intellidjent

Rhythmic Development Automation

[intellidjent] Presenting the Problem

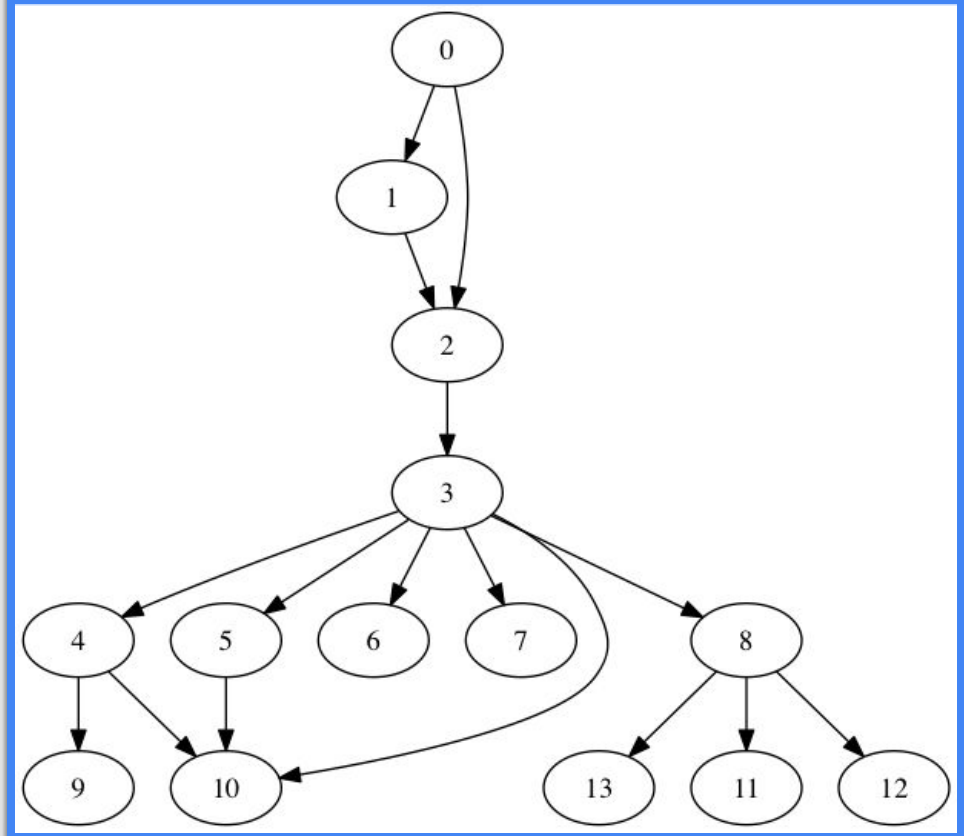
- Simplify the act of creating Continuous Integration (CI) tests for large and complex Maven module situations
- Guarantee that all affected modules are tested
- Guarantee an isolated development environment to reduce inter-engineer conflicts

[intellidjent]

EXAMPLE:

A dependency graph that has 14 Github Repositories that contain 1 or more Maven modules. Each of the leaves is a processing module that will be packaged as a shaded jar to be sent to Hadoop.

intellidjent has to understand the module graph structure and return a workflow representation that Jenkins can process.

