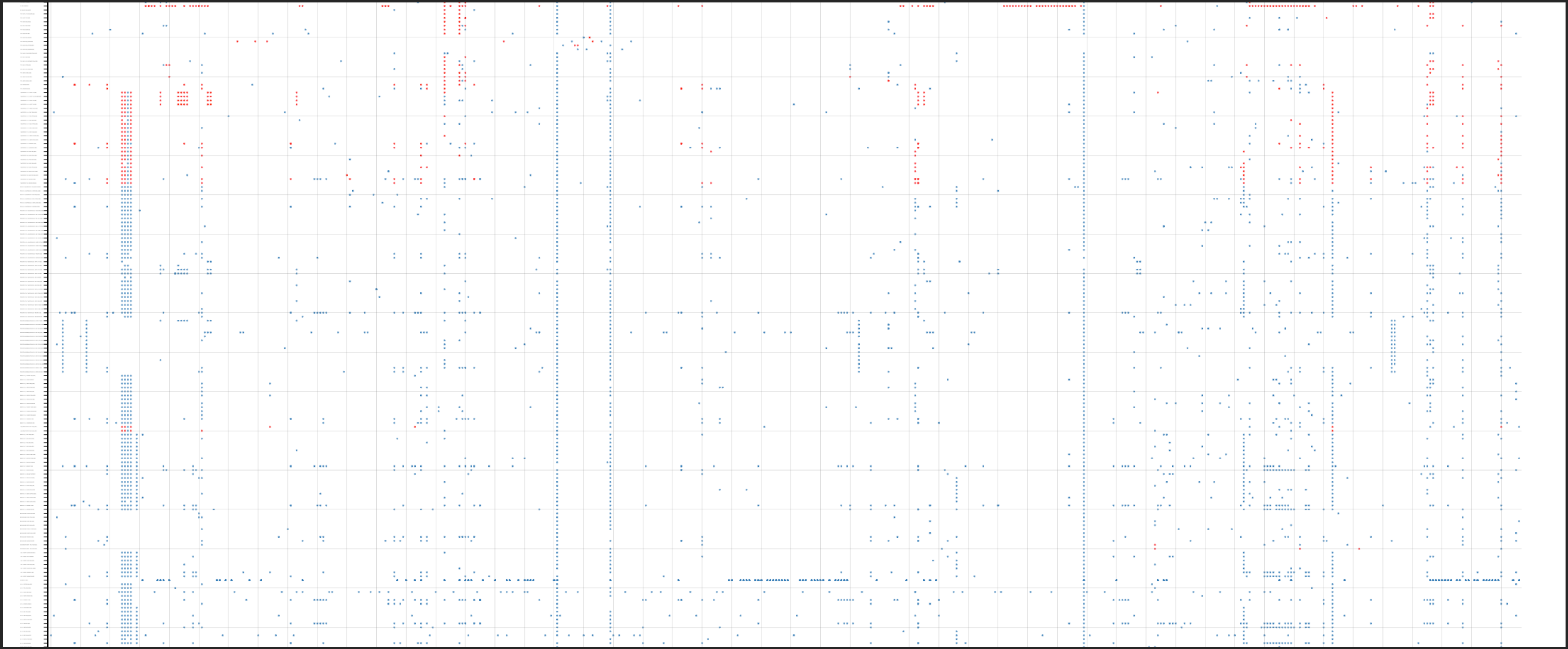


Note: A dot represents one or more failures in a test suite

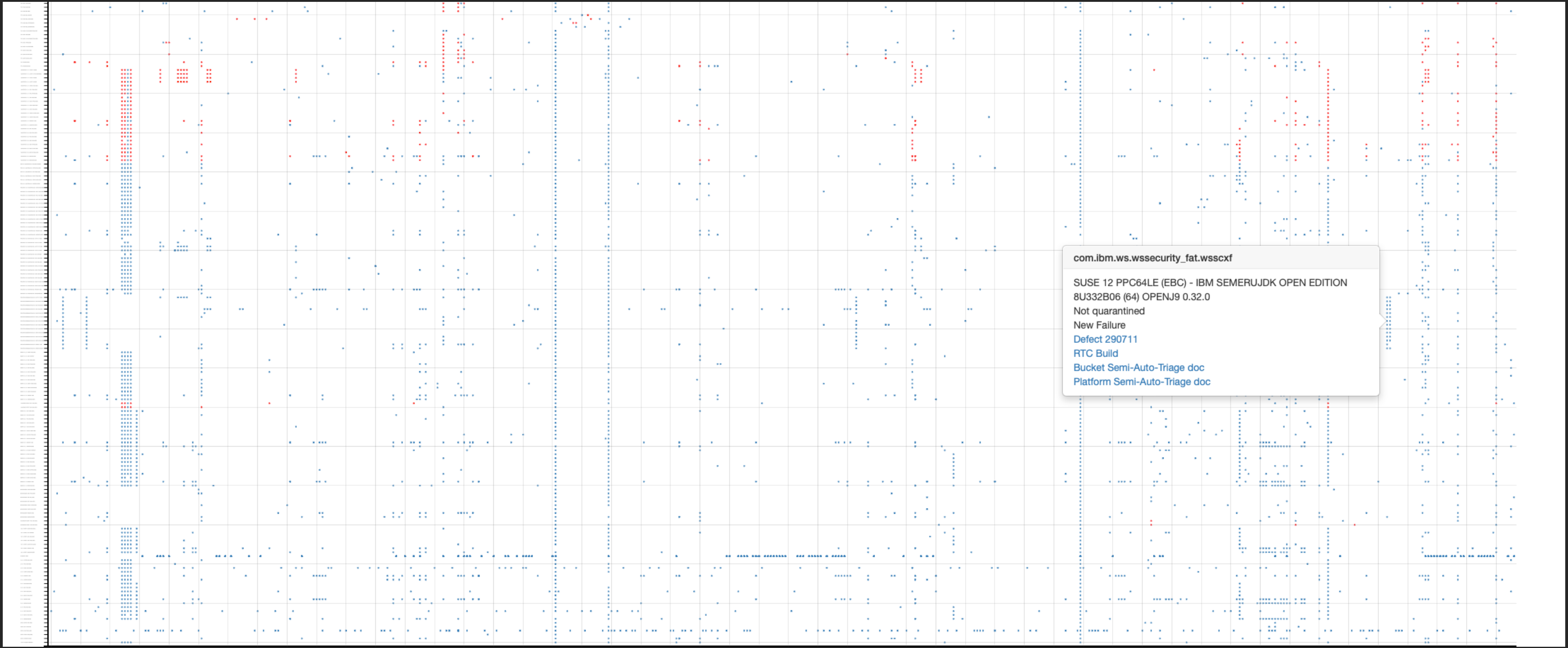
Recent Example – OS Sorting (99.91% pass rate)



Recent Example – JDK Sorting (99.91% pass rate)



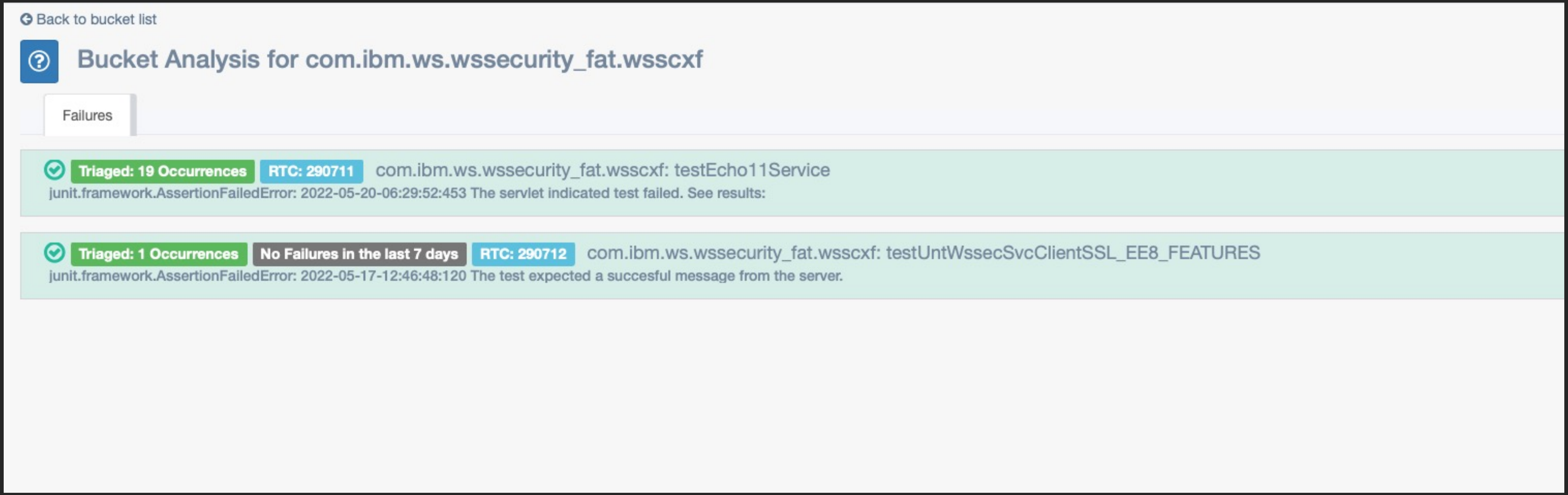
Select dots and request auto-triage



com.ibm.ws.wssecurity_fat.wsscxf

SUSE 12 PPC64LE (EBC) - IBM SEMERUJDK OPEN EDITION 8U332B06 (64) OPENJ9 0.32.0
Not quarantined
New Failure
Defect 290711
RTC Build
Bucket Semi-Auto-Triage doc
Platform Semi-Auto-Triage doc

