BEEFING UP RELEASE PIPELINES

Imad Dodin - 260713381

Fouad El Bitar - 260719196

Jules Boulay de Touchet - 260710129

Ege Odaci - 260722818

Irmak Pakis - 260733837

Baris Utku Cincik - 260730586

What is the Skrape-it?

{Skrape.it}

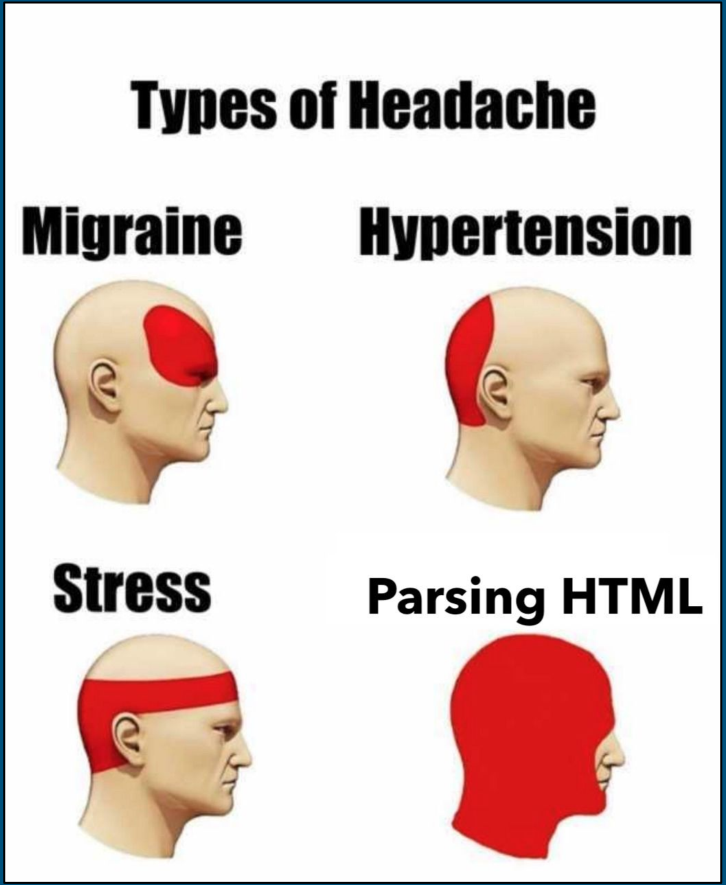
Kotlin based HTML/XML Testing and Web Scraping Library tool

- Helps you generate HTTP requests
- Provides easy to use interface
- Target specific elements with HTTP requests

What is the Skrape-it?

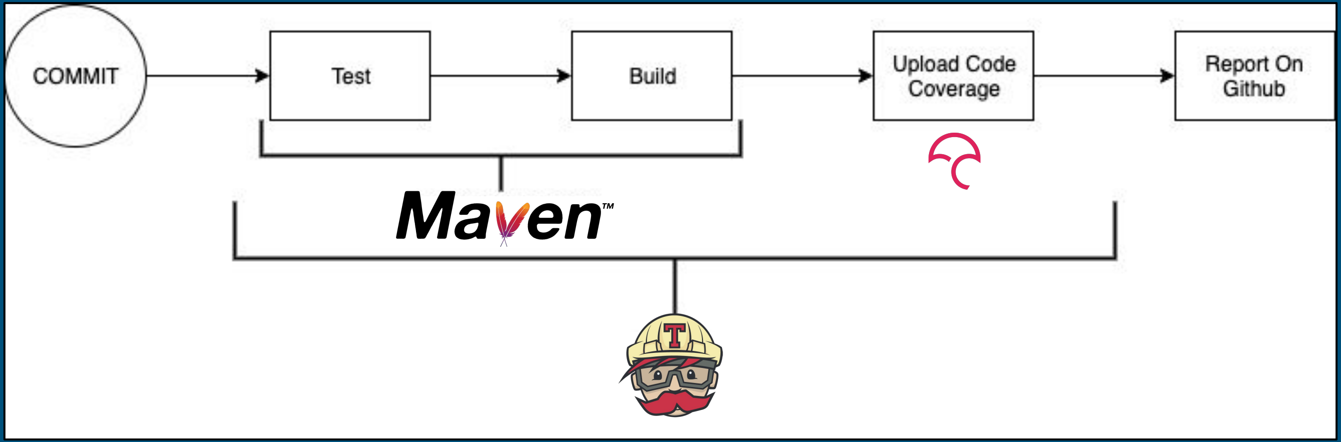
Parsing HTML is tedious!

- Analyze and extract HTML data
- Offers Domain Specific Language
- Pick elements from the HTML using the domain specific language



How does the system work?

Before



How does the system work?

