AppSentinels.ai
Application Security Re-invented

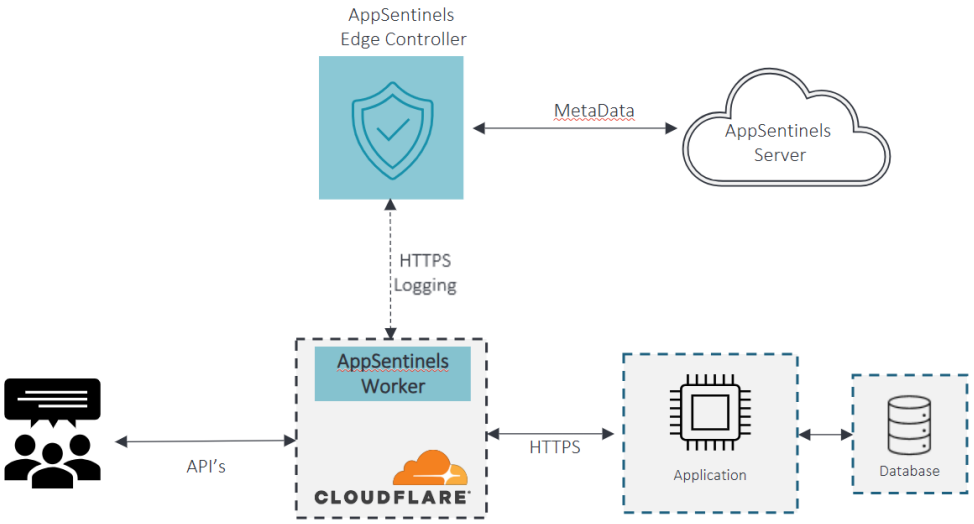
AppSentinels API Security Platform

Cloudflare Integration

Table of Contents

1. Introduction

AppSentinels' Cloudflare Worker performs API traffic logging at the edge to provide real-time visibility and optional enforcement. AppSentinels will provide a configurable JS based worker which will selectively perform logging and enforcement of API traffic in conjunction with AppSentinels edge controller.



This document will help,

- Deploy the AppSentinels Worker

- Configure secure HTTPS logging to the Edge Controller

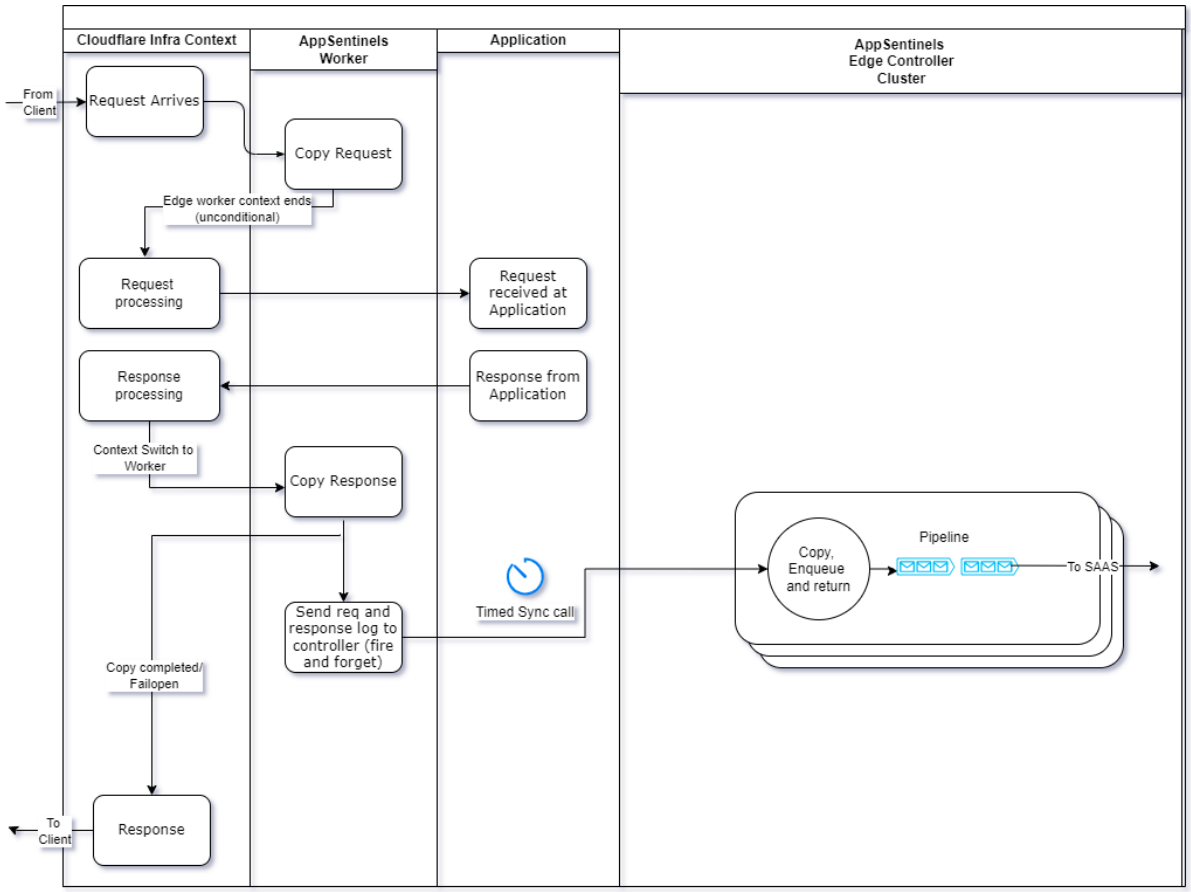
- Verify and Debug

---

2. Architecture & Design

Flow

1. Client → Cloudflare Edge → **AppSentinels Worker** (observes request)
2. Worker proxies to origin → receives response, observes it
3. Worker assembles **merged log** (req/resp + timing + headers)
4. Worker **asynchronously** ships merged log to **Edge Controller** over HTTPS (fail-open)



