# Rubrik
# Integrating with ELK

Steven Tong

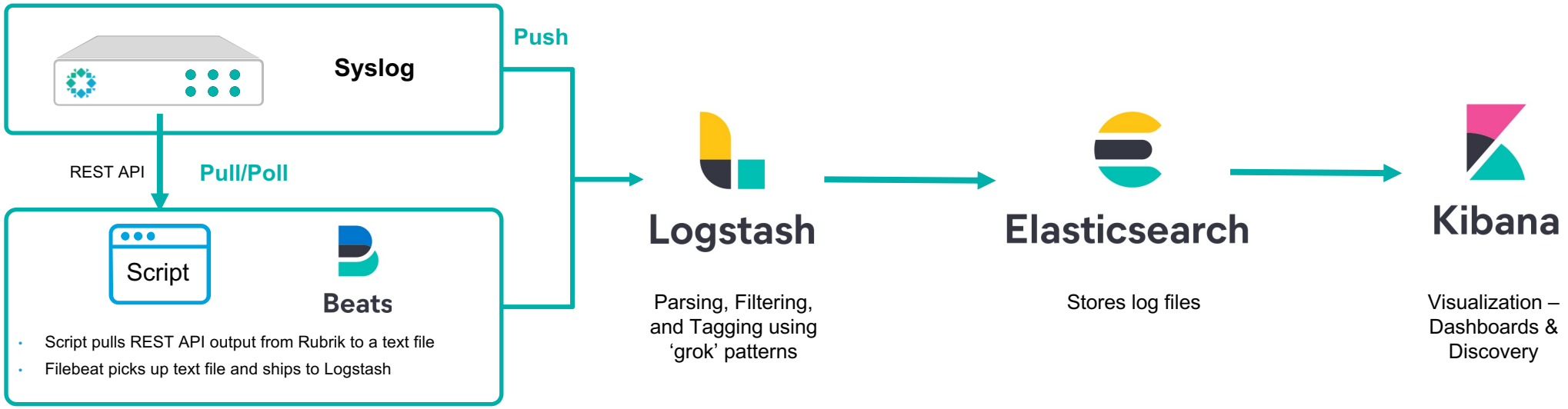
February 2021

rubrik

# Monitoring Rubrik

- Can Rubrik integrate with <ELK, Splunk, LogRhythm, SolarWinds, Datadog, log management/SIEM application of your choice>?
  - Yes, Rubrik can send (push) Syslog events or SNMP traps to the log/SIEM app or you can pull/poll for info using our REST API
  - In v5.3 we can also send Syslog events as SNMP traps – requires installing our three MIBs (download MIBs from the SNMP config screen in the CDM UI)
- What type of events does Rubrik send?
  - Things like job activity, audit events, SLA changes, hardware failures
- What if I wanted to build dashboards for things like capacity and SLA compliance?
  - That is when you would use the REST API to pull/poll for information to bring in any additional info you want

# Rubrik for ELK




https://github.com/stevenctong/rubrik/tree/main/elk

# ELK for Syslog

1. **Rubrik** – Configure Syslog to send events to Logstash
2. **Logstash** – Configure .conf files to parse Syslog messages to send to an Elasticsearch index
   - /etc/logstash/conf.d/
3. **Kibana** – Create an ‘index pattern’ in order to use the Elasticsearch index
4. **Kibana** – Discover Syslog data and create dashboards

