

nuts, the Java Package Manager

<https://github.com/thevpc/nuts> (git repo)

<https://thevpc.github.io/nuts> (website)

nuts.packagemanager@gmail.com

thevpc, 2025-01-04

Plan

1. Why a package manager
2. **nuts** features
3. Demo

1. Why a Package Manager

¥ Popularity of a language is proportional to popularity of its PM

! Javascript: npm/npx/yarn

! Python: pip, conda

! Ruby: rubygems

¥ Newcomer languages already include a PM

! go lang package manager (modules)

¥ Java ecosystem already have more than 7M packages deployed

1.1. Java Package Manager?

¥ **maven, gradle**

- ! Build tools
- ! Dependency-management tools
- ! Poor package/deployment management (**maven** 's **depl oy** is a build time stage)
- ! Lack of deployment lifecycle (install/uninstall/update)

1.2. Example

Let's take a **hello-world** example with dependencies :

```
package net.thevpc.nuts.doc.baseproject;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Main {
    private static final Logger LOG = LoggerFactory.getLogger(Main.class);
    public static void main(String[] args) {
        LOG.debug("A simple app with dependencies. Won't work out of the box!,
unless...");
    }
}
```

1.3. pom.xml

A minimal **pom.xml** is:

```
<?xml version="1.0" encoding="UTF-8"?><project xmlns="..."><modelVersion>
4.0.0</modelVersion>
<groupId>net.thevpc.nuts.doc</groupId>
<artifactId>base-project</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
<dependency>
Ê   <groupId>ch.qos.logback</groupId>
Ê   <artifactId>logback-classic</artifactId>
Ê   <version>1.2.10</version>
</dependency>
</dependencies>
</project>
```

1.4. Example

- ¥ With a minimal `pom.xml` we cannot execute unless we add the transitive `dependencies` to the classpath
- ¥ We also need to adjust the `pom.xml` to include the main class too!

1.5. Alternatives for deployment

- ¥ Java Web Start
- ¥ System PM / Installers
- ¥ jpackage, jlink
- ¥ Portable Installers
- ¥ Custom Deployments
- ¥ Build time Processors (Fat Jars)

1.6. Java Web start

- ¥ Run Remote App using **j n l p** file (with all of its dependencies)
- ¥ Special packaging
- ¥ Execution Sandbox : More Limitations
- ¥ Deprecated!! since Java9
- ¥ No Shared Dependencies / Centralized Dep Mgt
- ¥ The same applies to alternatives : **trivrost**, **OpenJNLP**

1.7. System PM / Installers

¥ rpm, deb, dmg, msi

- ! Native integration with OS/Env
- ! Centralized management
- ! Automatable (cmdline)
- ! Not portable
- ! Multiple deployment packages for each release
- ! Problem with installing multiple versions of the same package

1.8. JPackage jlink

¥ rpm, deb, dmg, msi

- ! All System PM / Installers applies
- ! Not portable
- ! java 8- not supported
- ! requires all dependencies to be packaged as rpm/deb or be bundled for each app
- ! JRE bundled each time!

1.9. Portable Installers

¥ Install Anywhere, GetDown, IzPack, BitRock Installer

- ! Good integration with OS/Env
- ! No centralized management
- ! Disk and network overload of dependencies
- ! Graphical! not suitable for automation (most of the time)
- ! Still Manual

1.10. Custom Deployers

¥ Custom (tomcat, netbeans) with multiple formats (tarball, zip)

- ! Manual
- ! No centralized management
- ! Difficult to automate
- ! Lack of integration with environment
- ! Disk and network overload of dependencies

