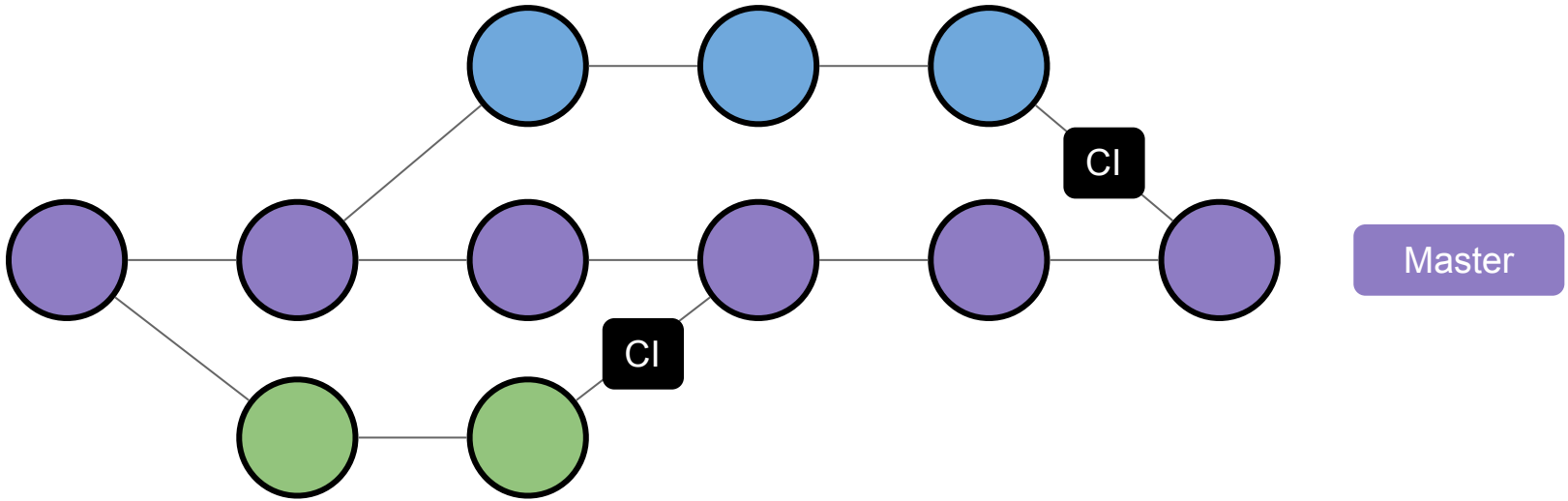
Integração Contínua

Prof. Matheus Sousa Faria

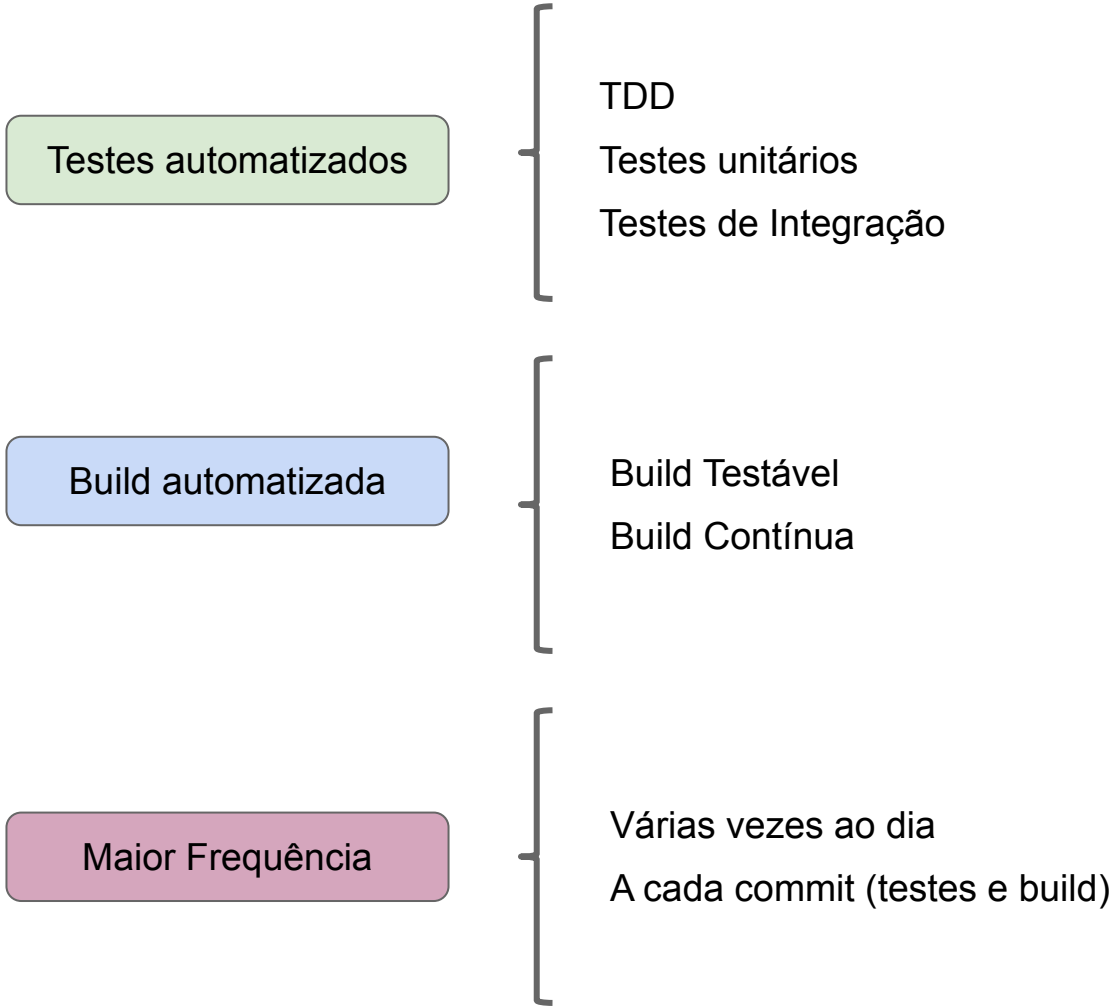
O que é?

Integração frequente das diferentes branches de trabalho


CI
CI
Master

XP

eXtreme Programming



Benefícios e Boas Práticas

1. Build Automática
2. Build Testável
3. Commits diários na master
4. Commits testados e funcionando na master
5. Build rápida
6. Deploy Automático

Integração Contínua

!=

Deploy Contínuo

Entrega contínua em produção ao cliente

Deploy Contínuo

Entrega Contínua (Continuous Delivery) ! = Deploy Contínuo

Entrega Contínua
(Continuous Delivery)

A habilidade de gerar uma build estável a qualquer momento

Integração Contínua
(Continuous Integration)

Entrega Contínua
(Continuous Delivery)

Deploy Contínuo
(Continuous Deployment)

Desenvolvimento | **Integração** | **Produção**

Contextos de Aplicação

Auto-hospedagem (Self-hosted)

- Roda em Hardware próprio
- VM, Cloud proprietária, Workstation
- Sua responsabilidade de instalar e manter

Software como Serviço (SaaS)

Provedores de Serviços na Nuvem (Cloud Providers)

Serviços de Repositórios (Github / Gitlab)

Self-Hosted

Vantagens:
- Máxima flexibilidade
- Controle das informações (Segurança)

Desvantagens:
- Instalação e manutenção
- Escala limitada à infra local
- Tempo de set-up inicial

SaaS

Vantagens:
- Fácil configuração
- Opções gratuitas
- Triggers automáticos

Desvantagens:
- Alto custo para aplicações grandes
- Segurança

Provedores de Serviços na Nuvem

Vantagens:
- Fácil configuração
- Fácil integração com outras ferramentas
- Escala fácil (Capacidade computacional)

Desvantagens:
- Preso a uma solução proprietária

Servidores de Repositórios

Vantagens:

- CI / CD em uma só ferramenta
- Containers Docker como opção
- Opções gratuitas

Desvantagens:

- Alto custo em escala

Ferramentas

Travis CI

CODESHIP

Jenkins

GitLab CI

circleci

Travis CI

CODESHIP

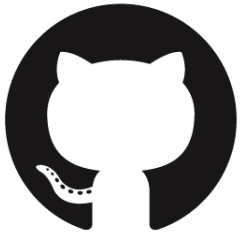
Jenkins

GitLab CI

Bamboo

circleci

Travis CI



GitHub

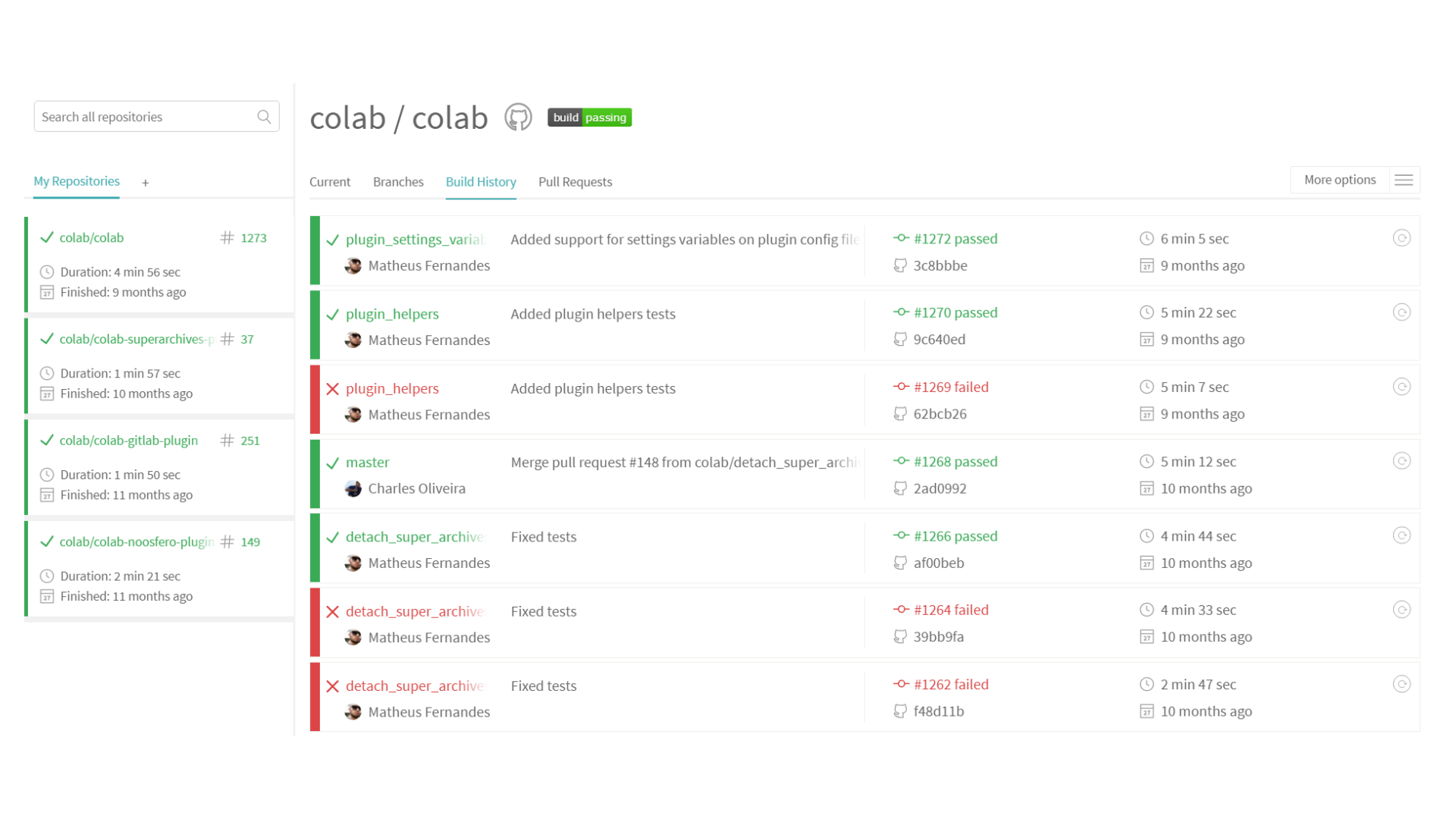
Travis CI

.travis.yml

```
language: python
python:
  - "2.6"
  - "2.7"
  - "3.2"
  - "3.3"
install:
  - pip install .
  - pip install -r
requirements.txt
script: pytest
```

YAML

Shell-like


Search all repositories
My Repositories
+
colab/colab
1273
Duration: 4 min 56 sec
Finished: 9 months ago
colab/colab-superarchives-p
37
Duration: 1 min 57 sec
Finished: 10 months ago
colab/colab-gitlab-plugin
251
Duration: 1 min 50 sec
Finished: 11 months ago
colab/colab-noosfero-plugin
149
Duration: 2 min 21 sec
Finished: 11 months ago
colab / colab
build passing
Current
Branches
Build History
Pull Requests
More options
plugin_settings_varia
Added support for settings variables on plugin config file
Matheus Fernandes
#1272 passed
3c8bbbe
6 min 5 sec
9 months ago
plugin_helpers
Added plugin helpers tests
Matheus Fernandes
#1270 passed
9c640ed
5 min 22 sec
9 months ago
plugin_helpers
Added plugin helpers tests
Matheus Fernandes
#1269 failed
62bcb26
5 min 7 sec
9 months ago
master
Merge pull request #148 from colab/detach_super_archi
Charles Oliveira
#1268 passed
2ad0992
5 min 12 sec
10 months ago
detach_super_archive
Fixed tests
Matheus Fernandes
#1266 passed
af00beb
4 min 44 sec
10 months ago
detach_super_archive
Fixed tests
Matheus Fernandes
#1264 failed
39bb9fa
4 min 33 sec
10 months ago
detach_super_archive
Fixed tests
Matheus Fernandes
#1262 failed
f48d11b
2 min 47 sec
10 months ago

✓ **plugin_settings_variables** Added support for settings variables on plugin config file

#1272 passed

Restart build

Signed-off-by: Matheus Fernandes <matheus.souza.fernandes@gmail.com>

Ran for 6 min 5 sec

9 months ago

Commit 3c8bbbe

Compare 3c8bbbedda83

Branch plugin_settings_variables

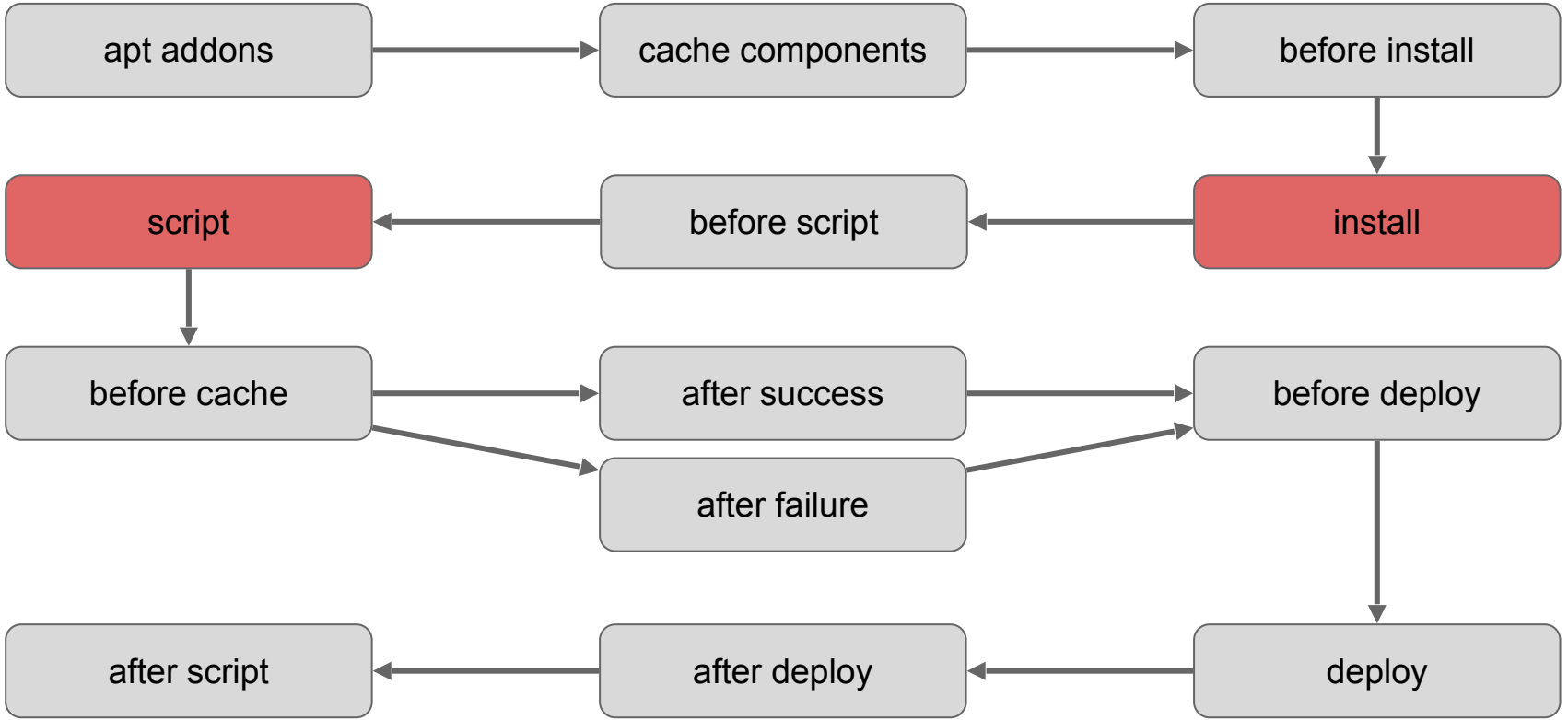
Matheus Fernandes authored and committed

Job log | View config

Remove log | Raw log

```
1   Using worker: worker-linux-docker-03303d85.prod.travis-ci.org:travis-linux-8
2
3   Build system information                                                    system_info
70
71  $ export DEBIAN_FRONTEND=noninteractive                                     fix.CVE-2015-7547
110 $ git clone --depth=50 --branch=plugin_settings_variables https://github.com/colab/colab.git colab/colab   git.checkout  0.97s
121 Installing APT Packages (BETA)                                              apt
```

Travis Lifecycle



Travis Lifecycle

apt addons

Exclusivo do Ubuntu

```
addons:
  apt:
    sources:
    - ubuntu-toolchain-r-test
    packages:
    - gcc-4.8
    - g++-4.8
    - cmake
```

