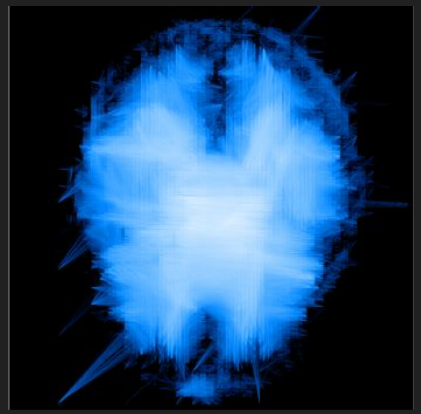
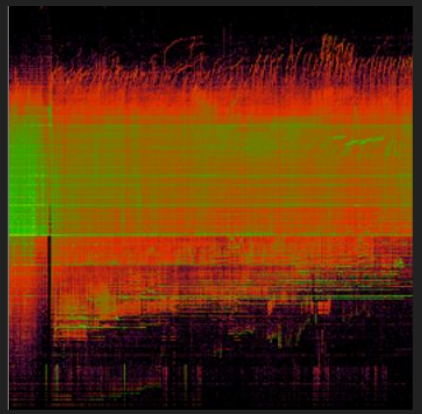
Continuous Integration for Spark Apps

Hi, I’m Sean!





uncharted

It’s hard to test Spark Apps :(

Case Study: Uncharted Spark Pipeline

```
import software.uncharted.sparkpipe.Pipe
import software.uncharted.sparkpipe.ops

// Data injest pipe
@transient val injest = Pipe(sqlContext)
.to(ops.core.dataframe.io.read(path = "hdfs:///data/twitter/trump/*/*", format = "json", schema = TRUMP_SCHEMA))
.to(_.selectExpr(
  "id_str", "user.id", "user.name", "timestamp_ms", "text",
  "retweeted_status.retweet_count as orig_retweet_count",
  "retweeted_status.favorite_count as orig_favorite_count",
))
.to(ops.core.dataframe.castColumns(Map(
  "timestamp_ms" -> "double",
  "orig_retweet_count" -> "double",
  "orig_favorite_count" -> "double"
)))
.to(ops.core.dataframe.cache)
```

Case Study: Uncharted Spark Pipeline

Some key issues:

- Ensure reliability
- Prevent regressions
- Maintain compatibility with multiple versions of Spark
- Open-source - need a quick and easy way to evaluate PRs

What is Continuous Integration?

“Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.”

-- ThoughtWorks

“Continuous Integration (CI) is a development practice that is pretty damned important for writing quality software.”

-- Me

So, What is Continuous Integration?

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches
6. Test in a clone of the production environment

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

} duh.

Best Practices, courtesy of Wikipedia

1. Maintain a code repository (Git)
2. Automate the build (Gradle)
3. Tests should be part of the build (ScalaTest)
4. Commit/push feature branches often
5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

