THE DEFINITIVE GUIDE FOR PRODUCTION TOOLS

by. Josh Dreyfuss

Introduction

Deploying an application in production is immensely complicated. The proliferation of hardware, 3rd party interactions, and environmental specifications means that there are a lot of elements to consider. Very few apps work in complete isolation, and there are whole ecosystems out there dedicated to improving the experience of working in production.

Understanding what tools are out there and which are worth checking out for you is critical to succeed in the modern development landscape.

This guide is designed to provide some direction and overview into the world of production tools for developers. We'll be covering alerting tools, log management tools, visualization and metrics tools, and APM tools for production. As you can't get an app into production without some pre-production tools, we'll also cover a few of those, including build tools and deployment automation tools. Since we at Takipi are focused on JVM-based languages, we only look at tools that support Java here and provide our analysis from a Java-based perspective. Many of these tools support multiple languages and heterogeneous environments however, and we do our best to incorporate that. So essentially, this is a Java-focused, but not Java-exclusive guide.

If we're missing your favorite tool or you're having trouble understanding something or you just want to tell us that we're the best (or worst), feel free to contact us at hello@takipi.com.

Chapter 1.....Alerting Tools

Chapter 2.....Log Management Tools

Chapter 3.....Visualization and Metrics Tools

Chapter 4.....Application Performance Management Tools

Chapter 5.....Build Tools

Having all the automation and management in the world can only take you so far if you have no clue what's actually happening in your production app. Alerting tools provide insight into the workings and status of your app. Some reach out to you to notify you when something worth your attention pops up, some provide a dashboard for error tracking, and others test your app on a continuous basis. There are alerting tools for the developer-side with issues like errors and exceptions, and for the operations side with issues like uptime.

Visibility is an important consideration for alerting tools. The more a tool requires you to integrate it into your environment, the more you are dependent on the tool's frameworks, which can lead to issues down the road.

Exceptions / Errors :

Sentry

Takipi

Airbrake

Raygun

Ops / Uptime:

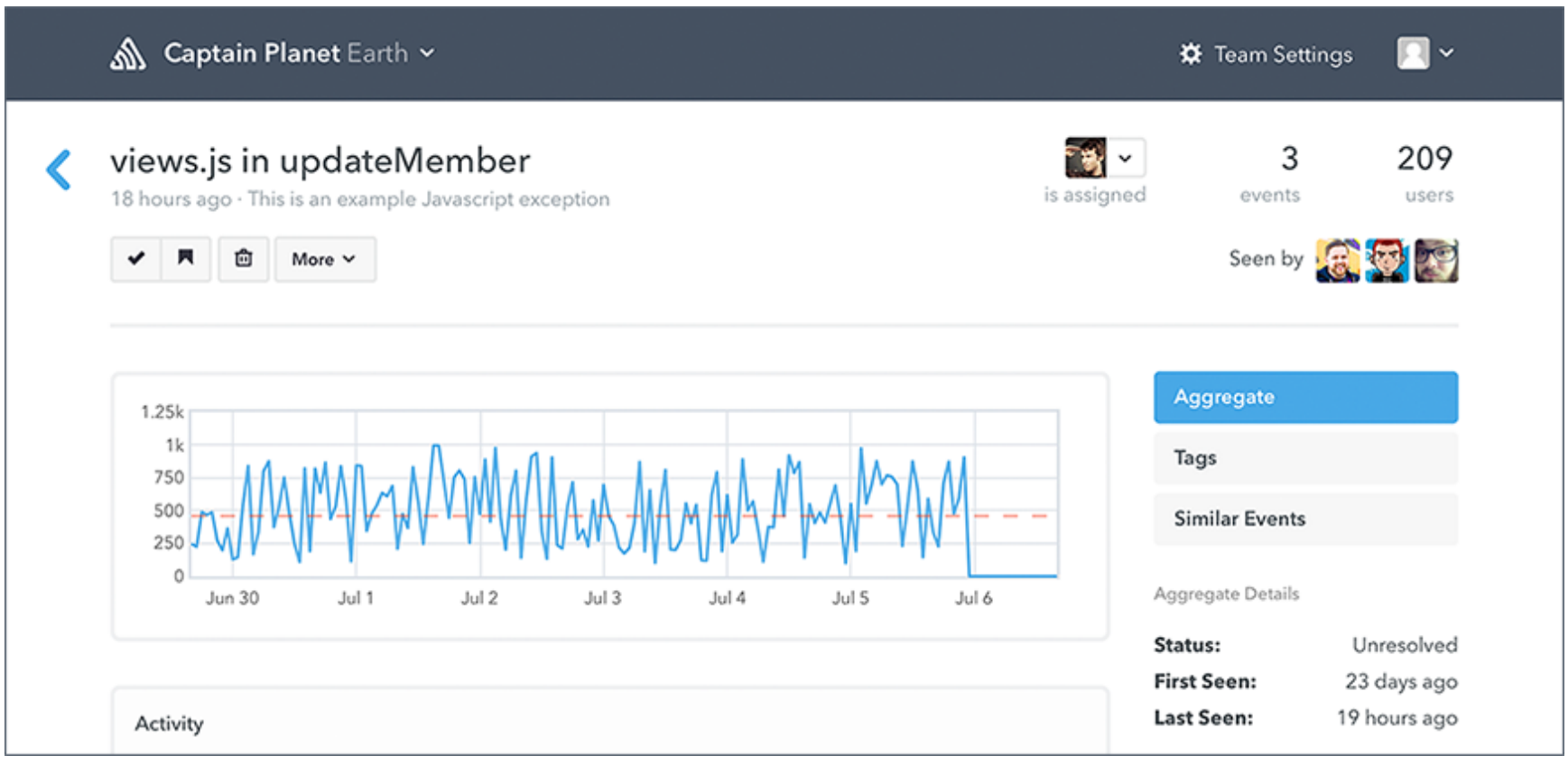
PagerDuty

Pingdom

Exceptions / Errors

Sentry:

Sentry is an open-source solution that serves as a real time event logging and aggregation platform. It monitors errors and displays when, where, and to whom they happen, promising to do so without relying solely on user feedback. Supported languages and frameworks include Ruby, Python, JS, Java, Django, iOS, .NET and more.



When to use it: A differentiator for Sentry is the fact that it's open source. If an open source tool with a greater number of SDKs and plugins is something you value, then you should consider Sentry. It's also one of the most popular error tracking tools in terms of deployment numbers, so if you like going with the most popular tool, that would be another reason to look here.

Price: Three tiers: $24/mo, $79/mo, $199/mo. Tiers are based on data retention and the number of events you want to track per minute.

Pros:

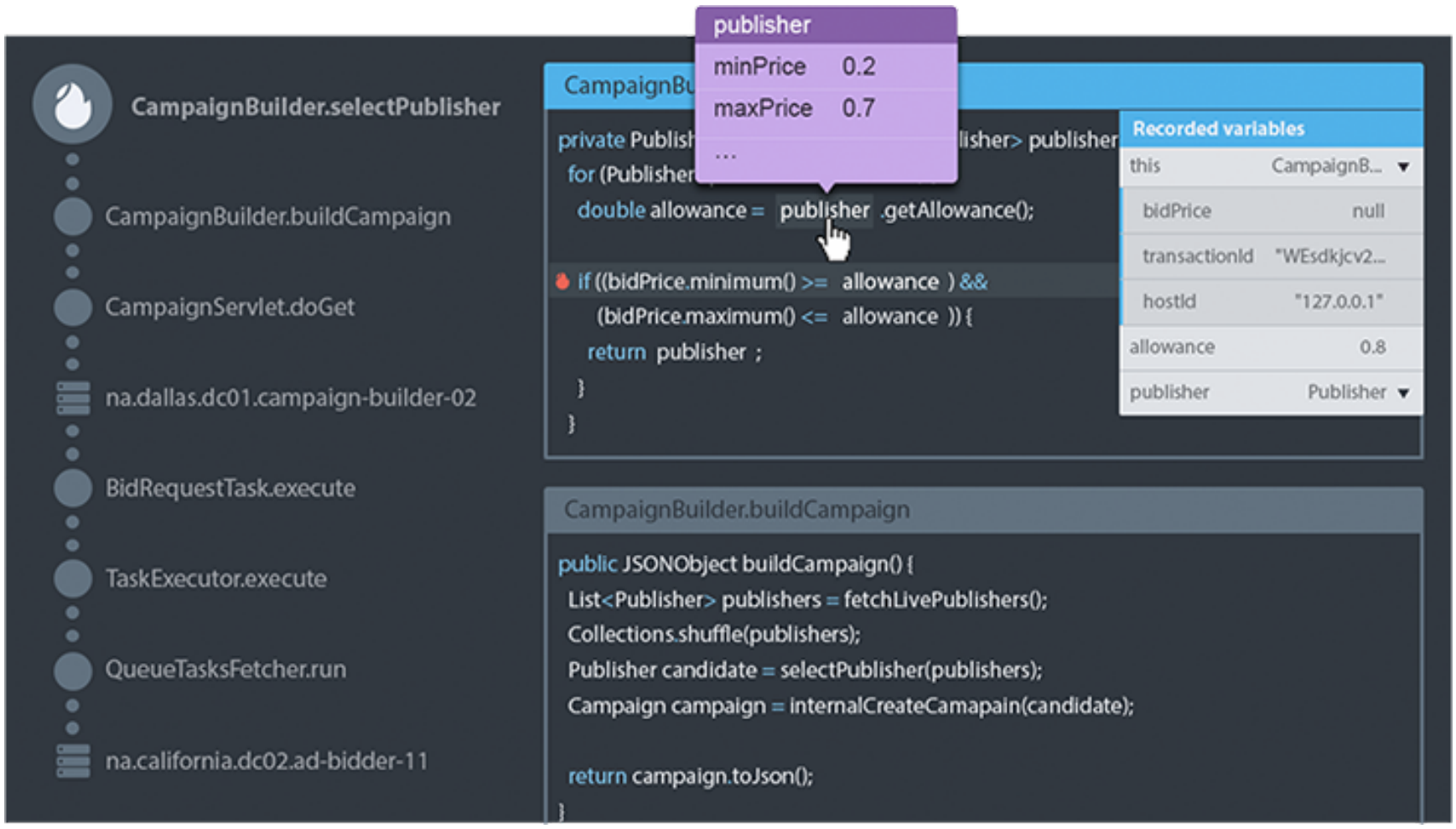
- See the impact of new deployments in real time.
- Provide support to specific users interrupted by an error.
- Detect and thwart fraud as it's attempted – Sentry provides notifications of unusual amounts of failures on purchases, authentication, and other sensitive areas.
- External Integrations – GitHub, HipChat, Heroku, and many more.
- Through their Java client Raven, Sentry supports the widest range of Java logging frameworks among the alerting tools that rely on them, including Logback, Log4j, and Log4j2.

Cons:

- Requires a binary dependency to operate.
- Documentation is fairly limited.
- If you're not using one of the supported frameworks, the alternative is a manual setup
- Security can be a concern, as filtering for personally identifiable information can be impossible, requiring manual effort on your part.

Takipi

Takipi tells you when and why your Java/Scala code breaks in production. It detects all types of errors and gives you the code and variable state when they happened. Takipi runs as a Java agent and does not require code changes, binary dependencies, or build configurations.



When to use it: If you have a production environment in Java, Scala, or other JVM-based language. If you want to capture the entire picture of your errors and want the actionable information you need to fix them. If you deploy frequently and want to track errors that result from new deployments. If you're facing complex errors in production that are almost impossible to reproduce.

Price: Free tier, sliding scale pro tier of $69/server/month, and enterprise tier. Tiers are based on data retention, error analyses numbers, and deployment options.

Pros:

- Detects all errors - caught and uncaught exceptions, HTTP errors, and log errors.
- Shows you the code and variable state of errors right when they occurred.
- Works at the JVM level, with no reliance on log files or logging frameworks.
- Production performance - self-throttles to maintain under 3% CPU and IO overhead.
- Automatically detects code deployments and alerts you when they introduce new errors.
- Integrates with JIRA, New Relic, and log management tools.

