How to monitor Kubernetes (k8s) clusters using free open-source tools - Prometheus and Grafana based container monitoring

Description

By Binu Nadarajan, Practice Head For Cloud Services

Background

Over The Past Decade, Software Development Using Agile Methodologies Has Brought About Many Changes In Organizational Processes. It Resulted In A Rethink Of Software Architecture That Allows For Faster Delivery Of New Features To Customers, Without Impacting The Whole Application. In Addition, Devops Tools And Techniques Enable Organizations To Increase Collaboration Between Teams And Automate The Building, Testing, And Deployment Of These Services.

Microservices-Based Architecture Has Come As A Boon For Organizations Who Wish To Achieve Quick Delivery Cycles. In This Architecture, An Application Is Broken Into Multiple Independently Deployable Services And Run As Containers. Many Container Orchestrators Are Available To Deploy And Manage Containers – And Kubernetes, Which Was Initially Designed By Google And Is Now Maintained By The Cloud Native Computing Foundation, Is By Far The Most Popular Open-Source Orchestrator Today.

Self-Managed Kubernetes Is The Oft Preferred Approach To Deploy And Maintain Containers To Circumvent Concerns About Vendor Lock-In. Typically, A Cluster Consists Of Many Nodes And Each Node Runs Hundreds Of Containers. Hence, After The Deployment Of Such A Large Number Of Containers, There Is Also The Need To Monitor Their Health, The Communications Between Them, And The Operations Inside Those Containers. Traditional Monitoring Tools And Processes Are Not Adequate To Monitor Kubernetes Clusters And Do Not Provide Sufficient Visibility Since Container Environments Have The Following Characteristics:

- They are built on a large distributed system
- They have a multi-layered architecture and hence we need to track numerous metrics
- They have a dynamic environment due to self-healing and auto-scaling capabilities
- Communications occur through virtual interfaces within and across nodes

This Blog Covers An Overview Of Open-Source Tools That Could Be Used To Effectively Monitor A Self-Managed Kubernetes Cluster.

