SETNS.RUN/FALCO-LIVESTREAM-YOUTUBE TWITCH.TV/KRISNOVA

THURSDAY | OCTOBER 10TH, 2019 | 1PM - 3PM (PT)

LIVESTREAM WITH KRIS NOVA OF FALCO

Join Kris Nova of Falco for a casual live-hacking to see OPA and Gatekeeper in action, along with an evaluation of different tools in the space.

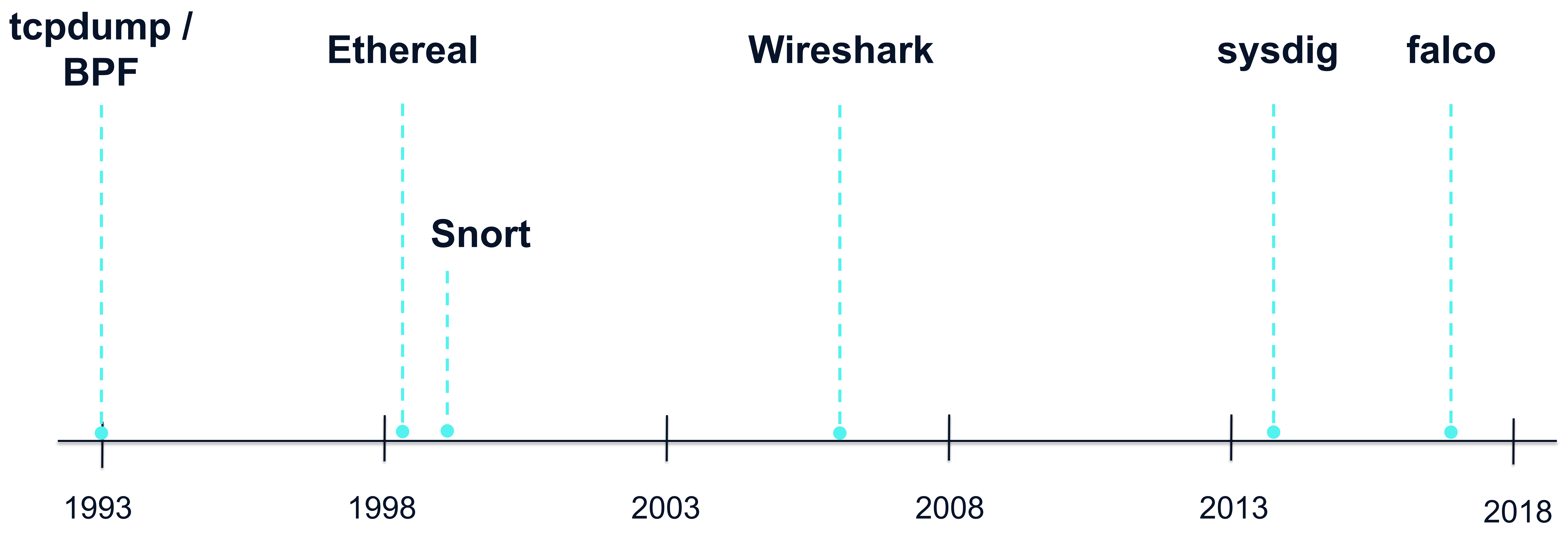
Sysdig Falco

Falco

Container runtime security with Falco.

Chris Kranz, Sysdig
@ckranz

A bit of history.



@ckranz

Snort : Wireshark = falco : sysdig

Falco.

A behavioral activity monitor

- Detects suspicious activity defined by a set of rules
- Uses sysdig's flexible and powerful filtering expressions

With full support for containers/orchestration

- Utilises sysdig's container & orchestrator support

And flexible notification methods

- Alert to files, standard output, syslog, programs

Open Source

- Anyone can contribute rules or improvements
- CNCF sandbox project

Falco: a CNCF sandbox project.

Runtime security for cloud native platforms

- Detect abnormal behavior in applications, containers, and hosts.
- Audit orchestrator activity.

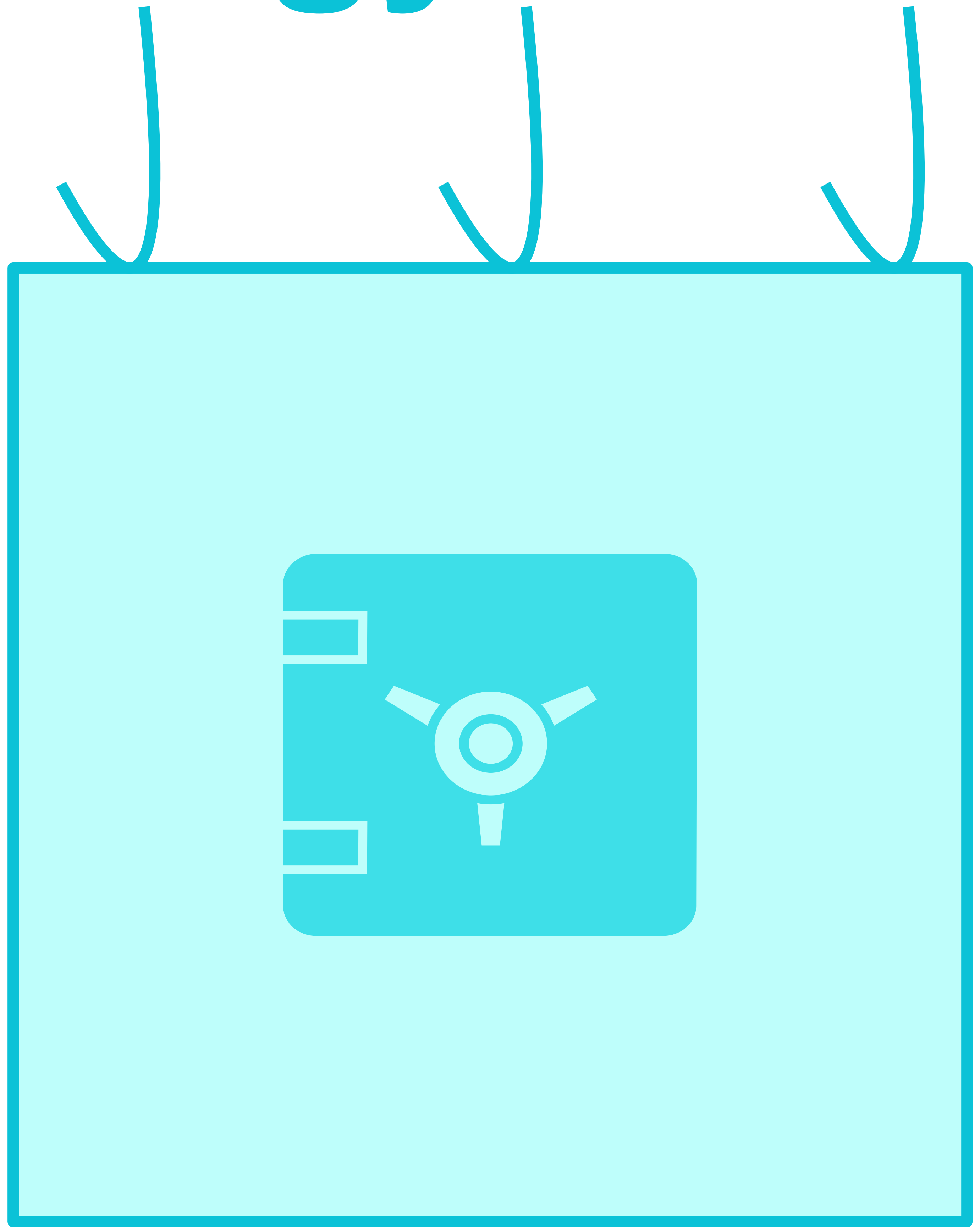
Cloud Native Computing Foundation (CNCF)

- Sandbox level project
- sysdig.com/blog/falco-cncf-sandbox

Home security analogy.

Prevent Intrusion

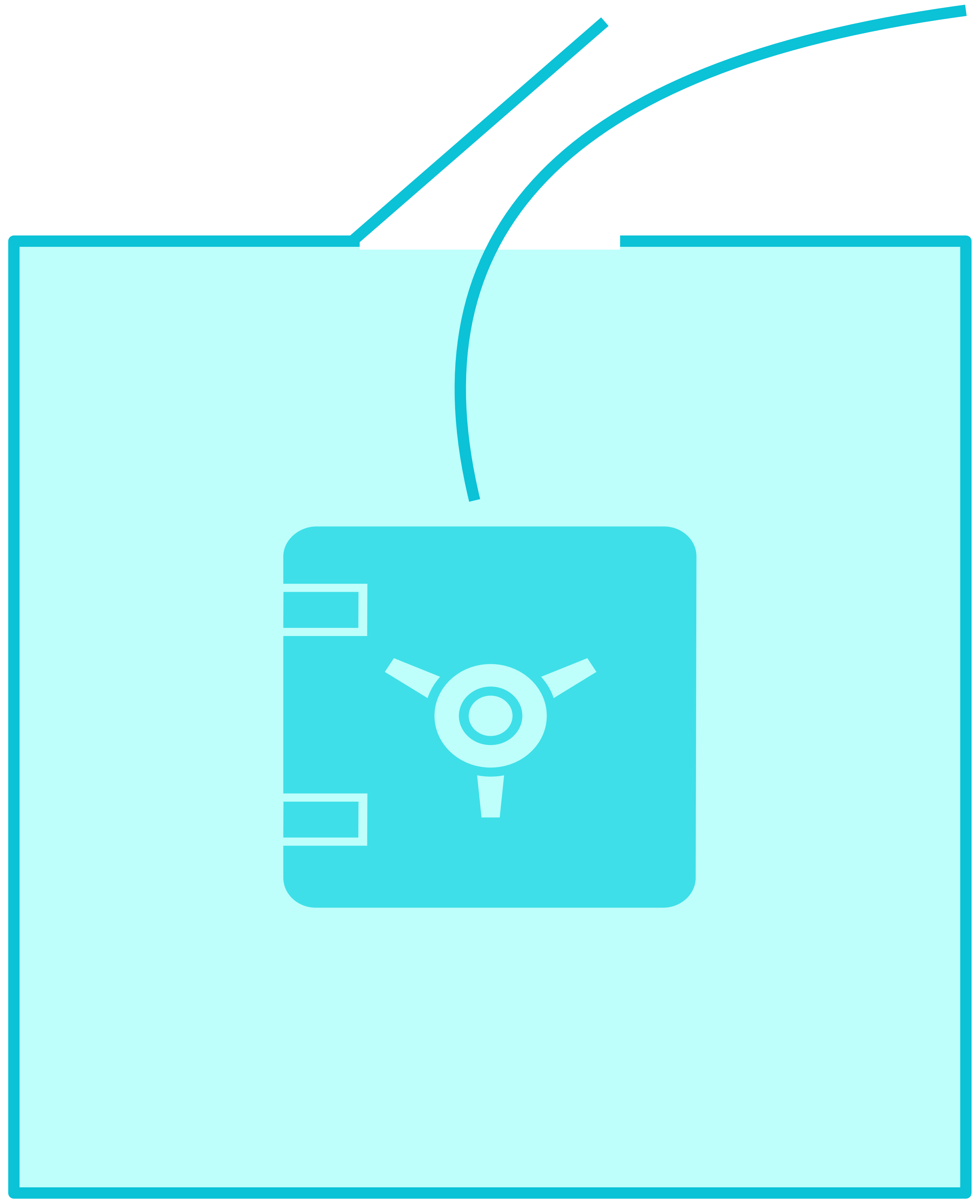
- Door locks
- Window sensors
- Bars on ground floor windows
- Exterior camera



Home security analogy.

