Android Cast — Git flow (green master)

Field	Value
Version	1.1
Published	2026-05-21
Author(s)	Anton Afanaasyeu
Markdown source	docs/GIT_FLOW.md
Diagrams (PDF)	Mermaid → SVG → PNG @ 200 DPI
Diagrams (vector)	docs/_pdf_build/git_flow_diagram_*.svg

PDF diagrams are high-resolution raster with correct aspect ratio (not stretched). For infinite zoom, open the SVG files in Okular or a browser. Okular: View → Zoom for PDF; SVG opens as native vector.

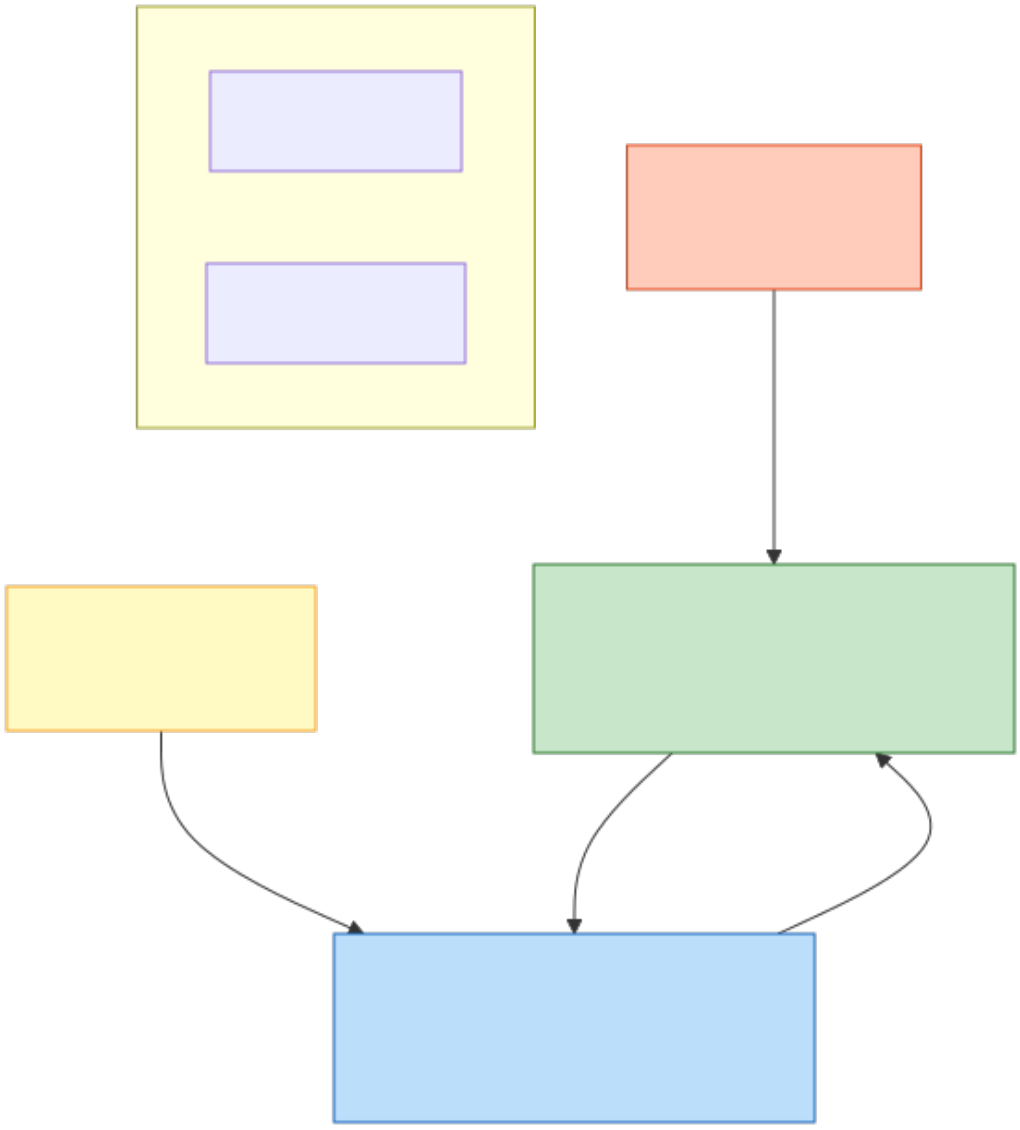
1. Goals

Goal	Mechanism
Always shippable master	Merge to master only when CI/tests pass
Safe integration	Features land on next first
Clear production line	Release tags only on master
Fast production patches	hotfix/* from master, then sync to next