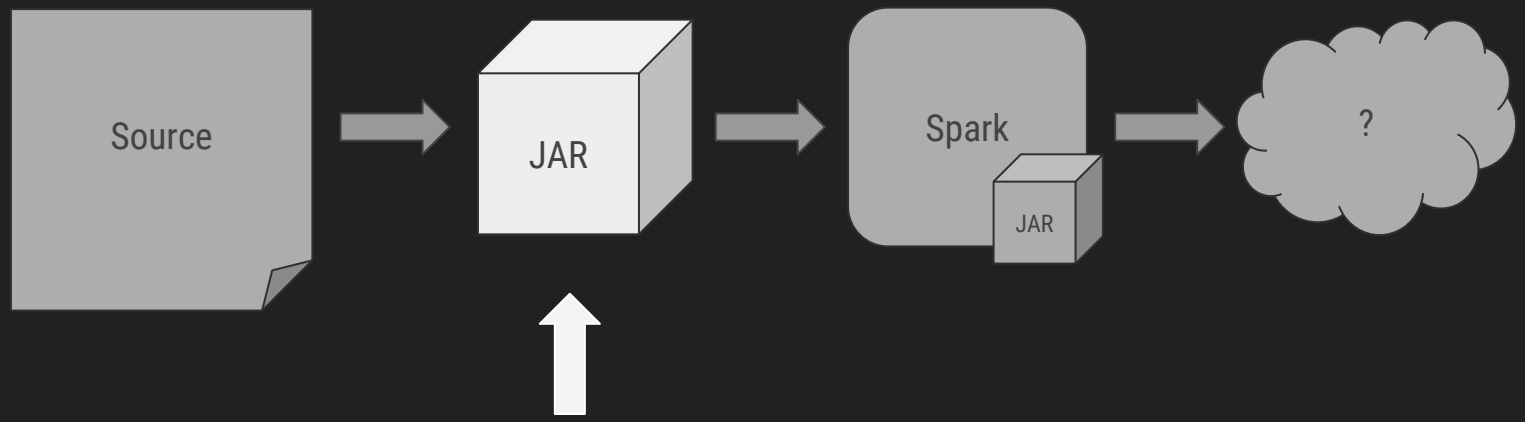
} ...less duh.

Why are these difficult with Apache Spark?

5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

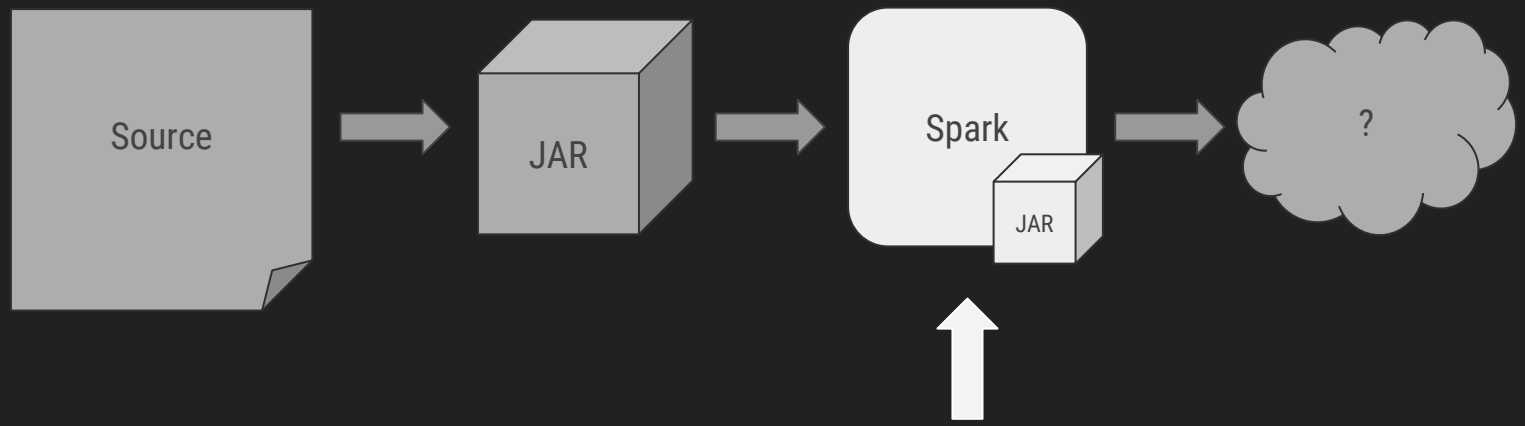
What is a Spark App?

What is a Spark app?



