title: Kubernetes Audit Logs Integration description: Sematext Kubernetes Audit Logs integration is configured by pointing the Kubernetes API Server to send audit logs from your infrastructure to Sematext Logs.

Sematext offers a Kubernetes Audit logs receiver endpoint. Everything you need to do is to configure the Kubernetes API Server to send logs to it.

## Sematext Kubernetes Audit Logs Quick Start

Kubernetes audit logs are detailed descriptions of each call made to the Kubernetes API Server. They provide a chronological sequence of activities that lead to the state of the system at a specific moment. They are extremely useful for security and compliance purposes, telling you exactly who did what, when, and how. You can configure Kubernetes Audit Logs by using any of the two options below.

We recommend you use the Dynamic Backend because it does not require you to install any agents. You only need to configure the Kubernetes API Server to send audit logs to the URL we provide you with.

The Log Backend is a viable option if you already have Logagent running in your Kubernetes cluster.

### Kubernetes Audit Dynamic Backend

Configuring the Dynamic Backend simpler than a Log Backend. You need to editing the configuration of your Kubernetes API Server and add three new configuration parameters.

### Kops

If you use Kops for cluster management you run `kops edit cluster <cluster>` to open the configuration. You only need to set one field to enable the dynamic audit configuration:

```
spec:
  kubeAPIServer:
    auditDynamicConfiguration: true
```

By enabling this feature you are allowing for auditsinks to be registered with the API server. You can read more about it here.

Next, you have to add an auditsink resource. Create a file named `auditsink.yaml`.

```
apiVersion: auditregistration.k8s.io/v1alpha1
kind: AuditSink
metadata:
  name: k8sauditsink
  policy:
```

```
    level: Metadata
    stages:
    - ResponseComplete
  webhook:
    throttle:
      qps: 10
      burst: 15
    clientConfig:
      url: "https://logsene-k8s-audit-receiver.sematext.com/<LOGS_TOKEN>/"
      # For EU Region
      # url: "https://logsene-k8s-audit-receiver.eu.sematext.com/<LOGS_TOKEN>/"
```

Now apply the auditsink.yaml to the Kubernetes cluster.

```
kubectl apply -f auditsink.yaml
```

That's it. You're now shipping Kubernetes Audit logs to Sematext Logs for safekeeping.

### Kubeadm / Minikube

With Kubeadm or Minikube the Kubernetes API Server configuration will be in the `/etc/kubernetes/manifests/kube-apiserver.yaml` file, on the master node. Edit the file and add these three new configuration values.

```
...
spec:
  containers:
  - command:
    - kube-apiserver
    ...
    - --audit-dynamic-configuration
    - --feature-gates=DynamicAuditing=true
    - --runtime-config=auditregistration.k8s.io/v1alpha1=true
```

Save the changes and exit the file. This will trigger a restart of the Kubernetes API server.

Next, you have to add an auditsink resource. Create a file named `auditsink.yaml`.

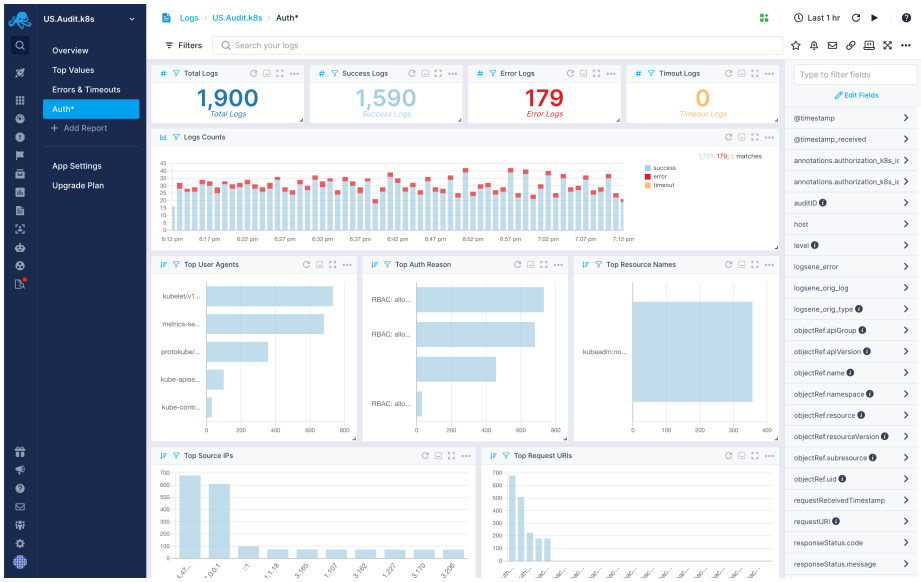
```
apiVersion: auditregistration.k8s.io/v1alpha1
kind: AuditSink
metadata:
  name: k8sauditsink2
  policy:
    level: Metadata
    stages:
    - ResponseComplete
```

```
webhook:
  throttle:
    qps: 10
    burst: 15
  clientConfig:
    url: "https://logsene-k8s-audit-receiver.sematext.com/<LOGS_TOKEN>/"
    # or for the EU Region
    # url: "https://logsene-k8s-audit-receiver.eu.sematext.com/<LOGS_TOKEN>/"
```

Now apply the auditsink.yaml to the Kubernetes cluster.

```
kubectl apply -f auditsink.yaml
```

That's it. You're now shipping Kubernetes Audit logs to Sematext Logs for safekeeping.




