Cycode Saas & Broker Considerations

Architecture Overview: SaaS + Broker

The architecture relies on the Cycode Broker, a lightweight container (Docker/Kubernetes) deployed within your network. The Broker acts as a bridge, maintaining the security perimeter by ensuring all connections are initiated from your environment outward.

Key Architectural Constraints:

- **Outbound Only**: The Broker communicates via HTTPS (Port 443) to the Cycode SaaS API and S3 endpoints.
- **No Inbound Access**: Cycode SaaS has no network route to initiate a connection to your environment, Broker, or SCM.

Step-by-Step: New Scan Trigger and Scheduling

The following process describes how a scheduled scan (e.g., a nightly full scan) is executed without Cycode entering your network.

1. Task Generation (SaaS Side)

When the scheduled time arrives:

- The **Scan Service** in the SaaS backend evaluates the schedule and identifies which repositories require scanning.
- It generates specific **Scan Tasks** (e.g., "Sync Metadata" or "Clone Repository").
- These tasks are placed into a queue on the **Broker Server** (a component within the SaaS environment), waiting to be picked up. **The SaaS platform does not push these tasks to the customer.**

2. The Handshake: Long-Polling (Customer Side)

This is the critical step that ensures no inbound connectivity is required.

- The Broker in your network maintains a continuous Long-Polling cycle. It initiates an outbound HTTPS request to api.cycode.com asking, "Are there any pending tasks for me?".
- The Broker waits on this open outbound connection for up to a configurable timeout (default 180 seconds).
- **If no tasks exist:** The connection closes, and the Broker immediately establishes a new outbound connection to repeat the question.
- **If the Scheduled Task exists:** The SaaS Broker Server responds to the Broker's existing outbound request with the task payload (e.g., instructions to scan a specific repo).
-

3. Execution and Response (Customer Side)

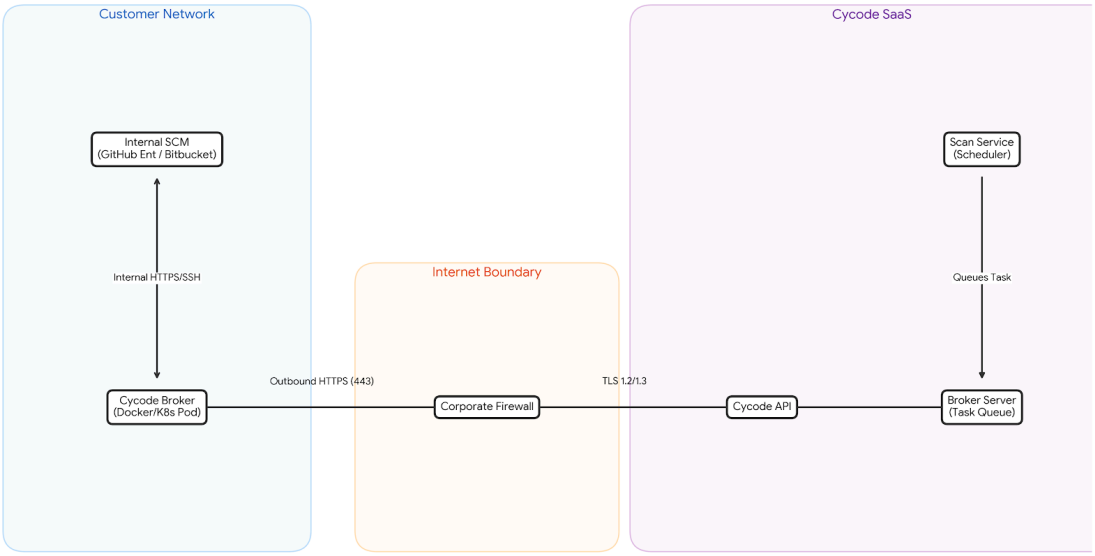
Once the Broker receives the task via the response payload:

- **Execution:** The Broker internally calls your SCM (e.g., GitHub Enterprise, Bitbucket) via internal API/SSH to fetch metadata or perform a shallow clone.
- **Data Return:** The Broker initiates a new outbound HTTPS connection to upload the results (metadata or encrypted zip of the clone) to Cycode SaaS or the designated S3 bucket.
- **Cleanup:** Local temporary data is deleted, and the Broker returns to the Long-Polling state (Step3).