Before

Workflow:

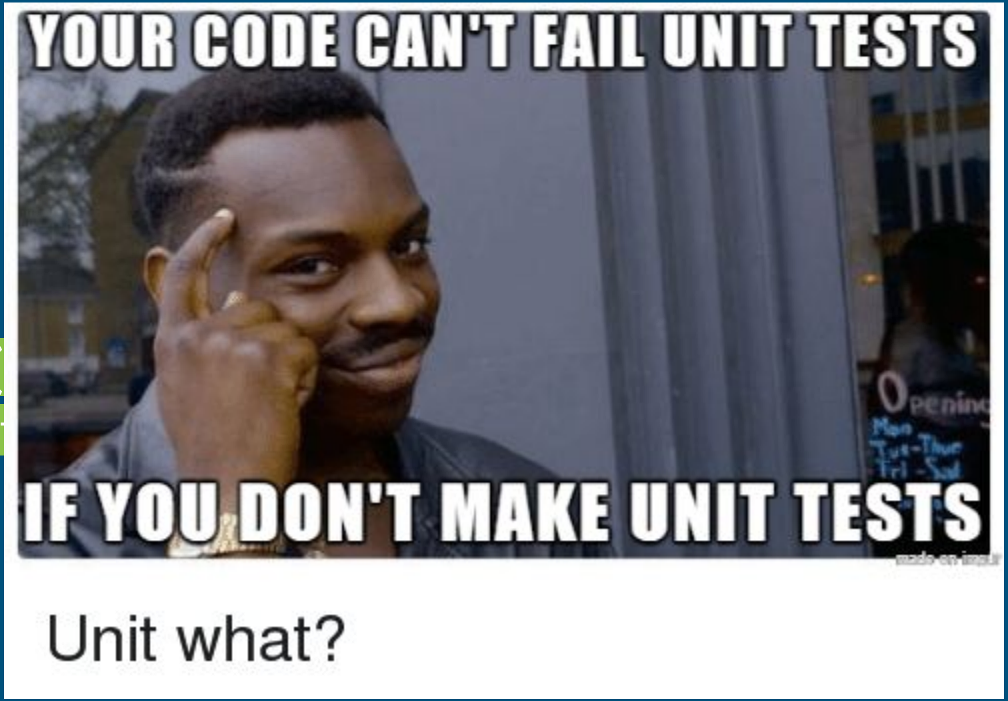
- Forking workflow to allow open source contribution
  - Protect master, each contributor maintains their own repository, merges have to be checked by owner/administrator
- Contributors keep up to date by pulling from main repo

How does the system work?

Before

Tools used:

- **Codecov (**
  that the tes



How does the system work?

Before

Tools use

- **Code** that tl e
- **Detel** detec

shirtwoot!
BUG HUNT
IT'S PROBABLY A SEMI-COLON
Not-working
You made 1 little change and now it's
Not-working
Welcome to the next 15 hours of your life
YOU MIGHT WANT TO USE THIS

detekt

Static code analysis for Kotlin

How does the system work?

Before

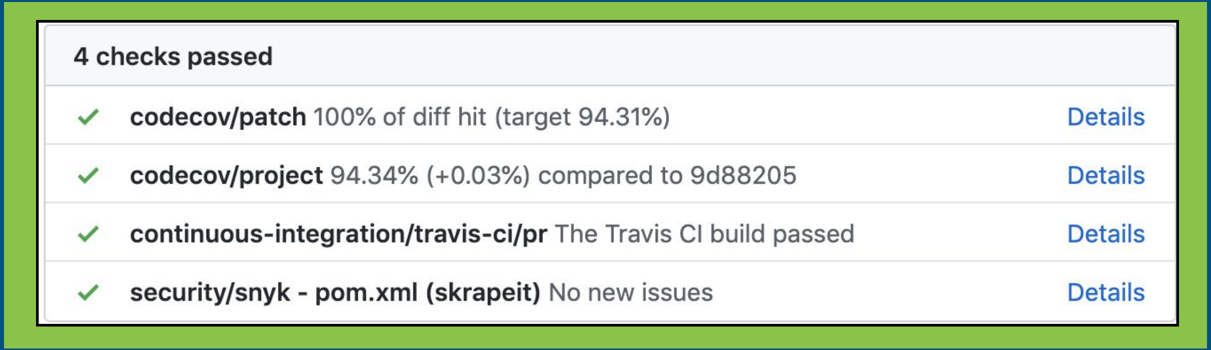
Tools used:

- **Codecov (code coverage tool)** - assesses the lines of code that the test suite covers
- **Detekt (static analysis tool) -** added to maven as plugin, detects code antipatterns and potential areas for bugs
- **Snyk (Vulnerability and dependency management tool) -** analyzes the dependencies of the project and checks for vulnerabilities

How does the system work?

Before

These tools are integrated in the pipeline so that when a change is made and build is executed, it is displayed on github...


4 checks passed

✓ **codecov/patch** 100% of diff hit (target 94.31%) Details

✓ **codecov/project** 94.34% (+0.03%) compared to 9d88205 Details

✓ **continuous-integration/travis-ci/pr** The Travis CI build passed Details

✓ **security/snyk - pom.xml (skrapeit)** No new issues Details

detekt

Static code analysis for Kotlin

snyk

What solutions did you try and why?

1. **Caching**
   a. **Speed up Continuous Integration time**
2. **Migrating from Maven → Gradle**
   a. **Speed up build times**
   b. **Offer more flexibility**
3. **Migrating from Travis CI → Github Actions**
   a. **Speed up pipeline execution**
   b. **Provide workflows for future customization**

Why not Travis?

