Exploring SlapOS Internals

by SlapOS Team.
▼ Details

You can view this tutorial in presentation mode.

Agenda

- SlapOS Node Directory Structure
- Software Release Internals
- Software Instance Internals
- SlapOS Git Repository Structure
- Cron: the magic that glues everything together

Requirements

- **Good** Buildout knowledge (via tutorials)
- SlapOS Tutorial: Day 1
- SlapOS Tutorial: Day 2
- Basic git knowledge

SlapOS Node Directory Structure

- /opt/slapos: SlapOS installation
- /opt/slapgrid: Software Releases
- /srv/slapgrid: Instances
- /etc/opt/slapos: SlapOS configuration

▼ Details

/opt/slapos

Here is installed SlapOS itself.

/opt/slapgrid

This is the directory containing all Software Releases installed on this node (wordpress, erp5, ...).

Each Software Release is self-contained in its own subdirectory, so that no Software Release can pollute another one. The name of this directory is the URL of the Software Release, hashed by MD5. So for example wordpress version "slapos-0.90", which URL is

http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.90:/software/wordpress/software.cfg

will be installed in

/opt/slapgrid/e884cf83212b2caf90bbb5cba481f58b

/srv/slapgrid

This directory contains all Computer Partitions.

/etc/opt/slapos

This directory contains all configuration files for SlapOS.

Note: location of this directory is compatible with FHS 2.3.

Software Release internals

- Buildout as build system
- Slapgrid: "just" a Buildout wrapper

▼ Details

Buildout

SlapOS uses Buildout as its build system. Imagine slapgrid as a simple wrapper of buildout. When slapgrid is run, it will run buildout to install Software Releases and to deploy instances.

Software Profiles

SlapOS uses Buildout to install software and deploy instances.

In SlapOS, a Software Release is defined by its URL. This URL is in fact a Buildout Profile. When SlapOS installs a Software Release, it launches Buildout with this URL. For example, to install wordpress, slapgrid will run something like (simplified):

```
buildout -c http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.90:/software/wordpress/software.cfg
```

in the correct directory.

Once a Software Release has successfully been installed, it will be frozen, i.e SlapOS Node will just check for its existence but won't process it anymore.

Instance Internals (1)

Deployment of an instance:

- Slapgrid runs instance.cfg of Software Release
- Recipes used by this profile are in slapos.cookbook

▼ Details

While installing a Software Release, as a convention, buildout generates an "instance.cfg" file, describing how to deploy the service.

When deploying an instance of a Software Release, slapgrid will use this file, running something like (simplified):

```
cd /srv/slapgrid/slappart12/
cp /opt/slapgrid/e884cf83212b2caf90bbb5cba481f58b/instance.cfg .
buildout
```

SlapOS will process (i.e run Buildout) each instance at least once per day by default. This periodicity can be overwritten by the Software Release by creating a "$SOFTWARE_RELEASE_INSTALL_DIRECTORY/periodicity" text file containing the periodicity in seconds.

Instance Internals: recipes

Example of recipes:

- "Create a directory"
- "Create a configuration file"
- "Deploy Apache"
- "Send service URL to SlapOS Master"

▼ Details
This insance.cfg file uses custom SlapOS recipes. If you have read the Buildout tutorial, you know that recipes are Python Eggs. Recipes achieving high quality level are included in a single Egg: slapos.cookbook. You can of course use your own egg(s).

Git repository architecture

slapos.git, the official git repository containing all the needed profiles and recipes, is separated into several directories.

- /software: all end-user services
- /component: all libraries/binaries used in software
- /stack: easy-to-use bundles, like LAMP.
- /slapos/recipe: official instantiation recipes.

▼ Details

/software

Example: Wordpress

You can find here all Software Releases, which can be considered as "end user services". Each Software Release represents a full service (like Wordpress, or ERP5).

In each Software Release directory, there is at least a "software.cfg" Buildout profile.

Because it is a Buildout profile, it can reuse other Buildout profiles from "component" or "stack" directory using the Buildout "extends" mechanism.

/component

Example: Apache, MySQL, OpenSSL

Here are all atomic components (all libraries or binaries, like Apache, MariaDB, CouchDB, cURL, OpenSSL, libxml, ncurses, ...) used in the Software Releases.

Those profiles describe how to install a component: how to compile it, how to fetch it, what are the dependencies...

/stack

Example: LAMP

If you need, you can also use what is called "stack". A stack is a group of components working together, in a reusable way. Most stack provide an instance.cfg file.

For example, when creating a PHP Sofware Release, you can extend the "LAMP" stack, containing all what is needed to run PHP applications (Apache, MySQL, etc) so that you can define only what is specific to your software in a few lines. Your software.cfg extending lamp stack will only contain a few lines.

Note: a special stack is stack/slapos.cfg, defining what is usually needed in every Software Releases, like using a special version of Buildout or fetching the slapos.cookbook egg. It should be always used to ease development of Software Release.

/slapos/recipe

Example: Apache deployment description

Here are all recipes used for deployment of instances, called from instance.cfg file of a Software Release. This directory forms content of the "slapos.cookbook" egg.

Cron and SlapOS commands

All the SlapOS Node processes are run automatically via simple cron entries:

- "slapos node software": install Software Releases
- "slapos node instance": deploy Instances
- "slapos node report": destroy instances
- "slapos node format": check/add IPs

▼ Details

Of course, you can manually run those commands if you want to.