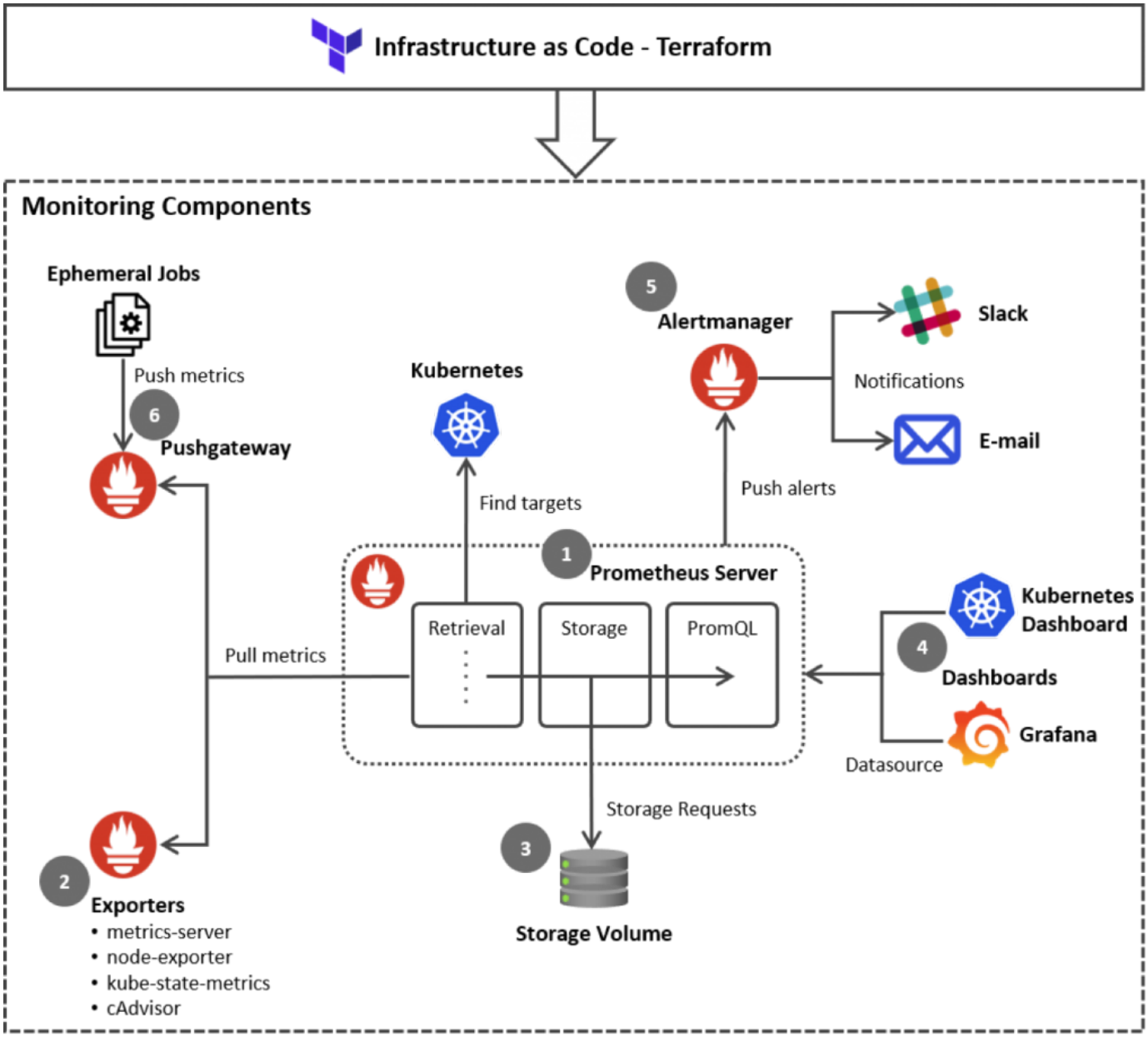
Key Performance Metrics

Host/Node – Examples Of These Metrics Are Resources Utilization At Every Node Like Network Bandwidth, Disk, Cpu, And Memory. Using These Metrics, One Can Determine Whether Or Not To Increase Or Decrease The Number And Size Of Nodes In The Cluster.

Pod – Examples Of These Metrics Are Utilization Of Resources (Like Cpu, Memory, File System, Etc.) And The State Of Each Pod. The Number Of Pods Running Shows If The Number Of Nodes Available Is Sufficient And If They Will Handle The Entire Workload If A Node Fails.

Docker Container – Examples Of These Metrics Relate To Utilization Of Resources (Like Cpu, Memory, Network Bandwidth, File System, Etc.)And The State Of Each Container. The Containers' Performance Reveals Whether The Resources Allotted For The Pod Are Sufficient, And It Helps To Decide The Placement Of Containers In Pods.

Object – Examples Of These Metrics Are Namespaces, Deployments, Secrets, Persistent Volumes And Claims, Service Accounts, Secrets, Network Policies, Daemon Sets, Etc., And The State Of Each Kubernetes Object. They Help In Troubleshooting And To Identify The Longevity And Stability Of The Cluster.



Prometheus Architecture Overview

As Shown In The Above Diagram, A Typical Kubernetes Cluster Monitoring System Built Using Prometheus Would Typically Need The Following Components:

- Prometheus Server to scrape and store time-series data.
- Exporters, namely client libraries or tools to export metrics from third-party systems.
- Storage including persistent storage on a local, on-disk time-series database. It can also be integrated with remote storage via remote read/write APIs.

- Dashboards to access visualizations of a collected time-series data in a few different ways using Prometheus native dashboard (WebUI) or advanced tools like Grafana.
- Alertmanager to notify a human operator about an occurrence of a specific condition, i.e., a rule defined as PromQL query. The actual alert is issued via integrations with external alerting tools like Slack, PagerDuty, Teams, etc.
- Pushgateway to push all the collected data to the database for short-lived jobs used for achieving and cleaning metrics.

The Following Is A Quick Overview Of The Tools Being Used.

Open-Source Container Monitoring Tools

Prometheus

Prometheus Consists Of A Central Component Named Prometheus Server. It Helps To Monitor Nodes, Which Are Called Targets. It Can Be A Single Target Or Multiple Targets Monitored For Different Metrics Like Cpu Usage, Memory Usage, Etc. It Stores All Its Data As A Time Series And Every Time Series Is Uniquely Identified By Its Metric Name And Optional Key-Value Pairs. This Data Can Be Queried Via The Promql, Functional Query Language And Visualized With A Built-In Expression Browser Or Consumed By External Systems Like Grafana Via The Http Api.