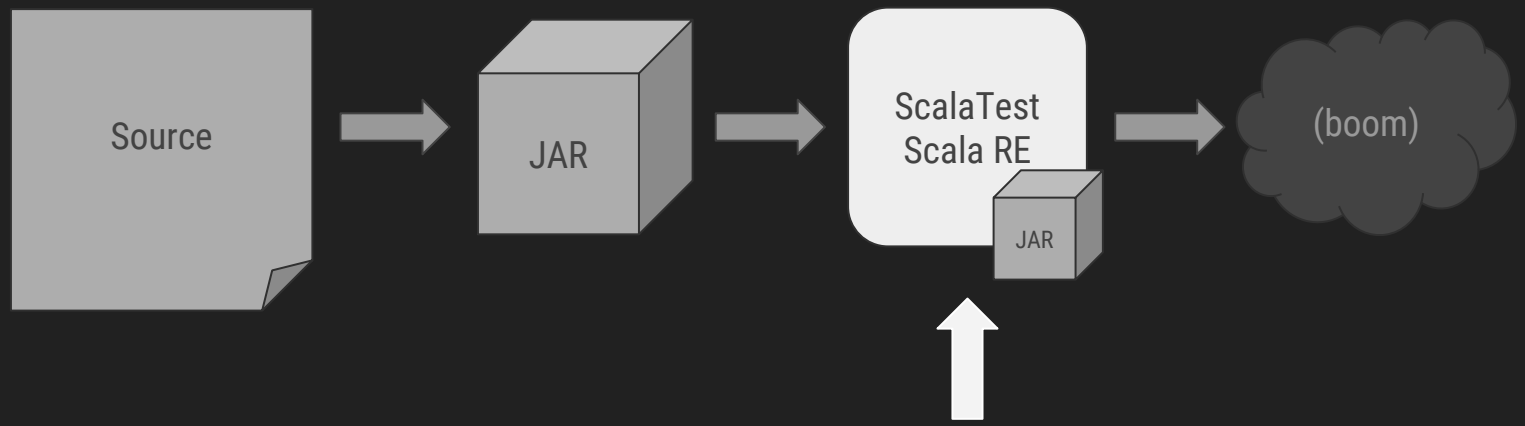
This thing.

And...



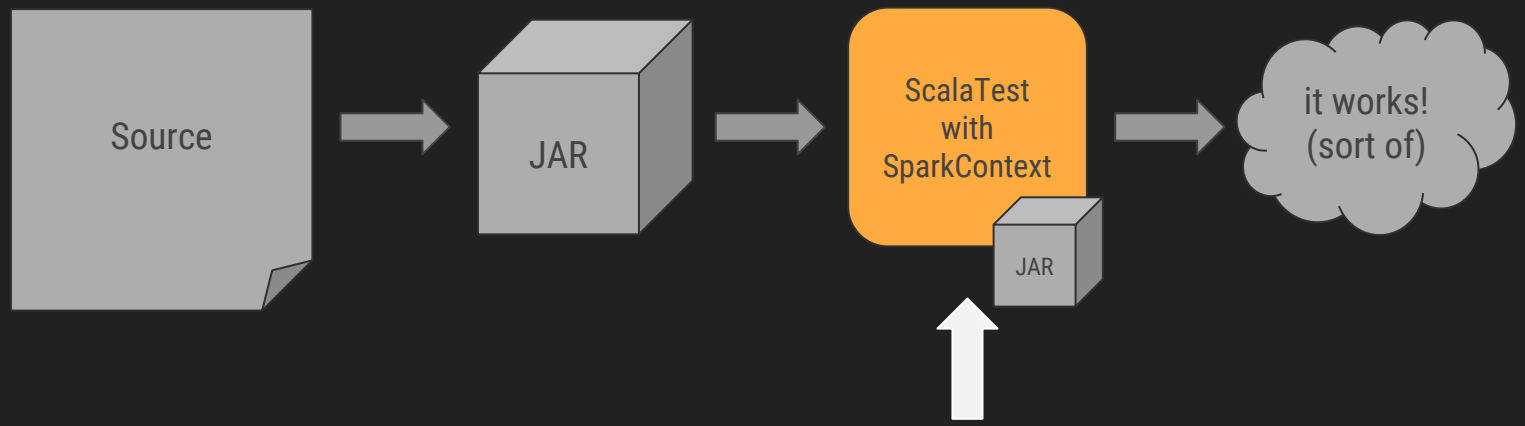
We need to test this

But...



