CloudStack Maintainers Guide

Preface

CloudStack is an open source project that lives and grows thanks to the hard work of a large number of contributors (of both source code and documentation). The code and documentation changes suggested by community members are taken by a subset of committers who have volunteered to maintain and nurture specific sections of the CloudStack project.

The CloudStack maintainers bear several responsibilities

- Review, and potentially acceptance, of code changes from the community. Contributors and maintainers share responsibility for testing that new contributions work and do not break the application, and that the code changes are of high quality.
- Mentorship of non-maintainers. Another role of the maintainer is to mentor non-maintainers into productive, recurring contributors or maintainers in the future.

While maintainers may develop and contribute code changes to CloudStack, more inherent in their role is that they act as a nurturing interface between the CloudStack community and the source tree.

It is important to note that maintainers are not so much granted *authority* as they are given *ownership*. Summed up, a maintainers role is to make their module shine, and to foster community growth around their module and CloudStack in general.

CloudStack Code Maintainers

Maintainer Requirements

CloudStack maintainers are skilled developers (or document writers) who have significant experience developing (or documenting) CloudStack. These individuals are familiar with the CloudStack coding standards, and are capable reviewers of code submissions.

Maintainer Scope

Each CloudStack maintainer has a specific module for which they are responsible and is to be their primary focus. They are intimately familiar with the module they are responsible for, act as the primary point of code review for code change submissions to the module, and interact with other members of the community on development topics related to the module and it's functionality. In general, maintainers only have commit rights on the module for which they are responsible.

Proven Maintainers

A select group of maintainers who have a history of successful contributions across multiple modules belong to a "Proven Maintainers" group. Proven Maintainers have commit rights for the whole source tree. The intention of this is not to provide access for working on more modules, but to allow a Proven Maintainer to help out in the event that another module's maintainer is unable to perform his or her duties. Proven Maintainers are still primarily responsible for the module for which they are the maintainer.

Current Maintainers Per Component

The following are the list of maintainers for the major components of CloudStack as volunteered by the community.

Component	Maintainers	JIRA subscriptions

CloudStack API	Alena Prokharchyk, Rohit Yadav	Subscribe
EC2 API	Prachi Damle, Likitha	Subscribe
S3 API	Chiradeep Vittal	
UI	Rohit Yadav, Brian Federle, Jessica Wang , Pranav Saxena	
Storage/Volumes	Chiradeep Vittal	
Networking	Rohit Yadav, Sheng Yang, Alena Prokharchyk, Chiradeep Vittal, Murali Reddy, Hugo Trippaers	
Deployment Planners /Allocators	Prachi Damle , Nitin Mehta	
System VM/Virtual Router	Sheng Yang , Jayapal Reddy, Rohit Yadav	
Console Proxy	Kelven Yang, Rohit Yadav	
Snapshots	Anthony Xu	
XenServer	Anthony Xu , Abhinandan Prateek , Devdeep Singh	Subscribe
XCP	Devdeep Singh	
KVM	Edison Su , Wido den Hollander, Rohit Yadav	Subscribe
VMWare	Kelven Yang, Sateesh, Vijayendra, Rohit Yadav	Subscribe
OVM	Frank Zhang	
Hyper-V	Anshul, Rajesh, Devdeep	
BareMetal	Frank Zhang	Subscribe
Authenticators/LDAP	Abhinandan Prateek ,Hugo Trippaers, Rohit Yadav	
VM Sync/HA	Abhinandan Prateek , Anthony Xu, Rohit Yadav	
Usage	Kishan Kavala	
Events	Kishan Kavala, Rohit Yadav	
Database and Upgrades	Kishan Kavala, Rohit Yadav	
Templates	Nitin Mehta	
Capacities	Nitin Mehta	
Netscaler	Rajesh Battala, Murali Reddy	
Simulator	Prasanna Santhanam	
cloudmonkey	Rohit Yadav	
Marvin	Prasanna Santhanam, Rohit Yadav	
SRX	Jayapal Reddy	
F5	???	
Setup/Install	Frank Zhang , Wido den Hollander, Hugo Trippaers, Rohit Yadav	
Ubuntu (Deb) Packaging	Wido den Hollander, Rohit Yadav	
RHEL (RPM) Packaging	Hugo Trippaers, Rohit Yadav	
Maven Build System	Hugo Trippaers, Edison Su, David Nalley, Rohit Yadav	
Website	Joe Brockmeier, Rohit Yadav	
Documentation	Joe Brockmeier, Rohit Yadav	