Prometheus Also Has An Alert Manager Component To Send Alerts Via E-Mail, Slack, Or Other Alerting Tools. One Can Define Rules Which Prometheus Server Reads And Fires Alerts When Defined Condition Gets Triggered.

Metrics Server

Metrics Server Collects Resource Metrics Like Cpu Or Memory Consumption For Containers Or Nodes From Kubelets And Exposes Them In The Kubernetes Api Server Through Metrics Api.

```
[root@master-node ~]# kubectl top nodes
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master-node    538m         13%    2257Mi          62%
worker-node1   266m         6%     1675Mi          46%
worker-node2   395m         9%     1736Mi          48%
worker-node3   395m         9%     1202Mi          33%
[root@master-node ~]# kubectl top pods --all-namespaces
NAMESPACE              NAME                                       CPU(cores)   MEMORY(bytes)
calico-system          calico-kube-controllers-5c6f449c6f-h6xwh   5m           18Mi
calico-system          calico-node-416d6                          47m          88Mi
calico-system          calico-node-gmllp                          30m          102Mi
calico-system          calico-node-ncvql                          44m          82Mi
calico-system          calico-node-w94lw                          27m          84Mi
calico-system          calico-typha-8bd664b68-25tj9               4m           26Mi
calico-system          calico-typha-8bd664b68-59qw9               3m           22Mi
calico-system          calico-typha-8bd664b68-dn9fc               3m           27Mi
calico-system          calico-typha-8bd664b68-z72sf               2m           22Mi
db-backend             db-backend-mysql-84c976476b-qnbkb          1m           474Mi
```

Sample Output Of Top Command

Node Exporter

Node Exporter Is Deployed In Every Node As Daemonset, And It Collects And Exposes A Wide Variety Of Hardware- And Linux Os-Related Metrics.

Sample Nodes' Metrics Screen

Kube-State-Metrics

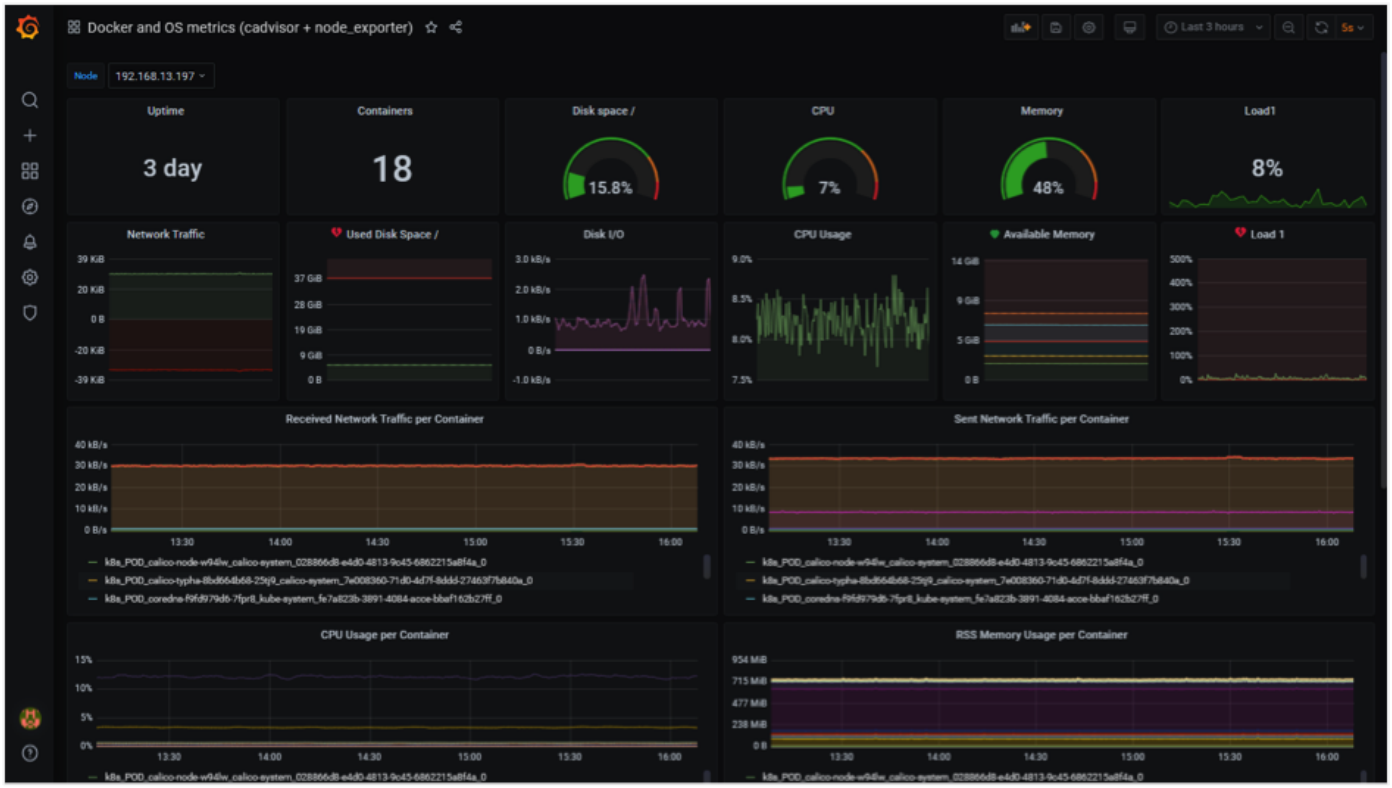
Kube-State-Metrics Listens To The Kubernetes Api Server And Exposes The State Of Kubernetes Objects Such As Config Map, Cron Job, Daemon Set, Deployment, Endpoint, Network Policy, Node, Persistent Volume, Pod, Service, Etc.

