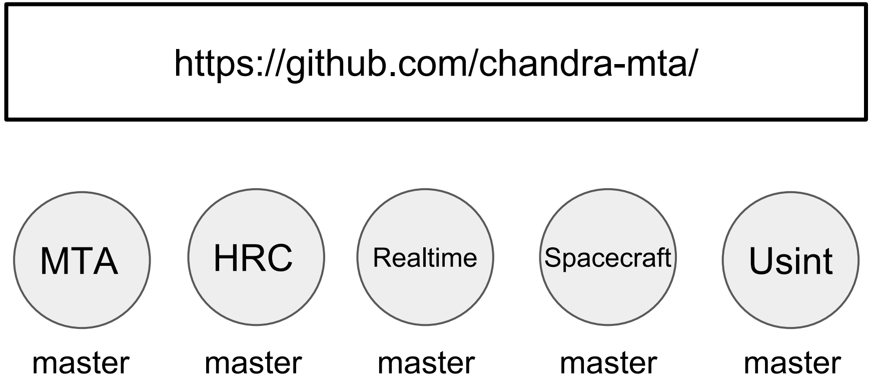


https://github.com/chandra-mta/

MTA master

HRC master

Realtime master

Spacecraft master

Usint master

- In each of the 5 repositories, the Github code on master branch should be identical