



Cortex Cloud Application Security

Confidential - Copyright © Palo Alto Networks

1. Cortex Cloud Application Security

2. Onboard Data Sources

2.1. Onboard version control systems

- 2.1.1. AWS CodeCommit
- 2.1.2. Azure DevOps
 - 2.1.2.1. Azure DevOps onboarding system architecture
- 2.1.3. Bitbucket Cloud
- 2.1.4. Bitbucket Data Center
- 2.1.5. GitHub Cloud
- 2.1.6. GitHub Enterprise (On-Prem)
- 2.1.7. GitLab SaaS
- 2.1.8. GitLab Self Managed (On-Prem)

2.2. Onboard CI/CD systems

- 2.2.1. Onboard CircleCI for CI/CD pipeline scans
- 2.2.2. Onboard Jenkins for CI/CD pipeline scans

2.3. Integrate CI tools

- 2.3.1. AWS CodeBuild
- 2.3.2. Onboard CircleCI for code scans
- 2.3.3. Connect Cortex CLI
- 2.3.4. GitHub Actions
- 2.3.5. Onboard Jenkins for code scans
- 2.3.6. Onboard Terraform Cloud (Run Tasks)
- 2.3.7. Onboard Terraform Enterprise (Run Tasks)

2.4. CLI pipeline code snippets

2.5. Onboard private package registries

- 2.5.1. JFrog Artifactory
- 2.5.2. Onboard JFrog Artifactory

2.6. Supported third-party data sources

- 2.6.1. Ingest Semgrep data
- 2.6.2. Ingest Snyk data
- 2.6.3. Ingest SonarQube SAST data
- 2.6.4. Ingest Veracode SAST data
- 2.6.5. Generic 3rd Party AppSec Collector
 - 2.6.5.1. Onboard the 3rd Party AppSec Collector

2.7. Manage data source integrations

2.8. Transporter over Broker VM

- 2.8.1. Set up a Transporter applet on Broker VM
- 2.8.2. Set up a Transporter on your VCS

3. Cortex Cloud Application Security dashboard

4. Application Security Posture Management (ASPM)

- 4.1. ASPM use cases
- 4.2. ASPM key features
- 4.3. ASPM user roles and permissions

4.4. Code to Cloud

- 4.4.1. Supported integrations
- 4.4.2. Code to Cloud visibility
- 4.4.3. Code to Cloud troubleshooting

4.5. ASPM Command Center

- 4.5.1. ASPM Command Center workflow

4.6. Coverage

- 4.6.1. Coverage in the user interface

4.7. Compliance for Cortex Cloud Application Security

- 4.7.1. Infrastructure-as-Code (IaC) compliance
- 4.7.2. Manage IaC compliance
- 4.7.3. CI/CD Compliance
 - 4.7.3.1. Create CI/CD compliance reports

4.8. Urgency

- 4.8.1. Urgency metrics

4.9. Backlog baseline

- 4.9.1. Backlog use cases
- 4.9.2. Issue/Finding classification by scanner
- 4.9.3. Using Backlog

4.10. Service Lead Agreements (SLA)

- 4.10.1. Configure and monitor Cortex Cloud Application Security SLAs

4.11. Applications

- 4.11.1. Defining Business Applications
 - 4.11.1.1. Defining business applications by Criteria
 - 4.11.1.2. Define applications by VCS criteria
 - 4.11.1.3. Manage application criteria
 - 4.11.1.4. Define applications by cloud tag-based criteria
 - 4.11.1.5. How to manually build an application
- 4.11.2. Application management and visibility
- 4.11.3. Business application assets
 - 4.11.3.1. Business application expanded asset details
 - 4.11.3.2. Export business application data
- 4.11.4. Scope user access to applications (Application SBAC)
 - 4.11.4.1. Enable SBAC in the Cortex Cloud tenant
 - 4.11.4.2. Create an application-based Asset Group
 - 4.11.4.3. Scope user access to an application
 - 4.11.4.4. Create application-scoped policies

4.12. Repositories as assets

- 4.12.1. Explore repository assets
- 4.12.2. In-depth repository asset information
- 4.12.3. Manage Repository assets
- 4.12.4. Export Software Bill of Materials (SBOM)
- 4.12.5. Manage issues detected in repositories

4.13. Manage 3rd party findings and generated issues

4.14. Manage code weaknesses

- 4.14.1. Code weaknesses issue inventory
- 4.14.2. Detailed code weakness issue information
- 4.14.3. Code weakness findings

5. CI/CD Security

5.1. CI/CD Security user roles and permissions

5.2. CI/CD Assets

- 5.2.1. CI/CD Instances as assets
 - 5.2.1.1. Explore CI/CD Instance assets
 - 5.2.1.2. In-depth CI/CD pipeline instance asset information
 - 5.2.1.3. Manage CI/CD pipeline instances
- 5.2.2. CI/CD Pipelines as assets
 - 5.2.2.1. Explore CI/CD Pipeline assets
 - 5.2.2.2. In-depth CI/CD pipeline asset information
 - 5.2.2.3. Manage CI/CD pipeline assets
- 5.2.3. Version Control System (VCS) Organizations as assets
 - 5.2.3.1. Explore VCS Organization assets
 - 5.2.3.2. In-depth VCS Organization asset information
 - 5.2.3.3. Manage VCS organization assets
- 5.2.4. VCS Collaborators-as-assets
 - 5.2.4.1. In-depth Collaborator asset information
 - 5.2.4.2. Manage Collaborator assets

5.3. Supply Chain Inventories

- 5.3.1. Supply Chain Tools
 - 5.3.1.1. Supply Chain Tools
 - 5.3.1.2. Expanded Supply Chain tool information
- 5.3.2. Supply Chain Catalog

5.4. CI/CD Risks

- 5.4.1. CI/CD pipeline issues
- 5.4.2. Expanded CI/CD risks issue information
- 5.4.3. VCS and CI/CD pipeline risk findings

5.5. CI/CD Rules

- 5.5.1. CI/CD rules roles and permissions
- 5.5.2. CI/CD rules inventory

5.6. CI/CD Policies

- 5.6.1. CI/CD policies user roles and permissions
- 5.6.2. CI/CD policies inventory
- 5.6.3. Create custom CI/CD policies
- 5.6.4. Manage CI/CD policies

6. Code Security

- 6.1. Code Security user roles and permissions
- 6.2. Code Security assets
- 6.3. Software packages as assets
 - 6.3.1. Explore software package assets
 - 6.3.2. In-depth software package asset information
- 6.4. Infrastructure-as-Code (IaC) resources as assets
 - 6.4.1. Explore IaC assets
 - 6.4.2. In-depth IaC resource asset information
- 6.5. Code Security scanners
- 6.6. Secrets scanners
 - 6.6.1. Secrets issues
 - 6.6.2. Secrets issues inventory
 - 6.6.3. Detailed Secrets issue information
 - 6.6.4. Secrets findings
 - 6.6.5. Manage Secrets issues
- 6.7. Infrastructure as Code (IaC) misconfiguration scanner
 - 6.7.1. Supported frameworks and languages

- 6.7.2. IaC misconfiguration issues
- 6.7.3. IaC misconfiguration issues inventory
- 6.7.4. Detailed IaC misconfiguration issue information
- 6.7.5. IaC misconfiguration findings
- 6.7.6. Manage IaC misconfiguration issues

6.8. IaC Drift Detection scans

- 6.8.1. IaC Drift Detection issues
- 6.8.2. IaC Drift Detection issue inventory
- 6.8.3. Detailed IaC drift detection issue information
- 6.8.4. IaC Drift Detection findings

6.9. Software Composition Analysis (SCA) scanners

- 6.9.1. Supported Software Composition Analysis (SCA) frameworks and languages
- 6.9.2. Software Composition Analysis (SCA) vulnerability issues
 - 6.9.2.1. Vulnerability issues inventory
 - 6.9.2.2. Detailed vulnerability issue information
 - 6.9.2.3. CVE vulnerabilities findings
 - 6.9.2.4. Manage SCA CVE vulnerability issues
- 6.9.3. License miscompliance issues
 - 6.9.3.1. License miscompliance issues inventory
 - 6.9.3.2. Expanded License miscompliance issues information
 - 6.9.3.3. License miscompliance findings
 - 6.9.3.4. Open-source software license categories
 - 6.9.3.5. Manage license miscompliance issues
- 6.9.4. Package Integrity
 - 6.9.4.1. Package Integrity inventory
 - 6.9.4.2. Expanded Package Integrity issues inventory information
 - 6.9.4.3. Package Integrity findings
 - 6.9.4.4. Manage Package Integrity issues

6.10. Application Security scans management

- 6.10.1. Overview
- 6.10.2. Branch periodic scans
- 6.10.3. Pull Request scans
- 6.10.4. CI scans
- 6.10.5. Manage repository scan configurations
- 6.10.6. Monitor data source instances health

6.11. Application Security Policies

- 6.11.1. User roles and permissions
- 6.11.2. Policies inventory
- 6.11.3. AI-recommended guardrails
 - 6.11.3.1. Manage AI-recommended guardrails
- 6.11.4. Create Cortex Cloud Application Security policies
 - 6.11.4.1. Create code security policies
 - 6.11.4.1.1. Cortex Cloud Application Security code policy Condition attributes
 - 6.11.4.2. Create IaC Drift Detection policies
 - 6.11.4.3. Manage Cortex Cloud Application Security policies

6.12. Application Security Rules

- 6.12.1. Roles and permissions
- 6.12.2. Rules inventory
- 6.12.3. Create custom Cortex Cloud Application Security rules
- 6.12.4. Manage Cortex Cloud Application Security custom rules
- 6.12.5. Configure YAML file properties

6.13. Application Security CLI

- 6.13.1. Connect Cortex CLI
- 6.13.2. Cortex CLI usage for Cortex Cloud Application Security
- 6.13.3. CLI pipeline code snippets

6.13.4. Cortex CLI Cortex Cloud Application Security command line reference

6.13.5. Cortex CLI common command line reference guide

6.13.6. Git Hooks

 6.13.6.1. Cortex CLI pre-commit hooks

 6.13.6.1.1. Pre-commit hook usage

 6.13.6.2. Cortex CLI pre-receive hooks

 6.13.6.2.1. Pre-receive hook usage

6.14. IDE

 6.14.1. System requirements

 6.14.2. Visual Studio (VS) Code and VS Code compatible IDEs

 6.14.2.1. How to use the Cortex Cloud extension in VS Code

 6.14.3. JetBrains

 6.14.3.1. How to use the JetBrains Cortex Cloud extension

6.15. Developer Suppressions

7. API documentation

1 | Cortex Cloud Application Security

Cortex Cloud Application Security

Abstract

Application Security provides unified visibility and control over app security throughout the lifecycle, identifying vulnerabilities to protect data and integrity.

The Cortex Cloud Application Security module provides comprehensive security for your applications throughout their entire lifecycle. It offers unified visibility and control over your application's security from development through to deployment.

Cortex Cloud Application Security license requirements

The following active licenses are required to enable and utilize the components of the Application Security module:

- **ASPM:** Requires one of the following licenses: Cloud (Posture Security or Runtime Security), or XSIAM Premium
- **CI/CD Security:** Requires one of the following licenses: Cloud (Posture Security or Runtime Security), or XSIAM Premium
- **Code Security:** This license is an add-on module and requires an existing base license, such as Cloud (Posture Security or Runtime Security) or XSIAM Premium. It must be purchased separately

Cortex Cloud Application Security use cases

- **Application Security Posture Management (ASPM):** Provides a consolidated view of application risks and vulnerabilities across your environment, enabling you to understand and manage your overall security posture. For more information refer to Application Security Posture Management (ASPM)
- **CI/CD Security:** Focuses on securing your continuous integration and continuous delivery pipelines, ensuring the integrity and security of your automated build and deployment processes. For more information refer to CI/CD Security
- **Code Security:** Identifies and helps mitigate security issues directly within your source code, including vulnerabilities in Infrastructure-as-Code (IaC) and open-source components, from the earliest stages of development. For more information refer to Code Security

Upgrade from Prisma Cloud to Cortex Cloud

Prisma Cloud customers can use the Upgrade Helper to copy Application Security data and configurations from their Prisma Cloud tenant to their new Cortex Cloud tenant. For more information refer to Upgrade from Prisma Cloud to Cortex Cloud.

2 | Onboard Data Sources

Abstract

Onboard VCS, integrate CI tools, registries and ingest third-party data for a comprehensive view of your application and supply chain security.

Onboard your Version Control Systems (VCS), integrate your CI tools, private registries (currently JFrog Artifactory), and ingest third-party data from external vendors, to gain a comprehensive view of your application and supply chain security.

Onboard VCS systems

By onboarding your VCS systems, you gain complete visibility into your repositories and pipeline assets and out-of-the-box CI/CD system capabilities.

- **Visibility**
 - **Asset mapping and inventory:** All repositories and their associated pipeline assets (such as GitHub Actions) are mapped, creating a complete inventory of your environment. This includes discovering forgotten or unauthorized devices and their connections
 - **Vulnerability management:** The inventory allows you to identify and prioritize security vulnerabilities since you can't secure what you don't know exists
 - **Compliance and auditing:** It provides the necessary data and logs to prove compliance during audits
 - **Attack surface understanding:** It helps you understand and manage your potential attack surface
- **CI/CD system capabilities:** Onboarding a VCS automatically integrates with specific CI/CD systems, triggering automated scans that identify supply chain security risks within your pipelines. For more information see Onboard CI/CD systems below

To onboard VCS systems, refer to Onboard version control systems.

Onboard CI/CD systems

You can onboard CI/CD systems to scan for configuration threats in your organization's instance, pipelines, and individual repositories. While onboarding supported version control systems includes out-of-the-box CI/CD capabilities, you must explicitly onboard CircleCI and Jenkins to get code scanning for these systems. Onboarding CI/CD systems provides the following scans.

- **Organization instance configuration threats:** This type of scan detects security issues at the level of the overall organization's instance of a version control system (VCS), such as GitHub. For example, it can flag risks such as **Project webhook SSL verification disabled** or **Variable is not scoped to an environment**.
- **Pipeline configuration risks:** This scan identifies security risks within the configuration of your pipelines. Examples of risks it detects include **Excessive GitHub Action permissions**, **using an unpinned container image in a pipeline**, or **CI instance accesses cloud provider using insecure long-term credentials**
- **Repository configuration issues:** This scan checks problems with the settings and configurations of individual code repositories. Examples include **Forking of a private repository is allowed** and **A change in settings so that a review is no longer required before merging code**

To onboard CI/CD systems, refer to Onboard CI/CD systems.

Integrate CI tools to enable code scans through Cortex CLI

By integrating CI tools, you get two main benefits: code scans and streamlined security workflows. This is achieved by inserting code snippets directly into your existing CI workflows, which then run through the unified Cortex CLI to trigger automated security checks.

- **Code scans:**
 - **Code scanning for IaC (Infrastructure as Code):** Finds misconfigurations in your IaC files, ensuring your cloud and infrastructure environments are secure from the start
 - **Software Composition Analysis (SCA) scans:** Identify vulnerabilities in open-source libraries and third-party components, along with license misconfigurations and package integrity issues
 - **Secrets detection:** Finds hardcoded secrets, such as API keys and passwords, in your code and pipelines to prevent unauthorized access and data breaches
- **Streamlined workflows:** By integrating security scans directly into your CI/CD pipelines, you achieve a shift-left security model, moving security from a final check to an early, continuous process within the development lifecycle.
 - **Early threat detection:** You can identify and fix security threats as soon as they are introduced
 - **Automated and seamless integration:** The use of code snippets and a unified CLI makes the security checks a seamless part of your existing CI process, requiring no manual intervention

To integrate CI tools through code snippets, refer to CLI pipeline code snippets

Integrate with JFrog Artifactory

Integrate with JFrog Artifactory to provide the Cortex Cloud Application Security SCA scanner direct access to packages stored in Artifactory, a private registry. This access allows the scanner to retrieve dependency metadata and package contents, enabling full visibility, accurate dependency trees, and reliable detection of supply-chain vulnerabilities.

Ingest third-party data

- **Expanded security coverage:** While Cortex Cloud Application Security provides robust native scanning, ingesting data from other security tools (such as SAST) expands your overall coverage. This creates a more comprehensive security profile, leaving no potential vulnerabilities unmonitored
- **Contextual Enrichment:** Third-party data adds context to your existing security information. By ingesting this data, Cortex Cloud automatically correlates its findings with threats detected by Cortex Cloud Application Security scans. This helps you prioritize which vulnerabilities to address first based on the actual risks they pose, enabling more strategic and efficient remediation
- **Leveraging existing investments:** You can maximize the value of your current security tools through Cortex Cloud , which powers the consolidation of your security data. Instead of operating in separate silos, your tools' data is integrated into a single, consolidated view of your security state within Cortex Cloud. This ensures that the data you have already collected from various sources is actively used to inform and strengthen your overall security strategy

To ingest third-party data, refer to Supported third-party data sources.

Code replication and retention policies

Cortex Cloud does not replicate or store your application code unless your organization has subscribed to the Application Security add-on license. The data collected and displayed relates only to security findings and metadata, preserving the integrity and location of your source code.

Disclaimer

While Cortex Cloud Application Security provides guidance during integration and explain the steps involved when you are redirected to third party version control systems (such as GitHub SaaS , GitLab SaaS and so on), Cortex Cloud Application Security does not assume responsibility for changes or variations in these platform processes. Always refer to the official documentation of the third party to ensure you are following their most current and precise instructions.

2.1 | Onboard version control systems

Connect Cortex Cloud Application Security with your version control systems (VCS) to gain comprehensive visibility into, and monitor, the systems, technologies, configurations, and pipelines that make your VCS platform.

These integrations trigger both periodic scans and scans on pull requests (PRs) via a webhook, enabling security scans to identify and remediate Infrastructure-as-Code (IaC) misconfigurations, exposed secrets and license non-compliance in your VCS environment. Scan results are displayed directly in PR comments and reports, allowing you to analyze, prioritize and fix issues as soon as they are detected.

NOTE:

Cortex Cloud Application Security (which includes IaC and Secrets scanning), is an add-on to a license (such as Posture Security) that must be purchased separately.

Supported VCS data sources

Cortex Cloud Application Security currently supports the following VCS data source integrations:

- AWS CodeCommit
- Azure DevOps
- Bitbucket Cloud
- Bitbucket Data Center
- GitHub Cloud
- GitHub Enterprise (On-Prem)
- GitLab SaaS
- GitLab Self Managed (On-Prem)

Each integration requires a unique set of permissions and subscribed events.

How to onboard a VCS data source

VCS data sources are listed in the Cortex data source catalog.

1. Navigate to Settings → Data Sources & Integrations → + Add New → Show More → Code Repositories.

TIP:

Navigate to Settings → Data Sources & Integrations → + Add New → and enter your VCS data source in the search bar.

2. From the search results, select a data source and follow the instructions in its configuration wizard to complete the settings configuration process.

NOTE:

Disclaimer: When onboarding with third-party data sources, we outline the required steps for setup, but we do not monitor these external resources, and they may change over time. Always refer to the relevant third-party documentation for the most current integration steps.

Onboard an additional data source instance

To onboard an additional data source instance:

1. On the Data Sources & Integrations page, select an integration from the table and click Add Instance.
2. Complete the onboarding through the configuration wizard.

Verify data source connectivity status and connected repositories

You can verify the connectivity status of data source instances and their connected repositories through one of these methods:

- Navigate to Settings → Data Sources & Integrations. This page displays all data sources with their connected instances, including connectivity status and additional instance details.
- When browsing the Data Source catalog, click a data source to view its details.

Manage VCS instances

You can manage VCS data source instances. Hover over an instance and right-click to access the following actions:

1. Select Settings → Data Sources & Integrations.
2. Click a data source to see a list of its connected instances.
3. Hover over an instance and right-click to access the following actions:
 - Details: View details of the data source instance, including a list of connected repositories and organization, connectivity status, last scan date, and when initially connected.
 - Edit instance: Opens the Select Repositories step of the integration wizard, allowing you to edit connected repositories. You can also edit the instance configuration by navigating back to the previous step of the wizard and modifying relevant details
 - Delete instance: Deletes the entire instance
 - Remove a connected repository: Right-click on a repository in the list, and click Remove Repository

Manage findings and issues

For information about managing findings detected after onboarding data sources, and issues generated from findings refer to Code Security scanners.

2.1.1 | AWS CodeCommit

Integrate Cortex Cloud Application Security with your AWS CodeCommit version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This allows you to analyze, prioritize, and resolve detected issues efficiently.

How the integration works

To ensure security, the platform does not store or use your personal AWS credentials for scanning. Instead, the integration utilizes a cross-account trust relationship through a dedicated IAM Service Role. This relationship is secured using an External ID, a unique security identifier that prevents unauthorized third-party access.

- **Deployment:** You deploy a CloudFormation template provided during onboarding. This template creates the necessary IAM roles and permissions automatically, requiring no manual configuration
- **The service role:** This template creates a specific IAM role for cross-account access that trusts the platform. The trust policy is automatically configured with a unique **External ID** generated for your tenant. This role follows least privilege principles by requiring only necessary CodeCommit permissions and is limited to the platform AWS account
- **Auditability:** All actions performed by the service role are logged in AWS CloudTrail, providing a permanent audit trail of all repository access and scanning activity for compliance monitoring
- **Scanning policies:** The role includes the required policies for scanning operations and permissions for CodeCommit repository access
- **Events:** The template configures a Simple Notification Service (SNS) topic with an HTTP subscription to the platform webhook URL. The template automatically applies an SNS Access Policy that allows CodeCommit to publish events and authorizes the platform to subscribe to the topic. When code changes occur, this topic pushes a notification to the webhook, triggering the platform to assume the service role and initiate a scan

PREREQUISITE:

Before you begin, ensure the following:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- **AWS user permissions:** To onboard CodeCommit, the user logged into the AWS Console must have permissions to deploy the CloudFormation stack and authorize the creation of the following resources:
 - **cloudformation:CreateStack:** Required to deploy the integration template
 - **iam:CreateRole:** Required to allow the template to provision the Service Role for scanning operations
 - **sns:CreateTopic:** Required to allow the template to provision notification triggers. **Note:** You must ensure your account is prepared to create an SNS topic for each required region if your Cloud account and stack are in different regions, as AWS requires SNS events to reside in the same region as the stack

NOTE:

During deployment, you must acknowledge the CAPABILITY_IAM setting in the AWS Console to allow the creation of these resources.

- **Required scanning and policy permissions**

The system requires specific permissions to access repositories and evaluate security conditions:

- **Scanning permissions:** Rights for the Service Role to access and scan CodeCommit repositories.
- **Policy permissions:** Rights to detect findings and handle issues generated from policies based on repository conditions and scan results.

Once the stack is created, the new IAM Service Role will automatically possess permissions to perform scans and handle policy-generated issues. For the complete list of permissions, refer to Technical appendix: IAM Service Role permissions below

NOTE:

The permissions are configured entirely by the CloudFormation template; no manual action is required.

Onboarding steps

1. Generate the template in the Cortex Cloud tenant.
 - a. Navigate to Settings → Data Sources & Integrations → + Add New.
 - b. Enter AWS CodeCommit in the search bar.
 - c. Hover over the AWS CodeCommit card in the catalog and click Add.Select Add Another Instance if the data source instance has already been configured.
The AWS CodeCommit onboarding wizard is displayed.
 - d. Download and save the CloudFormation template (YAML file) or copy the link for your administrator.
2. Create a stack on the AWS console.
 - a. Login to the AWS Console → search for CloudFormation → Create stack.
 - b. In the Specify template section, select Upload a template file → Choose file → upload the downloaded CloudFormation template → Next.

- c. Provide a stack name → proceed through the configuration options..
 - d. Review the stack configurations → acknowledge the IAM resource creation → Submit.
3. Select the repositories to be scanned from the Cortex Cloud tenant.
- a. On the Data Sources & Integrations page, filter for AWS CodeCommit → select the AWS CodeCommit data source.
 - b. Locate and right-click click on your newly created connector.

TIP:

The instance ID is identical to the stack ID on the AWS platform.

- c. Under Selection Options, choose the repositories to be connected to the instance:
 - Permit all existing repositories
 - Permit all existing and future repositories
 - Select Choose from repository list and select repositories from the list
 - d. Click Save.
4. Verify integration through the tenant or in AWS using either of these options:
- In Cortex Cloud: On the Data Sources & Integrations page, filter for AWS CodeCommit, select the AWS CodeCommit data source that is displayed, and verify that the status of your instance (connector) is Connected
 - In AWS: Open CloudFormation → Stacks. Verify that the integration is displayed with a Create Complete status

Validate repository scan and view scan results

After connection, the platform automatically triggers a security scan of the repository. Scanning is supported for Infrastructure as Code (IAC) analysis, Software Composition Analysis (SCA), and Secrets detection.

1. Navigate to Modules → Application Security → Periodic scans.
2. Filter by Provider → AWS CodeCommit.
3. Verify that the scan health of your repository is Completed.
4. Select the repository.
5. Review a summary of findings detected by the scans and issues generated by policies targeting the repository.
6. **Next step:** Navigate to a dedicated issue table (such as Secrets) to understand and remediate the issue.

Data protection

Cortex Cloud ensures the security and integrity of your code:

- **Isolated scanning:** Repository contents are scanned within a strictly isolated sandbox environment to prevent cross-contamination
- **Tenant isolation:** All security findings are stored with tenant isolation to ensure your data remains private and inaccessible to others
- **No Persistence:** No repository credentials or sensitive secrets are stored within the platform infrastructure
- **Temporary access:** Access is managed through secure cross-account IAM role assumption which provides temporary permissions without the need for static keys

Troubleshooting

Review the following common issues and resolutions to resolve errors during stack creation, repository connection, or scanning processes.

- **CloudFormation stack creation fails**

Stack status shows CREATE_FAILED or ROLLBACK_COMPLETE

- Verify IAM permissions for stack creation
- Check for naming conflicts with existing stacks
- Review CloudFormation events for specific error messages
- Ensure CAPABILITY_IAM is granted

- **Connection status shows WARNING or ERROR**

Instance status not CONNECTED

- Verify CloudFormation stack is in CREATE_COMPLETE state
- Check IAM role trust relationship
- Ensure CodeCommit repository exists and is accessible
- Verify cross-account access permissions

- **No scan results**

Repository connected but no findings in tables

- Check repository contains scannable files
- Verify scan job completed successfully
- Review scanner logs for errors
- Ensure repository is not empty

- **codecommit:GitPull:** Allows users to pull Git repository changes
- **codecommit>ListBranches:** Grants the ability to list branches within a repository
- **codecommit:GetBranch:** Required to get details about a branch in a repository
- **codecommit:GetPullRequest:** Enables fetching details of a specific pull request
- **codecommit:GetFolder:** Required to view the contents of a specified folder in a repository from the CodeCommit console
- **codecommit:GetFile:** Required to view the encoded content of an individual file and its metadata in a repository from the CodeCommit console
- **codecommit:GetBlob:** Allows fetching of an object (such as a file) from a repository
- **codecommit:GetCommitsFromMergeBase:** Grants access to commits from the merge base of a branch
- **codecommit:GetCommentsForPullRequest:** Allows retrieval of comments associated with a pull request
- **codecommit:PostCommentReply:** Required to create a reply to a comment on a comparison between commits or on a pull request
- **codecommit:UpdateComment:** Allows updating of comments on pull requests
- **codecommit:PostCommentForPullRequest:** Required to post a comment on a pull request in a repository
- **codecommit:GetComment:** Permits retrieval of a specific comment on a pull request
- **codecommit:GetCommit:** Allows fetching details of a specific commit
- **codecommit:GetDifferences:** Grants access to differences (changes) between commits, branches, and so on
- **codecommit:BatchGetRepositories:** Enables batch retrieval of repository details
- **codecommit:getRepository:** Permits fetching details of a specific repository
- **codecommit>ListRepositories:** Grants the ability to list repositories within an account
- **codecommit:getRepositoryTriggers:** Allows fetching of triggers configured for a repository
- **codecommit:PutRepositoryTriggers:** Enables configuration of repository triggers
- **codecommit:TestRepositoryTriggers:** Allows testing of repository triggers
- **codecommit:GetTree:** Required to view the contents of a specified tree in a repository from the CodeCommit console. This is an IAM policy permission only, not an API action that you can call
- **codecommit:GetReferences:** Permits fetching of references (branches, tags, etc.) in a repository
- **codecommit:GetObjectIdentifier:** Grants access to object identifiers within a repository
- **codecommit:GetCommitHistory:** Allows fetching of commit history for a repository
- **codecommit:BatchGetPullRequests:** Required to return information about one or more pull requests in a repository. This is an IAM policy permission only, not an API action that you can call

- **codecommit:BatchGetCommits**: Enables batch retrieval of commit details
- **codecommit:GetCommentsForComparedCommit**: Required to return information about comments made on the comparison between two commits in a repository
- **codecommit:PostCommentForComparedCommit**: Required to create a comment on the comparison between two commits in a repository
- **codecommit:PostCommentReply**: Enables posting replies to comments on pull requests
- **codecommit>ListPullRequests**: Required to return information about the pull requests for a repository
- **codecommit>DeleteCommentContent**: Required to delete the content of a comment made on a change, file, or commit in a repository. Comments cannot be deleted, but the content of a comment can be removed if the user has this permission
- **codecommit>CreateBranch**: Permits creation of branches within a repository
- **codecommit:GetBranch**: Permits retrieval of branch details
- **codecommit>CreateCommit**: Allows creation of commits in a repository
- **codecommit>CreatePullRequest**: Enables creation of pull requests in a repository
- **codecommit:PutFile**: Required to add a new or modified file to a repository from the CodeCommit console, CodeCommit API, or the AWS CLI
- **codecommit>ListAssociatedApprovalRuleTemplatesForRepository**: Grants access to associated approval rule templates for a repository
- **codecommit>ListApprovalRuleTemplates**: Allows listing of approval rule templates
- **codecommit:GetApprovalRuleTemplate**: Required to return information about an approval rule template in an Amazon Web Services account
- **codecommit>ListRepositoriesForApprovalRuleTemplate**: Permits listing of repositories associated with an approval rule template

2.1.2 | Azure DevOps

Integrate Cortex Cloud Application Security with your Azure DevOps version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

System architecture overview: Cortex utilizes a secure Delegated Access Model, executing operations under the user's identity rather than an autonomous service account. This architecture supports multi-tenant configurations, allowing you to onboard organizations across distinct Microsoft Entra ID tenants using a single email identity. For more information, refer to Azure DevOps onboarding system architecture.

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- **Azure DevOps permissions:** Ensure the user performing the integration holds one of the following roles in Azure DevOps:
 - Project Administrator: This permission is required to subscribe to webhooks. For more information, refer to the Microsoft Integrate with service hook documentation
 - Member of Project Collection Administrators: Required to subscribe to build.complete events and download the permissions report for CI/CD scans. As Organization owners are automatically part of this group, they also possess this permission
- **Scope:** The Cortex application requires the following authorization scopes. These scopes are granted automatically when authorizing via Microsoft Entra ID. If you authenticate using a Personal Access Token (PAT), you must manually select these scopes during token creation

NOTE:

These required Cortex application permissions are displayed by Microsoft during authorization. Each permission includes a scope description, available from the dropdown next to it.

Scope	Description
User.Read	Sign in and read user profile
vso.agentpools	Agent Pools (read)
vso.analytics	Analytics (read)
vso.auditlog	Audit Read Log
vso.build	Build (read)
vso.code_write	Code (read and write)
vso.entitlements	Entitlements (Read)
vso.extension	Extensions (read)
vso.graph	Graph (read)
vso.identity	Identity (read)

Scope	Description
vso.memberentitlementmanage	MemberEntitlement Management (read)
vso.packaging	Packaging (read)
vso.project	Project and team (read)
vso.release	Release (read)
vso.serviceendpoint	Service Endpoints (read)
vso.taskgroups_write	Task Groups (read, create)
vso.tokens	Delegated Authorization Tokens
vso.variablegroups_read	Variable Groups (read)
vso.work_write	Work items (read and write)

- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

Onboarding procedure

Step 1: Initiate in Cortex

1. In Cortex, navigate to Settings → Data Sources & Integrations → + Add New.
2. Search for Azure DevOps.
3. Hover over the card and click Add (or Add Another Instance if one already exists).

Step 2: Select authentication method

Select the method that aligns with your organization's security policy. Microsoft Entra ID is the recommended standard for long-term support.

Option A: Authorize with Microsoft Entra ID (recommended)

This method supports multi-tenant configurations.

1. Select Microsoft Entra ID authentication → Authorize.

IMPORTANT:

When redirected to the Microsoft login screen, do not immediately enter your email.

2. Select Sign-in options → Sign in to an organization.

3. Enter the specific Domain Name of the tenant you wish to onboard and click Next.

NOTE:

This forces Azure to bypass browser cookies and issue a token for the correct directory.

4. Enter your Email address, review the requested scopes, and click Accept on the permissions prompt.

Option B: Authorize with a Personal Access Token (PAT)

1. In Azure DevOps: Navigate to User Settings → Personal access tokens → + New Token.

2. **Organization:** Select All accessible organizations.

3. **Scopes:** Manually select all custom-defined scopes listed in the **Prerequisites** above.

4. Copy and paste the generated token into the Access Token field in the Cortex onboarding wizard and click Authorize.

NOTE:

PATs are static. To onboard a different tenant, you must log in to that specific environment to generate a new token.

Step 3: Configure repositories

1. Once authorized, you are redirected to the Select Repositories step.

2. Select which repositories to scan from the Selection Options menu:

- Permit all existing repositories
- Permit all existing and future repositories (recommended)
- Choose from repository list

3. Click Save.

Verification

Confirm the instance status shows as Connected in the Data Sources list.

1. Navigate to Settings → Data Sources & Integrations.

2. Search for and select the Azure DevOps data source that is presented.

3. In the Azure DevOps instances list, displayed, locate your specific instance and ensure the status shows as Connected.

Post-onboarding: subscribed events

Once successfully integrated, Cortex subscribes to the following events to trigger scans and notifications:

Category	Event	Description
Repositories	—	—
—	git.pullrequest.created	This event is triggered when a new pull request is created in a Git repository. It allows systems to be notified whenever a new pull request is initiated, enabling integration with other services or actions
—	git.pullrequest.updated	This event is triggered when an existing pull request is updated with new changes, comments, or other modifications. It allows systems to stay synchronized with the latest changes in pull requests
—	git.push	This event is triggered when new commits are pushed to a Git repository. It enables systems to track changes to the repository and perform actions such as triggering builds or running tests
—	git.pullrequest.merged	This event is triggered when a pull request is successfully merged into the target branch. It allows systems to take action after a pull request has been merged, such as deploying changes or updating related tasks
Organizations	—	—

Category	Event	Description
—	<code>build.complete</code>	This event is triggered when a build process is completed within an Azure DevOps organization. It allows systems to react to the completion of build tasks, such as notifying stakeholders or triggering subsequent stages in a deployment pipeline

Validation: You can validate the subscription by triggering an action in Azure DevOps and checking for a scan initiation. For example, to verify `git.push`: Push a commit to a connected repository. This should trigger a scan for secrets and IaC misconfigurations.

2.1.2.1 | Azure DevOps onboarding system architecture

This topic details the authentication mechanisms, identity models, and multi-tenant data structures used by Cortex to integrate with Azure DevOps.

Authentication architecture

Supported methods

- Microsoft Entra ID (OAuth 2.0): The recommended standard for Cortex integrations, utilizing dynamic delegated tokens

NOTE:

Microsoft has announced the deprecation of legacy OAuth methods by 2026, making Entra ID the required standard for long-term support.

- Personal Access Tokens (PAT): A static token model where authentication is handled via a token generated directly within Azure DevOps. A PAT is cryptographically bound to the specific organization and user account active at the time of creation

Global configuration

- **Ownership:** The Azure application used for OAuth is verified and owned by Cortex. It is registered in a fixed Cortex Microsoft Entra ID tenant that never changes
- **Regional specificity:** To ensure regional compliance and performance, each Cortex region (such as US, EU) utilizes its own dedicated Azure application registration

The delegated execution and identity model

Cortex operates under a Delegated Access Model, distinguishing it from Service Principal or App-only access models.

Secure conduit principles

The Azure application registered by Cortex serves as the OAuth client and trusted identity, acting solely as a secure conduit to:

- Authenticate the user
- Obtain delegated access tokens
- Call Azure DevOps APIs on behalf of that user

Comparison: Cortex vs. GitHub Apps

Unlike GitHub Apps, where the application becomes a first-class principal (autonomous actor) inside the customer environment, the Cortex Azure application **never operates independently** of a user's delegated identity.

Security and auditability

- **Permissions enforcement:** Every API call is evaluated using the user's existing permissions. Cortex cannot exceed or bypass the access rights of the authenticated user. If a user lacks permission for an operation, the request is denied
- **Audit trail:** In Azure DevOps audit logs, all activity appears as originating from the **individual user's identity**, not a Cortex-owned service account

Multi-tenant and multi-domain logic

Cortex employs a composite identifier logic to support complex organizational structures **[User Email] + [Microsoft Entra ID Tenant ID]**.

Operational comparison: dynamic vs. manual

The architectural difference between OAuth and PATs directly impacts onboarding effort.

- **Entra ID (dynamic discovery):** Because OAuth uses a dynamic tenant selection flow, a single authentication session can discover and connect multiple organizations tied to that tenant automatically
- **Personal Access Token (PAT) (static/manual onboarding):** A PAT lacks cross-organization visibility; it is strictly limited to the specific organization selected during its creation
 - **Result:** Multi-tenant or multi-org onboarding via PAT is a strictly manual process. An administrator must generate and provide a unique PAT for every individual organization they wish to onboard

The email + tenant logic

- **One email, multiple integrations:** A single user identity (email) can own multiple distinct integrations if they target different tenants (such as a Production tenant and a Sandbox tenant)
- **Uniqueness constraint:** You cannot create two integrations for the same email on the same tenant
- **Organization mapping:** Multiple Azure DevOps organizations can be mapped to a single integration if they reside under the same Entra ID tenant. Organizations in different tenants require separate integration instances
- **PAT requirement:** Because a PAT is tied to a specific organization, users must provide a separate PAT for each organization they wish to onboard

Architecture example

The following scenario illustrates how Cortex maps users, tenants, and organizations. In this example, two Cortex integrations are created; one per email + tenant combination; even though the same user email is used across all environments.

User email: dev.user@company.com

Microsoft Entra ID tenants:

- Tenant A (company.onmicrosoft.com)
- Tenant B (subsidiary.onmicrosoft.com)

Azure DevOps organizations:

- Org-1 → connected to Tenant A
- Org-2 → connected to Tenant A
- Org-3 → connected to Tenant B

Resulting Cortex integrations:

Cortex integration	Email	Entra ID tenant	Azure DevOps organizations
Integration #1	dev.user@company.com	Tenant A	Org-1, Org-2
Integration #2	dev.user@company.com	Tenant B	Org-3

User authorization and session handling

When using the recommended Microsoft Entra ID flow, Cortex redirects the user to the Microsoft identity platform: <https://login.microsoftonline.com/common/oauth2/v2.0/authorize>. To ensure security and multi-tenant accuracy, the architecture is designed to bypass standard browser session defaults.

Standard OAuth flow defaults to the user's last active session based on browser cookies. To prevent authorization against the wrong environment, Cortex enforces an explicit domain selection flow.

Session Management

- **The problem:** Without architectural enforcement, Azure may automatically sign a user into their "Home" or "Last Used" directory, leading to token issuance for the wrong tenant
- **The mechanism (explicit bypass):** Users are directed to use the Sign-in option → Sign in to an organization workflow. By entering a specific Tenant ID or Domain, the user overrides cached browser credentials
- **Result:** This ensures the authorization token is issued for the intended directory

Consent

After the explicit tenant selection and successful authentication, Microsoft prompts the user to consent to the requested scopes specifically for that tenant. This consent grant is what allows Cortex to act as a delegated agent, performing actions on the user's behalf.

Authorization scopes and event architecture

Cortex requires specific scopes to enable both API-based data retrieval and its event-driven architecture.

Event subscription (Webhooks)

Cortex subscribes to real-time events (such as `git.push`, `build.complete`) to trigger automated scans.

- **Functional dependency:** There is no single `Webhook` permission in Azure DevOps. The ability to create subscriptions is derived from standard scopes
- **Required scopes:**
 - `vso.code_write`: Required for code-related events
 - `vso.build`: Required for build-related events
- **Impact:** Without these specific permissions, the event-driven architecture cannot function, and Cortex will revert to scheduled (polling) synchronization only

2.1.3 | Bitbucket Cloud

Abstract

Integrate Bitbucket Cloud to scan for secrets, IaC misconfigurations, vulnerabilities, and license compliance to strengthen your VCS security posture.

Integrate Cortex Cloud Application Security with your Bitbucket Cloud version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

How to integrate Bitbucket Cloud

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- In **Bitbucket**, grant the user performing the Cortex application authorization the following permissions. The level of access required depends on the modules you intend to use:
 - For **code scanning**: The user must have Write access:
 - **Workspace group with default repository access:** Add the user to a workspace group whose default repository access is set to Write
 - **Repository permissions:** Ensure the user has Write permissions on each repository that the Cortex application needs to access: Go to Bitbucket > Repository Settings and grant the user write access to the relevant repositories
 - For **CI/CD security module**: The user requires **Administrator** permissions for both **Projects** and **Repositories**

NOTE:

If you intend to use CI/CD security, you must grant Administrator access now to prevent integration errors later.

For more information on Bitbucket Cloud permissions refer to the Bitbucket Authentication methods documentation.

- **Scope:** The Cortex application requires the following authorization scopes:

[Read more...](#)

- **project**: Provides access to view the project or projects. This scope implies the repository scope, giving read access to all the repositories in a project or projects
- **repository**: Provides read access to a repository or repositories. Note that this scope does not give access to a repository's pull requests. Includes 'access to the repo's source code', 'clone over HTTPS', 'access the file browsing API', 'download zip archives of the repo's contents', 'the ability to view and use the issue tracker on any repo (created issues, comment, vote, etc)', 'the ability to view and use the wiki on any repo (create/edit pages)'
- **repository:write**: Provides write (not admin) access to a repository or repositories. No distinction is made between public and private repositories. This scope implicitly grants the **repository** scope, which does not need to be requested separately. This scope alone does not give access to the pull requests API. Includes 'push access over HTTPS' and 'fork repos'
- **pullrequest**: Provides read access to pull requests. This scope implies the repository scope, giving read access to the pull request's destination repository. Includes 'see and list pull requests', 'create and resolve tasks' and 'comment on pull requests'
- **pullrequest:write**: Implicitly grants the **pullrequest** scope and adds the ability to create, merge and decline pull requests. This scope also implicitly grants the **repository:write** scope, giving write access to the pull request's destination repository. This is necessary to allow merging. Includes 'merge pull requests', 'decline pull requests', 'create pull requests' and 'approve pull requests'
- **issue**: The ability to interact with issue trackers the way non-repo members can. This scope doesn't implicitly grant any other scopes and doesn't give implicit access to the repository. Includes 'view, list and search issues', 'create new issues', 'comment on issues', 'watch issues' and 'vote for issues'
- **issue:write**: This scope implicitly grants the issue scope and adds the ability to transition and delete issues. This scope doesn't implicitly grant any other scopes and doesn't give implicit access to the repository. Includes 'transition issues' and 'delete issues'
- **webhook**: Gives access to webhooks. This scope is required for any webhook-related operation.

This scope gives read access to existing webhook subscriptions on all resources the authorization mechanism can access, without needing further scopes. For example, a client can list all existing webhook subscriptions on a repository. The repository scope is not required. Existing webhook subscriptions for the issue tracker on a repo can be retrieved without the issue scope. All that is required is the webhook scope.

To create webhooks, the client will need read access to the resource. For example, for issue:created, the client will need to have both the webhook and the issue scope. Includes 'list webhook subscriptions on any accessible repository, user, team, or snippet' and 'create/update/delete webhook subscriptions'

- **snippet**: Provides read access to snippets. No distinction is made between public and private snippets (public snippets are accessible without any form of authentication). Includes 'view any snippet' and 'create snippet comments'
- **email**: Ability to see the user's primary email address. This should make it easier to use Bitbucket Cloud as a login provider for apps or external applications
- **account**: When used for:
 - **user-related APIs**: Gives read-only access to the user's account information. Note that this doesn't include any ability to change any of the data. This scope allows you to view the user's: email addresses, language, location, website, full name, SSH keys, user groups
 - **workspace-related APIs**: Grants access to view the workspace's: users, user permissions, projects
- **pipeline**: Gives read-only access to pipelines, steps, deployment environments and variables

- **pipeline:write**: Gives write access to pipelines. This scope allows a user to: stop pipelines, rerun failed pipelines, resume halted pipelines and trigger manual pipelines
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On the Cortex Cloud console.

- Select Settings → Data Sources & Integrations → + Add New.
- Enter Bitbucket Cloud in the search bar → Hover over the displayed search result → Connect.
Select Add Another Instance if the data source instance has already been configured.
- Click Authorize on the Configure account step of the Bitbucket Cloud onboarding wizard.

You are redirected to Bitbucket Cloud to authorize Cortex Cloud Application Security access.

2. Authorize Cortex Cloud Application Security on Bitbucket Cloud: Review the requested permissions and then select Grant access.

You are redirected to the Select Repositories step of the integration wizard.

3. Choose the repositories to be connected to the instance:

- Permit all existing repositories
- Permit all existing and future repositories
- Select Choose from repository list and select repositories from the list

4. Select Save to confirm the repository selection and then Close on the final step of the wizard.

NOTE:

Ensure that you receive the Instance Successfully Created message on this step, indicating successful instance creation.

5. Verify integration and confirm that the your integrated Bitbucket Cloud instance has a status of Connected.

- On Data Sources, search for Bitbucket Cloud in the search bar.
- Hover over the resulting entry and click View More.
- Verify that the status of your Bitbucket Cloud instance is Connected.

6. Next step: View repository assets and mitigate detected issues.

NOTE:

To create an additional Bitbucket Cloud instance: Hover over the Bitbucket Cloud card in the catalog and click Connect Another.

[Manage Bitbucket Cloud integrations](#)

To manage Bitbucket Cloud integrations, refer to Manage data source integrations.

Subscribed events

Below is a comprehensive list of events to which Cortex Cloud Application Security is subscribed (excluding events for the CI/CD module - see below). These events encompass various actions and changes occurring within your Bitbucket Cloud environment that trigger notifications and integrations with Cortex Cloud Application Security.

Read more...

- **repo:push:** This event is triggered whenever a push operation occurs within a repository, indicating that new commits have been added or existing commits have been updated
- **repo:fork:** This event occurs when a repository is forked, creating a copy of the original repository within the same or a different workspace
- **repo:updated:** This event is triggered when there are updates or changes made to the repository settings or configuration
- **repo:commit_comment_created:** This event occurs when a new comment is created on a commit within the repository
- **repo:commit_status_created:** This event is triggered when a new status or check is created for a commit within the repository
- **repo:commit_status_updated:** This event occurs when the status or check of a commit within the repository is updated
- **issue:created:** This event is triggered when a new issue is created within the repository
- **issue:comment_created:** This event occurs when a new comment is added to an existing issue within the repository
- **issue:updated:** This event is triggered when an existing issue within the repository is updated or modified
- **pullrequest:created:** This event occurs when a new pull request is created within the repository
- **pullrequest:updated:** This event is triggered when an existing pull request within the repository is updated or modified
- **pullrequest:fulfilled:** This event occurs when a pull request is fulfilled or merged into the target branch
- **pullrequest:rejected:** This event is triggered when a pull request is rejected or closed without being merged

Troubleshooting Instance Path Errors

If your VCS instance shows an error with the message **Path was not approved in the egress**, you must ensure that your VCS organization's path is approved in the Cortex Gateway. For more information, refer to Egress Configurations.

2.1.4 | Bitbucket Data Center

Integrate Cortex Cloud Application Security with your Bitbucket Data Center version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

Supported versions: This integration supports Bitbucket Data Center and Data Center Server versions 6.7 and later.

How to integrate Bitbucket Data Center

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- In **Bitbucket**, grant the user performing the Cortex application authorization the following permissions:
 - **Administrator** permissions for projects
 - **Administrator** permissions for repositories
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On Bitbucket Server, create and copy a Personal Access Token (PAT).

- a. Navigate to Bitbucket Server → Manage account → Account settings → Personal access tokens.
- b. Provide a token name.
- c. Select the Permissions scope.
 - Projects: Administrator permissions
 - Repositories: Administrator permissions

NOTE:

- By default, the permissions of the access token are set according to your current access level. It is essential to define two levels of permissions, Project and Repository permissions. The Repository permissions inherit from Project permissions, requiring Repository permissions to match or exceed Project permissions
- Providing read and write permissions to the necessary repositories enables Cortex Cloud Application Security to copy files for scanning and access repository settings. This enables automated responses to pull requests, including creating fix PRs and adding comments

d. Select the Expire automatically option.

NOTE:

For additional security, it is recommended to set an expiry automatically. The expiry date of a token cannot be changed after it is created. You can see the expiry dates for all your tokens on Profile picture → Manage account → Personal access tokens.

e. Click Create.

f. Copy the generated token from the dialog.

IMPORTANT:

Always refer to the Bitbucket documentation for information relating to creating a PAT.

2. On the Cortex Cloud console.

a. Navigate to Settings → Data Sources & Integrations → + Add New.

b. Enter Bitbucket Data Center in the search bar.

c. Hover over the Bitbucket Data Center card in the catalog and click Connect.

Select Add Another Instance if the data source instance has already been configured.

d. Enter your domain in the Configure Domain step of the wizard and click Next.

e. Optional: Connect a Transporter: Select your Broker VM and associated Transporter applet from the provided menus.

NOTE:

For more information about the Transporter, including setup instructions, refer to Transporter over Broker VM.

f. Click Next.

g. On the Create a Personal Access Token step of the wizard: Paste the Bitbucket PAT generated in **step 1** above in the provided field, and click Next.

h. Under Selection Options of the Select Repositories step of the wizard:

- Choose the repositories to be connected to the instance:
 - Permit all existing repositories
 - Permit all existing and future repositories
 - Select Choose from repository list and select repositories from the list
- Click Save.

i. Click Close on the final step of the wizard.

NOTE:

Ensure that you receive the Instance Successfully Created message on this step, indicating successful instance creation.

3. Verify integration and confirm that the your integrated Bitbucket Data Center instance has a status of Connected.

a. On the Data Sources & Integrations page, search for Bitbucket Data Center in the search bar.

b. Hover over the resulting entry and click View More.

c. Verify that the status of your Bitbucket Data Center instance is Connected.

4. Next step: View repository assets and mitigate detected issues.

NOTE:

To create an additional Bitbucket Data Center instance: Hover over the Bitbucket Data Center card in the catalog and click Connect Another.

Manage Bitbucket Data Center integrations

To manage Bitbucket Data Center integrations, refer to Manage data source integrations.

Rotate integration tokens

Rotate integration tokens to enhance security and prevent unauthorized access.

Create a PUT request: `PUT /code/api/v1/integration/token/&<integration_id>` with the following body:

```
{  
  "token": "new token"  
}
```

To locate your integration ID:

1. Under Cortex Cloud Application Security select Settings → Data Sources & Integrations.
2. Hover over Bitbucket Data Center and click View Details.
3. Select the required instance from the list and retrieve the cas_connector_id from the URL.

Subscribed events

Below is a comprehensive list of events to which Cortex Cloud Application Security is subscribed. These events encompass various actions and changes occurring within your Bitbucket Data Center environment that trigger notifications and integrations with Cortex Cloud Application Security.

Read more...

- **pr:merged**: This event occurs when a pull request is successfully merged into the repository
- **pr:updated**: This event happens when the reviewer list for a pull request is updated
- **pr:opened**: This event occurs when a new pull request is opened
- **repo:added**: This event happens when a comment is added to the repository
- **repo:forked**: This event occurs when a repository is forked
- **repo:refs_changed**: This event happens when references in the repository are changed
- **repo:edited**: This event occurs when a comment in the repository is edited
- **pr:declined**: This event occurs when a pull request is declined
- **pr:deleted**: This event happens when a pull request is deleted
- **pr:comment_deleted**: This event occurs when a comment on a pull request is deleted
- **repo:comment_deleted**: This event happens when a comment in the repository is deleted
- **pr:edited**: This event occurs when a comment on a pull request is edited
- **pr:unapproved**: This event happens when a reviewer unapproves a pull request
- **pr:modified**: This event occurs when a pull request is modified
- **mirror:repo_synchronized**: This event occurs when a mirrored repository is synchronized
- **pr:needs_work**: This event happens when a reviewer marks a pull request as needing work
- **pr:approved**: This event occurs when a reviewer approves a pull request
- **repo:modified**: This event occurs when the repository is modified
- **pr:added**: This event occurs when a comment is added to a pull request

Subscribed events for the CI/CD module

These events are specific to the CI/CD module to which Cortex Cloud is subscribed. They encompass various actions and changes occurring within your CI/CD environment that trigger notifications and integrations with Cortex Cloud.

[Read more...](#)

Event	Description
Project: proj:modified	This event occurs when a project undergoes modifications, such as changes to its name, description, or configuration settings.

Event	Description
Repository: repo:refs_changed	This event occurs when a push operation is performed, typically resulting in changes to the repository's references.
Repository: repo:forked	This event occurs when a repository is forked, creating a separate copy of the repository under a different user or organization.
Repository: repomodified	This event occurs when the repository itself undergoes modifications, such as changes to its settings or configuration.
Repository: repoadded	This event occurs when a new comment is added to a commit within the repository.
Repository: repoedited	This event occurs when an existing comment on a commit is edited within the repository.
Repository: repodeleted	This event occurs when a comment on a commit is deleted within the repository.
Pull Request: pr:opened	This event occurs when a pull request is opened, indicating the initiation of a request to merge changes into the repository.
Pull Request: pr:from_ref_updated	This event occurs when the source branch of a pull request is updated with new changes.
Pull Request: pr:to_ref_updated	This event occurs when the target branch of a pull request is updated with new changes.
Pull Request: pr:modified	This event occurs when a pull request undergoes modifications, such as changes to its title, description, or metadata.

Event	Description
Pull Request: prupdated	This event occurs when the list of reviewers assigned to a pull request is updated.
Pull Request: prapproved	This event occurs when a reviewer approves a pull request.
Pull Request: prunapproved	This event occurs when a previously approved review on a pull request is revoked.
Pull Request: prneeds	This event occurs when a reviewer requests changes to be made to a pull request before it can be approved.
Pull Request: prmerged	This event occurs when a pull request is successfully merged into the repository.
Pull Request: prdeclined	This event occurs when a pull request is declined or rejected, typically due to not meeting certain criteria or requirements.
Pull Request: prdeleted	This event occurs when a pull request is deleted, either intentionally by a user or automatically due to certain conditions.
Pull Request: pradded	This event occurs when a new comment is added to a pull request.
Pull Request: predited	This event occurs when an existing comment on a pull request is edited.
Pull Request: prdeleted	This event occurs when a comment on a pull request is deleted.

2.1.5 | GitHub Cloud

Integrate Cortex Cloud Application Security with your GitHub SaaS version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

How to integrate GitHub SaaS

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- In GitHub, grant the user performing the Cortex application authorization the following permissions:
 - **Organization Owner:** Only an Organization Owner can directly authorize and install the application
- **Scope:** The Cortex application requires the following authorization scopes:
 - **Read** access to Dependabot alerts, actions, actions variables, administration, deployments, discussions, metadata, packages, repository hooks, secret scanning alerts, secrets, and security events
 - **Read and write** access to checks, code, commit statuses, issues, and pull requests

NOTE:

In contrast to GitLab SaaS , GitLab Self Managed (On-Prem) and Azure Repos, there is no individual record of each token used for authentication on the integrations page. However, Cortex Cloud Application Security retains and uses these tokens for necessary actions. Removing an integration will delete all associated tokens.

- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On the Cortex XSIAM console.

- a. Select Settings → Data Sources & Integrations → + Add New.
- b. Enter GitHub (SaaS) in the search bar.
- c. Hover over GitHub (SaaS) card in the catalog and click Connect.

Select Add Another Instance if the data source instance has already been configured.

- d. Click Authorize on the Configure account step of the GitHub SaaS onboarding wizard.

You are redirected to your GitHub SaaS account in order to install and authorize Cortex AppSec), the GitHub App application handling the Cortex Cloud Application Security functionality.

2. Install and authorize Cortex AppSec on GitHub SaaS.

- a. Select your organization on which will be installed.
- b. Select the repositories to be authorized.
- c. Review the permissions granted the application.

d. Click Install & Authorize.

You are redirected to the Select Repositories step of the GitHub SaaS installation wizard on the console.

Refer to the GitHub documentation for more on authorizing and installing GitHub SaaS Apps.

3. On the Cortex XSIAM console.

a. Under Selection Options, choose the repositories to be connected to the instance:

- Permit all existing repositories
- Permit all existing and future repositories
- Select Choose from repository list and select repositories from the list

b. Click **Save**.

4. Verify integration: On Data Sources, select Code Providers → GitHub SaaS → View more and confirm that the status of your integrated GitHub instance is 'Connected'.

5. Verify integration and confirm that the your integrated GitHub SaaS instance has a status of Connected.

a. On Data Sources, search for GitHub SaaS in the search bar.

b. Hover over the resulting entry and click View More.

c. Verify that the status of your GitHub SaaS instance is Connected.

6. View repository assets and mitigate detected issues.

NOTE:

To create an additional GitHub SaaS instance: Hover over the GitHub SaaS card in the catalog and click Connect Another.

[Manage GitHub SaaS integrations](#)

To manage GitHub SaaS integrations, refer to Manage data source integrations.

[Subscribed events](#)

Below is a comprehensive list of events to which Cortex Cloud Application Security is subscribed. These events encompass various actions and changes occurring within your GitHub SaaS environment that trigger notifications and integrations with Cortex Cloud Application Security.

[Read more...](#)

Event	Description
Create	Indicates the creation of a branch or tag

Event	Description
Commit comment	Refers to comments made on a commit or a 'diff' comment, which compares changes within a commit
Issues	Includes a wide range of actions such as opening, editing, deleting, transferring, pinning, unpinning, closing, reopening, assigning, unassigning, labeling, unlabeling, milestone management (milestoned, demilestoned), and locking or unlocking an issue
Public	Denotes changes made to a repository from private to public
Pull request	Represents actions related to pull requests, including assignment, enabling or disabling auto merge, closing, conversion to draft, demilestoning, dequeuing, editing, enqueueing, labeling, locking, milestone assignment, opening, readiness for review, reopening, removal of review requests, request for review, synchronization, unassignment, unlabeling, and unlocking
Pull request review comment	Indicates the creation, editing, or deletion of a comment on a pull request's diff
Push	Refers to a Git push operation performed on a repository
Repository	Includes actions such as creation, deletion, archiving, unarchiving, publicizing, privatizing, editing, renaming, or transferring of a repository

Troubleshooting Instance Path Errors

If your VCS instance shows an error with the message **Path was not approved in the egress**, you must ensure that your VCS organization's path is approved in the Cortex Gateway. For more information, refer to Egress Configurations.

2.1.6 | GitHub Enterprise (On-Prem)

Integrate Cortex Cloud Application Security with your GitHub Enterprise (On-Prem) version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

[How to integrate GitHub Enterprise \(On-Prem\)](#)

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- In **GitHub**, you must have Organization Owner permissions to install the Cortex application. Users with only repository-level admin permissions cannot complete the installation unless the organization explicitly allows non-owners to install GitHub Apps (in this instance the Cortex application)
- **Scope:** The Cortex application requires the following authorization scopes:

Read more...

- **repo:** Grants full access to public and private repositories, including read and write access to code, commit statuses, repository invitations, collaborators, deployment statuses, and the capability to subscribe the repository to receive new webhook notifications or events

NOTE:

In addition to repository-related resources, the repository scope also grants access to manage organization-owned resources, including projects, invitations, team memberships, and webhooks. This scope also grants the ability to manage projects owned by users

- **read:user:** Grants access to read a user's profile data
 - **read:repo_hook:** Grants read and ping access to hooks in public or private repositories
 - **read:org:** Provides read-only access to organization membership, organization projects, and team membership
 - **read:public_key:** Allows listing and viewing details for public keys
 - **workflow:** Provides the ability to add and update GitHub Actions workflow files. Workflow files can be committed without this scope if the same file (with both the same path and contents) exists on another branch in the same repository. Workflow files can expose GITHUB_TOKEN, which may have a different set of scopes. For more information, refer to the GitHub Actions Automatic token authentication token authentication documentation
 - **admin:org_hook:** Grants read, write, ping, and delete access to organization hooks. Note: OAuth tokens will only be able to perform these actions on organization hooks created by the OAuth app. Personal access tokens will only be able to perform these actions on organization hooks created by a user
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

- To enable access from your environment to your Cortex Cloud tenant, add the applicable Cortex IP addresses to your allow list. This ensures that your network can receive inbound connections from Cortex Cloud when required.

Egress proxy IPs

Add the egress proxy IP addresses for your specific region to your allow list as follows.

Read more...

<ul style="list-style-type: none"> ◦ AU (Australia) <ul style="list-style-type: none"> ▪ 34.151.83.236 ▪ 34.116.67.90 ◦ BR (Brazil) <ul style="list-style-type: none"> ▪ 34.151.223.178 ▪ 34.39.232.219 ◦ CA (Canada) <ul style="list-style-type: none"> ▪ 35.203.108.13 ▪ 35.203.101.16 ◦ CH (Switzerland) <ul style="list-style-type: none"> ▪ 34.65.108.153 ▪ 34.65.155.169 ◦ DE (Germany) <ul style="list-style-type: none"> ▪ 35.234.118.195 ▪ 34.89.183.45 ◦ EU (Europe) <ul style="list-style-type: none"> ▪ 34.147.107.51 ▪ 34.91.26.125 ◦ IN (India) <ul style="list-style-type: none"> ▪ 35.200.175.78 ▪ 34.93.9.198 ◦ ID (Indonesia) <ul style="list-style-type: none"> ▪ 34.128.126.138 ▪ 34.128.82.158 	<ul style="list-style-type: none"> ◦ DL (Delhi) <ul style="list-style-type: none"> ▪ 34.131.41.243 ▪ 34.131.45.169 ◦ FA (France) <ul style="list-style-type: none"> ▪ 34.155.5.117 ▪ 34.155.41.247 ◦ IL (Israel) <ul style="list-style-type: none"> ▪ 34.165.33.165 ▪ 34.165.27.131 ◦ IT (Italy) <ul style="list-style-type: none"> ▪ 34.154.23.156 ▪ 34.154.186.12 ◦ JP (Japan) <ul style="list-style-type: none"> ▪ 35.200.3.131 ▪ 34.146.181.233 ◦ PL (Poland) <ul style="list-style-type: none"> ▪ 34.118.48.171 ▪ 34.116.202.235 ◦ PR (Puerto Rico) <ul style="list-style-type: none"> ▪ 35.224.117.2 ▪ 34.173.28.243 ◦ QT (Qatar) <ul style="list-style-type: none"> ▪ 34.18.34.118 ▪ 34.18.39.155 	<ul style="list-style-type: none"> ◦ SA (Saudi Arabia) <ul style="list-style-type: none"> ▪ 34.166.61.81 ▪ 34.166.58.213 ◦ SG (Singapore) <ul style="list-style-type: none"> ▪ 35.240.243.57 ▪ 34.126.183.208 ◦ ZA (South Africa) <ul style="list-style-type: none"> ▪ 34.35.42.196 ▪ 34.35.79.219 ◦ KR (South Korea) <ul style="list-style-type: none"> ▪ 34.64.93.168 ▪ 34.64.237.45 ◦ ES (Spain) <ul style="list-style-type: none"> ▪ 34.175.46.46 ▪ 34.175.80.182 ◦ TW (Taiwan) <ul style="list-style-type: none"> ▪ 34.80.133.68 ▪ 35.234.18.10 ◦ UK (United Kingdom) <ul style="list-style-type: none"> ▪ 35.242.180.163 ▪ 34.105.173.229 ◦ US (United States) <ul style="list-style-type: none"> ▪ 34.132.108.184 ▪ 34.69.63.16
---	--	--

1. On the Cortex Cloud console.

a. Navigate to Settings → Data Sources & Integrations → + Add New.

- b. In the search bar, input GitHub → hover over GitHub Enterprise (On-Prem) card in the catalog → Connect.

Select Add Another Instance if the data source instance has already been configured.

- c. Enter your domain in the Configure Domain step of the wizard.

NOTE:

The domain is the hostname associated with your GitHub Enterprise (On-Prem) instance.

- d. Optional: Connect a Transporter: Select your Broker VM and associated Transporter applet from the provided menus.

NOTE:

For more information about the Transporter, including setup instructions, refer to [Transporter over Broker VM](#).

- e. Click Register.

You are redirected to your GitHub Enterprise (On-Prem) instance to register Cortex AppSec as an OAuth application. Additionally, the Register OAUTH App step of the integration wizard is displayed.

- f. Copy the Application Name, Homepage URL and Authorization Callback URL values from their respective fields.

2. On the Register a new OAuth application screen of the GitHub Enterprise (On-Prem) console:

- a. Paste the values copied in **step 1d** above in their respective fields.

- b. Click Register application.

- c. Once created, copy and save the Client ID and Client Secret values for the new Cortex AppSec application.

Click Authorize to complete the setup.

3. On the Cortex Cloud console.

- a. Select Next on the the Register OAUTH App step of the integration wizard.

The Set Client ID and Secret step of the wizard is displayed.

- b. Paste the Client ID and Client Secret values copied in **step 2c** above, and click Authorize.

- c. Under Selection Options of the Select Repositories step of the wizard, choose the repositories to be connected to the instance:

- Permit all existing repositories
- Permit all existing and future repositories
- Choose from repository list and select repositories from the list

- d. Click Save.

- e. Click Close on the final step of the wizard.

NOTE:

Ensure that you receive the Instance Successfully Created message on this step, indicating successful instance creation.

4. Verify integration: On the Data Sources & Integrations page, select Code ProvidersGitHub Enterprise (On-Prem), confirm the integration status as Connected.

5. View repository assets and mitigate detected issues.

NOTE:

To create an additional GitHub Enterprise (On-Prem) instance: Hover over the GitHub Enterprise (On-Prem) card in the catalog and click Connect Another.

[Manage GitHub Enterprise \(On-Prem\) integrations](#)

To manage GitHub Enterprise (On-Prem) integrations, refer to Manage data source integrations.

[Subscribed events](#)

The following list describes events that Cortex Cloud Application Security monitors on your GitHub Enterprise (On-Prem), covering actions and changes that trigger notifications and integrations.

- **Repository** events: All events related to repositories
- **Organization** events: Includes ['organization', 'membership', 'team'] events

2.1.7 | GitLab SaaS

Integrate Cortex Cloud Application Security with your GitLab SaaS version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

[How to integrate GitLab SaaS](#)

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- In Gitlab, the following permissions are required to integrate the application:
 - **Maintainer** (Project-level). Grants sufficient permissions to configure external integrations, manage repository access, and adjust CI/CD settings
 - **Administrator** (Repository-level): Required to scan pull requests (PRs). This enables Cortex Cloud to set up subscription webhooks for the selected repositories
- **Scope:** The Cortex application requires the following authorization scope:
 - **api**: Grants full read and write access to the API, including all groups and projects, as well as permissions to interact with the container registry, the dependency proxy, and the package registry
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On the Cortex Cloud console.

- a. Navigate to Settings → Data Sources & Integrations → + Add New.
- b. Hover over GitLab (Code Scanning) and click Connect.
- c. Click Authorize on the Configure account step of the GitLab SaaS onboarding wizard.

You are redirected to your GitLab SaaS account in order to install and authorize Cortex AppSec, the GitLab App application handling the Cortex Cloud Application Security functionality.

2. On GitLab SaaS: Review the requested permissions and click Authorize Cortex AppSec.

You are redirected to the Select Repositories step of the installation wizard on the console.

3. On the Cortex Cloud console.

- a. Under Selection Options, choose the repositories to be connected to the instance:
 - Permit all existing repositories
 - Permit all existing and future repositories
 - Choose from repository list → select repositories from the list
- b. Click Save.

NOTE:

A repository can only be integrated with a single instance. The first instance that connects with the repository will be the one that the repository is assigned to. This means that if multiple integrations attempt to connect to the same repository, only the first integration to establish the connection will be associated with that repository.

4. Verify integration and confirm that the your integrated GitLab SaaS instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for GitLab SaaS in the search bar.
- b. Hover over the resulting entry and click View More.
- c. Verify that the status of your GitLab SaaS instance is Connected.

5. View repository assets and mitigate detected issues.

NOTE:

To create an additional GitLab SaaS instance: Hover over the GitLab SaaS card and click in the catalog and Connect Another.

Manage GitLab SaaS integrations

To manage GitLab SaaS integrations, see Manage VCS instances.

Subscribed events

Below is a comprehensive list of events to which Cortex Cloud Application Security is subscribed. These events encompass various actions and changes occurring within your GitLab SaaS environment that trigger notifications and integrations with Cortex Cloud Application Security:

Read more...

Category	Event	Description
Projects	c	—
—	merge_requests_events	This event is triggered when merge or pull requests are created, updated, merged, closed, or have changes made to them
—	push_events	This event occurs whenever code changes are pushed to a repository, indicating new commits being added to the version control history
—	tag_push_events	This event is triggered when new tags are pushed to a repository
—	note_events	This event is generated when comments or notes are added to various objects within GitLab, such as issues, merge requests, or commits
—	confidential_note_events	Similar to note_events, but specifically for confidential comments or notes that are restricted to certain users or groups

Category	Event	Description
—	issues_events	This event is triggered when issues are created, updated, closed, or have changes made to them
—	confidential_issues_events	Similar to issues_events, but specifically for confidential issues that are restricted to certain users or groups
—	job_events	This event occurs when jobs defined in CI/CD pipelines are created, updated, started, finished, or have changes made to them
—	pipeline_events	This event is generated when pipelines are created, updated, started, finished, or have changes made to them
—	wiki_page_events	This event occurs when changes are made to wiki pages within GitLab, including creation, updates, and deletions
—	deployment_events	This event is triggered when deployments are created, updated, started, finished, or have changes made to them
—	releases_events	This event occurs when releases are created, updated, published, or have changes made to them
Groups	—	—
—	subgroup_events	This event is specific to GitLab groups and occurs when changes are made to subgroups within a group hierarchy

Troubleshooting Instance Path Errors

If your VCS instance shows an error with the message **Path was not approved in the egress**, you must ensure that your VCS organization's path is approved in the Cortex Gateway. For more information, refer to Egress

Configurations.

2.1.8 | GitLab Self Managed (On-Prem)

Integrate Cortex Cloud Application Security with your GitLab Self Managed (On-Prem) version control system (VCS) to enable security scans for exposed secrets, infrastructure-as-code (IaC) misconfigurations, vulnerabilities, package operational risks, and license compliance issues in your repositories. This integration allows you to analyze, prioritize, and resolve detected issues efficiently.

How to integrate GitLab Self Managed (On-Prem)

PREREQUISITE:

- Authorize the user integrating Cortex Cloud Application Security with your GitLab Self Managed (On-Prem) instances with the following permissions:
 - **Maintainer** permissions. Grants sufficient permissions to configure external integrations, manage repository access, and adjust CI/CD settings
 - **api**: Grants full read and write access to the API, including all groups and projects, as well as permissions to interact with the container registry, the dependency proxy, and the package registry
 - **Administrator repository permissions**: In order to scan pull requests (PRs), the user performing the integration must have administrative privileges for the repositories. This enables Cortex Cloud Application Security to set up subscription webhooks for the selected repositories
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On the Cortex Cloud console.

- a. Navigate to Settings → Data Sources & Integrations → + Add New.
- b. Hover over GitLab Self Managed (On-Prem) and click Connect.
- c. Enter your domain in the Configure Domain step of the wizard and click Register.

NOTE:

The domain is the hostname associated with your GitLab Self Managed (On-Prem) instance.

You are redirected to your GitLab Self Managed (On-Prem) instance register Cortex AppSec as an application. Additionally, the Register OAUTH App step of the integration wizard is displayed.

- d. Optional: Connect a Transporter: Select your Broker VM and associated Transporter applet from the provided menus.

NOTE:

For more information about the Transporter, including setup instructions, refer to Transporter over Broker VM.

- e. Copy the Application Name, Homepage URL and Authorization Callback URL values from their respective fields.

2. On the GitLab Self Managed (On-Prem) console:

- a. Access GitLab Self Managed (On-Prem) → User Settings → Applications.
- b. Paste the values copied in **step 1d** above in their respective fields.
- c. Select api as the application scope and then Save.
- d. Once created, copy and save the generated Application ID and Secret values for the new Cortex AppSec application.

3. On the Cortex Cloud console.

- a. Select Next on the Register OAUTH App step of the wizard.

The Set Client ID and Secret step of the wizard is displayed.