Analysis identifies multiple issues



Back to bucket list

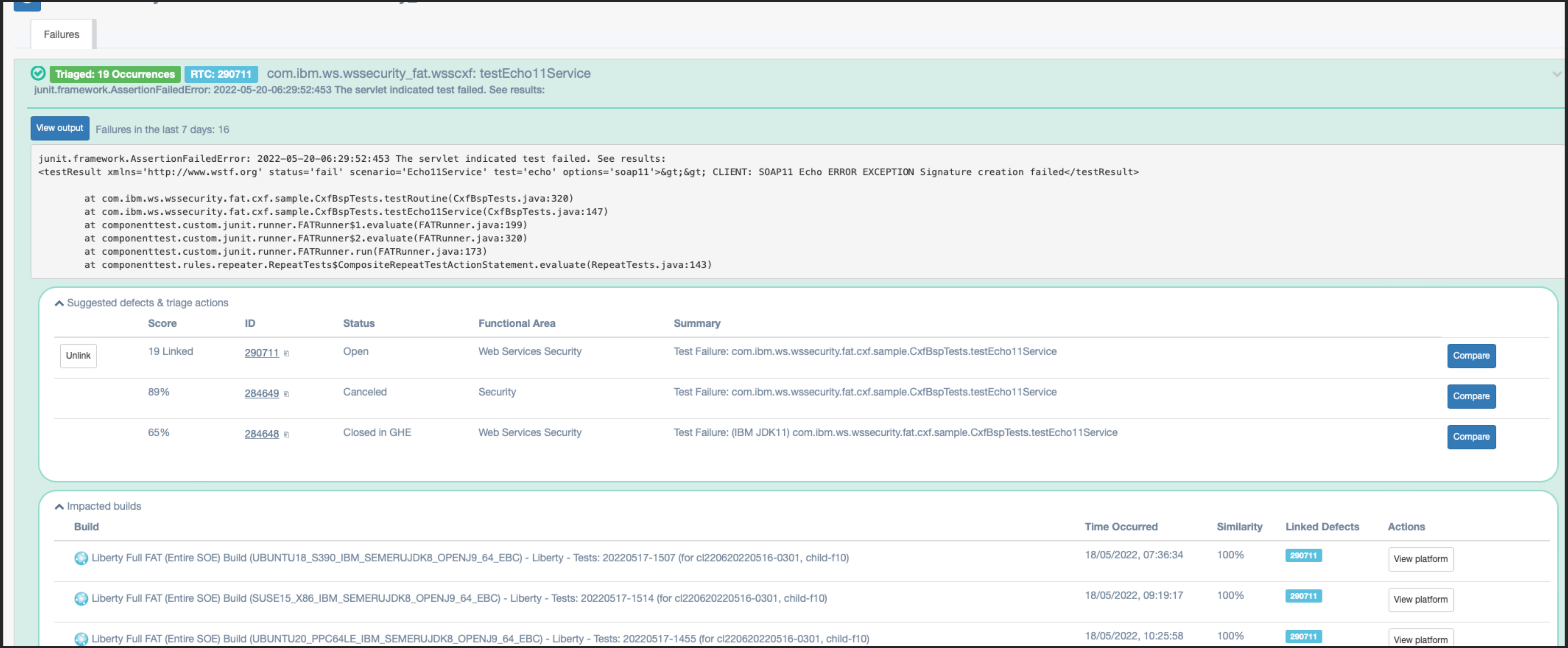
Bucket Analysis for com.ibm.ws.wssecurity_fat.wsscxf

Failures

Triaged: 19 Occurrences RTC: 290711 com.ibm.ws.wssecurity_fat.wsscxf: testEcho11Service
junit.framework.AssertionFailedError: 2022-05-20-06:29:52:453 The servlet indicated test failed. See results:

Triaged: 1 Occurrences No Failures in the last 7 days RTC: 290712 com.ibm.ws.wssecurity_fat.wsscxf: testUntWssecSvcClientSSL_EE8_FEATURES
junit.framework.AssertionFailedError: 2022-05-17-12:46:48:120 The test expected a succesful message from the server.

