Gradle to Bazel

A Gradle user’s perspective

Introductions

- Chip Dickson
  - CTO, SUM Global Technology
  - cdickson@sumglobal.com

- Charlie Walker
  - Chief Architect, SUM Global Technology
  - cwalker@sumglobal.com

Introductions

A shameless plug for our company which has graciously allowed us to be here and to have the time to work on this presentation

SUM Global Technology

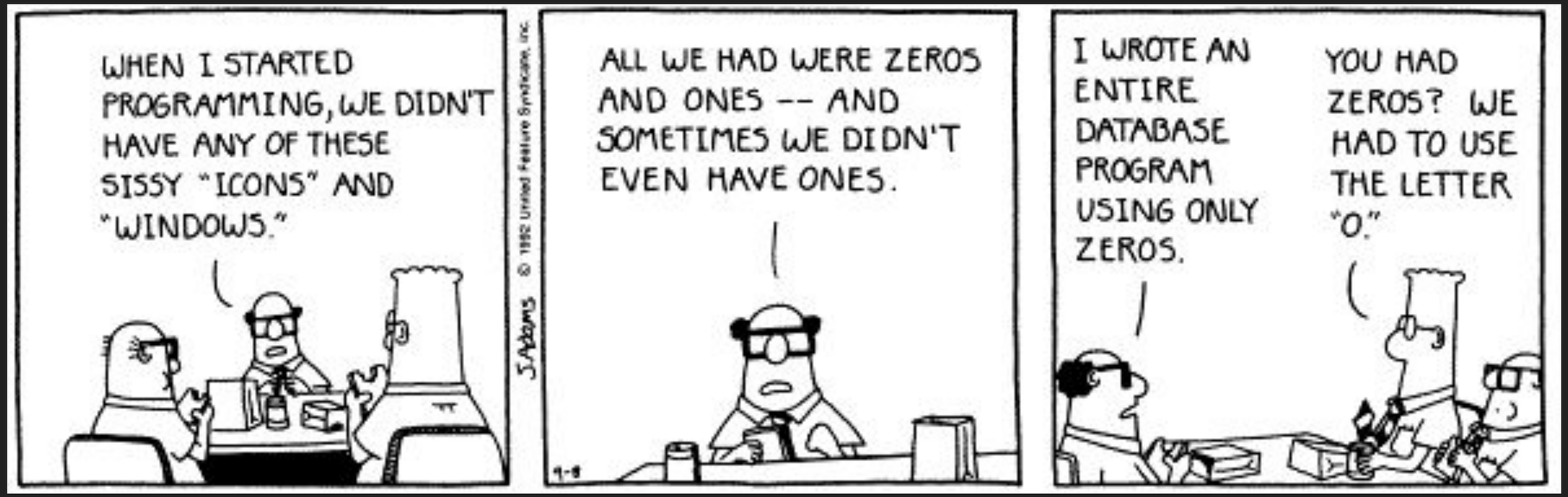
http://sumglobal.com

Come check us out, we would welcome the opportunity to work with you and your team

Who are we?

- Many years of experience with many build tools
  - Bazel, Gradle, Maven
- Bazel Community Expert
- Build and CI/CD consulting for over 8 years
- 20+ years experience architecting and developing enterprise software
  - “The Grey Beards”

SUM GLOBAL


WHEN I STARTED PROGRAMMING, WE DIDN'T HAVE ANY OF THESE SISSY "ICONS" AND "WINDOWS."
© 1992 United Feature Syndicate, Inc.
ALL WE HAD WERE ZEROS AND ONES -- AND SOMETIMES WE DIDN'T EVEN HAVE ONES.
I WROTE AN ENTIRE DATABASE PROGRAM USING ONLY ZEROS.
YOU HAD ZEROS? WE HAD TO USE THE LETTER "O."

Getting Started

Gradle

gradle init

Interactively prompts for the type of project and build file language (Groovy or Kotlin)

Bazel

Bazel has no equivalent. You need a WORKSPACE and a BUILD file in the root of your project.

For a JVM based project, you can start with the Java tutorial or one of the Java examples

For a GO projects there is Gazelle, nothing on the JVM side

What about managing different versions of the tools?

