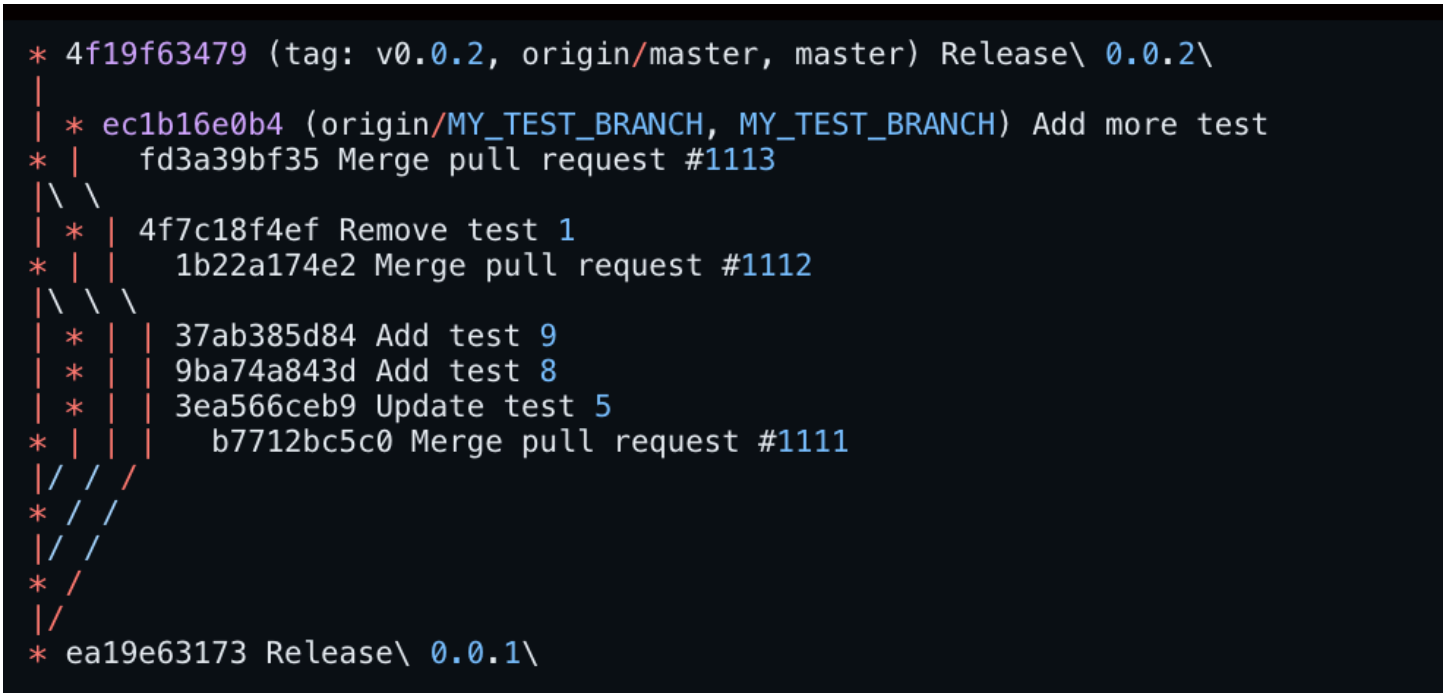
MERGING STRATEGY



Test Plan

To ensure a smooth and safe merging of MY_TEST_BRANCH into master, the approach must prioritize validation through comprehensive testing and version control best practices

Objectives

- Validate the MY_TEST_BRANCH by running all tests on the latest master
- Ensure compatibility between the MY_TEST_BRANCH and the target master branch
- Safeguard against regressions and integration issues

Workflow

Step 1: Update MY_TEST_BRANCH with the latest master

To incorporate the latest changes from master into MY_TEST_BRANCH, you can use one of two merging strategies: **merge** or **rebase**

Merge Strategy

- Fetch and merge the latest master into MY_TEST_BRANCH
- Benefits: Retains a clear commit history showing where master is integrated

```
git checkout MY_TEST_BRANCH
git fetch origin
git merge origin/master
```

- If conflicts arise, resolve and commit the changes

Rebase Strategy

- Reapply your branch commits on top of the latest master
- Benefits: Keeps a linear history

```
git checkout MY_TEST_BRANCH
git fetch origin
git rebase origin/master
```

- If conflicts arise, resolve and commit the changes

Step 2: Verify the Combined Codebase

Run all test suites for the updated MY_TEST_BRANCH to ensure compatibility and stability

```
npm run test
npm run lint
npm run integration-test
npm run performance-test
```

Step 3: Use CI/CD for Validation

Ensure the branch builds and passes tests on your CI/CD pipeline:

Push the updated MY_TEST_BRANCH

```
git push origin MY_TEST_BRANCH
```

Configure the CI/CD pipeline to:

- Build and Deploy the application
- Run unit, integration, and functional tests
- Execute performance and load tests
- Deploy to a staging environment for end-to-end testing

Step 4: Review Test Results

- Check for test failures or performance regressions
- Debug and fix issues on MY_TEST_BRANCH if necessary
- Rerun tests until the branch is stable

Step 5: Review and Create a Pull Request

- Create a pull request from MY_TEST_BRANCH to master in version control system
- Conduct code review to check for code issues before merging

Step 6: Merge into master

- If the PR checks are passed and there are no further issues, merge the PR into master

Step 7: Verify Post-Merge

- Run the regression test suite on master to ensure no issues were introduced.
- Deploy to a staging/production environment and execute smoke tests.

Test Categories

1. **Unit Tests**: Validate individual functions and components.
2. **Integration Tests**: Verify interactions between modules.
3. **Functional Tests**: Validate end-to-end workflows.
4. **Performance Tests**: Test response times and throughput.
5. **Regression Tests**: Ensure previous functionality remains intact.
6. **Security Tests**: Validate authentication and authorization.

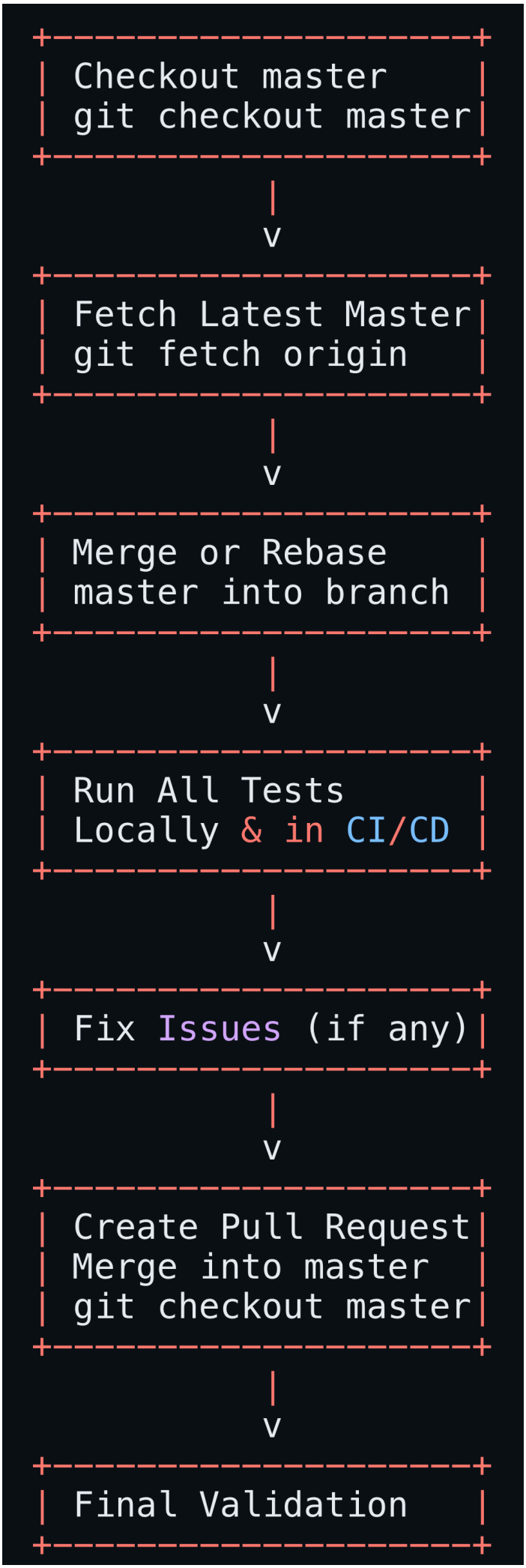
Puesdo Code

```
# Step 1: Update Branch
checkout_branch('MY_TEST_BRANCH')
fetch_updates('master')
merge_branch('master', 'MY_TEST_BRANCH')

# Step 2: Run Tests
if not run_tests('MY_TEST_BRANCH'):
    handle_failures()
    exit_process()

# Step 3: Merge into Master
create_pull_request('MY_TEST_BRANCH')
merge_branch('MY_TEST_BRANCH', 'master')
run_tests('master')
```

Flow Diagram


Checkout master
git checkout master
Fetch Latest Master
git fetch origin
Merge or Rebase
master into branch
Run All Tests
Locally & in CI/CD
Fix Issues (if any)
Create Pull Request
Merge into master
git checkout master
Final Validation

This strategy ensures that MY_TEST_BRANCH can be safely merged into master