Guided Triage and Defect Linking



Triaged: 19 Occurrences RTC: 290711 com.ibm.ws.wssecurity_fat.wsscxf: testEcho11Service

junit.framework.AssertionFailedError: 2022-05-20-06:29:52:453 The servlet indicated test failed. See results:

Failures in the last 7 days: 16

```
junit.framework.AssertionFailedError: 2022-05-20-06:29:52:453 The servlet indicated test failed. See results:
<testResult xmlns='http://www.wstf.org' status='fail' scenario='Echo11Service' test='echo' options='soap11'>&gt;&gt; CLIENT: SOAP11 Echo ERROR EXCEPTION Signature creation failed</testResult>

        at com.ibm.ws.wssecurity.fat.cxf.sample.CxfBspTests.testRoutine(CxfBspTests.java:320)
        at com.ibm.ws.wssecurity.fat.cxf.sample.CxfBspTests.testEcho11Service(CxfBspTests.java:147)
        at componenttest.custom.junit.runner.FATRunner$1.evaluate(FATRunner.java:199)
        at componenttest.custom.junit.runner.FATRunner$2.evaluate(FATRunner.java:320)
        at componenttest.custom.junit.runner.FATRunner.run(FATRunner.java:173)
        at componenttest.rules.repeater.RepeatTests$CompositeRepeatTestActionStatement.evaluate(RepeatTests.java:143)
```