Diagrams for Architecture Review

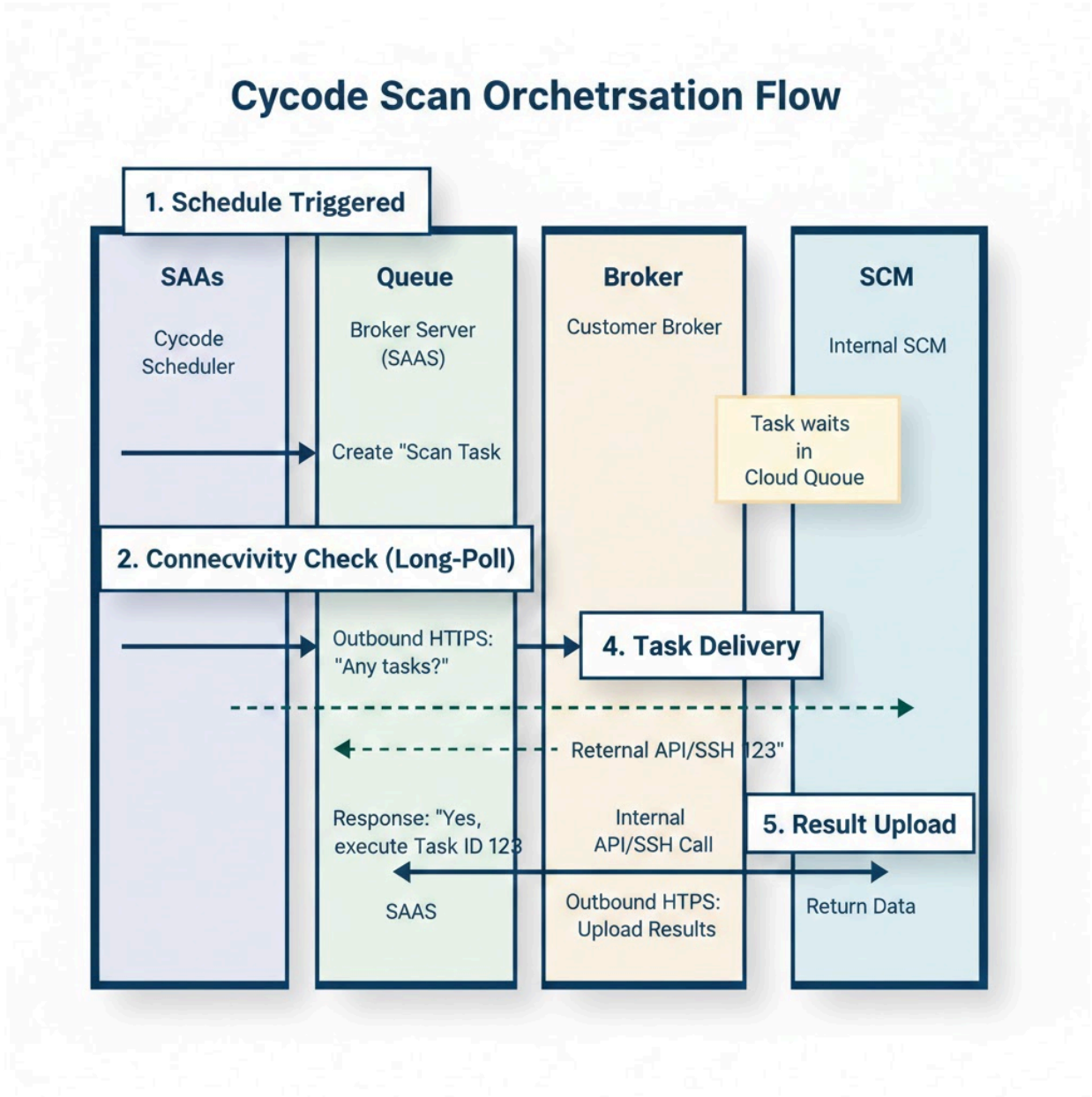
1. Connectivity Architecture

This diagram illustrates that the Broker is the only active initiator of traffic.



