Nebula: Netflix's OSS Gradle Plugins

Rob Spieldenner @robspieldenner

http://netflix.github.io/

https://github.com/nebula-plugins

Why Open Source

- Help out other teams using gradle
- Contributions
- Hiring

Current Plugin Build/Publish Infrastructure

- Github
  - branch per gradle release
  - version based on gradle release 2.4.x for gradle-2.4
- Cloudbees (Jenkins) for CI and release: Job DSL to setup a snapshot and release job per branch
- Bintray (jcenter) as host

New Plugin Build/Publish Infrastructure

- Github
  - matrix testing for different versions of gradle
  - semantic versioning
- Travis CI for CI, release on github tag
- Bintray (jcenter) as host
- Publish with gradle plugin portal

Support, Issues, etc.

- Github issues
  - We love pull requests
  - or breaking tests
- gitter.im

Naming Scheme

- nebula- : indicates a highly opinionated plugin
- gradle- : indicates we believe everyone would benefit from the functionality

nebula-plugin-plugin

https://github.com/nebula-plugins/nebula-plugin-plugin

nebula-plugin-plugin

- Opinions for publishing other plugins
- Eliminate boiler plate configuration
- Use nebula plugins to build themselves

nebula-test

https://github.com/nebula-plugins/nebula-test

nebula-test

will be replaced by Gradle TestKit

- ProjectSpec
- PluginProjectSpec
- IntegrationSpec
- Dependency Generation

PluginProjectSpec

```
import nebula.test.PluginProjectSpec
class PluginExampleSpec extends PluginProjectSpec {
  @Override
  String getPluginName() { 'plugin-example' }

  def ‘check task is setup’() {
    when:
    project.plugins.apply(PluginExample)

    then:
    def exampleTask = project.tasks.get(‘example’)
    exampleTask.exampleProperty == 'myConfiguredValue'
  }
}
```

IntegrationSpec

```
import nebula.test.IntegrationSpec
class MyExampleSpec extends IntegrationSpec {
  def 'example test'() {
    buildFile << """\
      apply plugin: 'java'
      ${applyPlugin(MyExamplePlugin)}
    """.stripIndent()

    when:
    def results = runTasksSuccessfully('exampleTask')

    then:
    results.wasExecuted(':exampleTask')
    results.wasUpToDate(':dependentTask')
    results.standardOutput.contains 'Example task output'
    fileExists('build/example/mycache.tmp')
  }
}
```

Dependency Generation

```
import nebula.test.dependencies.DependencyGraphBuilder
import nebula.test.dependencies.ModuleBuilder

// ...
  def 'generate dependencies'() {
    given:
    def graph = new DependencyGraphBuilder().addModule('g0:a0:0.0.1')
        .addModule('g1', 'a1', '1.0.1')
        .addModule(new ModuleBuilder('g2:a2:2.0.1').build())
        .addModule(new ModuleBuilder('g3:a3:3.0.1')
            .addDependency('g4:a4:4.0.1')
            .addDependency('g5', 'a5', '5.0.1').build()
        ).build()
    def generator = new GradleDependencyGenerator(graph)
    def mavenRepo = generator.generateTestMavenRepo()
    // def ivyRepo = generator.generateTestIvyRepo()

    // rest of test
  }
```

nebula-core

https://github.com/nebula-plugins/nebula-core

nebula-core

Utility tasks for other plugins

- Download
- Unzip
- Untar

nebula-project-plugin

https://github.com/nebula-plugins/nebula-project-plugin

nebula-project-plugin

- Opinionated about what a responsible project should do
    - Builds Javadoc and Sources jars
    - Record information about the build and stores it in the .jar, via gradle-info-plugin
    - Easy specification of people involved in a project via gradle-contacts-plugin
- facets to ease setting up new source sets

gradle-netflixoss-project-plugin

https://github.com/nebula-plugins/gradle-netflixoss-project-plugin

gradle-netflixoss-project-plugin

- Provide release process
- Configure publishing
- Recommend license headers
- Add some error handling for javadoc in jdk8

gradle-ospackage-plugin

https://github.com/nebula-plugins/gradle-ospackage-plugin

gradle-ospackage-plugin

```
apply plugin: 'nebula.os-package'

ospackage {
  installUtils file('scripts/utils.sh')
  preInstall file('scripts/preInstall.sh')
  postInstall file('scripts/postInstall.sh')
  preUninstall 'touch /tmp/myfile'
  postUninstall file('scripts/postUninstall.sh')

  requires('qux')

  into '/'
  from 'root'
}

buildRpm {
  requires('bar', '2.2', GREATER | EQUAL)
  requires('baz', '1.0.1', LESS)
  link('/etc/init.d/foo’, '/opt/foo/bin/foo.init')
}

buildDeb {
  requires('bat', '1.0.1')
  link('/etc/init.d/foo', '/opt/foo/bin/foo.upstart')
}
```

nebula-ospackage-plugin

https://github.com/nebula-plugins/nebula-ospackage-plugin

nebula-ospackage-plugin

- `nebula.nebula-ospackage-daemon` - Setup daemontools in a system package
- `nebula.nebula-ospackage-application` - Put contents of application zip into /opt/$applicationName
- `nebula.nebula-ospackage-application-daemon` - Combine the above, auto setup daemontools script for the application plugin output

gradle-dependency-lock-plugin

https://github.com/nebula-plugins/gradle-dependency-lock-plugin

gradle-dependency-lock-plugin

```
plugins {
  id 'nebula.dependency-lock' version '2.2.3'
  id 'groovy'
}

repositories { jcenter() }
dependencies {
  compile 'org.codehaus.groovy:groovy-all:2.+'
  compile 'com.google.guava:guava:latest.release'
  testCompile 'junit:junit:[4.0, 5.0)'
}
```