rubrik

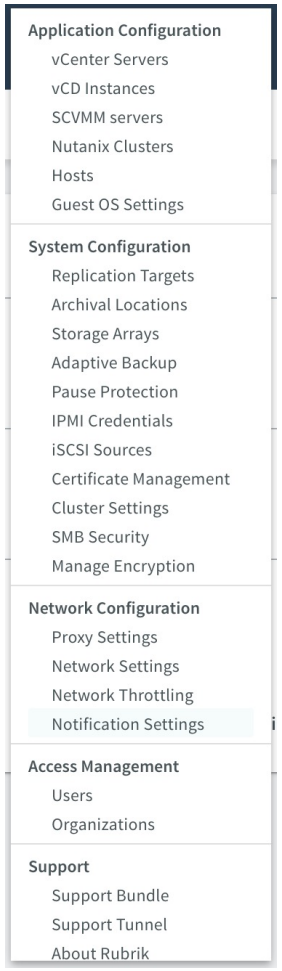
# ELK for Monitoring Status / Metrics

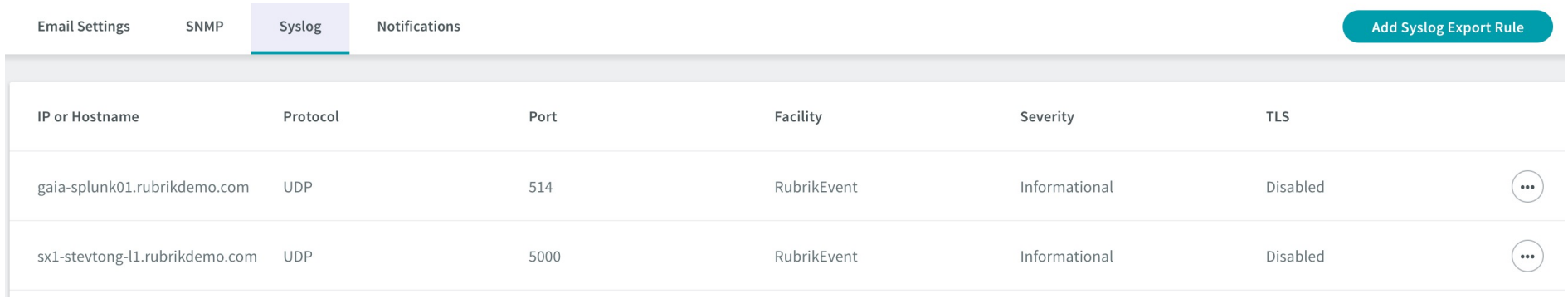
1. **Script Host** – Configure and schedule the script to poll REST API information at an interval and write the output to a log file in JSON format
   - get_rubrik_stats.sh + rubrik_cluster.conf – Bash script
   - get_rubrik.stats.ps1 – Powershell script
2. **Script Host** – Configure Filebeat to monitor the log file folder and send any updates to Logstash for further processing
   - /var/log/rubrikelk/*.log
3. **Logstash** – Configure .conf file to format fields to send to an Elasticsearch index
   - /etc/logstash/conf.d/
4. **Kibana** – Create an ‘index pattern’ in order to use the Elasticsearch index
5. **Kibana** – Discover the data and create dashboards

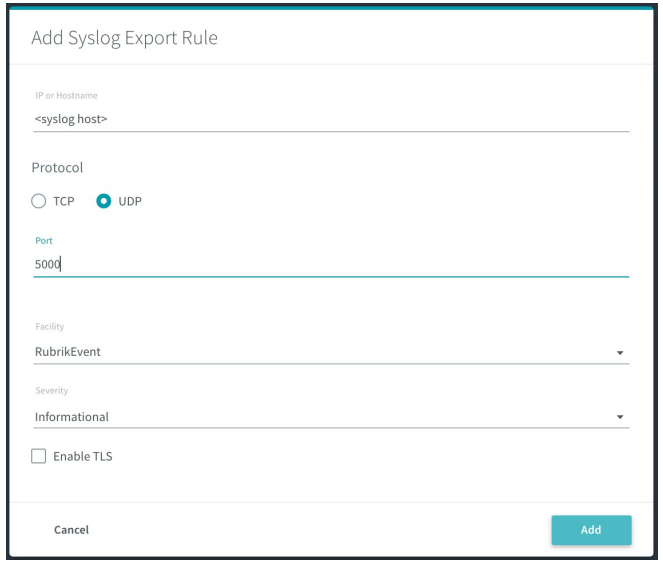
Note: We’re using Filebeat (logs) and not Metricbeat (metrics) so the data in Elasticsearch is log-based and not metric-based but we can still create the dashboards we want

rubrik

# Rubrik – Adding a Syslog Rule




- **Gear → Notification Settings**
- **Syslog** tab
- **Add Syslog Export Rule** button
- Fill in IP or hostname of your Logstash host, protocol (TCP or UDP), and port #
- Facility “RubrikEvent” and Severity “Informational” should be fine or change as you see fit

rubrik

# Logstash (Syslog)

- Logstash configuration files are used to configure Inputs → Filters → Output
  - Default directory – /etc/logstash/conf.d

- **Inputs** – Monitors the TCP & UDP ports for incoming Syslog messages
  - Syslog messages are received over non-privileged ports and then parsed
- **Filter** – Uses grok to parse the Syslog messages to fields
- **Output** – Fields are sent to Elasticsearch & any grok failures are also logged
  - Sends to a specific 'index' in Elasticsearch