By default, we have this

v1: Squish Spark inside ScalaTest

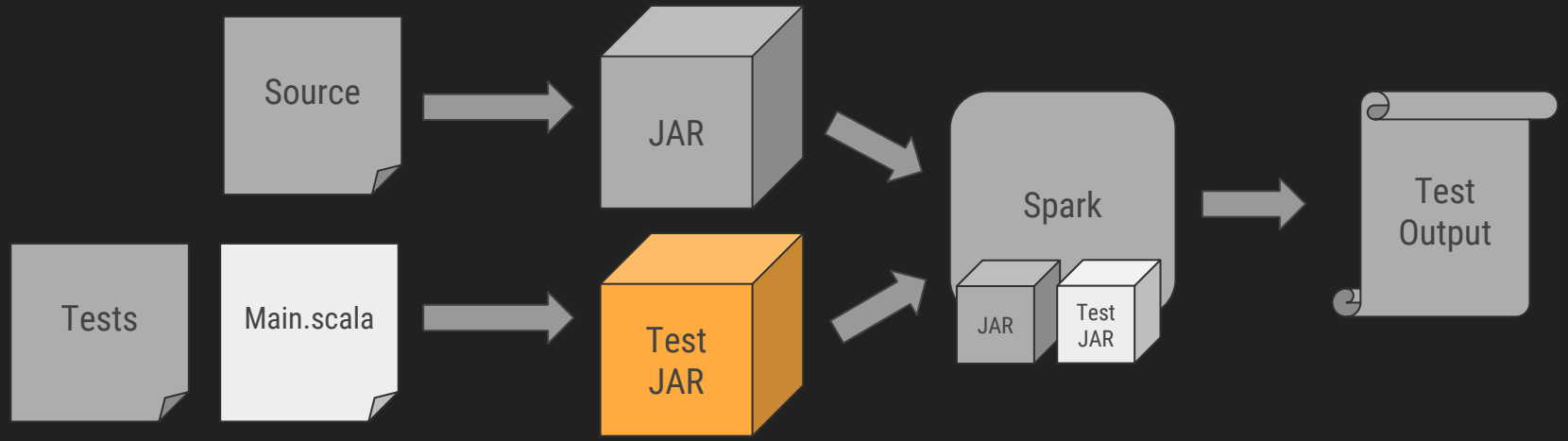


So, we try this

it works!
(sort of)

6. Test in a clone of the production environment

v2: Squish ScalaTest into Spark



Main.scala

```scala
import org.scalatest.tools.Runner

object Main {
  def main(args: Array[String]): Unit = {
    val testResult = Runner.run(Array("-o", "-R", "build/classes/test"))
    if (!testResult) {
      System.exit(1) // exit with an error code if a test failed
    }
  }
}
```