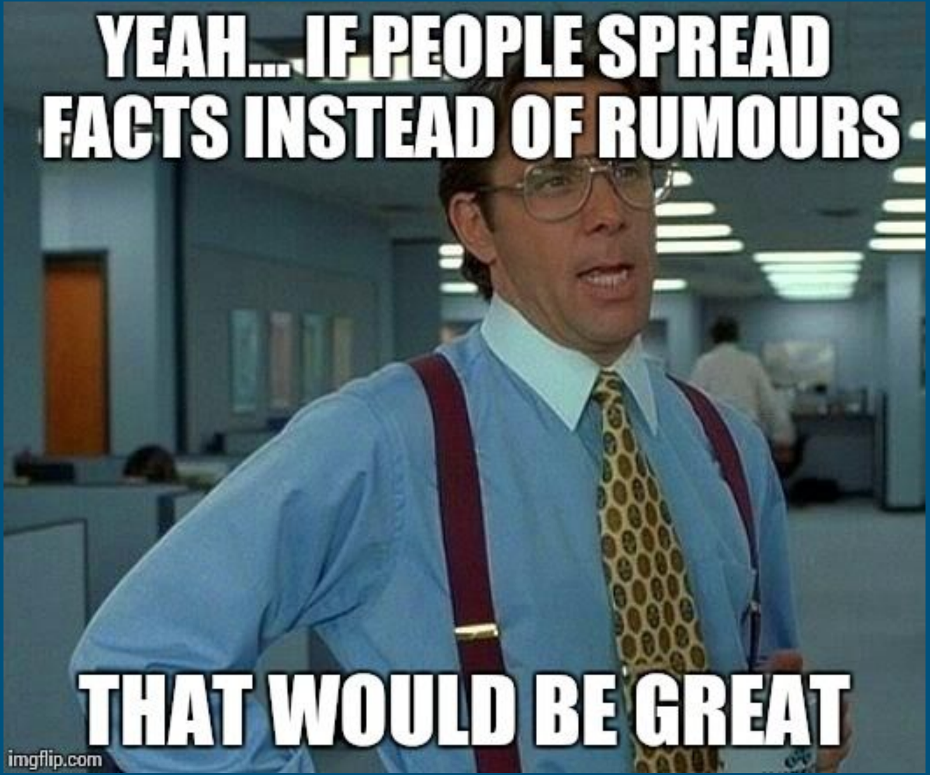
Why not Travis?

1. **Issues with Integration system Travis CI**

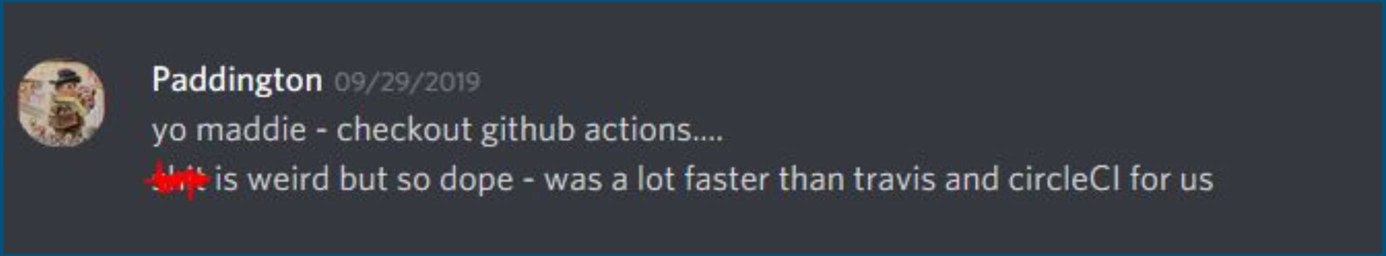
- No option to discriminate between Pull Requests and Commits
    - I know… I know… we could use environment variables….

Why GitHub Actions?

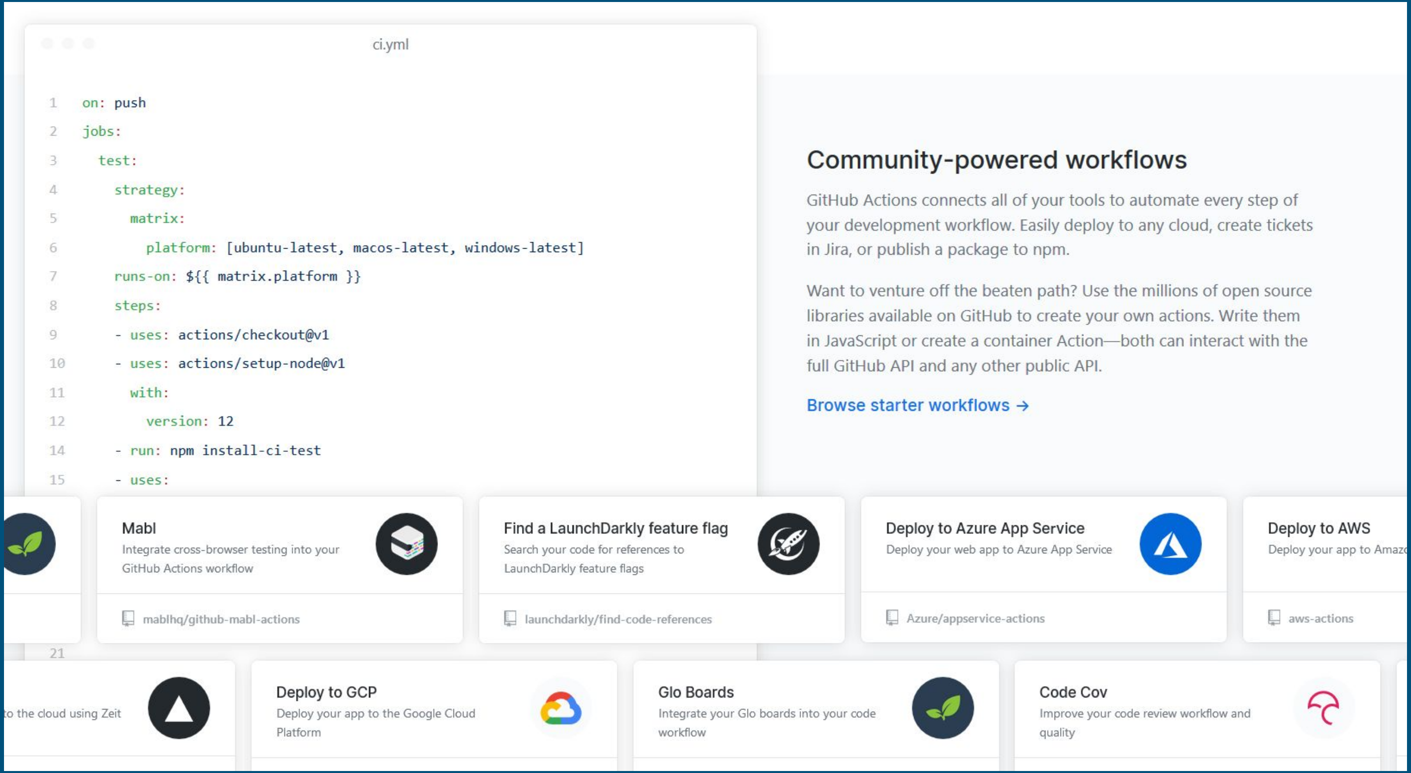
Faster CI!