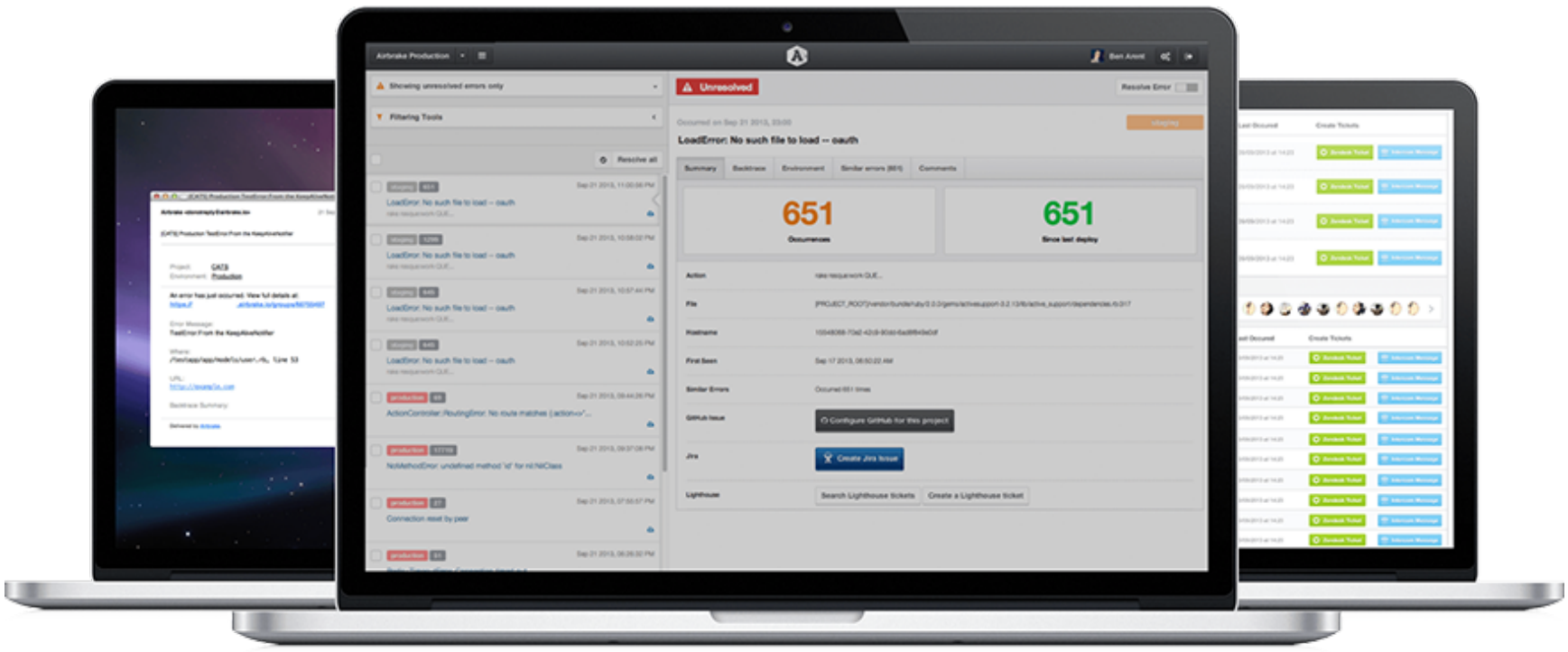
Cons:

- Non-JVM-based languages are not supported.
- Value is limited for non-production environments, such as staging.
- Does not currently have APIs to integrate with other alerting tools.

View video

Airbrake:

Rackspace's Airbrake has taken on the mission of "No More Searching Log Files." It provides users with an interface that includes a dashboard with error details and an application specific view. Supported languages include Ruby, PHP, Java, .NET, Python, Swift, JavaScript, and recently, iOS and Android.



When to use it: If you have a mixed environment that includes Ruby, as Airbrake is strong with Ruby. If you like Rackspace as a company. If you want more detail in your stack traces, such as error grouping and trends, from your error tracker. If you want to track errors in mobile apps.

Price: Three tiers: $39/mo, $89/mo, $199/mo. Tiers are based on the number of users and projects you want.

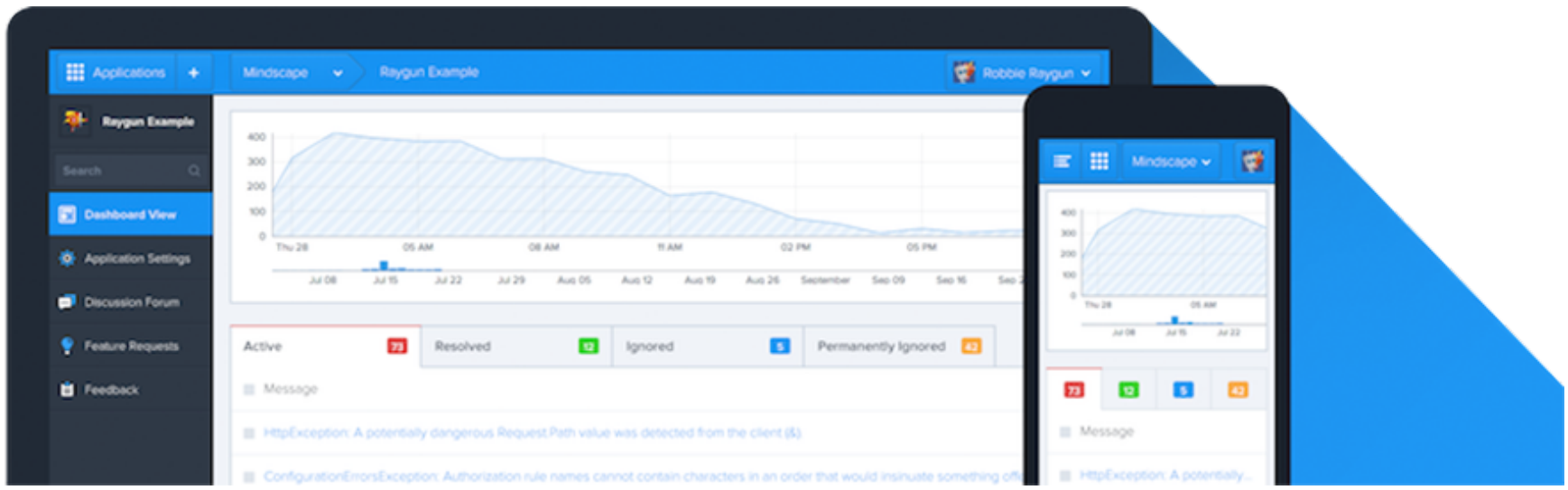
Pros:

- Detailed stack traces, grouping by error type, users and environment variables
- Team productivity – Filter important errors from the noise
- Team collaboration – See who's causing bugs and who's fixing them
- External Integrations – HipChat, GitHub, JIRA, Pivotal and over 30 more
- Mobile tracking abilities with iOS and Android support.

Cons:

- More difficult setup relative to other tools in this space
- Less friendly of an interface than other tools
- On the Java front, support is partial at best currently. Airbrake only supports Log4j. There's no Log4j2 or Logback support. If you don't use Log4j, you'll have to set your app up to send data directly to Airbrake.
- Installation for Java requires build changes and only works for Maven as far as build tools go. Going manual is the only other option.
- Security can be a concern, as filtering for personally identifiable information can be impossible, requiring manual effort on your part.

Mindscape's Raygun is an error management system that keeps track of exceptions coming from your apps, particularly on the mobile side. It supports various desktop, mobile, and web programming languages, including Java, Scala, .NET, Python, PHP, iOS, Android, and JavaScript. Besides that, sending errors to Raygun is possible through a REST API and a few more Providers (that's what they call language and framework integrations) that came to life thanks to developer community involvement.



When to use it: If you're looking for full stack traces from your error tracking tool and mobile is part of your environment. For Java users, if you feel the downside of dealing with the manual installation is worth what else you get from Raygun.

Price: Three tiers: $49/mo, $149/mo, $499/mo. Tiers are based on the number of applications and events tracked.

Pros:

- Error grouping – Every occurrence of a bug is presented within one group with access to single instances of it, including its stack trace.
- Full text search – Error groups and all collected data is searchable.
- View app activity – Every action on an error group is displayed for all your team to see: status updates, comments and more.
- Affected users – Counts of affected users appear by each error.
- External integrations – Github, Bitbucket, Asana, JIRA, HipChat and many more.

- Visualizations and level of detail - provides a deeper level of detail than some other error tracking tools thanks to full stack traces, and delivers good graph visualizations

Cons:

- Incomplete error tracking - On the Java front at least, it only has simple handling for uncaught exceptions, according to its documentation. Caught exceptions, HTTP errors, and log errors can only be tracked individually and manually.
- Manual set up - Everything is manual in terms of setting up Raygun for Java. This puts the onus of implementation (and therefore the quality of error tracking) on the developer.
- Security can be a concern, as filtering for personally identifiable information can be impossible, requiring manual effort on your part.

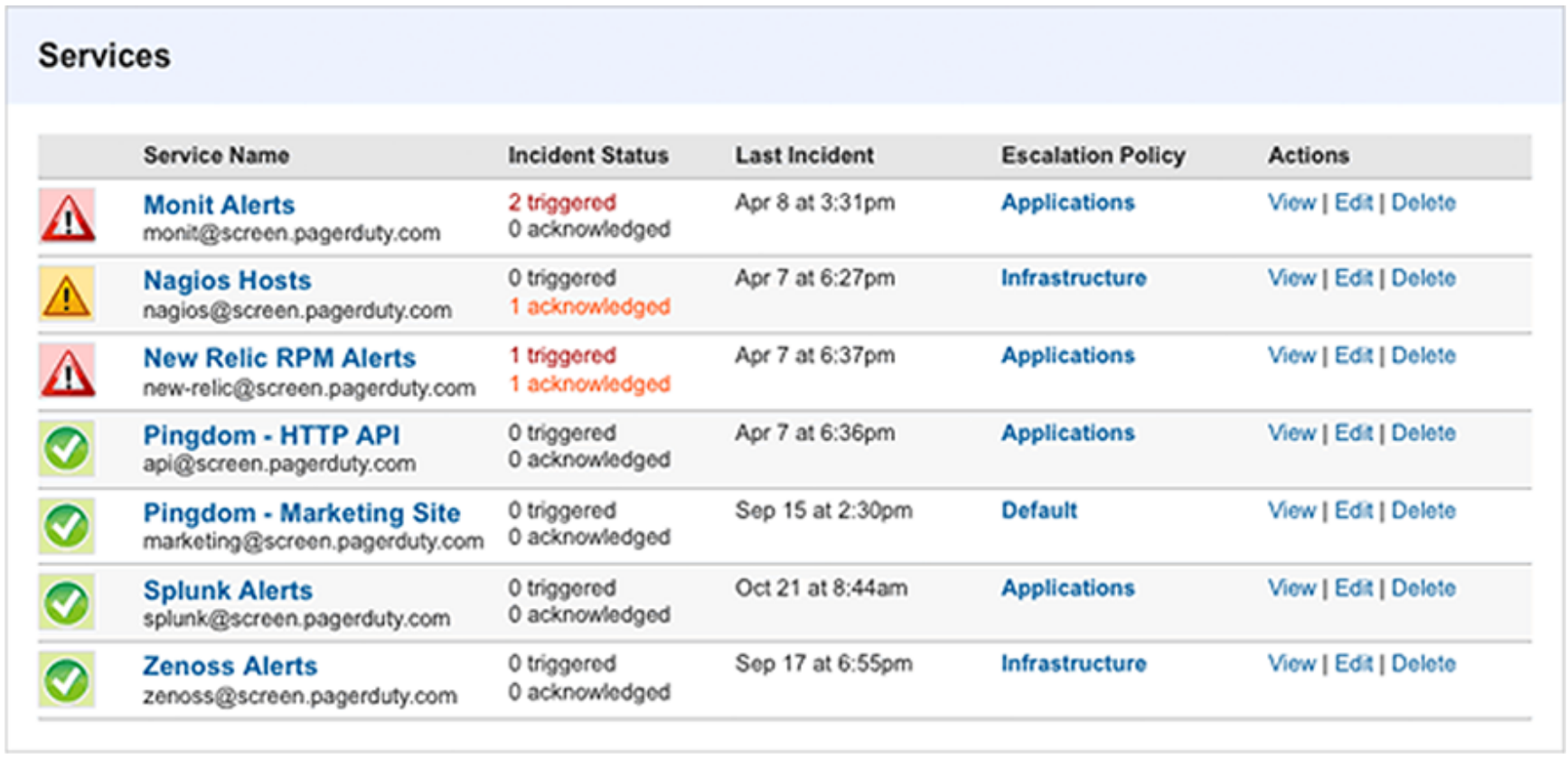
View video

Sentry vs. Airbrake vs. Raygun:

These tools accomplish largely similar things, but they have a few differences. Sentry is open source, while Raygun and Airbrake are not. Raygun is good for mobile environments with an extensive support for mobile languages. Airbrake is good for Ruby. Beyond that, most of the differences come down to installation and supported frameworks. The best approach is to have a clear understanding of your environment and choose the tool that requires the least manual implementation for your specific needs. Comparing the tools across factors that are important to you can provide some clarity as well. Some examples are: data retention, the amount of events captured, performance overhead, languages covered (outside of Java), scalability, and the types of deployments supported (e.g. Hadoop clusters vs web apps).

Ops / Uptime

PagerDuty is an alerting tool that pings you when issues arise in the different monitoring tools that you're having it watch. It doesn't monitor anything on it's own, but takes alerts generated by other tools and sends them out to you and your team based on escalation and priority rules you set up. It can send out alerts through email, phones, and several other means of contact. You can use it to collect alerts from your monitoring tools and create schedules to coordinate your team.