to the version that is installed and running locally

- The slides below propose a workflow for the following cases:

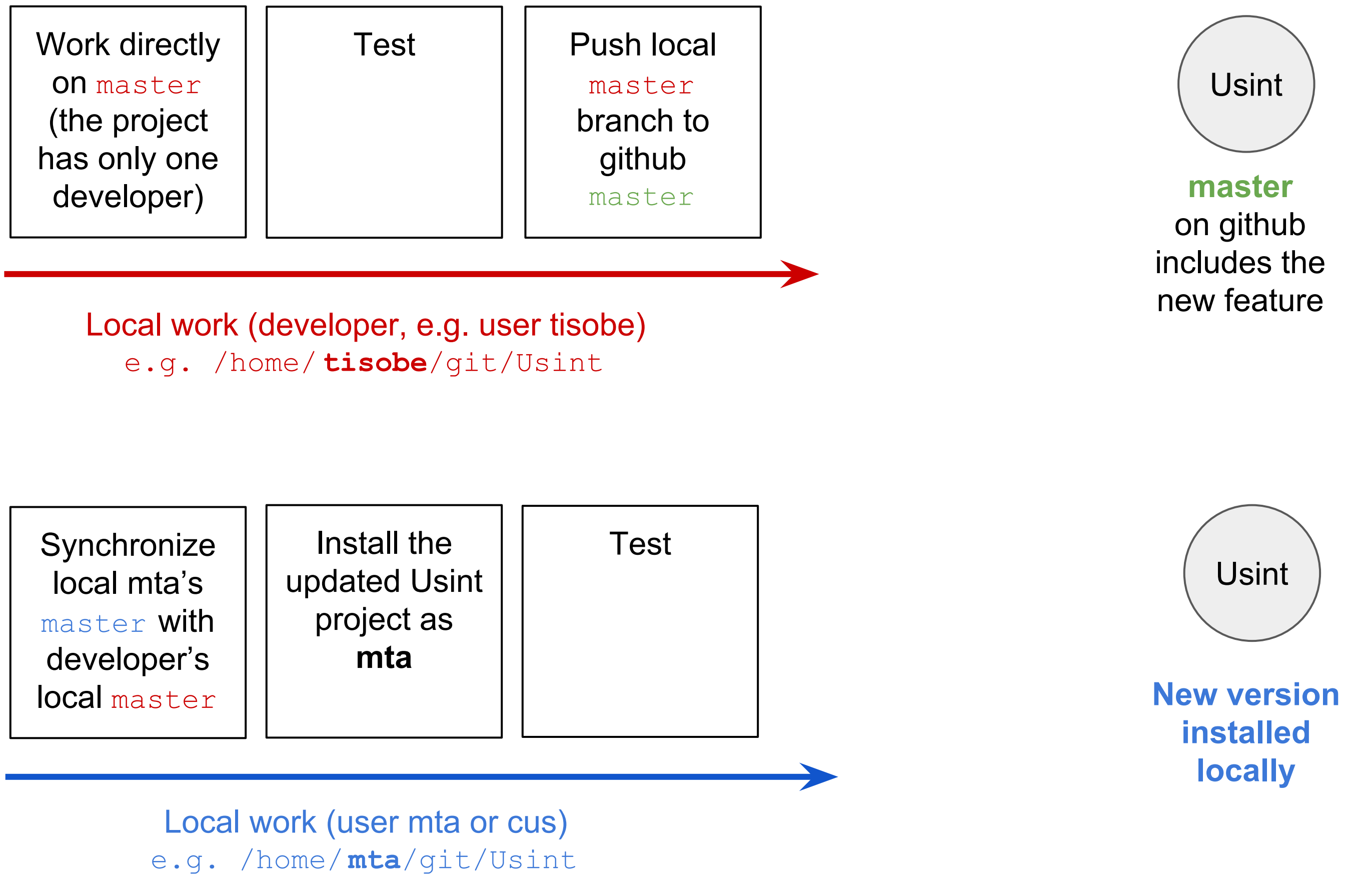
  - a project is developed by only one person, and does not require a review

