Metrics, monitors, alerts

Summary

Issue

We want to use metrics, monitors, and alerts, because we want to know how well our applications are working, and to know when there's a problem.

Decision

WIP.

Status

Gathering information. We are starting with the plausible ends of the spectrum: the most-recommended older free tool (Nagios) and the most-recommended newer paid tool (New Relic).

Details

Assumptions

We want to create web apps that are modern, fast, reliable, responsive, etc.

We want to buy rather than build.

Constraints

We want tooling that works well with our devops pipeline and with our deployment clouds.

Positions

We are researching positions now.

- AlertManager
- AppDynamics
- AppOptics
- Azure Monitor/Analytics/Insights/Dashboards
- Bosun
- Checkly
- Circonus
- Cloudwatch
- EFK

- ELK
- Grafana
- Grafana
- Graphite
- Graylog
- Healthchecks.io
- Heroku
- icinga2
- InfluxDB
- Instana
- Kafka
- Logagent
- Logz.io
- Loki
- Monitis
- Nagios
- Nagios
- Nagiosgraph
- NewRelic
- OpsGenie
- Outlyer
- PagerDuty
- Pagerduty
- PagerDuty
- Papertrail
- Prometheus
- Rollbar
- Scalyr

- Sematext Metrics, Logs, Experience, Tracing
- Sensu
- SignalFX
- Slack
- Splunk
- Stackdriver
- Stackstorm
- Telegraf
- Telegraf
- Thanos
- VictorOps
- Wavefront
- Zabbix

Argument

So far, Nagios and New Relic are the ends of the spectrum. Nagios is the oldest, simplest, free, viable tool. New Relic is the newest-featured, most-complete, paid, viable tool. We will start with evaluations of these. As needed, we will migrate into the spectrum.

So far, Zabbix has the best recommendations, and also offers the most complete capabilities.

So far, ELK has the best open-source build-over-buy popularity.

So far, Prometheus + Graphana have the best popularity.

Implications

TODO.

Related

Related decisions

The choices will affect testability, telemetry, and likely other systems such as for customer service, site reliability engineering, etc.

Related requirements

TODO.

Related artifacts

TODO.

Related principles

Easily reversible.

Need for speed.

Notes

A pretty good open source stack is:

- Prometheus for metrics and alerting based on metrics
- Grafana to display metrics
- Elasticsearch/Logstash/Kibana (ELK) for logs and structured events
- Pushover for mobile notifications

Freeform text messages vs structured event messages

Freeform text messages: for example, the kind of random stuff you would find in /var/log/messages, and something generated intentionally by the application. The messages are useful for identifying other things that are happening on the box like out of memory or hardware errors, but have a lot of junk.

Structured event messages: generated by the application, with a fixed or dynamic set of attributes, e.g. a HTTP request log, an accounting log, a user login.

Generally speaking, it's nice to log details about every request in a way that you can drill down based on attributes. So adding e.g. a userid or sessionid to everything lets you trace. Explicit tracing is also good, of course. Using ELK for this is kind of a poor man's https://www.honeycomb.io/

Graylog is easier

Graylog is easier to spin up from my experience.

Prometheus take some tuning

I am generally happy with Prometheus for metrics. The alerting takes some tuning, but is pretty good. It depends on your application. I think it's best to alert on end-user visible conditions, not underlying causes. For example, page load time is good, number of requests per second is not. Though zero requests per second indicates that something is wrong.

The advantage of a service is that they offer additional intelligence out of the box. I generally like Datadog. The services can be frighteningly expensive if you have a lot of data, and sometimes have pricing models which are not cloud friendly, e.g. charging per instance, when instances are dynamic. There is also a difference between services where every request is from a paid user and ones that are advertising related, so only some small percentage of the requests make you money. You can end up with a lot of data and not that much budget.

I work on some services that get 1B requests a day, so it makes sense to host our own monitoring and logging. If your volumes are lower, then hosted services are easier.

AWS services are mixed

My experience with AWS services has been mixed. Their Elasticsearch service has been flaky, so we run our own instances for that. CloudWatch metrics are expensive, so we generally only use them for "infrastructure" level metrics rather than the application, i.e. health-related metrics where AWS can know better what's going on than software running on the instance. CloudWatch Logs can be slow to update and don't have that much metadata. Running ELK helps with that. If I really want real time data, then using Kafka as the transport for logs is better. That's pretty well supported by Logstash. Managing a Kafka cluster is not for the faint hearted, though, there is a lot of exposed plumbing.

Kafka

Comment: Kafka can be super tricky at times, or Kafka can be rock solid and you almost forget about it being there tying everything together.