6. Test in a clone of the production environment

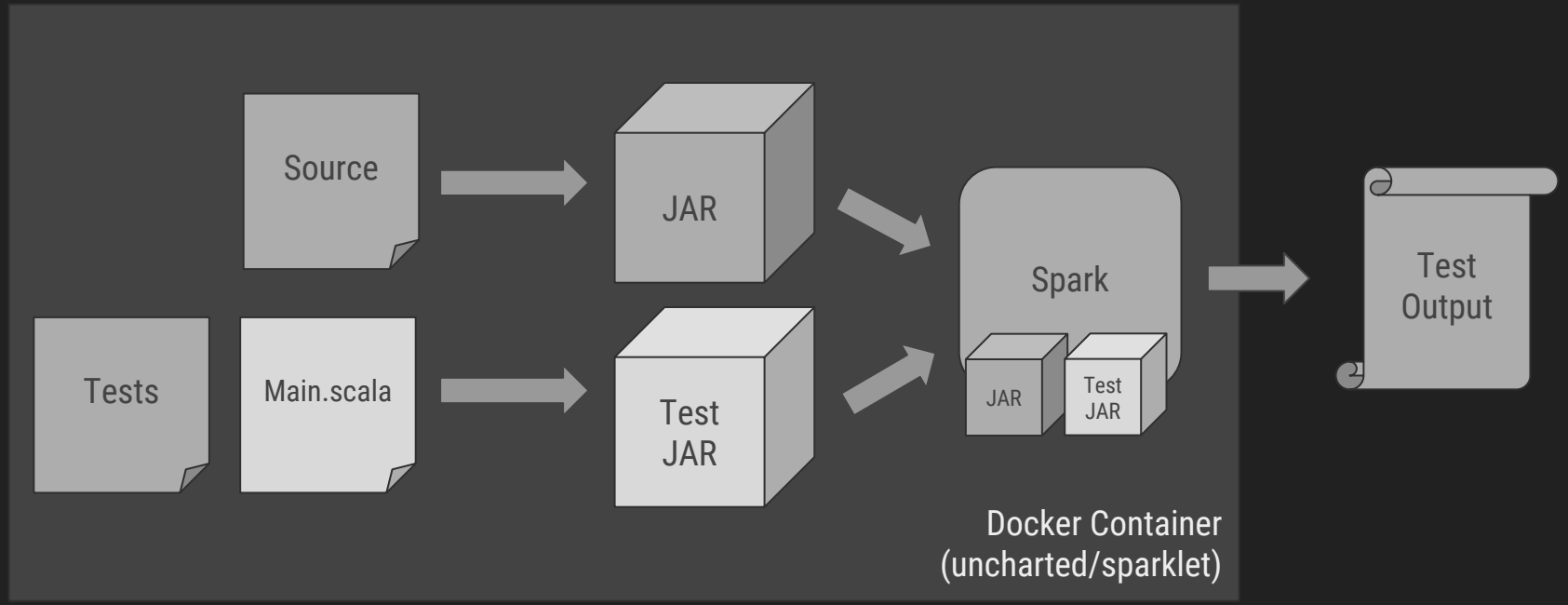
Progress?

5. Build (and test) All The Branches
6. ✔ Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

What now?

5. Build (and test) All The Branches
6. ✔ Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

v3: Squish Spark and Test JAR into Docker



test.sh

```
#!/bin/sh

docker run \
# forward ports for Spark UI and debugging to local machine
-p 8080:8080 \
-p 9999:9999 \
# this log4j.properties file silences some of the verbose output from spark
-v /$(pwd)/src/test/resources/log4j.properties:/usr/local/spark/conf/log4j.properties \
# mount our code as a shared volume within the container
-v /$(pwd):/opt/mysrc \
# set working directory where we'll have the container prompt start
--workdir="//opt/mysrc" \
# which version of Spark do we want to test against?
uncharted/sparklet:$SPARK_VERSION \
# run gradle test
./gradlew test
```