Code Testing and Code Quality

Besides reviewing code change submissions for logic and coding standard practices, maintainers are expected to test submissions to a reasonable level to ensure that inclusion of the change does not introduce defects into the source. If a code change to the source tree is found to break the build, the change must be reverted, with the maintainer to troubleshoot the issue.

A suite of automated QA tools does not yet exist for CloudStack, but as a suite is created, maintainers are expected to test submissions against the suite before committing changes to CloudStack.

Development Workflow

Details about CloudStack's git development workflow can be found at Git workflow in the brave new world

Branching

The CloudStack Source tree is made up of several branches: The source trunk, release branches, and feature branches.

The source trunk is the active main development tree for CloudStack ("master"). Except for feature branches, this will always have the latest code.

Release branches represent code used to create a General Availability release. Approximately one month before the GA release, a branch from HEAD will be made, named after the version to be released (e.g. "release-4.0"). From within that branch, Release Candidate packages will be created and tested until the community feels the quality of the code is sufficient for a GA release. Each time a Release Candidate is made, that revision of the code will be tagged appropriately (e.g. "release-4.0RC1"). Once the GA release is made, no more changes will be applied to that branch. Any defects found after a GA release will be addressed in trunk, and released in a future release of the product.

When working on a new feature that requires significant changes or work from multiple contributors to CloudStack, a feature branch may be created for this development work. Once the changes are complete and are merged into trunk, the feature branch will be removed.

Individual contributors or groups of contributors are welcome to maintain their own private repository for development purposes. When code changes are ready to be submitted to CloudStack for inclusion in the public project, patches or git pull requests can be sent to the CloudStack Development mailing list (see Patch Submission, below).

Patch Submission

Patches are how the community submits code or documentation change requests to CloudStack maintainers. Patches are submitted under one of two conditions: either to fix a defect found in the CloudStack, or to add functionality to CloudStack.

When submitting a patch to address a defect, a bug report should be created which describes the defect, including the release version the defect was found in, and how to reproduce the defect. If the bug reporter intends to submit a patch, this should be noted in the bug report notes.

Similarly, a feature request should be documented before submitting a patch to provide new functionality.

The bug reporter/feature requester and the patch submitter do not need to be the same individual. Before performing work on a solution, a contributor should make a note on the bug/feature stating that they are working on it, presuming nobody has already made a similar statement. Once the contributor believes they have a working solution, a patch should be created using the git-format-patch command as described at CloudStack Development mailing list. If changes have been made across multiple modules, multiple patches should be created, one for each module. Patches should be sent to the CloudStack Development mailing list, along with a short description of what the patch is addressing, and any relevant bug IDs or feature request IDs.

Responsiveness to patch submissions

A maintainer will strive to respond to a patch submission within two weeks (14 calendar days) of initial submission. This is an important guideline for CloudStack; Community contributions are a main source of innovation and growth, and failure to respond to contributions may result in less submissions in the future. Repeated failure of this guideline may result in loss of maintainer status.

In the event where a maintainer fails to respond to a patch submission within a reasonable period of time, the contributor may raise the issue on the CloudStack Development mailing list.

Inability to contact submitter

In the case where somebody has stated they would work on the issue, and a significant period of time has passed without a patch submission, the maintainer should contact the individual to check for an update. If, after 30 days, the maintainer is unable to contact the submitter, the bug/feature should be released for others to submit patches against.