Detect Intrusion

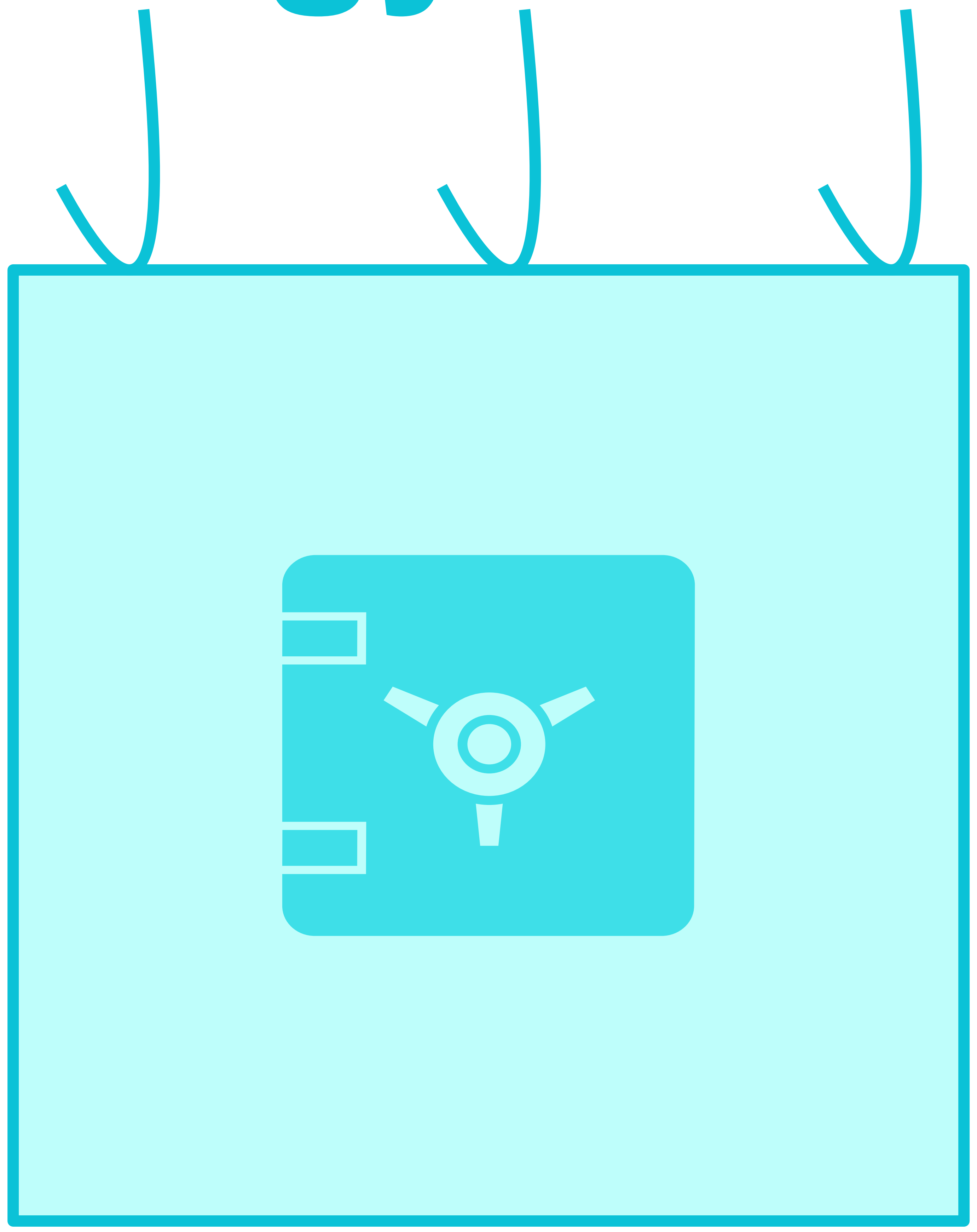
- Motion sensors
- Interior cameras



Home security analogy.

Prevent Intrusion

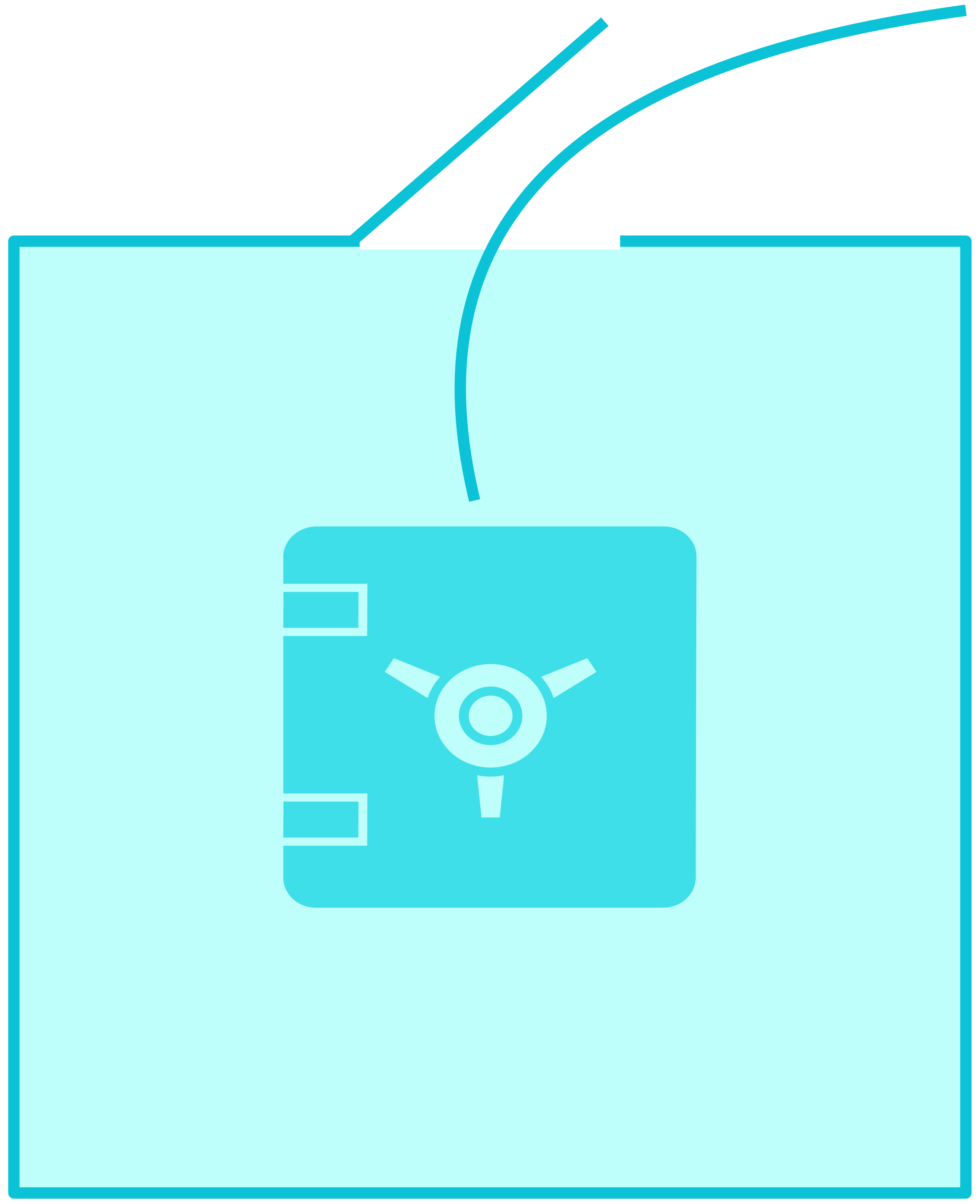
- Passwords
- Two-Factor Authentication
- Container Image Scanning
- Admission Controllers
- Network Policy



Home security analogy.

Detect Intrusion

- Kubernetes Audit Logging
- System Call Instrumentation
- Both methods essential for full protection



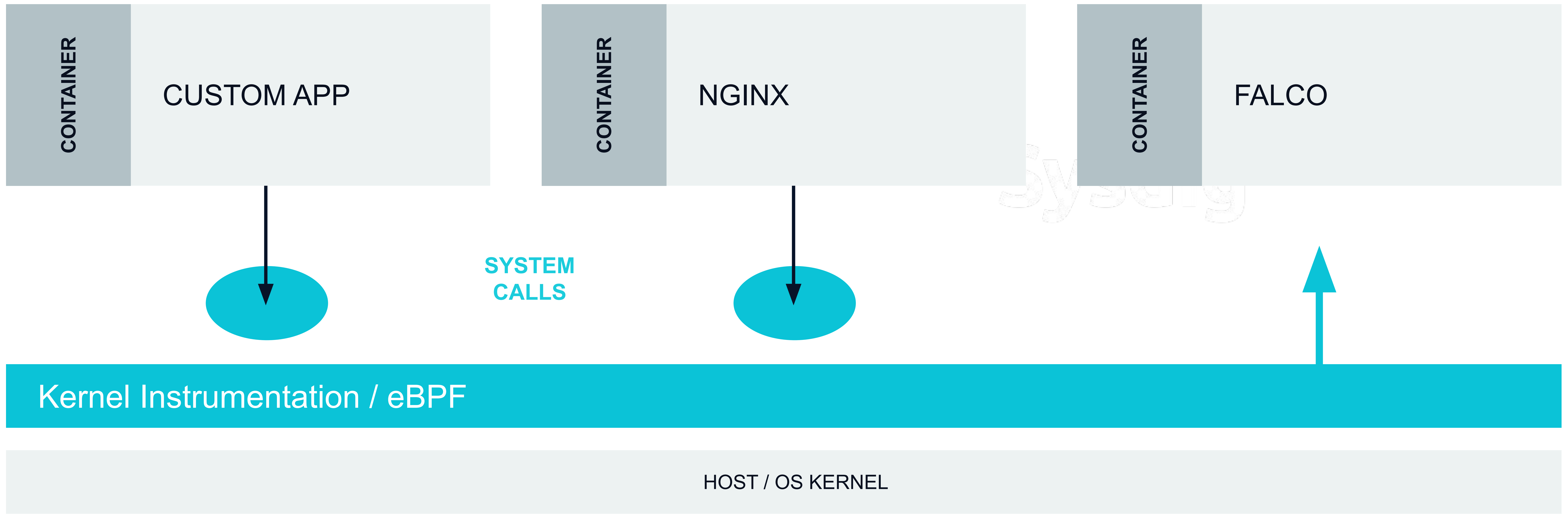
@ckranz

Anomaly detection.