  - a project is developed by more than one person, or it requires a review
    - we want to grant mta/cus access to github.com/chandra-mta
    - we do not want to grant mta/cus access to github.com/chandra-mta

The workflow of projects that

are developed by one person
and
do not require a review

The workflow of projects that are developed by only one person and do not require a review



Work directly on master (the project has only one developer)

Test

Push local master branch to github master

Usint

master on github includes the new feature

Local work (developer, e.g. user tisobe)

`e.g. /home/`**`tisobe`**`/git/Usint`

Synchronize local mta's master with developer's local master

Install the updated Usint project as **mta**

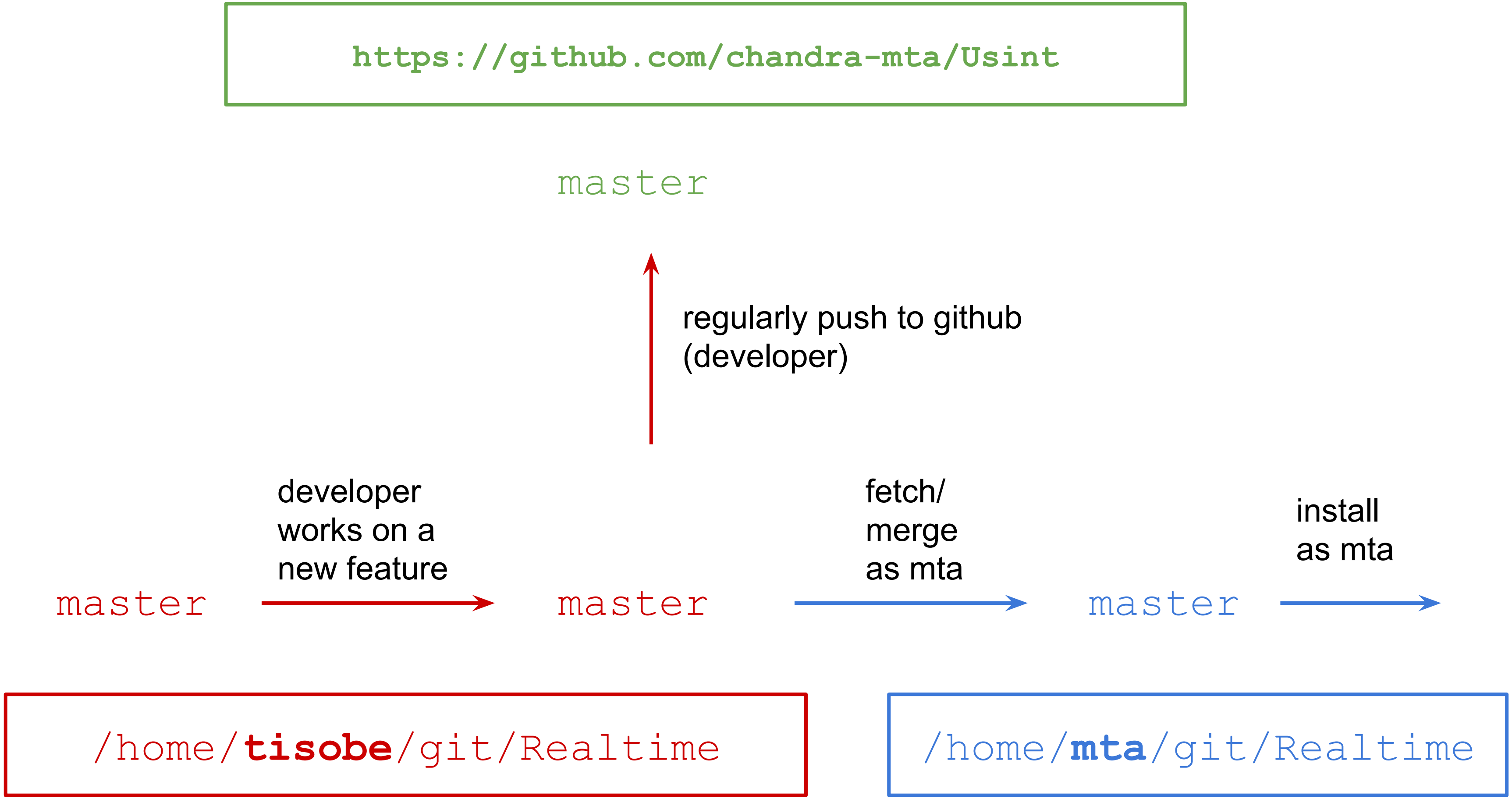
Test

Usint

New version installed locally

Local work (user mta or cus)

`e.g. /home/`**`mta`**`/git/Usint`



https://github.com/chandra-mta/Usint

master

regularly push to github
(developer)