Kube-State-Metrics Are Different From Resource Utilization Metrics Of Kubernetes Objects Exposed By Metrics Server. Resource Utilization Metrics Are Directed Towards The Performance And Health Aspects Such As Network, Memory, And Cpu. In Contrast, Kube-State-Metrics Expose The Count And State Of Objects, Helping To Get An Overall Visibility Of What Is Going On Within The Kubernetes Cluster.

Sample Kubernetes Cluster Configuration Screen

Cadvisor

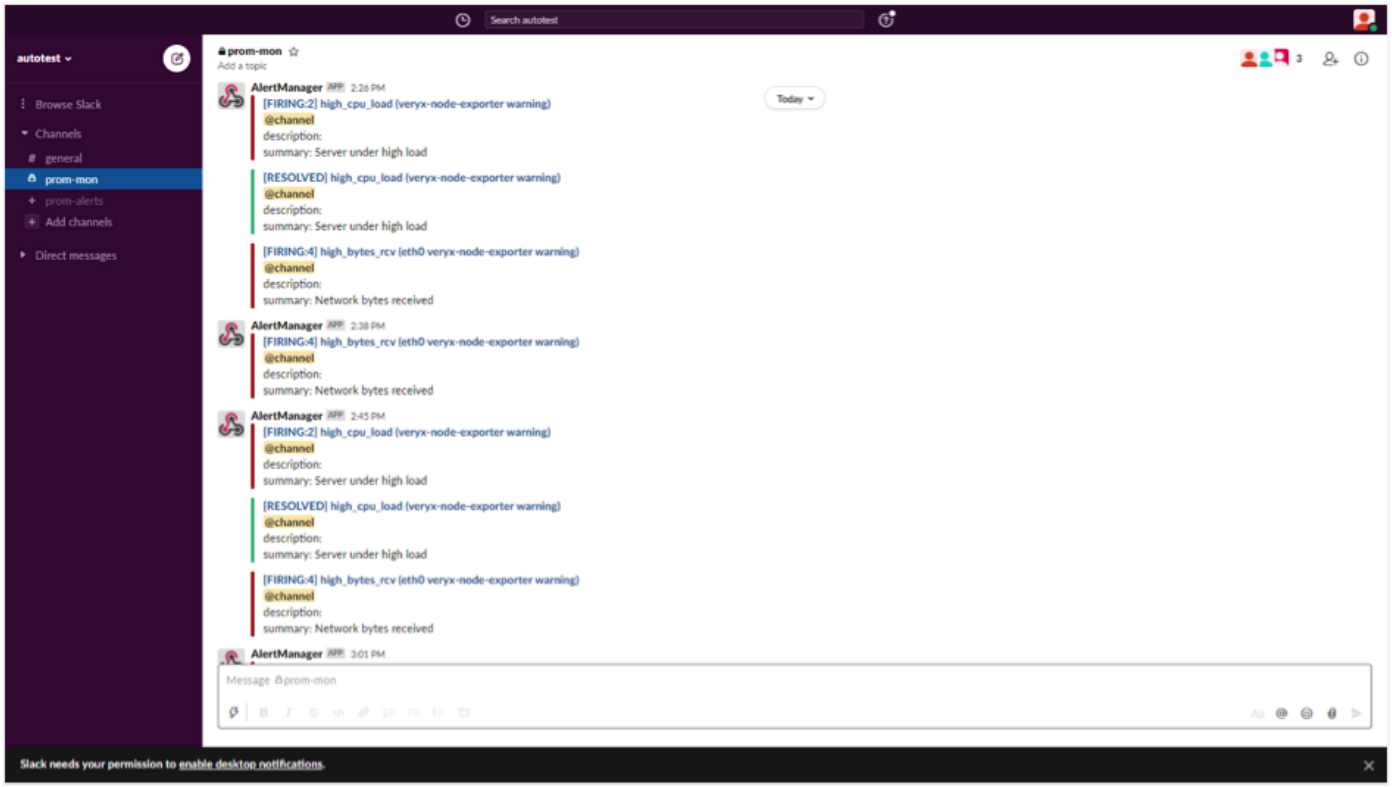
Cadvisor (Container Advisor) Provides An Understanding Of Their Running Containers' Resource Usage And Performance Characteristics. It Is A Running Daemon That Collects, Aggregates, Processes, And Exports Information About Running Containers In Each Node. Cadvisor Has Native Support For Docker Containers. The Kubelet Fetches Data From The Cadvisor And Exposes The Aggregated Pod Resource Usage Statistics Through The Metrics-Server Resource Metrics Api.