Comment: Kafka has been solid, but it was a surprising amount of work getting it going. I think of it like a relational database but you are only working at the "physical" layer, e.g. tablespaces, files and partitions. There were some times early on where the management utilities were lacking, and we had to write programs to e.g. reset a consumer group. http://howfuckedismydatabase.com/nosql/

Comment: We use Kafka as a "buffer" for log messages and a place where we can do real time stream processing on data that is coming from multiple servers. If we get a DDOS attack, then we need a way of analyzing data across multiple instances. If we are logging directly from the servers to ELK, the load can blow up the Elasticsearch cluster.

Comment: Kafka is good for us because if we get a DDOS attack, then we need a way of analyzing data across multiple instances. If we are logging directly from the servers to ELK, the load can blow up the Elasticsearch cluster.

Comment: Kafka does less work and is more efficient, so it can handle the load better. And we queue the Kafka work and retry. And having Kafka be overloaded doesn't affect users who are trying to do interactive work with Kibana, the way it would if Elasticsearch is struggling.

Comment: Stream processing is mostly looking for abuse, e.g. too much traffic from a single IP across the whole cluster, and then share the block across the whole cluster.

Comment: The logstash-output-kafka plugin is quite unreliable at the moment though. I've been bit by several of the issues on its GitHub issues page, that never seem to get fixed. I want to move away from using it, to sending directly from our apps to Kafka.

Comment: We are now sending structured events directly from the app to Kafka. The main motivation was touching the log data fewer times and avoiding reading and writing the disk multiple times. In high volume systems, logging can take more work than the app itself. I am thinking about making journald send logs directly as well, from a C program.

Loki

Keep a close eye on Loki. It’s not ready yet but when it is I’d expect it to be a better fit in this stack. Loki is a log aggregator created by grafana labs, it uses similar a scraping and tag syntax as Prometheus.

Prometheus + alertmanager + Rollbar + Graylog + Grafana

Prometheus + alertmanager for metrics. ❤️ Prometheus.

Rollbar/Graylog for logging/error reporting (there's some overlap here; a small service probably doesn't need both).

Currently, alerts just go to one of a few Slack channels that interested parties have notifications turned on for. If we were more serious about on-call they would go to PagerDuty/VictorOps/etc.

Grafana for graphs and dashboarding. Also eagerly looking forward to seeing if their upcoming logging facilities will obviate Graylog.

Thanos

We use Thanos as a front-end for our HA setup. It knows how to de-duplicate HA pairs.

We're currently keeping 6 months of local Prometheus data. This works reasonably well for us. But I'm just in the middle of rolling out bucket storage to our Thanos setup for long-term data storage. In theory, GCS storage will be about 30% cheaper than the GCE standard persistent disk we use right now.

We don't backup Prometheus data right now. The data just isn't really important to us beyond having enough for alerting. Our overall fleet deployment changes so much year to year that historical data older than a few months just isn't that interesting. It might be interesting to have a few core stats year-over-year, I may setup a core-stats set of recording rules and store those with Federation or just let Thanos take care of it.

EDIT: Minor disclaimer, I'm a Prometheus developer.

Prometheus HA

HA in Prometheus is done by duplication you run multiple poppers there’s ways to poll multiple and deduplicate the data

The scaling is by deciding network and having different Prometheus poll different parts of the network

Longterm storage isn’t proms strong point but is offloaded to something like influxes or timescaledb (which technically does the HA checkmark as well) article I read on it https://blog.timescale.com/prometheus-ha-postgresql-8de68d19b6f5?gi=7df160f10e07

Haven’t tried the longterm stuff yet as I’m still only experimenting and using it for short term graphs while librenms monitors my network for longterm

Datadog + PagerDuty + Threat Stack

We use Datadog (w/ PagerDuty) and Threat Stack and couldn't be happier. My only gripe with DD is the relatively high cost of metric storage.

Zabbix

Zabbix with custom scripts to monitor almost everything. Works like a charm.

Outlyer

I'm using Outlyer, but then I must disclaim that I work here, and dog fooding is a must.

Still need Graylog, Sentry and Statuscake to enhance.

Sounds biased, but having happily run Nagios and other monitoring systems internally, I would buy a hosted solution in any new gig and offload that pain.

Nagios + Nagiosgraph

We run Nagios for all monitoring and alerting. Alerts happen through e-mail (warnings and critical notifications) and audible app notifications (for critical alerts).

Nagiosgraph is used for visualisations.

This set-up has been very effective in keeping us comprehensively informed on what's happening in our environment. We run and monitor about 110 mission-critical servers and about 760 data points, and have had this morning system in place for over seven years.

I would like to also aggregate logs with Graylog or ELk at some point.

Prometheus + Grafana + AlertManager

Prometheus + Grafana + AlertManager via the awesome Prometheus Operator helm chart. Log still goes to LogDNA birch plan as we noticed ELK is too heavyweight for our humble min 3 max 5 nodes on GKE.

