Solution Guide

anchore

Shift Security Left with Anchore Enterprise

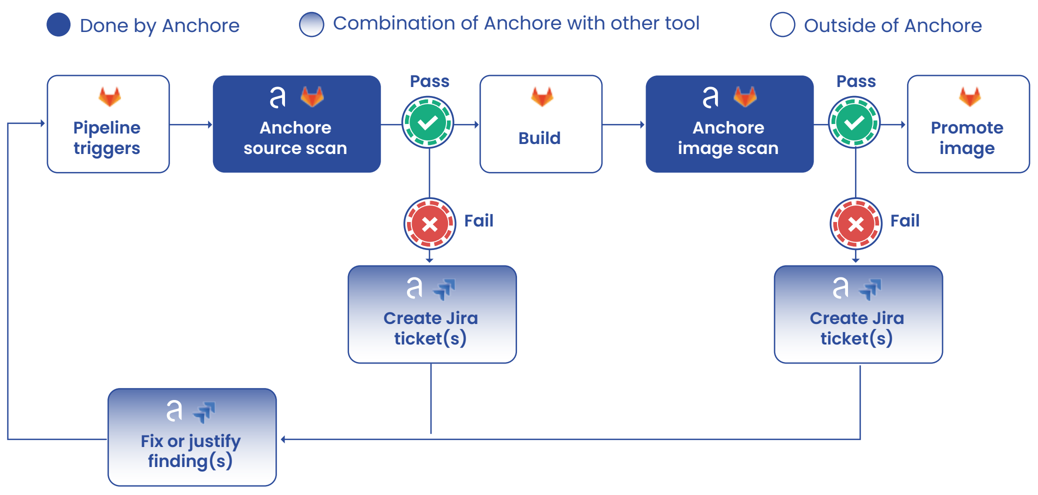
Shift left security is a secure development practice that integrates vulnerability scanning into continuous integration (CI) pipelines. It is also one of the primary use cases Anchore Enterprise is addressing for customers today. Developers gain early insights to potential vulnerabilities or policy infringements then are able to prioritize findings or immediately remediate the issue.

Shifting left offers a number of advantages over traditional security processes, in which security is addressed only after the product has been released:

1. Lower cost of remediation
2. Faster time to market
3. Improved overall security posture
4. Increased user trust

Automate early vulnerability detection and remediation for priority security findings (CISA KEV) with superior developer experience

In this guide we present a battle-tested, shift-left developer workflow with the help of Anchore Enterprise. The workflow infrastructure will include GitLab as the continuous integration (CI) pipeline, Anchore Enterprise as the vulnerability scanner and Jira as the remediation tracking solution—the most commonly used tools by federal customers. The graphic below depicts the implementation of these tools into a complete shift left security solution. Anchore Enterprise is agnostic to CI and remediation tracking tools, this same automation can be achieved with your preferred tooling.



While implementing container security can be seen as difficult, Anchore Enterprise is designed to preserve the developer experience—a modern DevOps experience without the friction or delays of legacy security tooling. Using Anchore as a blocking gate in the CI pipeline is an optional choice. Anchore's policy engine can also be configured to only stop pipelines if there are critical security findings or have actionable remediation recommendations available. Flexible policies allow for security teams to create policy checks that reflect the optimal balance between risk and developer velocity.

» Requirements

To implement the workflow reference architecture above, you will need:

1. A GitLab account
2. AnchoreCTL configured in your GitLab project >> see our docs
3. Access to a Jira project with the ability to create new tickets
4. Access to an Anchore Enterprise deployment with permissions to edit policy

Note: If you don't have an Anchore Enterprise deployment, you can get started in minutes with a free trial on AWS commercial cloud. If you don't have access to an AWS account, contact sales@anchore.com to get a license and support.

- Container registry credentials added to Anchore Enterprise to access images built in CI >> see our docs

Implement an end-to-end, automated DevSecOps + compliance workflow

Below we cover how to work with Anchore policies, how to integrate Anchore into individual CI pipeline stages and how to remediate or justify policy violations.

The resources needed to follow along with this guide are found in this GitHub repository.