- b. Paste the GitLab Self Managed (On-Prem) Application ID and Secret values copied in *step 2d* above and click Next.
- c. Under Selection Options of the Select Repositories step of the wizard, choose the repositories to be connected to the instance:
 - Permit all existing repositories
 - Permit all existing and future repositories
 - Choose from repository list and select repositories from the list

d. Click Save.

e. Click Close on the final step of the wizard.

NOTE:

Ensure that you receive the Instance Successfully Created message on this step, indicating successful instance creation.

4. Verify integration and confirm that your integrated GitLab Self Managed (On-Prem) instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for GitLab Self Managed (On-Prem) in the search bar.
- b. Hover over the resulting entry and click View More.
- c. Verify that the status of your GitLab Self Managed (On-Prem) instance is Connected.

5. View repository assets and mitigate detected issues.

NOTE:

To create an additional GitLab Self Managed (On-Prem) instance: Hover over the GitLab Self Managed (On-Prem) card in the catalog and click Connect Another.

Manage GitLab Self Managed (On-Prem) integrations

To manage GitLab Self Managed (On-Prem) integrations, refer to Manage data source integrations.

Subscribed events

Below is a comprehensive list of events to which Cortex Cloud Application Security is subscribed. These events encompass various actions and changes occurring within your GitLab Self Managed (On-Prem) environment that trigger notifications and integrations with Cortex Cloud Application Security.

[Read more...](#)

Category	Event	Description
Projects	—	—
—	merge_requests_events	This event is triggered when merge or pull requests are created, updated, merged, closed, or have changes made to them
—	push_events	This event occurs whenever code changes are pushed to a repository, indicating new commits being added to the version control history
—	tag_push_events	This event is triggered when new tags are pushed to a repository
—	note_events	This event is generated when comments or notes are added to various objects within GitLab, such as issues, merge requests, or commits
—	confidential_note_events	Similar to note_events, but specifically for confidential comments or notes that are restricted to certain users or groups

Category	Event	Description
—	issues_events	This event is triggered when issues are created, updated, closed, or have changes made to them
—	confidential_issues_events	Similar to issues_events, but specifically for confidential issues that are restricted to certain users or groups
—	job_events	This event occurs when jobs defined in CI/CD pipelines are created, updated, started, finished, or have changes made to them
—	pipeline_events	This event is generated when pipelines are created, updated, started, finished, or have changes made to them
—	wiki_page_events	This event occurs when changes are made to wiki pages within GitLab, including creation, updates, and deletions
—	deployment_events	This event is triggered when deployments are created, updated, started, finished, or have changes made to them
—	releases_events	This event occurs when releases are created, updated, published, or have changes made to them
Groups	—	—

Category	Event	Description
—	subgroup_events	This event is specific to GitLab groups and occurs when changes are made to subgroups within a group hierarchy
System	repository_update_events	This event occurs whenever there are updates or changes made to a GitLab repository, including actions such as new commits, branch operations, tag updates, and modifications to repository settings

2.2 | Onboard CI/CD systems

Onboard CI/CD systems to scan for configuration threats in your organization's instance, pipelines, and individual repositories. By onboarding supported version control systems (such as GitHub and GitLab), you gain out-of-the-box CI/CD scanning capabilities. However, you must explicitly onboard CircleCI and Jenkins to enable scanning for these systems.

2.2.1 | Onboard CircleCI for CI/CD pipeline scans

Integrate Cortex Cloud Application Security CI/CD Security with your CircleCI system to enable automated and continuous scanning of your CI/CD pipelines. This integration provides proactive security checks, triggered by pipeline events or configuration changes, ensuring security issues are detected and remediated throughout the entire deployment lifecycle.

Pipeline scans are executed using the Cortex CLI, and include automated actions based on scan results to enforce security policies and prevent vulnerable deployments.

NOTE:

- This integration utilizes a Personal Access Token (PAT) for authentication
- CircleCI onboarding offers both code and CI/CD scanning. A single integrated instance supports either code or CI scanning, but not both. If you require both code and CI scanning for your CircleCI environment, you must create two separate integrations, selecting the appropriate scanning type for each. To onboard CircleCI for code scans, refer to Onboard CircleCI for code scans

PREREQUISITE:

Before you begin:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- **CircleCI user requirements:**
 - **Permissions:** To enable Cortex Cloud visibility for all CircleCI projects, a version control system (VCS) user with integration permissions must be authorized (For example, Organization Owner permissions are required to onboard GitHub SaaS, while in GitLab SaaS you must be a Maintainer). This is because CircleCI's user base integrates with the VCS, inheriting its user permissions. For example, if a GitHub user has access to specific organizations and repositories, these entities are visible and available in CircleCI
 - **Best practice:** Create a dedicated VCS user to integrate CircleCI with Cortex Cloud, to prevent the integration breaking if the user leaves the organization
 - **Ensure that the dedicated user follows all the organization's projects in CircleCI**
 - **Create a personal API token in CircleCI (see step 1 below).** This is required to allow reading the configurations from CircleCI for all projects the user has access to

1. Generate a personal API token on CircleCI.

- a. Login to your CircleCI instance with your VCS user credentials.
- b. Create and save a personal API token. For more information about CircleCI tokens, refer to <https://circleci.com/docs/managing-api-tokens/#creating-a-personal-api-token>.

2. On the Cortex Cloud console:

- a. Navigate to Settings → Data Sources & Integrations → + Add New.
- b. Enter CircleCI in the search bar → hover over the displayed search result → Connect.
- c. On the Select Integration step of the integration wizard, select CI/CD System Scan → Next.
- d. On the Enable CI/CD system scanning step of the integration wizard:
 - i. Enter an instance name: This can be any name you choose; it serves as an alias for your integration.
 - ii. Paste the CircleCI personal API token that you generated in step 1 above → Done.
- e. Verify that the Instance Successfully Created message is displayed in the last step of the wizard and click Close.

3. Verify integration and confirm that the your integrated CircleCI instance has a status of Connected.

- a. On the Data Sources & Integrations page, enter CircleCI in the search bar.
- b. Hover over the resulting entry and click [number of instances] Configured.

Verified that your connected CircleCI instance displays a Connected status.

- c. Hover over the CircleCI instance and select View Details.
- d. Verify that the status of your CircleCI instance is Connected and that Pipeline Risks is the instance type.

4. Next step: View scan results and mitigate issues.

NOTE:

To add an additional CircleCI instance, navigate to Settings → Data Sources & Integrations → , select the connected CircleCI integration, click → Add Instance, and repeat the onboarding steps above.

2.2.2 | Onboard Jenkins for CI/CD pipeline scans

Integrate Cortex Cloud Application Security CI/CD Security with your Jenkins servers to enable automated and continuous scanning of your CI/CD pipelines. This integration provides proactive security checks, triggered by pipeline events or configuration changes, ensuring security issues are detected and remediated throughout the entire deployment lifecycle.

Pipeline scans are executed using the Cortex CLI, and include automated actions based on scan results to enforce security policies and prevent vulnerable deployments.

NOTE:

Jenkins onboarding offers both code and CI/CD scanning. A single integrated instance supports either code or CI scanning, but not both. If you require both code and CI scanning for your Jenkins servers, you must create two separate integrations, selecting the appropriate scanning type for each. To onboard Jenkins for code scans, refer to Onboard Jenkins for code scans.

PREREQUISITE:

Prerequisite

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- **In Jenkins:**
 - To install and configure the Cortex plugin in Jenkins, you must be a Jenkins Administrator with **Overall/Administer** permissions
 - Ensure the build server allows outbound HTTPS (Port 443 traffic to the Cortex API URL

1. On the Cortex Cloud console:

- a. Select Settings → Data Sources & Integrations → + Add New.
- b. Enter Jenkins in the search bar → hover over the displayed search result → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

2. On the Select Integration step of the wizard, select CI/CD System Scan → Next.
3. On the Create Instance step of the integration wizard: Provide a Jenkins plugin connector name → Next.
4. On the Plugin installation step of the wizard:
 - a. Click Download to download the Cortex Cloud Application Security Jenkins **Cortex.Cloud.hpi** plugin file.
 - b. Copy and save the generated **JWT** token.
 - c. Click Done.

NOTE:

The integration is added on the console but integration is pending, and will only be completed after completing step 5 below. You can view the pending integration on the Jenkins Instances page: Select Data Sources → Jenkins → View Details. The type of integration is Pipeline Risks

5. Install and configure the Cortex Cloud plugin on your Jenkins server:

- a. Open Jenkins and select: Manage Jenkins → Plugins (under System Configuration) → Advanced settings.
- b. Select Choose File (under the Deploy Plugin section) → browse for the Cortex.Cloud Plugin.hpi file → Upload → Deploy.
- c. Configure the plugin:
 - i. Open Jenkins → Select Manage Jenkins → System (under System Configuration) → Cortex Cloud.
 - ii. Fill in the provided fields:
 - Cortex JWT Token: Paste the JWT token copied in *step 4b* above.
 - Cortex Reports Recurrence Period (Value: minutes): The frequency with which reports are generated. We recommend that you do not change the default value
 - iii. Click Save.

The Cortex Cloud Application Security plugin is integrated with your Jenkins system.

NOTE:

Always refer to the official Jenkins documentation when installing plugins on Jenkins servers.

NOTE:

To add an additional Jenkins instance, navigate to Settings → Data Sources → select the menu for your connected Jenkins instance → + New Instance, and repeat the onboarding steps above.

2.3 | Integrate CI tools

By integrating CI tools, you get two main benefits: code scans and streamlined security workflows. This is achieved by inserting code snippets directly into your existing CI workflows, which then use the Cortex CLI to trigger automated security checks. This integration enables the platform to scan and detect exposed secrets, misconfigurations in your infrastructure-as-code (IaC) files, vulnerabilities in your Software Composition Analysis (SCA) packages and license non-compliance within your CI/CD pipelines.

You can integrate your CI tools and systems through the platform wizard or by directly adding a code snippet to your pipelines in supported systems.

Integrate CI tools via the tenant UI wizard

Cortex Cloud Application Security supports the following CI tools for onboarding via the UI wizard:

- AWS CodeBuild
- CircleCI for code scans (For CircleCI CI/CD pipeline scans, refer to CI/CD)
- Cortex CLI. For information about using the Cortex CLI, refer to Cortex CLI
- GitHub Actions
- Jenkins for code scans (For Jenkins CI/CD pipeline scans, refer to CI/CD)
- Terraform Cloud (Run Tasks)
- Terraform Enterprise (Run Tasks)

Manage CI Tools

To access CI tool management, navigate to Settings → Data Sources & Integrations → hover over a CI tool → View Details.

You can perform the following actions on CI tools:

- **Delete an instance:** Right-click on an instance of the CI tool → Delete instance → Delete
- **Remove a connected repository:** Select an instance of the CI → right-click on a repository → Remove Repository
- **Select the repository branches** to be scanned: Select an instance of the CI → right-click on a repository → Set Scanned Branches → select a branch/multiple branches → Save
- Perform a **manual scan** of the repository: Select an instance of the CI → right-click on a repository → Scan Repository

2.3.1 | AWS CodeBuild

Integrate Cortex Cloud Application Security with your AWS CodeBuild instance to allow dynamic, automated, and context-specific scans within your development workflow. This includes continuous scanning of your workflow whenever changes are pushed or triggered, integrating security checks, and catching issues as soon as they are introduced. Additionally, it automates shift-left actions such as notifying developers or creating tickets, based on scan results.

PREREQUISITE:

Before you begin:

- 1. On the Cortex Cloud console:
 - a. Navigate to Settings → Data Sources & Integrations → + Add New.
 - b. Enter AWS CodeBuild in the search bar → Hover over the displayed search result → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

- c. On the Add Environment Variables step of the AWS CodeBuild integration wizard.

- i. Select Generate API key.

The API key secret and API key ID values are generated and populate their respective fields.

- ii. Select the system architecture that your tool runs on.

- iii. Click Next.

2. Store your generated Cortex Cloud API key and API key ID in AWS Secrets Manager.

- If you have an API key:

1. Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID variable names from their respective fields in the wizard.
2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables (secrets) to the AWS Secrets Manager.

- If you do not have an API key:

1. Click Generate API key → Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values from their respective fields.
2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables (secrets) to the AWS Secrets Manager.

NOTE:

Do not change the names of the environment variables provided by Cortex Cloud. They are required for proper integration and functionality.

For more information on storing secrets in AWS Secrets Manager, refer to AWS Secrets Manager Documentation.

3. Grant the **IAM service role** associated with your AWS CodeBuild project the necessary permissions to read the Cortex Cloud API key and Cortex Cloud API key ID from AWS Secrets Manager.
4. Copy and paste the pre-populated sample code from the Configure Subscription step of the integration wizard into your **buildspec.yaml** configuration.

NOTE:

The code is only a reference. Replace the placeholder values with your build-specific values.

5. Select Done in the wizard.
6. Ensure that the **Connector Created Successfully** message is displayed in the final step of the wizard, and click **Close**.
7. Verify integration and confirm that the your integrated AWS CodeBuild instance has a status of Connected.
 - a. On the Data Sources & Integrations page, search for AWS CodeBuild in the search bar.
 - b. Hover over the resulting entry and click View Details.
 - c. Verify that the status of your AWS CodeBuild instance is Connected.

8. Next step: View scan results and mitigate issues.

NOTE:

To add an additional AWS CodeBuild instance, navigate to the Settings → Data Sources & Integrations → page, select the connected AWS CodeBuild integration, click → Add Instance, and repeat the onboarding steps above.

AWS CodeBuild code scan workflow template

This AWS CodeBuild workflow example automates code scanning using the Cortex CLI. The workflow contains placeholder values (often in brackets) and generic terms (such as dev) that you must replace with your environment-specific information before use.

```
version: 0.2

env:
  variables:
    CORTEX_API_URL: "https://api-viso-hdkbk6qphxpbehy758elo.xdr-qa2-uat.us.paloaltonetworks.com"
    CORTEX_CLI_VERSION: "0.8.11"
  secrets-manager:
    CORTEX_API_KEY: "CORTEX_API_KEY"
    CORTEX_API_KEY_ID: "CORTEX_API_KEY_ID"

phases:
  install:
    runtime-versions:
      docker: 19
    commands:
      - echo "Installing dependencies"
      - yum -y update
      - yum -y install jq curl

  pre_build:
    commands:
      - echo "Fetching temporary token"
      - |
        export TOKEN_RESPONSE=$(curl --location "${CORTEX_API_URL}/public_api/cas/v1/cortex-cli/create-token" \
          --header "Authorization: ${CORTEX_API_KEY}" \
          --header "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
          --header "Content-Type: application/json" \
          --data "{}" -s)
      - export TEMP_TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.token')
      - echo "Temporary token fetched"

      - echo "Pulling Docker image"
      - docker pull --platform linux/arm64 distributions-dev.traps.paloaltonetworks.com/cli-docker/${TEMP_TOKEN}/method:arm64-${CORTEX_CLI_VERSION}-dev

      - echo "Tagging Docker image"
      - docker tag distributions-dev.traps.paloaltonetworks.com/cli-docker/${TEMP_TOKEN}/method:arm64-${CORTEX_CLI_VERSION}-dev cortexcli:${CORTEX_CLI_VERSION}

    - echo "Setting Extra Environment Variables"
    - |
      export CODEBUILD_ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
      export CODEBUILD_GIT_BRANCH="$(git symbolic-ref HEAD --short 2>/dev/null)"
      if [ "$CODEBUILD_GIT_BRANCH" = "" ] ; then
        export CODEBUILD_GIT_BRANCH="$(git rev-parse HEAD | xargs git name-rev | cut -d' ' -f2 | sed 's/remotes\//origin\///g')"
      fi
      export CODEBUILD_PROJECT=${CODEBUILD_BUILD_ID%:$CODEBUILD_LOG_PATH}

    echo "==> AWS CodeBuild Extra Environment Variables:"
    echo "==> CODEBUILD_ACCOUNT_ID = $CODEBUILD_ACCOUNT_ID"
    echo "==> CODEBUILD_GIT_BRANCH = $CODEBUILD_GIT_BRANCH"
    echo "==> CODEBUILD_PROJECT = $CODEBUILD_PROJECT"
```

```

build:
  commands:
    - echo "Running Docker container"
    - |
      docker run --rm --platform linux/arm64 cortexcli:${CORTEX_CLI_VERSION} \
        --api-base-url ${CORTEX_API_URL} \
        --api-key ${CORTEX_API_KEY} \
        --api-key-id ${CORTEX_API_KEY_ID} \
        code scan \
        --directory . \
        --repo-id $CODEBUILD_ACCOUNT_ID/$CODEBUILD_PROJECT \
        --branch $CODEBUILD_GIT_BRANCH

```

2.3.2 | Onboard CircleCI for code scans

Integrate Cortex Cloud Application Security with your CircleCI system to allow dynamic, automated, and context-specific code scans across your codebase. This integration provides continuous scanning of your workflows, triggered by code changes or pipeline events, ensuring security checks are performed and issues are detected as early as possible.

Code scans are executed using the Cortex CLI, and include automated shift-left actions based on scan results.

NOTE:

CircleCI onboarding offers both code and CI/CD scanning. A single integrated instance supports either code or CI scanning, but not both. If you require both code and CI scanning for your CircleCI environment, you must create two separate integrations, selecting the appropriate scanning type for each. To onboard CircleCI for CI/CD scans, refer to Onboard CircleCI for CI/CD pipeline scans.

PREREQUISITE:

Before you begin:

-
1. On the Cortex Cloud console:
 - a. Navigate to Settings → Data Sources & Integrations → + Add New.
 - b. Enter CircleCI in the search bar → Hover over the displayed search result → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

2. On the Select Integration step of the CircleCI integration wizard, select Code Scan → Next.
3. On the Add Environment Variables step of the wizard.
 - a. Select Generate API key.

The API key secret and API key ID values are generated and populate their respective fields.
 - b. Select your system architecture.
 - c. Click Next.
4. Create a context in CircleCI and name it cortex-secrets.

IMPORTANT:

The cortex-secrets naming convention for the context is mandatory to ensure functionality and must not be changed.

5. Store your Cortex Cloud API Key and API ID within the cortex-secrets context.

- If you have an API key:

1. Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID variable names from their respective fields in the wizard.
2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables (secrets) to the cortex-secrets context.

- If you do not have an API key:

1. Click Generate API key → Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values from their respective fields.
2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables to the cortex-secrets context.

NOTE:

Do not change the names of the environment variables provided by Cortex Cloud. They are required for proper integration and functionality.

For more information on context in CircleCI, refer to Using contexts in CircleCI.

6. Copy and paste the pre-populated code from the Configure Job step of the integration wizard into your `.circleci/config.yaml` file, and click Done.

7. In your `.circleci/config.yaml` file:

- Verify that the YAML file includes a Docker container image
- Verify that the context is `cortex-secrets`
- In the `docker run` command, replace `--repo-id REPO_OWNER/REPO_NAME` values with your repository owner and repository name

8. Check that the The integration will be created once CircleCI authorizes message is displayed in the final step of the wizard and click Close .

9. Verify integration and confirm that the your integrated CircleCI instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for CircleCI in the search bar.
- b. Hover over the resulting entry and click View Details.
- c. Verify that the status of your CircleCI instance is Connected and that the instance type Code is displayed.

10. Next step: View scan results and mitigate issues.

NOTE:

To add an additional CircleCI instance, navigate to Settings → Data Sources & Integrations → select the menu for your connected CircleCI instance → + New Instance, and repeat the onboarding steps above.

CircleCI code scan workflow template

This circle workflow example automates code scanning using the Cortex CLI. The workflow contains placeholder values (often in brackets) and generic terms (such as dev) that you must replace with your environment-specific information before use

```
version: 2.1

executors:
  docker-executor:
    docker:
      - image: cimg/base:stable # Replace with a suitable image or executor
        environment:
          CORTEX_API_URL: "https://{{CORTEX_URL}}
          CORTEX_CLI_VERSION: "0.8.11"

jobs:
  setup-environment:
    executor: docker-executor
    steps:
      - checkout
      - setup_remote_docker
      - run:
          name: Get Temporary Token and Pull Docker Image
          command: |
            export TOKEN_RESPONSE=$(curl --location "${CORTEX_API_URL}/public_api/cas/v1/cortex-cli/create-token" \
              --header "Authorization: ${CORTEX_API_KEY}" \
              --header "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
              --header "Content-Type: application/json" \
              --data "{}" -s)
            export TEMP_TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.token')
            docker pull distributions-dev.traps.paloaltonetworks.com/cli-
            docker/${TEMP_TOKEN}/method:amd64-${CORTEX_CLI_VERSION}-dev
            docker tag distributions-dev.traps.paloaltonetworks.com/cli-
            docker/${TEMP_TOKEN}/method:amd64-${CORTEX_CLI_VERSION}-dev cortexcli:${CORTEX_CLI_VERSION}
          - run:
              name: Run Cortex CLI Container
              # Replace owner/repo with your actual repository information
              command: |
                docker run --rm cortexcli:${CORTEX_CLI_VERSION} \
                  --api-base-url ${CORTEX_API_URL} \
                  --api-key ${CORTEX_API_KEY} \
                  --api-key-id ${CORTEX_API_KEY_ID} \
                  code scan \
                  --directory .
                  --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
                  --branch "${CIRCLE_BRANCH}"

workflows:
  version: 2
  build:
    jobs:
      - setup-environment:
          context: cortex-secrets
```

2.3.3 | Connect Cortex CLI

Connect Cortex CLI to scan supported Cortex Cloud modules and gain insights into your security posture. The onboarding process involves downloading the architecture-specific binary, installing required runtime dependencies, and authenticating the client against your Cortex Cloud tenant.

PREREQUISITE:

- **System requirements** (OS-specific):

The following operating system dependencies are required to run the Cortex Cloud CLI binary on specific architectures.

- **macOS** (Intel Core i7, such as Sequoia):
 - To ensure all functionalities work correctly, you must install the `vectorscan` dependency via **Homebrew**. Use this command: `brew install vectorscan`
- **RHEL 8.10 and Red Hat UBI9**:
 - Install `patchelf`
 - Install `zstd`
- **Ubuntu 20**:
 - Requires the `prefetch` utility
- **Ubuntu (for linux-amd64)**
 - Requires the `libhyperscan5` library in addition to the `prefetch` utility. To install, run `sudo apt install libhyperscan5`

- Windows: Supported on AMD 64 and ARM 64 architectures

- **Application Security module requirements:**

These requirements are in addition to the base operating system dependencies listed above. You must ensure the base CLI binary is functional (such as `vectorscan` or `libhyperscan` is installed) before addressing these module-specific prerequisites.

- **Runtime requirement:**

- You must have Node.js v22 installed on your host machine. The CLI relies on this specific version to execute JavaScript analysis during code scans. To verify your version, run node –v

- **Supported Linux specifications for the Application Security module:**

For Linux users, the AppSec module requires specific Kernel and GLIBC versions to function correctly. Support is provided for systems meeting the following specifications.

Distribution	Kernel Version	GLIBC Version
RHEL 10:	6.12	2.39
Debian	6.1.27	2.36
Ubuntu 18.04	4.15	2.27
Ubuntu 20.04	5.4	2.31
Ubuntu 22.04	5.15	2.35
Ubuntu 24.04	6.8	2.39

- **API Security and CWP requirements**

Java: Java version 11 or higher is required specifically for API Security and Cloud Workload Protection (CWP) scans. To verify your version, run java –version

- **Installation utilities (For cURL-based downloads)**

To perform **cURL-based** downloads and installation, curl and jq are required.

- **Installation Commands for jq**

- On **Ubuntu/Debian-based Linux** distributions: sudo apt-get install jq
- On **RedHat/CentOS/Fedora**: sudo yum install jq
- **macOS** (using Homebrew): brew install jq
- **Windows:**
 - Download the executable from jq GitHub releases
 - If Chocolatey is installed: choco install jq

- **Best Practice** (required for SCA vulnerability suppression):

- **Path and repository matching**

- **Execution directory:** Run the CLI within your current working directory <current_directory_path>. It is recommended to use the **absolute file path** for your current working directory
- **Repository ID structure:** Ensure that the --repo-id parameter includes the <repo_owner_name>/<repo_name> structure
- **Directory name match:** The <repo_name> part of the ID must precisely match the name of the directory you are scanning

Example 1.

The present working directory is Users/test/<repo_name>. Therefore, the --repo-id parameter must be --repo-id <repo_owner_name>/<repo_name>, ensuring that <repo_name> precisely matches the directory name within the structure.

- **Windows execution environment:** For terminal actions performed by IDE extensions on Windows, Command Prompt (CMD) is the supported environment. PowerShell is not supported for these actions

If you suppress vulnerabilities from Cortex Cloud IDE extensions on Windows, the action depends on the shell environment in use

- **Supported Shell:** Command Prompt (CMD) is the supported environment for these actions.
- **Unsupported Shell: PowerShell is not supported** for terminal actions performed by Cortex Cloud IDE extensions. If the extension attempts to execute the suppression command via PowerShell, the action will fail

1. Navigate to Settings → Data Sources & Integrations → + Add New.

2. Enter Cortex CLI in the search bar and click Add.

NOTE:

Select Add Another Instance if a CLI instance is already onboarded.

3. On the Configure step of the integration wizard.

- Select your operating system → Next.
- Download the CLI binary: copy (or download) the command provided in the wizard and paste into your terminal.
- Click Next.

4. On The Authenticate step of the wizard.

This step combines credential configuration and binary installation. You have two options for authentication: generating a new key (Required for CWP) or using an existing one, provided it has CLI permissions.

a. Configure authentication. Select one of the following options:

- i. Option A: Generate a New API Key:

A. Select Generate API key.

- **Critical:** This option is required for CWP image scans. The CWP module does not support Read-only roles because it must upload offline image scan results to the tenant. If you deselect this, CWP scans will fail
- This option creates a CLI role for the API key with CLI View/Edit role. It is recommended as it grants the API key permissions to not only access data, but also to upload or send data back
- If you do not select this option, the generated API key creates a CLI Read Only role with CLI View permissions only

B. Optional: create the API key with upload results permissions.

WARNING:

Using With upload results permissions may incur additional costs as per your license agreement.

- C. Save credentials: Copy the the generated API Key ID and API key that are displayed in their respective fields. Save your key immediately, as it is displayed only once.
- D. Verify: You can verify that the generated API key is displayed under the API Keys inventory.

ii. Option B: Use an existing API Key

You can use a key generated previously, provided it has the necessary CLI permissions for your specific module (excepting CWP). CLI View/Edit permissions correspond to selecting With upload results permissions, while CLI Read Only or View permissions corresponds to not selecting the With upload results permissions.

b. Download and install.

The code block displayed in the wizard is a script that automates the download and setup.

What this code does: Whether you copy the command or download the file (see options below), this script performs the following:

- **Fetch binary:** Uses `curl` to download the architecture-specific binary (such as Linux AMD64) from your tenant
- **Authenticate:** Authenticates the download request using the API Key you insert
- **Set permissions:** Saves the file as `cortexcli` and automatically runs `chmod +x` to make it executable

Select one of the following options to execute this code:

i. Option A Graphical installer (macOS only).

Recommended: Ensure you have **Administrator** permissions to install.

- A. **Download:** Click the Download icon next to the command (not the copy).
- B. Move the downloaded installation bundle to the required folder.

C. **Locate file:** Open the folder containing the downloaded bundle and locate the file named `cortexcli.pkg`.

D. **Install:** Double-click `cortexcli.pkg` to launch the installer.

E. Wizard steps: Select Continue → Install → Close.

ii. **Option B: Copy and paste the code.**

A. Click the copy button or manually copy the code block shown in the wizard.

B. **Edit (Critical):** You cannot run this code immediately because it contains a placeholder for your security credentials.

- Open your terminal.
- Create a new directory: Run `mkdir [FOLDER_NAME]`
- Navigate to the folder: `CD [FOLDER_NAME]`
- Paste the copied command from the wizard in your terminal
- Replace `${CORTEX_API_KEY}` in the code with your API key
- Replace `${CORTEX_API_KEY_ID}` in the code with your API key ID

C. **Run:** Execute the modified command in your terminal.

iii. **Option C: Manual binary download.**

A. **Download the binary:** Run the following curl command (ensure you replace the variables with your specific tenant and version details):

- `curl -k -u ${CORTEX_API_ID}::${CORTEX_API_KEY} --output ./cortexcli ${CORTEX_FQDN}/api/v2/remote-li/{version}/{platform}/artifacts`

B. **Execute the CLI** Run the permissions command to make the binary executable:

- `chmod +x cortexcli`

C. Verify installation: Run `cortexcli -v`.

NOTE:

In the command above, `{version}` refers to the CLI version you wish to install (such as latest or a specific number), and `{platform}` refers to your operating system (such as, linux, darwin, windows).

c. Click Next → Close.

2.3.4 | GitHub Actions

Integrate Cortex Cloud Application Security with GitHub Actions to allow dynamic, automated, and context-specific scans within your development workflow. This includes continuous scanning of your workflows

whenever changes are pushed or triggered, integrating security checks, and detecting issues as soon as they are introduced.

How to integrate GitHub Actions

PREREQUISITE:

Before you begin:

- 1. On the Cortex Cloud console:
 - a. Navigate to Settings → Data Sources & Integrations → + Add New.
 - b. Enter GitHub Actions in the search bar → Hover over the displayed search result → Connect.
 2. On the Add Environment Variables step of the wizard.
 - a. Select Generate API key.

The API key secret and API key ID values are generated and populate their respective fields.
 - b. Optional: Change the default system architecture detected by the system.
 - c. Click Next.
 3. Store your Cortex Cloud API key and API key ID in the GitHub Actions Secrets credential store.
 - If you have an API key:
 1. Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID variable names from their respective fields in the wizard.
 2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables (secrets) to the GitHub Actions Secrets credential store.
 - If you do not have an API key:
 1. Click Generate API key → Copy the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values from their respective fields.
 2. Add the CORTEX_API_KEY and CORTEX_API_KEY_ID and their corresponding values as separate environment variables (secrets) to the GitHub Actions Secrets credential store.

NOTE:

Do not change the names of the environment variables provided by Cortex Cloud. They are required for proper integration and functionality.

For more information on passing secrets as environment variables to GitHub Actions, refer to Using secrets in GitHub Actions.

4. Copy and paste the pre-populated sample code from the Configure Job step of the integration wizard into your GitHub Actions job configuration → Done.

NOTE:

The code is only a reference. Replace the placeholder values with your build-specific values.

5. Ensure that the **Connector Created Successfully** message is displayed in the final step of the wizard, and click **Close**.
6. Verify integration and confirm that the your integrated GitHub Actions instance has a status of **Connected**.
 - a. On the Data Sources & Integrations page, search for GitHub Actions in the search bar.
 - b. Hover over the resulting entry and click **View Details**.
 - c. Verify that the status of your GitHub Actions instance is **Connected**.
7. Next step: View scan results and mitigate issues.

NOTE:

To add an additional GitHub Actions instance, navigate to **Settings** → **Data Sources & Integrations** → select the menu for your connected GitHub Actions instance → **+ New Instance**, and repeat the onboarding steps above.

GitHub Actions code scan workflow template

This GitHub Actions workflow example automates code scanning using the Cortex CLI. The workflow contains placeholder values (often in brackets) and generic terms (such as **dev**) that you must replace with your environment-specific information before use.

```
name: Cortex CLI Code Scan

on:
  push:
    branches:
      - main
  workflow_dispatch:

env:
  CORTEX_API_KEY: ${{secrets.CORTEX_API_KEY}}
  CORTEX_API_KEY_ID: ${{secrets.CORTEX_API_KEY_ID}}
  CORTEX_API_URL: https://<CORTEX_URL>
  CORTEX_CLI_VERSION: 0.8.11

jobs:
  download-and-execute:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2

      - name: Set up QEMU
        uses: docker/setup-qemu-action@v2
        with:
          platforms: arm64

      - name: Install Dependencies
        run: |
          sudo apt-get update
          sudo apt-get install -y jq curl

      - name: Get Temporary Token
        run: |
          TOKEN_RESPONSE=$(curl --location "${CORTEX_API_URL}/public_api/cas/v1/cortex-cli/create-token" \
```

```

--header "Authorization: ${CORTEX_API_KEY}" \
--header "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
--header 'Content-Type: application/json' \
--data '{}')
TEMP_TOKEN=$(echo $TOKEN_RESPONSE | jq -r '.token')
echo "TEMP_TOKEN=$TEMP_TOKEN" >> $GITHUB_ENV

- name: Pull Docker Image
  run: |
    docker pull distributions-dev.traps.paloaltonetworks.com/cli-
docker/${{env.TEMP_TOKEN}}/method:arm64-${{env.CORTEX_CLI_VERSION}}-dev
    docker tag distributions-dev.traps.paloaltonetworks.com/cli-
docker/${{env.TEMP_TOKEN}}/method:arm64-${{env.CORTEX_CLI_VERSION}}-dev cortexcli:${{env.CORTEX_CLI_VERSION}>

- name: Run Docker Container
  run: |
    docker run --rm --platform linux/arm64 cortexcli:${{env.CORTEX_CLI_VERSION}} \
    --api-base-url ${CORTEX_API_URL} \
    --api-key ${CORTEX_API_KEY} \
    --api-key-id ${CORTEX_API_KEY_ID} \
    code scan \
    --directory . \
    --repo-id ${github.repository}

```

2.3.5 | Onboard Jenkins for code scans

Integrate Cortex Cloud Application Security with your Jenkins server to allow dynamic, automated, and context-specific code scans across your codebase. This integration provides continuous scanning of your workflows, triggered by code changes or pipeline events, ensuring security checks are performed and issues are detected as early as possible.

Code scans are executed using the Cortex CLI, and include automated shift-left actions based on scan results.

NOTE:

Jenkins onboarding offers both code and CI/CD scanning. A single integrated instance supports either code or CI scanning, but not both. If you require both code and CI scanning for your Jenkins servers, you must create two separate integrations, selecting the appropriate scanning type for each. To onboard Jenkins for CI/CD scans, refer to Onboard Jenkins for CI/CD pipeline scans.

PREREQUISITE:

- Grant **Administrator** permissions to the user integrating Cortex Cloud Application Security with Jenkins
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On the Cortex Cloud console:

- a. Navigate to Settings → Data Sources & Integrations → + Add New.
- b. Enter Jenkins in the search bar → Hover over the displayed search result → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

2. On the Select Integration step of the Jenkins integration, select Code Scan → Next.

3. On the Add Environment Variables step of the wizard.

- Select Generate API key.

The API key secret and API key ID values are generated and populate their respective fields.

- Select your system architecture.

- Click Next.

4. Store your Cortex Cloud API Key and API ID in the Jenkins Credentials store.

PREREQUISITE:

- For Cortex Cloud Application Security CI tools, you must store secrets in Jenkins Credentials for use in your Jenkins pipelines using either of these methods:
 - **Plain text storage:** Store secrets directly as plain text in Jenkins Credentials. Access them in your pipeline using the `credentials` function, which retrieves the secret directly as plain text
 - **Credentials Binding Plugin:** Use the `withCredentials` function (requires installing the Credentials Binding Plugin) to securely bind credentials to environment variables within your pipeline
- The variable names `CORTEX_API_KEY` and `CORTEX_API_KEY_ID` must be used exactly as provided. They are part of a predefined system and cannot be changed without causing errors
- If you have an API key:
 1. Copy the `CORTEX_API_KEY` and `CORTEX_API_KEY_ID` variable names from their respective fields in the wizard.
 2. Add the `CORTEX_API_KEY` and `CORTEX_API_KEY_ID` names and their corresponding values as separate environment variables (secrets) to the Jenkins Credentials store.
- If you do not have an API key:
 1. Click Generate API key → Copy the `CORTEX_API_KEY` and `CORTEX_API_KEY_ID` and their corresponding values from their respective fields.
 2. Add the `CORTEX_API_KEY` and `CORTEX_API_KEY_ID` names and their corresponding values as separate environment variables (secrets) to the Jenkins Credentials store.

5. On the Set repository step of the wizard: (Optional): Add the URL of the repository to be scanned, or skip this step if you are adding code scanning to an existing pipeline → Next.

NOTE:

- This step is only required for new pipelines
- For private repositories, ensure the necessary credentials are configured in Jenkins Credentials

6. On the Configure Subscription step of the integration wizard.

- Copy and paste the code from the Configure Subscription step of the integration wizard into your Jenkins pipeline.
- In the `labels` property of your Jenkins configuration file, enter the label of a Jenkins node that is configured with Docker.

NOTE:

This ensures your build runs within a Docker environment. If a node without Docker is used, the build will fail.

- c. Optional: The provided code assumes that your Cortex Cloud access key and ID are stored as plain text in Jenkins Credentials. You can replace this method with your preferred secret management solution (such as the `withCredentials` function).

- d. Click Done.

7. Verify you receive the confirmation message on the last step of the wizard → Close.

8. Verify integration and confirm that the your integrated Jenkins instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for Jenkins in the search bar.
- b. Hover over the resulting entry and click View More.
- c. Verify that the status of your Jenkins instance is Connected and that the instance type Code is displayed.

9. Next step: View scan results and mitigate issues.

NOTE:

To add an additional Jenkins instance, navigate to Settings → Data Sources & Integrations → select the Jenkins integration, click → Add Instance, and repeat the onboarding steps.

Jenkins code scan workflow template (without checkout)

This Jenkins workflow example automates code scanning using the Cortex CLI. It does not include a step to checkout a repository. The workflow contains placeholder values (often in brackets) and generic terms (such as dev) that you must replace with your environment-specific information before use.

Read more...

```
pipeline {  
    agent {  
        docker {  
            image 'jenkins/agent:alpine'  
            args '-u root --privileged -v /var/run/docker.sock:/var/run/docker.sock'  
            label '<REPLACE WITH LABEL OF NODE WITH DOCKER INSTALLED>' // Use a docker agent with docker  
installed  
        }  
    }  
  
    environment {  
        CORTEX_API_KEY = credentials('CORTEX_API_KEY')  
        CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')  
        CORTEX_API_URL = 'https://api-viso-hdkbzk6qphxpbehy758elo.xdr-qa2-uat.us.paloaltonetworks.com'  
        CORTEX_CLI_VERSION = '0.8.11'  
    }  
  
    stages {  
        stage('Install Dependencies') {  
            steps {  
                sh '''  
                apk add --no-cache jq docker  
                '''  
            }  
        }  
    }  
}
```

```

stage('Get Temporary Token') {
    environment {
        TEMP_TOKEN = ""
    }
    steps {
        script {
            def response = sh(script: """
                curl --location '${env.CORTEX_API_URL}/public_api/cas/v1/cortex-cli/create-token' \
                    --header 'Authorization: ${env.CORTEX_API_KEY}' \
                    --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \
                    --header 'Content-Type: application/json' \
                    --data '{}' \
                    -s
            """, returnStdout: true).trim()

            env.TEMP_TOKEN = sh(script: """echo '${response}' | jq -r '.token'""", returnStdout: true).trim()
        }
    }
}

stage('Pull Docker Image') {
    steps {
        sh """
            docker pull distributions-dev.traps.paloaltonetworks.com/cli-
            docker/${env.TEMP_TOKEN}/method:amd64-${env.CORTEX_CLI_VERSION}-dev
            docker tag distributions-dev.traps.paloaltonetworks.com/cli-
            docker/${env.TEMP_TOKEN}/method:amd64-${env.CORTEX_CLI_VERSION}-dev cortexcli:${env.CORTEX_CLI_VERSION}
        """
    }
}

stage('Run Docker Container') {
    // Replace the repo-id with your repository like: owner/repo
    steps {
        unstash 'source'
        env.BRANCH = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim()
        sh """
            docker run --rm --platform linux/amd64 cortexcli:${env.CORTEX_CLI_VERSION} \
                --api-base-url ${env.CORTEX_API_URL} \
                --api-key ${env.CORTEX_API_KEY} \
                --api-key-id ${env.CORTEX_API_KEY_ID} \
                code scan \
                --directory . \
                --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
                --branch $BRANCH
        """
    }
}
}

```

Jenkins code scan workflow template (with checkout)

This Jenkins workflow example automates code scanning using the Cortex CLI. It includes a step to checkout a repository. The workflow contains placeholder values (often in brackets) and generic terms (such as dev) that you must replace with your environment-specific information before use.

Read more...

```

pipeline {
    agent {
        docker {
            image 'jenkins/agent:alpine'
            args '-u root --privileged -v /var/run/docker.sock:/var/run/docker.sock'
            label '<REPLACE WITH LABEL OF NODE WITH DOCKER INSTALLED>' // Use a docker agent with docker
        installed
    }
}

```

```

        }
    }

environment {
    CORTEX_API_KEY = credentials('CORTEX_API_KEY')
    CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')
    CORTEX_API_URL = 'https://api-viso-hdkbz6qphxpbehy758elo.xdr-qa2-uat.us.paloaltonetworks.com'
    CORTEX_CLI_VERSION = '0.8.11'
}

stages {
    stage('Checkout Repository') {
        steps {
            git branch: 'main', url: 'https://github-example.com/example-repo'
            stash includes: '**/*', name: 'source'
        }
    }

    stage('Install Dependencies') {
        steps {
            sh '''
                apk add --no-cache jq docker
            '''
        }
    }

    stage('Get Temporary Token') {
        environment {
            TEMP_TOKEN = """
        }
        steps {
            script {
                def response = sh(script: """
                    curl --location '${env.CORTEX_API_URL}/public_api/cas/v1/cortex-cli/create-token' \
                        --header 'Authorization: ${env.CORTEX_API_KEY}' \
                        --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \
                        --header 'Content-Type: application/json' \
                        --data '{}' \
                        -s
                """, returnStdout: true).trim()

                env.TEMP_TOKEN = sh(script: """echo '${response}' | jq -r '.token'""", returnStdout: true).trim()
            }
        }
    }

    stage('Pull Docker Image') {
        steps {
            sh """
                docker pull distributions-dev.traps.paloaltonetworks.com/cli-docker/${env.TEMP_TOKEN}/method:amd64-${env.CORTEX_CLI_VERSION}-dev
                docker tag distributions-dev.traps.paloaltonetworks.com/cli-docker/${env.TEMP_TOKEN}/method:amd64-${env.CORTEX_CLI_VERSION}-dev cortexcli:${env.CORTEX_CLI_VERSION}
            """
        }
    }

    stage('Run Docker Container') {
        // Replace the repo-id with your repository like: owner/repo
        steps {
            unstash 'source'
            env.BRANCH = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim()
            sh """
                docker run --rm --platform linux/amd64 cortexcli:${env.CORTEX_CLI_VERSION} \
                    --api-base-url ${env.CORTEX_API_URL} \
                    --api-key ${env.CORTEX_API_KEY} \
                    --api-key-id ${env.CORTEX_API_KEY_ID} \
                    code scan \
            """
        }
    }
}

```

```
--directory . \
--repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
--branch $BRANCH
.....
}
}
}
}
```

2.3.6 | Onboard Terraform Cloud (Run Tasks)

Integrate Cortex Cloud Application Security with Terraform Cloud (Run Tasks) to enable dynamic, automated, and context-specific scans in your Terraform workspace. Cortex Cloud Application Security scans Terraform (TF) frameworks for misconfigurations based on default and custom policies whenever changes are triggered, ensuring seamless security checks. It identifies issues such as infrastructure-as-code (IaC) misconfigurations, Software Composition Analysis (SCA) vulnerabilities, exposed secrets, and license non-compliance, depending on the security scanners that you have subscribed to.

You can monitor and remediate issues directly in the Cortex Cloud Application Security console. Run statuses and violation details can be tracked in both Cortex Cloud Application Security and Terraform Cloud through streamlined run task reviews. For more information about streamlined tasks, refer to <https://www.hashicorp.com/blog/terraform-cloud-adds-streamlined-run-task-reviews>.

PREREQUISITE:

Before you begin:

- Procure a Terraform cloud license that is either a trial license or a TF Cloud license at the TEAM & GOVERNANCE level
 - Terraform permissions: Grant the user or team the following permissions, depending on integration:
 - *Manage Workspaces* permissions at the organization level. These permissions are required to attach and manage the run task on workspaces or:
 - *Administrator* permissions on the workspace(s)
 - Create a Terraform **Organization**. For more information, refer to the Terraform documentation
 - Create a Terraform **Workspace**: For more information, refer to the Terraform documentation
 - Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On your Terraform Cloud platform, create a Terraform api token.
 - a. Select your user/profile icon → User Settings.
 - b. Select the Tokens section from the left side menu.
 - c. Click Create an API token → provide a description → Create API token .
 - d. Copy and save the token+ Done.

NOTE:

Skip this step if you plan on using an existing token.

For more information about Terraform API tokens, refer to the [Terraform API Tokens](#) documentation.

2. On the Cortex Cloud console.

- a. Select **Settings** → **Data Sources** (under **Data Collections**) → **+ Data Source**.
- b. Enter **Terraform Cloud (Run Tasks)** in the search bar → hover over the displayed search result → **Connect**.

NOTE:

Select **Add Another Instance** if the data source instance has already been configured.

3. Provide your Terraform user or team API token on the **Configure Account** step of the wizard → **Next**.

4. Select an organization from the **Select Organization** step of the wizard → **Next**.

5. On the **Select Workspace** step of the wizard.

- a. Select repositories from the **Selection Options** field.

- Permit all existing repositories
- Permit all existing and future repositories
- Choose from repository list

- b. Select a run plan from the **Run Stage** field.

- Pre-plan: The scan runs before Terraform generates the plan
- Post-plan: The scan runs after Terraform generates the plan

NOTE:

Cortex Cloud Application Security performs a scan of Terraform templates on selected workspaces based on the Run Stage.

- c. Click **Save** and then **Close** in the final verification step of the wizard.

6. Verify integration and confirm that the your integrated Terraform Cloud (Run Tasks) instance has a status of **Connected**.

- a. On **Data Sources**, search for **Terraform Cloud (Run Tasks)** in the search bar.
- b. Hover over the resulting entry and click **View Details**.
- c. Verify that the status of your Terraform Cloud (Run Tasks) instance is **Connected**.

7. Next step: View scan results and mitigate issues.

NOTE:

To add an additional Terraform Cloud (Run Tasks) instance, navigate to **Settings** → **Data Sources** → select the menu for your connected your Terraform Cloud (Run Tasks) instance → **+ New Instance**, and repeat the onboarding steps.

2.3.7 | Onboard Terraform Enterprise (Run Tasks)

Integrate Cortex Cloud Application Security with Terraform Enterprise (Run Tasks) to enable dynamic, automated, and context-specific scans in your Terraform workspace. Cortex Cloud Application Security scans Terraform (TF) frameworks for misconfigurations based on default and custom policies whenever changes are triggered, ensuring seamless security checks. It identifies infrastructure-as-code (IaC) misconfigurations, Software Composition Analysis (SCA) vulnerabilities^{*}, exposed secrets, and license non-compliance issues, depending on the security scanners that you have subscribed to.

NOTE:

For container image vulnerabilities, Cortex Cloud Application Security performs 'Image Referencer' scans within Terraform Enterprise (Run Tasks), as full SCA scans are not currently supported.

You can monitor and remediate issues directly in the Cortex Cloud Application Security console. Run statuses and violation details can be tracked in both Cortex Cloud Application Security and Terraform Enterprise through streamlined run task reviews. For more information about streamlined tasks, refer to <https://www.hashicorp.com/blog/terraform-cloud-adds-streamlined-run-task-reviews>.

PREREQUISITE:

Before you begin:

- Ensure access to a Terraform Enterprise console to enable you to provide a user or team token that authorizes Cortex Cloud Application Security to access workspaces and helps regulate run configurations
- Terraform Enterprise version compatibility: Ensure *Run Tasks* for workspaces on is compatible with version 1.1.9 and above
- Terraform Enterprise user or team permissions: For a workspace integration of run tasks you need to ensure that the token used has the following permissions. These permissions enable Cortex Cloud to configure run tasks in the environment and scan plan files from your runs:
 - **Manage run tasks** permissions at the organizational level. These permissions are required to create and manage the run task in the organization
 - **Manage Workspaces** permissions at the organization level. These permissions are required to attach and manage the run task on workspaces or:
 - **Administrator** permissions on the workspace(s)

NOTE:

For more on Terraform Run Task permissions refer to Manage Run Tasks permissions.

- Create a Terraform **Organization**. For more information, refer to the Terraform documentation
- Create a Terraform **Workspace**: For more information, refer to the Terraform documentation
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. On your Terraform Enterprise platform, create a Terraform api token.

- a. Select your user/profile icon → User Settings.
- b. Select the Tokens section from the left side menu.
- c. Click Create an API token → provide a description → Create API token .
- d. Copy and save the token+ Done.

NOTE:

Skip this step if you plan on using an existing token.

For more information about Terraform API tokens, refer to the Terraform API Tokens documentation.

2. On the Cortex Cloud console:

- a. Navigate to Settings → Data Sources & Integrations → + Add New.
- b. Enter Terraform Enterprise (Run Tasks) in the search bar → Hover over the displayed search result → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

3. Provide your Terraform user or team API token on the Configure Account step of the wizard → Next.

4. Select an organization from the Select Organization step of the wizard → Next.

5. On the Select Workspace step of the wizard:

- a. Select repositories from the Selection Options field.
 - Permit all existing repositories
 - Permit all existing and future repositories
 - Choose from repository list
- b. Select a run plan from the Run Stage field.
 - Pre-plan: The scan runs before Terraform generates the plan
 - Post-plan: The scan runs after Terraform generates the plan

NOTE:

Cortex Cloud Application Security performs a scan of Terraform templates on selected workspaces based on the Run Stage.

- c. Click Save.

6. Click Save and then Close in the final verification step of the wizard.

7. Verify integration and confirm that the your integrated Terraform Enterprise (Run Tasks) instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for Terraform Enterprise (Run Tasks) in the search bar.
- b. Hover over the resulting entry and click View More.
- c. Verify that the status of your Terraform Enterprise (Run Tasks) instance is Connected.

8. Next step: View scan results and mitigate issues.

NOTE:

To add an additional Terraform Enterprise (Run Tasks) instance, navigate to Settings → Data Sources & Integrations → page, select the Terraform Enterprise (Run Tasks) integration, click → Add Instance, and

repeat the onboarding steps above.

2.4 | CLI pipeline code snippets

You can integrate the Cortex CLI directly into your CI/CD pipelines to enable automated code scans by adding code snippets to your build script or pipeline configuration, such as a **YAML** or **Groovy** file. Both **ARM** and **AMD** architectures are supported, ensuring you can scan your codebase regardless of your runner's environment.

PREREQUISITE:

User permissions: Ensure the user performing the integration has permissions to edit pipeline configurations (such as YAML files).

User permissions: Ensure the user performing the integration has permissions to edit pipeline configurations (such as YAML files).

You must replace placeholder variables with your own credentials and environment-specific details.

AWS CodeBuild

- For AMD architecture

```
version: 0.2
env:
  variables:
    CORTEX_API_URL: <your_cortex_api_url>
    CORTEX_CLI_VERSION: "0.13.14"
  secrets-manager:
    CORTEX_API_KEY: "CORTEX_API_KEY"
    CORTEX_API_KEY_ID: "CORTEX_API_KEY_ID"
phases:
  install:
    commands:
      - apt-get update
      - apt-get install -y curl jq git
  pre_build:
    commands:
      - echo "Getting repo name"
      - export CODEBUILD_ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
      - export CODEBUILD_GIT_BRANCH="$(git symbolic-ref HEAD --short 2>/dev/null)"
      - |
        if [ "$CODEBUILD_GIT_BRANCH" = "" ] ; then
          export CODEBUILD_GIT_BRANCH="$(git rev-parse HEAD | xargs git name-rev | cut -d' ' -f2 | sed 's/remotes\|origin\|//g')";
        fi
      - export CODEBUILD_PROJECT=${CODEBUILD_BUILD_ID%:$CODEBUILD_LOG_PATH}
      - echo "Downloading cortexcli"
      - |
        crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
          -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
          -H "Authorization: ${CORTEX_API_KEY}")
        crtx_url=$(echo "$crtx_resp" | jq -r ".signed_url")
        curl -o cortexcli "$crtx_url"
        chmod +x cortexcli
        ./cortexcli --version

  build:
    commands:
      - |
        ./cortexcli \
          --api-base-url "${CORTEX_API_URL}" \
          --api-key "${CORTEX_API_KEY}" \
          --api-key-id "${CORTEX_API_KEY_ID}" \
          code scan \
          --directory "$(pwd)" \
          --repo-id $CODEBUILD_ACCOUNT_ID/$CODEBUILD_PROJECT \
          --branch $CODEBUILD_GIT_BRANCH \
          --source AWS_CODE_BUILD \
          --create-repo-if-missing
  artifacts:
    files:
      - '**/*'
```

- For ARM architecture

```
version: 0.2
env:
  variables:
    CORTEX_API_URL: <your_cortex_api_url>
    CORTEX_CLI_VERSION: "0.13.16"
  secrets-manager:
    CORTEX_API_KEY: "CORTEX_API_KEY"
    CORTEX_API_KEY_ID: "CORTEX_API_KEY_ID"
phases:
  install:
```

```

commands:
  - apt-get update
  - apt-get install -y curl jq git

pre_build:
  commands:
    - echo "Getting repo name"
    - export CODEBUILD_ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    - export CODEBUILD_GIT_BRANCH="$(git symbolic-ref HEAD --short 2>/dev/null)"
    - |
      if [ "$CODEBUILD_GIT_BRANCH" = "" ] ; then
        export CODEBUILD_GIT_BRANCH="$(git rev-parse HEAD | xargs git name-rev | cut -d' ' -f2 | sed 's/remotes\|origin\//g')";
      fi
    - export CODEBUILD_PROJECT=${CODEBUILD_BUILD_ID%:$CODEBUILD_LOG_PATH}
    - echo "Downloading cortexcli"
    - |
      crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}")
    - crtx_url=$(echo "$crtx_resp" | jq -r ".signed_url")
    - curl -o cortexcli "$crtx_url"
    - chmod +x cortexcli
    - ./cortexcli --version

build:
  commands:
    - |
      ./cortexcli \
        --api-base-url "${CORTEX_API_URL}" \
        --api-key "${CORTEX_API_KEY}" \
        --api-key-id "${CORTEX_API_KEY_ID}" \
        code scan \
        --directory "$(pwd)" \
        --repo-id $CODEBUILD_ACCOUNT_ID/$CODEBUILD_PROJECT \
        --branch $CODEBUILD_GIT_BRANCH \
        --source AWS_CODE_BUILD \
        --create-repo-if-missing

artifacts:
  files:
    - '**/*'

```

Azure Pipelines

- For AMD architecture

```
trigger:
  branches:
    include: ['*']
pr:
  branches:
    include: ['*']
pool:
  vmImage: ubuntu-latest
variables:
  CORTEX_API_URL: <your_cortex_api_url>
  MIN_LOG_LEVEL: "DEBUG"
steps:
- checkout: self
  clean: true
- task: NodeTool@0
  displayName: "Use Node.js 22.x"
  inputs:
    versionSpec: "22.x"
- bash: |
    set -euo pipefail
    sudo apt-get update
    sudo apt-get install -y --no-install-recommends jq ca-certificates curl
    BASE="${CORTEX_API_URL%/*}"
    URL="$BASE/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=amd64"
    set +x
    CRTX_URL=$(curl -fsSL "$URL" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
    set -x
    curl -fsSL -o cortexcli "$CRTX_URL"
    chmod +x cortexcli
  displayName: "Download cortexcli (amd64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
- bash: |
    set -euo pipefail
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "$(Build.SourcesDirectory)" \
      --repo-id "$(Build.Repository.Name)" \
      --branch "$(Build.SourceBranchName)" \
      --source "CORTEX_CLI" \
      --create-repo-if-missing
  displayName: "Cortex CLI Code Scan"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
  MIN_LOG_LEVEL: $(MIN_LOG_LEVEL)
```

- For ARM architecture

```
trigger:
  branches:
    include: ['*']
pr:
  branches:
    include: ['*']
variables:
```

```

CORTEX_API_URL: <your_cortex_api_url>
pool:
  name: arm
  demands:
    - Agent.OS -equals Linux
steps:
- checkout: self
  clean: true
- task: NodeTool@0
  displayName: "Use Node.js 22.x"
  inputs: { versionSpec: "22.x" }
- bash: |
    set -euo pipefail
    BASE="${CORTEX_API_URL%/*}"
    URL="$BASE/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=arm64"
    set +x
    CRTX_URL=$(curl -fsSL "$URL" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
    set -x
    curl -fsSL -o cortexcli "$CRTX_URL"
    chmod +x cortexcli
  displayName: "Download cortexcli (arm64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
- bash: |
    set -euo pipefail
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "$(Build.SourcesDirectory)" \
      --repo-id "$(Build.Repository.Name)" \
      --branch "$(Build.SourceBranchName)" \
      --source "CORTEX_CLI" \
      --create-repo-if-missing
  displayName: "Cortex CLI Code Scan (ARM64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)

```

Bitbucket

- For AMD architecture

```
image: ubuntu:24.04
clone:
  depth: full
pipelines:
  default:
    - step:
        name: Cortex CLI Code Scan (Hosted AMD64)
        script:
          - set -euo pipefail
          - apt-get update && apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip
file
          - curl -fsSL https://deb.nodesource.com/setup_22.x | bash -
          - apt-get install -y nodejs
          - node -v && npm -v
          - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
"$CORTEXCLI_HOME"
          - |
            CRTX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
            -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
            -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
            curl -fsSL -o cortexcli "$CRTX_URL"
            chmod +x cortexcli
            ./cortexcli --version
          - |
            ./cortexcli \
              --api-base-url "${CORTEX_API_URL}" \
              --api-key "${CORTEX_API_KEY}" \
              --api-key-id "${CORTEX_API_KEY_ID}" \
              code scan \
              --directory "${BITBUCKET_CLONE_DIR}" \
              --repo-id "${BITBUCKET_REPO_FULL_NAME}" \
              --branch "${BITBUCKET_BRANCH}" \
              --source "CORTEX_CLI" \
              --create-repo-if-missing \
              --upload-mode no-upload
artifacts:
  - cortexcli
```

- For ARM architecture

```
image: node:22-bookworm

pipelines:
  default:
    - step:
        name: Cortex CLI Code Scan
        runs-on:
          - self.hosted
          - linux.arm64
        script:
          - set -euo pipefail
          - apt-get update && apt-get install -y --no-install-recommends curl jq ca-certificates file
          - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
"$CORTEXCLI_HOME"

          - |
            set +x
            CRTX_URL=$(curl -fsSL "${CORTEX_API_URL%}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
            -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
            -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
            set -x
            curl -fsSL -o cortexcli "$CRTX_URL"
```

```
chmod +x cortexcli
./cortexcli --version

# Run the scan
- |
  ./cortexcli \
    --api-base-url "${CORTEX_API_URL}" \
    --api-key "${CORTEX_API_KEY}" \
    --api-key-id "${CORTEX_API_KEY_ID}" \
    code scan \
    --directory "${BITBUCKET_CLONE_DIR}" \
    --repo-id "${BITBUCKET_REPO_FULL_NAME}" \
    --branch "${BITBUCKET_BRANCH}" \
    --source "CORTEX_CLI" \
    --create-repo-if-missing
artifacts:
- cortexcli
```

CircleCI

- For AMD architecture

```
version: 2.1
jobs:
  cortex-code-scan:
    docker:
      - image: cimg/node:22.17.0 # Replace with a suitable image or executor
        environment:
          CORTEX_API_URL: <your_cortex_api_url>
    steps:
      - checkout
      - run:
          name: Download cortexcli
          command: |
            set -x
            crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
            -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
            -H "Authorization: ${CORTEX_API_KEY}")
            crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
            curl -o cortexcli $crtx_url
            chmod +x cortexcli
            ./cortexcli --version
      - run:
          name: Run Cortex CLI Code Scan
          command: |
            ./cortexcli \
              --api-base-url "${CORTEX_API_URL}" \
              --api-key "${CORTEX_API_KEY}" \
              --api-key-id "${CORTEX_API_KEY_ID}" \
              code scan \
              --directory "$(pwd)" \
              --repo-id "${CIRCLE_PROJECT_USERNAME}/${CIRCLE_PROJECT_REPONAME}" \
              --branch "${CIRCLE_BRANCH}" \
              --source "CIRCLE_CI" \
              --create-repo-if-missing
    workflows:
      cortex-scan-workflow:
        jobs:
          - cortex-code-scan:
              context: cortex-secrets
```

- For ARM architecture

```
version: 2.1
jobs:
  cortex-code-scan:
    docker:
      - image: <Replace with image supporting node js version 22 or higher>
        environment:
          CORTEX_API_URL: <your_cortex_api_url>
    steps:
      - checkout
      - run:
          name: Download cortexcli
          command: |
            set -x
            crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
            -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
            -H "Authorization: ${CORTEX_API_KEY}")
            crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
            curl -o cortexcli $crtx_url
            chmod +x cortexcli
            ./cortexcli --version
      - run:
```

```
name: Run Cortex CLI Code Scan
command: |
  ./cortexcli \
  --api-base-url "${CORTEX_API_URL}" \
  --api-key "${CORTEX_API_KEY}" \
  --api-key-id "${CORTEX_API_KEY_ID}" \
  code scan \
  --directory "$(pwd)" \
  --repo-id "${CIRCLE_PROJECT_USERNAME}/${CIRCLE_PROJECT_REPONAME}" \
  --branch "${CIRCLE_BRANCH}" \
  --source "CIRCLE_CI" \
  --create-repo-if-missing
workflows:
  cortex-scan-workflow:
    jobs:
      - cortex-code-scan:
          context: cortex-secrets
```

GitHub Actions

- For AMD architecture

```
name: Cortex CLI Code Scan
on:
  push:
    branches:
      - main
  workflow_dispatch:
env:
  CORTEX_API_KEY: ${{secrets.CORTEX_API_KEY}}
  CORTEX_API_KEY_ID: ${{secrets.CORTEX_API_KEY_ID}}
  CORTEX_API_URL: <your_cortex_api_url>

jobs:
  cortex-code-scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2

      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 22
      - name: Verify Node.js Version
        run: node -v
      - name: Download cortexcli
        run: |
          set -x
          crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=amd64" \
          -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
          -H "Authorization: ${CORTEX_API_KEY}")
          crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
          curl -o cortexcli $crtx_url
          chmod +x cortexcli
          ./cortexcli --version
      - name: Run Cortex CLI Code Scan
        run: |
          ./cortexcli \
            --api-base-url "${CORTEX_API_URL}" \
            --api-key "${CORTEX_API_KEY}" \
            --api-key-id "${CORTEX_API_KEY_ID}" \
            code scan \
            --directory "${{github.workspace}}" \
            --repo-id "${{github.repository}}" \
            --branch "${{github.ref_name}}" \
            --source "GITHUB_ACTIONS" \
            --create-repo-if-missing
```

- For ARM architecture

```
name: Cortex CLI Code Scan
on:
  push:
    branches:
      - main
  workflow_dispatch:
env:
  CORTEX_API_KEY: ${{secrets.CORTEX_API_KEY}}
  CORTEX_API_KEY_ID: ${{secrets.CORTEX_API_KEY_ID}}
  CORTEX_API_URL: <your_cortex_api_url>

jobs:
  cortex-code-scan:
    runs-on: ubuntu-latest
```

```
steps:
- name: Checkout Repository
  uses: actions/checkout@v2

- name: Set up Node.js
  uses: actions/setup-node@v4
  with:
    node-version: 22
- name: Verify Node.js Version
  run: node -v
- name: Download cortexcli
  run: |
    set -x
    crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=arm64" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}")
    crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
    curl -o cortexcli $crtx_url
    chmod +x cortexcli
    ./cortexcli --version
- name: Run Cortex CLI Code Scan
  run: |
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "${{github.workspace}}" \
      --repo-id "${{github.repository}}" \
      --branch "${{github.ref_name}}" \
      --source "GITHUB_ACTIONS" \
      --create-repo-if-missing
```

GitLab Runner

- For AMD architecture

```
stages: [scan]
variables:
  CORTEX_API_URL: <your_cortex_api_url>
cortex_code_scan:
  image: node:22-bookworm@sha256:bb6834c0669aa71cbc8d94606561a721adf489f6b93d7b8b825f0cf1b498c2c4
  tags: ["amd64"]
  stage: scan
  rules:
    - when: on_success
  before_script:
    - uname -m
    - set -euo pipefail
    - apt-get update
    - apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip file
    - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p "$CORTEXCLI_HOME"
    - |
      # avoid leaking secrets in logs
      set +x
      CRTLX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=amd64" \
        -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
        -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
      set -x
      curl -fsSL -o cortexcli "$CRTLX_URL"
      chmod +x cortexcli
      ./cortexcli --version
  script:
    - |
      ./cortexcli \
        --api-base-url "${CORTEX_API_URL}" \
        --api-key "${CORTEX_API_KEY}" \
        --api-key-id "${CORTEX_API_KEY_ID}" \
        code scan \
        --directory "${CI_PROJECT_DIR}" \
        --repo-id "${CI_PROJECT_PATH}" \
        --branch "${CI_COMMIT_REF_NAME}" \
        --source "CORTEX_CLI" \
        --upload-mode no-upload \
        --create-repo-if-missing
  artifacts:
    when: always
    paths: [cortexcli]
    expire_in: 1 day
```

- For ARM architecture

```
stages: [scan]
variables:
  CORTEX_API_URL: <your_cortex_api_url>
cortex_code_scan:
  image: node:22-bookworm
  stage: scan
  rules:
    - when: on_success
  before_script:
    - set -euo pipefail
    - apt-get update
    - apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip file
    - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p "$CORTEXCLI_HOME"
    - |
      # avoid leaking secrets in logs
      set +x
```

```
CRTX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
-H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
-H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
set -x
curl -fsSL -o cortexcli "$CRTX_URL"
chmod +x cortexcli
./cortexcli --version
script:
- |
./cortexcli \
--api-base-url "${CORTEX_API_URL}" \
--api-key "${CORTEX_API_KEY}" \
--api-key-id "${CORTEX_API_KEY_ID}" \
code scan \
--directory "${CI_PROJECT_DIR}" \
--repo-id "${CI_PROJECT_PATH}" \
--branch "${CI_COMMIT_REF_NAME}" \
--source "CORTEX_CLI" \
--upload-mode no-upload \
--create-repo-if-missing
artifacts:
when: always
paths: [cortexcli]
expire_in: 1 day
```

Jenkins

- For AMD architecture

```
pipeline {  
    agent {  
        docker {  
            image 'cimg/node:22.17.0' // Replace with a suitable image or executor  
            args '-u root'  
        }  
    }  
    environment {  
        CORTEX_API_KEY = credentials('CORTEX_API_KEY')  
        CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')  
        CORTEX_API_URL = <your_cortex_api_url>  
    }  
    stages {  
        stage('Checkout Repository') {  
            steps {  
                git branch: 'main', url: 'this-is-repository-url-example'  
                stash includes: '**/*', name: 'source'  
            }  
        }  
        stage('Install Dependencies') {  
            steps {  
                sh ''''  
                apt update  
                apt install -y curl jq git  
                '''  
            }  
        }  
        stage('Download cortexcli') {  
            steps {  
                script {  
                    def response = sh(script: '''''  
                        curl --location '${env.CORTEX_API_URL}/public_api/v1/unified-  
cli/releases/download-link?os=linux&architecture=amd64' \  
                        --header 'Authorization: ${env.CORTEX_API_KEY}' \  
                        --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \  
                        --silent  
                    ''''', returnStdout: true).trim()  
                    def downloadUrl = sh(script: '''''echo '${response}' | jq -r '.signed_url'''',  
returnStdout: true).trim()  
                    sh '''''  
                        curl -o cortexcli '${downloadUrl}'  
                        chmod +x cortexcli  
                        ./cortexcli --version  
                    '''''  
                }  
            }  
        }  
        stage('Run Scan') {  
            // Replace the repo-id with your repository like: owner/repo  
            steps {  
                script {  
                    unstash 'source'  
                    sh '''''  
                    ./cortexcli \  
                    --api-base-url "${env.CORTEX_API_URL}" \  
                    --api-key "${env.CORTEX_API_KEY}" \  
                    --api-key-id "${env.CORTEX_API_KEY_ID}" \  
                    code scan \  
                    --directory "$(pwd)" \  
                    --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \  
                    --branch <REPLACE WITH BRANCH> \  
                    --source "JENKINS" \  
                    --create-repo-if-missing  
                }  
            }  
        }  
    }  
}
```

```

        }
    }
}
}

• For ARM architecture

pipeline {
    agent {
        docker {
            image 'cimg/node:22.17.0' // Replace with a suitable image or executor
            args '-u root'
        }
    }
    environment {
        CORTEX_API_KEY = credentials('CORTEX_API_KEY')
        CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')
        CORTEX_API_URL = <your_cortex_api_url>
    }
    stages {
        stage('Checkout Repository') {
            steps {
                git branch: 'main', url: 'this-is-repository-url-example'
                stash includes: '**/*', name: 'source'
            }
        }
        stage('Install Dependencies') {
            steps {
                sh '''
                    apt update
                    apt install -y curl jq git
                    ...
                '''
            }
        }
        stage('Download cortexcli') {
            steps {
                script {
                    def response = sh(script: """
                        curl --location '${env.CORTEX_API_URL}/public_api/v1/unified-
                        cli/releases/download-link?os=linux&architecture=arm64' \
                        --header 'Authorization: ${env.CORTEX_API_KEY}' \
                        --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \
                        --silent
                    """, returnStdout: true).trim()
                    def downloadUrl = sh(script: """
                        echo '${response}' | jq -r '.signed_url'
                    """, returnStdout: true).trim()
                    sh """
                        curl -o cortexcli '${downloadUrl}'
                        chmod +x cortexcli
                        ./cortexcli --version
                    """
                }
            }
        }
        stage('Run Scan') {
            // Replace the repo-id with your repository like: owner/repo
            steps {
                script {
                    unstash 'source'
                    sh """
                        ./cortexcli \
                        --api-base-url "${env.CORTEX_API_URL}" \
                        --api-key "${env.CORTEX_API_KEY}" \
                        --api-key-id "${env.CORTEX_API_KEY_ID}" \
                        code scan \
                        --directory "$(pwd)" \
                        --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
                        --branch <REPLACE WITH BRANCH> \
                    """
                }
            }
        }
    }
}

```

```
--source "JENKINS" \
--create-repo-if-missing
.....
}
}
}
}
```

2.5 | Onboard private package registries

Onboard your private package registries to secure your internal software supply chain. While private registries protect your source code from public access, they also create visibility gaps for standard security scanners. By connecting these registries, you grant the Software Composition Analysis (SCA) scanner the access required to resolve private dependencies and build artifacts. This ensures that vulnerabilities hidden within your internal libraries are detected, enabling comprehensive risk visibility across your development environment.

Cortex Cloud Application Security currently supports JFrog registries.

2.5.1 | JFrog Artifactory

Onboard JFrog Artifactory to authorize the Software Composition Analysis (SCA) scanner to resolve packages stored in your private Artifactory instance. By retrieving dependency metadata directly from the registry, the scanner can build accurate dependency trees and reliably detect vulnerabilities in your private libraries.

Integration scope

- SCA vs Image scanning: to build accurate dependency trees. It does not support container image scanning. To scan images, you must configure a separate JFrog Artifactory instance. To integrate JFrog Artifactory for image scans, refer to Connect JFrog container registry.
- You can onboard only one JFrog Artifactory instance. Within that instance, you can configure one integration for each supported package manager type

Supported environments

Supports both JFrog Artifactory Cloud (SaaS) and JFrog Artifactory Self-Hosted (On-Premises) environments.

Supported package managers

- **Maven:** Supports mirroring
- **Gradle**
- **NPM:**

2.5.2 | Onboard JFrog Artifactory

Follow the steps below to configure the connection to your JFrog Artifactory instance.

PREREQUISITE:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- JFrog permissions:
 - The permissions associated with the user configured during the onboarding process determine the scope of scan results. Only repositories and artifacts the user can access are included

Repository access: The Artifactory user must have Read access to the specific repositories you want to scan
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

1. Navigate to Settings → Data Sources & Integrations → + Add New.
2. Hover over JFrog Artifactory and click Add.
3. Select Package resolution for code scanning as the integration type.
4. Provide an instance name → Enable access by IPs (optional) → Next.

NOTE:

To authorize the scanner to connect through your firewall, select Enable access by IPs, and copy the displayed source IPs to your organization's allowlist.

5. Fill in the provided fields and click Next.

- Registry URL: Enter your JFrog Artifactory URL.

Example 2. Examples

- For JFrog SaaS integrations: <https://example.jfrog.io>
- For JFrog on-premises integrations: <https://artifactory.example.com>, where `<artifactory.example.com>` is your server domain or IP address
- Username (required): Your JFrog user name
- Password (required): Your JFrog password

6. Select a package manager to configure a registry as private instead of the default public registry.

NOTE:

- For Maven:
 - Select Mirror Registry if this repository mirrors an external repository
 - Use the Mirror Of value to define the duplication scope:
 - * mirrors all requests
 - Type a request [value]: Mirrors only specific requests (such as central).
- Package managers not listed will default to the public registry
- You can only proceed after selecting at least one package manager.

