How to get the program to analyse all the previous commits and increase the version number in a cumulative way #816

Closed **darkpandarts** opened this issue 2 weeks ago · 9 comments

Labels question

darkpandarts commented 2 weeks ago

I'm interested in using the "release flow" branching strategy which relies on (briefly)

1. having a single `main` branch
2. `feature` branches are cut from main and then merged back in
3. at the end of a sprint, a `release` branch is cut from `main`

At that point I expected python-semantic-release to analyse the conventional commits to cumulatively increase the version number. For example

1. My current version number is `0.5.0` in terms of last tag in main and what is in `__version__.py` .
2. I have made the following commits since `0.5.0`

```
* 4cd3a20 fix: version should be 0.8.1
* 539864a feat: version should be 0.8.0
* f064516 feat: version should be 0.7.0
* 5e5cd40 feat: version should be 0.6.0
* d95d656 (tag: 0.5.0, origin/master, origin/HEAD, master) 0.5.0
```

However, when I run `poetry run semantic-release --noop -vv --config pyproject.toml version --print` in the release branch, the version comes out as **`0.6.0`** . See debug log below

I expected this to be 0.8.1 as I thought the commit changes were cumulative. Is there a way to enable this or is this just not the way it is done?

Many thanks! :)

```
[20:15:13] DEBUG    [semantic_release.cli.commands.main] DEBUG main.main: logging le
set to: DEBUG
main.py:105
🛡 You are running in no-operation mode, because the '--noop' flag was supplied
```

```
          DEBUG    [semantic_release.cli.commands.main] DEBUG main.main: global cli
options: GlobalCommandLineOptions(noop=True, verbosity=2, config_file='pyproject.toml',
strict=False)                                   main.py:123
          INFO     [semantic_release.cli.util] INFO util.load_raw_config_file: Loading
configuration from pyproject.toml
util.py:78
          DEBUG    [semantic_release.cli.util] DEBUG util.load_raw_config_file: Trying
to parse configuration pyproject.toml in TOML format
util.py:81
          INFO     [semantic_release.cli.config] INFO config.select_branch_options:
Using group 'master' options, as 'master|feat/*' matches 'feat/another-something'
config.py:238
          DEBUG    [semantic_release.cli.config] DEBUG config.from_raw_config: hvcs
token is not set
config.py:363
          DEBUG    [semantic_release.changelog.template] DEBUG template.environment:
{'template_dir': 'templates', 'block_start_string': '{%', 'block_end_string': '%}',
'variable_start_string': '{{',                  template.py:55
                   'variable_end_string': '}}', 'comment_start_string': '{#',
'comment_end_string': '#}', 'line_statement_prefix': None, 'line_comment_prefix': None,
'trim_blocks': False, 'lstrip_blocks': False,
                   'newline_sequence': '\n', 'keep_trailing_newline': False,
'extensions': (), 'autoescape': True, 'autoescape_value': True}
          DEBUG    [semantic_release.version.translator] DEBUG
translator._invert_tag_format_to_re: inverted tag_format '{version}' to '(?
P<version>.*)'
translator.py:40
          DEBUG    [semantic_release.cli.commands.version] DEBUG
version.is_forced_prerelease: force_prerelease = False, force_level = None, prerelease
= False                                                         version.py:47
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.1.0' as Version
version.py:121
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.1.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.1.0
version.py:148
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.2.0' as Version
version.py:121
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.2.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.2.0
version.py:148
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.3.0' as Version
version.py:121
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.3.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.3.0
version.py:148
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.4.0' as Version
version.py:121
```

```
