Pysa

Thibault BRONCHAIN <thibault@mc2.io>
Ken CHEN <ken@mc2.io>
Michael CHO <michael@mc2.io>

Date: 2013-07-23 (Wed, 24 Jul 2013)

Contents

Info Reverse Engineer Server Configurations

Organisation (c) 2013 - MADEIRACLOUD LTD.

Revision v0.2.5a

Description Pysa scans your system and reverse engineers its configurations for easy replication.

NAME

pysa - Reverse Engineer Server Configurations

SYNOPSIS

pysa [**-hqps**] [**-m** *module-name*] [**-o** *output-path*] [**-c** *config-file-path*] [**-f** *filter-config-path*]

DESCRIPTION

pysa scans your system and reverse engineers its configurations for easy replication.

pysa is able to scan your system, looking for different resources to deploy and generates some "autoconf" tools script to deploy it later on another computer.

See RESOURCES_ section for complete list of managed resources.

pysa is able to generates the configuration in Puppet (see Puppet documentation) or SaltStack (see SaltStack documentation) format.

OPTIONS

Options list

-h, –help

Display the short help.

-q, –quiet

Activate quiet mode and displays only error messages. By default, **pysa** displays all log messages.

-p, –puppet

Generates Puppet output.

-s, –salt

Generates SaltStack output.

-m module-name, –module module-name

Choose output module name. Default value: *pysa*

-c config-file-path, –config config-file-path

Specify a configuration file. See examples file for more details *pysa.cfg*

-o output-path, –output output-path

Choose the output filter for generated scripts. Default value: *output*

-f filter-config-path, –filter filter-config-path

Specify a filter configuration file. See FILTERS_ section for more details.

REPLICATION

pysa generates a puppet module containing several configuration scripts.

There are two ways to use **pysa** 's output:

- You can manually configure the configuration manager and add **pysa** 's module to it
- If you're using Puppet module, you can use the *pysa2puppet* script to deploy a complete and standalone setup. The script is interactive and will ask you all necessaries info (see usage first). A SaltStack version will be published soon.

RESOURCES

This section describes all the resources scanned by **pysa**

By default, all item described are scanned. However, you can apply some filters to avoid or specify some. See the FILTERS_ section.

At the current state, the resources objects and keys are similar to Puppet types. Jump to *pysa/scanner/object/* for a compelte object description. These objects will be documented soon.

Please see AUTOCONF TOOLS MODULES_ section to be sure to be able to handle all scanned resources.

Pleese note that in main cases, the scan must be done by an admin user (mostly root).

configuration files - file

pysa scans (and stores in output module) all files located in a specific location. Default **/etc** and **/root/.ssh**

Primary key: *path*

packages - package

pysa is able to scan all packages provided by *yum*, *apt-get*, python pypi (*pip*), ruby *gems*, nodejs packaged modules (*npm*) and php packages managers (*pear* and *pecl*).

Furthermore, **pysa** is able to detect repositories *rpm* packages if *yum* is not present.

Primary key: *name*

services - service

pysa detects all startup services managed by *upstart* and *SysV init* scripts.

Please see NOTES_ section.

Primary key: *name*

hosts - host

pysa scans and reproduces existing hostname associations (default */etc/hosts*).

Primary key: *name*

users - user

pysa scans and reproduces existing users (*/etc/passwd*).

Primary key: *name*

groups - group

pysa scans and reproduces existing groups (*/etc/groups*).

Primary key: *name*

mounts - mount

pysa scans and reproduces existing mount points (*/etc/fstab*).

Primary key: *device*

crons - cron

pysa scans and reproduces user's crons.

Primary key: *name*

ssh keys - key

pysa scans and reproduces root SSH keys (default */root/.ssh*).

SSH keys are manages as files.

Primary key: *name*

sources repositories - source

pysa is able to recognize all source repositories managed by the most common SCM (*subversion*, *git* and *mercurial*) present in the system.

Primary key: *path*

Puppet only The sources scanner is not able to scan sources repositories for SaltStack yet.

package managers repositories - repository

pysa scans and reproduces *yum* and *apt-get* repositories.

Primary key: *name*

AUTOCONF TOOLS MODULES

This section lists the autoconf tools' modules which may be used.

Modules are used for particular features and are only needed in some particular cases. Of course, modules (as with the autoconf tools) have to be installed on the new machine, not the original one.