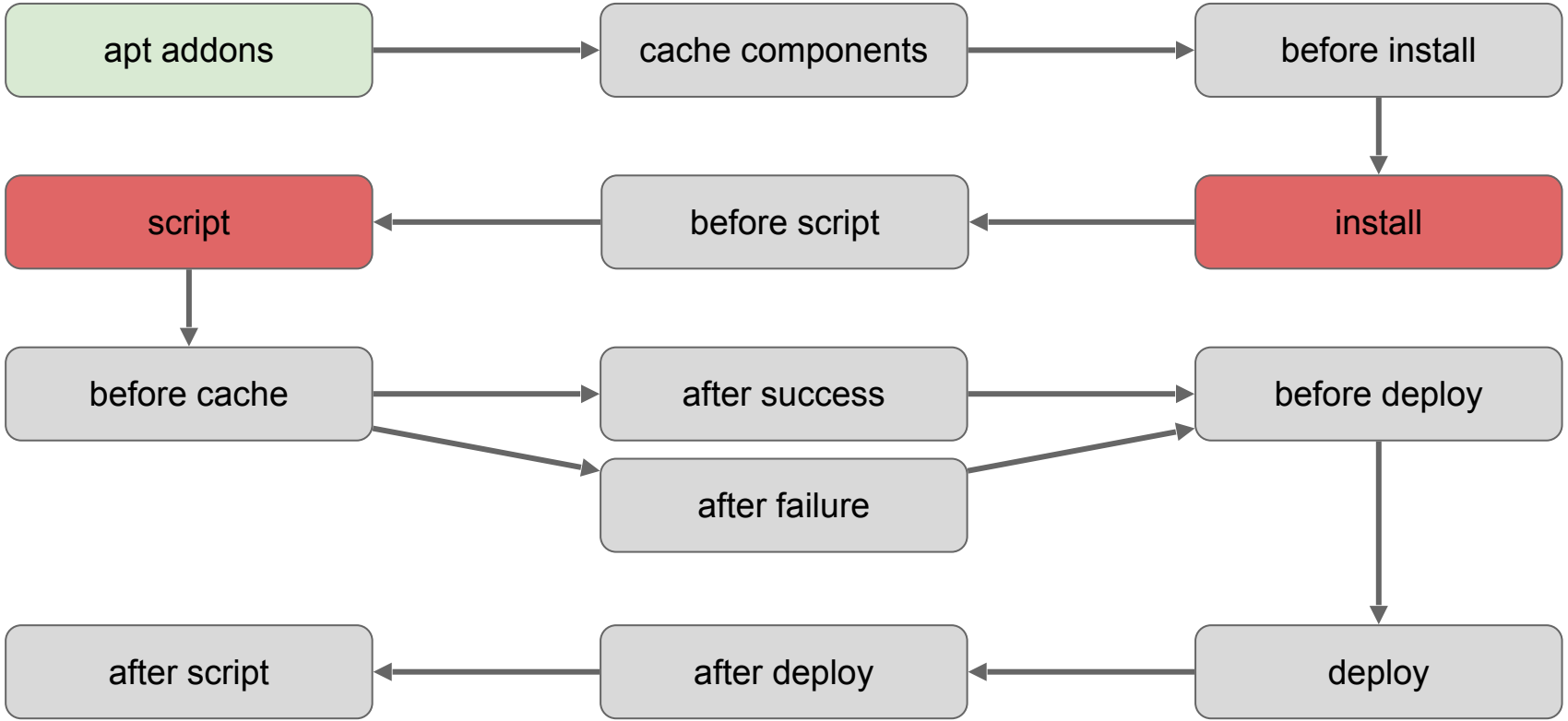
Travis Lifecycle

apt addons

Exclusivo do Ubuntu

```
addons:
  apt:
    sources:
    - deadsnakes
    - sourceline: 'ppa:ubuntu-toolchain-r/test'
    - sourceline: 'deb https://packagecloud.io/chef/stable/ubuntu/
precise main'
      key_url: 'https://packagecloud.io/gpg.key'
```

Travis Lifecycle



Travis Lifecycle

Marcar o que deve ser salvo após a build

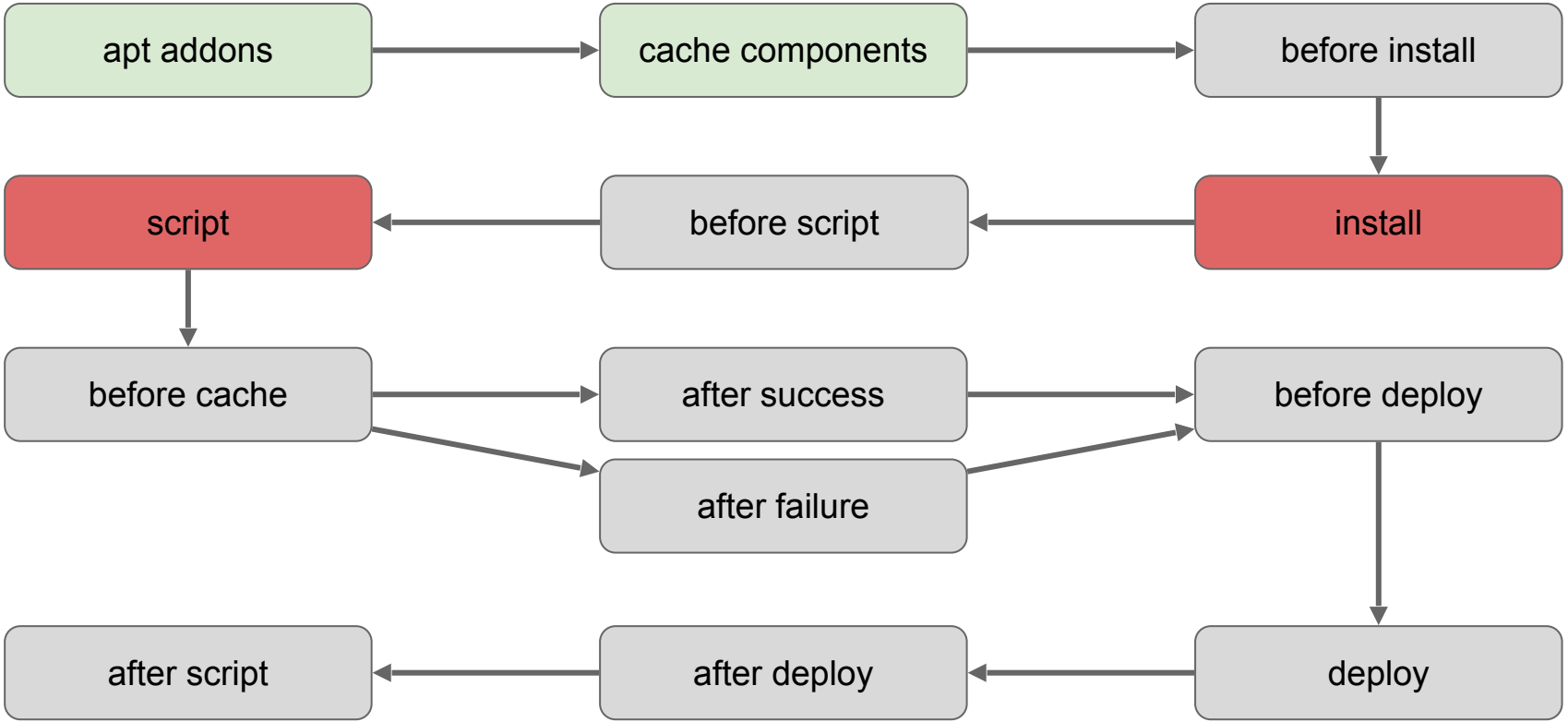

```
cache:
  - bundler
  - cocoapods
```

```
cache:
  directories:
    - $HOME/.cache/pip
```

Evita downloads lentos durante a build

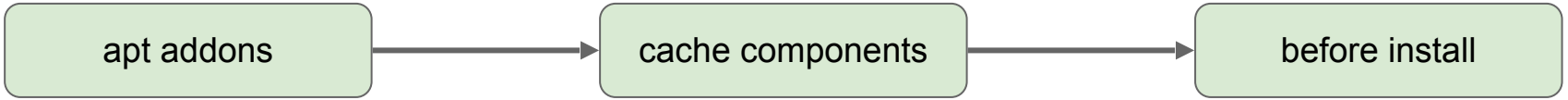
```
cache:
  bundler: true
  directories:
  - node_modules # NPM packages
```

Travis Lifecycle



Travis Lifecycle

Install para qualquer SO



```
before_install:
  - sudo apt-get -qq update
  - sudo apt-get install -y libxml2-dev
```

```
before_install:
  - brew update
  - brew install beanstalk
```

```
before_install:
  - wget http://pngquant.org/pngquant_1.7.1-1_i386.deb
  - sudo dpkg -i pngquant_1.7.1-1_i386.deb
```

