nuts, the Java Package Manager
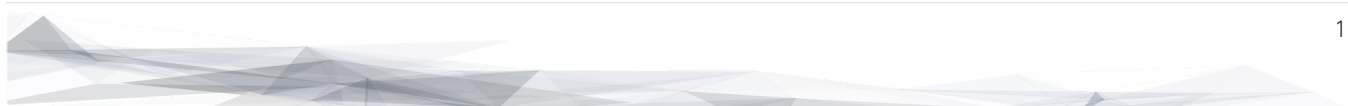

https://thevpc.github.io/nuts

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

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

nuts.packagemanager@gmail.com
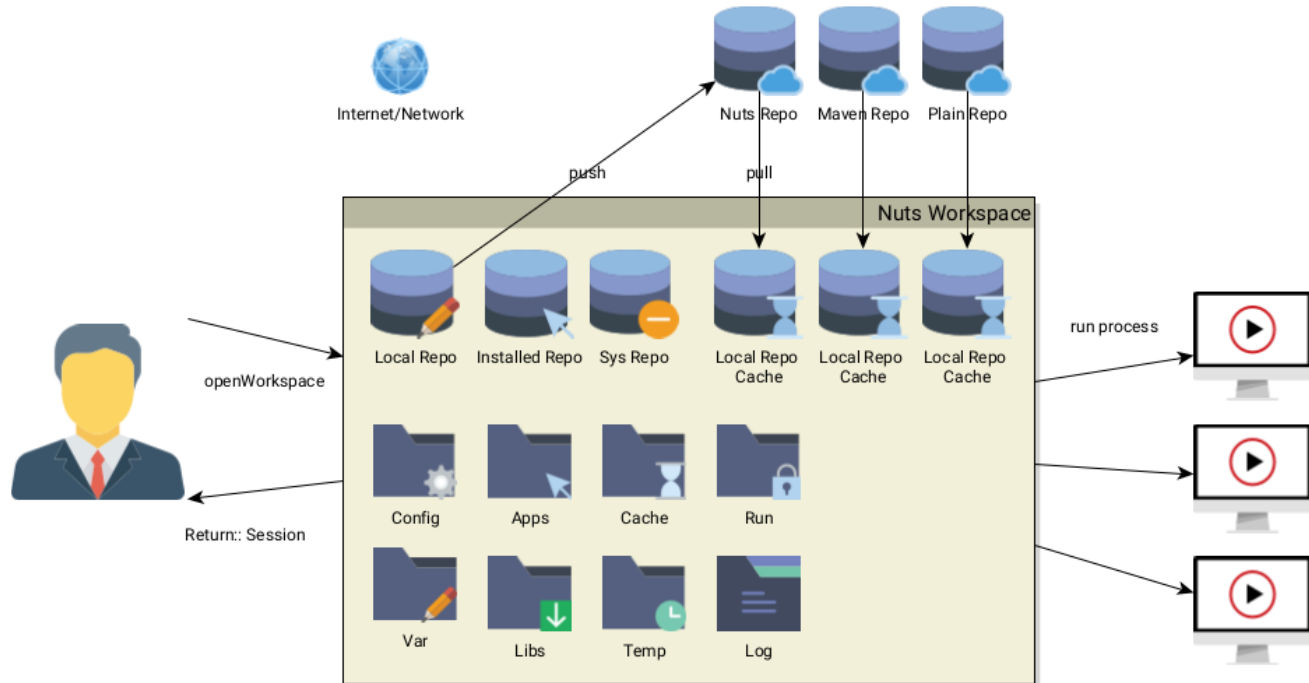
thevpc, 2025-01-04

Plan

1. Architecture

2. API

3. Demo

1. Architecture

2. Nuts Commandline

- SYNOPSYS

```
nuts [options]
```

- Options can be long or short

```
nuts -S
nuts --standalone
```

- Options order is (generally) irrelevant
- Most relevant option :)

```
nuts --help
```

3. Nuts Commandline Values

- Options can be valued

```
nuts -w=my-name
nuts -w my-name
nuts --workspace my-name
nuts --workspace=my-name

nuts -S
nuts -S=true

nuts -!S
nuts -~S
nuts -S=false
nuts -S=no
```

4. Selecting workspace

- User Workspace defaults to ~/.config/nuts/default-workspace

```
nuts
```

- System workspace defaults to /etc/opt/nuts/default-workspace

```
nuts -g
```

- You can customize it

```
nuts --workspace <workspace-location>
```

5. Creating Workspaces

- Workspace created on the first use

- Open existing workspace or fail if not already existing

```
nuts --open
```

6. Creating Workspaces

- Defining the default repositories with 'archetypes'

```
nuts --archetype=default
nuts --archetype=server
nuts --archetype=minimal
```

7. Folders

- Each (installed) App has distinct folders
  - app : app (executable) binaries
  - lib : lib (non executable) binaries
  - config : config files
  - var : variable data files
  - log : log files
  - temp : temp files
  - cache : cache files
  - run : run files (locks)

8. Folders

- Mainly folder are following two layout families
    - Linux,Unix,MacOS (XDG)
    - Windows (AppData)

9. Folders in Linux (user)

- 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"`

10. Folders in Linux (global/system)

- apps : `"/opt/nuts/apps"`

- lib : `"/opt/nuts/lib"`

- config : `"/etc/opt/nuts"`

- var : `"/var/opt/nuts"`

- log : `"/var/log/nuts"`

- temp : `"/tmp/nuts/global"`

- cache : `"/var/cache/nuts"`

- run : `"/tmp/run/nuts/global"`

11. Folders in Windows

- 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"`

12. Customize Folder Layout

- Mimic another OS

```
nuts --store-layout=system
nuts --store-layout=windows
nuts --store-layout=linux
```

