

DevOps | Security Controls, Shared Repos

Austen Rhyce Erickson
CSD380:Mod 11
2/25/25



DevOps | Security Controls, Shared Repos

What are we talking about?

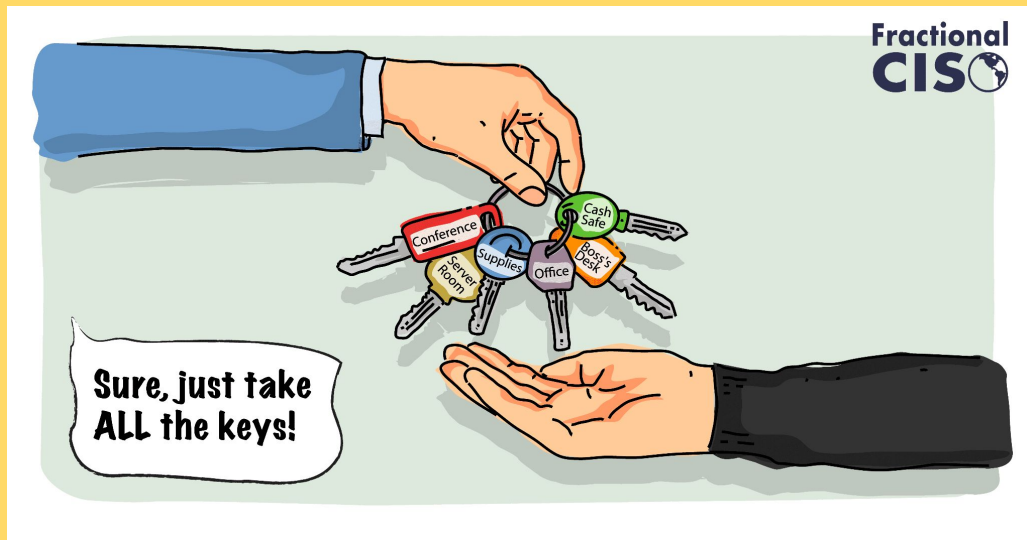
In shared code repositories involving multiple users, sensitive data must be protected, code integrity must be maintained, and unauthorized access must be prevented. Security controls help us do this.



DevOps | Security Controls, Shared Repos

Access Control & Least Privilege

- Use role-based access control to limit permissions based on user roles
- Enforce least privilege so users have only the access they need, no more
- Use multi-factor authentication to secure repository access



DevOps | Security Controls, Shared Repos

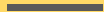
Code Reviews & Pull Request Policies

- Enforce peer code reviews
- Use protected branches
- Implement branch protection rules
 - Require pull request approvals



**GitHub Branch
Protection**

DevOps | Security Controls, Shared Repos



Set GitHub Branch Protection Rules

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Autolink references

Protect matching branches

☒ Require a pull request before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ Require approvals

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1

☐ Dismiss stale pull request approvals when new commits are pushed

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☐ Require review from Code Owners

Require an approved review in pull requests including files with a designated code owner.

☐ Require approval of the most recent reviewable push

Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☐ Require status checks to pass before merging

Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☐ Require conversation resolution before merging

When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more.](#)

☒ Require signed commits

Commits pushed to matching branches must have verified signatures.