Sequence Diagram: The "No Inbound" Trigger

This sequence shows that the SaaS creates the job, but the Broker retrieves it.



1. Data Access & Collection Scope

To perform security testing, the solution accesses specific data points. This retrieval is performed entirely by the Broker within your network, ensuring the SaaS never directly queries your SCM.

- **Repository Metadata:** Names, IDs, URLs, visibility flags, branch lists, etc.
- **Workflow Metadata:** Pull Request (PR) details (IDs, titles, source/target branches) and Commit metadata (hash, author, timestamp).
- **Repository Content:** A snapshot of tracked files (code, configuration, manifests, Dockerfiles) required for analysis. For Secrets or Leakage modules, this will include commit history if specifically enabled.

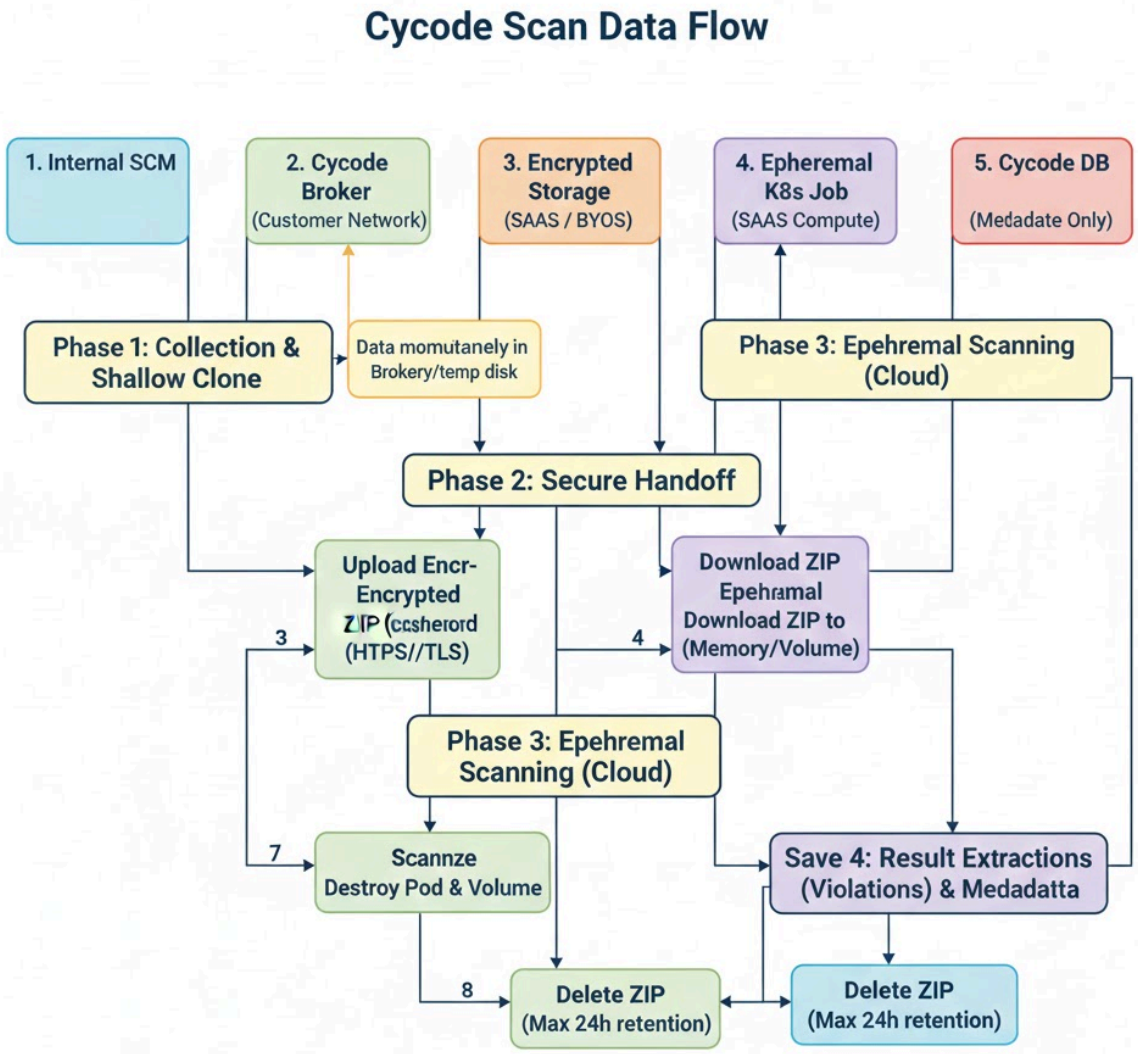
2. Execution Architecture: Who Scans What?