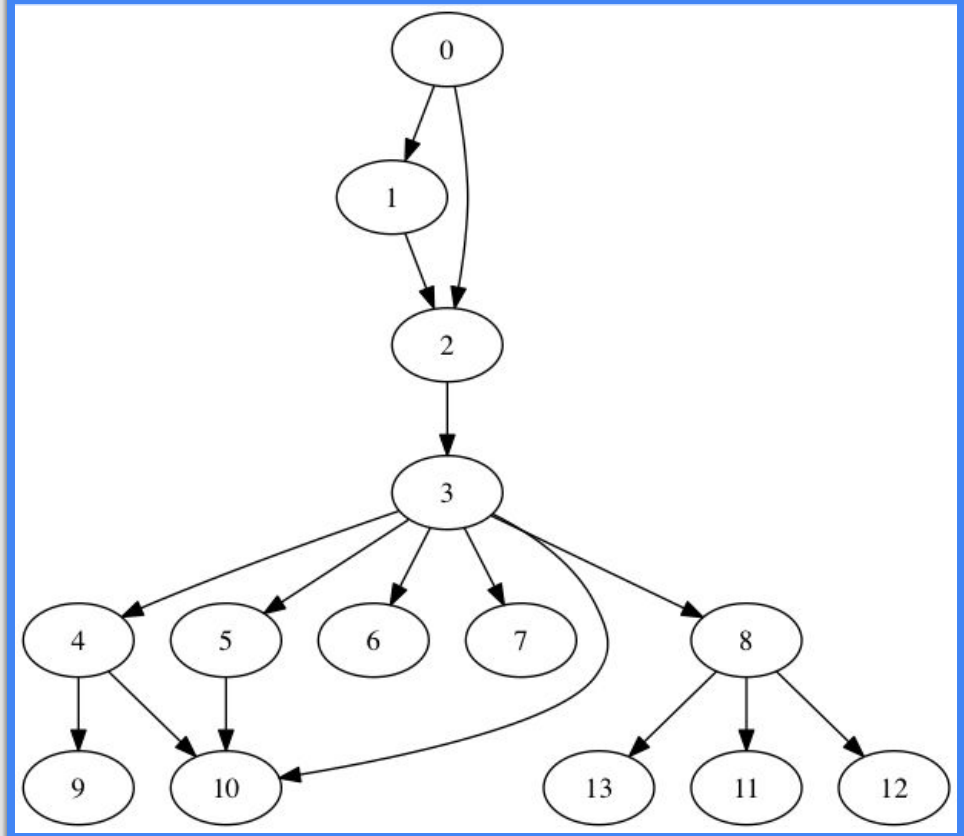


[intellidjent]

EXAMPLE cont:

If it is assumed that this graph given is a cohesive tech stack then all 14 nodes will need to build continuously for our Stable builds.

Intellidjent, alpha version, will build our Stable stack in sequential order by following a topological sorted order (i.e. 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 13 -> 11 -> 12)

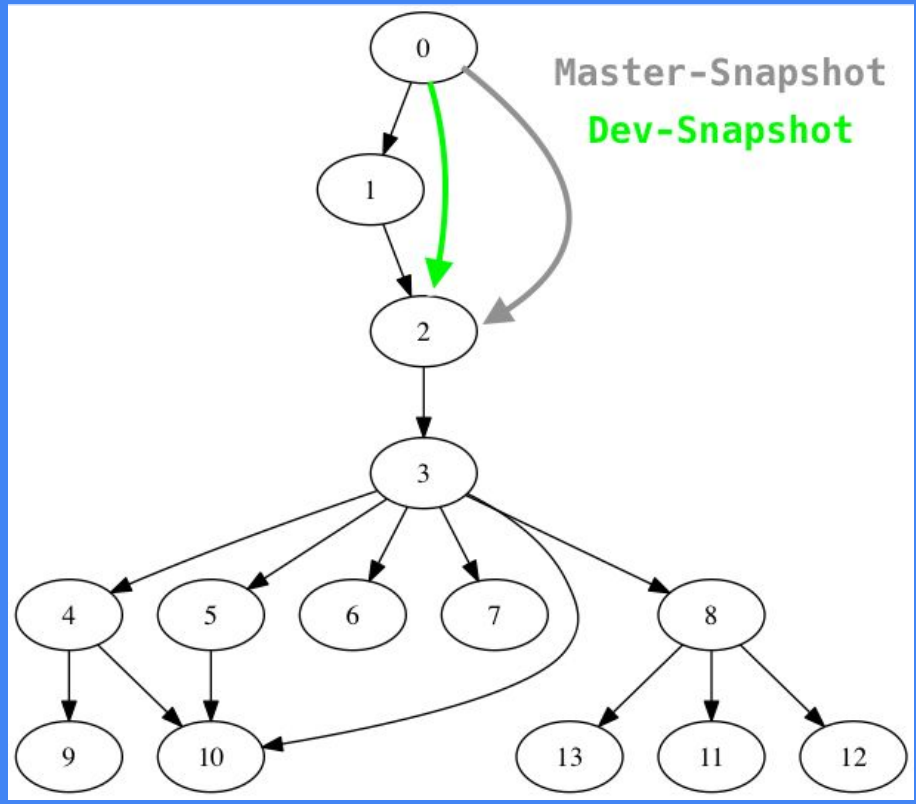


[intellidjent]

DEV STORY EXAMPLE:
TWO IMPORTANT QUESTIONS

What happens when I'm developing on repository 2 and a fellow engineer is developing on repository 0?

Will I consume their development builds that are considered unstable or will I be consuming the latest code from master that is considered stable?