7. (Optional): Select Add a package manager to set up an additional package manager.
8. Click Save.
9. **Verify integration:** Verify integration and confirm that the your integrated JFrog Artifactory instance is Connected.
 - a. Navigate to Settings → Data Sources & Integrations.
 - b. Filter the table by Provider=JFrog.
 - c. Select the resulting displayed instance.
 - d. On the Data Source side panel, verify that the Status displays Connected.

2.6 | Supported third-party data sources

Cortex Cloud Application Security supports the following third party ingestions:

- Ingest Veracode SAST data
- Ingest SonarQube SAST data
- Ingest Semgrep data
- Generic 3rd Party AppSec Collector

NOTE:

Only onboarded and scanned repositories can be mapped.

View ingested data

SAST findings and issues

- Code Weaknesses issues page: View dedicated SAST issues on the SAST code weaknesses (CWEs) page
- Asset inventories (Code Weaknesses tab):
 - Repositories Refer to for more information
 - Application : Refer to for more information

CVE vulnerabilities findings and issues

- Vulnerabilities issues page. View CVE vulnerabilities on the Software Composition Analysis (SCA) vulnerability issues page
- Asset inventories (Vulnerabilities tab):
 - Repositories: Refer to for more information
 - Application: Refer to for more information

License miscompliance findings and issues

- Licenses issues page. View license miscompliance on the License miscompliance issues page
- Asset inventories (Package Integrity tab):
 - Repositories: Refer to for more information
 - Application: Refer to for more information

2.6.1 | Ingest Semgrep data

Ingest findings from Semgrep's Static Application Security Testing (SAST) and Software Composition Analysis (SCA) scanners directly into Cortex Cloud. Cortex analyzes and prioritizes these findings—both security weaknesses in your code (SAST) and vulnerabilities in open-source dependencies (SCA)—so you can focus on the most critical issues first. Consolidating findings from native and third-party scanners in a single platform streamlines remediation and provides a clear, comprehensive view of your application security posture.

Supported outputs: Semgrep supports json, text and SARIF output formats.

How to connect Semgrep with Cortex Cloud Application Security

PREREQUISITE:

- **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
- Ensure that you have a connected version control system (VCS) and repositories
- Create a Semgrep API token

NOTE:

- To create a Semgrep API token, in Semgrep, navigate to Settings → Tokens → API tokens
- Ensure you select the **Web API** scope (sometimes labeled as Management or API access depending on your plan)
- The Web API permission authorizes Cortex Cloud to query the Semgrep API and retrieve your SCA/SAST findings

For more information on Semgrep API tokens, refer to [Create a Semgrep app token](#).

- Create an egress path to establish the designated route for outbound data transmission from Semgrep to Cortex Cloud

1. Select Settings → Data Sources & Integrations → + Add New.

2. Enter Semgrep in the search bar → Hover over the Semgrep card → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

3. On the Configure Integration step of the integration wizard: Provide your Semgrep API key → Authorize.
4. On the Select Issues Types step of the integration wizard.

Select the type of data findings to ingest: SAST, SCA or both → Next.

5. On the Select Projects step of the integration wizard.

Review the detected Semgrep projects and confirm or manage their repository mappings:

a. Options:

- Accept the displayed mapping as detected by Cortex Cloud. This does not require any action on your part
- Automatically map projects: Select Automatically map future Semgrep projects to ensure maximum security coverage by automatically mapping current and future Semgrep projects to Cortex Cloud repositories
- Configure unmapped or mismatched projects: Manually configure mapping if Cortex Cloud cannot match a project to a repository or an update to the mapping is required: From the list of detected projects, select the project from the list, then choose the correct repository from the Repository dropdown menu

b. Click Save.

NOTE:

- Mapping establishes relationships between Semgrep projects and Cortex Cloud code repositories, simplifying access management and enabling risk analysis at the repository level, including displaying findings on the tenant
- Only mapped projects are ingested

6. Verify integration and confirm that your integrated Semgrep instance has a status of Connected.

- a. Navigate to Settings → Data Sources & Integrations → Filter for Semgrep (for example Provider=Semgrep).
- b. Select the required instance from the list.
- c. Verify that the status of your Semgrep connector displays Connected.
- d. Verify the Semgrep projects are connected: Select the connector and verify that the mapped projects are displayed and connected.

[Manage Semgrep connections](#)

To manage Semgrep connections, refer to Manage ingested third-party connections.

[Manage SAST code weaknesses generated from ingested Semgrep findings](#)

You can view SAST code weaknesses generated from ingested Semgrep findings:

- On the Code Weaknesses page under Cortex Cloud Application Security Issues
- Under the Code Weaknesses tab of the Repositories assets page

For more information on SAST code weaknesses, refer to SAST code weaknesses (CWEs).

[Manage SCA issues generated from ingested Semgrep findings](#)

You can view SCA vulnerabilities generated from ingested Semgrep findings.

- CVE vulnerabilities:
 - On the Vulnerabilities page under Cortex Cloud Application Security Issues
 - Under the Vulnerabilities tab of the Software Packages assets page
- License miscompliance issues:
 - On the Licenses page under Cortex Cloud Application Security Issues
 - Under the Package Integrity tab of the Software Packages assets page

For more information on SCA issues, refer to Overview.

2.6.2 | Ingest Snyk data

Ingest findings from Snyk's Static Application Security Testing (SAST) and Software Composition Analysis (SCA) scanners directly into Cortex Cloud. Cortex analyzes and prioritizes these findings, both security weaknesses in your code (SAST) and vulnerabilities in open-source dependencies (SCA), so you can focus on the most critical issues first. Consolidating findings from native and third-party scanners in a single platform streamlines remediation and provides a clear, comprehensive view of your application security posture.

How to connect Snyk with Cortex Cloud Application Security

PREREQUISITE:

- **Permissions:** The following user permissions are required:
 - **Cortex Cloud user permissions:** Ensure you have View/Edit permissions for Data Sources and Integrations (RBAC: AppSec Admin or Instance Administrator)
 - Ensure that you have a connected version control system (VCS) and repositories
- **Snyk API token:**
 - Recommended. Generate an API token from a Service Account (Service accounts are decoupled from individual users, ensuring the integration remains uninterrupted even if an employee leaves the organization or changes roles)
 - **Role:** Create a Service Account and assign the Group Viewer role (for Group-level access) or Org Collaborator role (for specific organizations)
 - This role authorizes Cortex Cloud to list applications and retrieve findings via the API. It provides the necessary read-only access without granting excessive administrative privileges
 - Once generated, copy the key and store it securely

1. Navigate to Settings → Data Sources & Integrations → + Add New.

2. Enter Snyk in the search bar → Hover over the Snyk card → Connect.

NOTE:

Select Add Another Instance if the data source instance has already been configured.

3. On the Configure Integration step of the integration wizard.

a. Configure Snyk parameters:

- Select your Snyk API URL from the menu (for example API URLsNYK-US-02 (<https://api.us.snyk.io/rest>))
 - Enter your Snyk API token
- b. Click Authorize.
4. On the Select Organization step of the wizard: Enter your Snyk Organization → Next:
5. On the Select Issue Types step of the wizard: Select the type of data findings to be ingested: SAST, SCA or both → Next.
6. On the Map to Repositories step of the wizard, review the detected Snyk projects and confirm or manage their repository mappings
- Select Automatically map future Snyk applications to automatically map current and future Snyk projects to Cortex Cloud repositories. This is recommended to ensure maximum security coverage
 - Configure unmapped or mismatched applications: Manually configure mapping if Cortex Cloud cannot match an application to a repository or an update to the mapping is required: From the list of detected applications, select the application from the list, then choose the correct repository from the Repository dropdown menu
- b. Click Save.

NOTE:

- Mapping establishes relationships between Snyk applications and Cortex Cloud code repositories, simplifying access management and enabling risk analysis at the repository level, including displaying findings on the tenant
- Only mapped applications are ingested

7. Verify integration and confirm that the your integrated Snyk instance has a status of Connected.
- a. Navigate to Settings → Data Sources & Integrations → Filter for Snyk (for example Provider=Snyk).
 - b. Select the required instance from the list.
 - c. Verify that the status of your Snyk connector displays Connected.
 - d. Verify the Snyk applications are connected: Select the connector and verify that the mapped applications are displayed and connected.

[Manage Snyk connections](#)

To manage Snyk connections, refer to Manage ingested third-party connections.

[Manage SCA issues generated from Snyk findings](#)

You can view and manage SCA issues generated from ingested Snyk SCA findings to assess and manage vulnerabilities: Navigate to Modules → Application Security → Vulnerabilities (under Issues).

For more information about SCA issues and findings, refer to Software Composition Analysis (SCA) vulnerability issues.

Manage SAST code weaknesses generated from ingested Snyk findings

You can view and manage SAST code weaknesses generated from ingested Snyk findings:

- On the Code Weaknesses page under Cortex Cloud Application Security Issues
- Under the Code Weaknesses tab of the Repositories assets page

For more information on SAST code weaknesses, refer to SAST code weaknesses (CWEs).

2.6.3 | Ingest SonarQube SAST data

You can ingest SAST findings directly from SonarQube into Cortex Cloud Application Security. This allows you to use Cortex Cloud Application Security's analysis and visualization tools to identify critical vulnerabilities, prioritize remediation efforts, and improve your application code security.

SonarQube supports json output format.

How to connect SonarQube with Cortex Cloud Application Security

PREREQUISITE:

- Permissions: Ensure you have System Admin, AppSec Admin or GRBAC permissions. For more information on AppSec Admin permissions, refer to Code Security user roles and permissions
- Ensure that you have a connected version control system (VCS) and repositories
- Generate and copy a SonarQube API token. Ensure to assign Web API scope to the API token. Refer to the SonarQube documentation for more information
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to Egress configurations

NOTE:

The egress path is required for onboarding a self-hosted instance of SonarQube.

1. Navigate to Settings → Data Sources & Integrations → + Add New.
2. Select SonarQube under the '3rd Party Ingestion' section in the catalog.
3. On the Configure Integration step of the integration wizard:

- a. Fill in the provided fields:

- API Token: Paste the generated SonarQube API token
- URL and Port: Provide the URL of your SonarQube instance. Port is optional
- Organization: The SonarQube organization to be associated with the data ingestion. Required for SonarQube Cloud

- b. Click Accept.

4. On the Select Applications step of the integration wizard:
 - a. Select an option:

- Accept the displayed mapping as detected by Cortex Cloud Application Security. This does not require any action on your part
- Manually configure mapping if Cortex Cloud Application Security could not match a project to a repository: Select Set in the Cortex Cloud Application Security Repository column, and select a repository from the list that is displayed
- Automatically map future SonarQube projects
- Manually modify mapping: Click Replace next to the existing mapped Cortex Cloud Application Security repository. This will open an option to select a different repository from the displayed list, allowing you to update the mapping

NOTE:

- Mapping establishes relationships between SonarQube Applications and Cortex Cloud Application Security code repositories, simplifying access management and enabling risk analysis at the repository level, including displaying findings on the tenant
- Only mapped projects will be ingested

b. Click Save.

5. Select Close on the Status step of the wizard to complete the integration, initiating an automatic ingestion of data from the integrated SonarQube projects.

NOTE:

Verify that the Connector Created Successfully message is displayed on the page.

6. Verify integration and confirm that the your integrated SonarQube instance has a status of Connected.

- a. On the Data Sources & Integrations page, search for SonarQube in the search bar.
- b. Hover over the resulting entry and click View More.
- c. Verify that the status of your SonarQube instance is Connected.

[View SAST code weaknesses generated from ingested SonarQube findings](#)

You can view SAST code weaknesses generated from ingested SonarQube findings:

- On the Code Weaknesses page under Cortex Cloud Application Security Issues
- Under the Code Weaknesses tab of the Repositories assets page

For more information on SAST code weaknesses, refer to SAST code weaknesses (CWEs).

[Manage SonarQube connections](#)

To manage SonarQube connections, refer to Manage ingested third-party connections.

2.6.4 | Ingest Veracode SAST data

You can ingest SAST findings directly from Veracode into Cortex Cloud Application Security. This allows you to use Cortex Cloud Application Security's analysis and visualization tools to identify critical vulnerabilities,

prioritize remediation efforts, and improve your application code security.

Veracode supports **Cyclonedx**, **json** and table output formats.

How to connect Veracode with Cortex Cloud Application Security

PREREQUISITE:

- Permissions: The following user permissions are required:
 - Cortex Cloud: Instance Admin, AppSec Admin or GRBAC permissions. For more information on AppSec Admin permissions, refer to [Code Security user roles and permissions](#)
 - Veracode: At minimum, Reviewer permissions are required
- Ensure that you have a connected version control system (VCS) system and repositories
- Generate and copy a Veracode access key. The access key includes a key ID and secret
- Create an egress path to establish the designated route for outbound data transmission from Cortex Cloud to third party services. For more information about configuring egress paths, refer to [Egress configurations](#)

1. Navigate to Settings → Data Sources & Integrations → + Add New.

2. Select Veracode under the '3rd Party Ingestion' section in the catalog.

3. On the Configure Integration step of the integration wizard:

a. Fill in the provided fields:

- Enter the Veracode key ID and secret from step **1b** into their respective fields
- Select your Veracode region from the Region dropdown

b. Click Authorize.

The integrationSelect Applications step of the integration wizard is displayed, including a list of Veracode applications automatically mapped to Cortex Cloud Application Security repositories.

4. Select an option, and click Save.

- Select Automatically map future Veracode applications to automatically map all future applications to Cortex Cloud Application Security repositories
- Manually map Veracode applications to Cortex Cloud Application Security repositories: Click on a Cortex Cloud Application Security repository and select the required repository

NOTE:

Only mapped applications will be ingested.

- a.
 - All current applications
 - All current and future applications

NOTE:

This is the recommended option to ensure complete coverage and successful operation of all features.

- Only selected applications, and then select the applications from the menu
- b. Click Next.

5. On the Map to Repositories step of the wizard:

- a. Select an option:
 - Accept the displayed mapping as detected by Cortex Cloud Application Security . This does not require any action on your part
 - Manually configure mapping if Cortex Cloud Application Security could not match a project to a repository: Select Set in the Cortex Cloud Application Security Repository column, and select a repository from the list that is displayed
 - Reject mapping: Check the Don't map any applications box
 - Manually modify mapping: Click Replace next to the existing mapped Cortex Cloud repository. This will open an option to select a different repository from the displayed list, allowing you to update the mapping

NOTE:

- Mapping establishes relationships between Veracode projects and Cortex Cloud Application Security code repositories, simplifying access management and enabling risk analysis at the repository level, including displaying findings on the tenant
- Only mapped projects will be ingested

- b. Click Next.

6. Select Done on the Status step of the wizard to complete the integration, initiating an automatic ingestion of data from the integrated Veracode projects.

NOTE:

Verify that the Connector Created Successfully message is displayed on the page.

7. Verify integration and confirm that your integrated Veracode instance has a status of Connected.
 - a. On the Data Sources & Integrations page, search for Veracode in the search bar.
 - b. Hover over the resulting entry and click View More.
 - c. Verify that the status of your Veracode instance is Connected.

Limitations

- Currently, Veracode SAST ingestion supports Veracode periodic and CLI scans. Pull Request scans and other types are not supported
- History, deduplication and DevEx features such as PR comments, IDE, CLI and enforcement are not supported

View SAST code weaknesses generated from ingested Veracode findings

You can view SAST code weaknesses generated from ingested Veracode findings:

- On the Code Weaknesses page under Cortex Cloud Application Security Issues
- Under the Code Weaknesses tab of the Repositories assets page

For more information on SAST code weaknesses, refer to SAST code weaknesses (CWEs).

Manage connections

To manage Veracode connections, refer to Manage ingested third-party connections.

2.6.5 | Generic 3rd Party AppSec Collector

The 3rd Party AppSec Collector automatically uploads Static Application Security Testing (SAST) data from third-party tools that support SARIF (Static Analysis Results Interchange Format) output. This allows you to view your SAST data directly within Cortex Cloud. Once SARIF files are uploaded, they are parsed to create code findings. These findings can then be elevated to issues, either manually or automatically, depending on your configured policies.

IMPORTANT:

File uploads are limited to a maximum size of 10 MB.

After onboarding the 3rd Party AppSec Collector, you can view SAST code weaknesses generated from ingested SARIF findings in these locations:

- On the Code Weaknesses page under Cortex Cloud Application Security Issues
 - For more information about SAST issues under Code Weaknesses issues, refer to Manage code weaknesses
- Under dedicated Code Weaknesses tabs on the Repositories or Business Applications assets pages, where relevant SAST issues have been identified for that asset.
 - For more information about SAST issues on the Code Weaknesses tab under Repositories assets, refer to In-depth repository asset information
 - For more information about SAST issues on the Code Weaknesses tab under Business Applications assets, refer to Business application expanded asset details

2.6.5.1 | Onboard the 3rd Party AppSec Collector

Before you begin, fulfill these prerequisites.

PREREQUISITE:

- **Permissions:** The following Cortex Cloud user roles or RBAC permissions are required:

- **User roles:** Cortex Cloud Instance Admin, AppSec Admin
- **RBAC:** View/Edit permissions for Data Sources configurations are required when not using a dedicated user role

For more information about user permissions and groups, refer to [Assign user roles and groups](#).

- Onboard the repository into the system before SARIF findings for that repository can be uploaded
- **SARIF specifications:** The following table outlines the mandatory and optional JSON fields required to successfully validate and ingest SAST findings.

Cortex Cloud Application Security supports only valid SARIF files that strictly adhere to the SARIF standard (v2.1.0). The collector will not ingest files with invalid formats or schema violations. Ensure your third-party tool output is validated before upload.

Field Path	Requirement	Description And Validation
version	Mandatory	The SARIF version number. Must be explicitly set to the string value <code>2.1.0</code>
tool.driver.name	Mandatory	A string identifying the primary analysis tool used
runs.tool.driver.rules OR runs.tool.extensions.rules	Mandatory	At least one of these must be populated to define all ruleId's used in the results
run.results.ruleId	Mandatory	The unique identifier for the specific rule violated. This must exactly match an id defined in the rules array
run.results[].locations[] Mandatory	Location.artifact	The relative or absolute URI for the file where the finding was detected
run.results[].locations[] Mandatory	Location.region	The specific region within the file where the finding is located. This object must contain the startLine property to identify the location
run.results.message.text	Mandatory	A human-readable, plain-text description of the finding. This is the primary text displayed to the user.

Field Path	Requirement	Description And Validation
run.results.level	Optional	The severity level of the finding. Allowed values are error, warning, note, or none. If omitted, a Low severity level is set by default

1. Select Settings → Data Sources & Integrations → + Add New → and enter 3rd Party AppSec Collector in the search bar.

2. On the Configure Collector step of the integration wizard.

- a. Provide a Collector Name (required): This is a free-text field. You can input any descriptive name.

NOTE:

We recommend using the `tool.driver.name` from the SARIF file.

- b. Click Generate API Key to obtain the collector instance API key ID and key secret.

3. On the API Key step of the integration wizard.

- a. Copy and save the generated API key ID and API secret.

WARNING:

The API key ID and API secret cannot be retrieved once the wizard is closed.

- b. Copy the API URL.

NOTE:

This is the newly created generic collector API URL endpoint.

- c. Select Next.

4. (Optional): Validate the file format on the Test step of the wizard to ensure it meets all ingestion requirements.

NOTE:

Only the validity of the format is tested. No findings will be generated from the test file.

- a. Browse and upload the required file.

After upload, you'll see one of the following validation outcomes:

- Validation completed successfully
- Validation finished with warnings: One or more of the properties of at least one of the `results` of the SARIF format failed validation
- Validation failed: one of the mandatory fields is missing/wrong value. See SARIF mandatory fields in the prerequisites above

- b. Select Done.

5. Setup the data extraction to programmatically send the SARIF SAST results to the Cortex Cloud 3rd Party AppSec Collector instance via API.

You need to add the following values to the API request:

- The Cortex Cloud API key ID and secret generated in **step 2** above
- The Cortex Cloud URL copied in **step 3** above
- Your Cortex Cloud repository ID. To retrieve the repository ID, under Inventory, navigate to All Assets → Repositories (under Code) → select a repository → copy the Asset ID value from the Properties section of the side card
- (Optional) Branch: Default unless specified

Example 3. Example Setups

- **CURL**

```
curl -X POST {API_URL_FROM_RESPONSE}?repository_id={repository_asset_id}&branch={branch_name} -H 'x-crtx-auth-id: {token_id}' -H 'Authorization: {api_token}' -H 'Content-Type: application/json' -d '{"example": "value"}'
```

- Full cURL example

Read more...

```
curl --location '{base-URL}' \
--header 'Authorization:{API_KEY}' \
--header 'x-crtx-auth-id: {API_KEY_ID}' \
--header 'Content-Type: application/json' \
--data '{
    "$schema": "https://raw.githubusercontent.com/oasis-tcs/sarif-spec/master/Schemata/sarif-schema-2.1.0.json",
    "version": "2.1.0",
    "runs": [
        {
            "tool": {
                "driver": {
                    "name": "Veracode Static Analysis Policy Scan",
                    "rules": [
                        {
                            "id": "78",
                            "name": "Improper Neutralization of Special Elements used in an OS Command ('\\''OS Command Injection'\\')",
                            "shortDescription": {
                                "text": "CWE-78: Improper Neutralization of Special Elements used in an OS Command ('\\''OS Command Injection'\\')")
                            },
                            "helpUri": "https://cwe.mitre.org/data/definitions/78.html",
                            "properties": {
                                "category": "STATIC",
                                "tags": [
                                    "STATIC"
                                ]
                            },
                            "defaultConfiguration": {
                                "level": "error"
                            }
                        },
                        {
                            "id": "89",
                            "name": "Improper Neutralization of Special Elements used in an SQL Command ('\\''SQL Injection'\\')",
                            "shortDescription": {
                                "text": "CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('\\''SQL Injection'\\')")
                            },
                            "helpUri": "https://cwe.mitre.org/data/definitions/89.html",
                            "properties": {
                                "category": "STATIC",
                                "tags": [
                                    "STATIC"
                                ]
                            },
                            "defaultConfiguration": {
                                "level": "error"
                            }
                        }
                    ],
                    "version": "1.0"
                }
            },
            "results": [
                {

```

```
        "level": "error",
        "rank": 5,
        "message": {
            "text": "<span>This call to java.lang.ProcessBuilder.start() contains a command injection flaw. The argument to the function is constructed using untrusted input. If an attacker is allowed to specify all or part of the command, it may be possible to execute commands on the server with the privileges of the executing process. The level of exposure depends on the effectiveness of input validation routines, if any. start() was called on the processBuilder object, which contains tainted data. The tainted data originated from an earlier call to AnnotationVirtualController_vc_annotation_entry.</span> <span>Validate all untrusted input to ensure that it conforms to the expected format, using centralized data validation routines when possible. When using blocklists, be sure that the sanitizing routine performs a sufficient number of iterations to remove all instances of disallowed characters. Most APIs that execute system commands also have a \\"safe\\" version of the method that takes an array of strings as input rather than a single string, which protects against some forms of command injection.</span> <span>References: <a href=\"https://cwe.mitre.org/data/definitions/78.html\">CWE</a> <a href=\"https://owasp.org/www-community/attacks/Command_Injection\">OWASP</a></span>" },
        "locations": [
            {
                "physicalLocation": {
                    "artifactLocation": {
                        "uri": "com/scalesec/vulnado/Cowsay.java"
                    },
                    "region": {
                        "startLine": 16
                    }
                },
                "logicalLocations": [
                    {
                        "name": "Cowsay.java",
                        "fullyQualifiedName": "com.scalesec.vulnado.Cowsay.run",
                        "kind": "function"
                    },
                    {
                        "fullyQualifiedName": "java.lang.ProcessBuilder.start",
                        "kind": "member",
                        "parentIndex": 0
                    }
                ]
            }
        ],
        "ruleId": "78",
        "partialFingerprints": {
            "context_guid": "",
            "file_path": "",
            "procedure": ""
        }
    },
    {
        "level": "error",
        "rank": 4,
        "message": {
            "text": "<span>This database query contains a SQL injection flaw. The call to java.sql.Statement.executeQuery() constructs a dynamic SQL query using a variable derived from untrusted input. An attacker could exploit this flaw to execute arbitrary SQL queries against the database. The first argument to executeQuery() contains tainted data from the variable query. The tainted data originated from an earlier call to AnnotationVirtualController_vc_annotation_entry. </span> <span>Avoid dynamically constructing SQL queries. Instead, use parameterized prepared statements to prevent the database from interpreting the contents of bind variables as part of the query. Always validate untrusted input to ensure that it conforms to the expected format, using centralized data validation routines when possible.</span> <span>References: <a href=\"https://cwe.mitre.org/data/definitions/89.html\">CWE</a> <a href=\"https://owasp.org/www-community/attacks/SQL_Injection\">OWASP</a></span>" },
        "locations": [
            {
                "physicalLocation": {
                    "artifactLocation": {
```

```

        "uri": "com/scalesec/vulnado/User.java"
    },
    "region": {
        "startLine": 49
    }
},
"logicalLocations": [
{
    "name": "User.java",
    "fullyQualifiedName": "com.scalesec.vulnado.User.fetch",
    "kind": "function"
},
{
    "fullyQualifiedName": "java.sql.Statement.executeQuery",
    "kind": "member",
    "parentIndex": 0
}
]
],
"ruleId": "89",
"partialFingerprints": [
{
    "context_guid": "",
    "file_path": "",
    "procedure": ""
}
]
}
]
}
}'
```

- Python 3

```

import requests
import json

def generic_collector(token_id, api_token):
    headers = {
        "Authorization": {token_id},
        "x-crtx-auth-id": {api_token},
        "Content-Type": "application/json"
    }

    body = json.dumps({"example": "value"})
    res = requests.post(url="{API_URL_FROM_RESPONSE}?repository_id={repository_asset_id}&branch={branch_name}",
                         headers=headers,
                         data=body)
    return res
```

6. Validate the connector status.

- a. Navigate to Settings → Data Sources & Integrations → and enter 3rd Party AppSec Collector in the search bar.
- b. Select View Details.
- c. Check that your instance is displayed on the Collector Instances page, and that the status of your instance is Connected.

Manage 3rd Party AppSec Collector instances

1. Navigate to Settings → Data Sources & Integrations → and enter 3rd Party AppSec Collector in the search bar.
2. Select View Details.
3. Right-click on your instance → Edit instance.

You are redirected to the Generic 3rd Party AppSec Collector onboarding wizard to modify required content.

2.7 | Manage data source integrations

Manage integrations to align with evolving requirements and ensure they remain current.

1. Navigate to Settings → Data Sources & Integrations and search for the required data source integration.

TIP:

To quickly find your data source integration, use the search bar.

2. Hover over a data source and click View Details.

A list of instances for the selected data source is displayed.

3. Right-click on an instance and select an option:

- Edit instance: Redirects to the Select Repositories step of the integration wizard, where you can modify configurations for the selected instance. For more details, refer to the relevant integration guide
- Delete instance: When confirmed, deletes the instance, including data from previous scans

2.8 | Transporter over Broker VM

Abstract

Transporter over Broker VM sets up a secure communication channel between your VCS and Cortex.

The Transporter over Broker VM enables secure communication between your self-hosted Version Control Systems (VCS) and Cortex Cloud. This solution addresses the need for secure code scanning without exposing your internal network to the cloud.

Use cases

- **Enhanced security:** Establish a secure network tunnel for your self-hosted VCS, eliminating the need to open direct IP access to the cloud for scanning
- **Seamless Integration:** Leverage the existing Cortex Cloud Broker VM infrastructure to deploy and manage the Transporter applet
- **Simplified management:** Configure and manage the Transporter applet directly through the Cortex Cloud console
- **Automated updates:** The Broker VM automatically receives updates and enhancements, providing new capabilities to the Transporter without manual intervention

License

To gain access to the Transporter applet, you must have a Cloud license (Posture Management or Runtime Management) or a XSIAM Premium license.

WARNING:

The Transporter applet is not supported for FedRAMP customers.

Key components

The Transporter over Broker VM solution includes the following key components:

- **Transporter applet** This component runs within your internal network, specifically on a Broker VM. The applet requires access to your self-hosted version control providers (VCS) such as GitHub Server.

The Transporter:

- Establishes and maintains secure, long-lived connections to both your tenant and your VCS providers
- Operates based on events from the VCS provider or triggers initiated from your tenant
- **Broker VM:** In the Cortex Cloud Application Security environment, the Broker VM is a secured virtual machine (VM) designed to host the Transporter applet within your network. The Transporter applet is installed as an application or connector directly onto the Broker VM.

NOTE:

- The broker enables multiple connections, allowing a single Transporter applet to connect to various VCS instances
- Multiple Transporter applets cannot be created on a single Broker VM. Each Broker VM is limited to a single Transporter applet instance
- **Cortex Tenant:** Serves as the central cloud platform with several key functions in the Transporter system:
 - Acts as the cloud endpoint for the secure, long-lived WebSocket tunnel established by the Transporter applet running on your Broker VM
 - Links the Transporter applet to your self-managed VCS data sources, utilizing the secure tunnel for all communication.
 - Receives the data uploaded by the Transporter from your environment
 - Server as the scanner execution environment and results
 - Provides the interface for configuring and managing the Transporter system, as well as reviewing and managing the scan results
 - Monitors the status and health of your Transporter applets, providing visibility into their operation

Set up the Transporter

The order for setting up the Transporter solution is as follows:

PREREQUISITE:

- Ensure you have the necessary permissions and have already set up your Broker VM
- Confirm that your Broker is v 28 or above
- Whitelist IP addresses to enable access to Cortex Cloud resources. The IP addresses for the Transporter are in the Broker VM Resources section of the Enable access to required PANW resources document
- Open port 4052 (inbound), which is required for the Transporter's IP address communication
- Open Port 443 (outbound), which is required for the Broker VM to pull data from the your version control system (VCS)

1. Set up the Transporter applet on your Broker VM.

2. Onboard the Transporter on your VCS.

2.8.1 | Set up a Transporter applet on Broker VM

Abstract

Setup a Transporter applet on Broker VM.

This section describes how to set up and configure a Transporter applet on a Broker VM.

PREREQUISITE:

Permissions: To configure and manage Transporter applet settings, you must have permissions to manage Broker Service configurations (such as an Instance Administrator)

1. Setup a Broker VM.

Refer to Set up and configure Broker VM for more information.

2. Select Settings → Configurations → Broker VMs.

3. Select the Brokers tab → locate your Broker VM → hover and click + Add under the Apps column → AppSec Transporter.

A connector for AppSec Transporter is opened.

NOTE:

You cannot add a Transporter applet through Clusters.

4. Configure the Transporter connection in the provided fields:

- Transporter Name (required). Requires a unique name as you can integrate multiple applets for different integrations
- Provider Self Signed CA Certificate Path: Specify the file path for a custom Certificate Authority (CA) certificate used by the Transporter to securely communicate with services

5. Click Save.

6. Verify integration and connectivity: Locate your AppSec Transporter applet under the Apps column. Click it to confirm it displays a Connected status.

Proceed to configure the Transporter applet on your self-managed VCS data sources instance.

Add Connections

The broker enables multiple connections, allowing a single Transporter applet to connect to various VCS instances (such as GitHub and GitLab).

1. Select your AppSec Transporter under the App column → Configure → Add Connection (on the redirected AppSec Transporter setup page).
2. Repeat **steps 4-6** of Set up a Transporter applet on Broker VM above.

Manage Transporter applets

To manage Transporter applet configurations, disable connections, or deactivate an applet, navigate to the Broker VMs page. From there, select your Appsec Transporter under the App column.

- **Edit applet configurations:** Select the Appsec Transporter under the App column → Configure. You are redirected to the Transporter applet settings to manage its configurations
- **Disable applet connection for a single integration:**

1. Select the Appsec Transporter under the App column → Configure.
2. On the Transporter applet configurations page, click on the specific Transporter applet → Disable.

This disables the specific integration, but it can be re-enabled.

- **Deactivate an applet (all connections):** Select the Appsec Transporter under the App column → Deactivate → Confirm when prompted

All existing connections are deleted but their configurations are saved in the database. When adding a new connection, you'll be prompted if you want to reuse previous configurations.

2.8.2 | Set up a Transporter on your VCS

Abstract

Setup a Transporter on your version control system.

You configure a Transporter when onboarding your on-premises version control system (VCS). This setup is performed on the Configure Domain step of the onboarding wizard. GitHub Server, GitLab self-managed and Bitbucket Data Center version control systems support the Transporter integration, where it functions as a network tunnel.

PREREQUISITE:

- Ensure you have user permissions to onboard data sources.

NOTE:

The dedicated AppSec Admin role includes the required permissions.

- Before you begin you must setup a Transporter applet on your Broker VM. Refer to Set up a Transporter applet on Broker VM for more information

1. Onboard your VCS on-premises data source: Select Settings → Data Sources & Integrations → + Add New → , and search for your VCS on-premises data source.

2. Configure the Transporter on the Configure Domain step of the wizard:

- a. Select your Broker VM from the provided menu.
- b. Select the Transporter applet associated with the selected Broker VM from the Transporter Connection menu.

3. Complete the steps required to onboard the VCS data source: Refer to Onboard version control systems for more information about onboarding VCS data sources.

NOTE:

GitLab Enterprise webhook internal IP restriction: GitLab Enterprise's security policies prevent webhook subscriptions to internal IP addresses (such as broker addresses). These subscriptions can only be enabled through specific instance configuration. For more information, refer to <https://docs.gitlab.com/security/webhooks/>.

3 | Cortex Cloud Application Security dashboard

Abstract

Monitor and analyze your Application Security posture with the Application Security dashboard.

The Cortex Cloud Application Security dashboard offers AppSec practitioners a centralized platform for monitoring and analyzing your organization's security posture. It provides visibility into your software development lifecycle assets, and offers insights into issues detected in your environment, helping you to prioritize and remediate security risks.

How to access the Cortex Cloud Application Security dashboard

You can access the dashboard from either the Command Center or the Application Security module.

- From the Command Center: Select Application Security from the drop-down menu
- From the Application Security module: Select Modules → Application Security → AppSec Dashboard

Dashboard controls

- **Page-level filters:** You can filter dashboard data by Applications, Repositories, Pipelines and Backlog status. When you apply a filter, all widgets on the dashboard update to display data relevant to the selected items If you select multiple values (for example, several repositories or pipelines), the dashboard aggregates and displays combined data for all selected items
- **Widget-specific filters:** You can manage the data displayed in each widget through filters built into the widgets

Dashboard assets

The Cortex Cloud Application Security dashboard provides an overview of the applications, repositories and CI/CD pipelines in your environment. It displays the total count of each asset type, along with the number of critical and high issues associated with them. Selecting the number in an asset card redirects to the asset page.

Actions

- Select a number in an asset card to redirect to the corresponding asset page
- Select the Critical or High severity level in a Repositories or Pipelines asset card to redirect to the main Issues inventory. The inventory is filtered by AppSec Issues, the scanner type (for Repositories assets, all Cortex Cloud Application Security scanner types; for CI/CD Pipeline assets, CI/CD Risks, and the selected severity level
- Click Create Application from the Applications asset card to redirect to the Application Builder, where you can define and set up a new application. For more information, refer to How to manually build an application

Standard widgets

- **Top Issues to Address**

Displays the highest priority issues to address across all scanner types, including their severity level, the total number of times each issue was detected, and labels providing context, such as whether the issue can be fixed, whether it was found in a public repository, if an issue is part of the top 10 OWASP risks and so on.

- **Actions:**

- Selecting an issue redirects to the main Issues inventory, filtered by the issue
 - You can filter issues by type of scanner (such as IaC) to display the top issues to address in a specific scan category. When selecting a scanner, the Show All option is available, allowing you to redirect to the main issue inventory displaying all issues according to the scan type

- **Open Issues by AppSec Scanner**

Displays the total amount of open issues with critical and high severity, along with a breakdown by scanner category.

- Actions:**

- Selecting a scanner type (such as Secrets) redirects to the corresponding Cortex Cloud Application Security issues inventory, filtered to show open issues with critical and high severity by default
 - You can filter the graph by severity level

- **Open Issues by Urgency and Scan Type**

Displays the total amount of open issues by Urgency (a context-aware metric to help you focus remediation efforts on the issues that pose the greatest real-world risk in your code) and the category of scanner.

- Actions:**

- You can filter the scanner graph by urgency level
 - Selecting a scanner type redirects to the corresponding Cortex Cloud Application Security issues inventory, filtered to show open issues with the selected urgency level

- **Open Issues by Severity**

Displays the total number of open issues, including a breakdown by severity level with the count for each

- Actions:**

- You can filter the widget by scanner type to view open issues in a specific scan category
 - Selecting a severity level (such as Critical) redirects to the main Issues inventory, filtered by Critical issues detected by the type of Cortex Cloud Application Security scanner selected in the dashboard

- **Open Issues by SLA**

Displays the number of open issues that are approaching or past their SLA, grouped by scanner type (for example, Vulnerabilities).

Actions:

- Selecting a scan type redirects to the corresponding Cortex Cloud Application Security issues inventory, filtered to show the issues that are approaching or past their SLA
- Filter the graph by Overdue or Approaching SLA to display only open issues of the selected SLA type

- **Top Policies to Address**

Displays the top policies across all scanner types that resulted in the highest number of detected issues.

This widget is organized into three tabs:

- Issue generators: Displays policies that generated the most issues in your code or configurations. Includes the total amount of critical and high issues as well as context such as the amount of issues that have fixes and the types of scanner that detected the issues
- Build blockers: Displays policies that resulted in the highest count of failed builds, providing information about the number of builds blocked by the policy, and the scan type (such as vulnerability scanner) that caused the build to fail. Selecting a policy from the list opens its side car on the AppSec Policies page for more details
- PR blockers: Displays policies that resulted in the highest number of blocked pull requests (PR), including information about the amount of builds blocked by the policy, and the scan type (such as IaC scanner) that blocked the PR

Actions:

- Selecting a policy opens its side car on the AppSec Policies page for more details
- You can filter to view top policies at risk in a specific scan category
- Click Show All to redirect to the AppSec Policies page

- **Top Repositories at Risk**

Displays the repositories with the highest count of critical and high issues.

Actions:

- Selecting a repository opens the main Issues page filtered by critical and high issues detected in the repository
- You can filter to view top repositories at risk in a specific scan category

- **Top Pipelines at Risk**

Displays the pipelines with the highest count of critical and high issues.

Actions:

- Selecting a pipeline opens the main Issues page filtered by critical and high issues detected in the pipeline

4 | Application Security Posture Management (ASPM)

Abstract

ASPM centralizes AppSec monitoring across the SDLC. It aggregates findings from tools such as IaC and SCA to provide a holistic view and prioritize risks.

ASPM is a comprehensive and integrated approach designed to centralize and automate the continuous monitoring, evaluation, and enhancement of an organization's application security throughout its entire software lifecycle - from initial code development through deployment and ongoing operations in cloud or on-premises environments.

ASPM functions as a unified governance layer. It aggregates, correlates, and assesses security signals and findings from various application security testing tools (such as SAST, DAST, and SCA) and other critical data sources, providing a holistic, real-time, and actionable view of an application's overall security landscape. This unified approach directly addresses challenges like fragmented visibility, siloed security teams, inefficient resource allocation, and delayed remediation.

ASPM secures your applications by treating each as a dynamic, logical entity, encompassing its complex array of components. This includes custom code, open-source libraries, APIs, microservices, Infrastructure as Code (IaC) configurations, runtime environments (like containers), and data flows. By identifying and prioritizing these applications as single entities, ASPM enables focused monitoring and safeguarding of high-priority assets across the entire Software Development Lifecycle (SDLC), including VCS repositories, CI/CD pipelines, and container registries. This ensures you can effectively identify, prioritize, and remediate critical issues that could impact key business systems, thereby continuously reducing the risk of vulnerabilities and breaches.

4.1 | ASPM use cases

The primary use cases for the ASPM platform include:

- **Comprehensive visibility:** Gain a holistic, unified view of your application security posture across all stages of the software development lifecycle (SDLC), enabling you to identify and address vulnerabilities and misconfigurations
 - **Applications:** insights into the SDLC of your critical business applications
 - **Code to cloud:** Visualize and understand the relationship between your source code and deployed cloud resources, enabling you to identify and prioritize risks associated with your deployments
 - **CI/CD systems:** Monitor and analyze the security configurations and activities within your CI/CD pipelines to identify and mitigate risks introduced during the build and deployment processes
 - **Third party ingestion:** Integrate security findings from third-party scanners and security tools to gain a centralized view of your security posture and correlate findings across your development lifecycle
- **Contextual risk prioritization and proactive detection:** Prioritize remediation efforts based on a data-driven risk assessment that combines code-level vulnerabilities, runtime behaviors, and infrastructure configurations. Proactively detect critical issues such as exposed application secrets, IaC misconfigurations, SCA CVE vulnerabilities, package operational risks, and license miscompliance, all within the context of your application's architecture and potential impact on business operations. This allows you to focus on the most impactful threats and address them before they are exploited, ensuring a robust security posture
- **Effective prevention:** Enforce security policies and prevent security risks from impacting your applications
- **Actionable remediation:** Improve your security posture with actionable remediation guidance for identified security risks. The platform offers automated remediation for IaC misconfigurations and CVE vulnerabilities, in addition to clear steps for manual fixes for all categories of detected risks

4.2 | ASPM key features

The Cortex Cloud ASPM solution provides the following key features to help you gain comprehensive control over your application security posture:

- **Command Center:** A central dashboard providing a real-time overview of your organization's Application Security program, including security coverage, issue distribution (total, prevented, prioritized), and identification of riskiest applications
- **Coverage:** Provides detailed visibility into the security monitoring status of Application Security assets (such as VCS repositories and CI/CD pipelines) by security scanners, highlighting coverage percentages, gaps, and scanner success/failure rates
- **Backlog Management:** Enables categorization of issues and findings as New or Backlog (technical debt), allowing for differentiated management, filtering, and policy enforcement to prioritize new vulnerabilities
- **Application Builder:** Facilitates the definition and management of applications, automatically discovering and associating all relevant assets across the SDLC (from code to cloud) to provide a centralized and holistic view of application risk
- **Application side card:** Offers a unified, high-level summary of risks across the full application lifecycle, aggregating insights from multiple security domains, displaying key risk metrics, and showing application topology
- **Code-to-Cloud context:** Provides end-to-end visibility by mapping code-level issues (such as IaC misconfigurations and CVEs) to deployed assets and runtime issues, enabling full lifecycle insight from development through CI/CD to production
- **Generic 3rd party SARIF Collector:** Automates the ingestion of findings from diverse third-party scanners directly into the platform via SARIF file uploads, providing unique API access, real-time ingestion status, and detailed execution logs to streamline security workflows

4.3 | ASPM user roles and permissions

The AppSec Admin role is the dedicated user role for ASPM, granting full permissions for all application security-related activities. They can create and modify detection rules within the Code/Build domain, track progress, and adjust enforcements as needed. Additionally, they can triage and investigate findings, issues, and cases spanning from code to cloud. The role also includes complete visibility into all cloud assets.

Permissions assigned to the AppSec Admin role cannot be modified. However, you can save this role as a new custom role which can then be edited to meet specific organizational needs, offering a balance between standardized roles and customizable access control.

You can view AppSec Admin permissions in the tenant by navigating to Settings → Configurations → Roles → AppSec Admin.

4.4 | Code to Cloud

Abstract

Code to Cloud context maps asset lineage across the SDLC. By connecting repos, pipelines, and infra, it provides end-to-end traceability from code to runtime.

Code to Cloud context is the correlation engine that maps and maintains the full lineage of assets across the SDLC. By connecting repositories, pipelines, and infrastructure, it provides end-to-end traceability from source code to runtime resources.

Full Code to Cloud (C2C) coverage is achieved when Cortex Cloud can resolve the following chain:

Repository → Pipeline → Image → (optional Registry) → Runtime resources (including VMs, VM images, and IaC-defined infrastructure). This lineage mapping connects repositories, pipelines, images, VMs, VM images, IaC-defined infrastructure, and runtime assets back to their originating code, pipelines, IaC resources, and OSS packages—providing full lifecycle context rather than isolated runtime visibility.

When relationships cannot be resolved—for example, due to missing YOR tags or pipeline integrations, Cortex Cloud detects coverage gaps and provides specific configuration steps to fix them. If any part of this chain is unsupported or disconnected, the tenant provides only a partial view, isolating code-side or cloud-side components without end-to-end links. This limits bidirectional impact analysis: code issues cannot be traced to runtime for prioritization by actual exposure, and vulnerabilities cannot be traced back to their source code or owner.

By establishing deterministic links between code, build artifacts, and runtime infrastructure, Code to Cloud context enables the following:

- **Bidirectional traceability:** Trace runtime issues back to the specific line of code, developer, or pipeline that introduced them
- **Contextual application grouping:** Automatically groups related assets into business applications, allowing you to manage risk based on actual business impact
- **Precision prioritization:** Prioritize remediation based on actual exposure, such as whether a vulnerability is active or internet-facing, and its potential business impact
- **Drift detection:** Compare the intended state defined in Infrastructure-as-Code (IaC) templates against the actual state of runtime resources to identify unmanaged changes.
 - **For IaC drift detection policies:** To configure the security baselines and specific rule mappings that drive this detection logic, see Create IaC Drift Detection policies
 - **For IaC Drift Detection issues:** To investigate, prioritize, and remediate the specific runtime discrepancies identified by these scans, see IaC Drift Detection scans

Core components and mechanisms

- **Asset Lineage Graph:** This queryable graph database automatically maps the relationships between upstream assets (code repos, packages, IaC resources) and downstream assets (container images, virtual machines, runtime workloads, and cloud services). It is populated by ingesting metadata from:
 - **VCS and CI/CD:** Captures repository metadata, build logs, and pipeline run data
 - **Build artifacts:** Extracts deterministic links (such as image digests) from container and VM image builds
 - **Runtime Scanners:** Maps running workloads back to their build sources via the Cloud or Kubernetes connectors
- **Infrastructure-as-Code (IaC) Traceability (YOR):** To bridge the gap between static IaC files and dynamic cloud resources, Cortex Cloud leverages **YOR** tags:
 - **Automated mapping:** YOR applies unique trace tags to IaC resources. Cortex Cloud uses these tags to link a Terraform or CloudFormation template to the specific cloud asset it provisioned (for example, IaC Resource → Cloud Asset)
 - **Gap analysis:** If YOR tags are missing, the Asset Lineage Graph cannot complete the link. The system will prompt you to initiate tagging via the yor website to unlock full visibility
- **Drift detection logic:** Code to Cloud enables drift detection by treating Git as the single source of truth. The system correlates the declared state (from VCS) with the runtime state (from CSPM integrations). Drift is only flagged when a runtime change violates a security policy that is not violated in the source code, ensuring focus on security-relevant regressions rather than operational noise

4.4.1 | Supported integrations

To build the asset lineage, Cortex Cloud requires integration with specific providers across your stack.

Component	Supported Provider/Build Tool
VCS	GitHub, GitLab, Bitbucket, Azure DevOps
CI/CD	GitHub Actions, GitLab CI/CD, Azure Pipeline, Jenkins, CircleCI
Containers	Docker CLI → Container image, docker compose, docker buildx, Kaniko
VM images	AWS EC2 Image Builder, Packer → AWS AMI
IaC	Terraform (.tf), CloudFormation (.yml, .json)

4.4.2 | Code to Cloud visibility

Abstract

Manage security risk across the SDLC by tracing technical asset lineage. View asset dependencies and runtime context in assets, issues, and policies.

Code to Cloud context is integrated throughout the user interface to help you visualize dependencies and enforce security.

- **Topology graph:** Located in the **Business Applications** side card, this tab visualizes the entire path to production (**Code → Build → Deploy → Run**), allowing you to see how assets are interconnected
- Dedicated Cortex Cloud Application Security asset inventories: Repository, Software Package and IaC Resources side cards include Code to Cloud tab. This graph maps the specific asset to its upstream source and downstream runtime deployments
- Issue investigation: Vulnerabilities and IaC Misconfiguration issues include a Code to Cloud tab. This view traces the specific defect from the code file to the impacted runtime resource, helping verify if a vulnerability is actively deployed
- **Policy enforcement:** Policies can be configured with runtime conditions. For example, you can block a build only if the detected vulnerability affects an asset that is destined for an internet-facing environment. For more information on creating policies, refer to Create Cortex Cloud Application Security policies

4.4.3 | Code to Cloud troubleshooting

If the Code to Cloud lineage is incomplete, specific signals may be missing. Common issues include:

- **Missing YOR Tags (IaC Resources):** IaC resources without tags cannot be mapped to runtime. The system will prompt you to tag these resources
- **Missing Pipeline Integrations (Repositories):** If pipeline integrations are missing, the link between code and build artifacts breaks
- **Inactive Pipeline:** Lineage is generated during pipeline runs. If a pipeline is integrated but has not run, trigger a build to generate the necessary artifacts and establish the connection

NOTE:

Empty state messages and troubleshooting guidance for missing lineage are only visible to users with an active license.

4.5 | ASPM Command Center

Abstract

The ASPM Command Center is your central hub for real-time application security posture management across the SDLC. It offers critical insights to identify risks, track compliance, and enable secure development, transforming data into actionable cases.

The ASPM Command Center acts as a central hub for managing your application security posture across the entire Software Development Lifecycle (SDLC), from code to cloud. It offers a high-level, real-time view of your organization, providing critical insights into:

- **Overall security trends:** Track how your application security posture is evolving over time
- **Policy compliance:** Monitor adherence to your security policies throughout the SDLC
- **Coverage:** Gain visibility into scanned and unscanned assets, helping you identify and address security gaps
- **Critical risks:** Quickly identify and understand your most significant security threats

The ASPM Command Center provides a single, centralized view to track progress, prioritize actions, and explore key areas such as vulnerabilities, security coverage, and issue prioritization. It visualizes the data flow of issues from various sources—through blocking and prioritization—to their aggregation into actionable cases, helping you identify critical gaps, reduce risk, and focus on the most significant threats.

Use cases

- **Data source and issue overview:** View all your connected data sources including version control systems (VCS), data ingested from 3rd-party sources, CI/CD pipelines, and registries, along with the issues generated by Cortex Cloud from findings detected within these data sources
- **Coverage:** Assess your total security coverage of scanned assets, including coverage per data source, providing immediate insight into the completeness of your security monitoring efforts and highlighting areas where coverage needs to be improved
- **Deduplication and prioritization:** Analyze security issues by determining which were blocked by guardrails and which were deduplicated and aggregated for further action. This streamlines the volume of issues, ensuring your focus remains on the most impactful threats by presenting a refined set of prioritized issues
- **Riskiest applications:** Identify your applications most at risk. This view presents a list of critical applications and the overall count of high and critical cases across all applications, enabling rapid prioritization of top security concerns

Key performance indicators (KPIs) and widgets

The ASPM Command Center also features Key Performance Indicators (KPI) and widgets for a high-level overview:

- Total Coverage widget: Shows the percentage and absolute number of total assets scanned. Clicking this widget redirects you to the AppSec Coverage page
- Riskiest Applications widget: Presents a list of your most vulnerable applications, detailing the number of cases and their criticality. Selecting an app from this list opens the application case side panel directly within the ASPM Command Center, without the need to leave the command center and navigate to the dedicated Cases page
- AI Guardrails: AI-recommended guardrails help shift your security posture from detection to prevention by analyzing scan findings and suggesting high-impact blocking policies. Recommendations address both protecting known-good code (Lockdown) and reducing recurring risk patterns (Stop the bleeding), and are prioritized using context-aware analysis of deployed environments and backend risk scoring. Select Explore to redirect to the AppSec Policies page, where you can review, customize, and enforce the recommended guardrails

Refer to AI-recommended guardrails for more information.

For tailored analysis, the ASPM Command Center Overview page includes severity filters, allowing you to refine the displayed data based on the criticality of issues.

4.5.1 | ASPM Command Center workflow

The ASPM Command Center is designed to provide a comprehensive, interactive, and actionable overview of your application security posture. It streamlines the complex journey of security findings from their origin, through prioritization and aggregation, to resolution, empowering you to maintain a strong application security posture.

The ASPM Command Center is an interactive graph that visually represents your application security workflow, moving from general data sources to a prioritization funnel, to specific, actionable cases. This visualization helps you quickly understand your security posture.

Data sources and coverage

The first section of the graph focuses on your data sources, including both onboarded and third-party sources, categorized into CI/CD Pipelines and VCS & 3rd Party data sources. In this section interactive elements provide more detailed information:

- **Data source overview:** Hovering over a data source provides a quick overview of its coverage
- **Total coverage insights:** The More Coverage Details tab opens a dedicated Total Coverage page within the ASPM Command Center. This page displays the overall amount and percentage of each data source type out of your total, along with a granular breakdown of scanner coverage (SAST, SCA, Secrets, IAC), showing the same detailed metrics
- **Increase coverage:** The Click to Increase Coverage link directs you to the AppSec Coverage page, which enables you to enhance your security coverage
- **Issue count:** Each data source also displays the number of issues detected

Prioritization and aggregation funnel

Clicking on the prioritization and aggregation part of the graph provides a dedicated view within the ASPM Command Center, displaying the security funnel. This page includes:

- Issues: The initial volume of security issues

NOTE:

Breakdown by type: Issues are further broken down by type (such as IAC or Secrets).

- Open After Guardrails: The amount of issues that persist after being filtered by your security guardrails, including those blocked in PRs
- Prioritized: Issues refined by parameters such as context, impact, probability, and issues not found in deployed applications
- Cases generated: At the end of this funnel, the number of cases generated post-prioritization and aggregation are displayed, with a breakdown prioritized by application, type, and severity

Case management

In the ASPM Command Center Overview page, the final section of the graph summarizes your cases: It displays the total count of open and closed cases. Open cases are further broken down by critical and high severity, alongside a summary of closed cases.

4.6 | Coverage

In Application Security Posture Management (ASPM), Coverage provides a comprehensive overview of your security posture across the SDLC (code, build, deploy). It offers visibility into scanned and unscanned assets, active scanners, and implemented guardrails, enabling you to identify and address security gaps for continuous protection.

Use cases

- **Data source coverage:** Identify onboarded and partially onboarded version control systems (VCS) and third-party integrations, ensuring all relevant data is available for analysis
- **Scanner coverage:** Determine which Static Application Security Testing (SAST) (sourced solely from third-party ingestion), Software Composition Analysis (SCA), Secrets, Infrastructure as Code (IaC), CI/CD misconfiguration and security posture management (SPM), and malware (image scans) are actively scanning your codebase and build
- **Guardrail coverage:** Understand which security policies and guardrails are applied at each stage, and assess their effectiveness
- **Stage-specific maturity:** Evaluate the coverage and maturity of your security posture at each stage of the SDLC, enabling targeted improvements
- **Application-specific coverage:** Gain a granular view of your application's security coverage, including data sources, scanners, and policies applied at each stage
- **Global and application scores:** Understand the global and application-specific security coverage scores, reflecting the overall maturity of your security posture

4.6.1 | Coverage in the user interface

To access Coverage, navigate to Modules, → Application Security → AppSec Coverage.

This interface offers a comprehensive overview of your application security coverage, presenting key metrics and visualizations related to application data sources, scanners, and guardrails. Interactive widgets provide a summary of your coverage. When you apply a filter through a widget, all data displayed on the dashboard, including the asset inventory, will dynamically reflect the selected filter criteria. The asset inventory provides detailed information about application assets in your SDLC for in-depth analysis of your security posture. You can also see all issues for specific assets by selecting them in the inventory table.

Application-specific coverage

You can focus on the security posture of your critical business applications, allowing you to prioritize remediation efforts for your most important assets. To view application-specific data, select Add Filters → Applications → enter the unique application name as provided when creating it.

Coverage by data source

This widget provides metrics based on the coverage of the data source such as version control systems (GitHub and so on), CI tools (Jenkins and so on), Repositories (JFrog) and third party data sources (such as Veracode). In addition, insights are provided, such as the amount of assets added recently or whether a data source is not connected.

Coverage by status

This widget provides coverage metrics based on the percentage of scanned repositories out of the total amount of repositories. Values: scanned, partially scanned, unscanned.

Coverage by scanner type

This widget provides metrics based on the coverage of the scan types, including code scanners (vulnerabilities, code weaknesses, secrets, IaC misconfigurations) and images (malware).

Asset coverage inventory table

The asset coverage inventory table displays a list of assets. Table properties include:

Property	Description
Asset Type	The type of asset scanned, such as repositories or container image repositories
Name	The name of the scanned asset
Applications	The type of applications associated with the asset

Property	Description
Scanners Data	Informations about the scanners that were applied to the asset. Upon hovering on the scanner, additional data the type of scanner and its status; enabled or disabled
Last scan status	The status of the last scan: Completed, not scanned yet, in progress and error

4.7 | Compliance for Cortex Cloud Application Security

Cortex Cloud Application Security integrates compliance controls directly into your development ecosystem, enabling a shift-left approach to regulatory adherence. The framework maps **Infrastructure as Code (IaC)** misconfigurations and **CI/CD security risk** findings to supported industry standards.

The compliance framework automatically maps IaC misconfiguration rules (such as unencrypted storage or open security groups) and CI/CD security risks (such as insecure runner configurations or lack of branch protection) to supported industry standards. This enables targeted reporting and granular filtering by specific controls, such as CIS Benchmarks or OWASP requirements.

By configuring automated policies to alert developers or block builds when violations occur, you ensure that neither vulnerable infrastructure nor insecure delivery pipelines reach production.

Monitor and track compliance adherence

Monitor and track compliance adherence for your infrastructure code and CI/CD pipeline assets by checking whether your templates and configurations adhere to industry standards or your organization's best practices.

To view compliance-related details, navigate to Posture Management → Compliance.

For more information about managing compliance in Cortex Cloud, including assessments and reports, refer to Monitor and track compliance adherence.

4.7.1 | Infrastructure-as-Code (IaC) compliance

IaC compliance focuses on the security posture of your cloud resource definitions (Terraform, CloudFormation) before deployment. By analyzing templates, Cortex Cloud identifies misconfigurations that violate specific regulatory frameworks.

Supported IaC compliance standards

The IaC scanner maps findings to the following compliance standards and frameworks:

Industry standards

- PCI DSS (Payment Card Industry Data Security Standard)
 - PCI DSS v3.2.1 - Payment card data protection requirements
 - PCI DSS v4.0 - Latest PCI DSS requirements
 - PCI DSS v4.0.1 - Updated PCI DSS v4.0 requirements
- NIST (National Institute of Standards and Technology)
 - NIST 800-53 Rev4 - Security and privacy controls for federal information systems
 - NIST 800-53 Rev 5 - Updated security and privacy controls
 - NIST SP 800-171 Revision 2 - Protecting Controlled Unclassified Information
 - NIST SP 800-171 Revision 3 - Latest CUI protection requirements
 - NIST SP 800-172 - Enhanced security requirements for CUI
 - NIST CSF - Cybersecurity Framework
 - NIST CSF v2.0 - Updated Cybersecurity Framework
- ISO Standards
 - ISO 27001:2013 - Information security management systems
 - ISO/IEC 27001:2022 - Latest information security management standard
- HIPAA (Health Insurance Portability and Accountability Act)

Security and privacy requirements for healthcare data
- GDPR (General Data Protection Regulation)

European Union data protection and privacy requirements
- SOX (Sarbanes-Oxley Act)

Financial reporting and corporate governance requirements
- CCPA (California Consumer Privacy Act)

California data privacy requirements

Cloud provider benchmarks

- CIS (Center for Internet Security) Benchmarks

- AWS:
 - CIS v1.2.0 (AWS)
 - CIS AWS 3 Tier Web Architecture Benchmark v.1.0.0
- Azure:
 - CIS v1.1 (Azure)
 - CIS v1.2.0 (Azure)
 - CIS v1.3.0 (Azure)
 - CIS v1.3.1 (Azure)
 - CIS v1.4.0 (Azure)
 - CIS v1.5.0 (Azure) - Level 1
 - CIS v2.0.0 (Azure) Level 1
 - CIS v2.1 (Azure) Level 1
 - CIS v2.1.0 (Azure) Level 1
- GCP (Google Cloud Platform):
 - CIS v1.0.0 (GCP)
 - CIS v1.1.0 (GCP)
 - CIS v1.2.0 (GCP)
 - CIS v1.3.0 (GCP)
 - CIS v2.0.0 (GCP) Level 1
 - CIS v3.0 (GCP) Level 1
 - CIS v3.0.0 (GCP) Level 1
 - CIS v4.0.0 (GCP) Level 1
- GKE (Google Kubernetes Engine):
 - CIS v1.1.0 (GKE)
 - CIS v1.2.0 (GKE)
 - CIS v1.3.0 (GKE) - Level 1
 - CIS v1.4.0 (GKE) - Level 1
 - CIS v1.5.0 (GKE) - Level 1
- OCI (Oracle Cloud Infrastructure):

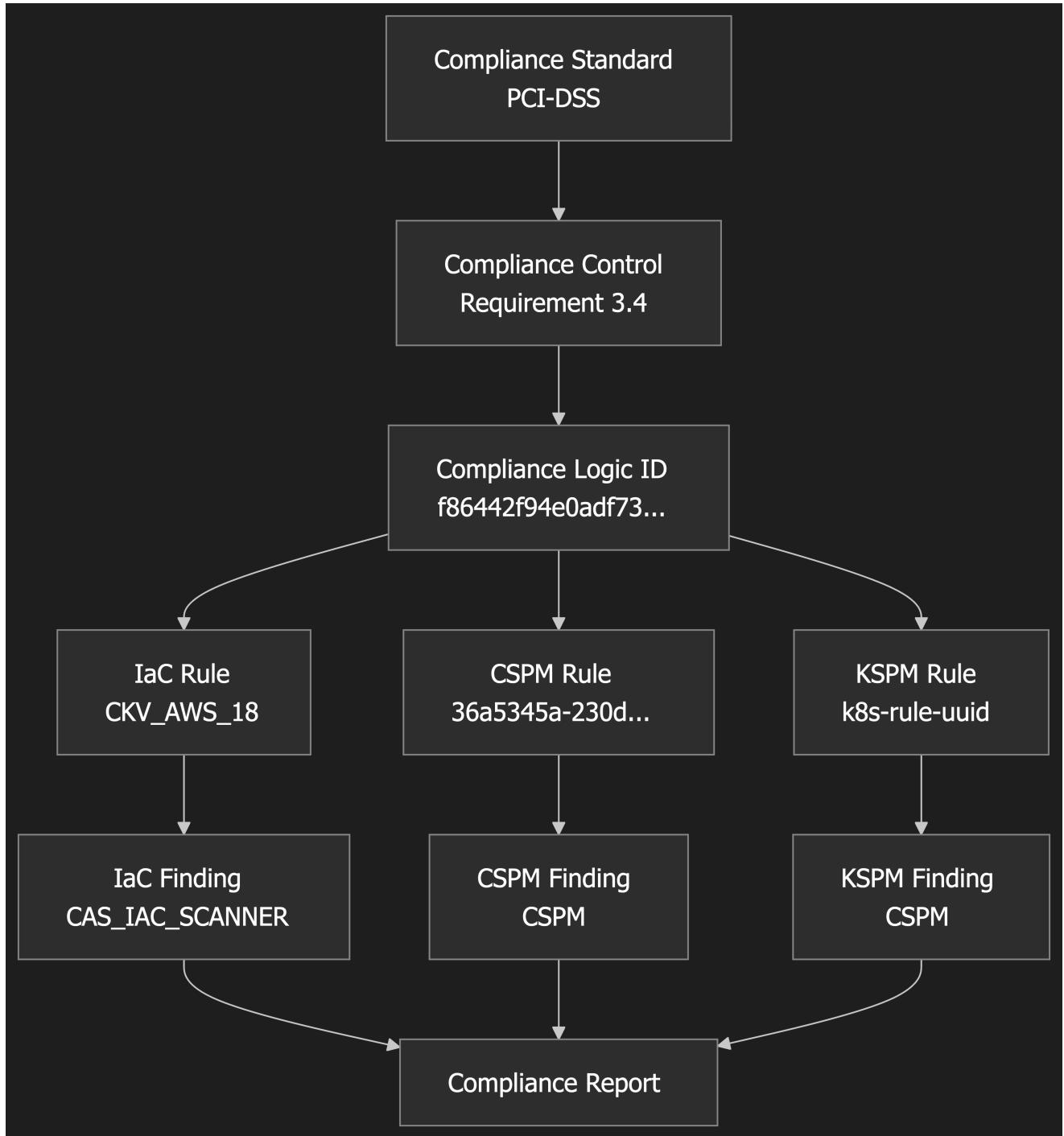
- CIS v1.2.0 (OCI)
 - CIS v2.0.0 (OCI) - Level 2
 - CIS v3.0.0 (OCI) - Level 2
- CIS Controls
 - CIS Controls v7.1 - Implementation groups for cybersecurity
 - CIS Controls v8 - Updated cybersecurity controls
 - CIS Controls v8.1 - Latest CIS Controls version

Rule mapping logic

To ensure consistency between build-time and run-time security, IaC compliance relies on a unified mapping logic.

IaC rules in Cortex Cloud are mapped to corresponding runtime CSPM (Cloud) or KSPM (Kubernetes) rules.

If an IaC rule is mapped to a runtime rule, it automatically inherits the compliance standards and controls associated with that runtime rule. This ensures that a violation detected in code (IaC) is categorized under the same compliance control as if it were detected in the cloud.



Scan types

IaC compliance scanning is available in the following scan types:

- Periodic scans: scheduled repository scans
- PR scans: pull request validation
- Branch scans: branch-specific scans
- External project scans: third-party project analysis

4.7.2 | Manage IaC compliance

Manage IaC compliance assessments and reports directly in the tenant to generate and download audit-ready compliance evidence. You can view mapped rules, enforce compliance via policies, and filter issues by specific regulatory controls.

[View IaC compliance rules mapped to compliance standards and controls](#)

You can view and modify compliance standards mapped to specific IaC rules in AppSec Rules to control which rules are evaluated for compliance and ensure that findings are correctly attributed to the intended compliance framework.

1. Navigate to Modules → Application Security → AppSec Rules.
2. Filter the table by IaC -supported Compliance Standards OR Compliance Controls attributes.

PREREQUISITE:

Add these properties to the IaC Rules table through the Table Settings Menu, as they are not exposed by default.

[Create Cortex Cloud Application Security policies with IaC compliance conditions](#)

Create policies to include or exclude findings based on specific IaC compliance standards and controls. This provides precise control over automated issue creation and build-blocking.

1. Navigate to Modules → Application Security → AppSec Policies → Add Policy.
2. Follow the standard procedure in the policy wizard. The configuration for all steps remains the same, except for the Conditions step.
3. On the Conditions step of the wizard.
 - a. Apply a compliance filter: Select either Compliance Standard or Compliance Control as the attribute.
 - b. Select the required values for the standard or control.

[Manage IaC compliance issues and findings](#)

IaC Compliance issues and findings are found under the respective IaC Misconfigurations tables. You can filter IaC misconfiguration findings and issues by compliance standards and controls to isolate risks relevant to specific regulatory frameworks. This allows you to prioritize remediation based on your organization's required security controls.

PREREQUISITE:

Add these properties to the tables table through the Table Settings Menu, as they are not exposed by default.

1. Navigate to Modules → Application Security → IaC Misconfigurations (under Issues).

2.
 - For IaC Compliance issues:
 - Filter the table by IaC-supported Compliance Standards OR Compliance Controls attributes.
 - View Compliance Standards and Compliance Controls in the Issues table by adding these properties through the Table Settings Menu.
 - The IaC issues side card includes a Compliance Standards tab with additional details about IaC Compliance Standards and Controls.
 - For IaC Compliance findings:
 - Select the Findings tab in the IaC issues page.
 - View IaC-supported Compliance Standards and Compliance Controls in the Findings table by adding these properties through the Table Settings Menu.
 - The Compliance Standards tab in the IaC findings side panel displays the specific standards and controls mapped to the Cortex Cloud Application Security rules.

4.7.3 | CI/CD Compliance

Abstract

CI/CD compliance ensures adherence to industry standards: CIS GitLab/GitHub and OWASP Top 10.

Cortex Cloud enhances your CI/CD compliance posture by assessing pipeline asset adherence to industry standards and your organization's best practices. Cortex Cloud supports compliance checks against the CIS GitLab Benchmark v1.0.1, CIS GitHub Benchmark v1.0.0, and the OWASP Top 10 CI/CD Risks v2025.

To access Compliance, select Posture Management → Compliance → Reports or Assessment.

4.7.3.1 | Create CI/CD compliance reports

The following steps describe the workflow for creating CI/CD compliance reports.

Step	Description
Step 1. Create an Asset Group.	Create an Asset Group
Step 2. Create an Assessment Profile.	Create an Assessment Profile
Step 3. View reports.	View and access reports

Create an Asset Group

Create an asset group to define a logical collection of your CI/CD assets (such as specific repositories or pipelines within a provider like GitHub). This step scopes your security assessments, ensuring that subsequent compliance checks and scans performed by an assessment profile are applied to the relevant resources.

1. Navigate to Inventory → Groups → + Add Group.
2. On the Create New Assets Group screen:
 - a. Provide a group name (required) and description.
 - b. From the Filter menu in the Assets table, select Provider → [Type of provider].

NOTE:

The CI/CD module supports GitHub and GitLab provider types.

- c. Select Create Dynamic Group, or select assets from the list that is displayed, and click Create Static Group.

NOTE:

For more information about Asset Groups, refer to Asset Groups.

Create an Assessment Profile

Create an assessment profile, which configures the specific security standards and initiates the scans against the assets defined in your asset group.

1. Navigate to Posture Management → Compliance → Assessment Profiles → Create New Assessment.
2. On the General step of the wizard.
 - a. Provide a profile name (required) and description (optional), and select Generate a scheduled report.
 - b. Specify the email recipients for the report.
 - c. Set the Evaluation frequency (required).
 - d. Click Next.
3. On the Standards and Asset Group step of the wizard.
 1. Select a standard.
 - NOTE:**

CIS GitLab Benchmark, CIS GitHub Benchmark, and the OWASP Top 10 CI/CD Risks standards are supported.
 2. Select your asset group from the list and click Next.
4. Review the details on the Summary step of the wizard and click Create.

NOTE:

For more information about assessment profiles, refer to [Use an assessment profile to run compliance checks on your assets](#).

View and access reports

The email recipients defined in the assessment profile will receive the compliance report.

To view the compliance scan results:

- Navigate to Posture Management → Compliance → Reports.

For more information about compliance assessment reports, refer to [View and manage compliance assessments and reports](#).

4.8 | Urgency

Prioritize issues by Urgency, a context-aware metric to help you focus remediation efforts on the issues that pose the greatest real-world risk in your code. Unlike **Severity**, which is a static measure of an issue's technical risk, Urgency dynamically evaluates risk based on deployment context.

The Urgency enrichment highlights risks based on specific, high-impact factors:

- **Deployment status:** Identifies issues actively deployed in production
- **Runtime exposure:** Indicates deployed assets that are in use, exposed to the internet, or can be exploited to leverage privileged capabilities in case of an attack
- **Data impact:** Flags deployed assets that can access sensitive data
- **Business criticality:** Prioritizes issues impacting your most critical business assets

How Urgency is calculated

Urgency is calculated as a dynamic risk score derived from **Probability** (likelihood of exploitation) and **Impact** (potential damage). It leverages multiple sources of data across your code-to-cloud environment:

- **Applications:** Data from applications created in Cortex Cloud
- **Runtime:** Data from deployed and running assets
- **Code:** Findings from native Cortex Cloud Application Security scanners and ingested third party sources
- **Risk metadata:** External threat context aggregated from third-party sources
- **Asset metadata:** Contextual information associated with the asset where the finding was detected, including any enrichments. For example, a vulnerability is a finding on a software package, and its enrichment could include related repository data

For a detailed breakdown of the parameters defined by each scanner type, see [Urgency metrics](#)

Urgency levels

- Top Urgent: Requires immediate mitigation
- Urgent: Mitigate as soon as possible
- Not Urgent: Requires attention, but can be addressed within your organization SLA (Service Level Agreement)
- Not Applicable: Includes two options:
 - For periodic scans: Urgency has not yet been calculated and will be calculated in the next periodic scan
 - Not calculated for PR (Pull Request) or CI (Continuous Integration) scans

How Urgency appears in the tenant

Persona and permissions: This feature is designed for AppSec Admins, DevOps or Developer persona, and requires the View role under RBAC.

In Cortex Cloud Application Security, Urgency is applied to issues detected from CVE Vulnerabilities, Secrets, IaC Misconfigurations, and Code Weaknesses periodic scanners. It is displayed in the tenant as follows:

- **Issue Inventory:** An Urgency column displays the urgency score assigned to the issue. You can filter by this column to focus on the most critical issues
- **Issue side-card:** The Overview tab of an issue side panel displays a Urgency Details section, listing all the code-to-cloud data sources used to calculate the Urgency score. A code-to-cloud graph displays where the Urgency context was detected across your software development lifecycle

For more information on Urgency by type of scanner issue, refer to:

- CVE vulnerabilities: Detailed vulnerability issue information
- Secrets: Detailed Secrets issue information
- IaC misconfiguration: Detailed IaC misconfiguration issue information
- Code weaknesses: Detailed code weakness issue information

4.8.1 | Urgency metrics

The following table outlines Urgency metrics by scanner, detailing each metric's description, values, and evidence.