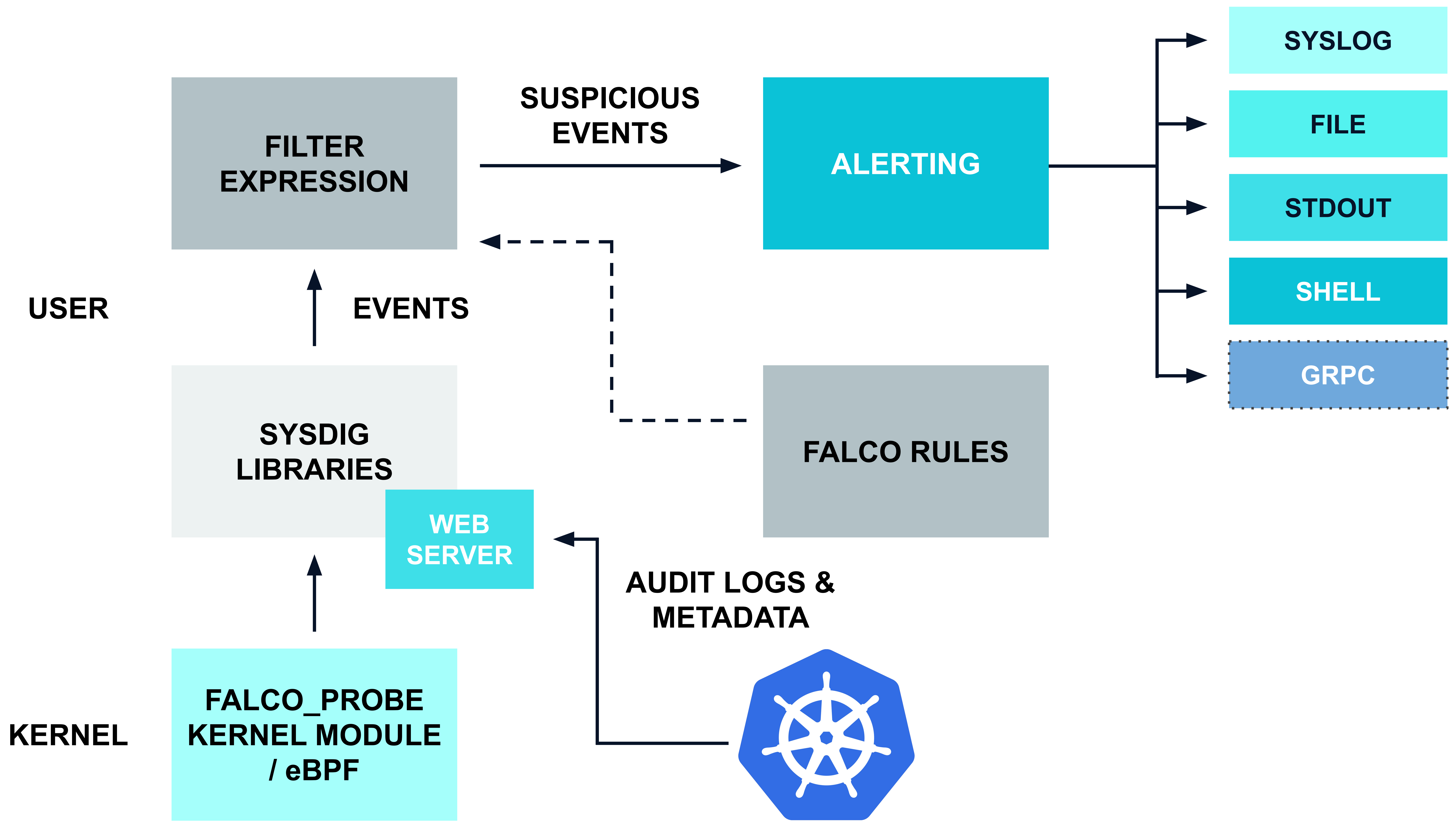
- Containers are **isolated** processes.
- Processes are **scoped** as to what's expected.
- Container images are **immutable, runtime environments** often **aren't.**
- How do you **detect abnormal behavior?**

Architecture.

Instrumentation.



Falco architecture.



@ckranz

Falco Rules

Falco rules.

yaml file containing Macros, Lists, and Rules

```
- list: bin_dirs
  items: [/bin, /sbin, /usr/bin, /usr/sbin]
- macro: bin_dir
  condition: fd.directory in (bin_dirs)
- rule: write_binary_dir
  desc: an attempt to write to any file below a set of binary directories
  condition: bin_dir and evt.dir = < and open_write and not
package_mgmt_procs
  output: "File below a known binary directory opened for writing
    (user=%user.name command=%proc.cmdline file=%fd.name)"
  priority: WARNING
```

Falco rules.

Macros

- **name:** text to use in later rules
- **condition:** filter expression snippet

Lists

- **name:** text to use later
- **items:** list of items

Rules

- **name:** used to identify rule
- **desc:** description of rule
- **condition:** filter expression, can contain macro references
- **output:** message to emit when rule triggers, can contain formatted info from event
- **priority:** severity of rule (WARNING, INFO, etc.)

Falco rules.

Filtering Expressions

- Use the same format as sysdig
- Full container, Kubernetes, Mesos, Docker Swarm support

Rule Execution Order

- Falco rules are combined into one giant filtering expression, joined by ors
- Each rule must contain at least one evt.type expression
- i.e. evt.type=open and …
- Allows for very fast filtering of events.

Conditions and Sysdig Filter Expressions.

Based on “Field Classes”. Supported classes include:

fd - File Descriptors	**container -** Container metadata
process - Processes	**fdlist -** FD poll events
evt - System events	**k8s -** Kubernetes metadata
user - Users	**ka -** Kubernetes Audit Logs
group - Groups	**mesos -** Mesos metadata
syslog - Syslog messages	

Quick examples.

A shell is run in a container	container.id != host and proc.name = bash
Overwrite system binaries	fd.directory in (/bin, /sbin, /usr/bin, /usr/sbin) and write
Container namespace change	evt.type = setns and not proc.name in (docker, sysdig)
Non-device files written in /dev	(evt.type = create or evt.arg.flags contains O_CREAT) and proc.name != blkid and fd.directory = /dev and fd.name != /dev/null
Process tries to access camera	evt.type = open and fd.name = /dev/video0 and not proc.name in (skype, webex)

@ckranz

Alerts + outputs.

Sending alerts

- Events matching filter expression result in alerts
- Rule's output field used to format event into alert message
- Falco configuration used to control where alert message is sent

Any combination of..

- Syslog
- File
- Standard Output
- Shell (e.g. mail -s "Falco Notification" someone@example.com)

A custom Falco rule.

```
- rule: Node Container Runs Node
  desc: Detect a process that’s not node started in a Node container.
  condition: evt.type=execve and container.image startswith node and proc.name!=node
  output: Node container started other process (user=%user.name
                 command=%proc.cmdline %container.info)
  priority: INFO
  tags: [container, apps]
```

Something is executing a program

In a container based on the Node image

And the process name isn’t node

Kubernetes audit log events.

- New in K8s v1.11
- Provides chronological set of records documenting changes to cluster
- Each record is a JSON object
- Audit policy controls which events are included in event log
- Log backend controls where events are sent
  - Log file
  - Webhook
  - AuditSink (alpha as of 1.13)

@ckranz

K8s audit events.

```
{
    "kind": "Event",
    "timestamp": "2018-10-26T13:00:25Z",
    "stage": "ResponseComplete",
    "verb": "delete",
    "requestURI": "/api/v1/namespaces/foo",
    "user": { "username": "minikube-user" },
    "responseStatus": { "code": 200 },
    "objectRef": { "resource": “namespaces", "namespace": “foo” },
    "level": "Request",
    "auditID": "693f4726-2430-450a-83e1-123c050fde98",
    "annotations": { "authorization.k8s.io/decision": "allow" }
}
```

Supporting Kubernetes audit log events.

- Create a new “Generic Event” interface
  - Event time, ability to extract values using fields
- Create a K8s Audit Event object
  - Event data is json object, stored in event
- Define new fields to extract values from K8s Audit Events
  - Uses Json Pointers to extract values
- Each Falco Rule now has a source
  - Default “syscall”, “k8s_audit” for K8s Audit Events

Kubernetes audit log fields.

- `jevt.value[<json_pointer>]`
  - Access any field from json object
- `jevt.time`
  - Access event timestamp
- `ka.verb, ka.uri, ka.user.name, ka.target.resource, …`
  - Access specific values from object
  - Implemented as macros:
    - ka.verb -> jevt.value[/verb]
    - ka.target.resource -> jevt.value[/objectRef/resource]
  - Full list: `falco –list=k8s_audit`

K8s audit log rule example.

```
- macro: contains_private_credentials
  condition: >
    (ka.req.configmap.obj contains "aws_access_key_id" or
     ka.req.configmap.obj contains "aws_s3_access_key_id" or
     ka.req.configmap.obj contains “password")

- macro: configmap
  condition: ka.target.resource=configmaps

- macro: modify
  condition: (ka.verb in (create,update,patch))

- rule: Create/Modify Configmap With Private Credentials
  desc: Detect creating/modifying a configmap containing a private credential
    (aws key, password, etc.)
  condition: configmap and modify and contains_private_credentials
  output: K8s configmap with private credential (user=%ka.user.name
          verb=%ka.verb name=%ka.req.configmap.name
          configmap=%ka.req.configmap.name config=%ka.req.configmap.obj)
  priority: WARNING
  source: k8s_audit
  tags: [k8s]
```

@ckranz

Extending rules/macros/lists.

Can combine rulesets to extend/modify behavior

```
falco -r <rules-file> -r <additional-rules-file> …
```

```
- macro: my macro
  condition: …
- list: my list
  items: …
- rule: my rule
  desc: …
  condition: …
  output: …
```

+

```
- macro: another macro
  condition: …
- list: another list
  items: …
- rule: another rule
  desc: …
  condition: …
  output: …
```

Installing and Integrations

Installing Falco.

- **Debian Package**
  - apt-get -y install falco
- **Redhat Package**
  - yum -y install falco
- **Installation Script**
  - curl -s s3.amazonaws.com/download.draios.com/stable/install-falco | sudo bash
- **Docker container**
  - docker pull sysdig/falco
- **Full instructions**
  - github.com/draios/falco/wiki/How-to-Install-Falco-for-Linux

Installing Falco on kubernetes.

- **Use Helm**
  - $ helm install --name sysdig-falco-1 stable/falco
  - https://sysdig.com/blog/falco-helm-chart/

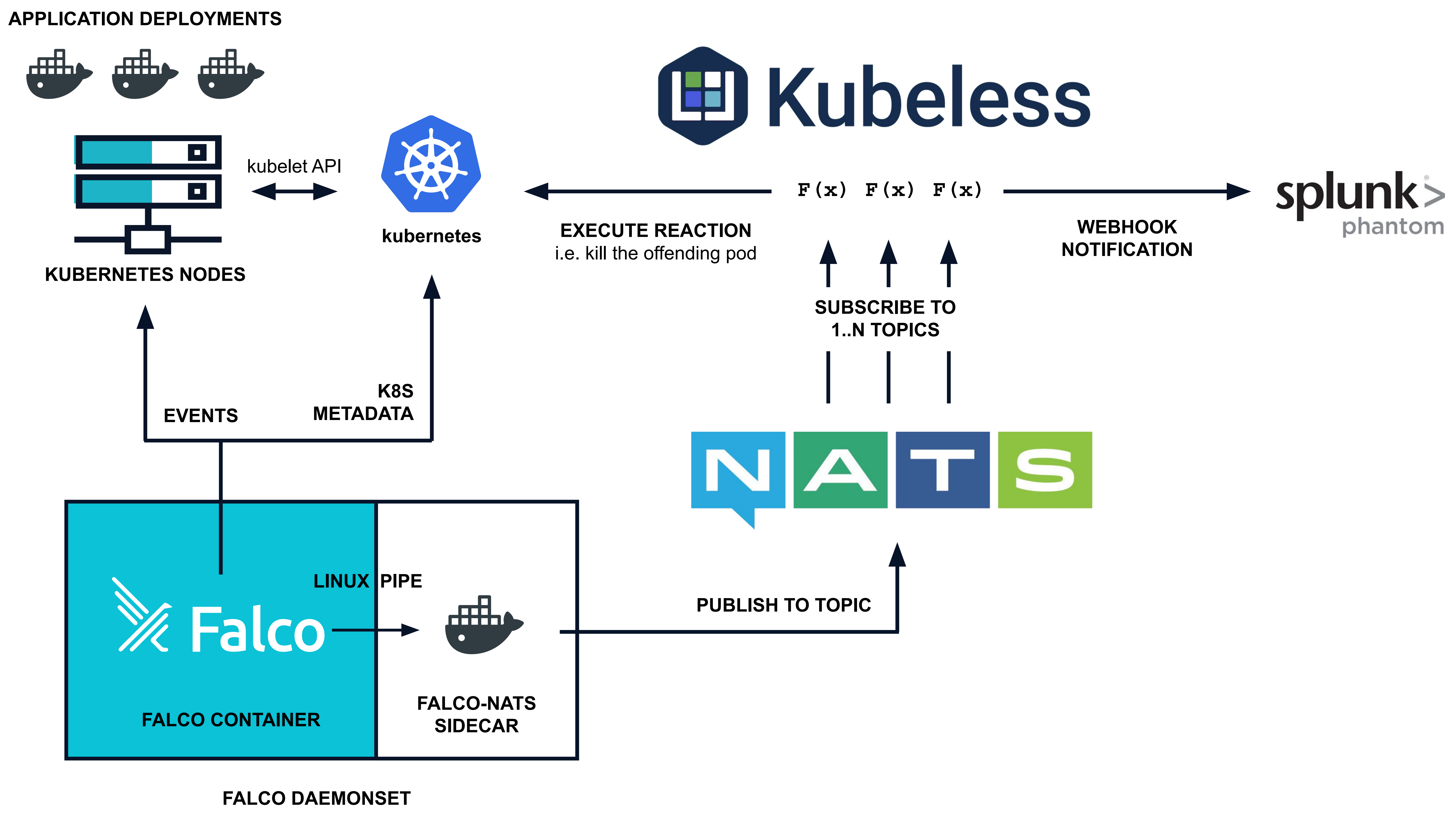
- **Install Falco as Kubernetes Daemonset**
  - https://github.com/draios/falco/tree/dev/examples/k8s-using-daemonset
  - Configuration stored in Kubernetes ConfigMaps
  - Conditions in a Falco Rule can leverage Kubernetes metadata to trigger events
  - Falco events can include Kubernetes metadata to give notification context:
    - name, id, labels for Pods, ReplicationController, Service, Namespace, ReplicaSet, and Deployment

How can you use Falco?