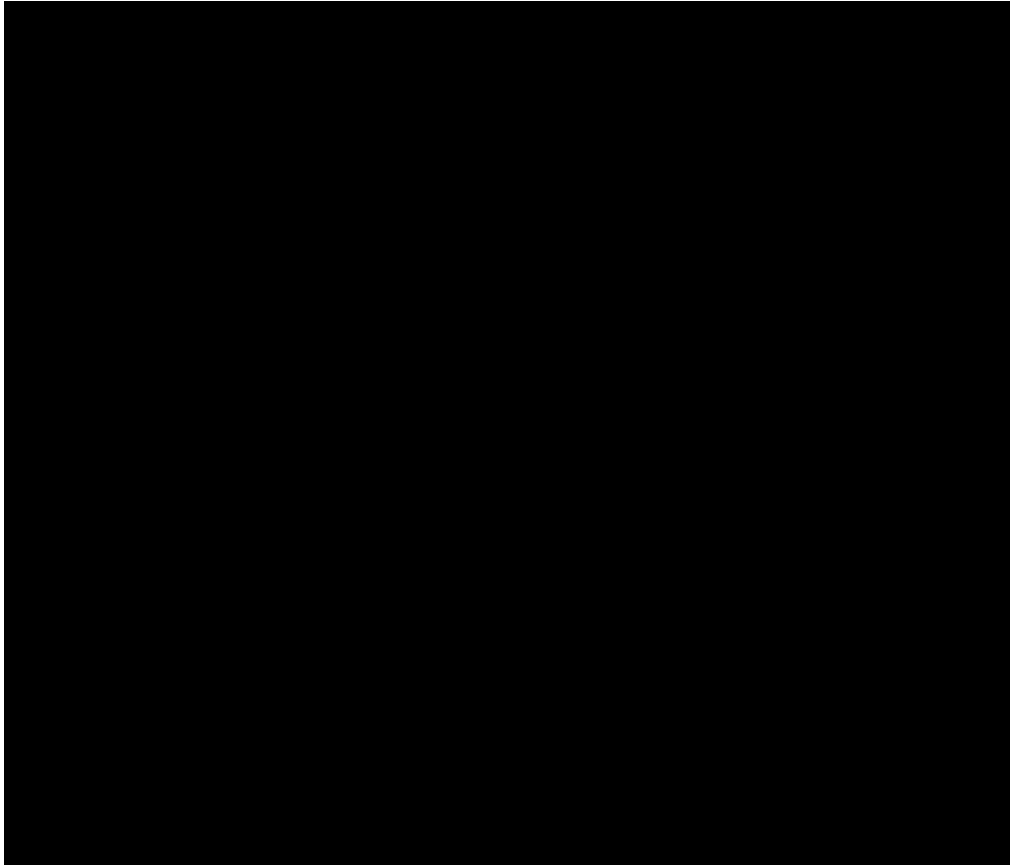
1.11. Fat Packages: maven-dependency-plugin

