Kubernetes on-premise cluster auditing.

For kubeadm made clusters only.

Alexey Nizhegolenko Follow
May 11, 2019 · 6 min read

Hello, in this short article I'll show a way of auditing the on-premise Kubernetes cluster, that was created by using kubeadm utility. When you put your brand-new Kubernetes cluster in production and give extended access to it for many peoples, you may wanna know who and what was doing on it. Especial if someone deletes the important deployments or same things.

In this case the Kubernetes team created an auditing inside the cluster, so you can enable it and get the full information about what are happens inside you cluster, in fact there is a lot of data you can get from audit.

Please use this documentation page to get more acquainted with auditing in kubernetes.

Well, in this article I'll show an easy way of how to enable the auditing for the kubeadm created cluster with saving logs in a specific log-file on master nodes. Not long ago the kubeadm utility got an option, that you can use for enabling audit when you create cluster with kubeadm. But you also can enable auditing for the existing cluster by making some modification in kubernetes API server container manifest.

So let's do it and see. Login on your first kubernetes master node and create an auditing policy file, in this file we'll specify the things we wanna get in to our audit logs, because by default you will get a plenty of records, so we need to configure only things we need.

First create the directory with audit policy file:

```
master1# mkdir /etc/kubernetes/policies

master1# vi /etc/kubernetes/policies/audit-policy.yaml

apiVersion: audit.k8s.io/v1beta1
kind: Policy
rules:
  # Do not log from kube-system accounts
  - level: None
    userGroups:
    - system:serviceaccounts:kube-system
  - level: None
    users:
    - system:apiserver
    - system:kube-scheduler
    - system:volume-scheduler
    - system:kube-controller-manager
    - system:node

  # Do not log from collector
  - level: None
    users:
    -
system:serviceaccount:collectorforkubernetes:collectorforkubernetes

  # Don't log nodes communications
  - level: None
    userGroups:
    - system:nodes

  # Don't log these read-only URLs.
  - level: None
    nonResourceURLs:
    - /healthz*
    - /version
    - /swagger*

  # Log configmap and secret changes in all namespaces at the
metadata level.
  - level: Metadata
    resources:
    - resources: ["secrets", "configmaps"]

  # A catch-all rule to log all other requests at the request level.
  - level: Request
```

Now we need to make some modifications for the Kubernetes API server pod manifest, the modifications are marked with bold text:

```
master1# vi /etc/kubernetes/manifests/kube-apiserver.yaml

apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=xx.xx.xx.xx
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-
servers=https://xx.xx.xx.xx:2379,https://xx.xx.xx.xx:2379,https://xx.
```

```
xx.xx.xx:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-
kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-
client.key
    - --kubelet-preferred-address-
types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-
client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-
client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-
ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    - --audit-policy-file=/etc/kubernetes/policies/audit-policy.yaml
    - --audit-log-path=/var/log/apiserver/audit.log
    - --audit-log-maxage=30
    - --audit-log-maxsize=200
    - --feature-gates=AdvancedAuditing=false
    image: k8s.gcr.io/kube-apiserver:v1.14.0
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: xx.xx.xx.xx
        path: /healthz
        port: 6443
        scheme: HTTPS
      initialDelaySeconds: 15
      timeoutSeconds: 15
    name: kube-apiserver
    resources:
      requests:
        cpu: 250m
    volumeMounts:
    - mountPath: /etc/ssl/certs
      name: ca-certs
      readOnly: true
    - mountPath: /etc/ca-certificates
      name: etc-ca-certificates
      readOnly: true
    - mountPath: /etc/kubernetes/pki
      name: k8s-certs
      readOnly: true
    - mountPath: /usr/local/share/ca-certificates
```

```
      name: usr-local-share-ca-certificates
      readOnly: true
    - mountPath: /usr/share/ca-certificates
      name: usr-share-ca-certificates
      readOnly: true
    - mountPath: /etc/kubernetes/policies
      name: policies
      readOnly: true
    - mountPath: /var/log/apiserver
      name: log
  hostNetwork: true
  priorityClassName: system-cluster-critical
  volumes:
  - hostPath:
      path: /etc/ssl/certs
      type: DirectoryOrCreate
    name: ca-certs
  - hostPath:
      path: /etc/ca-certificates
      type: DirectoryOrCreate
    name: etc-ca-certificates
  - hostPath:
      path: /etc/kubernetes/pki
      type: DirectoryOrCreate
    name: k8s-certs
  - hostPath:
      path: /usr/local/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-local-share-ca-certificates
  - hostPath:
      path: /usr/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-share-ca-certificates
  - hostPath:
      path: /etc/kubernetes/policies
      type: DirectoryOrCreate
    name: policies
  - hostPath:
      path: /var/log/apiserver
      type: DirectoryOrCreate
    name: log
status: {}
```

