BEST PRACTICES

TECH & TOOLING - LINTING, SEMANTIC ANALYSIS, CI/CD, DEVELOPER ENVIRONMENTS

CSC491 | UTORONTO

IN THIS LECTURE

1. We will not go in depth on any topic as there are many many options and that would not be feasible

2. I intend to introduce you to the concepts and provide some examples

3. You should leave here understanding a breadth of options and understand new concepts

4. I will help you evaluate and pick options if required for your projects

5. I can answer questions based on my own experiences and opinions, but remember these are not the "correct" solution

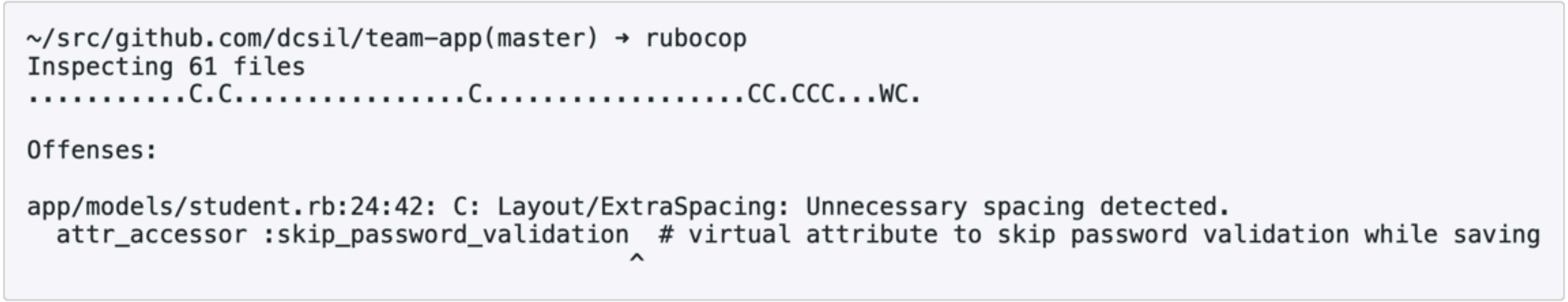
IN THIS LECTURE

1. What is linting? How do I use it?
2. What is semantic analysis?
3. What is CI/CD and how does it work?
4. A look into developer environments

LINTING

LINT, OR A LINTER, IS A TOOL THAT ANALYZES SOURCE CODE TO FLAG PROGRAMMING ERRORS, BUGS, STYLISTIC ERRORS, AND SUSPICIOUS CONSTRUCTS.

THERE ARE SO MANY OPINIONS ON THIS TOPIC THAT NONE OF THEM ARE RIGHT. THIS LECTURE WILL COVER SOME PRO-TIPS ON HOW TO NAVIGATE THIS FIELD



```
~/src/github.com/dcsil/team-app(master) → rubocop
Inspecting 61 files
...........C.C................C..................CC.CCC...WC.

Offenses:

app/models/student.rb:24:42: C: Layout/ExtraSpacing: Unnecessary spacing detected.
  attr_accessor :skip_password_validation  # virtual attribute to skip password validation while saving
                                         ^
```

LINTING

- Saves time on the boring stuff (who wants to debate how many spaces to put after a {?)
- Consistent Code. Code is your UI. This makes your devs faster because the code is easier to read
- Can catch bugs

LINTING

- Can also help a developer do the "right" thing!
- Linting can ensure proper use of methods in OSS repos, for example.

