



# Nuts, the Java Package Manager

Version v0.8.1, 2021-07-01

## Table of Contents

1. Introduction .....	1
1.1. Introduction .....	1
1.2. Nuts and Maven .....	2
1.2.1. You'd still be Maven, yet you gonna be Nuts .....	2
1.2.2. Nuts Package manager .....	3
1.2.3. Nuts Workspaces .....	6
1.2.4. Application Framework .....	6
1.2.5. Nuts ? Really ? .....	6
1.2.6. Let's start the journey .....	6
1.2.7. System Requirements .....	8
1.2.8. Installation .....	8
1.2.9. Linux .....	9
1.2.10. MacOS .....	9
1.2.11. Windows .....	10
1.2.12. *NIX wget .....	10
1.2.13. *NIX curl .....	11
1.2.14. Any Java enabled OS .....	11
1.2.15. Test Installation .....	12
1.2.16. Run a command .....	13
2. Introduction .....	14
2.1. Change Log .....	14
2.1.1. nuts 0.8.2.0 (NEXT VERSION) .....	14
2.1.2. nuts 0.8.1.0 (DEV VERSION) .....	14
2.1.3. nuts 0.8.0.0 (PUBLISHED VERSION) .....	16
2.1.4. nuts 0.7.2.0 .....	17
2.1.5. nuts 0.7.1.0 .....	18
2.1.6. nuts 0.7.0.0 .....	18
2.1.7. nuts 0.6.0.0 .....	19
2.1.8. nuts 0.5.8.0 .....	20
2.1.9. nuts 0.5.7.0 .....	20
2.1.10. nuts 0.5.6.0 .....	21
2.1.11. nuts 0.5.5.0 .....	22
2.1.12. nuts 0.5.4.0 Change Log .....	23

2.1.13. nuts 0.5.3.0 Change Log .....	24
2.1.14. nuts 0.5.2.0 Change Log .....	26
2.1.15. nuts 0.5.1.0 Change Log .....	26
2.1.16. nuts 0.5.0.0 Change Log .....	27
2.2. Frequently Asked Questions .....	27
2.2.1. Why do we need a package manager for Java. Isn't <b>Maven</b> enough? .....	27
2.2.2. What does Nuts mean and why ? .....	27
2.2.3. Does nuts support only jar packaging .....	28
2.2.4. Can I contribute to the project .....	28
2.2.5. Where can I find Documentation about the Project .....	28
2.2.6. How can I make my application "Nuts aware" .....	29
2.2.7. Why should I consider implementing my terminal application using Nuts Application Framework (NAF) .....	29
2.2.8. Can I use NAF for non terminal applications, Swing or JavaFX perhaps .....	30
2.2.9. What is the License used in Nuts .....	30
2.3. License .....	30
2.4. Running Nuts .....	31
2.4.1. Running a deployed artifact .....	31
2.4.2. Artifact Long Ids .....	31
2.4.3. Artifact Installation .....	32
2.4.4. Multiple Artifact version Installation .....	32
2.4.5. Searching artifacts .....	33
2.4.6. Running local jar file with its dependencies .....	34
2.5. Troubleshooting .....	35
2.5.1. recover mode .....	35
2.5.2. reset mode .....	35
2.5.3. Kill mode .....	35
2.5.4. After invoking reset mode .....	36
3. Introduction .....	37
3.1. Command Line Arguments .....	37
3.1.1. Short vs Long Options .....	37
3.1.2. Option Values .....	37
3.1.3. Boolean Options .....	37
3.1.4. Combo Simple Options .....	38
3.1.5. Ignoring Options, Comments .....	38

3.1.6. Nuts Option Types .....	38
3.2. Repository Structure .....	39
3.2.1. Core Nuts projects .....	39
3.2.2. Companion tools projects .....	40
3.2.3. Toolbox projects .....	41
3.2.4. Library Projects .....	44
3.2.5. Extensions .....	45
3.2.6. Other Projects .....	45
3.2.7. Honorable mentions .....	45
4. Introduction .....	47
4.1. Aliases .....	47
4.2. Automation .....	47
4.3. File system .....	47
4.3.1. Store Locations .....	47
4.3.2. Store Location Strategies .....	49
4.3.3. Custom Store Locations .....	50
4.4. Portable Workspaces .....	51
4.5. Style Guide .....	51
4.5.1. Nuts Text Format .....	51
5. Introduction .....	55
5.1. Install Command .....	55
5.1.1. Purpose .....	55
6. Introduction .....	57
6.1. Building .....	57
6.2. Your first Nuts Application .....	57
7. Introduction .....	58

---

## Chapter 1. Introduction

This is an introduction

### 1.1. Introduction

**nuts** stands for **Network Updatable Things Services** tool. It is a simple tool for managing remote artifacts, installing these artifacts to the current machine and executing such artifacts on need. Each managed package is also called a **nuts** which is a **Network Updatable Thing Service** . **nuts** artifacts are stored into repositories. A **repository** may be local for storing local **nuts** or remote for accessing remote artifacts (good examples are remote maven repositories). It may also be a proxy repository so that remote artifacts are fetched and cached locally to save network resources.

One manages a set of repositories called a workspace. Managed **nuts** (artifacts) have descriptors that depicts dependencies between them. This dependency is seamlessly handled by **nuts** (tool) to resolve and download on-need dependencies over the wire.

**nuts** is a swiss army knife tool as it acts like (and supports) **maven** build tool to have an abstract view of the the artifacts dependency and like **npm**, **pip** or **zypper/apt-get** package manager tools to install and uninstall artifacts allowing multiple versions of the very same artifact to be installed.

**nuts** common verbs are:

- **exec** : execute an artifact or a command
- **install**, **uninstall** : install/uninstall an artifact (using its fetched/deployed installer)
- **deploy**, **undeploy** : manage artifacts (artifact installers) on the local repositories
- **update** : update an artifact (using its fetched/deployed installer)
- **fetch**, **push** : download from, upload to remote repositories
- **search** : search for existing/installable artifacts
- **welcome** : a command that does nothing but bootstrapping **nuts** and showing a welcome message.

---

## 1.2. Nuts and Maven

### 1.2.1. You'd still be Maven, yet you gonna be Nuts

Is there any package manager for Java™ applications? You can google for it and you will find that many have queried this on blogs and forums. In most cases responses point to maven and gradle, the tremendous build tools. However, both maven and gradle are build tools, while helping build packages they lack of deployment features. They bundle every dependency in every package (think of wars, ears and standalone jars). They do not handle installation or upgrading. Apache ivy, as well, while competing with maven build tool does not provide more than transitive dependency management. The main idea behind a package manager is the automation of installation, update, configuration and removal of programs or libraries in a coherent manner with the help of a database that manages binaries and metadata. maven, to consider one, sticks to the build process, and goes no further.

You may also ask, "Why ever, do we need a package manager for Java™ applications". Okay, let's take some example of Java™ applications. How can we install apache netbeans IDE ? The proper way is to browse to the editor's website, select the proper mirror if applicable, download the archive, uncompress it, chmod the main binary (i'm a linux guy) and adjust PATH environment variable to point to this binary; Quite a pain. What do we do to update it now? Hopefully, the IDE has a solid plugin architecture and an in-app update/upgrade tool that will help the process (in a gui manner of course). The same applies to eclipse and apache tomcat with the exception that apache tomcat does not even bundle an in-app update tool. The same applies too when dealing with other operating systems (Windows, MacOS, ...). Managing Java™ applications is far from helpful.

Furthermore, as Java™ applications are (usually) not bundled in OS-aware installers, you will end up with a spaghetti home directory with applications installed all over your partitions, which - simply - does not mix up with all the work OS-developers have done to separate logs from data, from temporary files, from binaries, etc. Each application will handle it's files in a very specific manner that would make it hard to manage own's disk (automatic archive/backup/restore) or roaming applications across machines, etc.

Moreover, in a world of containers and devops, deployments of Java™ applications need to be automatable and reproducible with the highest level of simplicity, configurability and integrability. Installing tomcat on a custom port should not not be as painful as using a custom Docker image or a complicated Dockerfile or even a custom apache tomcat bundle.

When we recall that Java™ is the one language that has the more versatile number of libraries, frameworks and tools, I find it annoying not to have a decent package manager to make the leap and provide facilities I find prime in other languages and platforms ( `pip/python`, `npm/nodejs/javascript`) and most of linux distribution (`zypper/opensuse`, `dnf/redhat apt-get/debian/ubuntu`)

Hence I'm introducing here a humble attempt to provide a tiny (300ko) yet powerful package manager for Java™ applications (but not only) that should handle jar files seamlessly (with little or no modification) and that comes with a set of portable tools that makes this management at a higher level. I'm not talking about redefining the wheel. I'm aware that many tools such as maven, are already very good at what they do, I just needed to make the leap for deployments. You will be able to deploy your applications without bundling all of their dependencies : `nuts` will take care of that.

So you'd still be maven, yet you gonna be `nuts`.

### 1.2.2. Nuts Package manager

`nuts` is actually :

- a transitive dependency resolution manager
- package manager (backports maven and supports maven repositories)
- automation tool
- feature rich toolset
- application framework
- decentralized

- sandbox based

#### 1.2.2.1. Transitive dependency resolution manager

**nuts** calculates transitive dependencies of an application to resolve other packages to download at install or update/upgrade time. So typically, deployed applications should no more bundle their dependencies within the deployed archive. Thus we avoid the annoying fat jars (using maven plugins like 'maven-assembly-plugin' and 'maven-shade-plugin') and lib folders (using 'maven-dependency-plugin'). It will also reuse dependencies and packages across multiple installed applications and hence save disk space, and network bandwidth.