Services

	Service Name	Incident Status	Last Incident	Escalation Policy	Actions
⚠	**Monit Alerts** monit@screen.pagerduty.com	2 triggered 0 acknowledged	Apr 8 at 3:31pm	Applications	View | Edit | Delete
⚠	**Nagios Hosts** nagios@screen.pagerduty.com	0 triggered 1 acknowledged	Apr 7 at 6:27pm	Infrastructure	View | Edit | Delete
⚠	**New Relic RPM Alerts** new-relic@screen.pagerduty.com	1 triggered 1 acknowledged	Apr 7 at 6:37pm	Applications	View | Edit | Delete
✓	**Pingdom - HTTP API** api@screen.pagerduty.com	0 triggered 0 acknowledged	Apr 7 at 6:36pm	Applications	View | Edit | Delete
✓	**Pingdom - Marketing Site** marketing@screen.pagerduty.com	0 triggered 0 acknowledged	Sep 15 at 2:30pm	Default	View | Edit | Delete
✓	**Splunk Alerts** splunk@screen.pagerduty.com	0 triggered 0 acknowledged	Oct 21 at 8:44am	Applications	View | Edit | Delete
✓	**Zenoss Alerts** zenoss@screen.pagerduty.com	0 triggered 0 acknowledged	Sep 17 at 6:55pm	Infrastructure	View | Edit | Delete

When to use it: PagerDuty is the main player in this field with only a few competitors (such as VictorOps), so if you're using several tools and want one unified alerting tool, it's the one to check out. Also give it a try if an inside-your-environment alerting tool is what you're looking for in general.

Price: $19-$49 per month per user, depending on feature needs

Pros:

PagerDuty integrates with a huge variety of different tools to gather their alerts and notifications. It provides a fairly extensive and nuanced means of getting alerts out to the right people at the right level of urgency. It has means for escalating alerts as well.

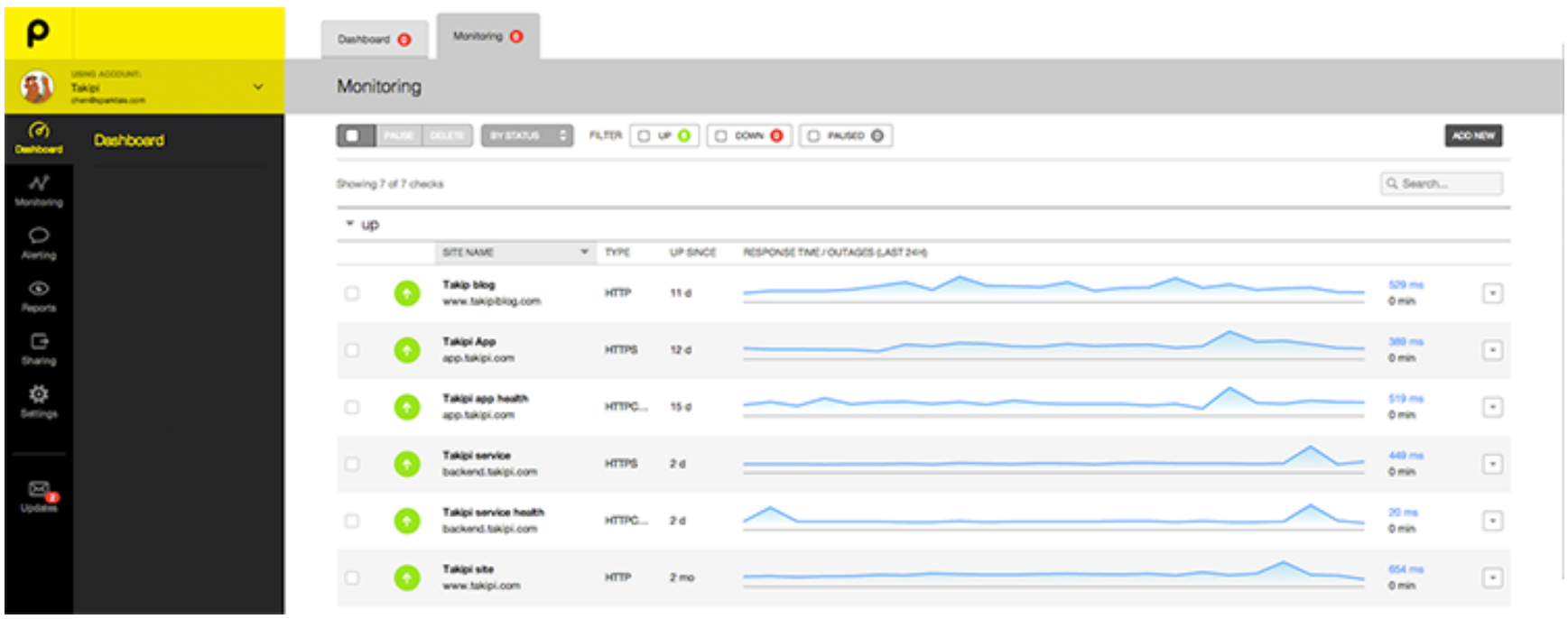
Cons:

It does require some installation and set up. To get the most out of it, you have to spend a bit of time hooking it up with your different tools and entering in the rules for the different alerts you want it to give.

View video

pingdom

Pingdom is a service that provides tracking and alerting on website's response times, 24/7. It helps answer a crucial question that may seem trivial at first blush: Is your website available? By probing it from different locations all over the globe, it can help differentiate actual downtime from routing and access problems.



When to use it: If you're looking for an outside your environment alerting tool. If you don't want to have any setup or installation.

Price: $13-$495 per month, depending on the scale you need

Pros:

Simulates external users and tests your app from the outside. Doesn't require any installation or changes to your environment whatsoever. The onboarding and setup process is super easy and the price is certainly right.

Cons:

Only usable for external-facing apps and environments. Doesn't provide a sophisticated level of alerting options.

View video

PagerDuty vs. Pingdom:

These aren't really true competitors. They are more complementary tools. In fact, PagerDuty has an integration with Pingdom to apply its more nuanced alerting capabilities to the alerts Pingdom generates. PagerDuty is an internal alerting tool, while Pingdom is an external alerting tool.

Resources:

5 Error Tracking Tools Java Developers Should Know

Other than shrinking release cycles, another property of the modern development lifecycle is ever expanding log files that can reach GBs per day. Let's say some issue arises after a new deployment: If you'd like to produce a timely response, dealing with GBs of unstructured data from multiple sources and machines is close to impossible without the proper tooling. In this space we can essentially divide the tools to the heavy duty enterprise on-premise Splunk, its SaaS competitors like Sumo Logic, Loggly and PaperTrails, and open source options like Logstash and Graylog.

Splunk

Sumo Logic

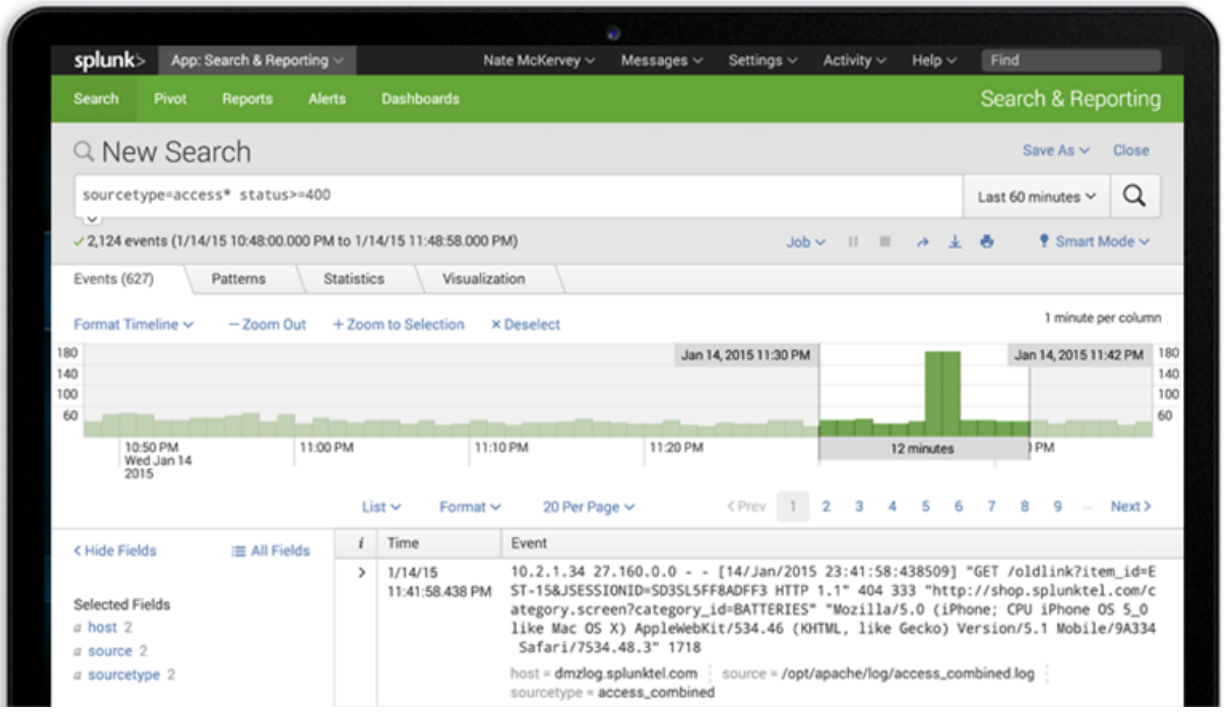
Loggly

PaperTrail

Logstash

GrayLog

Splunk is the biggest tool in the log management space. It's established, full-featured, and enterprise-class. It's unique in this space as it's an on-premises tool (although they have come out with a SaaS version as well).



When to use it: Enterprise companies with lots of feature needs and a variety of data that needs analyzing.

Price: $1,800/year to $60,000/year, depending on the data volume you need

Pros:

Splunk is probably the most feature-rich solution in the space. It's got hundreds of apps (I counted 537) to make sense of almost every format of log data, from security to business analytics to infrastructure monitoring. Splunk's search and charting tools are feature-rich to the point that there's probably no set of data you can't get to through its UI or APIs.

Cons:

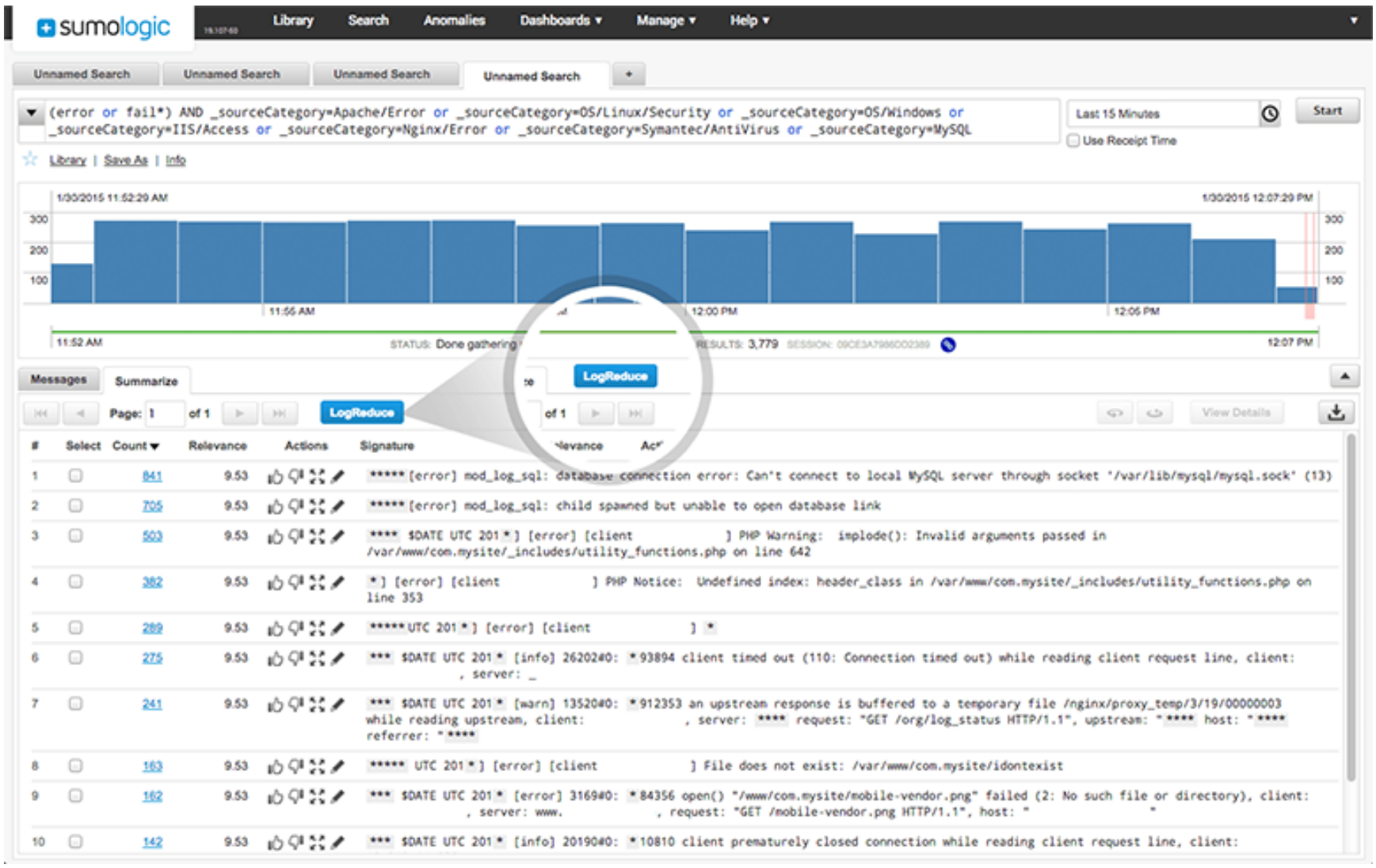
Splunk has two major cons. The first, which is more subjective, is that it's an on-premises solution, which means that setup costs in terms of money and complexity are high. To deploy in a high-scale environment you will need to install and configure a dedicated

cluster. As a developer, it’s usually something you can’t or don’t want to do as your first choice.

