Branching Naming Convention

Create feature branches from `master` with consistent, discoverable names.

- **Pattern examples:**
  - `<binary_name>_golden_master`
  - `<binary_name>_integration_test`
  - `<feature_name>_<description>`
  - `bugfix_<issue_description>`
- **Examples:**
  - `csv_importer_golden_master`
  - `csv_importer_integration_test`
  - `parser_add_json_support`
  - `bugfix_memory_leak_handler`

Pull Requests (PR)

Before merging a specific branch into the `master` branch, the following steps should be validated:

- A code review by one or several contributors
- All CI/CD checks that perform format and lint validation, etc.
- Comprehensive documentation of the feature
- All tests passing successfully
- No merge conflicts with the target branch

Commit Message Convention

Follow a clear and consistent commit message format:

- **Format:** `<component>: <short description>`
- **Examples:**

- `csv_importer: add golden test harness`
- `parser: fix memory leak in JSON handler`
- `docs: update API documentation`

Git Workflow

```
# Start from master
git checkout master
git pull

# Create a focused feature branch
git checkout -b csv_importer_golden_master

# Make your changes and commit as you go
git add -A
git commit -m "csv_importer: add golden test harness"

# Keep your branch up to date with master
git checkout master
git pull
git checkout csv_importer_golden_master
git rebase master

# Resolve any conflicts if they arise
# After resolving conflicts:
git add -A
git rebase --continue

# Push your branch (use --force-with-lease after rebasing)
git push -u origin csv_importer_golden_master
# or if already pushed and rebased:
git push --force-with-lease origin csv_importer_golden_master

# Open a PR on the repository platform (GitHub, GitLab, etc.)
```

Handling Merge Conflicts

If conflicts occur during rebase:

1. Open the conflicting files and resolve the conflicts manually
2. Stage the resolved files: `git add <file>`
3. Continue the rebase: `git rebase --continue`
4. If you want to abort: `git rebase --abort`

After PR Approval and Merge

```
# Switch back to master
git checkout master
git pull

# Delete the local feature branch
git branch -d csv_importer_golden_master

# Delete the remote feature branch (optional, often done automatically)
git push origin --delete csv_importer_golden_master
```

Branch Protection Rules

The `master` branch is protected:

- Direct pushes are disabled
- All changes must go through a Pull Request
- Required approvals: minimum 1 reviewer
- CI/CD checks must pass before merging
- Branch must be up to date before merging

Best Practices

- Keep commits atomic and focused on a single change
- Write clear, descriptive commit messages
- Rebase your branch regularly to stay up to date with master
- Test your changes locally before pushing

- Keep PRs small and focused for easier review
- Respond promptly to review comments
- Delete branches after they are merged