# VA GitHub Technology Standard

## 1. Introduction

This document is part of the series of Enterprise Design Patterns (EDP) that establishes the official technology standard for GitHub in VA. As such, many of the links included in this document are only available to VA staff with network access. The standard applies to all new projects that are using, or planning to use, VA GitHub.com. Products in sustainment using VA Enterprise Cloud (VAEC) hosted repositories (Github.ec.va.gov) also follow the configuration management guidance provided by Software Product Management (SPM).

## 2. OIT Policy and Technical Direction

All new projects are able to use an enterprise instance of GitHub, either via the enterprise hosted version (GitHub.ec.va.gov) or through the externally hosted GitHub.com. Existing non-GitHub repositories will soon be required to use GitHub as well to ensure consistent use throughout VA. 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). Future updates of this document will reflect all policy changes related to the use of these tools for managing source code repositories.

## 3. Guidance for GitHub Users

The current handbook for VA GitHub.com, supported by the DevSecOps Tools Suite (DOTS) team in the Office of the Chief Technology Officer (OCTO), is at VA GitHub Handbook | Welcome (department-of-veterans-affairs.github.io).

Existing repositories using Github.ec.va.gov (VAEC-hosted instance of GitHub) follow the configuration management guidance provided by SPM:

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 expanded in future updates with more information on templates and links to additional implementation guidance as it becomes available.

## 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, 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
  - CodeQL Getting Started: https://docs.github.com/en/codespaces/guides
» 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/

## 5. Configuration Management Guidelines

Project teams follow the below guidelines to ensure they are doing configuration management with GitHub properly. Refer to the GitHub.com and Github.ec.va.gov handbooks listed in Section 2 for more detailed information. Section 6 contains supplemental technical references.

» 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, HashiCorp Vault, Azure Key Vault, or CyberArk per the Secrets Management Enterprise Design Pattern.
  - 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. Supplemental Technical References

» VA Enterprise Design Patterns (EDP) - library of approved EDPs including this document
» VA Technical Reference Model (TRM) – list of approved software at the version level
» Process Asset Library (PAL) for Configuration Management: Display Framework ->SDPM (sharepoint.com) – process required for all GitHub users.
» VA Software Factory Security Pattern Handbook - provides security guidelines and specify minimum security requirements for software factory implementations at VA.
» Tech Tuesday: Secrets Management & GitHub Advanced Security Demo: Provides a demo for securely and efficiently managing the creation, rotation, revocation, and storage of digital authorization credentials (January 2023).
» FedRAMP Vulnerability Scanning Requirements for Containers Handbook - bridges the vulnerability scanning compliance gaps between traditional cloud systems and containerized cloud systems.
» OIS Authorization Requirements Standard Operating Procedures (SOP) - details the technical scans/testing and security documentation requirements for each boundary https://dvagov.sharepoint.com/sites/OITOIS/KnowledgeService/eMassDocumentLibrary/eMASS_Authorization_Requirements_SOP_Guide.pdf.
» https://department-of-veterans-affairs.github.io/ois-swa-wiki/docs/devsecops/
- provides guidance on applying Fortify concerns in DevSecOps environment.

## 7. Document Version Control

| Date | Version | Change Details | By |
|---|---|---|---|
| 2/13/2023 | 0.1 | Original Draft | Mike Dance |
| 2/27/2023 | 0.1 | Added Secrets Management | Bridget Stiggers |
| 2/27/2023 | 0.4 | Updated Configuration Management Checklist | Bridget Stiggers |
| 3/24/2023 | 0.5 | Updated version with edits from SPM, OIS, and GitHub | Mike Dance |
| 4/28/2023 | 0.8 | Further refinement of scope and improvements to guidance links based on SPM feedback | Mike Dance |
| 5/10/2023 | 1.0 | Final draft addressing feedback from SPM and OCTO | Mike Dance |
| 12/14/2023 | 1.5 | Updated version reflecting changes to status of Copilot and Codespaces and enterprise standards. | Mike Dance |

**Disclaimer:** This document serves both internal and external customers. Links displayed throughout this document may not be viewable to all users outside the VA domain. This document may also include links to websites outside VA control and jurisdiction. VA is not responsible for the privacy practices or the content of non-VA websites. We encourage you to review the privacy policy or terms and conditions of those sites to fully understand what information is collected and how it is used.