☐ Require linear history

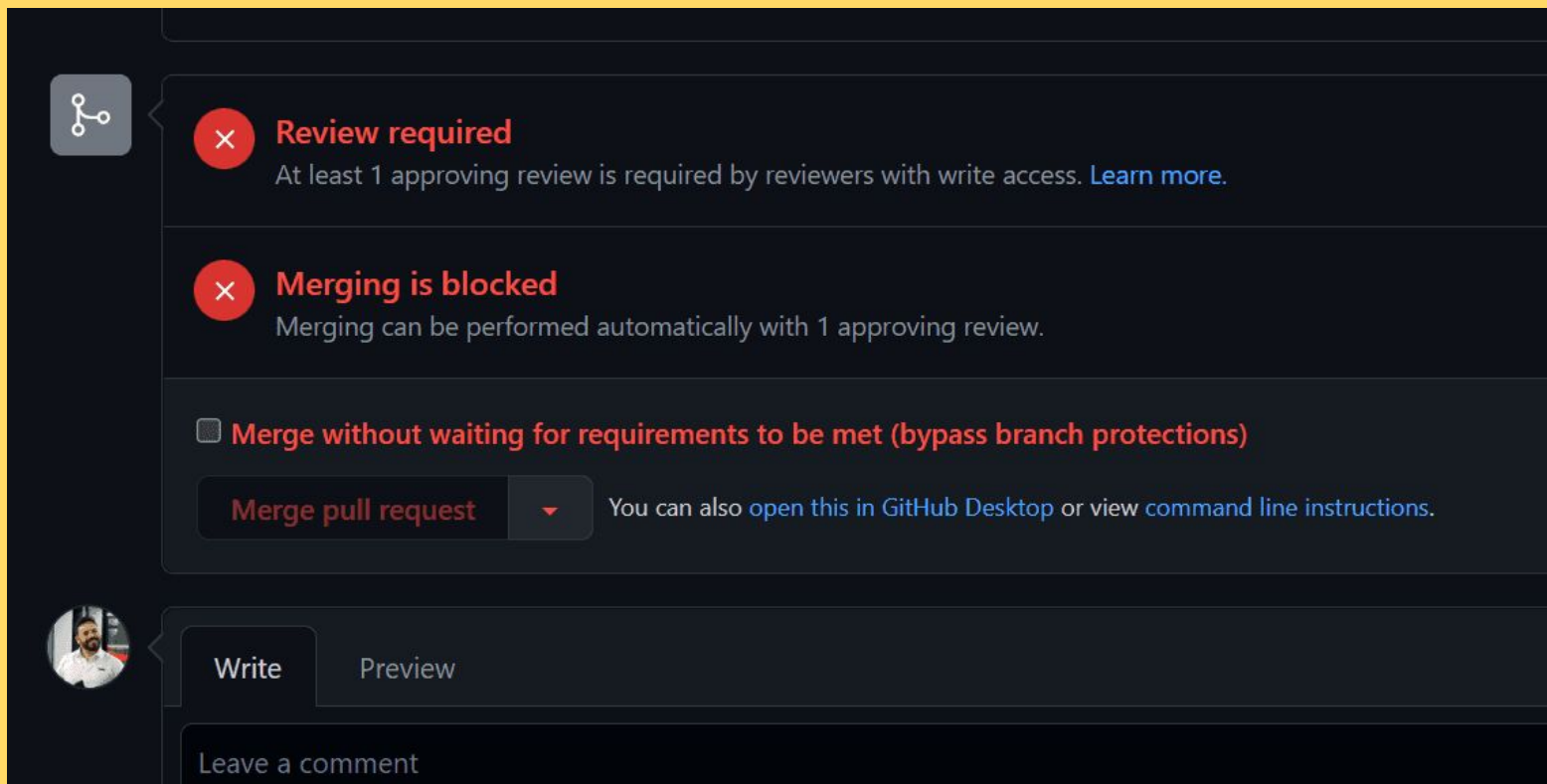
Prevent merge commits from being pushed to matching branches.

☐ Require deployments to succeed before merging


Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.


DevOps | Security Controls, Shared Repos

Encountering Branch Protections




The screenshot displays a GitHub pull request interface with a dark theme. On the left, a sidebar shows a repository icon and a user profile picture. The main content area contains two error messages, each preceded by a red circle with a white 'x' icon. The first message, 'Review required', states that at least one approving review is needed by reviewers with write access, with a 'Learn more' link. The second message, 'Merging is blocked', states that merging can be performed automatically with one approving review. Below these messages is a checkbox labeled 'Merge without waiting for requirements to be met (bypass branch protections)'. At the bottom, there is a 'Merge pull request' button with a dropdown arrow, followed by a text link: 'You can also open this in GitHub Desktop or view command line instructions.' At the very bottom, there is a comment section with 'Write' and 'Preview' tabs and a text input field labeled 'Leave a comment'.