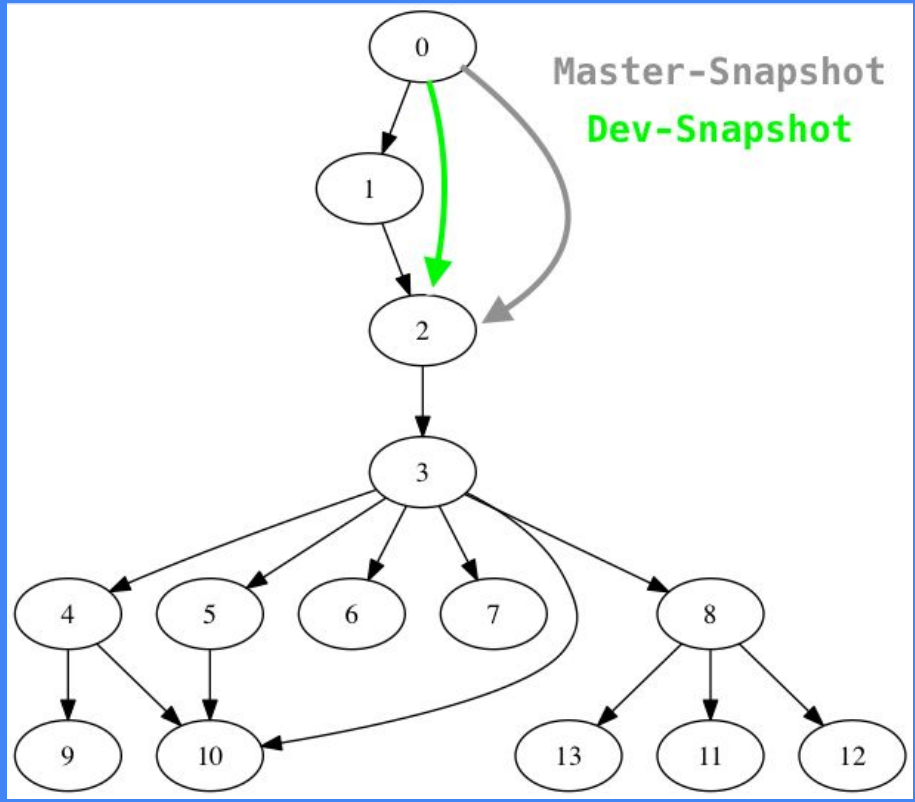


[intellidjent]

DEV STORY EXAMPLE:
QUESTION 1

What happens when I'm developing on repository 2 and a fellow engineer is developing on repository 0?

Option 1: She could have isolated her snapshot (unique snapshot identifier that I can't consume unless I purposely try)

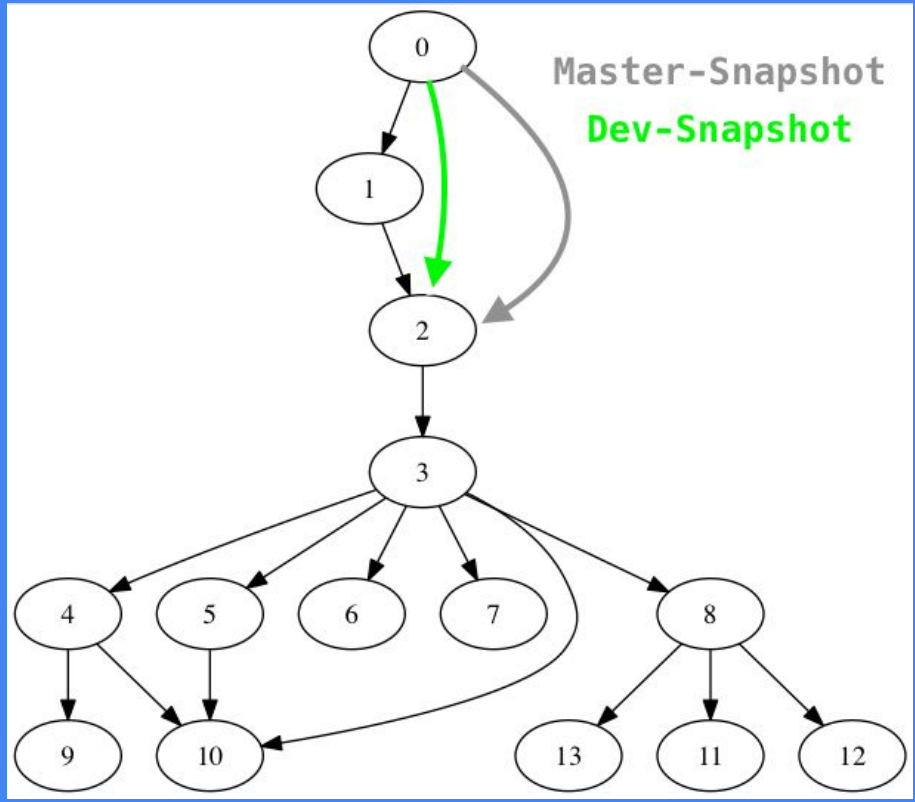


[intellidjent]

DEV STORY EXAMPLE:
QUESTION 1

What happens when I’m developing on repository 2 and a fellow engineer is developing on repository 0?