2. Branches

Branch	Role	Direct commits?
master	Production; green; annotated tags vX.Y.Z	No — merge only
next	Integration / release candidate	No — merge only
feature/*, fix/*	Short-lived work	Yes; merge → next
hotfix/*	Urgent fix for released code	Yes; merge → master first

Branch model (green master)



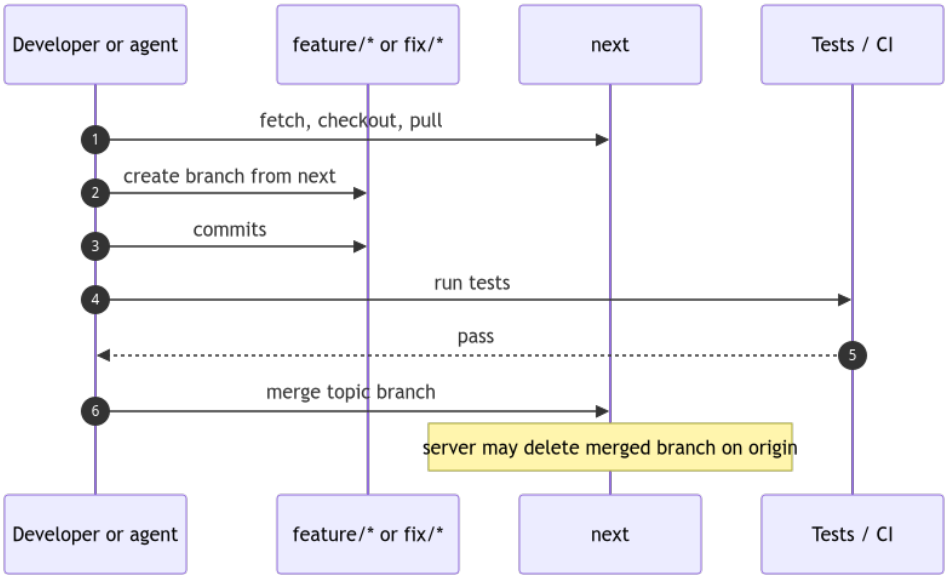
3. Daily development

- git fetch && git checkout next && git pull
- git checkout -b feature/short-description
- Commit on topic branch; run tests before merge
- Merge topic → next (after CI green)

Topic branch cleanup: optional locally. On the server, enable *delete branch on merge* (Gitea/GitLab/GitHub) so merged feature/* branches are removed automatically on origin.

Agents: never push directly to master or next unless the user explicitly requests a release or hotfix merge.

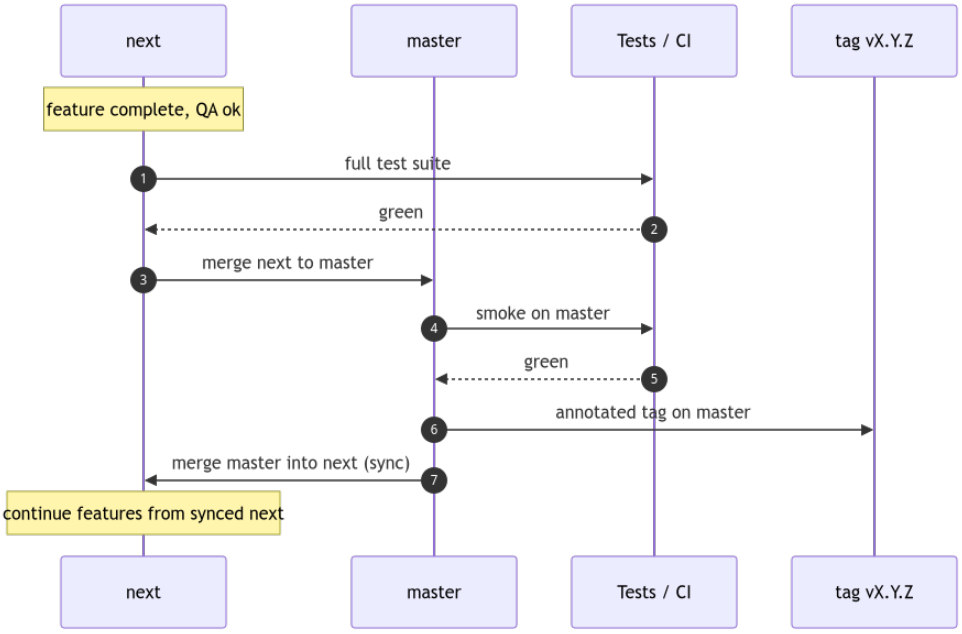
Daily development



4. Release (next → master)

- Confirm next is QA-ready and CI green
- Merge next → master
- Smoke / CI on master
- Tag on master: git tag -a v0.1.2 -m "Release v0.1.2"
- Sync: git checkout next && git merge master && git push
- Continue feature branches from updated next