Scanner type	Metric	Description	Values	Evidence
Vulnerabilities	Application Business Criticality	The highest criticality level among all applications linked to an issue's affected assets. If no application is attached, the value is None . Link the affected asset to its relevant application to ensure the urgency is calculated correctly	Critical, High, Medium, Low, Info, None + Name	The Application Name + ID (only one)
	Access Sensitive Data	At least one deployed asset affected by this issue has access to sensitive data	True, False + Finding ID	Finding ID (only one)
	Leverage Privileged Capabilities	At least one deployed asset affected by this issue has the ability to leverage privileged capabilities. In the next version, we'll provide permission-level details	True, False + Finding ID	Finding ID (only one)
	Used in Image	Indicates whether the vulnerable package present in the code is also included in the built image	True, False	—
	Is Deployed	At least one deployed asset is affected by this issue	True, False	—
	Internet Exposed	At least one affected deployed asset is accessible from the internet, increasing the likelihood of exploitation	True, False	—
	Loaded into Memory	The vulnerable package is actively loaded into memory in at least one deployed asset	True, False, Unknown	—

Scanner type	Metric	Description	Values	Evidence
OpenVAS	Runtime Agent Protection	Percentage of affected deployed assets with runtime protection enabled. Only active agents are counted. Effective if $\geq 80\%$ of deployed assets are protected	0–100%	—
	EPSS Score	Estimated probability that this CVE will be exploited within the next 30 days	0–100%	—
	CISA KEV	Indicates whether this CVE is listed in CISA's Known Exploited Vulnerabilities catalog	True, False	—
	CVSS Score	Industry-standard severity score for vulnerabilities (Common Vulnerability Scoring System)	0–10	—
	Exploit Maturity	Level of confidence in the existence of a known exploit	POC, Active, None	—
	Exploit Availability	Indicates whether an exploit is available to attackers	Public, Private	—
	Package Operational Risk	Risk level based on low maintenance, limited popularity, or outdated support	High, Medium, Low	—
	Fixable	Indicates whether a known fix is available for this CVE	True, False	—

Scanner type	Metric	Description	Values	Evidence
Secrets	Application Business Criticality	The highest criticality level among all applications linked to an issue's affected assets. If no application is attached, the value is None . Link the affected asset to its relevant application to ensure urgency is calculated correctly	Critical, High, Medium, Low, Info, None + Name	The Application Name + ID (only one)
	Access Sensitive Data	Indicates whether the secret provides access to sensitive data	True, False + Finding ID	Finding ID (only one)
	Leverage Privileged Capabilities	Indicates whether the secret can be used to perform privileged operations	True, False + Finding ID	Finding ID (only one)
	Visibility	Indicates whether the code repository where the secret was found is public	Private, Public	—
	Validation	Indicates whether the exposed secret is valid and whether it has high privileges	Privileged, Valid, Invalid, Unavailable	—
	Found in History	Indicates whether the secret was found in the version history of the repository	True, False	—

Scanner type	Metric	Description	Values	Evidence
IaC Misconfigurations	Application Business Criticality	The highest criticality level among all applications linked to an issue's affected assets. If no application is attached, the value is None . Link the affected asset to its relevant application to ensure urgency is calculated correctly	Critical, High, Medium, Low, Info, None + Name	The Application Name + ID (only one)
	Access Sensitive Data	At least one deployed asset affected by this issue has access to sensitive data	True, False + Finding ID	Finding ID (only one)
	Leverage Privileged Capabilities	At least one deployed asset affected by this issue has the ability to leverage privileged capabilities. In the next version, we'll provide permission-level details	True, False + Finding ID	Finding ID (only one)
	# Affected Assets	The number of deployed cloud assets affected by this issue	Number	—
	Severity	The issue's inherent severity (static rating, not including any user-applied override)	Critical, High, Medium, Low, Info, None + Name	—
	Internet Exposed	At least one affected deployed asset is accessible from the internet, increasing the likelihood of exploitation	True, False	—
	Is Deployed	At least one deployed asset is affected by this issue	True, False	—

Scanner type	Metric	Description	Values	Evidence
Code Weaknesses	Application Business Criticality	The highest criticality level among all applications linked to an issue's affected assets. If no application is attached, the value is None . Link the affected asset to its relevant application to ensure urgency is calculated correctly	Critical, High, Medium, Low, Info, None + Name	The Application Name + ID (only one)
	Access Sensitive Data	At least one deployed asset affected by this issue has access to sensitive data	True, False + Finding ID	Finding ID (only one)
	Leverage Privileged Capabilities	At least one deployed asset affected by this issue has the ability to leverage privileged capabilities. In the next version, we'll provide permission-level details	True, False + Finding ID	Finding ID (only one)
	Is Deployed	At least one deployed asset is affected by this issue	True, False	Asset ID (only one)
	Severity	The issue's inherent severity (static rating, not including any user-applied override)	Critical, High, Medium, Low, Info, None	—
	Internet Exposed	At least one affected deployed asset is accessible from the internet, increasing the likelihood of exploitation	True, False	—
	Runtime Agent Protection	Percentage of affected deployed assets with runtime protection enabled. Only active agents are counted. Effective if $\geq 80\%$ of deployed assets are protected	0–100%	—

Scanner type	Metric	Description	Values	Evidence
	CWE Top 25	Indicates whether the issue maps to one of the CWE Top 25 Most Dangerous Software Weaknesses	True, False	—
	OWASP Top 10	Indicates whether the issue maps to one of the OWASP Top 10 Web Application Security Risks	True, False	—

4.9 | Backlog baseline

Abstract

Backlog represents pre-existing code issues discovered by a scanner's first run or by new rules.

In managing application security, it's crucial to distinguish between backlog issues and new issues. Backlog issues represent the security technical debt- vulnerabilities that existed in a code repository or branch before a security scanner's initial run, or were uncovered by new scanner capabilities or rules. This collection defines a historical **backlog point** for a codebase. New issues are fresh vulnerabilities introduced into the codebase, typically through recent changes such as pull requests. This classification allows you to implement tailored security policies and prioritize remediation efforts more effectively.

4.9.1 | Backlog use cases

- **Gain granular visibility into security technical debt:** AppSec Admins need a clear, detailed picture of their security technical debt. This means being able to see the number of issues classified as **backlog** and **new** per repository and branch within an Cortex Cloud Application Security dashboard. This granular view allows managers to understand the full scope of their technical debt and identify areas where new vulnerabilities are still being introduced (bleeding)
- **Monitor and optimize security program performance:** To effectively manage security initiatives, AppSec Admins require a way to track the trend of **backlog** and **new** issues over time, both in total and broken down by each scanner (IaC, Secret, Vulnerability, SAST). Displaying this trend data in a dashboard widget helps you understand the pace of issue resolution across teams and pinpoint areas that may require escalation or additional resources
- **Streamline prioritization and remediation by scanner:** Cortex Cloud Application Security practitioners, development managers, and business owners need to efficiently prioritize and act on security findings. This is achieved by seeing, for each scanner, which issues are classified as **backlog** and **new**. This scanner-specific view allows for targeted remediation efforts, ensuring teams can focus on the most relevant and impactful issues based on their origin and status. This also includes the ability to integrate scanner-specific issue counts into broader dashboards like **DevChamp** and **DevSecOps** for cross-functional awareness
- **Implement differentiated security policies:** To ensure the system accurately reflects an organization's specific context, it must allow for customizable issue classification. This includes defining a historical cutoff date for when issues in existing repositories are considered **backlog** and **new**, and setting parameters such as treating new vulnerabilities discovered on existing Software Bill of Materials (SBOMs) as **new** issues. This flexibility ensures the backlog/new distinction is meaningful and actionable for all stakeholders.

4.9.2 | Issue/Finding classification by scanner

This table details how security issues and findings are classified as either Backlog or New based on their originating scanner and specific detection scenarios

Scanner	Backlog	New Issue	Comment
IaC	The first time an IaC detection rule ran against the code repository	Issues added through pull requests that are created by a detection rule which previously ran against this repository	If a new AppSec rule runs against the code repository, the detected issue is considered a Backlog issue

Scanner	Backlog	New Issue	Comment
Secrets	The first time a secret was detected on the code repository with a specific signature (out-of-the-box or customer-created)	A secret that was added in a pull request	If a new signature is added/changed in the secret signature engine (by the vendor or by the user), its first run will be considered a Backlog issue
SCA Vulnerabilities	The first time the SCA scanner created an SBOM of the code repository and identified vulnerabilities	A vulnerability found in a pull request on a new or updated package	<ul style="list-style-type: none"> • If there is a new vulnerability on an existing package version, it is considered a Backlog issue • If you set the global parameter issues on existing SBOM are considered new, it will be considered a new issue

Scanner	Backlog	New Issue	Comment
SAST	The first time the SAST scanner sends results on this code repository and file	A SAST finding that was found on a pull request	<p>This classification also applies if you import a SARIF file for a repository.</p> <p>NOTE:</p> <ul style="list-style-type: none"> • In some cases/vendors, this is not accurate as findings are deleted every time new findings are uploaded. In such cases, the feature may not be accurate or supported • For SAST, the vendor does not support policy in pull requests

NOTE:

Scanner updates and new detections: When a security scanner is updated to support new languages, detection rules, or capabilities, any issues discovered by these new features for existing code are classified as part of the backlog.

4.9.3 | Using Backlog

You can leverage the Backlog and New issue classifications across the platform as follows.

- **Role-Based Access Control (RBAC) / Standard-Based Access Control (SBAC):** Access and permissions will be managed systematically:
 - By default, only AppSec Admins have permission to configure the issues on existing SBOM are considered new setting
 - Permissions for all other capabilities, such as viewing issues or applying policies, are defined by the existing RBAC/SBAC policies and the user's specific issue management capabilities
- **Policies/Scope:** The system supports Backlog and New attributes for policies, allowing for differentiated enforcement. Refer to Create Cortex Cloud Application Security policies for more information
- **Multi-Branch Support:** The Backlog/New classification is consistent across development workflows:
 - The Backlog/New classification is maintained independently for every branch
 - The system allows policies to be defined and applied for specific branches, enabling you to tailor security rules (for example, enforcing stricter policies for New critical issues on main branches, or allowing Backlog issues on development branches) based on their classification
- You can filter the **Cortex Cloud Application Security dashboard** to display information according to the Backlog/New classification
- **Issues and Findings:** The Backlog/New classification is standardized across data for both findings and issues under the Backlog Status field, which is found under the Overview tab of both findings and issues side cards. For example, refer to Secrets issues
- The **API** provides comprehensive access to classified issue data:
 - You can retrieve all backlog issues, or filter them per scanner
 - You can retrieve all new issues, or filter them per scanner
 - You can retrieve new issues within the backlog, either all of them or filtered per scanner
 - You can disable the issues on existing SBOM are considered new flag programmatically, provided you have the required permissions

4.10 | Service Lead Agreements (SLA)

Abstract

Application Security SLA defines deadlines for fixing security issues based on severity, ensuring timely remediation and improving team performance.

Cortex Cloud Application Security SLA defines remediation timeframes for security issues based on their severity, ensuring timely fixes and improving team performance. It sets clear expectations for how quickly threats must be addressed and provides a measurable metric for tracking responsiveness, identifying bottlenecks, and strengthening overall security posture.

Cortex Cloud Application Security SLAs apply to issues detected during periodic code scans. Each severity level has an assigned remediation timeframe to support consistent issue management.

The default target remediation timeframes are:

- Critical: 7 days
- High: 14 days
- Medium: 30 days
- Low: 90 days

You can modify these values as required.

SLA status and monitoring

SLA status provides immediate risk context for prioritization of issues. The system automatically calculates and updates each issue's SLA status based on periodic scans and the configured timeframes.

There are three SLA status values:

- On Track: The issue is within its assigned remediation timeframe
- Approaching: The issue's SLA will be breached in a configurable number of days (the Approaching threshold). This status alerts you before an issue becomes overdue
- Overdue: The issue has breached its SLA

Roles and responsibilities

- **AppSec practitioners:**
 - Define and configure the SLA targets for each severity level
 - Track the SLA status for all Cortex Cloud Application Security issues across the organization
 - Generate reports and dashboards to measure team performance and identify trends
- **Developers / DevSecOps:**
 - Be aware of and adhere to the SLA commitments for all assigned issues
 - Actively monitor and prioritize issues that are nearing or have exceeded their SLA

Use cases and features

- For AppSec practitioners:
 - **Overdue dashboard:** Get a clear overview of all Approaching and Overdue issues. This allows you to quickly identify problematic areas and contact the relevant teams
 - **Centralized SLA tracking:** The SLA status for each issue is displayed directly in the Cortex Cloud Application Security issues tables
 - **Categorized overdue issues:** Filter overdue issues by domain (such as SAST, SCA, IaC, Secrets) to understand which areas require additional attention
 - **Detailed issue information:** A side panel on each issue provides a comprehensive view of its SLA details, including the configured time and how many days have passed, helping you understand its priority
- For developers :
 - **SLA visibility:** See the specific SLA you need to follow for each issue, ensuring you are always aware of your commitments
 - **Overdue issues:** Easily identify and filter issues that are past their SLA, so you can prioritize and fix them immediately
 - **Upcoming overdue issues:** Anticipate and prepare for issues that will become overdue in a configurable number of days, allowing you to take preventative action

4.10.1 | Configure and monitor Cortex Cloud Application Security SLAs

These procedures defining remediation timeframes and the methods available for monitoring issue compliance against the defined SLAs.

Configure SLA Remediation Targets

Define remediation timeframes to track issue compliance with the configured SLAs. The system automatically calculates and updates each issue's SLA status during periodic scans based on these timeframes.

1. Navigate to Settings → Configurations → Application Configuration (under Application Security).
2. Define the target SLA for each severity level: assigning a value in days → Save.

Default values:

- Critical: 7 days
- High: 14 days
- Medium: 30 days
- Low: 90 days

3. Set the approaching SLA threshold: Specify the number of days → Save.

This threshold enables proactive remediation and minimizes the risk of issues becoming Overdue.

Monitor SLA status

SLA status provides immediate risk context for prioritization. Status tracking is integrated across the Cortex Cloud Application Security Command Center dashboard, the Issues tables, and their side panels.

- **Cortex Cloud Application Security Dashboard:**

Displays a widget showing the number of Critical and High severity issues that are Overdue or Approaching SLA. The widget breaks down SLA status by scanner (for example, Secrets or IaC). Selecting a scanner opens the relevant issues page, filtered by scanner, severity, and SLA status.

You can access the Cortex Cloud Application Security dashboard from the the Application Security dashboard.

- **Issues table:**

The SLA status is integrated directly into the issues table, to provide context and help you track each issue.

To view SLA under Issues tables, under **Modules** select Application Security → [type of issue such as Secrets. If SLA is not displayed by default, select it from the Table Settings Menu.

SLA values:

- On Track: The issue is within its assigned remediation timeframe
- Approaching: The issue's SLA will be breached in a configurable number of days (the Approaching threshold). This status alerts you before an issue becomes overdue
- Overdue: The issue has breached its SLA

Hovering over an issue's SLA status will show a tooltip with additional details.

- **SLA in an issue side panel:**

Clicking on any individual issue opens a side panel. The Overview tab displays the current SLA status of the issue (such as Overdue). Hovering over this status provides additional details, including the issue severity, the total time allotted for remediation, and a countdown of the time remaining until the SLA is breached or since it was breached.

4.11 | Applications

Abstract

Build and manage applications as holistic entities. Gain centralized visibility across the SDLC to monitor assets and remediate threats based on business risk.

Applications are a single, holistic entity that encompasses their entire lifecycle and all its components, from custom code to open-source libraries and infrastructure configurations. This dynamic, logical entity allows for focused monitoring and protection of high-priority assets throughout your software delivery life-cycle (SDLC). Cortex Cloud provides you with the tools to build, manage, and gain visibility into your applications.

- **Build applications to your needs.** You can create business applications by selecting and associating components, starting with either code or cloud assets
- **Manage application assets:** The business application assets inventory gives you a centralized view of all applications and their interconnected assets throughout your SDLC
- **Application management and visibility:** Cortex Cloud provides tools for managing the security issues detected in your applications, allowing you to prioritize, analyze, and mitigate threats based on business criticality

By centralizing these functions, Cortex Cloud helps you to identify, prioritize, and remediate issues that could impact your most critical business systems.

4.11.1 | Defining Business Applications

Abstract

Define Business Applications automatically using tag-based criteria or manually with the Application Builder to map assets and prioritize app risk.

Business Applications allow you to define, group, and maintain assets that constitute a logical application with a unified business context. This enables a precise Code to Cloud security posture by correlating risks across the entire development lifecycle.

You can build applications using one of these methods:

- **Application Criteria:** Automatically create and maintain multiple applications in bulk by defining Application Criteria. Criteria allow you to dynamically set rules that group assets into applications based on existing cloud tags or code-based attributes. This ensures consistent grouping and scalable application visibility while reducing manual effort
- **Application Builder:** Provide a starting point from either code or cloud, and Cortex Cloud Application Security automatically maps related assets across the application lifecycle

4.11.1.1 | Defining business applications by Criteria

Abstract

The Criteria process uses tag-based criteria as the single source of truth to automatically define application assets and enable Code-to-Cloud risk correlation in real-time.

Defining business applications by Criteria allows you to automatically create and maintain application boundaries using organizational metadata from integrated sources. You can define criteria based on **Cloud tags** (for example AWS tags) or code-based **Version Control System (VCS) entities**, such as organizations, projects, and repositories.

These criteria define the authoritative definitive logic for application grouping, enabling Cortex Cloud to correlate security risks across the entire Code to Cloud lifecycle—from source code through to CI/CD pipelines, deployments, and runtime environments.

By automatically linking assets and enriching applications with business context (such as criticality, owner, and business unit), criteria-based applications through Criteria eliminate manual boundary management, and ensure consistent, real-time application visibility as your environment evolves.

- **Automated application mapping:** Define Criteria based on cloud tags or code-based VCS entities to automatically create and scale applications that align with your code structure and organizational patterns
- **Automatic asset enrichment:** Generate applications enriched with code, build, deploy, and runtime assets using the Cortex Cloud relationship algorithm
- **Reduced manual effort:** Eliminate time-consuming, manual mapping of assets to their applications
- **Real-time accuracy:** Update application maps automatically in response to changes in underlying infrastructure and asset tagging or code repositories
- **Holistic security posture:** Gain a complete application-centric view of your security posture across your SDLC

4.11.1.2 | Define applications by VCS criteria

Use VCS Criteria to automatically generate and maintain Business Applications based on your code hierarchy. Unlike manual creation, this method creates a dynamic rule set: as developers create new repositories that match your criteria, they are automatically recognized and onboarded as Business Applications without manual intervention.

PREREQUISITE:

- **Data source:** Your Version Control System (e.g., GitHub, GitLab) must already be onboarded as a Data Source
- **Permissions:** You must have View/Edit permissions for Access Management
- **SBAC:** You can only create applications from VCS entities (Organizations, Projects, or Repositories) that are already included in your SBAC Asset Groups

1. Under Modules, select Application Security → Business Applications (under Application Management)
→ Create Applications → New Criteria.

2. On the General step.

- a. Select Code as the source type.

NOTE:

This workflow allows you to unify assets across disparate providers (such as grouping a GitHub repository and a Bitbucket repository into one application) if they share naming conventions

- b. Provide a Criteria name (required) and description.
- c. Click Next.

3. On the Define Criteria step.

Define Grouping logic: Determine how Cortex Cloud constructs the boundaries of your applications. These settings control whether an application is defined as a single repository or a broader organization, and how the system handles assets with identical names across your environment. Connected runtime

and deployment assets are automatically linked to these boundaries to provide a complete view of the application lineage

- a. **Group by** (required): Select the VCS entity level that represents a distinct application in your architecture (such as Organization, Project, or Repository).

This setting defines the application perimeter. For example, selecting Repository creates a separate application for every repository found, whereas selecting Organization aggregates all assets within an organization into a single application.

- b. Merge organizations/projects/repositories with identical names (optional): Group entities with identical names within the selected provider.
- c. Unify applications across providers (optional): Group entities with identical names across all selected providers.

d. Click Next.

4. On the Scope step.

Apply filters to strictly define which VCS entities are processed. Select any combination of provider entities (one or multiple) to refine the application's scope, for instance, by organization alone or by organization + project + repository for precision. This ensures the scope aligns precisely with the desired segment of your VCS structure.

The scope remains dynamic: newly matching assets join automatically as your environment evolves, while those that stop matching drop out. This maintains an up-to-date inventory tied to the chosen VCS entities.

- a. Select a VCS provider to evaluate.
- b. Configure rules to limit scope. Set specific conditions, such as: Organization Name Contains Production.

Only assets matching these rules will trigger the creation of an application.

c. Click Next.

5. On the Metadata step.

Configure rules to automatically assign ownership and risk levels to the generated applications.

- a. Business Owner: Assign application ownership by syncing with your VCS provider (such as GitHub or GitLab)
- b. Configure Business Criticality:
 - i. Select a default severity level.
 - ii. **Severity override:** It is recommended to enable **Internet exposure override** to automatically elevate the severity of internet-exposed assets to Critical, ensuring accurate risk prioritization.

c. Click Done.

Cortex Cloud will begin processing your criteria. Navigate to the Business Applications list to verify that your new applications have been generated and populated with assets.

4.11.1.3 | Manage application criteria

After defining your application criteria, you can monitor their status and manage their lifecycle from a centralized view.

[View application Criteria](#)

View and manage the rules that govern your application definitions.

To view Application Criteria, under Modules, select Application Security → Application Criteria (under Application Management).

The table below lists all configured Application Criteria, detailing the properties that define how Cortex Cloud groups assets:

- Name: The user-defined name for this criteria set
- Tags: The specific tag keys selected from the cloud provider
- Creation Method: Indicates the origin of the criteria definition
- Created By: The entity that created the criteria
- Assets: The count of logical applications and the total number of associated assets grouped by this criteria
- Last Updated: The timestamp of the last modification to the criteria set

[Manage application Criteria](#)

- You can delete application Criteria: On the Application Criteria screen, right-click on a criteria in the table → Delete.
- To delete **all** application Criteria, select all Criteria → Delete.

Deleting the criteria will remove all applications created by it. This process is irreversible.

4.11.1.4 | Define applications by cloud tag-based criteria

Creating applications through cloud entities (Accounts, Subscriptions, or Resource Groups) is the foundational step for runtime-driven risk correlation. By defining these grouping rules, you enable Cortex Cloud to unify infrastructure assets within a specific Cloud Service Provider (CSP) and automatically link runtime environments to their associated deployment pipelines and originating source code. This environmental context is required for precise exploitability-aware prioritization and comprehensive Cloud to Code visibility.

Grouping is limited to assets within a single cloud provider, and assets with the same tag key and value across accounts or projects in the same provider are included in the same application. Cross-provider grouping is not supported.

PREREQUISITE:

Permissions: You must have View/Edit permissions for Access Management, or a role that includes these permissions.

1. Under Modules, select Application Security → Business Applications (under Application Management) → Create Applications → New Criteria.

2. On the General step of the application Criteria wizard.

- Select Cloud.

NOTE:

You can only create applications based on the entities from onboarded Cloud accounts listed in the Cloud card.

- Provide a Criteria name (required) and description.

- Click Next.

3. On the Define Criteria step of the wizard.

- Select a cloud provider to define where assets will be discovered for the application.

The system automatically retrieves all available tag keys from that provider.

- Select the organizational tags you will use for automatic asset grouping:

- Selection limit:** Select between one and five tags
- Grouping logic:** When multiple tags are selected, an **AND** condition is applied. Only assets that contain **all** the chosen tag keys will be included in the resulting application

NOTE:

Kubernetes (K8S) labels: K8S labels are supported as tags for asset grouping only when they originate from a supported cloud provider; AWS, GCP, Azure, or OCI.

- Click Next.

4. On the Metadata step.

- Map application metadata:** Map existing infrastructure tags to these application metadata fields.

This ensures the automatically created application definition inherits the required security and business context from its grouped assets and reduces the need for manual updates after the application is created.

- Application Name:** Specifies which tag key should be used to derive the application name (for example, if you specify `app-name` as the source tag, applications will be named based on values found in the `app-name` tag)
- Business Criticality:** Determines which tag key contains business criticality information (such as **criticality**). How it works: Extracts criticality levels (Critical, High, Medium, Low) from the specified tag and assigns the highest criticality level found across assets. If not specified or no value found, defaults to Medium
- Business Owner:** Map to a tag key which contain business owner information, allowing you to define the entity responsible for the application (such as `owner`)
- Business Unit:** Map to a tag key containing business unit information by defining the relevant department within the organization that uses or owns the application (such as `org`)

- b. Select Done.

The configured values are assigned to their corresponding application fields, creating the criteria set and ensuring that mapped metadata is applied to all matching assets.

- c. Verification: On the Business Applications page, confirm the success notification is displayed, and that the newly created applications, based on the defined criteria, are displayed in the list. You may need to wait some time for the applications to populate, especially for large applications gathering substantial data.

4.11.1.5 | How to manually build an application

Abstract

Manually build an application by adding assets, starting from either the code or run side.

You can build your application by adding assets, starting from either the code or run side. This process covers your entire code-to-cloud journey: Code, Build, Deploy, Run. Your application is then automatically built from the assets you select on your chosen starting side, and other assets are added automatically based on their connections.

PREREQUISITE:

Before you begin, ensure you have connected the necessary data sources. Refer to [How to onboard data sources](#) for more information.

1. Under Modules, select Application Security → Business Applications (under Application Management) → New Application → New Application.
2. In the Add a New Application dialog box:
 - a. Provide the required details:
 - Application name (required): A user-provided name
 - Category (required): Default - Business Application
 - Description: A description of the application. Provides context for users interacting with the application
 - Business Criticality (required): The level of importance of the application to your organization. This helps prioritize resources and attention based on the application's impact to your business objectives
 - Business owner: The individual or team responsible for the application from a business perspective
 - b. Click Create.
3. On the Applications page: Add assets to your application, starting with either code or cloud assets:

- Add code assets from the Code pane:
 1. Select a version control system (VCS) from the list that is displayed.
 2. Choose one or more of the following from their respective dropdown lists (multiple selections allowed): a specific VCS instance, an organization, or a repository.

NOTE:

These filters are represented by icons displayed on the Code pane after selecting a VCS.

3. Click Done (on the Code pane).

After connecting your VCS, Prisma: Cortex automatically identifies and associates all build-time (such as source code, build scripts, Dockerfiles, CI tools), deploy (such as Kubernetes manifests, Helm charts, Terraform scripts), and runtime assets (such as running containers, virtual machines, cloud instances).

- Add cloud assets from the Run pane:

1. Select an option:
 - Click Add Assets by Provider to select a cloud service provider
 - Click Add Assets by Tag or optionally select Kubernetes Namespace, Kubernetes Cluster, VPC, Organization or Resource tag to automatically populate the application with runtime assets associated with those entities. These filters are represented by an icon displayed under Run (after selecting a cloud provider)
2. Select an instance of your connected provider from the list that is displayed.
3. Click Done (on the Run pane).

After connecting your cloud service provider, Cortex Cloud automatically identifies, associates and displays all Code (VCS repositories), build-time (such as source code, build scripts, Dockerfiles, CI/CD pipelines) and deploy assets (such as Kubernetes manifests, Helm charts, Terraform scripts).

4. Click Finish to create the application.

The application is displayed on both the All Applications and its dedicated asset page (business).

NOTE:

To edit application assets, click the Clear All icon before clicking Finish. This clears all application data, allowing you to restart the application building process from the beginning.

4.11.2 | Application management and visibility

You can view and manage your applications from the following interfaces:

- The **Business Applications** asset inventory provides a focused view that allows you to analyze and manage business assets, including seeing all of your interconnected assets, tracing their path to production, and managing issues directly from the asset's side card: Navigate to Inventory → All Assets → Business Applications (under Application)
- Dedicated Application tabs in the side panels of Cortex Cloud Application Security asset categories, such as IaC Resources, Repositories and Software Packages, which list assets associated with applications. For example, see Applications

4.11.3 | Business application assets

The Business Application asset inventory provides visibility into all business applications and their interconnected assets generated throughout your software development lifecycle (SDLC), serving as a centralized repository for business application inventory management. Additionally, the interface details the risks detected in your business applications, allowing you to prioritize, manage, and mitigate potential threats based on business criticality.

To access the Business Application asset inventory, under Inventory, select All Assets → Business Application.

The Business Application asset inventory includes a dashboard with a widget of all issues detected in the application by severity level and a table including a list of applications.

Controls: You can filter the table to narrow results or export the table data from the Download icon.

The following fields are exposed in the application inventory table. To add additional table properties, select Menu settings → [property].

Field	Description
Name	The application name
Business Owner	The individual or team responsible for the application from a business perspective, as provided when creating the application
Criticality	The importance of the application to the business as defined when creating the application
Assets	The amount of assets associated with the application
Creation Method	Whether the application was created using criteria (Auto) or manually

Field	Description
Risk	Represents the overall assessed risk level for the application
Criteria Name	The configured criteria name
Last Updated	Timestamp showing the most recent application update

4.11.3.1 | Business application expanded asset details

Click an application in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Topology tab (providing context on the application path to production), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

Overview

The Overview tab summarizes application highlights, metadata and properties.

- **Highlights:** Includes properties such as deployment status
- **Visibility timeline:** When the application was first and last detected
- Asset properties, including Asset Id, Asset Category, Asset Groups and associated with the application
- Application risks:
 - **Risk summary:** The amount of risks associated with the application assets grouped by category (cases, issues and findings) and their severity level. For more information about issues, refer to Cortex Cloud Application Security code scanners
 - Risk Score: A value representing the overall security risk of an application, based on various underlying metrics. This helps in assessing and prioritizing the application's security posture and potential vulnerabilities
- Coverage: Evaluate the application security coverage via its scanned asset percentage
- Business Criticality: As defined when creating the application. See How to manually build an application for more information
- Business Owners: The entity associated with the application
- **Criteria:** The criteria used to create the application
- Creation Method: Indicates if the application was created through a manual selection of assets or automatically (such as via automation or discovery)

Topology

The Topology tab visualizes your application's asset relationships across the entire software development lifecycle (SDLC). It maps interconnected assets including code repositories, pipelines, container images, and workloads, providing a comprehensive representation of the code-to-cloud journey. You can view the topology either as a visual representation or as an asset inventory by selecting the Graph or Inventory (default) tabs respectively.

NOTE:

The topology graph is available only when all application components (code, pipeline, build and deploy), are configured.

Topology graph

The graph displays the application path to production, organized into four key SDLC sections:

- **CODE:** Displays source code repositories and VCS organizations, allowing you to understand code organization and repository structure:
 - Providers: GitHub, GitLab, Azure Repos, Bitbucket
 - Key relationships: Organizations contain repositories; repositories are forked from others
- **BUILD:** Displays CI/CD pipelines, visualizing build processes and pipeline dependencies:
 - Providers: GitHub Actions, GitLab CI/CD, Jenkins, Azure Pipelines, CircleCI
 - Key relationships: Repositories trigger pipelines; pipelines build container images
- **Deploy:** Displays container registries and image repositories, allowing you to track image lineage and registry organization:
 - Providers: Docker Hub, Google Artifact Registry (GAR), Amazon ECR, Azure ACR
 - Key relationships: Registries contain image repositories; pipelines build specific container images
- **Run:** Displays runtime architecture, including compute, storage, networking, and identity assets, allowing you to understand runtime architecture and resource dependencies
 - Assets: Kubernetes clusters/workloads, virtual machines, serverless functions, storage buckets, load balancers, and IAM policies
 - Providers: AWS, GCP, Azure
 - Key relationships: Images run on instances, workloads use service accounts, functions access storage buckets

Navigating the graph

Use the following controls to manage the view and investigate assets:

- **Node actions:** Click any asset node to view basic details. Select View Details in the popup to open the asset side-car for comprehensive information without leaving the topology view
- **Search and highlight:** Search for specific assets by name to highlight matching nodes and navigate directly to them in the graph
- **Group nodes:** Toggle this to organize assets into logical clusters (such as Container Images), simplifying complex graphs. Click a group to expand it
- **Layers:** Apply filters to view assets based on specific criteria, such as public internet exposure, related cases, or associated runtime events

Filtering and layout options

Customize the display to focus on relevant information:

- **Section filtering:** Toggle visibility for specific SDLC sections (CODE, BUILD, DEPLOY, RUN) to isolate parts of the lifecycle
- **Provider filtering:** Filter assets by cloud or VCS provider (such as Show only AWS or GitHub assets)
- **Layout options:** Choose a visualization style:
 - Hierarchical: Top-to-bottom flow (Code → Build → Deploy → Run).
 - Force-Directed: Physics-based layout.
 - Circular: Circular arrangement.

Understanding relationships

Edges connecting nodes represent specific interactions or dependencies, including:

- **CONTAINS:** Hierarchical containment (such as Org → Repo)
- **TRIGGERS:** Activation (such as Repo → Pipeline)
- **BUILDS:** Creation (such as Pipeline → Image)
- **RUNS ON:** Runtime execution (such as Image → Container Instance)
- **USES/ACCESSES:** Resource usage or data access

Common workflows

- **Investigate critical vulnerabilities:** Identify a critical CVE, locate the affected repository in the graph, and trace relationships forward to see if vulnerable versions are currently deployed as running instances
- **Track Code to Cloud misconfigurations:** Identify IaC issues (code) and trace them to deployed cloud resources to ensure fixes are applied at the source to prevent future misconfigured deployments
- **Audit secret exposure:** Locate repositories with privileged secrets and trace them to the DEPLOY or RUN sections to see if those secrets are active in production environments
- **Understand application architecture:** Filter for the RUN section to identify runtime components, then trace back to source repositories to document deployment paths for compliance.

Topology inventory

The Inventory table displays all assets associated with the business application. Selecting an asset opens its side card directly without having to navigate away to the dedicated asset inventory.

- **Asset details:** Displays properties such as Name, Provider, Type, Region, and timestamps for First/Last Observed
- **Risk context:** Includes breakdowns of associated cases, critical issues, and vulnerability severity
- **Table controls:** Filter the table by property or adjust the table settings to add/remove columns
- **Export icon:** Download the inventory as a **.tsv** file. See Export business application data for more information

Vulnerabilities

The Vulnerabilities tab displays SCA vulnerability issues detected across the application assets. This tab includes a continuous funnel graph and a section detailing the riskiest repositories.

The graph displays the following vulnerability metrics, filtered by default for Critical and High severity:

- All: The total amount of vulnerabilities detected in the application and its assets
- Exploitable: The subset of total vulnerabilities that are exploitable
- Fixable: The subset of total vulnerabilities that have an available fix
- Deployed: The subset of vulnerabilities detected in deployed application assets

You can filter the graph to display any combination of severities (Critical, High, Medium, and Low). Selecting any stage of the funnel (such as Fixable) redirects you to the main Issues inventory, filtered to display vulnerabilities that match the criteria you selected (for example, issues that have available fixes).

A known limitation is that only up to 4,000 issues will be displayed in the Issues inventory when redirecting from the graph, even if the count in a particular stage (such as Deployed) is higher.

The Riskiest repositories section lists the repositories with the highest risk, based on the number and severity of known vulnerabilities detected in the application. It also displays risk metrics such as whether the repository is deployed.

This section displays the following details for each repository:

- VCS
- Repository location
- Branch
- Last commit date

Selecting a repository from the list redirects you to the main Issues inventory, filtered to display all vulnerability issues for that specific repository. It includes the total number and a breakdown of issues by severity level.

Selecting the branch link opens that repository's asset side-card directly, allowing you to view more details without navigating away.

Configurations

The Configurations tab displays IaC misconfiguration issues detected across the application assets. This tab includes a graph and a section detailing top IaC misconfiguration rules.

The graph displays the following IaC misconfiguration metrics, filtered by default for Critical and High severity:

- All: The total number of misconfigurations detected in the application and its assets
- Fixable: The total number of misconfigurations that have an available fix
- Deployed: The total number of misconfigurations detected in deployed application assets

You can filter the graph to display any combination of severities (Critical, High, Medium, and Low). Selecting any of these categories (such as Fixable) redirects you to the tenant's main Issues inventory. This page will be filtered to display all IaC Misconfiguration issues for this specific application that match the criteria you selected (for example, issues that have available fixes).

A known limitation is that only up to 4,000 issues will be displayed in the Issues inventory when redirecting from the graph, even if the count in a particular category (such as Deployed) is higher.

The Top IaC misconfiguration rules section helps you identify and focus on the most urgent issues by highlighting misconfigurations detected from a matching rule in both the source code and the deployed cloud environment. It includes the total number and a breakdown of issues by severity level.

Selecting one of these matching rule sets redirects you to the main Issues inventory, filtered to display all IaC misconfiguration issues detected by that specific IaC rule set.

Secrets

The Secrets tab displays exposed Secrets issues detected across the application assets. This tab includes a graph and a section detailing the Riskiest repositories.

The graph displays the following Secrets metrics, filtered by default for Critical and High severity:

- All: The total number of Secrets detected in the application and its assets
- Valid: The total number of detected Secrets that have been verified as active and functional
- Privileged: The total number of Secrets that are valid and provide high-level access

You can filter the graph to display any combination of severities (Critical, High, Medium, and Low). Selecting any of these categories (such as Valid) redirects you to the tenant's main Issues inventory. This page will be filtered to display all Secrets issues for this specific application that match the criteria you selected (for example, issues that are validated).

A known limitation is that only up to 4,000 issues will be displayed in the Issues inventory when redirecting from the graph, even if the count in a particular category (such as Valid) is higher.

The Riskiest repositories section identifies the repositories with the highest risk, based on the number and severity of known Secrets detected in its assets. It includes the total number and breakdown of issues by severity level.

- VCS
- Repository location
- Branch
- Last commit date

Selecting a repository from the list redirects you to the main Issues inventory, filtered to display all Secrets issues for that specific repository.

Selecting the branch link opens that repository's asset side-card directly, allowing you to view more details without navigating away.

Code Weaknesses

The Code Weaknesses tab displays SAST code weakness issues detected across the application assets. This tab includes a graph and a section detailing the Riskiest repositories.

The graph displays the following code weakness metrics, filtered by default for Critical and High severity:

- All: The total number of code weaknesses detected in the application and its assets
- Labels: The total number of code weaknesses that are categorized by specific labels
- Deployed: The total number of code weaknesses detected in deployed application assets

You can filter the graph to display any combination of severities (Critical, High, Medium, and Low). Selecting any of these categories (such as Deployed) redirects you to the main Issues inventory. This page will be filtered to display all Code Weakness issues for this specific application that match the criteria you selected.

A known limitation is that only up to 4,000 issues will be displayed in the Issues inventory when redirecting from the graph, even if the count in a particular category is higher.

The Riskiest repositories section identifies the repositories with the highest risk, based on the number, severity, and type of code weaknesses detected—including those deployed to production.

This section displays the total count and type of issues for each repository, along with:

- VCS
- Repository location
- Branch
- Last commit date

Selecting a repository item redirects you to the tenant's main Issues inventory, which is filtered to display all code weakness issues for that specific repository.

Selecting the branch link opens that repository's asset side card directly, allowing you to view more details without navigating away.

4.11.3.2 | Export business application data

You can export application security data for reporting, sharing metrics, or audit evidence. Cortex Cloud offers two export workflows: a portfolio-level overview or an application-level deep dive. Data is downloaded to your local host in a **.tsv** file format.

Export global portfolios

You can export the high-level inventory for all defined business applications. This is used for reporting on the organization's overall risk posture, business criticality, and security coverage.

1. Navigate to Inventory → All Assets → Business Applications.
2. Select the Export icon on the main table header.

A file containing high-level summary data of all your business applications is downloaded.

Export individual application asset data

You can export the granular technical details for a single Business Application. This allows for tracing the Code to Cloud lineage and verifying the security status of every asset within a specific service.

1. From the Business Application inventory, click on an application name to open the Application side card.
2. Select the Topology tab.
3. Ensure the view is set to Inventory.
4. Select the Export icon within the Topology section.

A file containing data of all the assets associated with the business application is downloaded.

4.11.4 | Scope user access to applications (Application SBAC)

Scope user access to applications to ensure users only have permission, visibility, and actions within the applications explicitly assigned to them. This enforces clear security boundaries and provides consistent, application-level control across all application-related assets and issues, minimizing a user's broad or unnecessary access by enforcing per-user application-level control and ensuring users can only access what's relevant to them.

Application SBAC defines security boundaries and policies around the application entity itself. It provides granular, application-aware control, transitioning from infrastructure-wide permissions to application-specific enforcement.

Key features

- **Granular access control (Implicit Deny Model):** Enforces explicit user access to specific applications and their associated assets—such as repositories, packages, and vulnerabilities. Access to any application or asset not explicitly listed is automatically denied
- **Contextual data filtering:** Use the Business Application Names as a universal filter to scope data views (such as dashboards) to a selected application

Application-based scope across the platform

- ASPM Command Center: Limits the interactive security workflow graph in the ASPM Command Center to only the applications the user is authorized for
- Dashboards: Application scope automatically narrows platform-wide data into application-specific insights
- Coverage: Evaluate the security maturity of your application by identifying connected data sources and their coverage status of the application's assets, assessing the scanner coverage status of onboarded assets, and understanding which scanners (such as SCA, Secrets, IaC) that are actively analyzing the application's codebase and build

Application SBAC setup and workflow

1. Platform enablement: Enable SBAC at the tenant level.
2. Create or edit an Asset Group to include application assets.
3. Scope user access to an application.
 - a. Assign application-based SBAC to a User Group.
 - b. Add users to the User Group.
4. **Resulting visibility:** Users see only the applications and related assets they are authorized to manage, based on the applied application scope.

Manage user access

Configure user scopes in Cortex Cloud by navigating to Settings → Configurations → Access Management. You must possess the necessary View/Edit RBAC permissions for Access Management. These permissions are granted by default to the Account Admin and Instance Administrator roles.

4.11.4.1 | Enable SBAC in the Cortex Cloud tenant

Before configuring Application scope, SBAC must be enabled at the tenant level.

PREREQUISITE:

RBAC permissions: To configure user scopes you must have Administrator or View/Edit RBAC permissions for Access Management (under Configurations).

- Navigate to Settings → Configurations → General → Server Settings → Enable Scope Based Access Control.

NOTE:

Exclusions (roles not governed by SBAC): Certain roles cannot have SBAC applied. For these roles, access and permissions are managed through Role-Based Access Control (RBAC). You must manually ensure that these roles have all necessary base permissions (for example **Edit/View permissions to assets**), because SBAC is bypassed and does not impose its usual restrictions. As a result, functional access for these roles is determined solely by their RBAC configuration.

4.11.4.2 | Create an application-based Asset Group

Create an application-based Asset Group if no appropriate group exists or if the application's permissions must be isolated from existing groups.

1. Navigate to Inventory → Groups → + Add Group.
2. On the Create New Assets Group page.
 - a. Provide a Group Name (required) and Description (optional).
 - b. Enable the Use only the fields supported by scoping in Access Management configuration.
 - c. Select Filter panel → Business Application Names → choose an application.
 - d. Click Create Dynamic Group.

The Asset Group is scoped to applications.

NOTE:

You cannot create SBAC based on static groups.

For more information about Asset Groups, refer to Asset Groups.

4.11.4.3 | Scope user access to an application

Scoping user access by application ensures that permissions are applied consistently across all related assets. Users receive access through their membership in application-scoped User Groups.

Assign application-based SBAC to a User Group

Define a User Group with SBAC permissions by setting its scope to include assets in Asset Groups that have application properties configured.

1. Navigate to Settings → Configurations → User Groups (under Access Management).
2. Right-click on a group in the table → Edit Group → select the Scope tab.
3. Define the application scope:
 - a. **Scope assets:** Select Assets → Select asset groups → select an Asset Group associated with applications.
 - b. **Scope cases and issues:** Select Cases and Issues → All cases and issues.
4. Click Save.

NOTE:

For more information about User Groups, refer to User group management.

Add users to the application-scoped User Group

Add users to the User Group so they inherit the application-specific permissions and access to all related child resources, such as repositories.

1. Select Settings → Configurations → Users (under Access Management).

2. Right-click the relevant user → select Edit User Permissions.
3. Select the Scope tab.
4. Scope assets: Select the chevron icon (>) in the Assets field → Select Asset groups → select the user group scoped to the application (see above).
5. Scope cases and issues: Select the chevron icon (>) in the Cases and Issues field → Select All Cases and issues.
6. Click Save.

4.11.4.4 | Create application-scoped policies

The process for creating an Cortex Cloud Application Security application-scoped policy is the same as for a standard policy. The only difference is on the Scope step of the wizard, where you can restrict the policy to a specific application(s) and their associated assets. If your user access is application-scoped, you can create policies only within your assigned scope. All other steps remain unchanged.

NOTE:

Application-scoped policies apply to both code and CI/CD configuration policies.

1. Navigate to Modules → Application Security → AppSec Policies (under Policy Management) → Add Policy.
2. Configure the General and Conditions steps of the wizard.
3. On the Scope step of the wizard.
 - a. Select Asset Types as the scope.
 - b. Select Add Filter → Business Application Names → enter the required application name.
 - c. Click Next.
4. Complete the remaining steps in the wizard to create the policy.

The policy is displayed in the general AppSec Policies table, which reflects your application scope, displaying only the policies associated with applications you can access. Users with broader permissions can filter by Business Application Names to find application-scoped policies.

For more information about creating Cortex Cloud Application Security policies, refer to Create Cortex Cloud Application Security policies.

4.12 | Repositories as assets

The Repository asset inventory provides comprehensive visibility of all your repositories integrated with Cortex Cloud Application Security, providing detailed information and insights into repository artifacts, configurations, and dependencies. You can directly access issues, and findings related to repository assets from the Repository assets page, allowing you to prioritize and remediate them without having to navigate to a separate remediation section.

4.12.1 | Explore repository assets

To access repository assets, under Inventory, select All Assets → Repositories (under Code).

The Repositories assets page includes a dashboard and an inventory.

Repository dashboard

The dashboard includes two widgets:

- Providers: Displays connected version control providers (such as GitHub and GitLab) and the number of repositories found in each provider
- Privacy State: Shows the distribution between public and private repositories and the amount of repositories in each category

Selecting an item in either widget filters the table accordingly.

Repository asset inventory

The following table describes selected Repository properties of the inventory table.

Property	Description
Repository Name	The name of the repository in the version control system (VCS).
Repository Provider	The VCS platform hosting the repository, such as GitHub, GitLab, or Bitbucket
Repository Organization	The organizational structure (such as project, team, platform) that contains and manages the repository
Repository labels	Labels associated with the repository
Application Ids	The identifier of the application to which the repository belongs, indicating it is part of the application's assets.
First observed	The date the repository was initially detected in a scan
Observation time	The date the repository was last updated
Scanned Branches	The branch of the repository that is scanned (default: main/master)

4.12.2 | In-depth repository asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Code to Cloud tab (providing context on the asset's path to production), an Applications tab (displaying the applications associated with this asset), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

Repository asset summary

The repository asset summary, displayed at the top of the card, provides concise details about the repository, including the organization and repository and the version control system to which it belongs.

Overview

The Overview tab summarizes repository highlights, properties and scan information.

Highlights provide key security and operational insights related to the repository:

- Critical/ High issues: An aggregation of critical and high issues discovered within the repository assets across all scan types (IaC, Secrets, SCA) as well as ingested third party SAST findings. Selecting this field redirects to the main issues table, filtered by the repository and its critical and high issues
- Deployed: Whether the repository is deployed
- Public: Whether the repository is public
- **Risk summary:** The amount of cases, issues and findings associated with the repository, including their severity. Selecting cases or issues redirects to their respective main pages, automatically filtered by the repository, where you can view more detailed information

For more information about issues and findings, refer to Cortex Cloud Application Security code scanners

- **Visibility timeline:** When the repository issues were first and last detected

Properties:

- Asset details, including the Asset Id, Asset Category, Asset Group and Account ID associated with the repository
- Repository details: Provides information about the repository. This includes the provider (such as a version control system hosting the repository, for example as GitHub), the scanned branch, the programming languages or technologies used within the repository (such as Terraform), its visibility configuration (public or private), whether it's exposed to the internet, archived, the timestamp of the last commit, and a list of owners

Scan information

A list of scans conducted on the repository. Details include the scan type (Periodic, CI, PR), the specific branch of the package that was scanned, the timestamp of the last scan, the health status of the scan (Completed, Failed, Partially, In progress), and the PR/CI status of the scan (Passed, Failed).

For more information about scan management, refer to Overview.

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Code to Cloud

The Code to Cloud tab describes the integrated flow of a selected Infrastructure as Code (IaC) asset, from development to its deployed state. The graph visualizes the path to production, showing the IaC resource's journey from the repository node where it's hosted, through the CI/CD system, and finally to the traced runtime resource it provisions.

For more information on Cortex Cloud Code to Cloud, refer to Code to Cloud.

Applications

The Applications tab provides an overview of the applications associated with the repository, including a graphical representation of their path to production, which incorporates the repository role within the workflow.

For more information about applications, refer to Applications.

Vulnerabilities

The Vulnerabilities tab provides a list of vulnerabilities identified within the repository in your environment. Each vulnerability includes details regarding its severity level, associated CVE identifier, CVSS score, initial detection date, and assigned team member or group responsible for remediation.

The table includes the following default properties. Click on the Table Settings Menu for additional properties.

- Severity: The vulnerability severity level
- Issue Name: The CVE identifier
- Asset Name: The asset in which the vulnerability was detected. Selecting this attribute displays the asset side card without having to navigate away from the Repository page
- Branch: The branch in which the vulnerability was detected
- CVSS Score: The Common Vulnerability Scoring System score that quantifies the severity of the vulnerability
- Assigned To: The person or team responsible for addressing the vulnerability
- Dependency Type: Indicates whether the dependency is direct (explicitly declared in your project) or transitive (pulled in by one of your other dependencies)
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability)
- Creation Date: The date when the vulnerability was detected

For more information about SCA vulnerabilities, Software Composition Analysis (SCA) vulnerability issues

IaC Configurations

The IaC Configurations tab displays an inventory of IaC misconfiguration across all repository assets.

The table includes the following default properties. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the IaC misconfiguration
- Issue Name: The IaC misconfiguration identifier
- Asset Name: The name of the IaC resource in which the misconfiguration occurred. Selecting this attribute displays the asset side card without having to navigate away from the Repository page
- Branch: The branch in which the IaC misconfiguration was detected
- Assigned To: The person or team responsible for addressing the issue
- Creation Date: The date when the issue was detected

For more information about IaC misconfiguration, refer to Infrastructure as Code (IaC) misconfiguration scanner.

CI/CD Configuration

The CI/CD Configuration tab includes a table with the following exposed properties, listing CI/CD Risks. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the risk
- Name: The risk identifier
- Asset Name: The asset in which the risk was detected
- Description: A description of the risk
- Asset Category: The asset category associated with the risk

Secrets

The Secrets tab displays an inventory of Secrets detected within the repository.

The table includes the following exposed properties. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the exposed Secrets
- Issue Name: The Secrets identifier
- Branch: The branch in which the secret was detected
- Assigned To: The person or team responsible for addressing the Secrets
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability)
- Creation Date: The date when the Secrets were initially detected

For more information about Secrets, refer to Secrets scanners.

Package Integrity

The Package Inventory tab provides details about the popularity and maintenance of packages identified within the repository. It also includes an inventory of package operational risk issues and license issues,

offering a comprehensive view of the package's overall health and compliance.

The License Issue table includes the following properties. Click on the Table Settings Menu for additional properties.

- **Severity** level: Indicates the level of severity of the package license miscompliance
- Issue Name: The package license miscompliance identifier
- License Name: The name of the license associated with the package. This indicates the specific license agreement that is potentially being violated
- Asset Name: The name of the asset that uses the package with the license miscompliance. This identifies where the license issue occurs
- Branch: The branch of the codebase where the asset with the license issue is located

The Operational Risk Issues table includes the following properties. Click on the Table Settings Menu for additional properties.

- Severity: Indicates the level of severity of the package operational risk
- Issue Name: The package operational risk identifier
- Asset Name: The name of the asset that uses the package with the package operational risk
- Branch: The branch of the codebase where the asset with the package operational risk is located
- Assigned To: The person or team responsible for addressing the package operational risk
- Creation Date: The date when the package operational risk was initially detected

For more information on Package Operational Risks, refer to [Package Integrity](#).

Code Weaknesses

The Code Weaknesses tab provides an inventory of ingested SAST (Static Application Security Testing) CWEs (Common Weakness Enumerations) identified within the repository. Each CWE is listed with its corresponding severity level, allowing you to prioritize remediation efforts based on the potential impact on the repository's security posture.

The CWE table includes the following properties. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the CWE issue
- Name: The CWE identifier
- Branch: The branch of the codebase where the asset with the code weakness is located
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability)
- Assigned To: The person or team responsible for addressing the CWE issue
- Creation Date: The date when the CWE issue was detected

For more information about about third party code weaknesses, refer to [Manage code weaknesses](#).

4.12.3 | Manage Repository assets

You can perform these actions on repository assets.

Right-click on a row in the inventory table to take the following actions:

- Open in new tab: Opens the asset description card in a new tab
- View asset data: Display asset data. Formats: JSON, Tree View
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Show/hide rows with [Asset_Name]: Show/hide rows matching the [asset name] of the selected row
- Open in Cortex Assistant/Open in Cortex Agentic Assistant: Displays the repository asset in Cortex Assistant or Cortex Agentic Assistant.
- Ingest Sarif: Allows you to upload a file to ingest third party Sarif data

4.12.4 | Export Software Bill of Materials (SBOM)

You can generate and export a Software Bill of Materials (SBOM) for a specific repository to gain a comprehensive inventory of its software components and their dependencies. To create a SBOM:

1. Select a repository from the Repository asset inventory.
2. Click more options (represented by three dots).
3. Configure the following settings from the Export SBOM dialog box:
 - a. Level: Level of data: Select the scope of data to include in the SBOM: Options: Repository, Organization (downloads the SBOM for the entire VCS organization associated with the repository)
 - b. Format: Output format: Select the output format for the SBOM. Options:
 - CycloneDX v1.4: XML or JSON
 - CycloneDX v1.5: XML or JSON
 - CycloneDX v1.6: XML or JSON
 - SDPX v2.3: JSON

4.12.5 | Manage issues detected in repositories

The **Repositories** assets inventory provides an overview of the security issues identified by various scanners that analyze the repository's code and configuration. This includes the number and severity of issues detected in each repository.

You can remediate these issues directly from the asset inventory:

1. Select a repository from the inventory table.

A card is displayed with expanded repository details, including these types of issues detected during repository scans organized by tab according to category. Refer to In-depth repository asset information for more details about available issue categories in repository assets.

2. Click on a tab including an issue.

A list of issues for the selected type is displayed.

3. Select an issue from the list.

A card with detailed issue information, including remediation options, is displayed.

4. Remediate the issue:

- For Secrets exposure, refer to Secrets issues
- For IaC misconfiguration, refer to IaC misconfiguration issues
- SCA vulnerabilities:
 - For CVE vulnerabilities, refer to Software Composition Analysis (SCA) vulnerability issues
 - For package operational risks, refer to Package integrity issues
 - For package integrity (license miscompliance), refer to License miscompliance issues
- For SAST CWE weaknesses, refer to SAST code weaknesses (CWEs)

NOTE:

You can also find the repository issues in the general issue inventory table, and in the dedicated inventory of issues for each scanner type (*see step 4 above for details*).

4.13 | Manage 3rd party findings and generated issues

Cortex Cloud Cloud platform provides a centralized view for managing security findings from both native scanners and third-party tools. It ingests code weakness findings (CWEs) from third-party sources. Software Composition Analysis (SCA) CVE vulnerabilities are ingested from third-party tools and are also detected by the platform's native scanners (In addition, native scanners also detect Infrastructure as Code (IaC) misconfigurations and exposed Secrets). By unifying all of this data, Cortex produces actionable issues that streamline remediation and give you a single, clear view of your application security posture.

For more information on SCA CVE issues, refer to Vulnerability issues inventory.

4.14 | Manage code weaknesses

Abstract

Ingest third-party SAST findings to create actionable issues, enabling you to prioritize and track remediation and enhancing your security posture.

You can ingest Static Application Security Testing (SAST) findings from third-party vendors. These findings are raw security observations from your external scanners that Cortex Cloud then uses to generate actionable

issues. These issues are prioritized by Cortex Cloud Application Security, enabling you to remediate specific code weaknesses.

Cortex Cloud Application Security default policies enrich and categorize ingested Critical and High SAST findings detected in your organization's environment as issues (also known as Code Weaknesses). Issues represent the smallest unit for remediating SAST-identified CWEs.

NOTE:

You can customize policies to define which findings are categorized as issues.

How to access code weakness issues

To access SAST code violation issues, under Modules, select Application Security → Issues → Code Weaknesses.

You can also view SAST CWE issues in dedicated tabs under other sections when available:

Read more...

TIP:

- In code Asset inventories, navigate to Inventory → All Assets → Code:
 - On the Code Weaknesses tab under the Repositories asset inventory. Refer to In-depth repository asset information for more information
 - Under the All Code asset inventory: Select an asset from the table → Code Weaknesses
- In the Application asset inventory: navigate to Inventory → All Assets → Applications → select an option from the Applications menu → select an application from the inventory → Code Weaknesses tab
- In Cases and Issues; perform a query. Select Issues → AppSec Issues (under the All Domains menu) → SAST Scanner (as the Detection Method value)

4.14.1 | Code weaknesses issue inventory

The SAST code weakness issues inventory includes the following fields.

Read more...

Property	Description
Severity	Severity level of the code weakness issue (such as Critical, High, Medium, Low)
Name	Short, descriptive name of the code weakness issue (such as "SQL Injection," "Cross-Site Scripting")

Property	Description
CWE(s)	Common weakness enumeration (CWE) identifiers associated with the issue
OWASP Categories	Top 10 OWASP categories
Asset Name	Name of the repository affected by the CWE issue (such as library name, file name)
Language	Programming language in which the CWE issue was detected (such as Java, Python, JavaScript)
Branch	The specific branch or version of the code where the CWE issue was detected
File Path	Path to the file or location within the code where the CWE issue was detected
Data Source	The 3rd party data source for the code weakness such as GitLab or GitHub
Risk Factors	Classifies the issue based on industry-standard categories, such as OWASP Top 10, CWE Top 25, providing a standardized understanding of its type and prevalence
Status	The issue status. Values: New, In Progress, Resolved. You can set the status as required
Backlog Status	Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner

Property	Description
Created	When the issue was created
Assignee	The entity assigned to mitigate the issue

Selecting an issue in the table opens a card with tabs including additional information about the issues, including suggested remediation.

Summary

A summary of the code weakness including the severity level, the CWE identity and the type of engine that detected the weakness.

Overview

The Overview tab provides general details of the SAST CWE:

- Description: Provides a summary of the CWE and its potential impact
- Status: Displays the current state of the issue
- **Timestamps**: Provides the date the issue was created and last updated
- Assignee: Assign the CWE issue to the appropriate team member remediation using the dropdown menu
- Affected Assets: Identifies the version control system and repository containing the CWE
- Evidence:
 - Details and the location in the codebase of the code containing the CWE, including vulnerability classifications (such as OWASP) specific code lines and functions
 - **Commit details**: Includes the commit hash, committer, and the assigned user responsible for remediation
- AppSec Rule: The detection rule that flagged the issue
- Weakness Details: The CWE identifier with a link to the weakness in the MITRE database
- Remediation: Suggested manual remediation steps to address the CWE issue

Actions

The Actions tab displays suggested steps to mitigate the CWE issue.

4.14.2 | Detailed code weakness issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- Issue metadata

- Timestamp: When the issue was created and last updated
- Status: Displays the current state of the issue. Values: New, In Progress, Resolved. You can set the status manually as required.

Note:

- Status changes are permanent in the current state (no automatic reopening)
- If a previously resolved finding reappears in a new scan, a new issue may be created
- Resolved status: The issue is marked as addressed and removed from active management; it no longer affects system metrics
- Assignee: You can assign the issue to a person responsible for resolving this issue. Human entities are not supported
- Rule: The Cortex Cloud Application Security rule that detected the finding. Selecting the link in the field redirects to the AppSec Rules table, filtered by the selected rule (Only applies to IaC, Secrets and CI/CD rules)
- Policy: The violated security standard that led to the detection and creation of the issue. Selecting the link in the field redirects to the AppSec Policies table, filtered by the selected policy
- SLA (Service Level Agreement): Indicates the remediation timeline status for security issues. For example Overdue indicates that the issue has exceeded the target remediation timeframe. For more information about SLA, refer to Service Lead Agreements (SLA)
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog or new, refer to Issue/Finding classification by scanner
- Description: Provides a concise technical summary of the specific security finding detected. It identifies what the issue is by referencing standard identifiers (such as the CVE ID for vulnerabilities or the CWE name for code weaknesses) and explaining the nature of the flaw, misconfiguration, or risk within the asset
- Impact: Defines the security or operational consequences of an unresolved or exploited issue. Use this field to translate technical findings into business risk, such as unauthorized access
- Affected Assets: The specific asset in which the issue was identified. Clicking on the asset opens the asset side card without needing to navigate away to the asset table
- Related affected assets: Assets associated with the primary asset based on the specific scanner used. Examples include package managers (such as pip) for vulnerabilities, IaC frameworks (such as Terraform) for misconfigurations, and third-party detection engines (such as Semgrep) for code weaknesses
- Linked Cases: The number and severity of cases associated with this issue. Selecting the link opens the Cases side-card for more information without having to navigate away
- Evidence: Provides evidence and contextual details about the issue:

- Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
- Location: The exact technical context of a finding, linking security findings to specific lines of code or infrastructure files:

Examples:

- **Vulnerabilities (SCA)**: Identifies the manifest file path (such as `package-lock.json`) and the specific line where the vulnerable package is declared, providing a declaration snippet as context
- **IaC misconfigurations**: Points to the configuration file (such as `main.tf`) and the resource block's start and end lines. Context includes the full resource configuration and the specific resource identifier
- **Secrets**: Locates the file and line number of the exposure, providing a code snippet with the secret redacted for security
- **Code Weaknesses (SAST)**: References the source code file path and the start/end lines of the flaw. Context displays the vulnerable code snippet and the affected function or method name
- Collaborator: The individual or team responsible for contributing to the code or configuration where the issue was identified
- Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
- Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
- Urgency Details includes:

- **Summary:** The issue's urgency level, a breakdown of its contributing metrics, and the date it was last updated (Tip: Hover over a metric for more information)
- **Urgency context graph:** The Urgency graph provides a node-and-edge visualization that maps the structural relationship between a vulnerable asset and its connected infrastructure. By surfacing the asset hierarchy and deployment paths, the graph offers the context necessary to evaluate the scope and potential blast radius of an issue's urgency across the environment. This visualization allows you to analyze the specific deployment and runtime dependencies where the issue was detected. The urgency level itself is determined by metrics derived from the analysis of the connected assets.

Supported scanners: Vulnerabilities (SCA), Secrets, Code Weaknesses (ingested from third -party vendors) and IaC misconfigurations.

Graph structure:

- **Nodes (assets):** Represents assets such as repository or pipeline, that are linked to the specific asset where the issue was detected.

Clicking on a node opens a side card showing: Clicking on a node opens a side card showing initial details about the asset. Selecting View Details opens the asset side card without navigating away, displaying asset details, asset-specific information, and related context for that asset.

- **Edges (relationships):** The edges in the Urgency graph represent relationships between different asset types in the code-to-cloud deployment pipeline. These relationships trace how code flows from repositories through to runtime environments, providing the structural data required to calculate urgency metrics. By analyzing these relationships, Cortex Cloud determines the urgency level based on metrics such as whether the code is actively deployed, internet-exposed, accessing sensitive data, or leveraging privileged capabilities within its runtime environment.

- **Interactive controls:**

- **Layers control:** Toggle visibility of Code, Build, or Runtime layers to focus on specific pipeline segments
- **Group nodes:** Collapse multiple related nodes (such as assets within the same organization or namespace) into a single group node to manage visual density
- **Search and filter:** Locate specific nodes by typing an asset name or ID. Matching nodes are highlighted and the graph auto-focuses on the results
- **Zoom and pan:** Navigate large topologies using zoom buttons, Fit to View, or drag-and-drop panning

For more information about Urgency levels, refer to [Urgency](#).

- **Remediation:** Suggested steps to mitigate the issue. For the most efficient resolution, use the Actions tab, which provides a complete list of remediation options, including PR fixes where available

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

The Actions tab displays suggested steps to mitigate the CWE issue. Only suggested manual fixes are available.

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

4.14.3 | Code weakness findings

SAST CWE findings are based on ingested third party (such as Semgrep) data. Findings are potential security vulnerabilities identified within your source code based on common weakness enumerations (CWEs). These insights help assess and analyze the security posture of your applications by identifying weaknesses in your codebase.

NOTE:

Findings on the Cortex platform are not intended for direct action; but rather represent data collected by the platform. They must be promoted to issues to enable mitigation efforts to secure your codebase.

To access code weakness findings, navigate to code weakness issues (see SAST code weaknesses (CWEs)) and click the Findings tab.

The following table displays selected code weakness findings properties.

Read more...

Property	Description
Name	Short, descriptive name of the CWE finding (such as "SQL Injection," "Cross-Site Scripting")
CWE(s)	CWE identifier(s) associated with the finding (such as CWE-79, CWE-119)
OWASP Categories	Relevant Top 10 OWASP categories associated with the finding (but can be from different years)
Asset Name	Name of the repository affected by the CWE finding
Language	Programming language in which the CWE finding was detected (such as Java, Python, JavaScript)

Property	Description
Branch	The specific branch or version of the code where the CWE finding was detected
File Path	Path to the file or location to the code wherein the CWE finding was detected
Git User	Username of the Git user who last modified the file containing the finding
Data Source	Source of the CWE finding information
Created	Timestamp of when the CWE finding was first detected.
Finding ID	Unique identifier assigned to a specific finding

Selecting a finding from the table provides additional details:

- Overview: Includes when the finding was last updated, the category associated with the finding, and the name and link to the asset where the finding was detected
- Details: The location of the finding, the third party data source that detected the finding, the CWE category, the initial hash and commit, and rule ID

5 | CI/CD Security

Abstract

CI/CD security safeguards software components and process throughout the SDLC.

CI/CD security scans for code and configuration violations in the pipeline safeguard software components and processes throughout the automated CI/CD pipeline. These scans address the security of the initial code commit and the automated build processes by identifying code vulnerabilities and misconfigurations within your codebase and pipeline configurations. By catching these violations early, they contribute to maintaining the integrity, confidentiality, and availability of modern software systems and fostering trust within the pipeline's operations.

CI/CD Security use cases

- **CI/CD onboarding:** Integrate your CI/CD pipelines and related systems with the CI/CD module to gain centralized visibility and consolidate code risks, enabling security insights and pipeline hardening
- **CI/CD assets:** Identify and manage all critical CI/CD assets, including collaborators, CI/CD instances, CI/CD pipelines and version control system (VCS) organizations, to gain a comprehensive view of your attack surface
- **Supply Chain security:** Secure your SDLC by gaining visibility into and controlling third-party dependencies, open-source components, build artifacts, and CI/CD pipeline configurations and activities. Identify and mitigate risks introduced throughout the build and deployment processes, ensuring a secure software delivery
- **CI/CD risks:** Understand and mitigate common CI/CD risks, including OWASP Top 10 CI/CD security risks, by gaining visibility into pipeline security posture, visualizing breach pathways, and identifying exposed credentials
- **CI/CD rules:** Define and enforce granular security rules within your CI/CD pipelines to block insecure code from reaching production and apply consistent security policies across repositories, registries, and runtime environments
- **CI/CD policies:** Create and apply comprehensive security policies across your entire software supply chain, enforcing them throughout the SDLC to harden delivery pipelines and ensure only vetted code reaches production
- **CI/CD Compliance:** Scan CI/CD rules against selected compliance standards such CIS GitHub to generate comprehensive and recurring compliance reports. This ensures continuous, verifiable adherence for auditing

5.1 | CI/CD Security user roles and permissions

Cortex Cloud offers dedicated user roles for CI/CD Security: AppSec Admin and DevSecOps, each with specific areas of responsibilities. When assigning roles to users, it's recommended to align them with the user's required responsibilities within the application security framework.

The AppSec Admin has full permissions for all Cortex Cloud Application Security-related activities. They can create and modify detection rules within the Code/Build domain, track progress, and adjust enforcements as needed. Additionally, they can triage and investigate findings, issues, and cases spanning from code to cloud. The role also includes complete visibility into all cloud assets.

The DevSecOps role is specifically designed as an intermediary, possessing more permissions than a Developer but fewer than an AppSec Admin. This role actively manages security processes and tools to embed security directly into development and operations workflows. Responsibilities include managing and resolving security issues, performing scan management, and improving the overall application security posture by integrating security practices throughout the development and operations lifecycle.

Permissions assigned to predefined roles cannot be modified. However, you can save a predefined role as a new custom role. This custom role can then be edited to meet specific organizational needs, offering a balance between standardized roles and customizable access control.

Dedicated CI/CD Security roles include permissions that extend beyond CI/CD Security itself. In addition to these dedicated users, other roles within your tenant are also granted specific permissions to CI/CD Security.

You can view all permissions granted to user roles in your tenant by navigating to Settings → Configurations → Roles → select a role.

5.2 | CI/CD Assets

Abstract

Identify and manage all CI/CD assets—collaborators, VCS, instances, pipelines—for a complete view of your software supply chain attack surface.

CI/CD assets are the fundamental components that enable your automated software delivery pipeline. These include your collaborators, version control system (VCS) organizations (the top-level structures within VCS platforms that contain your repositories, code, and configurations), CI/CD instances (the tools that execute your pipeline) and CI/CD pipelines (the automated workflows that build, test, and deploy your software).

VCS Repositories are part of both ASPM and CI/CD asset modules. Refer to [Repositories as assets](#) for more information about repository assets.

Visibility into these assets allows you to understand your CI/CD environment. In addition, the assets display associated risks identified through CI/CD security scans, allowing you to remediate CI/CD issues directly from the dedicated asset inventories.

5.2.1 | CI/CD Instances as assets

The CI/CD Instances inventory provides a centralized view of all CI/CD pipeline tool instances (such as GitHub Actions and Jenkins) across your environment, enabling efficient tracking and management of your CI/CD instances. You can access and analyze instance details and properties, as well as review CI instance configurations and identify configuration issues.

5.2.1.1 | Explore CI/CD Instance assets

To access CI/CD pipeline instances, under Inventory, select All Assets → Code → Category → CI/CD Instances.

The CI/CD Instances assets page includes a dashboard and an inventory. The dashboard includes a Provider widget, displaying the types of CI/CD providers configured (for example, GitHub Actions) and the number of instances for each provider. You can filter the inventory by selecting a provider type.

[CI/CD instance inventory](#)

The following table describes selected CI/CD instance properties displayed in the inventory table.

[Read more...](#)

Property	Description
Name	The name of the CI/CD pipeline instance, often including a descriptor that indicates its environment or purpose, such as "Jenkins-prod" for a Jenkins instance in a production environment
Provider	The provider or system of the CI/CD pipeline instance, such as Jenkins
URL	The web address where the CI/CD pipeline instance can be accessed
Last Observed	The date when the last scan was conducted on the CI/CD pipelines in the instance. Use the column filter to choose a specific date range (custom, 7 days, 30 days) instead of the default 24 hours
Pipeline Count	The amount of CI/CD pipelines associated with the instance. Clicking on the value opens the Pipelines tab of the description card for additional information about the associated pipelines

5.2.1.2 | In-depth CI/CD pipeline instance asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Code to Cloud tab (providing context on the asset's path to production), an Applications tab (displaying the applications associated with this asset), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

CI/CD pipeline instance summary

The CI/CD pipeline instance summary, displayed at the top of the card, provides concise details about the CI/CD instance, such as its name, the provider (for example GitHub Actions), and specific pipeline configurations.

Overview

The Overview tab summarizes the CI/CD pipeline instance highlights and properties.

Highlights:

- **Critical/High issues:** An aggregation of critical and high issues associated with the CI/CD pipeline instance
- **Deployed:** Indicates whether the CI/CD pipeline instance has been deployed and is currently active within your cloud environment or infrastructure
- **Risk summary:** The amount of issues and findings associated with the pipeline instance and their severity level. Selecting an issue or finding redirects to the issues or findings page, filtered by the selected issue or finding
- **Visibility timeline:** When the CI/CD pipeline instance was first and last detected

Properties:

- Asset details, including Asset Id, Asset Category (fixed value: CI/CD Instance), Provider (CI tool such as Jenkins and GitHub Actions) and Asset Groups associated with the CI/CD pipeline instance
- Instance URL: Link to the specific CI/CD instance

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Configurations

The Configurations tab provides a table of CI/CD instance configuration issues. The following table provides selected CI/CD configuration instance issue properties.