Option 2: She could have left the version of repository 0 in her dev branch as X.X.X-Master-SNAPSHOT

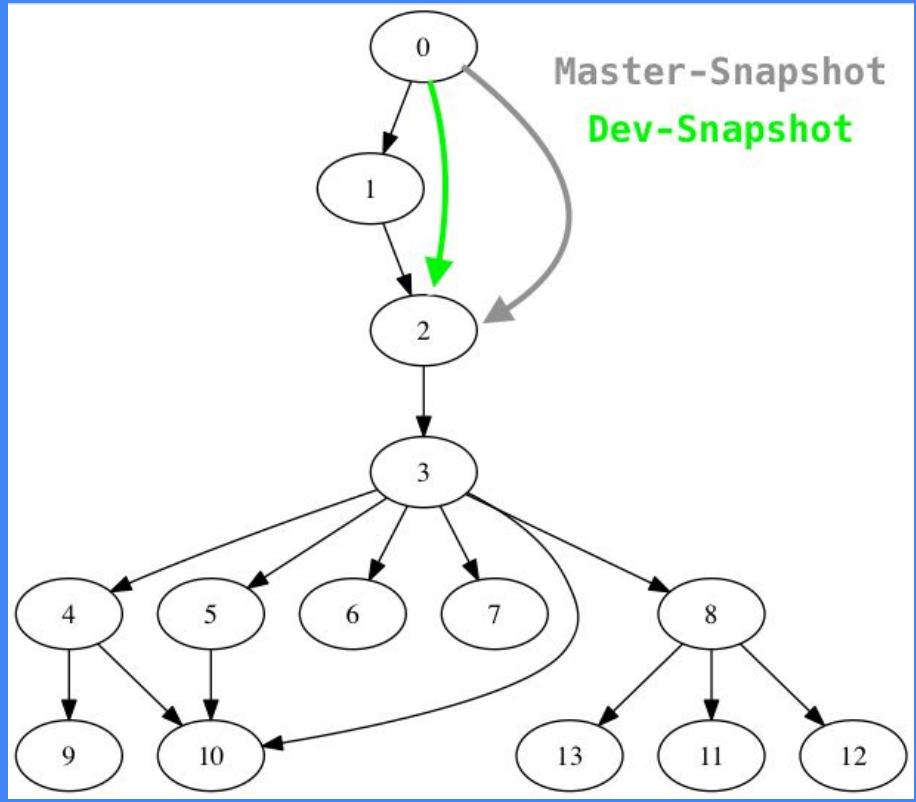


[intellidjent]

DEV STORY EXAMPLE:
QUESTION 2

Will I consume their development builds that are considered unstable or will I be consuming the latest code from master that is considered stable?

Option 1: Because my fellow engineer isolated their module version, I won't consume her changes. But we can't guarantee every engineer will be as proactive.

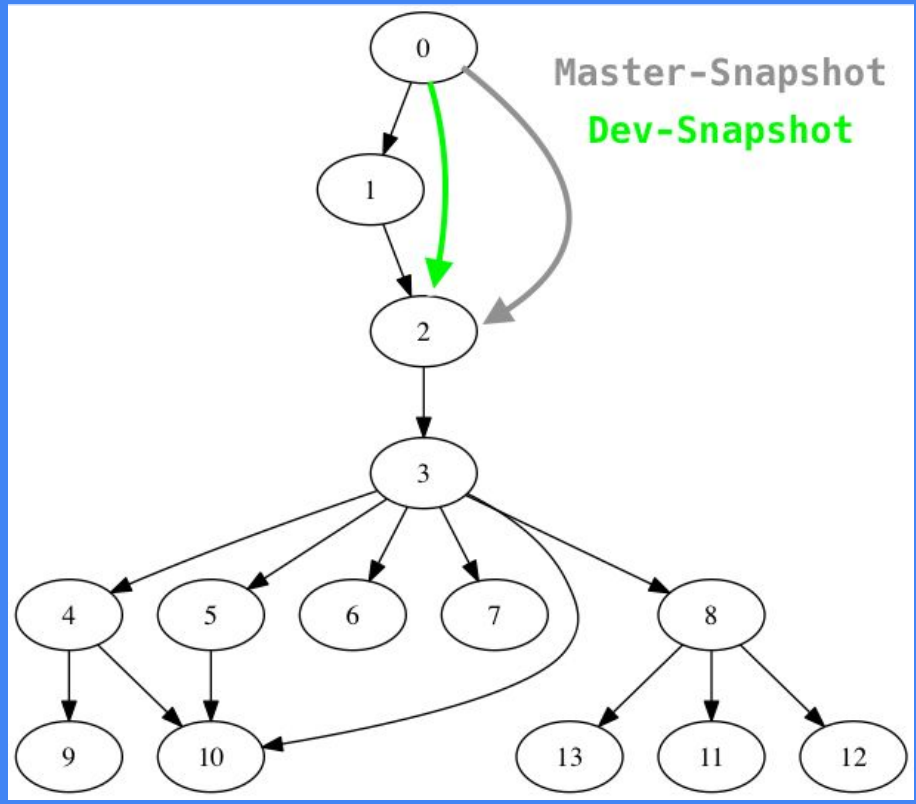


[intellidjent]

DEV STORY EXAMPLE:
QUESTION 2

Will I consume their development builds that are considered unstable or will I be consuming the latest code from master that is considered stable?

