International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

REAL TIME MONITORING OF SERVERS WITH PROMETHEUS AND GRAFANA FOR HIGH AVAILABILITY

ARUN KUMAR K[1], VINUTHA B S[2], VINAYADITYA B V[3]

[1]Assistant Professor, School of CS & IT, Jain University, Bangalore, Karnataka, India
[2,3]PG Scholar – School of CS & IT, Jain University, Bangalore, Karnataka, India

---

Abstract – Host content on the internet is all about connecting an interested audience to that content. Subtract any one element- audience, content or connection- from the equation and your website fails. Fortunately, even in the craziest online scenarios, you can ensure that your audience will always reach your content. It's called high availability. Three system admin concepts defined here work in concert to create a highly availability. Monitoring isdetermining if and when a server goes down. Monitoring software, like the daemon, will actively check to see if a server is reachable or not. Redundancy is having multiple versions of a server or component of the server operational. This also means you need to replicate the server's data. Without redundancy, when a server goes down, its content is inaccessible, but having a second server that can distribute its own replicated content will ensure anyone can still reach it. Failover is sending website traffic to an accessible server from an offline server.

1. INTRODUCTION

The combination of Prometheus and Grafana is becoming a more and more common monitoring stack used by Devops teams for storing and visualization time series data. Prometheus acts as the storage backend and Grafana as the interface for analysis and visualization. We don't regularly monitor our servers and we won't notice when things go badly, before they become a problem. The answer to that is probably "not really, at least, not before it's too late!" Admins who think they can just react when things fall down and go boom, or who feel they can check all their servers every day the good old fashioned way, by logging onto them, are either crazy, reckless, insomniacs, or they don't have enough servers to actually be considered system admins. You need to monitor your servers for resources, performance, and errors, as well as monitoring the apps they provide. Consider a file server. What happens when it runs out of space or an email server that can no longer send emails because there's a problem with a connector, or DNS. What about any server running at 100% CPU utilization. How responsive do you think it will be to your users. There's more to monitoring though, as anyone who has had a disk fail can tell you. Most disks start to throw errors long before they go code brown. If only you had a way to notice those errors before it was too late.

Server monitoring is basically a preventative measure to help you detect any issues before they cause any major issues that affect your productivity and your customer. Server monitoring is a process of continuously scanning servers on a designated network and scans the network for any failures or any irregularities that are detected by server monitoring software.

2. BACK END LANGUAGE

In this paper we used PHP programming language because, PHP is a general - purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites.

i. Version: PHP 7.0
ii. Java Script with bootstrap
iii. Backend: Elastic cloud computing (EC2)

2.1 DEPLOYMENT PLATFORM

For deploying of this application, we used amazon web Services. Amazon web services provide servers on rent to deploy application. Amazon web services is the one of the popular cloud based platform that Provide on-demand cloud computing platforms to Individuals, companies and governments, on a paid Subscription basis.

Platform: amazon web services (EC2 instance – Ubuntu 16.4 servers).

3. EXISTING SYSTEM AND PROBLEM STATEMENT

Before the use of the Prometheus, we were using cloudwatch to monitoring the servers where it's basically a metrics repository. Here are some of limitations of cloudwatch:

Limitation 1: Actions- 5 / alarm. This limit cannot be changed.

Limitation 2: Alarms- 10 / month. 5000 per region per account.

Limitation 3: Period: maximum value is one day (86,400 seconds). This limit cannot be changed.

PROBLEM STATEMENT

Before the use of the Prometheus, we were using cloudwatch to monitoring where exporting alarm alerting data is not available for further post-processing or analysis and have to navigate through various screens to get the metrics you want to add. It does not give you any recommendations, so you have to know what you're doing. This causes failure in monitoring the servers.

4. SYSTEM ARCHITECTURE

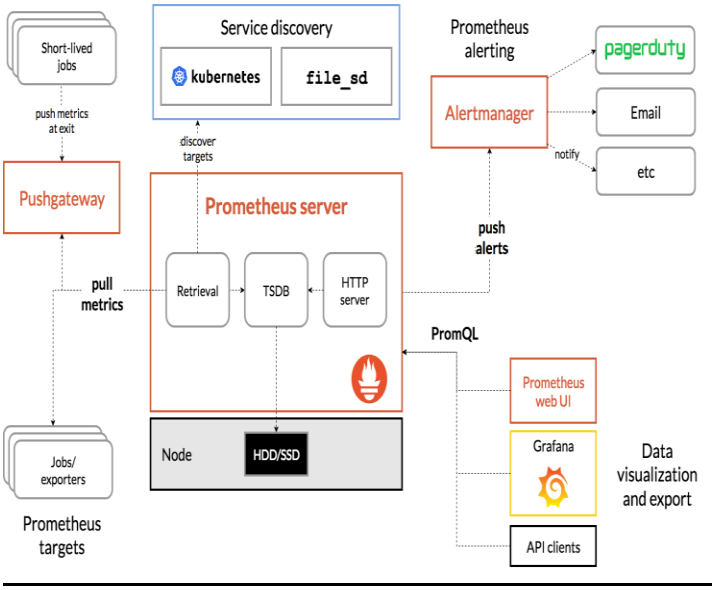

Fig 4.1 Prometheus and grafana workflow of servers monitoring

Prometheus is an open source monitoring and alerting toolkit for containers and microservices. It has become the mainstream, open source monitoring tool of choice for those that learn heavily on containers and microservices. The combination of Prometheus and grafana for storing and visualizing time series data. The data model identifies each time series not just with a name, but also with an unordered set of key-value pairs called labels. The PromQL query language allows aggregation across any of these labels, so you can analyse not just per process but also per datacenter and per service or by any other labels that you have defined. These can be graphed in dashboard systems such as Grafana.