DataDog + Sentry + PagerDuty.

I used to run all my own monitoring solutions using all kinds of software including Nagios, Icinga, Zabbix, ELK, Greylog2, Influx, and many other tools, but the truth is that there's just too much effort involved in running your own monitoring infrastructure, especially when you can pay someone else such low rates for someone else to do it for you!

Paying others to run the monitoring infra frees my clients up to focus on running their platforms instead of monitoring the monitoring, meaning that the value they gain from the stability of their platform far outweighs any costs of Monitoring as a Service.

Sensu + Graphite + ELK

My company is real big on self-hosted stuff.

Sensu -> PagerDuty

Graphite/Grafana

ELK (Elasticsearch, Logstash, Kibana)

Prometheus + Alertmanager

Prometheus + Alertmanager for alerting, my team believe that simple monitoring is good monitoring.

Other systems like logging and tracing will provide rich context for diagnosing when the on-call receive an alert, but we never build alerting on these.

Sensu + Grafana + Graylog + Kibana + NewRelic.

Sensu, grafana, graylog, kibana, newrelic.

Prometheus + Circonus

Prometheus instrumented services => Circonus analytics and visualization

icinga2 + VictorOps + NewRelic + Sentry + Slack

We are using below services:

icinga2 for monitoring and VictorOps for alerting

NewRelic for details monitoring of the service

Sentry for error tracking in the service

Slack/Email is part of alerting which triggers from NewRelic or icinga2

AppDynamics + Papertrail + PagerDuty + Healthchecks.io + Stackdriver

AppDynamics

Papertrail

PagerDuty

Healthchecks.io

Stackdriver

icinga2 + elasticsearch

icinga2 with elasticsearch integration for analysis and graphite+grafana integration for graphs.

thanks to the flexibility of apply rules in icinga2, the developers can only see services they receive notifications for.

and through icinga2 director, programmers can easily define their own checks (which they do, every few days - 100 checks go out, 100 other checks go in) on big scale with zero hassle.

DataDog + New Relic + ELK + EFK + Sentry + Alertmanager + VictorOps

What do we have now:

DataDog for metrics

New Relic for application monitoring

ELK (Elastic Search + Logstash + Kibana) for the logs

Sentry (self-hosted) for logging exceptions

Emails + Slack + VictorOps for alerting (based on severity)

What do we want to have:

Prometheus for metrics (Grafana for visualization)

New Relic (probably Elastic Search APM) for the application monitoring

EFK (elastic search + fluentd + kibana) for logging. Probably, Loki by Grafana would be prod-ready till the time we get here

Sentry for the exceptions

Alertmanager + email + VictorOps for the alerts

Wavefront + Scalyr + PagerDuty + Stackstorm + Slack

Wavefront + Scalyr + PagerDuty + Stackstorm + Slack (Disclaimer: works at VMware)

Telegraf + Prometheus + InfluxDB + Grafana

Telegraf for server metrics like CPU, Disk, Memory and Network. We also use Telegraf for SNMP monitoring of our network devices.

Prometheus for application metrics. We code health checks into our application that Prometheus scrapes.

InfluxDB for time-series storage. This is where our Telegraf data gets sent.

Grafana for dashboards and alerts. The alerting engine isn't super robust, but it does the job. We also fire alerts into Slack.

What I don't have right now is a centralized logging solution. ELK is powerful but difficult to set up and manage, and I don't know of any free alternatives that are close enough to look into.

Sematext + Logagent + Experience

Sematext for metrics, for logs, for traces, soon for real user monitoring, too. Simpler/cheaper than using N different tools/services, IMHO.

For log shipping we used to use rsyslog and then we switched to Logagent.

For frontend crash reporting we use Sentry, but we'll switch to Experience shortly.

Disclaimer: I'm a Sematextan.

Azure Monitor/Analytics + OpsGenie

I wish Log Analytics had a better interface. We’re moving away from splunk, which was much easier to navigate.

Prometheus + Alertmanager + Grafana + Splunk + PagerDuty

Prometheus, Alertmanager, Grafana, Splunk, PagerDuty

You really do not want to run your own notification system. You can replace Splunk with ELK unless your Security team prefers Splunk.

Telegraf + Prometheus + Grafana + Alertmanager

Telegraf as a collector, Prometheus + Alertmanager for monitoring and alerting, integrated with slack channels and pagerduty for critical alerts. Grafana for host metrics visualization.

Prometheus + Grafana + Cloudwatch + sentry + kibana + elasticsearch

Prometheus for metrics + alerts

Grafana for Prometheus dashboards

Cloudwatch monitoring Prometheus instances

sentry for exception tracking

kibana + elasticsearch

graylog

prometheus Push Gateway for batch/cronjobs

SOP https://github.com/rapidloop/sop to „push/forward“ metrics from 1 Prometheus instance to another

clients either use the Prometheus clients . We try to use opencensus.io on the client side

PagerDuty + Monitis

PagerDuty + Monitis. Also some bespoke Azure Functions for testing the health of some services.