Múltiplos Sistemas Operacionais

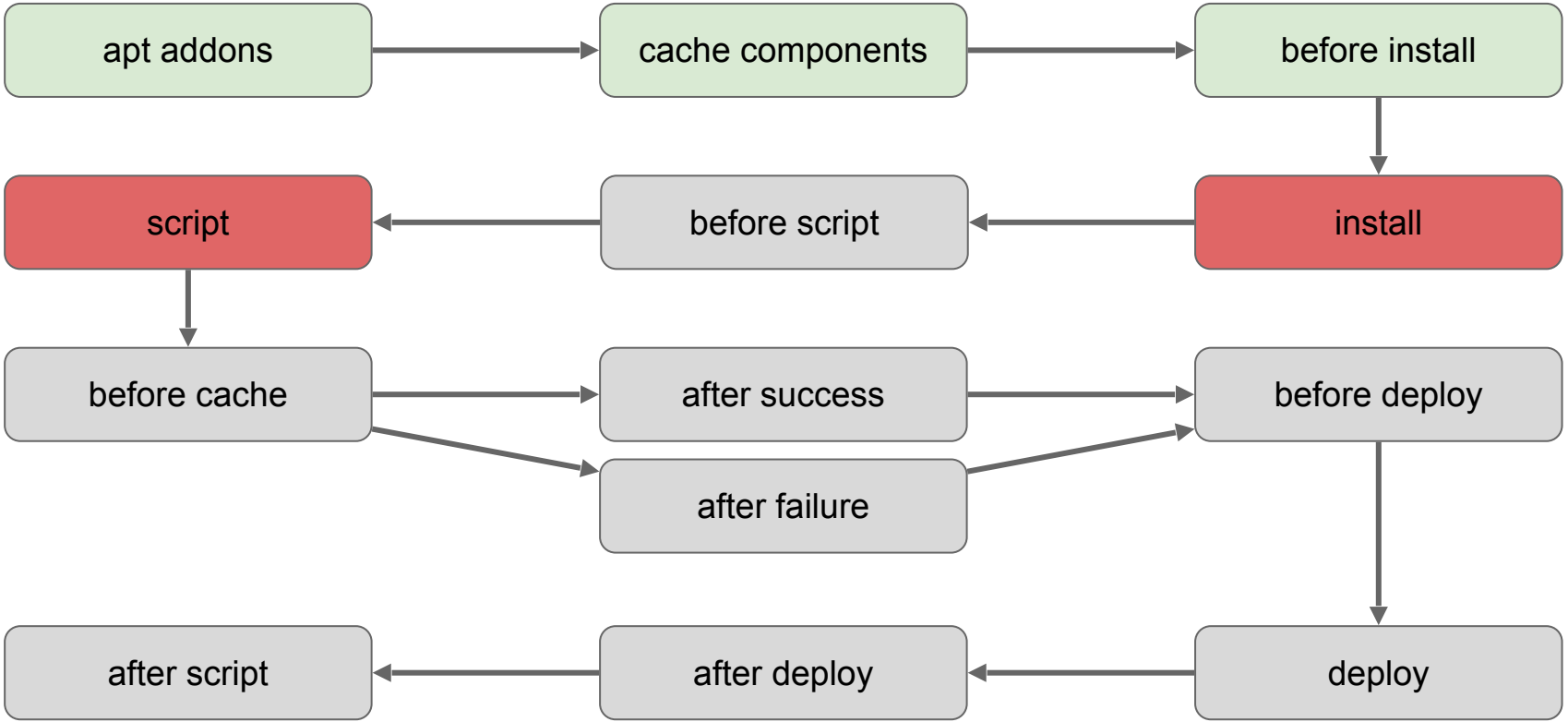
Trusty

Precise

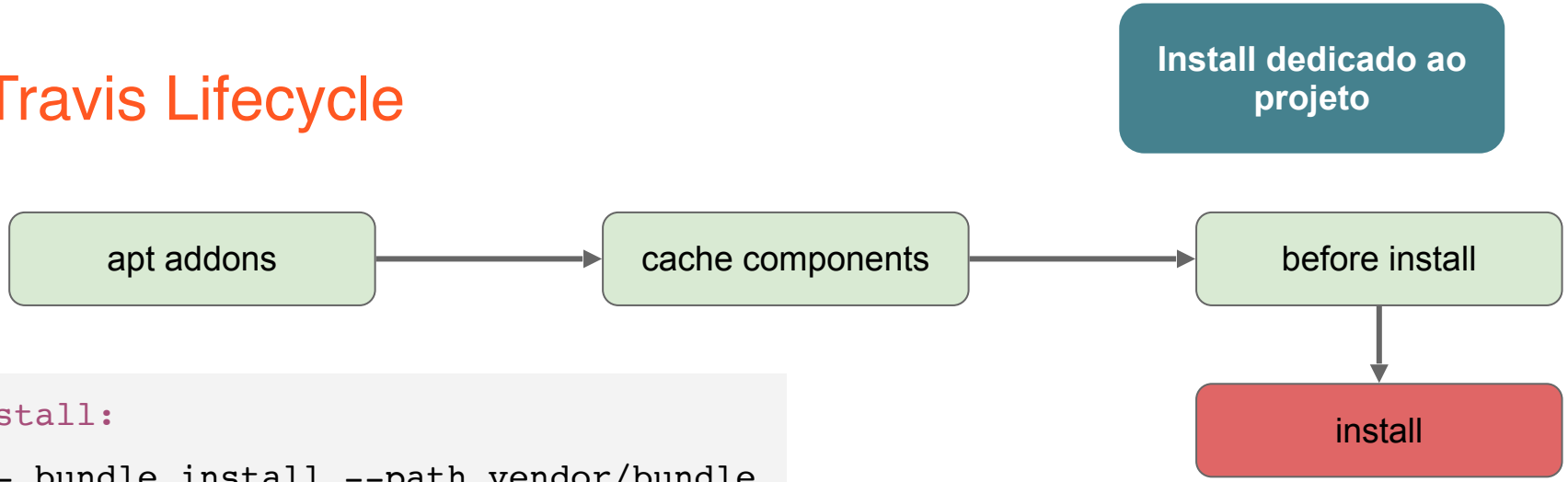
```
os:
  - linux
  - osx
```

10.11.6

Travis Lifecycle



Travis Lifecycle




```
install:
  - bundle install --path vendor/bundle
  - npm install
```

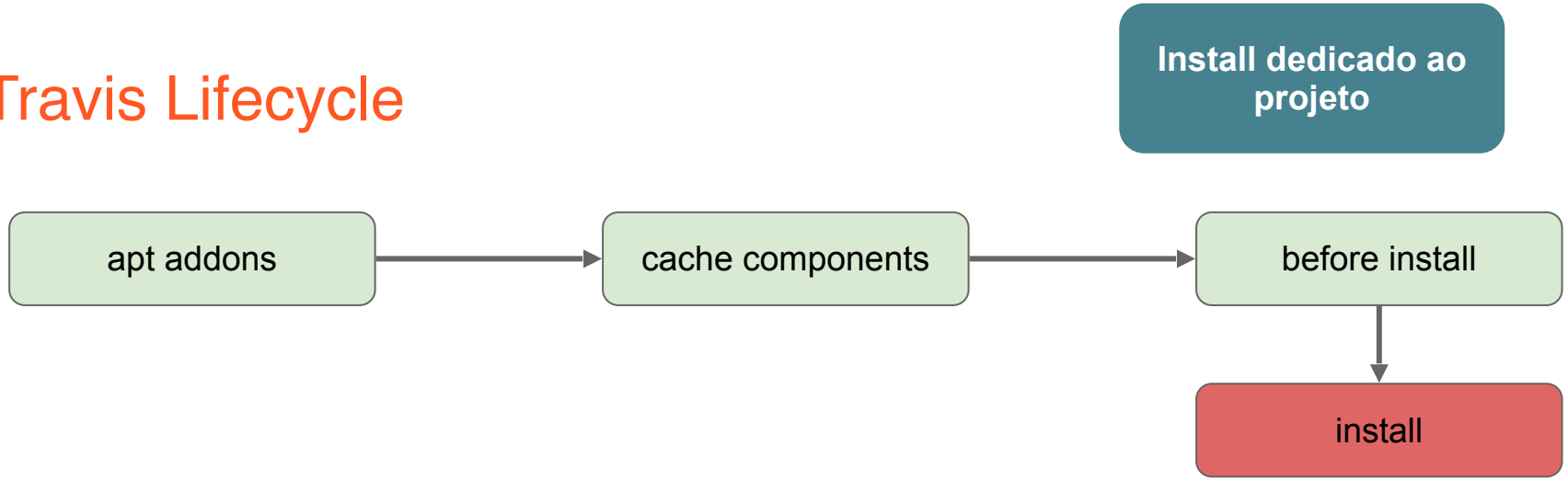
```
install: true
```

Pula o install

```
install: ./install-
dependencies.sh
```

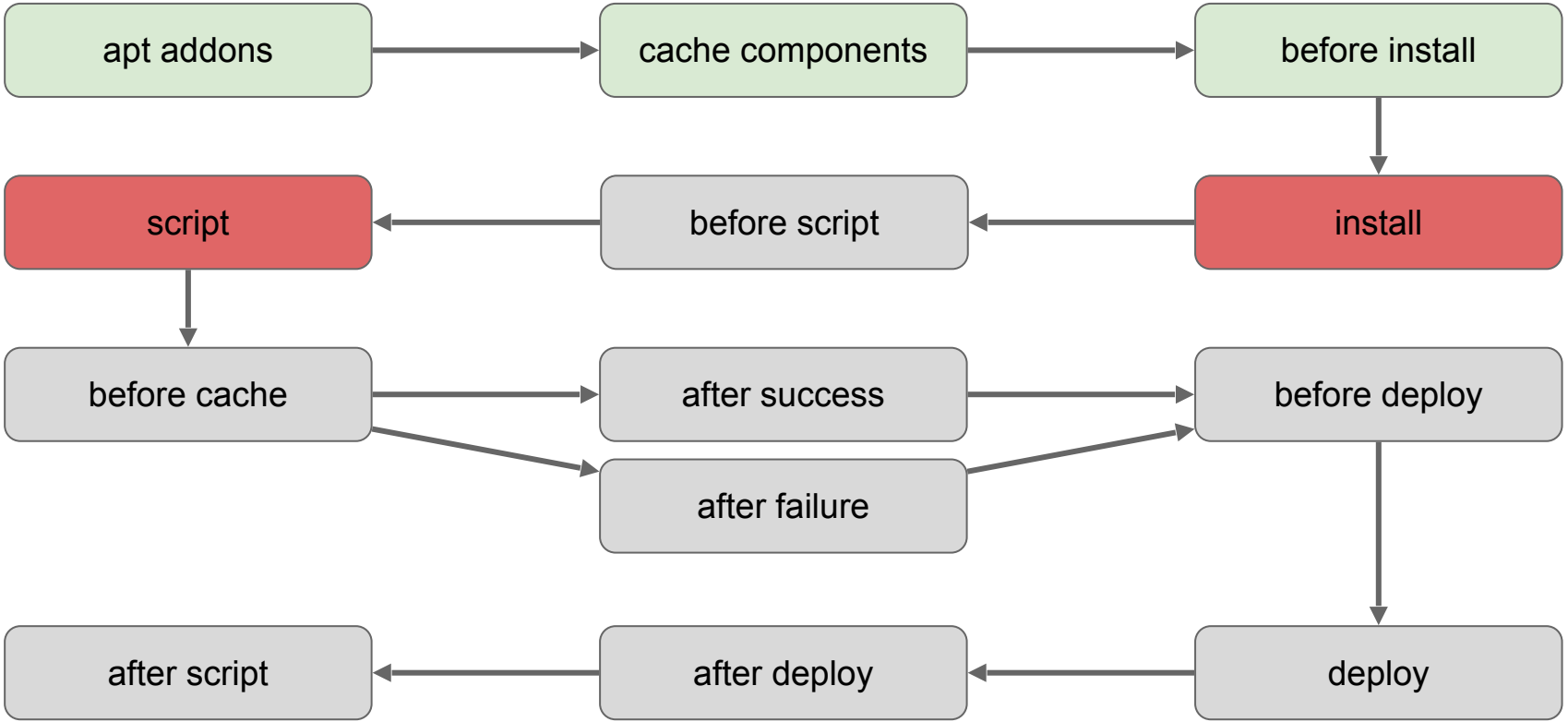
Deve ter o #!/usr/bin/env sh

Travis Lifecycle

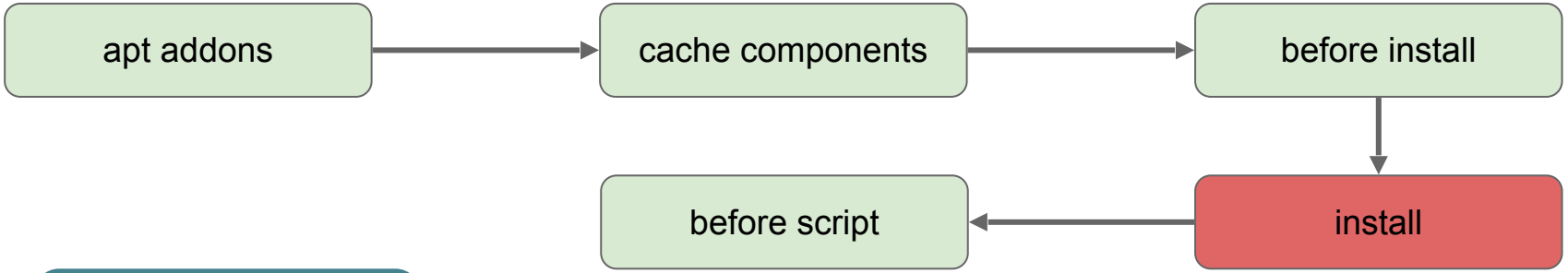



```
install:
  - if [ $TRAVIS_OS_NAME = linux ]; then sudo apt-get install foo; else brew install
bar; fi
```

