C++ Binary Dependency Management with Gradle

Hugh Greene <hughg@tameter.org>

C++ Binary **Dependency Management** with Gradle

Getting
consistent versions
of things needed
to build your software
and to use it

C++ **Binary Dependency Management** with Gradle

- **Why?**
  - Saves time
  - Identical binaries → confidence in testing
  - Fewer global installs of build tools
  - Licence costs for compilers, SDKs
  - Restricted source access

C++ Binary Dependency Management with Gradle

- **Newer languages/platforms have their own**
  - Python: PIP
  - Ruby: Gems
  - JVM: Ivy, Maven, Gradle, …
  - .Net: NuGet
- **C++ (in 2013)**
  - NuGet: poor native variant support
    - + other disadvantages

C++ Binary Dependency Management with Gradle

- **Getting versions: package manager**
  - *nix: rpm, apt, etc. … maybe Nix
  - Windows: nothing (in 2013)
  - … or download src, make, install
- **Connecting to build**
  - *nix: ./configure
  - Cross-platform: CMake
    - Awkward with Visual Studio

C++ Binary Dependency Management with Gradle

- **The Holy Gradle**
  - and friends
- **Developed and used commercially in-house**
  - OSS but **NOT** supported externally
  - since 2013

C++ Binary **Dependency Management** with Gradle

- **Some requirements**
  - Easy to understand version set
  - Reproducible builds
  - Robust against tool failure/bugs
  - Binary repository server
    - Stop storing binaries in Subversion!
  - Disconnected sites
  - Windows platform
  - Visual Studio (mostly)

Gradle

- **What?**
  - Java-land Dependency Management + Build tool
    - CMake for Java ≈ building a classpath
  - Modules: ID/coordinate = “group:name:version”
  - Repositories: http(s)://, file://, with URL patterns
  - Configurations: “slices” joining modules/artifacts/tasks
  - Artifacts: files
    - + metadata files: ID, configurations, artifacts, dependencies
  - Tasks: like make, nmake, MSBuild

C++ Binary Dependency Management with **Gradle**

- **Why?**
  - Dependency Management is hard
    - learn from others
  - Groovy DSLs are developer-friendly
    - … more than Ant or MSBuild, at least!
  - Not much else in 2013
    - See later for 2017 udpate

C++ Binary Dependency Management with Gradle

- **How? “The Holy Gradle” plugins**
  - ZIP artifacts
    - Unpack cache + symlinks in project workspace
  - Offline repo export (for disconnected sites)
  - Source dependencies
    - Multiple source repo graph
    - Binaries published together
    - Build & test all
  - Windows Credential Store integration
  - … currently stuck on Gradle 1.4 :-/

C++ Binary Dependency Management with Gradle

```
buildscript {
    gplugins.use "intrepid-plugin:7.7.2"
}
gplugins.apply()

group = "com.example-corp.teamA"
version = System.getenv("NEXT_VERSION_NUMBER") ?: Project.DEFAULT_VERSION

repositories.ivy {
    url "http://artifactory-server/artifactory/libs-release"
    credentials {
        username my.username("Artifactory")
        password my.password("Artifactory")
    }
}

configurationsSets {
    main { type configurationSetTypes.DLL_64 }
    test {
        type configurationSetTypes.EXE_64
        prefix "test"
    }
}
```

C++ Binary Dependency Management with Gradle

```
sourceDependencies {
    framework {
        git "http://git-server/path/to/framework"
        configurationSet configurationSets.main, configurationSetTypes.DLL_64
    }
    doc {
        svn "http://hg-server/path/to/my-doc"
        // No configuration mapping because it's not buildable, just doc.
    }
}

packedDependencies {
    "dep/RenderingLib" {
        dependency "com.example-corp.rendering:RenderingLib:2012a2"
        configurationSet configurationSets.main, configurationSetTypes.LIB_64
    }
    "dep/NUnit" {
        dependency "org.nunit:NUnit:2.5.10"
        def testRuntimeConfs = configurationSets.test.configurationNamesMap.findAll { k, v ->
            k[stage] == 'runtime'
        }
        configuration "${testRuntimeConfs.join(',')}->bin"
        unpackToCache = false
    }
}
```

C++ Binary Dependency Management with Gradle

```
packageArtifacts {
    import_common {
        include "src/**/*.h"
    }
    configurationSets.main.axes['Configuration'].each { conf ->
        "import_x64_${conf}" {
            include "lib/${conf}/*.lib"
        }
        "runtime_x64_${conf}" {
            include "bin/${conf}/*.dll"
        }
        "debugging_x64_${conf}" {
            include "bin/${conf}/*.pdb"
        }
    }
}

publishPackages {
    repositories.ivy {
        credentials {
            username my.username("Artifactory")
            password my.password("Artifactory")
        }
        url "http://artifactory-server/artifactory/my-integration-repo-local/"
    }
}
```

C++ Binary Dependency Management with Gradle

- Future …?
  - Binary → source replacement in workspace
  - Update to Gradle 2.x/3.x
    - Java 8; better performance; CMake-alike for C/C++
    - Kotlin for statically-checked build scripts!
  - Publicly buildable
  - Publicly published
  - Auto-generate deploy scripts
  - Auto-generate MSBuild or CMake fragments

C++ Binary Dependency Management with Gradle

- **NuGet**
  - Support for native variants still poor
  - Best with Visual Studio (but modifies projects)
  - Multiple copies of binaries
  - No source packages
- **Biicode**
  - Defunct

C++ Binary Dependency Management with Gradle

- **conan.io**
  - Use CMake or invent your own workspace integration
  - Source and binary packages
  - Written in Python
  - Artifactory support
- **vcpkg**
  - Source builds only
  - Visual Studio only

C++ Binary Dependency Management with Gradle

- **Links**
  - https://holygradle.bitbucket.io
  - https://bitbucket.org/nm2501/holy-gradle-plugins
  - https://docs.gradle.org/1.4/userguide/userguide.html
  - https://www.jfrog.com/confluence/display/RTF

- **Questions?**