- Gradle has the gradle wrapper
  - Gradle version specified in the “gradle-wrapper.properties”
  - Downloads the specified version of Gradle and runs the build with that version
- Bazel has Bazelisk
  - https://github.com/bazelbuild/bazelisk

What about those dependencies?

build.gradle

```
repositories {
    mavenCentral()
     jcenter()
}
dependencies {
    implementation group: 'org.springframework', name: 'spring-web', version: ‘3.2.1'
    implementation group: 'org.apache.httpcomponents', name: 'httpclient'
   implementation 'com.brsanthu:google-analytics-java:2.0.0'
}
```

- Per project
  - Sub-projects can inherit dependencies from parent project

WORKSPACE

- Using rules_jvm_external

```
maven_install(
    artifacts = [
        "junit:junit:4.12",
        "org.hamcrest:hamcrest-library:1.3",
        “com.brsanthu:google-analytics-java:2.0.0”
    ],
    repositories = [
        "https://jcenter.bintray.com/",
        "https://maven.google.com",
        "https://repo1.maven.org/maven2",
    ],
)
```

SUM GLOBAL

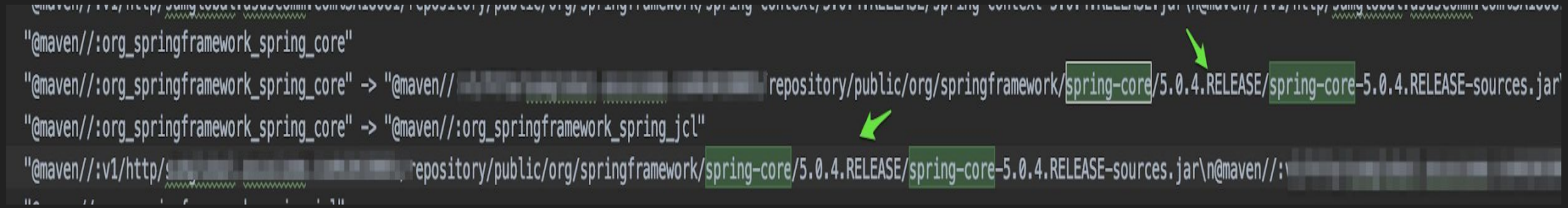
Some more about dependencies

- Gradle dependencies are resolved with highest version wins
  - Can specify dynamic versions
  - Can create version lock files
  - Transitive dependencies directly referenced in a multi-project build
  - For example: Project A depends on Guava, Project B depends on Project A and without listing Guava, can directly instantiate Guava classes.
    - Easy
    - Convenient
  - Gradle Strategy doesn't always give you what you think you are getting

```
+--- org.springframework.security:spring-security-core -> 5.0.3.RELEASE
|    +--- org.springframework:spring-aop:5.0.4.RELEASE
|    |    +--- org.springframework:spring-beans:5.0.4.RELEASE
|    |    |    \--- org.springframework:spring-core:5.0.4.RELEASE
|    |    |         \--- org.springframework:spring-jcl:5.0.4.RELEASE
|    |    \--- org.springframework:spring-core:5.0.4.RELEASE -> 5.0.7.RELEASE
|    +--- org.springframework:spring-beans:5.0.4.RELEASE (*)
```

Some more about dependencies (cont)

- Achieving hermetic, correct builds takes work
- Bazel's promise of Fast, Correct is something we have found reliable
- Same project, from the Bazel side, uses the explicit dependency declared in the project



SUM GLOBAL

Why all the extra time on dependencies?

- In a Gradle to Bazel conversion, it’s the single biggest challenge
  - You find out how much you don’t know about your Gradle build and what it depends on
  - Some things quit working because they were hidden by Gradle’s dependency management
    - What if, in our previous example, the code used something fixed in Spring Core 5.0.7 but the project is built on 5.0.4?
      - VERY hard to figure out why things are broken with Bazel build and not Gradle build
      - First reaction is Bazel is broken
        - In fact, neither is Gradle,
          - Specifically your code is broken, Bazel just pointed it out
        - Should have upgraded to Spring 5.0.7 (or some other chosen release), but Gradle hid the problem
          - Someone found the issue someplace, Gradle won’t just make up a version, something brought it in

So how do I specify the java version?

build.gradle

```
sourceCompatibility = 1.8
targetCompatibility = 1.8
```

Bazel command line or .bazelrc file

```
--javacopt="-source 8 -target 8 -encoding UTF-8"
```

OR

In the .bazelrc file

```
build --javacopt="-source 8 -target 8 -encoding UTF-8"
```

Viewing the dependency graph

./gradlew dependencies

./gradlew -q dependencyInsight --configuration compile --dependency slf4j-api

./gradlew dependentComponents

To run on a sub-project syntax is:

./gradlew :sub-project:dependencies

bazel query “deps(//foo)”

--output=graph

--notool_deps

Many other options

How do I declare a “task”?

```
task(‘hello’) {
    doLast {
        println “hello”
    }
}
```

rules.bzl

```
def _print_impl(ctx):
    print('Hello World')

print_rule = rule(implementation = _print_impl)
```

BUILD

```
load("//rules.bzl","print_rule")
print_rule(name=”hello”)
```

You put that build file where?

