Adopting GitHub in Enterprise World

by Max Yermakhanov

automagically

Max Yermakhanov

Senior Consultant

Microsoft Azure DevOps MVP

GitHub FastTrack Partner

Email: max@automagically.io
max@objectsharp.com

Blog: medium.com/@yermax

Twitter: @yermax

automagically

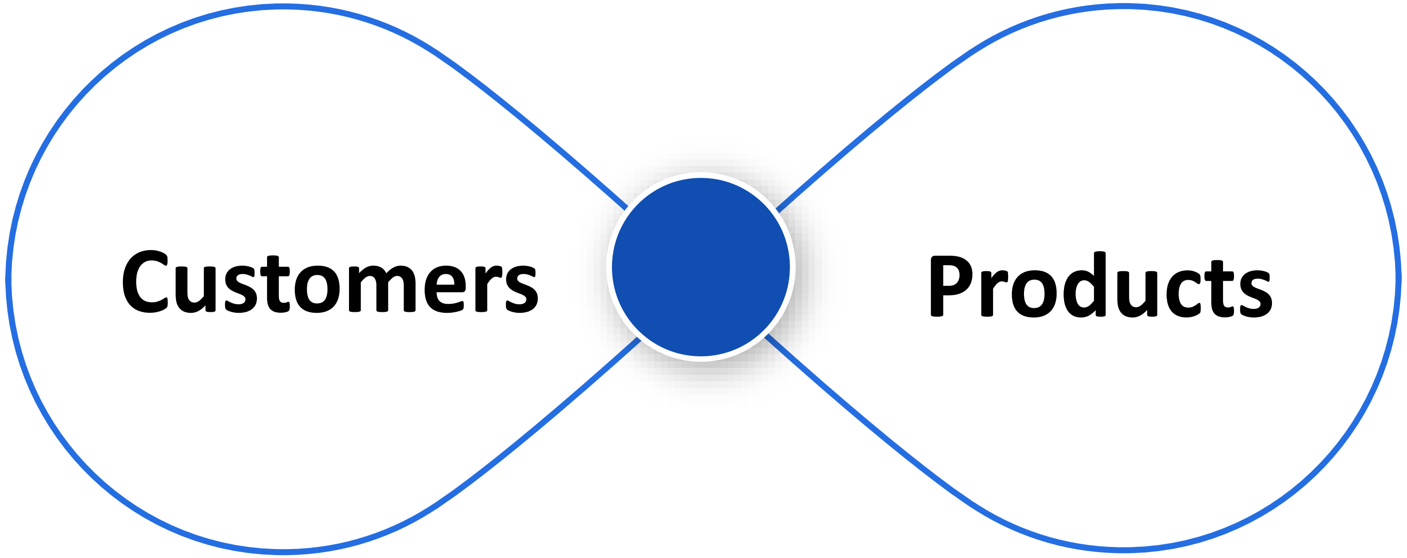
Developer experience matters!

- **Fully integrated platform** from idea-to-production with **end-to-end traceability**
- **Simple management** with one platform to onboard, one set of policies, and one self-contained security model
- **Collaborative, automated** workflows
- **Seamless access** to open source and innersource
- **Developer focused** security and compliance

Happier customers

Happier developers

Faster feedback



Customers

Products

Ship faster

- faster deployments
- shorter lead times
- more features customers care about

Better experiences

higher customer ratings

Ship more securely

- higher software quality
- fewer change failures
- faster remediation times

Overview

- Enterprise Account(s)
  - Security
  - Enterprise Policies
  - Billing and usage
- Organization(s)
  - Organization policies
- Repositories
- Teams

Enterprise Managed Users

- AAD or Okta (7k integrations)
  - mass onboarding, one-click removal
- SAML: Single Sign-On, Multi-factor authentication
- Team membership provisioning & automatic user removal with SCIM
- Company-owned accounts
  - managed user identified
  - user audit trail
- Private repositories only
  - reduce IP leakage
- Read-only access to OSS/GitHub.com

Access Control - Roles

- Default access roles:
  - Read
  - Write
  - Triage
  - Maintain
  - Admin
- You can create custom roles as well

One organization or multiple?

One GitHub organization

- High level of collaboration is required between business units
- Low administrative overhead

Multiple GitHub organizations

- High level of separation is required between business units
- High(er) administrative overhead

Pro Tip: Less is more

automagically

Collaboration and Planning

- Keep repositories as open as possible within the company, and encourage collaboration
  - Create internal communities & celebrate wins
  - Encourage Innersourcing by tagging repos to indicate reusability
  - Contribute ideas & content, not just code
  - Use “needs help” tag & encourage peer learning
  - Define contribution policies

Collaboration and Planning (continued)

- Train developers to look for existing code before writing something new
- Protected Branches ensure collaborators on your repo can’t make irrevocable changes
  - Code review approval
  - Required Status Checks
  - Enforce signed commits
  - Include administrators

Collaboration and Planning (continued)

- Pull Request early & often (keep them small)
  - PR early, PR often, and keep them small
  - Not every pull request has to be merged!
  - Use Draft PRs
  - Use Pull Request templates

- Innersourcing goes beyond raw code sharing
  - Use private packages to reuse code as versioned dependencies
  - Config-as-code = collaborate on env setup, workflows, & security

- Automate, automate, then automate some more!