We just specified the audit policy file, then set the log file name and some log rotate options, also added the host path and mount path parameters too. More about other options you can found in this documentation.

Also, I was need to disable the **AdvancedAuditing** option, to make it work in old style with storing data in to log file, that why I set the **feature-**

gates=AdvancedAuditing=false.

After we have done these modifications, we need to restart **kubelet** service on master node, to apply our changes:

```
master1# systemctl restart kubelet
```

You may need to wait a bit before the audit log file will be available, because the Kubernetes write audit logs not always immediately. But if all was done right, you must got an audit log file in **/var/log/apiserver** directory. There you can find all events that was pass the specified filter in audit policy file previously.

```
master1# tail /var/log/apiserver/audit.log

{"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Request","aud
itID":"566dceca-c6a8-44e1-9d24-
40301fa341ea","stage":"RequestReceived","requestURI":"/apis/extension
s/v1beta1/namespaces/default/deployments/kubyk-
deployment/scale","verb":"update","user":{"username":"kubernetes-
admin","groups":
["system:masters","system:authenticated"]},"sourceIPs":
["xx.xx.xx.xx"],"userAgent":"kubectl/v1.14.0 (linux/amd64)
kubernetes/641856d","objectRef":
{"resource":"deployments","namespace":"default","name":"kubyk-
deployment","apiGroup":"extensions","apiVersion":"v1beta1","subresour
ce":"scale"},"requestReceivedTimestamp":"2019-05-
10T09:15:23.152283Z","stageTimestamp":"2019-05-10T09:15:23.152283Z"}
{"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Request","aud
itID":"566dceca-c6a8-44e1-9d24-
40301fa341ea","stage":"ResponseComplete","requestURI":"/apis/extensio
ns/v1beta1/namespaces/default/deployments/kubyk-
deployment/scale","verb":"update","user":{"username":"kubernetes-
admin","groups":
["system:masters","system:authenticated"]},"sourceIPs":
["xx.xx.xx.xx"],"userAgent":"kubectl/v1.14.0 (linux/amd64)
kubernetes/641856d","objectRef":
{"resource":"deployments","namespace":"default","name":"kubyk-
deployment","uid":"a1ef6d6c-6cbb-11e9-93cb-
763fc8adcb06","apiGroup":"extensions","apiVersion":"v1beta1","resourc
eVersion":"18148141","subresource":"scale"},"responseStatus":
{"metadata":{},"code":200},"requestObject":
{"kind":"Scale","apiVersion":"extensions/v1beta1","metadata":
{"name":"kubyk-
deployment","namespace":"default","selfLink":"/apis/extensions/v1beta
1/namespaces/default/deployments/kubyk-
deployment/scale","uid":"a1ef6d6c-6cbb-11e9-93cb-
763fc8adcb06","resourceVersion":"18148141","creationTimestamp":"2019-
05-02T09:21:14Z"},"spec":{"replicas":3},"status":
{"replicas":3,"selector":
{"app":"kubyk"},"targetSelector":"app=kubyk"}},"requestReceivedTimest
amp":"2019-05-10T09:15:23.152283Z","stageTimestamp":"2019-05-
10T09:15:23.177291Z","annotations":
{"authorization.k8s.io/decision":"allow","authorization.k8s.io/reason
":""}}
```

OK, now we can see what happening in our cluster, who is creating, deleting and scaling resources and so on. You may need to repeat this configuration on all other master nodes, so they also will be storing logs if master1 will be out of service.