Splunk’s second con is that it’s expensive. To support a real-world application you’re looking at tens of thousands of dollars, which most likely means you’ll need sign offs from higher-ups in your organization, and the process is going to be slow. If you’ve got a new app and you want something fast that you can quickly spin up and ramp as things progress, then Splunk is not the right tool for you.

View video

Sumo was founded as a SaaS version of Splunk, going so far as to imitate some of Splunk's features and visuals early on. Since then, Sumo Logic has developed into a full fledged enterprise-class log management solution in its own right. Sumo Logic is the most enterprise-focused of the SaaS log analyzers.



When to use it: If you're an enterprise-type company but are willing to sacrifice some features for the benefits of SaaS, Sumo Logic is worth exploring. It's also good if you have a strong focus on security. It's not just developer-oriented as a tool either, with benefits for security teams and business purposes.

Price: Sliding scale from free to $1800/mo, depending on data volume and user needs.

Pros:

Sumo Logic is chock-full of features to reduce, search, and chart mass amounts of data. Out of all the SaaS log analyzers, it's probably the most feature-rich. Also, being a SaaS offering, it inherently means setup and ongoing operation are easier than an on-premises or open source tool. One of Sumo Logic's main points of attraction is the ability to establish baselines and to actively notify you when key metrics change after an event, such as a new version rollout or a breach attempt.

Cons:

This one is shared across all SaaS log analyzers, which is that you need to get the data to the service to actually do something with it. This means that you'll be looking at possible GBs (or more) uploaded from your servers. This can create issues on multiple fronts:

As a developer, if you're logging sensitive info or PII, you need to make sure it's redacted.

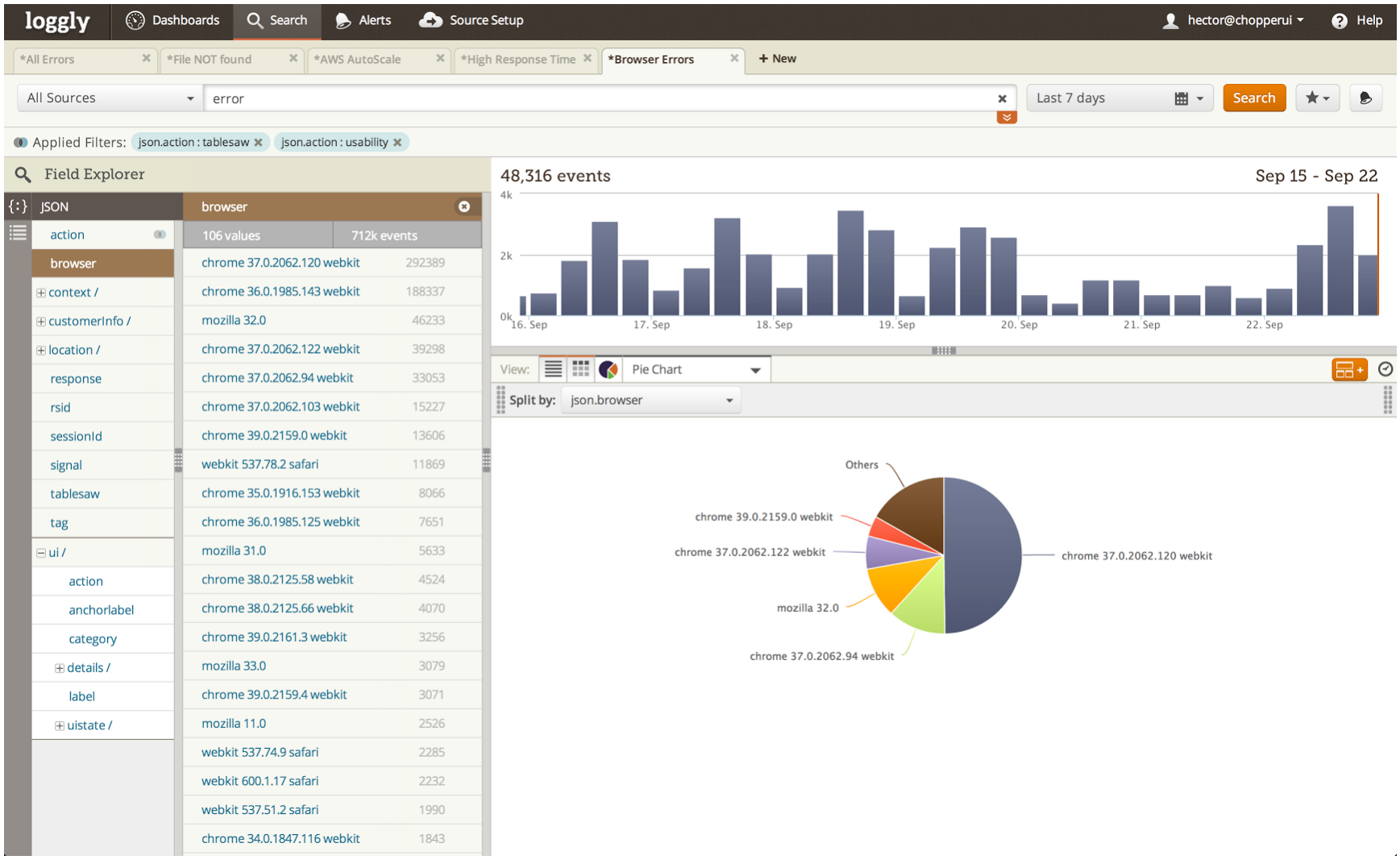
There may be a lag between the time data is logged and the time it's visible to the service.

There's additional overhead on your machines transmitting GBs of data, which really depends on your logging throughput.

View video

loggly

Loggly is a robust log analyzer, focusing on simplicity and ease of use for a DevOps audience. It's targeted for developers and DevOps - making it less enterprise-focused.



When to use it: Primary use cases are for troubleshooting and customer support scenarios. It's a good tool for a DevOps team.

Price: Sliding scale ranging from $49/mo to $5,100/mo, depending on data volume needs. Loggly has a free lite version as well.

Pros:

Loggly is geared more towards helping DevOps find and fix operational problems. This makes it very developer-friendly. Things like creating custom performance and DevOps

dashboards are super easy to do. Pricing is also transparent, which makes the start of use easier.

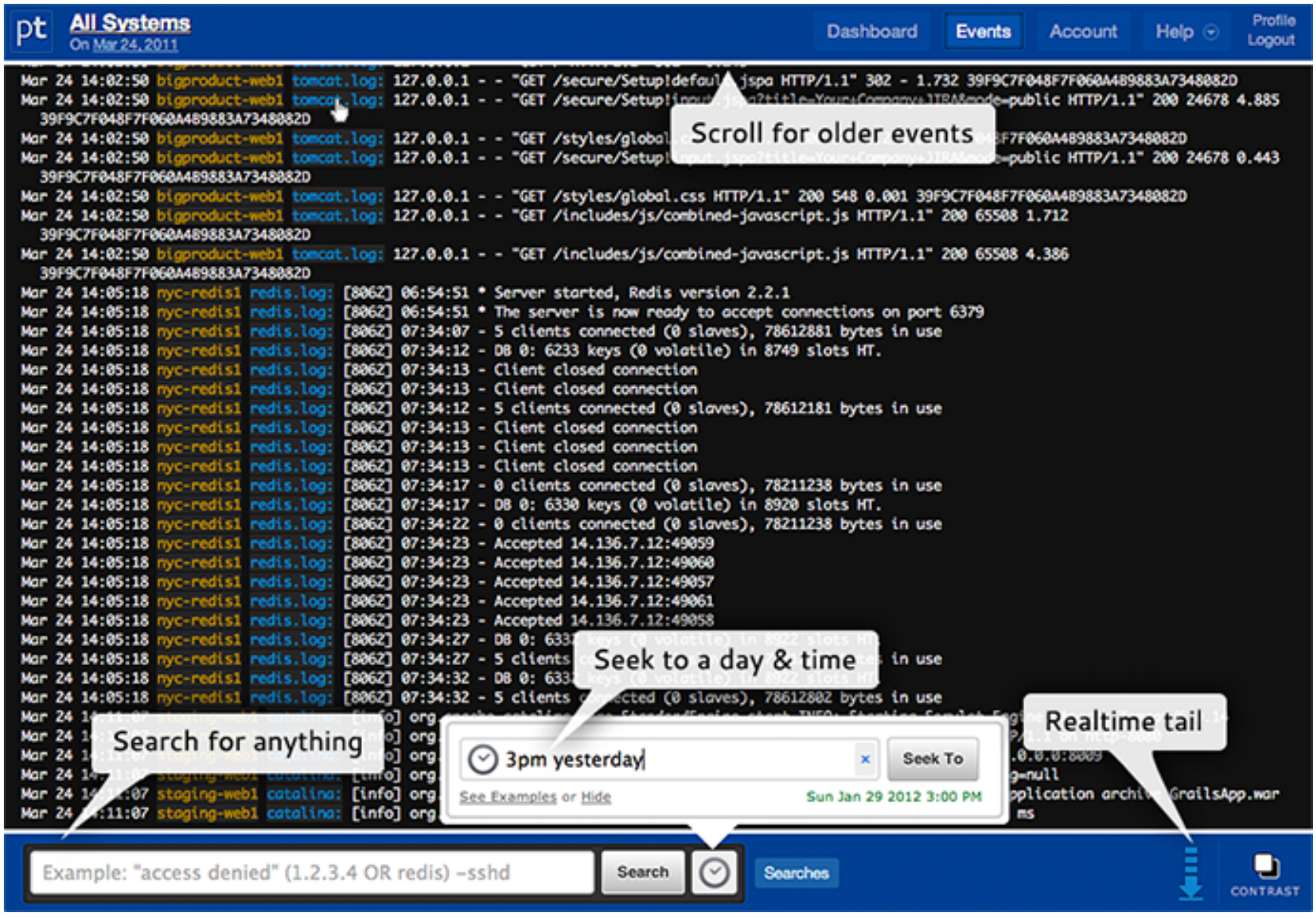
Cons:

Don’t expect Loggly to scale into a full blown infrastructure, security, or analytics solution. If you need forensics or infrastructure monitoring, then you should consider other tools. This is a tool mainly for DevOps teams to parse data coming from your app servers. Anything beyond that you’ll have to build yourself.

View video

papertrail

PaperTrail is a simple way to look and search through logs from multiple machines, in one consolidated easy-to-use interface. It's a SaaS tool designed to enhance the logs you already collect or generate.



When to use it: If you want a simple and straightforward tool without lots of extra bells and whistles. If you want a stripped down and basic log analyzer that is good for looking at log files in aggregation and doesn't try to be anything more.

Price: Sliding scale from free to $3,920/mo, based on data volume and the searchable duration you need.

