Source ID: How the adoption of feature toggles correlates with branch merges and defects in software development
All Coded Practices:

- Use short-lived branches
- Commit directly to trunk
- Staged rollout with rollback support
- Shared ownership
- Requires strong discipline and team coordination (-)
- Use feature flags reduce merge effort
- Feature flag complexity and debt management required (-)
- Use feature flags to reduce need for long-lived branches
- Use feature flags to decouple deployment from release
- Use internal/custom-built tools for specialized needs

Source ID: An empirical study on principles and practices of continuous delivery and deployment
All Coded Practices:

- Use a single shared trunk
- Staged rollout with rollback support
- Accelerate release cycles (+)
- Use feature flags to decouple deployment from release
- Automate test execution (unit, integration, e2e)
- Promote frequent and rapid integration (CI) (+)
- Commit frequently (daily)
- Use short-lived branches
- Transparency is emphasized, ensuring developers are aware of the current build status and rollout information.
- Reduce merge conflicts (+)
- Shortens automated feedback cycles by testing each trunk commit against the main codebase through CI/CD pipelines (+).
- Simplifies the development workflow (+)
- Feature flag complexity and debt management required (-)
- Enforce discipline for stable builds and testing
- Cultural or organizational resistance is common (-)
- Integrate reliable CI/CD and automation tooling
- Initial migration effort was significant. (-)

Source ID: Feature Toggles: Practitioner Practices and a Case Study
All Coded Practices:

- Use feature flags to reduce need for long-lived branches
- Use feature toggles to decouple deployment from release

- Use feature flags to Support progressive delivery (A/B testing, dark launches)
- Integrate reliable CI/CD and automation tooling
- Shortens automated feedback cycles by testing each trunk commit against the main codebase through CI/CD pipelines (+).
- Accelerate release cycles (+)
- Requires robust testing and test infrastructure (-)
- Staged rollout with rollback support
- Feature flag complexity and debt management required (-)

Source ID: Best Practices for Version Control and Release Management Using GitLab CI/CD
All Coded Practices:
Use short-lived branches
Commit frequently (daily)

- Staged rollout with rollback support
- Tag releases for versioning and tracking
- Require code reviews before merges
- Automate code quality checks (linting, testing, analysis)
- Use of CI/CD tools and pipelines
- Accelerate release cycles (+)
- Promote frequent and rapid integration (CI) (+)
- Reduce merge conflicts
- Simplifies the development workflow
- Requires strong discipline and team coordination (-)
- Integrate reliable CI/CD and automation tooling
- Requires robust testing and test infrastructure (-)
- Needs reliable CI/CD and automation tooling

Source ID: DORA Platform: DevOps Assessment and Benchmarking
All Coded Practices:

- Use short-lived branches
- Use a single shared trunk
- Merge changes quickly
- Use CI/CD pipelines for build, test, and deployment
- Cloud-hosted or scalable infrastructure tools used
- Use of CI/CD tools and pipelines
- Accelerate release cycles (+)
- Needs reliable CI/CD and automation tooling
- Cultural or organizational resistance is common

Source ID: Towards Continuous Delivery by Reducing the Feature Freeze Period: A Case Study
All Coded Practices:

- Use short-lived branches
- Use temporary release branches
- Use a single shared trunk
- Use feature flags to decouple deployment from release
- Use CI/CD pipelines for build, test, and deployment
- Promote shared or collective code ownership
- Encourage team collaboration and communication
- Containerization and orchestration tools used
- Use of custom or internal tooling
- Use of CI/CD tools and pipelines
- Accelerate release cycles (+)
- Automate test execution (unit, integration, e2e)
- Feature flag complexity and debt management required
- Needs reliable CI/CD and automation tooling
- Cultural or organizational resistance is common

Source ID: Benefits and Challenges of Continuous Integration and Delivery: A Case Study
All Coded Practices:
Commit frequently (daily)
Use short-lived branches
Use a single shared trunk
Cherry-pick changes to release branches
Reduce merge conflicts (+)
Shortens automated feedback cycles by testing each trunk commit against the main
Use feature flags to safely integrate incomplete features
Integrate CI/CD tools
Maintain trunk in a releasable state
Promote shared or collective code ownership
Use pair programming or mob programming for collaboration
Encourage team collaboration and communication
Mandatory code review policy
Automate code quality checks (linting, testing, analysis)
Enforce discipline for stable builds and testing
Adopt CI/CD pipelines for automated integration and delivery
Encourage team collaboration and shared ownership
Feature flag complexity and debt management required

Source ID: Continuous Integration! You Keep Using Those Words. I Do Not Think It Means What You Think It Means
All Coded Practices:
Shortens automated feedback cycles by testing each trunk commit against the main
Enforce discipline for stable builds and testing
Encourage team collaboration and communication
Use of CI/CD tools and pipelines
Accelerate release cycles (+)
Improve software quality and stability
Needs reliable CI/CD and automation tooling
Cultural or organizational resistance is common

