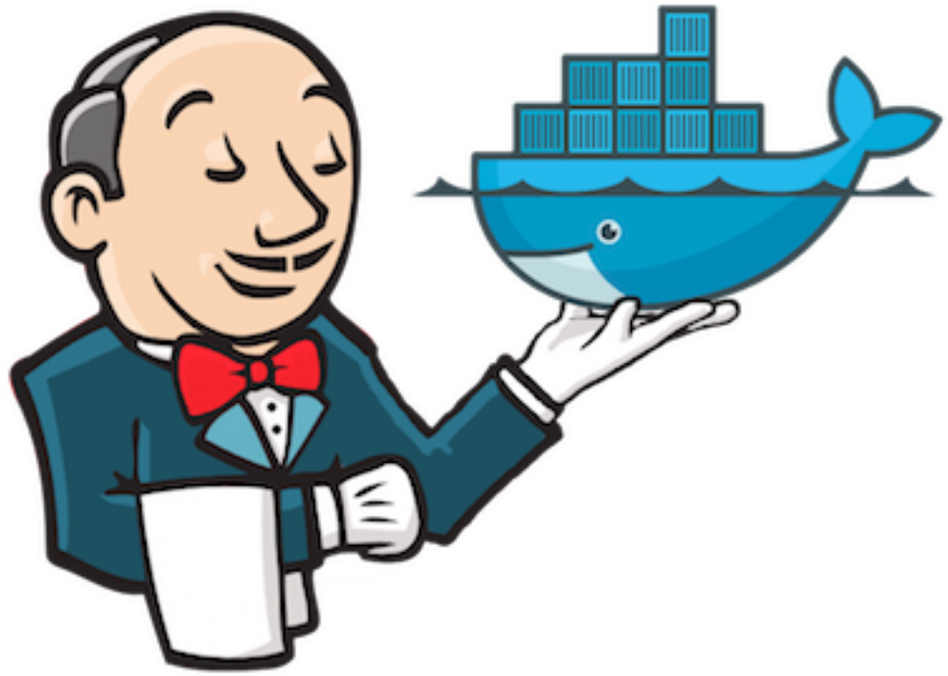
Continuous Builds

Jenkins, Docker, and Phabricator

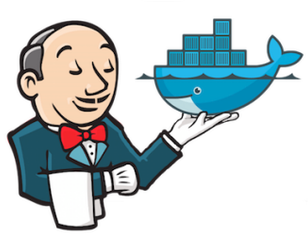


Matt Dickenson

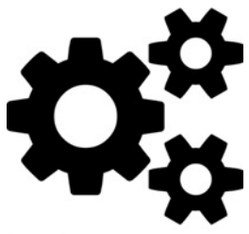
December 15, 2016

Agenda

Jenkins Basics



Docker on Jenkins

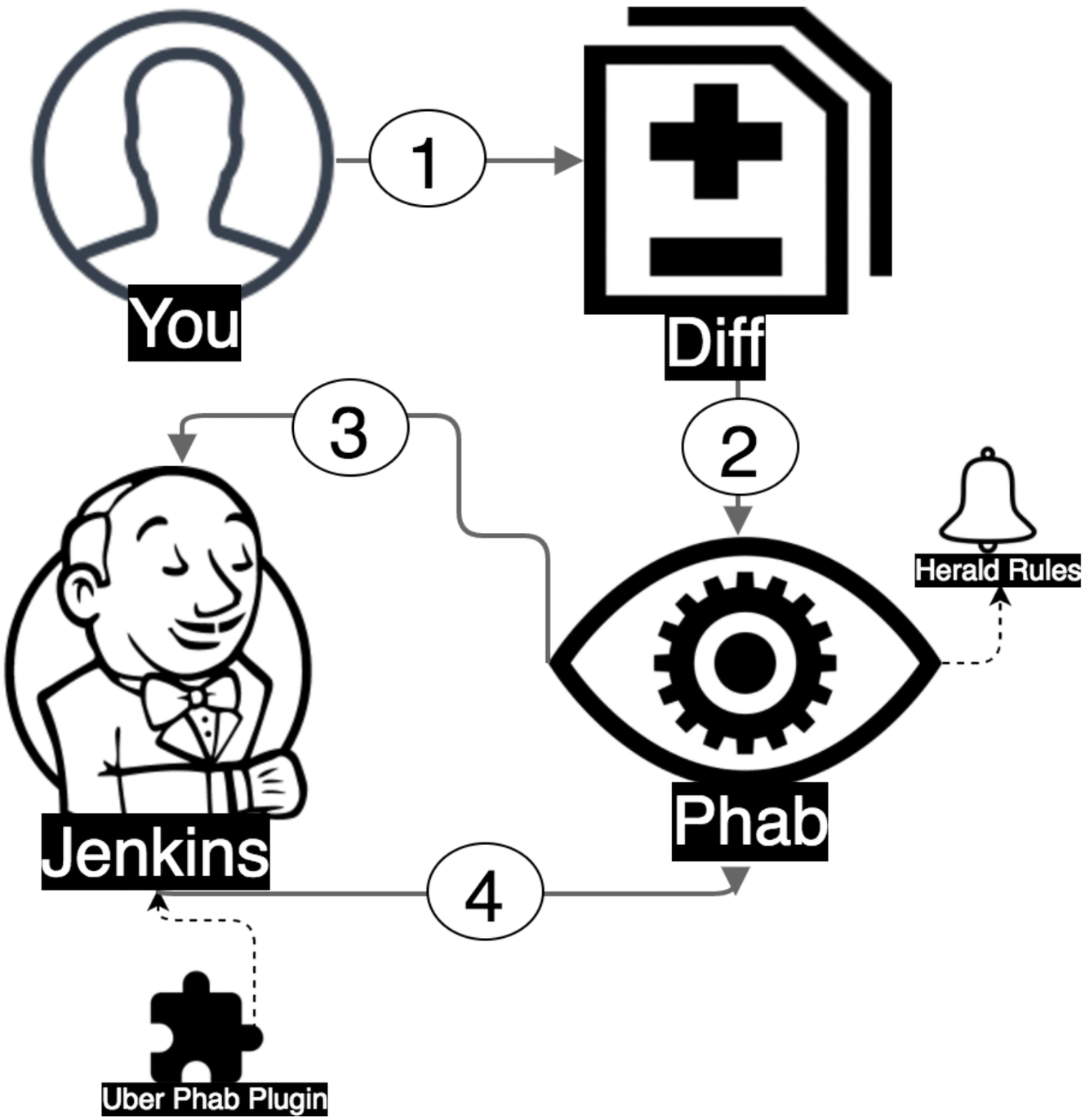


Phabricator

Debugging on Jenkins

Takeaways

Overview



Jenkins Build Status

 **Succeeded**	Build exited with code 0 (nothing reported a failure or instability)
 **Failed**	Build exited with code 1 (most likely a test failure)
 **Unstable**	The build itself completed, but something went wrong
 **Aborted**	Someone/something cancelled the build

Jenkins “Weather Report”

All recent builds succeeded

1 (of 5) recent builds failed

2 recent builds failed

3-4 recent builds failed

All recent builds failed

Jenkins DSL

job template

```
new GenericTest().override {
    steps {
        shell('scripts/ci-build.sh')
    }
    ...
}.extend {
    publishers {
        archiveJunit('test_results/**/nosetests.xml')
        cobertura('test_results/**/coverage.xml')
    }
}.build(this,
    'test-iei-docker-build',
    ...
)
```

Jenkins DSL

build script

```
new GenericTest().override {
    steps {
        shell('scripts/ci-build.sh')
    }
    ...
}.extend {
    publishers {
        archiveJunit('test_results/**/nosetests.xml')
        cobertura('test_results/**/coverage.xml')
    }
}.build(this,
    'test-iei-docker-build',
    ...
)
```

Jenkins DSL

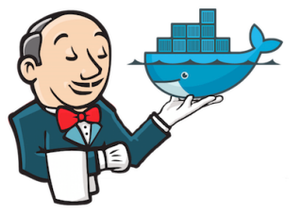
```
new GenericTest().override {
    steps {
        shell('scripts/ci-build.sh')
    }
    ...
}.extend {
    publishers {
        archiveJunit('test_results/**/nosetests.xml')
        cobertura('test_results/**/coverage.xml')
    }
}.build(this,
    'test-iei-docker-build',
    ...
)
```