Pros:

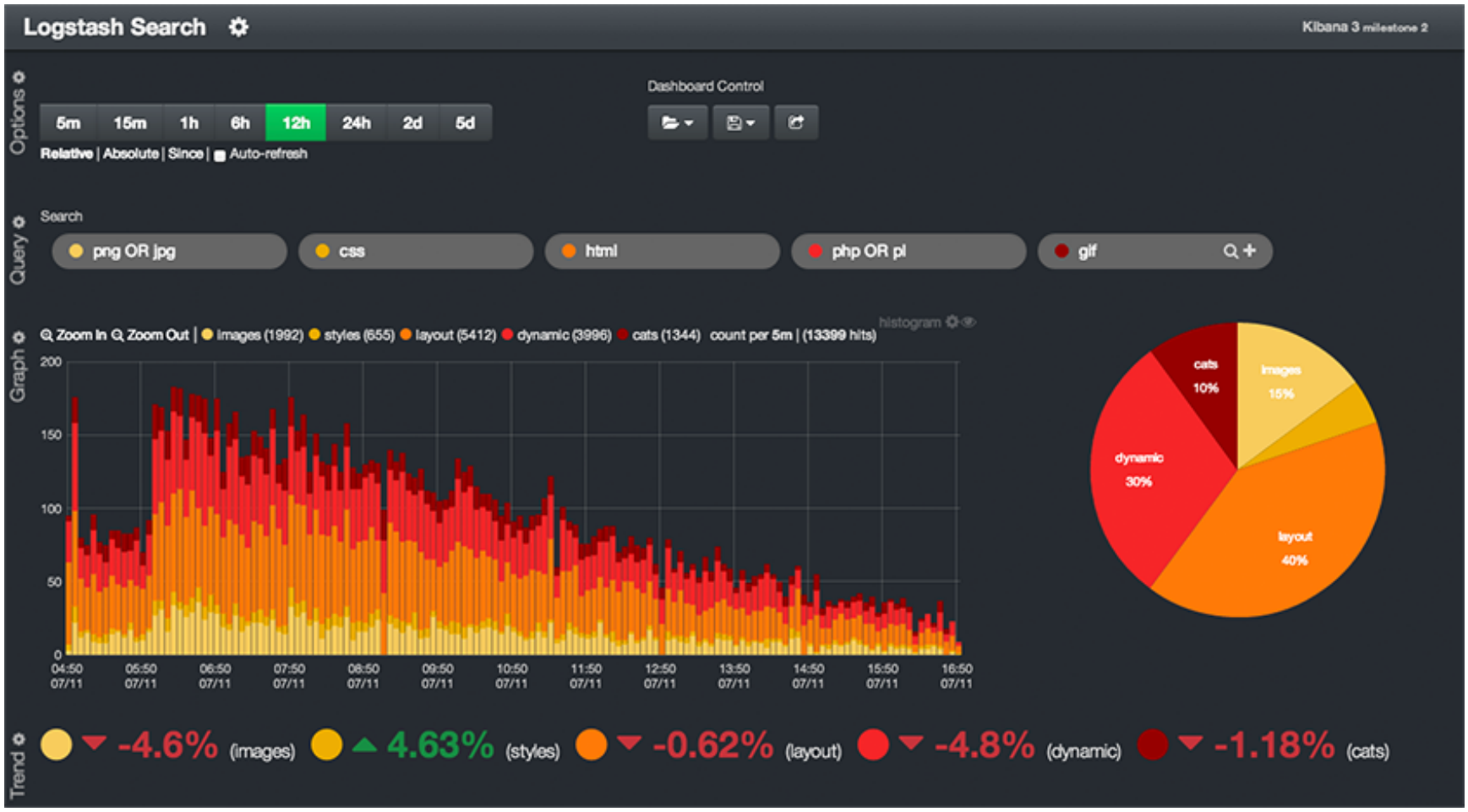
PaperTrail is a simple way to look at log files from multiple machines in a singular view in the cloud. The UX itself is very similar to looking at a log on your machine, and so are the search commands. It aims to do something simple and useful, and does it elegantly. It's also very affordable.

Cons:

PaperTrail is mostly text-based. If you're looking for any advanced integrations, predictive, or reporting capabilities, then you this isn't the tool for you.

logstash

Logstash is an open source tool for collecting and managing log files. It's part of the open-source ELK stack which includes ElasticSearch for indexing and searching through data and Kibana for charting and visualizing data. Together they form a powerful log management solution.



When to use it: If you want an open source tool. If you're interested in implementing the entire ELK stack or at least see value separately in using ElasticSearch or Kibana and want the interactive benefits that come from combining these tools.

Price: Free

Pros:

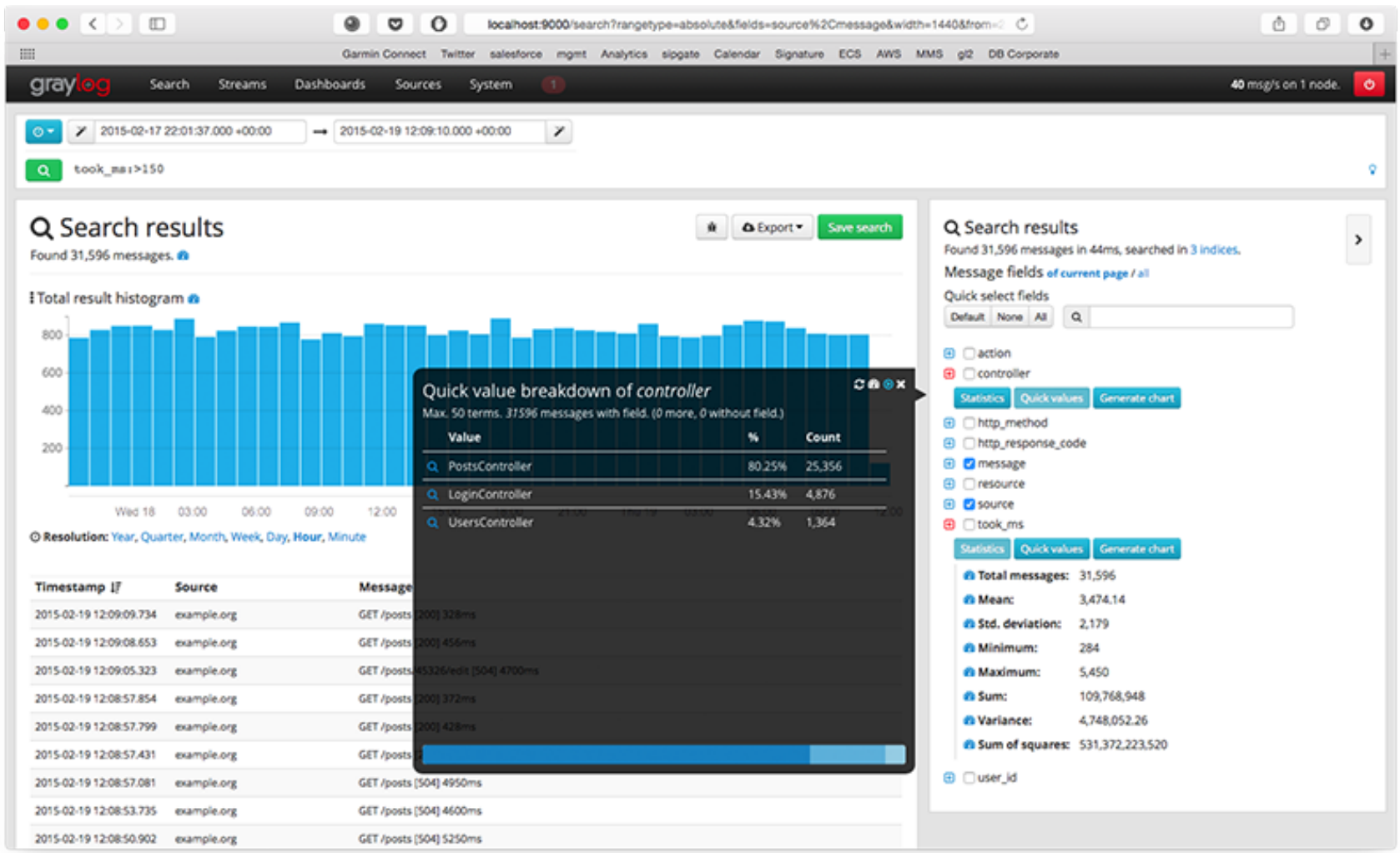
Since Logstash is an open-source solution, you're inherently getting a lot of a control and a very good price. Logstash uses three mature and powerful components, all heavily maintained, to create a very robust and extensible package. For an open-source solution, it's also very easy to install and start using.

Cons:

As Logstash is generally deployed as part of the ELK stack, it means you're dealing with three different products. That means that extensibility also becomes complex. Logstash filters are written in Ruby, Kibana is pure Javascript, and ElasticSearch has its own REST API as well as JSON templates. When you move to production, you'll also need to separate the three into different machines, which adds to the complexity.

View video

A fairly new player in the space, GrayLog is an open-source log analyzer backed by MongoDB as well as ElasticSearch (similar to Logstash) for storing and searching through log errors. It's mainly focused on helping developers detect and fix errors in their apps, but they recently released an official Graylog v1.0 GA designed to be an enterprise-ready platform.



When to use it: Graylog is more targeted towards developers than other open source log management tools. Its alerting features are strong, so if that is important to you in your log management tool, then consider Graylog. If you want a log management tool that aims to be both enterprise-ready and is open source, take a look at the new Graylog 1.0 release.

Price: Free

Pros:

- Can handle an extensive range of data formats
- Simple interface
- Fine grained control for authentication and user permissions
- Alerting for streams you designate makes for quick analytics
- Uses a REST API for distributing and being sent data

Cons:

- Not particularly management friendly on the dashboard front
- Reporting functionality is a bit lacking
- As a newer tool, particularly on the enterprise front, it’s not as mature and fully vetted as other options

View video

Splunk vs SumoLogic vs Loggly vs PaperTrail vs Logstash vs GrayLog (phew):

This can be broken down into essentially three categories - Splunk, SaaS, and open source. If you're a large enterprise company, you should consider Splunk first since it's the established enterprise player. Otherwise, SaaS vs open source comes down to your preferences and why you're looking at a log management tool. Sumo Logic is well suited for non-developer needs in enterprise-like companies. Loggly is great for troubleshooting and customer support. PaperTrail is best for a no-frills straightforward experience. GrayLog is good for open source audience that wants alerting functionality. And Logstash is good for an open source audience that's exploring the ELK stack.

Resources:

The 7 Log Management Tools Java Developers Should Know

Being able to generate data about your app is crucial, but you also need to be able to understand it. That's where visualization and metrics tools come in. These tools take (or are given) your data and provide you with key metrics and/or digestible visualizations of what's happening.

The first decision (after deciding to use one of these tools at all), is between open source (we cover Graphite and ELK stack) and SaaS (we cover Keen IO, Librato, and DataDog) options. The downside to the SaaS ones is that you have to weave them into your code, which is essentially a big investment. Weaving them into your code creates a huge dependency, so if you get to high scale, you're in trouble. The downside to open source tools is they require you to install and set them up yourself. This requires a lot of machines if you get to high scale that now require monitoring, which you don't have with SaaS. Either way, the cost of changing tools in the visualization/metrics or log analyzer space is huge.

There can be some uncertainty around using these tools instead of, or in addition to, log management tools. There is plenty of overlap between these two areas (the ELK stack covers both for example), but there are differences too. Your decision will come down to your budget and environment, but one thing to be aware of is the implicit cost of log analyzers. Any data center you're hosted on will charge you for the bandwidth you use, and log analyzers tend to use much more bandwidth than metrics tools. One way to decide if you need log analyzers or visualization/metrics tools is journaling. If you need journaling, which is info on specific events or users, you have to use a log analyzer. If you don't need journaling, you can use one of the metrics tools.

ELK stack

Graphite (and Grafana)

Keen IO

Librato

DataDog

ELK stack

The ELK stack is a combination of three open source products from Elastic: Elasticsearch's search and analytics capabilities, Logstash as the logs aggregator, and Kibana for the fancy dashboard visualization. We've been using it at Takipi for a while, feeding it from Java through our logs and Redis, and it's in use both by developers and for BI. Today, Elasticsearch is pretty much built-in with Logstash, and Kibana is an Elastic product as well, making integration and setup very simple.

When a new deployment rolls out, the dashboards follow custom indicators that you can set up about your app's health. These indicators update in real time, allowing close monitoring when freshly delivered code takes its first steps after being uploaded to production.

When to use it: If you want to deploy open source tools. The ELK stack crosses several tool areas (log management, search and analytics, and visualizations), so if you're interested in using tools in all those areas and want them to be tightly integrated. You can implement them separately as well, but much of the benefit is in the combined group.

Price: Free, but there are Elastic paid plans that include different levels of professional support for the tools.

