VA API Technology Standards and Tooling Guide

Main References

API Standards for the VA

Lighthouse Delivery Infrastructure (DI) documentation

Continuous Authority to Operate (cATO) playbook

Catalog of VA APIs

External References (for both VA and 3rd Party applications)

- API publishing standards
- API status page
- Continuous ATO (cATO) and SecRel guidelines

The Lighthouse API platform (developer.va.gov) simplifies API discovery and API integration for both VA and 3rd-party application teams while simultaneously reducing the burden on API providers (teams building APIs). Loan Guaranty (LGY) has been quite successful in supporting nearly 50 3rd-party lender applications in production on this with their APIs. Information as to how this is done can be found at:

https://code.va.gov/docs/default/component/va-api-standards/general-guidelines

https://code.va.gov/docs/default/component/lighthouse-guide-for-public-facing-va-apis

https://developer.va.gov/production-access

https://developer.va.gov/api-publishing

The following sections are also located in CODE VA but provided here in document form.

Lighthouse Delivery Infrastructure (LHDI) Tools

The Lighthouse Delivery Infrastructure (LHDI) is a product that supplies operating environments for applications using Amazon-managed Elastic Kubernetes Service (EKS) clusters. As described by NIST, we are a common controls provider, which allows us to manage the security posture of the platform instead of our product teams. And because our platform leverages the VA's cloud environment, we are common controls consumer as well. We recommend these tools as the way to production.

Core Platform Technologies

- **Kubernetes:** EKS (Default namespaces (dev, qa, sandbox, prod))
- **Service Mesh:** Istio (Default ingresses, canary deployments, retries/timeouts)

A **service mesh** is an infrastructure layer that provides capabilities to a collection of distributed services that share a business domain. Key capabilities include traffic management, security, and observability.

From Istio's What is a Service Mesh?:

Modern applications are typically architected as distributed collections of microservices, with each collection of microservices performing some discrete business function. A service mesh is a dedicated infrastructure layer that you can add to your applications. It allows you to transparently add capabilities like observability, traffic management, and security, without adding them to your own code. The term “service mesh” describes both the type of software you use to implement this pattern, and the security or network domain that is created when you use that software.

For LHDI customers, the most basic use case will usually include:

- Creating a virtual service in front of your Kubernetes service, and
- Ensuring that virtual service exposes your application on a gateway appropriate for the environment.

Deploying a virtual service associated with an appropriate gateway should be enough to expose your application to any consumers on the VA network. If you need to publish your application outside of the VA network, please see the documentation for publishing your application on the Lighthouse external platform.

More examples and features will be coming to LHDI in the future, but feel free to explore more advanced capabilities of the Istio service mesh in the Istio documentation on your own in the meantime.

Core Services

- **Code Repository:** GitHub
- **Continuous Integration:** Github Actions
- **Continuous Deployment Server:** Argo
- **Custom LHDI pipeline integrations:** via Github Actions
- **General Artifact Store:** GitHub Packages
- **Release Notes:** Github Releases

- **Code Analysis Reporting:** CodeClimate
- **Logging and Monitoring:** Datadog
- **Secrets Management:** Vault
- **AWS Resource Management (S3, RDS):** Amazon Controller for Kubernetes (ACK)
- **PVC Snapshots:** DLM
- **Blue/Green Deployments:** Argo Rollouts

Standard Frameworks for APIs, CLIs

- **API:** OpenAPI (aka Swagger)
- **Tracing:** OpenTracing
- **DB Migration:** Flyway
- **Image Build:** Docker
- **Image Base:** Distroless, Alpine (executor base)
- **Deployment Definition:** Helm
- **Lightkeeper CLI**
    - Lightkeeper is a CLI application used to automatically provision infrastructure within AWS for new and existing applications. Today, its primary use is to generate and obtain a kubeconfig for access to your cluster environments. Down the road, we plan to incorporate additional features of the Lightkeeper CLI for tasks like provisioning and managing secrets, S3 buckets, and RDS instances.
    - Available Lightkeeper commands

Sources for CIS Testing

- **aquasecurity:** kube-bench
- **Docker:** docker-bench
- **dev-sec:** kubernetes, Docker
- **Mitre:** Docker, Kubernetes

Secure Release Pipeline

The secure release pipeline is a continuous integration pipeline that manages and enforces both the secure and release stages of integration and delivery. An application development team's

ability to release code to users in production is gated and controlled, only allowing teams to achieve a signed image for deployment if the code adheres to our secure release policy. It works as a re-usable GitHub Action workflow, which provides:

- Automated Static Application Security Testing (SAST) and Software Composition Analysis (SCA) through Snyk, and
- Container/image scanning using Aqua

By adopting our secure and release states into their pipelines, application teams can deliver products into production quickly and safely. Each time your application teams submit code, they get:

- Immediate feedback on security vulnerabilities for their source code, 3rd party libraries, and images.
- Guidance on combating security weaknesses.
- Access to cybersecurity education resources.
- Information on how to quickly remediate blockers in real-time so that our pipeline can digitally sign their images.

Secure Release Pipeline Tools

The secure release pipeline uses Aqua and Snyk, with cATO vulnerabilities handled through SD Elements. How-to videos for SecRel tools are available on our resources page.

Aqua Scanning

Aqua scans images for operating system vulnerabilities, malware, and insecure configurations and monitors for images that are deployed into production.

Snyk Scanning

Snyk scans developer source code and third-party libraries, or dependencies, for known vulnerabilities. Once you have successfully integrated and run our SecRel pipeline, the results are made available in the GitHub Security Center. To browse vulnerability alerts, select the Security tab in your team's GitHub repository.

Security by Design Elements (SD Elements)

SD Elements supports threat modeling your system, and translating risks into actionable backlog that will address security and privacy requirements.

Roadmap

The following lists represent the next features and capabilities that the LHDI platform team intend to integrate (in order of priority):

High Priority

- **Service Mesh custom ingresses.**
- **Docker Registry:** ECR

Additional Capabilities

- **Ability for API teams to create custom namespaces.**
- **OAuth2/IDP flow**
- **Kubernetes Deployment Definition for system-level resources**
- **CI Monitoring**
- **Feature Toggle**
- **Contract Testing**
- **API Documentation**
- **Policy as Code**
- **Code Analysis Reporting**
- **Extending k8s-api**