The scanning process is a hybrid operation designed to keep heavy compute in the cloud while keeping connectivity control on-premise.

Component	Location	Role
Cycode Broker	Your Network (Docker/K8s)	Acts as the data fetcher. It performs the git clone or API calls to your SCM to retrieve metadata and zip the source code,.
Cycode's Scanners	Cycode SaaS	Runs the actual analysis. For some scanners, like SCA or SAST, These are ephemeral Kubernetes jobs (pods) that spin up on-demand to scan the data and destroy themselves immediately after. For secrets, currently, it's long-living k8s pods that are not ephemeral (Planned for Q1 to be changed)
S3 Bucket	Cycode SaaS (or BYOS)	Intermediate encrypted storage. The Broker uploads the zipped code here so the SaaS scanners can access it without entering your network.

3. The Lifecycle of a Scan (Data Handling Flow)

The following diagram and step-by-step walkthrough illustrate the "Life of a Repository Clone," demonstrating that source code is transient and strictly managed.



Step-by-Step Data Handling

1. Collection (Customer Environment)

- The **Broker** receives a task to clone a specific repository.
- It performs a local `git clone` (shallow clone usually) to its local file system.
- This data exists on the Broker container only long enough to be compressed into a ZIP file.

2. Transmission (Transit)

- The Broker uploads the ZIP file via outbound HTTPS (TLS 1.2/1.3) to a designated **S3 Bucket**.

- This bucket is encrypted (AES-256) and private.
- Note: If you use the "Bring Your Own Storage" (BYOS) option, this bucket resides in your own AWS account, giving you full control over the data at rest,.

3. Analysis (SaaS Environment)

- The upload triggers the **Scan Service** to spawn an **Ephemeral Kubernetes Job** or to run a task inside a k8s pod (as described above).
- This job mounts a temporary, ephemeral volume and downloads the ZIP from S3.
- Scanning engines (e.g., SAST, SCA, etc) read files from this ephemeral volume, identify violations, and generate a report, but not (yet) for secrets (as described above)

4. Cleanup & Retention (Post-Scan)

- **Immediate Deletion:** Once the scan completes, the ephemeral Kubernetes job and its attached volume are destroyed.
- **S3 Retention:** The zipped source code in the S3 bucket is deleted immediately after processing is confirmed, with a hard-stop policy of 24 hours. It is never backed up.
- **Persistence:** The only data persisted long-term in the Cycode Database is **Metadata** (repo names, timestamps) and **Findings** (the specific lines of code containing the vulnerability, snippet context, and remediation advice). This can be controlled as well (saved/not saved at all/saved partially)