master

developer
works on a
new feature

master

fetch/
merge
as mta

master

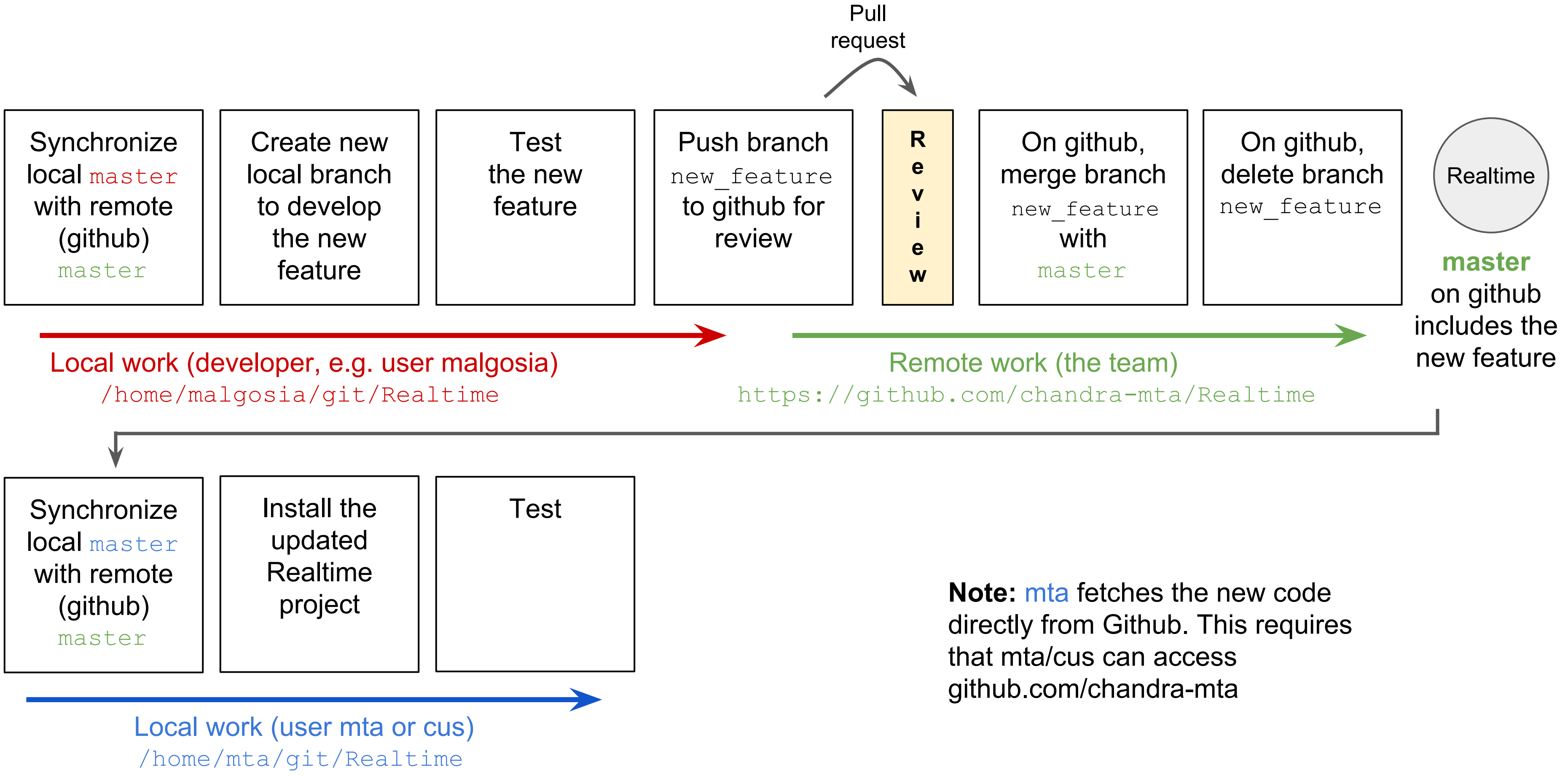
install
as mta

/home/**tisobe**/git/Realtime

/home/**mta**/git/Realtime

The workflow of projects that

are developed by more than one person
or
require a review



Synchronize local master with remote (github) master

Create new local branch to develop the new feature

Test the new feature

Pull request

Push branch new_feature to github for review

Review

On github, merge branch new_feature with master

On github, delete branch new_feature

Realtime

master on github includes the new feature