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.4.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.4.0
version.py:148
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.5.0' as Version
version.py:121
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.5.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.5.0
version.py:148
          INFO     [semantic_release.version.algorithm] INFO
algorithm.tags_and_versions: found 5 previous tags
algorithm.py:60
          INFO     [semantic_release.version.algorithm] INFO algorithm.next_version:
Found 5 full releases (excluding prereleases)
algorithm.py:255
          DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.0.0' as Version
version.py:121
          DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.0.0 parsed as a non-prerelease
version.py:145
          DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.0.0
version.py:148
          INFO     [semantic_release.version.algorithm] INFO algorithm.next_version:
The last full release was 0.5.0, tagged as '0.5.0'
algorithm.py:277
          DEBUG    [git.cmd] DEBUG cmd.execute: Popen(['git', 'merge-base', '0.5.0',
'feat/another-something'],
cmd.py:1057

cwd=/media/bsgt/jogi1/XX_local_PSYNC_linux2/XXX_CONTRACTING/stablecaps/0000_STABLECAPS_GI
stdin=None, shell=False, universal_newlines=False)
          DEBUG    [git.cmd] DEBUG cmd.execute: Popen(['git', 'cat-file', '--batch-
check'],
cwd=/media/bsgt/jogi1/XX_local_PSYNC_linux2/XXX_CONTRACTING/stablecaps/0000_STABLECAPS_GI
cmd.py:1057
                    stdin=<valid stream>, shell=False, universal_newlines=False)
          DEBUG    [git.cmd] DEBUG cmd.execute: Popen(['git', 'cat-file', '--batch'],
cwd=/media/bsgt/jogi1/XX_local_PSYNC_linux2/XXX_CONTRACTING/stablecaps/0000_STABLECAPS_GI
cmd.py:1057
                    stdin=<valid stream>, shell=False, universal_newlines=False)
          DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.bfs: checking
if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
d95d656314e0c70132cd8cbf6de1625630c9d3bc                algorithm.py:89
          INFO     [semantic_release.version.algorithm] INFO algorithm.bfs: found
latest version in branch history: '0.5.0' (d95d656)
algorithm.py:96
          INFO     [semantic_release.version.algorithm] INFO
algorithm._bfs_for_latest_version_in_history: the latest version in this branch's
history is 0.5.0                                                    algorithm.py:115
          INFO     [semantic_release.version.algorithm] INFO algorithm.next_version:
The last full version in this branch's history was 0.5.0
algorithm.py:302
```

```
           DEBUG    [git.cmd] DEBUG cmd.execute: Popen(['git', 'rev-list', '0.5.0...',
'--'],
cwd=/media/bsgt/jogi1/XX_local_PSYNC_linux2/XXX_CONTRACTING/stablecaps/0000_STABLECAPS_GI
cmd.py:1057
                     stdin=None, shell=False, universal_newlines=False)
           DEBUG    [semantic_release.commit_parser.angular] DEBUG angular.parse:
commit b0232162b07f53eea12cdf5e92c17493a462320d introduces a minor level_bump
angular.py:125
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
adding minor to the levels identified in commits_since_last_full_release
algorithm.py:328
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
b0232162b07f53eea12cdf5e92c17493a462320d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.4.0' (7c884c266b67e405d6b97c1d3784819be8867b9d) matches commit
b0232162b07f53eea12cdf5e92c17493a462320d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.3.0' (c80a7aec7472d11a2d5f51552f3ce3760dc45750) matches commit
b0232162b07f53eea12cdf5e92c17493a462320d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.2.0' (3c88647f5746212faf5abf086c0f4ca69b14f691) matches commit
b0232162b07f53eea12cdf5e92c17493a462320d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.1.0' (dcae5b5b75d98a6dd929b5137d488c0e114e2305) matches commit
b0232162b07f53eea12cdf5e92c17493a462320d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
no tags correspond to commit b0232162b07f53eea12cdf5e92c17493a462320d
algorithm.py:369