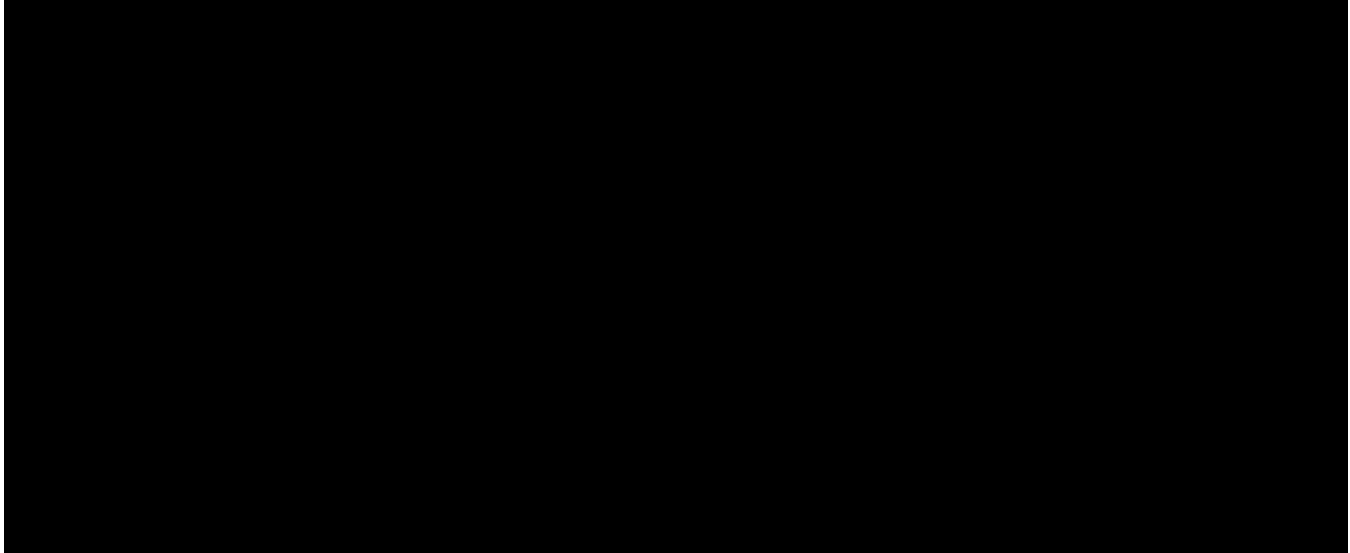
¥ **maven-dependency-plugin**

- ! Maven plugin

- ! Jars included in the "lib" folder

- ! Still need to bundle the jar and the lib folder (zip with **maven-antrun-plugin**)