YEAH... IF PEOPLE SPREAD FACTS INSTEAD OF RUMOURS
THAT WOULD BE GREAT
imgflip.com

Faster CI!


Paddington 09/29/2019
yo maddie - checkout github actions....
[redacted] is weird but so dope - was a lot faster than travis and circleCI for us

Modular Pipelines



ci.yml

```
on: push
jobs:
  test:
    strategy:
      matrix:
        platform: [ubuntu-latest, macos-latest, windows-latest]
    runs-on: ${{ matrix.platform }}
    steps:
    - uses: actions/checkout@v1
    - uses: actions/setup-node@v1
      with:
        version: 12
    - run: npm install-ci-test
    - uses:
```

Community-powered workflows

GitHub Actions connects all of your tools to automate every step of your development workflow. Easily deploy to any cloud, create tickets in Jira, or publish a package to npm.

Want to venture off the beaten path? Use the millions of open source libraries available on GitHub to create your own actions. Write them in JavaScript or create a container Action—both can interact with the full GitHub API and any other public API.

Browse starter workflows →

Mabl
Integrate cross-browser testing into your GitHub Actions workflow
mablhq/github-mabl-actions

Find a LaunchDarkly feature flag
Search your code for references to LaunchDarkly feature flags
launchdarkly/find-code-references

Deploy to Azure App Service
Deploy your web app to Azure App Service
Azure/appservice-actions

Deploy to AWS
Deploy your app to Amaz
aws-actions

to the cloud using Zeit

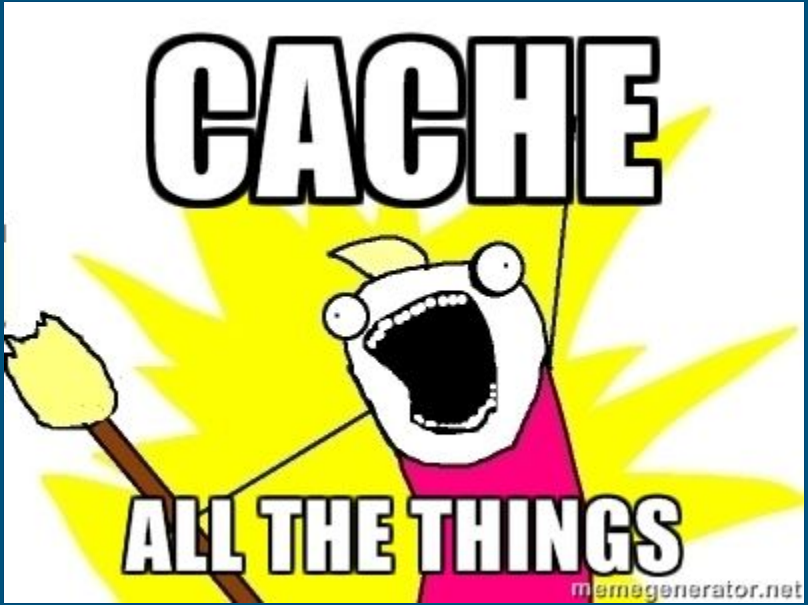
Deploy to GCP
Deploy your app to the Google Cloud Platform

Glo Boards
Integrate your Glo boards into your code workflow

Code Cov
Improve your code review workflow and quality

Why cache?