Sample Containers' Metrics Screen

Alertmanager

Alertmanager Is A Single Binary That Handles Alerts Sent By Prometheus Server And Notifies End-User Through E-Mail, Slack, Or Other Tools. Alert Rules Are Defined In Prometheus Server Configuration. Prometheus Server Scrapes Metrics From Its Client Applications. If An Alert Condition Hits, It Sends Alerts To The Alertmanager, Which Manages The Alerts Through Its Pipeline Of Grouping, Inhibition, Silencing, And Sending Out Notifications.



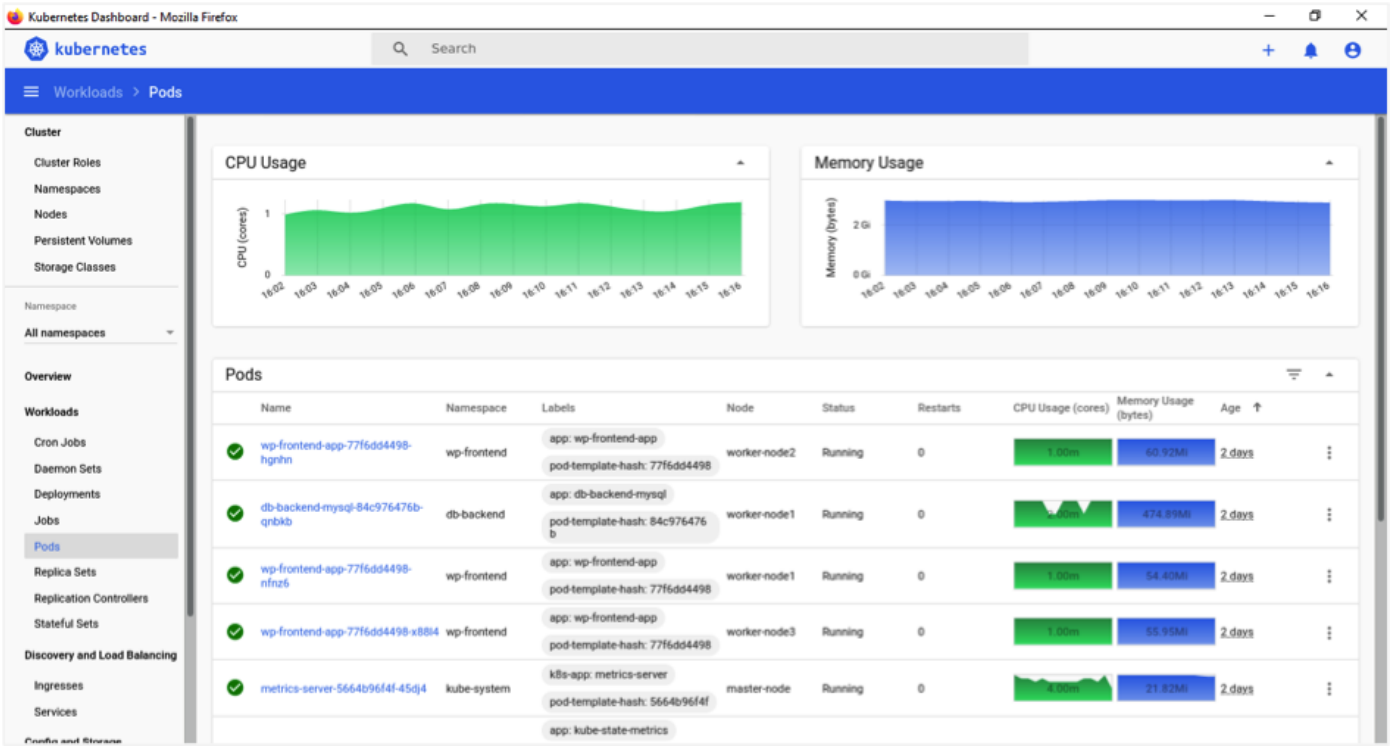
Sample Slack's Screen

Pushgateway

Pushgateway Is An Intermediary Service That Allows Pushing Time Series From Temporary Batch Jobs To An Intermediary Job That Prometheus Can Scrape.

Kubernetes Dashboard

Kubernetes Dashboard Is The Official Web-Based Ui For Kubernetes. The Dashboard Provides An Overview Of The Kubernetes Cluster As Well As The Individual Resources Running In It. It Has Many Features That Allow Users To Create And Manage Workloads And Do Discovery, Load Balancing, Configuration, Storage, And Monitoring.



Sample Kubernetes Dashboard Screen

Grafana

Grafana Is Used To Visualize Time-Series Data. Prometheus Acts As The Storage Backend And Grafana As The Interface For Analysis And Visualization. After Connecting Grafana With The Prometheus Data Source, It Helps To Create Dashboards For The Metrics To Be Monitored While Also Incorporating Readily Available Dashboards From The Official Website.

Terraform