Looking to introduce Prometheus and Grafana this year

Prometheus + Grafana + Bosun

Prometheus for storing the time series data. Grafana for visualization. Bosun for alert management.

Azure Monitor/Analytics/Insights/Dashboards

Azure only shop, Azure Monitor, Log Analytics, App Insights, Azure Dashboards + Pager Duty

Grafana + Monitis + OpsGenie + Slack

Grafana for monitoring container services in Kubernetes via Prometheus

Monitis for end-to-end service monitoring mainly for web APIs and web applications

OpsGenie for alert management

Slack for getting status information from our systems

Checkly + AppOptics + Cloudwatch + Heroku + Pagerduty + Papertrail

Long time (dev)ops engineer here. Grew up on Nagios. Would love to have opinions on my bootstrapped SaaS https://checklyhq.com. We do API monitoring & site transaction monitoring with pretty in depth alerting.

I started Checkly because active / synthetic monitoring in the API space was a bit limited (and pricey). Browser based / scripted monitoring is even more proprietary and expensive. We use Puppeteer and keep pricing as low as possible.

Our monitoring stack:

Checkly (dog fooding...)

AppOptics (custom graphing)

AWS Cloudwatch & SNS for SMS messages.

built-in Heroku alerting.

Pagerduty

Papertrail

Instana + Logz.io + slack

Instana alerts us in slack about infrastructure problems or performance degradation, and we've configured logz.io to alert in slack on a certain volume of error level logs from the application layer.

SignalFX + Splunk + PagerDuty + Slack

Currently using: SignalFX, Splunk, PagerDuty and Slack. I'm not a huge fan of SignalFX although their support team is super friendly and responsive. I like Splunk (worth it if you can pay for it), PagerDuty and Slack.

I used to use the TICK stack where most of the C was really a G, that is Grafana although I did use Chronograf a little bit. That was awesome but it was a pain to manage. The classic SaaS vs self-hosting dilemma.

I have used DataDog, New Relic, Graylog, ELK and BugSnag. I like DataDog and New Relic a lot, Graylog is pretty good. I'm not a big ELK fan though. BugSnag is nice, I actually feel like tracking errors/exceptions is a pretty good standin for full log monitoring, in many cases.

ELK + Prometheus + Grafana

Just like others, we use ELK for logs and Prometheus+Grafana for everything else.

Maintaining this setup is easy if you give yourself permission to occasionally lose data. For example, if our ElasticSearch database gets into a funk (which happens every 2-3 months for us unfortunately) we don't bother with HA and instead dump the data and get on with our lives. If you absolutely must have HA or long term retention, good luck.

Datadog + Prometheus + Grafana

I set up Datadog on a month to month because when I got here, there was no monitoring and no alerting. Only a couple of our sites were being monitored every 5m for uptime. Datadog hands down is the easiest to

setup. When I finish addressing all the other issues, I will change to Prometheus+Grafana. Not 100% decided on log management yet.

Nagios + ELK

We have over 100+ products that we support.

For on-prem, it's mostly Nagios and ELK. For the cloud, we're migrating from DataDog to NewRelic.

Datadog vs. Site24x7 + StatusCake + PagerDuty + SumoLogic + Slack

We used to use datadog but found it to be way to expensive for our needs. Don't get me wrong its amazing but, it does have a huge cost. We were able to setup site24x7.com with an annual subscription for about 2-3 months of cost from DD.

Our monitoring stack:

Site24x7 - APM, External URL monitoring, SMTP mailflow monitoring, ssl expirations and process monitoring.

StatusCake - for URL monitoring and confirmation - It's our backup just in case site24x7 misses something (It does not) but SC is more flexible for external port and service monitoring for our needs.

Both tools escalate to PagerDuty, and then we get our escalations in slack.

SumoLogic - for log monitoring (it’s a great tool but a little complicated for our needs)

From slack we can ack, or remedy the alert.

We then have a lot of site24x7 automations that then connect to commando.io for 'BedOps' as we call it - where an alert gets triggered, we kick of a few scripts or automations as an attempt to remedy the situation (99% of the time the automation + our scripts keep us out of trouble)

We have internal runbooks in our KB for when the automatons fail or if there is something out of the scope that needs to be fixed.

Prometheus + AlertManager + Grafana + Stackdriver

Prometheus (Operator) / AlertManager / Grafana for metrics in our GKE clusters and VMs.

Google Stackdriver for Logs (as it’s included and active by default and is currently enough for our needs).

Zabbix

Zabbix for everything. No additional software needed.