All what **nuts** needs is a descriptor file withing the jar file that defines the immediate dependencies. It then calculates all transitive dependencies automatically. And guess what, all maven built jars already contain that descriptor : the pom.xml file. So basically all maven applications are already **nuts** aware applications.

#### 1.2.2.2. Package manager

**nuts** uses this dependency resolution to help install, update, remove and search for applications. To be able to use an application, it has to be installed and configured with all of its dependencies. This is the main goal of **nuts**. When we ask to install tomcat, for instance, it will search for the best version in registered repositories, download it, and configure it to be ready for execution. The best version is not always the the latest one. Actually it would be the latest valid one, thus the latest one that matches some constraints. Constraints include the version of the running java (tomcat 8 works on java 7 but not 6 for instance), the running operating system (windows, linux, ... to help selecting the proper binaries), may be the hardware architecture or even the operating distribution (for linux based systems). Constraints will filter the search result to include the best, the most accurate version to install. Installation also would configure the installed application and even may run another artifact to help this configuration.

**nuts** also handles search for newer versions and update the installed application at request. Updating a software does not necessarily delete the older version. Both version can coexist and it is up to the user the decide whether or not to retain both versions.



Indeed, one of the key features of **nuts** is the ability to install and hence run multiple versions of the same software in parallel. You would never see an error message telling you can't install that software because a dependency of it is installed with different version. All software and all libraries can coexist peacefully.

Software artifacts are stored in repositories. **nuts** can handle multiple repositories, remote and local ones. Installed software are stored in special local repositories. Supported repositories include maven repositories and github repositories. Actually a fresh installation of **nuts** is configured with maven central repository so that, you already have access to thousands of installable artifacts.

At some point, you may need to uninstall an artifact and that's to undo the artifact installation. Installation will help you choose between uninstalling binaries only and keeping data/config files or remove permanently all of the artifact files. In all ways, uninstalling will not affect other artifacts that use the same dependencies if ever.

#### 1.2.2.3. Feature rich Toolset

**nuts** is intended to be used either by human users or by robots and other applications. It comes with portable, feature rich toolset, a versatile library and a handy parsable result.

**nuts** is mainly a commandline program that helps installing, uninstalling, searching, updating and running artifacts. To help desktop integration, **nuts** installs by default a set of other companion tools such as **nsh** (a portable bash-compatible implementation), and **nadmin** (an administration tool for **nuts** to configure users, authorizations, repositories, create scripts,...);

**nsh** brings the bash facilities to all environments (windows included) in a very portable manner. Besides it integrates well with the installed **nuts** version. Several common commands are ported to **nsh** such as **cat**, **head**, and **ssh**, as well as core features like pipes, redirection and scripts.

**nadmin** is intended for configuring **nuts** workspaces, managing repositories and users. It helps also configuring sub commands and aliases to make **nuts** usage even easier.

### 1.2.3. Nuts Workspaces

One of the key features of **nuts** is the ability to support multiple isolated workspaces, each managing its own repositories, applications and libraries; each defining its sandbox security constraints. Thus non-root installation is made easy while it remains possible to overlap between workspaces by sharing repositories. Roaming is also supported, so that a workspaces can be copied/moved across machines.

### 1.2.4. Application Framework

**nuts** can also be embedded as a library in your application. This enables you to wire classes on the fly by its network dependency-aware classloading mechanisms. The library allows as well building solid and well integrated applications, mainly console applications. Indeed, **nuts** comes with rich outputs that support automatic formatting to json, xml, table, tree and plain texts. It handles standard File Systems layouts; XDG Base Directory Specification is implemented for linux and MacOS. A compatible one is also implemented in Windows systems. And of course, it helps seamlessly install, update and remove events.

### 1.2.5. Nuts ? Really ?

In every palace you will find the wizard and the fool, the **maven** and the **nuts**; There's no exception in the java kingdom! If you do prefer acronyms here is another reason : **nuts** stands for Network Updatable Things Services. It should be able to facilitate things deployment and update over the wire where things resolve here to any piece of software depending (or not) on other piece of software.

### 1.2.6. Let's start the journey

we start by opening a new terminal (termm, konsole or whatever you prefer) then download **nuts** using this command : On linux/MacOS system we issue :

```
wget https://github.com/thevpc/vpc-public-  
maven/raw/master/net/vpc/app/nuts/nuts/0.8.1/nuts-0.8.1.jar
```

Let's check that java is installed :

```
java --version
```

Now we run **nuts**

```
java -jar nuts-0.8.1.jar -zy
```

We used the flags **-y** to auto-confirm and **-z** to ignore cached binaries (combined here as **-zy**). These flags are not required. We use them here to make installation work in all cases. Installation may last several minutes as it will download all required dependencies, companions and tools.

You should then see this message

```
Welcome to nuts. Yeah, it is working...
```

**nuts** is well installed, just restart your terminal.

Now we will install apache tomcat. So in your terminal type:

```
nuts install ntomcat  
nuts ntomcat start --set-port 9090
```

The first two commands will install tomcat helper tool (ntomcat) that will download latest version of tomcat and configure it to 9090 port. The last command will start tomcat. Let's check tomcat status now

```
nuts tomcat status
```

Now we will do the same with derby database. We will install it and run it.

```
nuts install nderby  
nuts nderby start
```

As you can see, simple commands are all you need to download, install, configure and run tomcat or derby or any application that is deployed in the maven repository.

So please visit **nuts** [website](https://thevpc.github.com/nuts) or [github repository](https://github.com/thevpc/nuts) for more information. === Installation

### 1.2.7. System Requirements

Here are all **nuts** requirements :

- **Java** : **nuts** requires a valid Java Runtime Environment (JRE) or Java Development Kit (JDK) version 8 or above to execute.
- **System Memory**: **nuts** memory footprint is very little and has no minimum RAM requirements.
- **Disk**: 2.5Mo on the disk are required for the **nuts** installation itself. In addition to that, additional disk space will be used for your local Nuts workspace. The size of your local workspace will vary depending on usage but expect at least 500MB.
- **Operating System**: **nuts** is able to run on any java enabled Operating System including all recent versions of Windows, Linux and MacOS.

To check if you have a valid java installation type

```
java -version
```

The result would be equivalent to the following. Just be sure the version is 1.8 or over. In this example, the java version is 1.8.0\_211

```
$> java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

### 1.2.8. Installation

### 1.2.9. Linux

#### using curl:

```
NDVER=0.8.1 && curl -OL https://repo.maven.apache.org/maven2\
/net/thevpc/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -zy && . ~/.bashrc
```

#### using wget:

```
NDVER=0.8.1 && wget -https://repo.maven.apache.org/maven2\
/net/thevpc/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -zy && ~/.bashrc
```

#### Using helper script:

```
curl -OL -http://thevpc.net/nuts-install.sh && . ./nuts-install.sh
```

Linux Systems installation is based on bash shell. First launch will configure "~/.bashrc" so that **nuts** and other companion tool commands will be available in any future terminal instances. Using **nuts** on unix-like system should be seamless. A simple bash terminal (Gnome Terminal, KDE Konsole,...) is already a nuts-aware terminal.

All Linux versions and distributions should work with or without XWindow (or equivalent). Graphical system is required only if you plan to run a gui application using **nuts**. All tests where performed on OpenSuse Tumbleweed.

#### TIP:

Any bash terminal application is a nuts-aware terminal.

### 1.2.10. MacOS

```
NDVER=0.8.1 && curl -OL https://github.com/thevpc/vpc-public-maven/raw/master\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

MacOS Systems installation is based on **bash** shell. First launch will configure "~/.bashrc" so that **nuts** and other companion tool commands will be available in any future terminal instances. Using **nuts** on MacOS system should be seamless. A simple bash terminal (MacOs Terminal App) is already a nuts-aware terminal.

TIP:

Any bash terminal application is a nuts-aware terminal.

### 1.2.11. Windows

download [nuts-0.8.1.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.0.8.1/nuts-0.8.1.jar>)

```
java -jar -y nuts-0.8.1.jar
```

On Windows systems, first launch will create a new **nuts** Menu (under Programs) and a couple of Desktop shortcuts to launch a configured command terminal.

- **nuts-cmd-0.8.1** : this shortcut will open a configured command terminal. **nuts** command will be available as well as several nuts companion tools installed by **nadmin** by default
- **nuts-cmd** : this shortcut will point to the last installed **nuts** version, here 0.8.1

Any of these shortcuts will launch a nuts-aware terminal.

Supported Windows systems include Window 7 and later.

TIP:

Any of the created shortcuts for windows is a nuts-aware terminal.

### 1.2.12. \*NIX wget

```
NDVER=0.8.1 && rm -f nuts-$NDVER.jar && wget https://github.com/thevpc/\
vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar &&\
java -jar nuts-$NDVER.jar -y
```

TIP:

Any bash terminal application is a nuts-aware terminal.

