Flow One-Pagers

Heliconia Labs ApS

JULY 2026

Introduction

With abapGit flow, Heliconia Labs provides an end-to-end development and deployment process for SAP ABAP systems. It brings modern software delivery practices to ABAP: Git-based collaboration, pull requests, quality gates, code review, and independent changes that can move safely toward production.

Together with open tooling, this creates a modern workflow where feedback is visible early, quality is built in, and ABAP code becomes accessible to contemporary engineering tools. This foundation is essential for successful LLM implementations, where clean versioned code, automated checks, and repeatable workflows determine how effectively AI can assist.

Across all of these topics, the focus is developer experience and productivity: fewer manual checklists, clearer review context, faster feedback, and workflows that let SAP developers work with the same confidence and ergonomics as the rest of the engineering organization.

The following one-pagers highlight the workflows and focus areas that make this foundation practical, actionable, and ready to adopt: build quality in, automate test data and scenarios, review changes with LLM assistance, inner source across teams, onboard developers faster, roll back with confidence, build on open tooling instead of walled gardens, keep security simple with SAP-initiated communication and no new infrastructure, govern clean core decisions, monitor repository health, and sequence transports with preview deployments.

CONTENTS

Quality Built In

Make SAP quality part of the development workflow instead of a late project phase. Continuous integration runs the same checks every time a change is proposed, so developers do not need to remember what to run and teams get objective feedback before review, testing, transport sequencing, or release decisions become expensive.

With abaplint.app, ABAP static analysis becomes fast, configurable, and visible in the pull request. Developers see issues while the code is still fresh, reviewers get cleaner evidence, and quality standards become daily habits. For LLM-based changes, the same gates keep humans in the loop before code moves forward.

FLOW

1. Developer pushes

ABAP source, configuration, documentation, and tests land in GitHub as one reviewable change.

2. CI starts

Continuous integration runs static analysis, tests, security checks, and policy gates automatically.

3. Feedback returns

abaplint.app reports Clean ABAP, syntax, naming, and project-specific findings directly in the PR.

4. Human approves

Developers fix issues before approval, while reviewers validate LLM-based changes and final intent.

WHY IT MATTERS

- Shift feedback left: problems surface at commit and pull request time, not during transport or production validation.
- No checklist burden: developers do not need to remember which commands, static analysis, or test suites to run.
- Consistent standards: every change is checked against the same automated rules before expert review time is spent.

SAP FIT

Quality gates complement SAP transport discipline by adding earlier evidence. abaplint.app, abapGit, unit tests, and repository documentation form an open, inspectable toolchain that supports both ECC and S/4HANA work without locking teams into a closed quality process.

CI

CHECKS EVERY PULL REQUEST

abaplint.app

STATIC ANALYSIS FOR ABAP

Built In

QUALITY BEFORE TRANSPORT

Test Data and Scenario Automation

Make SAP testing repeatable by treating unit tests, test scenarios, expected outcomes, and representative test data as part of the development workflow.

Quality gates prove that code can pass checks, but business confidence also depends on realistic process paths. Pull requests can connect code changes with unit test coverage, scenarios, data, and evidence needed to show that the behavior works before release pressure builds.

FLOW

1. Scenario is described

The PR links code changes to business scenarios, expected behavior, and relevant SAP process steps.

2. Test data is prepared

Representative materials, customers, orders, roles, or fixture records are documented or generated repeatably.

3. Unit tests run early

Focused unit tests prove the code path first, then integration checks, regression scripts, or preview validation exercise the broader scenario.

4. Evidence stays attached

Results, screenshots, logs, and reviewer decisions remain connected to the pull request.

WHY IT MATTERS

- Earlier defect discovery: unit tests catch logic regressions before full SAP scenario validation is needed.
- Repeatable validation: teams reduce manual retesting of the same SAP process paths.
- Clearer business review: reviewers can see expected behavior, relevant data, and proof of execution.
- Better AI guardrails: LLM-assisted changes must prove behavior, not only syntax or style.

SAP FIT

SAP changes often depend on realistic business data and process context. Connecting ABAP code, unit tests, scenario descriptions, test data, automated checks, and review evidence in GitHub makes validation easier to repeat across ECC, S/4HANA, and preview environments.

Unit Tests

CODE PATHS PROVEN EARLY

Data

REPEATABLE TEST SETUP

Evidence

RESULTS LINKED TO PRS

LLM-Based Code Review for SAP Changes

Bring fast, consistent review feedback into the GitHub pull request before SAP experts spend their time.

A reviewer gets cleaner ABAP, clearer context, and fewer avoidable defects. The team keeps the four-eyes principle, with an AI assistant handling the first pass. The LLM provider is replaceable: use the model that fits the risk, data, and enterprise sourcing requirements.