Property	Description
Severity (icon)	The severity level of the issue
Issue Name	The issue identifier
Asset Name	The asset in which the issue was detected
Assigned To	The person or entity assigned to remediate the issue
Creation Date	When the issue was initially detected
Issue Status	The status of the issue

Pipelines

The Pipelines tab displays a list of CI/CD pipelines associated with the CI/CD instance.

Property	Description
Pipeline Name	The name of the CI/CD pipeline.
Pipeline File Path	The location of the configuration file that defines the CI/CD pipeline within the associated repository
Related Repositories	A list of repositories that are associated with the CI/CD pipeline
Last Observed	The timestamp indicating the most recent time this specific CI/CD pipeline was detected
Application IDs	Unique identifiers assigned to the application(s) that this CI/CD pipeline builds, tests, or deploys

5.2.1.3 | Manage CI/CD pipeline instances

You can perform the following actions CI/CD instances.

Asset actions

- Right-click on an asset in an inventory table to access the Actions menu, where you can perform the following actions:
 - Open in new tab: Opens the description tab of the asset for detailed analysis of the issue
 - View asset data: Opens a new pop-up window displaying the data retrieved for the asset during the most recent scan in either JSON (default) or tree view. This raw data provides a comprehensive and unformatted view of the asset's properties and attributes as they were initially ingested
 - Copy text to clipboard: Copies the selected text to the clipboard
 - Copy entire row: Copies the entire selected row data
 - Show/hide rows: Stand on data in a row and filter the entire inventory to show or hide assets based on the selected attribute
 - Open in Cortex Assistant/Open in Cortex Agentic Assistant: Opens the repository in Cortex Assistant or Cortex Agentic Assistant.
- **Export asset data:** Click the download icon (showing Export to file when hovering over the icon) in the top right of any asset page to export the asset data
- **View Dashboard:** Redirects to the Application Security dashboard

5.2.2 | CI/CD Pipelines as assets

The CI/CD Pipelines assets provide a centralized view of all CI/CD pipeline assets across your environments, enabling efficient tracking and management. You can access and analyze CI/CD pipeline details, properties and insights including deployment status, whether deployed, active or new and a summary of findings and issues associated with pipelines. This allows you to assess the security and operational status of your pipelines.

5.2.2.1 | Explore CI/CD Pipeline assets

To access CI/CD pipelines assets, under Inventory, select All Assets → Code → CI/CD Pipelines.

The CI/CD pipelines assets page includes a dashboard and an inventory.

CI/CD Pipeline assets dashboard

The dashboard includes a widget detailing the CI pipeline providers. You can filter by provider.

CI/CD Pipeline assets inventory

The following table describes selected CI/CD Pipeline asset properties displayed in the inventory table.

Property	Description
Name	The unique identifier assigned to the pipeline
Provider	The tool or service that supplied the pipeline, such as GitHub Actions or Jenkins
Repository	The code repository which stores the source code, pipeline configurations, and related assets used for the CI/CD process
CI File Path	The specific location or directory path where the CI configuration file is stored
CI Instance	The individual occurrence of the CI associated with the pipeline

5.2.2.2 | In-depth CI/CD pipeline asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Code to Cloud tab (providing context on the asset's path to production), an Applications tab (displaying the applications associated with this asset), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

CI/CD Pipeline asset summary

The CI/CD Pipeline asset summary, displayed at the top of the card, provides concise details about the CI/CD pipeline assets, such as its name, the platform used (for example GitHub Actions) and specific pipeline configurations.

Overview

The Overview tab summarizes CI/CD pipeline asset highlights and properties.

Highlights:

- Critical/High issues: An aggregation of critical and high issues associated with the CI/CD pipeline asset
- Deployed: Indicates whether the CI/CD pipeline asset has been deployed and is currently active within your cloud environment or infrastructure
- New: Indicates whether the CI/CD pipeline asset was created during the past 30 days
- Active: Indicates whether the CI/CD pipeline asset is active and processing tasks
- **Risk summary:** Displays the total amount of risks associated with the pipeline asset grouped by category (cases, issues and findings) and their severity level. Selecting a risk category will redirect you to more information
- **Visibility timeline:** When the CI/CD pipeline assets were first and last detected

Properties:

- Asset details, including Asset Id, Asset Types and Asset Groups associated with the CI/CD pipelines asset
- Applications: Lists the applications that include this pipeline assets as part of their defined assets or configuration
- CI/CD configuration:
 - Provider: The platform or service that hosts and manages the CI/CD pipeline, such as Jenkins and GitHub Actions
 - CI File Repository: The location or repository where the configuration files for the CI/CD pipeline are stored
 - CI Instance: The specific instance or environment where the CI/CD pipeline is executed
- Last Job Execution: The most recent execution of a job within the CI/CD pipeline
- Contributors: The individuals or entities who have made contributions to the CI/CD pipeline. This information allows collaboration within the CI/CD pipeline's development process

NOTE:

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Code to Cloud

The Code to Cloud tab describes the integrated flow of a selected CI/CD pipeline, showing its journey from code commit to deployment. The graph visualizes the path to production, showcasing the pipeline's central role in orchestrating the flow from the repository, through its build and test stages, to the image it creates, and finally to the traced runtime resource it deploys.

For more information on Cortex Cloud Code to Cloud, refer to [Code to Cloud](#).

Applications

The Applications tab provides an overview of the applications associated with this CI/CD pipeline, including the application risk score, business criticality, business owners and path to production. The path to production

provides a graphical representation the application software development lifecycle, including the CI/CD pipeline role within the workflow.

For more information about applications, refer to Applications.

Instances

The Instances tab displays a list of CI instances associated with the pipeline. The following table displays CI instance properties.

Property	Description
Instance Name	The name of the specific CI/CD instance
Instance ID	The identifier for the specific CI/CD instance
Instance Provider	The name of the CI/CD tool or platform that manages the instance (such as Jenkins, GitHub Actions)
Instance URL	An address that allows you to access the user interface or details page of the specific CI/CD instance within its native platform
Last Observed	The timestamp indicating the most recent time this specific CI/CD instance was detected

5.2.2.3 | Manage CI/CD pipeline assets

You can perform the following actions on CI/CD pipeline assets.

Asset actions

- Right-click on an asset in an inventory table to access the Actions menu, where you can perform the following actions:
 - Open in new tab: Opens the description tab of the asset for detailed analysis of the issue
 - View asset data: Opens a new pop-up window displaying the data retrieved for the asset during the most recent scan in either JSON (default) or tree view. This raw data provides a comprehensive and unformatted view of the asset's properties and attributes as they were initially ingested
 - Copy text to clipboard: Copies the selected text to the clipboard
 - Copy entire row: Copies the entire selected row data
 - Show/hide rows: Stand on data in a row and filter the entire inventory to show or hide assets based on the selected attribute
 - Open in Cortex Assistant/Open in Cortex Agentic Assistant: Opens the repository in Cortex Assistant or Cortex Agentic Assistant.
- **Export asset data:** Click the download icon (showing Export to file when hovering over the icon) in the top right of any asset page to export the asset data
- **View Dashboard:** Redirects to the Application Security dashboard

5.2.3 | Version Control System (VCS) Organizations as assets

The VCS Organizations asset inventory provides a centralized view of all VCS Organizations that are integrated with Cortex Cloud, including their repositories and properties. The platform enables efficient discovery and management of these VCS organization assets, providing insights and analysis to contextualize their importance within your ecosystem and assess their risk posture. You can directly access related issues and findings within the VCS Organizations asset inventory, allowing you to prioritize and remediate them without navigating to a separate remediation section.

5.2.3.1 | Explore VCS Organization assets

To access VCS Organization assets, under Inventory, select All Assets → Code → VCS Organizations.

The VCS Organization assets page includes a dashboard and an inventory.

VCS Organization asset dashboard

The Providers widget is displayed by default on the dashboard, showing the distribution of VCS organizations across integrated version control systems. Selecting a VCS within the widget sorts the inventory table accordingly.

VCS Organization asset inventory

The asset inventory table includes exposed properties as well as selected key properties that are available but not displayed by default.

Property/Attribute	Description
VCS Organization Name	The VCS organization name
VCS Organization Provider	The version control system (such as GitHub) that the organization is associated with
VCS Organization URL	The web address (URL) of the VCS organization's main page on the respective version control system provider (such as the main GitHub organization page)
First Observed	The date of the scan that initially detected the VCS organization
Observation Time	The date that the VCS organization data was last updated
VCS Organization Connected Apps	A list of applications created within Cortex Cloud that have a connection or association with this specific VCS organization entity
Application IDs	A list of the unique identifiers (IDs) assigned to the applications that are connected to or associated with this specific VCS organization entity

5.2.3.2 | In-depth VCS Organization asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including Configurations (displaying an inventory of configurations for all associated VCS organization assets) and Identity (providing a view of users within the VCS Organization).

VCS Organization asset summary

The VCS Organization asset summary, displayed at the top of the card, provides concise details about the VCS Organization, such as its name and associated VCS.

Overview

The Overview tab summarizes VCS Organization highlights, properties, repositories and members.

Highlights:

- Critical/High issues: An aggregation of critical and high issues associated with the VCS Organization
- Deployed: Indicates whether the VCS Organization has been deployed and is currently active within your cloud environment or infrastructure
- Public Repository: Indicates whether the VCS Organization is a public repository, indicating its visibility to the public
- **Risk summary:** The amount of risks associated with the VCS Organization grouped by category (cases, issues and findings) and their severity level
- **Visibility timeline:** When the collaborator was first and last detected

Properties:

- Asset (VCS organization) details, including Asset Id, Asset Category and Asset Groups associated with the VCS Organization
- Provider: the VCS associated with the VCS Organization
- **Organization Owners:** Users with full administrative control over the version control organization, including members, repositories, and settings

Repositories: A table of repositories associated with the VCS Organization. Enables you to quickly identify specific repositories of interest and gain a comprehensive understanding of the organization's overall repository structure. The table repository properties such as name, scanned branch, visibility, last commit and associated technologies. Selecting a repository opens its asset card directly within VCS Organization assets, allowing quick access to repository details without having to redirect to the dedicated Repository assets page.

NOTE:

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Configurations

The Configurations tab displays an inventory of configurations for all associated VCS organization assets, and an inventory of top configurations issues (VCS & CI/CD Risks) related to the organization.

The table includes the following properties:

- Severity level (icon): Indicates the level of severity of the configuration issue
- Asset Name: The name of the resource in which the misconfiguration was detected
- Assigned To: The person or team responsible for addressing the vulnerability
- Creation Date: The date when the vulnerability was detected

Identity

The Identity tab provides a view of users within the VCS Organization, outlining their access levels and the repositories they are collaborators on, along with the timestamp of the latest commit for each repository

5.2.3.3 | Manage VCS organization assets

You can perform the following actions on VCS organization assets.

Asset actions

- Right-click on an asset in an inventory table to access the Actions menu, where you can perform the following actions:
 - Open in new tab: Opens the description tab of the asset for detailed analysis of the issue
 - View asset data: Opens a new pop-up window displaying the data retrieved for the asset during the most recent scan in either JSON (default) or tree view. This raw data provides a comprehensive and unformatted view of the asset's properties and attributes as they were initially ingested
 - Copy text to clipboard: Copies the selected text to the clipboard
 - Copy entire row: Copies the entire selected row data
 - Show/hide rows: Stand on data in a row and filter the entire inventory to show or hide assets based on the selected attribute
 - Open in Cortex Assistant/Open in Cortex Agentic Assistant: Opens the repository in Cortex Assistant or Cortex Agentic Assistant.
- **Export asset data:** Click the download icon (showing Export to file when hovering over the icon) in the top right of any asset page to export the asset data
- View Dashboard: Redirects to the Application Security dashboard

5.2.4 | VCS Collaborators-as-assets

The VCS Collaborator (collaborators) asset inventory provides a centralized view of all collaborators (users) who interact with VCS repositories, CI/CD pipelines, and related assets. The platform enables efficient discovery and management of these assets, providing insights and analysis to contextualize their importance within your ecosystem and assess their risk posture. You can directly access related issues and findings within the Collaborator asset inventory, allowing you to prioritize and remediate them without navigating to a separate remediation section.

How to access Collaborator assets

- To access Collaborator assets, under Identity, select Human Identities → select VCS Collaborator from the Type filter.

5.2.4.1 | In-depth Collaborator asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs.

Collaborator asset summary

The Collaborator asset summary, displayed at the top of the card, provides concise details about the collaborator, such as their name and the version control system that the collaborator is associated with.

Overview

The Overview tab summarizes the collaborator highlights and properties.

Highlights include:

- Critical/High issues: An aggregation of critical and high issues associated with the collaborator
- New: Whether the collaborator was recently added to the VCS organization
- Inactive: Indicates whether the collaborator has had no commits within the last time period

Properties include:

- **Risk Summary:** The amount of risks associated with the collaborator grouped by category (cases, issues and findings) and their severity level
- Asset details, including Asset Id, Asset Types and Asset Groups associated with the collaborator
- **Visibility timeline:** When the collaborator was first and last detected
- **Identity and Affiliation:**
 - Username: the alias associated with the collaborator
 - User Type: The classification of the collaborator's account (such as individual user, service account, bot), indicating the nature of their access and activity
 - VCS Organization: The specific version control system organization to which the collaborator belongs or has access
 - Emails: "The email addresses associated with the collaborator's account, used for communication and notifications
- **Access and Activity:**
 - Team Membership: The teams or groups within the VCS Organization to which the collaborator belongs, defining their access permissions and collaborative roles
 - Last Commit: The timestamp of the collaborator's most recent commit to a repository within the associated VCS Organization, indicating their recent activity

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Access

The Access tab provides a list of assets that a VCS Collaborator has access to within your environment. It details the specific assets they can interact with, the level of permission granted, and the timestamp of their most recent code commit to that asset.

5.2.4.2 | Manage Collaborator assets

You can perform the following actions collaborator assets.

Asset actions

- Right-click on an asset in an inventory table to access the Actions menu, where you can perform the following actions:
 - Open in new tab: Opens the description tab of the asset for detailed analysis of the issue
 - View asset data: Opens a new pop-up window displaying the data retrieved for the asset during the most recent scan in either JSON (default) or tree view. This raw data provides a comprehensive and unformatted view of the asset's properties and attributes as they were initially ingested
 - Copy text to clipboard: Copies the selected text to the clipboard
 - Copy entire row: Copies the entire selected row data
 - Show/hide rows: Stand on data in a row and filter the entire inventory to show or hide assets based on the selected attribute
 - Open in Cortex Assistant/Open in Cortex Agentic Assistant: Opens the repository in Cortex Assistant or Cortex Agentic Assistant.
- **Export asset data:** Click the download icon (showing Export to file when hovering over the icon) in the top right of any asset page to export the asset data
- View Dashboard: Redirects to the Application Security dashboard

5.3 | Supply Chain Inventories

Abstract

Supply Chain: Gain full visibility by tracking detected tools in your environment and cross-referencing them against a catalog of Cortex-recognized, trusted technologies.

The Software Supply Chain provides comprehensive visibility into the tools, services, and third-party integrations that operate across your software development and delivery processes. It includes two complementary inventories:

- Supply Chain Tools: Lists tools and their associated risk factors detected in your environment
- Supply Chain Catalog: Cortex Cloud's centralized registry of recognized supply-chain tools and their associated risk factors

Together, these inventories allow you to assess tool usage, coverage, and security posture—identifying unused, vulnerable, or unapproved tools before they expand your attack surface.

Execution environments

Cortex Cloud supports these Supply Chain Tool execution environments:

- **Third party pipelines:** Third-party plugins integrated with Cortex Cloud, provide visibility into installations, locations, and CVE vulnerabilities within your pipeline environment. This allows for prioritized remediation, effectively reducing your attack surface by identifying and removing unused or vulnerable plugins.

Supported pipeline environments include:

- GitHub Actions
- Jenkins plugins
- CircleCI Orbs
- Azure Extensions

Additionally, these pipelines often incorporate third-party executables into their workflows. Cortex Cloud offers enhanced visibility into these third-party services, transforming unreadable data into actionable insights for improved security posture.

- **VCS third parties: VCS Apps.** Third-party applications and webhooks in your version control system. This enables removal of unused assets, management of permissions, and adherence to the principle of least privilege
- **Executables:** Standalone programs or scripts executed within your CI/CD pipelines. These may include custom scripts, third-party command-line tools, or other executable files. The inventory provides insights into their usage, deployment locations, and potential security risks
- **Remote Scripts (URL).** Executable scripts fetched from a remote URL during pipeline execution. The inventory provides insights into their origin, usage, and potential security risks, addressing the unique challenges of untrusted remote code
- **Webhooks:** Automated, event-driven communications that trigger actions across your CI/CD pipeline and integrated services. Cortex Cloud provides an inventory of these webhooks, enabling you to assess their usage, coverage, and potential security risks

Tool status

Tools are categorized by status: Approved, Pending Review, or Rejected (but still in use). When initially detected, tools are assigned a Pending Review status by default, requiring further action to change the status to Approved or Rejected. Rejected does not mean the tool is not in use. It allows application security practitioners and DevOpsSec personnel to search for and remove these tools as needed.

You can modify the tool status by right-clicking on a tool in both Supply Chain Tool and Supply Chain Catalog inventories:

- In the inventory table, right-click on a tool → Change Status → select a status
- From the Overview tab on the Supply Chain side-panel.
 - Initial selection: Select a status from the available options
 - When modifying a previous selection: Select Edit → select a status

For information about changing a tool status, refer to Overview

Using the inventories

- Use Supply Chain Tools to view and manage tools detected in your environment, review usage, and prioritize remediation
- Use the Supply Chain Catalog to cross-reference detected tools against Cortex Cloud-supported ones, identify coverage gaps, and evaluate risk before integrating new tools or replacing existing ones

NOTE:

Although attributes are identical across inventories, their values for the same tool can differ, most commonly in Risk Factors and Type. This is because the inventory reflects your live environment, which may include different versions or configurations than the catalog—for example, a package may not have been upgraded or may be deployed differently.

5.3.1 | Supply Chain Tools

The Supply Chain Tools inventory provides detailed information about individual tools, including deployments, areas of non-use, and functionality detected in your SDLC. This data allows you to assess tool usage, coverage, and potential security risks.

Use cases

- **Tool visibility:** Gain visibility into all tools used across CI/CD pipelines and VCSs
- **Third-Party tools:** Discover and monitor all external tools (webhooks, executables, apps, plugins) integrated into your CI/CD pipelines and VCSs
- **Detailed Tool insights:** Access detailed information on each tool (creator, risk factors such as deprecation or low usage, usage evidence, first seen date, category) to evaluate approval status and assess risk
- **Rejected Tool Monitoring:** Manage tool approval status by approving or rejecting tools found in pipelines/VCSs, identifying non-compliant usage
- **Usage:** View tool usage indicators by category across pipelines/VCSs for management reporting internal as well as external usage

5.3.1.1 | Supply Chain Tools

The Supply Chain Tools inventory table provides a detailed list of your organization's CI/CD pipeline tools and VCS Apps, allowing you to view and manage your organization's supply chain tools from a single, centralized location. You can review tool usage, third-party integrations, and risk assessments, including creator information, usage evidence, and category details. Additionally, you can filter tools by status (approved, rejected, uncategorized) and category, search for specific tools, and identify top risks to ensure policy adherence and prioritize remediation.

How to access Supply Chain Tools

To access Supply Chain Tools, select Modules → Application Security → Supply Chain Tools (under 3rd Party Tools).

The inventory table describes the exposed Supply Chain tool properties. You can view additional properties through the Table Settings Menu.

Property/ Attribute	Description
Name	The name of the Supply Chain tool
Risk Factors	<p>Risk factors associated with the tool, as assessed by Cortex Cloud, help you identify and prioritize potential risks for tools and components based on their likely impact and exploitability. Values include Archived, Not verified, Unsecured URL and Outdated Version.</p> <p>For tools in your environment, risk factors are specific to the exact version you have, whereas catalog risk factors reflect the tool's general profile. To understand the specific reasoning behind a risk factor, hover over it to view a detailed explanation</p>
Status	The tool status. Values: Approved, Pending, Rejected
Usage	The amount of CI/CD pipelines in which the tool was used. Includes a link which opens the location in which the tool is used
Type	The type of tool
Category	The category associated with the tool, such as Version Control System (VCS), Continuous Integration (CI) Servers and Build Automation Tools

5.3.1.2 | Expanded Supply Chain tool information

When you click a tool's entry in the inventory table, a side card will open to display detailed information. The information is organized into three tabs: the Overview tab, which provides a summary of the tool's key details and is the default view; the Vulnerabilities tab, which lists any associated security vulnerabilities (CVEs); and the Actions tab, which outlines available mitigation options for the tool.

Overview

The Overview tab includes these details:

- Name: The name of the tool
- Description: A description of the tool usage and a link to its third-party origin, such as a public repository, documentation portal, or the vendor's official website
- **PAN insights:** Cortex Cloud mitigation recommendations based on risk factors to address relevant supply-chain threats
- **Timestamp:** When the tool was initially detected
- Category: The tool type, such as code scanning and analytics
- Usage: The amount of assets using this tool
- Status: The current status of the tool. Values include Approved, Pending Review, Rejected. You can manually override the system-assigned status
- Approve / Reject: Approve or reject the tool.

NOTE:

AppSec Admin user permissions are required to perform these actions.

- **Execution environments:** A list of execution environments associated with the tool. Details include: name, the number of assets using the tool and the scan type

Usage

The Usage tab provides a list of the execution environments associated with the tool. Details include

- The asset name in which the tool runs (such as a pipeline). Selecting the name opens the asset in a side-car without having to navigate to the dedicated Assets page
- The type of asset, such as CI/CD pipeline, CI/CD instance, or Organization (for VCS Apps)
- Evidence of the tool in the environment - the location of the file containing the tool
- When the tool was initially detected.

Vulnerabilities

The **Vulnerabilities** tab is displayed whenever one or more tools has a risk factor that is a result of a Common Vulnerability and Exposure (CVE). This tab provides a consolidated list of all CVEs impacting the tools. The table includes these properties:

- Name. The unique identifier for the CVE entry. For example, CVE-2023-25764. This name is a clickable link that directs you to a detailed report on the vulnerability from a public database
- CVSS Score. The numerical score assigned to the vulnerability based on the Common Vulnerability Scoring System (CVSS). This score indicates the severity of the vulnerability, with a higher number representing a greater risk
- Asset: The specific asset affected by the CVE

Comments

Select the Comments icon in the side-panel to add comments directly to catalog items, enabling collaboration and internal notes between security and development teams regarding component usage, justification, or deprecation status.

5.3.2 | Supply Chain Catalog

The Supply Chain Catalog is Cortex Cloud's centralized registry of Cortex Cloud supported supply-chain tools and their associated risk factors. The catalog is distinct from the inventory displayed on the Supply Chain Tools page, which lists tools detected in your environment. Some tools may be displayed in both inventories - for example if you use Semgrep, which is also included in the catalog.

Use case

Use the catalog to cross-reference against your inventory to identify coverage gaps, assess exposure, and benchmark your security posture before integrating new tools or to replace existing ones that may be at risk.

How to access the Supply Chain Catalog

To access the Supply Chain Catalog, select Modules → Application Security → Supply Chain Catalog (under 3rd party tools).

Supply Chain Catalog inventory

This inventory includes a list of all supply chain tools in the Catalog. The inventory table properties are identical to the Supply Chain Tools inventory table. For information about these properties, refer to Supply Chain Tools.

Expanded Supply Chain catalog information

When you click a tool's entry in the inventory table, a side card opens to display detailed information. The information is organized into three tabs:

- Overview: Provides a summary of the tool's key details and is the default view
- Vulnerabilities Lists any associated security vulnerabilities (CVEs)
- Actions: Outlines available mitigation options for the tool

The details provided in these tabs are identical to the details displayed in the expanded Supply Chain Tool Catalog. For information about these properties, refer to Expanded Supply Chain tool information.

5.4 | CI/CD Risks

Abstract

CI/CD risks identify vulnerabilities and misconfigurations in pipelines, then prioritize them into actionable issues for efficient remediation.

CI/CD pipeline risks are a set of predefined rules that identify pipeline vulnerabilities. Scans analyze both code and configurations of integrated VCS and CI/CD systems and pipelines, as well as their inter-connectivity, to detect these risks. The risks are classified based on security categories including attack vectors, misconfigurations, and bad practices found throughout your CI/CD pipelines.

CI/CD pipeline risk findings, detected during scans, are displayed in a dedicated table for analysis and investigation. Cortex Cloud Application Security then applies context and prioritizes these findings to create CI/CD pipeline risk issues. These issues represent the smallest unit of risk that can be remediated, and are displayed in their own dedicated inventory. You can remediate CI/CD pipeline risk issues manually by applying suggested fixes.

NOTE:

Cortex Cloud Application Security CI/CD pipeline scans create a comprehensive inventory of all CI/CD pipelines in your environment. For more information refer to [CI/CD Pipelines as assets](#).

5.4.1 | CI/CD pipeline issues

All Medium, High and Critical CI/CD pipeline findings detected in an organization's environment are categorized by Cortex Cloud as CI/CD risk issues. This approach allows for targeted remediation efforts. Only manual fixes are available for CI/CD pipeline risk issues.

The CI/CD Risks Issues table is a filtered instance of the broader Issues table found under Cases & Issues, meaning it exclusively displays issues categorized as CI/CD risks. However, the CI/CD Risks Issues table only displays issues generated from findings detected during periodic scans. In contrast, the comprehensive Issues table includes all CI/CD risks issues, regardless of their detection source, such as periodic, pull request (PR), and continuous integration (CI) scans.

How to access CI/CD pipeline risk issues

To access CI/CD pipeline risks issues, under Modules, select Application Security → Issues → CI/CD Risks.

CI/CD pipeline risk issue inventory

Below are selected properties of the CI/CD pipeline risks issues inventories.

Property	Description
Severity	The CI/CD pipeline risk severity level. Values: Critical, High, Medium, Low, Informational, unknown
Issue Name	The name assigned to the CI/CD pipeline risk issue. Corresponds to the CI/CD rule that detected the risk
Category	The type of issue. Values: Code, Configuration

Property	Description
Description	A description of the issue
Finding ID	The identifier of the finding on which the issue is based
Provider	The version control system (such as GitHub) or CI tool (such as GitHub Actions) hosting the CI/CD pipeline in which the issue was detected
Asset Name	The name of the asset in which the issue was detected
Asset ID	The identifier of the asset in which the issue was detected
Asset Category	The category of the asset (such as a repository, CI/CD Pipeline) in which the issue was detected
Status	The status of the issue. Values: New, Resolved, Under Investigation
Domain	Fixed value: Posture
Last Updated	The most recent scan that detected the finding which generated the issue
Backlog Status	Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner

5.4.2 | Expanded CI/CD risks issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- **Timestamp:** When the issue was created and last updated
- Status: The issue status. Values: New, In Progress, Resolved. You can set the status as required
- Assignee: The entity assigned to mitigate the issue. You can assign the issue from the menu in the field
- Description: A description of the risk and the impact that the issue could potentially have on your SDLC
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the IaC resource was identified)
- Evidence: Provides evidence and contextual details within your SDLC containing the IaC misconfiguration issue:
 - **Issue source**
 - Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Category: The scanner category. Configuration is the immutable value
 - AppSec Rule: The security rule that detected this issue. Includes a link to the rule
 - AppSec Policy: The violated security standard that lead to the creation of the issue. Includes a link to the policy
 - Collaborator: The individual or team responsible for contributing to the code or configuration where the issue was identified
 - **Code context**
 - Scanner Type: AppSec CI/CD Risk Scanner is the immutable scanner type
 - Scanner Source: Cortex AppSec is the immutable scanner source
 - Repository Name: The name of the version control repository where the issue was located
 - Branch: The specific branch within the repository containing the issue
 - Framework: The infrastructure as code (IaC) framework used (such as CloudFormation, Terraform)
 - File Path: The exact location of the issue within the repository file structure
 - First Hash: The commit hash of the first commit where this specific issue was introduced or detected
 - Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
 - Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
 - Remediation: Suggested steps to remediate the issue

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

Actions

Provides suggested solutions. No automated solutions are available for CI/CD risk issues.

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

5.4.3 | VCS and CI/CD pipeline risk findings

VCS and CI/CD pipeline scans produce findings, which are potential security risks in your VCS repositories and CI/CD pipeline configurations. These insights help assess and analyze the security posture of your VCS's and CI/CD pipelines.

The CI/CD risks Findings table is a filtered instance of the broader Findings table found under Cases & Issues, meaning it exclusively displays findings categorized as CI/CD pipeline risk findings. However, CI/CD pipeline risk Findings only displays findings detected during periodic scans. In contrast, the comprehensive Findings table includes all CI/CD pipeline risk findings regardless of their detection source, such as periodic, pull request (PR), and continuous integration (CI) scans.

The following table describes selected properties of the Findings table.

Property	Description
Name	The name of the finding
Created	When the finding was initially detected
Last Updated	The last detection date of the finding
Provider	The VCS including the CI/cD pipeline

Property	Description
Sub Category	<p>The CI/CD category that the findings belongs to. Values include:</p> <ul style="list-style-type: none"> • 3rd Party Services • Artifact Integrity Validation • Credential Hygiene • Data Protection • Dependency Chains • Identity & Access Management • Input Validation • Flow Control Mechanisms • Pipeline-Based Access Controls (PBAC) • Poisoned Pipeline Execution (PPE) • System Configuration
Detection Method	The engine used to detect VCS and CI/CD findings. Default value: CI/CD Security
Finding ID	The unique identifier assigned to the finding

Expanded Findings details

Click on a finding in the inventory table to open the Findings side card, which provides additional details about the finding.

- **Finding summary:** Found at the top of the card. Includes the finding name, ID and type (Configuration for CI/CD risk findings)
- Description: A description of the finding including its location
- **Timestamp:** When the finding was last updated
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the CI/CD risk was identified)
- Evidence: Provides evidence and contextual details within your SDLC containing the CI/CD risk finding:
 - **Finding source**
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected
 - Collaborator: The individual or team responsible for contributing to the code or configuration where the finding was identified
 - **Code context**
 - Repository: The name of the version control repository where the finding was located
 - Branch: The specific branch within the repository containing the finding
 - File Path: The exact location of the finding within the repository file structure
 - First Hash: The commit hash of the first commit where this specific finding was introduced or detected
 - **Scan metadata**
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected
- Code: The file and code including the CI/CD risk in which the finding was detected

5.5 | CI/CD Rules

Abstract

CI/CD rules detect security threats within your pipelines.

CI/CD rules are designed to detect security threats within your application security environment, which includes the various components, configurations, and interactions within your application that can potentially introduce vulnerabilities or pose risks to its security. CI/CD rules identify and flag issues based on predefined criteria, ensuring that potential threats are proactively detected and addressed to enhance the overall security posture of your application.

CI/CD rules cover a wide range of security best practices, inspired by compliance frameworks such as OWASP top 10 CI/CD Risks, as well as additional best practices beyond regulatory requirements.

NOTE:

- Out-of-the-box rules cannot be modified
- Custom CI/CD rules are not supported

5.5.1 | CI/CD rules roles and permissions

These user roles and permissions are required for CI/CD rules:

- Dedicated AppSec Admins and DevSecOps user roles only have **View** permissions
- Roles with Rules Read permission can view detection rules

5.5.2 | CI/CD rules inventory

The CI/CD rules inventory includes both out-of-the-box and custom rules.

To access the inventory:

1. Under Modules, select Application Security → AppSec Rules (under Policy Management).
2. In the Filter panel, select Scanner → CI/CD Security.

The following list lists the exposed AppSec Rules table properties. Additional settings are found under Menu settings.

Field/Property	Description
Severity	The priority level assigned to findings identified by the rule
Rule Name	The rule name
Rule Description	A description of the rule
Framework/Language	The framework or language that the detection rule applies to (for example, GitHub, Terraform, JavaScript)
Labels	Labels assigned to the rule

Field/Property	Description
Policies Count	The amount of policies that include the rule. Selecting the count redirects to the AppSec Policies page, sorted by the policies associated with the rule
Issues Count	The amount of issues detected by the rule. Selecting the count redirects to the CI/CD Risks page, sorted by the issues detected by the rule
Scanner	CI/CD Security. This value is immutable
Last modified	The date and time when the rule was most recently updated
Mapped Cloud Security Rule	The specific runtime cloud security policy that correlates with this AppSec rule. This mapping enables you to trace a security issue from its definition in code to its manifestation in the live cloud environment, ensuring end-to-end visibility

5.6 | CI/CD Policies

Abstract

Control CI/CD security: view system & custom policies. Create and manage policies to ensure pipeline integrity and compliance.

CI/CD policies define how a system should respond to threats in pipelines. It includes conditions that trigger the policy, the scope of its application, and the actions to be taken when these conditions are met. When a policy detects a threat, it generates an issue for remediation.

Cortex Cloud provides out-of-the-box CI/CD policies. In addition, you can create custom policies to tailor it to your specific business or infrastructure requirements. Out-of-the-box policies cannot be modified directly. However, you can create a custom policy by cloning the existing one. This allows you to make changes to the original policy according to your requirements. Refer to Manage CI/CD policies for more information.

TIP:

For Cortex Cloud Code policies, refer to Application Security Policies.

5.6.1 | CI/CD policies user roles and permissions

These user roles and permissions are required for CI/CD policies:

- Roles with Policies View/Edit permissions can create and modify detection policies
- Roles with Policies Read permissions can view detection policies
- An AppSec Admins user role has View/Edit permissions
- A DevSecOps user role only has View permissions

5.6.2 | CI/CD policies inventory

The CI/CD policies inventory includes both out-of-the-box and custom policies.

To access the inventory:

1. Under Modules, select Cortex Cloud Application Security → AppSec Policies (under Policy Management).
2. In the filter panel, select Scan Type → CI/CD Risk.

The following list describes the policy fields/properties exposed in the inventory table. Select Table Settings Menu to view and add additional properties to the table.

Properties/Attribute	Description
Policy Name	The name of the CI/CD policy
Status	Whether the policy is enabled or disable
Description	A description of the CI/CD policy
Scan Type	CI/CD Risks is the immutable value
Conditions	The specific criteria that trigger the policy
Actions	Actions to take when the policy detects its target risk
Scope	The assets to be evaluated by the policy

Properties/Attribute	Description
Trigger	Trigger types that define when the condition will be evaluated. Options include Periodic scan, Pull Request scan and CI scan
Last Triggered	The last time that the policy was triggered
Created By	The user or entity that created the policy
Modified by	The user or entity that modified the policy
Modification Time	The timestamp of the most recent change to the policy
Open Issues	The amount of issues detected by the policy that remain unresolved

Expanded policy details

Selecting a policy opens a side panel where you can review additional details:

- **Metadata:**
 - **Policy details:** Name and description of the policy
 - **Policy ownership:** Information on the policy's creator and last modifier
- NOTE:**

To view all out-of-the-box (OOTB) policies, filter by **Policy Owner = System**.
- **Timestamps:** The last time the policy was modified and last triggered
- **Scope:** The asset type the policy applies to, along with a table summarizing the policy conditions, trigger, and actions, displayed as follows:
 - When: The trigger that initiates the policy action, such as Periodic, Pull Request, or CI scans
 - If: Conditions that are applied to the policy. For example: (**Finding Type = CI/CD Risks**) AND (**Severity = Critical**)
 - Then: Triggered actions for the policy, such as Create issue and Block PR

5.6.3 | Create custom CI/CD policies

Create custom CI/CD policies to enable tailored security checks across your pipelines.

1. Under Modules, select Application Security → AppSec Policies (under Policy Management).
2. Click Add Policy on the Policies page.
3. Provide a policy name (required) and description on the General step of the wizard that is displayed, and click Next.
4. Define the policy conditions on the Define Conditions step of the wizard.
 - a. Define the criteria for your policy to evaluate.
 - Trigger (Required): For CI/CD policies, the only supported trigger type is Periodic scan.

NOTE:

PR Scan and CI Scan triggers are automatically disabled and unchecked. They can only be enabled if other scan types (that is non-CI/CD risk scans such as Secrets) are also selected, and will only run on non-CI/CD risks types of scans

- Conditions (Required). Configure conditions for CI/CD risks by selecting CI/CD Risks as the Finding Type: Select Add Filter → Finding Type → CI/CD Risks.

NOTE:

You can combine multiple conditions to create complex rules for when the policy should apply.

Example 4. EXAMPLE

Create conditions that apply to a CI/CD policy which detects high severity CI/CD risks on GitHub: Select Add Filter → Finding Type → CI/CD Risks → AND → Provider → [VCS/CI/CD system] → AND → Severity: → High.

For more information about CI/CD Risks as the finding type, refer to Finding Type attribute for CI/CD Risks below.

- Developer Suppressions: A developer suppression is an inline comment added during code development to exclude specific findings from being evaluated by a policy. You can choose to either apply or skip these suppressions
 - Apply: When you select this option, a new condition is automatically added to each existing condition group, ensuring the policy respects developer suppressions
 - Skip: This option causes the policy to evaluate all findings, even those suppressed by a developer

b. Click Next.

5. Define the policy scope on the Define Scope step of the wizard.

- a. Define the scope of the assets to be evaluated by the policy using one of these methods. Options include Asset Groups and Matching Criteria.

- Asset Groups: Select the asset groups on which this policy and its chosen detection rules will be evaluated. You can only select asset groups that are assigned to you as part of your scope

For more information about Cortex Cloud Application Security Asset Groups, refer to SBAC Scope-based access control for Cortex Cloud Application Security

- Matching Criteria: When selected, displays a list of assets. You can define the policy's scope by using Matching Criteria, which allows you to build filters using the query builder. Only those assets that satisfy these criteria will be included in the policy's scope

NOTE:

- SBAC scope limitations do not apply to Matching Criteria
- For CI/CD policies, you can filter Matching Criteria by VCS Organization Name, CI/CD Pipeline [Name/ID], CI/CD Instance [Name/ID] or Collaborator [Name/Email/MFA Enabled/Last Observed]

b. Click Next.

6. Periodic Scan trigger: This trigger is automatically enabled and locked (checked and disabled from being unchecked), as CI/CD Risks scans only run on Periodic scans

Define **actions** to be taken when a policy is triggered on the Define Action step of the wizard.

a. Specify which actions to take when the policy detects its target risk:

- Create a new issue: Create an issue if policy conditions within the selected scope are met. This is the only available action for CI/CD Risks policies
- Block:
 - Block a CI run if the policy conditions within the selected scope are met. Available when CI scan is selected as the evaluation method
 - Block a build pull request (PR) if the policy conditions within the selected scope are met. Available when CI scan is selected as the evaluation method
- Report:
 - Enable reporting via CLI if policy conditions within the selected scope are met. Available when CI scan is selected as the evaluation method
 - Enable reporting of a pull request (PR) comment if policy conditions within the selected scope are met. Available when PR scan is selected as the evaluation method

b. Click Create.

The policy is created. You are redirected to the Policies page, which displays the newly created policy.

Finding Type attribute for CI/CD Risks

When CI/CD Risks is selected as the finding type, the following apply:

- Periodic Scan trigger: This trigger is automatically enabled and locked (checked and disabled from being unchecked), as CI/CD Risks scans only run on Periodic scans
- PR Scan and CI Scan triggers: These triggers are automatically disabled and unchecked. They can only be enabled if other scan types (that is non-CI/CD risk scans such as Secrets) are also selected, and will only run on non-CI/CD risks types of scans
- CI/CD risks attributes include Severity, AppSec Rule, AppSec Rule Label, Backlog Status, Subcategory, Provider (such as GitHub) and Finding Type (which enables you to enlarge the policy scope to include additional scanner types)

For more information about Backlog Status, refer to Backlog baseline.

5.6.4 | Manage CI/CD policies

Manage your custom CI/CD policies to maintain an effective application security posture and adapt your security rules to evolving threats and requirements.

To manage policies, right-click on a policy in the table or select a policy and then select the menu in the side panel. The following actions are available:

- Edit policy: Redirects to the policy wizard, allowing you to modify the policy

NOTE:

You cannot edit out-of-the-box (OOTB) policies.

- Duplicate policy: Clone OOTB policies as templates for creating custom policies. When this option is selected, the policy wizard is displayed with the original policy configurations, allowing you to modify them as required

NOTE:

The duplicated policy will include the word "clone" in its name and must be renamed.

- Disable policy: Deactivate the policy without deleting it. Future scans will not trigger the policy, but existing issues detected by the policy will persist. Bulk actions are supported, allowing you to disable multiple policies simultaneously
- Delete policy: Permanently remove the policy from your environment. Issues detected by the policy will persist. Bulk deletions are supported

6 | Code Security

Abstract

Code Security identifies and mitigates vulnerabilities, secrets, and IaC misconfigurations, shifting security left to protect data and maintain trust.

Code Security provides automated, native scanning tools designed to enhance the security posture of your applications and infrastructure. Integrating these scanners into your development workflow facilitates a shift-

left approach to security, enabling the proactive identification and remediation of security issues early in your software development lifecycle (SDLC), preventing them from reaching production environments.

Findings generated by native scanners are evaluated and elevated to actionable issues based on Cortex Cloud rules and policies. Code Security issues are displayed in dedicated issue tables, organized by the native Code Security scanner type (Secrets, SCA and IaC misconfigurations), providing security teams with focused, manageable views.

These tables are filtered instances of the main Issues table located under Cases & Issues. They only display issues generated from findings detected during periodic scans. In contrast, the comprehensive Issues table includes all issues, regardless of their detection source, such as periodic, pull request (PR), and continuous integration (CI) scans.

Code Security use cases

- **Prevent production incidents:** By integrating security scans into CI/CD pipelines, organizations can block insecure code from being merged into production, preventing critical vulnerabilities from reaching live environments
- **Ensure compliance and reduce risk:** Automated scanning for known vulnerabilities (CVEs) and open-source license risks helps organizations adhere to regulatory requirements and minimize legal and compliance exposure
- **Improve developer efficiency and productivity:** Providing immediate feedback on security issues directly within the IDE allows developers to fix problems as they code, reducing rework and accelerating secure development
- **Gain comprehensive visibility into software assets:** Building a complete inventory of open-source packages and assessing your operational risk to help understand your software supply chain and proactively manage potential threats from outdated or abandoned components
- **Streamline security operations:** Centralized scan management, customizable policies, and API integrations enable security teams to automate workflows, enforce security standards, and gain a holistic view of your Application Security posture
- **Enhance traceability and remediation:** Mapping security rules from code to cloud environments allows for end-to-end traceability of issues, accelerating the identification of root causes and prioritization of remediation efforts

Code Security key features

- Comprehensive code scanning:
 - Software Composition Analysis (SCA):
 - **Vulnerability scanning:** Identifies known vulnerabilities (CVEs) in open-source dependencies
 - **License scanning:** Manages open-source license obligations and identifies compliance risks
 - **Package Operational Risk scanning:** Assesses the popularity, maintenance status, and potential long-term risks of open-source packages (such as deprecation, maintainer activity)
 - **Secrets scanning:** Detects hardcoded credentials, API keys, and other sensitive information within source code
 - **Infrastructure as Code (IaC) scanning:** Identifies misconfigurations and security flaws in IaC templates (such as Terraform, CloudFormation)
- Seamless integration and shift-left capabilities:
 - **Version control system (VCS) and CI tool onboarding:** Integrates with popular VCS (such as GitHub, GitLab, Bitbucket) and CI tools (such as Jenkins or GitHub Actions) for automated scanning
 - **CLI scans**
 - **Integration with CI tools:** Allows for easy implementation in CI pipelines by adding a snippet, enabling blocking of pipelines based on security policies
 - **Dedicated Application Security CLI module** (as part of Cortex Cloud unified CLI (which includes Image, API, and code scanning)), enabling local scans or pipeline steps to detect IaC, secrets, vulnerabilities, license, and package integrity findings
 - **IDE support** (JetBrains, VS Code): Plugins for popular Integrated Development Environments (IDEs) to enable shift-left security, allowing developers to detect and fix issues (IaC, Secrets, CVE, License, Package Integrity (Operational Risk)) directly as they write code, including local fixes and suppression options
- Centralized scan management:
 - **Review of all scan types:** Provides a unified view for periodic, Pull Request (PR), and Continuous Integration (CI) scans
 - **Detailed scan insights:** Overview tabs for scan details and health, along with dedicated tabs for each finding type (Vulnerabilities, Configurations, Secrets, Package Integrity)
 - **Issue/Finding card access:** Allows you to drill down into individual issues from the scan management page
 - **Reporting and blocking:** Configurable policies to report findings or block pipelines based on predefined Application Security policies
- Out-of-the-box and customizable rules and policies:

- Out-of-the-box rules and policies, as well as the ability for custom creation
- Customization of conditions and scope for pipeline blocking
- Custom code to cloud rules mapping: Provides the ability to map custom Application Security IaC rules to custom runtime CSPM (Cloud Security Posture Management) rules, improving issue traceability and accelerating remediation by linking code-level issues to cloud runtime anomalies
- Secure communication and APIs:
 - Transporter: A secure communication channel between your SDLC environment and Cortex Cloud
 - Application Security APIs: Public APIs for scan management, policies, and applications, enabling enhanced automation, integration capabilities, and granular control with enforced user permissions

Related topics

- Application Security Policies
- Application Security Rules
- Application Security scans management
- 3rd party scanner findings and generated issues

6.1 | Code Security user roles and permissions

Cortex Cloud offers dedicated predefined user roles for Code Security: AppSec Admin, DevSecOps and Developer, each with specific areas of responsibilities. When assigning roles to users, it's recommended to align them with the user's required responsibilities within the application security framework.

The AppSec Admin has full permissions for all Cortex Cloud Application Security-related activities. They can create and modify detection rules within the Code/Build domain, track progress, and adjust enforcements as needed. Additionally, they can triage and investigate findings, issues, and cases spanning from code to cloud. The role also includes complete visibility into all cloud assets.

The DevSecOps role is specifically designed as an intermediary, possessing more permissions than a Developer but fewer than an AppSec Admin. This role actively manages security processes and tools to embed security directly into development and operations workflows. Responsibilities include managing and resolving security issues, performing scan management, and improving the overall application security posture by integrating security practices throughout the development and operations lifecycle.

Developers within the Code Security environment have limited permissions primarily focused on viewing and monitoring security information. They can access and analyze scan results, track progress, and collaborate with security teams. However, they typically do not have the authority to modify detection rules, enforcements, or directly address security issues.

Permissions defined in the predefined roles cannot be changed. However, you can save a predefined role as a new custom role. This custom role can then be edited to meet specific organizational needs, offering a balance between standardized roles and customizable access control.

Dedicated Code Security roles include permissions that extend beyond Code Security itself. In addition to these dedicated users, other roles within your tenant are also granted specific permissions to Code Security.

You can view all permissions granted to user roles in your tenant by navigating to Settings → Configurations → Roles → select a role.

6.2 | Code Security assets

Code Security provides a comprehensive view of assets detected by native **Infrastructure-as-Code (IaC)** and **Software Composition Analysis (SCA)** scanners. These assets are displayed in dedicated IaC Resources and Software Package inventories. Both are specialized, filtered views within the broader All Assets inventory. This focused approach allows you to manage and analyze your **Code Security** assets separately from other types in your environment.

Code Security assets are a part of the broader Cortex Cloud Application Security asset suite, including:

- **ASPM** assets, which include Repositories. For more information about Repositories, refer to Repositories as assets.
- **CI/CD Security** assets, which include VCS Organizations, CI/CD Instances and CI/CD Pipelines. For more information about **CI/CD Security** assets, refer to CI/CD assets.
- Identity assets, which include Collaborators. For more information about Collaborators as assets, refer to VCS Collaborators-as-assets.

Code Security asset use cases

- **Visibility and context**
 - **Asset inventory:** Maintain a comprehensive inventory of all detected assets, including their metadata and relationships to other assets. This provides a centralized view of all components within the environment
 - **Code to cloud mapping:** A graphical representation of the SDLC, highlighting the asset's location within. This visualization allows for a clear understanding of the asset's journey and its relationship to other components
 - **Application path to production:** Trace the asset's path through the application lifecycle, from its origin in code repositories to its deployment. This includes identifying all intermediate stages and dependencies
- **Security risks**
 - **Infrastructure as Code (IaC) misconfigurations:** Identify misconfigurations associated with the IaC asset configuration, and provide details such as severity, location, when created and the assignee
 - **SCA CVE vulnerabilities:** Identify known vulnerabilities in open-source packages associated with an asset, including details such as severity, the CVE issue, CVSS score, when discovered and the assignee
 - **License miscompliance:** Identify and detail license miscompliance issues within packages associated with an asset, including severity, license category (such as strong copyleft), location, when discovered, and the assignee
 - **Package Integrity:** Identify and detail any package operational issues in packages associated with the asset, including severity, location, when created and the assignee

6.3 | Software packages as assets

The Software Packages asset inventory provides a centralized view of all open source software packages and their details across your environments. The platform enables efficient tracking and management of your software package assets, ensuring compliance with security and governance standards. You can directly access package vulnerabilities, operational risks and license misconfiguration cases, issues, and findings within the Software Packages asset inventory, allowing you to prioritize and remediate them without having to navigate to a separate remediation section.

6.3.1 | Explore software package assets

To access software package assets, under Inventory, select All Assets → Software Packages (under Code).

The Software Packages assets page includes a dashboard and an inventory.

Software package dashboard

The dashboard includes two widgets:

- Package Managers: A breakdown showing the package managers (such as npm and pip) in your environment, and the number of software packages found in each package manager
- Dependency Types: A breakdown showing the amount of direct and transitive (indirect) software packages

Selecting an item in either widget filters the software package asset inventory accordingly.

Software package asset inventory

The software package inventory table includes the following exposed attributes. Use the Table Settings Menu to view additional properties.

Property	Description
Name	The specific identifier given to a software package
Version	The specific version or release of a software package
Licenses	The legal permissions or rights granted for the use of the software package. For more information about licenses, refer to Open-source software license categories
Operational Risk	The operational risk score assigned to the vulnerability. Refer to Package Integrity for more information about operational risks

Property	Description
Package Manager	Specifies the package manager (such as pip, npm, Maven) that installed and manages the software package and its dependencies
Dependency Type	Whether the package is a direct (a package that is explicitly listed as a requirement) or transitive (a package that is indirectly required by another package) dependency
Organization	The entity responsible for publishing or maintaining the software package
Repository	The repository containing the package's source code. Includes a link which opens the repository asset side card without having to navigate away from the Software Packages page
Provider	The platform (such as GitHub) hosting and supplying the software package
Branch	The scanned branch in which the vulnerability was detected
File Path	The location of the software package file in your environment
Business Application Names	The Business Applications using the software package. See Defining Business Applications for more information about Business Applications
First Observed	The initial sighting or detection of the software package
Last Observed	The last sighting or detection of the software package

6.3.2 | In-depth software package asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Code to Cloud tab (providing context on the asset's path to production), an Applications tab (displaying the applications associated with this asset), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

Software package summary

The software package asset summary, displayed at the top of the card, provides concise details about the asset's key attributes, including the package name and originating package manager.

Overview

The Overview tab summarizes software package package highlights and properties.

Highlights include:

- **Critical/High issues:** An aggregation of critical and high vulnerability issues associated with the package. If the package has a single issue, selecting it opens the issue side panel in place; if multiple issues exist, you're redirected to the main Issues table in a new tab, displaying only the issues associated with that package
- **New:** Indicates whether the package was first detected in your environment during the past 30 days
- **Root:** Indicates whether this package is the top-level package within its dependency tree, meaning it is not a dependency of any other package within the project
- **Deprecated:** Whether the package was officially deprecated by its maintainers. This indicates that it is no longer recommended for use and could potentially include security risks
- **Risk summary:** The amount of vulnerabilities associated with the package grouped by category (cases, issues and findings) and their severity. For more information about OSS package vulnerabilities, refer to Overview
- **Visibility timeline:** When the package vulnerabilities were first and last detected

Properties:

- Asset details, including Asset Id, Asset Types and Asset Groups associated with the package
- **Applications:** Lists the applications that include this package as part of their defined assets or configurations. See Application below for more details
- **Package source:** Includes details about the package origin, including its provider (such as PyPI, npm), the repository and branch hosting the package source code, the path to its primary file, the license under which it's distributed, its dependency type (direct or transitive), and a list of contributors
- Scan information: A list of Software Composition Analysis (SCA) scans conducted on the package. Scan details include the name of the scanner used, the specific branch of the package that was scanned, the timestamp of the last scan, and the overall status of the scan (such as completed or failed).

For more information about scan management, refer to Overview.

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Code

The Code tab identifies the package's location within your codebase by providing the repository, file path, and specific line number. Additionally, it presents the package's dependency tree, viewable as either a graph or a hierarchical list, presenting its relationships with other components.

Code to Cloud

The Code to Cloud tab visually represents the software development lifecycle (SDLC), focusing on the package's role in the path to production. The graph describes the links between the repository node hosting the package, the pipeline, image, and cluster.

For more information on Cortex Cloud Code to Cloud, refer to [Code to Cloud](#).

Applications

The Applications tab provides an overview of the application associated with this package, including a graphical representation of its path to production, which incorporates the package's role within the workflow.

For more information about applications, refer to [Applications](#).

Vulnerabilities

The Vulnerabilities tab provides a list of vulnerabilities identified within the package in your environment. Each vulnerability includes details regarding its severity level, associated CVE identifier, CVSS score, initial detection date, and assigned team member or group responsible for remediation.

The table includes the following default properties. Click on the Table Settings Menu for additional properties.

- Severity: The vulnerability severity level
- Issue Name: The CVE identifier
- Asset Name: The asset in which the vulnerability was detected. Selecting this attribute displays the asset side card without having to navigate away from the Repository page
- Branch: The branch in which the vulnerability was detected
- CVSS Score: The Common Vulnerability Scoring System score that quantifies the severity of the vulnerability
- Assigned To: The person or team responsible for addressing the vulnerability
- Dependency Type: Indicates whether the dependency is direct (explicitly declared in your project) or transitive (pulled in by one of your other dependencies)
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability)
- Creation Date: The date when the vulnerability was detected

For more information about SCA vulnerabilities, Software Composition Analysis (SCA) vulnerability issues

Package Integrity

The Package Inventory tab provides details about the popularity and maintenance of packages identified within your SDLC. It also includes an inventory of package operational risk issues and package license issues, offering a comprehensive view of the package's overall health and compliance.

The License Issue table includes the following properties. Click on the Table Settings Menu for additional properties.

- **Severity** level: Indicates the level of severity of the package license miscompliance
- Issue Name: The package license miscompliance identifier
- License Name: The name of the license associated with the package. This indicates the specific license agreement that is potentially being violated
- Asset Name: The name of the asset that uses the package with the license miscompliance. This identifies where the license issue occurs
- Branch: The branch of the codebase where the asset with the license issue is located

The Operational Risk Issues table includes the following properties. Click on the Table Settings Menu for additional properties.

- Severity: Indicates the level of severity of the package operational risk
- Issue Name: The package operational risk identifier
- Asset Name: The name of the asset that uses the package with the package operational risk
- Branch: The branch of the codebase where the asset with the package operational risk is located
- Assigned To: The person or team responsible for addressing the package operational risk
- Creation Date: The date when the package operational risk was initially detected

For more information on Package Operational Risks, refer to [Package Integrity](#).

6.4 | Infrastructure-as-Code (IaC) resources as assets

The IaC resource asset inventory provides a centralized view of all infrastructure-as-code (IaC) resources and their details across your environments. The platform enables efficient tracking and management of your IaC resources, ensuring compliance with security and governance standards. You can directly access IaC misconfiguration issues and findings within the IaC assets inventory, allowing you to prioritize and remediate them without having to navigate to a separate remediation section.

6.4.1 | Explore IaC assets

To access IaC assets, under Inventory, select All Assets → Code → IaC Resources.

The IaC Resources assets page includes a dashboard and an inventory.

IaC resources dashboard

The dashboard includes three widgets. To focus the IaC asset inventory on a specific set of resources, select a value in a widget and then choose Filter in, or Filter out to exclude a specific resource from the results.

- Cloud Providers: Displays the total amount of IaC resources categorized by connected cloud providers (such as AWS and GCP) and the number of IaC resources found in each provider.
- Frameworks: Displays connected frameworks (such as Terraform and Kubernetes) and the number of IaC resources found in each framework
- Drifted Resources: Shows the total number of IaC resources with detected drift, broken down by cloud provider. Each provider displays its own drift count.

Asset inventory filters

IaC Asset Inventory filters allow you to refine the displayed list of IaC resources based on precise criteria such as framework, provider, resource type and name and so on, enabling focused security investigation.

You can filter specifically by Drift Resources to narrow the inventory to resources with configuration deviations from the IaC source, helping you focus on high-risk, untracked changes.

IaC resource asset inventory

The following table describes selected IaC resource properties of the inventory.

Property	Description
Asset ID	The identifier assigned to the IaC resource
Provider	The version control system containing the IaC resource
Name	The unique identifier for the IaC resource within the system
Resource Type	The type of configurations or artifacts represented in the Infrastructure as Code (IaC) assets, such as Dockerfile, AWS Internet Gateway and AWS EKS Addon
Type	The classification of artifacts or resources within IaC assets, such as Dockerfile resource and Terraform resource
Cloud Provider	The cloud provider where this IaC resource is deployed, such as AWS, Azure, or GCP
Repository	The version control system repository where the IaC code for this resource is stored

Property	Description
File Path	The relative path to the IaC provider file within the repository that defines this resource
End Line	The line number indicating the conclusion or endpoint of the configuration code for a particular IaC resource within the specified file. Example: for <code>/vpc.tf (62–66)</code> , the end line refers to line 66, marking the conclusion of the configuration related to the IaC resource defined in the 'vpc.tf' Terraform file
Framework	The IaC framework used to define and manage this resource, such as Terraform or CloudFormation
Branch	The specific branch of the repository where the IaC asset is located
First Observed	The date and time when this IaC resource was first discovered by the system.
Type Category	Fixed value: IaC Resource

6.4.2 | In-depth IaC resource asset information

Click an asset in the inventory table to open its side card, providing in-depth information organized into several tabs. The Overview tab (default display) offers highlights and a general summary. Additional contextual tabs provide specific details, including a Code to Cloud tab (providing context on the asset's path to production), an Applications tab (displaying the applications associated with this asset), and tabs focusing on specific issue types detected within the asset, such as Secrets and Vulnerabilities.

IaC resource summary

The IaC resource asset summary, displayed at the top of the card, provides concise details about the IaC resource, such as its name (such as `aws_s3_bucket.website`), type (such as S3 bucket), and the associated framework (such as Terraform).

Overview

The Overview tab summarizes IaC resource highlights and properties.

Highlights provide key security and operational insights related to the asset:

- Critical/High issues: An aggregation of critical and high issues associated with the IaC resource
- Internet Exposed Runtime Asset: Indicates whether the IaC resource, when deployed, results in a runtime asset (such as a container), that is directly accessible from the public internet
- Deployed: Indicates whether the IaC resource has been deployed and is currently active within your cloud environment or infrastructure
- New: Indicates whether the IaC resource was created during the past 30 days
- Sensitive Data in Runtime: Indicates whether the IaC resource, when deployed, handles or stores sensitive data within its runtime environment
- **Risk summary:** The amount of cases, issues and findings associated with the IaC resource. Each of these types includes two clickable values. The first value redirects to the corresponding type table sorted by the entry, while the second value opens the description card of the specific type within the IaC resource asset inventory without requiring redirection
- **Visibility timeline:** When the IaC resource was first and last detected

Properties:

- Asset details, including Asset Id, Asset Category and Account ID associated with the IaC resource
- **Source Control Information:** Displays the origin and location of the IaC resource's code. It includes the cloud provider (such as AWS), the provider hosting the code (such as GitHub, GitLab), the linked repository and branch in which the code is hosted, associated tags, and the exact file path to the resource's definition
- **Version History and Metadata:** Provides insights into the resource's development history, authorship, and associated metadata. It lists collaborators who have modified the IaC code, and details about the initial commit hash and time related to this IaC asset
- **Scan information:** A list of scans conducted on the IaC resource. Scan details include the name of the scanner used, the specific branch of the package that was scanned, the timestamp of the last scan, and the overall status of the scan (such as completed or failed)

For more information about scan management, refer to Overview.

The Highlights section and other asset properties only display attributes when their corresponding indicators are present. For example, if an asset is not deployed, its deployment-related attributes will not show up; similarly, if there are no detected issues, those highlights or properties will not appear.

Code

The Code tab describes the directory path and file name where the asset's resource code resides within your IaC repository. In addition, the code defining the resource block is displayed.

Code to Cloud

The Code to Cloud tab describes the integrated flow of a selected Infrastructure as Code (IaC) asset, from development to its deployed state. The graph visualizes the path to production, showing the IaC resource's journey from the repository node where it's hosted, through the CI/CD system, and finally to the traced runtime resource it provisions.

For more information on Cortex Cloud Code to Cloud, refer to Code to Cloud.

Applications

The Applications tab provides an overview of the application associated with this IaC asset, including a graphical representation of its path to production, which incorporates the IaC assets role within the workflow.

For more information about applications, refer to [Applications](#).

Configurations

The Configurations tab displays an inventory of IaC misconfiguration detected in the asset.

The table includes the following default properties. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the IaC misconfiguration
- Issue Name: The IaC misconfiguration identifier
- Asset Name: The name of the IaC resource in which the misconfiguration occurred. Selecting this attribute displays the asset side card without having to navigate away from the Repository page
- Branch: The branch in which the IaC misconfiguration was detected
- Assigned To: The person or team responsible for addressing the issue
- Creation Date: The date when the issue was detected

For more information about IaC misconfiguration, refer to [Infrastructure as Code \(IaC\) misconfiguration scanner](#).

Secrets

The Secrets tab displays an inventory of exposed Secrets in the IaC asset.

The table includes the following exposed properties. Click on the Table Settings Menu for additional properties.

- Severity level: Indicates the level of severity of the exposed Secrets
- Issue Name: The Secrets identifier
- Branch: The branch in which the secret was detected
- Assigned To: The person or team responsible for addressing the Secrets
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability)
- Creation Date: The date when the Secrets were initially detected

For more information about Secrets, refer to [Secrets scanners](#).

6.5 | Code Security scanners

The Code Security module provides automated, native scanning tools designed to enhance the security posture of your applications and infrastructure. Integrating these scanners into your development workflow

facilitates a "shift-left" approach to security, enabling the proactive identification and remediation of vulnerabilities early in your software development lifecycle (SDLC).

The module includes both native scanners and the capability to ingest data from third-party scanners, giving you comprehensive visibility into your security posture.

Code Security scanners include:

- **Software Composition Analysis (SCA) scanners:** Modern applications frequently incorporate numerous open-source and third-party packages. Code Security SCA scanners automate the inspection of these dependencies. They identify known vulnerabilities (tracked as CVEs), assess license compliance to mitigate legal risks, and detect package operational risks such as outdated or unmaintained components. This provides critical insight into your software's complete composition, enabling informed decisions about external code. For more information refer to Software Composition Analysis (SCA) scanners
- **Secrets scanner:** The accidental exposure of sensitive credentials—including API keys, passwords, or tokens—within source code or configuration files represents a significant security risk. The Code Security module secrets scanner detects hardcoded secrets across your repositories and code. Early identification of these exposures prevents unauthorized access and potential data breaches. For more information refer to Secrets scanners
- **Infrastructure as Code (IaC) scanners:** Analyze your infrastructure configuration files prior to deployment to detect misconfigurations, insecure defaults, and compliance violations, thereby preventing the introduction of vulnerabilities into your operational environments. For more information refer to Infrastructure as Code (IaC) misconfiguration scanner

In addition to the native code security scanners, Cortex Cloud Cloud ingests data from third-party scanners. This allows you to consolidate your security findings and manage them all within a single platform for a holistic view of your security posture. For more information about third party scanners, refer to Supported third-party data sources

6.6 | Secrets scanners

Abstract

Cortex Cloud Application Security Secrets scans identify sensitive data embedded in your codebase. This proactive scanning approach ensures that sensitive information is detected and addressed early in the Software Development Lifecycle, significantly reducing the risk of such issues reaching production environments.

Secrets use cases

- **Detect exposed sensitive data:** Identify hardcoded credentials, API keys, passwords, tokens, encryption keys, certificates, and pass-phrases found directly within your source code or configuration files
- **Prevent unauthorized access and data breaches:** By enabling the early identification of these exposures, secrets scans help prevent unauthorized access and potential data breaches that could compromise your infrastructure and applications

Supported file types: Cortex Cloud Application Security scans any plaintext files that are not encrypted, not compressed (for example, not .zip files) and not compiled (for example, not .jar files), for secrets. Additionally, entropy findings look for keywords to lower the noise, and those keywords must be in line with the high entropy string to be flagged.

Entropy Analysis: Cortex Cloud Application Security provides signatures that analyze the randomness of strings within the file. Highly random strings, often referred to as high entropy, can be indicative of a potential secret. To reduce false positives, Cortex Cloud Application Security considers specific keywords that might be associated with secrets alongside the randomness of the data for better accuracy.

You can create policies to automatically detect and prevent the introduction of vulnerable open-source components into your codebase. These policies ensure that all third-party dependencies align with your organization's security standards and risk tolerance, helping you maintain a secure software supply chain. For more information about creating secrets policies, refer to Create Cortex Cloud Application Security policies.

6.6.1 | Secrets issues

Valid secret findings exposed during scans, and classified as Critical or High by Cortex Cloud Application Security policies, are categorized as issues, which represent the smallest unit for remediation in Cortex Cloud. Each issue includes a description, its potential impact, and evidence of the finding that generated the issue in the code.

The Cortex Cloud Application Security Secrets inventory is a pre-filtered view of the comprehensive Cortex Cloud Issues inventory (located at Cases & Issues → Issues). This Secrets inventory displays only issues generated by Cortex Cloud Application Security scanners, while the platform Cortex Cloud Issues inventory unifies all issues from all platform scanners.

Cortex Cloud Application Security provides contextual risk indicators that extend beyond basic severity scoring, providing a clearer view of an issue's exposure and priority. The proprietary Urgency metric dynamically evaluates risk based on factors such as visibility and validation, providing additional context by assessing exploit probability and inherent severity.

Cortex Cloud Application Security provides detailed manual guidance to remediate Secrets issues. You can access these remediation steps directly in the Actions tab of the issue side panel.

How to access Secrets issues

Under Modules, select Application Security → Issues → Secrets.

You can also view Secrets issues in dedicated tabs under other sections when available:

- In Code asset inventories: navigate to Inventory → All Assets → Code:
 - On the Secrets tab under Repositories. Refer to In-depth repository asset information for more information
 - Under the All Code inventory: Select an asset from the table → Secrets
- In Application asset inventories: navigate to Inventory → All Assets → Application → select an option from the Application menu → select an item from the inventory → Secrets
- In Cases and Issues; perform a query. Select Issues → AppSec Issues (under the All Domains menu) → Secrets (as the Detection Method value)

6.6.2 | Secrets issues inventory

The Secrets issues inventory includes the following properties. Use the Table Settings Menu to view additional properties.

Property	Description
Severity	Severity level of the secret exposure (such as Critical, High, Medium, Low)
Name	Name or description of the secret exposure type
Asset Name	Name of the resource or asset where the secret was exposed. Selecting an Asset Name in the table opens the asset's side card, displaying information about the asset, without having to navigate away from the issue page.
Urgency	A context-aware metric to help you focus remediation efforts on the issues that pose the greatest real-world risk in your code. Urgency enrichment highlights risks based on specific, high-impact factors such as deployment status and runtime exposure. For more information about Urgency, refer to Urgency
Risk Factors	Specific conditions that increase the exposure or exploitability of a secret. Options: Found in history, Valid, Privileged
AppSec Policy	The Application Security policy that identified the specific security policy that was violated, which triggered the scanner to generate this issue. Selecting a policy in the table opens the policy side card, displaying a summary of the policy, without having to navigate away from the issue page. For more information about Application Security policies, refer to Application Security Policies
Data Source	The system or integration from which the issue data was originally pulled (such as GitHub, GitLab)

Property	Description
SLA	Application Security SLA defines deadlines for fixing security issues based on severity, ensuring timely remediation and improving team performance. For more information about Application Security SLA, refer to Service Lead Agreements (SLA)
Rule Category	The category associated with the rule that detected the secret
Status	The current state of the issue. Options: New, In Progress, Resolved
File Path	Path to the file or location within the code where the secret was exposed
Assignee	The individual responsible for addressing and resolving the issue
Created	Timestamp of when the secret exposure was first detected
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced vulnerability (New). For more on Backlog status, refer to Backlog baseline
Business Applications	Business applications that includes assets in which the exposed secret was detected
Organization	The organization within the VCS in which the issue was detected
Branch	The specific branch or version of the code where the secret exposure was detected

Property	Description
Repository	The version control repository where secret resides

Secrets validation

You can filter secrets based on their validation status. Options include:

- Valid: The secret has been verified as active and functional
- Invalid: The secret has been verified as no longer active or functional
- Privileged: The secret is valid and provides access to sensitive resources or functions
- No Validation: Validation was not attempted because the secret type or source does not support verification
- Unavailable: Validation could not be performed because the secret source was inaccessible or the required permissions were missing

6.6.3 | Detailed Secrets issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- Issue metadata

- **Timestamp:** When the issue was created and last updated
- **Status:** Displays the current state of the issue. Values: New, In Progress, Resolved. You can set the status manually as required.

Note:

- Status changes are permanent in the current state (no automatic reopening)
- If a previously resolved finding reappears in a new scan, a new issue may be created
- **Resolved status:** The issue is marked as addressed and removed from active management; it no longer affects system metrics
- **Assignee:** You can assign the issue to a person responsible for resolving this issue. Human entities are not supported
- **Rule:** The Cortex Cloud Application Security rule that detected the finding. Selecting the link in the field redirects to the AppSec Rules table, filtered by the selected rule (Only applies to IaC, Secrets and CI/CD rules)
- **Policy:** The violated security standard that led to the detection and creation of the issue. Selecting the link in the field redirects to the AppSec Policies table, filtered by the selected policy
- **SLA (Service Level Agreement):** Indicates the remediation timeline status for security issues. For example Overdue indicates that the issue has exceeded the target remediation timeframe. For more information about SLA, refer to Service Lead Agreements (SLA)
- **Backlog Status:** Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog or new, refer to Issue/Finding classification by scanner
- **Description:** Provides a concise technical summary of the specific security finding detected. It identifies what the issue is by referencing standard identifiers (such as the CVE ID for vulnerabilities or the CWE name for code weaknesses) and explaining the nature of the flaw, misconfiguration, or risk within the asset
- **Impact:** Defines the security or operational consequences of an unresolved or exploited issue. Use this field to translate technical findings into business risk, such as unauthorized access
- **Affected Assets:** The specific asset in which the issue was identified. Clicking on the asset opens the asset side card without needing to navigate away to the asset table
- **Related affected assets:** Assets associated with the primary asset based on the specific scanner used. Examples include package managers (such as pip) for vulnerabilities, IaC frameworks (such as Terraform) for misconfigurations, and third-party detection engines (such as Semgrep) for code weaknesses
- **Linked Cases:** The number and severity of cases associated with this issue. Selecting the link opens the Cases side-card for more information without having to navigate away
- **Evidence:** Provides evidence and contextual details about the issue:

- Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
- Location: The exact technical context of a finding, linking security findings to specific lines of code or infrastructure files:

Examples:

- **Vulnerabilities (SCA)**: Identifies the manifest file path (such as `package-lock.json`) and the specific line where the vulnerable package is declared, providing a declaration snippet as context
- **IaC misconfigurations**: Points to the configuration file (such as `main.tf`) and the resource block's start and end lines. Context includes the full resource configuration and the specific resource identifier
- **Secrets**: Locates the file and line number of the exposure, providing a code snippet with the secret redacted for security
- **Code Weaknesses (SAST)**: References the source code file path and the start/end lines of the flaw. Context displays the vulnerable code snippet and the affected function or method name
- Collaborator: The individual or team responsible for contributing to the code or configuration where the issue was identified
- Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
- Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
- Urgency Details includes:

- **Summary:** The issue's urgency level, a breakdown of its contributing metrics, and the date it was last updated (Tip: Hover over a metric for more information)
- **Urgency context graph:** The Urgency graph provides a node-and-edge visualization that maps the structural relationship between a vulnerable asset and its connected infrastructure. By surfacing the asset hierarchy and deployment paths, the graph offers the context necessary to evaluate the scope and potential blast radius of an issue's urgency across the environment. This visualization allows you to analyze the specific deployment and runtime dependencies where the issue was detected. The urgency level itself is determined by metrics derived from the analysis of the connected assets.

Supported scanners: Vulnerabilities (SCA), Secrets, Code Weaknesses (ingested from third -party vendors) and IaC misconfigurations.

