How to Set Up Commit Signing

GPG, SSH, and Sigstore configuration guide

Option A: GPG Signing

Traditional approach, widely supported:

- Generate GPG key: gpg --full-generate-key (RSA 4096 or Ed25519)
- Configure Git: git config --global user.signingkey
- Enable signing: git config --global commit.gpgsign true
- Upload public key to GitHub/GitLab settings
- Verify: git log --show-signature shows 'Good signature'

Option B: SSH Signing (Recommended)

Simpler setup, uses your existing SSH key:

- Configure Git: git config --global gpg.format ssh
- Set signing key: git config --global user.signingkey ~/.ssh/id_ed25519.pub
- Enable signing: git config --global commit.gpgsign true
- Upload SSH signing key to GitHub/GitLab (separate from auth key)

Option C: Sigstore/Keyless Signing

Best for CI/CD — no key management required:

- Install gitsign: brew install sigstore/tap/gitsign
- Configure: git config --global commit.gpgsign true && git config --global gpg.x509.program gitsign && git config --global gpg.format x509
- Signs using your OIDC identity (GitHub, Google, Microsoft)
- Verification via Rekor transparency log

Enforce in CI

Configure branch protection rules to require signed commits. This ensures all code in protected branches has verified authorship.