### 1.2.13. \*NIX curl

```
NDVER=0.8.1 && curl -OL https://github.com/thevpc/vpc-public-maven/raw/master/\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

TIP:

Any bash terminal application is a nuts-aware terminal.

### 1.2.14. Any Java enabled OS

```
NDVER=0.8.1 && curl -OL https://github.com/thevpc/vpc-public-maven/raw/master/\
/net/vpc/app/nuts/nuts/$NDVER/nuts-$NDVER.jar && java -jar \
nuts-$NDVER.jar -y
```

TIP:

Any bash terminal application is a nuts-aware terminal.

You should then see some log like the following :

[illegible]

As you can see, installation upon first launch, will also trigger installation of other optional programs called "companion tools". Actually they are recommended helpful tools :

- **nsh** which stands for **Nuts Shell** , a bash compatible shell implementation program that will run equally on linux an windows systems.
- **nadmin** an administration tool for **nuts**

IMPORTANT:

After installation, you need to restart the terminal application for the configuration to take effect.

### 1.2.15. Test Installation

To test installation the simplest way is to open a nuts-aware terminal and type :

```
nuts --version
```



It should show a result in the format : nuts-api-version/nuts-impl-version

```
0.8.1/##:version0:0.8.1.0##
```

### 1.2.16. Run a command

To run a command using **nuts** just type