Pros:

- Strong communities
- Kibana has a wide range of uses, including business analytics, that other visualization and metrics tools do not
- The combination of Elasticsearch for metrics and Kibana for visualizations is very smooth and tightly integrated
- No price barrier

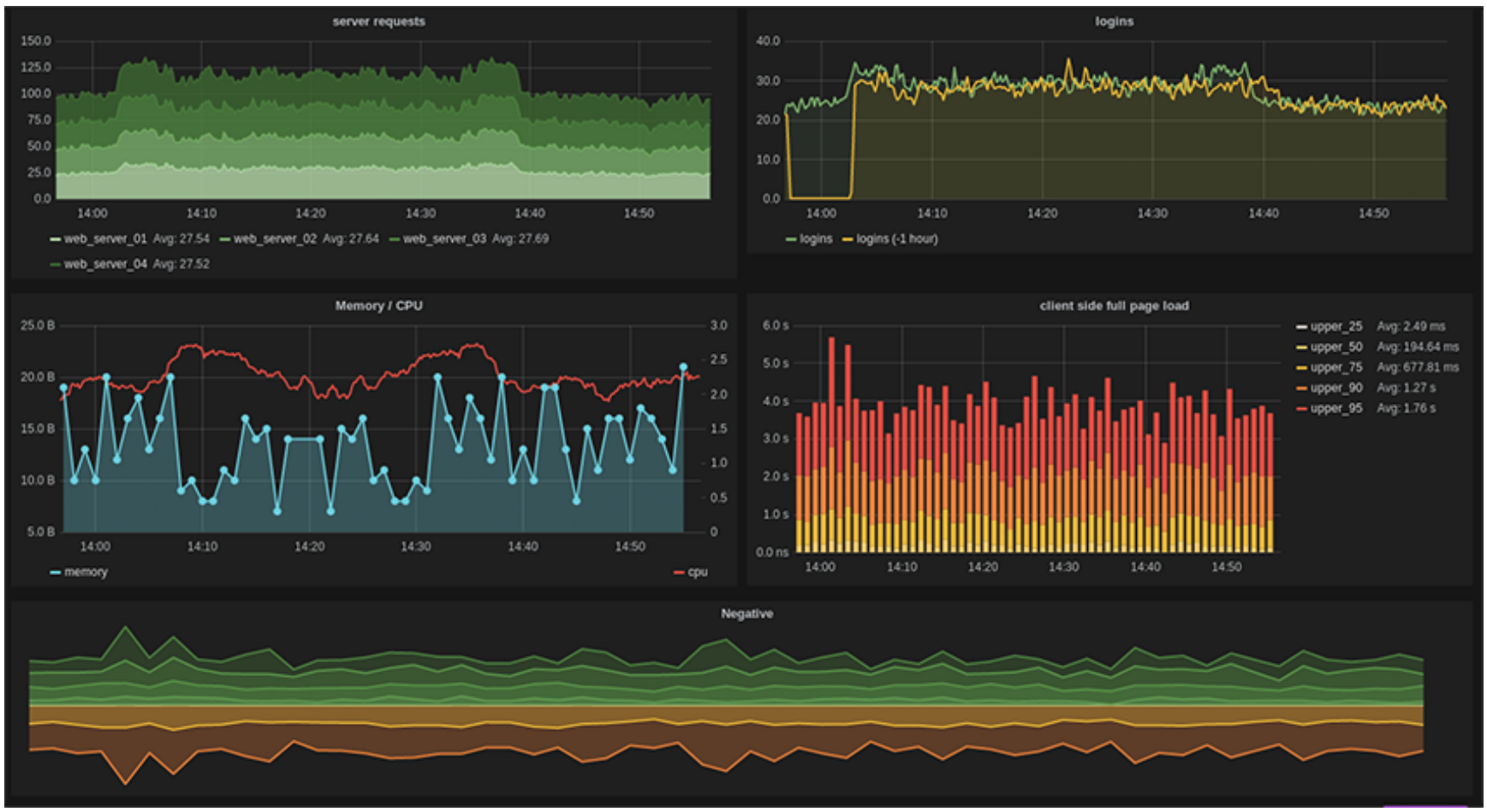
Cons:

- As they are open source tools, you need to install and set them up yourself
- Requires a lot of machines that need monitoring if you use at higher scale

View video

Graphite (and Grafana)

Graphite is an open source tool written in Python for storing data and providing graphs and visualizations of it on demand. It does not collect any data itself, but instead works with other tools for that purpose. Grafana is an open source tool that provides a metrics dashboard and graph editor for Graphite. It essentially enhances what you can get from Graphite and provides more features. Graphite and Grafana are designed to be language agnostic. Also worth noting is Hosted Graphite, which essentially offers Graphite and Grafana as a paid SaaS model.



When to use it: If you want an open source tool that is focused only on visualizations and metrics without diving into more complicated analytics or data collection. If you want a strong DevOps focused tool.

Price: Free

Pros:

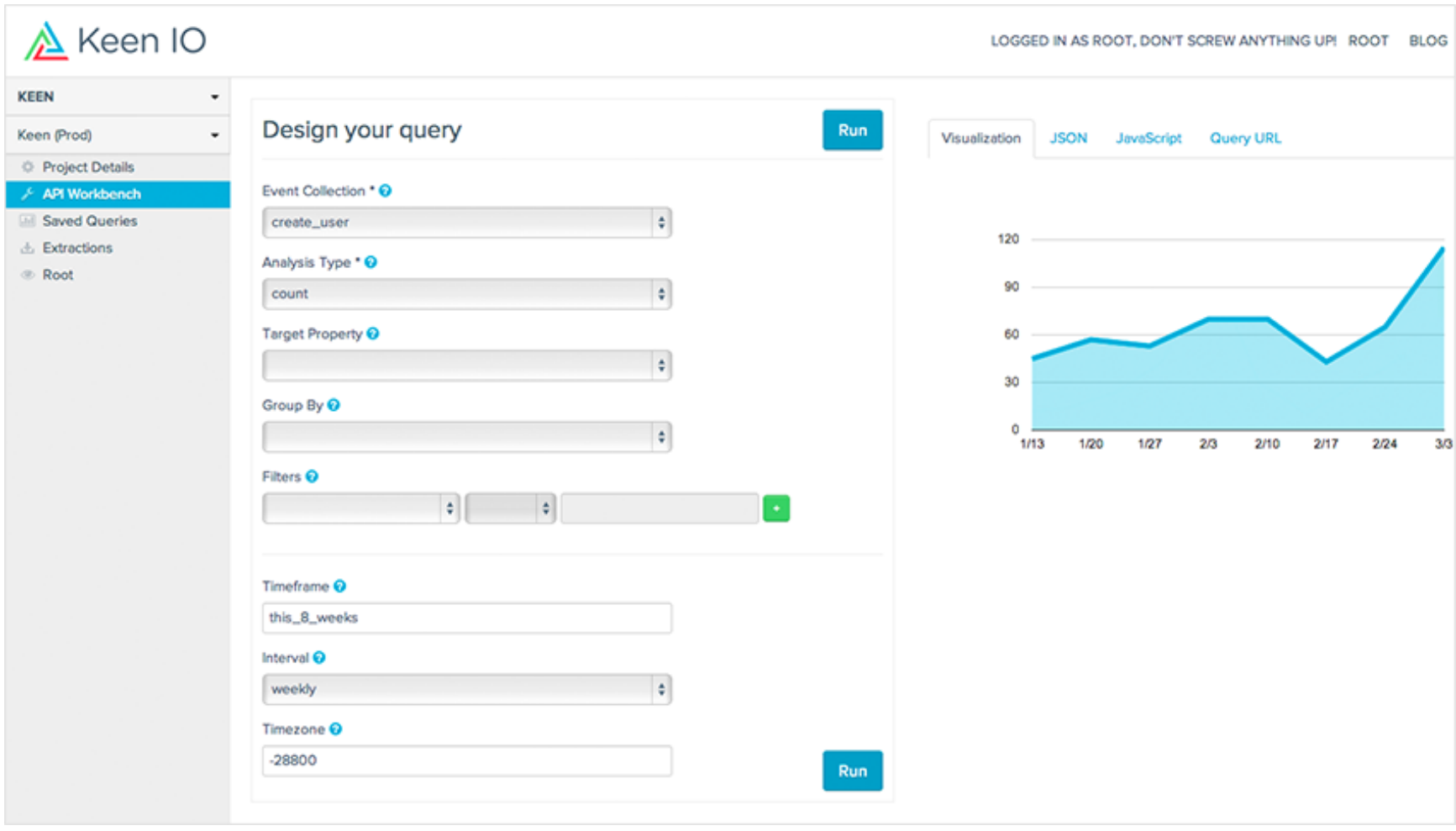
- Reduces reliance on log shipping. Looking at Graphite's graphs can diagnose some issues without diving into logs
- Excellent at creating real-time graphs of numeric and time-differentiated data
- No price barrier
- Grafana provides many additional features and spiffy looking visuals to Graphite

Cons:

- Requires a separate effort (and coding) to send data to it, as it collects nothing on its own
- Some lulls with development at times, but it seems to be active these days
- Open source tool, so the installation, setup, and scaling concerns fall on your shoulders
- Graphite on its own is fairly unappealing from a GUI standpoint

View video

Keen IO is a SaaS analytics infrastructure platform. It collects the data you specify from a wide range of possibilities, and its APIs perform the analytics and visualization tasks that you request. It's designed around providing flexibility and choice to users for whom other visualization and metrics tools aren't satisfying.



When to use it: If you can't get the exact data you want from other tools but can't or don't want to build your own custom analytics infrastructure. If analytics is a primary focus or concern for you, and you want more than an out-of-the-box solution provides.

Price: Sliding scale from free to $2000/month, based on the number of events per month you need.

Pros:

- Requires relatively low overhead compared to similar tools
- Strong visualizations for real-time or archived data
- Gives you the ability to find out nearly anything about your data
- Easily shareable and extractable data

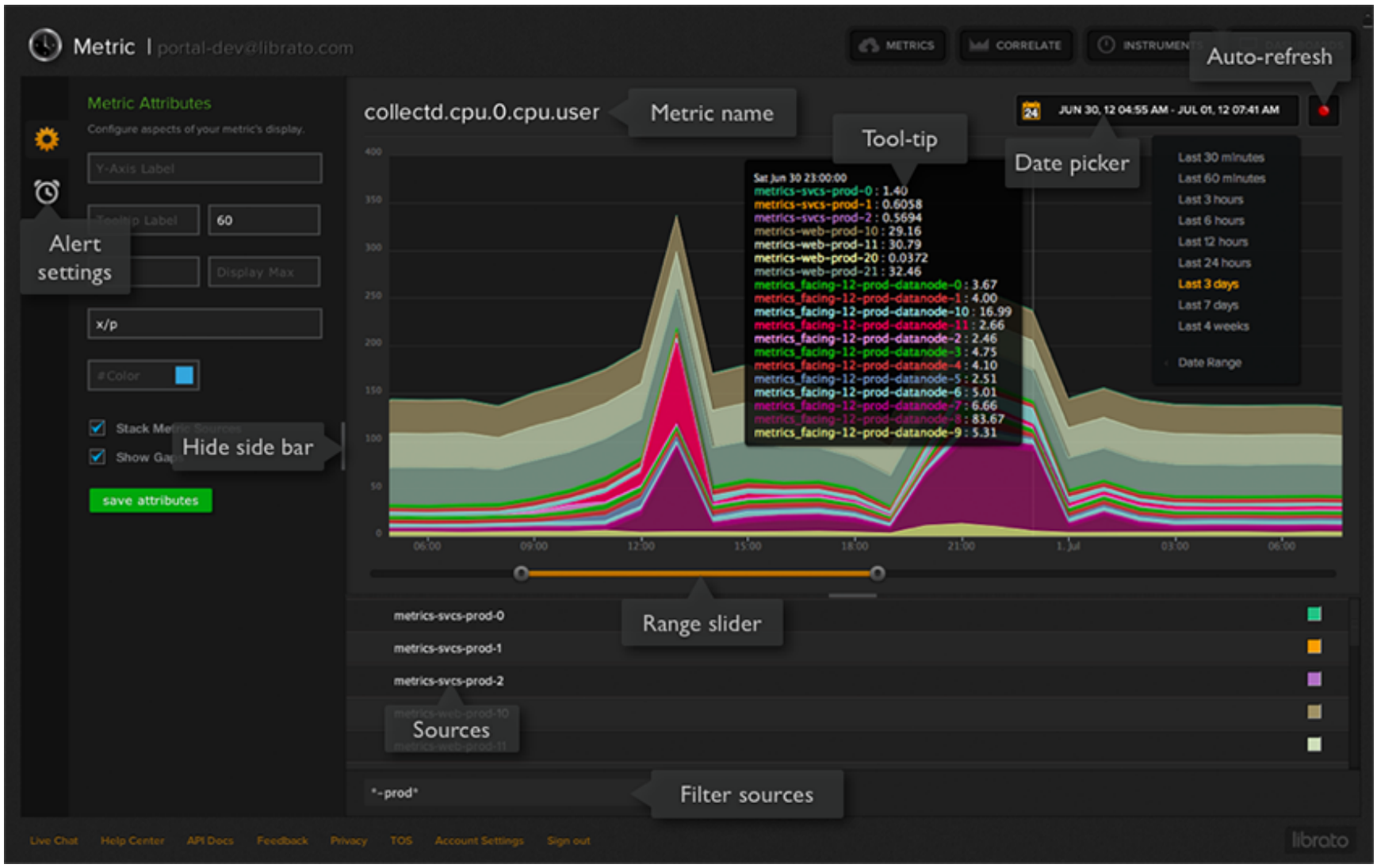
Cons:

- Requires you to weave it into your code, which creates dependencies and scaling challenges
- Using a customizable tool like this means that you'll have to customize it, which makes installation and setup a greater lift

View video

librato™

A hosted service for monitoring and managing cloud applications, Librato can create custom dashboards in seconds without a need to set up or deploy any software. It focuses on graphics and visuals to create a very smooth look and feel compared to other dashboards. It recently became part of SolarWinds.



When to use it: If you want to be able to work with highly customized metrics and reports. If you want beautiful visualizations and good alerting for the data that you send it.