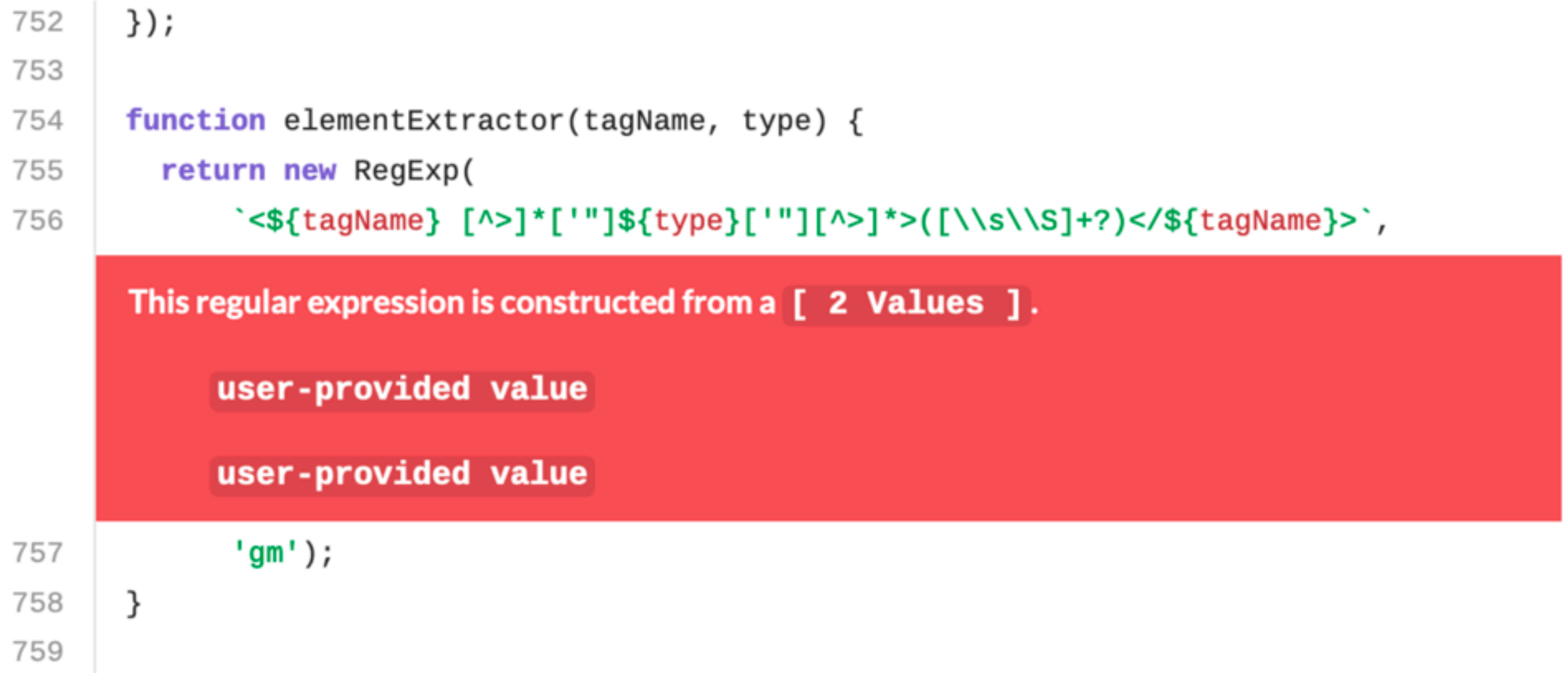
SEMANTIC ANALYSIS

SEMANTIC ANALYSIS OR CONTEXT SENSITIVE ANALYSIS IS A PROCESS IN COMPILER CONSTRUCTION, USUALLY AFTER PARSING, TO GATHER NECESSARY SEMANTIC INFORMATION FROM THE SOURCE CODE.

INCLUDES SUCH THINGS AS TYPE CHECKING TOO.

SEMANTIC ANALYSIS

- Catch bugs
- Catch security issues



```
});

function elementExtractor(tagName, type) {
  return new RegExp(
      `<${tagName} [^>]*['"]${type}['"][^>]*>([\\s\\S]+?)</${tagName}>`,
```

This regular expression is constructed from a `[ 2 Values ]`.