FLOW

1. Developer opens PR

ABAP, config-as-code, documentation, and tests are visible in one review surface.

2. CI evidence appears

Static analysis, unit tests, security scanning, and policy checks give objective signals.

3. LLM review comments

The assistant highlights risky diffs, missing tests, unclear naming, guideline drift, and architecture impact.

4. Human approves

SAP experts decide with better signal, traceability, and shared context.

WHY IT MATTERS

- Higher review quality: consistent first-pass checks against Clean ABAP, local SAP guidelines, and PR evidence.
- Shift feedback left: comments arrive while the developer still has the change in working memory.
- Better use of experts: senior reviewers focus on architecture, business risk, and SAP-specific judgment.
- Auditable decisions: every suggestion, response, approval, and human override stays attached to the pull request.

SAP FIT

Complements a standard pull request process and a GitHub-based platform direction. It builds on CI evidence, open tooling, and repository-based documentation. Keeps accountability with the human approver while enabling real autonomous first-pass analysis from any suitable provider.

Fast

FEEDBACK AT PR CREATION

Any Provider

MODEL OF YOUR CHOICE

Learn

GUIDELINES REINFORCED DAILY

Inner Sourcing for SAP Teams

Use GitHub pull requests to make SAP knowledge discoverable, reusable, and easier to improve across teams.

SAP development stops being hidden in isolated systems and becomes part of the shared engineering ecosystem, helping teams upskill into normal software delivery habits without reinventing a separate SAP-only wheel.

FLOW

1. Discover

Teams find existing SAP packages, APIs, extensions, and examples in GitHub.

2. Learn

README files, PRs, ADRs, checks, and architecture notes show how the solution is meant to evolve.

3. Contribute

Developers propose improvements through a familiar pull request workflow.

4. Reuse

Shared components become easier to trust because ownership and quality signals are visible.

WHY IT MATTERS

- Less duplicate work: teams can see and reuse existing solutions before creating another variant.
- Better onboarding: new SAP developers learn from real repositories, reviews, accepted patterns, and visible architecture decisions.
- Stronger ownership: CODEOWNERS, branch protection, and PR review make responsibility explicit.
- Cross-domain collaboration: non-SAP engineers can contribute documentation, APIs, tests, and automation.

SAP FIT

Fits organizations that have standardized on GitHub for development transparency. Supports ABAP and non-ABAP assets in one collaboration model. Gives SAP teams a practical path into modern engineering habits without losing review governance or creating a walled garden.

Find

EXISTING SAP ASSETS

Share

PATTERNS ACROSS TEAMS

Own

WITH VISIBLE CODEOWNERS

Developer Onboarding and Enablement

Make SAP development easier to join, understand, and improve by moving knowledge into repositories, pull requests, automated checks, and repeatable workflows.

New developers, consultants, and cross-functional engineers should not depend on tribal knowledge as the main interface to the SAP landscape. Repositories can show how the system is organized, which standards apply, who owns what, and how a change moves safely from idea to production.

FLOW

1. Repository explains

README files, package structure, ADRs, examples, and ownership metadata show how the solution is organized.

2. Checks teach standards

abaplint.app, CI, tests, and PR templates reinforce local rules every time a developer contributes.

3. Reviews share context

Pull requests make naming, architecture, business intent, and SAP-specific tradeoffs visible.

4. Developers contribute

New contributors learn from working examples, early feedback, and repeatable delivery paths.

WHY IT MATTERS

- Faster onboarding: developers learn from real repositories, accepted patterns, review history, and automated feedback.
- Less expert dependency: teams reduce reliance on a few long-tenured SAP specialists for everyday delivery knowledge.
- Standards in daily work: Clean ABAP, local conventions, security expectations, and testing habits are reinforced continuously.
- Reusable learning material: pull requests, comments, checks, and documentation become examples for the next contributor.

SAP FIT

Supports the move from SAP-only delivery habits to shared engineering practices without losing SAP-specific guidance. Helps ABAP developers, consultants, architects, and non-SAP engineers understand the same repository evidence, quality gates, ownership model, and release path.

Learn

STANDARDS VISIBLE IN REPOSITORIES

Contribute

GUIDED BY PRS AND CHECKS

Scale

LESS RELIANCE ON TRIBAL KNOWLEDGE

Easy Rollback with GitHub as the Change Memory

Make every SAP change easier to understand, compare, and reverse by keeping the source, review, and release evidence in GitHub.

Rollback becomes a prepared operational path, not an emergency reconstruction exercise. Teams can compare production state with reviewed source, understand why a change moved forward, and propose a controlled revert using the same pull request workflow.