Local work (developer, e.g. user malgosia)
`/home/malgosia/git/Realtime`

Remote work (the team)
`https://github.com/chandra-mta/Realtime`

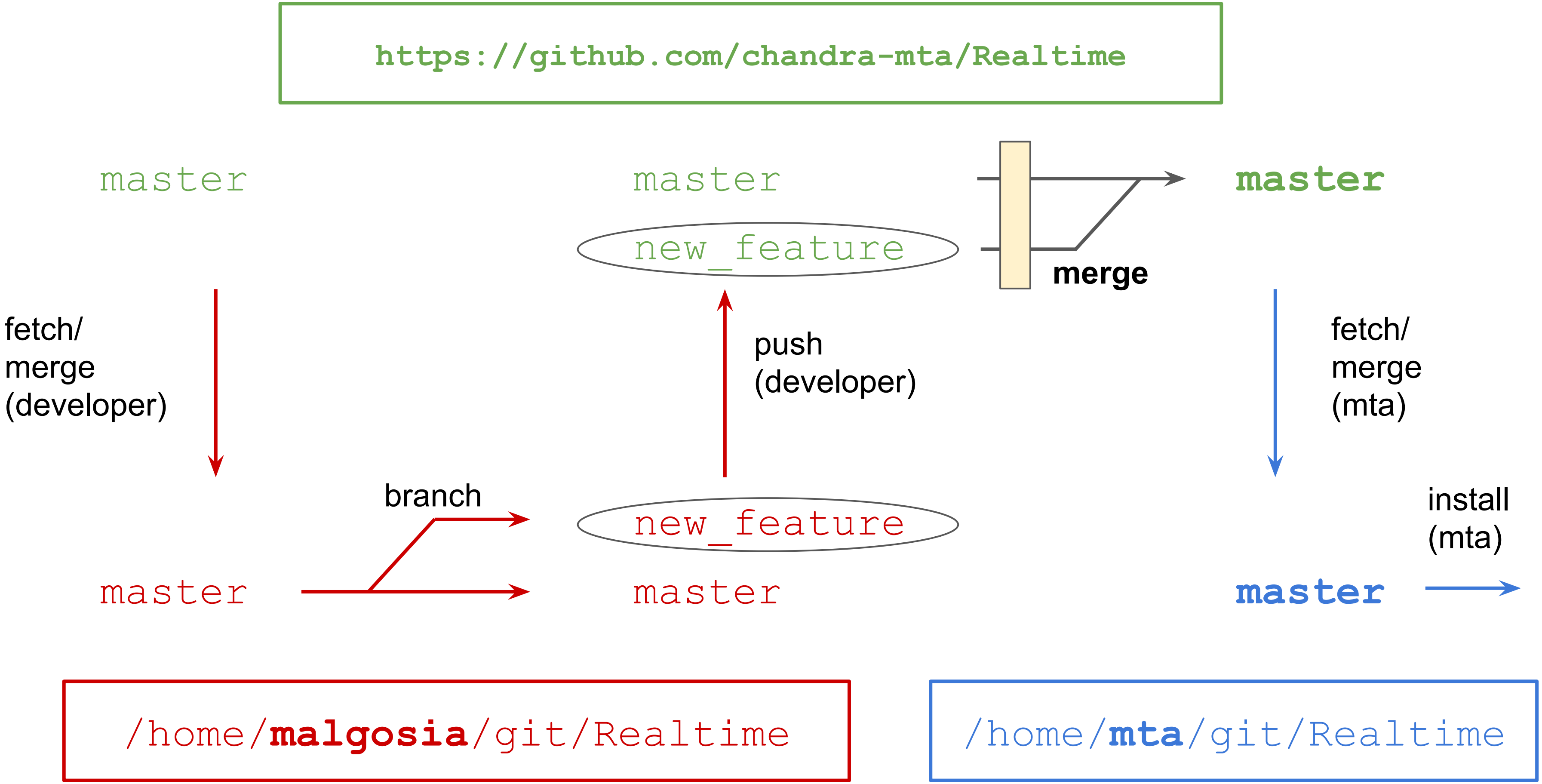
Synchronize local master with remote (github) master

Install the updated Realtime project

Test

Local work (user mta or cus)
`/home/mta/git/Realtime`

Note: mta fetches the new code directly from Github. This requires that mta/cus can access github.com/chandra-mta



https://github.com/chandra-mta/Realtime

master

master

new_feature

merge

master

fetch/
merge
(developer)

push
(developer)

fetch/
merge
(mta)

branch

new_feature

install
(mta)