2. **Caching**



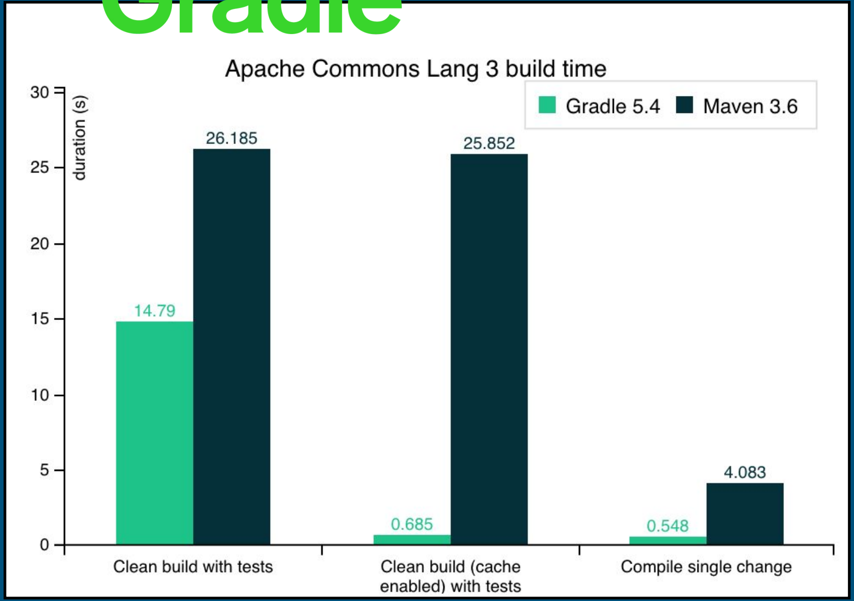
Why cache?

2. **Caching**

- Travis was **not caching** any files
- Although caching Maven build files does not have a strong effect as it does for Bundler or NPM, we tested and found an improvement from around 5 min to 2 min

Why not Maven?

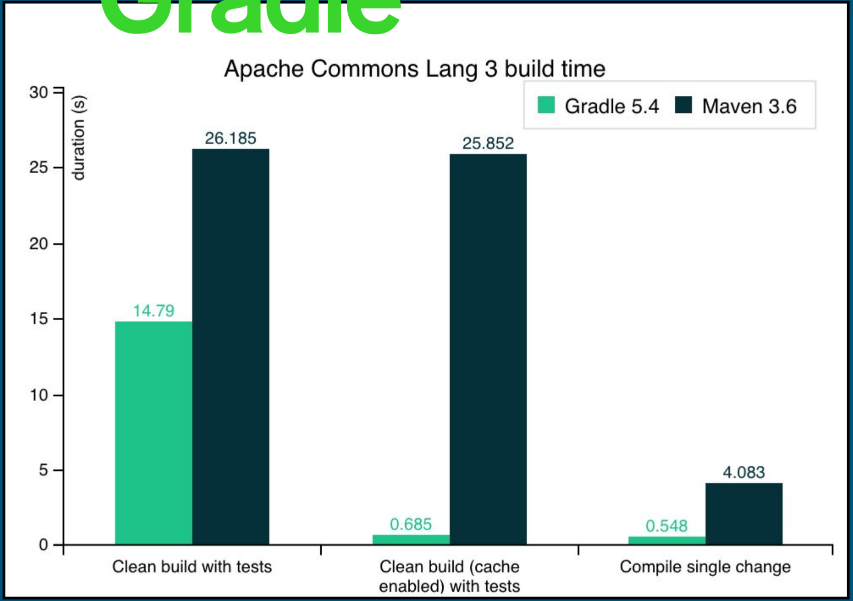
Why not Maven?



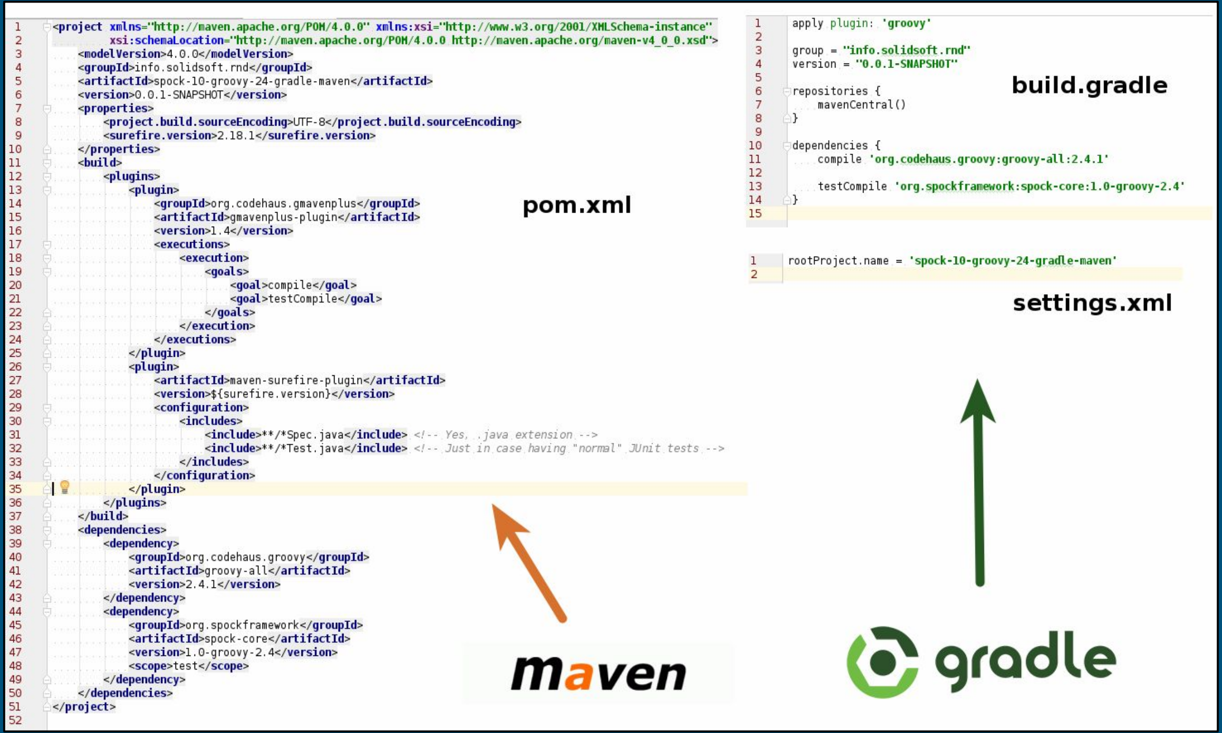
Why not Maven?