Graph structure:

- **Nodes (assets):** Represents assets such as repository or pipeline, that are linked to the specific asset where the issue was detected.

Clicking on a node opens a side card showing: Clicking on a node opens a side card showing initial details about the asset. Selecting View Details opens the asset side card without navigating away, displaying asset details, asset-specific information, and related context for that asset.

- **Edges (relationships):** The edges in the Urgency graph represent relationships between different asset types in the code-to-cloud deployment pipeline. These relationships trace how code flows from repositories through to runtime environments, providing the structural data required to calculate urgency metrics. By analyzing these relationships, Cortex Cloud determines the urgency level based on metrics such as whether the code is actively deployed, internet-exposed, accessing sensitive data, or leveraging privileged capabilities within its runtime environment.

- **Interactive controls:**

- **Layers control:** Toggle visibility of Code, Build, or Runtime layers to focus on specific pipeline segments
- **Group nodes:** Collapse multiple related nodes (such as assets within the same organization or namespace) into a single group node to manage visual density
- **Search and filter:** Locate specific nodes by typing an asset name or ID. Matching nodes are highlighted and the graph auto-focuses on the results
- **Zoom and pan:** Navigate large topologies using zoom buttons, Fit to View, or drag-and-drop panning

For more information about Urgency levels, refer to [Urgency](#).

- **Remediation:** Suggested steps to mitigate the issue. For the most efficient resolution, use the Actions tab, which provides a complete list of remediation options, including PR fixes where available

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

Includes a manual fix where applicable. Automated fixes are not available for secrets issues.

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

6.6.4 | Secrets findings

Secrets findings display sensitive information, such as credentials or API keys, that may be exposed within your assets. These insights help assess and analyze the potential exposure of secrets in your environment.

The Cortex Cloud Application Security Secrets Findings table is a pre-filtered view of the comprehensive Findings inventory (located at Cases & Issues → Issues → Findings Table tab).

This table exclusively displays findings from the Cortex Cloud Application Security Secrets scanner that were detected during periodic scans. In contrast, the comprehensive Findings table unifies vulnerabilities findings from all sources, including periodic, pull request (PR), and continuous integration (CI) scans.

NOTE:

Findings are informational and, as such, are not directly mitigable. Remediation is performed on issues derived from findings.

To access Secrets findings, select Application Security → Issues → Secrets → click the Findings tab.

Secrets findings inventory

The Secrets Findings inventory includes the following properties. Use the Table Settings Menu to view additional properties.

Property	Description
Name	The name or title of the finding.
Asset Name	Name of the asset affected by the finding. Selecting an Asset Name in the table opens the asset's side card, displaying information about the asset, without having to navigate away from the Findings page.
Risk Factors	Quantifiable attributes of a finding, allowing you to analyze and assess the risk. Options: Found in history, Valid, Privileged

Property	Description
Data Source	Source of the finding information (the version control system)
Rule Category	The category assigned to the rule that detected the finding
Repository	Name of the repository hosting the asset in which the finding was detected
Branch	The branch of code or version control branch where the finding was detected
File Path	The file path or location within the repository where the finding was located
Backlog Status	Backlog Status: Indicates if the finding is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how findings are categorized as backlog/new, refer to Issue/Finding classification by scanner

Expanded Findings details

Clicking on a finding in the inventory table opens the Findings side card which provides additional details.

- **Finding summary:** Found at the top of the card. Includes the finding name, ID and type (Data for Secrets findings)
- Description: A description of the finding including its location
- Impact: The potential security risk the finding poses to your environment
- **Timestamp:** When the finding was last updated
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the secret was identified, , such as **JavaScript Package**)
- Evidence: Provides evidence and contextual details within your software development lifecycle containing the finding:
 - **Finding source**
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected
 - **Code context**
 - Branch: The specific branch within the repository containing the finding
 - File Path: The exact location of the finding within the repository file structure
 - First Hash: The commit hash of the first commit where this specific finding was introduced or detected
 - First Commit Date: The date of the commit that introduced the problematic code or dependency into the repository. This helps understand how long an issue has existed and for prioritizing remediation efforts based on its age
 - Backlog Status: See the inventory table above
 - Collaborator. The entity associated with the finding, contributing to or responsible for the evidence supporting it
 - Code: the code line including the finding
 - **Scan metadata**
 - Rule ID: The ID of the rule that triggered the finding
 - Prioritization Labels: Labels assigned to a finding (such as No Validation), which allow you to prioritize findings for further analysis or mitigation
 - **Manual Fix Suggestion:** Remediation options

6.6.5 | Manage Secrets issues

You can take the following actions to address and manage Secrets exposure issues:

Right-click on a row in the inventory table to access the following actions

- Change Status: Modify the status of the issue. Values: New, In Progress, Resolved
- Change Severity: Modify the severity level of the issue. Values: Critical, High, Medium, Low
- Change Assignee: Change the user or identity assigned to address the issue
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Copy issue URL: Duplicate the URL associated with the issue, to share or reference the issue
- Show/hide rows with the [severity level]: Show/hide rows matching the [severity level] of the selected row

6.7 | Infrastructure as Code (IaC) misconfiguration scanner

Abstract

IaC scanners safeguard cloud infrastructure by identifying misconfigurations before deployment, preventing vulnerabilities in your operational environment.

IaC misconfiguration scanners safeguard your cloud infrastructure by identifying security risks in your infrastructure configurations before they are deployed.

By analyzing your infrastructure configuration files prior to deployment, IaC scanners help prevent the introduction of vulnerabilities into your operational environments.

IaC use cases

- **Proactive detection:** IaC scans identify and flag critical issues such as misconfigurations, insecure defaults, and compliance violations directly within your IaC templates. This includes detecting potential risks in files for various frameworks (see Supported frameworks and languages for supported frameworks)
- **Issue remediation:** All Critical and High IaC misconfiguration findings are categorized as actionable issues. The platform provides both manual and automated fixes to resolve these issues, streamlining remediation efforts. Automated fixes for IaC misconfigurations can modify the configuration directly
- **Integration and developer efficiency:** IaC scanning integrates seamlessly into development workflows. Findings can be detected locally using the Cortex CLI or directly within supported IDEs (Visual Studio Code, JetBrains) via plugins, providing real-time security feedback to developers as they write code. This helps developers fix problems early, reducing rework and accelerating secure development
- **Policy enforcement:** You can create and apply custom policies and rules that define how the system responds to IaC threats, allowing for tailored security checks and automated actions such as blocking CI runs or pull requests based on detected misconfigurations

6.7.1 | Supported frameworks and languages

Cortex Cloud Application Security supports the following infrastructure-as-code (IaC) frameworks.

Ansible	Dockerfile	OpenAI
ARM	Helm	OpenTofu
Bicep	Kubernetes	Terraform
CloudFormation	Kustomize	Terraform Plan

6.7.2 | IaC misconfiguration issues

All Critical and High IaC misconfiguration findings detected in an organization's environment are categorized as issues, which represent the smallest unit for remediating IaC resource misconfigurations. Where applicable, both manual and automated fixes are provided to resolve these issues.

The IaC misconfiguration Issues table is a filtered instance of the broader Issues table found under Cases & Issues, meaning it exclusively displays issues categorized as IaC misconfigurations. The IaC misconfiguration Issues table only displays issues generated from findings detected during periodic scans. In contrast, the comprehensive Issues table includes all IaC misconfiguration issues, regardless of their detection source, such as periodic, pull request (PR), and continuous integration (CI) scans.

How to access IaC misconfiguration issues

To access IaC misconfiguration issues, under Modules, select Application Security → Issues → IaC misconfiguration.

TIP:

You can also view IaC misconfiguration issues in dedicated tabs under other sections when available:

- In code Asset inventories, navigate to Inventory → All Assets → Code:
 - On the Configuration tab under *Repository assets*. Refer to In-depth repository asset information for more information
 - Under the All Code asset inventory: Select an asset from the table → click the Configuration tab
- In Application asset inventories, when an application includes an IaC asset that includes a detected misconfiguration: navigate to Inventory → All Assets → Applications → select an option from the Applications menu → select an application from the inventory → Configurations
- Under Cases and Issues; perform a query. Select Issues → AppSec Issues (under the All Domains menu) → IaC Scanner (as the Detection Method value)

6.7.3 | IaC misconfiguration issues inventory

The IaC misconfiguration issues inventory includes the following properties. Use the menu to view additional fields.

Read more...

Property	Description
Severity	The level of risk or impact associated with the finding, typically categorized (such as Critical, High, Medium, Low).
Name	The specific name or title of the finding, clearly identifying the detected issue
Asset Name	The name of the asset (such as a repository) where the issue was identified
Framework	The Infrastructure as Code (IaC) framework or language used for the asset (such as CloudFormation, Terraform, Azure Resource Manager, Kubernetes)
Repository	The version control repository where the problematic code or configuration resides
Data Source	The system or integration from which the data was originally pulled (such as GitHub, GitLab)
Branch	The specific branch within the repository where the issue was detected
File Path	The exact path to the file within the repository's structure where the issue is located
Status	The current state of the issue (New, In Progress, Resolved)

Property	Description
Assignee	The individual responsible for addressing and resolving the issue
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced misconfiguration (New). For more on Backlog status, refer to Backlog baseline

6.7.4 | Detailed IaC misconfiguration issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- Issue metadata

- **Timestamp:** When the issue was created and last updated
- Status: Displays the current state of the issue. Values: New, In Progress, Resolved. You can set the status manually as required.

Note:

- Status changes are permanent in the current state (no automatic reopening)
- If a previously resolved finding reappears in a new scan, a new issue may be created
- Resolved status: The issue is marked as addressed and removed from active management; it no longer affects system metrics
- Assignee: You can assign the issue to a person responsible for resolving this issue. Human entities are not supported
- Rule: The Cortex Cloud Application Security rule that detected the finding. Selecting the link in the field redirects to the AppSec Rules table, filtered by the selected rule (Only applies to IaC, Secrets and CI/CD rules)
- Policy: The violated security standard that led to the detection and creation of the issue. Selecting the link in the field redirects to the AppSec Policies table, filtered by the selected policy
- SLA (Service Level Agreement): Indicates the remediation timeline status for security issues. For example Overdue indicates that the issue has exceeded the target remediation timeframe. For more information about SLA, refer to Service Lead Agreements (SLA)
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog or new, refer to Issue/Finding classification by scanner
- Description: Provides a concise technical summary of the specific security finding detected. It identifies what the issue is by referencing standard identifiers (such as the CVE ID for vulnerabilities or the CWE name for code weaknesses) and explaining the nature of the flaw, misconfiguration, or risk within the asset
- Impact: Defines the security or operational consequences of an unresolved or exploited issue. Use this field to translate technical findings into business risk, such as unauthorized access
- Affected Assets: The specific asset in which the issue was identified. Clicking on the asset opens the asset side card without needing to navigate away to the asset table
- **Related affected assets:** Assets associated with the primary asset based on the specific scanner used. Examples include package managers (such as pip) for vulnerabilities, IaC frameworks (such as Terraform) for misconfigurations, and third-party detection engines (such as Semgrep) for code weaknesses
- Linked Cases: The number and severity of cases associated with this issue. Selecting the link opens the Cases side-card for more information without having to navigate away
- Evidence: Provides evidence and contextual details about the issue:

- Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
- Location: The exact technical context of a finding, linking security findings to specific lines of code or infrastructure files:

Examples:

- **Vulnerabilities (SCA)**: Identifies the manifest file path (such as `package-lock.json`) and the specific line where the vulnerable package is declared, providing a declaration snippet as context
- **IaC misconfigurations**: Points to the configuration file (such as `main.tf`) and the resource block's start and end lines. Context includes the full resource configuration and the specific resource identifier
- **Secrets**: Locates the file and line number of the exposure, providing a code snippet with the secret redacted for security
- **Code Weaknesses (SAST)**: References the source code file path and the start/end lines of the flaw. Context displays the vulnerable code snippet and the affected function or method name
- Collaborator: The individual or team responsible for contributing to the code or configuration where the issue was identified
- Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
- Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
- Urgency Details includes:

- **Summary:** The issue's urgency level, a breakdown of its contributing metrics, and the date it was last updated (Tip: Hover over a metric for more information)
- **Urgency context graph:** The Urgency graph provides a node-and-edge visualization that maps the structural relationship between a vulnerable asset and its connected infrastructure. By surfacing the asset hierarchy and deployment paths, the graph offers the context necessary to evaluate the scope and potential blast radius of an issue's urgency across the environment. This visualization allows you to analyze the specific deployment and runtime dependencies where the issue was detected. The urgency level itself is determined by metrics derived from the analysis of the connected assets.

Supported scanners: Vulnerabilities (SCA), Secrets, Code Weaknesses (ingested from third -party vendors) and IaC misconfigurations.

Graph structure:

- **Nodes (assets):** Represents assets such as repository or pipeline, that are linked to the specific asset where the issue was detected.

Clicking on a node opens a side card showing: Clicking on a node opens a side card showing initial details about the asset. Selecting View Details opens the asset side card without navigating away, displaying asset details, asset-specific information, and related context for that asset.

- **Edges (relationships):** The edges in the Urgency graph represent relationships between different asset types in the code-to-cloud deployment pipeline. These relationships trace how code flows from repositories through to runtime environments, providing the structural data required to calculate urgency metrics. By analyzing these relationships, Cortex Cloud determines the urgency level based on metrics such as whether the code is actively deployed, internet-exposed, accessing sensitive data, or leveraging privileged capabilities within its runtime environment.

- **Interactive controls:**

- **Layers control:** Toggle visibility of Code, Build, or Runtime layers to focus on specific pipeline segments
- **Group nodes:** Collapse multiple related nodes (such as assets within the same organization or namespace) into a single group node to manage visual density
- **Search and filter:** Locate specific nodes by typing an asset name or ID. Matching nodes are highlighted and the graph auto-focuses on the results
- **Zoom and pan:** Navigate large topologies using zoom buttons, Fit to View, or drag-and-drop panning

For more information about Urgency levels, refer to [Urgency](#).

- **Remediation:** Suggested steps to mitigate the issue. For the most efficient resolution, use the Actions tab, which provides a complete list of remediation options, including PR fixes where available

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

The Code to Cloud tab provides a trace from the IaC misconfiguration issue to its impact on your runtime environment. By correlating static code analysis with live production data, this feature allows security teams to understand the real-world implications of misconfigurations. This tab displays a list of traced runtime issues and findings, offering visibility into which cloud resources currently running in production were created from the IaC code being analyzed.

Scope-Based Access Control (SBAC): The tab respects your access permissions by filtering the data displayed. You will only see cloud assets and findings that belong to asset groups you are authorized to view.

The tab displays the Traced Runtime Issues and Traced Runtime Findings tables. These tables include standard exposed properties, which you can expand by adding additional attributes from the Table Settings Menu.

- The Traced Runtime Issues table includes the following exposed properties.
 - Severity: The level of risk associated with the issue
 - Issue Name: The name of the specific issue detected
 - Assigned to: Indicates the individual responsible for remediating the issue
 - Asset Name: The name of the runtime asset where the issue was detected
 - Detection Method: The method used to identify the issue, such as a specific scanner or security tool
 - Observation Time: The time and date when the issue was first observed in the runtime environment
- The Traced Runtime Findings table includes these exposed properties
 - Finding Name: The specific name of the finding detected on the asset
 - Asset Name: The name of the runtime asset where the finding was detected
 - Detection Method: The method used to identify the finding
 - Observation Time: The time and date when the finding was first observed

Actions

The Actions tab provides remediation and suggested steps to fix the issue:

- **Manual remediation instructions:** Framework-specific instructions (Terraform, CloudFormation, and so on)
- **Automated fix suggestions:** Available for supported misconfiguration types
- **Open Fix Pull Request:** Can create pull requests with configuration fixes

PR Fixes are available when:

- Issue has automated fix suggestion
- Misconfiguration has a known remediation pattern
- Repository integration supports PR creation
- Detection method is the dedicated Cortex Cloud Application Security IaC security scanner

Bulk operations:

- Can fix up to ten issues in a single request
- Issues from the same repository are grouped into one PR
- Each repository gets a separate PR

Prerequisites for fix PRs:

- User must have write permissions to repository
- Repository must be connected via supported integration (GitHub, GitLab, Bitbucket, Azure Repos)

PR Fixes are not available when:

- Complex misconfigurations requiring manual review
- Multi-resource dependencies
- Custom policy violations

Example actions:

- **Manual fix:**
 - Add encryption configuration to S3 bucket
 - Enable logging for RDS instance
 - Configure security group rules
- **Automated fix:**
 - Creates PR with updated IaC configuration
 - Title: Cortex Cloud: Fixing IaC issue (resource-name)

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

6.7.5 | IaC misconfiguration findings

IaC misconfiguration scans produce findings, which are potential security risks in your Infrastructure-as-Code (IaC) definitions. These insights help assess and analyze the security posture of your IaC assets.

The IaC misconfiguration Findings table is a filtered instance of the broader Findings table found under Cases & Issues, meaning it exclusively displays findings categorized as IaC misconfiguration findings.

NOTE:

Findings are informational and, as such, are not directly mitigable. Remediation is performed on issues derived from findings.

To access IaC misconfiguration findings, under Modules, select Application Security → Issues → IaC Misconfigurations → click the Findings tab.

[Findings inventory](#)

The Findings inventory includes the following exposed properties.

[Read more...](#)

Property	Description
Name	The specific name or title of the finding (follows the name of the rule)
Asset Name	The name of the asset (such as a repository) where the finding was identified
Repository	The version control repository where the problematic code or configuration resides
Data Source	The system or integration from which the finding data was originally pulled (such as GitHub, GitLab)
Branch	The specific branch within the repository where the finding was detected
File Path	The exact path to the file within the repository's structure where the finding is located

[Expanded Findings details](#)

Clicking on a finding in the inventory table opens the Findings side card which provides additional details about the finding.

[Read more...](#)

- **Finding summary:** Found at the top of the card. Includes the finding name, ID and type (Configuration for IaC findings)
- Description: A description of the finding including its location
- **Timestamp:** When the finding was last updated
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the IaC resource was identified)
- Evidence: Provides evidence and contextual details within your SDLC containing the IaC misconfiguration finding:
 - **Finding source**
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected
 - Collaborator: The individual or team responsible for contributing to the code or configuration where the finding was identified
 - **Code context**
 - Repository: The name of the version control repository where the finding was located
 - Branch: The specific branch within the repository containing the finding
 - File Path: The exact location of the finding within the repository file structure
 - First Hash: The commit hash of the first commit where this specific finding was introduced or detected
 - First Commit Date: The date of the commit that introduced the problematic code or dependency into the repository. This helps understand how long an issue has existed and for prioritizing remediation efforts based on its age
 - **Scan metadata**
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected

6.7.6 | Manage IaC misconfiguration issues

You can take the following actions to address and manage IaC misconfiguration issues:

Right-click on a row in the inventory table to access the following actions

- Change Status: Modify the status of the issue. Values: New, In Progress, Resolved
- Change Severity: Modify the severity level of the issue. Values: Critical, High, Medium, Low
- Change Assignee: Change the user or identity assigned to address the issue
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Copy issue URL: Duplicate the URL associated with the issue, to share or reference the issue
- Show/hide rows with the [severity level]: Show/hide rows matching the [severity level] of the selected row
- Open Fix PR: Create a Pull Request (PR) with a suggested fix to remediate the issue

6.8 | IaC Drift Detection scans

Abstract

IaC Drift Detection identifies runtime configurations that diverge from code. It flags security-critical discrepancies to focus remediation efforts.

IaC Drift Detection preserves Git as the single source of truth (SSOT) by correlating declared infrastructure templates with live cloud resources using **Code-to-Cloud** lineage. Cortex evaluates drift through a misconfiguration-based detection model, surfacing discrepancies only when a runtime resource violates a security or compliance rule that is not violated in its corresponding IaC definition. This ensures drift findings reflect security-relevant divergence rather than expected operational variance.

Prerequisites

To enable drift detection, ensure your environment meets the following requirements

- **Cloud service provider integration (for CSPM):** The target cloud accounts (AWS, Azure, or GCP) must be successfully onboarded, returning data, and able to read the actual state of your infrastructure from the live environment
- **Version control system (VCS) integration:** The platform must be able to read the intended state of your infrastructure from your code repositories.
 - **Repository integration:** A valid integration with a supported VCS provider (such as GitHub, GitLab, Bitbucket) must be active
 - **Supported formats:** The repository must contain valid Terraform (`.tf`) or CloudFormation (`.yml/.json`) templates
 - **File structure:** The integration must have visibility into the root directory where the IaC templates are located
- **Resource tagging:** Drift detection requires a common identifier to correlate the code block in the VCS with the live resource in the cloud. Use either of the following methods:
 - **Automatically using the Tagging Bot:** Refer to Manage repository scan configurations to enable the Tagging Bot
 - **Manually set-up yor trace tags.** Refer to the Yor documentation for more information
- **Rule mapping (Critical):** Drift is calculated only for IaC rules that are mapped to a corresponding CSPM rule. Rules without this mapping lack a runtime signal and cannot generate drift findings. If a declared resource cannot be resolved to a specific runtime resource via this mapping, drift is not evaluated

6.8.1 | IaC Drift Detection issues

The IaC Drifts issues page provides focused visibility into runtime configuration divergence, allowing you to prioritize remediation efforts efficiently.

IaC drift detection issues are generated when the following conditions are met:

- A deployed cloud resource's runtime configuration diverges from its IaC definition
- The specific divergence violates a security policy

How to access IaC Drift Detection issues

IaC drift detection is managed through the IaC Drifts issues page: Navigate to Modules → Application Security → IaC Drifts.

6.8.2 | IaC Drift Detection issue inventory

The IaC drift detection issues inventory includes these default properties. Use the Table Settings Menu to view additional fields.

Property	Description
Severity	The level of risk associated with the drift finding. Values: Informational, Low , Medium, High, Critical, Unknown
Name	The specific name or title of the finding, clearly identifying the detected security issue resulting from the configuration drift
Code IaC Resource	The name of the IaC resource (code asset) in the repository that corresponds to the drifted Cloud Resource. This is the source of truth that failed to prevent the drift
Cloud Resource	The specific type of the live cloud asset that experienced drift (such as AWS Internet Gateway, Azure Security Group)
Framework	The Infrastructure as Code (IaC) framework or language used to define the original code asset (such as CloudFormation, Terraform)
Repository	The version control repository where the problematic IaC code or configuration files reside
AppSec Policy ID	The ID of the Application Security policy that created the drift issue
Data Source	The version control system from which the original data for the IaC asset was pulled (such as GitHub, GitLab)
Cloud Provider	The cloud service provider hosting the drifted resource (such as AWS, Azure, GCP)

Property	Description
Branch	The specific branch in the version control repository containing the IaC definition
SLA	The deadline or status of the issue remediation timeline based on the organization's Service Level Agreement. For more information, refer to Service Lead Agreements (SLA)
File Path	The full directory path to the specific IaC file within the repository
Status	The current lifecycle state of the drift issue. Values: Approaching, On Track, Overdue
Created	The timestamp when the issue was created
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced misconfiguration (New). For more information, refer to Backlog baseline
Assignee	The individual responsible for addressing and resolving the issue
Business Application	The business application associated with the drifted IaC resource and the corresponding cloud asset. For more information about business applications, refer to Defining Business Applications

6.8.3 | Detailed IaC drift detection issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists

available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- Issue metadata
 - Timestamp: When the issue was created and last updated
 - Status: Displays the current state of the issue. Values: New, In Progress, Resolved. You can set the status as required
 - Severity: The severity level of the issue. You can modify the severity level
 - Assignee: The entity assigned to mitigate the issue. You can assign the issue to a person responsible for resolving this issue
 - AppSec Policy: The violated security standard that led to the detection and creation of the issue. Selecting the link in the field redirects to the AppSec Policies table, filtered by the selected policy. To configure these baselines, refer to Create IaC Drift Detection policies
 - Backlog: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner
- Description: A description of the issue

Example 5. Example

The AWS S3 Bucket Foo has been manually modified and now has drifted configuration from the Terraform resource aws_s3_bucket.bar in the repository panw/xxxx that can cause operational and security risks.

- Impact: The potential risks resulting from the configuration drift

Example 6. Example

Drift can introduce security vulnerabilities and compliance issues if changes occur outside the established Software Development Life Cycle (SDLC) processes. Consequently, addressing IaC drift is critical to ensure secure, efficient, and predictable cloud infrastructure operations.

- Affected Assets: Identifies the resources impacted by the security violation across the Code and Runtime environments
 - Affected Code Asset: The specific IaC resource (the source of truth) in the repository that corresponds to the drifted cloud resource. This asset is critical for understanding the original intended configuration and generating fix PRs.
 - Affected Cloud Asset: Identifies the live runtime cloud resource (the deployed asset) on which the security policy violation was detected. This is the entity currently exposed to risk due to configuration drift.
 - Framework: The specific Infrastructure as Code (IaC) framework used to define the original code asset (such as Terraform, CloudFormation)
- Evidence: Provides evidence and contextual details about the issue:

- Data Source: The system or integration from which the issue data was originally pulled (such as GitHub). Click the link to navigate to the data source itself
- Location: The location of the issue, including the repository, branch, file path, and the IaC resource

Resolution Actions

Remove the discrepancy and restore the cloud asset to its intended IaC-defined configuration. Two required remediation paths are available:

- Align Drift in Code: Use this action when the configuration change in the cloud asset is intentional, required, or approved. The IaC file must be updated to reflect the runtime configuration.
 - Objective: Make the IaC configuration the new source of truth by removing the drift in code
 - Method: Generate a pull request (PR) which updates the original IaC file to match the deployed and compliant configuration, bringing the codebase back in sync with the runtime asset.
 1. Select Align Drift in Code.
 2. Open a Pull Request in the IaC repository that contains the drifted configuration captured in the provided Terraform template.
 3. Run the following commands to sync your local state with the updated configuration:

```
terraform refresh
terraform plan // Reads the actual state from the live cloud environment (which includes the
refresh function) and compares it against the desired IaC state to show the calculated
changes needed to fix the drift.
terraform apply // Executes the computed plan, applying the configuration defined in your
updated IaC code to the live cloud environment
```

- Reconcile Cloud Asset: Use this action when the drift results from an unmanaged or risky modification made directly in the cloud environment. The cloud asset must be reverted to the secure configuration defined in the IaC file.
 - Objective: Remove the security risk by restoring the asset to its last IaC-defined, secure state
 - Method: Re-apply the IaC configuration to the affected resource
 1. Select Reconcile Cloud Asset.
 2. Re-apply IaC configuration to the <affected cloud resource> so that the runtime aligns with the compliant IaC state.
 3. Run:

```
terraform plan --refresh=false
terraform apply
```

6.8.4 | IaC Drift Detection findings

IaC Drift Detection findings are generated when runtime cloud resources do not match the expected configuration defined in their correlated IaC rules. These findings indicate untracked or unmanaged changes in the live environment.

To access IaC drift detection findings, navigate to Modules → Application Security → IaC Drifts (under Issues) → click the Findings tab.

The properties displayed in the IaC Drift Detection Findings table are identical to those displayed in the IaC Drift Detection Issues page. This provides a consistent data structure for security teams transitioning from raw findings to prioritized issues. Refer to IaC Drift Detection issue inventory for more information.

6.9 | Software Composition Analysis (SCA) scanners

Abstract

Cortex Cloud Application Security SCA scanners inspect and manage the security and compliance of your application's open-source and third-party dependencies. They are part of the Cortex Cloud shift-left security strategy, enabling organizations to proactively identify and mitigate risks associated with external code components early in the development lifecycle.

Currently, support for SCA is limited to static analysis, meaning that only direct dependencies are scanned. However, if lock files are present, support is extended to include the analysis of transitive dependencies as well.

SCA use cases

SCA provides a comprehensive approach to securing your software supply chain by enabling you to achieve these objectives:

- **Gain comprehensive visibility into Software Composition:** SCA tools build a complete inventory of all open-source packages and their dependencies, providing critical insight into your software's entire composition. This unified view helps you understand and manage your overall software supply chain. Refer to Software packages as assets for more information
- **Improve application code security and prioritizing remediation:** SCA scans identify critical vulnerabilities and prioritize remediation efforts based on a data-driven risk assessment that combines code-level vulnerabilities and potential business impact
- **Enable informed decisions about external code:** By providing detailed insights into vulnerabilities, licenses, and operational risks, SCA empowers development and security teams to make informed decisions about the use, update, or replacement of external code components

SCA achieves these objectives by performing the following core functions:

- **Identify known vulnerabilities (CVEs):** Automatically detect and flag known security vulnerabilities (tracked as Common Vulnerabilities and Exposures or CVEs) present in your open-source and third-party dependencies. This provides crucial insights into potential weaknesses that could be exploited
- **Assess license compliance:** Manage open-source license obligations and identify potential compliance risks introduced by your application's components. This helps you adhere to regulatory requirements and minimize legal and compliance exposure
- **Detect Package Operational Risks:** Evaluate the operational health and potential long-term risks of open-source packages, such as assessing their popularity, maintenance status (such as outdated or unmaintained components), and deprecation status. This allows you to proactively manage threats from abandoned or insecure components

6.9.1 | Supported Software Composition Analysis (SCA) frameworks and languages

The following table provides the SCA languages supported by Cortex Cloud.

Language	Vulnerabilities	Operational Risk	License Detection	Transitive Dependencies	Risk Detection
JavaScript	Yes	Yes	Yes	Yes	Yes
Python	Yes	Yes	Yes	Yes	Yes
Go	Yes	Yes	Yes	Yes	Yes
Java	Yes	Yes	Yes	Yes	Yes
Kotlin	Yes	Yes	Yes	Yes	Yes
Ruby	Yes	No	No	Yes	No
.NET (C#)	Yes	Yes	Yes	Yes	Yes
PHP	Yes	No	Yes	Yes	No
Rust	Yes	Yes	Yes	Yes	Yes

SCA support matrix by language

The following table provides a high-level overview of Cortex Cloud SCA's language support. Clicking on any listed language will navigate to a dedicated section with more in-depth details about the language.

Language	Package Managers	Manifest Files	Supported Versions
JavaScript	npm	package.json package-lock.json	Lockfile v1, v2, v3
	yarn	package.json yarn.lock	Yarn v1, v2, v3, v4
Python	pip	requirements.txt Pipfile Pipfile.lock	Python v3.9 and above
GO	Go	go.mod go.sum Gopkg.lock	Go v1.11 and above
Java	Maven	pom.xml	Maven v3.x and v4.x
	Gradle	build.gradle settings.gradle	Gradle v4.x through v9.x
Kotlin	Maven	pom.xml	Maven v3.x and v4.x
	Gradle	build.gradle.kts settings.gradle.kts	Gradle v4.x through v9.x

Language	Package Managers	Manifest Files	Supported Versions
Ruby	—	Gemfile Gemfile.lock gemspec	Ruby v2.5.x through v3.4.x
.NET	—	.csproj packages.config project.assets.json packages.lock.json paket.lock	.NET v5.0 through v10.0 .NET Framework v4.6 through v4.8.x .NET Core v2.0 through v3.x
PHP	Composer	composer.json composer.lock	PHP v8.3 and above
Rust	Cargo	Cargo.toml Cargo.lock	Any version of Rust

6.9.2 | Software Composition Analysis (SCA) vulnerability issues

Vulnerability findings detected in software packages with a Critical or High severity level are categorized as SCA issues, which represent the smallest actionable unit for remediations in Cortex Cloud. Each issue includes a description, its potential impact, and evidence of the finding that generated the issue in the code.

The Cortex Cloud Application Security Vulnerabilities inventory is a pre-filtered view of the comprehensive Cortex Cloud Issues inventory (located at Cases & Issues → Issues). This Vulnerabilities inventory displays only issues generated by Cortex Cloud Application Security scanners and ingested third party vulnerability findings, while the platform Cortex Cloud Issues inventory unifies all issues from all platform and third-party scanners.

Cortex Cloud Application Security provides contextual risk indicators that extend beyond basic severity scoring, providing a clearer view of an issue's exposure and priority. The proprietary Urgency metric dynamically evaluates risk based on factors such as deployment context, runtime exposure, and exploit likelihood, while industry-recognized standards such as EPSS and CVSS provide additional context by assessing exploit probability and inherent severity.

Cortex Cloud Application Security provides detailed manual guidance and automated fixes, when available, to remediate issues. You can access these remediation steps directly in the Actions tab of the issue side panel.

How to access CVE vulnerability issues

To access CVE vulnerability issues, under Modules, select Application Security → Issues → Vulnerabilities.

You can also view SCA vulnerabilities in dedicated tabs under other sections:

Read more...

TIP:

- In Asset inventories, navigate to Inventory → All Assets → Code:
 - On the Vulnerabilities tab under Repositories. Refer to In-depth repository asset information for more information
- Under the All Code inventory: Select an asset from the table → Vulnerabilities
- In the Applications inventory: Navigate to Inventory → All Assets → Applications → select an option from the Applications menu → select an application from the inventory → Vulnerabilities
- In Cases and Issues; perform a query. Select Issues → AppSec Issues (under the All Domains menu) → AppSec CVE Scanner (as the Detection Method value)

6.9.2.1 | Vulnerability issues inventory

The CVE vulnerability issues inventory includes the following SCA attributes. Use the Table Settings Menu to view additional properties.

Field/Attribute	Description
Severity	The severity level of the issue. You can change the severity level directly in the table: Right-click on an issue → Change Severity → select a severity level
Name	The unique identifier assigned to the CVE
Asset Name	Name of the asset affected by the issue. Selecting an Asset Name in the table opens the asset's side card, displaying information about the asset, without having to navigate away from the issue page. For more information about software package assets, refer to Software packages as assets

Field/Attribute	Description
Urgency	A context-aware metric to help you focus remediation efforts on the issues that pose the greatest real-world risk in your code. Urgency enrichment highlights risks based on specific, high-impact factors such as deployment status and runtime exposure. For more information about Urgency, refer to Urgency
CVSS Score	Common Vulnerability Scoring System (CVSS) score representing the severity of the vulnerability
EPSS Score	The Exploit Prediction Scoring System (EPSS) provides a data-driven probability score (from 0 to 1, or 0% to 100%) that indicates the likelihood of a given CVE being exploited in the wild within the next 30 days
Repository	Name of the repository where the affected package is hosted
AppSec Policy	The Application Security policy that identified the specific security policy that was violated, which triggered the scanner to generate this issue. Selecting a policy in the table opens the policy side card, displaying a summary of the policy, without having to navigate away from the issue page. For more information about Application Security policies, refer to Application Security Policies
Data Source	The system or integration from which the data was originally pulled (such as GitHub, GitLab)
SLA	Application Security SLA defines deadlines for fixing security issues based on severity, ensuring timely remediation and improving team performance. For more information about Application Security SLA, refer to Service Lead Agreements (SLA)

Field/Attribute	Description
Branch	The specific branch or version of the code where the vulnerability was detected
File Path	Path to the file or location within the code where the vulnerability was detected
Status	The current state of the issue (New, In Progress, Resolved). You can change the issue status directly in the table: Right-click on an issue → Change Status → select a status option. Note that you can also change the status from the Overview card in the issue side panel
Assignee	The individual responsible for addressing and resolving the issue. You can change an assignee directly in the table: Right-click on an issue → Change Assignee → select a user → OK
Scanner	The scanner that detected the issue
Created	Timestamp of when the issue was first detected.
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced vulnerability (New). For more information about Backlog Status, refer to Backlog baseline
Business Application Names	Business applications which includes assets in which the issue was detected
CVE Risk Factors	Attributes (such as Severity, Attack Vector, Attack Complexity, Exploit Exists) indicating the potential severity and likelihood of exploitation for a given CVE

Field/Attribute	Description
Automatic Fix Available	Whether automatic fixes are available to remediate the issue. Values: Yes/No
Manual Fix Available	Whether manual fixes are available to remediate the issue. Values: Yes/No

6.9.2.2 | Detailed vulnerability issue information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- Issue metadata

- **Timestamp:** When the issue was created and last updated
- Status: Displays the current state of the issue. Values: New, In Progress, Resolved. You can set the status manually as required.

Note:

- Status changes are permanent in the current state (no automatic reopening)
- If a previously resolved finding reappears in a new scan, a new issue may be created
- Resolved status: The issue is marked as addressed and removed from active management; it no longer affects system metrics
- Assignee: You can assign the issue to a person responsible for resolving this issue. Human entities are not supported
- Rule: The Cortex Cloud Application Security rule that detected the finding. Selecting the link in the field redirects to the AppSec Rules table, filtered by the selected rule (Only applies to IaC, Secrets and CI/CD rules)
- Policy: The violated security standard that led to the detection and creation of the issue. Selecting the link in the field redirects to the AppSec Policies table, filtered by the selected policy
- SLA (Service Level Agreement): Indicates the remediation timeline status for security issues. For example Overdue indicates that the issue has exceeded the target remediation timeframe. For more information about SLA, refer to Service Lead Agreements (SLA)
- Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog or new, refer to Issue/Finding classification by scanner
- Description: Provides a concise technical summary of the specific security finding detected. It identifies what the issue is by referencing standard identifiers (such as the CVE ID for vulnerabilities or the CWE name for code weaknesses) and explaining the nature of the flaw, misconfiguration, or risk within the asset
- Impact: Defines the security or operational consequences of an unresolved or exploited issue. Use this field to translate technical findings into business risk, such as unauthorized access
- Affected Assets: The specific asset in which the issue was identified. Clicking on the asset opens the asset side card without needing to navigate away to the asset table
- **Related affected assets:** Assets associated with the primary asset based on the specific scanner used. Examples include package managers (such as pip) for vulnerabilities, IaC frameworks (such as Terraform) for misconfigurations, and third-party detection engines (such as Semgrep) for code weaknesses
- Linked Cases: The number and severity of cases associated with this issue. Selecting the link opens the Cases side-card for more information without having to navigate away
- Evidence: Provides evidence and contextual details about the issue:

- Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
- Location: The exact technical context of a finding, linking security findings to specific lines of code or infrastructure files:

Examples:

- **Vulnerabilities (SCA)**: Identifies the manifest file path (such as `package-lock.json`) and the specific line where the vulnerable package is declared, providing a declaration snippet as context
- **IaC misconfigurations**: Points to the configuration file (such as `main.tf`) and the resource block's start and end lines. Context includes the full resource configuration and the specific resource identifier
- **Secrets**: Locates the file and line number of the exposure, providing a code snippet with the secret redacted for security
- **Code Weaknesses (SAST)**: References the source code file path and the start/end lines of the flaw. Context displays the vulnerable code snippet and the affected function or method name
- Collaborator: The individual or team responsible for contributing to the code or configuration where the issue was identified
- Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
- Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
- Urgency Details includes:

- **Summary:** The issue's urgency level, a breakdown of its contributing metrics, and the date it was last updated (Tip: Hover over a metric for more information)
- **Urgency context graph:** The Urgency graph provides a node-and-edge visualization that maps the structural relationship between a vulnerable asset and its connected infrastructure. By surfacing the asset hierarchy and deployment paths, the graph offers the context necessary to evaluate the scope and potential blast radius of an issue's urgency across the environment. This visualization allows you to analyze the specific deployment and runtime dependencies where the issue was detected. The urgency level itself is determined by metrics derived from the analysis of the connected assets.

Supported scanners: Vulnerabilities (SCA), Secrets, Code Weaknesses (ingested from third -party vendors) and IaC misconfigurations.

Graph structure:

- **Nodes (assets):** Represents assets such as repository or pipeline, that are linked to the specific asset where the issue was detected.

Clicking on a node opens a side card showing: Clicking on a node opens a side card showing initial details about the asset. Selecting View Details opens the asset side card without navigating away, displaying asset details, asset-specific information, and related context for that asset.

- **Edges (relationships):** The edges in the Urgency graph represent relationships between different asset types in the code-to-cloud deployment pipeline. These relationships trace how code flows from repositories through to runtime environments, providing the structural data required to calculate urgency metrics. By analyzing these relationships, Cortex Cloud determines the urgency level based on metrics such as whether the code is actively deployed, internet-exposed, accessing sensitive data, or leveraging privileged capabilities within its runtime environment.

- **Interactive controls:**

- **Layers control:** Toggle visibility of Code, Build, or Runtime layers to focus on specific pipeline segments
- **Group nodes:** Collapse multiple related nodes (such as assets within the same organization or namespace) into a single group node to manage visual density
- **Search and filter:** Locate specific nodes by typing an asset name or ID. Matching nodes are highlighted and the graph auto-focuses on the results
- **Zoom and pan:** Navigate large topologies using zoom buttons, Fit to View, or drag-and-drop panning

For more information about Urgency levels, refer to [Urgency](#).

- **Remediation:** Suggested steps to mitigate the issue. For the most efficient resolution, use the Actions tab, which provides a complete list of remediation options, including PR fixes where available

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

The Code to Cloud tab provides a trace from the vulnerability issue to its impact on your runtime environment, displaying a list of traced runtime issues and traced runtime findings. By correlating static code analysis with live production data, this feature allows security teams to understand the real-world implications of vulnerabilities. This tab displays a list of traced runtime issues and findings, offering visibility into which cloud resources currently running in production were created from the IaC code being analyzed.

Scope-Based Access Control (SBAC): The tab respects your access permissions by filtering the data displayed. You will only see cloud assets and findings that belong to asset groups you are authorized to view.

Information is organized into Traced Runtime Issues and Traced Runtime Findings tables. These tables include standard exposed properties, which you can customize by adding additional attributes from the Table Settings Menu.

- The Traced Runtime Issues table includes the following exposed properties:
 - Severity: The level of risk associated with the issue.
 - Issue Name: The name of the specific issue detected.
 - Assigned to: Indicates the individual responsible for remediating the issue.
 - Asset Name: The name of the runtime asset where the issue was detected.
 - Detection Method: The method used to identify the issue, such as a specific scanner or security tool.
 - Observation Time: The time and date when the issue was first observed in the runtime environment.
- The Traced Runtime Findings table includes these exposed properties:
 - Finding Name: The specific name of the finding detected on the asset.
 - Asset Name: The name of the runtime asset where the finding was detected.
 - Detection Method: The method used to identify the finding.
 - Observation Time: The time and date when the finding was first observed.

Actions

The Actions tab provides remediation and suggested steps to fix the issue:

- **Manual remediation instructions:** Instructions on how to upgrade vulnerable packages
- **Automated fix suggestions:** Code changes that can be automatically applied. Available for direct dependencies with known fix versions
- **Open Fix Pull Request:** Create pull requests to upgrade packages

PR Fixes are available when:

- Issue has a fix version available
- Package is a direct dependency (not transitive)
- Repository integration supports PR creation
- Detection method is the dedicated Cortex Cloud Application Security SCA CVE scanner

Bulk operations:

- Can fix up to ten issues in a single request
- Issues from the same repository are grouped into one PR
- Each repository gets a separate PR

Prerequisites for a fix PR:

- User must have write permissions to repository
- Repository must be connected via supported integration (GitHub, GitLab, Bitbucket, Azure Repos)

PR Fixes are not available when:

- No fix version exists
- Package is a transitive dependency
- Repository is archived or read-only

Example Actions:

- **Manual fix:**
 - Upgrade package from version X.Y.Z to A.B.C
 - Update package.json/requirements.txt/pom.xml
- **Automated fix:**
 - Creates PR upgrading: package-name 1.0.0 → 1.2.3

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

6.9.2.3 | CVE vulnerabilities findings

CVE vulnerabilities findings are specific, known security threats identified in your assets based on the CVE system, providing insights into potential risks.

The Cortex Cloud Application Security Vulnerabilities Findings table is a pre-filtered view of the comprehensive Findings inventory (located at Cases & Issues → Issues → Findings Table tab).

This table exclusively displays findings from Cortex Cloud Application Security SCA scanners that were detected during periodic scans, and ingested third-party data. In contrast, the comprehensive Findings table unifies vulnerabilities findings from all sources, including periodic, pull request (PR), and continuous integration (CI) scans.

NOTE:

Findings are informational and, as such, are not directly mitigable. Remediation is performed on issues derived from findings.

To access CVE findings, under Modules, select Application Security → Issues → Vulnerabilities → click the Findings tab.

The CVE vulnerability findings inventory includes the following SCA attributes. Use the Table Settings Menu to view additional properties.

Field/Attribute	Description
CVSS Severity	The Common Vulnerability Scoring System (CVSS) severity level assigned to the finding
CVE	Common Vulnerabilities and Exposures identifier associated with the finding
CVSS Score	The Common Vulnerability Scoring System severity score representing the severity of the vulnerability
EPSS Score	Exploit Prediction Scoring System: The probability (0-1) that estimates the likelihood of the vulnerability being exploited in the wild within the next 30 days
Name	The CVE identifier
Package Manager	The package management system used (such as npm, Maven, pip)
Asset Name	Name of the asset affected by the finding. Selecting an Asset Name in the table opens the asset's side card, displaying information about the asset, without having to navigate away from the Findings page.

Field/Attribute	Description
Risk Factors	Quantifiable attributes of a finding, allowing you to analyze and assess the risk. Options: Found in history, Valid, Privileged
Dependency Type	Type of dependency (direct, transitive)
Repository	Name of the repository hosting the asset in which the finding was detected
Branch	The specific branch or version of the code where the vulnerability finding was detected
File Path	Path to the file or location within the code where the vulnerability finding was detected
Data Source	Source of the finding information (the version control system)
Scanner	The type of scanner that detected the finding
Backlog Status	Backlog Status: Indicates if the finding is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how findings are categorized as backlog/new, refer to Issue/Finding classification by scanner

Expanded Findings details

Clicking on a finding in the inventory table opens the Findings side card which provides additional details.

- **Finding summary:** Found at the top of the card. Includes the finding ID and type (Vulnerability for CVE vulnerability findings)
- Description: A description of the finding including its location
- Impact: The potential security risk the finding poses to your environment
- **Timestamp:** When the finding was last updated
- Asset details: Includes Asset (The impacted asset, such as **cookie**. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the CVE vulnerability was identified, such as **JavaScript Package**)
- Evidence: Provides evidence and contextual details within your software development lifecycle containing the finding:

- Finding source
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected
- Code context
 - Repository: The name of the version control repository where the finding was located
 - Package Manager: The dependency management system (such as npm, Maven) used to include or declare the software component where the finding was detected
 - Branch: The specific branch within the repository containing the finding
 - File Path: The exact location of the finding within the repository file structure
 - First Hash: The commit hash of the first commit where this specific finding was introduced or detected
 - First Commit Date: The date of the commit that introduced the problematic code or dependency into the repository. This helps understand how long an issue has existed and for prioritizing remediation efforts based on its age
 - Root Package Name: The name of the primary software component analyzed by the scan
 - Root Package Version: The version of the primary software component analyzed by the scan
- CVE Information

- CVE ID: A unique identifier assigned to publicly known cybersecurity vulnerabilities, allowing for standardized tracking and referencing.
- CVE Description: A concise summary detailing the nature of the vulnerability, including its affected products, impacts, and potential exploitation methods.
- CVSS Severity: A qualitative ranking (e.g., Low, Medium, High, Critical) derived from the CVSS score, indicating the overall severity of the vulnerability.
- CVSS Score: A numerical score (0-10) representing the Common Vulnerability Scoring System assessment of the vulnerability's characteristics and severity.
- EPSS Score: The Exploit Prediction Scoring System score, a probability (0-1) estimating the likelihood of a vulnerability being exploited in the wild within the next 30 days.
- Fix Version: The specific software or component version in which the vulnerability has been patched or resolved by the vendor.
- Vendor Link: A direct URL to the official advisory, patch, or documentation provided by the vendor regarding the vulnerability.
- Risk Factors: Specific attributes or conditions that contribute to an issue's likelihood or the severity of its impact, such as a lack of validation, hardcoded secrets, or unpatched vulnerabilities.
- Traced Runtime Findings: A list of vulnerabilities findings that were detected in your running application

6.9.2.4 | Manage SCA CVE vulnerability issues

Right-click on a row in the inventory table to access the following actions

- Change Status: Modify the status of the issue. Values: New, In Progress, Resolved
- Change Severity: Modify the severity level of the issue. Values: Critical, High, Medium, Low
- Change Assignee: Change the user or identity assigned to address the issue
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Copy issue URL: Duplicate the URL associated with the issue, to share or reference the issue
- Show/hide rows with the [severity level]: Show/hide rows matching the [severity level] of the selected row

6.9.3 | License miscompliance issues

All license miscompliance findings detected in software packages within an organization's environment are defined as issues, which represent the smallest unit for remediating these violations. Manual fix suggestions are provided to help resolve these issues.

The Licenses issues and findings tables are filtered views of the main issues and findings tables under Cases & Issues, dedicated solely to license issues and findings detected during periodic scans.

Open source software licenses define the terms under which open source software can be used, modified, and distributed. Cortex Cloud offers three types of default license categories out of the box, providing comprehensive coverage for managing license compliance within your environment: Non Permissive Licenses, Strong Copyleft Licenses, and Weak Copyleft Licenses. For more information refer to Open-source software license categories.

How to access license miscompliance issues

To access license miscompliance issues, under Modules, select Application Security → Issues → Licenses.

Read more...

TIP:

You can also view license miscompliance issues in dedicated tabs under other sections when available:

- In Application asset inventories: navigate to Inventory → All Assets → Application → select an option from the Application menu → select an item from the inventory → Package Integrity
- In Asset inventories, navigate to Inventory → All Assets → Code:
 - On the Licenses tab under Repositories. Refer to In-depth repository asset information for more information
 - Under the All Code inventory: Select an asset from the table → Package Integrity
- In Cases and Issues; perform a query. Select Issues → AppSec Issues (under the All Domains menu) → CAS License Scanner (as the Detection Method value)

6.9.3.1 | License miscompliance issues inventory

This table includes selected fields of the license miscompliance issues issues inventory.

Read more...

Field/Attribute	Definition
Severity	The severity level assigned to a detected license compliance issue (such as high, medium, low).
Name	The full name or text of the software license
License	The identifier or short name of the associated license (such as MPL-2.0).

Field/Attribute	Definition
Asset Name	The name of the asset such as application, service) that uses this license. Selecting the asset opens its side panel, allowing you to view asset details directly without leaving the issue
Package Manager	The package manager used to install or manage the software package including the license (such as, npm, pip, maven)
License Category	A categorization of the license based on its type (such as strong copyleft)
Scan Source	The type of scan that detected the issue, such as periodic or pull request
Assignee	The individual or team responsible for addressing the license compliance issue
Created	The date and time when the license compliance issue was first detected or recorded

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Issue summary

The issue summary displayed at the top of the card provides concise details including the package manager and the name and type of package in which the license miscompliance issue was detected.

Overview

The Overview tab provides general details of the license miscompliance:

- Description: Provides a summary of the license miscompliance
- Impact: The potential impact the license miscompliance issue could have on your SDLC
- Status: The status of the issue. Values: New, In Progress, Resolved. You can modify the status of the issue from the menu in the field
- Assignee: The entity assigned to resolve the issue. By default the value is Unassigned. Select an assignee from the menu in the field
- Timestamps: Provides the date the issue was created and last updated
- Affected Assets: Identifies the asset impacted by the issue. Selecting the asset opens its side panel, allowing you to view asset details directly without leaving the issue or navigating to the dedicated Assets section under Inventory
- Linked Cases: The cases that the issue is linked to, including the amount of cases, their description and severity. Selecting a case opens its side panel, allowing you to view case details directly without leaving the issue or navigating to the dedicated Cases section under Cases & Issues
- Evidence: Details and location in the codebase of the package containing the license miscompliance issue

- Issue source and classification
 - Scanner Type: The type of security analysis tool that identified the license miscompliance (The default is AppSec Package License Scanner)
 - Scanner Source: The specific platform or vendor providing the scanner. Cortex AppSec is the default source
 - AppSec Policy: The Cortex Cloud Application Security policy that was triggered, leading to the creation of this issue. Select the icon next to the policy name to navigate to the policy on the AppSec Policies page
 - Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced license miscompliance issue). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner
- Code context
 - Package Manager: The system (such as npm, pip) managing application dependencies that contains the non-compliant license issue
 - Repository Name: The name of the version control repository where the license issue was found
 - Branch: The specific development branch within the repository containing the license issue
 - File Path: The exact location of the license issue within the repository's file structure
 - Commit Hash: The unique identifier of the code change (commit) that is associated with the license issue
- Metadata
 - Data Source: The system or integration from which the issue was originally detected (such as GitHub or CI/CD pipeline). Select the icon next to the data source redirects to the data source
- Remediation: Suggested remediation steps to address the issue. See Actions below for detailed information

Actions

The Actions tab displays suggestions to fix the license miscompliance, including contacting your legal team for further investigation.

6.9.3.2 | Expanded License miscompliance issues information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related

resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

The issue summary displayed at the top of the card provides concise details including the package manager and the name and type of package in which the license miscompliance issue was detected.

Overview

The Overview tab provides general details of the license miscompliance:

- Description: Provides a summary of the license miscompliance
- Impact: The potential impact the license miscompliance issue could have on your SDLC
- Status: The status of the issue. Values: New, In Progress, Resolved. You can modify the status of the issue from the menu in the field
- Assignee: The entity assigned to resolve the issue. By default the value is Unassigned. Select an assignee from the menu in the field
- Timestamps: Provides the date the issue was created and last updated
- Affected Assets: Identifies the asset impacted by the issue. Selecting the asset opens its side panel, allowing you to view asset details directly without leaving the issue or navigating to the dedicated Assets section under Inventory
- Linked Cases: The cases that the issue is linked to, including the amount of cases, their description and severity. Selecting a case opens its side panel, allowing you to view case details directly without leaving the issue or navigating to the dedicated Cases section under Cases & Issues
- Evidence: Details and location in the codebase of the package containing the license miscompliance issue

- Issue source and classification
 - Scanner Type: The type of security analysis tool that identified the license miscompliance (The default is AppSec Package License Scanner)
 - Scanner Source: The specific platform or vendor providing the scanner. Cortex AppSec is the default source
 - AppSec Policy: The Cortex Cloud Application Security policy that was triggered, leading to the creation of this issue. Select the icon next to the policy name to navigate to the policy on the AppSec Policies page
 - Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced license miscompliance issue). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner
- Code context
 - Package Manager: The system (such as npm, pip) managing application dependencies that contains the non-compliant license issue
 - Repository Name: The name of the version control repository where the license issue was found
 - Branch: The specific development branch within the repository containing the license issue
 - File Path: The exact location of the license issue within the repository's file structure
 - Commit Hash: The unique identifier of the code change (commit) that is associated with the license issue
- Metadata
 - Data Source: The system or integration from which the issue was originally detected (such as GitHub or CI/CD pipeline). Select the icon next to the data source redirects to the data source
- Remediation: Suggested remediation steps to address the issue. See Actions below for detailed information

Code to Cloud

The Code to Cloud tab provides a trace from the IaC misconfiguration issue to its impact on your runtime environment, displaying a list of traced runtime findings and traced runtime issues:

- The Traced Runtime Issues table includes these exposed properties
 - Severity: The level of risk associated with the issue.
 - Issue Name: The name of the specific issue detected.
 - Assigned to: Indicates the individual responsible for remediating the issue.
 - Asset Name: The name of the runtime asset where the issue was detected.
 - Detection Method: The method used to identify the issue, such as a specific scanner or security tool.
 - Observation Time: The time and date when the issue was first observed in the runtime environment.
- The Traced Runtime Findings table includes these exposed properties
 - Finding Name: The specific name of the finding detected on the asset.
 - Asset Name: The name of the runtime asset where the finding was detected.
 - Detection Method: The method used to identify the finding.
 - Observation Time: The time and date when the finding was first observed.

Actions

The Actions tab displays suggestions to fix the license miscompliance, including contacting your legal team for further investigation.

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

6.9.3.3 | License miscompliance findings

License miscompliance findings are potential licensing vulnerabilities in your open-source software packages. These findings allow you to assess and analyze your package license compliance. Promoting these findings to issues allows you to address license non-compliance. This ensures compliance with licensing requirements and maintain the integrity of your software supply chain.

The Licenses Findings table is a filtered instance of the broader Findings table found under Cases & Issues, meaning it exclusively displays findings categorized as license miscompliance findings. However, the Licenses Findings table only displays findings detected during periodic scans. In contrast, the comprehensive Findings table includes all license miscompliance findings regardless of their detection source, such as periodic, pull request (PR), and continuous integration (CI) scans.

How to access license miscompliance findings

To access license miscompliance findings, under Modules, select Application Security → Issues → Licenses → click the Findings tab.

License miscompliance findings inventory

The license findings inventory includes the following properties.

Read more...

Field/Attribute	Definition
Name	The finding name including the category assigned, such as Weak copyleft
License	The specific license under which the software package is distributed (such as MIT, GPL, Apache 2.0).
Asset Name	The name of the asset such as application, service) that uses this license
Package Manager	The package manager used to install or manage the software package including the license (such as, npm, pip, maven)
Dependency Type	Indicates whether the dependency is direct or transitive
Repository	The repository hosting the code in which the license miscompliance was detected
Branch	The branch containing the repository with the license miscompliance
License Category	A categorization of the license based on its type (such as strong copyleft)

Field/Attribute	Definition
Data Source	The version control system hosting the repository with the license miscompliance
Created	The date and time when the license compliance issue was first detected or recorded
Finding ID	The unique identifier assigned to the license finding
Backlog Status	Backlog Status: Indicates if the issue is categorized as Backlog (pre-existing technical debt) or New (a recently introduced vulnerability). To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner

[Expanded Findings details](#)

Clicking on a finding in the inventory table opens the Findings side card, which provides additional details about the finding.

[Read more...](#)

- **Finding summary:** Found at the top of the card. Includes the finding name, ID and type (Code for license miscompliance findings)
- Description: A description of the finding including its location
- **Timestamp:** When the finding was last updated
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the license miscompliance was identified)
- Evidence: Provides evidence and contextual details within your SDLC containing the IaC misconfiguration finding:
 - **Finding source**
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Collaborator: The individual or team responsible for contributing to the code or configuration where the finding was identified
 - **License information**
 - License: The license (MPL-2.0) including the miscompliance
 - License Category: The category assigned to the license. Refer to Open-source software license categories for more information about license categories
 - OSI Approved: Whether the license is OSI approved or not
 - **Code context**
 - Repository: The name of the version control repository where the finding was located
 - Package Manager: The tool (such as PIP) that manages the dependency in which the license miscompliance was detected
 - Branch: The specific branch within the repository containing the finding
 - File Path: The exact location of the finding within the repository file structure
 - First Hash: The commit hash of the first commit where this specific finding was introduced or detected
 - **Scan metadata**
 - Run ID: The unique identifier of the specific scan execution during which this finding was detected

6.9.3.4 | Open-source software license categories

Open-source software licenses define the terms under which open-source software can be used, modified, and distributed. In Cortex Cloud Application Security, licenses are scanned as part of the SCA vulnerability

scan for open-source packages. All Critical, High and Medium license miscompliance detected in open-source software packages within an organization's environment are defined as issues. This enables structured vulnerability management and focused remediation. Where applicable, manual and automated fixes are provided.

Cortex Cloud Application Security offers three types of default license categories out of the box, providing comprehensive coverage for managing license compliance within your environment:

- Non-Permissive Licenses (High severity)
- Strong Copyleft Licenses (High severity)
- Weak Copyleft Licenses (Medium severity)

Within each license type, SPDX identifiers are organized and sorted based on their characteristics and attributes. SPDX Identifiers are unique codes assigned to software licenses by the Software Package Data Exchange (SPDX) project. These identifiers are used to categorize and accurately identify software packages distributed under various types of licenses, including strong copyleft, weak copyleft, and non-permissive licenses. By associating each software package with a specific SPDX Identifier, it becomes easier to track and manage license compliance across different licensing policies, ensuring that the correct license type is identified and adhered to.

Non-permissive licenses

Non Permissive licenses policies identify software packages distributed under non-permissive or restrictive licenses. These licenses restrict how you can use, modify, and distribute the software. They may limit your ability to integrate the software into certain projects or require you to purchase a commercial license for specific uses.

Non-permissive SPDX identifiers

The following list displays supported SPDX identifiers for non-permissive licenses.

[Read more...](#)

- BUSL-1.1
- C-UDA-1.0
- CC-BY-NC-3.0-DE
- CC-BY-NC-ND-3.0-DE
- CC-BY-NC-ND-3.0-IGO
- CC-BY-NC-SA-2.0-DE
- CC-BY-NC-SA-2.0-FR
- CC-BY-NC-SA-2.0-UK
- CC-BY-NC-SA-3.0-DE
- CC-BY-NC-SA-3.0-IGO
- CC-BY-ND-3.0-DE
- Hippocratic-2.1
- JPL-image
- MS-LPL
- NCGL-UK-2.0
- PolyForm-Noncommercial-1.0.0

Strong copyleft licenses

Strong Copyleft licenses policies identify software packages distributed under strong copyleft licenses, such as the GNU General Public License (GPL). These licenses require derivative works to be distributed under the same copyleft license terms as the original work. This ensures broader access and modification rights.

Strong copyleft SPDX identifiers

The following list displays supported SPDX identifiers for strong copyleft licenses.

[Read more...](#)

- AGPL-1.0
- AGPL-1.0-only
- AGPL-1.0-or-later
- AGPL-2.0
- AGPL-3.0
- AGPL-3.0-only
- AGPL-3.0-or-later
- Arphic-1999
- CERN-OHL-S-2.0
- copyleft-next-0.3.0
- copyleft-next-0.3.1
- GPL-2.0
- GPL-3.0
- Linux-man-pages-copyleft
- OpenPBS-2.3

Weak copyleft licenses

Weak Copyleft Licenses policies identify software packages distributed under weak copyleft licenses. These licenses permit combining code with other licenses, including proprietary licenses, without mandating the entire derivative work to be released under the same copyleft license.

Weak copyleft SPDX identifiers

The following list displays supported SPDX identifiers for weak copyleft licenses.

[Read more...](#)

- Artistic-1.0
- Artistic-2.0
- APSL
- CAL-1.0-Combined-Work-Exception
- CC-BY-SA-2.0-UK
- CC-BY-SA-2.1-JP
- CC-BY-SA-3.0-AT
- CC-BY-SA-3.0-DE
- CC-BY-SA-4.0
- CDDL-1.0
- CDLA-Sharing-1.0
- CERN-OHL-W-2.0
- CPOL-1.02
- EPL-1.0
- EPL-2.0
- eCos-2.0
- EUPL-3.0
- FDK-AAC
- LGPL-2.0
- LGPL-2.1
- LGPL-3.0
- MPL-1.1
- MPL-2.0
- MS-RL
- OSL-3.0
- QPL-1.0-INRIA-2004
- Sendmail-8.23
- SimPL-2.0
- TAPR-OHL-1.0
- TPL-1.0

6.9.3.5 | Manage license miscompliance issues

Right-click on a row in the inventory table to access the following actions

- Change Status: Modify the status of the issue. Values: New, In Progress, Resolved
- Change Severity: Modify the severity level of the issue. Values: Critical, High, Medium, Low
- Change Assignee: Change the user or identity assigned to address the issue
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Copy issue URL: Duplicate the URL associated with the issue, to share or reference the issue
- Show/hide rows with the [severity level]: Show/hide rows matching the [severity level] of the selected row

6.9.4 | Package Integrity

Package Integrity (also referred to as Package Operational Risk) assesses the operational risk and potential impact of each package in your codebase by examining package maintainer and popularity factors, and other relevant metrics. This analysis results in open-source package operational risk severity levels being categorized into High, Medium and Low. By prioritizing risks based on these categories, you can effectively focus remediation efforts on the most critical issues.

The Package Integrity issues and findings tables are filtered views of the main issues and findings tables under Cases & Issues, dedicated solely to issues and findings categorized as package operational risks and detected during periodic scans.

The following table defines the operational risk metrics used to assess open-source packages.

Metric	Property
Maintainer	Level: Indicates the level of maintenance based on various computed factors
—	Versions
—	Last release
—	Last commit
—	Created

Metric	Property
—	Open issues
—	Open pull requests
Popularity	Level: Indicates the level of popularity based on various computed factors
—	Weekly downloads
—	Number of stars
—	Number of forks
—	Contributors

Limitation

Package Operational Risk relies on metadata from the package's associated GitHub repository (such as activity, popularity, deprecation status). If the package manager entry does not include a valid GitHub repository URL, operational risk data may be unavailable for that package.

Package Operational Risk policy enforcement

You can create policies to automatically detect and prevent the introduction of operational risks. These policies ensure that all third-party dependencies align with your organization's security standards and risk tolerance, helping you maintain a secure software supply chain. For more information about creating policies, refer to Create Cortex Cloud Application Security policies.

6.9.4.1 | Package Integrity inventory

The following table describes selected fields in the Package Integrity issues inventory table.

Read more...

Property	Description
Severity	Severity level of the package integrity issue (such as Critical, High, Medium, Low)
Name	Name of the package integrity issue
Asset Name	Name of the asset affected by the package integrity issue
Package Manager	The package management system used (such as npm, Maven, pip), in which the package with the issue was detected
Repository	Name of the repository where the package was obtained
Branch	The specific branch of the code where the package was used
File Path	The exact path to the file within the repository's structure where the issue is located
Data Source	The sources from which the issue was generated or collected
Dependency Type	Type of dependency (direct, transitive)
Created	Timestamp of when the package integrity issue was first detected
Status	The current state of the issue (New, In Progress, Resolved)

Property	Description
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced risk (New). For more on Backlog status, refer to Backlog baseline
Assignee	The individual or identity responsible for addressing the package integrity issue

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata, and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Issue summary

The issue summary, displayed at the top of the card, provides concise details about the issue, including its name, severity, type (such as configuration or data), the number of days the issue has been open, and the type of scanner that detected the issue (such as IaC scanner).

NOTE:

You can modify the severity level from the severity field menu.

Overview

The Overview tab provides general details of the package integrity issue:

- Description: Provides a summary of the issue
- Impact: The potential impact the issue could have on your SDLC
- Status: The status of the issue. Values: New, In Progress, Resolved. You can modify the status of the issue from the menu in the field
- Assignee: The entity assigned to resolve the issue. By default the value is Unassigned. Select an assignee from the menu in the field
- Timestamps: Provides the date the issue was created and last updated
- Affected Assets: Identifies the asset impacted by the issue. Selecting the asset opens its side panel, allowing you to view asset details directly without leaving the issue or navigating to the dedicated Assets section under Inventory
- Linked Cases: The cases that the issue is linked to, including the amount of cases, their description and severity. Selecting a case opens its side panel, allowing you to view case details directly without leaving the issue or navigating to the dedicated Cases section under Cases & Issues
- AppSec Policy: The policy (with its defined rules and conditions) that was triggered, leading to the creation of this issue. Includes a link to the policy in the general Cortex Cloud Application Security policy inventory
- Evidence: Details and location in the codebase of the package containing the package operational risk issue

- Issue source and classification
 - Scanner Type: The type of security analysis tool that identified the package integrity issue
 - Scanner Source: The specific platform or vendor providing the scanner. Cortex AppSec is the default source. This field is not selectable
 - Package Manager: The software package management system used (such as npm, pip, Maven, NuGet)
 - Package Registry URL: The URL of the repository or registry from which the package was sourced. Select the icon next to the policy name to navigate to the package registry
 - Backlog Status: Indicates if the finding is categorized as Backlog (pre-existing technical debt) or New (a recently introduced package integrity issue)
 - AppSec Policy: The Cortex Cloud Application Security policy that was triggered, leading to the creation of this issue. Select the icon next to the policy name to navigate to the policy in the the AppSec Policies page. To understand how issues are categorized as backlog/new, refer to Issue/Finding classification by scanner
- Code context
 - Branch: The specific development branch within the repository containing the package integrity issue
 - File Path: The exact location of the relevant file (such as package.json, pom.xml) within the repository's file structure
 - Commit Hash: The unique identifier of the code change (commit) that is associated with the package integrity issue
 - Collaborator: The user that authored the relevant code commit
 - Commit Time: The date and time when the associated code commit was made
- Metadata
 - Data Source: The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Select the icon next to navigate to the data source
 - Risk Factors: Specific attributes or conditions that contribute to the severity or exploitability of the package integrity issue
- Remediation: Suggested manual remediation steps to address the operational risk. See Actions for more detailed information

Actions

The Actions tab provides suggested steps to address package operational risk issues.

6.9.4.2 | Expanded Package Integrity issues inventory information

Clicking an issue in the table opens a detailed side card that serves as a centralized workspace for investigation and remediation. The card opens on the Overview tab, which presents general details, metadata,

and a summary of the evidence, while the War Room tab provides an audit trail of all automatic and manual actions taken on the issue, offering context on how it has been investigated over time. The Actions tab lists available remediation options, and for IaC and vulnerability issues, the Code to Cloud tab displays related resources and lineage between code and runtime, allowing you to understand the impact of the issue across environments.

Overview

- **Timestamp:** When the issue was created and last updated
- Status: The issue status. Values: New, In Progress, Resolved. You can set the status as required
- Assignee: The entity assigned to mitigate the issue. You can assign the issue
- Description: Provides details about the package integrity issue, including the location where the package was detected (such as specific repository, file path, and line number) and the operational risks identified
- Impact: The impact that the issue could potentially have on your SDLC
- Asset details: Includes Asset (The impacted asset. Clicking on the asset opens the asset side card without needing to navigate away to the asset section) and Asset Type (The specific asset type in which the IaC resource was identified)
- Evidence: Provides evidence and contextual details within your SDLC containing the package operational risk:

- Issue source
 - Data Source: The system or integration from which the issue data was originally pulled (such as GitHub or a CI/CD pipeline). Click the icon next to the data source to navigate to the data source itself
 - Category: Package Operational Risk is the immutable value
 - AppSec Policy: The violated security standard that led to the creation of the issue. Includes a link to the policy
 - Scanning context
 - Scanner Type: Code. The value is immutable
 - Scanner Source: Cortex AppSec. The value is immutable
 - Package Manager: The package manager (such as PIP, npm, Maven) in which the vulnerability was detected
 - Package Registry URL: The URL of the package registry (e.g., npm, PyPI, Maven Central) from which the vulnerable package was obtained
 - Code location
 - Repository Name: The name of the version control repository where the issue was located
 - Branch: The specific branch within the repository containing the issue
 - File Path: The exact location of the issue within the repository file structure
 - Commit Hash: The commit hash of the most recent commit that modified the code where the issue was detected
 - Commit Time: The timestamp of the most recent commit that modified the code where the issue was detected
 - Risk analysis
 - Risk Factors: Specific attributes or conditions that contribute to an issue's likelihood or the severity of its impact
- Remediation: Suggested mitigation for the package operational risk.

NOTE:

Different issue types include different properties; therefore, not all properties are available for every issue.

Actions

This tab includes suggested steps to fix the issue. No automatic fixes are available for Package Operational Risks.

War Room

The War Room provides an audit trail of all automatic or manual actions taken on an issue, serving as a dedicated space to review and interact with your issue. Each issue has a unique War Room. With machine learning insights, the Cortex Cloud platform suggests the most effective analysts and command sets to help you address issues efficiently.

6.9.4.3 | Package Integrity findings

Package Integrity scans produce findings, which are potential package operational risks within your software packages. These insights help assess and analyze the security posture of your software supply chain.

The Package Integrity Findings table is a filtered instance of the broader Findings table found under Cases & Issues, meaning it exclusively displays findings categorized as Package Integrity findings.

NOTE:

Findings cannot be mitigated. They must be promoted to issues to enable remediation efforts to secure your software packages.

To access package integrity findings, navigate to Package Integrity issues (see Package integrity issues), and click the Findings tab.

The following table describes selected fields in the Package Integrity Findings inventory table.

Read more...

Property	Description
Name	The name of the package operational risk
Asset Name	The asset in which the finding was detected
Finding ID	A unique identifier assigned to the finding
Data Source	The version control system hosting the repository which includes the package in which the finding was detected
Package Manager	The package manager hosting the package in which the finding was detected
Repository	The repository hosting the package in which the finding was detected
Branch	The branch in which the finding was detected

Property	Description
File Path	The exact path to the file within the repository's structure where the issue is located
Dependency Type	Whether the package is direct or transitive
Created	When the finding was first detected
Backlog Status	Indicates whether the issue is classified as pre-existing technical debt (Backlog) or a newly introduced misconfiguration (New). For more on Backlog status, refer to Backlog baseline

[Expanded Findings details](#)

Click on a finding in the inventory table to open the Findings side card, which provides additional details about the finding.