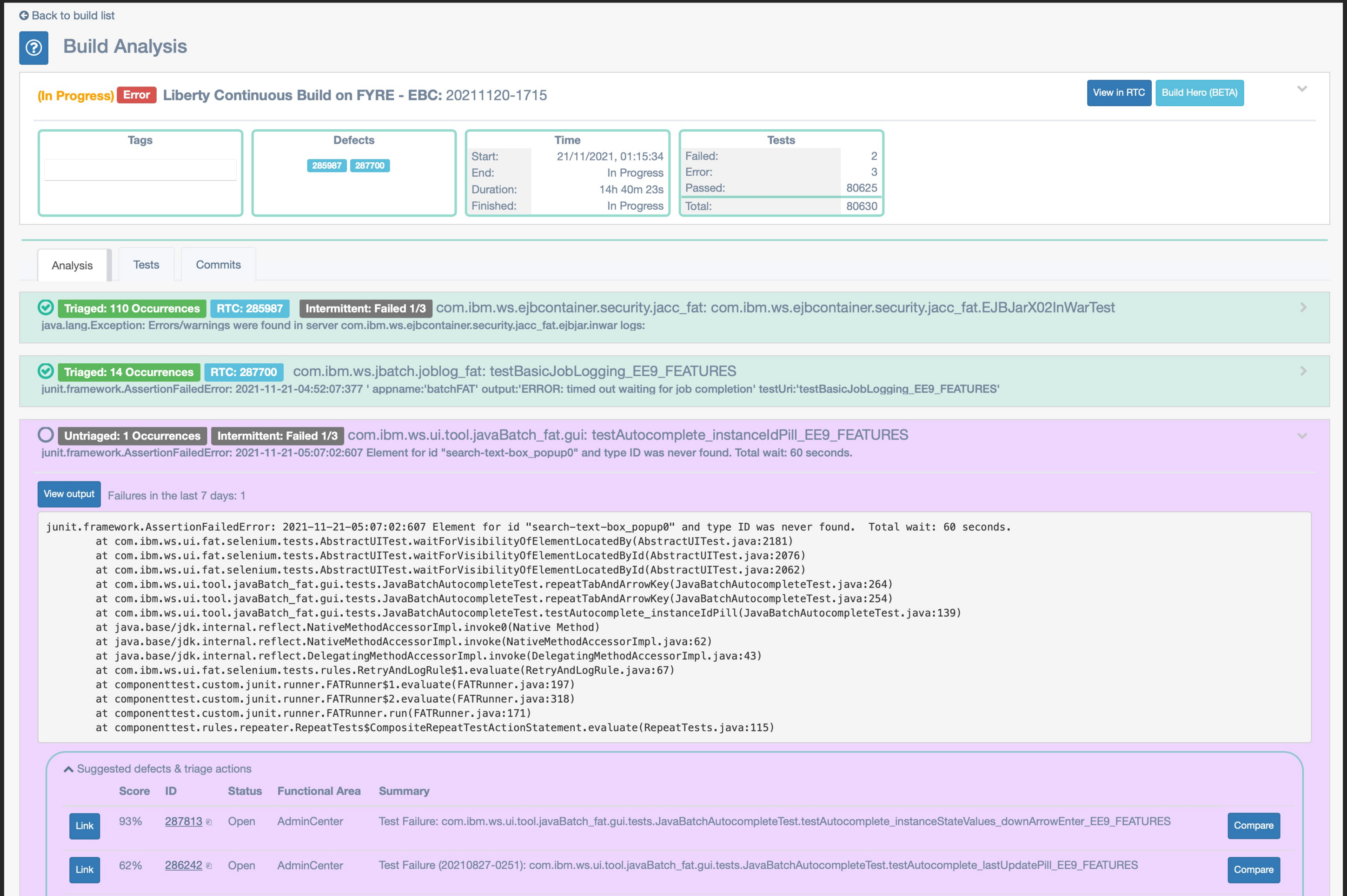
Suggested defects & triage actions

	Score	ID	Status	Functional Area	Summary	
Unlink	19 Linked	290711	Open	Web Services Security	Test Failure: com.ibm.ws.wssecurity.fat.cxf.sample.CxfBspTests.testEcho11Service	Compare
	89%	284649	Canceled	Security	Test Failure: com.ibm.ws.wssecurity.fat.cxf.sample.CxfBspTests.testEcho11Service	Compare
	65%	284648	Closed in GHE	Web Services Security	Test Failure: (IBM JDK11) com.ibm.ws.wssecurity.fat.cxf.sample.CxfBspTests.testEcho11Service	Compare

Impacted builds

Build	Time Occurred	Similarity	Linked Defects	Actions
Liberty Full FAT (Entire SOE) Build (UBUNTU18_S390_IBM_SEMERUJDK8_OPENJ9_64_EBC) - Liberty - Tests: 20220517-1507 (for cl220620220516-0301, child-f10)	18/05/2022, 07:36:34	100%	290711	View platform
Liberty Full FAT (Entire SOE) Build (SUSE15_X86_IBM_SEMERUJDK8_OPENJ9_64_EBC) - Liberty - Tests: 20220517-1514 (for cl220620220516-0301, child-f10)	18/05/2022, 09:19:17	100%	290711	View platform
Liberty Full FAT (Entire SOE) Build (UBUNTU20_PPC64LE_IBM_SEMERUJDK8_OPENJ9_64_EBC) - Liberty - Tests: 20220517-1455 (for cl220620220516-0301, child-f10)	18/05/2022, 10:25:58	100%	290711	View platform

Build Monitoring: Automated Triage



Back to build list

Build Analysis

(In Progress) Error **Liberty Continuous Build on FYRE - EBC:** 20211120-1715

View in RTC | Build Hero (BETA)

Tags

Defects: 285987 287700

Time	
Start:	21/11/2021, 01:15:34
End:	In Progress
Duration:	14h 40m 23s
Finished:	In Progress

Tests	
Failed:	2
Error:	3
Passed:	80625
Total:	80630

Analysis | Tests | Commits

Triaged: 110 Occurrences | RTC: 285987 | Intermittent: Failed 1/3 com.ibm.ws.ejbcontainer.security.jacc_fat: com.ibm.ws.ejbcontainer.security.jacc_fat.EJBJarX02InWarTest
java.lang.Exception: Errors/warnings were found in server com.ibm.ws.ejbcontainer.security.jacc_fat.ejbjar.inwar logs:

Triaged: 14 Occurrences | RTC: 287700 com.ibm.ws.jbatch.joblog_fat: testBasicJobLogging_EE9_FEATURES
junit.framework.AssertionFailedError: 2021-11-21-04:52:07:377 ' appname:'batchFAT' output:'ERROR: timed out waiting for job completion' testUri:'testBasicJobLogging_EE9_FEATURES'

Untriaged: 1 Occurrences | Intermittent: Failed 1/3 com.ibm.ws.ui.tool.javaBatch_fat.gui: testAutocomplete_instanceIdPill_EE9_FEATURES
junit.framework.AssertionFailedError: 2021-11-21-05:07:02:607 Element for id "search-text-box_popup0" and type ID was never found. Total wait: 60 seconds.