Prometheus has a main central component called Prometheus Server. As a monitoring service, Prometheus servers monitor a particular thing. That thing could be anything: it could be an entire Linux server, a stand-alone Apache server, a single process, a database service or some other system unit that you want to monitor. In Prometheus terms, we call the main monitoring service the Prometheus Server and the things that Prometheus monitors are called Targets. So the Prometheus server monitors Targets.

4.1 Alerting - Knowing when things are going wrong is usually the most important thing that you want monitoring for. You want the monitoring system to call in a human to take a look.

4.2 Debugging - Now that you have called in a human, they need to investigate to determine the root cause and ultimately resolve whatever the issue.

4.3 Trending - Alerting and debugging usually happen on time scales on the order of minutes to hours. While less urgent, the ability to see how your systems are being used and changing over time is also useful. Trending can feed into design decisions and processes such as capacity planning.

4.4 Plumbing - At the end of the day all monitoring systems are data processing pipelines. Sometimes it is more convenient to appropriate part of your monitoring system for another purpose, rather than building a bespoke solution.

Each unit of a target such as current CPU status, memory usage (in case of a Linux server Prometheus Target) or any other specific unit that you would like to monitor is called a metric. So Prometheus server collects metrics from targets, stores them locally or remotely and displays them back in the Prometheus server. The Prometheus server scrapes targets at an interval that you define to collect metrics from specific targets and store them in a time-series database.

Prometheus provides client-libraries in a number of languages that you can use to provide health-status of your application. But Prometheus is not only about application monitoring, you can use something called Exporters to monitor third-party systems (Such as a Linux Server, MySQL daemon). An Exporter is a piece of software that gets existing metrics from a third-party system and export them to the metric format that the Prometheus server can understand.

5. IMPLEMENTATION

The implementation involves:

1. Creating service users –

For security purposes, we'll begin creating two new user accounts, Prometheus and node_exporter. We'll use these

accounts to isolate the ownership on Prometheus' core files and directories. Create these two users, and use the --no-create-home and --shell /bin/false options so that these users can't log into the server.

- sudo useradd --no-create-home --shell /bin/false prometheus
- sudo useradd --no-create-home --shell /bin/false node_exporter

2. Downloading Prometheus - First, downloads and unpacks the current stable version of Prometheus into your home directory.

- Cd
- https://github.com/prometheus/prometheus/releases/download/v2.0.0/prometheus-2.0.0.linux-amd64.tar.gz

3. Configuring Prometheus – In the etc/ Prometheus directory, use nano or your favorite text editor to create a configuration file named prometheus.yml. For now, this file will contain just enough information to run Prometheus for the first time.

-Sudo nano /etc/prometheus/prometheus.yml
- scrape_configs:
- job_name: 'prometheus'

scrape_interval: 5s

static_configs:

- Targets: ['localhost: 9090']

4. Running Prometheus – Startup Prometheus as the Prometheus user, providing the path to both the configuration file and the data directory.

-Sudo -u Prometheus /usr/local/bin/prometheus \

--config.file /etc/prometheus/prometheus.yml \

--storage.tsdb.path /var/lib/prometheus/ \

--web.console.templates=/etc/prometheus/consoles\
web.console.libraries=/etc/prometheus/console_libraries.

[Unit] Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=Prometheus
Group=Prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
web.console.templates=/etc/prometheus/consoles \
web.console.libraries=/etc/prometheus/console_libraries
[Install] WantedBy=multi-user.target.

6. FRONT END SCREEN

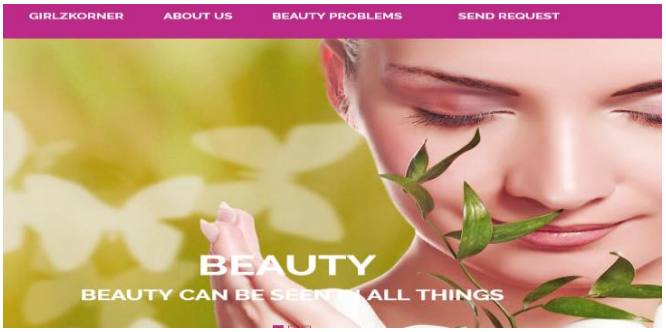

Fig 6.1 – WEB Application front page

This fig 6.1 consists of information about web application TheGirlzKorner.com where it provides the all information about the beauty problems.

```
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_
.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']
~
```

