tarent solutions

Maven Packaging Plugin

Robert Schuster

BED-con 2012

What?

tarent solutions

- maven plugin to create installable software artifacts
- reusing a powerful and practice-proven infrastructure

= package management system

Excursus: package mgmt system

↪ tarent solutions

- core of every (major) GNU/Linux and BSD flavor
  - embedded, desktop, server
- reproduceable way of outfitting a machine
  - same set of installed packages = identical feature set
  - consistency checks during installation/removal/upgrade
  - handling of config files (merges)
- highly integrated into system (= tools)

Excursus: package mgmt system

↪ tarent solutions

- higher level management tools available
- manage pools of machines

e.g. puppet

How?

tarent solutions

- using plugin configuration from pom.xml
  - describes package content
  - main artifact
  - aux. files: icons, D-Bus or Policy Kit descriptors, *.desktop, etc.
- project's dependencies

How?

tarent solutions

- invoke Maven:

„mvn pkg:pkg“

(or through your IDE)

How?

tarent solutions

- results are created:

```
> ls foobar/target

foobar_1.0.deb
```

How?

tarent solutions

- multiple packages per project are possible:

```
> ls foobar/target

foobar_1.0.deb
foobar-config_1.0.deb
```

How?

tarent solutions

- could also be a different packaging type:

```
> ls foobar/target

foobar_1.0.rpm
foobar-config_1.0.rpm
```

How?

tarent solutions

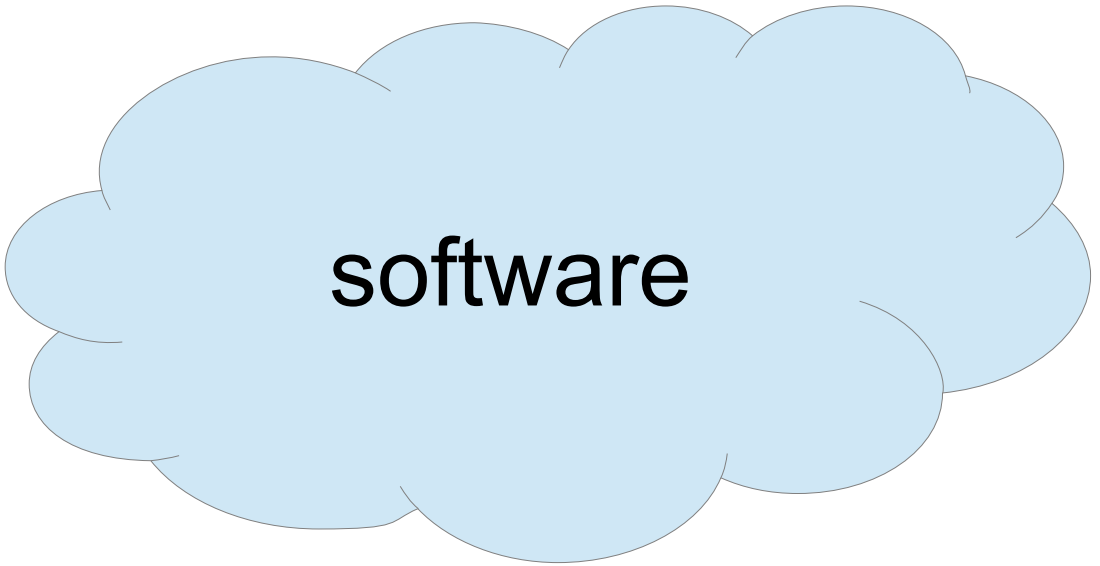
- or two configurations at the same time:

```
> ls foobar/target

foobar_1.0.deb
foobar-config_1.0.deb
foobar_1.0.rpm
foobar-config_1.0.rpm
```