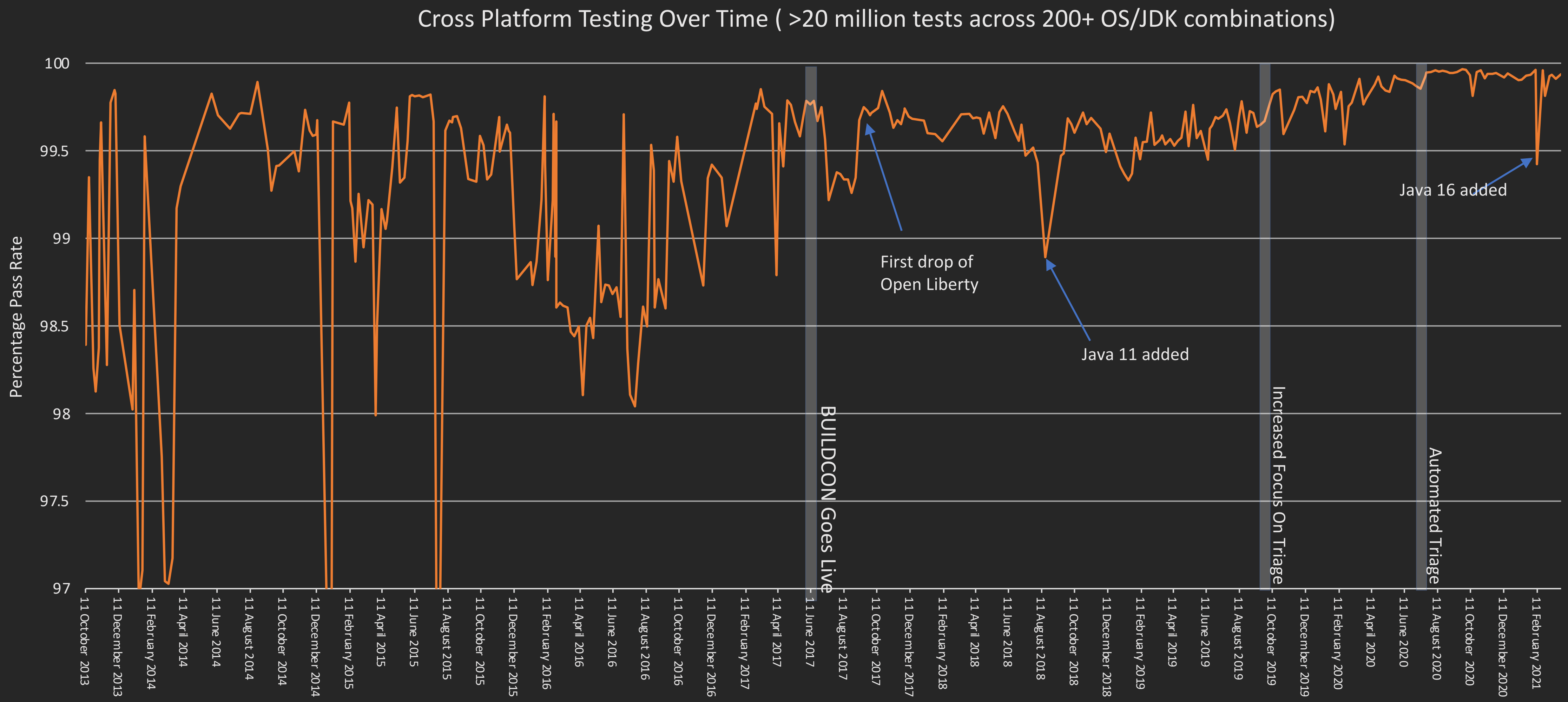
View output Failures in the last 7 days: 1

```
junit.framework.AssertionFailedError: 2021-11-21-05:07:02:607 Element for id "search-text-box_popup0" and type ID was never found.  Total wait: 60 seconds.
	at com.ibm.ws.ui.fat.selenium.tests.AbstractUITest.waitForVisibilityOfElementLocatedBy(AbstractUITest.java:2181)
	at com.ibm.ws.ui.fat.selenium.tests.AbstractUITest.waitForVisibilityOfElementLocatedById(AbstractUITest.java:2076)
	at com.ibm.ws.ui.fat.selenium.tests.AbstractUITest.waitForVisibilityOfElementLocatedById(AbstractUITest.java:2062)
	at com.ibm.ws.ui.tool.javaBatch_fat.gui.tests.JavaBatchAutocompleteTest.repeatTabAndArrowKey(JavaBatchAutocompleteTest.java:264)
	at com.ibm.ws.ui.tool.javaBatch_fat.gui.tests.JavaBatchAutocompleteTest.repeatTabAndArrowKey(JavaBatchAutocompleteTest.java:254)
	at com.ibm.ws.ui.tool.javaBatch_fat.gui.tests.JavaBatchAutocompleteTest.testAutocomplete_instanceIdPill(JavaBatchAutocompleteTest.java:139)
	at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at com.ibm.ws.ui.fat.selenium.tests.rules.RetryAndLogRule$1.evaluate(RetryAndLogRule.java:67)
	at componenttest.custom.junit.runner.FATRunner$1.evaluate(FATRunner.java:197)
	at componenttest.custom.junit.runner.FATRunner$2.evaluate(FATRunner.java:318)
	at componenttest.custom.junit.runner.FATRunner.run(FATRunner.java:171)
	at componenttest.rules.repeater.RepeatTests$CompositeRepeatTestActionStatement.evaluate(RepeatTests.java:115)
```

