
Lago Documentation

Release 0.3

David Caro

January 22, 2016

| | | |
|----------|--------------------------------------|----------|
| 1 | Getting started | 1 |
| 2 | Releases | 3 |
| 2.1 | Release process | 3 |
| 3 | Contents | 7 |
| 3.1 | lago package | 8 |
| 3.2 | lago_template_repo package | 8 |
| 3.3 | ovirtlago package | 8 |
| 4 | Indices and tables | 9 |

Getting started

Check out the awesome README!

Releases

2.1 Release process

2.1.1 Versioning

For Iago we use a similar approach to semantic versioning, that is:

```
X.Y.Z
```

For example:

```
0.1.0
1.2.123
2.0.0
2.0.1
```

Where:

- Z changes for each patch (number of patches since X.Y tag)
- Y changes from time to time, with milestones (arbitrary bump), only for backwards compatible changes
- X changes if it's a non-backwards compatible change or arbitrarily (we don't like Y getting too high, or big milestone reached, ...)

The source tree has tags with the X.Y versions, that's where the packaging process gets them from.

On each X or Y change a new tag is created.

For now we have only one branch (master) and we will try to keep it that way as long as possible, if at some point we have to support old versions, then we will create a branch for each X version in the form:

```
vX
```

For example:

```
v0
v1
```

There's a helper script to resolve the current version, based on the last tag and the compatibility breaking commits since then, to get the version for the current repo run:

```
$ scripts/version_manager.py . version
```

2.1.2 RPM Versioning

The rpm versions differ from the generic version in that they have the `-1` suffix, where the `-1` is the release for that rpm (usually will never change, only when repackaging without any code change, something that is not so easy for us but if there's any external packagers is helpful for them)

2.1.3 Repository layout

Tree schema of the repository:

```
lago
-- stable <-- subdirs for each major version to avoid accidental
| |             non-backwards compatible upgrade
| |
| | -- 0.0 <-- Contains any 0.* release for lago
| |   -- ChangeLog_0.0.txt
| |   -- rpm
| |     -- el6
| |     -- el7
| |     -- fc22
| |     -- fc23
| |   -- sources
| -- 1.0
|   -- ChangeLog_1.0.txt
|   -- rpm
|     -- el6
|     -- el7
|     -- fc22
|     -- fc23
|   -- sources
| -- 2.0
|   -- ChangeLog_2.0.txt
|   -- rpm
|     -- el6
|     -- el7
|     -- fc22
|     -- fc23
|   -- sources
-- unstable <-- Multiple subdirs are needed only if branching
  -- 0.0 <-- Contains 0.* builds that might or might not have
  | |             been released
  | -- latest <--- keeps the latest build from merged, static
  | |             url
  | -- snapshot-lago_0.0_pipeline_1
  | -- snapshot-lago_0.0_pipeline_2
  | |             ^ contains the rpms created on the pipeline build
  | |             number 2 for the 0.0 version, this is needed to
  | |             ease the automated testing of the rpms
  | -- ... <-- this is cleaned up from time to time to avoid
  |             using too much space
  -- 1.0
  | -- latest
  | -- snapshot-lago_1.0_pipeline_1
  | -- snapshot-lago_pipeline_2
  | -- ...
  -- 2.0
```



```
-- latest
-- snapshot-lago_2.0_pipeline_1
-- snapshot-lago_2.0_pipeline_2
-- ...
```

2.1.4 Promotion of artifacts to stable, aka. releasing

The goal is to have an automated set of tests, that check in depth lago, and run them in a timely fashion, and if passed, deploy to stable. As right now that's not yet possible, so for now the tests and deploy is done manually.

The promotion of the artifacts is done in these cases:

- New major version bump ($X+1.0$, for example $1.0 \rightarrow 2.0$)
- New minor version bump ($X.Y+1$, for example $1.1 \rightarrow 1.2$)
- If it passed certain time since the last X or Y version bumps ($X.Y.Z+n$, for example $1.0.1 \rightarrow 1.0.2$)
- If there are blocking/important bugfixes ($X.Y.Z+n$)
- If there are important new features ($X.Y+1$ or $X.Y.Z+n$)

2.1.5 How to mark a major version

Whenever there's a commit that breaks the backwards compatibility, you should add to it the pseudo-header:

```
Sem-Ver: api-breaking
```

And that will force a major version bump for any package built from it, that is done so in the moment when you submit the commit in Gerrit, the packages that are build from it have the correct version.

After that, make sure that you tag that commit too, so it will be easy to look for it in the future.

2.1.6 The release procedure on the maintainer side

1. Select the snapshot repo you want to release
2. **Test the rpms, for now we only have the tests from projects that use it:**
 - Run all the [ovirt tests](#) on it, make sure it does not break anything, if there are issues -> [open bug](#)
 - **Run [vdsm functional tests](#), make sure it does not break anything, if** there are issues -> [open bug](#)
3. **On non-major version bump $X.Y+1$ or $X.Y.Z+n$**
 - [Create a changelog](#) since the base of the tag and keep it aside
4. **On Major version bump $X+1.0$**
 - [Create a changelog](#) since the previous $.0$ tag ($X.0$) and keep it aside
5. Deploy the rpms from snapshot to dest repo and copy the `ChangeLog` from the tarball to `ChangeLog_X.0.txt` in the base of the `stable/X.0/` dir
6. Send email to [lago-devel](#) with the announcement and the changelog since the previous tag that you kept aside, feel free to change the body to your liking:

Subject: [day-month-year] New lago release - X.Y.Z

Hi everyone! There's a new lago release with version X.Y.Z ready for you to upgrade!

Here are the changes:

<CHANGELOG HERE>

Enjoy!

Contents

3.1 lago package

3.1.1 Subpackages

`lago.plugins` package

Submodules

`lago.plugins.cli` module

3.1.2 Submodules

3.1.3 `lago.brctl` module

3.1.4 `lago.config` module

3.1.5 `lago.constants` module

3.1.6 `lago.dirlock` module

3.1.7 `lago.log_utils` module

3.1.8 `lago.paths` module

3.1.9 `lago.subnet_lease` module

3.1.10 `lago.sysprep` module

3.1.11 `lago.templates` module

3.1.12 `lago.utils` module

3.1.13 `lago.virt` module

3.2 `lago_template_repo` package

3.3 `ovirtlago` package

3.3.1 Submodules

Indices and tables

- `genindex`
- `modindex`
- `search`