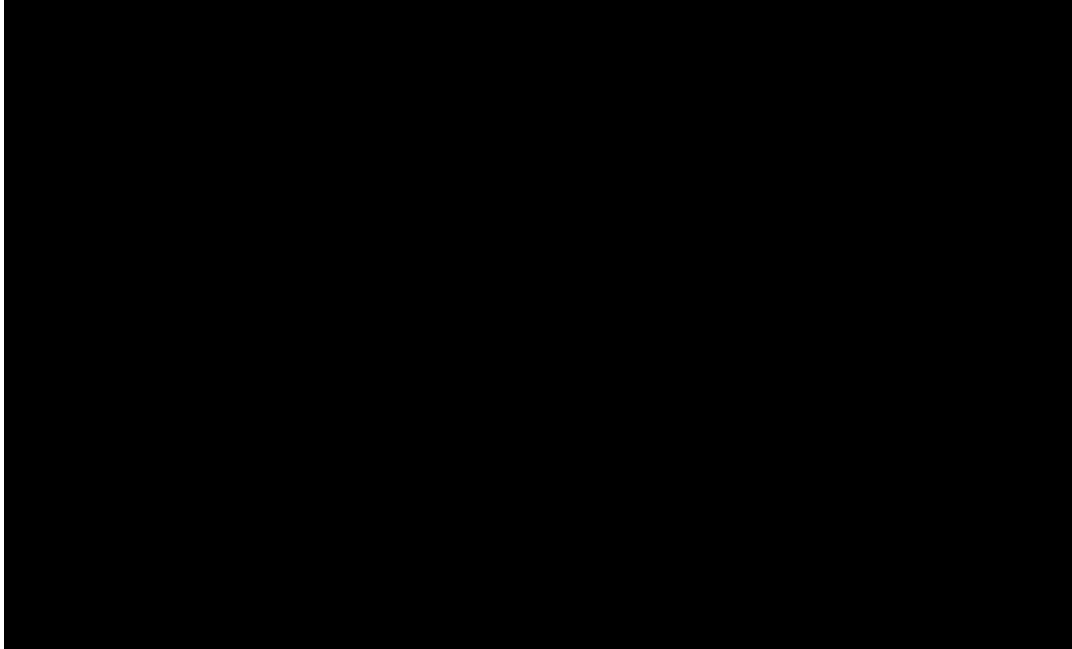
1.12. Fat Jars : Uber Jar

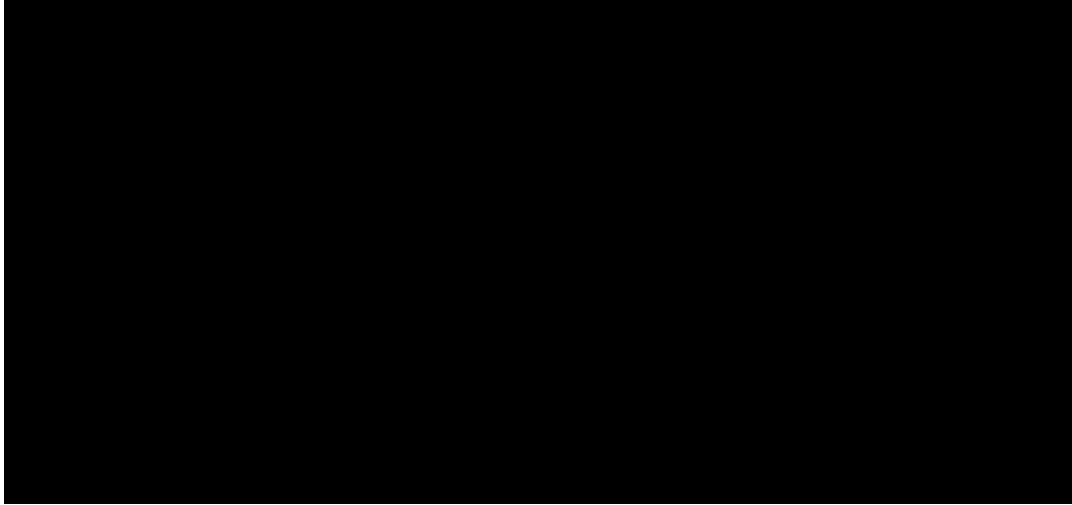
¥ `maven-assembly-plugin`

- ! Jars deflated into the same jar
- ! Can rewrite classes/resources

¥ `maven-shade-plugin`

- ! Jars deflated into the same jar
- ! Rewrites classes/resources
- ! Simpler than `maven-assembly-plugin`





1.13. Fat Jars : Jar Jar

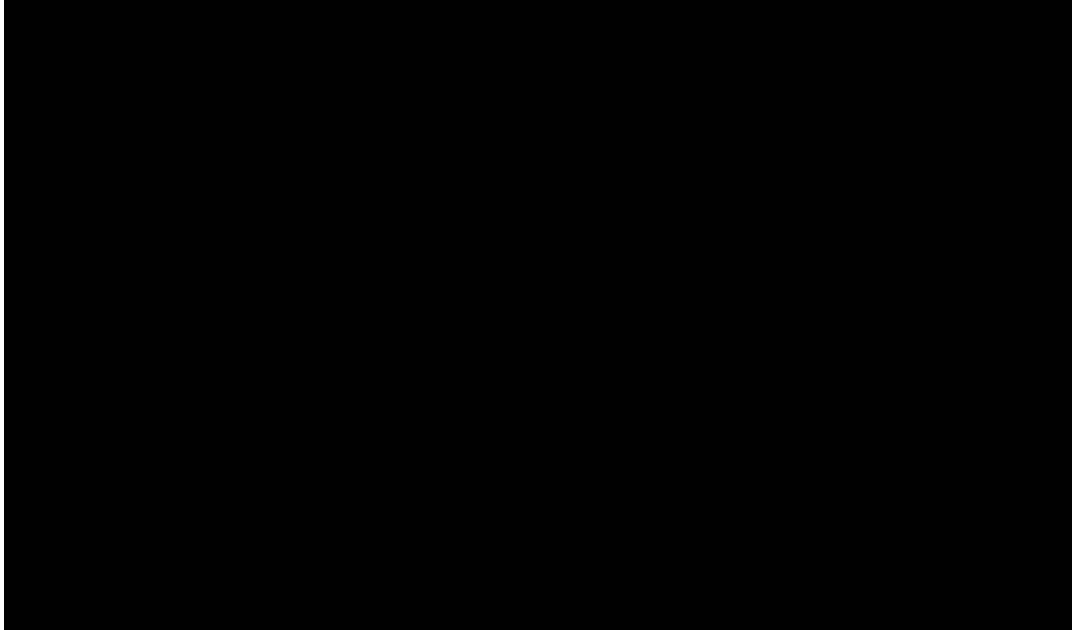
¥ onej ar-maven-pl ugi n

- ! Rewrites jar to include dependencies as jars!
- ! Adds bootstrap classes
- ! Changes classloader