Release Workflow

CloudStack intends to produce stable "General Availability" ("GA") for public consumption three times per calendar year. Approximately one month before an intended release date, the CloudStack development community will begin preparing for the GA release by creating Release Candidate ("RC") releases

With the goal of a "general availability" release, Apache CloudStack will use the following version labels to notate progress towards a GA release. In the future, the project may add additional labels (such as "Beta") or further voting to ensure the quality of releases.

Keep in mind that both Release Candidate and General Availability releases require voting, as described in the **Voting** section below.

Release Candidates

Approximately one month before a GA release is to be made, the CloudStack development community will package and release a Release Candidate ("RC"). This package manifests the community's interest in creating a GA release with the featureset found in the branched code, and requests the community to perform testing and QA on the RC to find bugs and other issues which should be mitigated before the GA release. Each release will be named in a format similar to "apache-cloudstack-4.0.0.RC2-incubating-src.tar.gz"

During the "RC" cycle, an RC release will be made, and the community will QA and report bugs, and another RC candidate will be made in a period between 1-2 weeks after the last. This cycle will continue until the community feels the release is reaching the quality worthy of a GA label.

General Availability Releases

Once the community feels that the QA process has found and fixed issues to create a release that will be stable and usable by the general public, a GA release can be made. Each GA release will be named in a format similar to "apache-cloudstack-4.0.0-incubating-src.tar.gz"

Voting

For all CloudStack releases (RCs or GA), a formal VOTE binding upon Apache is required before release. The only time formal voting is required is for approving a formal release of CloudStack. While in incubation, CloudStack will need at least 3 positive votes from Apache Incubator PMC members. Details about this process can be found here.

The standard submission/review/commit life-cycle should exist without need to cast votes. In the event of a disagreement, either the maintainer or contributor may bring the issue to the attention of the CloudStack Development list.

Version Numbers

Apache CloudStack releases will have a version number in the format of major.minor.point.

Changes in the major version number represents very significant changes - major updates to the code base, moving the project to a new home, etc.

Changes in the minor version number represents "standard" GA releases.

Changes in the point version number represents fixes applied to the relative GA release.

How to Perform a Release

This only includes a source code release right now. We need to evolve / edit when we get our binary build process sorted out.

Publishing Release Candidates

The following steps document the process of publishing an RC.

Assumptions:

1. The community member has gpg installed locally, and has added their public key to the KEYS file in the project's source repo.
2. The community member has commit rights on the CloudStack git repo.
3. The community member has commit rights for the CloudStack svn repo (for updating the cloudstack website).
4. The community member has commit rights for the https://dist.apache.org/repos/dist/dev/cloudstack svn repo.

CloudStack RC's are distributed via the ASF mirroring system, and are provided to the community via the http://cloudstack.apache.org/downloads.html page. In order to create the release candidate, you can follow the steps below to get it hosted within the ASF mirrors.

Steps:

Create (or identify) the release branch in git. Normally, we name release branches with the full x.y.z release number (but exclude the RCx portion that will be used during the release process). If the branch is being created, then ensure that it has been pushed to the ASF repo before moving on from here.

Run tools/build/build_asf.sh:

```
# tools/build/build_asf.sh -h
usage: tools/build/build_asf.sh -v version [-b branch] [-s source dir] [-o output dir] [-t [-u]] [-h]
  -v sets the version
  -b sets the branch (defaults to 'master')
  -s sets the source directory (defaults to ~/incubator-cloudstack/)
  -o sets the output directory (defaults to ~/cs-asf-build/)
  -t tags the git repo with the version
     -u sets the certificate ID to sign the tag with (if not provided, the default key is attempted)
  -h
```

