perforce

CI/CD Static Analysis Integration Checklist

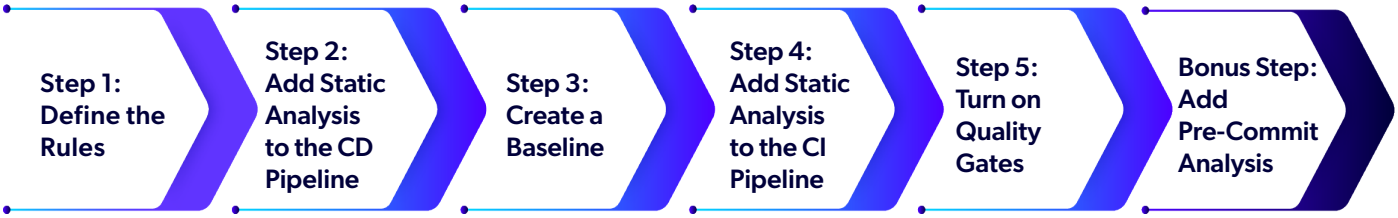
Modern software development moves fast. With Continuous Integration and Continuous Delivery (CI/CD) pipelines enabling teams to push code changes multiple times a day, maintaining code quality becomes both more critical and more challenging.

Adding analysis and checks to your CI/CD pipelines will prevent coding flaws and potential security vulnerabilities from slipping through to later phases of testing, or — worse still — into production.

To truly optimize development workflows and shift-left, integrating static analysis/SAST into both your CI and CD pipelines is essential — but it is also important to do this without obstructing the existing processes.

Follow the maturity steps from the checklist below to align with our recommended best practices.

Static Analysis CI/CD Integration Maturity Stages



6 Steps to Add Static Analysis to the CI/CD Pipeline

Step	Description/Benefits	Checklist
Step 1: Define the Rules	This step focuses your teams' thinking on quality, safety, and security. Defining rules is a team effort. Involving Development, Security, and QA teams ensures broader coverage of vulnerabilities, aligns rules with real-world workflows, and fosters shared ownership of code quality and security.	**Ask your teams:** ☐ What rules are **required** to be enforced? - Check for coding standards that help meet industry standards for quality, safety, and security. ☐ If not required, what rules would you **like** to be enforced? - You can also use static analysis tools, like Perforce QAC and Perforce Klocwork, to create your own internal standard.
	***Helpful Tip: Start with a smaller set of rules that are always strictly enforced, rather than having too many.***	
Step 2: Add Static Analysis to the CD Pipeline	This step begins automated enforcement of code quality and provides measurable insights into technical health. You can start tracking and optimizing your selected rules.	☐ Add your static analysis tool(s) to your existing CD pipelines for Master branch builds. ☐ Run these analyses nightly, weekly, or per-revision. ☐ Review results. ☐ Refine rules.
Step 3: Create a Baseline	Once you have full analysis results from the CD pipelines, a decision should be made about what to baseline. By starting from zero, you separate existing from new findings, creating new technical debt.	☐ Determine if the results are valid and valuable. ☐ Decide what to baseline: - What needs to be fixed now? - What can be left to the backlog?
Step 4: Add Static Analysis to the CI Pipeline	Now that you've addressed CD, it's time to implement the new static analysis stage to your CI pipelines to kick-start shift-left. In this phase, developers begin to get early visibility into code quality issues, enabling them to catch and fix problems before they escalate. This is the essence of shift-left, improving efficiency and reducing downstream defects.	☐ Run static analysis tools on each new branch as part of your branch and merge strategy. ☐ Report and triage new findings (the deltas) in the branch review process. ☐ Analyzing just the change code (change-set) will improve analysis times and minimize feedback loops.
Step 5: Turn on Quality Gates	Now it's time to turn on Quality Gates for your static analysis results in the CI pipelines. Quality gates enforce code quality and security. The development team can now code with confidence, trusting that substandard code will be prevented from passing through.	☐ Roll out static analysis rules enforcement through CI Quality Gates. ☐ New technical debt is now blocked from entering the codebase.

Bonus Step:
Add Pre-Commit Analysis

Optimal shift-left brings the process of finding and fixing to where the code is created: back to development.

By finding and addressing issues early, organizations save time, money, and a lot of rework.

- [ ] Optionally, make the tools available to developers on the desktop or within their IDE to check for new issues before they even commit.
- [ ] Ideally, provide developers with efficient options to resolve issues, such as AI code assistance tools.

Shift-Left development is a practice that helps developers find and fix vulnerabilities and coding errors as early as practical in the software development lifecycle (or, to the left of the linear development timeline, before the product has been released to end users).

Adding tools like static code analyzers enables you to adopt shift-left methodology faster, more accurately, and at scale.

Automate CI/CD With Perforce Static Analysis

Perforce Static Analysis tools — QAC and Klocwork — automate the CI/CD pipeline and beyond with quality checks, consistent policy enforcement, and security assurance — accelerating your time-to-market. On the desktop, developers get immediate feedback, improved code quality, faster code reviews, and the tools they need to be empowered.

Plus, with the integration of artificial intelligence (AI), developers can now go beyond issue detection to intelligent resolution. Perforce AI Remediation seamlessly integrates with the desktop environment, harnesses Klocwork/QAC analysis results, supporting flexible LLM selection, and enables interactive collaborative sessions — empowering teams to refine code faster and more effectively.

See for yourself how Perforce Static Analysis optimizes shift-left. Request your free trial today.

Free Trial ▶

www.perforce.com/products/sca/free-static-code-analyzer-trial