Terraform Is A Robust Infrastructure As A Code Tool That Follows A Declarative Programming Model. The Above-Mentioned Tools Can Be Easily Deployed With Terraform And Maintained With Its Unique In-Built Features Like Source Version Control, Remote Backend, And Workspaces. It Is Also Integrated With Ci/Cd Tools Like Jenkins, Bamboo, And Gitlab For Complete Devops Automation.

Conclusion

In This Blog, I Have Provided A Quick Overview Of Tools Used To Set Up A Complete Monitoring System For Kubernetes Clusters.

At Veryx Technologies, We Help Customers Build Their Own Container Monitoring System Quickly For Their Self-Managed Kubernetes Cluster Using Open-Source Tools And Deliver It As An Infrastructure As A Code Solution Using Terraform. To Know More About Our Microservices Capabilities Click Here.

References

Prometheus Full Documentation, Examples And Guides – Https://Prometheus.io
Metrics Server – Https://Github.com/Kubernetes-Sigs/Metrics-Server
Node Exporter – Https://Github.com/Prometheus/Node_Exporter
Kube-State-Metrics – Https://Github.com/Kubernetes/Kube-State-Metrics
Cadvisor – Https://Github.com/Google/Cadvisor
Alertmanager – Https://Prometheus.io/Docs/Alerting/Latest/Alertmanager
Pushgateway – Https://Github.com/Prometheus/Pushgateway
Kubernetes Dashboard (Webui) – Https://Github.com/Kubernetes/Dashboard
Grafana – Https://Grafana.com/Docs/Grafana/Latest/Installation/Docker
Infrastructure As Code – Terraform – Https://Www.terraform.io

About The Author: Binu Nadarajan

Binu Nadarajan Is A Networking Industry Veteran With 16 Years Of Experience. Currently, He Heads The Cloud Services Practice At Veryx, Helping Businesses To Successfully Build Infrastructure In The Cloud As Well As To Migrate Their Applications To The Cloud. In His Spare Time, Binu Enjoys Reading And Playing With His Little Son.