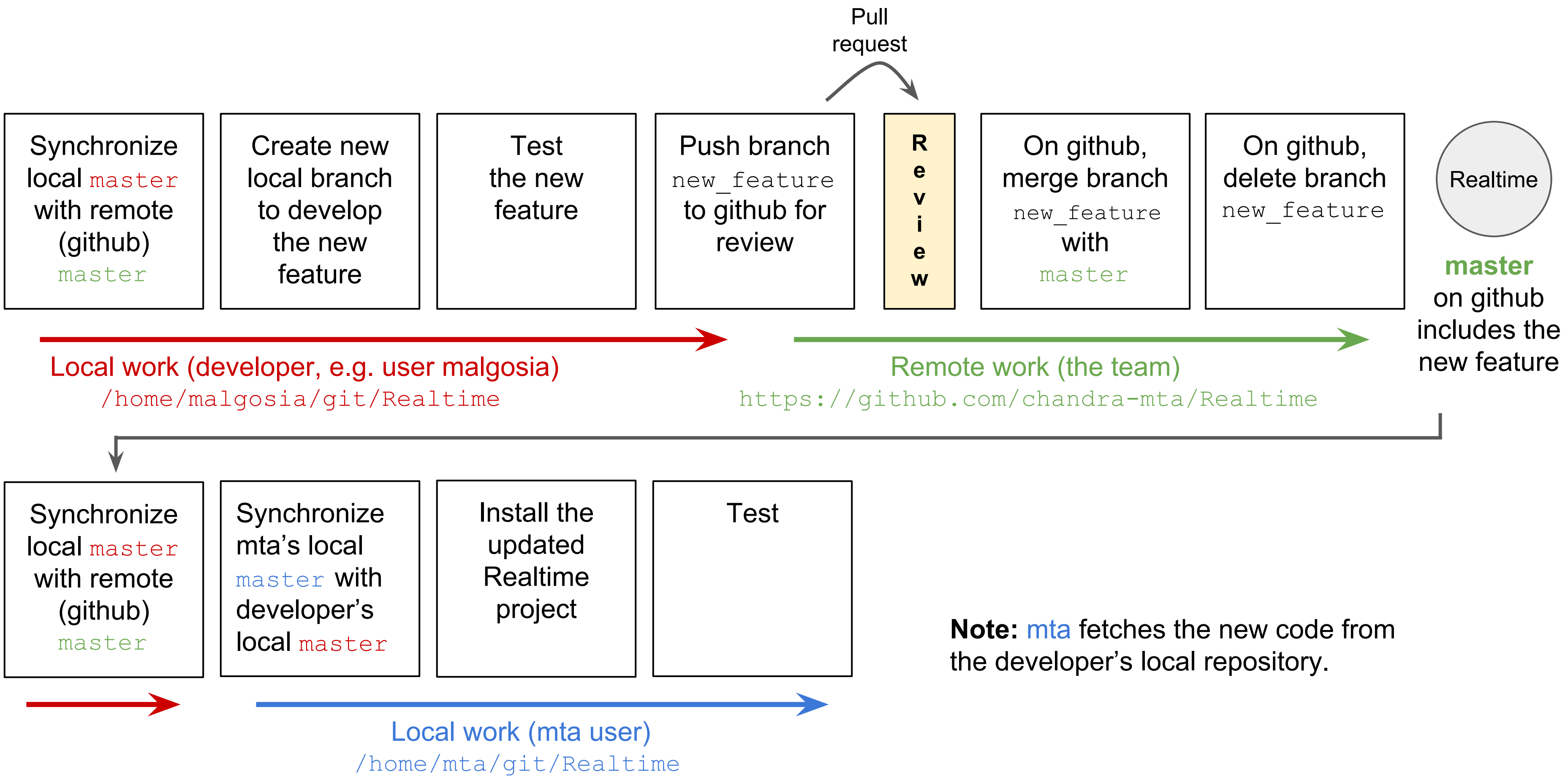
master

master

master

/home/**malgosia**/git/Realtime

/home/**mta**/git/Realtime



Pull request

Synchronize local master with remote (github) master

Create new local branch to develop the new feature

Test the new feature

Push branch new_feature to github for review

Review

On github, merge branch new_feature with master

On github, delete branch new_feature

Realtime

master on github includes the new feature

Local work (developer, e.g. user malgosia)
`/home/malgosia/git/Realtime`

Remote work (the team)
`https://github.com/chandra-mta/Realtime`