4. Summary of Data Storage & Encryption

State	Location	Storage Medium	Encryption	Retention
During Collection	Broker (On-Prem)	Container Filesystem	Disk/Volume Encryption (Customer Managed)	Seconds/Minutes (deleted after upload)
At Rest (Transit)	S3 Bucket	Object Storage	AES-256 (Server-Side Encryption)	< 24 Hours (Hard delete)
During Scan	SaaS Scanner	Ephemeral Volume and/or memory	K8s Ephemeral Disk OR in the k8s pod (depands on the scanner, see above)	Duration of scan only (Minutes)
Long-Term	Cycode DB	Managed Database	Encrypted at Rest	Metadata only & Findings retained for history/audit

Architectural Note for Reviewers: The "No Inbound" architecture means the SaaS never "reaches in" to scan. Instead, your Broker "brings out" a temporary snapshot, which is analyzed in a secure, ephemeral sandbox and then immediately discarded. The full source code is never permanently stored on the Cycode platform.

1. What Information is Transmitted to SaaS?

Data transmission occurs in two distinct phases: the temporary upload required for analysis and the persistent data stored for reporting.

A. Transmitted Data (The Upload)

To perform the analysis, the Broker initiates an outbound HTTPS connection to transmit:

- **Repository Metadata:** Git metadata including branch lists, commit hashes, author details, timestamps, and PR titles/IDs, etc.
- **Repository Content (Artifacts):** A **zipped snapshot** of the tracked files required for the scan. This is uploaded directly to the S3 bucket.
    - Note: If using the "Bring Your Own Storage" (BYOS) option, this zip file is uploaded to **your own** AWS bucket, not Cycode's.

B. Stored Data (Post-Processing)

Once the ephemeral scan job completes (and the zip file is deleted), the following data persists in the Cycode SaaS platform:

- **Violations/Findings:** The specific detection type (e.g., Secret, SAST issue), severity, and rule ID.
- **Location Pointers:** File paths, line numbers, and the associated commit hash or PR ID.
- **Code Snippets:** A small snippet of code surrounding the violation to aid in triage.
    - Security Note: For **Secrets** detection, raw values are **never** stored if configured that way. Only masked values (e.g., ******) or salted hashes are retained for validation.
- **Inventory & Context:** Dependency graphs, package versions, and audit workflow states (e.g., "Ignored", "Assigned to User", etc.

C. Format and Transport

Transport Protocol: All data flows from Broker $\to$ SaaS over outbound HTTPS (Port 443) using TLS 1.2/1.3.
Storage Encryption: Data at rest is encrypted using AES-256.

2. Practical Options to Minimize Exposure

You can configure the SaaS solution to significantly reduce the amount of sensitive data leaving your perimeter. Below are the standard "levers" available to architects.

Level 1: Configuration & Policies (Low Effort)

- **Snippet Redaction / Removal:** Cycode allows you to configure options to disable **snippet storage.** In this mode, the SaaS stores only the file path and line number. Alternatively, you can enforce heavy obfuscation (redacting string literals) within the UI.

Level 2: Architectural Adjustments (Medium Effort)

- **Bring Your Own Storage (BYOS)**: Instead of uploading the zipped code to Cycode's S3 bucket, you can configure the system to use an S3 bucket in your **own AWS account**.
    - **Benefit:** The raw source code artifacts remain within your cloud account and access control boundary at all times. Cycode's scanning engine accesses them only temporarily via a pre-signed URL and never retains a copy.

Summary Table for Decision Making

Strategy	Effect on Data Exposure	Trade-off
Default SaaS	Sends Code Zip + Snippets	Full visibility/context in SaaS UI.
Snippet Redaction	Sends Code Zip; Stores No Snippets	Triage requires developers to check local code (no context in UI).
BYOS	Code Zip stays in Your Cloud	Requires AWS S3 bucket management on your side.