Travis Lifecycle



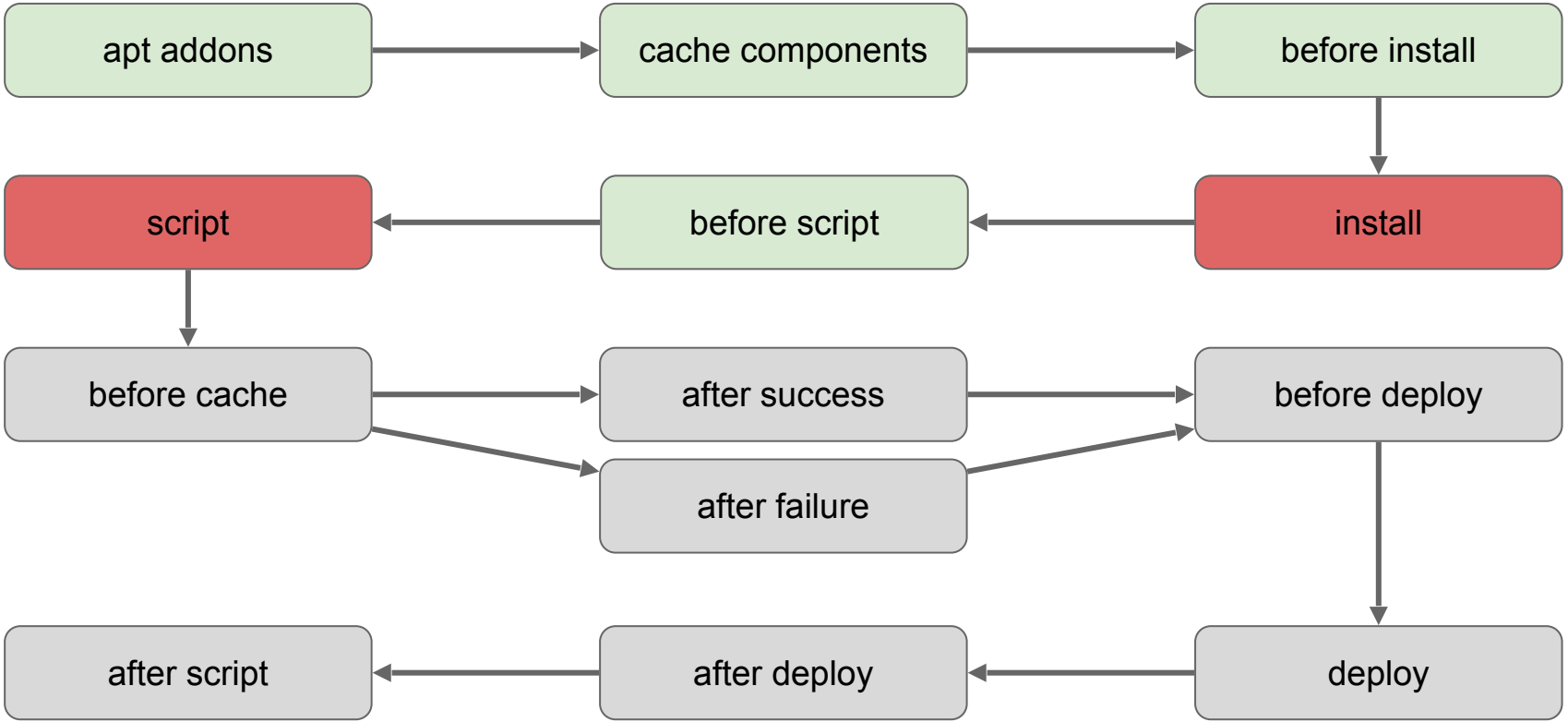
Travis Lifecycle



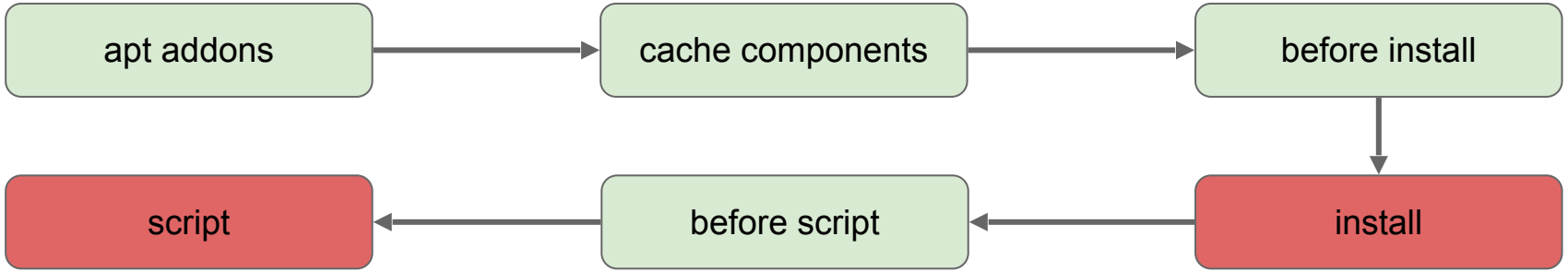
Prepara o ambiente para o script

```
before_script:
  - cp config/database.yml.travis config/database.yml
```

Travis Lifecycle



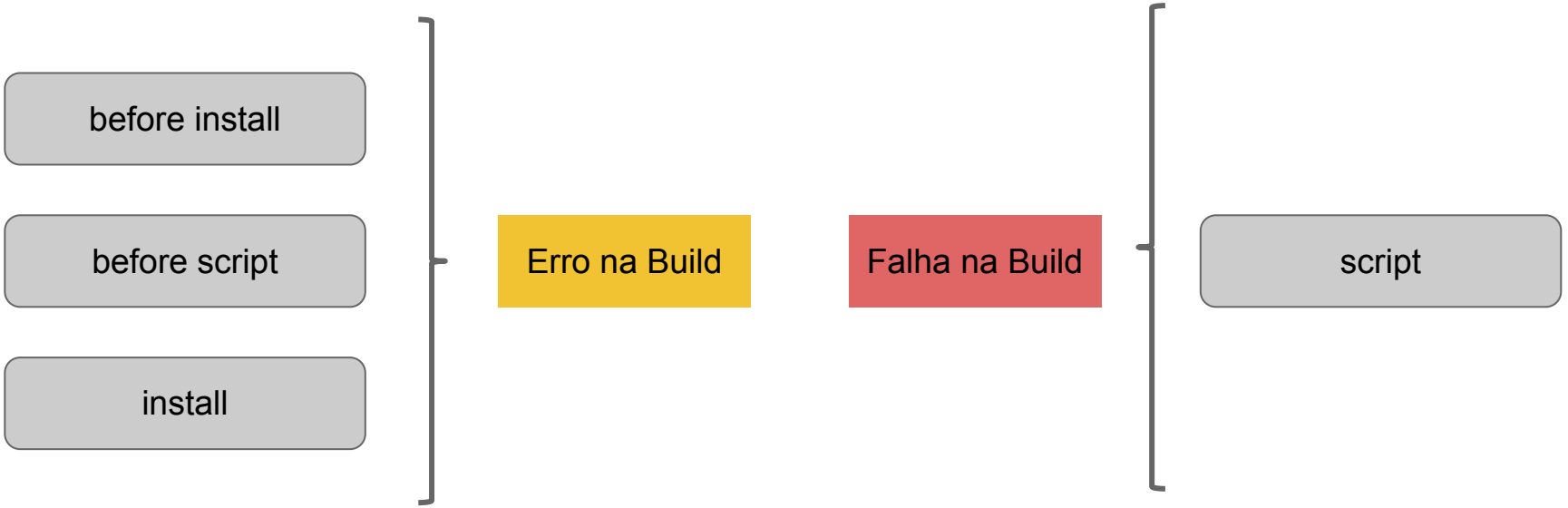
Travis Lifecycle



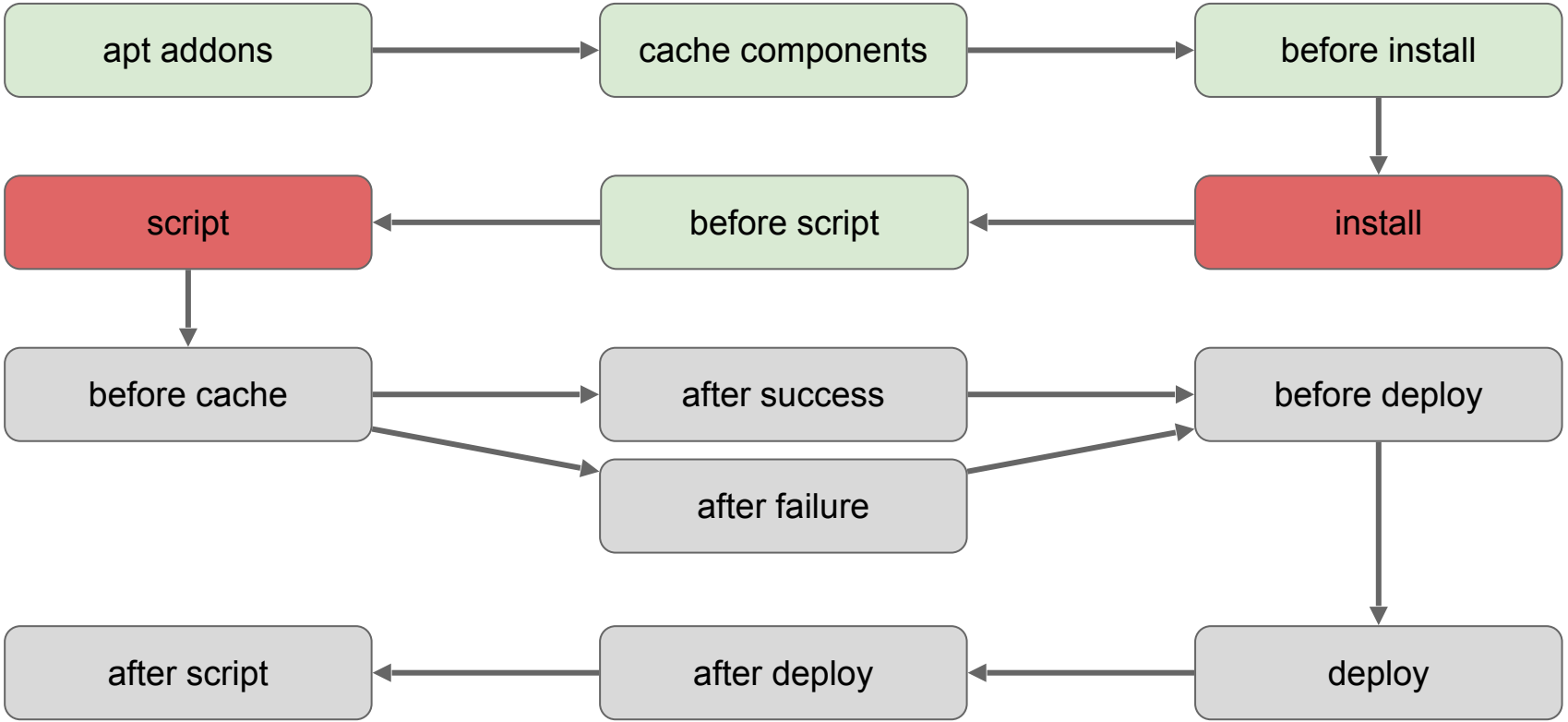

```
script:
  - python setup.py test
  - flake8 colab
```