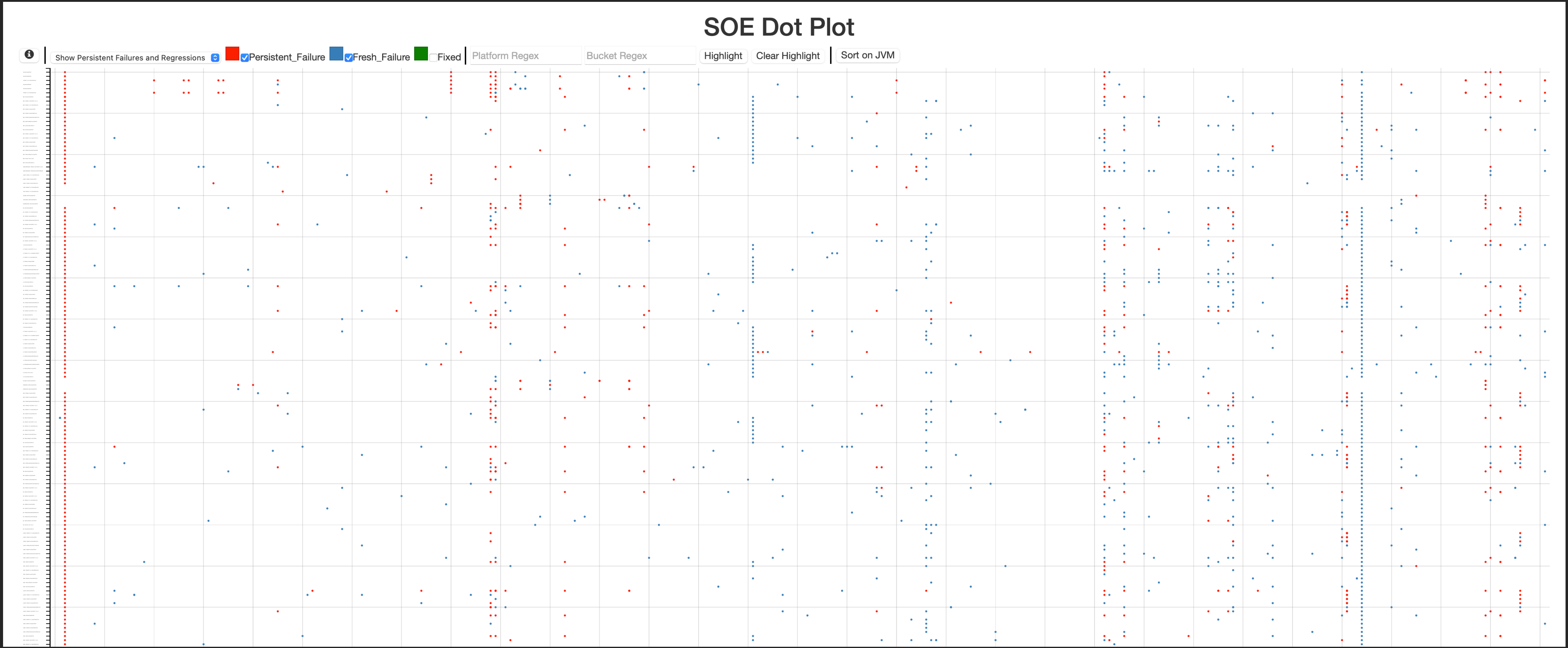
Suggested defects & triage actions

	Score	ID	Status	Functional Area	Summary	
Link	93%	287813	Open	AdminCenter	Test Failure: com.ibm.ws.ui.tool.javaBatch_fat.gui.tests.JavaBatchAutocompleteTest.testAutocomplete_instanceStateValues_downArrowEnter_EE9_FEATURES	Compare
Link	62%	286242	Open	AdminCenter	Test Failure (20210827-0251): com.ibm.ws.ui.tool.javaBatch_fat.gui.tests.JavaBatchAutocompleteTest.testAutocomplete_lastUpdatePill_EE9_FEATURES	Compare

