Empacotamento

Prof. Matheus Sousa Faria

Pra quê?

- Forma de distribuição
  - Instalação
  - Atualização
  - Remoção
- Pacote
  - Contém tudo que é necessário para o SW
  - Arquivos, meta-informações, dependências
- Padronização da distribuição
  - Repositórios oficiais
- Automação do processo de instalação
- Pacotes != Instaladores
  - Segue padrões
  - São gerenciados por package managers

Package Managers (oficiais)

APT

.deb

YUM

.rpm

PACMAN

.pkg.tar.xz

...

(Homebrew)

.dmg

“Google Play”

.apk

Package Managers (extra-oficiais)

pip /
easy_install

.egg

RubyGems

.gem

NPM

.npm

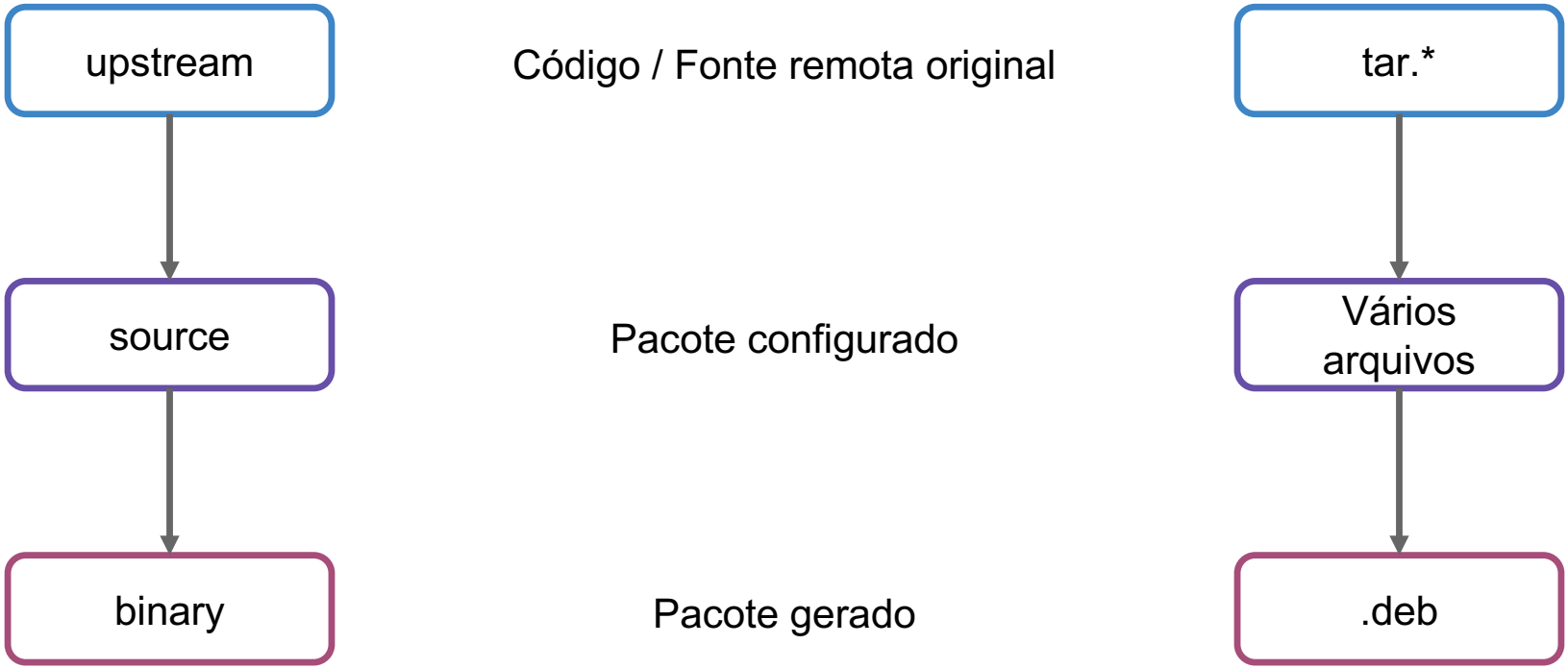
Snapcraft

.span

.deb

.deb

- Pacotes Debian-based
  - Ubuntu, Elementary
- Seguem regras e padrões pré estabelecidos
- Facilitam na instalação e configuração


upstream
Código / Fonte remota original
tar.*
source
Pacote configurado
Vários arquivos
binary
Pacote gerado
.deb

Como gerar um .deb?