Source ID: Shifting to Modern Software Development Infrastructures
All Coded Practices:

- Use short-lived branches
- Use feature flags to safely integrate incomplete features
- Integrate CI/CD tools
- Adopt CI/CD pipelines for automated integration and delivery
- Use tags for releases
- Mandatory code review policy
- Use pair programming or mob programming for collaboration
- Automate code quality checks (linting, testing, analysis)
- Reduce merge conflicts (+)
- Reduced lead time helps deliver features and fixes faster, respond to user feedback more quickly
- Improve software quality and stability
- Encourage team collaboration and shared ownership
- Maintains system reliability despite rapid delivery, thanks to automated testing and CI/CD safeguards.
- May be difficult for inexperienced developers (-)
- Cultural or organizational resistance is common (-)
- Initial migration effort was significant. (-)

Source ID: Facebook’s Trunk-Based Development (take 2)
All Coded Practices:

- Use temporary release branches
- Use branch by abstraction
- Use feature flags to Support progressive delivery (A/B testing, dark launches)
- Deploy frequently and provide fast feedback

- Make small, frequent commits
- Enforce discipline for stable builds and testing
- Integrate CI/CD tools
- Adopt CI/CD pipelines for automated integration and delivery
- Staged rollout with rollback support
- Rollback management and data compatibility needed
- Feature flag complexity and debt management required

Source ID: Avoiding 'Big Bang' for Branch By Abstraction
All Coded Practices:

- Branch by abstraction
- Use short-lived branches
- Trunk-aligned refactoring
- Automated build and tests always pass on the CI
- Enforce discipline for stable builds and testing

Source ID: Trunk-based development
All Coded Practices:

- Commit frequently (daily)
- Merge changes quickly
- Use a single shared trunk
- Use feature flags to safely integrate incomplete features
- Automate test execution (unit, integration, e2e)
- Use CI/CD pipelines for build, test, and deployment
- Make small, frequent commits
- Require code reviews before merges
- Enforce discipline for stable builds and testing
- Adopt CI/CD pipelines for automated integration and delivery
- Reduce merge conflicts
- Enable fast and continuous feedback
- Requires strong discipline and team coordination
- May be difficult for inexperienced developers
- Feature flag complexity and debt management required
- Needs reliable CI/CD and automation tooling

Source ID: Don’t you know trunk-based development is where it’s at?
All Coded Practices:

- Maintain a trusted build with test-driven practices

- Maintain trunk in a releasable state
- Promote shared or collective code ownership
- Require code reviews before merges
- Use pair programming or mob programming for collaboration
- Use of CI/CD tools and pipelines
- Encourage team collaboration and shared ownership
- Enable fast and continuous feedback
- Maintain traceability and code history
- Requires strong discipline and team coordination
- Requires robust testing and test infrastructure
- Needs reliable CI/CD and automation tooling
- CI + Trunk practices common in OSS
- Use of OSS-style contribution workflows

Source ID: Trunk-Based Deployment Just Makes Things Worse
All Coded Practices:

- Cherry-pick changes to release branches
- Use CI/CD pipelines for build, test, and deployment
- Use CI gates to block commits on failure
- Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
- Use of CI/CD tools and pipelines
- Requires robust testing and test infrastructure
- CI + Trunk practices common in OSS
- Use of OSS-style contribution workflows

Source ID: Why I love Trunk Based Development (or pushing straight to master)
All Coded Practices:

- Use short-lived branches
- Use Branch-by-Abstraction
- Use feature flags to safely integrate incomplete features
- Enable frequent and fast releases
- Promote shared or collective code ownership
- Encourage team collaboration and communication
- Use pair programming or mob programming for collaboration
- Use of CI/CD tools and pipelines
- Encourage team collaboration and shared ownership
- Maintain traceability and history
- Improve software quality and stability
- Requires strong discipline and team coordination
- Needs reliable CI/CD and automation tooling

Source ID: A Complete Guide to Trunk-Based Development
All Coded Practices:

- Use short-lived branches
- Use a single shared trunk
- Merge changes quickly
- Commit frequently (daily)
- Use Branch-by-Abstraction
- Use toggles to manage risk during development
- Use CI/CD pipelines for build, test, and deployment
- Make small, frequent commits
- Promote shared or collective code ownership
- Enforce discipline for stable builds and testing
- Use of CI/CD tools and pipelines
- Reduce merge conflicts
- Enable fast and continuous feedback
- Requires strong discipline and team coordination
- Feature flag complexity and debt management required
- Needs reliable CI/CD and automation tooling
- CI + Trunk practices common in OSS
- Use of OSS-style contribution workflows

Source ID: You Should be Doing Trunk Based Development
All Coded Practices:

- Use short-lived branches
- Merge changes quickly
- Commit frequently (daily)
- Use CI/CD pipelines for build, test, and deployment
- Use of Tags Instead of Branches for Releases
- Make small, frequent commits
- Encourage team collaboration and communication
- Require code reviews before merges
- Use of CI/CD tools and pipelines
- Improve software quality and stability
- Requires robust testing and test infrastructure
- Use of mailing lists for coordination
- Use of OSS-style contribution workflows

• Emphasis on shared code ownership

Source ID: Scaled Trunk-Based Development
All Coded Practices:
Use short-lived branches
Merge changes quickly
Cherry-pick changes to release branches
Use temporary release branches
Use a single shared trunk
Use feature flags to safely integrate incomplete features
Use CI/CD pipelines for build, test, and deployment
Maintain trunk in a releasable state
Use temporary release branches when needed
Cherry-pick commits to release branches
Promote frequent and rapid integration
Reduce merge conflicts
Requires strong discipline and team coordination
Requires robust testing and test infrastructure
Needs reliable CI/CD and automation tooling
Adoption of practices from known OSS projects
Use of OSS-style contribution workflows

Source ID: Trunk-based Development: Pros, Cons & Why You Should Consider Adopting It
All Coded Practices:
Commit frequently (daily)
Commit directly to trunk
Merge changes quickly
Use CI/CD pipelines for build, test, and deployment
Maintain trunk in a releasable state
Enforce discipline for stable builds and testing
Encourage team collaboration and communication
Require code reviews before merges
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Use of CI/CD tools and pipelines
Reduce merge conflicts
Encourage team collaboration and shared ownership
Improve software quality and stability
Requires strong discipline and team coordination
Requires robust testing and test infrastructure
Needs reliable CI/CD and automation tooling

Source ID: Trunk Based Development vs Feature Driven Development

All Coded Practices:
Use short-lived branches
Use a single shared trunk
Merge changes quickly
Commit frequently (daily)
Use CI/CD pipelines for build, test, and deployment
Maintain trunk in a releasable state
Encourage team collaboration and communication
May be difficult for inexperienced developers
Adoption of practices from known OSS projects
Use of OSS-style contribution workflows

Source ID: Development and Deployment at Facebook

All Coded Practices:
Commit frequently (daily)
Avoid long-lived branches
Use toggles to enable safe rollouts and quick rollback
Use CI/CD pipelines for build, test, and deployment
Deploy frequently and provide fast feedback
Support rollback and staged releases (e.g., canary)
Use feature toggles to decouple release and deployment
Require code reviews before merges
Automate code quality checks (linting, testing, analysis)
Cloud-hosted or scalable infrastructure tools used
Use of custom or internal tooling
Use of CI/CD tools and pipelines
Enable fast and continuous feedback
Leverage automation and CI/CD for reliability
Support safe experimentation and rollback
Requires robust testing and test infrastructure
Feature flag complexity and debt management required
Adoption of practices from known OSS projects
Use of OSS-style contribution workflows

Source ID: CI/CD Unleashed

All Coded Practices:
Commit frequently (daily)
Support Continuous Integration
Use a single shared trunk
Use CI/CD pipelines for build, test, and deployment

Enable frequent and fast releases
Enforce discipline for stable builds and testing
Support early failure detection with proper tooling
Use of CI/CD tools and pipelines
Tooling includes quality and monitoring support
Requires strong discipline and team coordination
Challenging in legacy systems or during migration
Needs reliable CI/CD and automation tooling

Source ID: Tech Leadership Playbook - Chapter 6: SDLC
All Coded Practices:
Use short-lived branches
Merge changes quickly
Commit frequently (daily)
Use CI/CD pipelines for build, test, and deployment
Use temporary release branches when needed
Gate production deployment with QA signoff
Tag releases for versioning and tracking
Support manual gating in staged release flows
Require code reviews before merges
Automate code quality checks (linting, testing, analysis)
Encourage team collaboration and shared ownership
Accelerate release cycles
Improve software quality and stability
Requires strong discipline and team coordination
Requires robust testing and test infrastructure
Needs reliable CI/CD and automation tooling
Adoption of practices from known OSS projects
Use of OSS-style contribution workflows

Source ID: DevSecOps Adventures - Chapter 4
All Coded Practices:
Support Continuous Integration
Merge changes quickly
Commit frequently (daily)
Enable frequent and fast releases
Enforce discipline for stable builds and testing
Containerization and orchestration tools used
Use of CI/CD tools and pipelines
Reduce merge conflicts
Encourage team collaboration and shared ownership

Accelerate release cycles
Requires strong discipline and team coordination
Needs reliable CI/CD and automation tooling
CI + Trunk practices common in OSS
Use of OSS-style contribution workflows

