Releasing collections with release branches

Collections MUST follow the semantic versioning rules. See :ref:`releasing_collections` for high-level details.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\community\collection_contributors\(ansible-devel)(docs)(docsite)(rst)(community)(collection_contributors)collection_release_with_branches.rst, line 6); *backlink***
>
> Unknown interpreted text role "ref".

Release planning and announcement

1. Announce your intention to release the collection in a corresponding pinned release issue/community pinboard of the collection and in the `#ansible-community` Matrix/IRC channel. Repeat the announcement in any other dedicated channels if they exist.
2. Ensure all the other repository maintainers are informed about the time of the following release.

Releasing major collection versions

The new version is assumed to be `X.0.0`.

1. If you are going to release the `community.general` and `community.network` collections, create new `backport-X` and `needs_backport_to_stable_X` labels in the corresponding repositories. Copy the styles and descriptions from the corresponding existing labels.
2. Ensure you are in a default branch in your local fork. These examples use `main`.

   ```
   git status
   git checkout main     # if needed
   ```

3. Update your local fork:

   ```
   git pull --rebase upstream main
   ```

Creating the release branch

1. Create a branch `stable-X`. Replace `X` with a correct number and push it to the **upstream** repository, NOT to the `origin`.:

   ```
   git branch stable-X main
   git push upstream stable-X
   ```

2. Create and checkout to another branch from the `main` branch:

   ```
   git checkout -b update_repo
   ```

3. Update the version in `galaxy.yml` in the branch to the next **expected** version, for example, `X.1.0`.

Creating the changelogs

1. Replace `changelogs/changelog.yml` with:

   ```
   ancestor: X.0.0
   releases: {}
   ```

2. Remove all changelog fragments from `changelogs/fragments/`. Removing the changelog fragments ensures that every

major release has a changelog describing changes since the last major release.
3. Add and commit all the changes made. Push the branch to the `origin` repository.
4. Create a pull request in the collection repository. If CI tests pass, merge the pull request since the `main` branch is expecting changes for the next minor/major versions
5. Switch to the `stable-X` branch.
6. In the `stable-X` branch, ensure that `galaxy.yml` contains the correct version number `X.0.0`. If not, update it.
7. In the `stable-X` branch, add a changelog fragment `changelogs/fragments/X.0.0.yml` with the content:

```
release_summary: |-
  Write some text here that should appear as the release summary for this version.
  The format is reStructuredText, but not a list as for regular changelog fragments.
  This text will be inserted into the changelog.
```

For example:

```
release_summary: This is release 2.0.0 of ``community.foo``, released on YYYY-MM-DD.
```

8. In the `stable-X` branch, generate the changelogs:

```
antsibull-changelog release --cummulative-release
```

9. In the `stable-X` branch, verify that the `CHANGELOG.rst` looks as expected.
10. In the `stable-X` branch, update `README.md` so that the changelog link points to `/tree/stable-X/` and no longer to `/tree/main/`, and change badges respectively, for example, in case of AZP, add `?branchName=stable-X` to the AZP CI badge (https://dev.azure.com/ansible/community.xxx/_apis/build/status/CI?branchName=stable-X).
11. In the `stable-X` branch, add, commit, and push changes to `README.md`, `CHANGELOG.rst` and `changelogs/changelog.yaml`, and potentially deleted/archived fragments to the **upstream** repository, NOT to the `origin`.