Creating the Anchore Developer Policy Bundle

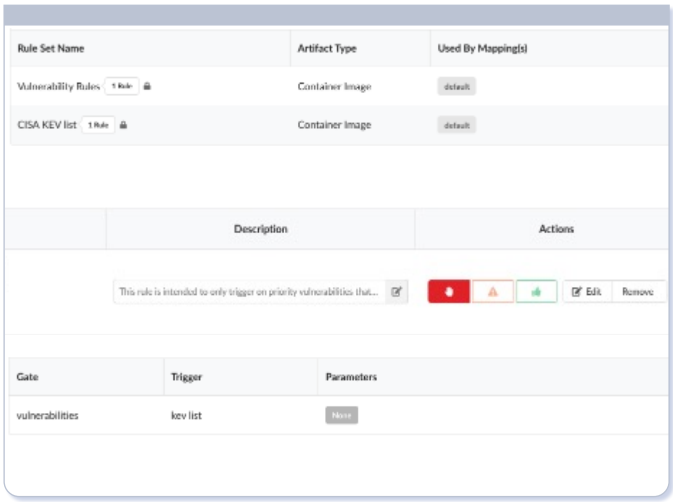
Policy in Anchore is a powerful yet flexible feature. You can use and modify any of Anchore's prebuilt policy bundles (e.g., NIST 800-53, CIS, Iron Bank, FedRAMP) or you can create your own custom policy bundles meeting the unique requirements of your organization.

In this example, we are using a policy bundle called anchore-developer-bundle. This bundle is made up of only two policy rules:

- Trigger on any CISA KEV findings, and
- Trigger if there are critical vulnerabilities in the application (i.e., non-OS) with an available fix.

You can access a working version of this bundle here.

Note: the CISA KEV policy gate was introduced in Anchore Enterprise 5.8. You will need a deployment with that version of higher to continue.



Rule Set Name	Artifact Type	Used By Mapping(s)
Vulnerability Rules 1 Rule	Container Image	default
CISA KEV list 1 Rule	Container Image	default

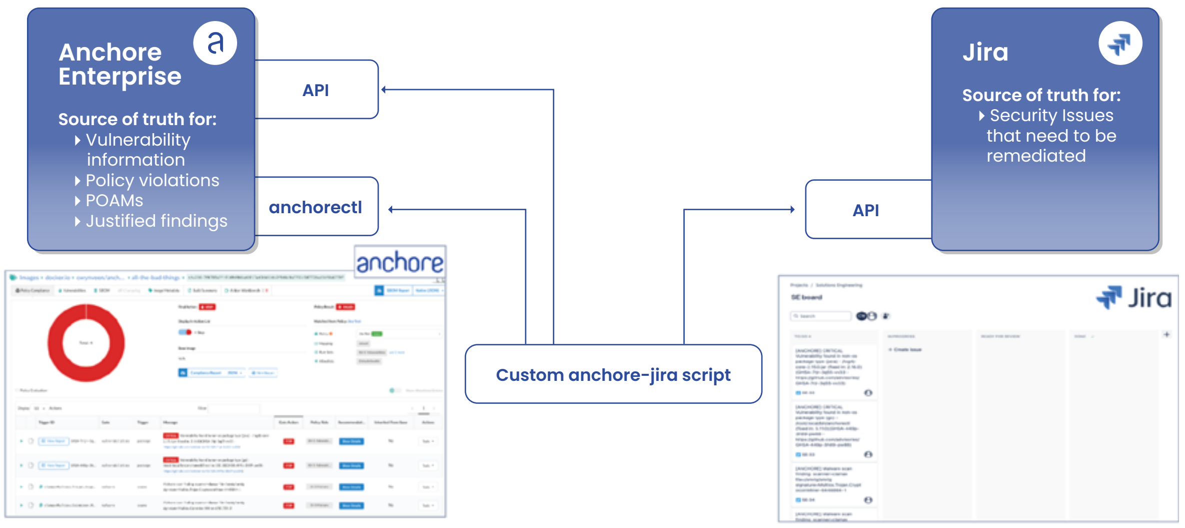
	Description	Actions
	This rule is intended to only trigger on priority vulnerabilities that...	Edit Remove