- **Summary:** A summary of the finding at the top of the card including name, id and the type of scanner (Code is the immutable value for Package Integrity issues) that detected the finding
- **Description:** Provides a summary of the finding
- **Impact:** The potential impact the issue could have on your SDLC
- **Timestamps:** Provides the date the issue was last updated
- **Affected Assets:** Identifies the asset impacted by the finding. Selecting the asset opens its side panel, allowing you to view asset details directly without leaving the finding or navigating to the dedicated Assets section under Inventory
- **Evidence:** Details and location in the codebase of the package containing the package operational risk finding:
 - **Issue source and classification**
 - **Package Manager:** The dependency management system (such as npm, Maven) used to include or declare the software component where the finding was detected
 - **Dependency Type:** Indicates whether the vulnerability originates from a direct dependency (explicitly declared in your project) or an indirect/transitive dependency (pulled in by one of your direct dependencies)
 - **Data Source:** The system or integration from which the finding data was originally pulled (such as GitHub or a CI/CD pipeline). Select the icon next to navigate to the data source
 - **Code context**
 - **Repository:** The name of the version control repository where the finding was located
 - **Branch:** The specific development branch within the repository containing the package integrity finding
 - **First Hash:** The commit hash of the earliest code change where this finding was identified
 - **First Commit Date:** The date of the commit that introduced the problematic code or dependency into the repository. This helps understand how long an issue has existed and for prioritizing remediation efforts based on its age
 - **Collaborator:** The user that authored the relevant code commit
 - **Code:** The file and code including the license miscompliance in which the finding was detected
 - **Scan Metadata**
 - **Run ID:** The unique identifier of the specific scan execution during which this finding was detected

6.9.4.4 | Manage Package Integrity issues

Right-click on a row in the inventory table to access the following actions

- Change Status: Modify the status of the issue. Values: New, In Progress, Resolved
- Change Severity: Modify the severity level of the issue. Values: Critical, High, Medium, Low
- Change Assignee: Change the user or identity assigned to address the issue
- Copy text to clipboard: Duplicate selected text for easy pasting elsewhere
- Copy entire row: Duplicate the entire row of data for easy pasting elsewhere
- Copy issue URL: Duplicate the URL associated with the issue, to share or reference the issue
- Show/hide rows with the [severity level]: Show/hide rows matching the [severity level] of the selected row

6.10 | Application Security scans management

Abstract

6.10.1 | Overview

You can manage Cortex Cloud Application Security scans through dedicated periodic branch and pull request (PR) scans inventories, which provide a central view of scan health, status, scope, and detected issues. This enables efficient tracking, analysis, and management of scans for vulnerability insights.

In addition, you can configure native scanners, optimize settings, and monitor scan health to streamline your scan management process and enhance your security posture.

[How to access Cortex Cloud Application Security scan management](#)

To access scan management:

1. Under Modules, select Application Security.

2. Under Scans, select a scan type:

- Branch Periodic Scanning: Scans code branches on a schedule to identify vulnerabilities early in development. For more information about branch periodic scans, refer to [Branch periodic scans](#)
- Pull Request Scans: Scans code changes within pull requests to prevent the introduction of new vulnerabilities. For more information about pull request scans, refer to [Pull Request scans](#)

[Scan data presentation](#)

Periodic and pull request scan details are presented on the Cortex Cloud console across three levels of granularity: an inventory table providing a list of scans, a side panel providing general scan details including a high-level breakdown of the findings and issues detected during the scan, and an expanded description card, providing detailed information about the issues generated from these scans.

NOTE:

While scans provide a comprehensive inventory of all issues detected during a scan, dedicated inventories are also maintained for specific scan types for more granular management. For more information, refer to

Infrastructure as Code (IaC) misconfiguration scanner, Secrets scanners and Software Composition Analysis (SCA) scanners.

6.10.2 | Branch periodic scans

Branch periodic scans are automated checks that assess the security posture of applications and infrastructure. These scans run at regular intervals using supported Cortex Cloud Application Security scanners to identify vulnerabilities and weaknesses. You can analyze the scans directly from a dedicated inventory table, which displays branch periodic scan details, including code context, scan date, health, detected findings, and generated issues.

How to access branch periodic scans

Under Modules select Application Security → Branch Periodic Scanning (under Scans).

Branch periodic scan inventory

The branch periodic scan inventory displays the following details:

The following fields are displayed in the scan inventory table.

- Repository: Identifies the scanned repositories
- Organization: Specifies the organization owning the repository
- Scanned Branch: Indicates the branch analyzed during the scan
- Scan Date: Records the timestamp of the last scan execution
- Scan Health: Displays the health of the scan. Values include:
 - Error
 - Partially: Indicates that the scan execution resulted in a partial completion, with some scan modules (for example IaC) succeeding and others (such as Secrets) failing
 - Completed

NOTE:

The inventory table displays scan issues for visibility only; remediation is not available here. To resolve issues, navigate to the dedicated issue type inventory, where you can manage and remediate them.

Expanded periodic scan details

Selecting a scan from the inventory opens its side car, which displays a general overview of the scan's details and provides access to details of issues and findings via dedicated scan type tabs.

Overview

The overview tab displays these scan details.

- General scan information
 - Organization: The specific business entity or organizational unit associated with the periodic scan
 - Scan Date: The date and time when the periodic scan was performed
 - Scan Health: The overall health status of the periodic scan, indicating its success, failure, or other relevant states
- Code context
 - Repository: The version control repository that was scanned
 - Scanned Branch: The specific branch within the repository that was subjected to the periodic scan
- Scan results summary
 - Issues: A breakdown by severity and count of actionable security issues identified from the scan's findings
 - Issues by Type: A categorization and count of identified issues based on their specific vulnerability types (such as IaC, Secrets)
 - Findings: A breakdown by severity and count of findings discovered by the scan before being converted into actionable issues
 - Findings by Type: A categorization and count of raw findings based on their specific detection types.

Vulnerabilities

When selecting the Vulnerabilities tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Vulnerabilities issues page.

For detailed information about vulnerabilities issues, refer to Software Composition Analysis (SCA) vulnerability issues.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Configurations

When selecting the Configurations tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated IaC misconfigurations issues page.

For detailed information about IaC misconfiguration issues, refer to Overview.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Secrets

When selecting the Secrets tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Secrets issues page.

For detailed information about Secrets issues, refer to Secrets issues.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Package Integrity

When selecting the Package Integrity tab, the Issues tab is displayed by default, displaying a list of package operational risk and license issues, with detailed properties for each entry. Selecting an entry then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Package Integrity issues page.

For detailed information about Package Integrity issues, refer to Package Integrity.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

6.10.3 | Pull Request scans

Pull Request (PR) scans are initiated by events triggered by version control systems such as GitHub, GitLab, Bitbucket and Azure Repos, or via webhooks. These scans are run on default or non-default branches containing open PRs or Merge Requests (MR) from your integrated repositories.

The system executes scans based on the following logic:

- **Initial Scan (PR/MR Open):** When a new PR or MR is opened, the system performs a full scan of all files included in the request
- **Subsequent Scan (New Commits):** For any new commit pushed to an existing open PR/MR, the system executes a delta analysis (diff scan). This focused scan analyzes only the modified files, which prevents redundant findings. The PR comments are updated according to the results of the latest commit scan

Scan results are based on default enforcement thresholds. You can analyze PR scans directly from the dedicated inventory table, which displays detailed metrics including code context, scan date, health, status,

detected findings, and generated issues.

How to access Pull Request scans

Under Modules select Application Security → Pull Request Scans (under Scans).

Pull Request scan inventory

The pull request scan inventory displays the following details:

- Repository: Identifies the scanned repositories
- Organization: Specifies the organization owning the repository
- Scanned Branch: Indicates the branch analyzed during the scan
- Scan Date: Records the timestamp of the last scan execution
- Pull Request ID: The unique identifier of the PR
- Commit: The commit included in the PR
- Scan Health: Displays the health of the scan. Values include:
 - Error
 - Partially: Indicates that the scan execution resulted in a partial completion, with some scan modules (for example IaC) succeeding and others (such as Secrets) failing
 - Completed
- PR Status: Status of the PR scan. Values: "Passed", 'Blocked', 'In Progress'

NOTE:

The inventory table displays scan issues for visibility only; remediation is not available here. To resolve issues, navigate to the dedicated issue type inventory, where you can manage and remediate them.

Expanded Pull Request scan details

Selecting a scan from the inventory opens its side car, which displays a general overview of the scan's details and provides access to details of issues and findings via dedicated scan type tabs.

Overview

The overview tab displays these scan details.

- General scan information
 - Organization: The specific business entity or organizational unit associated with the periodic scan
 - Scan Date: The date and time when the periodic scan was performed
 - Scan Health: The overall health status of the periodic scan, indicating its success, failure, or other relevant states
- Code context
 - Repository: The version control repository that was scanned
 - Scanned Branch: The specific branch within the repository that was subjected to the periodic scan
- Scan results summary
 - Issues: A breakdown by severity and count of actionable security issues identified from the scan's findings
 - Issues by Type: A categorization and count of identified issues based on their specific vulnerability types (such as IaC, Secrets)
 - Findings: A breakdown by severity and count of findings discovered by the scan before being converted into actionable issues
 - Findings by Type: A categorization and count of raw findings based on their specific detection types.

Vulnerabilities

Selecting the Vulnerabilities tab, the Issues tab is displayed by default. Selecting an entry then opens its side card (which includes remediation options) directly within Scans Management, eliminating the need to navigate to the dedicated Vulnerabilities issues page.

For detailed information about vulnerabilities issues, refer to Software Composition Analysis (SCA) vulnerability issues.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Configurations

When selecting the Configurations tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated IaC misconfigurations issues page.

For detailed information about IaC misconfiguration issues, refer to Overview.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Secrets

When selecting the Secrets tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Secrets issues page.

For detailed information about Secrets issues, refer to [Secrets issues](#).

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Package Integrity

When selecting the Package Integrity tab, the Issues tab is displayed by default, displaying a list of package operational risk and license issues, with detailed properties for each entry. Selecting an entry then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Package Integrity issues page.

For detailed information about Package Integrity issues, refer to [Package Integrity](#).

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

6.10.4 | CI scans

CI scans detect exposed secrets, misconfigurations in your infrastructure-as-code (IaC) files, vulnerabilities in your software composition analysis (SCA) packages, and license non-compliance in your CI pipelines. Scan results, including details of the scan, scan status and health, are displayed in a dedicated table. For deeper analysis, click on a scan in the table to open a side panel containing general scan details and a breakdown of CI findings and issues.

How to access CI scans

Under Modules select Application Security → CI Scans (under Scans).

CI scan inventory table

The CI scan inventory table displays the following details:

Property	Description
Repository	The scanned repository
Organization	The organization owning the repository
Scanned Branch	Indicates the branch analyzed during the scan
Scan Date	Records the timestamp of the last scan execution
Scan Health	<p>Displays the health of the scan. Values include:</p> <ul style="list-style-type: none"> • Completed • Partially: Indicates that the scan execution resulted in a partial completion, with some scan modules (for example IaC) succeeding and others (such as Secrets) failing • Error
CI Status	Displays the health status of the scan. Values: Passed, Blocked, In Progress

NOTE:

The inventory table displays scan issues for visibility only; remediation is not available in scan management. To resolve issues, navigate to the dedicated issue type inventory, where you can manage and remediate them.

Expanded CI scan details

Selecting a scan from the inventory opens its side car, which displays a general overview of the scan's details and provides access to details of issues and findings via dedicated scan type tabs.

Overview

The overview tab displays these scan details.

- General scan information
 - Organization: The specific business entity or organizational unit associated with the periodic scan
 - Scan Date: The date and time when the periodic scan was performed
 - Scan Health: The overall health status of the periodic scan, indicating its success, failure, or other relevant states
- Code context
 - Repository: The version control repository that was scanned
 - Scanned Branch: The specific branch within the repository that was subjected to the periodic scan
- Scan results summary
 - Issues: A breakdown by severity and count of actionable security issues identified from the scan's findings
 - Issues by Type: A categorization and count of identified issues based on their specific vulnerability types (such as IaC, Secrets)
 - Findings: A breakdown by severity and count of findings discovered by the scan before being converted into actionable issues
 - Findings by Type: A categorization and count of raw findings based on their specific detection types.

Vulnerabilities

When selecting the Vulnerabilities tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Vulnerabilities issues page.

For detailed information about vulnerabilities issues, refer to Software Composition Analysis (SCA) vulnerability issues.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Configurations

When selecting the Configurations tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated IaC misconfigurations issues page.

For detailed information about IaC misconfiguration issues, refer to Overview.

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Secrets

When selecting the Secrets tab, the Issues tab is displayed by default. Selecting an issue in the table that is presented then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Secrets issues page.

For detailed information about Secrets issues, refer to [Secrets issues](#).

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

Package Integrity

When selecting the Package Integrity tab, the Issues tab is displayed by default, displaying a list of package operational risk and license issues, with detailed properties for each entry. Selecting an entry then opens its side card directly within Scans Management, eliminating the need to navigate to the dedicated Package Integrity issues page.

For detailed information about Package Integrity issues, refer to [Package Integrity](#).

Select the Findings tab to open a list of findings associated with the issue, including the name of the finding, the asset in which the finding was detected, and the repository hosting the asset.

Click on a finding for additional details, including a description of the finding the asset type and group associated with the finding, when last updated, and evidence for the finding.

6.10.5 | Manage repository scan configurations

You can achieve granular control over repository security through scan configurations. Tailor scans by managing branches, enabling/disabling scanners, customizing PR behavior, excluding paths, and managing repository deletion.

To access repository scan configurations:

1. Select Settings → Data Sources & Integrations and search for the provider.
2. Select the relevant provider. You can see all the instances and their statuses.
3. Select an instance. The instance repositories and their statuses are displayed (including scan coverage).

Scan coverage indicates the scan engines that run against the repository, such as IaC, SCA, and Secrets.

4. Right-click on a selected repository and select Scan Configuration.
5. Configure scan settings:

- Active Scanners:
 - Enable or disable Code Security scanners (IaC, SCA, Secrets)
 - Enable or disable Git history scans to automatically detect secrets still present in a repository's past commits. By identifying these exposed credentials, you can proactively address risks and prevent potential data breaches
 - Enable/disable secrets validation to check a detected secret's activity status via public APIs. This allows you to prioritize and address only active, exposed secrets. Secrets validation scanning is available through the Cortex Cloud tenant and IDE extensions.
- PR Scanning:
 - Scan PR: Enable security scans to automatically trigger on a pull request (PR) to identify potential security risks or misconfigurations before the code is merged
 - Fail PR on scan error: Enable pull requests (PRs) to automatically fail on a scan error. This prevents security risks and misconfigurations from being merged
- Tagging Bot: Enable to automatically append a unique `yor_trace` ID to resource and module blocks when a pull request is opened. This identifier enables precise code-to-cloud tracing throughout the software development lifecycle
- Exclude Path: Specify the files and directories to exclude from scanning.
 - Exclude directories: Enter folder names to exclude all files inside, such as `node_modules/`, `tests/`, `dist/`
 - Exclude files by pattern: Use wildcards: `*` for any number of characters, `?` for a single character
 - `.*.log`: Excludes all log files
 - `*.min.js` Excludes all minified JavaScript files
 - Use a Comma-Separated List: Separate multiple exclusions with commas. For example, `node_modules/, dist/, docs/`
 - Avoid leading slashes: Use `dist/`, not `/dist/`

6. Click Save.

6.10.6 | Monitor data source instances health

You can monitor the health and status of your integrated data source instances and their repositories. Tracking and analyzing this data allows you to understand and troubleshoot errors, mitigate detected vulnerabilities, and ensure the reliability of your data sources.

How to view scans health

1. Navigate to Settings → Data Sources & Integrations.
2. Filter for a data source such as GitHub (SaaS).

The data source is displayed, including the number of connected instances and their statuses. Status options include Connected, Error, Warning, and Disabled.

3. Select the relevant data source.

A list of data source instances is displayed.

4. Select an instance.

5. Click the Review health link.

The health status page opens, displaying:

- The status of the instance, connected repositories and instance type, such as cloud are displayed on the top card
- A list of repositories connected to the instance, including scan details such as status, scan coverage (connected scanners such as SCA, Secrets and IaC and the scan status), location (scanned branch, organization, repository) and timestamp

Understanding errors and warnings

• An error status is displayed if:

- More than 50% of the repositories associated with the instance have errors
- There is a token error for a repository
- There are missing permissions (categorized as an error based on severity)
- Webhook subscription errors (categorized as an error based on severity)
- At least one repository includes a warning

• A warning status is displayed if:

- Less than 50% of the repositories associated with the instance have errors
- Missing permissions (categorized as a warning based on severity)
- Webhook subscription errors (categorized as a warning based on severity)

Troubleshoot scan health

You can troubleshoot both instance and repository issues.

1. Select an instance containing an error or warning status.

2. Review instance issues: click Review Health in the top bar.

A list of errors by repository and related to the instance are displayed. The list includes these unique properties:

- Classification: Options: Warning or Error
- Error Type: The type of instance error. Options: Permission, Clone, Connection, Webhook subscription, Token
- Repository: the repository containing the issue
- Scope: the scope of the issue

3. Review repository issues: select a repository in the table with a status of Error or Warning.

A list of repository-related warnings and errors is displayed, including details such as description, type, location, and timestamp.

6.11 | Application Security Policies

Abstract

AppSec policies define threat responses by setting conditions, scope, and actions. Use out-of-the-box policies or clone them to create custom ones.

An Cortex Cloud Application Security policies define how to respond to application security threats. The policy evaluates raw findings, such as CVE vulnerabilities or IaC misconfigurations detected by scanners, against defined conditions and scope to determine whether they should be raised as actionable issues. This keeps all scan data visible while ensuring that only findings that meet your risk criteria require action.

Cortex Cloud Application Security policies serve two primary functions:

- **Detection:** Automating the creation of issues for specific security findings to prioritize remediation
- **Prevention:** Blocking pull requests (PRs) or CI/CD builds that introduce security risks, ensuring threats are stopped before they reach production

Use cases

- **Automate workflows:** Automatically generate issues only when specific criteria are met, rather than for every raw scanner finding
- **Enforce gates:** Block PRs or fail CI/CD pipelines when critical vulnerabilities or misconfigurations are detected
- **Ensure compliance:** Enforce specific regulatory requirements such as PCI-DSS, HIPAA, or SOC2
- **Prioritize risk:** Focus remediation efforts on high-impact issues by filtering out noise based on severity or context

Policy types and categories

Policies are categorized based on their focus and their origin:

- By focus:
 - **Code security policies:** Address risks in the code-to-cloud workflow, including secrets, CVE vulnerabilities, IaC misconfigurations, and license violations
 - **CI/CD configuration policies:** Scan the pipeline infrastructure itself (such as, GitHub Actions, Jenkins) to detect misconfigurations and risky settings in workflows
 - **Drift detection policies:** Identify discrepancies between the desired infrastructure state defined in IaC templates and the actual runtime configuration of cloud resources to prevent unauthorized manual modifications and mitigate environmental drift
- By origin:
 - **Out-of-the-Box (OOTB):** Disabled by default. These Cortex Cloud Application Security-provided policies pre-configured according to security best practices. The policies are immutable but can be cloned to serve as templates for custom configurations
 - **AI Guardrails:** AI-driven policy recommendations generated through the analysis of historical security findings and organizational patterns to optimize detection accuracy
 - **Custom Policies:** User-defined policies tailored to meet specific organizational requirements or unique infrastructure environments

Core components

Cortex Cloud Application Security policies are built around core components, Conditions, Scope, Triggers, and Actions, that define their logic and execution. For a detailed explanation of each component, to Create Cortex Cloud Application Security policies.

6.11.1 | User roles and permissions

These user roles and permissions are required for AppSec policies:

- Roles with Policies View/Edit permissions can create and modify detection policies
- Roles with Policies Read permissions can view detection policies
- An AppSec Admins user role has View/Edit permissions
- A DevSecOps user role only has View permissions

6.11.2 | Policies inventory

The Cortex Cloud Application Security policies inventory includes both out-of-the-box and custom policies.

To access Cortex Cloud Application Security polices, under Modules, select Cortex Cloud Application Security → AppSec Policies (under Policy Management).

The following list describes the policy properties exposed by default in the inventory table. Additional properties can be added to the table from the Table Settings Menu.

Field/Attribute	Description
Policy Name	The name of the Cortex Cloud Application Security policy
Status	Whether the policy is enabled or disabled. Disabled policies are greyed out but remain clickable, allowing you to open the policy side panel
AI-Recommended (label)	Indicates the policy was generated from an AI-recommended guardrail without modification
Description	A description of the Cortex Cloud Application Security policy
Conditions	The specific criteria used to determine when policy actions are applied
Actions	The steps taken when the policy conditions and scope are met
Scope	The type of assets to be evaluated by the policy. See for more information about policy scope
Trigger	Trigger types that define when the condition will be evaluated. Options include Periodic scan, Pull Request scan and CI scan
Last Triggered	The last time that the policy was triggered
Created By	The user or entity that created the policy
Modified by	The user or entity that modified the policy

Field/Attribute	Description
Modification Time	The timestamp of the most recent change to the policy
Open Issues	The amount of issues detected by the policy that remain unresolved

Expanded policy details

Selecting a policy opens a side panel where you can review additional details:

- **Metadata:**
 - **Policy details:** Name and description of the policy
 - **Policy ownership:** Information on the policy's creator and last modifier
- **NOTE:**
 - To view all out-of-the-box (OOTB) policies, filter by **Policy Owner = System**.
- **Timestamps:** The last time the policy was modified and last triggered
- **Scope:** The asset type the policy applies to, along with a table summarizing the policy conditions, trigger, and actions, displayed as follows:
 - When: The trigger that initiates the policy action, such as Periodic, Pull Request, or CI scans
 - If: Conditions that are applied to the policy. For example: (**Finding Type = IaC Misconfiguration**) AND (**Severity = Critical**)
 - Then: Triggered actions for the policy, such as Create issue and Block PR

6.11.3 | AI-recommended guardrails

Abstract

AI-recommended guardrails automate risk prevention by analyzing your organization's security data to suggest granular, context-aware guardrails that lock down clean environments and block recurring risks.

The Cortex Cloud Application Security AI-recommended guardrails shift your security posture from reactive detection to proactive, automated prevention. By analyzing your organization's historical security data and risk profile, the AI engine detects gaps and suggests granular, context-aware policies tailored to your specific environment. Unlike static policies, these recommendations are data-driven responses designed to maximize immediate risk reduction with minimal operational overhead.

Enforcement strategies

The intelligence engine categorizes recommendations into two primary strategic models designed to reduce technical debt and prevent regression.

- **Lockdown scopes** (posture preservation): Maintains the integrity of clean code scopes. The AI identifies repositories or applications currently free of high-severity vulnerabilities and recommends policies to prevent new risks from being introduced into these clean scopes
- **Stop the bleeding** (risk containment): Targets recurring issue patterns. The engine analyzes the last 30 days of data to identify persistent risks, such as a specific vulnerable package appearing across multiple repositories, and suggests policies to block these components

Context-aware prioritization

The engine prioritizes guardrails for **Critical** and **High** severity issues, specifically those affecting assets detected in **deployed environments**, ensuring that guardrails address actual exposure rather than theoretical risk

Discovery and application

While guardrails are technically managed as policies, the workflow for AI recommendations spans two interface areas:

- **Discovery (ASPM Command Center)**: The Command Center serves as the discovery layer, presenting the most impactful opportunities for risk reduction
- **Enforcement (AppSec Policies)**: Recommendations are reviewed, customized, and formally applied as active enforcement policies within the AppSec Policies list

Supported scopes

Currently, AI-recommended guardrails are generated based exclusively on findings from the Software Composition Analysis (SCA) scanner. They do not currently support data from third-party scanners, Secrets, or Infrastructure as Code (IaC) scanners

6.11.3.1 | Manage AI-recommended guardrails

AI-recommended guardrails are tailored to your environment's findings. Applying these recommendations converts them into active enforcement policies across your organization.

For more information on AI-recommended guardrails, refer to AI-recommended guardrails.

NOTE:

Recommended guardrails are generated based exclusively on findings from the Cortex Cloud Application Security Software Composition Analysis (SCA) scanner. They do not support SCA data ingested from third-party scanners, nor do they currently support other finding categories, such as Secrets or Infrastructure as Code (IaC).

To review, implement, or dismiss AI-recommended guardrails:

1. Access AI-recommended policies:

- From the ASPM Command Center: Select Explore in the AI Guardrails section
- From the AppSec Policies page: Navigate to Modules → Application Security → AppSec Policies → click AI Recommendations

NOTE:

The total amount of available recommended guardrails is displayed on the AI Recommendations button.

2. Select an action based on your requirements:

- Apply: Immediately converts the recommendation into an active policy. Use this for high-confidence suggestions where you do not need to modify parameters
- View Assets: Redirects you to the Scope step of the AppSec Policies wizard, where you can validate the specific list of repositories or business applications that the AI engine has identified for inclusion in the policy's enforcement perimeter
- View Evidence: Redirects to the relevant Findings page, automatically filtered to display the specific instances of the identified risk, such as a vulnerable package
- View Details: Redirects to the Summary step of the AppSec Policies wizard, allowing you to validate the suggested policy configuration, including scope, conditions and actions, before creating the policy.
 - Review the parameters. Use the Back and Next buttons to navigate the wizard and modify policy parameters as needed
 - Select Done to approve and create the policy immediately without additional navigation
- Dismiss: If the recommendation is not relevant:

Monitor active guardrails

Once applied, policies based on AI guardrail recommendations are displayed in the AppSec Policies table and are tagged as AI-Recommended for easy filtering and reporting.

NOTE:

The AI-Recommended tag is only assigned to recommendations applied as-is, that is, without modifications. Edited recommendations generate policies without this label.

Next step: Investigate and remediate SCA issues

Investigate and remediate vulnerabilities detected by your AI-Recommended policies to secure your software dependencies and strengthen your software supply chain. For more information refer to Software Composition Analysis (SCA) vulnerability issues.

6.11.4 | Create Cortex Cloud Application Security policies

You can create custom policies to detect and prevent security risks across your software development environments and workflows. Cortex Cloud Application Security policies define how to respond to code and configuration drift, either by blocking risks (prevent) or creating issues for remediation (detect). To enhance and automate protection, you can use AI Guardrails, which analyze real security findings to recommend

tailored policies. These tools help enforce consistent, automated responses to threats and infrastructure deviations across your code and CI/CD environments before they reach production.

Cortex Cloud Application Security policies are categorized into the following types based on their focus:

- Code Security policies
- AI-recommended guardrails
- CI/CD Configuration policies
- Create IaC Drift Detection policies

Policy evaluation overview

AppSec Policies consists of four core pillars: Scope, Conditions, Triggers, and Actions. Together, these components define which assets are evaluated, what is evaluated, when evaluation occurs, and how enforcement is applied. AppSec Policies support multiple triggers and actions.

- **Conditions** (The logic)

Defines what constitutes a violation by selecting a scanner or multiple scanners and applying attribute filters.

- **Scope** (The target)

The scope resolves to a specific list of Repositories and or Business Applications. You can apply filters to narrow the scope, for example, to a single repository or application.

- **Triggers** (The timing)

Triggers define when and where a policy is evaluated. A policy can be evaluated in one or more of the following contexts:

- **Periodic scans**: Runs on a scheduled basis
- **Pull Requests (PRs)**: Runs whenever code is committed to a PR
- **Continuous Integration (CI) scans**: Runs during pipeline execution

The selected triggers determine where the policy logic is applied.

- **Actions** (The outcome)

Defines what happens when a condition is met within the scope at the triggered time.

- **Detect**: Create an issue
- **Prevent** : Block a pull request (PR) or fail a CI run (PR and CI triggers only)
- **Report**: Add a PR comment or CI report (PR and CI triggers only)

If multiple actions are selected for a single trigger, all selected actions are executed.

Example 7. Example

Parameters:

- **Condition**: Critical vulnerability
- **Trigger**: Pull request
- **Actions**: Prevent, Report, Detect

Result

When a pull request scan detects a critical vulnerability:

- The pull request is blocked
- A comment is added to the pull request
- An issue is created for the detected vulnerability

Multiple policies matching a single finding

If a single finding matches multiple policies, actions from all matching policies are evaluated. Each action is executed only once.

Example 8. Example

Parameters:

- **Policy A:** Prevent (Block PR), Detect (Create an issue)
- **Policy B:** Detect (Create an issue)
- **Policy C:** Report (PR comment)

Result:

When the finding is detected:

- The pull request is blocked
- One issue is created
- A PR comment is added

NOTE:

When an issue is created, only the first policy ID that triggered the issue is associated with it.

Viewing blocking policies

When a CI scan is blocked by a policy, the blocking policy is shown in the **CI Scan Health** view under the **Blocking Policy** column.

The same behavior applies to pull request (PR) scans. When a PR is blocked, the blocking policy appears in the PR scan details.

Third-party scanner limitations

Policies applied to third-party scanners are currently supported with the following limitations:

- Only **periodic scan** triggers are supported
- Only the Create an Issue action is available

6.11.4.1 | Create code security policies

Code policies automatically prevent and detect security risks across your code-to-cloud workflow, ensuring threats are stopped before they reach production. They address secrets, CVE vulnerabilities, IaC misconfigurations, code weaknesses, and package-related risks—including operational risks and license violations. These policies support results from third-party scanners and apply to periodic scans, pull requests, and CI-triggered scans.

Prioritize risk with context

To ensure security efforts are focused and effective, you can integrate contextual information into your policies:

- **Code-to-cloud context:** Create policies to prioritize and act on issues that pose the greatest security risk to deployed production assets
- **Application context:** Align Scope-Based Access Control (SBAC) policies with an application-specific purpose and business sensitivity to ensure relevant and effective security enforcement across its lifecycle. For more information about creating application-scoped policies, refer to Scope user access to applications (Application SBAC)

Using runtime and application contexts focuses efforts on high-impact issues and reduce noise.

1. Under Modules, select Application Security → AppSec Policies → + Add Policy.
2. On the General step of the policy creation wizard.
 - a. Select Code Scanners (default) as the policy type.
 - b. Provide a unique policy name (required) and description.
 - c. Click Next.
3. On the Conditions step of the wizard, select Add Filters to define the conditions that apply to the policy. By default, the Finding Type field is displayed with all scanner types selected.
 - a. Modify the Finding Type (optional): Select one or more scanners. The policy will only evaluate findings generated by the selected scanners.
Scanner options include: Select All, Vulnerability, Secrets, IaC Misconfiguration, License, Operational Risk, Code Weakness.

Considerations:

- You can select multiple finding types (scanners) to broaden the policy's coverage, allowing it to apply to multiple security categories simultaneously. By default multiple scanners are selected, limiting the available filters
 - Selecting specific scanners narrows the policy scope to particular risk categories (such as Secrets) and unlocks the filtering attributes unique to the scanner on the Conditions screen
- b. Click AND to select an attribute to filter the findings for the selected Finding Type (scanner).

When applying filters to the selected finding types, the system's logic adapts based on the attributes you choose, creating a tailored policy that targets the specific risk patterns you are looking for.

Filter logic:

Using **AND** and **OR** operators allows you to broaden policy coverage while maintaining dedicated logic for each Finding Type, giving precise control over how different conditions are evaluated.

- **AND** logic within a bracket: All conditions in a single bracket must be met for the policy action to apply
- **OR** logic between brackets: Multiple brackets can be combined with the OR operator to define separate evaluation rules for different sets of conditions

Example 9. Example

In this example, the policy applies to findings that meet the first set of conditions (Secrets with Critical severity) or the second set (IaC with High severity).

```
[Finding Type = Secrets AND Severity = Critical]  
OR  
[Finding Type = IaC AND Severity = High]
```

c. Select a value for the filter.

d. Click Next.

NOTE:

For a detailed list of all available attributes and their values by scanner type, refer to Cortex Cloud Application Security code policy Condition attributes .

4. On the Scope step of the wizard, narrow the policy scope to relevant assets using one of these methods: Asset Types or Asset Groups.

a. If selecting Asset Types:

- i. An unfiltered list of Cortex Cloud Application Security assets in the inventory is displayed.
- ii. Narrow the scope of the assets to be checked by the policy: Select Add Filter → [Filter type] → [Filter value].

You can filter assets by:

- Category. Values: Application, Repository.

Application context: If you select Business Applications as the type of Category, a list of business applications is displayed, and the policy is scoped to evaluate these applications. You can narrow the scope of the policy to specific applications by applying additional filters, such as Business Application Names.

- Business Application Names. Value: String
- Business Application Criticality. Values: Select All, Critical, High, Medium, Low
- Application Business Owner. Values: String
- Repository Name: Values: String
- Repository ID: Values: String
- Is Public Repository: Values: Select All, Yes, No
- Has Deployed Assets (Runtime context): Values: Select All, Yes, No
- Has Internet-exposed deployed assets (Runtime context): Values: Select All, Yes, No
- Has deployed assets with access to sensitive data (Runtime context): Values: Select All, Yes, No
- Has deployed assets with privileged capabilities (Runtime context): Values: Select All, Yes, No
- Provider: Values: AWS Code Build, AWS Code Commit, Azure Repos, Bitbucket, Bitbucket Data Center, Circle CI, Cortex CLI, GitHub, GitHub Actions, GitLab, GitLab Self-Managed, HCP Terraform Tasks, HCP Terraform Enterprise Run Tasks, Jenkins

NOTE:

SBAC scope-based limitations do not apply to Asset Types

- If selecting Asset Groups: Select the asset groups against which this policy and its chosen detection rules will be evaluated. You can only select asset groups that you are assigned to you as part of your scope.

Application context (SBAC): For more information about Cortex Cloud Application Security Asset Groups, refer to SBAC Scope-based access control for Cortex Cloud Application Security.

NOTE:

When selecting Asset Groups, the code policy is evaluated only on the application and repository assets included in the group.

- Click Next.

NOTE:

For each application that matches the criteria, the policy is evaluated only on the repositories associated with that application.

- On the Triggers & Action step of the wizard.

- Select one or more trigger types that define when the condition will be evaluated. You must select at least one trigger.

- Periodic Scan: Automated security scans that run every 12 hours
- PR (Pull Request)Scan: Security scans triggered automatically by events in your version control system when a Pull Request (PR) or Merge Request (MR) is created or updated
- CI Scan: Security scans that run as part of your build pipeline, including checks on your code and configurations.

NOTE:

If you select only Code Weaknesses as the Finding Type in the condition step (indicating a third-party scanner), the PR and CI triggers are disabled, leaving only Periodic Scan enabled.

- Specify which actions to take when the policy detects its target risk. You must select at least one action for each selected trigger.

Periodic Scan	PR Scan	CI Scan
<ul style="list-style-type: none"> • Create an issue (required) • Override Severity 	<ul style="list-style-type: none"> • Block PR • PR Comment • Create an issue • Override Severity 	<ul style="list-style-type: none"> • Block CI • CLI Report • Create an issue • Override Severity

Table Legend

- Create an issue: Create an issue if policy conditions within the selected scope are met. You can control when an issue is created—for example, during periodic scans but not in pull request scans. You can also assign different severities based on the trigger, allowing lower severity for issues detected after a block compared to those found during periodic scans
- Block: Block a run if the policy conditions within the selected scope are met
- PR Comment: Automatically posts a comment on a pull request when the policy conditions for the selected trigger are met
- CLI Report: Enable reporting via CLI if policy conditions within the selected scope are met. Available when CI scan is selected as the evaluation method
- Override Severity: Manually override the default severity that the system assigns to the issue

- Click Next.

- On the Summary step of the wizard: Review the policy settings and click Done.

This step provides an overview of the configured policy, including its name and description, the configured scope, and a table of conditions, triggers, and actions. The table splits conditions by trigger, since not all selected triggers apply to all selected Finding Types—for example, Code Weaknesses. It also displays the user who created the policy and the creation date.

You can view the custom policy that you created in the general policies table on the AppSec Policies page.

Next step: Investigate and remediate issues

Investigate and remediate issues identified by your scanners to mitigate risks across your code, dependencies, and infrastructure. For more information, refer to Code Security scanners.

6.11.4.1.1 | Cortex Cloud Application Security code policy Condition attributes

Abstract

Learn about Application Security code policy Condition filters and attributes.

Condition attributes are data fields used to filter findings. The list includes common attributes - including the Finding Type attribute - as well as scanner-specific attributes, which are displayed after you select a scanner as the Finding Type. You can select multiple attributes, from both common and scanner-specific lists, to create precise filters and target policies more effectively.

Common Condition attributes

- Severity. Values: Select All, Critical, High, Medium, Low
- Backlog Status. Values: Select All, Backlog, New
 - For more information about Backlog Status, refer to Backlog baseline
- Respect Developer Suppression. Values: Select All, Yes, No. For more information on developer suppressions, see Developer Suppressions below
- Category. The top-level domain for organizing security findings. Values: Configuration, Vulnerability, Malware, Identity, Data, Code, Posture, Brand Protection
- Finding Type: See below for detailed information

Finding Type-specific attributes (scanner-specific)

The Finding Type filter allows you to select the scan types that the policy will apply to. Multiple selection is supported.

Finding Type properties include: Select All, Secrets, IaC Misconfigurations, Vulnerabilities, Licenses, Operational Risk and Code Weaknesses

After selecting Finding Type as the filter, you can further refine the policy criteria by selecting specific attributes of the finding type using the **AND** operator.

Example 10. Example

When License is selected as the finding type, these following package attributes are available for filtering: License Type, Package Deprecated, Package Maintained, Package Operational Risk and Package Popularity. If you select License Type, a list of selectable licenses (such as Artistic 2.0, APSL) are displayed.

The following table describes the scanner type attributes.

Attribute	Description
Secrets	<ul style="list-style-type: none"> • Severity <ul style="list-style-type: none"> ◦ Logic: Equal to, greater than, less than ◦ Severity level: Critical, High, Medium, Low • AppSec Rule <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Rules: Multi-selection list of available detection rules or type-in field • Secret Validity <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Privileged, Valid, Invalid, Unknown • AppSec Rule Labels <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Labels: Multi-selection list of available labels

Attribute	Description
IaC misconfiguration	<ul style="list-style-type: none"> • Has an automated fix: <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Yes, No • AppSec Rule <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Rules: Multi-selection list of available detection rules • IaC tag <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Tags: [One or more tags, separated by commas or spaces] • AppSec Rule Labels <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Labels: [Multi-selection list of available labels or type-in field] • IaC Compliance <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: <ul style="list-style-type: none"> ▪ Compliance Standard ▪ Compliance Control

Attribute	Description
Vulnerabilities	<ul style="list-style-type: none"> • Risk Factors <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Critical severity, High severity, Medium severity, Has fix, Remote execution, DoS-Low, DoS-High, Recent vulnerability, Exploit exists in the wild, Exploit exists -POC, Attack complexity: low, Attack vector: network • CVE ID <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ CVE IDs: [Multiple CVE IDs can be entered, separated by commas or spaces] • CVE Fix Available Date: Range: Select a start and end date • CVE Publish Date: Range: Select a start and end time • CVSS Score <ul style="list-style-type: none"> ◦ Logic: Is equal to, is not equal to, is equal or greater than, is equal or less than ◦ Score: Number between 0-10 • EPSS <ul style="list-style-type: none"> ◦ Logic: Is equal to, is not equal to, is equal or greater than, is equal or less than ◦ Number between 0.00–1.00 • Has a fix: <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Yes, No • Is KEV <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Yes, No

Attribute	Description
	<ul style="list-style-type: none"> • Package Name <ul style="list-style-type: none"> ◦ Logic: Is, Is not, Contains, Does not contain ◦ Package Name: [Package name string] • Package Deprecated <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Yes, No • Package Maintained <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Infrequently Maintained, Moderately Maintained, Frequently Maintained • Package Popularity <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Low, Medium, High • Package Operational Risk <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Values: Select All, Low, Medium, High • Package Version <ul style="list-style-type: none"> ◦ Logic: Equal to, not equal to, greater than, less than ◦ Version: [Package version string]

Attribute	Description
License	<ul style="list-style-type: none"> • License Type <ul style="list-style-type: none"> ◦ Logic: Is, Is not ◦ Licenses: Multi-selection list of available license types • AppSec Rule: See above • AppSec Rule Label: See above • Package Deprecated: See above • Package Maintained: See above • Package Name: See above • Package Operational Risk: See above • Package Popularity: See above • Package Version: See above
Operational risk	<ul style="list-style-type: none"> • Popularity: See Package Popularity above • Maintained: See Package Maintained above • Deprecated: See Package Deprecated above • Package Name: See above • Package Version: See above

Attribute	Description
Code Weaknesses	<ul style="list-style-type: none"> • CWE: <ul style="list-style-type: none"> ◦ Logic: Contains, does not contain, equals, does not equal ◦ Values: [CWE ID (text string)] • Language: <ul style="list-style-type: none"> ◦ Logic: Contains, does not contain, equals, does not equal ◦ Values: [Supported program language, such as Java (text string)] • OWASP Category: <ul style="list-style-type: none"> ◦ Logic: Contains, does not contain, equals, does not equal ◦ Values: [Top 10 OWASP CWE categories (text string)] • Source: <ul style="list-style-type: none"> ◦ Logic: Contains, does not contain, equals, does not equal ◦ Values: [Text string]

Developer Suppressions

Apply developer suppressions through the Respect Developer Suppression attribute. Options: Select All, Yes, No.

- **No:** If you do not select the Respect Developer Suppressions attribute, no additional action is taken, and developer suppressions are not considered in the policy
- **Yes:** If you select the Respect Developer Suppressions attribute, a new condition is automatically added to each existing condition group, ensuring the policy respects developer suppressions

Precedence rule: If both a global suppression setting and a condition-level filter for Respect Developer Suppression are present, the condition-level filter will always take precedence during policy evaluation.

Build queries with Finding Types

You can select the Finding Type attribute at the beginning of a new query and also after each **OR** operator. When an AND operator follows a selected Finding Type, the available attributes for filtering will be limited to those relevant to that specific Finding Type. The OR operator, however, provides access to all available attribute options.

Example 11. Example

```
finding type = Vulnerabilities AND CVSS>9 AND has a fix = true) OR (finding type = Operational Risk AND maintenance = LOW) OR (finding type = IaC misconfiguration AND IaC tag = kuku
```

6.11.4.2 | Create IaC Drift Detection policies

IaC Drift Detection policies define the conditions under which configuration drift is identified, prioritized, and surfaced as security issues. By creating these policies, you control how the platform evaluates deviations between your IaC baseline and the live cloud configuration, ensuring that drift is detected consistently and aligned with your security and compliance requirements.

1. Navigate to Modules → Application Security → AppSec Policies → Add Policies.
2. On the General step of the wizard.
 - a. Select Drift Detection Scanner.
 - b. Provide a policy name (required) and description (optional).
 - c. Click Next.
3. On the Conditions step of the wizard:, IaC Drift is selected as the default Finding Type. Select the IaC Drift attributes that define the policy. Options include:
 - AppSec Rule: Filters the policy scope to trigger only when the detected drift specifically violates a particular named security rule (such as S3-Bucket-Public-Read-Access). This allows fine-grained control over which security findings count as a policy violation
 - AppSec Rule Label: Filters the policy based on custom tags or categories assigned to the underlying security rule. This is useful for grouping similar risks (such as grouping all rules labeled Networking-Exposed)
 - Backlog Status: Filters the policy based on whether the resulting issue is categorized as New (a recently introduced misconfiguration) or Backlog (pre-existing technical debt)
 - Severity: Filters the policy to trigger only when the drift is classified above a certain risk level (for example, only trigger an issue if the drift severity is Critical or High)
 - IaC Tag: Filters the policy based on custom metadata tags found within the IaC code (such as checking only resources tagged with Environment: Production). This is useful for isolating high-priority business applications
4. On the Scope step of the wizard.

Limit policy evaluation by defining which code repositories or business applications are scanned and which cloud resources are evaluated for drift. At evaluation time, the policy engine constructs the runtime resource set strictly based on the defined scope.

- If selecting Asset Types: Narrow the scope of the policy using these attributes:
 - Provider: Cloud service provider (such as AWS), version control systems (such as GitHub) and CI tools (such as Jenkins)
 - Category: Values include Application, Repository
 - Business Application Names: Enter a custom text string
 - Application Business Criticality: Define the policy according to a business application's criticality. Values: Select All, Critical, High, Medium, Low
 - Application Business Owner: Enter a custom text string
 - Cloud Account: The cloud account number or name
 - Cloud Region: The specific cloud region
 - Repository Name: The name of a specific repository. Enter a custom text string
 - Repository ID: The repository identifier
 - Is public repository: Whether the repository is public or if the application is built from public source code. Values: Select All, Yes, No
 - Has deployed assets: Whether the asset (repository or application) includes deployed assets. Values: Select All, Yes, No
 - Has internet-exposed deployed assets: Whether the asset (repository or application) includes deployed assets. Values: Select All, Yes, No
 - Has deployed assets with access to sensitive data: Whether the asset (repository or application) includes deployed assets with access to sensitive data (such as credit cards). Values: Select All, Yes, No
 - Has deployed assets with privileged capabilities: Whether the asset (repository or application) includes deployed assets with privileged capabilities (such as Admin permissions). Values: Select All, Yes, No

At evaluation time, only cloud resources that fall within this Cloud Asset scope are included, and issues are created exclusively for resources in this defined set.

- If selecting Asset Groups: Choose Asset Groups to apply mandatory runtime constraints to members of existing, pre-defined asset collections. For IaC Drift, Asset Groups support filtering based on deployed cloud environment characteristics:
 - Provider (Cloud)
 - Cloud Account (realm)
 - Cloud Region

At evaluation time, the policy engine constructs the runtime resource set strictly based on these filters.

5. On the Triggers & Actions step of the wizard.

1. Periodic Scan is currently the only supported trigger for Drift Detection policies. Create an issue is the action configured by default.
 2. Select Next.
6. On the Summary step: Review the policy settings and click Done.

This step provides an overview of the configured policy, including its name and description, the configured scope, and a table of conditions, triggers, and actions. The table also displays the user who created the policy and the creation date.

You can view the custom policy that you created in the general policies table on the AppSec Policies page.

Next step: Investigate and remediate drift

Once your policies identify configuration drift, investigate the issue and apply remediation options to align your runtime environment with your IaC baseline. For more information, refer to IaC Drift Detection scans.

6.11.4.3 | Manage Cortex Cloud Application Security policies

Manage your custom Cortex Cloud Application Security policies to maintain an effective application security posture and adapt your security rules to evolving threats and requirements.

To manage policies, right-click on a policy in the table or select a policy and then select the menu in the side panel. The following actions are available:

- Edit: Redirects to the policy wizard, allowing you to modify the policy

NOTE:

You cannot edit out-of-the-box (OOTB) policies.

- Duplicate: Clone default policies as templates for creating custom policies. When this option is selected, the policy wizard is displayed with the original policy configurations, allowing you to modify them as required

NOTE:

The duplicated policy will include the word "clone" in its name and must be renamed.

- Disable: Enable or deactivate the policy without deleting it. Future scans will not trigger the policy, but existing issues detected by the policy will persist. Bulk actions are supported, allowing you to disable multiple policies simultaneously
- Enable: Activates the policy configuration, making it active for all subsequent scans and enforcement gates.
- Delete policy (Custom policies only): Permanently remove the policy from your environment. Issues detected by the policy will persist. Bulk deletions are supported

6.12 | Application Security Rules

Abstract

AppSec rules detect security threats using predefined criteria based on standard compliance frameworks and best practices. Custom rules are supported.

Cortex Cloud Application Security rules are designed to detect security threats within your application security environment, which includes the various components, configurations, and interactions within your application that can potentially introduce vulnerabilities or pose risks to its security. Cortex Cloud Application Security rules identify and flag issues based on predefined criteria, ensuring that potential threats are proactively detected and addressed to enhance the overall security posture of your application.

Cortex Cloud Application Security rules cover a wide range of security best practices, inspired by compliance frameworks such as PCI, GDPR, ISO 27001:2013, and NIST, as well as additional best practices beyond regulatory requirements.

In addition to default rules, you can create custom rules to tailor to your specific security requirements.

NOTE:

Out-of-the-box rules cannot be modified directly. However, you can create a custom rule by cloning the existing one. This allows you to make changes to the original rule according to your requirements. Refer to [Manage Cortex Cloud Application Security custom rules](#) for more information.

6.12.1 | Roles and permissions

These user roles and permissions are required for AppSec rules:

- Users with Rules Read permission can view detection rules
- Users with rules Rules View/Edit permissions can create and modify detection rules
- Both AppSec Admins and DevSecOps roles only have **view** permissions

6.12.2 | Rules inventory

To access Cortex Cloud Application Security rules, under Modules select Application Security → AppSec Rules (under Policy Management).

The Cortex Cloud Application Security rules inventory includes both out-of-the-box and custom rules. The following list describes rules fields/properties displayed in the inventory table. By default, rules are displayed according to severity and then alphabetically. Details are provided for properties that require explanation. You can enable or disable rules by checking the box next to the rule name in the table.

Attribute/Property	Description
Rule Name	The rule name

Attribute/Property	Description
Rule Description	A description of the rule
Severity	The severity level assigned to findings identified by the rule
Scanner	The type of Application Security scanner configured to detect violations of this rule
Policies Count	The amount of policies that included the rule in its configuration
Last modified	The date and time when the rule was most recently updated
Labels	Labels assigned to the rule
Framework/Language	The framework or language that the detection rule applies to (for example, GitHub, Terraform, JavaScript)
Issues Count	The amount of issues generated from findings detected by the rule. Select the value to navigate directly to the dedicated Issues page for the corresponding scan type, Filtered by the issues detected by the rule
Mapped Cloud Security Rule	The corresponding Cloud Security Posture Management (CSPM) rule ID that is linked to this Application Security rule. This mapping enables unified policy enforcement and ensures consistent security governance from code to cloud

How to search for Cortex Cloud Application Security rules

Use filters to find specific rules or categories.

Example 12. Examples

- To filter rules relating to Secrets, select filter icon → Scanner (from the Select field) → Secrets (from the Value field).
- To view custom rules only, select Mode from the Select field, not equals as the operator, and Out-of-the-box as the value
- Sort rules according to their attributes, such as issue severity, to prioritize remediation efforts

6.12.3 | Create custom Cortex Cloud Application Security rules

Create custom Cortex Cloud Application Security rules to tailor your security measures to address specific and unique threats to your organization that are not covered by default rules. Custom rules run across branch periodic, PR, and CI scans.

Use the custom rule builder to create rules from scratch or clone and customize existing rules, enabling you to tailor them to meet your specific requirements effectively.

You can create custom rules for:

- **Secrets** scans
- **IaC** scans. Supported frameworks include Terraform, TFPlan (with automatic application of Terraform custom rules), CloudFormation, Kubernetes, Bicep, Helm, Kustomize, Helm and ARM. These scans also apply to serverless deployments

To create custom rules:

1. Under Modules select Application Security → AppSec Rules (under Policy Management) → Add Rule.
2. On the New Rule dialog box:
 - a. Provide these details:

- Name: The name of the rule
- Impact: Describes the potential impact of a detected violation. This description is displayed in the Issues page as well as PR Comments
- Severity(required): Determines the priority level assigned to findings identified by the rule
- Scanner (required): The type of scanner to be used to detect issues based on the rule
- Category (required): Refines the scope of the rule. Values include General, IAM, Monitoring, Networking, Public, Storage, Compute, Kubernetes, Logging, and AI/Machine Learning
- Subcategory (required): Further refines the scope of the rule by specifying particular attributes within the selected category

After selecting the category and sub-category, a description of the rule finding that will be based on these selections is displayed.

Example 13. Example

If IAM is the category, and Overly Permissive is the sub-category, the finding type description is: "Based on the categorization, finding type will be "Overly permissive IAM policies configuration found in infrastructure as code"".

- Framework: The framework or language that the rule is designed to apply to, such as Terraform, CloudFormation and ARM
- Labels: Assign tags to rules to help categorize, filter, and organize them for easier identification and management
- Mapped Runtime Rule: Select a runtime rule from the menu to map to your custom build time rule. This enhances your code-to-cloud visibility, allowing you to prioritize findings which are detected in both build and deployed environments

b. Click Next.

3. On the Rule Configuration screen.

a. Provide your rule definition as YAML.

NOTE:

See Configure YAML file properties below for more details.

b. Validate the code: Click Validate Code.

c. Provide suggested remediation in AsciiDoc format.

d. Click Done.

The rule is displayed in the rules inventory table.

NOTE:

Scanning/testing behavior is not supported.

6.12.4 | Manage Cortex Cloud Application Security custom rules

You can manage Cortex Cloud Application Security detection rules to customize and optimize your security configurations according to your specific needs and preferences: On the AppSec Rules inventory, right-click on a rule or click to open the side panel → select an option:

- Edit: Opens the Edit Rule wizard, allowing you to manage existing rules
- Duplication: Opens the selected rule in a New Rule dialog box, allowing you to save a copy of the rule. This allows you to customize default rules according to your requirements

6.12.5 | Configure YAML file properties

You can leverage YAML templates to create complex rules tailored to specific compliance or security requirements. Cortex Cloud Application Security rules support attribute-based and connection-state rules.

Rule attributes properties (YAML)

The following YAML attributes are used to define the properties of the rules.

- provider: Specifies the cloud provider or source for the resources
- definition: Contains the logic and conditions for the rule, including attributes, operators, and resource connections
- resource_type: Defines the type of the specific cloud resource
- cond_type: Represents the condition type for applying the rule. Options: attributes, connection, filter, resource
- attribute: Refers to the specific attribute or property of the cloud resource being evaluated
- value: Represents the value that the attribute of the cloud resource should meet for the rule condition

Attribute-based rules

Attributes define resource property configurations. The YAML syntax for attribute configurations aligns with the framework targeted by the rule, such as Terraform, to define the desired resource state. Cortex Cloud Application Security IaC rules identify and flag any resource that deviates from this defined state as a violation.

Each resource must include one of the following conditions:

- **Contain the specified attribute values.** For example, if a rule states that the `region` attribute must be `us-west-2`, then a resource will only pass this part of the rule if it includes the `region` attribute, and the value of that attribute is `us-west-2`
- **Match the attribute's presence.** For example, if a rule states "the `encryptionEnabled` attribute must be present," then a resource will only pass if it includes the `encryptionEnabled` attribute, regardless of its value
- **Match the attribute's absence:** For example, if a rule says "the `publicAccessAllowed` attribute must be absent," then a resource will only pass if it does not include the `publicAccessAllowed` attribute

Example 14. EXAMPLE

In this example, the attribute check flags any `aws_redshift_cluster` resource where the `automated_snapshot_retention_period` is not 0.

```
definition:  
  cond_type: "attribute"  
  resource_types:  
    - "aws_redshift_cluster"  
  attribute: "automated_snapshot_retention_period"  
  operator: "not_equals"  
  value: "0"
```

Supported Operators: Attribute operators apply differently based on the scan type:

- For **IaC** scans: All attribute operators are supported
- For **Secrets** scans: You must implicitly use the `regex` operator. Even if `regex` is not explicitly defined, pattern matching is applied automatically. For example, in the following secret rule, `regex` is implicitly applied:

```
cond_type: "secrets"  
value:  
  - "[A-Za-z0-9]{8,20}"  
  - "my-super-secret-password-regex"
```

The table below explains how to use attributes with matching keys and values.

Operators	Values
Equals	<code>equals</code>
Not Equals	<code>not_equals</code>
Regex Match	<code>regex_match</code>
Not Regex Match	<code>not_regex_match</code>
Exists	<code>exists</code>
Not Exists	<code>not_exists</code>
One Exists	<code>one_exists</code>
Any	<code>any</code>

Operators	Values
Contains	contains
Not Contains	not_contains
Within	within
Starts With	starting_with
Not Starts With	not_starting_with
Ends With	ending_with
Not Ends With	not_ending_with
Greater Than	greater_than
Greater Than Or Equal	greater_than_or_equal
Less Than	less_than
Less Than Or Equal	less_than_or_equal
Subset	subset
Not Subset	not_subset
Intersects	intersects
Not Intersects	not_intersects

Limitation of nesting in NOT blocks

Nesting connection condition types within a **NOT** block is not currently supported. The following example displays an unsupported 'NOT' block for connection condition types.

Example 15.

```
definition:  
  not:  
    cond_type: "connection"  
    resource_types:  
      - "aws_elb"  
    connected_resource_types:  
      - "aws_security_group"  
    operator: "exists"
```

Using JSONPath with operators

Operators within this system support advanced attribute targeting through *JSONPath* expressions. To apply an operator to a JSONPath result, prefix the operator with `jsonpath_`. This allows for flexible and precise data extraction and comparison. For example: `jsonpath_length_equals` or `jsonpath_length_exists`.

Connection-based rules

Connection state in a rule defines whether resources of different types are connected or disconnected. This helps enforce security controls and architectural constraints by specifying allowed or prohibited relationships between resources.

Example 16. EXAMPLE

In this example, `aws_lb` and `aws_elb` must be connected to `aws_security_group` or `aws_default_security_group` to be compliant.

```
definition:  
  cond_type: "connection"  
  resource_types:  
    - "aws_elb"  
    - "aws_lb"  
  connected_resource_types:  
    - "aws_security_group"  
    - "aws_default_security_group"  
  operator: "exists"
```

The table below explains how to use Connection State types with matching keys and values.

Key	Type	Value
<code>cond_type</code>	string	A connection must exist between the specified resources

Key	Type	Value
resource_type	collection of strings	Use either all or [included resource type from list]
connected_resource_types	collection of strings	Use either all or [included resource type from list]
operator	string	exists/not exists

The table below explains how to use Connection State operators:

Connection State Operators	Value
Exists	exists
Not Exists	not_exists

Logical operators (AND/OR)

A rule can include layers of defined attributes, connection state, or both. To define the relationship between them, use **AND/OR** logical operators. You can customize the attributes, connection state, or both across multiple layers.

Example 17.

In this example, the attribute property is evaluated using **OR** logic to enforce compliance checks for ensuring all AWS databases have a backup policy.

Read more...

```

metadata:
  name: "Ensure all AWS databases have Backup Policy"
  guidelines: "In case of non-compliant resource – add a backup policy configuration for the resource"
  category: "storage"
  severity: "medium"
scope:
  provider: "aws"
definition:
  or:
    - cond_type: "attribute"
      resource_types:
        - "aws_rds_cluster"
        - "aws_db_instance"
      attribute: "backup_retention_period"
      operator: "not_exists"
    - cond_type: "attribute"
  
```

```

resource_types:
- "aws_rds_cluster"
- "aws_db_instance"
attribute: "backup_retention_period"
operator: "not_equals"
value: "0"
- cond_type: "attribute"
resource_types:
- "aws_redshift_cluster"
attribute: "automated_snapshot_retention_period"
operator: "not_equals"
value: "0"
- cond_type: "attribute"
resource_types:
- "aws_dynamodb_table"
attribute: "point_in_time_recovery"
operator: "not_equals"
value: "false"
- cond_type: "attribute"
resource_types:
- "aws_dynamodb_table"
attribute: "point_in_time_recovery"
operator: "exists"

```

Example 18.

In this example, both **AND/OR** logical operators are utilized to evaluate both attribute and connection state properties in order to enforce compliance checks for ensuring that all Application Load Balancers (ALBs) are only connected to HTTPS listeners.

Read more...

```

metadata:
  name: "Ensure all ALBs are connected only to HTTPS listeners"
  guidelines: "In case of non-compliant resource – change the definition of the listener/listener_rule protocol value into HTTPS"
  category: "networking"
  severity: "high"
scope:
  provider: "aws"
definition:
  and:
    - cond_type: "filter"
      value:
        - "aws_lb"
      attribute: "resource_type"
      operator: "within"
    - cond_type: "attribute"
      resource_types:
        - "aws_lb"
      attribute: "load_balancer_type"
      operator: "equals"
      value: "application"
  - or:
    - cond_type: "connection"
      resource_types:
        - "aws_lb"
      connected_resource_types:
        - "aws_lb_listener"
      operator: "not_exists"
    - and:
      - cond_type: "connection"
        resource_types:
          - "aws_lb"
        connected_resource_types:
          - "aws_lb_listener"
        operator: "exists"

```

```

- cond_type: "attribute"
  resource_types:
    - "aws_lb_listener"
      attribute: "certificate_arn"
      operator: "exists"
- cond_type: "attribute"
  resource_types:
    - "aws_lb_listener"
      attribute: "ssl_policy"
      operator: "exists"
- cond_type: "attribute"
  resource_types:
    - "aws_lb_listener"
      attribute: "protocol"
      operator: "equals"
      value: "HTTPS"
- or:
  - cond_type: "attribute"
    resource_types:
      - "aws_lb_listener"
        attribute: "default_action.redirect.protocol"
        operator: "equals"
        value: "HTTPS"
  - cond_type: "attribute"
    resource_types:
      - "aws_lb_listener"
        attribute: "default_action.redirect.protocol"
        operator: "not_exists"
- or:
  - cond_type: "connection"
    resource_types:
      - "aws_lb_listener_rule"
    connected_resource_types:
      - "aws_lb_listener"
    operator: "not_exists"
- and:
  - cond_type: "connection"
    resource_types:
      - "aws_lb_listener_rule"
    connected_resource_types:
      - "aws_lb_listener"
    operator: "exists"
  - or:
    - cond_type: "attribute"
      resource_types:
        - "aws_lb_listener_rule"
          attribute: "default_action.redirect.protocol"
          operator: "equals"
          value: "HTTPS"
    - cond_type: "attribute"
      resource_types:
        - "aws_lb_listener_rule"
          attribute: "default_action.redirect.protocol"
          operator: "not_exists"

```

Example 19.

In this example, **OR** logic is applied to custom secrets defined as part of a policy aiming to enforce security measures by restricting the addition of certain types of secrets.

Read more...

```

metadata:
  name: "My Secret"
  guidelines: "Don't add secrets"
  category: "secrets"
  severity: "high"
definition:

```

```
cond_type: "secrets"
value:
- "[A-Za-z0-9]{8,}"
- "my-super-secret-password-regex"
```

6.13 | Application Security CLI

The Application Security CLI, part of the Cortex CLI, allows developers and security teams to integrate security checks directly into their application development workflows.

NOTE:

For detailed information about the Cortex Cloud CLI, refer to [Cortex CLI](#).

The Code Security CLI supports the following scan types:

- **Secrets:** Identifies exposed sensitive secrets within your codebase
- **Infrastructure-as-Code (IaC):** Analyzes infrastructure configuration files to detect potential security misconfigurations
- **Software Composition Analysis (SCA):** Performs vulnerability detection in third-party dependencies, assesses their license compliance and their package operational risk

In addition, the Code Security CLI serves as the integration mechanism for security scanning within supported CI tools such as Jenkins, GitHub Actions, and others. This is achieved by adding a code snippet containing the CLI command into the configuration files of your CI tool when integrating the CI tool with Cortex Cloud. It acts as a wrapper, enabling security scanning within your pipelines, and direct upload of results to the platform.

The CLI supports the following outputs:

- json
- spdx
- cli
- junitxml
- sarif
- cyclonedx
- cyclonedx_json

Code Security CLI scan behavior and output

- Scans generate assets (see Code Security assets, issues, and findings)
- If one scanner (such as Secrets) fails, the other scanners will continue to run and produce results
- Scan failures trigger an error message indicating the scanner that failed
- The Code Security CLI provides these output modes for flexible management and viewing of scan results:
 - Upload to platform: `--upload-mode = true` (default). Uploads scan results directly to the platform for centralized analysis and management
 - Upload findings only. `--upload-mode = false` (default). Upload findings, but without including the actual source code content. This prevents raw source code from leaving your local environment or being stored on the platform
 - CLI output only: `upload = false` (default). View scan results directly in your command-line interface without being uploaded to the platform

For more information about the output flags, refer to Cortex CLI Cortex Cloud Application Security command line reference.

Authentication

To authenticate the Code Security CLI, choose one of the following methods:

- **Using command-line flags:** Provide authentication details directly with your commands

The following flags are required to authenticate the Code Security CLI:

- `--api-base-url: [$CORTEX_API_BASE_URL]`
- `--api-key: [$CORTEX_API_KEY]`
- `--auth-id. [$CORTEX_AUTH_ID]`

For more information about these flags, refer to Cortex CLI common command line reference guide.

- **Using a `cortex.yaml` file:** Place your authentication details in a `cortex.yaml` file

Requirements

PREREQUISITE:

These requirements apply specifically to the Application Security CLI.

- **For the Cortex CLI binary:**

- Ensure you have Node.js v22 installed on your host machine before running any scans with the Cortex CLI. This is crucial to prevent runtime errors, as the CLI depends on Node.js for executing JavaScript analysis

NOTE:

- To check your version of Node.js, run `node -v`
- To download Node.js, refer to the official Node.js site
- For Linux OS systems, ensure that GLIBC (GNU C library) version 2.35 or greater is installed

NOTE:

This requirement does not apply when using the CLI as a container image.

- Ensure you have Cortex CLI version 0.15 or greater. Run `cortexcli -v` to verify your version
- **Permissions:** Ensure you have the required user permissions. Refer to Overview for more information
- Onboard and install the Cortex CLI. Refer to Connect Cortex CLI for more information

Configure proxy for the Code Security CLI

When operating the Code Security CLI within environments requiring internet access via a proxy server, you can configure the tool to route its traffic through your proxy using standard environment variables. For proxies that perform TLS inspection, you must also specify a CA certificate

- **Environment variables:** Set `HTTP_PROXY` and `HTTPS_PROXY` (or `http_proxy` and `https_proxy`) to your proxy address
- **CA Certificate:** Use the `--ca-certificate` flag or the `$CORTEX_CA_CERTIFICATE` environment variable to provide your CA certificate for proxies that perform TLS inspection. The flag is now global and must appear before `code scan`. It is currently limited to the Application Security CLI. You can either:
 - **Use the environment variable:**

```
cortexcli --api-base-url <API_URL> --api-key <API_KEY> --api-key-id <API_KEY_ID> --ca-certificate $CORTEX_CA_CERTIFICATE code scan --directory {{DIRECTORY}} --branch main --repo-id organization/repo-name --output json --output-file-path ./output.json
```
 - **Use the flag with a direct path:**

```
cortexcli --ca-certificate /path/to/cert.pem code scan --directory {{DIRECTORY}} --branch main --repo-id organization/repo-name --output json --output-file-path ./output.json
```
- **Skip certificate verification:** Use the `--no-cert-verify` flag or the `$CORTEX_NO_CERT_VERIFY` environment variable to skip SSL certificate validation. To enable, set the flag or environment variable to true. **Warning:** This configuration is insecure. Use only in development or testing environments where a valid CA certificate is unavailable

6.13.1 | Connect Cortex CLI

Connect Cortex CLI to scan supported Cortex Cloud modules and gain insights into your security posture, enabling you to identify, analyze and address potential risks.

PREREQUISITE:

- **System requirements:**

- **macOS** (Intel Core i7, such as Sequoia): To ensure all functionalities work correctly, you must install the vectorscan dependency via **Homebrew**, using this command: `brew install vectorscan`
- **RHEL 8.10 and Red Hat UBI9.** The following prerequisites must be met:
 - Install `patchelf`
 - Install `zstd`
- **Ubuntu 20** requires the `prefetch` utility
- **Ubuntu (for linux-amd64)** also requires the `libhyperscan5` library. To install, run `sudo apt install libhyperscan5`
- **Linux for AppSec Module:** Support is provided for systems meeting the following specifications:
 - **RHEL 10:** Kernel: 6.12, glibc: 2.39
 - **Debian 12:** Kernel: 6.1.27, glibc: 2.36
 - **Ubuntu:**
 - Version 18.04 - Kernel: 4.15, glibc: 2.27
 - Version 20.04 - Kernel: 5.4, glibc: 2.31
 - Version 22.04 - Kernel: 5.15, Glibc: 2.35
 - Version 24.04 - Kernel: 6.8, Glibc: 2.39
 - Windows: AMD 64 and ARM 64
- For cURL-based downloads:
 - `curl`
 - `jq`
 - On **Ubuntu/Debian-based Linux** distributions: `sudo apt-get install jq`
 - On **RedHat/CentOS/Fedora**: `sudo yum install jq`
 - **macOS** (using Homebrew): `brew install jq`
 - **Windows:**
 - Download the executable from `jq` GitHub releases
 - If Chocolatey is installed: `choco install jq`
- **Permissions:** Grant the user installing the CLI required permissions. For more information refer to Cortex CLI
- Best Practice (required for SCA vulnerability suppression):

- Run the CLI within your current working directory (<current_directory_path>). It is recommended to use the absolute file path for your current working directory
- Ensure that the --repo-id parameter includes the <repo_owner_name>/<repo_name> structure, with the <repo_name> matching the exact name of the directory

Example 20. Example

The present working directory is Users/test/<repo_name>. Therefore, the --repo-id parameter must be --repo-id <repo_owner_name>/<repo_name>, ensuring that <repo_name> precisely matches the directory name within the structure.

- For terminal actions performed by Cortex Cloud IDE extensions on Windows, Command Prompt (CMD) is the supported environment. PowerShell is not supported for these actions

1. Navigate to Settings → Data Sources & Integrations → + Add New → Show More → CI/CD.

TIP:

You can also locate your CI tool by typing its name (such as Jenkins) into the search bar on the Add Data Source or Integrations page after selecting + Add New.

2. Hover over Cortex CLI and click Connect.

TIP:

You can enter CLI in the search bar to locate the Cortex CLI tool.

3. In the Configure step of the integration wizard.

a. Select your operating system from the menu.

b. Download the CLI binary: copy (or download) the command provided in the wizard and paste into your terminal.

c. Click Next.

The Authenticate step of the wizard is displayed.

4. On The Authenticate step of the wizard.

a. Generate an API:

i. Select Generate API key.

- **IMPORTANT:** This option is required for CWP image scans
- This option creates a CLI role for the API key with CLI View/Edit options. It is recommended as it grants the API key permissions to not only access data, but also to upload or send data back
- If you do not select this option, the generated API key creates a CLI Read Only role with CLI View permissions only
- **Warning:** Using With upload results permissions may incur additional costs as per your license agreement

ii. Copy the the generated API Key ID and API key that are displayed in their respective fields.

iii. Copy and save the the generated API key from the Retrieve your API key field.

A code command is generated and displayed.

iv. Verify that the generated API key is displayed under the API Keys inventory.

NOTE:

Using an existing API Key (or verifying existing API Key permissions): If you are using an existing API key, verify it has CLI permissions. CLI View/Edit permissions correspond to selecting With upload results permissions, while CLI Read Only or View permissions corresponds to not selecting the With upload results permissions.

b. Download and save the CLI tool to your system:

i. Copy or download the provided code.

NOTE:

On macOS arm 64 architecture you must unpack the downloaded file to retrieve the executable.

ii. Replace \${API_KEY} in the code with your API key.

iii. Retrieve and paste the Cortex Cloud public API URL in the code: Navigate to Settings → API Keys (under Configurations) → click Copy API URL .

c. Run the command in your terminal.

d. Click Done.

5. Make the cortexcli file executable: run chmod +x cortexcli.

NOTE:

To add an additional CLI instance, navigate to Settings → Data Sources & Integrations → select the menu for your connected CLI instance → + New Instance, and repeat the onboarding steps.

Download and run the Cortex CLI

1. Download the CLI: Run curl -k -u \$CORTEX_API_ID:\$CORTEX_API_KEY --output ./cortexcli \$CORTEX_FQDN/api/v2/remote-li/{version}/{platform}/artifacts

2. Execute the CLI: Run chmod +x cortexcli.

3. Verify installation: Run cortexcli -v.

The CLI version is displayed.

Authentication

You can authenticate the Cortex CLI using one of two methods: command-line flags or an environment configuration file.

- **Using command-line flags:** Provide your API credentials and base URL directly in the command using the following flags

- `--api-base-url`: `[$CORTEX_API_BASE_URL]`
- `--api-key`: `[$CORTEX_API_KEY]`
- `--api-key-id` `[$CORTEX_KEY_ID]`

For more information about these flags, refer to Cortex CLI common command line reference guide.

- **Using an environment configuration file:** Instead of using flags, you can create an environment configuration file named `cortex.env`. Save this file in your working directory and add your credentials as variables:

- `CORTEX_API_KEY`: <api key id>
- `CORTEX_API_SECRET`: <secret>
- `CORTEX_API_BASE_URL`: <tenant URL>, for example `https://api-tenantname.paloaltonetworks.com/`

Cortex CLI usage

To execute a Cortex CLI scan, run `cortexcli [global flags] [module name] scan [module flags]`.

Command breakdown

- Global flags:
 - `--api-base-url <value>`
 - `--api-key <value>`
 - `--api-key-id <value>`
- `cortexcli` acts as the global option, establishing the environment for subsequent Cortex CLI commands
- Module name: Select the module (environment) to be scanned:
 - `api` for API Security. For more information about API Security scans, refer to Cortex CLI for API Security
 - `image` for CWP. For more information about CWP scans, refer to Cortex CLI for Cloud Workload Protection
 - `code scan` for Cortex Cloud Application Security. For more informations about Cortex Cloud Application Security refer to Cortex CLI for Code Security
- Module flags: The flags available for the selected command:
 - For flags common to all environments, refer to Cortex CLI common command line reference guide
 - For flags specific to CWP refer to Cloud Workload Protection command line reference
 - For flags specific to API Security, refer to Cortex CLI API Security command line reference guide
 - For flags specific to Cortex Cloud Application Security, refer to Cortex CLI Cortex Cloud Application Security command line reference

NOTE:

- For more information about CLI usage for CWP, refer to Cortex CLI for Cloud Workload Protection
- For more information about CLI usage for API Security, refer to Cortex CLI for API Security
- For more information about CLI usage for Cortex Cloud Application Security, refer to Cortex CLI usage for Cortex Cloud Application Security

6.13.2 | Cortex CLI usage for Cortex Cloud Application Security

To scan Cortex Cloud Application Security, run:

```
cortexcli --api-base-url <API URL> --api-key <API key from the "Authenticate" step in the CLI connector screen> --api-key-id <API Key ID> code scan --directory {{DIRECTORY}} --branch main --repo-id organization/repo-name - output json --output-file-path ./output.json
```

Command line reference

The command structure includes global flags which are used for authentication, and then specifies the module name and command specific to Cortex Cloud Application Security which are followed by dedicated flags unique to this module as well as flags common to all modules.