test output

Jenkins DSL

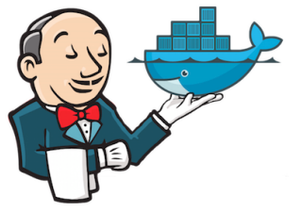
```
new GenericTest().override {
    steps {
        shell('scripts/ci-build.sh')
    }
    ...
}.extend {
    publishers {
        archiveJunit('test_results/**/nosetests.xml')
        cobertura('test_results/**/coverage.xml')
    }
}.build(this,
    'test-iei-docker-build',
    ...
)
```

job name



Docker on Jenkins

- How is it differently from building locally?
- No sudo access
- No ssh tunnels

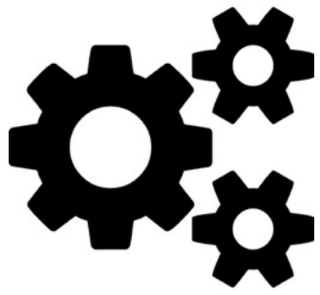


Docker on Jenkins

- Which images get built?
- How are names generated?
- How are tags generated?
- How are models fetched?
- How are tests run?

Build Artifacts Warning

- **Caution:** Be sure that your build scripts clean up any artifacts not owned by the Jenkins user (e.g. `coverage.xml`, `results.xml`, `.pyc`, and `__pycache__` files if generated within the Docker image)
- If you don't do this, the files will be stranded and the Jenkins user will not have permission to delete them (and you won't be able to change permissions later)
- The normal Jenkins post-build cleanup is not able to delete these files.



Phabricator

- Contains a link to Jenkins build
- Indicates build status
- Reports test failures
- Reports test coverage

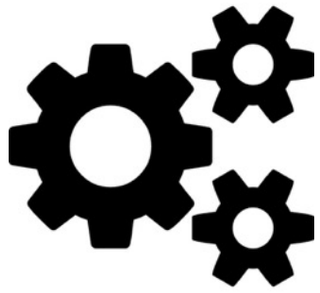
Phabricator

DIFF DETAIL

Diff 2101516	
Repository	rMCIEI mc/iei
Lint	★ Lint OK
Unit	★ No Unit Test Coverage
Branch	sirius_client
Build Status	✓ **Buildable 2844884**
	✓ Build 4468254: ci-test-iei-docker-build Jenkins
	✓ Build 4468253: Juno code-diff-validation Jenkins
	✓ Build 4468250: arc lint + arc unit

build status

link to build



Phabricator

test result

test output

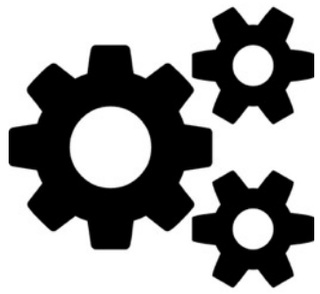
Unit Tests ✖ Failed

View All

	Time	Test

✖	7 ms	Jenkins > com.uber.maps.opus.publishing.staging.EvidenceIngestTest::initializationError
		`java.lang.TypeNotPresentException: Type [unknown] not present`
		`at sun.reflect.annotation.TypeNotPresentExceptionProxy.generateException(TypeNotPresentExceptionProxy.java:46)`
		`at sun.reflect.annotation.AnnotationInvocationHandler.invoke(AnnotationInvocationHandler.java:84)`

✔	0 ms	Coverage Data
✔	10 ms	Jenkins > com.uber.maps.opus.publishing.IndexedAvroConverter.CommandLineParserTest::indexedAvroConverterParseCommandLineArguments
✔	1 ms	Jenkins > com.uber.maps.opus.publishing.IndexedAvroConverter.CommandLineParserTest::indexedAvroConverterPaths
✔	0 ms	Jenkins > com.uber.maps.opus.publishing.IndexedAvroConverter.GeneralReadRecordsTest::indexedAvroConverterReadRecords
...		View Full Test Results (1 Failed · 12 Passed)