```
nuts <command>
```

Several commands are available, and you can always manually run any java and non java application. More info is available in the **nuts** official [wiki](<https://github.com/thevpc/nuts/wiki>) .

## Chapter 2. Introduction

This is an introduction

### 2.1. Change Log

View Official releases [here](<https://github.com/thevpc/nuts/releases>) : Starred releases are most stable ones.

#### 2.1.1. nuts 0.8.2.0 (NEXT VERSION)

- 2021/XX/XX      nuts      0.8.2.0      (\*)      released      [download      nuts-0.8.2.jar](<https://repo.maven.apache.org/maven2/net/thevpc/nuts/nuts/0.8.2/nuts-0.8.2.jar>)
- WARNING: API has evolved with multiple incompatibilities with previous versions
- TODO : {api} add comments support in 'element' and and propagate support for xml,json and so on
- TODO : {impl} better yaml parser
- TODO : {impl} implement yaml formatter (both print as raw and as colored)
- TODO : {impl} bundle tson parser

#### 2.1.2. nuts 0.8.1.0 (DEV VERSION)

- 2020/11/22      nuts      0.8.1.0      (\*)      released      [download      nuts-0.8.1.jar](<https://repo.maven.apache.org/maven2/net/thevpc/nuts/nuts/0.8.1/nuts-0.8.1.jar>)
- WARNING: API has evolved with multiple incompatibilities with previous versions
- CHANGED: {api} removed NutsApplicationLifeCycle and replaced with NutsApplication (an interface instead of a class)
- ADDED : {api} added support for parsing pom.xml (MAVEN) along with **.nuts (nuts descriptors)**

- ADDED : {api} added io killProcess support
- CHANGED: {api} added path API, implemented via nlib-ssh to add ssh support for paths
- CHANGED: {all} remove dependencies, runtime has no dependencies, and others have the bare minimum
- CHANGED: {api} session is from now on mandatory to perform any operation. A simple way to make it simple to use is to get a "session aware" workspace with `session.getWorkspace()`
- ADDED : {api} added support for Yaml with minimal implementation
- ADDED : {api} element now supports complex keys in Map Entries (Objects)
- ADDED : {api}{cmdline} added support for History and implemented in JLine extension
- ADDED : {api}{cmdline} added support for readline syntax coloring (using jline)
- ADDED : {api}{cmdline} added `--locale` option to support multi languages. The option is reflected to Session as well
- ADDED : {api}{cmdline} added `--key=value` options to support extra properties
- ADDED : {api}{cmdline} added `-S` short option, equivalent to `--standalone`
- ADDED : {api}{cmdline} added `NutsFormattedMessage` to support formatted messages in a uniform manner (C-style, {} positional)
- CHANGED: {api}{cmdline} both list and tree dependencies are now accessible as `NutsDependencies`
- ADDED : {runtime} added support to community maven repositories : jcenter, jboss, spring, clojars, atlassian, atlassian-snapshot, google, oracle to use the repository you can add it as a permanent repository or temporary. here are some examples:
  - `nuts nadmin add repository jcenter // add permanently the repository`
  - `nuts -r jcenter my-command // use temporarily the repository to run my-command`

- FIXED : {runtime} extension support (for JLine)
- ADDED : {runtime} added minimal implementation for YAM
- ADDED : {runtime} added fast implementation for JSON and removed gson dependency
- CHANGED: {runtime} revamped Nuts Text Format to support simplified syntax but more verbose styles. Now supports `), ), )` and so on as Title Nodes. It supports as well the common markdown 'code' format with anti-quotes such as ``code goes here...``. Other supported examples are: ``some command...`` ``error message...`` ``someKeyword``
- CHANGED: {runtime} help files now have extensions ".ntf" (for nuts text format) instead of ".help"
- ADDED : {njob} added --help sub-command
- FIXED : {nsh} fixed multiple inconsistencies and implemented a brand new parser
- REMOVED: {docusaurus-to-ascidoctor} tool fully removed as replaced by a more mature ndocusaurus
- REMOVED: {ndi}, removed project, merged into nadmin
- REMOVED: {nded}, removed project, temporarily code added to nadmin, needs to be refactored
- ADDED : {ntalk-agent} new modules nlib-talk-agent (library) and ntalk-agent (application using the library) that enable client to client communication. nlib-talk-agent is a broker that helps communication between nuts components with minimum overhead. nlib-talk-agent enables one workspace to talk with any other workspace without having to create one server socket for each workspace. It also enables singleton per location implementation

### 2.1.3. nuts 0.8.0.0 (PUBLISHED VERSION)

- 2020/11/8?      nuts      0.8.0.0      (\*)      released      [download      nuts-0.8.0.jar](<https://repo.maven.apache.org/maven2/net/thevpc/nuts/nuts/0.8.0/nuts-0.8.0.jar>)

- WARNING: this is the first version to be deployed to maven central. previous versions will no longer be supported
- WARNING: this is a **major version**, API has evolved with multiple incompatibilities with previous versions
- WARNING: The OSS License has changed from GPL3 to the more permissive Apache Licence v2.0
- CHANGED: changed packages from net.vpc to net.thevpc (required for central to be aligned with website)
- CHANGED: removed support for vpc-public-maven and vpc-public-nuts
- CHANGED: ``-Z`` will update `.bashrc` file and switch back to default workspace
- ADDED : when a dependency is missing it will be shown in the error message
- ADDED : nuts commandline argument `--N` (`--expire`) to force reloading invoked artifacts (expire fetched jars). a related `NutsSession.expireTime` is introduced to force reinstall of any launched application and its dependencies, example: ``-N ndi``
- ADDED : `install --strategy=install|reinstall|require|repair` introduced to select install strategy (or sub command)
- ADDED : `NutsInput` & `NutsOutput` to help considering reusable sources/targets
- ADDED : nuts commandline argument `--skip-errors` to ignore unsupported commandline args
- ADDED : new toolbox `njob`, to track service jobs (how many hours you are working on each service project)
- ADDED : new `next-term`, to support jline console extension into nuts
- ADDED : `workspace.str()` to create `NutsStringBuilder`
- ADDED : `'switch'` command in `ndi` to support switching from one workspace to another. example : ``switch -w other-workspace -a 0.8.0``

#### 2.1.4. nuts 0.7.2.0



this version is not deployed to maven-central

- 2020/09/23 nuts 0.7.2.0 (\*) released [download nuts-0.7.2.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.7.2/nuts-0.7.2.jar>)
- FIXED : execute non installed artifacts sometimes do not ask for confirmation
- ADDED : NutsCommandLineProcessor.prepare/exec/autoComplete
- ADDED : NutsApplicationContext.processCommandLine(cmdLine)
- ADDED : NutsApplicationContext.configureLast(cmdLine)
- RENAMED: feenoo renamed to ncode
- ADDED : Docusaurus Website
- ADDED : new toolbox ndocusaurus : Docusaurus Website templating

#### 2.1.5. nuts 0.7.1.0



this version is not deployed to maven-central

- 2020/09/14 nuts 0.7.1.0 (\*) released [download nuts-0.7.1.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.7.1/nuts-0.7.1.jar>)
- FIXED : reset stdout line when calling external processes
- FIXED : fixed several display issues.

#### 2.1.6. nuts 0.7.0.0



this version is not deployed to maven-central

- 2020/07/26 nuts 0.7.0.0 (\*) released [download nuts-0.7.0.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.7.0/nuts-0.7.0.jar>)
- ADDED : NutsApplicationContext.processCommandLine(c)
- ADDED : NutsWorkspaceCommand.copySession()
- RENAMED: derby renamed to nderby

- RENAMED: mysql renamed to nmysql
- RENAMED: tomcat renamed to ntomcat
- RENAMED: mvn renamed to nmvn

### 2.1.7. nuts 0.6.0.0



this version is not deployed to maven-central

- 2020/01/15 nuts 0.6.0.0 (\*) released [download nuts-0.6.0.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.6.0/nuts-0.6.0.jar>)
- CHANGED : config file format changed
- CHANGED : now installed packages are stored in 'installed' meta repository
- CHANGED : alias files have extension changed from **.njc** to **.cmd-alias.json**
- CHANGED : now nuts looks for system env variable NUTS\_WORKSPACE for default workspace location
- CHANGED : api and runtime are installed by default
- CHANGED : now distinguishes between installed primary and installed dependencies packages.
- ADDED : support for ROOT\_CMD execution (SYSCALL was renamed USER\_CMD)
- ADDED : support for Interrupting Copy
- ADDED : support to ps (list processes)
- ADDED : support progress options
- CHANGED : worky, searches now for modified deployments with same version but different content
- FIXED : encoding problem with json/xml
- REMOVED : NutsRepositorySession

### 2.1.8. nuts 0.5.8.0



this version is not deployed to maven-central

- 2019/09/02 nuts 0.5.8.0 (\*) released [download nuts-0.5.8.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.7/nuts-0.5.7.jar>)
- ADDED : support for Custom Monitor in Copy Command
- ADDED : support to javaw for windows (exec command supports --javaw or --win flag)
- ADDED : support to workspace custom logging (with support for colouring)
- ADDED : support to userProperties per repository
- ADDED : NutsString and NutsStringFormat to support 'Nuts Stream Format'
- ADDED : NutsWorkspaceAware to support initialize/dispose of NutsComponents
- ADDED : I/O Delete action
- ADDED : I/O Lock action
- ADDED : I/O Compress and Uncompress actions
- CHANGE : now if a command to execute ends with '!', we will force searching in installed only.
- CHANGE : removed install/uninstall in Terminal, replaced by NutsWorkspaceAware

### 2.1.9. nuts 0.5.7.0



this version is not deployed to maven-central

- 2019/07/23 nuts 0.5.7.0 (\*) released [download nuts-0.5.7.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.7/nuts-0.5.7.jar>)
- ADDED : support to Windows (Tested on Win 7) and MacOS(Not Tested) of Desktop Integration
- ADDED : added session and Nuts(Add/Update/Remove)Options where applicable
- ADDED : Initial support for uri based workspaces



- ADDED : --dry option to help dry-run commands (test execution without side effects)
- ADDED : NutsApplication getShared\*Folder() method for configuration shared between versions
- ADDED : flags (in Definition and search) : api, runtime, extension, companion
- CHANGED : Improved compatibility with Maven
- CHANGED : Improved Documentation (still too much to go though)
- CHANGED : Changed NutsCommandLine main api to simplify boot time implementations
- CHANGED : Renamed NutsEffectiveUser → NutsUser
- CHANGED : Renamed NutsRight → NutsPermission (and all subsequent methods)
- CHANGED : NutsExtensionInfo → NutsExtensionInformation
- CHANGED : NutsHttpConnectionFacade → NutsHttpConnection
- CHANGED : Added java.io.Serializable anchor when applicable
- REMOVED :  
NutsDefaultRepositoriesProvider, NutsSingletonClassLoaderProvider, NutsDefaultClassLoaderProvider, NutsWorkspaceSPI
- REMOVED : NutsRepositoryListener.onInstall(...)
- REMOVED : 'alternative' concept, and added NutsClassifierMapping so that classifier can be resolved according to env

#### 2.1.10. nuts 0.5.6.0



this version is not deployed to maven-central

- 2019/06/23 nuts 0.5.6.0 released [download nuts-0.5.6.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.6/nuts-0.5.6.jar>)
- ADDED : Implements XDG Base Directory Specification
- ADDED : Added Json Path support

- ADDED : Added NutsQuestionParser and NutsQuestionFormat
- CHANGED : Extensions are loaded by boot instead of impl so that one can change default impl behaviour
- CHANGED : All repositories are now cache aware.
- CHANGED : Refactored **Format to extends the very same interface.**
- CHANGED : Using to java.time package instead of older Date class
- CHANGED : Improved Documentation (still to much to go though)
- CHANGED : Prefer https repository urls
- FIXED : Fixed several issues
- REMOVED : [CommandLine] IMMEDIATE
- REMOVED : [Options] --term
- REMOVED : [Extensions] add/remove extensions from extension manager (should use install/uninstall commands)

#### 2.1.11. nuts 0.5.5.0



this version is not deployed to maven-central

- 2019/06/08 nuts 0.5.5.0 released [download nuts-0.5.5.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.5/nuts-0.5.5.jar>)
- REMOVED : Removed Nsh commands Console Deploy Info Install Fetch Uninstall,Push Update Exec Which
- REMOVED : Removed maven-github repository type support (web API)
- REMOVED : Removed nuts-cmd-app project dependency. A built-in NutsApplication is included in the api the help simplify extension.
- ADDED : Added support for XML, TABLE and TREE (along with JSON, PROPS and PLAIN) printing format to help automate result parsing
- ADDED : Added Better api in Nuts IO to handle SHA and MD5

- ADDED : json and xml nsh commands to help manipulating json and xml in commands outputs
- FIXED : Fixed fprintf issue with "" (empty string)
- FIXED : Fixed Update indexes/stats command
- FIXED : When installing nuts, lookup latest core implementation
- CHANGED : Renamed FindCommand to SearchCommand (and some of their methods too)
- CHANGED : NutsIdFilter.accept accepts workspace as a second argument
- CHANGED : Improved Help text
- CHANGED : Improved Documentation (still to much to go through)
- ADDED : (nsh) Builtin nsh commands basename and dirname
- CHANGED : (nsh) Builtin nsh command who renamed to whoami
- REMOVED : (nfind) Removed nfind companion (the built-in search command is a better replacement)

#### 2.1.12. nuts 0.5.4.0 Change Log



this version is not deployed to maven-central

- 2019/04/21 nuts 0.5.4.0 (\*) released [download nuts-0.5.4.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.4/nuts-0.5.4.jar>)

-Added lucene indexing facility (thanks to the excellent work of nasreddine bac ali) \*  
Removed dependencies to common,strings, io and utils (utility classes).

- Removed dependencies to asm (bytecode manipulation).
- From Now on only gson and jansi are retained.
- Layout changes
  - from now on configuration will be version specific. some migration should be done to import previous configs

- system (global) repo is no more created under the workspace. Only a link to it is registered in nuts-workspace.json
- added MacOS Layout. Help is needed for testing this !
- Better support for JDK 8+ (New IO, Predicates, Streams, ...)
- Added Comprehensive implementation of Iterator (Stream Like) to better handle result iteration while search is in progress
- Speed improvements
- Added JUnit test battery
- Added support to JSON, PROPS and PLAIN result, implemented in version and info. Should continue implementing in other commands.
- Removed --license, --update, --install, ... options, replaced by workspace "internal" commands new concept.
- Workspaces handle several type of executables that will be resolved in that order : "internal command", "aliases : aka workspace command aliases", "components", "path/unmanaged components" and system/native commands.
- Several Fixes
  - Fixed Problem with Layout
  - Fixed Problem coloring (fprint embedded library)
  - All System properties now start with "nuts."
  - System properties starting with "nuts.export." are exported to children processes
  - Added watch dog against infinite child process creation

### 2.1.13. nuts 0.5.3.0 Change Log



this version is not deployed to maven-central

- 2019/01/05 nuts 0.5.3.0 released [download nuts-0.5.3.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.3/nuts-0.5.3.jar>)

- (WINDOWS) First support to Windows platform
  - Support for Console coloring on windows
  - Storing to AppData\Local and AppData\Roaming folders
  - ndi is not yet supported!
- (LINUX,UNIX) ndi no more stores to ~/bin but instead it updates .bashrc to point to current workspace added a confirmation question.
- API Change
  - Moved getStoreRoot from NutsWorkspace to NutsWorkspaceConfigManager
  - Added StoreType : CACHE,LIB
  - Introduced  
NutsDeploymentBuilder,NutsIoManager,NutsParseManager,NutsFormatManager,  
DescriptorFormat
  - Introduced NutsSessionTerminal,NutsSystemTerminal
  - Added description, alternative (to support multi architecture nuts) descriptor properties
  - Removed descriptor/id 'ext' and 'file' parameters. 'packaging' should be more than enough
  - Removed Maps from config. Replaced by plain arrays
  - Removed workspace.cwd
  - Removed Temp File/Folder support
- Added Archetype "standalone" to help bundling and application with all its dependencies
- Several fixes
  - Fixed Log configuration, introduced --log-inherited to enable inherited log-handlers
  - Fixed support for install/uninstall hooks

- Fixed Repository Layout where ref repo folder is created twice
- Fixed Multiple pom download issue
- Fixed Gson parsing issue
- Fixed autocomplete support
- Fixed bad json format recovery
- nsh
  - introduced pwd,set unset,alias,unalias,autocomplete commands
  - fixed support to autocomplete
- TODO
  - Code Comments
  - Help files

#### 2.1.14. nuts 0.5.2.0 Change Log



this version is not deployed to maven-central

- 2018/12/28 nuts 0.5.2.0 released [download nuts-0.5.2.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.2/nuts-0.5.2.jar>)
- Global refactoring
  - Introduced NutsCommandExecBuilder, NutsDependencyBuilder, NutsDeploymentBuilder, NutsIdBuilder, NutsClassLoaderBuilder
- Extracted nsh commands as regular nuts package (nadmin, nfind) WORKING-ON :  
Fixing "mvn" start from nuts (handling, exclude, pom import and classifiers from maven)

#### 2.1.15. nuts 0.5.1.0 Change Log



this version is not deployed to maven-central

- 2018/12/18 nuts 0.5.1.0 released [download nuts-0.5.1.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.1/nuts-0.5.1.jar>)
- FIXED : Fixed problem with inheritIO from child process (added InputStreamTransparentAdapter and OutputStreamTransparentAdapter interfaces)
- FIXED : Added distinction between workspace config and runtime boot api/runtime values
- FIXED : Do not read workspace version and dependency config from child process (because it may require distinct version of nuts)
- FIXED : Mkdir,cp, etc... used incorrectly cwd. Fixed.
- CHANGED : Optimized pom.xml parse execution time (using DOM instead of SAX)
- CHANGED : moved cache from bootstrap folder to default-workspace/cache

#### 2.1.16. nuts 0.5.0.0 Change Log



this version is not deployed to maven-central

- 2018/11/25 nuts 0.5.0.0 released [download nuts-0.5.0.jar](<https://github.com/thevpc/vpc-public-maven/raw/master/net/vpc/app/nuts/nuts/0.5.0/nuts-0.5.0.jar>)
- Very first published version. older ones were used internally for internal projects only.

## 2.2. Frequently Asked Questions

### 2.2.1. Why do we need a package manager for Java. Isn't Maven enough?

Please read [Nuts Introduction, Why and What for](../intro/nuts-and-maven.md). In few words maven manages dependencies to build applications, nuts uses maven dependencies system to install applications.

### 2.2.2. What does Nuts mean and why ?

**nuts** stands for "Network Updatable Things Services". It helps managing things (artifacts

of any type, not only java). The Name also helps depicting another idea : **nuts** is a good companion and complement to Maven tool. The word maven (MAY-vin), from Yiddish, means a super-enthusiastic expert/fan/connoisseur/Wizard. And where wizards are, fools and **nuts** must be.

**nuts** is the foolish tool to support the deployment and not the build. Hence the name.

### 2.2.3. Does nuts support only jar packaging

Not only. **nuts** supports all packagings supported by maven. This includes pom , jar , maven-plugin , ejb , war , ear , rar. However **nuts** is also intended to support any "thing" including "exe" , "dll" , "so" , "zip" files, etc.

**nuts** differs from maven as it defines other properties to the artifact descriptor (aka pom in maven) : os (operating system), arch (hardware architecture), osdist (relevant for linux for instance : opensuse, ubuntu) and platform (relevant to vm platforms like java vm, dotnet clr, etc). Such properties are queried to download the most appropriate binaries for the the current environment.

### 2.2.4. Can I contribute to the project

I hoped you would ask this question. Of course. You can drop me an email (see my github profile email) to add you as contributor or fork the repository and ping a pull request. You can also open a new issue for feature implementation to invite any other contributor to implement that feature (or even implement it your self).

### 2.2.5. Where can I find Documentation about the Project

Mainly all of the documentation can be found in 2 places:

- this website: it includes both user documentation and javadocs (code documentation)
- each command help option. when you type