Synchronize local master with remote (github) master

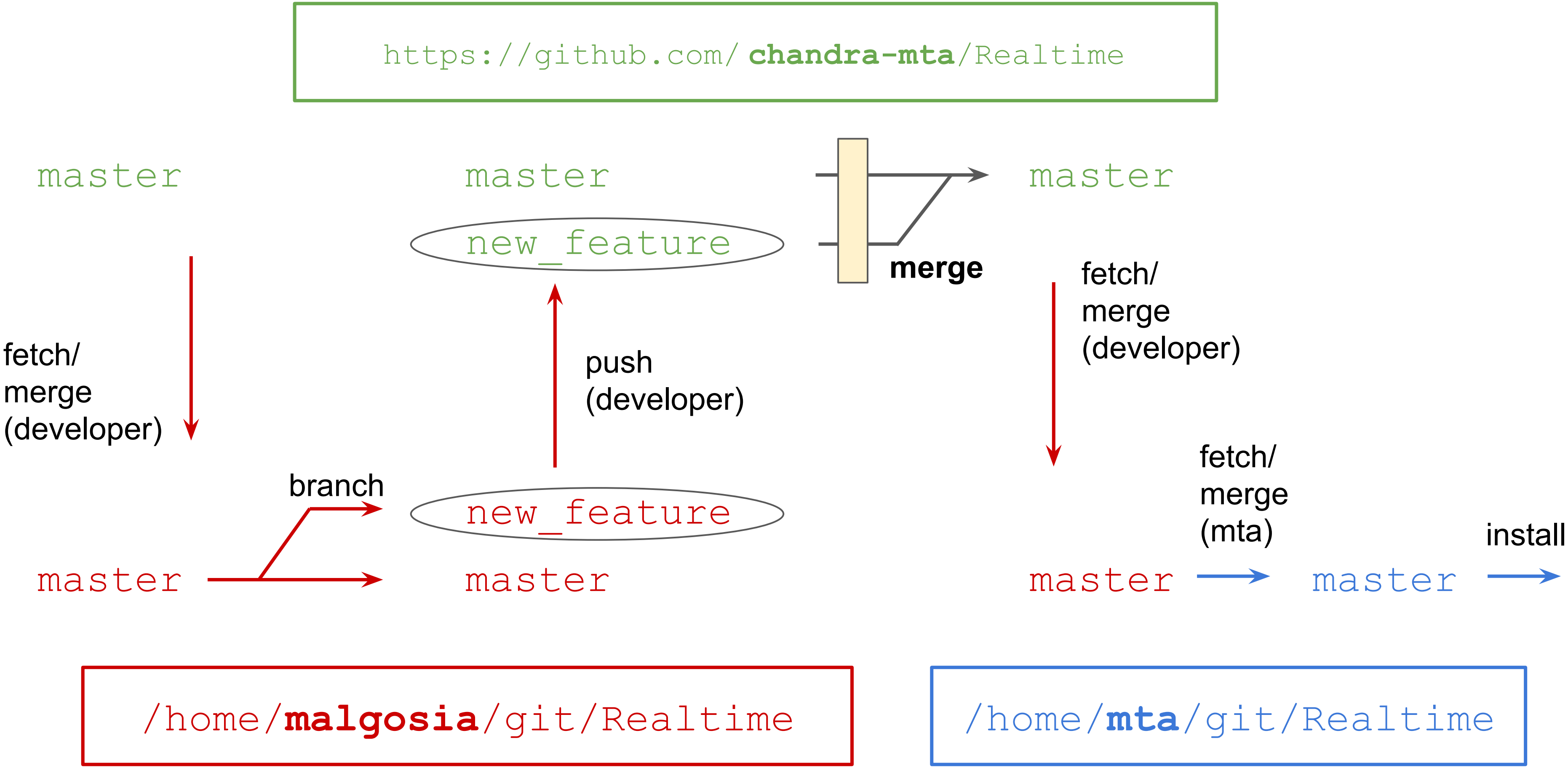
Synchronize mta's local master with developer's local master

Install the updated Realtime project

Test

Local work (mta user)
`/home/mta/git/Realtime`

Note: mta fetches the new code from the developer's local repository.



https://github.com/ **chandra-mta**/Realtime

master

master

new_feature

merge

master

fetch/
merge
(developer)

push
(developer)

fetch/
merge
(developer)

branch

new_feature

fetch/
merge
(mta)

install

master

master

master

master

/home/**malgosia**/git/Realtime

/home/**mta**/git/Realtime

How does it work in practise?