Fig 6.2 – Prometheus metrics to monitor server

This fig 6.2 shows how Prometheus collects metrics from monitored targets by scraping metrics endpoints on these targets. To edit the prometheus.yml file, use this command - > vi prometheus.yml.

```
./prometheus -config.file=prometheus.yml
[0000] Starting prometheus (version=1.5.2, branch=master, revision=bd1182d29f462c39544f94cc8
22830e1c64cf55b)  source=main.go:75
[0000] Build context (go=go1.7.5, user=root@a8af9200f95d, date=20170210-14:41:22)  source=ma
in.go:76
[0000] Loading configuration file prometheus.yml      source=main.go:248
[0000] Loading series map and head chunks...          source=storage.go:373
[0000] 0 series loaded.                               source=storage.go:378
[0000] Starting target manager...                     source=targetmanager.go:61
[0000] Listening on :9090                             source=web.go:259
```

Fig 6.3 – Prometheus installation for monitoring servers

This fig 6.3 shows the all the information about the Prometheus agent and files which is installed in the Ubuntu

server machine. To start the Prometheus this command from the Prometheus directory.

>./Prometheus –config.file=prometheus.yml

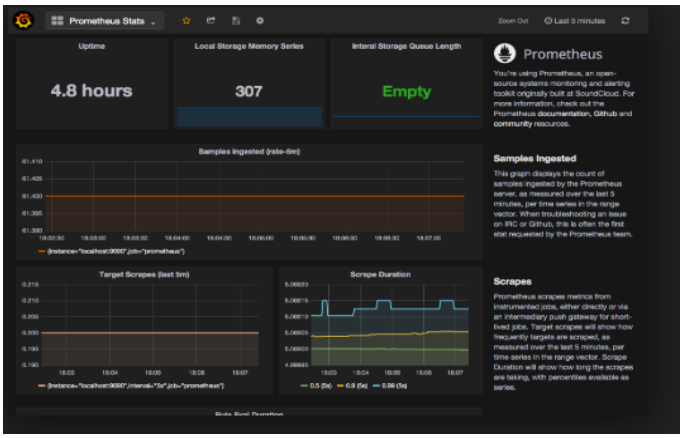

Fig 6.4 – Monitoring using Prometheus and grafana

This fig 6.4 shows the real time server monitoring using Prometheus and grafana for high availability.

7. CONCLUSION AND FUTURE ENHANCEMENT

Here we designed a real time monitoring of servers using the combination of Prometheus and grafana. Prometheus works well for recording any purely numeric time series. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength. It is designed for reliability, each Prometheus server is standalone, not depending on any network storage or other remote services. Grafana or other API (Application program interface)consumers can be used to visualize the collected data.

FUTURE ENHANCEMENT

we suggests to raise the awareness of Prometheus server importance at all layers of container cluster infrastructure, and work toward establishing best practices and standards for monitoring and metrics formats which improves the real time monitoring of servers for high availability.

REFERENCES

1. Cloud Computing: Concepts, Technology and Architecture (The Prentice Hall Service Technology Series from Thomas Erl) 1st Edition, Kindle Edition.
2. DeepQ Research Engineering Blog, Build a Monitoring Dashboard by Prometheus + Grafana.
3. Ikram Hawramani, Cloud computing for complete beginners: Building and scaling high performance web servers on the amazon cloud.
4. P. Mell and T. Grance, The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-145, 2011.
5. Andreas witting and Michael witting, Amazon web services in actions, ISBN- 1617292885, 17/10/2015.
6. James Turnbull, Monitoring with Prometheus, ISBN-9780988820289, June 2018.
7. Slawek Ligus, Effective monitoring and alerting: For web operations 1st edition.
8. Dave Avery, How to visualize your data with Grafana [Q+A], Big Data Zone, Sep-19.
9. G. Suciu, V. Suciu, R. Gheorghe, C. Dobre, F. Pop, and A. Castiglione. “Analysis of Network Management and Monoitoring Using Cloud Computing”, Computational Intelligence and Intelligent Systems, Springer, pp. 343-352, 2016.
10. M. Kim, Y. Kang, and Y. Yu, “Develop Total IT Service Monitoring System of Agentless Method for Total Management of based on Cloud Service Demand”, International Journal of Software Engineering and Its Applications Vol. 10, No. 1, pp. 1-14, 2016.
11. J. Swarna, C. S. Raja, D. Ravichandran, “Cloud Monitoring Based on SNMP”, Journal of Theoretical and Applied Information Technology, Vol. 40, No. 2, pp. 188-193, 2012.
12. M. Madan and M. Mathur, “Cloud Network Management Model – A Novel Approach to Manage Cloud Traffic”, International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol. 4, No. 5, pp. 9-20, 2014.
13. A. Anwar, A. Sailer, A. Kochut, C. O. Schulz, A. Segal and A. R. Butt, "Cost-Aware Cloud Metering with Scalable Service Management Infrastructure," 2015 IEEE 8th International Conference on Cloud Computing, New York City, NY, 2015, pp. 285-292.
14. A. Kaushik, “Use of Open Source Technologies for Enterprise Server Monitoring Using SNMP”, IJCSE, Vol. 2, No. 7, pp. 2246-2252, 2010.