           DEBUG    [semantic_release.commit_parser.angular] DEBUG angular.parse:
commit 4cd3a206c43c3da3e5c323bad9059644b0242f46 introduces a patch level_bump
angular.py:125
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
adding patch to the levels identified in commits_since_last_full_release
algorithm.py:328
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
4cd3a206c43c3da3e5c323bad9059644b0242f46       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.4.0' (7c884c266b67e405d6b97c1d3784819be8867b9d) matches commit
4cd3a206c43c3da3e5c323bad9059644b0242f46       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.3.0' (c80a7aec7472d11a2d5f51552f3ce3760dc45750) matches commit
4cd3a206c43c3da3e5c323bad9059644b0242f46       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.2.0' (3c88647f5746212faf5abf086c0f4ca69b14f691) matches commit
4cd3a206c43c3da3e5c323bad9059644b0242f46       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.1.0' (dcae5b5b75d98a6dd929b5137d488c0e114e2305) matches commit
4cd3a206c43c3da3e5c323bad9059644b0242f46       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
no tags correspond to commit 4cd3a206c43c3da3e5c323bad9059644b0242f46
algorithm.py:369
           DEBUG    [semantic_release.commit_parser.angular] DEBUG angular.parse:
commit 539864aaf19d59e07ef35d33fc0893a227ada14d introduces a minor level_bump
angular.py:125
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
adding minor to the levels identified in commits_since_last_full_release
algorithm.py:328
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
```

```
testing if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
539864aaf19d59e07ef35d33fc0893a227ada14d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.4.0' (7c884c266b67e405d6b97c1d3784819be8867b9d) matches commit
539864aaf19d59e07ef35d33fc0893a227ada14d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.3.0' (c80a7aec7472d11a2d5f51552f3ce3760dc45750) matches commit
539864aaf19d59e07ef35d33fc0893a227ada14d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.2.0' (3c88647f5746212faf5abf086c0f4ca69b14f691) matches commit
539864aaf19d59e07ef35d33fc0893a227ada14d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.1.0' (dcae5b5b75d98a6dd929b5137d488c0e114e2305) matches commit
539864aaf19d59e07ef35d33fc0893a227ada14d       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
no tags correspond to commit 539864aaf19d59e07ef35d33fc0893a227ada14d
algorithm.py:369
           DEBUG    [semantic_release.commit_parser.angular] DEBUG angular.parse:
commit f06451618e6ae44dec9e3d42f634ca4fd19b0212 introduces a minor level_bump
angular.py:125
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
adding minor to the levels identified in commits_since_last_full_release
algorithm.py:328
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
f06451618e6ae44dec9e3d42f634ca4fd19b0212       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.4.0' (7c884c266b67e405d6b97c1d3784819be8867b9d) matches commit
f06451618e6ae44dec9e3d42f634ca4fd19b0212       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.3.0' (c80a7aec7472d11a2d5f51552f3ce3760dc45750) matches commit
f06451618e6ae44dec9e3d42f634ca4fd19b0212       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.2.0' (3c88647f5746212faf5abf086c0f4ca69b14f691) matches commit
f06451618e6ae44dec9e3d42f634ca4fd19b0212       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.1.0' (dcae5b5b75d98a6dd929b5137d488c0e114e2305) matches commit
f06451618e6ae44dec9e3d42f634ca4fd19b0212       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
no tags correspond to commit f06451618e6ae44dec9e3d42f634ca4fd19b0212
algorithm.py:369
           DEBUG    [semantic_release.commit_parser.angular] DEBUG angular.parse:
commit 5e5cd40c8ac935d3789952c45e00056b497a1068 introduces a minor level_bump
angular.py:125
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
adding minor to the levels identified in commits_since_last_full_release
algorithm.py:328
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.5.0' (d95d656314e0c70132cd8cbf6de1625630c9d3bc) matches commit
