Application Management

René Serral-Gracià[1]

[1]Universitat Politècnica de Catalunya (UPC)

May 9, 2024

Lectures

1. System administration introduction
2. Operating System installation
3. User management
4. **Application management**
5. System monitoring
6. Filesystem Maintenance
7. Network services
8. Security and Protection
9. Introduction to Public Cloud

Outline

Goals

Knowledge

- Software distribution formats
- Software installation methods

Abilities

- App installation
  - Select install destination
  - Basic configuration
- Several versions maintenance

Outline

Manual installation procedure

1. Select install destination
2. Create destination directories
   - Executable
   - Libraries
   - Support files
   - Configuration files
   - Log files (logs)
   - Documentation
3. Distribute application files
4. Initial application configuration

Manual installation procedure

1. Select install destination
2. Create destination directories
   - Executable → $PREFIX/bin
   - Libraries → $PREFIX/lib i $PREFIX/lib64
   - Support files → $PREFIX/share
   - Configuration files → /etc
   - Log files (logs) → /var/log
   - Documentation → $PREFIX/share/doc
3. Distribute application files
4. Initial application configuration

Automatic installation process

Depending of the software deployment method

- Automate previously defined steps
- Generic procedure
- Delivers default configurations
  - Usually not suitable in the particular installation environment

Outline

From source code

1. Decompress the package into a separate directory (`/usr/src/app_name`)
2. Read the documentation — README, INSTALL...
3. Install Dependencies
   - Any method may be used
4. Adjust the code to the system's particular needs
   - Available libraries
   - Library and file locations
     - `autotools` based

```
$ ./configure --prefix=...
```

     - `cmake` based

```
$ cmake -DCMAKE_INSTALL_PREFIX=...
```

From source code

5. Compile

```
$ make
```

   - Solve any existing error

6. Install

```
$ sudo make install
```

   - Copy executables, libraries... to their final location
   - Generate default configuration files

7. Base configuration

Self-installable binaries

- Installation system suited to the application
- Implemented by the same company developing the application — Fits the application
- Closed feature set

Examples

- Java
- Matlab
- Most closed applications

Precompiled binaries

- Installation system suited for the whole distribution

```
# apt-get install ...
# dnf -i ...
# yast2 ...
```

- Created by the same distribution — Fits the system
- Predefined set of features, changeable with some efforts

Exercise

Discuss about the pros and cons of each software deployment method

- Source code
- Binary (precompiled)
- Binary (self-installable)

Outline

Application location and execution

- `$PATH` Environment variable
  - Determines the search path for executables
- Where to install the applications?
  - On its own directory
    - Then `$PATH` becomes very long
  - All in the same directory
    - Lack of organization and potential name clash
- Combine both options
  - Each application on its own directory
  - Creating soft-links to the executables in a common directory

Exercise

Where would you install the following applications?

- Libre Office - precompiled binaries
  - Text editor, spreadsheet, presentations
- Discord - Self installable binary
  - Development environment and Java virtual machine
- plasma/gnome - precompiled binaries
  - Window manager (graphical environment)
- vlc - source code
  - Multimedia player

Outline

Exercise

Assumen we already installed the applications of the previous case. . .

- Where would you install the following applications?
  - python 3.8 if the previous one was 2.7
  - Java 11 if we had version 8

Exercise

Assumen we already installed the applications of the previous case. . .

- Where would you install the following applications?
  - python 3.8 if the previous one was 2.7
  - Assuming that python was installed using precompiled binaries, in this case we could use any of the other alternatives considering that the application must have a $PREFIX different than `/usr`. Alternatively we could use the distribution enable facilities for this.
  - Java 11 if we had version 8
  - In this case since there is no percompiled binaries we have to use self-installable binaries and install it into a different $PREFIX, e.g., (`/usr/local/java/jdk11`)

Version maintenance

Only possible with very good planning

- Install the applications into different directories
  - It avoids filename and configuration clashes
- Keep a soft-link to the newest version (or more used)
  - Add old versions into each user's `$PATH`
  - Rename the old soft-link instead of removing it

Before installing a new version it must be checked that the changes do not impose issues for the previous versions

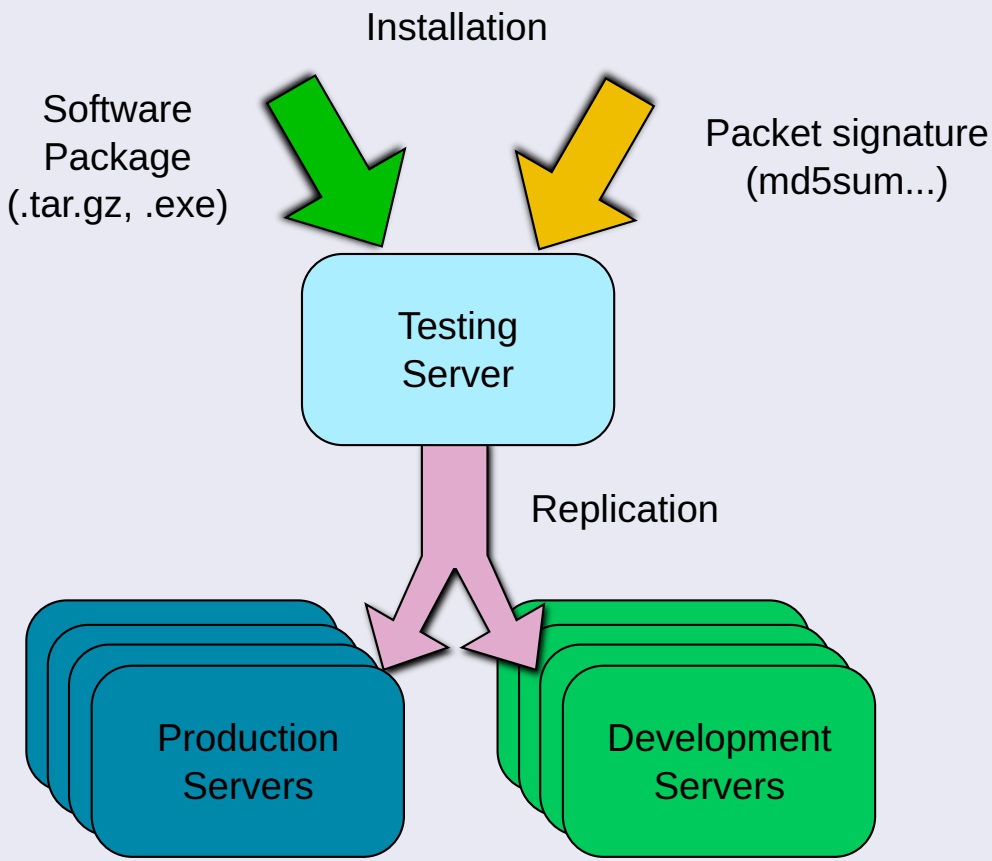
Outline

Other considerations

- Is the installed software secure?
  - Known vulnerabilities
  - Modified source code
  - Modified binaries
- How to keep it up-to-date?
  - 500+ software packages...
  - ...en 10+ servers
- Is the software stable?
  - +estable $\rightarrow$ -actual
  - Test server

Exercise

Discuss if the detailed issues are solved using this solution. . .



Homework

System monitoring commands

- `ps`, `top`
- `iostat`, `vmstat`
- `w`, `last`
- `du`, `df`