

UEDGE GitHub Workflow Guide

Parallel Major Versions (v8 / v9) - GitFlow-ish, PEP 621, Rebase-First

This document is the **authoritative workflow specification** for all development, maintenance, and release activity in **UEDGE**.

It is written for:

- New contributors unfamiliar with GitHub
- Experienced developers
- Release managers and maintainers

1. Supported Major Versions

v8 - Current Baseline

- Default development version
- Receives **all active physics, bugfix, and feature work**
- Branches:
 - `main-v8` (released code)
 - `develop-v8` (integration)

v9 - Next-Generation

- Major restructuring and reformatting
- Branches:
 - `main-v9`
 - `develop-v9`
- Receives **explicit backports** from v8 where applicable

2. Core Rules

1. `main-v8` and `main-v9` are always releasable

2. All work happens on **feature branches**
3. Feature branches target **exactly one major version**
4. Rebase only short-lived branches
5. Backports are labeled, tracked, and cherry-picked
6. Inactive branches are archived, then deleted

3. Branch Model

Long-lived branches

Branch	Purpose
main-v8	Released v8 code
develop-v8	Active v8 development
main-v9	Released v9 code
develop-v9	Active v9 development

Short-lived branches

Naming is **mandatory**:

- `feature/v8-*`, `bugfix/v8-*`
- `feature/v9-*`, `bugfix/v9-*`
- `release/v8-x.y.z`, `release/v9-x.y.z`
- `hotfix/v8-x.y.z`, `hotfix/v9-x.y.z`

Feature branches:

- Are based on the matching `develop-*`
- Are the **only branches that accept PRs**
- Must be rebased before merge

4. Pull Requests

- PRs target **feature branches only**
- Feature branches are merged into `develop-*`
- No direct PRs to `main-*` or `develop-*`
- Maintainers perform integration

5. Versioning (PEP 621 / PEP 440)

Defined in `pyproject.toml`:

```
[project]
version = "1.8.2"
```

Examples:

- Final: `1.8.2`
- RC: `1.9.0rc1`
- Hotfix: `1.8.3`

Git tags:

- `v1.8.2`
- `v1.9.0rc1`

6. Backport Policy (v8 → v9)

Required Labels

Exactly **one** required on every v8 PR:

- `backport-needed-v9`
- `backport-complete-v9`
- `no-backport`
- `v9-only`

Backport Method

Cherry-pick only:

```
git checkout develop-v9
git cherry-pick <commit>
git push
```

Squashing Guidance

To minimize cherry-picks:

- Squash or rebase feature PRs before merge
- Prefer one logical commit per PR

7. Release Workflow

1. Create `release/vX-Y.Z` from `develop-vX`
2. Stabilize (bugfixes, docs, packaging only)
3. Update version in `pyproject.toml`
4. Merge into:
 - `main-vX`
 - `develop-vX`
5. Tag on `main-vX`

This workflow is **recommended best practice** for parallel major versions.

8. Stale Branch Policy

A branch is **stale** if:

- No commits AND no PRs for 12 months

Lifecycle

1. Renamed to `stale/<original-name>`
2. Made read-only
3. Notification documented (issue or log)

- 4. Deleted after 12 more months unless approved

Stale branches:

- Are not valid PR targets
- Exist only for archival purposes

9. Decision Table

Change Type	Target
Bugfix to released code	develop-v8
Physics / features	develop-v8
Breaking / structural	develop-v9
Docs	Most relevant active branch

10. Breaking Changes & Deprecation

- Breaking changes allowed **only in v9**
- Must be documented in release notes
- Deprecations:
 - Warn in v8
 - Remove in v9

This document governs all UEDGE GitHub activity.