Puppet modules

willdurand/nodejs: add *npm* package manager support nodes/php: add *php* package manager support puppetlabs/vcsrepo: add *scm* (sources) support

to install a Puppet module: puppet module install *module-name*

FILTERS

pysa integrates a powerful filters engine, which allows you to adapt its behavior to your needs.

A filter file is composed of sections, keys and values. In some specific cases sections and/or keys can be split using a '.' (see below for more details).

A key can be tagged with '_' at the front to be considered as "action" key. An action key is a key representing a specific action in the section (see below).

If some parameters conflict then the result may be harmful, please use it carefully. Don't hesitate to report any abnormal output to us.

Some improvements are planned in this section.

common action keys

__contentrefer

This key acts as a pointer. All the content of the referred section will be interpreted in the section.

This key should be set alone, as all keys will be replaced.

addition section

addition section description

This section is used to add or modify some values.

It can sounds similar to the replace section, but works in a completely different way:

- The key is based on section key instead of content to replace
- It is replaced at the scanning step, while the *replacement* section is done at the output generation step

Remember that *addition* is used to add/set a concrete parameter, while *replace* is used to replace some content.

The section name can be separate in multiple subsections using a dot '.', always starting by *addition* keyword:

- addition.resource_type will replace values for all objects of resource_type
- addition.resource_type.key.value will replace only the values for the objects where the key/value match the requirement

The key represents the resource key. The value represents the resource value.

addition section format

section_key = section_value

addition section action keys

No action key for this section.

discard section

discard section description

This section is used to specify which object should or shouldn't be discard.

The key is separated in to two sub-keys by a dot '.', which represents the object type for the first one and the attribute name for the second one.

The values can be seen as a list of attributes separated by a coma ','.

The joker '*' can be used to specify to match all characters.

discard section format

object.attribute_name = attribute1, attribute2*, ...

discard section action keys

__resources: resource names Select which resources to be scanned, use it carefully, some resources might depend on others.

replace section

replace section description

This section is used to replace any kind of content.

The section name can be separated into multiple subsections using a dot '.', always starting by *replace* keyword:

- replace will replace all values for all objects.
- replace.object will replace all values for the selected object.
- replace.object.field will replace only the values associated with the field in the selected object.

The key represents the new value. The value(s) represents the target to replace.

replace section format

new_value = old_value1, old_value2, ...

replace section action keys

_replaceall:

- true/false
- REQUIRED
- Select the filtering mode (replace all except -true- or replace none except -false-)
- default: true _except: primary_keys_values

update section

update section description

This section is used to specify which *package* should be updated. This section has been created due to the lack of old packages in many repositories.

A list of package names is specified as values of the *except* key, separated by a coma ‘,’.

The joker ‘*’ can be used to specify to match all characters.

update section format

except = package1, package2*, *package3, *package4*, ...

update section action keys

_update:

- true/false
- REQUIRED
- Select the filtering mode (update all except -true- or update none except -false-)
- default: false

USAGE EXAMPLES

See *docs/examples* for configuration file examples.

NOTES

pysa has been inspired by a software called *Blueprint* (more information at http://devstructure.com/blueprint/).

The force of **pysa** lies on the following points:

- **pysa**'s "filters" and *Blueprint*'s "rules" are totally different. Please refer to the documentations for more details.
- **pysa**'s *Puppet* output is cleaner (the files are separated, the module is automatically created...).
- The dependency cycle is more resilient. **pysa** generates an attribute-based dependency cycle (each object relies and depends on its own dependencies) so if something fails the whole process isn't stopped.
- **pysa** is under active development and there is additional functionality under development (e.g., integration to *Madeira*'s services, *Salt*/*Chef* modules).

As an early-release, **pysa** does not (always) provide 100% functional results. This comes, in some cases, from the architectural choices that we've made. For example, **pysa** does not (yet) support the addition of user's packages, simply because we can't ensure the availability of these packages on the new system. It would lead to the generation of wrong output files.

Furthermore, **pysa** depends on "autoconf" tools. This means that if a feature is not supported by one of these tools, **pysa** can't provide it. For example, it is currently impossible to use upstart services on a *Redhat* based platform, as it is impossible to use npm package manager on *Ubuntu*.

Please don't hesitate to contact us for any kind of feedback, advice or requirement: pysa-user@googlegroups.com for public discussions and pysa@mc2.io for private messages.

If you have a question about a specific source file, you can also contact the author directly (first-name@mc2.io)

BUGS

No known bugs.