10-logstash-rubrik-syslog.conf

```
input {
  tcp {
    port => 5000
    type => "syslog"
  }
  udp {
    port => 5000
    type => "syslog"
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => {
        "message" => [
          '%{TIMESTAMP_ISO8601:timestamp} %{NOTSPACE:rubrikNodeSN} %{NOTSPACE:rubrikComponent} %{NOTSPACE:rubrikNum} %{GREEDYDATA}
clusterName=\"%{NOTSPACE:rubrikClusterName}\" errorCode=\"%{GREEDYDATA:rubrikErrorCode}\"
errorMessage=\"%{GREEDYDATA:rubrikErrorMessage}\" errorReason=\"%{GREEDYDATA:rubrikErrorReason}\"
errorRemedy=\"%{GREEDYDATA:rubrikErrorRemedy}\" eventId=\"%{NOTSPACE:rubrikEventId}\" eventName=\"%{NOTSPACE:rubrikEventName}\"
eventSeriesId=\"%{NOTSPACE:rubrikEventSeriesId}\" eventSeverity=\"%{NOTSPACE:rubrikEventSeverity}\"
eventType=\"%{NOTSPACE:rubrikEventType}\" locationName=\"%{GREEDYDATA:rubrikLocationName}\" nodeId=\"%{NOTSPACE:rubrikNodeId}\"
nodeIpAddress=\"%{NOTSPACE:rubrikNodeIpAddress}\" objectId=\"%{GREEDYDATA:rubrikObjectId}\"
objectName=\"%{GREEDYDATA:rubrikObjectName}\" objectType=\"%{NOTSPACE:rubrikObjectType}\"
status=\"%{NOTSPACE:rubrikStatus}\"]\[%{GREEDYDATA}\] %{GREEDYDATA:rubrikEventMessage}\n',
          '%{TIMESTAMP_ISO8601:timestamp} %{NOTSPACE:rubrikNodeSN} %{NOTSPACE:rubrikComponent} %{NOTSPACE:rubrikNum} %{GREEDYDATA}
clusterName=\"%{NOTSPACE:rubrikClusterName}\" errorCode=\"%{GREEDYDATA:rubrikErrorCode}\"
errorMessage=\"%{GREEDYDATA:rubrikErrorMessage}\" errorReason=\"%{GREEDYDATA:rubrikErrorReason}\"
errorRemedy=\"%{GREEDYDATA:rubrikErrorRemedy}\" eventId=\"%{NOTSPACE:rubrikEventId}\" eventName=\"%{NOTSPACE:rubrikEventName}\"
eventSeriesId=\"%{NOTSPACE:rubrikEventSeriesId}\" eventSeverity=\"%{NOTSPACE:rubrikEventSeverity}\"
eventType=\"%{NOTSPACE:rubrikEventType}\" locationName=\"%{GREEDYDATA:rubrikLocationName}\" nodeId=\"%{NOTSPACE:rubrikNodeId}\"
nodeIpAddress=\"%{NOTSPACE:rubrikNodeIpAddress}\" objectId=\"%{GREEDYDATA:rubrikObjectId}\"
objectName=\"%{GREEDYDATA:rubrikObjectName}\" objectType=\"%{NOTSPACE:rubrikObjectType}\" status=\"%{NOTSPACE:rubrikStatus}\"]
%{GREEDYDATA:rubrikEventMessage}\n'
        ]
      }
      remove_field => [ "message" ]
    }
  }
}

output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
    index => "logstash-rubrik-syslog"
  }
  stdout { codec => rubydebug }

  if [type] == "syslog" and "_grokparsefailure" in [tags] {
    file { path => "/var/log/grokfailures/failed_syslog_events-%{+YYYY-MM-dd}" }
  }
}
```