Response engine & security playbooks.

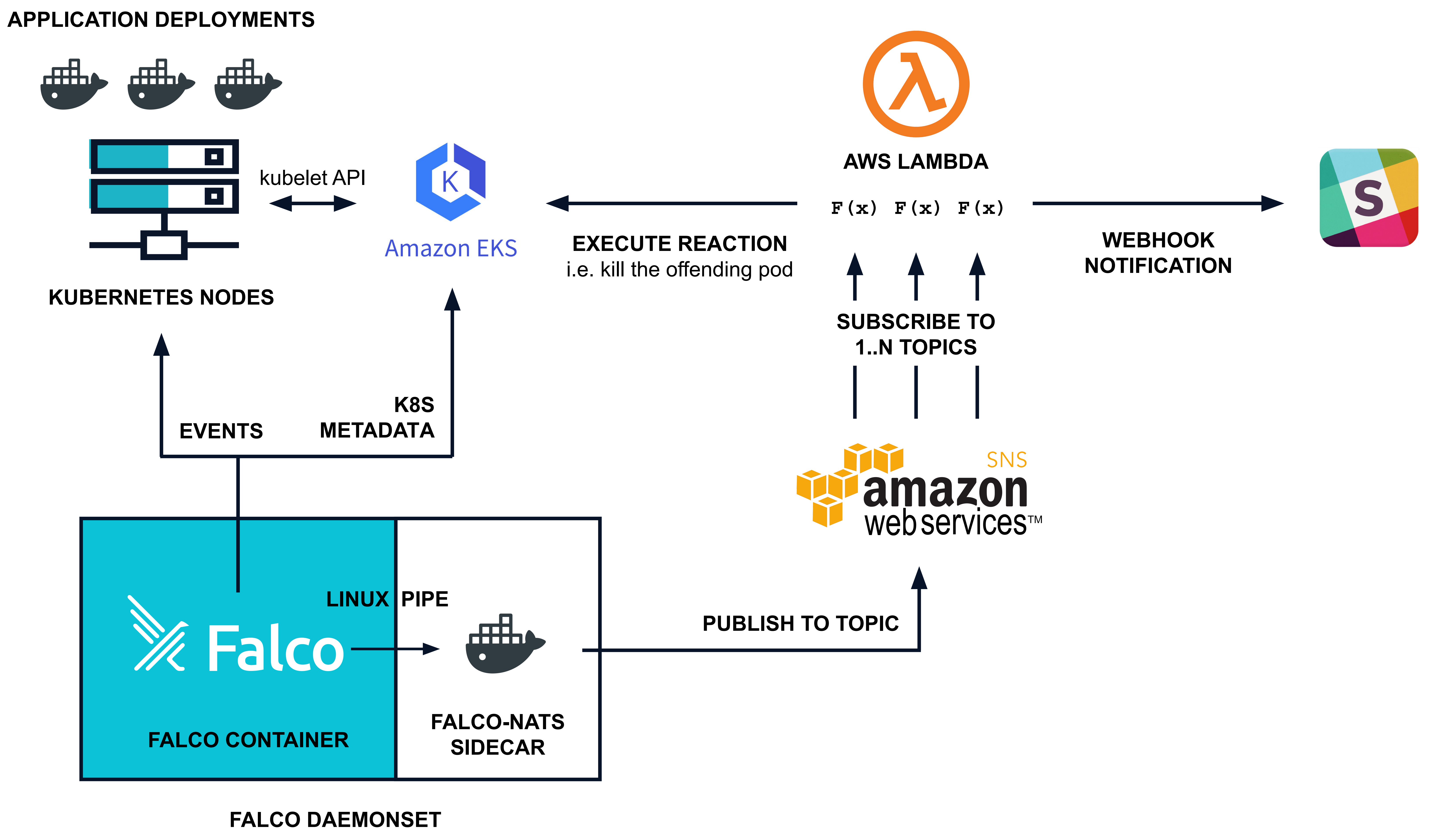
- **Detect abnormal events** with Falco
- **Publish alerts** to Pub/Sub service (NATS.io)
- Subscribers can **subscribe to various FALCO topics** to receive alerts:
  - FALCO.* - All alerts
  - FALCO.Notice - Alerts of priority “Notice” only
  - FALCO.Critical - Alerts of priority “Critical” only
- Subscribers can **take action** on alerts:
  - Kill offending Pod
  - Taint Nodes to prevent scheduling
  - Isolate Pod with Networking Policy
  - Send notification via Slack

Response engine & security playbooks.



https://sysdig.com/blog/container-security-orchestration-falco-splunk-phantom/

Response engine & security playbooks.



https://aws.amazon.com/blogs/opensource/securing-amazon-eks-lambda-falco/

@ckranz

Response engine & security playbooks.

Falco

Detects abnormal event, Publishes alert to NATS

NATS

Subscribers receive Falco Alert through NATS Server

Kubeless

Kubeless receives Falco Alert, firing a function to delete the offending Kubernetes Pod

https://sysdig.com/blog/oss-container-security-runtime/

Functions for operations.

- Easily write simple functions to react to security events
- Multiple subscribers can take multiple actions
  - One function to delete a pod
  - One function to notify teams
  - One function to log events
- Small, reusable components

SIEM with EFK.

- Security Information and Event Management
  - Collect security events
  - Easily allow reporting and correlation of events across various data sources

- Elasticsearch, Fluentd, Kibana
  - Fluentd - Cloud Native log aggregation
  - Elasticsearch - Schema free JSON data store
  - Kibana - powerful data visualization tool for Elasticsearch

- https://sysdig.com/blog/kubernetes-security-logging-fluentd-falco/

SIEM with EFK.

Falco

Detects abnormal event, Publishes alert to stdout

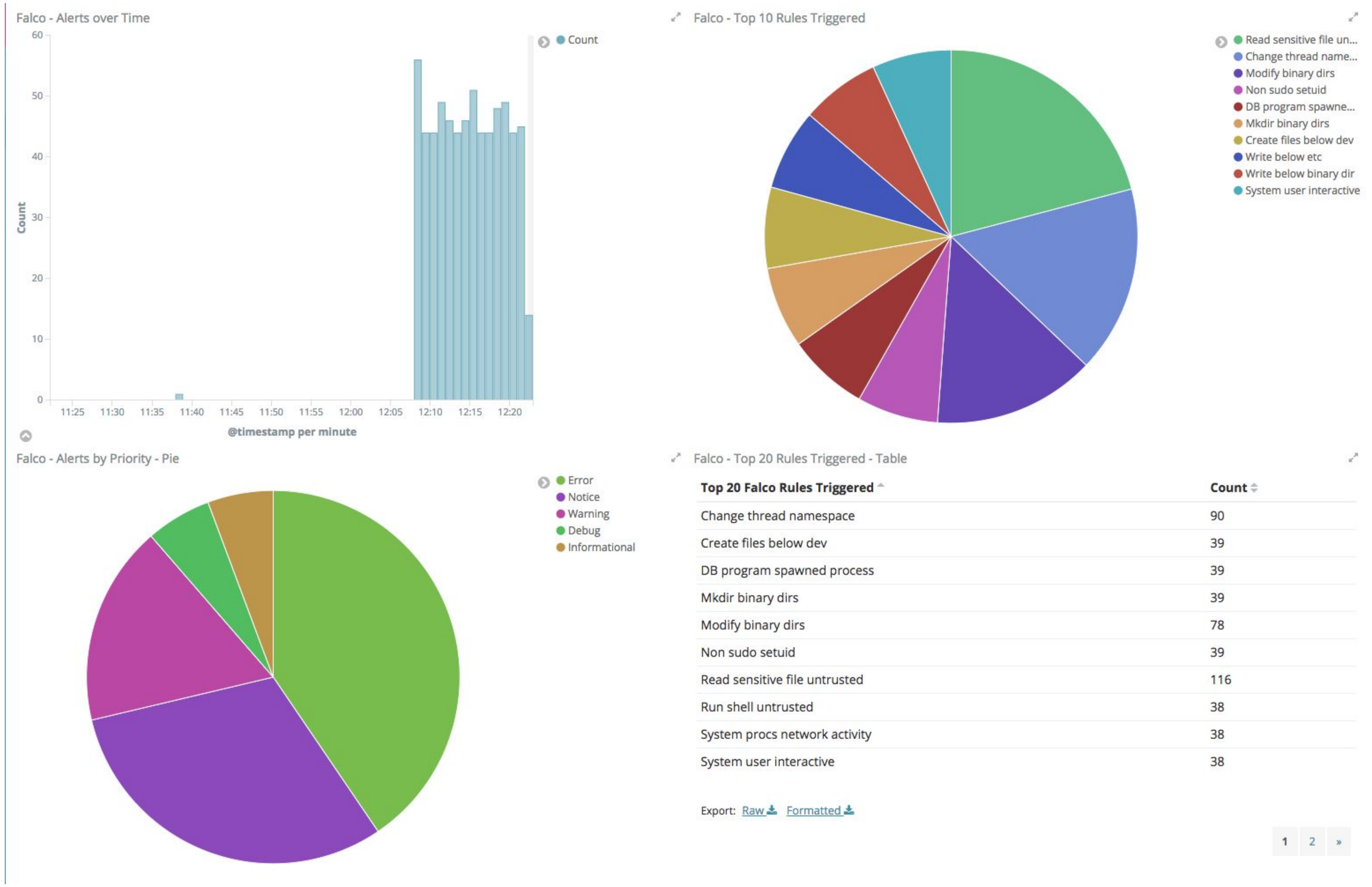
fluentd

Fluentd ships alerts to Elasticsearch

kibana

Kibana dashboards can be used to aggregate, filter, and report on alerts.

SIEM with EFK.



Top 20 Falco Rules Triggered	Count
Change thread namespace	90
Create files below dev	39
DB program spawned process	39
Mkdir binary dirs	39
Modify binary dirs	78
Non sudo setuid	39
Read sensitive file untrusted	116
Run shell untrusted	38
System procs network activity	38
System user interactive	38

Export: Raw Formatted

1 2 »

@ckranz

Roadmap / Help wanted.

Rules Library - Build library or rules profiles for common apps

- https://github.com/falcosecurity/profiles

Event Streams - Increase sources of events beyond system calls.

- Kubernetes Audit Events - Needs additional perf testing
- Application Level Events - CRUD Operations

Output Destinations - Integrate with more alert destinations.

- Messaging Services - SNS, Google Pub/Sub, Kafka
- Logging Services - Elasticsearch, Splunk, Stackdriver
- Web services - HTTPs, GRPC

@ckranz

Join the community.

