Release a new Clippy Version

NOTE: This document is probably only relevant to you, if you're a member of the Clippy team.

Clippy is released together with stable Rust releases. The dates for these releases can be found at the Rust Forge. This document explains the necessary steps to create a Clippy release.

1. Remerge the `beta` branch
2. Update the `beta` branch
3. Find the Clippy commit
4. Tag the stable commit
5. Update `CHANGELOG.md`

NOTE: This document is for stable Rust releases, not for point releases. For point releases, step 1. and 2. should be enough.

Remerge the `beta` branch

This step is only necessary, if since the last release something was backported to the beta Rust release. The remerge is then necessary, to make sure that the Clippy commit, that was used by the now stable Rust release, persists in the tree of the Clippy repository.

To find out if this step is necessary run

```
# Assumes that the local master branch is up-to-date
$ git fetch upstream
$ git branch master --contains upstream/beta
```

If this command outputs `master`, this step is **not** necessary.

```
# Assuming `HEAD` is the current `master` branch of rust-lang/rust-clippy
$ git checkout -b backport_remerge
$ git merge upstream/beta
$ git diff  # This diff has to be empty, otherwise something with the remerge failed
$ git push origin backport_remerge  # This can be pushed to your fork
```

After this, open a PR to the master branch. In this PR, the commit hash of the `HEAD` of the `beta` branch must exists. In addition to that, no files should be changed by this PR.

Update the `beta` branch

This step must be done **after** the PR of the previous step was merged.

First, the Clippy commit of the `beta` branch of the Rust repository has to be determined.

```
# Assuming the current directory corresponds to the Rust repository
$ git checkout beta
$ BETA_SHA=$(git log --oneline -- src/tools/clippy/ | grep -o "Merge commit '[a-f0-9]*' into .*" | head -1 | sed -e "s/Merge commit '\([a-f0-9]*\)' into .*/\1/g")
```

After finding the Clippy commit, the `beta` branch in the Clippy repository can be updated.

```
# Assuming the current directory corresponds to the Clippy repository
$ git checkout beta
$ git reset --hard $BETA_SHA
$ git push upstream beta
```

Find the Clippy commit

The first step is to tag the Clippy commit, that is included in the stable Rust release. This commit can be found in the Rust repository.

```
# Assuming the current directory corresponds to the Rust repository
$ git fetch upstream    # `upstream` is the `rust-lang/rust` remote
$ git checkout 1.XX.0   # XX should be exchanged with the corresponding version
$ SHA=$(git log --oneline -- src/tools/clippy/ | grep -o "Merge commit '[a-f0-9]*'
into .*" | head -1 | sed -e "s/Merge commit '\([a-f0-9]*\)' into .*/\1/g")
```

Tag the stable commit

After finding the Clippy commit, it can be tagged with the release number.

```
# Assuming the current directory corresponds to the Clippy repository
$ git checkout $SHA
$ git tag rust-1.XX.0               # XX should be exchanged with the corresponding
version
$ git push upstream rust-1.XX.0     # `upstream` is the `rust-lang/rust-clippy`
remote
```

After this, the release should be available on the Clippy release page.

Update the `stable` branch

At this step you should have already checked out the commit of the `rust-1.XX.0` tag. Updating the stable branch from here is as easy as:

```
# Assuming the current directory corresponds to the Clippy repository and the
# commit of the just created rust-1.XX.0 tag is checked out.
$ git push upstream rust-1.XX.0:stable  # `upstream` is the `rust-lang/rust-clippy`
remote
```

NOTE: Usually there are no stable backports for Clippy, so this update should be possible without force pushing or anything like this. If there should have happened a stable backport, make sure to re-merge those changes just as with the `beta` branch.

Update `CHANGELOG.md`

For this see the document on how to update the changelog.