Price: Librato is priced per metric you want on a metered time basis. The pricing is complicated to figure out, but is tailored to what you actually use more so than other pricing models in this space.

Pros:

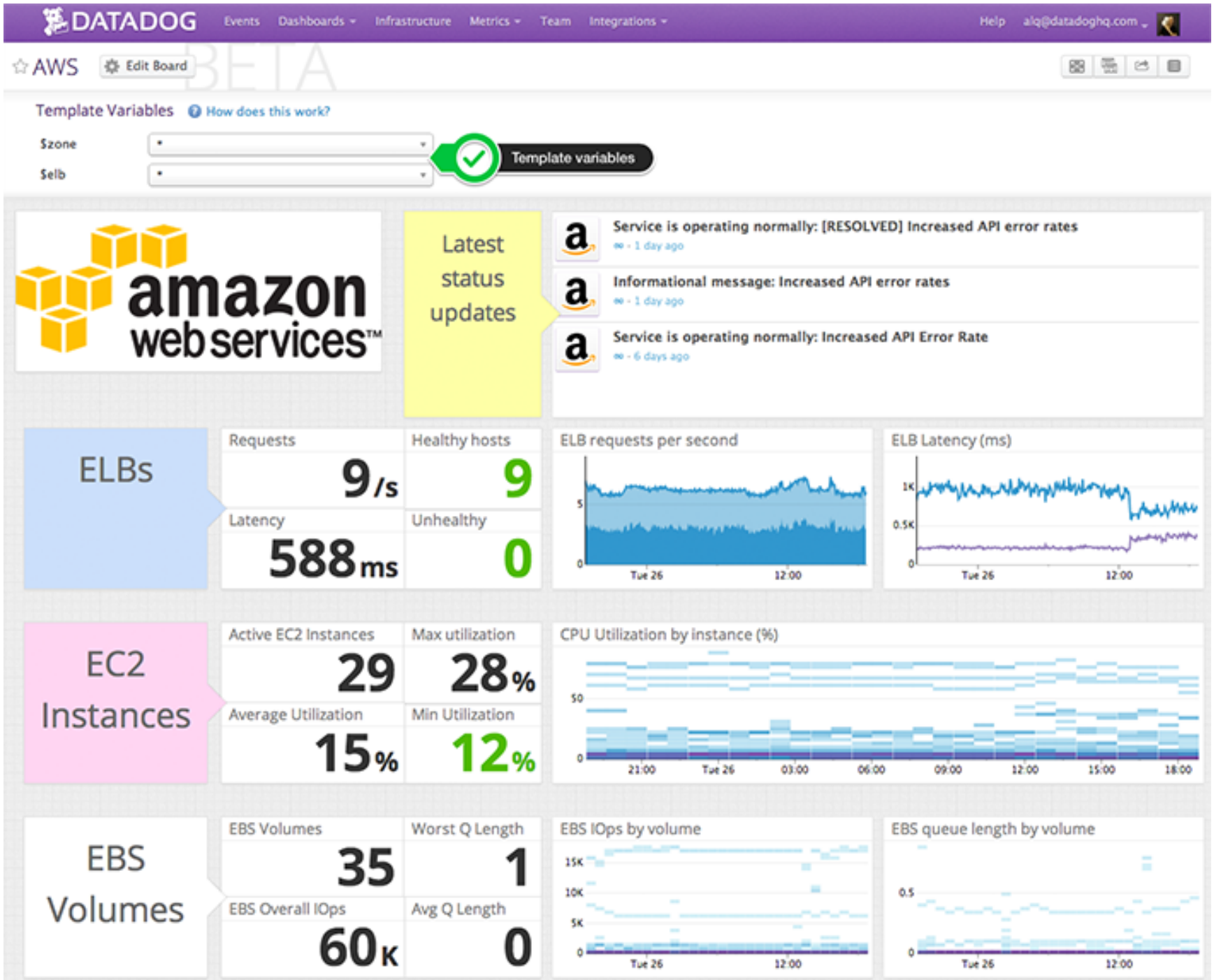
- It would be hard to find anything that Librato doesn't know how to talk with and help make sense of its data.
- Good at collating and visualizing data. Dashboards are fluid and look great at any size.

- Provides custom reports on metrics & alerts through email, HipChat, Campfire, and just HTTP POST requests to integrate with anything you can think of.
- Delivers alerts automatically when metrics cross certain thresholds.
- Strong for analytics on the data you send to it.

Cons:

- Pricing is complicated and obscure
- Requires you to weave it into your code, which creates dependencies and scaling challenges
- Doesn’t collect data on its own. It shows you only what you send to it.

DataDog is SaaS monitoring tool targeted for DevOps teams that takes data from your app and a wide range of other tools and provides metrics and visualizations. It unifies data produced from your infrastructure and software into one location. They are currently built around aggregating and presenting data rather than performing analytics.



When to use it: If you want a DevOps focused tool that works well with an extensive ecosystem of other tools. If you want a tool that presents you a unified view of data generated across your hardware and software.

Price: Free version, and a Standard version with a sliding scale of $15/host/month. Scale is based on number of hosts.

View video

ELK stack vs Graphite vs Keen IO vs Librato vs DataDog:

The first step to choosing which of these tools to use is if you want to go open source (Graphite/ELK stack) or SaaS (Keen/Librato/DataDog). The second step is to determine the importance of analytics to you vs more visualization-only tools. If analytics is a big driver, Keen IO or ELK stack is where you should start your search. If you're more DevOps focused, Graphite and DataDog are targeted for this audience. Librato is a bit broader and has its fingers in both camps. If visualizing what's going on in your app and environment is your driver, Librato, DataDog, Graphite, and ELK stack are worth exploring.

Resources:

7 New Tools Java Developers Should Know

15 Tools Java Developers Should Use After a Major Release

Application Performance Management (APM) has been around for a while, though it hasn't become ubiquitous in production yet. APM provides analytics around your application's performance – at the core this means timing how long it takes to execute different areas in the code and complete transactions. This is done either by instrumenting the code, monitoring logs, or including network/hardware metrics.

On top of this basic concept, many different implementations exist – but there are basic truths most can agree on: A modern solution should monitor production environments, so overhead (in terms of CPU and throughput) becomes very important. Also, it should display what the web/mobile end users are experiencing, which was not part of traditional APMs.

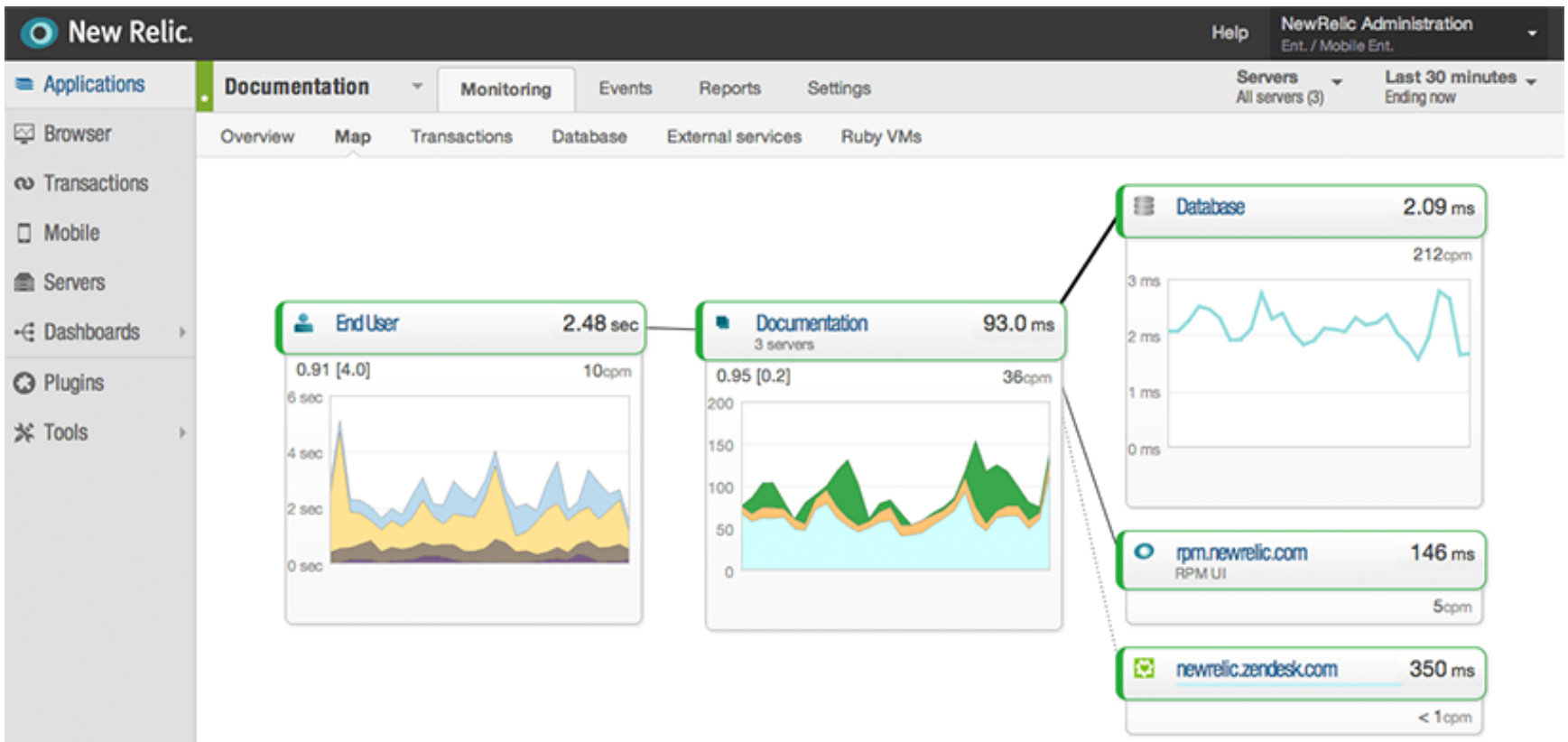
What was once considered a luxury is becoming more commonplace: Rapid new deployments in production mean more chances to introduce errors to your system's architecture, slow it down, and maybe even crash it. New generation APM tools, the ones we discuss in this section, are designed for these environments.

New Relic

AppDynamics

AppNeta

One of the two largest players in the APM space, New Relic is a SaaS tool designed to monitor the performance of your application on the backend, frontend, and mobile elements. It supports Java, Scala, .NET, PHP, Node.js, Ruby, Python, iOS, Android, and JavaScript, as well as a host of plugins, cloud platforms, and databases. They have additional products specifically for mobile apps, server monitoring, and other targeted areas. New Relic uses the Apdex score index to judge performance, which uses a user-defined response time threshold to imply end-user satisfaction.



When to use it: New Relic is generally targeted towards startups and SMBs, although is perfectly usable for enterprises as well. If you use Ruby and Python in your environment, New Relic supports those languages, while some other large APM tools don't.

Price: Three tier pricing, with a free version, a $149/mo/host version, and an Enterprise version.

Pros:

It has a very extensive range of supported environments, so for the majority of situations, New Relic will be usable for you.

Likewise, it has lots of integrations and plugins.

Their focus for performance is on bottom line response times, through the Apdex score.

New Relic has a strong focus on web applications.

Good out of the box tool.

Cons:

It’s only available as a SaaS model, so if you have on-premises requirements, you’ll need to look elsewhere.

New Relic is less focused on the global picture of your app than other APM tools.

It can have performance overhead issues at times.