3. Issues with the Build System Maven

- Migrating to Gradle **expected to speed up build times**
- Maven is rigid and **makes customization tedious.** Gradle is much more flexible



Gradle <3



pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>info.solidsoft.rnd</groupId>
    <artifactId>spock-10-groovy-24-gradle-maven</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <surefire.version>2.18.1</surefire.version>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.gmavenplus</groupId>
                <artifactId>gmavenplus-plugin</artifactId>
                <version>1.4</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>${surefire.version}</version>
                <configuration>
                    <includes>
                        <include>**/*Spec.java</include> <!-- Yes, .java extension -->
                        <include>**/*Test.java</include> <!-- Just in case having "normal" JUnit tests -->
                    </includes>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.codehaus.groovy</groupId>
            <artifactId>groovy-all</artifactId>
            <version>2.4.1</version>
        </dependency>
        <dependency>
            <groupId>org.spockframework</groupId>
            <artifactId>spock-core</artifactId>
            <version>1.0-groovy-2.4</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>

```

build.gradle

```groovy
apply plugin: 'groovy'

group = "info.solidsoft.rnd"
version = "0.0.1-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.1'

    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}

```

```groovy
rootProject.name = 'spock-10-groovy-24-gradle-maven'

```

settings.xml

At first glance, which one would you choose?

Automate *Everything*

4. Issues with Deployment

- Skrape.it is deployed to Maven Central ***manually***
- Ideally a **hook** should be added to the workflow to automatically deploy changes
  - *This would require changes to their development workflow by automating deployment off of master (since they push and merge changes to master, they would need to push and merge changes to deployment for a hook to be added)*