Release: next → master



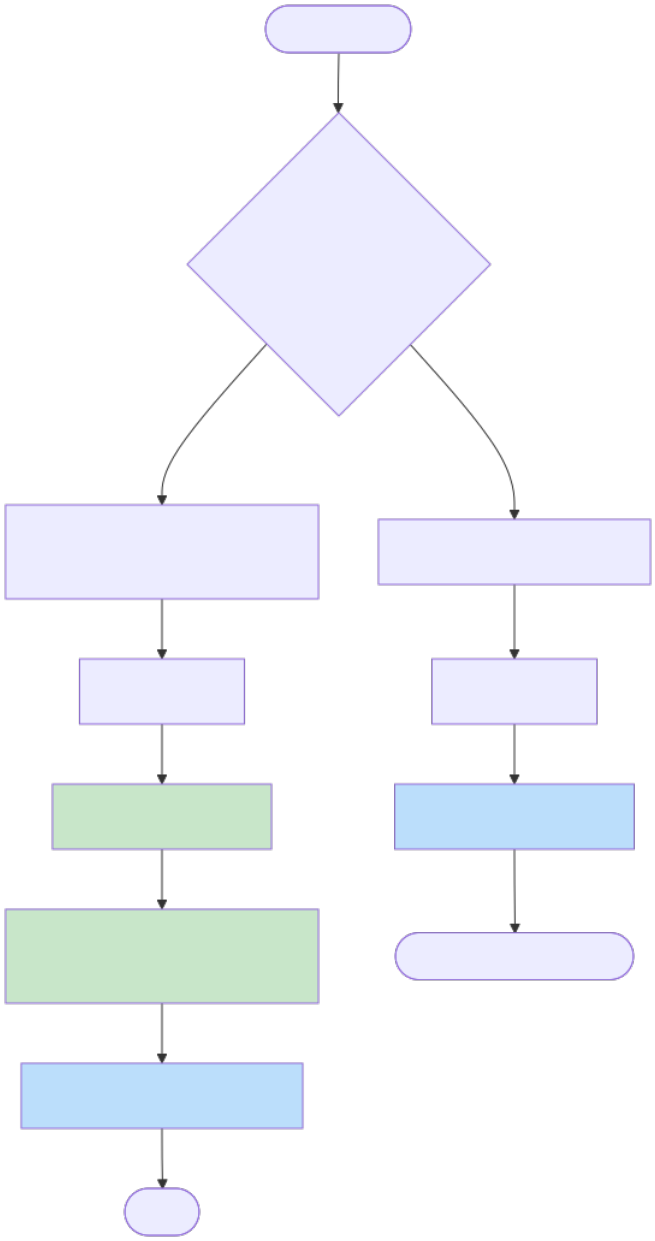
5. Hotfixes — recommended strategy

Default (option 1): branch hotfix/* from **master** when the bug affects released code. After fix and tests: merge → master, tag patch (e.g. v0.1.3), then **merge master → next** so integration does not miss the fix.

Exception (option 2): bug exists only on next — branch fix/* from next, merge to next only; ships with the next normal release.

Why not hotfix only on next? next often contains unreleased features; merging next to master for a patch would ship extra code or block the fix.

Hotfix decision and flow



5.1 Sync cheat sheet

Event	Action
After release + tag	merge master → next

After hotfix + tag	merge master → next
Feature done	merge topic → next only
Wrong base branch	rebase topic onto latest next

6. Branch protection and CI

Rule	master	next
CI required before merge	yes	yes
No WIP direct push	yes	yes
Annotated release tags	yes	no
Force-push	never	maintainer only, rare
Delete branch on merge	n/a	yes (topic branches)

Force-push on next: keep disabled by default. Allow only when you intentionally rewrite integration history (e.g. squash fixups before release) — never on master. Prefer merge/rebase on topic branches instead. If you force-push next, coordinate with anyone else using the repo and re-sync their clones.

7. Cursor / agent checklist

- Branch from next for features and non-production fixes
- Run tests before recommending merge to next
- Releases and tags only with explicit user approval
- Hotfix released code from master; sync master into next after tag
- Never force-push master; avoid force-push on next unless user requests

Regenerate: python3 docs/_pdf_build/build_git_flow_pdf.py