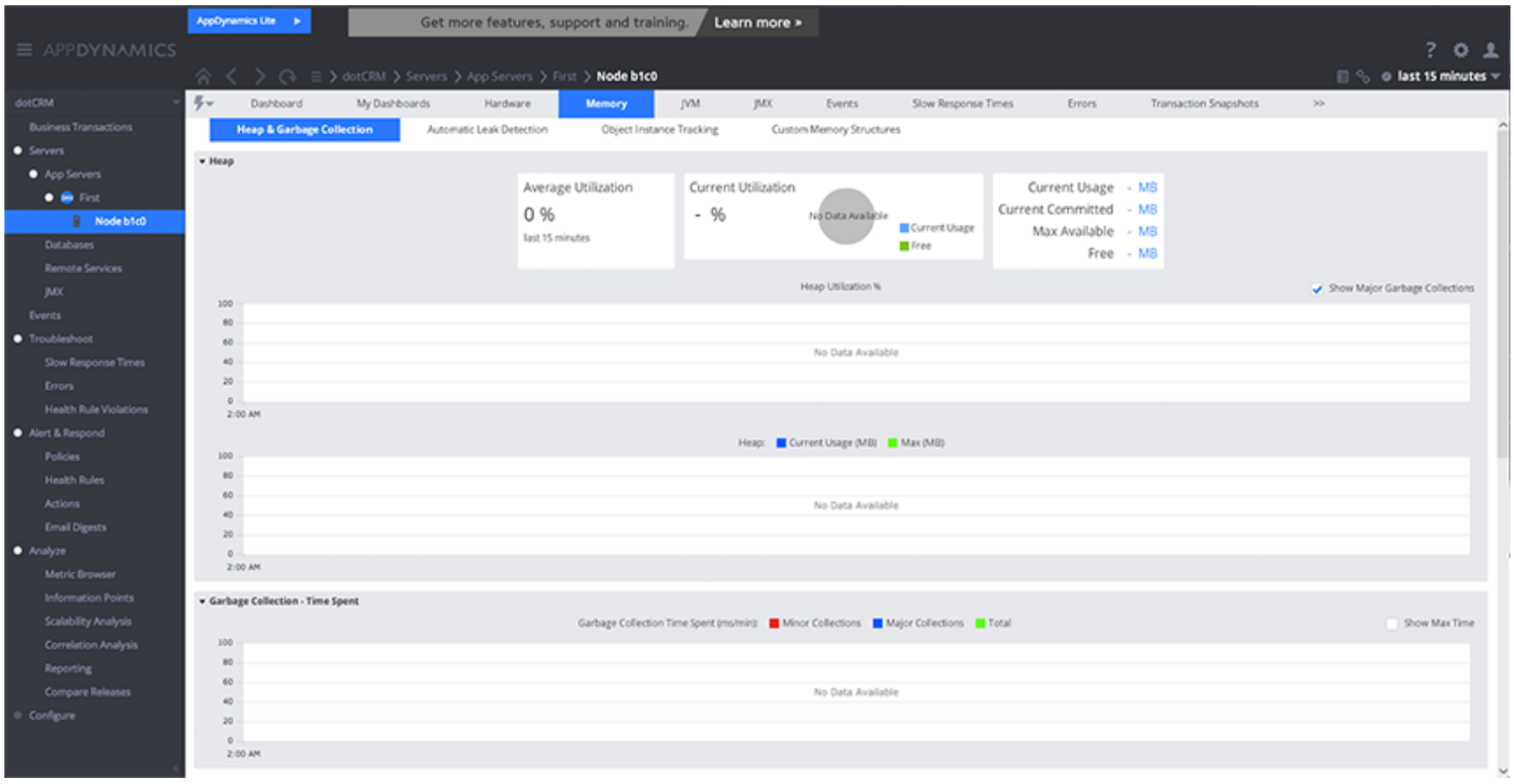
Part of using Apdex means that you need to go in and manually set the threshold measurements that New Relic relies on.

It’s an expensive tool. But there is flexibility in the price if you contact them.

View video

APPDYNAMICS

AppDynamics is the other primary player in the APM space. It comes in SaaS and on-premises versions, and has traditionally positioned itself as more of an Enterprise tool. It supports Java, Scala, .NET, PHP, Node.js, iOS, Android, and JavaScript, as well as a range of plugins, databases, and cloud platforms. AppDynamics determines your app performance through a dynamic scoring system that learns from your system.



When to use it: AppDynamics is generally targeted towards enterprise companies, but is fine for smaller companies as well. If you need or want an on-premises solution, then AppDynamics is easily the best choice. If you need more depth of features in Java or .NET, but don't need as wide a breadth of platforms or mobile support as other tools can provide, AppDynamics is worth exploring.

Price: Four tier pricing system: free version, then three Pro versions with pricing based on number of units. The pricing is a bit complicated to calculate, but comes roughly to $3000/unit/year. Pricing is designed for customized Enterprise plans, so there's a good deal of flexibility there.

Pros:

- Lots of integrations, and an open platform for plugin development.
- Focused on the global picture of your app and visualizing the stack from end-to-end.

- For database monitoring, the AppDynamics dashboard provides a richer look into things like resource consumption, wait states, user sessions, specific query calls and more.
- AppDynamics provides deeper insights into garbage collection and memory leak detection and distributed systems beyond the standard metrics of other tools
- AppDynamics performance measurement doesn't require you to manually input anything.

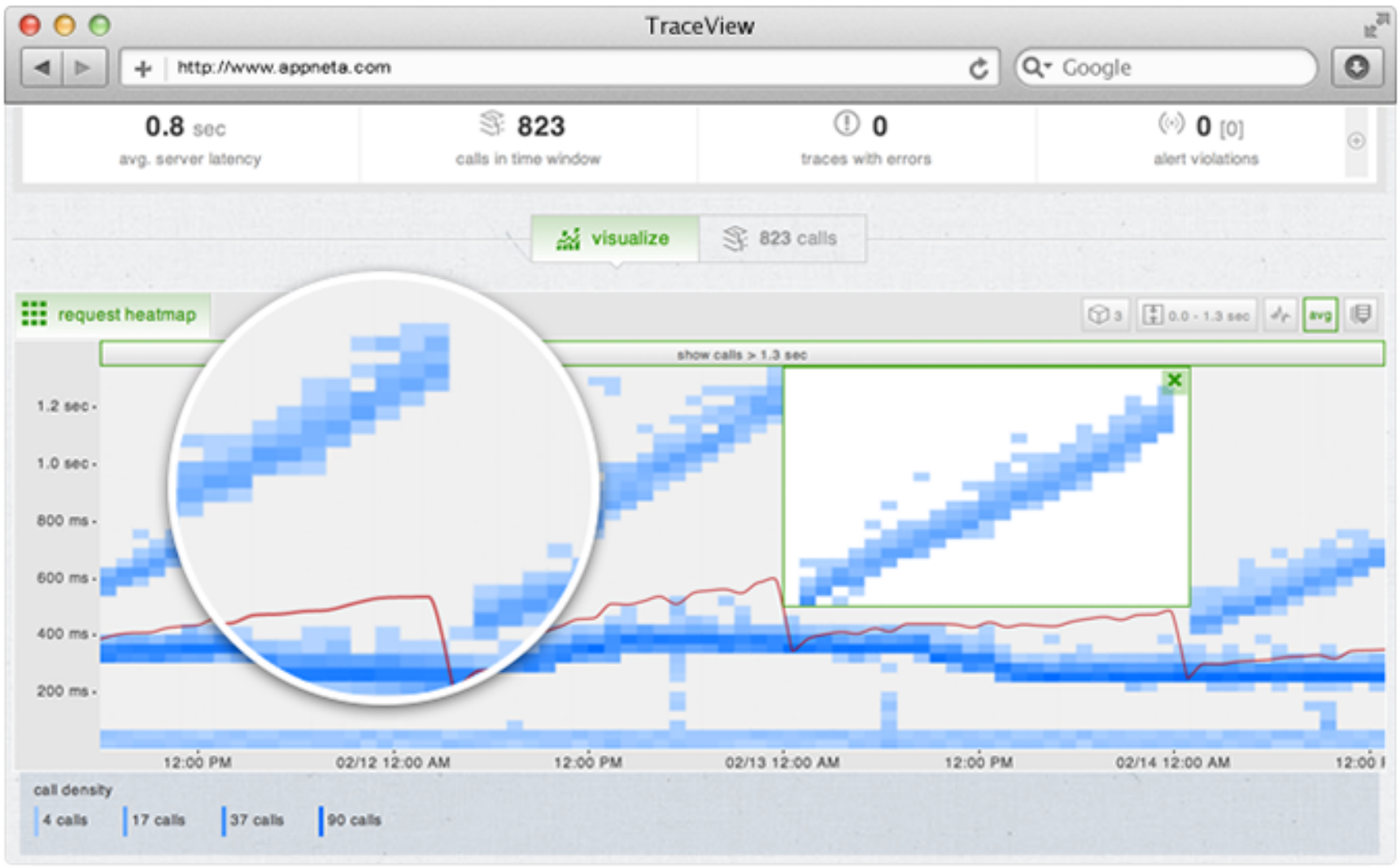
Cons:

- Dashboard is based on Flash, which provides some limitations.
- AppDynamics' method for calculating performance takes some time to learn your system.
- Response times can suffer and performance overhead issues can arise at times.
- More complicated of a tool from the usability standpoint than others in this space.
- It's an expensive tool. But there's flexibility in the price if you contact them.

View video

AppNeta

AppNeta is an APM tool for monitoring web apps, the network, and 3rd party SaaS apps. It focuses on full stack application tracing and is split into four product areas. It supports Java, .NET, Python, Ruby, PHP, and Node.js.



When to use it: If you have a service-oriented architecture with multiple languages. If you need longer data retention and are interested in outlier elements as opposed to averages. If pricing is a large concern for you.

Price: Three tier pricing, with a free version, startup version and enterprise version. Pricing differs between tools, but generally follows a monthly subscription per host model. TraceView, the general APM tool, is $79/host/month (startup) or $119/host/month (enterprise).

Pros:

- Pricing is cheaper than the other big tools in this space, with flexibility for dynamic environments where a per host pricing model might not make sense.
- The combined product suite can give you monitoring on all levels of your app with tools designed to work well together.

Cons:

- Doesn’t play in the mobile APM space today, with no support for iOS or Android.
- Not as feature-rich as other tools in this space.

View video

New Relic vs. AppDynamics vs. AppNeta:

It really comes down to your preferences as far as a few features go. The main exception is if you want an on-premises or hybrid deployment model, in which case AppDynamics is the easy choice. Otherwise, these tools hit a lot of the same spots. AppNeta focuses more on outlier elements and trends, while New Relic focuses more on averages and performance bottom lines.

Resources:

AppDynamics VS New Relic – Which Tool is Right For You? The Complete Guide

In order to bring your code into production, you need to have a usable software image that can be run and deployed. That's where build tools come in. They take your source code and compile it into an executable program. These days, build tools bring more to the table then just building your app, with features like dependency management as well.

In the Java world, there are three main build tools: Ant, Maven, and Gradle. These tools come before the actual production environment, but are a necessary element of the process, so we're covering them here. The primary differences for these tools come down to the amount of effort, usability, and external connections that the each has.

Ant

Maven

Gradle

Apache's Ant is an open source Java library and command-line tool used for automating software build processes. It's primarily used for building Java applications. Created in 2000, Ant is the original build tool in the Java space that's still being used today. You'll probably want to include Ivy with it if you want any dependency management capabilities.

When to use it: If you want nearly total control over how your build tool runs and are willing to put in the extra effort to get that.

Price: free

Pros:

- XML base means it works well with automatic tools.
- Once up and running, Ant gives you nearly full control over how things happen.
- Rich plugin ecosystem opens up a lot of possibilities, and it's easy to create custom plugins if what you need isn't available.
- Solid and extensive documentation.

Cons:

- XML base means less customization capabilities.
- Ant makes you do pretty much everything your self, which can be daunting.
- Build scripts are often very different, which makes understanding other projects difficult.
- As an old established tool, the community is fairly dead.

View video

Apache's Maven is a build automation tool primarily for Java projects, and is the most popular choice for Java developers today by the usage numbers. Unlike Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down.

When to use it: If you want the de facto tool and plugin repository. If you're running anything unusual with your other tools, Maven will support it. Well suited for large enterprises due to its very fast build speed.

Price: free

Pros:

- Extensive ecosystem for plugins.
- Common structure between builds makes understanding other projects easy.
- Full support for almost any CI, app server, or IDE tool.

Cons:

- Lots of download requirements for dependencies and plugins.
- Up and down documentation quality.
- Community is largely quiet.
- Customization is weak.

View video

Gradle is an open source build automation system. With version 1.0 released in 2012, Gradle aims to “combine the power and flexibility of Ant with the dependency management and conventions of Maven into a more effective way to build.” Its build scripts are written in Groovy, not XML, which creates a host of different advantages and disadvantages compared to Ant or Maven. Despite being a newer tool in this space, it’s seen widespread adoption.

When to use it: Gradle is designed for multi-project environments and incremental builds. It’s good if you’re comfortable with Groovy or are willing to get there. It’s also great for personal projects and SMBs.

Price: free

Pros:

- DSL base means you have a more customizable and streamlined tool.
- No required build script boilerplate makes for a simpler experience.
- Excellent documentation and active community. For example, Gradleware is a company designed around facilitating the adoption and use of Gradle through consultancy and other guidance.
- It’s simple to create custom plugins.

Cons:

- DSL base means you have a less straightforward and standardized tool.
- As the new kid on the block, the ecosystem for plugins and the like is less developed.
- As a newer tool, its support for CI tools and app servers isn’t as fleshed out as Maven or Ant.

View video

Ant vs. Maven vs. Gradle:

Which build tool you use will depend on your needs and preferences as a developer. If build speed is the most important element to you, Maven may be best. If the community and documentation is important, than Gradle could be the way to go. If full control is what you want, then you should lean towards Ant. All three of these are good options, so it really comes down to the different flavors and quirks of you and your environment. If you want to go with the flow of the crowd, Gradle seems to be rating the best these days in the majority of usage situations.

References:

Java Build Tools: Part 1 – An Introductory Crash Course to Getting Started with Maven, Gradle and Ant + Ivy

Java Build Tools – Part 2: A Decision Maker's Comparison of Maven, Gradle and Ant + Ivy

Feedback? Questions?

Contact us at hello@takipi.com