Website

- https://falco.org

Public Slack

- http://slack.sysdig.com/
- https://sysdig.slack.com/messages/falco

Blog

- https://sysdig.com/blog/tag/falco/

Github

- https://github.com/falcosecurity/falco

Documentation

- https://github.com/falcosecurity/falco/wiki

Docker Hub

- https://hub.docker.com/r/falcosecurity/falco/

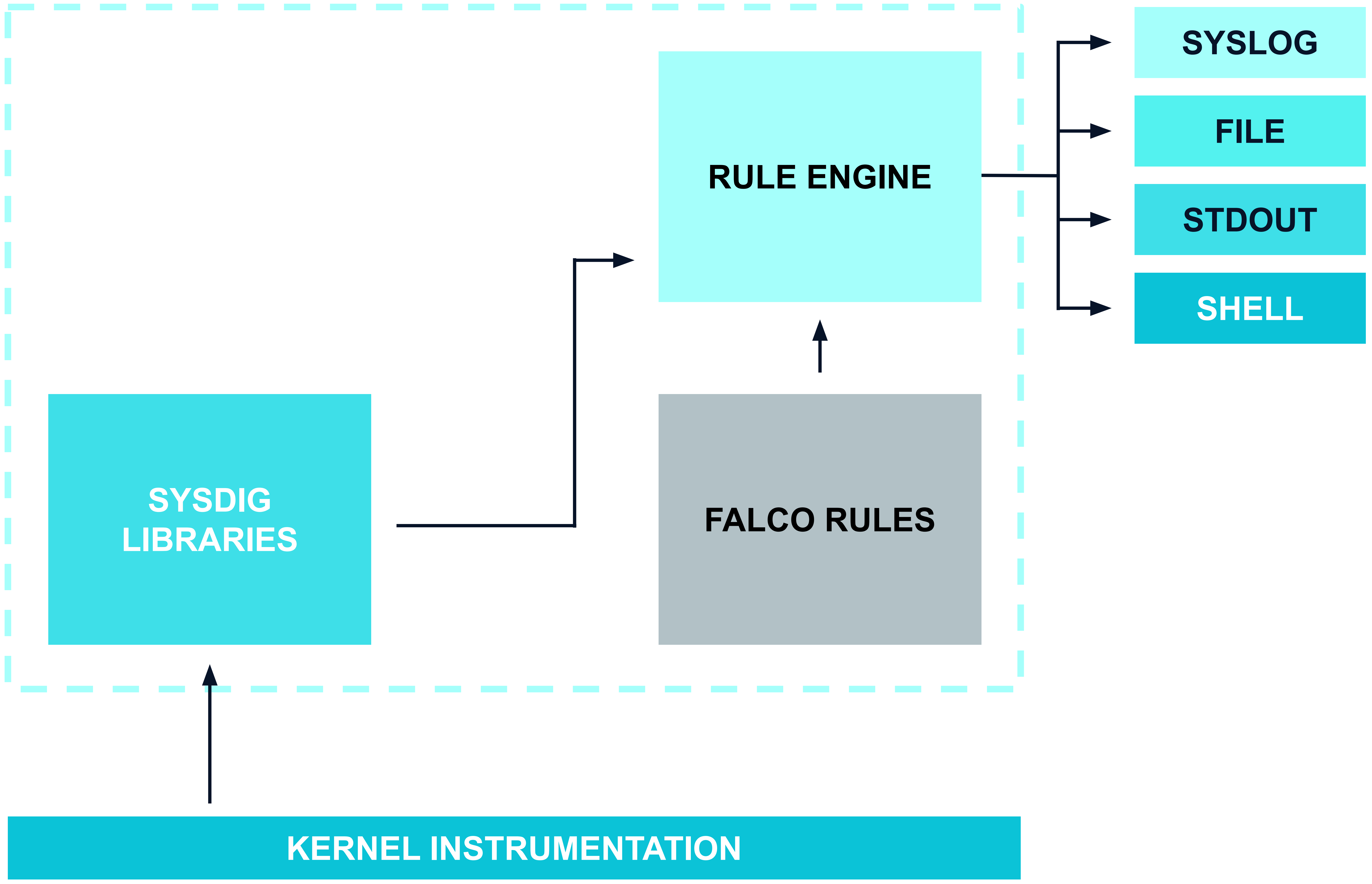
Thank you!

@mfdii

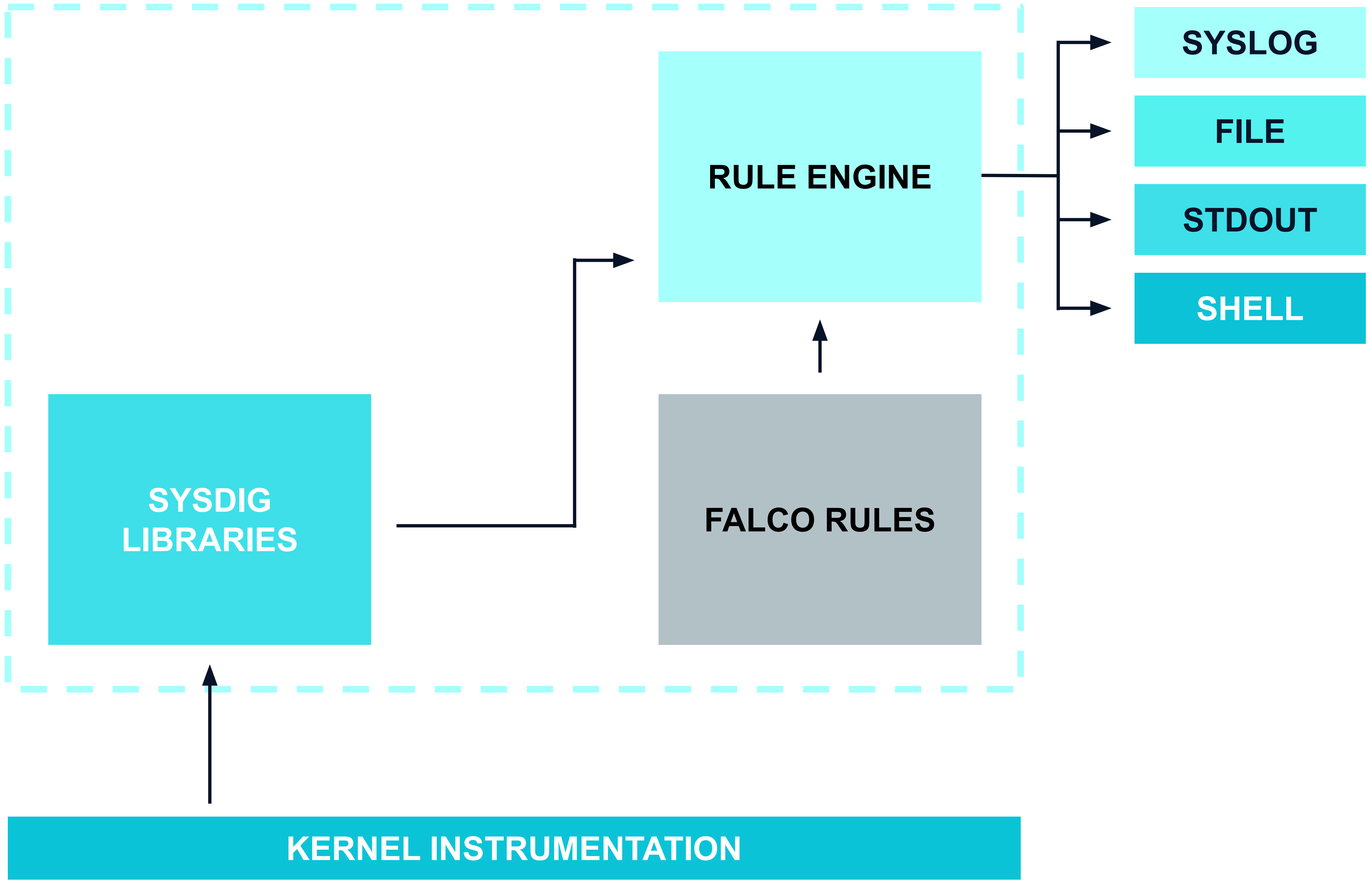
@mfdii

Additional Slides

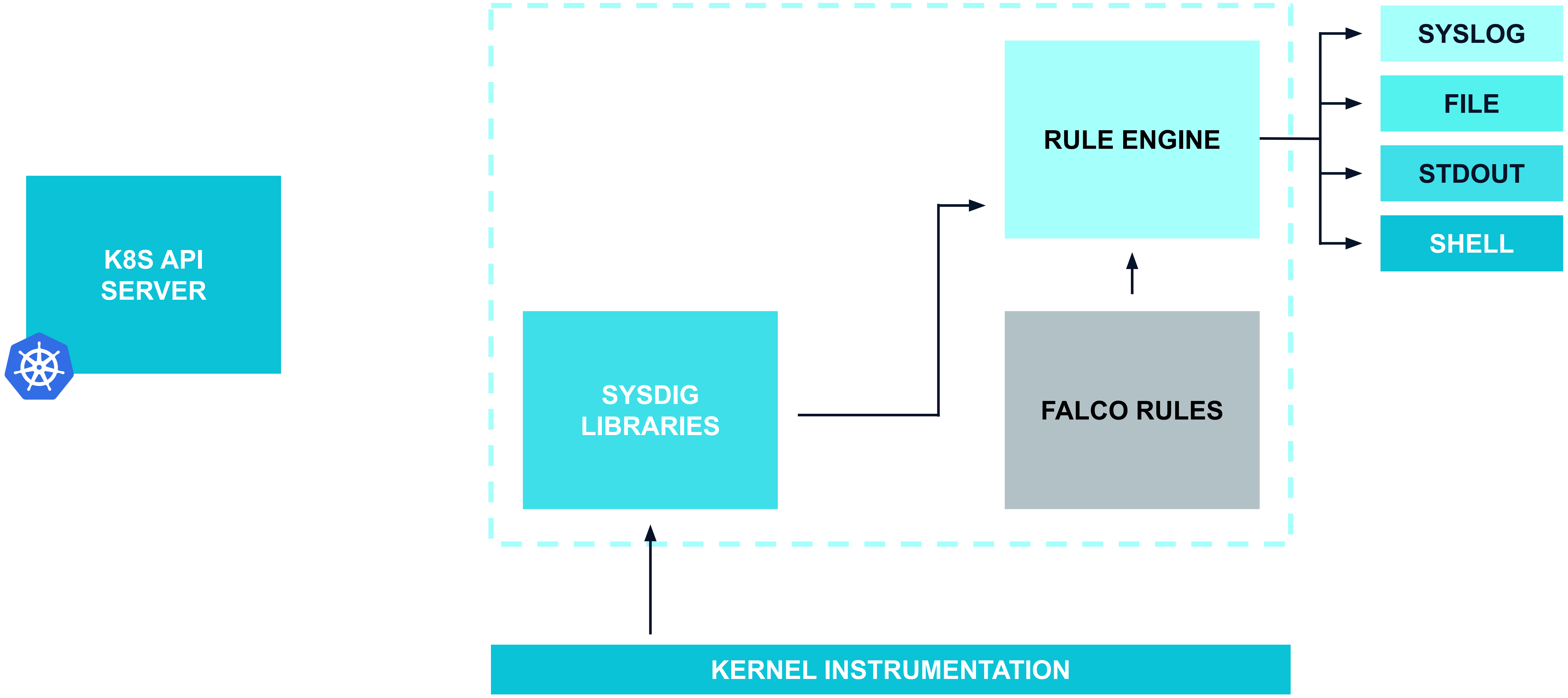
Falco internal architecture.