Dependências para geração:

```
debhelper
devscripts
build-essential
```

Como gerar um .deb?

1. Adiquirir o *tarball* do upstream
    a. O nome deve estar seguindo a regra

```
appname_1.0.orig.tar.gz

vim_3.4.orig.tar.gz
docker_17.3.orig.tar.gz
```

Como gerar um .deb?

2. Descompactar o *tarball*
    a. Deve haver uma pasta seguindo a convenção de nomes

```
appname-1.0/

vim-3.4/
docker-17.3/
```

Como gerar um .deb?

3. Configurar o pacote
    a. Criar pasta debian
    b. Criar arquivos:
        i. Changelog
        ii. Compat
        iii. Control
        iv. Rules
        v. Source/format

Como gerar um .deb?

Changelog

Guarda o histórico de mudanças do pacote

```
appname (1.0-1) UNRELEASED; urgency=low

  * Initial release. (Closes: #32)

 -- Matheus Faria <matheusfaria@unb.br>  Thu, 18 Nov 2010 17:25:32 +0000
```

Como gerar um .deb?

Compat

Guarda a versão compatível do debhelper para aquele pacote

```
9
```

Como gerar um .deb?

Control

Informações do pacote, seus mantenedores, meta-informação

```
Source: appname
Maintainer: Lars Wirzenius <liw@liw.fi>
Section: misc
Priority: optional
Standards-Version: 3.9.2
Build-Depends: debhelper (>= 9)

Package: appname
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: this is an app with a name
 This the long description for this app that has a name.
```

Como gerar um .deb?

Copyright

Licença de distribuição do pacote

```
MIT, LGPL, ...
```

Como gerar um .deb?

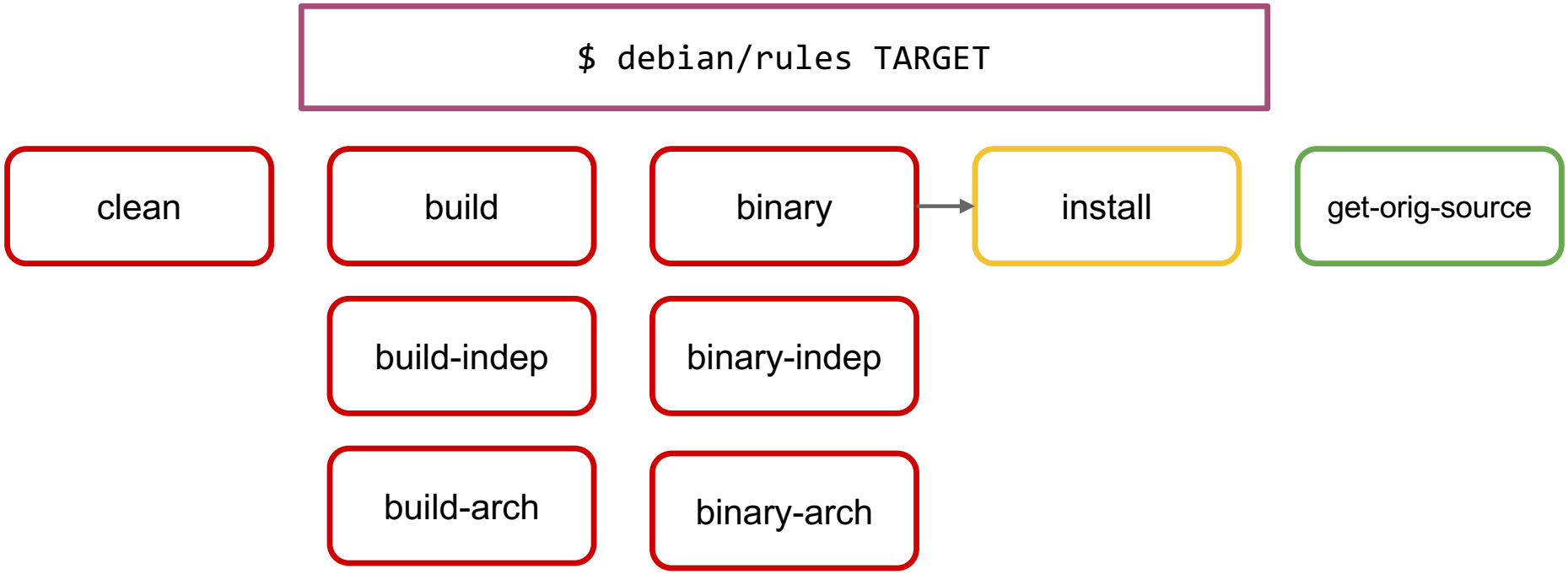
Rules (Makefile)

