Massiv.io

Combating Engineering Bureaucracy with Active Data Aggregation

In the previous two blog entries we discussed Integral, our platform for real-time architecture. One of the pillars of real-time architecture is "active data aggregation," the ability to assemble and disseminate architecture information from code, configuration, wikis, runtime metrics and other sources on a continuous basis. This article dives into the details of active data aggregation and how it helps cut through engineering bureaucracy.

***

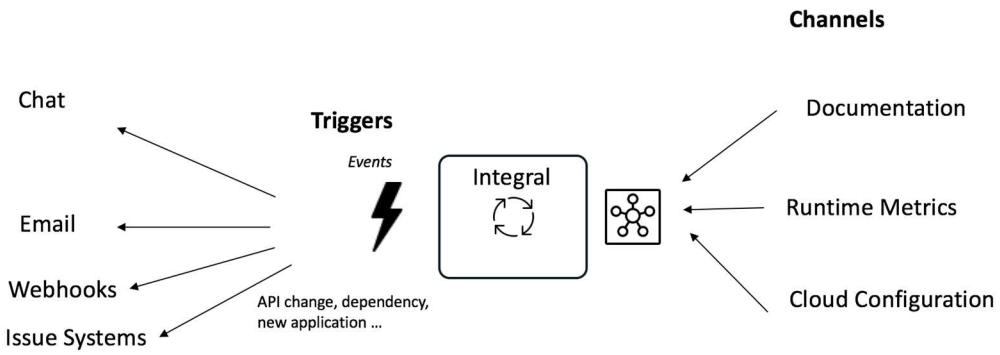
Despite advances in our engineering toolchains over the last ten years, architecture tooling remains a relative backwater. While the rest of our development process becomes increasingly automated, tools for managing application architecture remain stuck in the past:

- Whiteboard and paper – perhaps the most common architecture tooling – are total silos, existing in a conference room or on desk somewhere, quickly becoming obsolete as applications evolve
- Reverse engineering and UML tools provide some automation but they require manual setup and do not offer effective collaboration or sharing features

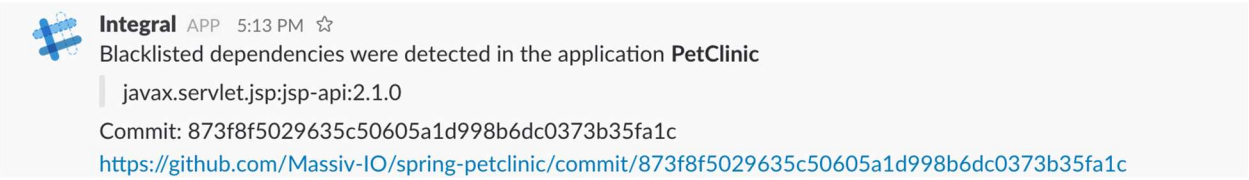
The lack of readily-accessible architecture knowledge not only affects ivory-tower architects. It impacts engineering team members from development to operations on a daily basis. Are you having trouble keeping track of which code repositories map to your enterprise systems and services? Do you wish you had an easy way to search for uses of a particular technology across all services in your organization? Would receiving notifications when a breaking API change is committed help your team plan better? How much time would be saved if operations could receive detailed dependency information on a new service well before it is released for deployment?

The result of poor architecture knowledge is engineering bureaucracy. Instead of writing code, performing reviews, and communicating application changes to stakeholders across an organization, time is consumed tracking down and producing reports that quickly become outdated.

As a first step in cutting through this bureaucracy, Integral weaves together disparate information sources to construct a holistic picture of your enterprise architecture. Data collection is done through *channels.* However, this is only half of the equation. After analysis, this architecture information is presented not only as diagrams but also delivered as events team members can act on. In Integral, events are created by *triggers,* for example, a check-in that adds a blacklisted library dependency. Active data aggregation looks like this:



As shown in the figure, Integral can deliver trigger events to wherever you prefer to receive notifications or your development workflow dictates. In this example, Integral has sent a Slack notification about the addition of a blacklisted dependency:

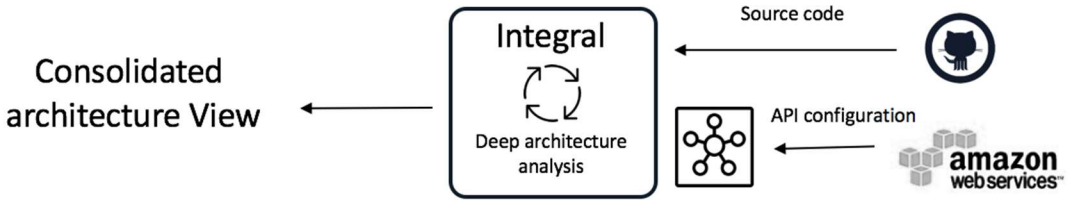


In keeping with our platform design, ***Integral is not another dashboard or portal, but a fabric for accessing and disseminating critical architecture information in your organization***. Now, let's go over *channels* and *triggers* in more detail.

Channels: Architecture Data Collection

Channels are plugins that give Integral the ability to collect information not only from code but from a variety of sources. These may include cloud service provider APIs, application performance management tools (APM), wikis, and document storage such SharePoint, Drobox or Google Drive.

In our previous blog entry on deep architecture analysis, we discussed how Integral analyzes source code and configuration to produce a service model that describes the core architecture of an application. Once this is complete, channels supply data which is overlaid on the model. For example, the Amazon Web Services Channel (AWS) contributes API gateway configuration which is added to the enterprise architecture view.



The gateway configuration is not only important for generating proper endpoint information. In many cases, such as highly distributed microservice environments, services themselves may communicate via the gateway. In order to wire these services correctly and show their dependencies, the current gateway configuration must be used.

At Massiv.io we are currently adding channels to draw in other sources of information. One of the most important of these will be metrics from Application Performance Management (APM) products such as NewRelic and AppDynamics. This will allow stakeholders to view runtime and SLA performance in the context of their enterprise architecture. It will also provide new insights such as how backend SLA performance impacts upstream applications and APIs by following application wiring in an enterprise architecture.

Another channel in development will index and correlate documentation from multiple sources such as cloud storage, chat services, wikis, and SharePoint to allow contextual distributed search and information retrieval.

Triggers: Architecture Events

Triggers are responsible for generating and delivering events based on certain actions. Previously we used the example of creating a trigger when a commit includes an incompatible API change. Let's walk through that.

Triggers can optionally have a filter and have a target. Once a trigger is selected, it can be optionally assigned a filter. In our example, we could choose to filter on application name using a regex or decide not to include a filter, in which case all applications will be monitored. Triggers are then assigned a target such as an email address or Slack channel.

When a trigger is enabled, any application matching the trigger filter will be monitored. If a commit for a matching application includes a change that modifies its API signature in an incompatible way, an event will be generated and sent to the target. Behind the scenes, when the commit is made, Integral reprocesses the source code, determines its new service model, and compares it against the previous model. If, for example, a REST operation is removed, the API trigger will be fired and the event sent.

This only touches the surface of what triggers are capable of automating. In the future, we will be opening our platform via web hooks, so users can write their own filters and targets, allowing them to initiate custom responses to architecture events.

***

In subsequent blog pieces we'll cover more aspects of our real-time architecture platform. If you would like additional information, a demo or want to try Integral out, please don't hesitate to contact us at support@massiv.io. In the meantime, keep up the fight against engineering bureaucracy!