Usage Concept

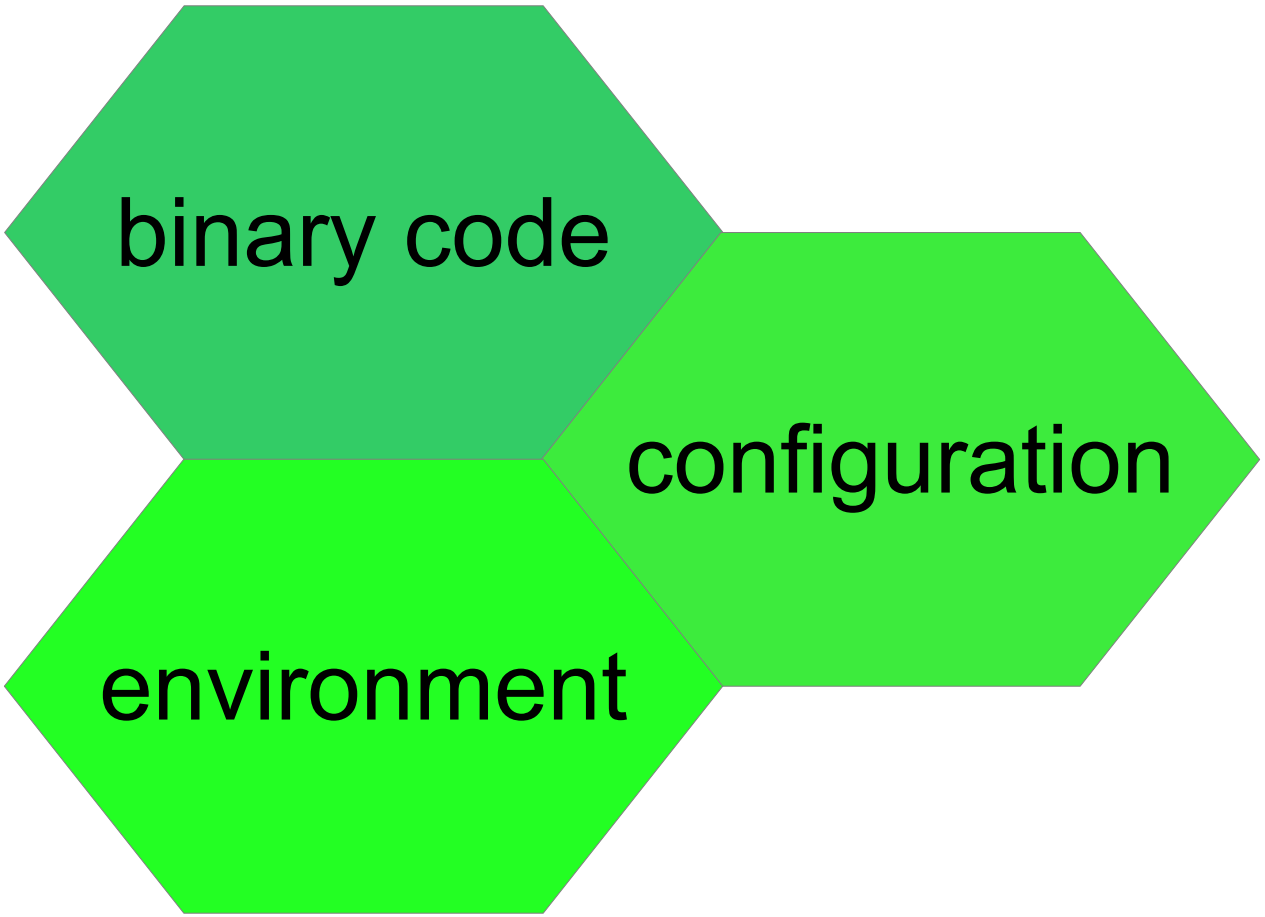
tarent solutions



source code = doesn't run

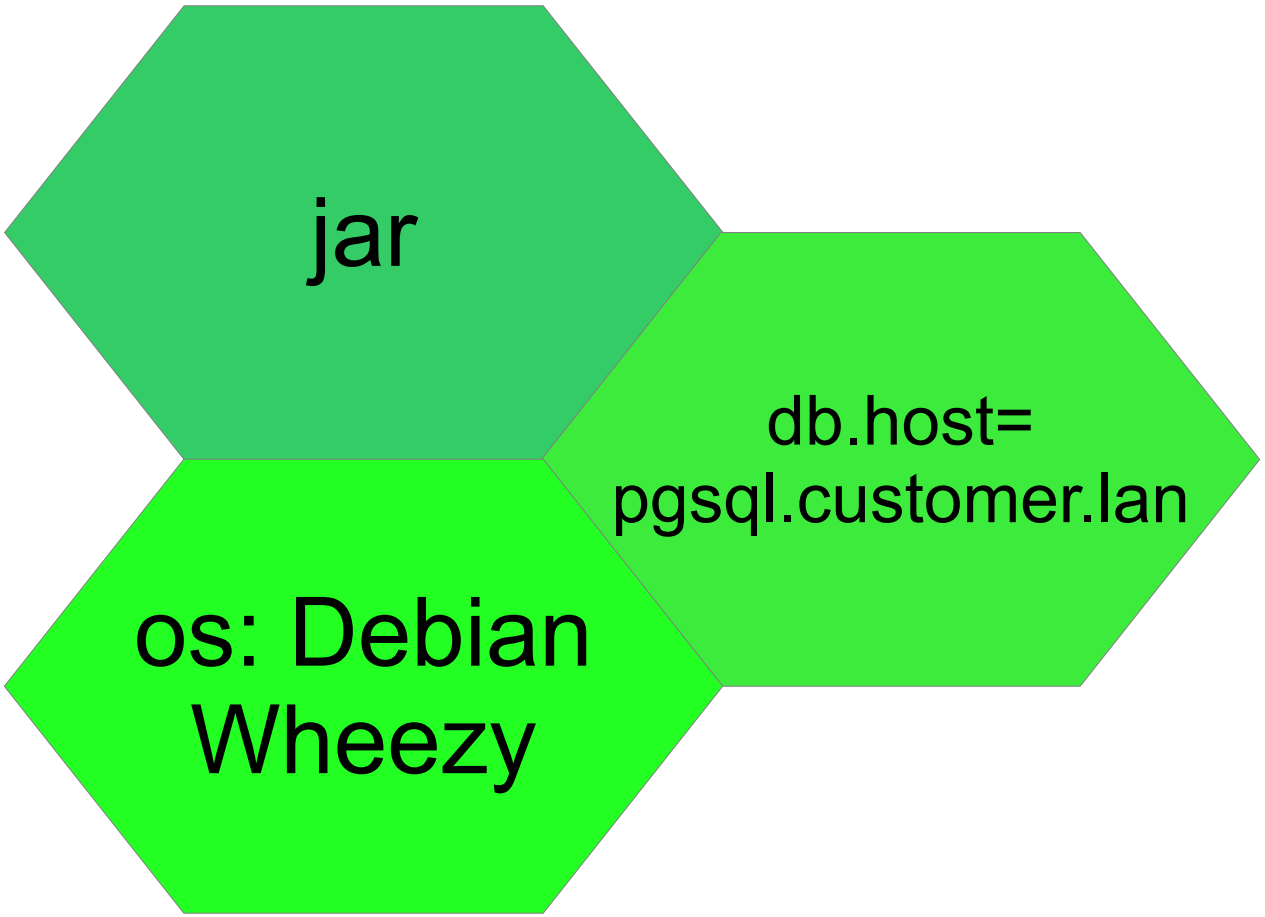
Usage Concept

tarent solutions



Usage Concept

tarent solutions



Usage Concept

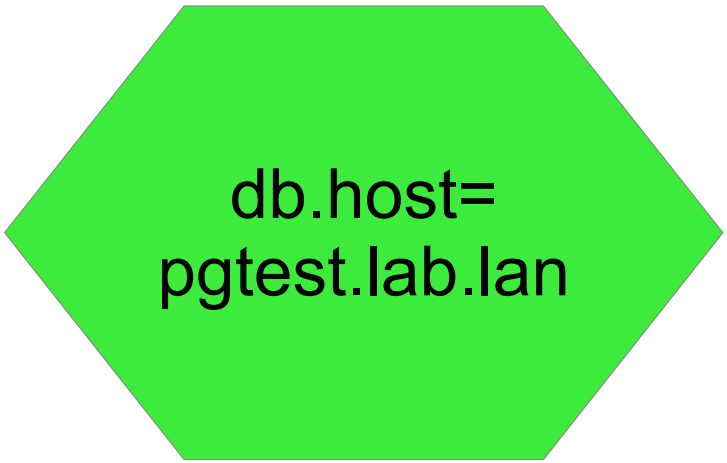
tarent solutions

- the tuple (binary code, configuration, environment) is a TargetConfiguration
- whenever one of the three components changes, a new TargetConfiguration might be necessary

Example

tarent solutions

staging



production

Policy

tarent solutions

- plugin does not impose rules
- project adjusts the plugin to its needs, not the other way round