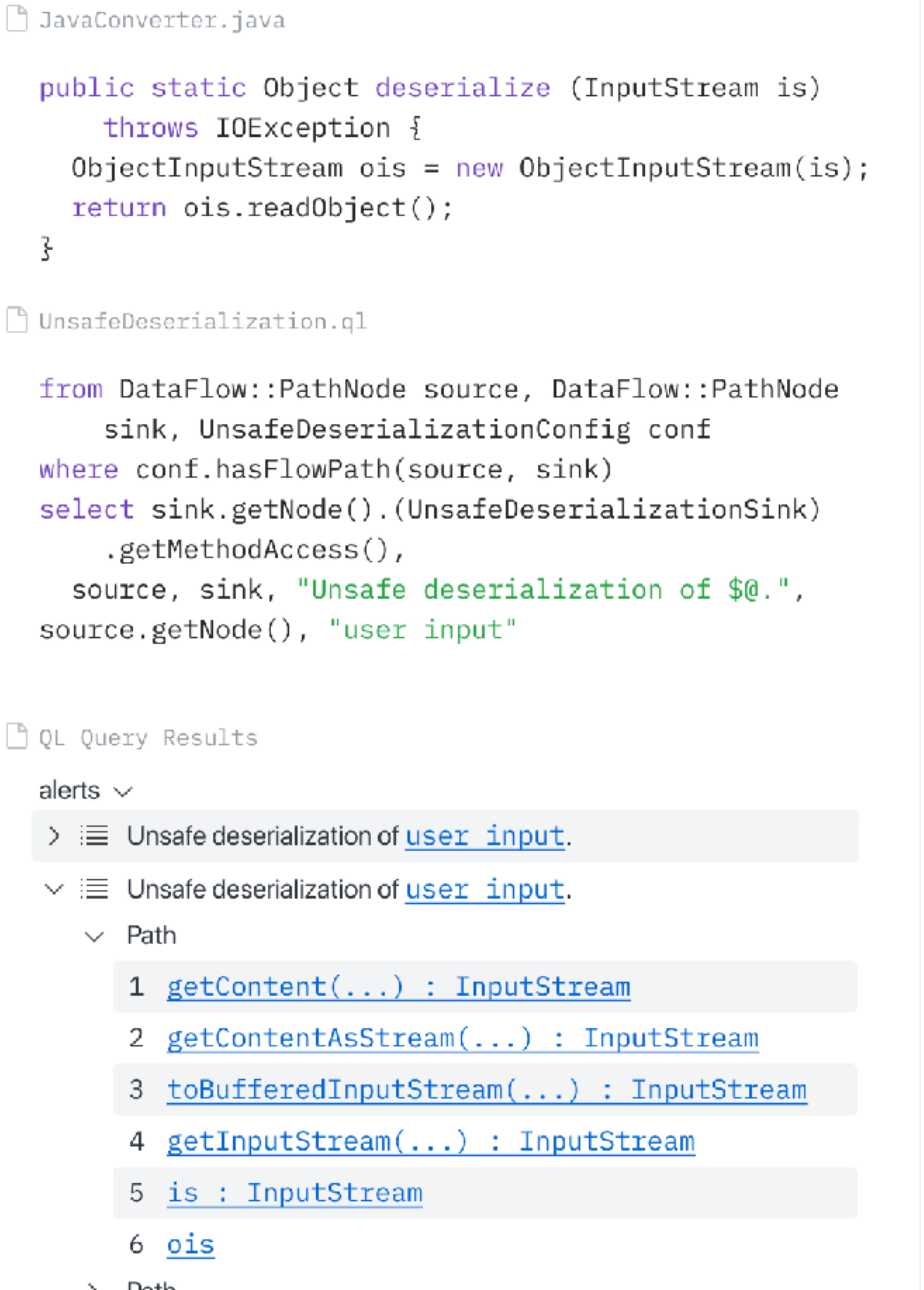
`user-provided value`

`user-provided value`

```
      'gm');
}
```

SEMANTIC ANALYSIS

- Can dive across layers and layers of code to find weird paths developers are not going to see
- Can be adapted to many languages and detect errors like deserialization, unsafe usage of user params, etc



JavaConverter.java

```
public static Object deserialize (InputStream is)
    throws IOException {
  ObjectInputStream ois = new ObjectInputStream(is);
  return ois.readObject();
}
```