rubrik

# Rubrik Syslog Examples

```
<134>1 2021-01-30T19:13:35.672Z RVM16CS014306 Rubrik-SprayServer 31932 Rubrik [eventDetail@49929 clusterName=\"sand1-rbk01\" errorCode=\"-\"
errorMessage=\"-\" errorReason=\"-\" errorRemedy=\"-\" eventId=\"884cc6df-293b-4d30-9b17-b62d1a0b3d56\"
eventName=\"Audit.CreatedManagedVolumeSnapshotAudit\" eventSeriesId=\"786a194d-1df6-44f6-90ea-0ec28ea59469\" eventSeverity=\"Informational\"
eventType=\"Audit\" locationName=\"-\" nodeId=\"RVM16CS014306\" nodeIpAddress=\"172.xx.yy.zz\" objectId=\"15bda393-bd81-4f1c-b731-5ae54f9500e2\"
objectName=\"-\" objectType=\"UserActionAudit\" status=\"Success\"] saphana_sh1-hxexsa-2_HXE created a snapshot 8ca08bdc-ea5c-48c1-8336-
02385cf192fa for managed volume 'sap_hana_sh1-hxexsa-2_HXE_SYSTEMDB_log'\n"
```

```
<134>1 2021-01-30T17:34:13.229Z RVM16CS014476 Rubrik-JobFetcherLoop 8833 Rubrik [eventDetail@49929 clusterName=\"sand1-rbk01\" errorCode=\"-\"
errorMessage=\"-\" errorReason=\"-\" errorRemedy=\"-\" eventId=\"f4f549fb-4a3b-4eae-837e-3e97d87cd708\"
eventName=\"Snapshot.LogBackupSucceeded\" eventSeriesId=\"2a517e35-4d40-4a57-a698-bc3174a602bd\" eventSeverity=\"Informational\"
eventType=\"Backup\" locationName=\"Oracle Host sx1-shawmcel-l3\" nodeId=\"RVM16CS014476\" nodeIpAddress=\"172.xx.yy.zz\" objectId=\"d805203a-
3eb4-4289-bf7b-10505b41e794\" objectName=\"smcdb01\" objectType=\"OracleDb\" status=\"Success\"][mdc@49929
JobContextQueryStatsSnapshotId=\"1731619182330813365\" instanceId=\"-1\" jobId=\"\" jobType=\"\" ndc=\"CREATE_ORACLE_LOG_SNAPSHOT_d805203a-
3eb4-4289-bf7b-10505b41e794:::906\" parentSpanId=\"0\" profile=\"false\" spanId=\"313acb5790465439\" taskId=\"\"
tracerId=\"b97fdd05fcaae4d6313acb5790465439\"] Completed log backup of Oracle Database 'smcdb01'\n
```

- Rubrik Syslog messages follow a standard format
- One of the formats has an extra array so two ‘grok’ rules will need to be created to parse the messages

rubrik

# Parsing With Grok

```
<131>1 2021-02-16T07:42:54.457Z RVM16CS014476 Rubrik-JobFetcherLoop 8833 Rubrik [eventDetail@49929 clusterName="sand1-rbk01" errorCode="RBK20700003" errorMessage="Unable to
resolve host name ‘sx1-test-w1.rubrikdemo.com'." errorReason="Failed to look up the IP address for host ‘sx1-test-w1.rubrikdemo.com'." errorRemedy="Make sure correct host
name is used and DNS server is configured correctly." eventId="a54c7b50-70b2-48e2-9de6-930a3f104c10" eventName="Snapshot.BackupFailed" eventSeriesId="5a39f449-7b31-42fc-
822c-6dd8653cace0" eventSeverity="Warning" eventType="Backup" locationName="-" nodeId="RVM16CS014476" nodeIpAddress="172.xx.yy.zz" objectId="f6d37b0f-32fa-49af-b543-
4d96f3e0e0e9" objectName=”sx1-test-w1.rubrikdemo.com volumes" objectType="VolumeGroup" status="Failure"][mdc@49929 JobContextQueryStatsSnapshotId="3510858650583934103"
instanceId="-1" jobId="" jobType="" ndc="CREATE_VOLUME_GROUP_SNAPSHOT_f6d37b0f-32fa-49af-b543-4d96f3e0e0e9:::8068" parentSpanId="0" profile="false" spanId="966367648eb0bf18"
taskId="" tracerId="e95fc547e0d25c2966367648eb0bf18"] Failed backup of Volume Group ‘sx-test-w1.rubrikdemo.com volumes'. Reason: RBK20700003 - Unable to resolve host name
‘sx1-test-w1.rubrikdemo.com'.
```

**Grok statements that parse the above example**

- %{TIMESTAMP_ISO8601:**timestamp**} - `2021-02-16T07:42:54.457Z`
- %{NOTSPACE:**rubrikNodeSN**} - `RVM16CS014476`
- %{NOTSPACE:**rubrikComponent**} - `Rubrik-JobFetcherLoop`
- %{NOTSPACE:**rubrikNum**} - `8833`
- %{NOTSPACE} - `Rubrik`
- %{GREEDYDATA} clusterName=\” - `[eventDetail@49929 clusterName=“`
- %{NOTSPACE:**rubrikClusterName**} - `sand1-rbk01`
- \“ errorCode=\” – `“ errorCode=“`
- %{GREEDYDATA:**rubrikErrorCode**} - `RBK20700003`
- \” errorMessage=\” – `“ errorMessage=“`
- %{GREEDYDATA:**rubrikErrorMessage**} - `Unable to resolve host name ‘sx1-test-w1.rubrikdemo.com’.`
- \” errorReason=\” – `“ errorReason=“`
- %{GREEDYDATA:rubrikErrorReason} - `Failed to look up the IP address for host ‘sx1-test-w1.rubrikdemo.com’.`

...