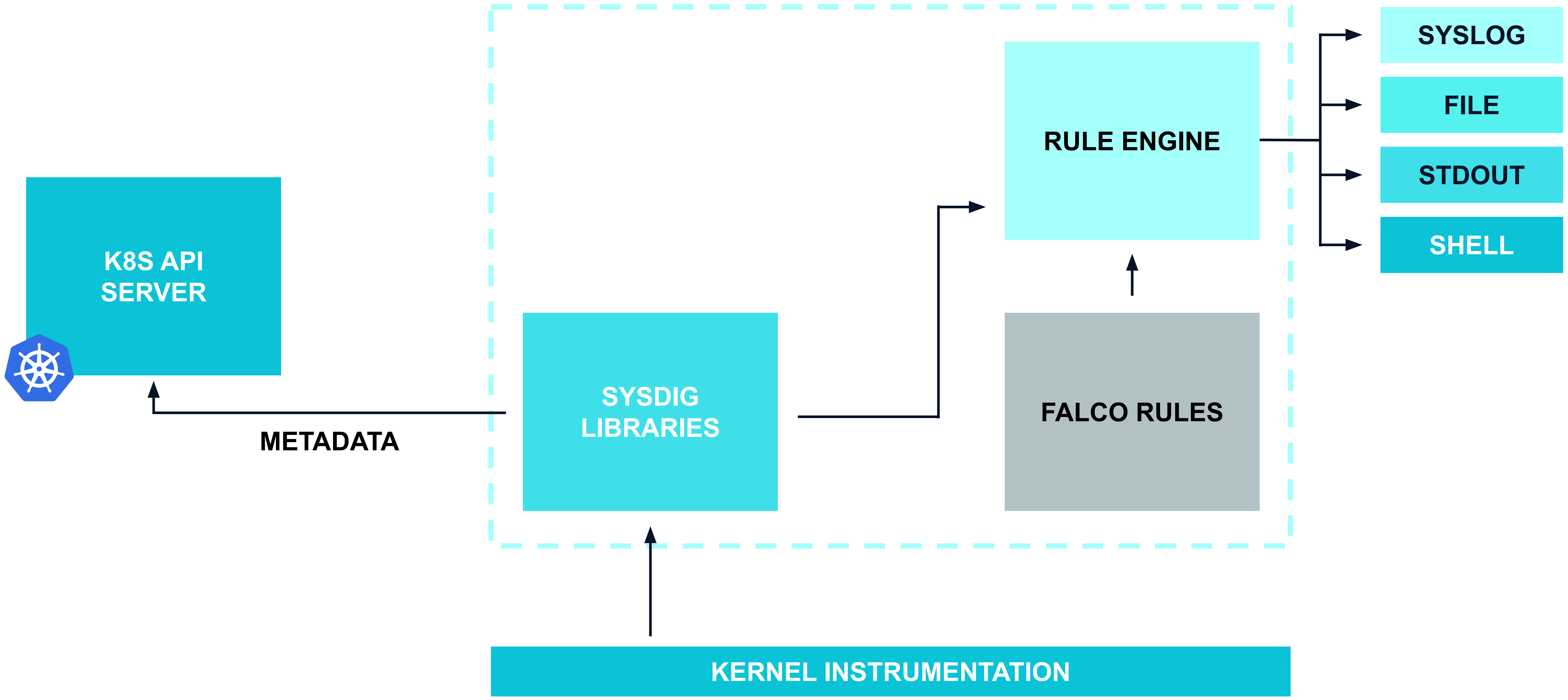
Falco internal architecture.



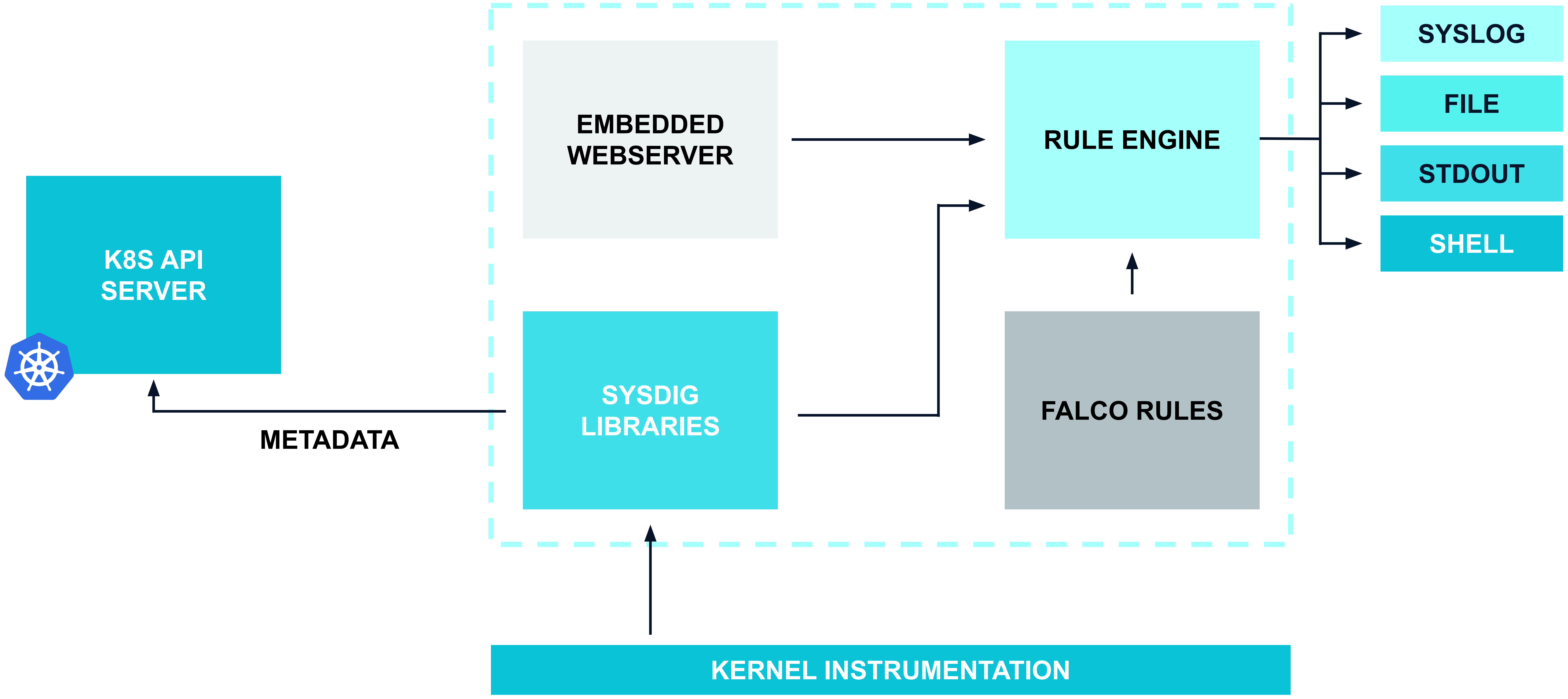
Falco internal architecture.



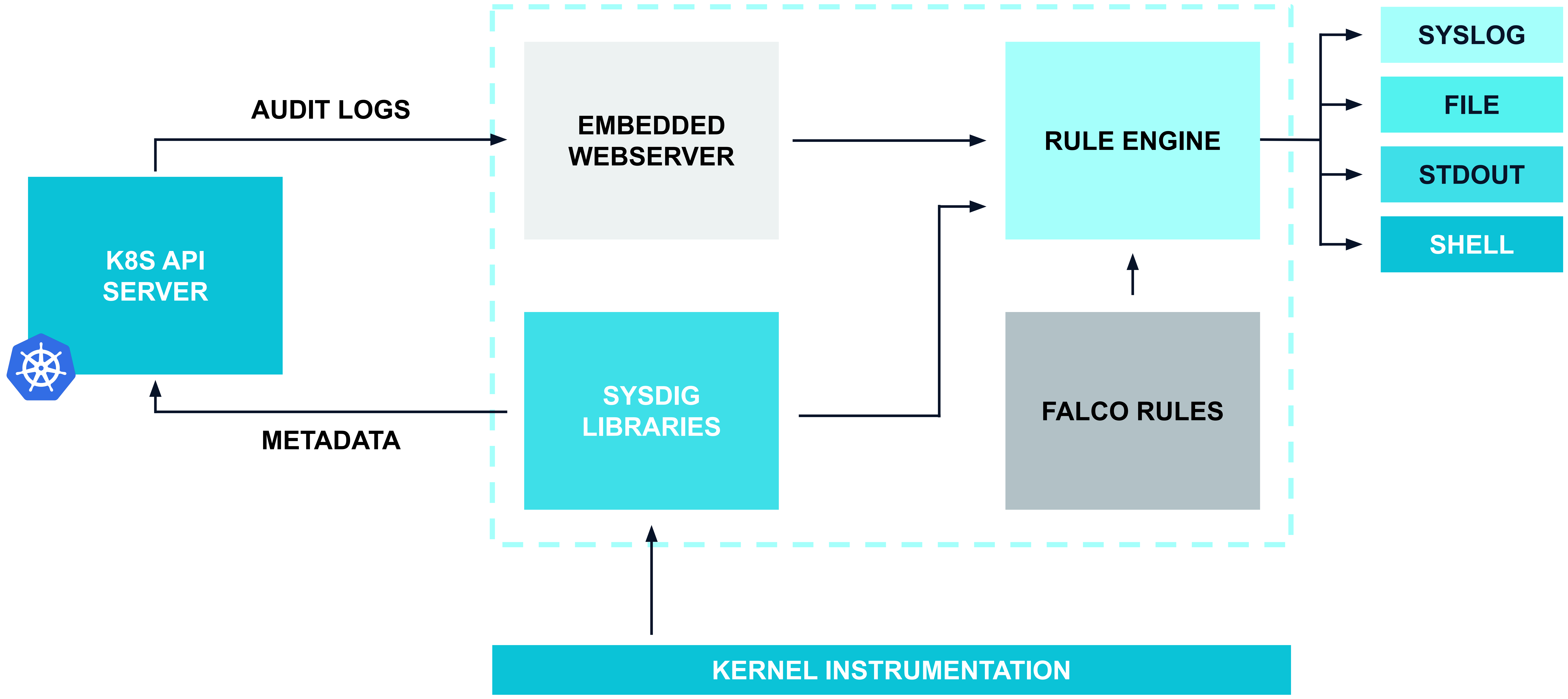
Falco internal architecture.



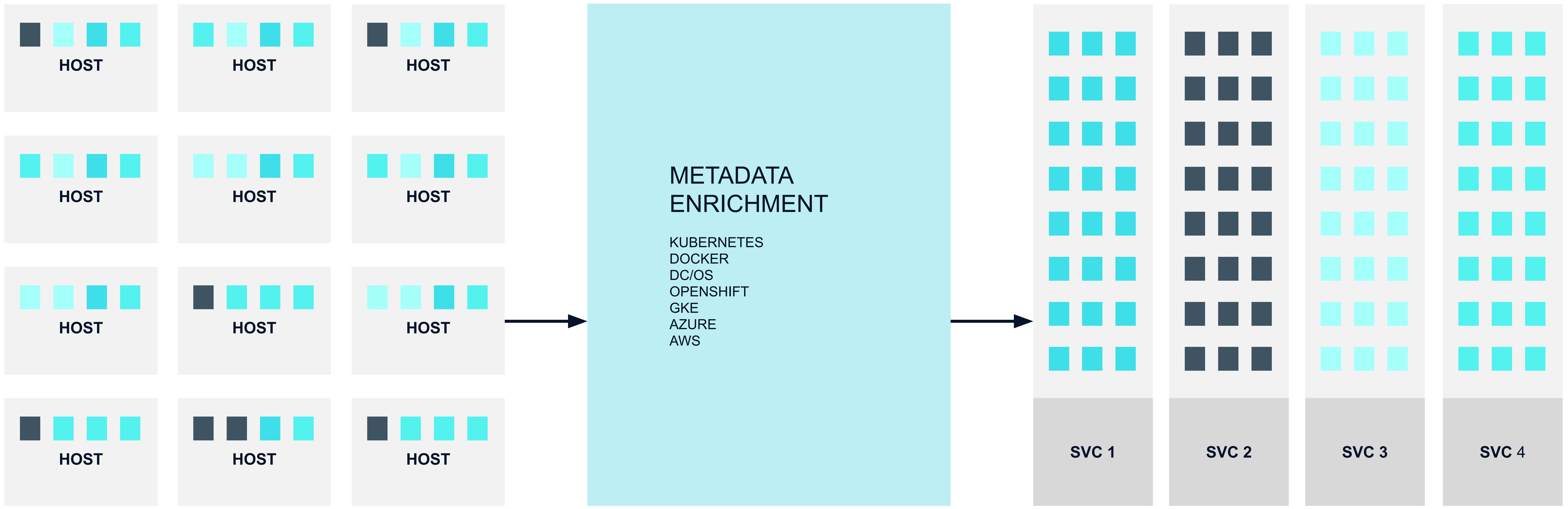
Falco internal architecture.



Falco internal architecture.



Enabling service-oriented intelligence.