Also, good idea to use **fluentd** or same tools for exporting this logs in to **ELK** stack or other logs storage.

In fact, we can send the audit logs in to the standard output of the API server container, this will provide us with availability to check the last audit logs just by running logs command:

```
kubectl logs kube-apiserver-kube-master1 -n kube-system
```

And this will be useful if you don't need a deep logging and just wanna have a quick look, about what were happening not so far away in cluster (really not so useful way, as for my opinion). Please mind that this method don't provide a deep logging at all.

For enabling this way auditing, we need to change the **audit-log-path** option in **/etc/kubernetes/policies/audit-policy.yaml** file like this:

```
..... ... . ..

- --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    - --audit-policy-file=/etc/kubernetes/policies/audit-policy.yaml
    - --audit-log-path=-
    - --audit-log-maxage=30
    - --audit-log-maxsize=200
    - --feature-gates=AdvancedAuditing=false
    image: k8s.gcr.io/kube-apiserver:v1.14.0
    imagePullPolicy: IfNotPres

..... ... ...
```

After restarting the **kubelet**, you can find the last audit logs by the Kubernetes API server pod logs command:

```
master1# systemctl restart kubelet


control# kubectl logs kube-apiserver-kube-master1 -n kube-system

{"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Request","aud
itID":"cb39199e-d1f6-4773-ad09-
b5bb9aa13626","stage":"ResponseComplete","requestURI":"/apis/extensio
ns/v1beta1/namespaces/default/deployments/kubyk-
deployment/scale","verb":"update","user":{"username":"kubernetes-
admin","groups":
["system:masters","system:authenticated"]},"sourceIPs":
["93.158.95.86"],"userAgent":"kubectl/v1.14.0 (linux/amd64)
kubernetes/641856d","objectRef":
{"resource":"deployments","namespace":"default","name":"kubyk-
deployment","uid":"a1ef6d6c-6cbb-11e9-93cb-
763fc8adcb06","apiGroup":"extensions","apiVersion":"v1beta1","resourc
eVersion":"18168853","subresource":"scale"},"responseStatus":
{"metadata":{},"code":200},"requestObject":
{"kind":"Scale","apiVersion":"extensions/v1beta1","metadata":
{"name":"kubyk-
deployment","namespace":"default","selfLink":"/apis/extensions/v1beta
1/namespaces/default/deployments/kubyk-
deployment/scale","uid":"a1ef6d6c-6cbb-11e9-93cb-
763fc8adcb06","resourceVersion":"18168853","creationTimestamp":"2019-
05-02T09:21:14Z"},"spec":{"replicas":1},"status":
{"replicas":2,"selector":
{"app":"kubyk"},"targetSelector":"app=kubyk"}},"requestReceivedTimest
amp":"2019-05-10T12:55:43.292027Z","stageTimestamp":"2019-05-
10T12:55:43.329950Z","annotations":
{"authorization.k8s.io/decision":"allow","authorization.k8s.io/reason
":""}}
I0510 12:55:43.587646       1 controller.go:102] OpenAPI
AggregationController: Processing item
k8s_internal_local_delegation_chain_0000000001
I0510 12:55:43.587787       1 controller.go:102] OpenAPI
AggregationController: Processing item
k8s_internal_local_delegation_chain_0000000002
I0510 12:55:44.588056       1 controller.go:102] OpenAPI
AggregationController: Processing item
k8s_internal_local_delegation_chain_0000000001
```

```
I0510 12:55:44.588257       1 controller.go:102] OpenAPI
AggregationController: Processing item
k8s_internal_local_delegation_chain_0000000002
I0510 12:55:45.588435       1 controller.go:102] OpenAPI
AggregationController: Processing item
k8s_internal_local_delegation_chain_0000000001
```

As you can see the logs are mixed with other Kubernetes API server logs, so this way may be not so clear, or you need to find a way to separate this logs.

That all for now, hope it'll helpful and you can make your Kubernetes cluster more "production ready", with enabling audit inside it.

Good luck.

Kubernetes Audit Logs Kubeadm