UnsafeDeserialization.ql

```
from DataFlow::PathNode source, DataFlow::PathNode
    sink, UnsafeDeserializationConfig conf
where conf.hasFlowPath(source, sink)
select sink.getNode().(UnsafeDeserializationSink)
    .getMethodAccess(),
  source, sink, "Unsafe deserialization of $@.",
source.getNode(), "user input"
```

QL Query Results

alerts

- Unsafe deserialization of user input.
- Unsafe deserialization of user input.
  - Path
    1 getContent(...) : InputStream
    2 getContentAsStream(...) : InputStream
    3 toBufferedInputStream(...) : InputStream
    4 getInputStream(...) : InputStream
    5 is : InputStream
    6 ois
  - Path

CONTINUOUS INTEGRATION (CI)

CONTINUOUS INTEGRATION

Overview of how it works

- When code is pushed to a remote repository, a system picks up the changes, clones them, runs the test suite, and reports the result to the code.

PARTS OF A CI SYSTEM

The basic parts of a CI system are:

<u>1. Event System</u>

Receives events from some remote source indicating a code change

<u>2. Scheduling system</u>

Schedules the job to be run

<u>3. Coordinator</u>

Coordinates workers to run a job. Sometimes will check out the code and create a cached setup for the workers.

PARTS OF A CI SYSTEM

The basic parts of a CI system are:

4. Workers

Checks out the code if not done in (3) and sets it up. Runs the test suite (or part of it)

5. Reporter

Aggregates any results and reports back to the source of the code change

PARTS OF A CI SYSTEM

- **We also need a UI to see the CI running and stream logs in real time**

- **We can sometimes need a test flakiness* detection system**

* test flakiness occurs when a test, that should pass, fails some of the time for unrelated reasons. This could be due to performance issues of the system, time based issues, or something else unknown.

PARTS OF A CI SYSTEM

Setup Time

- As your test suite grows so does the time required to run it.
- To combat this people often split their tests between multiple workers.
- Each of those workers has some fixed set up time.

PARTS OF A CI SYSTEM

Setup Time

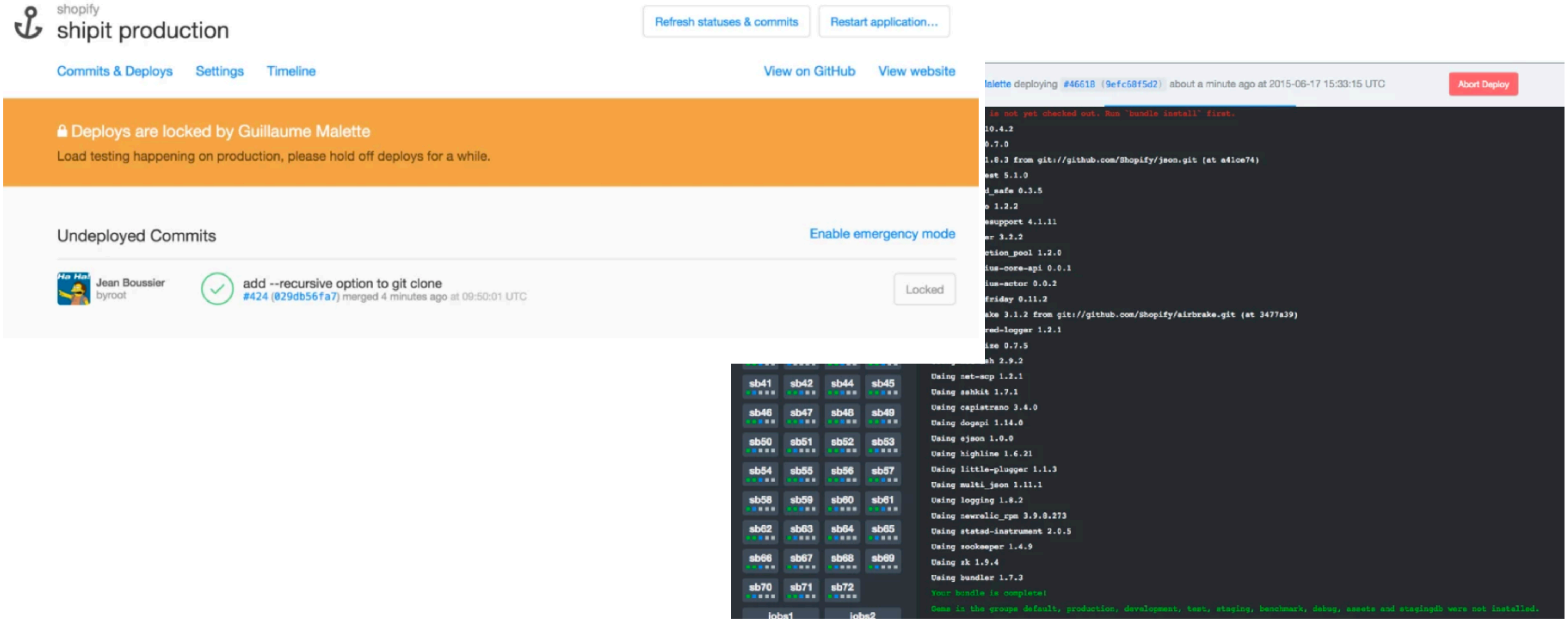
- **E.g. If you have 20 workers that run 30 minute of tests each but each takes 3.5 minutes to setup, then it still takes 4 minutes. It would be just as cost effective and almost the same time to halve the workers in this case.**
- **Work on reducing set up time by prebuilding an image, sharing a cache between builds (of packages for example), or having one setup done in a coordinator**