@ckranz

Falco Rules

Falco rules.

yaml file containing Macros, Lists, and Rules

```
- list: bin_dirs
  items: [/bin, /sbin, /usr/bin, /usr/sbin]
- macro: bin_dir
  condition: fd.directory in (bin_dirs)
- rule: write_binary_dir
  desc: an attempt to write to any file below a set of binary directories
  condition: bin_dir and evt.dir = < and open_write and not
package_mgmt_procs
  output: "File below a known binary directory opened for writing
    (user=%user.name command=%proc.cmdline file=%fd.name)"
  priority: WARNING
```

@ckranz

Falco rules.

Macros

- **name:** text to use in later rules
- **condition:** filter expression snippet

Lists

- **name:** text to use later
- **items:** list of items

Rules

- **name:** used to identify rule
- **desc:** description of rule
- **condition:** filter expression, can contain macro references
- **output:** message to emit when rule triggers, can contain formatted info from event
- **priority:** severity of rule (WARNING, INFO, etc.)

Falco rules.

Filtering Expressions

- Use the same format as sysdig
- Full container, Kubernetes, Mesos, Docker Swarm support

Rule Execution Order

- Falco rules are combined into one giant filtering expression, joined by ors
- Each rule must contain at least one evt.type expression
- i.e. evt.type=open and …
- Allows for very fast filtering of events.

Conditions and Sysdig Filter Expressions.

Based on “Field Classes”. Supported classes include:

fd - File Descriptors	**container -** Container metadata
process - Processes	**fdlist -** FD poll events
evt - System events	**k8s -** Kubernetes metadata
user - Users	**ka -** Kubernetes Audit Logs
group - Groups	**mesos -** Mesos metadata
syslog - Syslog messages	

Quick examples.

A shell is run in a container	container.id != host and proc.name = bash
Overwrite system binaries	fd.directory in (/bin, /sbin, /usr/bin, /usr/sbin) and write
Container namespace change	evt.type = setns and not proc.name in (docker, sysdig)
Non-device files written in /dev	(evt.type = create or evt.arg.flags contains O_CREAT) and proc.name != blkid and fd.directory = /dev and fd.name != /dev/null
Process tries to access camera	evt.type = open and fd.name = /dev/video0 and not proc.name in (skype, webex)

Alerts and outputs.

Sending Alerts

- Events matching filter expression result in alerts
- Rule's output field used to format event into alert message
- Falco configuration used to control where alert message is sent

Any combination of...

- Syslog
- File
- Standard Output
- Shell (e.g. mail -s "Falco Notification" someone@example.com)

A custom Falco rule.

```
- rule: Node Container Runs Node
  desc: Detect a process that's not node started in a Node container.
  condition: evt.type=execve and container.image startswith node and proc.name!=node
  output: Node container started other process (user=%user.name
                command=%proc.cmdline %container.info)
  priority: INFO
  tags: [container, apps]
```

Something is executing a program

In a container based on the Node image

And the process name isn't node

Kubernetes audit log events.

- New in K8s v1.11
- Provides chronological set of records documenting changes to cluster
- Each record is a JSON object
- Audit policy controls which events are included in event log
- Log backend controls where events are sent
    - Log file
    - Webhook

K8s audit events.

```
{
    "kind": "Event",
    "timestamp": "2018-10-26T13:00:25Z",
    "stage": "ResponseComplete",
    "verb": "delete",
    "requestURI": "/api/v1/namespaces/foo",
    "user": { "username": "minikube-user" },
    "responseStatus": { "code": 200 },
    "objectRef": { "resource": “namespaces", "namespace": “foo” },
    "level": "Request",
    "auditID": "693f4726-2430-450a-83e1-123c050fde98",
    "annotations": { "authorization.k8s.io/decision": "allow" }
}
```

@ckranz

Supporting kubernetes audit log events.

- Create a new “Generic Event” interface
  - Event time, ability to extract values using fields
- Create a K8s Audit Event object
  - Event data is json object, stored in event
- Define new fields to extract values from K8s Audit Events
  - Uses Json Pointers to extract values
- Each Falco Rule now has a source
  - Default “syscall”, “k8s_audit” for K8s Audit Events

Kubernetes audit log fields.

- `jevt.value[<json_pointer>]`
  - Access any field from json object
- `jevt.time`
  - Access event timestamp
- `ka.verb, ka.uri, ka.user.name, ka.target.resource, …`
  - Access specific values from object
  - Implemented as macros:
    - ka.verb -> jevt.value[/verb]
    - ka.target.resource -> jevt.value[/objectRef/resource]
  - Full list: `falco —list=k8s_audit`

K8s audit log rule example.

```
- macro: contains_private_credentials
  condition: >
    (ka.req.configmap.obj contains "aws_access_key_id" or
     ka.req.configmap.obj contains "aws_s3_access_key_id" or
     ka.req.configmap.obj contains “password")

- macro: configmap
  condition: ka.target.resource=configmaps

- macro: modify
  condition: (ka.verb in (create,update,patch))

- rule: Create/Modify Configmap With Private Credentials
  desc: Detect creating/modifying a configmap containing a private credential
    (aws key, password, etc.)
  condition: configmap and modify and contains_private_credentials
  output: K8s configmap with private credential (user=%ka.user.name
          verb=%ka.verb name=%ka.req.configmap.name
          configmap=%ka.req.configmap.name config=%ka.req.configmap.obj)
  priority: WARNING
  source: k8s_audit
  tags: [k8s]
```

@ckranz

Extending rules/macros/lists.

Can combine rulesets to extend/modify behavior

```
falco -r <rules-file> -r <additional-rules-file> …
```

```
- macro: my macro
  condition: …
- list: my list
  items: …
- rule: my rule
  desc: …
  condition: …
  output: …
```

+

```
- macro: another macro
  condition: …
- list: another list
  items: …
- rule: another rule
  desc: …
  condition: …
  output: …
```

Installing and Integrations

Installing Falco.

- **Debian Package**
  - apt-get -y install falco
- **Redhat Package**
  - yum -y install falco
- **Installation Script**
  - curl -s s3.amazonaws.com/download.draios.com/stable/install-falco | sudo bash
- **Docker container**
  - docker pull sysdig/falco
- **Full instructions**
  - github.com/draios/falco/wiki/How-to-Install-Falco-for-Linux

Installing Falco on kubernetes.

- **Use Helm**
  - $ helm install --name sysdig-falco-1 stable/falco
  - https://sysdig.com/blog/falco-helm-chart/

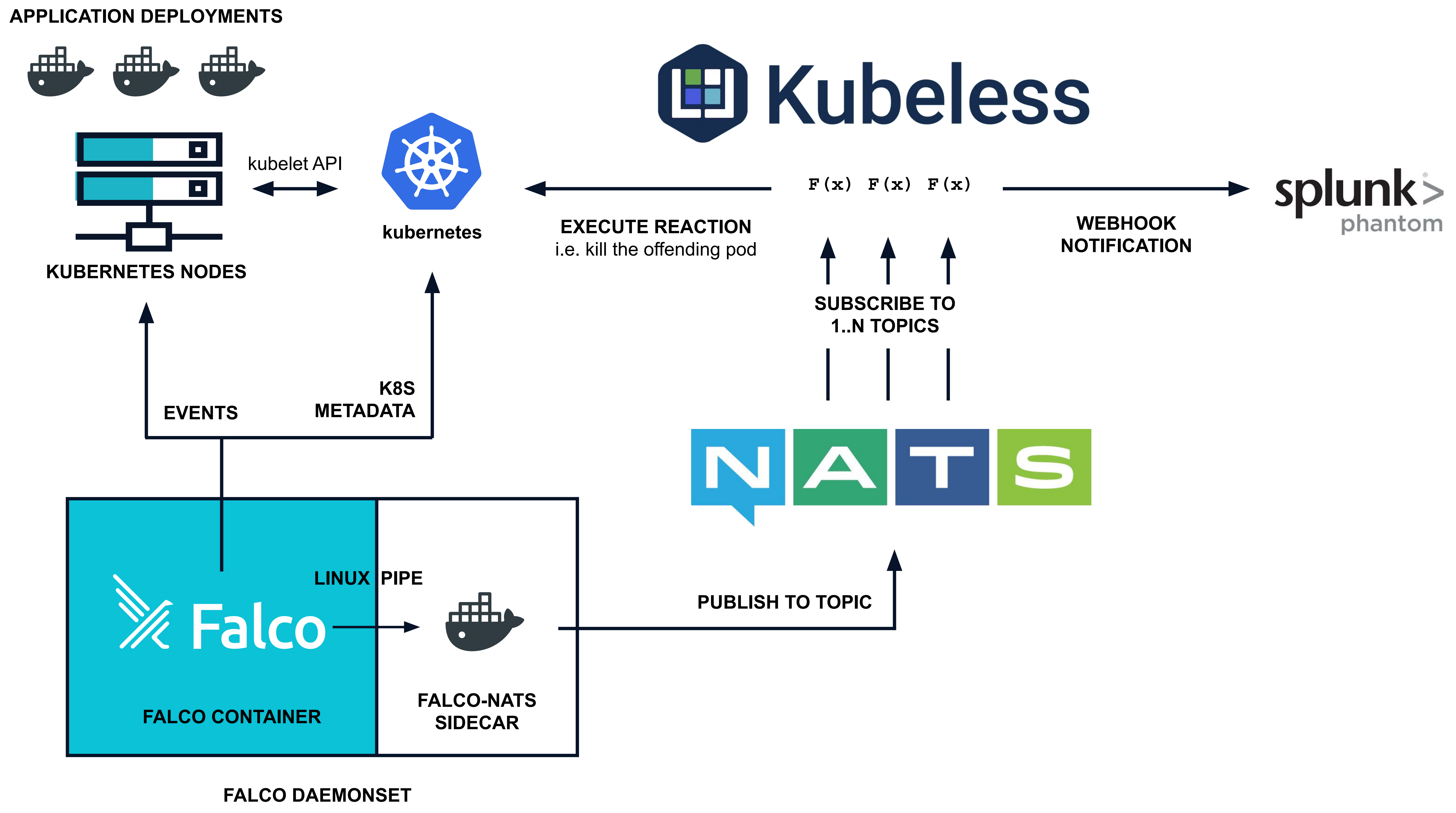
- **Install Falco as Kubernetes Daemonset**
  - https://github.com/draios/falco/tree/dev/examples/k8s-using-daemonset
  - Configuration stored in Kubernetes ConfigMaps
  - Conditions in a Falco Rule can leverage Kubernetes metadata to trigger events
  - Falco events can include Kubernetes metadata to give notification context:
    - name, id, labels for Pods, ReplicationController, Service, Namespace, ReplicaSet, and Deployment

@ckranz

Response engine & security playbooks.

