**Logging & Monitoring Kubernetes(EFK)**

1. Create a configmaps for fluentd configuration as below attachment.

<https://drive.google.com/file/d/1a7ksptV7pdSolJx-Qt4UaFpjck5R_nPy/view?usp=sharing>

We can get this config file from kubernetes official github repository. <https://github.com/kubernetes/kubernetes/blob/master/cluster/addons/fluentd-elasticsearch/fluentd-es-configmap.yaml>

1. Create the ServiceAccount, CLusterRole and RoleBinding for fluentd.

|  |
| --- |
| apiVersion: v1  kind: ServiceAccount  metadata:  name: fluentd-es  namespace: kube-system  labels:  k8s-app: fluentd-es  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  ---  kind: ClusterRole  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: fluentd-es  labels:  k8s-app: fluentd-es  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  rules:  - apiGroups:  - ""  resources:  - "namespaces"  - "pods"  verbs:  - "get"  - "watch"  - "list"  ---  kind: ClusterRoleBinding  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: fluentd-es  labels:  k8s-app: fluentd-es  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  subjects:  - kind: ServiceAccount  name: fluentd-es  namespace: kube-system  apiGroup: ""  roleRef:  kind: ClusterRole  name: fluentd-es  apiGroup: "" |

1. Create Daemonset objects for fluentd as we need the fluentd agent in all worker nodes, Which will push the logs to elasticsearch.

|  |
| --- |
| apiVersion: apps/v1  kind: DaemonSet  metadata:  name: fluentd-es-v2.2.0  namespace: kube-system  labels:  k8s-app: fluentd-es  version: v2.2.0  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  spec:  selector:  matchLabels:  k8s-app: fluentd-es  version: v2.2.0  template:  metadata:  labels:  k8s-app: fluentd-es  kubernetes.io/cluster-service: "true"  version: v2.2.0  # This annotation ensures that fluentd does not get evicted if the node  # supports critical pod annotation based priority scheme.  # Note that this does not guarantee admission on the nodes (#40573).  annotations:  scheduler.alpha.kubernetes.io/critical-pod: ''  seccomp.security.alpha.kubernetes.io/pod: 'docker/default'  spec:  priorityClassName: system-node-critical  serviceAccountName: fluentd-es  containers:  - name: fluentd-es  image: k8s.gcr.io/fluentd-elasticsearch:v2.2.0  env:  - name: FLUENTD\_ARGS  value: --no-supervisor -q  resources:  limits:  memory: 500Mi  requests:  cpu: 100m  memory: 200Mi  volumeMounts:  - name: varlog  mountPath: /var/log  - name: varlibdockercontainers  mountPath: /var/lib/docker/containers  readOnly: true  - name: config-volume  mountPath: /etc/fluent/config.d  # nodeSelector:  # beta.kubernetes.io/fluentd-ds-ready: "true"  terminationGracePeriodSeconds: 30  volumes:  - name: varlog  hostPath:  path: /var/log  - name: varlibdockercontainers  hostPath:  path: /var/lib/docker/containers  - name: config-volume  configMap:  name: fluentd-es-config-v0.1.4 |

1. Create ServiceAccount, ClusterRole and RoleBinding for elasticsearch.

|  |
| --- |
| apiVersion: v1  kind: ServiceAccount  metadata:  name: elasticsearch-logging  namespace: kube-system  labels:  k8s-app: elasticsearch-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  ---  kind: ClusterRole  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: elasticsearch-logging  labels:  k8s-app: elasticsearch-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  rules:  - apiGroups:  - ""  resources:  - "services"  - "namespaces"  - "endpoints"  verbs:  - "get"  ---  kind: ClusterRoleBinding  apiVersion: rbac.authorization.k8s.io/v1  metadata:  namespace: kube-system  name: elasticsearch-logging  labels:  k8s-app: elasticsearch-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  subjects:  - kind: ServiceAccount  name: elasticsearch-logging  namespace: kube-system  apiGroup: ""  roleRef:  kind: ClusterRole  name: elasticsearch-logging  apiGroup: "" |

1. Create a storage class for dynamic persistent Volume.

|  |
| --- |
| apiVersion: storage.k8s.io/v1  kind: StorageClass  metadata:  name: cloud-gp2  #namespace: kube-system  provisioner: kubernetes.io/aws-ebs  parameters:  type: gp2 |

1. Create a statefulset with 2 replicas for elasticsearch.

|  |
| --- |
| apiVersion: apps/v1  kind: StatefulSet  metadata:  name: elasticsearch-logging  namespace: kube-system  labels:  k8s-app: elasticsearch-logging  version: v6.2.5  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  spec:  serviceName: elasticsearch-logging  replicas: 2  selector:  matchLabels:  k8s-app: elasticsearch-logging  version: v6.2.5  template:  metadata:  labels:  k8s-app: elasticsearch-logging  version: v6.2.5  kubernetes.io/cluster-service: "true"  spec:  serviceAccountName: elasticsearch-logging  containers:  - image: k8s.gcr.io/elasticsearch:v6.2.5  name: elasticsearch-logging  resources:  # need more cpu upon initialization, therefore burstable class  limits:  cpu: 1000m  requests:  cpu: 100m  ports:  - containerPort: 9200  name: db  protocol: TCP  - containerPort: 9300  name: transport  protocol: TCP  volumeMounts:  - name: elasticsearch-logging  mountPath: /data  env:  - name: "NAMESPACE"  valueFrom:  fieldRef:  fieldPath: metadata.namespace  # Elasticsearch requires vm.max\_map\_count to be at least 262144.  # If your OS already sets up this number to a higher value, feel free  # to remove this init container.  initContainers:  - image: alpine:3.6  command: ["/sbin/sysctl", "-w", "vm.max\_map\_count=262144"]  name: elasticsearch-logging-init  securityContext:  privileged: true  volumeClaimTemplates:  - metadata:  name: elasticsearch-logging  spec:  storageClassName: cloud-gp2  accessModes: [ "ReadWriteOnce" ]  resources:  requests:  storage: 31Gi |

1. Create service for elasticsearch statefulset.

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: elasticsearch-logging  namespace: kube-system  labels:  k8s-app: elasticsearch-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  kubernetes.io/name: "Elasticsearch"  spec:  ports:  - port: 9200  protocol: TCP  targetPort: db  selector:  k8s-app: elasticsearch-logging |

1. Create kibina deployment

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: kibana-logging  namespace: kube-system  labels:  k8s-app: kibana-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  spec:  replicas: 1  selector:  matchLabels:  k8s-app: kibana-logging  template:  metadata:  labels:  k8s-app: kibana-logging  annotations:  seccomp.security.alpha.kubernetes.io/pod: 'docker/default'  spec:  containers:  - name: kibana-logging  image: docker.elastic.co/kibana/kibana-oss:6.2.4  resources:  # need more cpu upon initialization, therefore burstable class  limits:  cpu: 1000m  requests:  cpu: 100m  env:  - name: ELASTICSEARCH\_URL  value: http://elasticsearch-logging:9200  ports:  - containerPort: 5601  name: ui  protocol: TCP |

1. Create kibana service.

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: kibana-logging  namespace: kube-system  labels:  k8s-app: kibana-logging  kubernetes.io/cluster-service: "true"  addonmanager.kubernetes.io/mode: Reconcile  kubernetes.io/name: "Kibana"  spec:  ports:  - port: 5601  protocol: TCP  targetPort: ui  selector:  k8s-app: kibana-logging |

1. Finally create the ingress to access kibana and elasticsearch from outside the cluster(it can be public or private).

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  name: demo-ingress  namespace: kube-system  spec:  rules:  - host: kibana.pradipta.tech  http:  paths:  - path: /  backend:  serviceName: kibana-logging  servicePort: 5601  - host: es.pradipta.tech  http:  paths:  - path: /  backend:  serviceName: elasticsearch-logging  servicePort: 9200 |