VA GitHub Technology Standards

1. Introduction

This Enterprise Technology Guideline (ETG) establishes the official technology standard for using GitHub in VA. This standard applies to all new projects that are using, or planning to use, GitHub Enterprise Cloud (GHEC). The current direction is to use GitHub Enterprise Cloud – US (GHEC-US), a VA-managed, FedRAMP-aligned environment that ensures stronger protections and centralized access controls

Legacy on-premises repositories follow configuration management guidance provided by Digital Services Delivery (DSD), while all other repositories follow the official GitHub guidance contained in the GitHub Handbook.

2. OIT Policy and Technical Direction

All new software development projects are required to use an enterprise instance of GitHub, through the externally hosted GitHub.com. GitHub is required for all future products either in development or in sustainment, and it will be applied to testing, pre-production, and production environments. Other Git-based tools, such as GitLab and Bitbucket, are still currently approved in the Technical Reference Model (TRM) for sustainment projects. Future updates of this document will reflect all policy changes related to the use of these tools for managing source code repositories.

3. Guidelines for GitHub Users

The current handbook for the GHEC is at VA GitHub Handbook | Welcome (department-of-veterans-affairs.github.io).

The latest developer guidance for GitHub is located at the Catalog of Developer Essentials (CODE VA): GitHub | Docs | CODE VA.

Existing repositories using GHES (Github.ec.va.gov) follow the configuration management guidance provided by DSD: VA OIT SPM Configuration Management Department SharePoint site - Configuration Management at VA

More information about the current available GitHub instances is at 2.0 General GitHub Information (sharepoint.com).

Each resource provides onboarding guidance, contact information for GitHub issues, and links to collaboration channels, including Slack channels, for customer support. This section will be kept

up-to-date as these resources change, but detailed information will be updated in real-time on these sites.

4. Available GitHub Tools

The following tools are available to project teams that have existing repositories in VA GitHub.com or will be using GitHub.com in future projects. This section will be updated with more tools as they become available enterprise-wide, including Codespaces and Copilot, along with technical details to support consistent application.

- Codacy – a code quality tool that proactively monitors for code changes that introduce security or code quality concerns. This tool is maintained by the DevSecOps Tool Suite (DOTS) and support is provided by DOTS.
- CodeQL – GitHub's next-generation high-fidelity Static Application Security Testing (SAST) tool used to find vulnerabilities earlier in the development process than traditional security tools. This tool is maintained by GitHub and support is provided by GitHub Enterprise Support and the GitHub Expert Services team.
    - Getting Started: https://codeql.github.com/docs/
    - Example: https://github.com/department-of-veterans-affairs/vets-website/blob/main/.github/workflows/codeql-analysis.yml
- GitHub Advanced Security (GHAS) – GitHub's suite of security tooling encompassing CodeQL for SAST scanning, Dependabot for identifying and remediation vulnerable dependencies, and Secret Scanning which GitHub's state-of-the-art platform for identifying and validating leaked secrets in partnership with over 200 leading SaaS providers. This tool is maintained by GitHub and support is provided by GitHub Enterprise Support and the GitHub Expert Services team.
    - Dependabot Getting Started: https://docs.github.com/en/code-security/dependabot
    - Secret Scanning Getting Started: https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning
- Probot Settings – A Configuration-as-Code tool for defining repository settings in code, such that admins cannot mistakenly or intentionally update repository settings, such as modify branch protection rules, without consulting with the wider admin team. This tool is maintained by GitHub and support is provided by GitHub Enterprise Support and the GitHub Expert Services team.
    - Getting Started: https://github.com/repository-settings/app#usage
    - Example: https://github.com/department-of-veterans-affairs/.github/blob/master/.github/settings.yml

- Jira Smart Commits – A tool that allows developers to auto-link GitHub commits, branches, and pull requests against this Jira tickets. This tool is maintained by Atlassian, and support is provided on best effort by the GitHub Expert Services team, DOTS, and then ultimately Atlassian.
    - Getting Started: https://support.atlassian.com/jira-software-cloud/docs/process-issues-with-smart-commits/
- Copilot – Copilot is the AI-powered coding companion, designed to assist by suggesting entire lines or blocks of code as you type. This can help you code faster, reduce errors, and learn new coding techniques along the way. You can request access using this link. When requesting access, be sure to be signed in to GitHub and part of the Department of Veterans Affairs GitHub Organization. Please note that accounts are automatically removed after 30 days of inactivity to optimize license usage. More information on how activity is determined can be found here. Check out the CAIO GitHub Copilot SharePoint Page for more information and helpful resources, including recorded training webinars on fundamental and intermediate use of GitHub Copilot. If you have any questions or need further assistance, please reach out via the GitHub Copilot Teams Channel or email OITaicodedev@va.gov.

5. Configuration Management Guidelines

Project teams follow the below guidelines to ensure they are doing configuration management with GitHub properly.