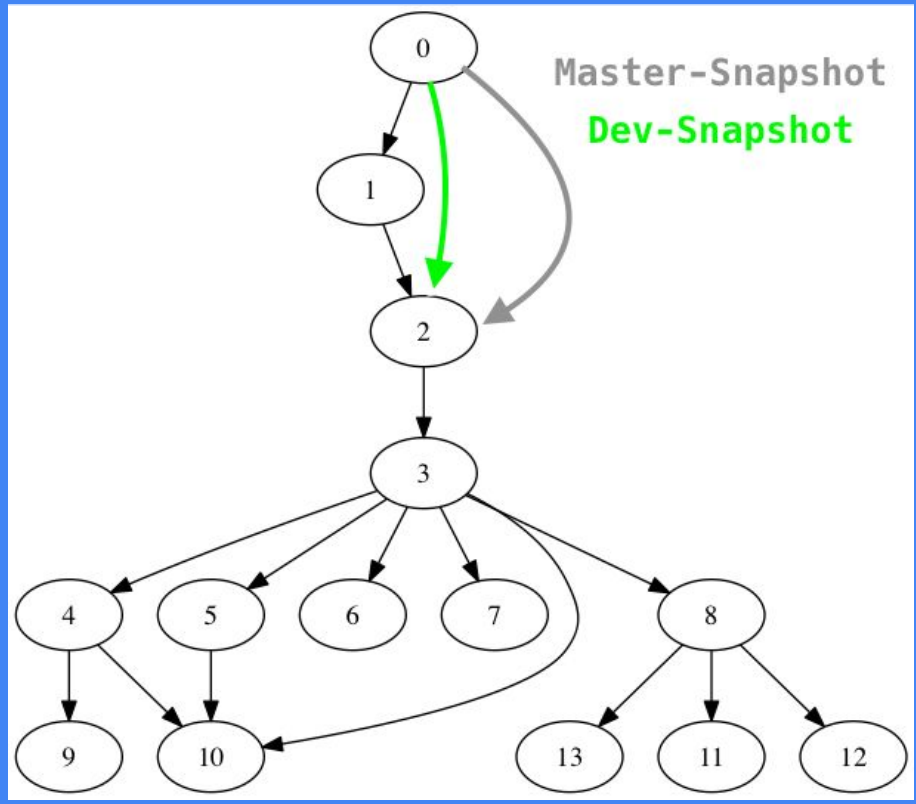
Option 2: My fellow engineer left their repository's code to be version X.X.X-Master-Snapshot. Not good news for me because I'm now going to consume her development code from a build 5 minutes ago compared to a true Master branch build from 10 minutes ago



[intellidjent]

DEV STORY EXAMPLE:
HYPOTHESIS

This is why each developer should have an isolated environment when building versions that will ultimately be hosted on the remote Maven SNAPSHOT repository.