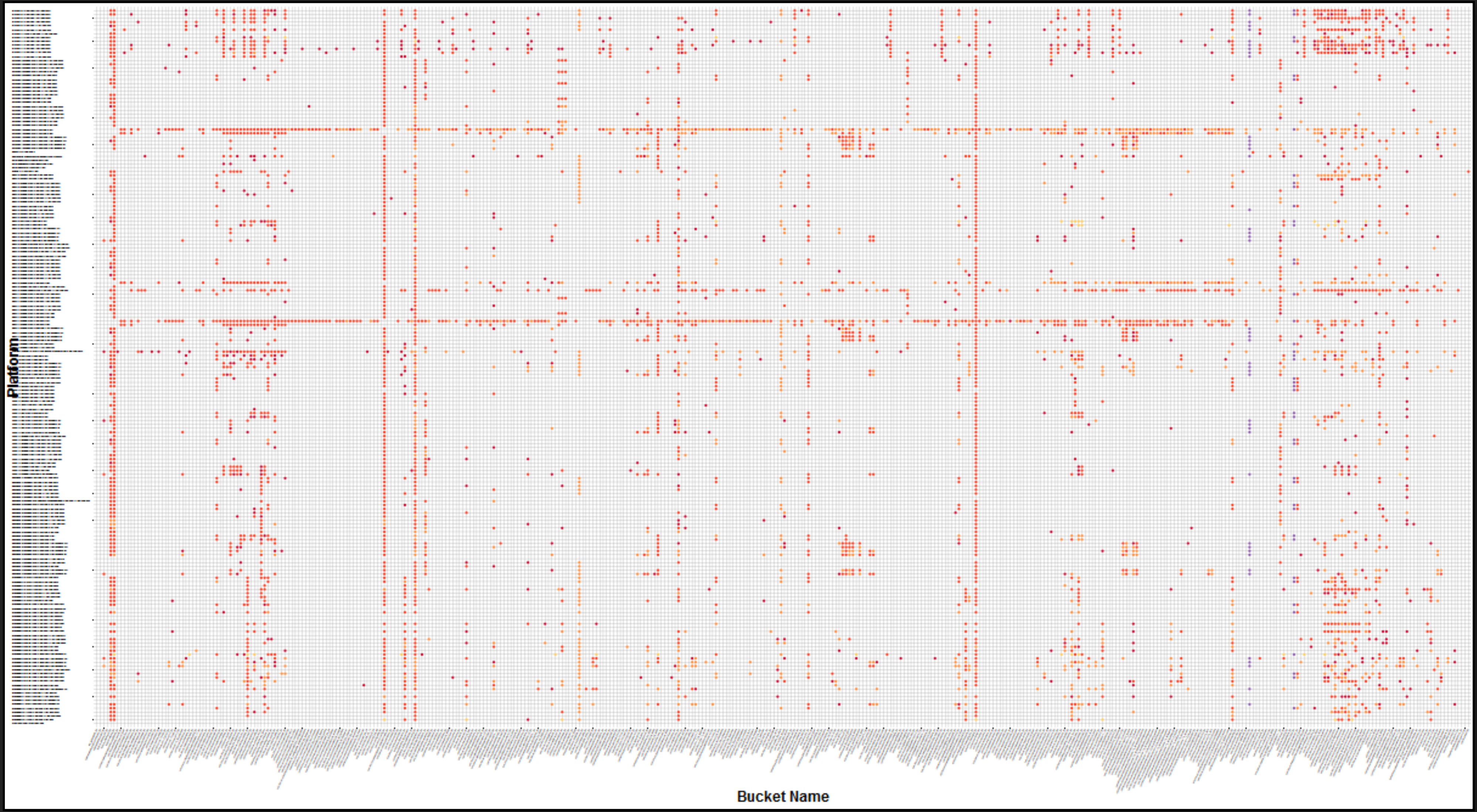
Highlighting the impact



Recent Example (99.93% pass rate)



Older example (99.5% pass rate)



Open Liberty

Useful Links

Why choose Liberty for Microservices
https://ibm.biz/6ReasonsWhyLiberty

Choosing the right Java runtime
https://ibm.biz/ChooseJavaRuntime

How to approach application modernization
https://ibm.biz/ModernizeJavaApps

Open Liberty Site
https://www.openliberty.io

Open Liberty Guides
https://www.openliberty.io/guides

A lightweight open framework for building fast and efficient cloud-native Java microservices.

Build cloud-native apps and microservices while running only what you need. Open Liberty™ is the most flexible server runtime available to Java™ developers in this solar system.

https://openliberty.io →

Questions?

Open Liberty