Gate	Trigger	Parameters
vulnerabilities	kev list	None

Anchore and Jira Integration

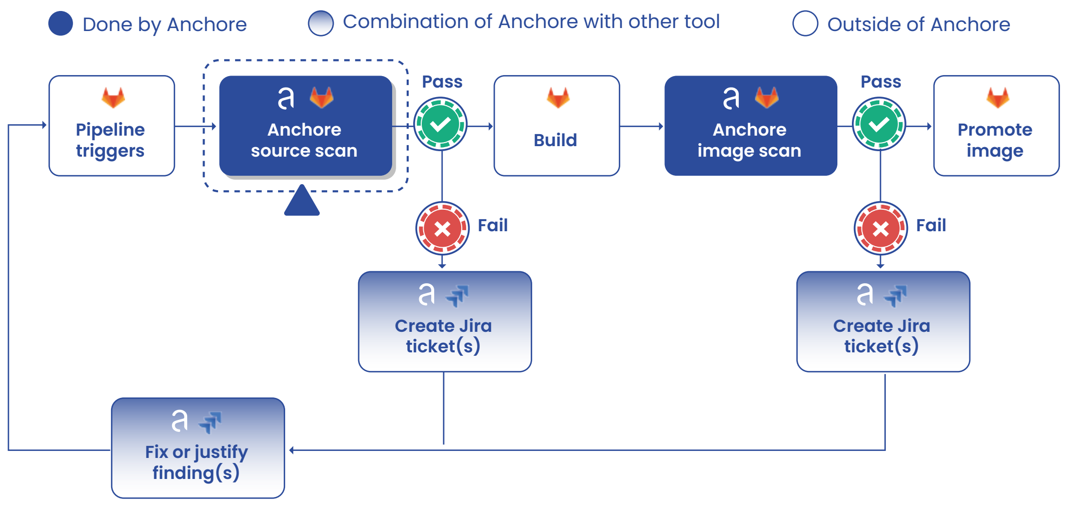
Anchore Enterprise comes with anchore-jira, a lightweight python script that integrates anchorectl and the Jira API, to automate the creation of Jira tickets when policy violations are triggered in the CI pipeline.

First, anchorectl pulls a policy evaluation for the source and image SBOM and creates a Jira ticket for each Stop action in the policy evaluation. You can review the script here and modify or extend it to your needs.



Anchore Source Scan Stage

This stage in the pipeline generates an SBOM from the source code repository used to build the container image. Anchore Enterprise identifies vulnerabilities and other security issues that require remediation. If the pipeline stops here, it is due to a direct dependency. Developers save time by receiving security feedback early in the development process to inform and prevent architectural decisions from coming back to bite them during the release and deploy stages.



The Source Scan Stage executes the following 5 steps:

1. Install the CLI tools:

- syft (Anchore's open source SBOM generation tool), and
- anchorectl

2. Syft generates an SBOM for the source code in the repository—then anchorectl pushes that SBOM to the centralized SBOM repository in the Anchore Enterprise deployment

3. Isn't there a step here where the AE deployment runs the SBOM against the organization's policies and generates a policy evaluation scan?

4. anchorectl then pulls the policy evaluation from the Anchore Enterprise deployment for the submitted SBOM

5. The generate_policy_tickets_jira.py script then evaluates the policy evaluation for any Stop actions and creates Jira tickets for all policy violations

6. anchorectl stops the pipeline if the policy evaluation had any Stop actions. If there are no Stop actions, the pipeline continues to run