Onde acontece a build, testes, lint, ...

Status de Retorno

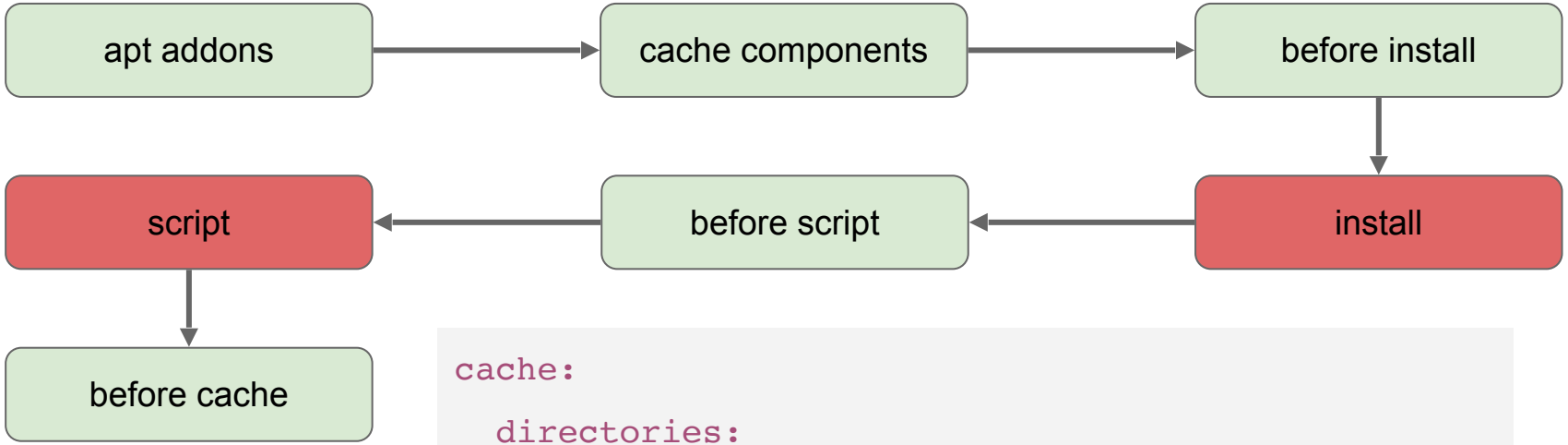


Travis Lifecycle



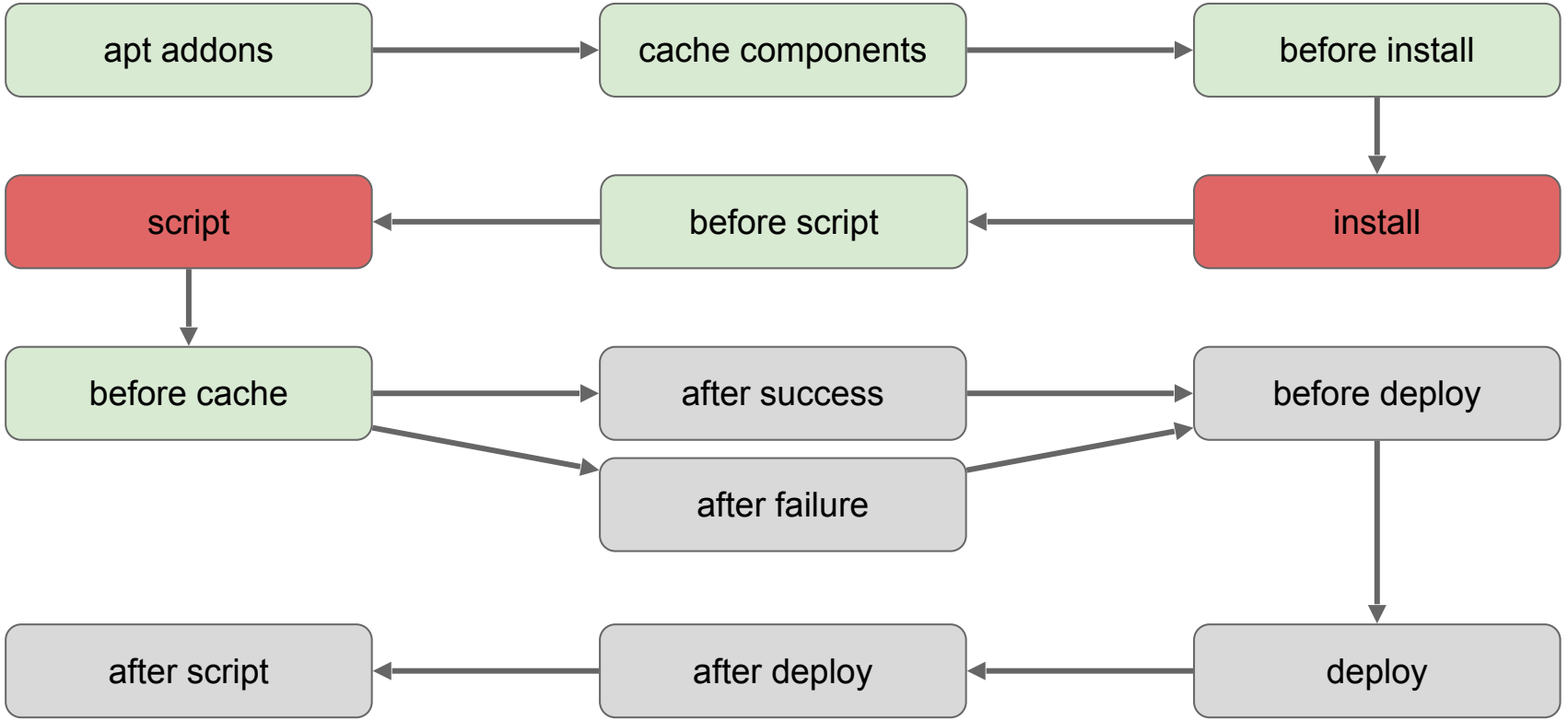
Travis Lifecycle

Acontece antes de salvar o cache

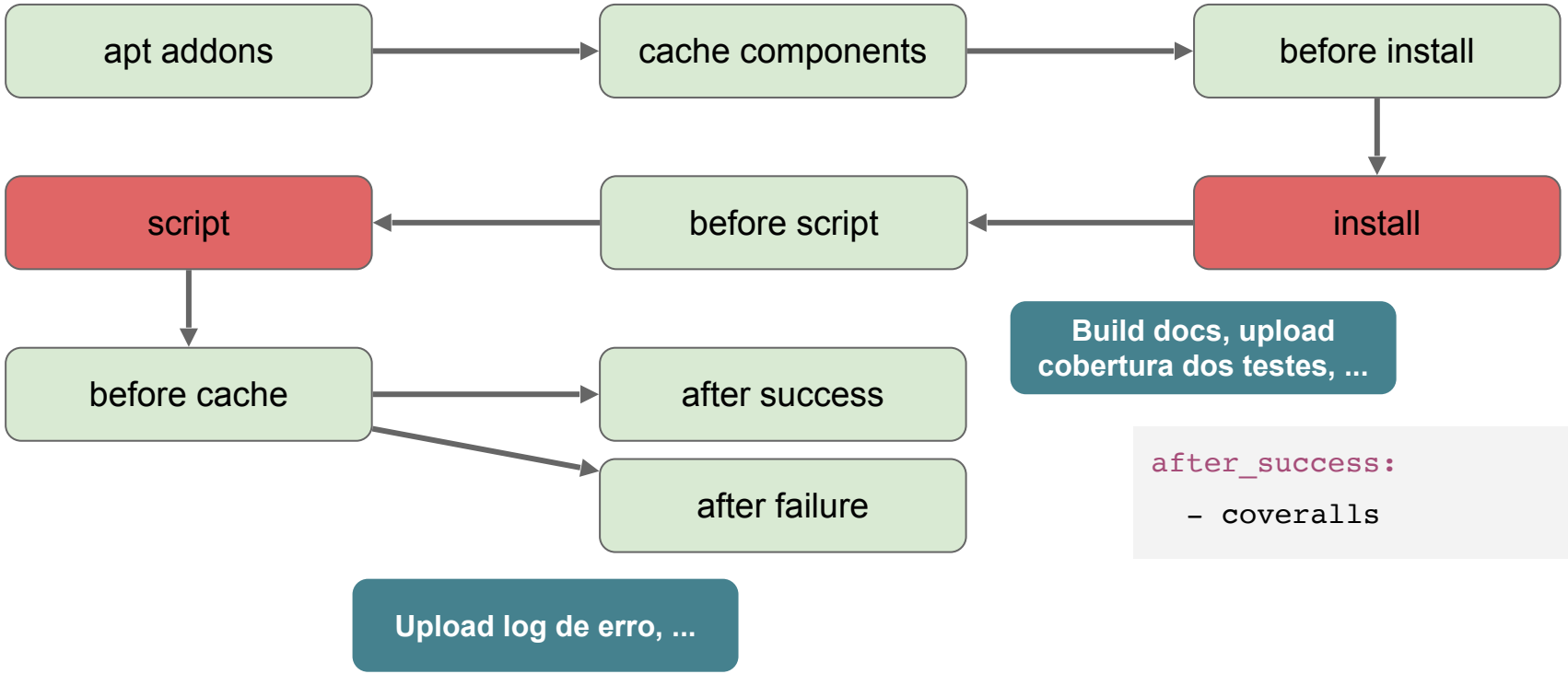



```
cache:
  directories:
    - $HOME/.cache/pip
before_cache:
  - rm -f $HOME/.cache/pip/log/debug.log
```