¥ spr i ng-boot-maven-pl ugi n

- ! Rewrites jar to include dependencies as jars!
- ! Adds bootstrap classes
- ! Changes classloader





1.14. SoÉ

- ¥ All alternatives are poor and/or ugly
- ¥ `pom.xml` polluted with +16-20 lines of code
- ¥ ~~Why do we need a package manager for Java~~
- ¥ Why don't we already have a package manager for Java!

2. nuts Package Manager for Java

Main Idea:

- ¥ Little to no Intrusion and Backward compatibility to support existing apps and repos
- ¥ Good Integration with Java ecosystem and popular build/deploy/devops tools
- ¥ Solid enough to support multiple platforms
- ¥ Simple but extensible
- ¥ Open Source

2.1. nuts: A Package Manager for Java

¥ Centralized package manager for Java Apps and Libs (not only)

! `install`, `uninstall`, `update`, `search` and `exec` for packages

! Optimized dependency resolution solver

! Cache for dependencies across installed apps

¥ Automation/devops friendly commandline tool

¥ Portable across Architectures, OSes, OS Distibs, Desktop Environments, Platforms (Java versions)

¥ Libre and Open Source, developed in java

2.2. nuts: A Package Manager for Java

Is Not:

- ¥ a replacement for **maven**, **gradle** or any build tool (used at deploy time)
- ¥ a plugin for **maven**, **gradle** or any build tool (do not change the build process)
- ¥ a replacement for **spring** framework or any other framework
- ¥ a replacement for **IzPack** or **InstallAnywhere** (but can do pretty much of it)
- ¥ a replacement for **ansible** or **chef** (but is conceptually driven by automation)
- ¥ a mere download tool

2.3. nuts: Maven & Gradle

- ¥ Integrates seamlessly with **maven**
 - ! No required modification of the build process
 - ! Does not alter/rewrite the package
 - ! No special **maven/gradle** plugin needed
- ¥ Supports local Jars, public packages (maven central), and private packages (local .m2, nexus repos,É)
- ¥ Solves at runtime what **maven/gradle** solve at build time
 - ! Supports **maven** and **gradle** dependency resolution algorithms, scopes, É

2.4. nuts: Dependency Optimization

¥ Downloads, Caches and Installs only relevant dependencies according to

- ! **arch** (hardware architecture: x86, x64, relevant for native dependencies)
- ! **os** (operating system: Win/Linux/Mac, relevant for specific tasks)
- ! **osDist** (operating distribution : Ubuntu/OpenSuse,É)
- ! **desktop** (desktop environment, relevant for icon/shortcut creation and environment integration)
- ! **platform** (java SE versions installed to know what dependencies to use)

2.5. nuts: Integration

¥ Solid integration with environments

- ! Uses OS's File System Layouts (XDG for Linux, É)
 - " separate folders per app
 - " separate folders for log, config, lib, cache, etc.
 - " portable across OSes (~/.config versus ~/AppData)
- ! Supports cmdline and gui apps (installs scripts, icons, menus, É)
- ! Supports **j ar** and **zi p** based apps

2.6. nuts: Toolbox

- ¥ Terminal Coloring on Linux/Windows
- ¥ Supports Windows `cmd/PowerShell` and *NIX `sh`, `bash`, `csch`, `zsh` and `fish` and their relative `rcfiles`
- ¥ Bundles a `bash/GNU binutils` compatible (still incomplete) but enhanced java implementations
 - ! `ls`, `cp`, `touch`, `mkdir`, `rmdir`, `É`
 - ! works on windows
 - ! adds some extra goodies (`ssh`, `json`, support `É`)

2.7. nuts: Existing Apps

¥ Supports out of the box

! **maven** 's repos (including central, spring, google, etc), more than 7M dependencies

! Apache repos (**netbeans**, **tomcat**, **derby**, etc)

2.8. nuts: Automation

¥ Powerful toolbox with customizable output formats

! props

! xml

! json

! yaml

! table

! tree

2.9. nuts: Unique features

- ¥ Is statically built and has (almost) no dependencies
- ¥ Can be used as a library to support transitive ClassPath resolution
- ¥ Has a clean and rich API

2.10. nuts: Stability

¥ Tested:

! over 160 regression tests with 3500+ lines of test-code in the repository.