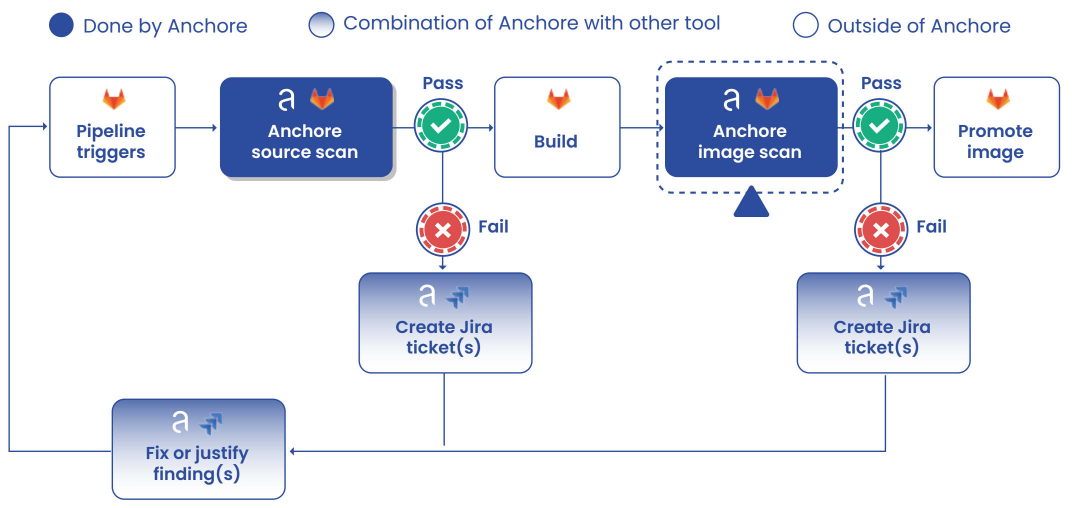
---

Below is an example of the GitLab build steps that will execute the process outline above:

```
anchore-source-scan:
  stage: anchore-source-scan
  image: ubuntu:latest
  script:
    - apt update
    - apt install curl -y

## Install anchorectl and syft

    - curl -sSfL https://anchorectl-releases.anchore.io/anchorectl/install.sh | sh -s -- -b /
usr/local/bin
    - curl -sSfL https://raw.githubusercontent.com/anchore/syft/main/install.sh | sh -s -- -b
/usr/local/bin

## Generate source code SBOM and push to Anchore Enterprise deployment

    - syft -o json . | anchorectl source add -a anchore-demo@latest gitlab.com/your-gitlab-
project/anchore-demo@$CI_COMMIT_SHA --from -

## Run Jira script to create tickets if there are Stop actions for this SBOM

python3 anchore_jira_ticket_generator.py

## Break pipeline if this SBOM fails an Anchore polict evaluation

anchorectl image check anchore-demo@latest gitlab.com/your-gitlab-project/anchore-
demo@$CI_COMMIT_SHA -p anchore-developer-bundle -f
```

Anchore Image Scan Stage

This stage in the pipeline follows the same logic as the source scan stage. After the image is built, Anchore generates an SBOM for the image. It then pushes that SBOM to Anchore Enterprise, and stops the pipeline if our anchore-developer-bundle policy detects a policy violation.



```
anchore-image-scan:
  stage: anchore-image-scan
  image: anchore/enterprise-gitlab-scan:v4.0.0
  needs: ["build"]

## Install anchorectl

    - curl -sSfL  https://anchorectl-releases.anchore.io/anchorectl/install.sh  | sh -s -- -b $HOME/.
local/bin

## Push the image to Anchore to generate the SBOM

    - anchorectl image add --no-auto-subscribe --wait --dockerfile Dockerfile --force --from registry
${ANCHORE_IMAGE}

## Run Jira script to create tickets if there are Stop actions for this SBOM

python3 anchore_jira_ticket_generator.py

## Break pipeline if this SBOM fails an Anchore policy evaluation

anchorectl image check --detail $ANCHORE_IMAGE -p anchore-developer-bundle -f
```

Fix or Justify Findings

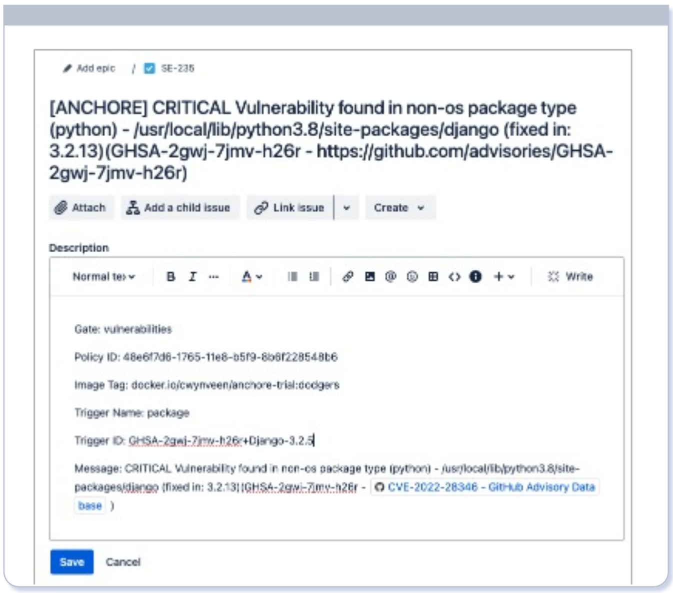
If Stop actions are present in a policy evaluation for an SBOM in the pipeline, the pipeline will fail. The policy violation will need to be remediated before the pipeline can be re-run. The most common remediation actions are:

1. Fix the issue, by upgrading the impacted component to a patched version,
2. Justify the finding, or
3. Create a Plan of Action & Milestone (POAM) to mitigate the issue at a future point in time.