Some ‘grok’ resources

- https://grokdebug.herokuapp.com/
- https://streamsets.com/documentation/datacollector/latest/help/datacollector/UserGuide/Apx-GrokPatterns/GrokPatterns_title.html
- https://www.kartar.net/2014/09/when-logstash-and-syslog-go-wrong/

- **Green** – The field + message that is stored in the matched pattern
- **Red** – Matched pattern that is not stored in a field (thrown away0)

rubrik

# Scripts & Filebeat

- A script can be scheduled to run periodically to gather status and metrics from Rubrik
  - get_rubrik_stats.sh (bash)
  - get_rubrik_stats.ps1 (Powershell)
- The data is captured in JSON format to a log file
- Filebeat is configured to look for updates to those log files in a certain directory and send it to Logstash for further processing
  - /var/log/rubrikelk/*.log

filebeat.yml

```
###################### Filebeat Configuration Example #########################

# ============================== Filebeat inputs ===============================

filebeat.inputs:

- type: log

  # Change to true to enable this input configuration.
  enabled: true

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    #- /var/log/*.log
    - /var/log/rubrikelk/*.log


  # In the event the Filebeat cannot send to output, close open files after timeout
  # Data loss is a potential side effect
  close_timeout: 1h


# ------------------------------ Logstash Output -------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

rubrik

# Logstash (Filebeat)

- **Inputs** – Uses ‘beats’ filter
  - Beats messages are received over non-privileged port and then parsed
- **Filter** – The data being sent should be in JSON format
  - Re-name fields as needed
  - Whitelist the fields that will be sent to Elasticsearch
- **Output** – Elasticsearch

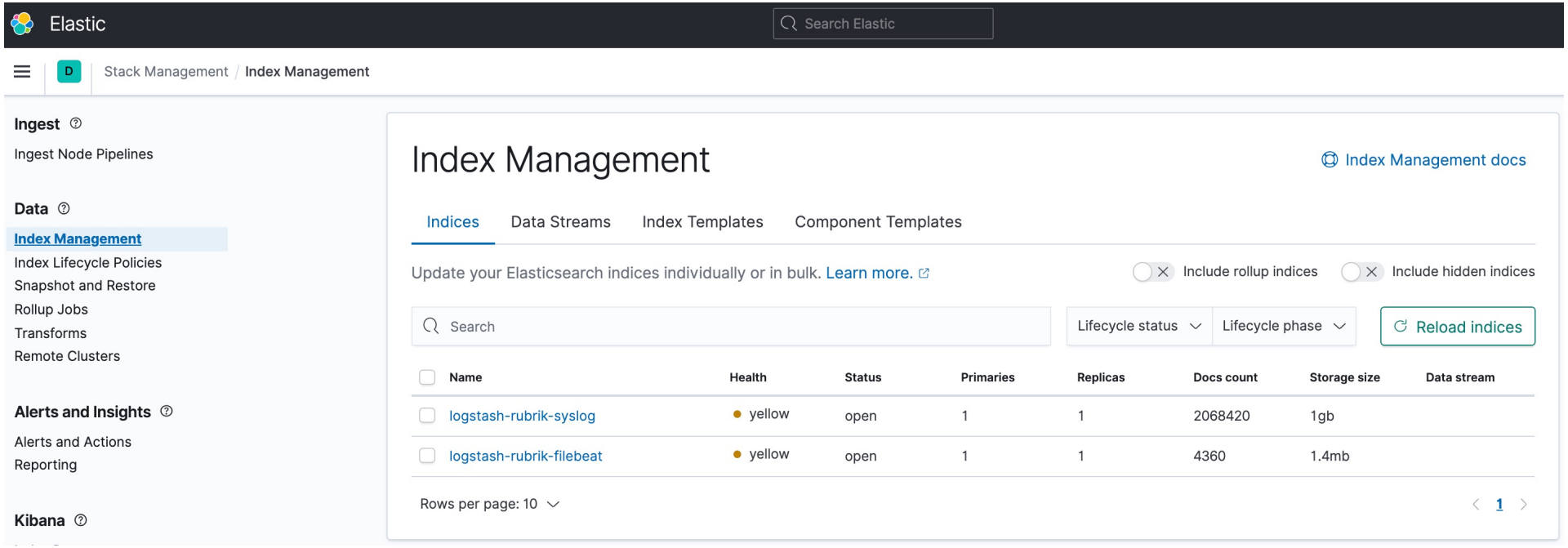
20-logstash-rubrik-filebeat.conf