Automate *Everything*

4. Issues with Deployment

Since Professor McIntosh already taught us hooks we thought we would go for a challenge…

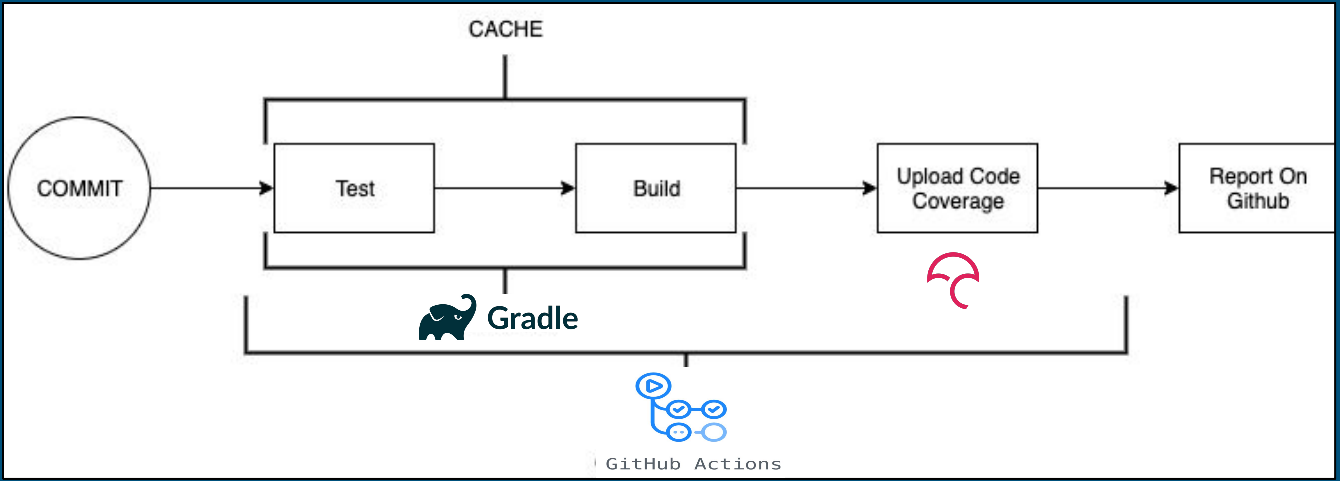
Kidding, we chose to tackle the other issues as they were more pressing and important to change before the project grows larger.

What solutions did you try and why?

1. Caching
   a. Speed up Continuous Integration time
2. Migrating from Maven → Gradle
   a. Speed up build times
   b. Offer more flexibility
3. Migrating from Travis CI → Github Actions
   a. Speed up pipeline execution
   b. Provide workflows for future customization

How does the system work?

After



So... Did it work?

To test our improvements we measured the entire time it took for the pipeline to execute.

The time it took was our dependent variable, this consisted of:

1. Time to spin up the VM
2. Execute and running the build time
3. Uploading the code to code coverage

So... Did it work?

Why did we choose pipeline execution speed as our measurement?

- Good representation of the impact of the technical improvements
- In contrast to just measuring the build time, measuring total time “from committing to feedback” is more representative of the concerns of the development team
- Lack of a better measurement. If the system was hosted on an infrastructure we could measure variables such as latency or puppet configuration.

It kind of worked:

We created 3 different experiments:

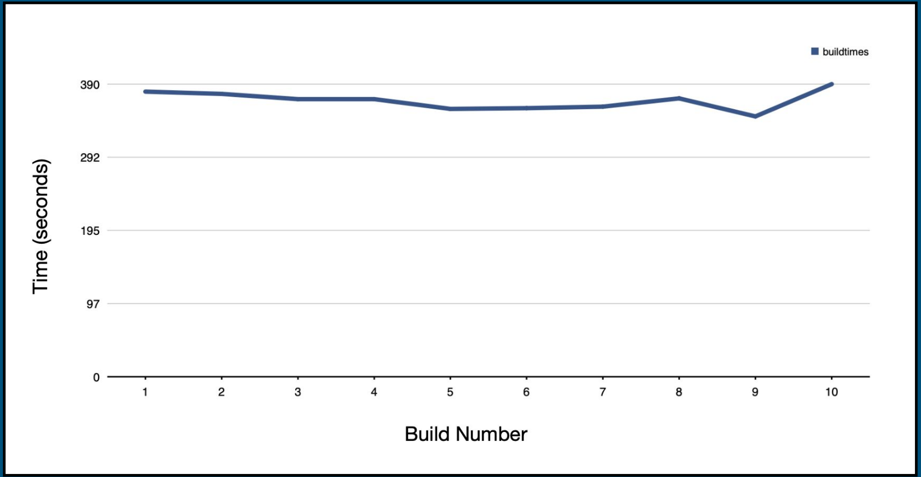
1. Maven build time with cache vs without
2. Gradle build time with cache vs without
3. Travis vs Github Actions time

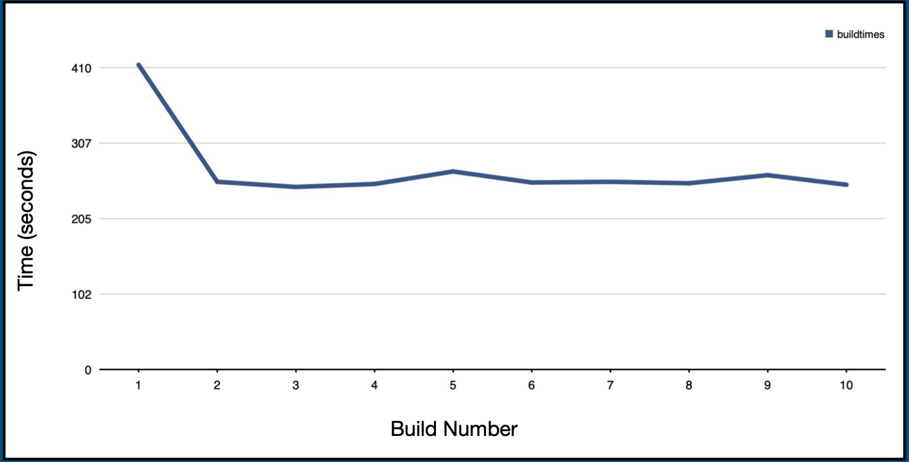
Each experiment is executed 10 times

It kind of worked.

Maven **without** cache

Maven **with** cache





It kind of worked.

Maven **without** cache	Maven **with** cache
Average time: 368 sec	**Average time:** 267 sec

Conclusion:

- After caching there is a sharp drop in time after the first build (28% drop).
- We save on having to fetch all the dependencies