CONTINUOUS DEPLOYMENT (CD)

CONTINUOUS DEPLOYMENT

- Fast, debuggable deploy
- Somewhere everyone can see, so many can debug if problems
- Coordinated deploys are easier

EXAMPLE OF A SYSTEM



shopify
shipit production

Commits & Deploys   Settings   Timeline

View on GitHub   View website

Refresh statuses & commits   Restart application...

🔒 Deploys are locked by Guillaume Malette

Load testing happening on production, please hold off deploys for a while.

Undeployed Commits

Enable emergency mode

Jean Boussier
byroot

✓ add --recursive option to git clone
#424 (029db56fa7) merged 4 minutes ago at 09:50:01 UTC

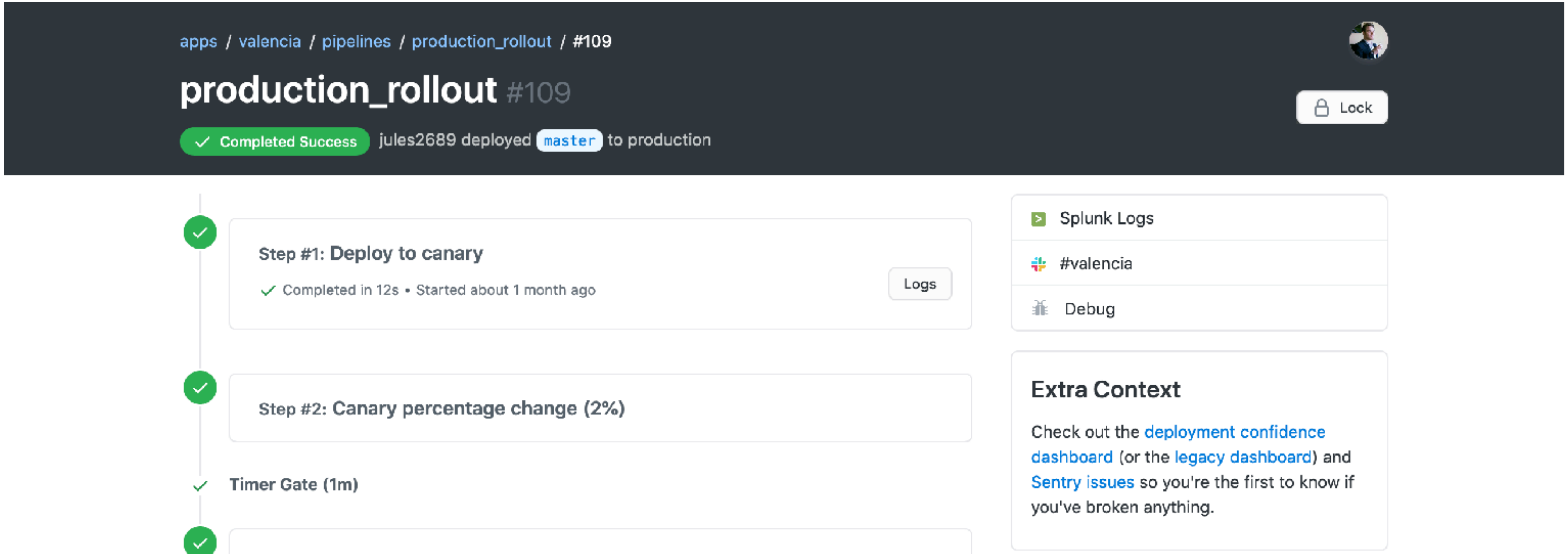
Locked

lalette deploying #46618 (9efc68f5d2) about a minute ago at 2015-06-17 15:33:15 UTC

Abort Deploy

```
 is not yet checked out. Run `bundle install` first.
10.4.2
0.7.0
1.8.3 from git://github.com/Shopify/json.git (at a41ce74)
est 5.1.0
d_safe 0.3.5
o 1.2.2
esupport 4.1.11
er 3.2.2
ction_pool 1.2.0
ius-core-api 0.0.1
ius-actor 0.0.2
friday 0.11.2
ake 3.1.2 from git://github.com/Shopify/airbrake.git (at 3477a39)
red-logger 1.2.1
ize 0.7.5
sh 2.9.2
Using net-scp 1.2.1
Using sshkit 1.7.1
Using capistrano 3.4.0
Using dogapi 1.14.0
Using ejson 1.0.0
Using highline 1.6.21
Using little-plugger 1.1.3
Using multi_json 1.11.1
Using logging 1.8.2
Using newrelic_rpm 3.9.8.273
Using statsd-instrument 2.0.5
Using zookeeper 1.4.9
Using zk 1.9.4
Using bundler 1.7.3
Your bundle is complete!
Gems in the groups default, production, development, test, staging, benchmark, debug, assets and stagingdb were not installed.
```

EXAMPLE OF A SYSTEM



apps / valencia / pipelines / production_rollout / #109

production_rollout #109