If you are not using the Dynamic Backend, then you can set up the Log Backend.

### Kubernetes Audit Log Backend

You need to enable audit logs only on the master nodes. First of all you need to create a policy to specify what will be recorded. A good example of the `audit-policy.yaml` file is the audit profile below from the official Kubernetes docs.

```
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
```

```yaml
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      # Resource "pods" doesn't match requests to any subresource of pods,
      # which is consistent with the RBAC policy.
      resources: ["pods"]
  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]

  # Don't log requests to a configmap called "controller-leader"
  - level: None
    resources:
    - group: ""
      resources: ["configmaps"]
      resourceNames: ["controller-leader"]

  # Don't log watch requests by the "system:kube-proxy" on endpoints or services
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
    - group: "" # core API group
      resources: ["endpoints", "services"]

  # Don't log authenticated requests to certain non-resource URL paths.
  - level: None
    userGroups: ["system:authenticated"]
    nonResourceURLs:
    - "/api*" # Wildcard matching.
    - "/version"

  # Log the request body of configmap changes in kube-system.
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["configmaps"]
    # This rule only applies to resources in the "kube-system" namespace.
    # The empty string "" can be used to select non-namespaced resources.
    namespaces: ["kube-system"]

  # Log configmap and secret changes in all other namespaces at the Metadata level.
```

```
  - level: Metadata
    resources:
    - group: "" # core API group
      resources: ["secrets", "configmaps"]

  # Log all other resources in core and extensions at the Request level.
  - level: Request
    resources:
    - group: "" # core API group
    - group: "extensions" # Version of group should NOT be included.

  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    # Long-running requests like watches that fall under this rule will not
    # generate an audit event in RequestReceived.
    omitStages:
      - "RequestReceived"
```

No matter which tool you use:

- Kops
- Minikube
- Kubeadm
- EKS
- GKE
- etc.

You need to add this policy file to your master nodes. To read more about auditing you can check out the official Kubernetes docs.

To enable the audit policy, you need to edit the definition of the Kubernetes API Server.

### Kops

If you use Kops for cluster management you run `kops edit cluster <cluster>` to open the configuration. In the `auditPolicyFile` field you need to specify the absolute path to your policy file.

```
spec:
  ...
  kubeAPIServer:
    auditPolicyFile: /etc/kubernetes/policies/audit-policy.yaml
    auditLogPath: - # log to stdout
    auditLogMaxAge: 10 # num days
    auditLogMaxBackups: 1 # the num of audit logs to retain
    auditLogMaxSize: 100 # the max size in MB to retain
```

Once you've configured logging the Audit logs to stdout you can use cluster-level logging to store these logs in a central location.

Install the Sematext Agent Helm chart:

```
helm install --name st-agent \
  --set infraToken=xxxx-xxxx \
  --set containerToken=xxxx-xxxx \
  --set logsToken=xxxx-xxxx \
  --set region=US \
  stable/sematext-agent
```

### Kubeadm / Minikube

Otherwise, if you're using Kubeadm or Minikube the Kubernetes API Server configuration will be in the `/etc/kubernetes/manifests/kube-apiserver.yaml` file, on the master node.

```
...
spec:
  containers:
  - command:
    - kube-apiserver
    ...
    - --audit-policy-file=/etc/kubernetes/policies/audit-policy.yaml
    - --audit-log-path=- # log to stdout
    - --audit-log-format=json
    ...
    volumeMounts:
    - mountPath: /etc/kubernetes/policies
      name: policies
      readOnly: true
  ...
  hostNetwork: true
  volumes:
  - hostPath:
      path: /etc/kubernetes/policies
      type: DirectoryOrCreate
    name: policies
```

Apply these changes by restarting the Kubelet.

```
sudo systemctl restart kubelet
```

Once you've configured logging the Audit logs to stdout you can use cluster-level logging to store these logs in a central location.

Install the Sematext Agent Helm chart:

```
helm install --name st-agent \
  --set infraToken=xxxx-xxxx \
  --set containerToken=xxxx-xxxx \
  --set logsToken=xxxx-xxxx \
  --set region=US \
  stable/sematext-agent
```

This will start both a log agent, and optionally a monitoring agent if you so wish. The agent will collect logs all logs from stdout, including the audit logs and send them to Sematext Logs.

If you prefer, you can set up Logagent with `kubectl` as well if you go here.