- This technique will not help when adding new dependencies

What solutions did you try and why?

1. Caching
   a. Speed up Continuous Integration time ✓

2. Migrating from Maven → Gradle
   a. Speed up build times
   b. Offer more flexibility

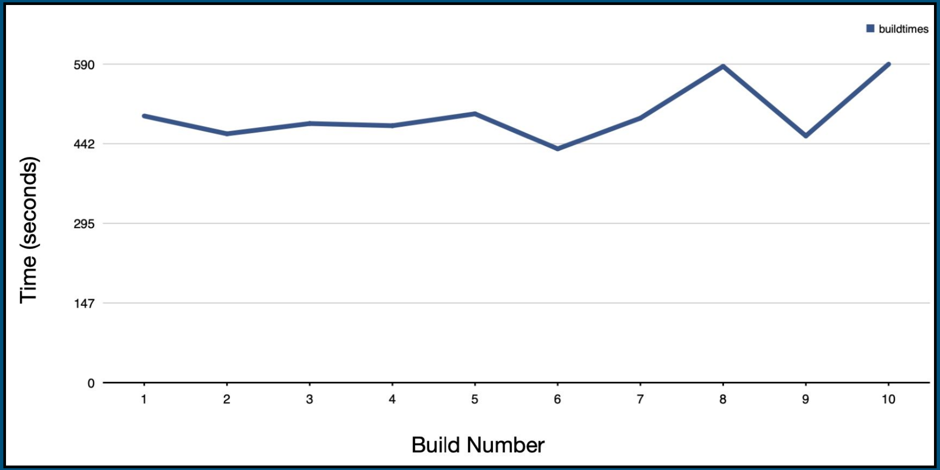
3. Migrating from Travis CI → Github Actions
   a. Speed up pipeline execution
   b. Provide workflows for future customization

Gradle

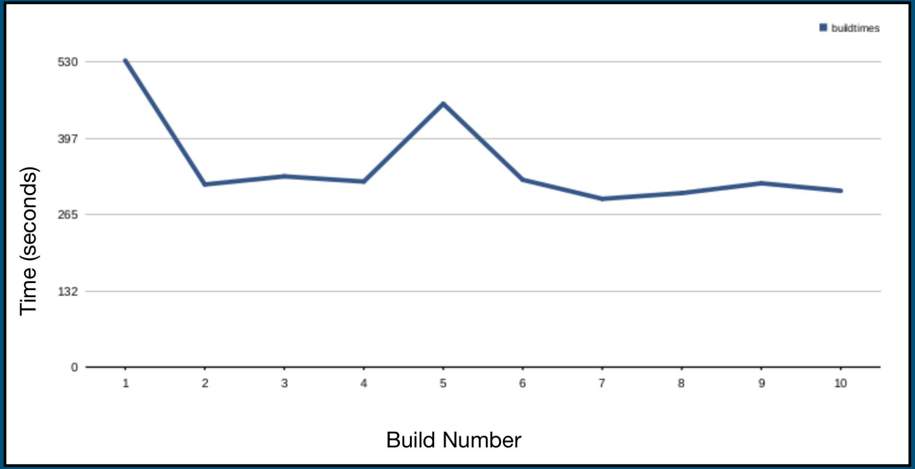
GitHub Actions

Gradle was a bust :(

Gradle **without** cache



Gradle **with** cache



Gradle was a bust :(

Gradle **without** cache

Average time: 496 sec

Gradle **with** cache

Average time: 287 sec (excluding the outlier)

Conclusion:

- 35% increase in job time when using Gradle without cache, contrary to what we expected
- Gradle jobs have a standard deviation of 49.3, hence build times vary more than with Maven
- Contrary to what was expected, Gradle times were larger

What solutions did you try and why?

1. Caching
   a. Speed up Continuous Integration time ✓

2. Migrating from Maven → Gradle
   a. Speed up bu[illegible] times ✗
   b. Offer more flexibility ✓

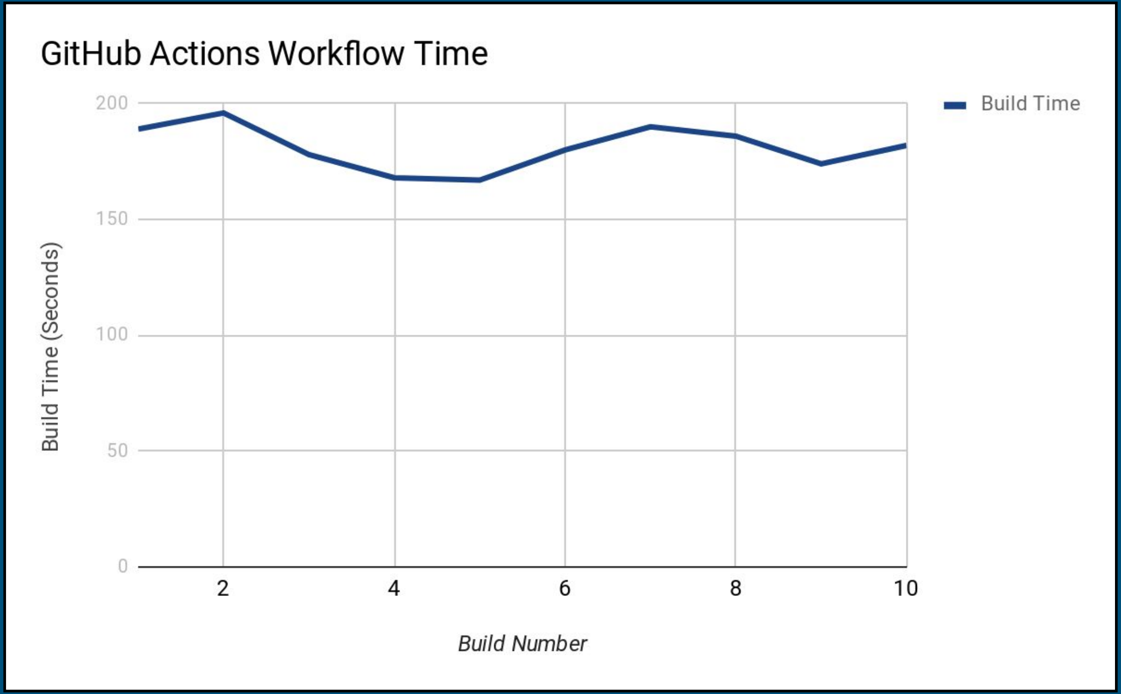
Gradle

3. Migrating from Travis CI → Github Actions
   a. Speed up pipeline execution
   b. Provide workflows for future customization

GitHub Actions

GitHub Actions was great!

Github Actions Workflow Time (on Gradle without caching)



Github Action was great!

Github Actions workflow time

Average time: 187 sec (from 368 sec)

Conclusion:

- Impressive decrease in pipeline time when migrating to Github Actions

What solutions did you try and why?

1. Caching
    a. Speed up Continuous Integration time ✓

2. Migrating from Maven → Gradle
    a. Speed up build times ✗
    b. Offer more flexibility ✓

3. Migrating from Travis CI → Github Actions
    a. Speed up pipeline execution ✓
    b. Provide workflows for future customization ✓

All in all...

- 49% decrease in build time (around 3 minutes saved)
- 245 commits → 12 hours and 15 minutes saved!
- Project now consistent with Kotlin standards (Gradle!)

Questions?