✓ Completed Success jules2689 deployed `master` to production

Step #1: Deploy to canary

✓ Completed in 12s • Started about 1 month ago

Step #2: Canary percentage change (2%)

✓ Timer Gate (1m)

Extra Context

Check out the deployment confidence dashboard (or the legacy dashboard) and Sentry issues so you're the first to know if you've broken anything.

DEVELOPER ENVIRONMENTS

DEVELOPER ENVIRONMENTS

- How will you run your app locally?
- How will the next person?
- What is the experience like running your app locally?

DEVELOPER ENVIRONMENTS

- Running your app locally often starts as a set of instructions in a doc somewhere
- Eventually it makes its way into an automated script when other people start wanting to run your software
- Many people eventually make Docker containers out of it
- Then they realize they need 10 Docker containers to run stuff locally and need to network into the Docker container to change files

DEVELOPER ENVIRONMENTS

- Not an easy task. Just as hard as production.
- Docker-Compose is a good start, may need to write code outside of the container and sync in?
- In Cloud options becoming more viable

DEVELOPER ENVIRONMENTS

- The local developer experience should be something you don't take for granted.
- While saving a few minutes of time on a script or task might seem meaningless, imagine the time saved when you have to do that task 100 times. Imagine the time saved when 100 people have to do the task 100 times.

CREDITS

- CONTENT
  - Types of Software Testing
  - Visual Regression Testing
  - Locust.io
  - Lint (Software)
  - LGTM from Semmle Example
  - Redis vs Memcached
  - GitHub report on top languages
  - Developer Productivity