5e5cd40c8ac935d3789952c45e00056b497a1068       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.4.0' (7c884c266b67e405d6b97c1d3784819be8867b9d) matches commit
5e5cd40c8ac935d3789952c45e00056b497a1068       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.3.0' (c80a7aec7472d11a2d5f51552f3ce3760dc45750) matches commit
5e5cd40c8ac935d3789952c45e00056b497a1068       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
testing if tag '0.2.0' (3c88647f5746212faf5abf086c0f4ca69b14f691) matches commit
5e5cd40c8ac935d3789952c45e00056b497a1068       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
```

```
testing if tag '0.1.0' (dcae5b5b75d98a6dd929b5137d488c0e114e2305) matches commit
5e5cd40c8ac935d3789952c45e00056b497a1068       algorithm.py:350
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
no tags correspond to commit 5e5cd40c8ac935d3789952c45e00056b497a1068
algorithm.py:369
           DEBUG    [semantic_release.version.algorithm] DEBUG algorithm.next_version:
parsed the following distinct levels from the commits since the last release:
{<LevelBump.PATCH: 2>, <LevelBump.MINOR: 3>}      algorithm.py:374
           INFO     [semantic_release.version.algorithm] INFO algorithm.next_version:
The type of the next release release is: minor
algorithm.py:380
           DEBUG    [semantic_release.version.algorithm] DEBUG
algorithm._increment_version: _increment_version: latest_version = 0.5.0,
latest_full_version = 0.5.0, latest_full_version_in_history = 0.5.0,
algorithm.py:142
                    level_bump = minor, prerelease = False, prerelease_token = rc,
major_on_zero = False
           DEBUG    [semantic_release.version.algorithm] DEBUG
algorithm._increment_version: reducing version increment due to 0. version and
major_on_zero=False
algorithm.py:150
           DEBUG    [semantic_release.version.algorithm] DEBUG
algorithm._increment_version: prerelease=false and 0.5.0 is not a prerelease; bumping
with a minor release                                              algorithm.py:231
           DEBUG    [semantic_release.version.version] DEBUG version.bump: performing a
minor level bump
version.py:223
0.6.0
           DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.1.0' as Version
version.py:121
           DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.1.0 parsed as a non-prerelease
version.py:145
           DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.1.0
version.py:148
           DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.2.0' as Version
version.py:121
           DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.2.0 parsed as a non-prerelease
version.py:145
           DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.2.0
version.py:148
           DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.3.0' as Version
version.py:121
           DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.3.0 parsed as a non-prerelease
version.py:145
           DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.3.0
version.py:148
           DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.4.0' as Version
version.py:121
           DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.4.0 parsed as a non-prerelease
```

```
version.py:145
           DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.4.0
version.py:148
           DEBUG    [semantic_release.version.version] DEBUG version.parse: attempting
to parse string '0.5.0' as Version
version.py:121
           DEBUG    [semantic_release.version.version] DEBUG version.parse: version
string 0.5.0 parsed as a non-prerelease
version.py:145
           DEBUG    [semantic_release.version.version] DEBUG version.parse: parsed
build metadata '' from version string 0.5.0
version.py:148
           INFO     [semantic_release.version.algorithm] INFO
algorithm.tags_and_versions: found 5 previous tags
algorithm.py:60
           INFO     [semantic_release.cli.github_actions_output] INFO
github_actions_output.write_if_possible: not writing GitHub Actions output, as no file
specified                                        github_actions_output.py:70
```