- For repositories hosted in VA Github.com, activate GitHub Advanced Security (GHAS) and apply it to all changes to the repository.
- Determine the branching approach (as defined in the README.md for existing repositories) before beginning development.
- Document the branching approach in an easy-to-read format for the team in the CONTRIBUTING.md in the root of your repository.
- Consult with the GitHub Expert Services team if the repository must have public visibility to understand the responsibilities and to have the repository made public.
- If necessary, apply a commonsense review process implementing the GitHub CODEOWNERS pattern to ensure appropriate teams are reviewing code they are responsible for maintaining.
- When uploading code to GitHub, the .env file must be added to the. gitignore file. This way, the environment variables file will not be committed to GitHub.
- When using API keys, developers must also set restrictions on the key, limiting access.
- Ensure that sensitive information should never be in config files that are visible to all users, even users with read only access to the repository.
- Document a plan for responding to compromised secrets and practice that plan yearly.

- Dynamic secrets with limited scope must be used for development whenever possible.
- Developers use the following approaches to avoid committing sensitive information:
    - Avoid the catch-all commands *git add* . and *git commit -a* on the command line—use *git add filename* and *git rm filename* to individually stage files, instead.
    - Use *git add* --interactive to individually review and stage changes within each file.
    - Use *git diff* --cached to review the changes that are staged for commit. This is the exact *diff* that *git commit* will produce as long as the *-a flag* is not used.
    - Consider using a visual program like GitHub Desktop or gitk to commit changes. Visual programs generally make it easier to see exactly which files will be added, deleted and modified with each commit.
- For Image Repositories:
    - When authentication is necessary during the container image build process, use an authentication management service to store secrets.
        - VA Public Key Infrastructure (PKI) should be used for generating, managing and securing credentials.
    - Use a secrets storage service such as AWS Secrets Manager, Azure Key Vault, or CyberArk per the Software Factory Utility for Secrets Management.
    - When using Elastic CI Stack for AWS, place secrets inside the stack's encrypted S3 bucket.
    - Buildkit with a flag called –secret safely provides a secret to Dockerfiles (for container images using Docker, as an example) at build time.
    - Every new image must be scanned for exposed and unencrypted secrets (especially private keys) initially; then scanned again for every new image version.
    - Scan the registry for secrets daily and implement corrective methods immediately.
    - Scanning tools such as GitGuardian Shield *ggshield*, and Trivy, can be used to scan base images for secrets (Note – these tools are subject to decisions rendered in the VA Technical Reference Model (TRM)).
    - Aqua scans for embedded secrets, scans for vulnerabilities in container images based on a constantly updated stream of aggregate sources (common vulnerability exposures (CVE), vendor advisories, and proprietary research).

- Aqua scans are used to find malware, open-source software licenses, and configuration issues in images to further reduce the attack surface. This scan is done in coordination with the other required scans to maintain an Authority to Operate (ATO).
- Secrets must be removed, moved or encrypted but only after the keys and tokens have been replaced. *Exposure of the secret should be assumed.*

6. Repository Standards

Repository standards focus on the full code repository lifecycle to ensure enough knowledge is captured to streamline external use of VA custom code.

- Publish repositories standards at the top level of GitHub – include guidance from Director of the Office and Management and Budget

Structure and Content - Each repository must include comprehensive documentation of the custom-developed code, configuration and artifacts required to develop, build, test, and deploy the custom-developed code.

- README.md: Overview of the project purpose, installation instructions, usage examples, and contact information.
- LICENSE.md: The license under which the code is released, ensuring legal clarity.
- API Documentation: Swagger output which provides documentation on APIs provided by the code, including endpoints, parameters, and example requests/responses.
- Setup.md: When feasible, a script of the installation instructions from README.md
- va.code.gov.md: relevant metadata is captured in this file that will be used to create the aggregate code inventory for VA found at www.va.gov/code.json.

Version Control and Change Management- VA must ensure that custom-developed code follows best practices for operating repositories and version control systems to keep track of changes and to facilitate collaboration among multiple developer.

- Use Git for version control.
- Adopt a branching strategy to manage development and releases.
- Document changes in each release, including new features, bug fixes, and known issues.
- Follow a consistent format for commit messages to provide clear context for changes.

Access and Permissions - VA must publish custom-developed software to a public repository or a private listing that can be accessed by other federal agencies. Once published, VA should use a role-based approach to determine who access, modify, and manage the repository. For code published to a private listing, controlled access to private repositories ensures that only authorized users can view and contribute to the code.

- For code stored in public repositories, the public will be free to access that source code
- For code stored in private listing, non-VA Federal employees gain access using the same process in place for VA GitHub intranet
- Log files should be setup to support metrics: code reuse frequency, access and modification
- Use secure authentication methods (e.g., OAuth, SSO) to verify user identities

Document Version Control

Date	Version	Change Details	By
2/13/2023	0.1	Original Draft	OCTO/TAS
5/10/2023	1.0	Final draft addressing feedback from SPM and OCTO	OCTO/TAS
12/14/2023	1.5	Updated version reflecting changes to status of Copilot and Codespaces and enterprise standards.	OCTO/TAS
12/12/2024	2.0	Update for FY25 that includes changes made during the past year.	OCTO/TAS
12/16/2025, 3/16/2026	3.0	Updates reflecting CoPilot availability, new repository standards, and organizational changes, including a hotfix update in March 2026.	OCTO/TAS