Phabricator

test coverage

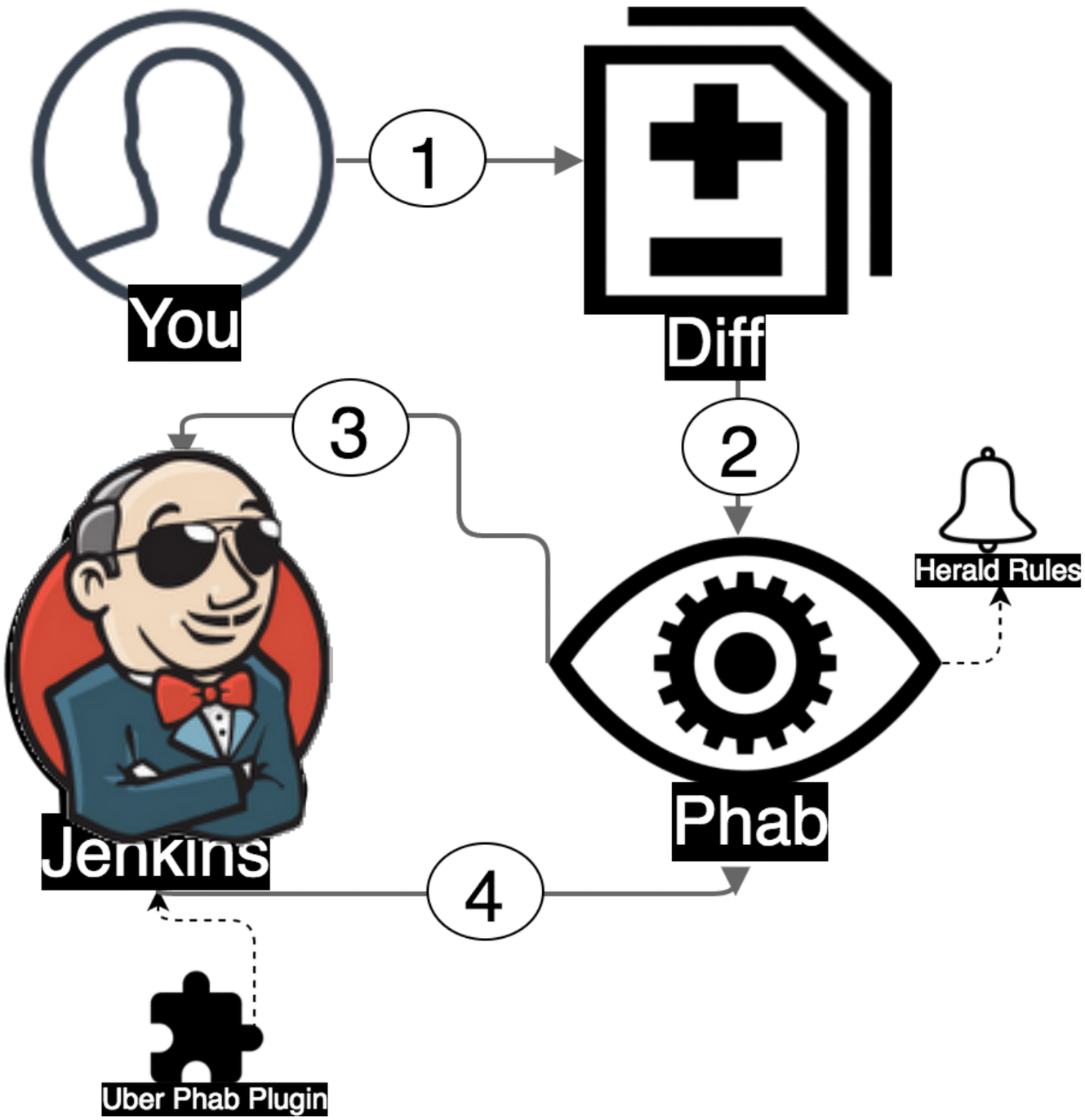
jenkins added a comment. Wed, Nov 2, 2:14 PM

Coverage remained the same (19.778%) when pulling **sirius_client** into `6416580` . https://ci.uberinternal.com/job/test-iei-docker-build/955/ for more details.

Code Coverage Gotchas

- Code that isn’t imported in tests isn’t included in coverage metrics
  - It’s possible to add tests and have coverage do *down*
- Coverage != effective tests
  - It’s possible to have 100% coverage with bad tests
- Coverage is more an indicator of *untested* than tested code

Putting It All Together



Takeaways

- Allow Jenkins to build before landing your diff
- Don’t land diffs that have build failures/instability
- When in doubt, check the Jenkins console output
- Unit tests are good, more of those!