 **darkpandarts** added the question label 2 weeks ago

codejedi365 commented 2 weeks ago • edited Contributor

> 1. having a single `main` branch
>
> 2. `feature` branches are cut from main and then merged back in

This is basically what is called GitHub Flow (which is separate from Git Flow), except for 3.

> 3. at the end of a sprint, a `release` branch is cut from `main`

In most cases, I would have to caution against doing a release branch and make only a release tag on main.

> I expected this to be 0.8.1 as I thought the commit changes were cumulative. Is there a way to enable this or is this just not the way it is done?

The commit changes are consolidated based on ranking between tags (not cumulative). The release tags themselves are evaluated cumulatively. I must caution against a cumulative result, as it is confusing to your users since you never actually gave them a `0.7` or `0.6`.

What many people do in your situation is release when your CI passes on the merge to main. If you don't want it automatic, (release at the end of a sprint) than run the CI with a manual job, a scheduled job, or manually run it locally.

If you did manual right now with the commits you have, 0.6.0 is the correct result based on the conventional commit standard. A single version bump can include a myriad of features and fixes.

darkpandarts commented 2 weeks ago Author

Thanks for the info! :)

> I must caution against a cumulative result, as it is confusing to your users since you never actually gave them a 0.7 or 0.6.

I see what you mean about the potential confusion. I would have thought that would be more with regards to the major version than the minor and patch versions. I suppose it is a question of style at the end of the day. It would be nice to have the ability to cumulatively update minor and patch version numbers, but I get it if that is something that would not be desirable to support.

> In most cases, I would have to caution against doing a release branch and make only a release tag on main.

May I ask if you could extrapolate on this please? Just asking because "Release Flow" seems to be a established branching strategy. So, should that not also be catered to, to some extent?

Some links:

1. Releaseflow
2. Release Flow: How We Do Branching on the VSTS Team

Thanks again for your detailed reply! :)

codejedi365 commented last week Contributor

> I see what you mean about the potential confusion. I would have thought that would be more with regards to the major version than the minor and patch versions. I suppose it is a question of style at the end of the day. It would be nice to have the ability to cumulatively update minor and patch version numbers, but I get it if that is something that would not be desirable to support.

Yes it is stylistic in nature. I always take the view of your consumers into account. This is where the version number and changelog matter. The info helps them choose what do they need to consider when adding the new software. I would suspect your changelog would be numerous versions just for one change a piece. This is definitely non-standard and people would be confused by it. You can do it but Python-Semantic Release is incompatible with this construct even if you used a custom commit parser. You will have to construct your own tooling.

> > In most cases, I would have to caution against doing a release branch and make only a release tag on main.

> May I ask if you could extrapolate on this please? Just asking because "Release Flow" seems to be a established branching strategy. So, should that not also be catered to, to some extent?

I'm glad that you have done your research. Honestly, it's the first time I'm hearing of an actual name to this strategy. I am however aware of branching based on a release and have had personal experience here.

I read the links and I'm still a bit skeptical of some aspects of ReleaseFlow. From what I can tell, Python Semantic Release does support this strategy but we currently do not have a test suite to evaluate this workflow. I did not mean to insinuate that it's not supported.

I was cautioning against creating arbitrary branches because as I placed the concept in a category when you are supporting multiple long term versions. I consider this an advanced strategy since most smaller scale projects only tend to support the latest version. In one of my experiences, I found that we arbitrarily created a release branch for `1.0`, although a month later we didn't have a `2.0`, which meant main was basically the next version, so why have a 1.0 branch.

It is also out of place to see Dave Farley quoted in these articles because currently his YouTube channel says "Don't Branch", without a consideration for ReleaseFlow.

darkpandarts commented last week Author

I think you have pretty much convinced me that the cumulative version number would be more trouble than it's worth.

I'm not sure I fully understood what you meant about the branching (mostly due to my ignorance). I think you possibly mean that it would be better to have a trunk based strategy with commits in the mainline being marked with a release tag? If so, wouldn't the release branch just be roughly the same in purpose? i.e. a series of committed code, the last commit being tagged as the the released state. So just another way of saving the released state. How would it make a difference to what code was supported? Wouldn't one be in the same situation (with regards to support) whatever release strategy was used?

If you have any links explaining what you meant, that would be much appreciated! :)

codejedi365 commented last week • edited Contributor

> I'm not sure I fully understood what you meant about the branching...

I think some of the confusion is from the specific `git` -relevant definitions I'm using. Git uses hashes to distinguish differences between states of a repository. A hash is therefore the core reference point to a change. A commit, is the state of a repo with a subject/summary to describe it. Everything in Git is centered around these base references. A branch is defined by its head reference but also represents a list of sequential commits together from a `merge-base` of another branch. A branch implies code can be added to this and the branch can move/grow. A tag, however, is a pointer essentially to a commit location, it cannot have changes added to it. In order for a tag to move, it must be deleted and applied again to a different commit. It's only a human readable pointer to a reference.

> I think you possibly mean that it would be better to have a trunk based strategy with commits in the mainline being marked with a release tag?