```
input {
  beats {
    port => 5044
    type => "beats"
  }
}

filter {
  if [type] == "beats" {

    json {
      source => "message"
    }
    mutate {
      rename => [ "lastUpdateTime", "timestamp" ]

    date {
      match => [ "scriptRunTime" , "ISO8601" ]
      target => "@timestamp"
    }
    prune {
      whitelist_names => [ "timestamp", "type", "index", "rubrikClusterName", "rubrikSpaceTotal",
        "rubrikSpaceUsed", "rubrikSpaceAvailable", "rubrikSpaceSnapshot", "rubrikSpaceLiveMount",
        "rubrikSpacePendingSnapshot", "rubrikSpaceCDP", "rubrikSpaceMisc", "rubrikUsedPct",
        "rubrikTotalProtected", "rubrikInCompliance", "rubrikOutCompliance",
        "rubrikPctInCompliance", "rubrikPctOutCompliance", "rubrikComplianceTime",
        "rubrikNodesGood", "rubrikNodesBad", "rubrikNodesTotal"  ]
    }
  }
}

output {
  if [type] == "beats" {
    elasticsearch {
      hosts => ["localhost:9200"]
      index => "logstash-rubrik-filebeat"
    }

    stdout { codec => rubydebug }

  }
}
```

rubrik

# Kibana – Indexes

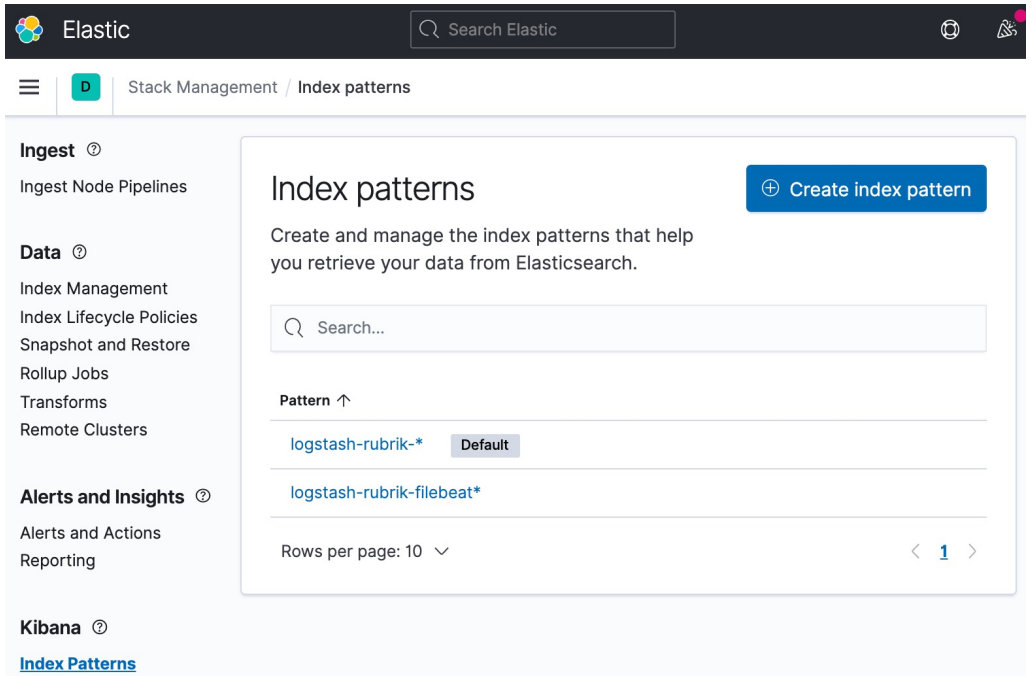


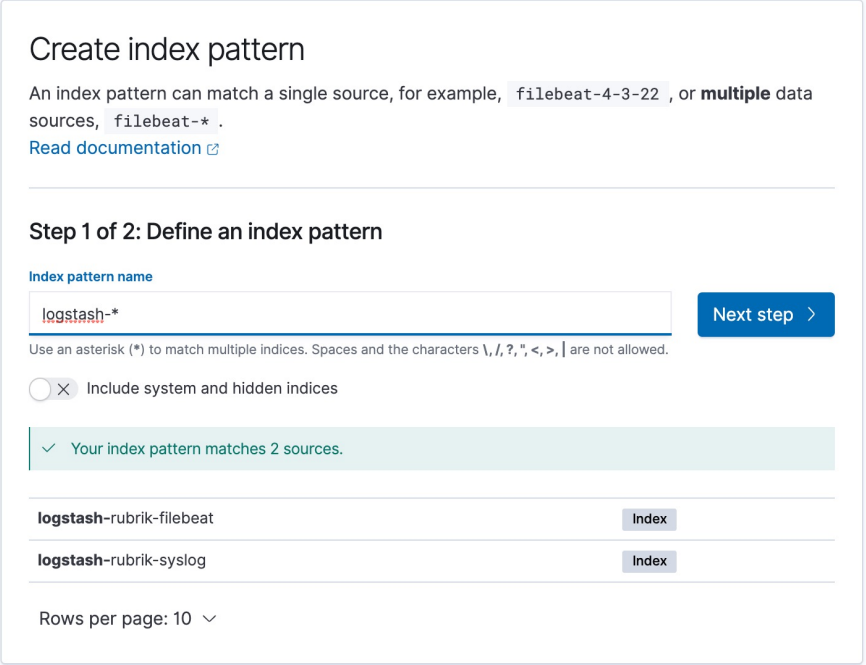

- Data sent to Elasticsearch indexes should start appearing under Stack Management → Index Management
- Log lifecycle and management will need to be configured in order to manage the size of the indexes