- Exploded vs standalone

```
nuts --standalone
nuts --exploded
```

13. Destroying a workspace

- Need to erase config (--reset, -Z) and skip boot (--skip-boot, -Q)

```
nuts -ZQ
```

14. Working with Cache

- Invalidate only used files (New only)

```
nuts -N
```

- Invalidate All cache

```
nuts -z
```

15. Running an Application

- Supported executables

  - artifact (ex: `org.jedit:jedit#5.0`,`jedit`)

  - local jar (ex: `./jedit.jar`)

  - accessible jar (ex: `http://jedit.org/jedit.jar`)

  - accessible jar (ex: `ssh://vpc@my-server/jedit.jar`)

  - native applications (ex: `ls`, `cp` etc on Linux and `cmd.exe` on Windows)

  - any alias command to another supported executable (ex: `nll=nsh -c ls -l`)
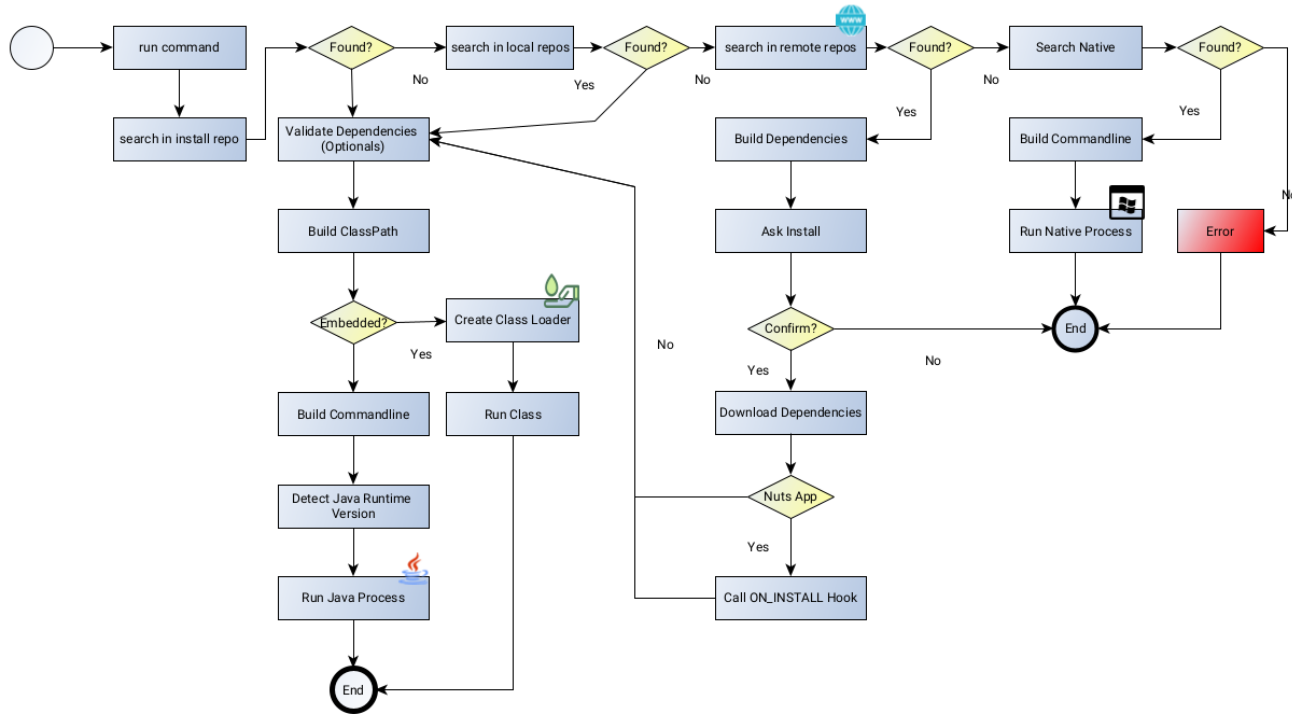
16. Running an Application

```
nuts my-application
nuts --exec my-application
nuts --exec --java-version=9 -Xmx2G my-application
nuts --embedded --java-version=9 -Xmx2G my-application
nuts ls
nuts --system ls
```

17. Running an Application

```
[start] → run command → Found?
                          │
run command → search in install repo
                          │
Found? --No--> Validate Dependencies (Optionals)
      --Yes (search in local repos)--

search in local repos → Found?
Found? --No--> search in remote repos
      --Yes--> Validate Dependencies (Optionals)

search in remote repos → Found?
Found? --No--> Search Native
      --Yes--> Build Dependencies

Build Dependencies → Ask Install → Confirm?
Confirm? --Yes--> Download Dependencies --No--> End
Download Dependencies → Nuts App
Nuts App --Yes--> Call ON_INSTALL Hook
         --No--> Validate Dependencies (Optionals)

Search Native → Found?
Found? --Yes--> Build Commandline --No--> Error
Build Commandline → Run Native Process → End
Error → End

Validate Dependencies (Optionals) → Build ClassPath → Embedded?
Embedded? --Yes--> Build Commandline → Detect Java Runtime Version → Run Java Process → End
          (Create Class Loader → Run Class → End)
```

18. Imports

```
nuts com.whitecape:white-app
nuts com.whitecape.yellow-app:yellow-app
nuts settings add import com.whitecape
nuts white-app
nuts yellow-app
nuts yellow-app#2.3
```

18.1. Conclusion

- `nuts` commandline is very rich

- `nuts` behaves seamlessly on linux/windows/macos

- 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