Publishing the collection

1. In the `stable-X` branch, add an annotated tag to the last commit with the collection version `X.0.0`. Pushing this tag to the `upstream` repository will make Zuul publish the collection on Ansible Galaxy.

```
git tag -n    # see current tags and their comments
git tag -a NEW_VERSION -m "comment here"    # the comment can be, for example, "community.foo: 2.0.0
git push upstream NEW_VERSION
```

2. Wait until the new version is published on the collection's Ansible Galaxy page. It will appear in a list of tarballs available to download.
3. Add a GitHub release for the new tag. The title should be the version and content, such as - `See https://github.com/ansible-collections/community.xxx/blob/stable-X/CHANGELOG.rst for all changes.`
4. Announce the release through the Bullhorn Newsletter.
5. Announce the release in the pinned release issue/community pinboard of the collection and in the `#ansible-community` Matrix/Libera.Chat IRC channel.
6. In the `stable-X` branch, update the version in `galaxy.yml` to the next **expected** version, for example, `X.1.0`. Add, commit and push to the **upstream** repository.

Releasing minor collection versions

The new version is assumed to be `X.Y.0`. All changes that should go into it are expected to be previously backported from the default branch to the `stable-X` branch.

Creating the changelogs

1. In the `stable-X` branch, make sure that `galaxy.yml` contains the correct version number `X.Y.0`. If not, update it.
2. In the `stable-X` branch, add a changelog fragment `changelogs/fragments/X.Y.0.yml` with content:

```
release summary: |-
  Write some text here that should appear as the release summary for this version.
  The format is reStructuredText, but not a list as for regular changelog fragments.
  This text will be inserted into the changelog.
```

3. In the `stable-X` branch, run:

```
antsibull-changelog release
```

4. In the `stable-X` branch, verify that `CHANGELOG.rst` looks as expected.
5. In the `stable-X` branch, add, commit, and push changes to `CHANGELOG.rst` and `changelogs/changelog.yaml`, and potentially deleted/archived fragments to the **upstream** repository, NOT to the origin.

Publishing the collection

1. In the `stable-X` branch, add an annotated tag to the last commit with the collection version `X.Y.0`. Pushing this tag to the `upstream` repository will make Zuul publish the collection on Ansible Galaxy.

   ```
   git tag -n    # see current tags and their comments
   git tag -a NEW_VERSION -m "comment here"    # the comment can be, for example, "community.foo: 2.1.0
   git push upstream NEW_VERSION
   ```

2. Wait until the new version is published on the collection's Ansible Galaxy page. The published version will appear in a list of tarballs available to download.
3. Add a GitHub release for the new tag. The title should be the version and content, such as - `See https://github.com/ansible-collections/community.xxx/blob/stable-X/CHANGELOG.rst for all changes.`
4. Announce the release through the Bullhorn Newsletter.
5. Announce the release in the pinned release issue/community pinboard of the collection and in the `#ansible-community` Matrix/IRC channel. Additionally, you can announce it using GitHub's Releases system.
6. In the `stable-X` branch, update the version in `galaxy.yml` to the next **expected** version, for example, if you have released `X.1.0`, the next expected version could be `X.2.0`. Add, commit and push to the **upstream** repository.
7. Checkout to the `main` branch.
8. In the `main` branch:

   1. If more minor versions are released before the next major version, update the version in `galaxy.yml` to `X.(Y+1).0` as well. Create a dedicated pull request and merge.
   2. If the next version will be a new major version, create a pull request where you update the version in `galaxy.yml` to `(X+1).0.0`. Note that the sanity tests will most likely fail since there will be deprecations with removal scheduled for `(X+1).0.0`, which are flagged by the tests.

   For every such deprecation, decide:

   - Whether to remove them now. For example you remove the complete `modules/plugins` or you remove redirects.
   - Whether to add ignore entries to the corresponding `tests/sanity/ignore-*.txt` file and create issues, for example for removed features in `modules/plugins`.

   Once the CI tests pass, merge the pull request. Make sure that this pull request is merged not too much later after the release for `verison_added` sanity tests not to expect the wrong version for the new feature pull request.

   > **Note**
   >
   > It makes sense to already do some removals in the days before the release. These removals must happen in the main branch and must not be backported.

Releasing patch versions

The new version is assumed to be `X.Y.Z`, and the previous patch version is assumed to be `X.Y.z` with `z < Z`. `z` is frequently``0`` snce patch releases are uncommon.