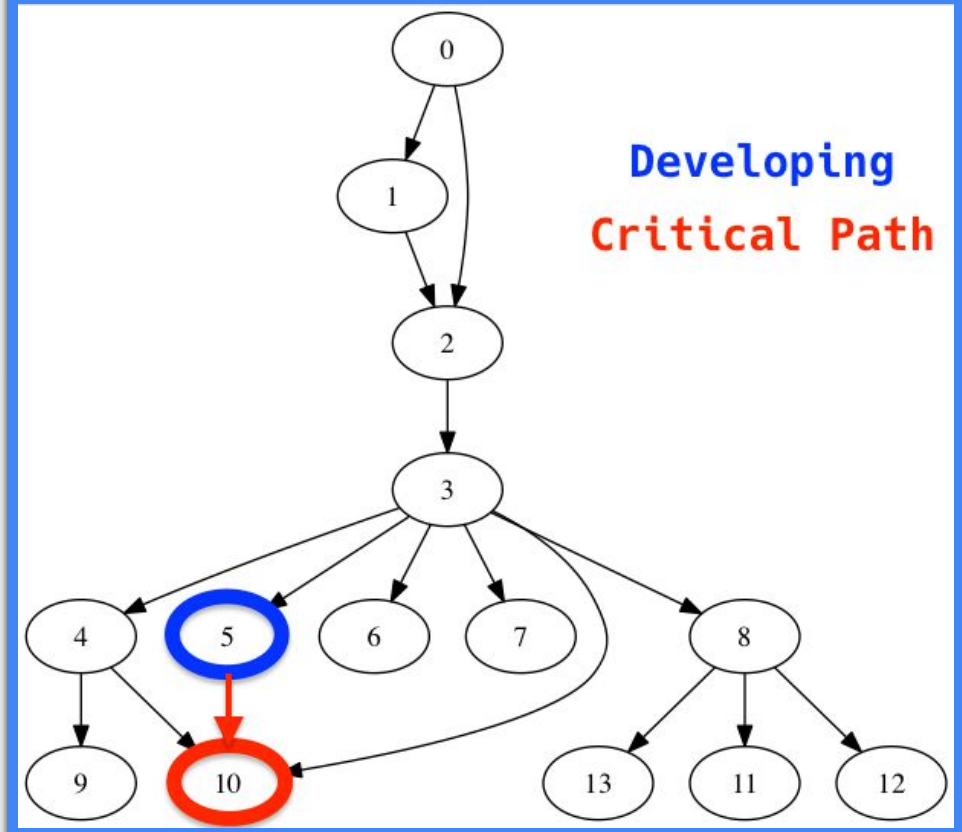
[intellidjent] Situation One

- If an engineer is working sufficiently close to the top of the tech stack[1], the only modules that need to be added to a workflow is itself and its descendants
- The developed module will simply consume all Stable builds of its direct ancestry

1. Top of the tech stack is considered a leaf node when looking at a DAG or a processing shaded-jar that is deployed as a runnable package

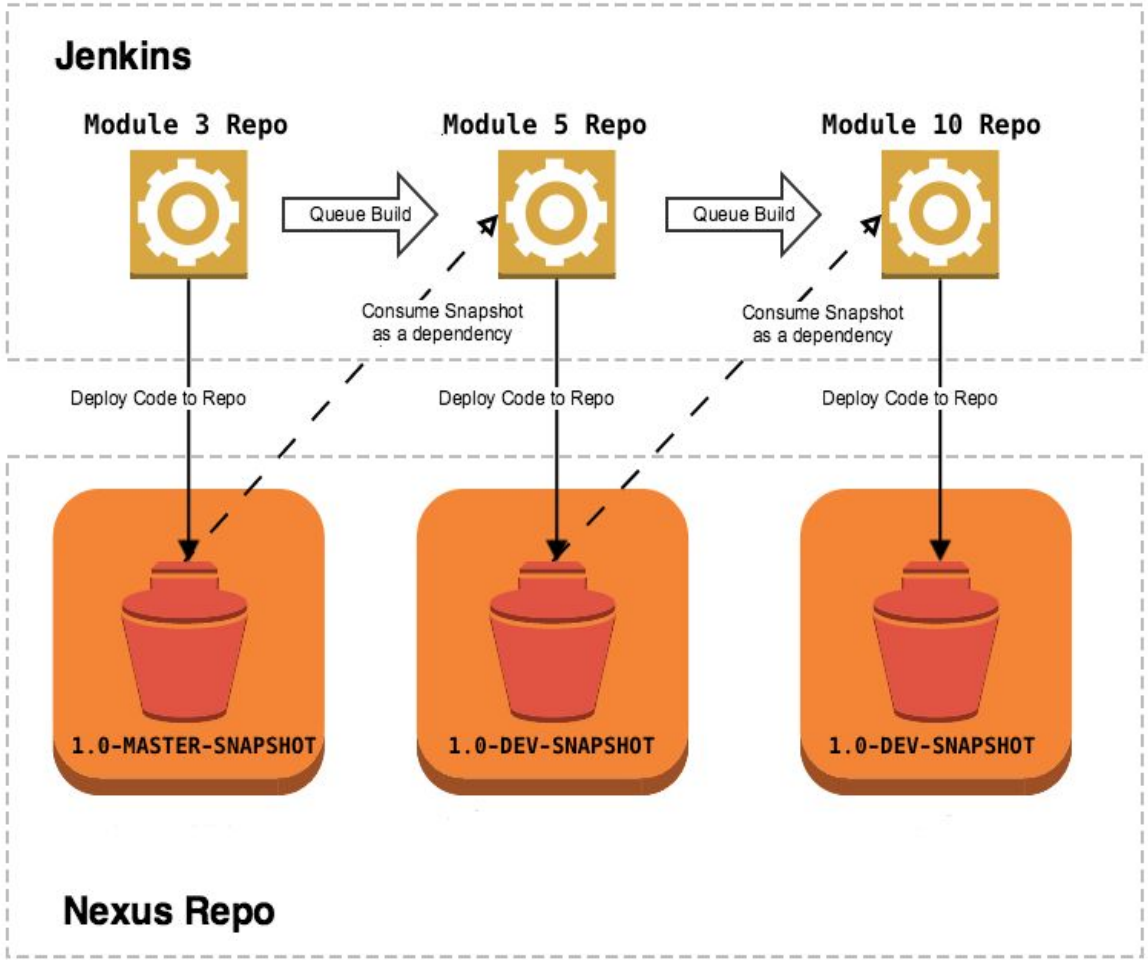
[intellidjent]

- **module 5** is the only module with a dev change for this example JIRA task.
- intellidjent will create a workflow that ingests master SNAPSHOTs as **module 5** and **module 10's** ancestors and create an isolated development SNAPSHOT of **module 5** and **module 10**
- Other intellidjent workflows are not aware of the development changes being made.