Regardless of which branching strategy used, I recommended that you should use release tags to point to a specific state of code which represents a specific version, not a branch. A branch is used to make an alternative line of history of commits. Tags are for pointing at specific permanent states of code. Continuous Delivery advocates, like Dave Farley, state that branching hinders integration as it is more complex to simultaneously maintain multiple branches of code. If you create a release branch, essentially, you are creating a fork to deviate from the mainline. Since ReleaseFlow uses branches after code has been integrated its not as cumbersome as feature branching or git flow can be, however, I foresee way too many branches for very little gain.

> If so, wouldn't the release branch just be roughly the same in purpose? i.e. a series of committed code, the last commit being tagged as the the released state. So just another way of saving the released state.

This is why I provided my definitions above. Yes, the head of the branch marks a reference with its HEAD but it's not the purpose of a branch. It also is way too easy to move the HEAD and versions should be as permanent as possible. PSR's logic is centered around tags so without a tag, PSR is not going to be of much use to you.

> How would it make a difference to what code was supported? Wouldn't one be in the same situation (with regards to support) whatever release strategy was used?

(pending)

darkpandarts commented last week — Author

Ok gotcha! That does make sense. I was going to use tags with Release flow, so think that PSR would work. However, based on your comments, I'm thinking that maybe a trunk based branching strategy may be better. However, I'm a bit undecided as I tend to favour long lived branches. May I ask what is the specific branching style you use?

codejedi365 commented 4 days ago • edited — Contributor

> May I ask what is the specific branching style you use?

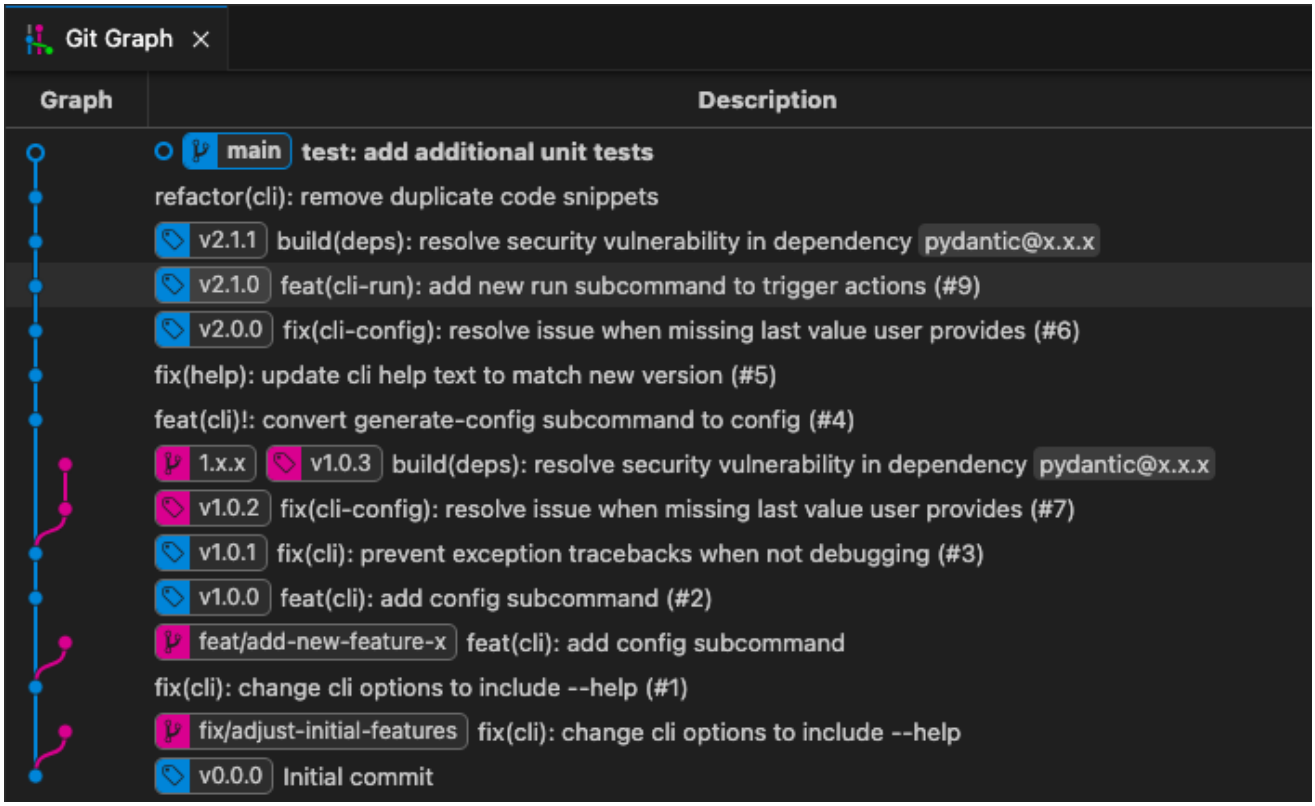
I personally use a GitHub Flow feature-branching strategy with squash merges. I do **NOT** implement the push to production or staging from an PR as the VSTS Team lead claimed was a particular nuance to GitHub Flow. I have built full CD pipelines upon merges to main, scheduled release pipelines, and manually triggered release pipelines for different projects.