- **Global flags:** These flags are part of the initial `cortexcli` command and are necessary to authenticate and connect to Cortex Cloud
 - `--api-base-url`: (Required = true). The public facing API URL. Refer to Connect Cortex CLI for more information
 - `--api-key`: (Required = true). The Cortex Cloud API key generated when onboarding the CLI as a data source. Refer to Connect Cortex CLI for more information
 - `--api-key-id`: (Required = true). The Cortex Cloud API key ID generated when onboarding the CLI as a data source

For a comprehensive list of Cortex Cloud Application Security global flags, refer to Cortex CLI Cortex Cloud Application Security command line reference

- **Cortex Cloud Application Security specifics:** Following the global flags, the command specifies the module and the commands required for initiating a scan using the Cortex Cloud Application Security module:
 - `code scan`: Required - true. This command instructs the CLI to perform an Cortex Cloud Application Security scan.
 - For the optional flags, refer to the dedicated Cortex Cloud Application Security command line reference

CLI Usage Examples

- **Send output to a file:** Direct the command's output to a specified file instead of displaying it in the console


```
./cortexcli --api-base-url <BASE_URL> --api-key <API_KEY> --api-key-id <API_KEY_ID> code scan --branch <branch name> --repo-id <repo name> --directory <path> --output json --output-file-path <path>
```
- **Perform a scan without upload:** Run a scan for local analysis or testing without uploading the results to Cortex Cloud. This command runs a code scan and saves all standard output (human-readable format) to `scan_results.txt`

```
./cortexcli --api-base-url <BASE_URL> --api-key <API_KEY> --api-key-id <API_KEY_ID> code scan --upload-mode no-upload --branch <branch name> --repo-id <repo name> --directory <path>
```

Sample outputs

The `cortexcli` provides different options for how scan results are presented.

- **Standard output (stdout):** When no specific output format flags (such as `--output json` or `--output sarif`) are provided, the Cortex CLI will produce standard output directly to your terminal or console
- **JSON output:** To obtain the output of a scan command as a JSON file, specify the flags `--output json` `--output-file-path ./output.json`. This command will save the detailed scan results in JSON format to `output.json` in the current directory.

Supported flags

The Cortex Cloud Application Security CLI supports both common Cortex CLI and dedicated Cortex Cloud Application Security flags.

- For dedicated Cortex Cloud Application Security flags, refer to Cortex CLI Cortex Cloud Application Security command line reference
- For common flags, refer to Cortex CLI common command line reference guide

6.13.3 | CLI pipeline code snippets

You can integrate the Cortex CLI directly into your CI/CD pipelines to enable automated code scans by adding code snippets to your build script or pipeline configuration, such as a **YAML** or **Groovy** file. Both **ARM** and **AMD** architectures are supported, ensuring you can scan your codebase regardless of your runner's environment.

PREREQUISITE:

User permissions: Ensure the user performing the integration has permissions to edit pipeline configurations (such as YAML files).

User permissions: Ensure the user performing the integration has permissions to edit pipeline configurations (such as YAML files).

You must replace placeholder variables with your own credentials and environment-specific details.

AWS CodeBuild

- For AMD architecture

```
version: 0.2
env:
  variables:
    CORTEX_API_URL: <your_cortex_api_url>
    CORTEX_CLI_VERSION: "0.13.14"
  secrets-manager:
    CORTEX_API_KEY: "CORTEX_API_KEY"
    CORTEX_API_KEY_ID: "CORTEX_API_KEY_ID"
phases:
  install:
    commands:
      - apt-get update
      - apt-get install -y curl jq git
  pre_build:
    commands:
      - echo "Getting repo name"
      - export CODEBUILD_ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
      - export CODEBUILD_GIT_BRANCH="$(git symbolic-ref HEAD --short 2>/dev/null)"
      - |
        if [ "$CODEBUILD_GIT_BRANCH" = "" ] ; then
          export CODEBUILD_GIT_BRANCH="$(git rev-parse HEAD | xargs git name-rev | cut -d' ' -f2 | sed 's/remotes\|origin\///g')";
        fi
      - export CODEBUILD_PROJECT=${CODEBUILD_BUILD_ID%:$CODEBUILD_LOG_PATH}
      - echo "Downloading cortexcli"
      - |
        crt_x_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
          -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
          -H "Authorization: ${CORTEX_API_KEY}")
        crt_x_url=$(echo "$crt_x_resp" | jq -r ".signed_url")
        curl -o cortexcli "$crt_x_url"
        chmod +x cortexcli
        ./cortexcli --version

  build:
    commands:
      - |
        ./cortexcli \
          --api-base-url "${CORTEX_API_URL}" \
          --api-key "${CORTEX_API_KEY}" \
          --api-key-id "${CORTEX_API_KEY_ID}" \
          code scan \
          --directory "$(pwd)" \
          --repo-id $CODEBUILD_ACCOUNT_ID/$CODEBUILD_PROJECT \
          --branch $CODEBUILD_GIT_BRANCH \
          --source AWS_CODE_BUILD \
          --create-repo-if-missing
artifacts:
  files:
    - '**/*'
```

- For ARM architecture

```
version: 0.2
env:
  variables:
    CORTEX_API_URL: <your_cortex_api_url>
    CORTEX_CLI_VERSION: "0.13.16"
  secrets-manager:
    CORTEX_API_KEY: "CORTEX_API_KEY"
    CORTEX_API_KEY_ID: "CORTEX_API_KEY_ID"
phases:
  install:
```

```

commands:
  - apt-get update
  - apt-get install -y curl jq git

pre_build:
  commands:
    - echo "Getting repo name"
    - export CODEBUILD_ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    - export CODEBUILD_GIT_BRANCH="$(git symbolic-ref HEAD --short 2>/dev/null)"
    - |
      if [ "$CODEBUILD_GIT_BRANCH" = "" ] ; then
        export CODEBUILD_GIT_BRANCH="$(git rev-parse HEAD | xargs git name-rev | cut -d' ' -f2 | sed 's/remotes\|origin\//g')"
      fi
    - export CODEBUILD_PROJECT=${CODEBUILD_BUILD_ID%:$CODEBUILD_LOG_PATH}
    - echo "Downloading cortexcli"
    - |
      crttx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}")
    - crttx_url=$(echo "$crttx_resp" | jq -r ".signed_url")
    - curl -o cortexcli "$crttx_url"
    - chmod +x cortexcli
    - ./cortexcli --version

build:
  commands:
    - |
      ./cortexcli \
        --api-base-url "${CORTEX_API_URL}" \
        --api-key "${CORTEX_API_KEY}" \
        --api-key-id "${CORTEX_API_KEY_ID}" \
        code scan \
        --directory "$(pwd)" \
        --repo-id $CODEBUILD_ACCOUNT_ID/$CODEBUILD_PROJECT \
        --branch $CODEBUILD_GIT_BRANCH \
        --source AWS_CODE_BUILD \
        --create-repo-if-missing

artifacts:
  files:
    - '**/*'

```

Azure Pipelines

- For AMD architecture

```
trigger:
  branches:
    include: ['*']
pr:
  branches:
    include: ['*']
pool:
  vmImage: ubuntu-latest
variables:
  CORTEX_API_URL: <your_cortex_api_url>
  MIN_LOG_LEVEL: "DEBUG"
steps:
- checkout: self
  clean: true
- task: NodeTool@0
  displayName: "Use Node.js 22.x"
  inputs:
    versionSpec: "22.x"
- bash: |
    set -euo pipefail
    sudo apt-get update
    sudo apt-get install -y --no-install-recommends jq ca-certificates curl
    BASE="${CORTEX_API_URL%/*}"
    URL="$BASE/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=amd64"
    set +x
    CRTX_URL=$(curl -fsSL "$URL" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
    set -x
    curl -fsSL -o cortexcli "$CRTX_URL"
    chmod +x cortexcli
  displayName: "Download cortexcli (amd64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
- bash: |
    set -euo pipefail
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "$(Build.SourcesDirectory)" \
      --repo-id "$(Build.Repository.Name)" \
      --branch "$(Build.SourceBranchName)" \
      --source "CORTEX_CLI" \
      --create-repo-if-missing
  displayName: "Cortex CLI Code Scan"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
  MIN_LOG_LEVEL: $(MIN_LOG_LEVEL)
```

- For ARM architecture

```
trigger:
  branches:
    include: ['*']
pr:
  branches:
    include: ['*']
variables:
```

```

CORTEX_API_URL: <your_cortex_api_url>
pool:
  name: arm
  demands:
    - Agent.OS -equals Linux
steps:
- checkout: self
  clean: true
- task: NodeTool@0
  displayName: "Use Node.js 22.x"
  inputs: { versionSpec: "22.x" }
- bash: |
    set -euo pipefail
    BASE="${CORTEX_API_URL%/*}"
    URL="$BASE/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=arm64"
    set +x
    CRTX_URL=$(curl -fsSL "$URL" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
    set -x
    curl -fsSL -o cortexcli "$CRTX_URL"
    chmod +x cortexcli
  displayName: "Download cortexcli (arm64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)
- bash: |
    set -euo pipefail
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "$(Build.SourcesDirectory)" \
      --repo-id "$(Build.Repository.Name)" \
      --branch "$(Build.SourceBranchName)" \
      --source "CORTEX_CLI" \
      --create-repo-if-missing
  displayName: "Cortex CLI Code Scan (ARM64)"
env:
  CORTEX_API_URL: $(CORTEX_API_URL)
  CORTEX_API_KEY_ID: $(CORTEX_API_KEY_ID)
  CORTEX_API_KEY: $(CORTEX_API_KEY)

```

- For AMD architecture

```

image: ubuntu:24.04
clone:
  depth: full
pipelines:
  default:
    - step:
        name: Cortex CLI Code Scan (Hosted AMD64)
        script:
          - set -euo pipefail
          - apt-get update && apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip
file
  - curl -fsSL https://deb.nodesource.com/setup_22.x | bash -
  - apt-get install -y nodejs
  - node -v && npm -v
  - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
"$CORTEXCLI_HOME"
  - |
    CRTX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
    curl -fsSL -o cortexcli "$CRTX_URL"
    chmod +x cortexcli
    ./cortexcli --version
  - |
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "${BITBUCKET_CLONE_DIR}" \
      --repo-id "${BITBUCKET_REPO_FULL_NAME}" \
      --branch "${BITBUCKET_BRANCH}" \
      --source "CORTEX_CLI" \
      --create-repo-if-missing \
      --upload-mode no-upload
artifacts:
  - cortexcli

```

- For ARM architecture

```

image: node:22-bookworm

pipelines:
  default:
    - step:
        name: Cortex CLI Code Scan
        runs-on:
          - self.hosted
          - linux.arm64
        script:
          - set -euo pipefail
          - apt-get update && apt-get install -y --no-install-recommends curl jq ca-certificates file
          - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
"$CORTEXCLI_HOME"

          - |
            set +x
            CRTX_URL=$(curl -fsSL "${CORTEX_API_URL%}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
              -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
              -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
            set -x
            curl -fsSL -o cortexcli "$CRTX_URL"

```

```
chmod +x cortexcli
./cortexcli --version

# Run the scan
- |
  ./cortexcli \
    --api-base-url "${CORTEX_API_URL}" \
    --api-key "${CORTEX_API_KEY}" \
    --api-key-id "${CORTEX_API_KEY_ID}" \
    code scan \
    --directory "${BITBUCKET_CLONE_DIR}" \
    --repo-id "${BITBUCKET_REPO_FULL_NAME}" \
    --branch "${BITBUCKET_BRANCH}" \
    --source "CORTEX_CLI" \
    --create-repo-if-missing
artifacts:
- cortexcli
```

- For AMD architecture

```

version: 2.1
jobs:
  cortex-code-scan:
    docker:
      - image: cimg/node:22.17.0 # Replace with a suitable image or executor
    environment:
      CORTEX_API_URL: <your_cortex_api_url>
    steps:
      - checkout
      - run:
          name: Download cortexcli
          command: |
            set -x
            crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
              -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
              -H "Authorization: ${CORTEX_API_KEY}")
            crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
            curl -o cortexcli $crtx_url
            chmod +x cortexcli
            ./cortexcli --version
      - run:
          name: Run Cortex CLI Code Scan
          command: |
            ./cortexcli \
              --api-base-url "${CORTEX_API_URL}" \
              --api-key "${CORTEX_API_KEY}" \
              --api-key-id "${CORTEX_API_KEY_ID}" \
              code scan \
              --directory "$(pwd)" \
              --repo-id "${CIRCLE_PROJECT_USERNAME}/${CIRCLE_PROJECT_REPONAME}" \
              --branch "${CIRCLE_BRANCH}" \
              --source "CIRCLE_CI" \
              --create-repo-if-missing
    workflows:
      cortex-scan-workflow:
        jobs:
          - cortex-code-scan:
              context: cortex-secrets

```

- For ARM architecture

```

version: 2.1
jobs:
  cortex-code-scan:
    docker:
      - image: <Replace with image supporting node js version 22 or higher>
    environment:
      CORTEX_API_URL: <your_cortex_api_url>
    steps:
      - checkout
      - run:
          name: Download cortexcli
          command: |
            set -x
            crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
              -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
              -H "Authorization: ${CORTEX_API_KEY}")
            crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
            curl -o cortexcli $crtx_url
            chmod +x cortexcli
            ./cortexcli --version
      - run:

```

```
name: Run Cortex CLI Code Scan
command: |
  ./cortexcli \
    --api-base-url "${CORTEX_API_URL}" \
    --api-key "${CORTEX_API_KEY}" \
    --api-key-id "${CORTEX_API_KEY_ID}" \
    code scan \
    --directory "$(pwd)" \
    --repo-id "${CIRCLE_PROJECT_USERNAME}/${CIRCLE_PROJECT_REPONAME}" \
    --branch "${CIRCLE_BRANCH}" \
    --source "CIRCLE_CI" \
    --create-repo-if-missing
workflows:
  cortex-scan-workflow:
    jobs:
      - cortex-code-scan:
          context: cortex-secrets
```

GitHub Actions

- For AMD architecture

```
name: Cortex CLI Code Scan
on:
  push:
    branches:
      - main
  workflow_dispatch:
env:
  CORTEX_API_KEY: ${secrets.CORTEX_API_KEY}
  CORTEX_API_KEY_ID: ${secrets.CORTEX_API_KEY_ID}
  CORTEX_API_URL: <your_cortex_api_url>

jobs:
  cortex-code-scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2

      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 22
      - name: Verify Node.js Version
        run: node -v
      - name: Download cortexcli
        run: |
          set -x
          crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=amd64" \
          -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
          -H "Authorization: ${CORTEX_API_KEY}")
          crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
          curl -o cortexcli $crtx_url
          chmod +x cortexcli
          ./cortexcli --version
      - name: Run Cortex CLI Code Scan
        run: |
          ./cortexcli \
            --api-base-url "${CORTEX_API_URL}" \
            --api-key "${CORTEX_API_KEY}" \
            --api-key-id "${CORTEX_API_KEY_ID}" \
            code scan \
            --directory "${github.workspace}" \
            --repo-id "${github.repository}" \
            --branch "${github.ref_name}" \
            --source "GITHUB_ACTIONS" \
            --create-repo-if-missing
```

- For ARM architecture

```
name: Cortex CLI Code Scan
on:
  push:
    branches:
      - main
  workflow_dispatch:
env:
  CORTEX_API_KEY: ${secrets.CORTEX_API_KEY}
  CORTEX_API_KEY_ID: ${secrets.CORTEX_API_KEY_ID}
  CORTEX_API_URL: <your_cortex_api_url>

jobs:
  cortex-code-scan:
    runs-on: ubuntu-latest
```

```
steps:
- name: Checkout Repository
  uses: actions/checkout@v2

- name: Set up Node.js
  uses: actions/setup-node@v4
  with:
    node-version: 22
- name: Verify Node.js Version
  run: node -v
- name: Download cortexcli
  run: |
    set -x
    crtx_resp=$(curl "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?os=linux&architecture=arm64" \
      -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
      -H "Authorization: ${CORTEX_API_KEY}")
    crtx_url=$(echo $crtx_resp | jq -r ".signed_url")
    curl -o cortexcli $crtx_url
    chmod +x cortexcli
    ./cortexcli --version
- name: Run Cortex CLI Code Scan
  run: |
    ./cortexcli \
      --api-base-url "${CORTEX_API_URL}" \
      --api-key "${CORTEX_API_KEY}" \
      --api-key-id "${CORTEX_API_KEY_ID}" \
      code scan \
      --directory "${{github.workspace}}" \
      --repo-id "${{github.repository}}" \
      --branch "${{github.ref_name}}" \
      --source "GITHUB_ACTIONS" \
      --create-repo-if-missing
```

- For AMD architecture

```

stages: [scan]
variables:
  CORTEX_API_URL: <your_cortex_api_url>
cortex_code_scan:
  image: node:22-bookworm@sha256:bb6834c0669aa71cbc8d94606561a721adf489f6b93d7b8b825f0cf1b498c2c4
  tags: ["amd64"]
  stage: scan
  rules:
    - when: on_success
  before_script:
    - uname -m
    - set -euo pipefail
    - apt-get update
    - apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip file
    - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
      "$CORTEXCLI_HOME"
    - |
      # avoid leaking secrets in logs
      set +x
      CRTX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
      os=linux&architecture=amd64" \
        -H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
        -H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
      set -x
      curl -fsSL -o cortexcli "$CRTX_URL"
      chmod +x cortexcli
      ./cortexcli --version
  script:
    - |
      ./cortexcli \
        --api-base-url "${CORTEX_API_URL}" \
        --api-key "${CORTEX_API_KEY}" \
        --api-key-id "${CORTEX_API_KEY_ID}" \
        code scan \
        --directory "${CI_PROJECT_DIR}" \
        --repo-id "${CI_PROJECT_PATH}" \
        --branch "${CI_COMMIT_REF_NAME}" \
        --source "CORTEX_CLI" \
        --upload-mode no-upload \
        --create-repo-if-missing
  artifacts:
    when: always
    paths: [cortexcli]
    expire_in: 1 day

```

- For ARM architecture

```

stages: [scan]
variables:
  CORTEX_API_URL: <your_cortex_api_url>
cortex_code_scan:
  image: node:22-bookworm
  stage: scan
  rules:
    - when: on_success
  before_script:
    - set -euo pipefail
    - apt-get update
    - apt-get install -y --no-install-recommends curl jq ca-certificates tar gzip file
    - export CORTEXCLI_HOME="/root/.cortexcli"; rm -rf "$CORTEXCLI_HOME" || true; mkdir -p
      "$CORTEXCLI_HOME"
    - |
      # avoid leaking secrets in logs
      set +x

```

```
CRTX_URL=$(curl -fsSL "${CORTEX_API_URL}/public_api/v1/unified-cli/releases/download-link?
os=linux&architecture=arm64" \
-H "x-xdr-auth-id: ${CORTEX_API_KEY_ID}" \
-H "Authorization: ${CORTEX_API_KEY}" | jq -r '.signed_url')
set -x
curl -fsSL -o cortexcli "$CRTX_URL"
chmod +x cortexcli
./cortexcli --version
script:
- |
./cortexcli \
--api-base-url "${CORTEX_API_URL}" \
--api-key "${CORTEX_API_KEY}" \
--api-key-id "${CORTEX_API_KEY_ID}" \
code scan \
--directory "${CI_PROJECT_DIR}" \
--repo-id "${CI_PROJECT_PATH}" \
--branch "${CI_COMMIT_REF_NAME}" \
--source "CORTEX_CLI" \
--upload-mode no-upload \
--create-repo-if-missing
artifacts:
when: always
paths: [cortexcli]
expire_in: 1 day
```

Jenkins

- For AMD architecture

```
pipeline {
    agent {
        docker {
            image 'cimg/node:22.17.0' // Replace with a suitable image or executor
            args '-u root'
        }
    }
    environment {
        CORTEX_API_KEY = credentials('CORTEX_API_KEY')
        CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')
        CORTEX_API_URL = <your_cortex_api_url>
    }
    stages {
        stage('Checkout Repository') {
            steps {
                git branch: 'main', url: 'this-is-repository-url-example'
                stash includes: '**/*', name: 'source'
            }
        }
        stage('Install Dependencies') {
            steps {
                sh '''
                    apt update
                    apt install -y curl jq git
                '''
            }
        }
        stage('Download cortexcli') {
            steps {
                script {
                    def response = sh(script: """
                        curl --location '${env.CORTEX_API_URL}/public_api/v1/unified-
cli/releases/download-link?os=linux&architecture=amd64' \
                        --header 'Authorization: ${env.CORTEX_API_KEY}' \
                        --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \
                        --silent
                    """, returnStdout: true).trim()
                    def downloadUrl = sh(script: """echo '${response}' | jq -r '.signed_url'""", returnStdout: true).trim()
                    sh """
                        curl -o cortexcli '${downloadUrl}'
                        chmod +x cortexcli
                        ./cortexcli --version
                    """
                }
            }
        }
        stage('Run Scan') {
            // Replace the repo-id with your repository like: owner/repo
            steps {
                script {
                    unstash 'source'
                    sh """
                        ./cortexcli \
                            --api-base-url "${env.CORTEX_API_URL}" \
                            --api-key "${env.CORTEX_API_KEY}" \
                            --api-key-id "${env.CORTEX_API_KEY_ID}" \
                            code scan \
                            --directory "$(pwd)" \
                            --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
                            --branch <REPLACE WITH BRANCH> \
                            --source "JENKINS" \
                            --create-repo-if-missing
                    """
                }
            }
        }
    }
}
```

```

    }
}
}

For ARM architecture

pipeline {
    agent {
        docker {
            image 'cimg/node:22.17.0' // Replace with a suitable image or executor
            args '-u root'
        }
    }
    environment {
        CORTEX_API_KEY = credentials('CORTEX_API_KEY')
        CORTEX_API_KEY_ID = credentials('CORTEX_API_KEY_ID')
        CORTEX_API_URL = <your_cortex_api_url>
    }
    stages {
        stage('Checkout Repository') {
            steps {
                git branch: 'main', url: 'this-is-repository-url-example'
                stash includes: '**/*', name: 'source'
            }
        }
        stage('Install Dependencies') {
            steps {
                sh ''
                apt update
                apt install -y curl jq git
                ''
            }
        }
        stage('Download cortexcli') {
            steps {
                script {
                    def response = sh(script: """
                        curl --location '${env.CORTEX_API_URL}/public_api/v1/unified-
cli/releases/download-link?os=linux&architecture=arm64' \
                        --header 'Authorization: ${env.CORTEX_API_KEY}' \
                        --header 'x-xdr-auth-id: ${env.CORTEX_API_KEY_ID}' \
                        --silent
                    """, returnStdout: true).trim()
                    def downloadUrl = sh(script: """echo '${response}' | jq -r '.signed_url'""", returnStdout: true).trim()
                    sh """
                        curl -o cortexcli '${downloadUrl}'
                        chmod +x cortexcli
                        ./cortexcli --version
                    """
                }
            }
        }
        stage('Run Scan') {
            // Replace the repo-id with your repository like: owner/repo
            steps {
                script {
                    unstash 'source'
                    sh """
                        ./cortexcli \
                            --api-base-url "${env.CORTEX_API_URL}" \
                            --api-key "${env.CORTEX_API_KEY}" \
                            --api-key-id "${env.CORTEX_API_KEY_ID}" \
                            code scan \
                            --directory "\$(pwd)" \
                            --repo-id <REPLACE WITH REPO_OWNER/REPO_NAME> \
                            --branch <REPLACE WITH BRANCH> \
                    """
                }
            }
        }
    }
}

```

```
        --source "JENKINS" \
        --create-repo-if-missing
.....
    }
}
}
}
```

6.13.4 | Cortex CLI Cortex Cloud Application Security command line reference

This reference guide documents the commands and flags unique to the Cortex Cloud Application Security CLI. For CLI commands common to all supported modules refer to Cortex CLI common command line reference guide.

IMPORTANT:

The Cortex CLI Cortex Cloud Application Security only supports single occurrences of each flag. If the same flag is passed multiple times, only the last provided value will be used. For example, in the following command, only TF CloudFormation will be the scanned framework.

Example 21.

```
./cortexcli --api-base-url <YOUR_API_URL> --api-key <YOUR_API_KEY> --auth-id <YOUR_AUTH_ID> --
framework terraform --framework "terraform clouformation"
```

Command/Variable	Description
--source	The source of execution. Default source: CLI. Examples: Jenkins, GitHub Actions, CLI

Command/Variable	Description
--repo-id	<p>Required for upload mode.</p> <p>Identity string of the repository. Format <code>repo_owner/repo_name</code>.</p> <p>NOTE:</p> <p>The repo-id flag must not end with <code>.config</code>, <code>.log</code> or <code>.ini</code>. <code>--config</code> is acceptable.</p> <p>Example 22.</p> <ul style="list-style-type: none"> • <code>--repo-id foo.config</code> will be blocked • <code>--repo-id foo-config</code> will pass <p>To retrieve the repository ID, under Inventory, navigate to All Assets → Repositories (under Code) → select a repository → copy the Asset ID value from the Properties section of the side card.</p>
--branch	<p>Required for upload mode.</p> <p>Selected branch of the persisted repository</p>
--directory	<p>Required.</p> <p>The directory path to scan. Cannot be used together with <code>--file</code></p>
-file	<p>The file path to scan. Cannot be used together with <code>--directory</code>. When using this option, the Cortex CLI will filter runners based on the file type provided. For example, if you specify a <code>.tf</code> file, only the Terraform and secrets frameworks will be included. You can further limit this (for example; skip secrets) by using the <code>--skip-framework</code> argument</p>

Command/Variable	Description
--var-file	<p>Variable files to load in addition to the default files. This feature is currently supported for both source Terraform (.tfvars files) and Helm chart scans (for providing custom values or variable overrides). Refer to https://www.terraform.io/docs/language/values/variables.html#variable-definitions-tfvars-files for more information</p>
--framework	<p>Filter to scan specific frameworks. Example: --framework arm.</p> <p>Syntax: Use a single flag with comma-separated values for multiple frameworks. Both quoted (<code>"arm,ansible"</code>) and unquoted (<code>arm,ansible</code>) formats are supported. Example: --framework arm,ansible.</p> <p>Constraint: Do not use multiple --framework flags: --framework terraform --framework sca_package.</p> <p>Environment variables: export CORTEX_CODE_FRAMEWORK=arm,ansible.</p> <p>Supported frameworks: ARM, ANSIBLE, BICEP, CLOUDFORMATION, DOCKER, DOCKERFILE, HELM, KUBERNETES, KUSTOMIZE, OPENAPI, SCA, SECRETS, SERVERLESS, TERRAFORM, TERRAFORMJSON, TERRAFORMPLAN</p>
--skip-framework	<p>Skip specific frameworks. Example: --skip-framework terraform.</p> <p>Syntax: Use a single flag with comma-separated values for multiple frameworks. Both quoted (<code>"arm,ansible"</code>) and unquoted (<code>arm,ansible</code>) formats are supported. Example: --skip-framework terraform, sca_package.</p> <p>Constraint: Do not use multiple skip --framework flags: --skip-framework terraform --skip-framework sca_package.</p> <p>Environment variables: export CORTEX_CODE_SKIP_FRAMEWORK="tf,sca"</p>

Command/Variable	Description
--ca-certificate	CA Certificate to use
--no-cert-verify	This flag disables TLS/SSL certificate verification (default: false). Skips TLS certificate verification when connecting to the API. Use only in test or development environments, as this reduces connection security
--summary-position	Sets the position for displaying the summary information
--upload-mode	<p>Upload mode determines the method or mode used to upload data, and includes these options:</p> <ul style="list-style-type: none"> • upload : Uploads scan results to the Cortex Cloud platform • no-upload : Disables uploads of scan results to the platform • no-code: Uploads scan findings to the platform, but without including the actual source code content (code blocks in the uploaded data)
--external-modules-download-path	Specifies the directory to download external modules to. Defaults to <code>.external_modules</code>
--output Supported formats: cli, json, spdx, junitxml, sarif, cyclonedx, cyclonedx_json	Output format for reporting
--output-file-path	Specifies the output path for the scan result file
--deep-analysis	Enables or disables deep analysis of the Terraform plan and related files

Command/Variable	Description
--repo-root-for-plan-enrichment	Enriches Terraform plan findings by mapping them to their original <code>.tf</code> files
--skip-path	Specifies a path (file or directory) that should be skipped during the scanning process. This option is useful for excluding specific files or directories that are not relevant to the scanning analysis, increasing the efficiency and accuracy of scan results
--create-repo-if-missing	Determines whether the system should create a repository if it is missing. This option allows users to automate the creation of repositories as needed and ensure that all required repositories are available for scanning. For example, when running automated scans or integrating with version control systems, enabling <code>--create-repo-if-missing</code> can help maintain consistency and prevent disruptions due to missing repositories
--compact	Do not display code blocks in the output
--no-fail-on-crash	Prevents the application from failing (blocking pipelines) in the event of a scanner or backend failure. Instead of returning a <code>2</code> exit code, it will return a <code>0</code> exit code in such scenarios.
--var-file	Variable files to load in addition to the default files, Currently only supported for source Terraform (<code>.tf</code> file) and Helm chart scans
CORTEX_APPSEC_VALIDATE_SECRETS	Controls whether secret validation is performed. By default, this feature is disabled. Set <code>CORTEX_APPSEC_VALIDATE_SECRETS = true</code> to enable it

Command/Variable	Description
--timeout	<p>Sets the maximum time the Cortex CLI will wait for triggered local scan processes to complete.</p> <p>Default value: 15 minutes.</p> <p>Syntax:</p> <ul style="list-style-type: none"> • To specify a duration: Use a numeric value followed by a unit (for example <code>--timeout 10m</code>) • Default unit: Numeric values entered without a unit are interpreted as seconds. For example, <code>30</code> is equal to 30 seconds. • Supported units: Milliseconds, seconds, minutes and hours
--help	Help

6.13.5 | Cortex CLI common command line reference guide

This reference guide describes the common command line flags used to manage the Cortex Cloud Application Security, Cloud Workload Protection (CWP) and API Security modules through the Cortex CLI, including the structure of base commands and subcommands.

Common Cortex CLI commands and flags

The following table describes CLI commands common to all supported Cortex CLI modules.

Command	Description
--api-base-url	<p>The public facing API URL. To retrieve the URL, under Settings, select Configurations → API Keys → copy API URL. Required: true.</p> <p>[\${CORTEX_API_BASE_URL}]</p>
--api-key	<p>The API key used for authorization. Required: true.</p> <p>[\${CORTEX_API_KEY}]</p>

Command	Description
--api-key-id	The API key ID. Required: true. [\$CORTEX_API_KEY_ID]
--soft-fail	<p>Identifies and reports errors identified during a scan but does not trigger a failing condition. Instead, the scan returns a successful result with an exit code of 0. Unlike skipped or suppressed checks, soft fail errors are still reported but do not cause the scan to fail. Required: false.</p> <p>[\$CORTEX_SOFT_FAIL]</p> <p>NOTE:</p> <p>For soft fails, a failed check matches the defined severity threshold. If multiple soft fail severities are specified, the highest severity acts as the threshold for determining a soft fail. However, a successful scan will always return an exit code of 0, even if block-level findings (which might trigger soft fails based on severity) are present.</p>
--log-level	Set the logging level (INFO, WARNING, ERROR) for Stdout output
--http-proxy	The HTTP proxy server URL to route traffic through [\$HTTP_PROXY]
--help	Show help options
--version	Retrieves the version of the Cortex CLI currently in use

6.13.6 | Git Hooks

Abstract

Pre-commit and pre-receive hooks scan code changes locally and on push to detect exposed secrets.

Cortex Cloud Git hooks automatically scan your code for exposed secrets via the Cortex CLI wrapper before code changes are pushed or committed.

The following hooks are supported:

- Pre-commit hooks: Run locally before changes are committed
- Pre-receive hooks: Run on the remote server before changes are pushed

6.13.6.1 | Cortex CLI pre-commit hooks

Abstract

Integrate Application Security secrets scanner as pre-commit hooks into your workflows to scan for errors on your machine before local commits.

Integrate the Cortex Cloud Application Security secrets scanner as a pre-commit hook by installing the Cortex CLI. The scanner executes the hook locally before a commit. This setup ensures that secrets checks are enforced before any changes are committed.

When setting up pre-commit hooks, you can choose between local hooks and global hooks.

- **Local:** Installs the hook in the `.git/hooks` directory of the **current** repository, ensuring that Cortex Cloud secrets scans automatically run on your code before every commit
- **Global:** Installs the hook for **all** Git repositories on your machine, so Cortex Cloud secrets scans will automatically run on your code before every commit, regardless of the project

How to configure pre-commit hooks

PREREQUISITE:

These common prerequisites are required for all types of installation (both local and global) of the Cortex CLI pre-commit hook.

- Ensure you have a license for Cortex Cloud Application Security
- Install the Cortex Cloud CLI binary locally. Refer to Connect Cortex CLI for information about onboarding the CLI
- Obtain Cortex Cloud API credentials (API Key ID and API Key) available from the CLI onboarding process (see above), and your API base URL. For more information on creating API keys, refer to <https://docs-cortex.paloaltonetworks.com/r/Cortex-XSIAM-REST-API/Create-a-new-API-key>
- **Git:** You must have Git installed on your machine. For installation instructions, refer to the official Git website

1. Create a directory:

```
mkdir -p ~/.cortexcli
```

2. Create a `.cortex.yaml` file in the `~/.cortexcli/` directory.

3. Open the `.cortex.yaml` file and add your Cortex Cloud API credentials and API base URL to the `yaml` file:

- CORTEX_API_BASE_URL: <replace with the base API URL>
- CORTEX_API_KEY_ID: <replace with API Key ID>
- CORTEX_API_KEY: <replace with API Key>

NOTE:

It is recommended you configure credentials for the Cortex CLI using a configuration file.

4. **For local hooks:** Install the Cortex CLI pre-commit hook package to set up a local hook for the current Git repository:

PREREQUISITE:

For local installation: Install the **pre-commit** framework version 3.2.0 or greater. Refer to <https://pre-commit.com/> for installation instructions.

- a.
 - For macOS, you can use Homebrew:

```
brew install pre-commit
```

- For other installations run:

```
pip install pre-commit
```

- b. Navigate to the root of your repository → run the following command:

```
cortexcli code pre-commit install --mode local
```

5. **For Global hooks:** Install the Cortex CLI pre-commit hook package to set up hooks for all Git repositories on your machine.

```
cortexcli code pre-commit install --mode global
```

NOTE:

The **pre-commit** framework is not required for global mode.

References

To set up the Cortex CLI as a pre-commit hook on supported platforms, refer to the following official Git documentation for managing hooks:

- **Git Hooks:** A comprehensive guide on all available Git hooks, including **Pre-commit**: <https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks>
- **Atlassian Git Tutorial:** A tutorial that explains the purpose and usage of both local and server-side hooks, including **pre-commit**: <https://www.atlassian.com/git/tutorials/git-hooks>

6.13.6.1.1 | Pre-commit hook usage

You can run secrets checks on your code, customize its behavior using supported flags, and suppress detected secrets when required.

By default, Cortex CLI pre-commit hooks:

- **Scan staged files only:** The scan performs a quick and efficient check by only analyzing the changes you are about to commit, rather than the entire codebase
- **Scan for secrets only:** Pre-commit hooks support secrets scans only
- **Do not upload results to the platform:** All scan results are kept local to your machine, ensuring your data remains private

Command flag reference

Use the following flags with the **cortexcli code pre-commit** command to customize scanner behavior.

- **--ignore-existing-secrets**: Ignores secrets that already exist from a periodic scan (default: false) [`[$CORTEX_CODE_IGNORE_EXISTING_SECRETS]`]
- **--validate-secrets**: Checks if the secrets are valid (default: false) [`[$CORTEX_CODE_VALIDATE_SECRETS]`]
- **--skip-path**: Specifies a file or directory path to skip during the scan [`[$CORTEX_CODE_SKIP_PATH]`]
- **--compact**: Prevents the display of code blocks in the output (default: false) [`[$CORTEX_CODE_COMPACT]`]
- **--summary-position**: Determines whether the summary appears on top (before the check results) or on bottom (after the check results). (default: top) [`[$CORTEX_CODE_SUMMARY_POSITION]`]
- **--no-fail-on-crash**: Returns exit code 0 instead of 2 in case of a failure in the integration with the platform (default: false) [`[$CORTEX_CODE_NO_FAIL_ON_CRASH]`]
- **--help, -h**: Displays a help message with available options

Secrets suppression

You can suppress secrets directly within your code by adding a comment. This is useful for secrets that are intentionally included or a false positive and are not a security risk. Currently, suppression is not supported in **JSON** files.

The comment format is:

```
cortex:skip=<SECRET_ID>:<suppression justification>
```

Replace **<SECRET_ID>** with the specific ID provided in the scan output, and provide a brief explanation for why the secret is being suppressed. The comment syntax will depend on the file type.

Example 23. Example

Comments in a Dockerfile begin with (#). Note the comment in the **After suppression** code-block below.

- **Before suppression:**

```
ENV SEC_1="ghp_3xyKmc3W7XanE82IKHJ3Z3AfHbV"
```

- **After suppression:**

```
# cortex:skip=APPSEC_SECRET_43: Suppress this key for testing purposes
ENV SEC_1="ghp_3xyKmc3W7XanE82IKHJ3Z3AfHbV"
```

6.13.6.2 | Cortex CLI pre-receive hooks

Abstract

Integrate the Application Security secrets scanner as a pre-receive hook into your workflows to scan for errors before code is accepted into your repository.

Integrate the Cortex Cloud Application Security secrets scanner as pre-receive hook into your workflows installing the Cortex CLI. The hook runs on the remote server before changes are pushed, allowing you to enforce checks before code is accepted into version control.

Supported version control systems: Pre-receive hooks are supported for GitHub Enterprise, GitLab self-managed, and Bitbucket Data Center. To setup pre-receive hook on these platforms refer to Setup on third-party platforms below.

Pre-receive hook workflow setup

1. Fulfill prerequisites.
2. Configure API credentials.
3. Install the pre-receive hook.
4. Setup the pre-receive hook on third party platforms.

Setup requirements

PREREQUISITE:

Before you begin, ensure you have:

- **Administrator** access to the VCS server and console
- A valid license for Cortex Cloud Application Security
- The **Cortex Cloud CLI binary** or Docker image installed on the server (requires GLIBC (GNU C library) version 2.35 or greater). Refer to Connect Cortex CLI for information about onboarding the CLI
- **Cortex Cloud API credentials** (API Key ID and API Key) and your API base URL. For more information on creating API keys, refer to <https://docs-cortex.paloaltonetworks.com/r/Cortex-XSIAM-REST-API/Create-a-new-API-key>
- **Git** installed on your machine. For installation instructions, refer to the official Git website

Configure credentials

It is recommended to configure credentials for the Cortex Cloud Application Security Cortex CLI using a configuration file, instead of embedding them directly in the hook script.

1. Create a directory:

```
mkdir -p ~/.cortexcli/.cortex.yaml
```

NOTE:

Make sure to create the directory under the home directory of the Linux user that runs the Git hooks. This user is typically not the root user.

2. Configure credentials: Open the `.cortex.yaml` file in the `~/.cortexcli/` directory and add the following configuration parameters:

- `CORTEX_API_BASE_URL`: <API base URL>
- `CORTEX_API_KEY_ID`: < API key ID >
- `CORTEX_API_KEY`: < API key>

To set up the Cortex CLI as a pre-receive hook on supported third-party platforms, refer to the official vendor documentation:

- **GitHub Enterprise:** About pre-receive hooks
- **GitLab self-managed:** Git server hooks
- **Bitbucket Enterprise:** Using repository hooks

Reference script

Use the script below as reference to extend or modify your existing pre-receive hooks in your VCS provider.

```
#!/usr/bin/env bash

# This script is used to run Cortex CLI in a pre-receive hook.

# Hide the update notice.
export CORTEX_HIDE_UPDATE_NOTICE=1

CORTEX_CLI="/usr/local/bin/cortexcli"
BASE_COMMAND="--api-base-url ${CORTEX_API_BASE_URL} --api-key-id ${CORTEX_API_KEY_ID} --api-key
${CORTEX_API_KEY} code pre-receive"
OPTIONAL_FLAGS=''

# Run cortex cli
${CORTEX_CLI} ${BASE_COMMAND:-''} ${OPTIONAL_FLAGS:-''}

exit_code=$?

exit $exit_code
```

6.13.6.2.1 | Pre-receive hook usage

The hook executes a script on every git push.

By default, Cortex CLI pre-receive hooks:

- **Only scans code changes:** It analyzes the code difference included in the pushed commits, not the entire repository
- **Scans for secrets only:** The analysis is focused on detecting sensitive information
- **Does not upload results to Cortex Cloud:** All scan results are kept local to your machine (on the server)

Understanding the script variables

- **CORTEX_CLI:** Defines the executable path, pointing to the absolute location of the `cortexcli` binary
- **BASE_COMMAND:** Assembles the core command string, including authentication flags (`--api-base-url`, `--api-key-id`, `--api-key`) and the primary command: `code pre-receive`. The use of `...{ }...` ensures authentication variables are injected as flag values
- **OPTIONAL_FLAGS:** An empty variable placeholder for adding optional runtime arguments

Command flag reference

Use the following flags with the pre-receive command to customize scanner behavior.

Example command structure:

```
$ cortexcli code pre-receive [options]
```

Option	Description
--ignore-existing-secrets	Ignores secrets that already exist in the periodic scan (default: false) [\$CORTEX_CODE_IGNORE_EXISTING_SECRETS]
--validate-secrets	Checks if the secrets are valid (default: false) [\$CORTEX_CODE_VALIDATE_SECRETS]
--skip-path	Specifies a file or directory path to skip during the scan [\$CORTEX_CODE_SKIP_PATH]
--compact	Prevents the display of code blocks in the output (default: false) [\$CORTEX_CODE_COMPACT]
--summary-position	Determines whether the summary appears on top (before the check results) or on bottom (after the check results). (default: top) [\$CORTEX_CODE_SUMMARY_POSITION]
--no-fail-on-crash	Returns exit code 0 instead of 2 in case of a failure in the integration with the platform (default: false) [\$CORTEX_CODE_NO_FAIL_ON_CRASH]
--help, -h	Displays a help message with available options

Breakglass: Bypassing the hook

The breakglass feature allows you to intentionally bypass the pre-receive hook security scan. This is useful in urgent situations where a push must go through immediately, but it should be used with caution as it overrides your security policies.

1. Configure your server to accept custom push options:

```
```bash
git config receive.advertisePushOptions true
````
```

2. Add the `-o breakglass` option to your `git push` command:

```
```bash
git push -o breakglass
```
```

Troubleshooting and recommendations

- Refer to the Cortex CLI for more information on the Cortex CLI.
- Modify the script as required based on the server running the VCS
- The Cortex CLI must be available on the server. This documentation does not describe the CLI installation process
- Update the Cortex CLI periodically
- Instead of adding the API URL and credentials directly in the script, consider creating a `~/.cortexcli/.cortex.yaml` configuration file (owned by the git user and group) with the following contents:

```
CORTEX_API_BASE_URL: <api base url>
CORTEX_API_KEY: <api key>
CORTEX_API_KEY_ID: <api key id>
```

6.14 | IDE

Abstract

Integrating the AppSec IDE security plugin to scan for misconfigurations, vulnerabilities, and secrets while coding, with in-IDE fixes.

Integrate the Cortex Cloud security plugin into your workflow with your IDE to directly scan your infrastructure-as-code (IaC) files for misconfigurations, detect vulnerabilities in your software composition analysis (SCA) packages, identify exposed secrets, and uncover license violations while coding.

This process seamlessly runs in the background without disrupting your coding experience. Security findings are flagged within your code, categorized by scan type and severity for identification and resolution within the IDE itself. Remediation options include fixes (when available), suppression, or referring to documentation. Supported IDEs include VS Code and all JetBrains offerings (such as IntelliJ, PyCharm and so on).

NOTE:

Not all remediation options are available for all findings or all type of scan category.

6.14.1 | System requirements

License: Running a supported IDE requires a license for the Cortex Cloud add-on.

Supported architectures:

- macOS: arm64 and x64 (x86-64 or amd64)
- Windows: x64

Supported frameworks: The IDE is compatible with all frameworks supported by Cortex Cloud Application Security

Limitations:

- Mono-repos are not supported
- Special characters (such as &, parentheses) are not supported in user names within a file path). For example, the é character in `c:\Users\JohnSmithé` is not supported
- For terminal actions performed by Cortex Cloud IDE extensions on Windows, only Command Prompt (CMD) is supported. PowerShell is not supported

6.14.2 | Visual Studio (VS) Code and VS Code compatible IDEs

Integrate the Cortex Cloud code security plugin with your Visual Studio (VS) Code or any VS Code-compatible IDEs (such as Cursor, VSCodium, or Windsurf) to enhance security during development. The plugin scans for security policy violations using both default and custom policies, enabling you to identify and resolve issues before committing code, reducing the risk of pull request failures caused by undetected problems.

Choose your security level

- **Enhanced security with Cortex AppSec integration** (recommended): For comprehensive security analysis, connect the plugin to Cortex Cloud using your Cortex Cloud API Key and API Key ID (see prerequisites below for more information about API keys). This option provides the full functionality of the platform security features
- **Basic open-source functionality**: If you prefer not to use access credentials, you can still benefit from the plugin's basic open-source features. These features provide a foundational level of security analysis, helping you identify common vulnerabilities. However, you will not have access to Cortex Cloud

Supported scan categories

The plugin scans these code security categories:

- Secrets: Identifies sensitive data embedded in code, such as API keys, encryption keys, OAuth tokens, certificates, PEM files, passwords, and pass-phrases
- IaC misconfigurations in IaC templates such as Kubernetes and Terraform. For a list of supported IaC frameworks see Supported frameworks
- SCA vulnerabilities: Includes security issues in both direct and transitive open-source dependencies
- Licenses: Software license noncompliance
- Package Integrity: Assesses the operational risk and potential impact of each package in your codebase

Prerequisites

PREREQUISITE:

Before you begin (These prerequisites apply to both VS Code and VS Code compatibles):

- **Permissions:** CLI Read only permissions. Refer to Cortex CLI for more information about permissions
- **Environment setup**
 - **macOS and Windows:** Install Python 3.9.x to 3.12.x
 - Install Node.js version 22 and above for SCA scans (such as vulnerabilities scans)
- On the Cortex Cloud console.
 - Retrieve your access key and URL for authentication purposes when setting up the plugin
 - Generate and copy a Cortex Cloud access key to enable access to Cortex Cloud. The access key includes a key ID and secret:
 1. Navigate to Settings → Configurations → API Keys (under Integrations) → + New Key.
 2. Copy and save the key.
 3. Retrieve your API Key ID from the ID column.

For more information about API keys, refer to <https://docs-cortex.paloaltonetworks.com/r/Cloud-Onboarding-Public-APIs/Understand-Cortex-Cloud-licenses>.

For more information about API keys, see <https://docs-cortex.paloaltonetworks.com/r/Cortex-XSIAM-REST-API/Cortex-XSIAM-APIs>.

For more information about API keys, see <https://docs-cortex.paloaltonetworks.com/r/Cortex-XDR-REST-API/Cortex-XDR-API-Overview>.

NOTE:

When generating an API key, ensure you select the Standard security level.

- Retrieve your Cortex Cloud API URL: Navigate to Settings → Configurations → API Keys (under Integration) → click Copy API URL

Installation

You can install the plugin directly from your IDE extensions panel or though the **Visual Studio Marketplace** (for VS Code) or the **Open VSX Registry** (for compatible IDEs). After completing any installation method, your IDE will activate the extension. Restart your IDE if prompted to ensure the necessary Cortex CLI components initialize correctly.

Install VS Code

- Install through VS Code IDE
 - Access the Activity bar → Extensions → Cortex Cloud → Install.
- Install from the Visual Studio Marketplace
 1. Access the Cortex Cloud extension from the Visual Studio Marketplace (for VS Code).
 2. Select Install → Open Visual Studio Code.app.

You are redirected to the Cortex Cloud extension on your IDE.

 3. Click Install.

- **Install the Cortex Cloud extension from within a compatible IDE**

1. Select the Extensions icon (represented by four squares) in the IDE's Activity Bar (usually on the far left).

TIP:

You can access Extensions using the keyboard shortcuts **Ctrl+Shift+X** (Windows) or **Cmd+Shift+X** (macOS).

2. Search the Registry: Enter Cortex Cloud in the search field at the top of the Extensions view.

The search results will pull the extension listing from the Open VSX Registry.

3. Click on the Cortex Cloud extension in the search results.

4. Select Install.

- **Install the Cortex Cloud extension from the Open VSX Registry**

1. Download the extension package.

- a. Open the Open VSX Registry

- b. Select the Download link to save the **.vsix** extension file to your local system.

2. Install manually via the IDE Command Palette:

- a. Open your compatible IDE.

- b. Access the **Command Palette** by pressing **Ctrl+Shift+P** (Windows/Linux) or **Cmd+Shift+P** (macOS).

- c. Type or select Install from VSIX (The exact name may vary slightly, such as Extensions: Install from VSIX...).

A file dialogue opens.

- d. Navigate to the location where you saved the **.vsix** file in Step 1b and select it.

The IDE will process the package and activate the extension.

Configure plugin settings

The configuration process depends on whether you're using the open-source or proprietary version. For the proprietary version, you will need your Cortex Cloud API Key, API Key ID and tenant URL to establish a secure connection between your environment and Cortex Cloud. These details authenticate you to your tenant. The open-source project does not require these settings.

NOTE:

Enforcement rules and CA certificates are not applicable to the open-source project.

1. Access the Cortex Cloud extension settings in one of these ways:

- Select Extensions → Cortex Cloud → navigate to the bottom menu bar of your VS Code editor → select the Cortex Cloud tab
- Select Extensions → Cortex Cloud → click the Settings (cogwheel) icon → Settings

2. Fill in the provided fields:

- API Key ID (required): The Cortex Cloud access key ID. See Prerequisites above
- API Key (required): The Cortex Cloud secret key. See Prerequisites above
- Platform URL (required): Your Cortex Cloud URL. See Prerequisites above

PREREQUISITE:

You must insert your API key and API ID values into the Settings before providing the tenant URL.

- CLI Version: Leave blank to use the latest CLI version (or enter 'latest'), or specify a version
- CLI Path: Specifies the path to the CLI scanner. Recommended: Leave empty to let the extension manage the scanner installation
- Disable Error Message Popups: Hide error message popups. You can view errors in the logs via the **Open Cortex Cloud Log** command
- Certificate: Add your Cortex Cloud CA certificate. Format: **.pem** file

Example 24. Example

- macOS/Linux: /Users/your_username/Documents/cacert.pem or ~/Documents/cacert.pem
- Windows: C:\Users\your_username\Documents\cacert.pem

- Ignore Gitignore files: Selected by default. Files that belongs to paths included in the **gitignore** file will not be scanned when opened or saved
- External Checks Directory: Provide the path to a folder containing custom security checks
- Specific Frameworks: Scan specific frameworks such as **ARM**. You can add multiple frameworks using spaces between the values in the command. Refer to Cortex CLI Cortex Cloud Application Security command line reference for more information about framework flags
- Environment Variables: Define specific environment variables and their values that will be accessible to the security scanner while it performs its analysis
 - To add variables, select Add item → provide the variable key/value pairs → OK
 - To edit or delete a variable: Select the edit or delete icons next to a variable in the table

UI layout

To view the extension, select the Cortex Cloud tab in the Activity bar. The extension UI layout is as follows:

- **Left pane:** The **Security scan panel**, which includes these features:
 - Full Scan button: manually initiate a full scan of your project
 - **Scan results.** Features a tree structure displaying detected issues by security category (IaC misconfigurations, Secrets, Vulnerabilities (SCA), and Licenses). Each category expands to reveal folders containing specific issues detected during a scan
 - Control buttons: Provide access to Settings, Test Connectivity, Full scan play button, and Extension Monitoring, which includes scan history and log files
- **Middle pane:** **Code editor.** Review your codebase, and view a list of issues related to a file or resource (for IaC misconfigurations), along with remediation options
- **Right pane:** **Details panel.** Provides a detailed view of a selected issue, including information such as the code difference when available, and remediation options

6.14.2.1 | How to use the Cortex Cloud extension in VS Code

The Cortex Cloud security extension allows you to conduct comprehensive scans, manage findings efficiently, and remediate issues within your coding environment. You can prioritize findings and address the most critical issues by filtering the scan results based on category and severity.

To use the extension:

1. Write your code.

The plugin provides real-time security feedback and suggestions.

2. Use the available actions and commands to resolve security issues and ensure compliance with best practices.

UI layout

To access the plugin, select the Cortex Cloud icon in the extension panel. The UI is divided into three sections:

- **Left pane (Navigation panel):** Displays a tree structure of issues according to security category: (IaC misconfigurations, Secrets, Vulnerabilities (SCA), and Licenses). Expand each category to reveal folders containing specific issues detected during a scan
- **Middle pane (Code editor).** Allows you to review your codebase, view findings related to resources (for IaC misconfigurations) or files, and access remediation options
- **Right pane (Problems Tool).** Provides detailed information about a selected issue, including code differences (when available), and remediation options

Scans

You can perform scans on an entire project or specific files:

- **Full scan:** Click the Play button for any security category (for example, IaC and Secrets) in the Navigation panel to scan the entire project. This action initiates scans across all categories
- **Selected file scan:** Open or save a specific file to trigger a scan for that file only

Findings

Manage findings through either the **Code editor** or the **Problems Tool**.

- Findings are grouped by security category in the Navigation panel for easy browsing
- Click on a category or folder to view associated findings
- Select an issue to display its details in the Code Editor and Problems Tool
- Use the filter icon in the Navigation panel to refine findings by severity

Manage findings in the Code editor

To manage findings directly in your code editor, the extension highlights secrets, misconfigurations, vulnerabilities, and license non-compliance issues with red marks next to affected lines. Hover over the marks to view the issue's details, severity and remediation options. For multiple issues, scroll to view all detected findings. For a more detailed view of any issue, click **Console** to open the Problems Tool and explore further information about the detected finding.

Manage findings through the Problems Tool

The Problems Tool provides a detailed view of selected issues, including available remediation options. You can access the Problems Tool by selecting an issue in the navigation bar or through the **Console** option in the Code editor.

Remediation

You can mitigate issues directly through both the Code editor and the Problems Tool. Options include **Fix**, **Suppress**, or **Documentation**.

NOTE:

Not all remediation options are available for all issues.

Fix issues

When selecting an issue in the Code editor or Problems Tool, a suggested fix is displayed when available. Fixes are applied automatically upon selection. Below is a list of the types of fixes available for different issue categories.

- Secrets issues: Follow the policy guidelines
- IaC misconfiguration: The fix modifies the configuration. The Problems Tool displays the code difference to be fixed
- SCA CVE vulnerabilities: You can directly fix the specific vulnerability that has been detected during the scan by upgrading the package to the version that includes a fix
- License mis-compliance: Follow the policy guidelines
- Package integrity:

Suppression

Suppress an issue to temporarily hide or ignore an issue without fixing it, allowing you to concentrate on more important issues.

NOTE:

The suppression is scoped to the file.

1. In the IDE, select an issue from the Navigation bar and click Suppress in either the Code editor or Problems Tool.
2. Provide a justification for the suppression and click Enter to confirm.

The justification will be added as a commented annotation to your source code.

After suppressing an issue, the file will not be scanned for two minutes. This is to prevent the issue from being re-triggered. Saving the file during the hold period will not trigger a scan.

Documentation

If automated fixes are not available, policy documentation can provide guidance on how to address the issue: Select an issue and click Documentation. You are redirected to the relevant policy documentation which includes suggested guidelines on how to solve the issue.

6.14.3 | JetBrains

Integrate the Cortex AppSec code security plugin with your JetBrains IDE instance to enhance security during development. The plugin scans for security policy violations using both default and custom policies, allowing you to identify and resolve issues before committing code, reducing the risk of pull request failures due to undetected problems.

NOTE:

The Cortex AppSec code security plugin supports all JetBrains products.

Supported scan categories

The plugin scans these code security categories:

- Secrets: Identifies sensitive data embedded in code, such as API keys, encryption keys, OAuth tokens, certificates, PEM files, passwords, and pass-phrases
- IaC misconfigurations in IaC templates such as Kubernetes and Terraform. For a list of supported IaC frameworks see [Supported frameworks](#)
- SCA vulnerabilities: Includes security issues in both direct and transitive open-source dependencies
- Licenses: Software license mis-compliance
- Package Integrity: Assesses the operational risk and potential impact of each package in your codebase

PREREQUISITE:

- **Permissions:** CLI Read only permissions. Refer to Cortex CLI for more information about permissions
- Environment setup
 - **macOS** and **Windows**: Install Python 3.9.x to 3.12.x
 - Install Node.js version 22 and above for SCA scans (such as vulnerabilities scans)
- On Cortex Cloud
 - Generate and copy a Cortex Cloud access key to enable access to Cortex Cloud. The access key includes a key ID and secret:
 1. Navigate to Settings → Configurations → API Keys (under Integrations) → + New Key.
 2. Copy and save the key.
 3. Retrieve your API Key ID from the ID column.

For more information about API keys, refer to <https://docs-cortex.paloaltonetworks.com/r/Cloud-Onboarding-Public-APIs/Understand-Cortex-Cloud-licenses>.

For more information about API keys, see <https://docs-cortex.paloaltonetworks.com/r/Cortex-XSIAM-REST-API/Cortex-XSIAM-APIs>.

For more information about API keys, see <https://docs-cortex.paloaltonetworks.com/r/Cortex-XDR-REST-API/Cortex-XDR-API-Overview>.

NOTE:

When generating an API key, ensure you select the Standard security level.

- Retrieve your Cortex Cloud API URL: Navigate to Settings → Configurations → API Keys → select Copy API URL.

Installation

You can install the plugin directly through the JetBrains IDE Plugins panel or the JetBrains Marketplace.

- Install through JetBrains IDE: Navigate to Settings → Plugins → select the Marketplace tab → search for Prisma Cloud → Install → OK
- Install from the JetBrains marketplace:
 1. Open the marketplace.
 2. Select a platform, search for the Prisma Cloud plugin, and click Get.
 3. Select Download from the Versions tab and then Got it in the popup.

Configure plugin settings

1. In your IDE, select Settings → Tools → Cortex Cloud.
2. In the plugin Settings screen, fill in these fields:

- Access Key: Your Cortex Cloud API key ID
- Secret Key: Your Cortex Cloud API secret key
- Cortex Cloud URL: Your Cortex Cloud URL.

PREREQUISITE:

You must insert your API key and API ID values into the Settings before providing the tenant URL.

- CA-Certificate (optional): Add your CA certificate. Format: .pem file

Example 25. Example

- macOS/Linux: /Users/your_username/Documents/cacert.pem or ~/Documents/cacert.pem
- Windows: C:\Users\your_username\Documents\cacert.pem

- CLI Version: Leave blank to use the latest Cortex Cloud Application Security version (or enter 'latest'), or specify a version
- CLI Path: Specifies the path to the CLI executable. Recommended: Leave empty to let the extension manage the CLI installation
- Ignore gitignore files: Selected by default. Files that belongs to paths included in the **gitignore** file will not be scanned when opened or saved
- External Checks Directory: Provide the path to a folder containing custom security checks
- Custom Environment Variables: Environment variables passed to the scanner during scans such as CORTEX_API_BASE_URL:
 - To add variables, provide the key/value pairs in the table under the Custom Environment Variables field
 - To remove variables, select the (—) sign

3. Click Apply → OK.

[Test connection](#)

You can test your connection by selecting the Test Connection field under Settings.

[Manage plugin configurations](#)

Use one of these methods to access plugin configurations and modify settings:

- Select the Settings icon in the Details panel
- Select Settings → Tools → Cortex Cloud

UI layout

- **Left pane:** Displays a tree structure that accurately mirrors the files and folders of the project you have opened.
- **Middle pane:** Displays the **Code editor**. When an issue is selected in the Details panel, the relevant file opens in this editor. This allows you to examine your codebase, see the issue in its specific context (such as an IaC misconfiguration), and find recommended remediation steps
- **Bottom pane:** This is the **Details panel**. Serves as the central hub for in-depth analysis and management of your code scan issues. It provides comprehensive information on selected issues, remediation options scan history, and various controls such as a manual scan option.

6.14.3.1 | How to use the JetBrains Cortex Cloud extension

The Cortex Cloud security extension for JetBrains IDEs integrates comprehensive scanning, efficient issue management, and remediation capabilities directly into your coding environment, without disturbing development.

Execute scans

You can scan your code for security issues using two primary methods: full project scans and single file scans.

- **Full project scans:** These scans comprehensively analyze your entire project. They can be initiated in two ways:
 - **Automatic scan:** Triggered automatically when you open a project
 - **Manual scan:** Initiated by clicking the Play button in the **Details panel**
- **Single file scan:** These scans focus on a specific file. They are triggered automatically when you open or save that file

View scan history

Select Scan History in the **Details panel** to view a record of past scanning activities. For each recorded scan, you can review details such as the start time and duration of the scan, the path that was scanned, the scan trigger (such as Manual or File Opened), the total number of issues detected, and the CLI command used to execute the scan. Under scan history, you can also run scans locally from the terminal for support purposes and so on.

Filters are available within the Scan History to view All Scans, Project Scans, or File Scans.

View log files

Select the Log file icon (next to the Play button in the Details panel) to view log files. These logs provide diagnostic information and details about the execution of your scans, which can be useful for troubleshooting.

Manage issues in the **Details panel**

The **Detail panel**'s main display features a series of tabs for categorizing issues. An Overview tab provides a summary of all detected issues. Alongside it, dedicated tabs exist for specific security categories: Issues are organized by security category (IaC, Secrets, Vulnerabilities, Licenses, and Package Integrity). The Overview

tab displays the total count of all issues, while each category-specific tab displays the total count of issues associated with its specific type.

You can filter issues:

- By severity level (L M H C) to filter issues by severity
- By fix availability (Fix Available) to filter for issues with an available fix

NOTE:

After selecting Fix Available, the number of issues displayed in the issue categories (such as IaC) reflect the number of fixable issues for that type.

The **Details panel** displays a hierarchical view of folders and files that contain issues, with this display dynamically updated based on the selected issue type tab.

1. Under a scan category, browse through folders/subfolders to locate and click on a file containing issues.
2. (Optional) Use a filter to prioritize issues.
3. Select an individual issue within the file to display its details in the right section of the **Details panel**.

NOTE:

The corresponding file simultaneously opens in the **Code editor**, highlighting the issue within its exact code context. See below for more information.

The detailed issue view provides information including:

- The name and description of the issue
- The code lines (or resource for IaC misconfigurations) in which the issue has been detected
- Contextual remediation options, provided specifically for each issue type to guide resolution. For more information about remediation options, refer to Remediation options by issue category below

[Manage issues in the **Code editor**](#)

Select an individual issue within the **Details panel** to open the file containing the issue in the **Code editor**.

Issues are marked by a red i icon next to the code line.

- Click the red i icon for basic details about an issue: name, severity, and remediation options (see below)
- For IaC resources with multiple issues, hovering over the line of code marked i displays a list of issues at the resource's starting line. Scroll to view all issues
- Select Console to display the issue in the **Details panel**

[Remediation options by issue category](#)

You can mitigate issues directly through both the **Code editor** or the **Details panel**. Options include Fix, Suppress, and Documentation.

NOTE:

Not all types of remediation are available for all issue categories. For example, fixes are not available for License issues.

Fixes

When selecting an issue in either the **Code editor** and **Details panel**, a suggested fix is displayed when available. Fixes are automatically applied to the code upon selection. The following list displays the type of fix available for the different categories of issues.

- **CVE vulnerabilities:** The fix bumps the package version. You can directly fix the specific CVE vulnerability that has been detected during the scan by upgrading the package to the version that includes a fix
- **IaC misconfigurations:** The fix modifies the configuration. The **Details panel** displays the code difference to be fixed
- **Secrets issues:** N/A
- **License mis-compliance:** N/A
- **Package Integrity:** N/A

Suppression

Suppress an issue to temporarily hide or ignore an issue without fixing it, allowing you to concentrate on more important issues. The suppression is scoped to the file.

1. Select an issue from the Details panel → click Suppress in either the Code editor or Details panel.
2. Provide a justification for the suppression → click OK.

The justification will be added as a commented annotation to your source code.

After suppressing an issue, the file will not be scanned for two minutes. This is to prevent the issue from being re-triggered. Saving the file during the hold period will not trigger a scan.

Documentation

If automated fixes are not available, policy documentation can provide guidance on how to address the issue: Select an issue → click Documentation in either the Code editor or Details panel. You are redirected to the relevant documentation which includes suggested guidelines on how to mitigate the issue.

NOTE:

Secrets and Licenses category issues are typically mitigated by following the guidance in the Documentation.

6.15 | Developer Suppressions

Developer suppressions enable you to temporarily ignore specific scan findings that are known or acceptable in your code. You can apply a suppression inline in the code using inline comments or an annotation, or manage them from supported IDEs (see IDE integrations).

When added in code, a suppression comment is placed next to the relevant line. The scanner still creates the finding, but marks it as Suppressed rather than excluding it. This ensures that the finding remains visible for audit, while removing it from active remediation workflows.

You can also create policies that use developer suppressions as a condition. For more information, refer to Application Security Policies.

Use suppressions only for valid exceptions (for example, a false positive or an approved design decision), and include a clear justification. Scope each suppression to the minimal code element (line or block) and review them regularly to prevent technical debt.

Guidelines

- **Default branch behavior:** Suppressions on the default branch apply to both pull request (PR) comments and CI/CD runs
- **Periodic scans:** When a suppression is added to a PR, it will not take effect in the periodic scan until the PR is merged
- **Merging PRs:** When a PR containing a suppression is merged into the default branch, the suppression specific to that asset is applied to the default branch. As a result, subsequent pull requests and periodic scans will no longer flag the suppressed issue

Suppressing checks

The general pattern for inline suppressions is:

```
#cortex:skip=<rule_id>:<suppression_comment>
```

- **rule_id** : The ID of the specific check
- **suppression_comment**: An explanation included in scan output

Examples

The following examples show how to add inline suppressions to ignore specific findings. Each suppression includes a check ID and an optional comment explaining the reason, such as a false positive or an approved deviation.

NOTE:

The check IDs and rule names are placeholders for illustration purposes and do not correspond to actual checks.

Terraform

The following comment skips the **APPSEC_AWS_20** check for the **foo-bucket S3** resource. This check ensures that the S3 bucket is private. It is skipped here because the bucket is intentionally public.

```
resource "aws_s3_bucket" "foo-bucket" {  
    region      = var.region  
    # cortex:skip=APPSEC_AWS_20:The bucket is a public static content host  
    bucket      = local.bucket_name  
    force_destroy = true  
    acl         = "public-read"  
}
```

CloudFormation

CloudFormation checks can be suppressed either with an inline comment next to the resource or using the resource's Metadata section.

- Option #1 Inline Comment

The following comment skips the **APPSEC_AWS_16** check for this RDS instance, with a comment explaining that encryption at rest is intentionally handled or not required.

```
Resources:  
  MyDB:  
    Type: 'AWS::RDS::DBInstance'  
    # cortex:skip=APPSEC_AWS_16:Ensure all data stored in the RDS is securely encrypted at rest  
    Properties:  
      DBName: 'mydb'  
      DBInstanceClass: 'db.t3.micro'  
      Engine: 'mysql'  
      MasterUsername: 'master'  
      MasterUserPassword: 'password'
```

- Option #2 Metadata Section

Suppresses a check using the CloudFormation Metadata section, with a reason provided.

```
Resources:  
  MyDB:  
    Metadata:  
      cortex:  
        skip:  
          - id: "APPSEC_AWS_157"  
            comment: "Ensure that RDS instances have Multi-AZ enabled"  
    Type: "AWS::RDS::DBInstance"  
    Properties:  
      DBName: "mydb"  
      DBInstanceClass: "db.t3.micro"  
      Engine: "mysql"  
      MasterUsername: "master"  
      MasterUserPassword: "password"
```

Dockerfile

Comments can be added to any line in the Dockerfile.

The following comment skips specific Dockerfile checks using inline comments while providing a reason for each suppression.

```
# cortex:skip=APPSEC_DOCKER_5: No need to skip the python check in this image  
# cortex:skip=APPSEC_DOCKER_7: No need to skip graph check in this image  
FROM alpine:3.3  
RUN apk --no-cache add nginx  
EXPOSE 3000 80 443 22  
# cortex:skip=APPSEC_DOCKER_1: required for nginx to run  
CMD ["nginx", "-g", "daemon off;"]
```

Kubernetes

To suppress checks in Kubernetes manifests, annotate using the following format:

```
cortex.io/skip#: <rule_id>=<suppression_comment>  
  
apiVersion: v1  
kind: Pod
```

```
metadata:
  name: mypod
  annotations:
    cortex.io/skip1: APPSEC_K8S_20=Ignore privilege escalation
    cortex.io/skip2: APPSEC_K8S_14
    cortex.io/skip3: APPSEC_K8S_11=BestEffort QoS, no CPU limits
spec:
  containers:
    - name: app
      image: nginx
```

Secrets

Inline suppressions for secrets can be added before, after, or next to the infringing line.

Resources:

```
MyDB:
  Type: 'AWS::RDS::DBInstance'
  Properties:
    DBName: 'mydb'
    DBInstanceClass: 'db.t3.micro'
    Engine: 'mysql'
    MasterUsername: 'master'
    # cortex:skip=APPSEC_SECRET_6 before it
    MasterUserPassword: 'password' # cortex:skip=APPSEC_SECRET_6 or next to it
```

Software Composition Analysis (SCA)

Inline suppressions allow you to ignore specific CVEs, license violations, or all CVEs for a specific package directly in dependency files. Each skip includes a check ID (or package identifier) and an optional reason for the suppression. The exact syntax depends on the package manager.

- For Python (`requirements.txt`), .NET (Paket), Java/Kotlin (`gradle.properties`), Ruby (`Gemfile`)

The skip comment can be added anywhere in the file.

```
# cortex:skip=CVE-2019-19844: Ignore CVE-2019-19844 for all packages in this file
# cortex:skip=jinja2[APPSEC_LIC_1]: Ignore non-OSI license violations for jinja2
django==1.2
jinja2==3.1.0
```

- For JavaScript (`package.json`, bower, json)

- Skip comments can be placed in the metadata section of the non-lock file. When scanning with lock files (for example `yarn.lock`), the suppression applies to related violations.

```
{
  "name": "my-package",
  "version": "1.0.0",
  "//": [
    "cortex:skip=express[APPSEC_LIC_2]: ignore unknown license violations for express",
    "cortex:skip=CVE-2023-123: ignore this CVE for this file"
  ],
  "dependencies": {
    "express": "4.17.1",
    "lodash": "4.17.21"
  }
}
```

- Single skip comment alternative:

```
//: "cortex:skip=CVE-2023-123: ignore this CVE for this file"
```

- For Java (`pom.xml`), .NET (`*.csproj`)

Skip comments can be placed anywhere in the file.

```
<!--cortex:skip=CVE-2023-123: ignore this CVE for the file-->
<!--cortex:skip=junit[APPSEC_LIC_1]: ignore non-compliant license findings for junit-->
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>5.3.9</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

- Go (`go.mod`), Java/Kotlin (`build.gradle`)

Skip comments can be added anywhere in the file. For Go, a skip in `go.mod` also applies to `go.sum`. The following example is for `go.mod`.

```
module example.com/myproject

go 1.17

require (
  github.com/gin-gonic/gin v1.7.4
  github.com/go-sql-driver/mysql v1.6.0
  // cortex:skip=CVE-2023-123: ignore this CVE for this file
```

```
// cortex:skip=github.com/go-sql-driver/mysql[APPSEC_LIC_2]: ignore license violations for this
package
)
```

7 | API documentation

The Cortex Cloud Application Security module includes an API for programmatic access to its features. This enables you to automate security workflows, integrate security insights into your existing tools, and create custom solutions. With the API, you can retrieve application security data, manage configurations, and trigger actions, helping to incorporate security into your development and operational processes.

Required license: Cortex Cloud Runtime Security or Cortex Cloud Posture Management.

For more information, refer to the Application Security API documentation.