Travis Lifecycle

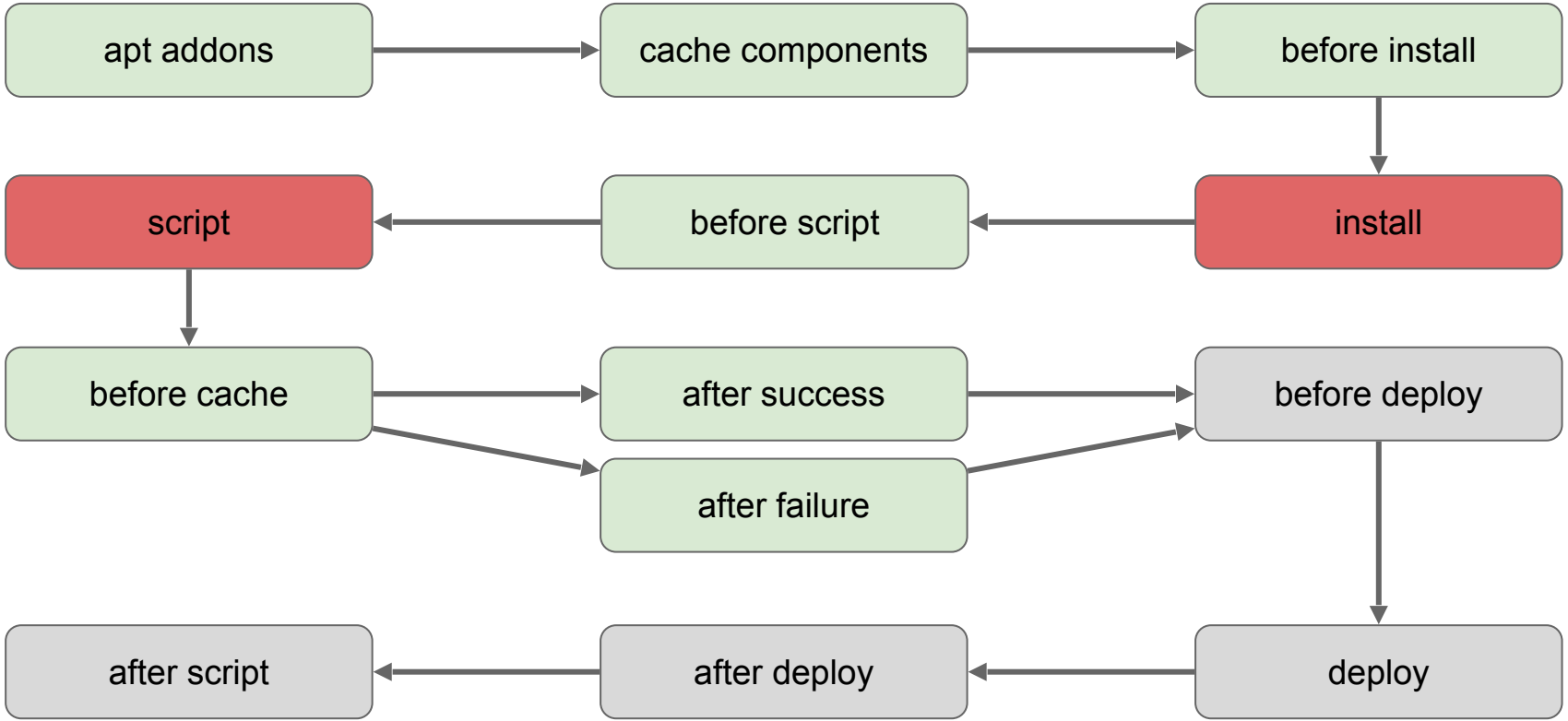


Travis Lifecycle

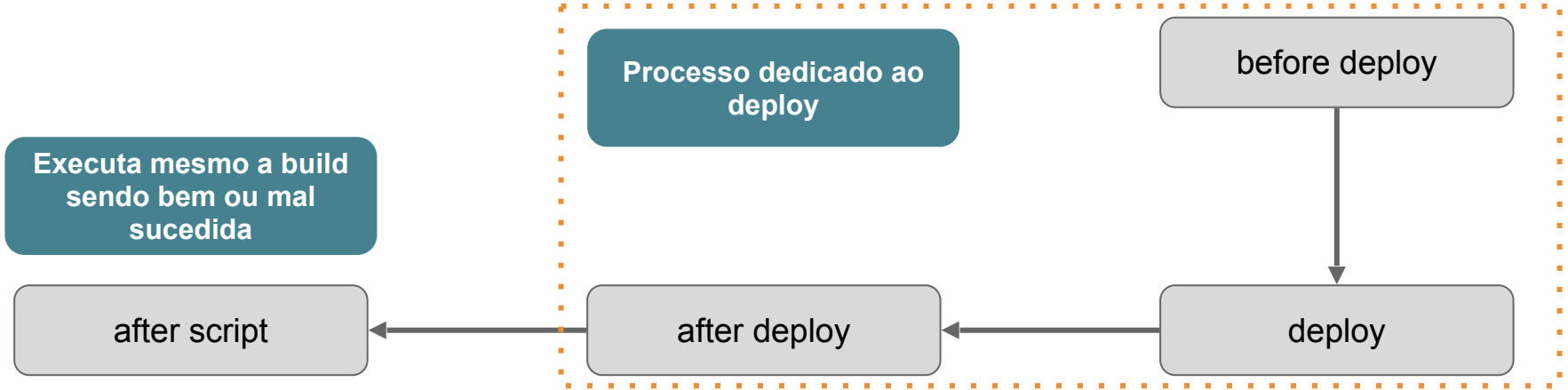



```
after_success:
  - coveralls
```

Travis Lifecycle



Travis Lifecycle



Travis Lifecycle

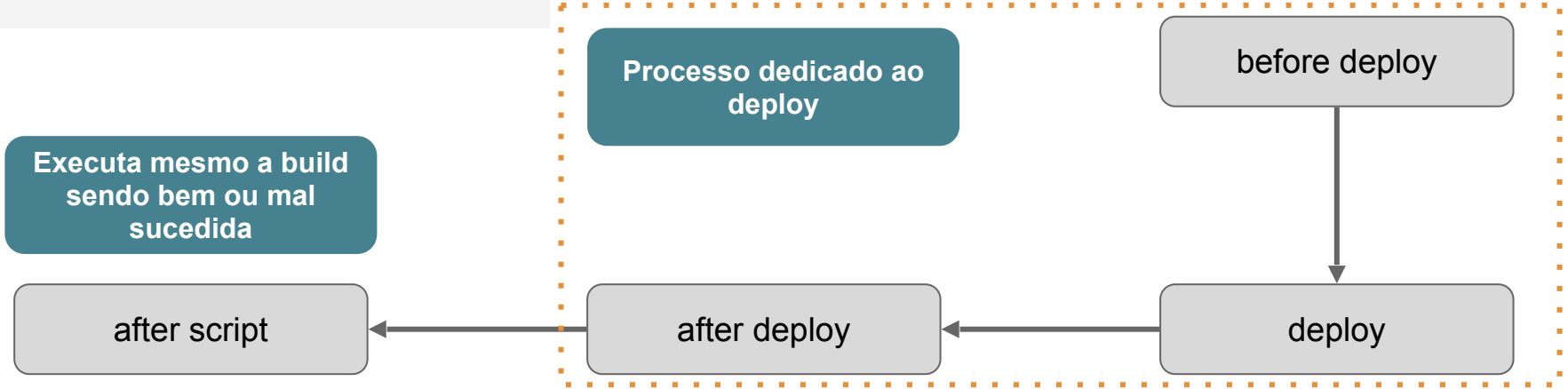
Entrega Contínua

heroku

```
deploy:
  provider: heroku
  api_key: ...
  app:
    master: my-app-staging
    production: my-app-
production
```

Geração da api_key

```
travis encrypt $(heroku auth:token) --add
deploy.api_key
```