Source ID: Why Google Stores Billions of Lines of Code in a Single Repository
All Coded Practices:
Use a single shared trunk
Automate test execution (unit, integration, e2e)
Promote shared or collective code ownership
Encourage team collaboration and communication
Require code reviews before merges
Cloud-hosted or scalable infrastructure tools used
Use of custom or internal tooling
Use of CI/CD tools and pipelines
Leverage automation and CI/CD for reliability
Needs reliable CI/CD and automation tooling

Source ID: Software Engineering at Google
All Coded Practices:
Use feature toggles to decouple release and deployment
Promote shared or collective code ownership
Require code reviews before merges
Use of custom or internal tooling
Use of CI/CD tools and pipelines
Tooling includes quality and monitoring support
Promote frequent and rapid integration
Reduce merge conflicts
Needs reliable CI/CD and automation tooling
CI + Trunk practices common in OSS
Use of OSS-style contribution workflows

Source ID: Optimizing Branching Strategies in Mono- and Multi-Repository Environments: A Comprehensive Analysis
All Coded Practices:
Avoid long-lived branches
Use CI/CD pipelines for build, test, and deployment
Small team

Encourage team collaboration and communication
Trunk-based more common in mono-repo structure
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Simplicity
Fast delivery
Requires strong discipline and team coordination
CI + Trunk practices common in OSS
Trunk-based preferred in small OSS projects

Source ID: CI/CD Configuration Practices in Open-Source Android Apps: An Empirical Study
All Coded Practices:
Integrate CI/CD tools (e.g., Jenkins, GitHub Actions)
Use CI/CD pipelines for build, test, and deployment
Tag releases for versioning and tracking
Automate code quality checks (linting, testing, analysis)
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Use of CI/CD tools and pipelines
Leverage automation and CI/CD for reliability
Adoption of practices from known OSS projects
Use of OSS-style contribution workflows

Source ID: Open Source-Style Collaborative Development Practices in Commercial Projects Using GitHub
All Coded Practices:
Use short-lived branches
Rebase before merge
Use CI/CD pipelines for build, test, and deployment
Use temporary release branches when needed
Tag releases for versioning and tracking
Cherry-pick commits to release branches
Promote shared or collective code ownership
Require code reviews before merges
Trunk-based more common in mono-repo structure
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Use of CI/CD tools and pipelines
Promote frequent and rapid integration
Reduce merge conflicts
Enable fast and continuous feedback
Requires strong discipline and team coordination
Feature flag complexity and debt management required

Needs reliable CI/CD and automation tooling
CI + Trunk practices common in OSS
Use of OSS-style contribution workflows

Source ID: How the Adoption of Feature Toggles Correlates with Branch Merges and Defects in OSS Projects
All Coded Practices:
Merge changes quickly
Use CI/CD pipelines for build, test, and deployment
Use feature toggles to decouple release and deployment
Enable frequent and fast releases
Encourage team collaboration and communication
Use of custom or internal tooling
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Feature flag complexity and debt management required
CI + Trunk practices common in OSS
Use of OSS-style contribution workflows

Source ID: Predicting Code Merge Conflicts in Large-Scale Projects: An Empirical Study
All Coded Practices:
Merge changes quickly
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Tooling includes quality and monitoring support
Leverage automation and CI/CD for reliability

Source ID: Monorepos: A Multivocal Literature Review
All Coded Practices:
Encourage team collaboration and communication
Leverage automation and CI/CD for reliability
Challenging in legacy systems or during migration
CI + Trunk practices common in OSS
Use of OSS-style contribution workflows

Source ID: Open-Source Projects and their Collaborative Development Workflows
All Coded Practices:
Merge changes quickly
Cherry-pick changes to release branches
Use temporary release branches

Support Continuous Integration
Commit frequently (daily)
Integrate CI/CD tools (e.g., Jenkins, GitHub Actions)
Use CI/CD pipelines for build, test, and deployment
Use temporary release branches when needed
Cherry-pick commits to release branches
Require code reviews before merges
Use of custom or internal tooling
Use of popular Git-based platforms (GitHub, GitLab, Bitbucket)
Use of CI/CD tools and pipelines
Maintain traceability and history
Needs reliable CI/CD and automation tooling
Use of mailing lists for coordination
Use of OSS-style contribution workflows
Emphasis on shared code ownership

Source ID: Putting it All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project

All Coded Practices:
Commit directly to trunk
Avoid long-lived branches
Merge changes quickly
Deploy frequently and provide fast feedback
Promote shared or collective code ownership
Require code reviews before merges
Cloud-hosted or scalable infrastructure tools used
Use of custom or internal tooling
Promote frequent and rapid integration
Enable fast and continuous feedback
Requires strong discipline and team coordination
Needs reliable CI/CD and automation tooling
Use of mailing lists for coordination
Use of OSS-style contribution workflows
Emphasis on shared code ownership