[intellidjent]

Focusing on the build path 3 -> 5 -> 10 for situation one, the reactive build looks like the adjacent workflow.



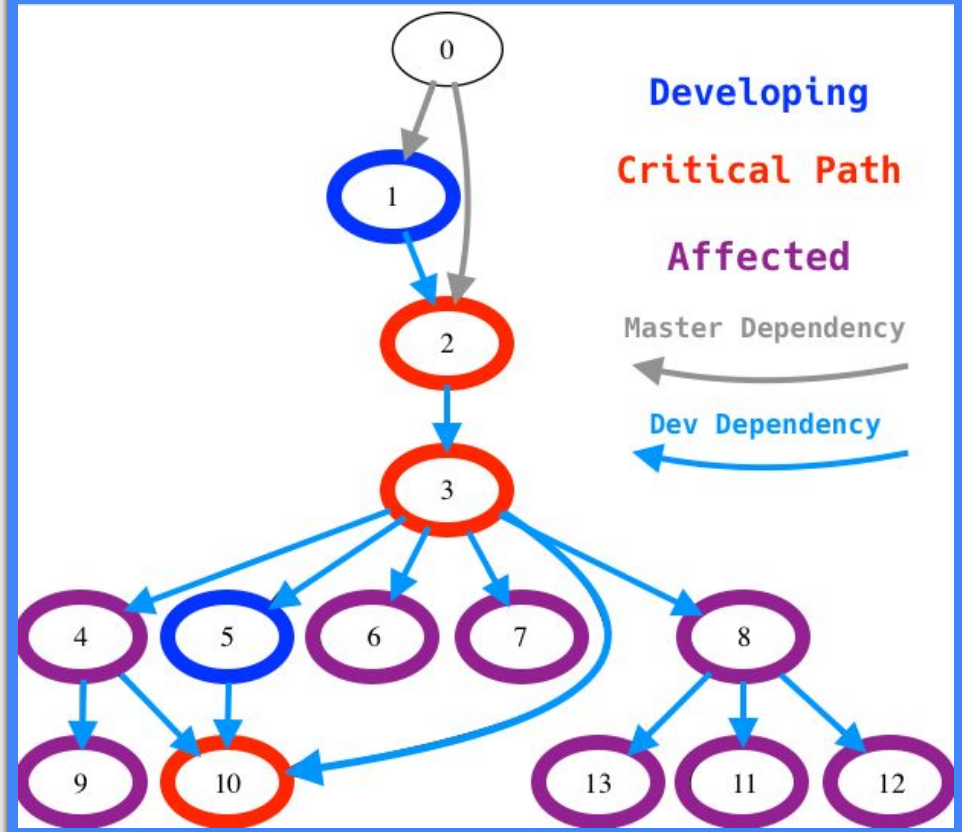
[intellidjent] Situation Two

- If an engineer is developing near the root of the maven ancestry [1] or across multiple modules, she must be aware of possible side-effects that can occur far away in the descendants. Passivity can only be guaranteed if all consumers are tested.
- This situation requires that all descendants of **module 1** are rebuilt with that new SNAPSHOT and proper integration testing is ran. At this point, the only module that is not isolated from master is **module 0**.

1. Modules near the root of the Maven ancestry tend to be services and infrastructure modules that are used and embedded in the runnable packaged jars from the leaf nodes.

[intellidjent]

- **Module 1** and **module 5** have development changes and possess several degrees of separation between one-another.
- **Critical Path** is the series of modules you are interested in testing for this development task.
- All highlighted module nodes are nodes that have to be tested to guarantee development changes are passive and are compatible with descendents