```
nuts --help
```



```
or
```

```
nsh --help
```

```
you will get more details on nuts or on the tool (here nsh)
```

### 2.2.6. How can I make my application "Nuts aware"

If by **nuts** aware you mean that you would download your application and run it using **nuts**, then you just need to create the application using maven and deploy your application to the public maven central. Nothing really special is to be done from your side. You do not have to use plugins like 'maven-assembly-plugin' and 'maven-shade-plugin' to include your dependencies. Or, you can also use NAF (**nuts** Application Framework) make your application full featured "Nuts aware" application.

### 2.2.7. Why should I consider implementing my terminal application using Nuts Application Framework (NAF)

First of all, NAF is a simple 300k jar so for what it provided to you, you would be surprised. Indeed, implementing your application using NAF will provide you a clean way to :

- seamless integration with **nuts** and all other NAF applications (obviously!)
- support standard file system layout (XDG) where config files and log files are not necessarily in the same folder see [Nuts File System](../advanced/filesystem.md) for more details.
- support application life cycle events (onInstall, onUninstall, onUpgrade),
- standard support of command line arguments
- dynamic dependency aware class loading
- terminal coloring, and terminal components (progress bar, etc...)
- json,xml,table,tree and plain format support out of the box as output to all your

---

## commands

- pipe manipulation when calling sub processes
- advanced io features (persistence Locks, monitored downloads, compression, file hashing....)
- standard ways to support and use installed platforms (installed JRE, JDK, ...)
- and lots more...

### 2.2.8. Can I use NAF for non terminal applications, Swing or JavaFX perhaps

Sure, you will be able to benefit of all the items in the preceding question but terminal coloring wont be relevant of course. Check netbeans-launcher in github. It's a good example of how interesting is to use NAF in non terminal applications.

### 2.2.9. What is the License used in Nuts

**nuts** is published under Licensed under the Apache License, Version 2.0.

## 2.3. License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Copyright © 2016-2020 thevpc

## 2.4. Running Nuts

### 2.4.1. Running a deployed artifact

You can run any jar using **nuts** as far as the jar is accessible from one of the supported repositories. By default, **nuts** supports:

- maven central
- local maven folder (~/.m2) You can configure other repositories or even implement your own if you need to.

The jar will be parsed to check from maven descriptor so that dependencies will be resolved and downloaded on the fly. Then, all executable classes (public with static void main method) are enumerated. You can actually run any of them when prompted. Any jar built using maven should be well described and can be run using its artifact long id.

### 2.4.2. Artifact Long Ids

**nuts** long ids are a string representation of a unique identifier of the artifact. It has the following form :

```
groupId:artifactId#version
```

for instance, to install **netbeans-launcher** (which is a simple UI helping launch of multiple instances of netbeans), you can issue

```
nuts net.vpc.app:netbeans-launcher#1.2.2
```

You do agree that this can be of some cumbersome to type. So you can simplify it to :

```
nuts netbeans-launcher
```

In this form, **nuts** will auto-detect both the **groupId** and the **version**. The group id is detected if it is already imported (we will see later import a groupId). By default, there is

a couple of groupIds that are automatically imported :

- `.vpc.app` (contains various applications of the author)
- `.vpc.nuts.toolbox` (contains various companion tools of `nuts`, such as `nsh`, `nadmin`, ...)

And it turns out, hopefully, that netbeans-launcher belongs to an imported groupId, so we can omit it. Besides, if no version is provided, `nuts` will also auto-detect the best version to execute. If the application is already installed, the version you choose to install will be resolved. If you have not installed any, the most recent version, obviously, will be detected for you.

### 2.4.3. Artifact Installation

Any java application can run using `nuts` but it has to be installed first. If you try to run the application before installing it, you will be prompted to confirm installation. To install our favorite application here we could have issued :

```
nuts install netbeans-launcher
```

But as we have tried to run the application first, it has been installed for us (after confirmation).

### 2.4.4. Multiple Artifact version Installation

One of the key features of `nuts` is the ability to install multiple versions of the same application. We can for instance type :

```
nuts install netbeans-launcher#1.2.2
# then
nuts install netbeans-launcher#1.2.0
```

Now we have two versions installed, the last one always is considered default one. you can run either, using it's version

```
nuts netbeans-launcher#1.2.2 &
# or
nuts netbeans-launcher#1.2.0 &
```

Actually, when you have many versions installed for the same artifact and you try to run it without specifying the version, the last one installed will be considered. To be more precise, an artifact has a default version when it is installed. This default version is considered when no explicit version is typed. In our example, when we type

```
nuts netbeans-launcher &
```

the 1.2.0 version will be invoked because the artifact is already installed and the default version points to the last one installed. So if you want to switch back to version 1.2.2 you just have to re-install it. Don't worry, no file will be downloaded again, nuts will detect that the version is not marked as default and will switch it to.

### 2.4.5. Searching artifacts

Now let's take a look at installed artifacts. We will type :

```
nuts search --installed
```

This will list all installed artifacts. We can get a better listing using long format :

```
nuts search --installed -l
```

you will see something like

```
I-X 2019-08-21 04:54:22.951 anonymous vpc-public-maven net.vpc.app:netbeans-
launcher#1.2.0
i-X 2019-08-21 04:54:05.196 anonymous vpc-public-maven net.vpc.app:netbeans-
launcher#1.2.2
```

The first column here is the artifact status that helps getting zipped information of the

artifact. the 'T' stands for 'installed and default' whereas, 'i' is simply 'installed'. The 'X' stands for 'executable application', where 'x' is simply 'executable'. Roughly said, executable applications are executables aware of (or depends on) **nuts**, as they provide a special api that helps nuts to get more information and more features for the application. As an example, executable applications have special OnInstall and OnUninstall hooks called by nuts. The second and the third columns are date and time of installation. The fourth column points to the installation user. When Secure mode has not been enabled (which is the default), you are running nuts as 'anonymous'. The fifth column shows the repository from which the package was installed. And the last column depicts the artifact long id.

#### 2.4.6. Running local jar file with its dependencies

Let's suppose that my-app.jar is a maven created jar (contains META-INF/maven files) with a number of dependencies. **nuts** is able to download on the fly needed dependencies, detect the Main class (no need for MANIFEST.MF) and run the application. If a Main-Class Attribute was detected in a valid MANIFEST.MF, it will be considered. If more than one class is detected with a main method, **nuts** will ask for the current class to run.

When you run a local file, **nuts** will behave as if the app is installed (in the given path, an no need to invoke install command). Local files are detected if they are denoted by a valid path (containing '/' or " depending on the underlying operating system). Dependencies will be downloaded as well (and cached in the workspace)

```
nuts ./my-app.jar some-argument-of-my-app
```

If you need to pass JVM arguments you have to prefix them with "--exec". So if you want to fix maximum heap size use

```
nuts --exe -Xms1G -Xmx2G ./my-app.jar argument-1 argument-2
```

---

## 2.5. Troubleshooting

Whenever installation fails, it is more likely there is a mis-configuration or invalid libraries bundles used. You may have to options to circumvent this which are two levels or workspace reinitialization.

### 2.5.1. recover mode

**recover mode** will apply best efforts to correct configuration without losing them. It will delete all cached data and libraries for them to be downloaded later and searches for a valid nuts installation binaries to run (it will actually do a forced update). To run nuts in recover mode type :

```
nuts --recover
```

### 2.5.2. reset mode

**reset mode** will apply all efforts to correct configuration by, actually, **deleting** them (and all of workspace files!!) to create a new fresh workspace. This is quite a radical action to run. Do not ever invoke this unless your are really knowing what you are doing. To run nuts in reset mode type :

```
nuts --reset
```

### 2.5.3. Kill mode

**Kill mode** is a special variant of reset mode where workspace will not be recreated after deletion. This can be achieved by using a combination of reset mode and `--skip-boot (-Q)`option. Do not ever invoke it unless your are really knowing what you are doing. To run nuts in reset mode type :

To run nuts in prune mode type :

```
nuts --reset -Q
```

---

#### 2.5.4. After invoking reset mode

After invoking reset mode, nuts commands (installed by nadmin) will not be available anymore. you should use the jar base invocation at least once to reinstall these commands.