Example:

```
# tools/build/build_asf.sh -v 4.0.0RC1 -b master -s ~/incubator-cloudstack -o ~/rc1 -t -u XXXXXXXX
Using version: 4.0.0RC1
Using source directory: /Users/apache.user/incubator-cloudstack
Using output directory: /Users/apache.user/rc1
Using branch: master
Tagging the branch with the version number, and signing the branch with certificate ID XXXXXXXX.

gpg: using PGP trust model
gpg: using subkey XXXXXXXX instead of primary key XXXXXXXX

You need a passphrase to unlock the secret key for
user: "Apache User <apache.user@apache.org>"
4096-bit RSA key, ID XXXXXXXX, created 2012-08-06 (main key ID XXXXXXXX)

gpg: writing to `cloudstack-source-4.0.0RC1.tar.gz.asc'
gpg: RSA/SHA1 signature from: "XXXXXXXX Apache User <apache.user@apache.org>"
gpg: using PGP trust model
gpg: using subkey XXXXXXXX instead of primary key XXXXXXXX

You need a passphrase to unlock the secret key for
user: "Apache User <apache.user@apache.org>"
gpg: using subkey XXXXXXXX instead of primary key XXXXXXXX
4096-bit RSA key, ID XXXXXXXX, created 2012-08-06 (main key ID XXXXXXXX)

gpg: writing to `cloudstack-source-4.0.0RC1.zip.asc'
gpg: RSA/SHA1 signature from: "XXXXXXXX Apache User <apache.user@apache.org>"
Version: GnuPG/MacGPG2 v2.0.18 (Darwin)
gpg: armor header:
Comment: GPGTools - http://gpgtools.org
gpg: armor header:
gpg: Signature made Tue Sep  4 15:48:44 2012 EDT using RSA key ID XXXXXXXX
gpg: using subkey XXXXXXXX instead of primary key XXXXXXXX
gpg: using PGP trust model
gpg: Good signature from "Apache User <apache.user@apache.org>"
gpg: binary signature, digest algorithm SHA1
Version: GnuPG/MacGPG2 v2.0.18 (Darwin)
gpg: armor header:
Comment: GPGTools - http://gpgtools.org
gpg: armor header:
gpg: Signature made Tue Sep  4 15:48:46 2012 EDT using RSA key ID XXXXXXXX
gpg: using subkey XXXXXXXX instead of primary key XXXXXXXX
gpg: using PGP trust model
gpg: Good signature from "Apache User <apache.user@apache.org>"
gpg: binary signature, digest algorithm SHA1

You need a passphrase to unlock the secret key for
user: "Apache User <apache.user@apache.org>"
4096-bit RSA key, ID XXXXXXXX, created 2012-08-06 (main key ID XXXXXXXX)
```

During the script's run, you should be prompted to enter your pass-phrase for your key during the signing process.

The output directory specified (or defaulted) should now contain the following files:

```
KEYS
cloudstack-source-4.0.0RC1.tar.gz
cloudstack-source-4.0.0RC1.tar.gz.asc
cloudstack-source-4.0.0RC1.tar.gz.md5
cloudstack-source-4.0.0RC1.tar.gz.sha
cloudstack-source-4.0.0RC1.zip
cloudstack-source-4.0.0RC1.zip.asc
cloudstack-source-4.0.0RC1.zip.md5
cloudstack-source-4.0.0RC1.zip.sha
```

You should verify the archive contents now, to confirm that they have packaged correctly. Once verified, it's time to push them into the ASF dist repo:

Check out the https://dist.apache.org/repos/dist/dev/cloudstack folder. Copy these files to that folder, issue an svn add, and svn commit with a sane message about the Release Candidate number.

⚠ We need to figure out the RC archiving policy.

Once the commit completes (it may take a good bit of time, due to the size of the CloudStack source tree), you now have to wait for the mirrors to catch up (24 hours).

After the mirrors have caught up, the cloudstack site needs to be edited. Check out https://svn.apache.org/repos/asf/cloudstack/site/trunk/content/ and edit the downloads.mdtext file's RC section to point to the new RC version.

Commit that site edit, and review it here: http://cloudstack.staging.apache.org/downloads.html

Once satisfied, use the ASF CMS to publish that page (from here: https://cms.apache.org/cloudstack/publish?diff=1)

Once completed, let the cloudstack-dev@i.a.o list know!

Publishing Approved Releases

TBD

Appendix

References:

- Linux Foundation'sHow to Participate in the Linux Community
- DrupalCore MaintainersGuide
- Apache ISIS Contributors Guide
- The Fedora Project's "Proven Packager" policy
- Apache Incubator Release Management