In Gradle, you put a “build.gradle” file at the top level of the project/subproject directory

While you can do the same in Bazel, it’s a better practice to put a “BUILD” file at each package level

And why aren’t my tests just running?

So in gradle, my unit tests are automatically run when I call “./gradlew build” or I can run them by calling “./gradlew test”.

Bazel works differently. To run a test target in Bazel:

bazel test //path/to:target

Or to run all tests:

bazel test //:...

What about Spring Boot?

In Gradle, there's a plugin for that…

Well supported by Spring

There are several older rule sets available for modification.
https://github.com/salesforce/bazel-springboot-rule is one of the better ones, but is 2 years old.

SUM Global is working on a rule set we hope to have available after the first of the year to generically support Spring Boot with Bazel 1.+ and when it's available Bazel 2.+

We have built our own for specific projects as needed so far ...

What about Micro-Profile?

Micro-Profile is still living in the Maven past, but is doable in Gradle

- Current tools lack solid Gradle support
  - Thorntail has Gradle plugin, but its not well supported as per the documentation
  - Open Liberty has ci.gradle plugin, updated regularly, but still second class to Maven tools, heavily relies on Ant tasks

With Bazel, you are on your own. These tools have documentation on how they are packaged etc. This is one on SUM Global's radar for building out support for thorntail.io and/or openliberty if there is interest.

I have my fat jar, what about my Docker image?

There are solid Docker plugins available. They execute the Docker installed on your system.

- Not hermetic, but it works
- Not run in parallel
  - Can take a long time if your project has multiple Docker containers
- Uses Dockerfile
- Publish using installed Docker

Bazel has rules_docker. It's a very different approach. Bazel has its own container builder.

- Hermetic
- Very fast
  - Runs in parallel
- Image configured via the rule
  - Different approach, takes planning and understanding
  - Shouldn't you know what you put in your container?
  - Publish via "run" using installed Docker - Same issue as Gradle for speed

I have my fat jar, what about my Docker image? - 2

- com.bmuschko:gradle-docker-plugin

```
// Copy artifacts to Docker input folder.
task copyArtifactsToDocker(type: Copy, dependsOn: dockerFile) {
  from "${buildDir}/libs/${project.name}-${project.version}-boot.jar"
  into 'build/docker'
}

// Build an image from the Dockerfile.
task dockerImage(type:
com.bmuschko.gradle.docker.tasks.image.DockerBuildImage, dependsOn:
copyArtifactsToDocker) {
  inputDir = project.file('build/docker')
  if(!project.hasProperty("debug")) {
      tags = ["${dockerRepository}/foo/${project.name}:${version}"]
  } else {
      tags = ["${dockerRepository}/foo/${project.name}:debug-${version}"]
  }
}
```

```
container_image(
    name = "image_{}".format(_local_build_properties["bazel.base.name"]),
    labels = {"maintainer" : "admin@foo.com"},
    creation_time = "{BUILD_TIMESTAMP}",
    stamp = True,
    base = "@java_base//image",
    ports = ["8080"] if DEBUG != True else ["8080","8001","1114"],
    files = ["{}-{}-boot.jar".format(_local_build_properties["pom.artifact.id"],
_local_build_properties["pom.version"])],
    tars = ["//{}:grpc_health_probe".format(_local_build_properties["pom.artifact.id"])],
   cmd = cmd if DEBUG != True else debugCmd,
)
```

Publishing a jar file?

Gradle, you add the plugin “maven-publish” and define the parameters. POM file and Gradle’s new module metadata file created and published.

Bazel itself doesn’t provide this functionality.

- The publishing part would probably be straightforward as it’s just http put calls.
  - The problem is in the creation of pom files. We have had success with @bazel_common//tools/maven:pom_file.bzl", "pom_file"
- While there are issues in Bazel opened around this problem, Grakn Labs open sourced a collection of rules to assemble and deploy to Maven, Pip, Homebrew, Packager and RPM. https://github.com/graknlabs/bazel-distribution

How to run Code Quality Tools

Gradle has plugins for the various tools.

