**Logging**:

Using EFK:

1. ElasticSearch
2. Fluentd
3. Filebeat is light weight agent for logging purpose. But will use fluentd.
4. MetricBeat can be used in conjunction with ES to populate metric data as well, but will use dedicated Prometheus server with Grafana.
5. Kibana

ElasticSearch is processing engine. Fluentd works as an agent and deployed as a daemon set within k8s cluster.

Kibana Works as Dashboard endpoint, we use it for querying the logging.

**Setup**:

1. Install Elasticsearch as deployment object within k8s cluster, and expose the service as ClusterIP.
2. Install FluentD agent as daemon Set within k8s Cluster, and configure the configuration to it either injecting the dependency through customizing the Dockerfile which comes shipped with image from the provider or create a configmap object and mount it as a volume by editing the dameonset yaml file. or simply create config map with the same name is required by the daemon set object.
3. Configure Fluentd to work with ES endpoint and username and password if configured.
4. Define Source and Match directive to read the data and push the data based on the respective type of plugin defined.
5. Enable the Kibana plugin by adding Kibana module to ES instance either through modifying the existing Dockerfile. Which in most of the case not required as Kibana plugin comes packed with ES distribution. which we can use for visualization purpose.

Alternatively, we can enable X-Pack for authentication purpose. or we can use Search Guard from third party to have AUTHN in place.

**Monitoring**:

Using Prometheus and Grafana Dashboard.

Prometheus runs as a deployment object and can be exposed as ClusterIP, Load balancer, External Names, NodePort Type, we can also start as headless service.

Prometheus uses:

1. Prometheus Server -- deploy -- configured to scrape service endpoint which is annotated with  
   prometheus.io/path: /context path/actuator/Prometheus (service should be instrumented with Prometheus through pom.xml)  
   prometheus.io/port: port  
   prometheus.io/scrape: true
2. Kube State Metrics -- deploy -- populates k8s specific object information.
3. TSDB -- uses the time series data to store /data folder.
4. PrometheusNodeExporter-- daemon Set -- exports the node specific metrics to Prometheus server.
5. AlertManager -- deploy -- it adds an alerting functionality to Prometheus server., which can also be achieved using Grafana. It can be used of alerting the team through slack channel based on the alert rule defined. We can setup SMTP details within Alert Manager configuration for sending the email.
6. Install Grafana on ec2 instance and configure it to use Prometheus data source.

Prometheus Server is being used as an interface to interact with time series data and run PQL query against time series data to gather aggregated data.

Internally it uses tsdb to store the time series data on /data.

Grafana: is a dashboard to visualize the Metrics coming from Prometheus server as DataSource configuration.

**ANOTHER SOLUTION:**

**TICK Stack:**

1. **Telegraf**
2. **InfluxDB**
3. **Chronograf**
4. **Kapacitor**

It supports processing of the time series data using tick script langaugae. Which support two types:

1. Stream Task
2. Batch Task