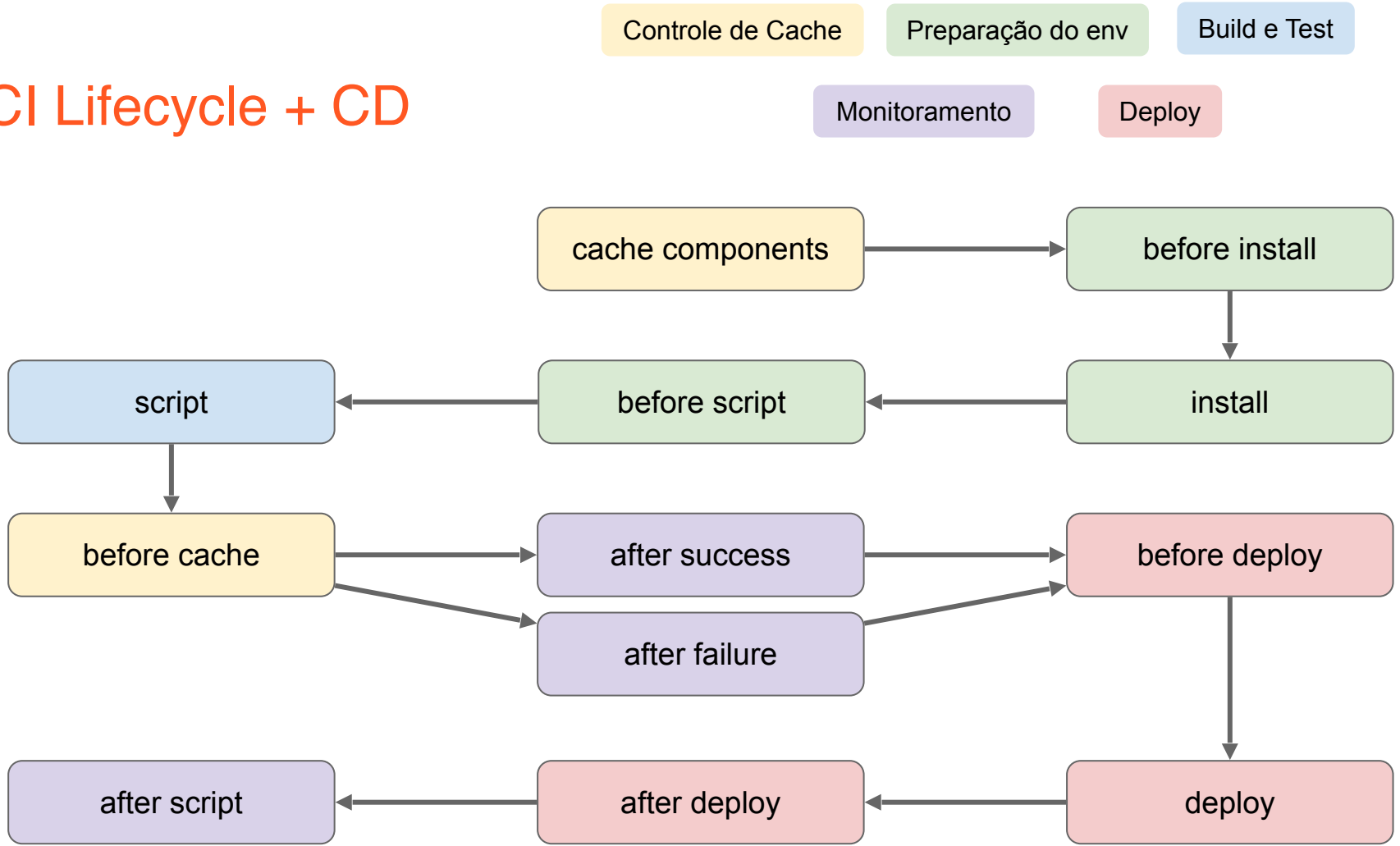


Validação do arquivo do Travis

Gem Travis

```
$ travis
lint .travis.yml
```

CI Lifecycle + CD



Travis CI

CODESHIP

Jenkins

GitLab CI

circleci

Travis CI

CODESHIP

Jenkins

GitLab CI

Bamboo

circleci


circleci


GitHub

circleci.yml

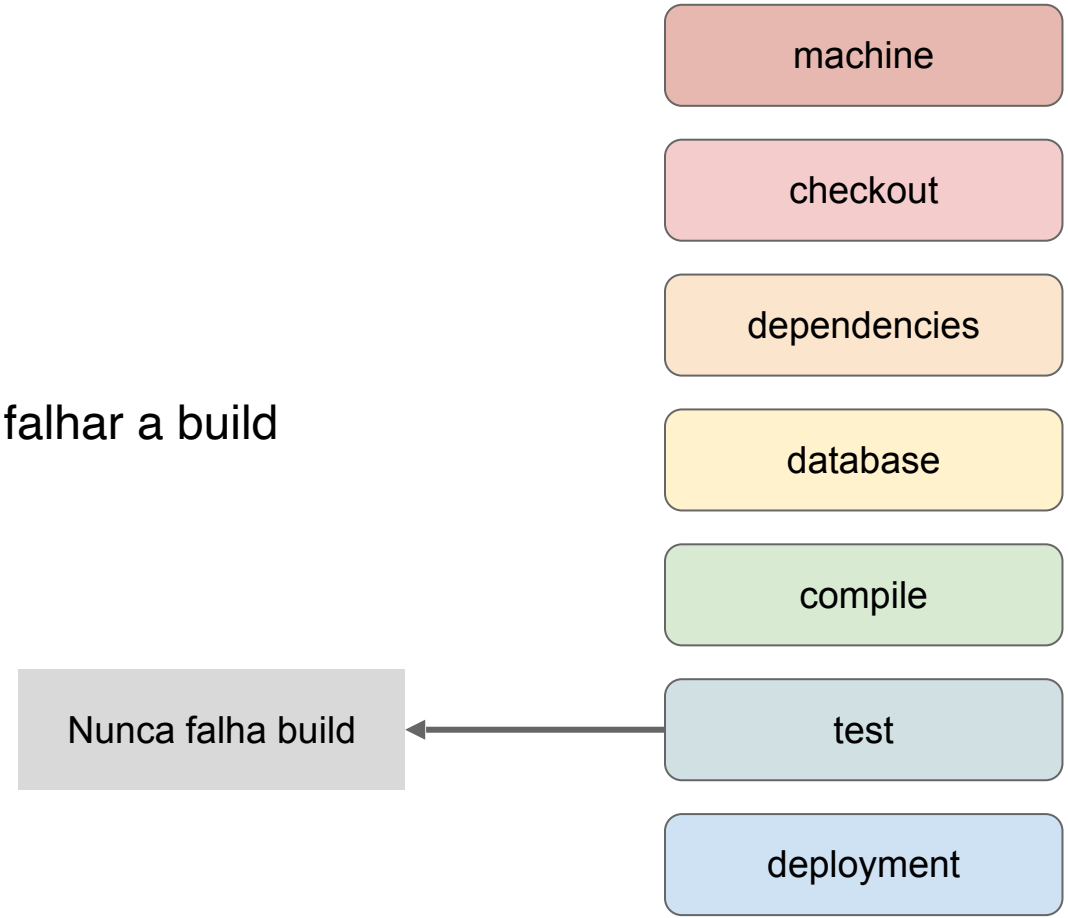
YAML

Shell-like

```
machine:
  timezone:
    America/Los_Angeles
  ruby:
    version:
      1.8.7-p358-falcon-perf
checkout:
  post:
    - git submodule sync
    - git submodule update --init
dependencies:
  pre:
    - npm install coffeescript
    - gem uninstall bundler
    - gem install bundler --pre
```

Circle CI Lifecycle

- 7 seções
- Seriais
- Independentes
- Todas as seções podem falhar a build
  - 0 == sucesso
  - !0 == falha



Circle CI Lifecycle

Configuração da máquina de build

machine

```
machine:
  timezone:
    America/Los_Angeles
  ruby:
    version: 1.9.3-p0-falcon
  hosts:
    dev.circleci.com: 127.0.0.1
    foobar: 1.2.3.4
  environment:
    foo: bar
    baz: 123
  services:
    - cassandra
    - elasticsearch
    - rabbitmq-server
    - redis
```

Circle CI Lifecycle

machine

checkout

Obter o código no seu estado correto

```
checkout:
  post:
    - git submodule sync
    - git submodule update --init
```

Pode se utilizar o default

Circle CI Lifecycle

```
dependencies:
  pre:
    - gem uninstall bundler
    - gem install bundler --pre
  override:
    - npm install
  cache_directories:
    - "assets/cache"
    - "~/assets/output"
```

Instalar as dependências do projeto

Sobrescrever o padrão

machine

checkout

dependencies

Circle CI Lifecycle

Configuração do banco de dados

```
database:
  override:
    - mv config/database.ci.yml config/database.yml
    - bundle exec rake db:create db:schema:load --trace
```

machine

checkout

dependencies

database

Circle CI Lifecycle

Comandos para uma build customizada

machine

checkout

dependencies

database

compile

test

```
compile:
  override:
    - bundle exec middleman build --
verbose
test:
  override:
    - phpunit my/special/subdirectory/
tests
```

Circle CI Lifecycle


```
deployment:
  staging:
    branch: master
    heroku:
      appname: foo-bar-123
```

- Opicional
- Faz o deploy para um provider
- 1+ branches especificadas
- SSH keys registradas no CircleCI

machine

checkout

dependencies

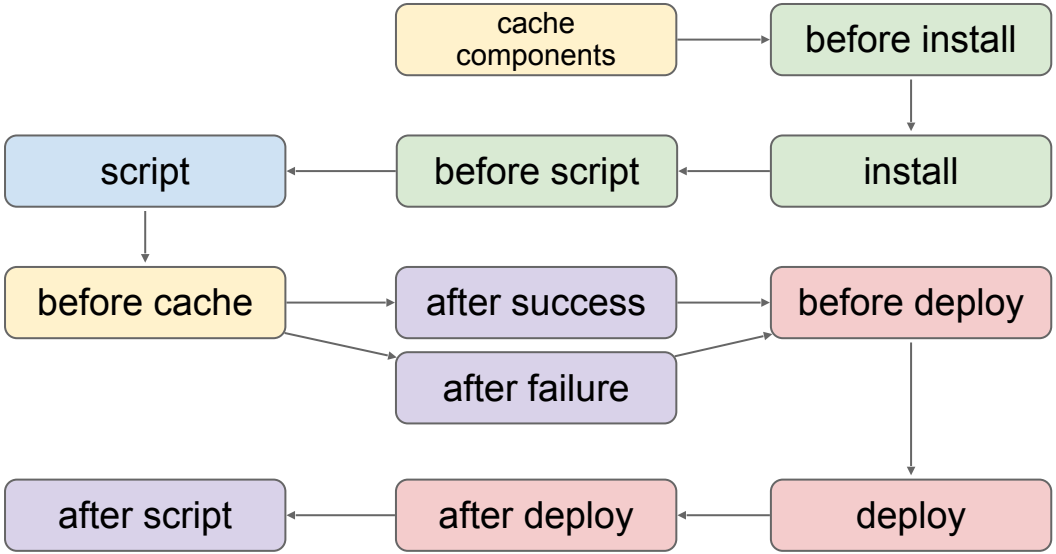
database

compile

test

deployment

Travis CI



circleci

- machine
- checkout
- dependencies
- database
- compile
- test
- deployment

O que mais?

Mais etapas para qualificar o pipeline

- Core Infrastructure Initiative (CII) - **Best Practices Badge Program** https://bestpractices.coreinfrastructure.org/en
- Análise estática:
  - Code Climate
  - Code Quality (Gitlab)
  - Stylesheet (https://github.com/collections/clean-code-linters)
    - EsLint (JavaScript)
    - PyCodeStyle (Python)
    - Flake8 (Python)
    - CSSlint
    - Rubocop (Ruby)
    - OClint (C / C++ / Objective C)