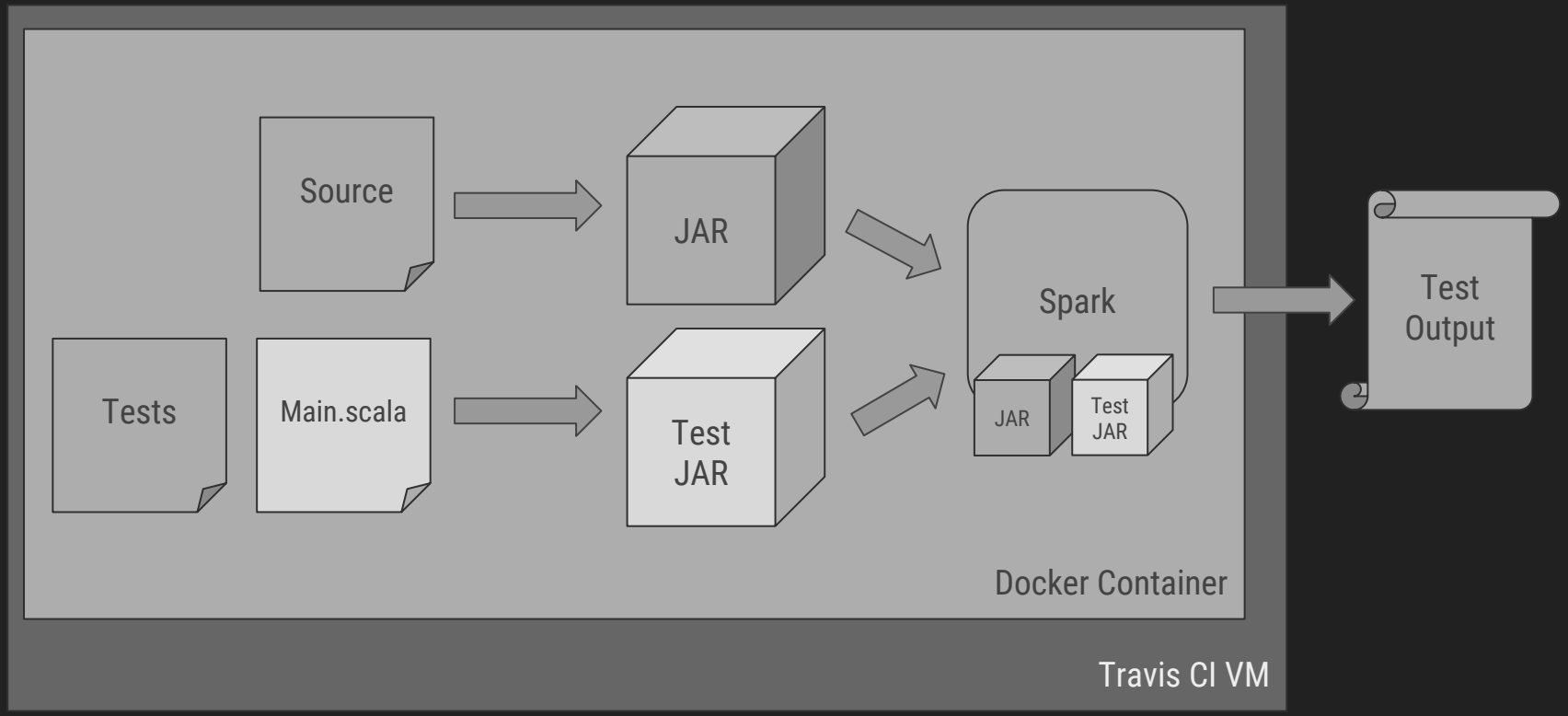
build.gradle (excerpt)

```
// override default test task to run spark-submit instead
task test(overwrite: true, type: Exec, dependsOn: [jar, testJar, scalaStyle]) {
  executable = 'spark-submit'
  args = [
    // the --packages flag allows us to place the scalatest jar on the classpath
    "--packages","org.scalatest:scalatest_${scalaBinaryVersion}:2.2.5",
    // the --jars flag lets us do the same thing with our code
    "--jars","/opt/src/build/libs/${artifactName}-${version}.jar",
    // Main.scala, as shown before, will kick off our tests
    "--class","com.mycompany.project.tests.Main",
    // submit the test jar as the application
    "build/libs/${artifactName}-${version}-tests.jar"
  ]
}
```

Progress?