```
java -jar nuts-0.5.7.jar
```



---

## Chapter 3. Introduction

This is an introduction

### 3.1. Command Line Arguments

**nuts** supports a specific format for command line arguments. This format is the format supported in **nuts** Application Framework (NAF) and as such all NAF applications support the same command line arguments format. Arguments in **nuts** can be options or non options. Options always start with dash (-).

#### 3.1.1. Short vs Long Options

Options can be long options (starts with double dash) or short options (start with a single dash). Many arguments support both forms. For instance "-w" and "--workspace" are the supported forms to define the workspace location in the nuts command.

#### 3.1.2. Option Values

Options can also support a value of type string or boolean. The value can be suffixed to the option while separated with '=' sign or immediately after the option. As an example "-w=/myfolder/myworkspace" and "--workspace /myfolder/myworkspace" are equivalent.

#### 3.1.3. Boolean Options

Particularly, when the value is a boolean, the value do not need to be defined. As a result "--skip-companions" and "--skip-companions=true" are equivalent. However "--skip-companions true" is not (because the option is of type boolean) and "true" will be parsed as a NonOption.

To define a "false" value to the boolean option we can either suffix with "=false" or prefix with "!" or "~" sign. Hence, "--skip-companions=false", "--!skip-companions" and "--~skip-companions" are all equivalent.

---

### 3.1.4. Combo Simple Options

Simple options can be grouped in a single word. "-ls" is equivalent to "-l -s". So one should be careful. One exception though. For portability reasons, "-version" is considered a single short option.

### 3.1.5. Ignoring Options, Comments

Options starting with "--/" and "--/" are simply ignored by the command line parser.

### 3.1.6. Nuts Option Types

Options in **nuts** are regrouped in multiple categories. An option can belong to multiple categories though.

- Create Options : such options are only relevant when creating a new workspace. They define the configuration of the workspace to create. They will be ignored when the workspace already exists. Examples include
  - --skip-companions
  - --archetype
  - --store-strategy
  - --standalone
- Open Options : such options are relevant when creating a new workspace or when opening an existing workspace. They define the way commands are executed. Examples include
  - --workspace
  - --bot
  - --reset
- Exported Options : are passed to sub-**nuts**-processes that will be created by **nuts**. For instance when nuts will call the **nsh** command it will spawn a new process. In such case, these options are passed to the sub-process as environment variable.

- `--workspace`
- `--bot`
- `--no-color`
- **Application Options** : are options that are by default supported by Applications using NAF (Nuts Application Framework) (as well as Nuts it self).
  - `--help`
  - `--version` all **nuts** options are described in the command help. Just type :