Example

Add new radiation alert to the `space_weather` set of scripts
that belong to the `/home/mta/git/Realtime` local repository

Github (remote) version `https://github.com/chandra-mta/Realtime`

Synchronize developer’s local master branch with the online code

```
malgosia> cd /home/malgosia/git/Realtime                  # go to the local Realtime repository
malgosia> git status
# On branch master
nothing to commit (working directory clean)
malgosia> git branch                                      # make sure you are on master branch
* master
malgosia> git remote -v                                   # check the setup of your paths
origin    git@github.com:chandra-mta/Realtime.git (fetch)
origin    git@github.com:chandra-mta/Realtime.git (push)
malgosia> git fetch origin                                 # fetch from github (to make sure
                                                           # you have the most recent version)
malgosia> git merge origin/master                          # merge
malgosia> ls
acorn/
config_mon/
firedrill/
snapshot/
space_weather/
…
malgosia> ls space_weather
space_weather/calculate.csh
space_weather/G13.calc
…
```

Create new local branch for the development of the new feature

```
malgosia> git branch new_feature                        # create a new branch, e.g. new_feature
malgosia> git branch
* master
  new_feature
malgosia> git checkout new_feature                      # switch to the branch new_feature
Switched to branch 'new_feature'
malgosia> git branch
  master
* new_feature

<local development, edit existing scripts, add new scripts, etc>
```

Push the branch containing the new development to Github

```
malgosia> git status
# On branch new_feature
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#    modified:  G13.calc
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    goes_hrc_proxy_viol.pl
no changes added to commit (use "git add" and/or "git commit -a")

malgosia> git add G13.calc goes_hrc_proxy_viol.pl             # add and commit the changes
malgosia> git commit -m “Added goes hrc proxy alert”
malgosia> git status
#  On branch new_feature
nothing to commit (working directory clean)
malgosia> git push origin new_feature                         # push the new_feature to github
```

<The workflow moves to Github. Github has now two branches: master and new_feature. The new development is visible to everyone for comparison with the prior version(s) and for review>

Synchronize mta’s local master branch with the updated Github master branch

<Pull request issued on github for the new branch. Code review on github, branch new_feature merged with master, github’s master code contains now the new radiation alert>

1. mta pulls the new code directly from Github

```
<user>> su mta                                              # any user logged in as mta

mta> cd /home/mta/git/Realtime                              # go to mta’s Realtime repo
mta> git status                                             # check the branch and status
# On branch master
nothing to commit (working directory clean)
mta> git remote -v                                          # check the paths
origin    git@github.com:chandra-mta/Realtime.git (fetch)
origin    git@github.com:chandra-mta/Realtime.git (push)
mta> git fetch origin                                       # fetch the new master from github
mta> git merge origin/master                                # merge
```

<Realtime code on mta’s local master branch contains now the new radiation alert and is ready for installation>

2. **or** The developer pulls the new code from Github, and mta pulls the new code from the developer - next page

<Pull request issued on github for the new branch. Code review on github, branch new_feature merged with master, remote master code contains now the new radiation alert>

```
malgosia> git remote -v                                    # confirm the paths
origin    git@github.com:chandra-mta/Realtime.git (fetch)
origin    git@github.com:chandra-mta/Realtime.git (push)
malgosia> git fetch origin                                 # fetch the new master from github
malgosia> git checkout master                              # checkout to branch master
malgosia> git merge origin/master                          # merge

malgosia> su mta                                           # log in as mta

mta> cd /home/mta/git/Realtime                             # go to mta’s Realtime repo
mta> git status                                            # check the branch and status
# On branch master                                         # checkout to master if not there
nothing to commit (working directory clean)
mta> git remote set-url origin /home/malgosia/git/Realtime         # update the paths if needed
mta> git remote -v                                         # double check
origin    /home/malgosia/git/Realtime (fetch)
origin    /home/malgosia/git/Realtime (push)
mta> git fetch origin                                      # fetch the new master from malgosia
mta> git merge origin/master                               # merge
```

<Realtime code on mta’s local master branch is ready for installation>