I suggest to everyone to consider your answers to the following questions to help guide their selection:

1. What is your risk tolerance of feature regressions that your user will experience?
   a. Do you value peer code review or are you the sole developer?
   b. How do you build common knowledge across the team (sustainment)?
   c. How thorough are you with integration/system testing?
2. How do you expect (tools, workflow, etc) to automatically deploy your product?
3. How many individual versions are you going to support?

Based on the answers to these questions will help give you a lens to look at each strategy for and how it might effect the workflow. I am less inclined for the trunk based development approach personally as it is less compatible with the tools I'm familiar with and my risk tolerance of my users receiving possible regressions. I also think it is extremely annoying to have to deal with the branch I'm working on moving while I'm writing it. Depending how good you are with `git` it can change things. When I pair program though, I consider code reviews less important (its happening as the group codes) and can see the appeal of trunk based development as that point the PRs are kinda pointless. If I made that consideration, then I would also adjust my deployment strategy, and use scheduled deployments that match the end of the sprint. I would not run automatic CD from a trunk based strategy though (unless your risk tolerance is high), especially since developers generally are awful testers. I'm sure there are other considerations that I'm not including right now but these are a few.

I recommend studying what trunk-based development was trying to solve as it is a solution to certain problems but can introduce its own problems if you (and the team) don't collectively understand the approach and underlying considerations to make when introducing code changes. I actually pull a lot of insight from trunk based approaches and personally apply it create a more rapid GitHub Flow.



I built this example workflow to illustrate an example workflow I would do to include supporting of 2 concurrent major versions (ie, v2.0.0 latest, & v1.0.0 maintenance). I built it using my version of GitHub Flow but after the squash merges it basically ends up looking like trunk-based development. If I had concurrent development efforts of multiple teammates, there would be more branches waiting for completion and merge into main. This should not be a slow process, for merge into main. IMO, feature branches should focus on one aspect of the code and should not live very long to prevent complicated rebases and merge conflicts. You will see that I branch from the v1 tag and then make the change. Since the maintenance patch has to be manual, the creation of the branch is manual and then I can call PSR on that branch. It's a personal thing not to keep branches around as they can always be re-created; I would next delete `v1.x.x` as it doesn't serve any purpose. If you use a Visual Git program such as the `Git Graph` VSCode extension, branches create clutter so I remove them as much as possible. I left some of the branches at the bottom to illustrate the squash merges but ultimately these would be pruned after merge.

❤️ 1

darkpandarts commented 3 days ago — Author

Wow! Thanks for the detailed reply! I think that is really helpful! I really appreciate the time you put into this answer because it is very helpful! :)

👍 1

darkpandarts closed this as completed 3 days ago

codejedi365 commented 3 days ago Contributor

@darkpandarts, you're welcome! Thanks for pushing me to consolidate all my thoughts into words! Cheers!

Assignees

No one assigned

Labels

question

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants