B.M.S. COLLEGE OF ENGINEERING

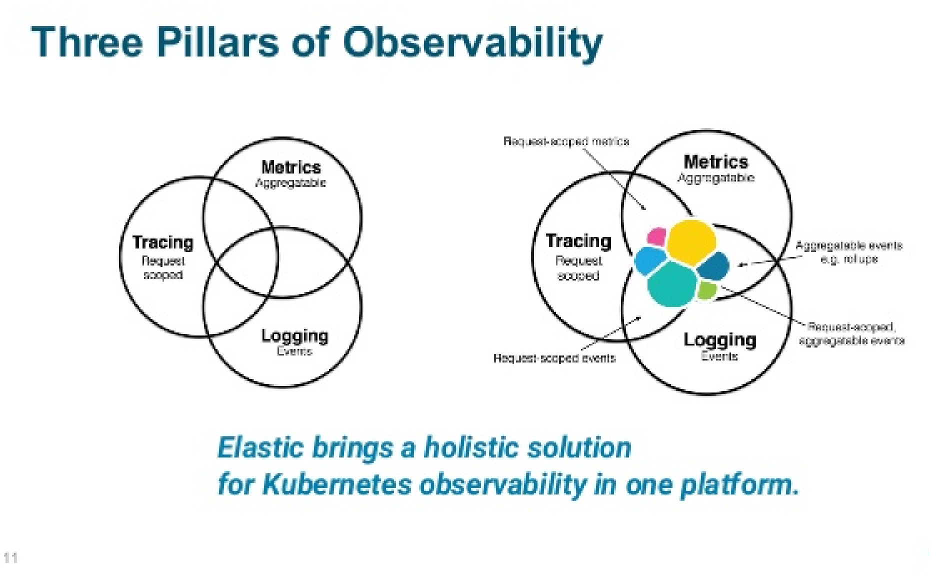
Autonomous Institute, Affiliated to VTU
Estd. 1946

DEPARTMENT OF CSE
CTY Project Work In collaboration with HPE

Project Title	Open source monitoring and observability stack on Kubernetes		
Student Team	Name: Pooja Srinivasan USN:1BM18CS069 SEM:UG-6th SEM	Name: Anusree Manoj K USN:1BM18CS017 SEM:UG-6th SEM	
	Name: Shikha N USN:1BM18CS149 SEM:UG-6th SEM	Name: Niha USN:1BM18CS060 SEM:UG-6th SEM	
Faculty Mentor	Dr. Nandhini V Associate Professor	**HPE Mentors**	Divakar Padiyar Sonu Sudhakaran
Review for the Period	19-03-2021	25-03-2021	
Task Given	Exploring and understanding concepts - Observability and Monitoring with respect to K8		
Difficulties Faced	None		
Libraries Used	None		
Github Link for the code:	None		
Code: None			
Observability [1,4]	The three pillars of observability are logs, traces and metrics. ● Logs represent discrete events and data in a structured textual form.They contain both the log the message emitted by a component of the service or the code(payload) and metadata, such as the timestamp, label, tag, or other identifiers.In Kubernetes, every container writes its logs to standard output which can be aggregated by a centralized logging solution for future analysis.		

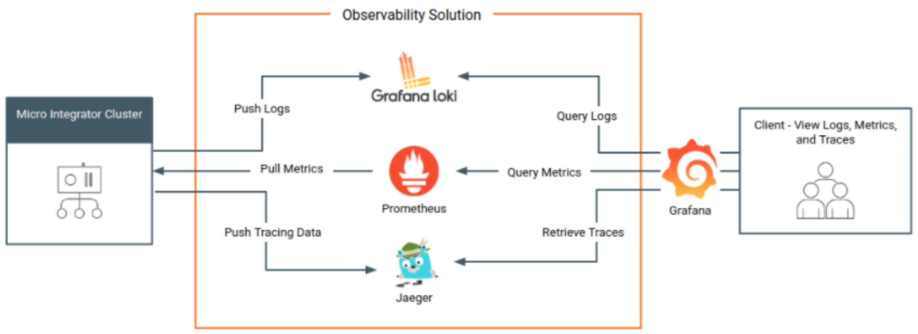
Some of the common options for consistently gathering and analyzing logs are: Collecting kubernetes logs with Fluentd**,**Storing and analyzing with ELK stack(Elasticsearch, Logstash, and Kibana).

- A trace is a representation of consecutive events which reflect an end-to-end request path in a distributed system.Traces helps us to understand the entire request flow, and hence it’s possible to troubleshoot most performance bottlenecks or analyze dependencies between different services. Tools used for Distributed traces are Istio,OpenCensus,Zipkin and Jaeger.
- A metric is a fundamental type of signal that can be emitted by a service or the infrastructure it's running on and numeric representation of data over intervals of time.It is used to understand the system health using telemetry signals.Metrics are stored in time series like Prometheus.
  Tools used for metrics are:
  Prometheus,Graphite,InfluxDB,OpenTSDB and OpenCensus.


Three Pillars of Observability
Metrics
Aggregatable
Tracing
Request scoped
Logging
Events
Request-scoped metrics
Metrics
Aggregatable
Tracing
Request scoped
Aggregatable events e.g. roll ups
Request-scoped, aggregatable events
Request-scoped events
Logging
Events
Elastic brings a holistic solution
for Kubernetes observability in one platform.
11

Tools for Observability are

- Jaeger - Distributed Systems Tracing with Kubernetes
- Fluentd - Collecting Kubernetes Logs
- Prometheus - Automating Collection and Storage of Observability Data
- Grafana - Visualizing Observability Data
- The ELK Stack—Analyzing Kubernetes Logs

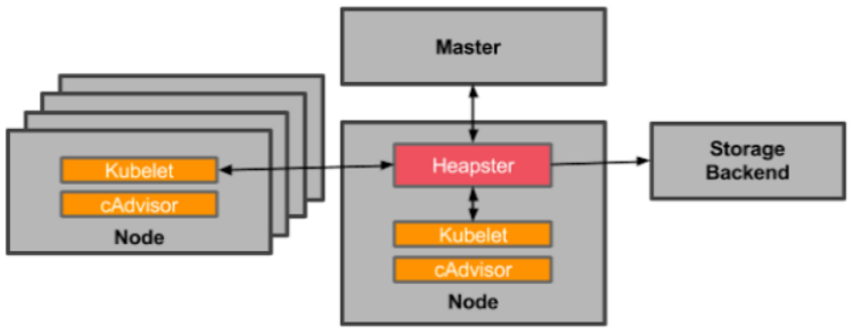
	 Observability Solution Micro Integrator Cluster Push Logs Grafana Loki Query Logs Pull Metrics Prometheus Query Metrics Push Tracing Data Retrieve Traces Jaeger Grafana Client - View Logs, Metrics, and Traces
Monitoring [2,3]	There are several Kubernetes metrics to monitor. These can be separated into two main components: ● Monitoring the cluster itself ● Monitoring pods. **Kubernetes Cluster Monitoring:** For cluster monitoring, the objective is to monitor the health of the entire Kubernetes cluster. As an administrator, we are interested in discovering if all the nodes in the cluster are working properly and at what capacity, how many applications are running on each node, and the resource utilization of the entire cluster. Measurable metrics: ● Node resource utilization – there are many metrics in this area, all related to resource utilization. Network bandwidth, disk utilization, CPU, and memory utilization are examples of this. Using these metrics, one can find out whether or not to increase or decrease the number and size of nodes in the cluster. ● The number of nodes – the number of nodes available is an important metric to follow. This allows you to figure out what you are paying for (if you are using cloud providers), and to discover what the cluster is being used for. ● Running pods – the number of pods running will show you if the number of nodes available is sufficient and if they will be able to handle the entire workload in case a node fails. **Kubernetes Pod Monitoring** ● Kubernetes Metrics - Using Kubernetes metrics, we can monitor how a specific pod and its deployment are being handled by the orchestrator. The

following information can be monitored: the number of instances a pod has at the moment and how many were expected (if the number is low, your cluster may be out of resources), how the on-progress deployment is going (how many instances were changed from an older version to a new one), health checks, and some network data available through network services.

- Pod container Metrics - These are available mostly through cAdvisor and exposed by Heapster, which queries every node about the running containers. In this case, metrics like CPU, network, and memory usage compared with the maximum allowed are the highlights.
- Application Metrics - These metrics are developed by the application itself and are related to the business rules it addresses

Monitoring Kubernetes Methods

- Using Kubernetes DaemonSets - This method consists of running metric-collection software usually, called an agent as a pod called a DaemonSet. DaemonSets ensure that every node in the cluster has a copy of the DaemonSet pod. Similarly, as nodes are terminated, the pod is removed as well.
- Using Heapster - Heapster is a bridge between a cluster and a storage designed to collect metrics. Unlike DaemonSets, Heapster acts as a normal pod and discovers every cluster node via the Kubernetes API. Using Kubelet (a tool that enables master-node communications) and Kubelet's open-source agent cAdvisor, Kubernetes and the bridge can store all relevant information about the cluster and its containers.



Monitoring with Heapster and cAdvisor in Kubernetes. Source: blog.kubernetes.io

	Different tools to Monitor kubernetes ● Heapster, InfluxDB, and Grafana with Kubernetes - This solution is used to monitor your Kubernetes cluster by using a combination of Heapster to collect metrics, InfluxDB to store it in a time series database, and Grafana to present and aggregate the collected information. ● Prometheus and Grafana with Kubernetes - Combination of these can be used for collecting Kubernetes metrics, with several tools, frameworks, and API's available for using it.Prometheus does not have a Heapster sink, so use of InfluxDB and Prometheus together for the same Grafana instance, collecting different metrics, whenever each one is easier to collect. ● Heapster and ELK Stack with Kubernetes - The ELK Stack is a combination of three components: Elasticsearch, Logstash, and Kibana, each responsible for a different stage in the data pipeline. ELK is widely used for centralized logging, but can also be used for collecting and monitoring metrics. There are various methods of hooking in ELK with Kubernetes. One method, for example, uses deploys fluentd as a DaemonSet. ● Datadog - Datadog is used to simplify monitoring by enabling organizations to monitor their applications and infrastructure more easily. Datadog uses a DaemonSet agent that will be deployed in every cluster node.
References	1. https://logz.io/blog/implementing-kubernetes-observability/ 2. https://logz.io/blog/kubernetes-monitoring/ 3. https://sematext.com/guides/kubernetes-monitoring/ 4. https://opensource.com/article/19/10/open-source-observability-kubernetes