Passos para a configuração, instalação, remoção...

```
#!/usr/bin/make -f
%:
	dh $@
```

Como gerar um .deb?

Rules (Makefile): Targets



Como gerar um .deb?

Rules (Makefile): Targets

clean

```
dh_testdir
dh_auto_clean
dh_clean
```

build

```
dh_testdir
dh_auto_configure
dh_auto_build
dh_auto_test
```

binary

```
dh_testroot
dh_prep
dh_installdirs
dh_auto_install
dh_install
dh_installdocs
dh_installchangelogs
dh_installexamples
dh_installman
dh_installcatalogs
dh_installcron
dh_installdebconf
dh_installemacsen
dh_installifupdown
dh_installinfo
dh_installinit
dh_installmenu
dh_installmime
dh_installmodules
dh_installlogcheck
dh_installlogrotate
dh_installpam
dh_installppp
dh_installudev
dh_installwm
dh_installxfonts
dh_bugfiles
dh_lintian
dh_gconf
dh_icons
dh_perl
dh_usrlocal
dh_link
dh_compress
dh_fixperms
dh_strip
dh_makeshlibs
dh_shlibdeps
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
```

Como gerar um .deb?

Rules (Makefile)

100% compatível com um Makefile **bem feito**

Dica para impressão de todos os comandos dados:

```
#!/usr/bin/make -f
DH_VERBOSE = 1
%:
	dh $@
```

Como gerar um .deb?

source/format

Versão do .deb em que o pacote debian foi criado

```
3.0 (quilt)
```

Como gerar um .deb?

4. Construir o pacote

```
debuild -us -uc
```

4. Instalar o pacote

```
sudo dpkg -i appname_1.0-1_amd64.deb
```

Analisando um pacote já existente

```
apt-get source vim
```

Estado do seu pacote

`lintian`

- Analisa padrões de empacotamento
- LINT para pacote
- Acusa pequenos erros
- Roda automaticamente pelo debuild

Mais a se explorar da pasta debian/

`init.d`	Para criação de serviços
`manpage.*`	Páginas manuais
`appname.cron.d`	Cron jobs

Scripts de hooks de instalação

- `prerm`
- `postrm`
- `preinst`
- `postinst`

Mais a se explorar da pasta debian/

`appname.dirs` Diretórios que serão alterados/criados durante a instalação

```
usr/bin
usr/share
var/lib
```

