What is a webhook?

Archived: 2026-04-05 23:40:18 UTC

Updated February 1, 2024•*8*-minute read

What is a webhook?

A webhook is a lightweight, event-driven communication that automatically sends data between applications via HTTP. Triggered by specific events, webhooks automate communication between application programming interfaces (APIs) and can be used to activate workflows, such as in GitOps environments.

Because webhooks can connect event sources to automation solutions, they are 1 way to launch event-driven automation to perform IT actions when a specified event occurs.

Webhooks for application development

What is an API?

An API is a set of definitions and protocols for building and integrating application software. Communication between APIs is sometimes referred to as a contract between an information user and an information provider—establishing the content required from the consumer (the call) and the content required by the producer (the response). This relationship is also described as the *client* app which calls the *server* app, but these roles can be reversed, depending on which app is requesting data in a given situation.

Web APIs typically use HTTP to request data from other apps and define the structure of response messages, which usually take the form of an XML or JSON file. Both XML and JSON are preferred formats because they present data in a way that's easy for other apps to manipulate.

When a client API requests data from a server API, it's calling to see if a certain event has occurred—in other words, whether the server's data has changed in a way that might be useful to the client. In this process (known as *polling*), the client sends an HTTP request at regular intervals until the server's API sends the relevant data, which is sometimes called the *payload*.

The client app doesn't know the state of the server app, so it polls the server's API for an update—calling over and over until the specific event occurs—but the server will only send the requested data once that information is available. The client app has to keep asking for the update and wait until the relevant event happens.

Read about best practices of successful API teams

How are Webhooks different?

To set up a webhook, the client gives a unique URL to the server API and specifies which event it wants to know about. Once the webhook is set up, the client no longer needs to poll the server; the server will automatically send

the relevant payload to the client's webhook URL when the specified event occurs.

Webhooks are often referred to as *reverse APIs* or *push APIs*, because they put the responsibility of communication on the server, rather than the client. Instead of the client sending HTTP requests—asking for data until the server responds—the server sends the client a single HTTP POST request as soon as the data is available. Despite their nicknames, webhooks are not APIs; they work together. An application must have an API to use a webhook.

The name *webhook* is a simple combination of *web*, referring to its HTTP-based communication, and the *hooking* programming function that allows apps to intercept calls or other events that might be of interest. Webhooks hook the event that occurs on the server app, and prompt the server to send the payload to the client via the web. Jeff Lindsay helped to popularize the concept with his 2007 blog post, "Web hooks to revolutionize the web."

IT teams use a variety of methods to protect apps that communicate via webhooks. Most webhook-enabled apps add a secret key to the request header of the payload, so that the client can confirm the server's identity. Webhooks are often protected with Mutual Transport Layer Security (mTLS) authentication, which verifies both client and server before the payload is sent. It is also common for client apps to use SSL encryption for the webhook URL, to ensure the transferred data remains private.

Webhooks:

- **Eliminate the need for polling.** This saves resources for the client application.
- **Are quick to set up.** If an app supports webhooks, they are easy to set up through the server app's user interface. This is where the client enters their app's webhook URL and sets some basic parameters, like which event they are interested in.
- **Automate data transfer.** The payload is sent as soon as the specified event happens on the server app. This exchange is initiated by the event, so it happens as quickly as the data can be transferred from server to client—as real-time as any data transfer can be.
- **Are good for lightweight, specific payloads.** Webhooks rely on the server to determine the amount of data it sends, leaving it to the client to interpret the payload and use it in a productive way. Since the client does not control the exact timing or size of the data transfer, webhooks deal with small amounts of information between 2 endpoints, often in the form of a notification.

Webhooks for infrastructure automation

Webhooks are most commonly used to simplify communication between two applications, but they can also be used to automate Infrastructure-as-code (IaC) workflows and enable GitOps practices.

What is Infrastructure as Code (IaC)?

Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of manual processes. Version control is an important part of IaC, and configuration files should be under source control just like any other software source code file. Deploying infrastructure as code also means that infrastructure can be divided into modular components that can then be combined in different ways through automation.

Automating infrastructure provisioning with IaC means that developers don't need to manually provision and manage servers, operating systems, storage, and other infrastructure components each time they develop or deploy an application. Codifying infrastructure provides a template to follow for provisioning. This can still be accomplished manually, but these processes can be automated with an enterprise-level desired state engine, such as Red Hat® Ansible® Automation Platform.

What is GitOps?

Often considered an evolution of IaC, GitOps is a strategic approach to managing infrastructure and application configurations using Git, an open source version control system. Following GitOps practices, developers use Git as a single source of truth for declarative infrastructure and applications, and use Git pull requests to automatically manage infrastructure provisioning and deployment. The Git repository contains the entire state of the system so that the record of changes is visible and auditable.

Where do webhooks come into it?