! opensuse, ubuntu, docker, windows7, windows10

! sh, bash, csh, zsh, fish

2.11. 'nuts'É really?

¥ Network Updatable Things Services

¥ The nuts (fool) companion for the maven (sage) in the Software Kingdom's Palace!

3. Demonstration

3.1. Install Nuts

1. Download `nuts.jar`
2. Run `java -jar nuts.jar -Zy`
3. Restart your terminal

3.2. Install Nuts (Linux)

¥ Install for Preview/Evaluation, most **recent**

```
$ wget https://thevpc.net/nuts/nuts-preview.jar -o nuts.jar
$ java -jar nuts.jar -Zy -r=+preview
$ exit
```

¥ Install for Production, most **stable**

```
$ wget https://repo.maven.apache.org/maven2/net/thevpc/nuts/nuts/0.8.3/nuts-0.8.3.jar
-o nuts.jar
$ java -jar nuts.jar -Zy
$ exit
```

¥ In all cases, do not forget to **restart** your terminal

3.3. Run the app

- ¥ We just run the app
- ¥ No modification is required
- ¥ We use the already built (by maven) jar
- ¥ The "artifactId" is (almost) sufficient to resolve the application to install

3.4. Demonstration : Install Application

¥ Or we can install the app

- ! All **required** dependencies will be resolved and downloaded
- ! dependencies are shared across multiple apps
- ! multiple versions of the same dependencies can coexist (required by different apps)

¥ And then we run it

3.5. Install Gui App

¥ We can run a gui app of course

¥ **nuts** will create for it

- ! a Desktop Shortcut (Icon)

- ! a Menu Item

3.6. Search for available applications

¥ We can search for installed or available (local/remote) apps

¥ We can search for apps and/or libs

3.7. Repositories

¥ We can configure Repositories used to install/update packages

¥ We can list Repositories used to install/update packages

¥ Supports

- ! Standard Maven Repositories

- ! Plain Folders

- ! Browsable HTTP folders (Parses HTML for common Webserver Directory Lists)

3.8. Integration and Formats

¥ Customize any command's output to use structured/parsable or user friendly output formats

¥ All Commands support options!

! structured (parsable) : `--j son`, `--xml`, `--props`, `--yaml`

! unstructured : `--pl ai n`, `--tabl e`, `--tree`

3.9. Companions

- ¥ We can use `nsh` instead of `bash` / `cmd`
- ¥ Implements common internal bash commands (`cd`,`É`) and constructs (`pipes`,`É`)
- ¥ Implements common `binutils` commands (`ls`,`mkdir`,`É`.)
- ¥ All commands support `json` (and `yaml` , `É`) out of the box
- ¥ All commands support `ssh` and extended path format (including URLs) out of the box, so that `cp` can be used as a simple alternative to `wget`

3.10. Bot Mode

¥ Running with `--bot` will disable all interaction and terminal coloring

3.11. Help

¥ An extensive help is available from within the command line

3.12. Conclusion

¥ **nuts** tries to be for **j** **ava** what **npm** is for javascript

¥ **nuts** is a versatile toolbox

¥ **nuts** is **2800+** classes, **600ko+** boot jar

¥ I invite you to

- ! Take a shot, try to use it and give feedback

- ! **Star(*)** the repository <https://github.com/thevpc/nuts>

- ! Spread the word

- ! Join the Core Team to enhance **nuts**

Thank you

please support us by starring our repo at

<https://github.com/thevpc/nuts> (git repo)

<https://thevpc.github.io/nuts> (website)

nuts.packagemanager@gmail.com