Technical University  
of Applied Sciences  
Würzburg-Schweinfurt

Faculty of Computer Science and Business Information Systems

Würzburg, on  
25.05.2023

# Cloud Native Computing

**Exercise: Kubernetes Monitoring, Logging and Service Mesh**

**EFK Stack**

**-- Recreate the cluster --**

**Note: Please use Kubernetes Version v1.23.17, since Fluentd still needs PodSecurityPolicy to be functioned. ( PodSecurityPolicy was deprecated in Kubernetes v1.21, and removed from Kubernetes in v1.25)**

Review: We have learn that kubernetes help you to manage log collected from your application.

**kubectl run nginx --image nginx**

**kubectl logs nginx**

In this exercise we will see, how you can install EFK stack in Kubernetes using Helm charts

First install Elasticsearch. Go to this page to read instruction for the installation. Read and install it. **Please install Elasticsearch version 7.13.0**

**https://artifacthub.io/packages/helm/elastic/elasticsearch**

How is it going?

Note: If you find any problem, try to debug the problem with "kubectl logs", reading installation page or stackoverflow

You might need to change some value in default values.yaml, use this commands

**helm show values elastic/elasticsearch**

**helm install elasticsearch elastic/elasticsearch --set <properties>=<values>,…**

**helm install elasticsearch elastic/elasticsearch --set replicas=1,minimumMasterNodes=1 --version 7.13.0**

**helm install elasticsearch elastic/elasticsearch --value <path-to-value-yaml>**

See which components get created

**kubectl get all**

After elasticsearch pod get successfully created, install fluentd using this page. **Please install Fluentd version 0.2.6**

**https://artifacthub.io/packages/helm/fluent/fluentd**

How is it going?

Note: If you find any problem, try to debug the problem with "kubectl logs", reading installation page or stackoverflow

Suggested command

**helm install fluentd fluent/fluentd --version 0.2.6**

See which components get created

**kubectl get all**

See default values.yaml of fluentd on web page or using

**helm show values fluent/fluentd**

To check whether Elasticsearch is working or not

First way: Change type of elasticsearch service to LoadBalancer

**kubectl get svc**

**kubectl edit svc/elasticsearch-master**

**## change service type to LoadBalancer**

**localhost:9200**

Second way: using port-forward

**kubectl port-forward svc/elasticsearch-master 9200**

**localhost:9200**

See YAML files that fluentd helm chart generate for us

**helm template fluentd fluent/fluentd**

**helm template elasticsearch elastic/elasticsearch**

After fluentd pod get successfully created, install kibana using this page. **Please install Kibana version 6.8.22**

**https://artifacthub.io/packages/helm/elastic/kibana**

How is it going?

Note: If you find any problem, try to debug the problem with "kubectl logs", reading installation page or stackoverflow

Suggested command

**helm install kibana elastic/kibana --version 6.8.22**

See which components get created

**kubectl get all**

See default values.yaml of kibana on web page or using

**helm show values kibana elastic/kibana**

To check whether Kibana is working or not

First way: Change type of Kibana service to LoadBalancer

**kubectl get svc**

**kubectl edit svc/kibana-kibana**

**## change service type to LoadBalancer**

**localhost:5601**

Second way: using port-forward

**kubectl port-forward svc/kibana-kibana 5601**

**localhost:5601**

See YAML files that kibana helm chart generate for us

**helm template kibana elastic/kibana**

Navigate through Kibana dashboard and create index for look for logs

**Go to Analytics à Discover**

**Click Create index pattern**

**use "fluentd" as a name**

**Go to Analytics à Discover, again**

Trying to search for log of some pod

**kubernetes.pod\_name: elasticsearch-master-0**

At the end, every Pods in EFK should run

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Elasticsearch use perisistentVolume to store data

**kubectl get pvc**

**kubectl get pv**

Testing by creating a pod to create some log by running this command

**cat << EOF | kubectl apply -f -**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: counter**

**spec:**

**containers:**

**- name: count**

**image: busybox**

**args: [/bin/sh, -c, 'i=0; while true; do echo "This is demo log $i: $(date)"; i=$((i+1)); sleep 1; done']**

**EOF**

Check for logs which are just created. You can also expand to see more detail for each log

**kubernetes.pod\_name: counter**

**Prometheus and Grafana**

**-- Recreate the cluster --**

To enable Prometheus and Grafana, we will install it using helm charts

**https://artifacthub.io/packages/helm/prometheus-community/kube-prometheus-stack**

How is it going?

Note: If you find any problem, try to debug the problem with "kubectl logs", reading installation page or stackoverflow

You might need to change some value in default values.yaml, use this commands

Suggested command

**helm install prometheus-stack prometheus-community/kube-prometheus-stack**

See which components get created

**kubectl get all**

See values.yaml of this helm charts

**helm show values prometheus-community/kube-prometheus-stack**

See all YAML files using for creating prometheus stack

**helm template prometheus-stack prometheus-community/kube-prometheus-stack**

You will see that prometheus use operator as deployment to manage its components

**kubectl get deployment prometheus-stack-kube-prom-operator**

See YAML files of prometheus operator

**kubectl get deployment prometheus-stack-kube-prom-operator -o yaml**

Prometheus use Time-Series-Database to store its metric. It is deployed as StatefulSet

**kubectl get sts prometheus-prometheus-stack-kube-prom-prometheus**

Prometheus also monitors itself and deploys its own export. It will be type daemonset, since it will collect metric from every node in the cluster.

**kubectl get daemonset prometheus-stack-prometheus-node-exporter**

Prometheus stack also require plenty of configMap and Secret

**kubectl get configmap**

**kubectl get secret**

Prometheus stack create many CRD for its own purpose

**kubectl get crd**

If you want to configure rules for sending alert, you need to set it in this configmap

**kubectl get configmap prometheus-prometheus-stack-kube-prom-prometheus-rulefiles-0 -o yaml**

**kubectl get prometheusrules prometheus-stack-kube-prom-prometheus -o yaml**

Easiest way to access prometheus UI is through port-forwarding

**kubectl get svc**

**kubectl edit svc/prometheus-stack-kube-prom-prometheus**

**## change service type to LoadBalancer**

**localhost:9090**

In Prometheus UI, you can see

**Rules**

**Targets**

**Prometheus configuration**

**etc.**

You can use PromQL to directly search for some metric like these

**container\_cpu\_usage\_seconds\_total**

**node\_memory\_Active\_bytes**

**and many more…..**

Instead of using PromQL query metric, you can use Grafana. Use port-forward to access Grafana dashboard.

**kubectl get svc**

**kubectl edit svc/prometheus-stack-grafana**

**## change service type to LoadBalancer**

**localhost**

You need to provide username: admin and password: prom-operator to access Grafana

In dashboard, you can see

Alert rules

Query using PromQL under Explore

See Dashboard

Status of each pod