```
nuts --help
```

### 3.2. Repository Structure

**nuts** repository is composed of several projects that can be organized in 5 categories

- **Core nuts** : These projects are the core/base of the **nuts** package manager
- **Companion Tools** : These projects are applications and tools to install with **nuts** itself. Their installation are prompted at first install of **nuts**
- **Toolbox** : These projects are applications and tools built on top of **nuts** Application Framework and are of common interest
- **Lib** : These projects are common libraries that can be used to enabled some **nuts** features in your application
- **Extension** : These projects are add features to the nuts platform. on example is the ability to add JLine library support to have smarter terminals.
- **Other** : All other apps that doe no fit in the previous categories

#### 3.2.1. Core Nuts projects

Core **nuts** projects include **nuts-builder**, **nuts-api** (/core/nuts), **nuts-runtime** (/core/nuts-runtime).

---

#### 3.2.1.1. **nuts-builder**

**nuts-builder** is a meta project (parent maven pom project) that helps building all the other projects.

#### 3.2.1.2. **nuts-api**

**nuts-api** is the effective "nuts" only required dependency. It defines the bootstrap application that is responsible of loading all necessary libraries for its execution.

**nuts-api** starts to load **nuts-runtime** which is responsible of implementing all features and interfaces declared by the **nuts-api** library. That implementation will handle further download, version dependency etc. Such architecture is considered to force loose coupling with nuts binaries.

**nuts-api** is a very thin bootstrapper : its size is about 300k. It can be used as a standalone application or as an embedded library.

#### 3.2.1.3. **nuts-runtime**

**nuts-runtime** is the effective and standard implementation of **nuts-api**.

**nuts-runtime** has a faster update pace than **nuts-api**. It focuses on performance and compliance to the **nuts** specifications declared by **nuts-api** interfaces. You are not required to add this dependency to your application if you want to embed **nuts**. The library will be loaded on the wire (if not yet present in the classpath of course).

**nuts-runtime** is designed to have very few dependencies : **gson** and **jansi**.

- **gson** trivially is used to support json serialization : the main format used in **nuts** to support configuration and descriptors.
- **jansi** is used to support terminal coloring and the "Nuts Text Format" (NTF), a simple text format (markdown like) that helps creating colorful terminal applications.

### 3.2.2. Companion tools projects

Companion tools include **nadmin** and **nsh**. These two applications are implemented

following the "**nuts** Application Framework" and hence are dependent on **nuts-api** library. They are recommended applications to install with **nuts** itself, however they are not mandatory and may be ignored particularly when using **nuts-api** as library.

#### 3.2.2.1. **nadmin**

**nadmin** (for **nuts** admin) is an administration tool to manage the **nuts** workspaces. It adds support for managing users, credentials, authorizations, workspaces and repositories by providing command line support for such actions.

**nadmin** adds also seamless integration of **nuts** commands in your favorite operating system and environment by creating scripts that point to your favorite workspaces and applications.

**nadmin** is responsible of creating script shortcuts to your common commands so that you can invoke tem directly from your environment. For instance it creates an "nadmin" script and configures your PATH environment to help calling the nuts admin tool instead of the common way to do so "nuts nadmin". On window system, **nadmin** will create shortcuts and menus.

#### 3.2.2.2. **nsh**

**nsh** (for **nuts** shell) is simply a portable POSIX bash compatible implementation. It supports all common builtin commands (ls, cd, rm, ...) and adds support to grep, ssh and scp in a seamless manner. It also supports command line, scripts (including commons constructs with if, do, case, ...) and pipes (|) and common bash syntax.

### 3.2.3. **Toolbox projects**

**nuts** comes with an array of tools out of the box you can install and play with. Here are some of them:

#### 3.2.3.1. **nversion**

**nversion** is a small tool that helps detecting files versions. It supports jar, war, ear, dll and exe file versions. It opens a file and looks for it's version in its meta-data.

---

### 3.2.3.2. **nmysql**

**nmysql** is a companion tool to the mysql and mariadb servers. The initial actions supported are backup and restore including push/pull mechanism from/to a couple of databases for synchronization. It supports jdbc and ssh based access to remote mysql/mariadb installation.

### 3.2.3.3. **ntomcat**

**ntomcat** is a companion tool to the tomcat http server. The initial actions supported are start, stop, status, configure (http ports etc..) and deploy. It supports as well installation of several versions of Tomcat and multi domain configuration for deployments.

### 3.2.3.4. **nderby**

**nderby** is a companion tool to the derby database server. The initial actions supported are start, stop, status and configure. It supports as well installation of several versions of Derby.

### 3.2.3.5. **nmvn**

**nmvn** is a companion tool to maven. It supports installations of several versions of it and running them seamlessly.

### 3.2.3.6. **noapi**

**noapi** (for Nuts OpenApi) is an OpenAPI documentation generator.

### 3.2.3.7. **ncode**

**ncode** is a small code search tool. It searches for files, files contents and classes within jars. You can search for files that contains some text or jars that contain some class, or jars of a specific version of java.

### 3.2.3.8. **nwork**

**worky** is a developer centered tool. The 'y' in **worky** refers to 'my' in the "Tunisian

Dialect" and hence means "my work". **Worky** is the tool we - maven users - need to check if the version of project we are working on is yet to be deployed to nexus or not. So basically it checks if the version is the same, and downloads the server's version and then compares binaries to local project's to check if we have missed to update the version in our pom.xml. I know I'm not the only one having pain with jar deployments to nexus. **Worky** does other things as well to help me on on daily basis.

#### 3.2.3.9. ntemplate

**ntemplate** is a file templating tool that replaces place-holders in the files with an evaluated expression.

#### 3.2.3.10. njob

**njob** is powerful terminal task manager

#### 3.2.3.11. ndoc

**ndoc** is a javadoc generator. It supports standard format and adds markdown format.

#### 3.2.3.12. ndocusaurus

**ndocusaurus** is a [Docusaurus 2](<https://docusaurus.io>) toolbox that adds several features to the initial tool:

- adds templating (using ntemplate)
- adds pdf generation

#### 3.2.3.13. ntalk-agent

**ntalk-agent** is a client-to-client communication broker

#### 3.2.3.14. nclown

**nclown** is an angular web application frontend for **nuts**. It helps navigating, searching and installing artifacts. It is intended to be a web admin tool as well.

---

#### 3.2.3.15. **nserver**

**nserver** is a standalone application that runs a small http server that will expose a workspace as a remote repository to other **nuts** installations. This is the simplest way to mirror a workspace and share artifacts between networked nodes.

#### 3.2.3.16. **nwar**

**nwar** (for **nuts** Web Application Archive) is a web application that exposes **nserver** as a war to be deployed on a more mature http server or container.

#### 3.2.3.17. **ndexer**

**ndexer** (for Indexer) is a lucene powered index for **nuts**. It can be shared across multiple **nuts** workspaces and processes.

### 3.2.4. Library Projects

Library projects are several libraries that add **nuts** support in a particular environment or domain.

#### 3.2.4.1. **nlib-tomcat-classloader**

This is a must-have feature in your web application if deployed on Tomcat. It solves the following problem : a simple war application is surprisingly fat with too many jars (hundreds of Megs) you need to upload each time you change a single file or class in your web project. Basically all the jars included in the lib folder of the war are to be uploaded each time to the remote Tomcat server. The common solution is to use "provided" scope in maven and put your jars in Tomcat lib or ext folders. This is a bad approach if you are using a single Tomcat process for multiple applications. **nuts-tomcat-classloader** simply uses **nuts** to download libraries when the application is deployed based on the **pom.xml** you provide and include them in the current web application class loader. Hence, the war becomes lighter than ever. **nuts** cache mechanisms optimizes bandwidth and makes this more convenient by sharing the same jar files between applications depending on the same versions. All you have to do is to add this library to your application and configure your **pom.xml** accordingly.



#### 3.2.4.2. **nlib-servlet**

Basically this is the simplest way to include **nserver** into your web application.

#### 3.2.4.3. **nlib-template**

This library provides helper methods to manipulate maven pom.xml and generate simple Java files while supporting **nuts** concepts. It is used in other tools that are meant to generate maven projects.

#### 3.2.4.4. **nlib-talkagent**

This library provides support for client to client communication

### 3.2.5. **Extensions**

Extensions provide extra feature to nuts.

#### 3.2.5.1. **next-term**

This library provides rich terminal support (auto-complete, history) based on the JLine library

### 3.2.6. **Other Projects**

Other projects you may encounter in the repository are WIP projects that may be continued or discontinued. This includes : **nutsc** (a native c bootstrapper) and **nuts-installer** (a **nuts** installer tool)

### 3.2.7. **Honorable mentions**

Although not included in this Git repository some other tools are based on **nuts** and hence are installable using `install the-app` command. Those tools are published in other repositories.

**3.2.7.1. netbeans-launcher** : this tool supports installation and launch of multiple netbeans instances in parallel. See [Netbeans Launcher GitHub Repository](<https://github.com/thevpc/netbeans-launcher>)

**3.2.7.2. upa-box** : this tool supports creation of UPA aware projects. UPA is a non structured ORM for the Java Language. See [Netbeans Launcher GitHub Repository](<https://github.com/thevpc/upa>)

**3.2.7.3. vr-box** : this tool supports creation of VR aware projects. VR is a web portal framework. See [Netbeans Launcher GitHub Repository](<https://github.com/thevpc/vr>)

---

## Chapter 4. Introduction

This is an introduction

### 4.1. Aliases

### 4.2. Automation

### 4.3. File system

**nuts** manages multiple workspaces. It has a default one located at `~/.config/nuts` (`~` is the user home directory). Each workspace handles a database and files related to the installed applications. The workspace has a specific layout to store different types of files relatives to your applications. **nuts** is largely inspired by [XDG Base Directory Specification](<https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html>) and hence defines several store locations for each file type. Such organization of folders is called Layout and is dependent on the current operating system, the layout strategy and any custom configuration.

#### 4.3.1. Store Locations

Supported Store Locations are :

**nuts** File System defines the following folders :

- **config** : defines the base directory relative to which application specific configuration files should be stored.
- **apps** : defines the base directory relative to which application executable binaries should be stored
- **lib** : defines the base directory relative to which application non executable binaries should be stored
- **var** : defines the base directory relative to which application specific data files (other than config) should be stored

- 
- **log** : defines the base directory relative to which application specific log and trace files should be stored
  - **temp** : defines the base directory relative to which application specific temporary files should be stored
  - **cache** : defines the base directory relative to which application non-essential data and binary files should be stored to optimize bandwidth or performance
  - **run** : defines the base directory relative to which application-specific non-essential runtime files and other file objects (such as sockets, named pipes, ...) should be stored

**nuts** defines such distinct folders (named Store Locations) for storing different types of application data according to your operating system.

On Windows Systems the default locations are :

- apps : "\$HOME/AppData/Roaming/nuts/apps"
- lib : "\$HOME/AppData/Roaming/nuts/lib"
- config : "\$HOME/AppData/Roaming/nuts/config"
- var : "\$HOME/AppData/Roaming/nuts/var"
- log : "\$HOME/AppData/Roaming/nuts/log"
- temp : "\$HOME/AppData/Local/nuts/temp"
- cache : "\$HOME/AppData/Local/nuts/cache"
- run : "\$HOME/AppData/Local/nuts/run" On Linux, Unix, MacOS and any POSIX System the default locations are :
- config : "\$HOME/.config/nuts"
- apps : "\$HOME/.local/share/nuts/apps"
- lib : "\$HOME/.local/share/nuts/lib"
- var : "\$HOME/.local/share/nuts/var"
- log : "\$HOME/.local/log/nuts"
- cache : "\$HOME/.cache/nuts"

- temp : "\$java.io.tmpdir/\$username/nuts"
- run : "/run/user/\$USER\_ID/nuts" As an example, the configuration folder for the artifact net.vpc.app:netbeans-launcher#1.2.4 in the default workspace in a Linux environment is

```
home/me/.config/nuts/default-workspace/config/id/net/vpc/app/netbeans-launcher/1.2.4/
```

And the log file "app.log" for the same artifact in the workspace named "personal" in a Windows environment is located at

```
C:/Users/me/AppData/Roaming/nuts/log/nuts/personal/config/id/net/vpc/app/netbeans-launcher/1.2.4/app.log
```

### 4.3.2. Store Location Strategies

When you install any application using the **nuts** command a set of specific folders for the presented Store Locations are created. For that, two strategies exist : **Exploded strategy** (the default) and **Standalone strategy**.

In **Exploded strategy** **nuts** defines top level folders (in linux ~/.config for config Store Location etc), and then creates withing each top level Store Location a sub folder for the given application (or application version to be more specific). This helps putting all your config files in a SSD partition for instance and make **nuts** run faster. However if you are interested in the backup or roaming of your workspace, this may be not the best approach.

The **Standalone strategy** is indeed provided mainly for Roaming workspaces that can be shared, copied, moved to other locations. A single root folder will contain all of the Store Locations.

As an example, in "Standalone Strategy", the configuration folder for the artifact net.vpc.app:netbeans-launcher#1.2.4 in the default workspace in a Linux environment is

```
home/me/.config/nuts/default-workspace/config/id/net/vpc/app/netbeans-
launcher/1.2.4/
```

And the log file "app.log" for the same artifact in the workspace named "personal" in the same Linux environment is located at

```
/home/me/.config/nuts/default-workspace/log/id/net/vpc/app/netbeans-
launcher/1.2.4/
```

You can see here that the following folder will contain ALL the data files of the workspace.

```
/home/me/.config/nuts/default-workspace
```

whereas in the **Exploded strategy** the Store Location are "exploded" into multiple root folders.

### 4.3.3. Custom Store Locations

Of course, your able to configure separately each Store Location to meet your needs.

#### 4.3.3.1. Selecting strategies

The following command will create an exploded workspace

```
nuts -w my-workspace --exploded
```

The following comman will create an standalone workspace

```
nuts -w my-workspace --standalone
```

#### 4.3.3.2. Finer Customization

The following command will create an exploded workspace and moves all config files to the SSD partition folder /myssd/myconfig

```
nuts -w my-workspace --system-config-home=/mysdd/myconfig
```

You can type help for more details.

