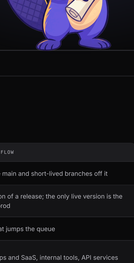
Which Git workflow? Gitflow, GitHub Flow, and trunk-based

Three questions and a decision matrix for choosing between Gitflow, GitHub Flow, and trunk-based development by product shape, team size, and deploy tempo.

01 GIT · WORKFLOW

Which Git workflow? Decide with three questions



Gitflow and GitHub Flow are both right, for different product shapes. Trunk-based is a decision about tempo. The answer lies not in the meeting but in the product itself.

Muhammet Şafak · October 3, 2026

02 THE MECHANICS OF TWO MODELS

Gitflow and GitHub Flow

TOPIC	GITFLOW	GITHUB FLOW
Branches	main, develop, feature/*, release/*, hotfix/*	A single main and short-lived branches off it
Release	Release-oriented; a separate release branch for each release	No notion of a release; the only live version is the one in prod
Hotfix	A separate hotfix/* line	A PR that jumps the queue
Natural habitat	Mobile apps, SDKs and libraries, on-premise or packaged software	Web apps and SaaS, internal tools, API services

03 THREE QUESTIONS, ONE DEFAULT

The product chooses the branch strategy

1 — How many versions are live at once?

`One → GitHub Flow. More than one → Gitflow.`

2 — Does the customer see my deploy frequency?

`No, I deploy whenever I want → GitHub Flow. Yes, there is a contract between releases → Gitflow.`

3 — Must a hotfix go through a release?

`No → GitHub Flow. Yes → Gitflow.`

4 — If the answers are mixed

`The product's shape hasn't settled yet; GitHub Flow for now. Moving from it to Gitflow is a small migration; unlike the reverse.`

04 TEAM AND DEPLOY TEMPO

Three workflows, four situations

The vertical axis shows team size, the horizontal axis deploy tempo. The cells are compiled from the Git in Production guide and the trunk-based post.

SMALL TEAM → LARGE TEAM

Gitflow
Versioned product
- Mobile app, App Store distribution
- Versioned package or library: a release branch helps

Trunk-based
In the guide: 50+ developers, 10+ deploys a day
- Continuous deployment culture
- Feature flag infrastructure in place

GitHub Flow or something simpler
3-person freelance agency
- Working directly without PRs is an option too
- In a team under 3, a trunk-based investment doesn't pay off

GitHub Flow
Team of 3–15
- SaaS, CI/CD in place
- One or two deploys a day

PLANNED, VERSIONED RELEASES → FREQUENT DEPLOYS

05 FROM THE GIT IN PRODUCTION GUIDE

Decision table

SITUATION	RECOMMENDATION
Mobile app, App Store distribution	Gitflow
SaaS, small-to-medium team, daily deploys	GitHub Flow
3-person freelance agency	GitHub Flow
50+ developers, 10+ deploys a day	Trunk-based
Versioned package or library	Gitflow
No plan, "whatever is needed"	GitHub Flow

06 WHEN TO GO TRUNK-BASED

Two different thresholds in the posts

TRUNK POST — Once deploy frequency reaches 5+ a week, the release branch stops being a structure and starts being an obstacle. Projects where trunk-based is advocated deploy 3–10 times a day.

GITFLOW POST — At 5+ deploys a day, even GitHub Flow's short-lived branches become a source of friction; beyond that threshold, trunk-based.

TIP — For a team deploying once a week, there is no reason forcing a move; Gitflow or GitHub Flow works fine at that rhythm.

07 TRUNK-BASED TARGETS

The numbers that carry the tempo

5–10 minutes
Target: pipeline time from merge to prod

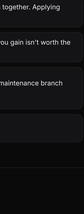
1–2 days
Threshold: the longest life of a feature branch

200–400 lines
Target: size of a PR with a single logical change

60–90 minutes
Review time for a PR of this size

08 BEFORE YOU SWITCH

The four foundations of trunk-based



- [ ] A green trunk at the contract level: guaranteed by CI and required PR checks
- [ ] Feature flag infrastructure: unfinished work enters the code but stays off for users
- [ ] A fast pipeline: 5–10 minutes from merge to prod
- [ ] A short-lived branch culture: closed the day it's opened or the next

09 TRUNK-BASED AND MEASUREMENT

Measure once you switch, and know when not to

Do

- Target for lead time for changes: p50 < 1 hour
- Target for change failure rate: < 15%
- Target for mean time to restore: < 30 minutes
- If the metrics get worse, look at CI and flag discipline first

Don't

- Don't switch when a regulated sector requires pre-prod sign-off
- Don't switch when 3.x and 4.x must be supported at the same time
- Don't switch on a team that can't guarantee a green CI
- Don't switch on a team under 3 that deploys once a week

10 MISMATCH ERRORS

Putting the model in the wrong place

ANTI-PATTERN — Mixing the three workflows: a develop branch, feature flags, and a release branch together. Applying half of three separate disciplines means getting the advantage of none.

GITFLOW IN SAAS — develop becomes a needless shadow prod; PRs get merged twice. The isolation you gain isn't worth the friction you pay.

GITHUB FLOW IN A PACKAGED PRODUCT — When a customer on an old version asks for a hotfix, the team goes looking for a maintenance branch that doesn't exist.

HYBRID MODEL — Applying one of two models correctly is always cheaper than inventing a third.

11 DECISION

Choose your Git workflow for the product; not the product for the workflow.

The choice is not "which is modern" but "which one does your product's shape make natural." When the number of versions, the customer type, or the deploy frequency changes, the decision reopens.