 **Review required**
At least 1 approving review is required by reviewers with write access. [Learn more.](#)

 **Merging is blocked**
Merging can be performed automatically with 1 approving review.

☐ **Merge without waiting for requirements to be met (bypass branch protections)**

Merge pull request ▼ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

 **Write** Preview

Leave a comment

DevOps | Security Controls, Shared Repos

Secrets Management

- Don't store API keys, passwords, or other secrets in the repository
 - Use environment variables
 - Use secret management tools
- Use a secret scanning tool to detect accidental exposure of credentials
- Rotate secrets
- Revoke exposed secrets



HashiCorp

Vault

DevOps | Security Controls, Shared Repos

Secure Repo Configuration

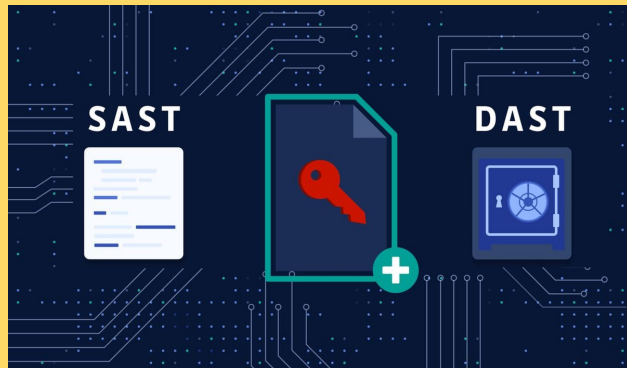
- Use branch protection rules
 - Disable force pushes
 - Disable direct commits to main branches
- Sign your commits with GPG or SSH keys to verify authenticity of the developer
 - Enforce trusted commit authors



DevOps | Security Controls, Shared Repos

Continuous Security Scanning

- Static Application Security Testing (SAST) can/should be integrated into CI/CD pipelines
- Use Software Composition Analysis (SCA) to identify vulnerable dependencies
- Infrastructure as Code (IaC) scanning tools can identify misconfigurations



Dependabot

DevOps | Security Controls, Shared Repos

Logging, Monitoring, Auditing

- Hosting platforms like GitHub have audit logging to track changes and detect anomalies – use it!
- Use Security Information & Event Management (SIEM) for real-time monitoring and alerting
- Don't assume adherence to security policies, conduct regular security audits and compliance checks



DevOps | Security Controls, Shared Repos

Secure CI/CD Pipelines

- Build environments should be isolated to prevent injection attacks
- Use signed artifacts and immutable builds to prevent supply chain attacks
- Utilize vaults/encrypted storage for CI/CD access secrets



DevOps | Security Controls, Shared Repos

Incident Response & Backup

- There should be a incident response plan for breaches/unauthorized access
- Use automated backups of repos, store the backups securely
- Leverage version control history to revert changes



DevOps | Security Controls, Shared Repos

Conclusion

- Automate the security controls
- Educate developers on security
- Regularly perform security testing
- Repositories should be private by default



When security controls and repo management best practices are done correctly, source code can be successfully safeguarded while code collaboration takes place

DevOps | Security Controls, Shared Repos

Sources

- “Protect Your Code Repository.” *Www.ncsc.gov.uk*, www.ncsc.gov.uk/collection/developers-collection/principles/protect-your-code-repository.
- Tal, Liran. “Securing Source Code in Repositories Is Essential: How to Get Started.” *Snyk*, 2023, snyk.io/articles/securing-source-code-repositories/.
- “Source Code Security Best Practices: A Complete Guide - Blog.” *Get.assembla.com*, 7 Aug. 2023, get.assembla.com/blog/source-code-security/.
- “Code Protection: How to Protect Your Source Code.” *Digitalguardian.com*, 2024, www.digitalguardian.com/blog/code-protection-how-protect-your-source-code.
- “Top 10 Source Code Security Best Practices in 2025.” *AIMultiple*, 2025, research.aimultiple.com/source-code-security/.