```
nuts help
```

## 4.4. Portable Workspaces

## 4.5. Style Guide

```

      ^- ^-
     / \ / \ _ _ / / _ _ _ _
    /  \ / / / / / _ / _ _ /
   /  ^ \ / / _ / / / _ ( _ )
  \ _ \ \ / \ _ _ / \ _ / _ _ /
                                     version v0.8.1

```

### 4.5.1. Nuts Text Format

**nuts** comes up with a simple coloring syntax that helps writing better looking portable command line programs. standard output is automatically configured to accept the "Nuts Text Format" (NTF) syntax. Though it remains possible to disable this ability using the `--no-color` standard option (or programmatically, see **nuts** API documentation). NTF will be translated to the underlying terminal implementation using ANSI escape code on linux/windows terminals if available.

Here after a showcase of available NTF syntax.

type (nuts help --colors) to display this help

#### NTF special characters:

```
# 0 \ ` are special characters
: { and } are also special character when used inside # styles
# defines primary styles
0 is a 'nop' character. it is used as a separator when required. It is not displayed.
`Text` defines verbatim text
``lang code-bloc`` defines formatted code in the given language/format (such as sh, java, json, ...)
\ You can escape special characters using \ character
```

#### NTF COLORS:

nuts supports 4 types of colors :

```
*/ primary (foreground) and secondary (background) colors are defined by themes and are customizable
syntax foreground : [#]+Text[#]+ or ##:p<0-15>: Text##
syntax background : ##:s<0-15>: Text##
*/ 4bit colors (0-15) define a predefined color palette of 16 colors as defined by ANSI terminal formats
syntax foreground : ##:f<0-15>: Text##
syntax background : ##:b<0-15>: Text##
*/ 8bit colors (0-255) define a predefined color palette of 256 colors as defined by ANSI terminal formats
syntax foreground : ##:f<0-255>: Text##
syntax background : ##:b<0-255>: Text##
*/ 24bit colors (r-g-b) define a predefined color palette of 16M colors as defined by ANSI terminal formats
syntax foreground : ##:fx<hexa-6-digits>: Text##
syntax background : ##:bx<hexa-6-digits>: Text##
```

Format	Display	Description
Text #Text#	Text #Text#	plain text
##Text##	Text	primary1 = title 1
###Text###	Text	primary2 = title 2
####Text####	Text	primary3 = title 3
#####Text#####	Text	primary4 = title 4
#####Text#####	Text	primary5 = title 5
#####Text#####	Text	primary6 = title 6
#####Text#####	Text	primary7 = title 7
#####Text#####	Text	primary8 = title 8
#####Text#####	Text	primary9 = title 9
##:1:Text## ##:p1:Text## ##:s1:Text##	Text Text Text	primary 1
##:2:Text## ##:p2:Text## ##:s2:Text##	Text Text Text	primary 2
##:3:Text## ##:p3:Text## ##:s3:Text##	Text Text Text	primary 3
##:4:Text## ##:p4:Text## ##:s4:Text##	Text Text Text	primary 4
##:5:Text## ##:p5:Text## ##:s5:Text##	Text Text Text	primary 5
##:6:Text## ##:p6:Text## ##:s6:Text##	Text Text Text	primary 6
##:7:Text## ##:p7:Text## ##:s7:Text##	Text Text Text	primary 7
##:8:Text## ##:p8:Text## ##:s8:Text##	Text Text Text	primary 8
##:9:Text## ##:p9:Text## ##:s9:Text##	Text Text Text	primary 9
##:10:Text## ##:p10:Text## ##:s10:Text##	Text Text Text	primary 10
##:11:Text## ##:p11:Text## ##:s11:Text##	Text Text Text	primary 11
##:12:Text## ##:p12:Text## ##:s12:Text##	Text Text Text	primary 12
##:13:Text## ##:p13:Text## ##:s13:Text##	Text Text Text	primary 13
##:14:Text## ##:p14:Text## ##:s14:Text##	Text Text Text	primary 14
##:15:Text## ##:p15:Text## ##:s15:Text##	Text Text Text	primary 15
##:/:Text## ##:_:Text## ##:%:Text##	Text Text Text	italic, underlined, blink
##:!:Text## ##:+:Text## ##:-:Text##	Text Text Text	reversed, bold, striked
##:primary3:Text##	Text	primary3
##:secondary5:Text##	Text	secondary5
##:{s12:AA##:12:BB##0##:6:CC##DD}##	AABBCCDD	composed colors, note the 0 separator
##:f158:AA## ##:f58:BB## ##:f201:CC##	AA BB CC	foreground 8bits colors
##:foreground158:Text##	Text	foreground 158 (8bits)
##:fxd787af:Text##	Text	foreground Pink (d787af in 24bits)
##:foregroundxd787af:Text##	Text	foreground Pink (d787af in 24bits)
##:b158:Text##	Text	background 158 (8bits)
##:bxd787af:Text##	:bxd787af:Text	background Pink (24bits)



**NTF SPECIAL CHARACTERS**

NTF uses special characters as escape format. characters have different meanings in different situations, but you will be able to escape any character using \. You can also escape a whole text using ```

Format	Display	Description
\\#\` ``` Text with # escaped```	\#ø Text with # escaped	escaped characters escaped text (note the starting space)

**NTF STYLES:**

Supported styles in NTF are defined in the following table:

Format	Display	Description
```underlined underlined```	<u>underlined</u>	underlined
```italic italic```	<i>italic</i>	italic
```striked striked```	<del>striked</del>	striked
```reversed reversed```	reversed	reversed
```error error``` ```warn warn``` ...	error warn info config comments string number boolean keyword option user_input operator separator success danger fail var pale	several token types
...`		
##:underlined underlined##	<u>underlined</u>	underlined
##:italic italic##	<i>italic</i>	italic
##:striked striked##	<del>striked</del>	striked
##:reversed reversed##	reversed	reversed
##:error error## ```warn warn## ...	error warn info config comments string number boolean keyword option user_input operator separator success danger fail var pale	several embeddable token types
...`		

**NTF TITLES:**

Tables are special formatted lines and are usable simply by suffix # chars with a parentheses at the start of any new line.

```

Title 1
#) Title 1
Title 2
##) Title 2
Title 3
###) Title 3
Title 4
####) Title 4
Title 5
#####) Title 5
Title 6
#####) Title 6
Title 7
#####) Title 7
Title 8
#####) Title 8
Title 9
#####) Title 9

```

**NTF SYNTAX COLORING:**

NTF supports as well a advanced features that allow you to syntax coloring popular formats. Especially Xml, Json, Bash and Java are supported.

```

Xml format
NTF_syntax
NTF_syntax
```xml <xml n='value'></xml>```
Coloring Result
<xml n='value'></xml>

Json format
NTF_syntax
```json {k:'value',n:'value'}```
Coloring Result
{k:'value',n:'value'}

Java format
NTF_syntax
```java
public class A{
    int a=12;
}
```
Coloring Result
public class A{
    int a=12;
}

Shell commandline format
NTF_syntax
```sh cmd arg user -option=true```
Coloring Result
cmd arg user -option=true

```

## Chapter 5. Introduction

This is an introduction

### 5.1. Install Command

#### 5.1.1. Purpose

The install command is used to install or reinstall packages.

- A+B : read A main package and B dependencies
- A+B? : ask, if confirmed, read A main package and B dependencies.
- require : deploy package as 'required'
- install : deploy package as 'installed'
- nothing : do nothing The available strategies are
- require : install the package and all of its dependencies as required class installed package
- install : install the package and all of its dependencies as first class installed package
- reinstall : re-install or re-required the package and all of its dependencies
- repair : repair (re-install or re-required) the given dependency "required class installed package" can be removed (uninstalled automatically by nuts when none of the depending package is anymore installed.

| Status/Strategy<br>→ Status | REQUIRE               | INSTALL               | REINSTALL             | REPAIR                |
|-----------------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| NOT_INSTALLED               | REQUIRED              | INSTALLED             | INSTALLED?            | ERROR                 |
| INSTALLED                   | INSTALLED<br>REQUIRED | INSTALLED?            | INSTALLED             | INSTALLED             |
| INSTALLED<br>REQUIRED       | INSTALLED<br>REQUIRED | INSTALLED<br>REQUIRED | INSTALLED<br>REQUIRED | INSTALLED<br>REQUIRED |

|                                   |                                   |                       |                       |                       |
|-----------------------------------|-----------------------------------|-----------------------|-----------------------|-----------------------|
| REQUIRED                          | REQUIRED                          | INSTALLED<br>REQUIRED | REQUIRED              | REQUIRED              |
| INSTALLED<br>OBSOLETE             | INSTALLED<br>REQUIRED<br>OBSOLETE | INSTALLED             | INSTALLED             | INSTALLED             |
| INSTALLED<br>REQUIRED<br>OBSOLETE | INSTALLED<br>REQUIRED             | INSTALLED<br>REQUIRED | INSTALLED<br>REQUIRED | INSTALLED<br>REQUIRED |
| REQUIRED<br>OBSOLETE              | REQUIRED<br>OBSOLETE              | INSTALLED<br>REQUIRED | REQUIRED              | REQUIRED              |

| Status/Strategy<br>→ action       | REQUIRE         | INSTALL          | REINSTALL        | REPAIR          |
|-----------------------------------|-----------------|------------------|------------------|-----------------|
| NOT_INSTALLED                     | require+require | install+require  | install+require? | error           |
| INSTALLED                         | nothing+nothing | install+require? | install+require  | install+nothing |
| INSTALLED<br>REQUIRED             | nothing+nothing | install+require? | install+require  | install+nothing |
| REQUIRED                          | nothing+nothing | install+nothing  | require+require  | require+nothing |
| INSTALLED<br>OBSOLETE             | install+require | install+require  | install+require  | install+nothing |
| INSTALLED<br>REQUIRED<br>OBSOLETE | install+require | install+require  | install+require  | install+nothing |
| REQUIRED<br>OBSOLETE              | require+require | install+require  | require+require  | require+nothing |

## Chapter 6. Introduction

This is an introduction

### 6.1. Building

To build **nuts** from sources == Nuts Applications

### 6.2. Your first Nuts Application

---

## Chapter 7. Introduction

This is an introduction