gradle-dependency-lock-plugin

```
./gradlew generateLock saveLock
```

```
{
  "com.google.guava:guava": { "locked": "19.0-rc1", "requested": "latest.release" },
  "junit:junit": { "locked": "4.12", "requested": "[4.0, 5.0)" },
  "org.codehaus.groovy:groovy-all": { "locked": "2.4.4", "requested": "2.+" }
}
```

gradle-dependency-lock-plugin

```
plugins {
  id 'nebula.dependency-lock' version '2.2.3'
  id 'groovy'
}

dependencyLock {
  includeTransitives = true
}

repositories { jcenter() }
dependencies {
  compile 'org.codehaus.groovy:groovy-all:2.+'
  compile 'com.google.guava:guava:latest.release'
  testCompile 'junit:junit:[4.0, 5.0)'
}
```

gradle-dependency-lock-plugin

```
./gradlew generateLock saveLock
```

```
{
  "com.google.guava:guava": { "locked": "19.0-rc1", "requested": "latest.release" },
  "junit:junit": { "locked": "4.12", "requested": "[4.0, 5.0)" },
  "org.codehaus.groovy:groovy-all": { "locked": "2.4.4", "requested": "2.+" },
  "org.hamcrest:hamcrest-core": { "locked": "1.3", "transitive": [ "junit:junit" ] }
}
```

gradle-extra-configurations-plugin

https://github.com/nebula-plugins/gradle-extra-configurations-plugin

gradle-extra-configurations-plugin

```
apply plugin: 'nebula.provided-base'

dependencies {
  provided 'example:providedlib:42.1.2'
}

// ------------------

apply plugin: 'nebula.optional-base'

dependencies {
  compile 'example:optionallib:2.3.4', optional
}
```

nebula-publishing-plugin

https://github.com/nebula-plugins/nebula-publishing-plugin

nebula-publishing-plugin

- nebula.javadoc-jar
- nebula.source-jar
- nebula.test-jar

nebula-publishing-plugin

- nebula.apache-license-pom.properties
- nebula.manifest-pom.properties
- nebula.maven-base-publishing.properties
- nebula.maven-java-publishing.properties
- nebula.resolved-pom.properties
- nebula.scm-pom.properties

nebula-publishing-plugin

nebula.maven-publishing/nebula.ivy-publishing

- resolve dynamic versions to specific
- dependencies for war
- add licenses
- add manifest/informational properties

nebula-publishing-plugin

```xml
<name>gradle-dependency-lock-plugin</name>
<description>Gradle plugin to allow locking of dynamic dependency versions</description>
<properties>
  <nebula_Manifest_Version>1.0</nebula_Manifest_Version>
  <nebula_Implementation_Title>com.netflix.nebula#gradle-dependency-lock-plugin;2.2.3</nebula_Implementation_Title>
  <nebula_Implementation_Version>2.2.3</nebula_Implementation_Version>
  <nebula_Built_By>jenkins</nebula_Built_By>
  <nebula_Build_Date>2015-04-02_10:02:13</nebula_Build_Date>
  <nebula_Gradle_Version>2.2.1</nebula_Gradle_Version>
  <nebula_Module_Origin>git@github.com:nebula-plugins/gradle-dependency-lock-plugin.git</nebula_Module_Origin>
  <nebula_Change>f91b5ad</nebula_Change>
  <nebula_Build_Id>2015-04-02_16-59-13</nebula_Build_Id>
  <nebula_Created_By>1.7.0_60-b19 (Oracle Corporation)</nebula_Created_By>
  <nebula_Build_Java_Version>1.7.0_60</nebula_Build_Java_Version>
  <nebula_X_Compile_Target_JDK>1.7</nebula_X_Compile_Target_JDK>
  <nebula_X_Compile_Source_JDK>1.7</nebula_X_Compile_Source_JDK>
</properties>
<scm>
  <url>git@github.com:nebula-plugins/gradle-dependency-lock-plugin.git</url>
  <connection>scm:git@github.com:nebula-plugins/gradle-dependency-lock-plugin.git</connection>
</scm>
<developers>
  <developer>
    <id>exampleuser</id>
    <name>Example User</name>
    <email>example@example.email</email>
  </developer>
</developers>
```

gradle-metrics-plugin

https://github.com/nebula-plugins/gradle-metrics-plugin

gradle-metrics-plugin

- Stopped development in favor of gradle.com
- Elaticsearch index of each build
    - Info - Gradle start parameters, system properties and environment variables. SCM and GIT information if the gradle-info-plugin has been applied
    - Project - name and version
    - Events - configuration, dependency resolution, task execution

gradle-override-plugin

https://github.com/nebula-plugins/gradle-override-plugin

gradle-override-plugin

```
apply plugin: 'nebula.nebula-override'

class MyExtension {
    String myProp
}

example {
    myProp = 'hello'
}
```

```
$ ./gradlew -Doverride.example.myProb=newValue <task>
$ ./gradlew -Doverride.sourceCompatibility=1.7 <task>
```

Other Plugins

- nebula-clojure-plugin
- nebula-bintray-plugin
- gradle-stash-plugin
- gradle-contacts-plugin
- gradle-scm-plugin
- gradle-git-scm-plugin

Questions

Netflix is Hiring

- Build Tools - Full Stack Engineer
- https://jobs.netflix.com/