Webhooks reduce the steps required to implement and manage git-centric deployment pipelines, and can be used to automatically launch entire IaC workflows. In a GitOps environment—with a git repository as the source of truth—a webhook functions the same as it does between 2 applications; when triggered by a specified event, 1 API sends a payload to another API. The difference lies in what type of event triggers the webhook and what the recipient does with the payload.

In this context, the git repository plays the role of the server app, while the desired state engine—which manages the state of infrastructure—plays the role of the client app. Webhooks can be used to notify the desired state engine every time a change is made in the git repository. If a piece of code is updated and pushed to the repository, this event will trigger the webhook. The repository then automatically sends the payload to the desired state engine's webhook address, informing it of the code change.

If the desired state engine supports automation, these webhooks can also launch IaC workflows—turning a code change into automated action. For example, system administrators can set up automation that runs whenever a webhook's payload is received, to automatically apply new code changes on their managed hosts and restore them to a default state. This method of using webhooks to trigger automation can be extended to perform other IT actions without human involvement, enabling a process known as event-driven automation.

The only difference is the source of truth; instead of connecting a desired state engine to a git repository—which relies on humans to push code updates—a webhook can connect it to a third-party tool that monitors a source for specific events. Once these event sources detect a targeted event and fire the webhook, the payload can launch automation that takes immediate action to resolve the event at any time of day, without requiring IT staff to press a single button.

Learn how Ansible Automation Platform uses webhooks for GitOps

Webhooks for event-driven automation

Event-driven automation is the process of responding automatically to changing conditions in an IT environment, to help resolve issues faster and reduce routine, repetitive tasks. With event-driven automation, IT teams can codify responses to any event—like hardware issues, distributed denial-of-service (DDoS) attacks, memory shortages, or application failures—so that the necessary action is automatically executed when the event occurs.

Event-driven solutions rely on third-party tools or plugins—like ServiceNow, Kafka, Prometheus, Sensu, Dynatrace, and Appdynamics—to monitor a source for events. Webhooks can be used to connect these event sources with an automation platform, so that when a source detects an event, the webhook triggers the appropriate automated response.

IT teams can adopt event-driven automation incrementally to reduce mean time to resolution (MTTR) and perform functions that still require human involvement—like automatically creating a ticket—and gradually work towards fully automatic remediation, so that the appropriate action is automatically taken when a particular issue occurs.

How can Red Hat help?

Red Hat Ansible Automation Platform is an end-to-end automation platform designed to help IT teams create, manage, and scale automation across the entire enterprise. You can use webhooks to integrate Ansible Automation Platform with a Git repository—via a service like GitHub or GitLab—to support IaC and GitOps practices. Once a repo link is set up, Ansible Automation Platform catches Git commits from the Git system and uses those events to trigger automation jobs to update projects, manage inventories, and perform deployments.

Webhooks allow you to automatically activate automation when events occur in your source control system. This eliminates the need for additional CI/CD tools—like Jenkins—to monitor repositories and launch automation jobs when changes occur, simplifying your GitOps workflow and streamlining operations. Because Ansible Automation Platform works with a wide variety of development and deployment tools, you can tailor your GitOps workflow to your preferred tools and processes.

Mission-critical automation with Event-Driven Ansible

As a part of Ansible Automation Platform, Event-Driven Ansible provides the event-handling capability needed to automate time-consuming tasks and respond to changing conditions in any IT domain. It can process events containing discrete intelligence about conditions in the IT environment, determine the appropriate response to the event, then execute automated actions to address or remediate the event.

Event-Driven Ansible can be used to automate IT service management tasks—such as ticket enhancement, remediation, and user management—and a variety of other IT processes. It connects *sources* of events with corresponding *actions* via *rules*. Ansible Rulebooks define the event source and explain—in the form of conditional “if-this-then-that” instructions—the action to take when the event occurs. Based on the rulebook you design, Event-Driven Ansible recognizes the specified event, matches it with the appropriate action, and automatically executes it.

You can use fully-supported generic webhooks to connect Event-Driven Ansible to event sources, but Event-Driven Ansible also offers an ecosystem library of source plugins built by partners for their specific technology.

These fully supported plugins allow you to build event-driven automation without having to write code or program webhooks for every new event. You just need to know which event you're interested in and what action you want to happen—then write these instructions into an Ansible Rulebook, which enables any event to automatically execute any preexisting Ansible Playbook or automation workflow you desire.

Learn more about Event-Driven Ansible

Hear from an expert

Keep reading

What is patch management?

Patch management is the process of identifying, testing, and installing system updates to improve security and performance in operating systems and applications.

Why choose Red Hat for automation?

Red Hat Ansible Automation Platform includes all the tools needed to share automation across teams and implement enterprise-wide automation.

What is an Ansible Playbook?

An Ansible Playbook is a blueprint of automation tasks executed on hosts.

Automation and management resources