Collaboration across tooling

- **GitHub Projects, JIRA, Azure Boards**
  - Deep-linking from work items to code
  - State synchronization between code & issues
  - Board state display (badges) in GitHub
- **Microsoft Teams, Slack**
  - View repository, PRs, & issues in Teams tab
  - Search repos, issue commands, get notifications
  - In-context conversations & holistic personal views
- **GitHub for Mobile**
  - Collaborate on the go! Edit issues & pull requests; search for users/repos/orgs; comment, react, and merge code in a portable mobile-optimized interface.

CICD and Automation

- Ensure safe use of public GitHub Actions
  - GitHub verification badge on public actions is not enough
  - Limit what actions can be used by your organization(s)
  - Create internal GitHub Actions catalog
  - Create separate GitHub organization to test actions
  - Review the source code and trust the publisher / action
  - Fork public GitHub Action repositories and take control

CICD and Automation (continued)

- Ensure safe use of public GitHub Actions (continued)
  - Use SHA hashes for public GitHub Actions, if needed
  - Set default GITHUB_TOKEN permissions to read
  - Take advantage of Dependabot for actions
- Store sensitive data as secrets in GitHub or external key vault
- Promote workflow best practices using reusable workflows and starter workflows

CICD and Automation (continued)

- Use GitHub Apps to improve your workflows
- Use private runners
  - Secure access to private runners
  - Implement ephemeral private runners hosted in a k8s cluster
  - Do not use private runners for public repos
  - Be aware of dangers of untrusted input and incoming PRs from the forks

Security & Compliance

- Access, Policies & Compliance
  - SSO Access Controls
  - Real-time inventory of dependency insights
  - License compliance
  - Policy management
  - Private Secret Scanning

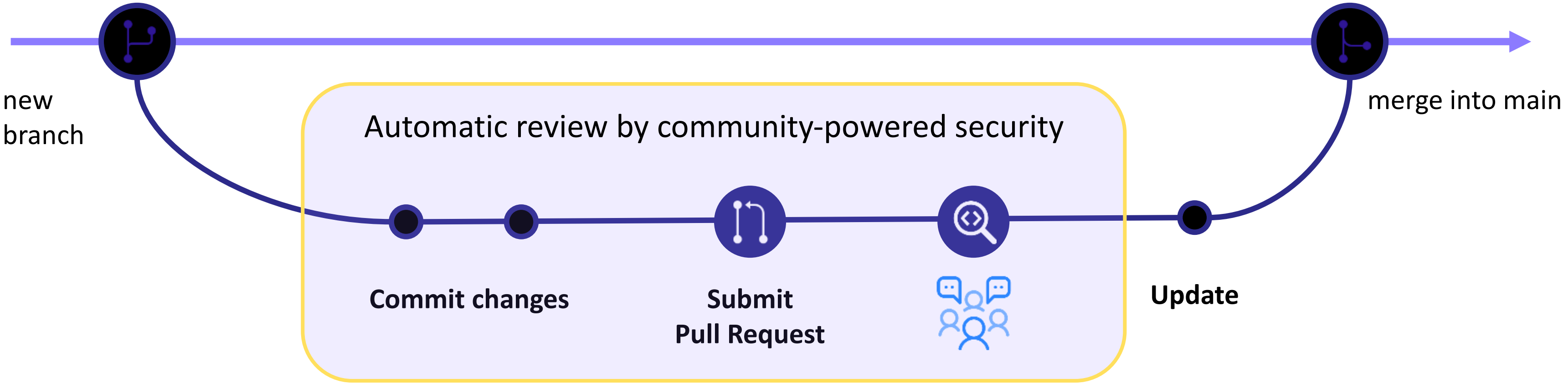
Security & Compliance (continued)

- Vulnerability Management
  - Dependency scanning
  - Largest vulnerability database
  - Automated security updates

Security & Compliance (continued)

- Advanced Security
    - Advanced code analysis
    - Vulnerability hunting tool
    - Community of top security experts
    - Private Secret Scanning

Integrated Security Analysis



automagically

Administration and Maintenance

- Take advantage of team synchronization with AAD
- Enable MFA, of course
- Tune GitHub notification settings
- Configure GitHub policies at various levels
- Audit log keeps track of changes in GitHub
- GitHub App is your best friend. Also, so is GitHub CLI
- Look into adopting Codespaces. Seriously, Codespaces are great!

Moving to GitHub?

- **Self-migrate** from any git-based solution via web-based wizard or command line/scripting
- **Advanced migration** using GitHub Enterprise Importer (GEI, formerly Octoshift)
- **Professional Services** available for managed migrations

Thank you