Releasing when more minor versions are expected

1. Checkout the `X.Y.z` tag.
2. Update `galaxy.yml` so that the version is `X.Y.Z`. Add and commit.
3. Cherry-pick all changes from `stable-X` that were added after `X.Y.z` and should go into `X.Y.Z`.
4. Add a changelog fragment `changelogs/fragments/X.Y.Z.yml` with content:

   ```
   release_summary: |-
     Write some text here that should appear as the release summary for this version.
     The format is reStructuredText but not a list as for regular changelog fragments.
     This text will be inserted into the changelog.
   ```

   Add to git and commit.

5. Generate the changelogs.

   ```
   antsibull-changelog release
   ```

6. Verify that `CHANGELOG.rst` looks as expected.
7. Add and commit changes to `CHANGELOG.rst` and `changelogs/changelog.yaml`, and potentially deleted/archived fragments.

Publishing the collection

1. Add an annotated tag to the last commit with the collection version `X.Y.Z`. Pushing this tag to the `upstream` repository will make Zuul publish the collection on Ansible Galaxy.

```
git tag -n    # see current tags and their comments
git tag -a NEW_VERSION -m "comment here"    # the comment can be, for example, "community.foo: 2.1.1
git push upstream NEW_VERSION
```

2. Wait until the new version is published on the collection's Ansible Galaxy page. It will appear in a list of tarballs available to download.
3. Add a GitHub release for the new tag. The title should be the version and content, such as - `See https://github.com/ansible-collections/community.xxx/blob/stable-X/CHANGELOG.rst for all changes.`

> **Note**
>
> The data for this release is only contained in a tag, and not in a branch, in particular not in `stable-X`. This is deliberate, since the next minor release `X.(Y+1).0` already contains the changes for `X.Y.Z` as well, since these were cherry-picked from `stable-X`.

4. Announce the release through the Bullhorn Newsletter.
5. Announce the release in the pinned release issue/community pinboard of the collection and in the `#ansible-community` *Matrix/IRC channel <https://docs.ansible.com/ansible/devel/community/communication.html#real-time-chat>*.

Releasing when no more minor versions are expected

1. In the `stable-X` branch, make sure that `galaxy.yml` contains the correct version number `X.Y.Z`. If not, update it!
2. In the `stable-X` branch, add a changelog fragment `changelogs/fragments/X.Y.Z.yml` with content:

```
release_summary: |-
  Write some text here that should appear as the release summary for this version.
  The format is reStructuredText, but not a list as for regular changelog fragments.
  This text will be inserted into the changelog.
```

3. Generate the changelogs in the `stable-X` branch.

```
antsibull-changelog release
```

4. In the `stable-X` branch, verify that `CHANGELOG.rst` looks as expected.
5. In the `stable-X` branch, add, commit, and push changes to `CHANGELOG.rst` and `changelogs/changelog.yaml`, and potentially deleted/archived fragments to the **upstream** repository, NOT to the origin.

Publishing the collection

1. In the `stable-X` branch, add an annotated tag to the last commit with the collection version `X.Y.Z`. Pushing this tag to the `upstream` repository will make Zuul publish the collection on Ansible Galaxy.

```
git tag -n    # see current tags and their comments
git tag -a NEW_VERSION -m "comment here"    # the comment can be, for example, "community.foo: 2.1.1
git push upstream NEW_VERSION
```

2. Wait until the new version is published on the collection's Ansible Galaxy page. It will appear in a list of tarballs available to download.
3. Add a GitHub release for the new tag. Title should be the version and content, such as: `See https://github.com/ansible-collections/community.xxx/blob/stable-X/CHANGELOG.rst for all changes.`
4. Announce the release through the Bullhorn Newsletter.
5. Announce the release in the pinned issue/community pinboard of the collection and in the `#ansible-community` Matrix/IRC channel.