rubrik

# Kibana – Index Patterns



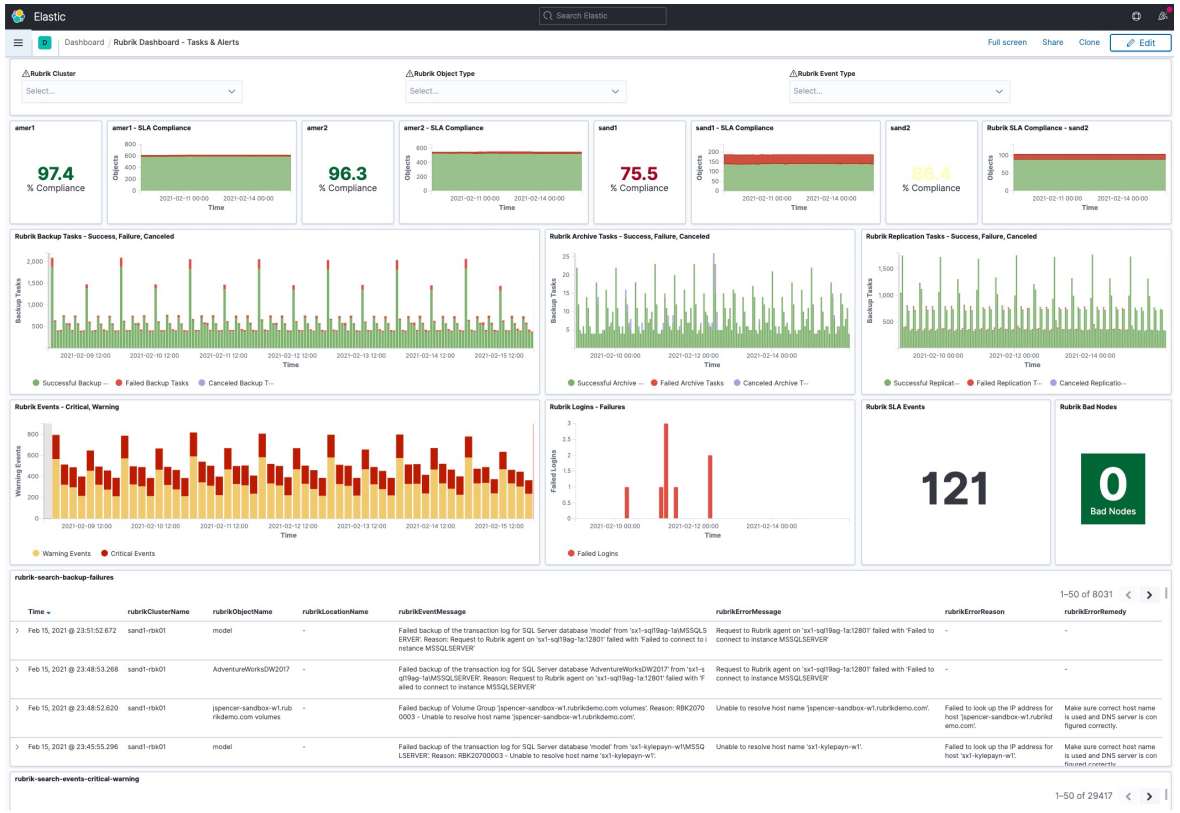



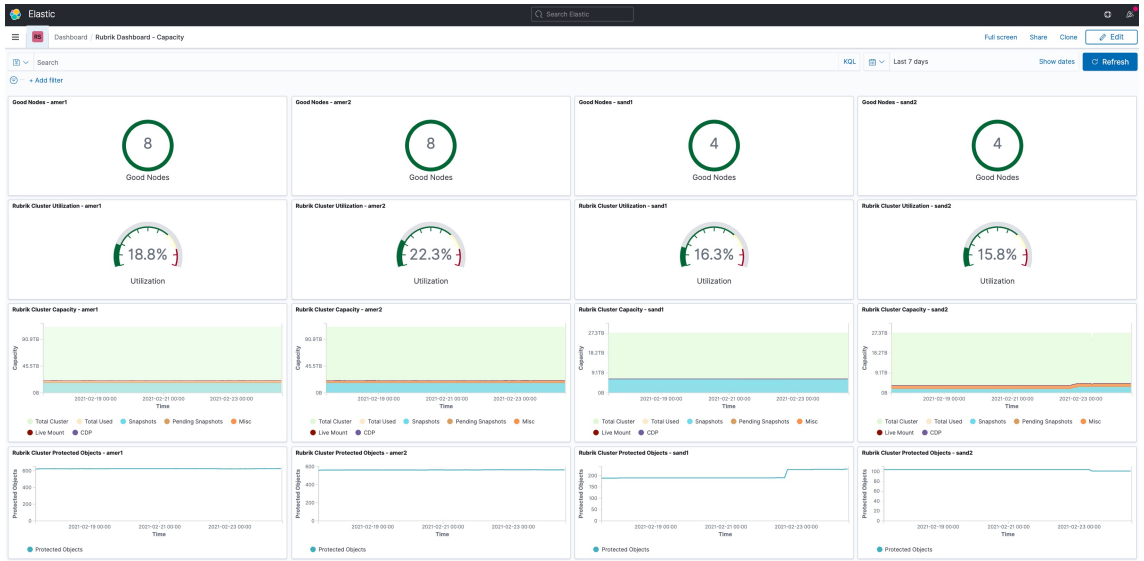

- Create an Index Pattern under Stack Management → Index Patterns
- One or more Index Patterns must be created in order to reference data stored in the Elasticsearch indexes
- When you create a pattern name it will let you know all indexes that match that pattern

rubrik

# Kibana – Dashboards




- Two example dashboards are created in Kibana
- For all dashboards you can change the time range and provide additional search filters to update the results on the dashboard

rubrik

# Dashboard - Tasks & Alerts

All tiles were created from Syslog events except for SLA Compliance which uses metrics polled from the REST API

Charts are preferable when possible for trend analysis

- **Drop-down Filter** – Filter the data shown by cluster, object type, and event type
- **Number + Chart – SLA Compliance** – For each cluster display as a % and chart
- **Chart – Backup Tasks** – Display successful, failed, and cancelled backup tasks
- **Chart – Archive Tasks** – Display successful, failed, and cancelled archive tasks
- **Chart – Replication Tasks** – Display successful, failed, and cancelled replication tasks
- **Chart – Events (Critical, Warning)** – Display events that are Warning or Critical level
- **Chart – Login Failures** – Display number of login failures
- **Number – SLA Failures** – Display a # of SLA type events (ie changes) over the current time range
- **Number – Bad Nodes** – Display the number of bad nodes currently seen against a colored background
- **Event Log – Backup Failures** – List failed backup events
- **Event Log – Critical, Warning** – List events that are Warning or Critical level
- **Event Log – SLA** – List events that modified SLA domains
- **Event Log – Failed Logins** – List events of failed logins

rubrik

# Dashboard - Capacity

All tiles were created using REST API metrics

Aggregate metrics are hard to create and do not make sense so separate tiles are created for each cluster

- **Number + Donut – Good Nodes** – Display the number of good nodes along with a colored, filled donut of the number of good nodes
- **Number + Gauge – Cluster Utilization** – Show current cluster utilization within a colored gauge that goes up to 100%
- **Chart – Cluster Capacity** – Display the cluster storage utilization by TB over time with capacity breakouts
  - It can be hard to hover over this chart – “Inspect” the chart to see the actual data values
- **Chart – Protected Objects** – Display number of protected objects over time

rubrik

Don’t Backup.

**Go Forward.**

rubrik