FLOW

1. Change is versioned

Every approved object change has a commit, pull request, and linked rationale.

2. Release state is known

Checks, approvals, transports, and deployment notes show what reached production.

3. Impact is compared

Teams inspect the current state against the original change, related PRs, and incident context.

4. Revert is proposed

A rollback PR can be reviewed, tested, approved, and transported using the same controls.

WHY IT MATTERS

- Less production stress: teams can compare production state, understand change history, and create a controlled revert path.
- Clear accountability: rollback decisions use the same transparent PR evidence as forward changes.
- Lower recovery time: fewer manual investigations across transports, screenshots, and local notes.
- Reusable patterns: rollback playbooks become repeatable across ECC, S/4HANA, and side-by-side extensions.

SAP FIT

Supports SAP transport discipline while adding modern version comparison and revert workflows. Helps separate source rollback decisions from deployment orchestration and environment constraints. Improves audit readiness by connecting commits, PR approvals, checks, releases, transports, and incident response.

Know

WHAT CHANGED IN PROD

Compare

SOURCE TO RELEASE STATE

Revert

THROUGH REVIEW CONTROLS

GitHub and Open Tooling

This approach builds on proven platform choices: GitHub for inner sourcing, GitHub Actions for automation, and open tooling for SAP developer feedback.

This is not a new isolated SAP platform. It is an extension of the engineering platform already chosen for transparency, collaboration, and automation, based on open tooling with a proven track record.

FLOW

1. Repository first

Source, docs, decisions, and checks live together in GitHub.

2. Automation built in

GitHub Actions runs checks, gates, and release workflows on every change.

3. Open SAP checks

abapGit, test tooling, and CI quality gates keep feedback portable.

4. Autonomous, governed

AI assistance can use any provider and work on PR evidence while humans retain control.

WHY IT MATTERS

- Reuse proven choices: for many organizations GitHub is already the consolidated engineering platform, and SAP development can join it.
- Avoid vendor lock-in: open tooling keeps checks inspectable, portable, proven, and easier to evolve.
- Meet teams where they work: PRs become the shared control point for review, quality, security, progress monitoring, and learning.

SAP FIT

Supports both ECC and S/4HANA transition needs by making code and decisions visible outside SAP GUI boundaries. Connects SAP governance with enterprise DevSecOps without forcing feedback into a closed SAP-only tool. Creates a platform for future automation, including real autonomous agents, Copilot-style assistance, and other provider-neutral LLM options.

Chosen

GITHUB PLATFORM DIRECTION

Automated

ACTIONS ON EVERY CHANGE

Open

PORTABLE SAP CHECKS

Security

Adopt abapGit flow without changing the security posture of the SAP landscape. There are typically no changes to network or VPN setup, no SAP Basis setup, and no additional infrastructure to run: no servers, agents, or middleware to install, operate, patch, or audit.

The SAP system initiates all communication, triggered by user actions and using each developer's existing accesses and authorizations. Nothing connects into the SAP system from outside, and no third-party component holds standing access. On the platform side, GitHub Advanced Security adds secret scanning, code scanning, and dependency visibility to every pull request.

FLOW

1. User acts

A developer triggers the exchange from the SAP system; nothing runs without a user-initiated action.

2. SAP calls out

The SAP system initiates outbound HTTPS to GitHub. No inbound connections or open ports are needed.

3. Existing controls apply

Existing SAP accesses and authorizations, TLS, and the existing outbound network path govern the exchange.

4. Platform checks run

GHAS secret scanning, code scanning, and dependency review add security signals to every PR.

WHY IT MATTERS

- Small attack surface: outbound-only communication means no inbound connectivity, no open ports, and no standing agents in the landscape.
- Fast security approval: with typically no firewall, network, or VPN changes to negotiate, adoption does not stall in infrastructure projects.
- Nothing new to operate: no SAP Basis setup and no additional servers to size, patch, monitor, or certify — existing SAP and GitHub operations cover it.
- No new access model: developers work with their existing accesses and authorizations, so authorization concepts and audits stay unchanged.

SAP FIT

Communication uses the SAP system's standard outbound HTTPS capabilities and runs in the context of the acting user with existing accesses and authorizations. Security review can focus on a single outbound connection to GitHub instead of a new platform footprint, and the model is the same for ECC and S/4HANA.

Outbound Only

SAP INITIATES ALL COMMUNICATION

No New Infra

NO SAP BASIS SETUP, NOTHING TO OPERATE

No VPN Changes

TYPICALLY NO NETWORK CHANGES

Clean Core and Extension Governance

Use GitHub-based review and automation to make SAP clean core decisions visible, deliberate, and repeatable.