5. Build (and test) All The Branches
6. ✓ Test in a clone of the production environment
7. ✓ Keep the build fast
8. Everyone can see the results of builds

v4: Squish Docker into Travis CI



.travis.yml

```yaml
sudo: required
language: bash
services:
  - docker
env:
  # build against multiple versions of spark!
  - SPARK_VERSION=1.4.1
  - SPARK_VERSION=1.5.2
  - SPARK_VERSION=1.6.0
before_script:
  # generate environment file so we can give
  # the travis environment to our test container
  - env | grep TRAVIS_ > travis.env
  - echo "CI_NAME=travis_ci" >> travis.env
  - echo "CI=true" >> travis.env
  - echo "TRAVIS=true" >> travis.env
  - echo "CONTINUOUS_INTEGRATION=true" >> travis.env
  - echo "DEBIAN_FRONTEND=noninteractive" >> travis.env
  - echo "HAS_JOSH_K_SEAL_OF_APPROVAL" >> travis.env
  - echo $SPARK_VERSION >> travis.env
```

```yaml
# this is more or less a one-line version of test.sh
script:
  # run test container
  - docker run --env-file travis.env -v $(pwd)
/src/test/resources/log4j.properties:
/usr/local/spark/conf/log4j.properties -v $(pwd):/opt/mysrc
--rm --workdir="/opt/mysrc"
uncharted/sparklet:$SPARK_VERSION ./gradlew test
```

Voilà!

build passing

Progress?

5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

Builds - unchartedsof

https://travis-ci.org/unchartedsoftware/sparkpipe-core/builds

Travis CI Blog Status Help Sean McIntyre SM

unchartedsoftware / sparkpipe-core

build:

Current Branches Build History Pull Requests Settings

✓ 0.9.5	Bumping version number for release. Bumping	#55 passed	8 min 46 sec
SM Sean McIntyre		1fb943e	11 days ago
✓ master	Bumping version number for release. Bumping	#54 passed	9 min 1 sec
SM Sean McIntyre		1fb943e	11 days ago
✓ master	Merge pull request #18 from unchartedsoftwar	#53 passed	8 min 24 sec
SM Sean McIntyre		c8b0b66	11 days ago
✓ feature/17	Added stopTermFilter and includeTermFilter in	#51 passed	9 min 5 sec
SM Sean McIntyre		612642d	11 days ago
✓ 0.9.4	Bumping version number for release	#50 passed	8 min 21 sec
SM Sean McIntyre		c414d92	23 days ago
✓ master	Bumping version number for release	#49 passed	8 min 46 sec
SM Sean McIntyre		c414d92	23 days ago
✓ master	Merge pull request #16 from unchartedsoftwar	#48 passed	8 min 23 sec

unchartedsoftware/s

https://travis-ci.org/unchartedsoftware/sparkpipe-core

Travis CI Blog Status Help Sean McIntyre SM

unchartedsoftware / sparkpipe-core

build:passed

Current Branches Build History Pull Requests Settings

✓ 0.9.5 Bumping version number for release. Bumping scala dependency to 2.10.6 (final 2.10.x maintenance release).

Commit 1fb943e

Compare 0.9.5

SM Sean McIntyre authored and committed

#55 passed

Elapsed time 2 min 57 sec

Total time 8 min 46 sec

11 days ago

Build Jobs

✓	# 55.1	no language set	SPARK_VERSION=1.4.1	2 min 57 sec
✓	# 55.2	no language set	SPARK_VERSION=1.5.2	2 min 54 sec
✓	# 55.3	no language set	SPARK_VERSION=1.6.0	2 min 55 sec

Search all repositories

All done!

5. Build (and test) All The Branches
6. Test in a clone of the production environment
7. Keep the build fast
8. Everyone can see the results of builds

Next Steps?

Alpine Linux

docker-compose

Windows (dev environment) support

python

https://github.com/unchartedsoftware/sparkpipe-core

https://hub.docker.com/r/uncharted/sparklet/

Questions?

smcintyre@unchartedsoftware.com

https://github.com/Ghnuberath

@Ghnuberath

uncharted