- Spotbugs
  - Successor to Findbugs
- PMD
- Checkstyle

Error Prone (http://errorprone.info) is already built in to Bazel and works straight out of the box. You can control what is checked and how its reported (Error or Warning) via command line options or .bazelrc file

EX: build --javacopt="-Xep:ArrayToString:WARN”

There are also rules for things like sonarqube (https://github.com/Zetten/bazel-sonarqube)

Specific tools will require custom rules

But don’t I need to “clean” the build?

Gradle does a good job of keeping you from running “clean”, but ...

- There are simply times you have no other choice.
  - Often when dependencies are updated to a different version or plugins are updated.

One of Bazel’s primary goals is to ensure correct incremental builds. How it works:

- Bazel maintains a database of all work previously done, and will only omit a build step if it finds that the set of input files (and their timestamps) to that build step, and the compilation command for that build step, exactly match one in the database, and, that the set of output files (and their timestamps) for the database entry exactly match the timestamps of the files on disk. Any change to the input files or output files, or to the command itself, will cause re-execution of the build step.
- The benefit to users of correct incremental builds is less time wasted waiting for rebuilds cause by the use of a “clean” command

Umm… So how do I build a war file?

So in Gradle, i just add the “war” plugin.

In Bazel, the java ruleset doesn’t contain any rules for building a war file.

BUT the appengine rules do.

```
load("@io_bazel_rules_appengine//appengine:java_appengine.bzl", "java_war")
java_war(
    name = "myapp",
    srcs = ["java/my/webapp/TestServlet.java"],
    data = glob(["webapp/**"]),
    data_path = "/webapp",
    deps = [
        "//external:appengine/java/api",
        "@io_bazel_rules_appengine//appengine:javax.servlet.api",
    ],
)
```

SUM GLOBAL

What if I need to do something special?

- Plugins
- Custom Tasks

- Bazel rules
- Tool Chains
- Platforms

Performance - A bit about the project

- Original Gradle build produces 11 libraries and 19 microservices each in their own Docker container
- Simple stats comparison is done running a clean build and producing the Docker containers in both Gradle and Bazel
- Original build was untouched
- Bazel build not optimized or broken down to the package level
  - BUILD files every place there is a build.gradle file
- There are things that could make both builds faster
  - Gradle - 80/20 rule applies, would take a lot of effort to optimize further
  - Bazel - further break down of the work units (BUILD files) could improve build speed

Performance and Build Results

- Gradle dependencies….
  - Oops… how did that get in there?
- Build speed
  - First run
    BUILD SUCCESSFUL in **1m 54s**
    142 actionable tasks: 142 executed
  - Running it again
    BUILD SUCCESSFUL in **1m 5s**
    142 actionable tasks: 19 executed, 123 up-to-date

- Bazel dependencies
  - Exactly what you asked for, right or wrong
- Build speed
  - First run:
    INFO: Elapsed time: **106.258s**, Critical Path: 37.23s
    INFO: 451 processes: 418 darwin-sandbox, 33 worker.
    INFO: Build completed successfully, 518 total actions
  - Running it again
    INFO: Elapsed time: **0.432s**, Critical Path: 0.06s
    INFO: 0 processes.
    INFO: Build completed successfully, 1 total action

Some other things

- Gradle - JVM languages with some native support
- Android
  - Gradle/Kotlin the de facto way to build Android apps these days

Gradle builds your Java based stack, must use other tools for other parts of your build

- Bazel builds many things besides Java well
  - Bazel Native
    - Yes Please!
  - Android
    - Solid support, still improving
  - Kotlin
    - Has some warts in the Kotlin space, but is improving every day
    - We have had success migrating Kotlin projects
  - Other JVM languages
    - Scala - Lots of good work here - *Ittai Zeidman is your person for answers here*
    - Groovy - Solid support
  - Javascript/Typescript/Angular

Bazel gives you a full stack approach

Lessons Learned

- It's not always the tool that is doing something wrong
- Dependencies are intentional and precise with Bazel
- Replacing plugins from Gradle’s ecosystem is not always a direct port
  - Many parts may already exist in Bazel, just not lumped in with Java tools
- Bazel “thinks” differently than Gradle
  - Bazel “With great power comes great responsibility” VS Gradle “Don’t you worry your pretty head about that…”
  - Both approaches have their merit

Questions?

SUM GLOBAL