Policy

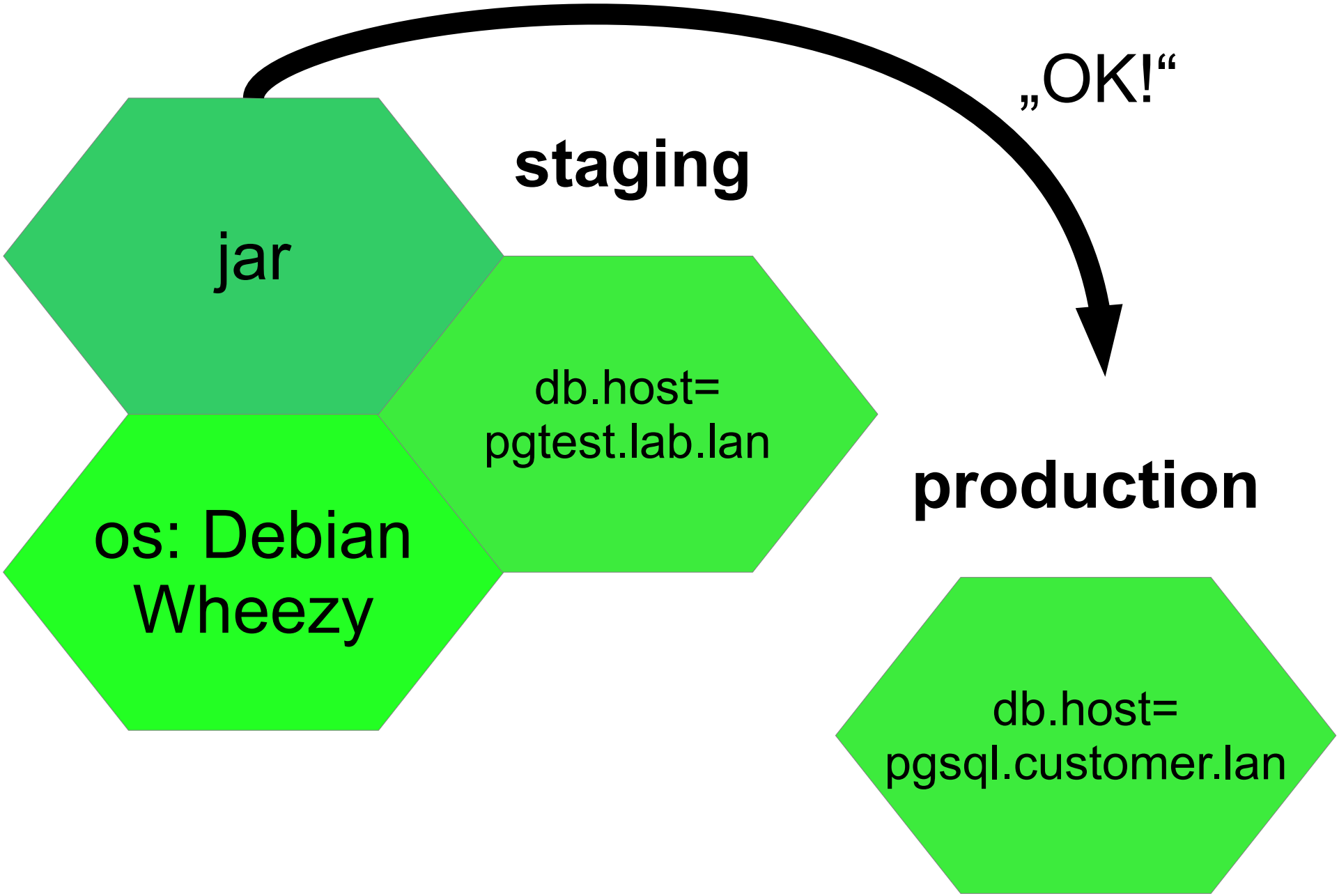
tarent solutions

- best practice
  - one package for code
  - one package per (configuration, env)

result:
same code that QA tested
and that ran through regression
and integration tests,
ends up on production system

Example cont'd

tarent solutions



The Big Picture

Jenkins

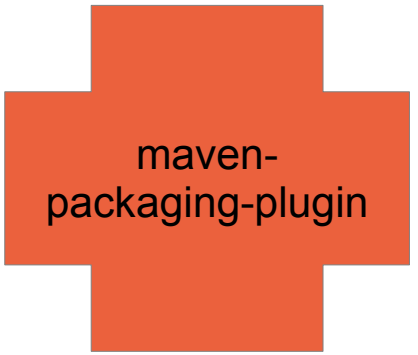
maven-packaging-plugin

puppet labs

The Big Picture

tarent solutions

- pom includes target confs for all target systems
- these are: QA, staging, production
  - single binary code package
  - 3 distinct configurations



The Big Picture

↪ tarent solutions

- on each release:
  - 4 packages are built and deployed to distinct Debian repos

Jenkins

The Big Picture

tarent solutions

- QA machines automatically upgrade
- on positive QA report
  - upgrade on staging machines
- on positive customer report