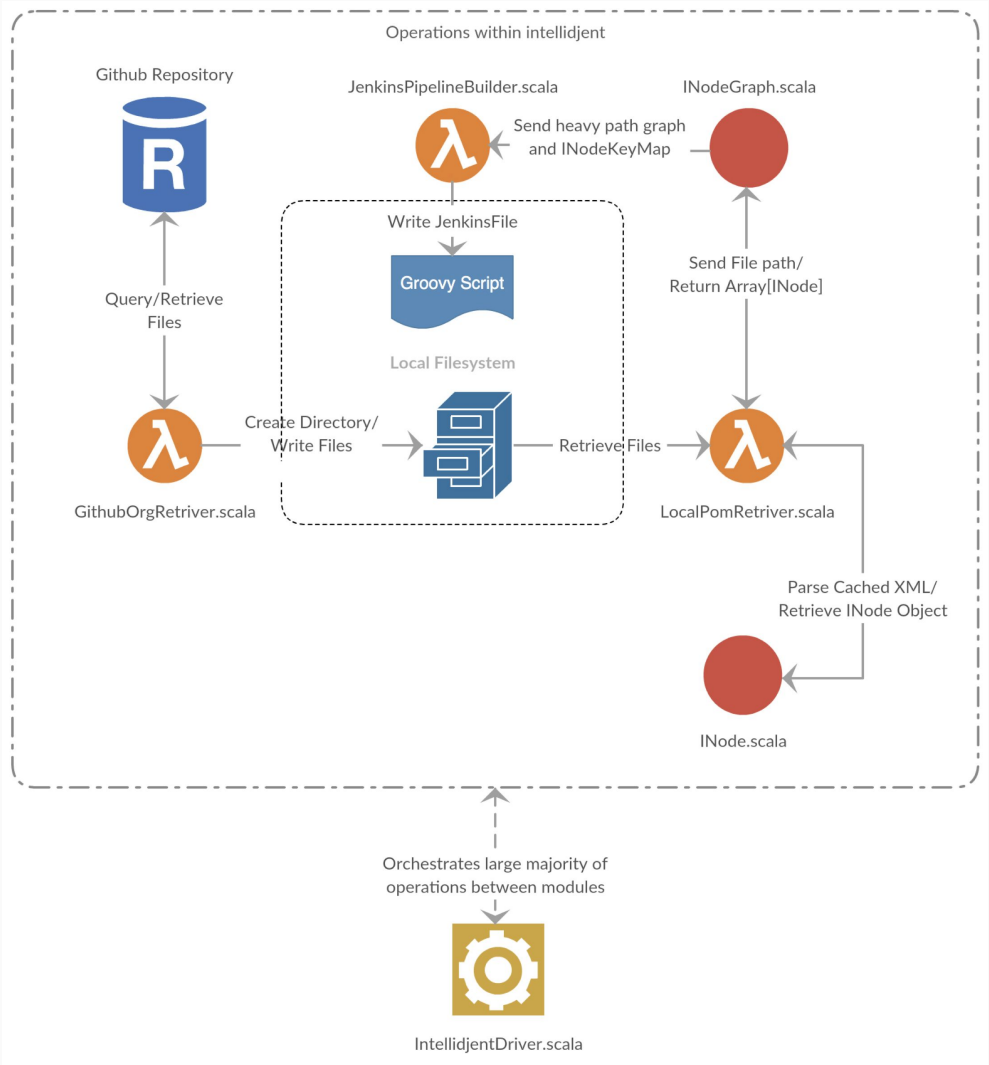


[intellidjent] Current State

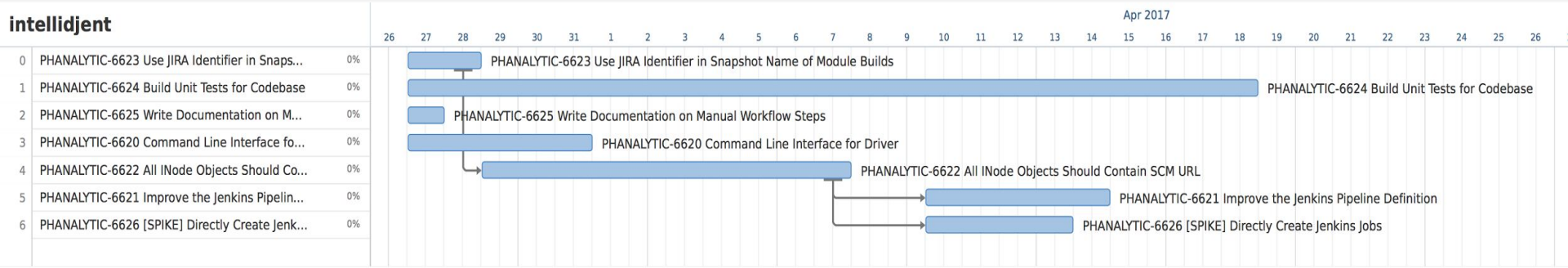
- The core operations that intellidjent needs to support have already been written out and functionally tested.
- JIRAs have been created under https://jira2.cerner.com/browse/PHANALYTIC-6619 that define what remains to be done.

[intellidjent] Architecture

Simple Abstraction that shows code communication and orchestration.



[intellidjent] Timeline to Finish Development



[intellidjent] What to Expect At Completion

Engineer usecase flowchart