Ajude o Debian

Seja um mantenedor de pacotes

pip

pip

- Pip Installs Packages
  - Pip Installs Python
- Gerenciador de pacotes do python
- Repositório Official: PyPI
  - Python Package Index
- Funcionalidades:
  - Instalar
  - Desinstalar
  - Listar
  - Instalar para um usuário específico

pip

```
pip install django
```

```
pip install --user django
```

Apenas para o usuário atual

```
pip install -r requirements.txt
```

Arquivo de dependencias

Python Package

setup.py

setup.cfg

→ Configuração do pacote

MANIFEST.in

→ Arquivos que serão distribuídos

README.rst

LICENSE.txt

→ Descrição do Projeto

meu_pacote/

→ Código

setup.py (Informações do Pacote)

```
from setuptools import setup, find_packages
setup(
    name = 'meu_pacote',
    version = '1.2.0',
    description = 'Descrição curta do pacote',
    long_description = '''Descrição Longa''',
    url = 'https://github.com/meunome/meu_pacote',
    author = 'Meu Nome',
    author_email = 'meunome@mail.com',
   license = 'MIT',
```

Utilizados no PyPI para identificação

setup.py (classificadores)

```
    classifiers=[
        'Development Status :: 3 - Alpha',

        'Intended Audience :: Developers',
        'Topic :: Software Development :: Build Tools',

        'License :: OSI Approved :: MIT License',

        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3',
    ],
    keywords='sample setuptools development',
```

Tags de indexação

setup.py

Dependências e arquivos do pacote

```
    packages=find_packages(exclude=['contrib', 'docs', 'tests']),

    install_requires=['django'],

    # $ pip install -e .[dev,test]
    extras_require={
        'dev': ['check-manifest'],
        'test': ['coverage'],
    },
```

setup.py

Comandos disponíveis após instalar

```
    entry_points={
        'console_scripts': [
            'meu_comando=meu_pacote.meu_pacote:main',
            'meu_comando2=meu_pacote.meu_pacote:execute',
        ],
    },
)
```

gem

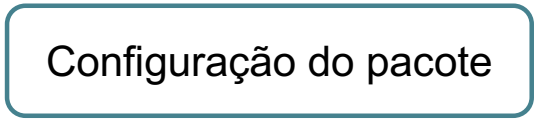
gem

- Rubygems: Gerenciador de pacotes do ruby
- Repositório Official: rubygems.org
- Funcionalidades:
  - Instalação
  - Desinstalação
  - Listagem das gems
  - Instalação local por diretório

gem

```
minha_gem.gemspec
lib/
        minha_gem/

        rubycode.rb
        minha_gem.r
b
```



Gemspec

```
gem build minhagem.gemspec
```

```
Gem::Specification.new do |s|
  s.name        = 'minha_gem'
  s.version     = '0.1.0'
  s.licenses    = ['MIT']
  s.summary     = "Descrição curta"
  s.description = "Descrição longa"
  s.authors     = ["Meu Nome"]
  s.email       = 'meunome@mail.com'
  s.files       = ["lib/example.rb"]
  s.homepage    = 'https://minhagem.io'
  s.metadata    = { "source_code_uri" => "https://github.com/" }
end
```

npm

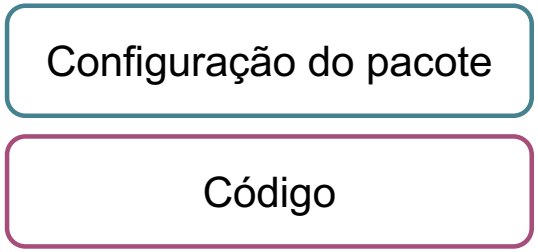
npm

- Node Package Manager
- Pacote == Node Module
- Repositório Official: npmjs.com
- Muito semelhante ao rubygems em nível de funcionalidade

npm

package.json

```
app/
    codigo.js
```



package.json

```
{
  "name": "meu_pacote",
  "version": "1.0.0",
  "description": "Breve Descrição",
  "main": "index.js",
  "scripts": {
    "gulp": "gulp"
  },
  "author": "Meu Nome",
  "license": "ISC",
  "dependencies": {
    "browser-sync": "^2.18.13",
    "del": "^3.0.0",
    "run-sequence": "^2.2.0"
  }
}
```

```
npm init
```

```
npm install run-sequence --save
```

fim.pkg