All of the Stop actions that get triggered by the anchore-developer-bundle will have a fix available. If upgrading the impacted component causes breaking changes to the application, then a justification or POAM can be specified. The justification can be added to Anchore Enterprise's Allowlist to either indefinitely allow the finding, or temporarily allow the finding for a specified amount of time. Once a finding is added to an Allowlist in Anchore, the Stop action will instead return a Go action in the policy evaluation and no longer cause pipeline to halt.

To read more about using Allowlists in Anchore, please refer to our docs. For example justifications, please look at the references listed at the end of this guide.

Below is an example screenshot of a Jira ticket that is generated by the anchore_jira_ticket_generator.py script. It captures which component is impacted, and what version the fix is available in.



[ANCHORE] CRITICAL Vulnerability found in non-os package type (python) - /usr/local/lib/python3.8/site-packages/django (fixed in: 3.2.13)(GHSA-2gwj-7jmv-h26r - https://github.com/advisories/GHSA-2gwj-7jmv-h26r)

Description

Gate: vulnerabilities

Policy ID: 48e6f7d6-1765-11e8-b5f9-8b6f228548b6

Image Tag: docker.io/cwynveen/anchore-trial:dodgers

Trigger Name: package

Trigger ID: GHSA-2gwj-7jmv-h26r+Django-3.2.5

Message: CRITICAL Vulnerability found in non-os package type (python) - /usr/local/lib/python3.8/site-packages/django (fixed in: 3.2.13) (GHSA-2gwj-7jmv-h26r - CVE-2022-28346 - GitHub Advisory Database)

Summary

This solution guide provides a comprehensive approach to integrating Anchore Enterprise into CI pipelines to enhance security measures for developers. Focusing on the most commonly used DevSecOps tools used by federal customers—GitLab and Jira. The guide outlines an automated workflow to identify and prioritize security findings without hindering the developer experience. Anchore's flexible policy engine achieves this by only flagging critical security issues with actionable remediations. This creates an optimal balance between the security risks and developer velocity for the organization.

The guide highlights the major steps to achieve this automated workflow:

- Create and customize a policy bundle to flag critical vulnerabilities
- Example CI/CD pipeline configurations for source and build stages that
- Generate SBOMs
  - Evaluate the SBOM against the policy bundle
  - Automatically create Jira tickets based on policy violations
  - Actions to remediate policy violations

This shift left strategy of identifying and remediating prioritized security issues early in the development process has significant benefits, including: lower cost of remediation, faster time to market, improved overall security posture, and increased user trust.

For further details on integrating Anchore Enterprise with your CI/CD pipeline, refer to the official Anchore Enterprise documentation or contact our support team.

References

GO anchore-shift-left-automation GitHub repo:
https://github.com/connorwynveen/anchore-shift-left-automation

Justifications Table

JUSTIFICATION	FINDING JUSTIFIED GUIDELINES	ADDITIONAL INFORMATION
False Positive	No mitigation or re-mediation required	False positives include items that a scanner incorrectly identifies such as a wrong package or version. This does **NOT** include findings that are mitigated or "not exploitable".
Disputed	No mitigation or re-mediation required	Issues marked as DISPUTED within the NVD. This does **NOT** include issues a contributor is disputing. It must be marked as such within the NVD.
Won't Fix	Must be mitigated; must be remediated	Upstream (not OS distribution such as Redhat, Debian, Ubun-tu, etc.) states they will not fix the security flaw.
Distro - Won't Fix	No mitigation or re-mediation required	Issues marked as WONT_FIX by the vendor. Reserved for OS distributions packages.
No Fix Available	Must be mitigated; must be remediated	There is no patch available. This **ONLY** considers the vulnera-ble library itself, not downstream products.
Distro - Pending Resolution	Must be mitigated; must be remediated	Vulnerability is for a library provided by the Operating System (OS) distribution. Only applicable when using the latest ver-sion of a distribution and library.
Mitigated	Mitigation is comple-te; must be reme-diated	Issue has a mitigation that reduces severity or risk.
Not Vulnerable	Mitigation is comple-te; must be reme-diated	Issue is not exploitable within application.
Unreleased	Must be mitigated; must be remediated	Fix is available in a branch for the next release but is not yet available.
Pending Resolution	Must be mitigated; must be remediated	Upstream project is aware of vulnerability and is tracking an issue ticket to fix.
True Positive	Must be mitigated; must be remediated	Image is vulnerable to this finding. Default state of a new finding.
Policy N/A	No mitigation or re-mediation required	Product functionality requires security policy exception. (Only applies to policy findings, not CVEs.)

anchore

sales@anchore.com
anchore.com