Design Goals

- **Fail-open**: never block user traffic due to logging issues
- **Predictable cost & latency**: Capped payload logging; short timeouts; static page bypasses, non-API method bypasses
- **Signal over noise**: headers/body captured only when useful

- **Operationally friendly**: simple config knobs; clear debugging paths

Key Components

- Cloudflare Worker (JavaScript)
- Edge Controller (HTTPS endpoint: POST `/mergedlog`)

3. Deployment

Prerequisites

- Cloudflare account and a proxied zone
- Edge Controller reachable via HTTPS with public DNS

Creating AppSentinels worker

1. Log into Cloudflare account
2. Choose the website for enabling AppSentinels worker
3. Select on Workers (sidebar) and name it as **appsentinels-apisec-logger**
4. Select HTTP handler and then click on Create service
5. Click on Quick Edit button and copy paste the JS worker code provided by AppSentinels
6. The JS code requires logging endpoint to be configured along with its scheme. Open the JS script and populate **REMOTE_CONTROLLER_ENDPOINT_URL,**

   For example:

   REMOTE_CONTROLLER_ENDPOINT_URL = https://onprem-controller:9004/mergedlog?sensor=cf

7. Save and deploy

Enable AppSentinels worker for a website

1. Go to the websites (on the sidebar) and select the website which need to monitored and protected by AppSentinels Security Solution
2. Click on Worker Routes and then Add Route
3. The worker can be enabled for specific routes on the website using wildcards. Possible combinations,

   website-hostname/*

   *website-hostname

   website-hostname/path*

   subdomain1. website-hostname/*

   In case edge controller is deployed as an origin server under a subdomain, it is mandatory to specify “no worker” for the controller’s subdomain. This will

ensure the worker logging doesn't create API loop inside the Cloudflare worker network

For ex: If controller was mapped to domain appsentinels-controller.website-hostname/*, then no worker for this route.

4. Save this config and worker should start seeing the API traffic now

4. Configuration

Primary knobs in the Worker

- `REMOTE_CONTROLLER_ENDPOINT_URL` – HTTPS URL to Edge Controller (must include query flags like `sensor=cf&cap=nohb`).
- `MAX_SUPPORTED_PAYLOAD` – Max bytes to capture from request/response bodies (default `131072`).
- `SUPPORTED_CONTENT_TYPES` – Substrings matched in `Content-Type` for body capture.
- `BYPASS_METHOD` – Methods to skip (default `OPTIONS, HEAD`).
- `DEFAULT_LOGGING_TIMEOUT` – Max ms the Worker will wait when sending logs (default `1000`).
- `SENSOR_INSTANCE` – Optional instance label injected into response headers for traceability.
- `SKIP_EXTENSIONS` – Static file extensions to bypass (avoid noisy/large bodies).

5. Secure HTTPS Logging to Controller

Cloudflare worker will be performing secure HTTPS logging onto AppSentinels controller which acts like a server. The worker will, as part of HTTPS, perform server validation. It is essential that controller be provided with server cert and private key.

There are couple of ways to deploy the controller depending upon your requirements,

Controller Deployment Option 1 - Deploy as independent internet endpoint

This deployment option requires controller be deployed as any other internet exposed endpoint and requires that,

1. Controller endpoint be exposed on the internet
2. DNS mapped to controller IP
3. Configure REMOTE_CONTROLLER_ENDPOINT_URL in the worker accordingly

4. Provide domain specific public and private certs to controller for HTTPS validation performed by AppSentinels worker

Choice of this deployment is preferred if you have complete control over deployment and capability to provision DNS/Certificates etc.

Controller Deployment Option 2 – Deploy as another origin server

This method leverages Cloudflare for secure logging. The method involves,

1. Provision controller and provide an IP address to it.
2. Create origin server certificate on Cloudflare (under domain go to SSL/TLS/Origin Server and then Create Certificate) and mount them onto controller
3. Create a subdomain for controller eg: subdomain1.website-hostname. Update Cloudflare DNS records by creating an A record for the subdomain and mapping it to publicly reachable controller IP
4. Configure REMOTE_CONTROLLER_ENDPOINT_URL in the worker accordingly

Do not attach the worker to this controller's route, ever! it is mandatory to specify "no worker" for the controller's subdomain. This will ensure the worker logging doesn't create API loop inside the Cloudflare worker network

For ex: If controller was mapped to domain appsentinels-controller.website-hostname/*, then no worker for this route.

8. Debugging & Verification

- **Cloudflare Dashboard** → Worker → *Logs*: live tail errors and metrics.
- **Wrangler**: `wrangler tail` for local log streaming.
- **Functional test**: Hit a routed API and confirm entries appear in AppSentinels (API catalog, latency, headers).
- **Network sanity**: If nothing arrives, validate DNS/TLS to the controller and check firewall rules.

9. Security Notes

- Always use **HTTPS** to the controller.

---

10. Upgrades & Maintenance

- Track **APPSENTINELS_PLUGIN_VERSION** in the Worker.

- When changing constants (timeouts, payload caps), roll out gradually and watch controller load.

Appendix A – Worker Code

Worker code:

The complete Worker source code is available upon request.

Revision	Date Modified	Author	Comments
1.0	01-Aug-24	Sachin	Initial Draft
1.1	04-Dec-24	Sachin	Secure logging
1.2	16-Aug-25	Sachin	Reformatted document
1.3	01-Nov-25	Sachin	Update for various controller deployment choices