Stable RT Releases

What is done, what do we need?

Steven Rostedt
rostedt@goodmis.org
srostedt@redhat.com

Current supported releases

3.2-rt - May 2018

3.4-rt - Sept 2016 (Over finally?)

3.10-rt - Oct 2017

3.12-rt - Jan 2017

3.14-rt - Oct 2017 (ended)

3.18-rt - Jan 2017

4.1-rt - Sep 2017

4.4-rt - Feb 2018

4.6-rt - not going to be supported

Doing stable releases

Mostly manual with a lot of scripts

http://rostedt.homelinux.com/rt-src

Releasing a stable

$ git tag | grep v4.4

$ git merge v4.4.x

Fix issues, run quick test if necessary

$ vim localversion-rt

$ commit-release

$ push-this-branch

On main machine run: tag-rt

$ git merge v4.4.x+1 ...

Releasing a stable

$ vim localversion-rt

$ ktest.pl atest.conf (Compiles an allmodconfig)

$ ktest.pl cross.conf (Compiles cross compilers)

arm64, alpha, arm, avr32, blackfin, cris, frv, ia64, m32r,
m68k, mips64, mips, openrisc, hppa64, hppa,
powerpc, powerpc64, s390, sh, sparc64, sparc,
tilegx, i386, x86_64

Using make defconfig + CONFIG_PREEMPT_RT_FULL

$ ktest.pl test.conf

(Compiles and boots on my test box)

Releasing a stable

On test box run:

- **cyclictest -p80 -i250 -n -a -t -q -d 0**
- **hackbench in loop**
- **distcc -j20 kernel build in loop**

$ ktest.pl ctest.conf

- **Builds and boots 6 i386 configs 6 x86_64 configs**

When all tests are done kill cyclictest and check results, then run

- **rt-migrate-test -c**
- **pi_mutex_test**
- **pi_mutex_hammer**
- **for i in `seq 5`; do stress_cpu_hotplug ; done**

Releasing a stable

$ vim localversion-rt

$ commit-release

$ push-this-branch

On main machine run:

$ tag-rt

$ push-branch `this-branch`

$ make-rt v4.4.${x+1}

$ make-patches v4.4.${x+1}

$ git checkout v4.4.${x+1}-rebase

Releasing a stable

On main machine run: (in rebase branch)

$ git rebase -i v4.4.${x+1}

examine and commit

Fix any conflicts

Fix localversion-rt

$ git commit localversion-rt

Change commit log to v4.4.${x+1}-rt$y REBASE

$ tag-rt-rebase

$ push-branch `this-branch` -f --tags

$ make-tarball v4.4.${x+1}-rt$y-rebase

Releasing a stable

On main machine run:

```
$ cd ../rt/4.4
$ mv patch* older
$ mv ../tmp/patch* .
$ move-old-patches
$ sign-patches
$ upload-patches
```

Send out email using /tmp/prog-4.4.${x+1}-rt$y

Releasing a release candidate

$ git checkout -b temp v4.6-rt

is the last update used

$ git show rt-devel/linux-v4.6.y-rt

take note of last stable that was merged

$ git merge v4.6.$x

$ git cherry -v HEAD rt-devel/linux-v4.6.y-rt > /tmp/list

$ make-git-patches /tmp/list ./patches/

makes a quilt series of patches added since last time

$ git checkout v4.6-rt

$ git merge rt-devel/linux-v4.6.y-rt

Releasing a release candidate

Check each patch in quilt series

- **Give priority to Cc: stable-rt@vger.kernel.org**

$ quilt push -a

$ quilt push -f

- **fix up conflicts**

$ quilt fork ${patch}-v4.4.patch

$ quilt refresh

$ quilt refresh

$ while quilt pop; do quilt refresh; done

Releasing a release candidate

$ git quiltimport

$ git checkout -b v4.4-rt-next

$ vim localversion-rt

  Add -rc release

$ commit-release

$ push-this-branch

Test just the same as stable release

Releasing a release candidate

On main machine run:

$ git checkout v4.4-rt-next

$ make-pre

Enter release date

Makes a quilt queue in ../sendpatches/patches

$ cd ../sendpatches

$ quilt mail ...

$ cd ../rt/4.4

$ mv patch* older

$ mv ../tmp/patch* .

$ sign-patches

$ upload-patches

Releasing a new release

$ vim localversion-rt

(non -rc version)

$ push-this-branch

On main machine run:

$ tag-rt

$ push-branch `this-branch`

$ make-rt v4.4.$x-rt${y-1} v4.4.$x-rt$y

$ make-patches v4.4.$x-rt${y-1} v4.4.$x-rt$y

$ git checkout v4.4-rt-rebase

Releasing a new release

On main machine run:

$ cherry-pick v4.4.$x-rt${y-1} v4.4.$x-rt$y

Does a cherry pick of the new patches that were added

Does a rebase -i to allow me to remove old v4.4.$x-rt${y-1}

$ vim localversion-rt

Fix rebase conflict

$ tag-rt-rebase

$ make-tarball v4.4.$x-rt$y-rebase

$ cd ../rt/v4.4

Releasing a new release

On main machine run:

```
$ remove-old-patches patch-4.4.$x-rt$y-rc$z.patch.xz
$ mv -i patch* older
$ mv ../tmp/patch* .
$ sign-patches
$ upload-patches
```

Discussion

What versions do you use?

What more should be done?

Better testing?