  - upgrade on production machines

puppet labs

Details to be worked out ...

- DB and/or LDAP schema migration
  - see Pedro's talk about Liquibase
- complex config file migration
- release process documentation
  - changelog, stakeholders, …
- source code provision
- complex J2EE deployment (cluster ...)

Future development

↪ tarent solutions

- Windows support
  - MSI: that's what Puppet supports
  - F/OSS pkg mgmt for Windows: http://coapp.org
- direct support for war-, ear-specifics

Resources

tarent solutions

- http://mvn-pkg-plugin.evolvis.org/
  - Quick Start Guide, Step By Step Guide,
  - User Manual
  - scm, bug/feature tracker, mailing-list

Danke!

Thanks!

¡Gracias!

Obrigado!

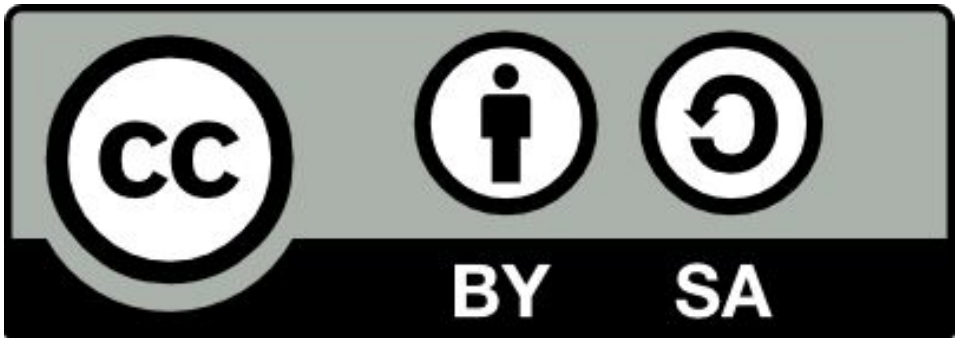
Merci!

Děkuji!

Teşekkürler!

License

tarent solutions



https://creativecommons.org/licenses/by-sa/3.0/