Teams should be able to see when a change belongs in standard configuration, an ABAP enhancement, a side-by-side extension, an API, or a temporary exception. Clean core becomes practical when architectural intent is reviewed before implementation choices harden.

FLOW

1. Intent is declared

The PR explains whether the change touches core ABAP, an enhancement point, an API, configuration, or an extension.

2. Rules apply

Review templates, CODEOWNERS, checks, and guidelines steer clean core decisions before merge.

3. Exceptions are owned

Necessary deviations are visible, justified, assigned, and time-bounded instead of becoming hidden debt.

4. Upgrade risk drops

Teams preserve a clearer path for S/4HANA upgrades, cloud readiness, and future SAP maintenance.

WHY IT MATTERS

- Practical clean core: architecture guidance appears in the pull request, not only in strategy documents.
- Lower upgrade risk: custom ABAP, modifications, and exceptions become easier to identify and manage.
- Lightweight governance: architects get a review point without slowing every low-risk change.
- Visible rationale: teams can compare extension options and understand why a decision was made.

SAP FIT

Supports ECC-to-S/4HANA transition work by connecting abapGit flow with enterprise architecture. Every change can be reviewed not only for correctness, but also for where it should live in the SAP landscape and how it affects future maintainability.

Decide

CORE, EXTENSION, OR CONFIG

Govern

ARCHITECTURE RULES IN PRS

Upgrade

LOWER FUTURE MIGRATION RISK

Repository Health and Progress Visibility

Turn SAP development from a hidden transport queue into an observable engineering system. GitHub repositories, pull requests, CI checks, review activity, security findings, and release signals make progress visible without asking teams to maintain separate status decks.

Repository health gives architects, leads, and managers a shared view of delivery quality. Teams can see where work is blocked, which changes carry risk, whether ownership is clear, and how modern SAP engineering habits are being adopted over time.

FLOW

1. Work is captured

ABAP source, documentation, tests, checks, and review discussions live together in GitHub.

2. Signals are collected

CI, abaplint.app, GHAS, PR status, review activity, and release metadata create objective evidence.

3. Progress is monitored

Teams see bottlenecks, quality trends, ownership gaps, and release readiness across repositories.

4. Decisions improve

Architects and managers act on real delivery signals instead of anecdotes or late-stage surprises.

WHY IT MATTERS

- Transparent delivery: SAP progress becomes visible through the normal pull request and release workflow.
- Less reporting overhead: status comes from real engineering activity instead of separate manual updates.
- Earlier risk signals: review queues, failing checks, unclear ownership, and risky changes surface before release pressure builds.
- Continuous governance: architecture and quality discussions can happen from repository evidence, not only in meetings.

SAP FIT

Complements SAP transport governance by showing the health of the work before it reaches release planning. Connects ABAP repositories, CI evidence, security signals, ownership, and deployment context into one inspectable view that supports both team-level improvement and leadership-level adoption tracking.

Visible

DELIVERY STATUS FROM PR EVIDENCE

Measurable

QUALITY, REVIEW, AND RELEASE SIGNALS

Actionable

BOTTLENECKS AND RISK SURFACED EARLY

Transport Sequencing and Preview Deployments

Make SAP releases easier to reason about before production by showing planned changes, dependencies, transport order, and preview evidence in one workflow.

Transport sequencing LABS PREVIEW helps teams see what can move, what must wait, and what release state they are targeting. Preview deployments LABS PREVIEW give reviewers a temporary environment to inspect change behavior before production release. Impact analysis LABS PREVIEW highlights the expected production effect before approval.

FLOW

1. Changes are collected

Approved PRs, release tags, transport candidates, and deployment intent are visible together.

2. Sequence is checked

LABS PREVIEW

Teams compare dependencies, target states, and transport order before changes move.

3. Preview is deployed

LABS PREVIEW

A temporary environment lets reviewers inspect behavior before production release.

4. Release is decided

Humans approve with clearer context about risk, order, and expected production impact.

WHY IT MATTERS

- Fewer release surprises: dependencies and ordering issues become visible before transports move.
- Better review evidence: reviewers inspect behavior and release context, not only source diffs.
- Safer transport planning: release state and transport intent become explicit.
- Stronger rollback readiness: known source and deployment state make reversals easier to reason about.

SAP FIT

Complements SAP transport discipline by adding GitHub-based visibility before release. Keeps transport execution governed while giving teams earlier insight into dependency order, release state, and preview behavior.

Sequence

LABS PREVIEW

PLAN TRANSPORT ORDER

Preview

LABS PREVIEW

INSPECT BEFORE RELEASE

Impact analysis

LABS PREVIEW

ASSESS PRODUCTION EFFECT