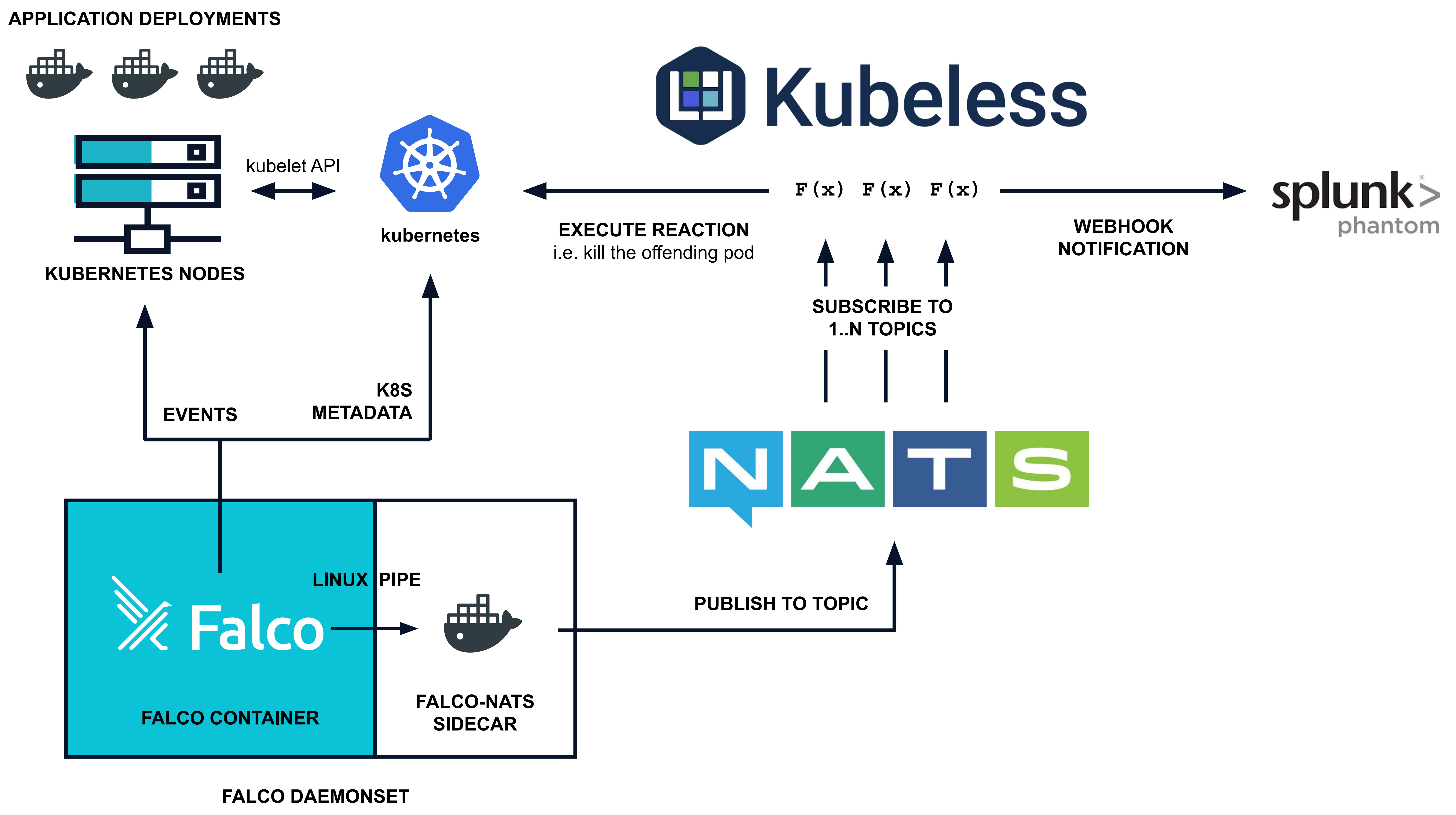
- **Detect abnormal events** with Falco
- **Publish alerts** to Pub/Sub service (NATS.io)
- Subscribers can **subscribe to various FALCO topics** to receive alerts:
  - FALCO.* - All alerts
  - FALCO.Notice - Alerts of priority “Notice” only
  - FALCO.Critical - Alerts of priority “Critical” only
- Subscribers can **take action** on alerts:
  - Kill offending Pod
  - Taint Nodes to prevent scheduling
  - Isolate Pod with Networking Policy
  - Send notification via Slack

Response engine & security playbooks.



@ckranz

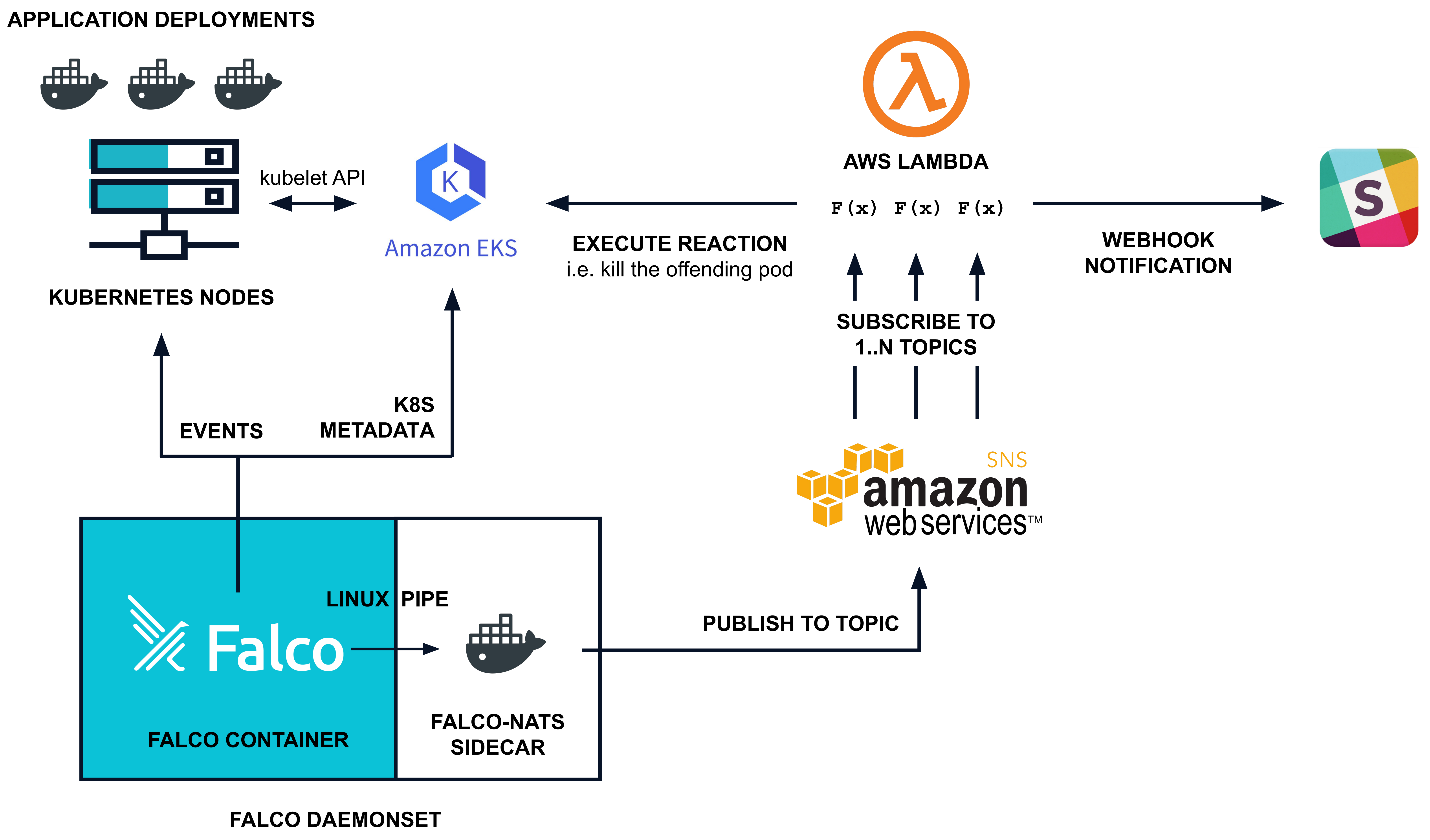
Response engine & security playbooks.



https://sysdig.com/blog/container-security-orchestration-falco-splunk-phantom/

@ckranz

Response engine & security playbooks.



https://aws.amazon.com/blogs/opensource/securing-amazon-eks-lambda-falco/

@ckranz

Response engine & security playbooks.

Detects abnormal event, Publishes alert to NATS

Subscribers receive Falco Alert through NATS Server

Kubeless receives Falco Alert, firing a function to delete the offending Kubernetes Pod

https://sysdig.com/blog/oss-container-security-runtime/

@ckranz

Functions for operations.

- Easily write simple functions to react to security events
- Multiple subscribers can take multiple actions
  - One function to delete a pod
  - One function to notify teams
  - One function to log events
- Small, reusable components

SIEM with EFK.

- Security Information and Event Management
  - Collect security events
  - Easily allow reporting and correlation of events across various data sources

- Elasticsearch, Fluentd, Kibana
  - Fluentd - Cloud Native log aggregation
  - Elasticsearch - Schema free JSON data store
  - Kibana - powerful data visualization tool for Elasticsearch

- https://sysdig.com/blog/kubernetes-security-logging-fluentd-falco/

SIEM with EFK.

Falco

Detects abnormal event, Publishes alert to stdout

fluentd

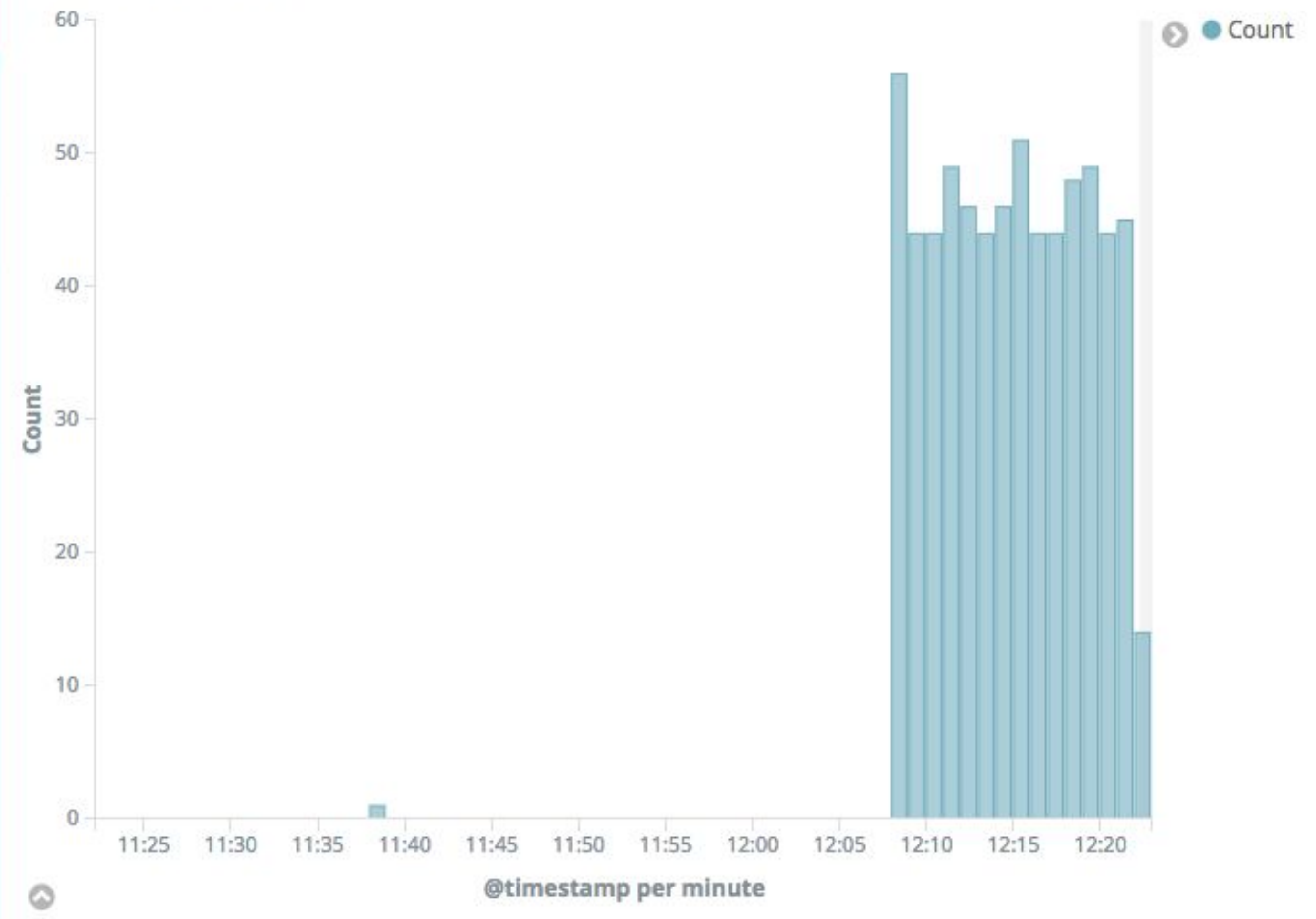
Fluentd ships alerts to Elasticsearch

kibana

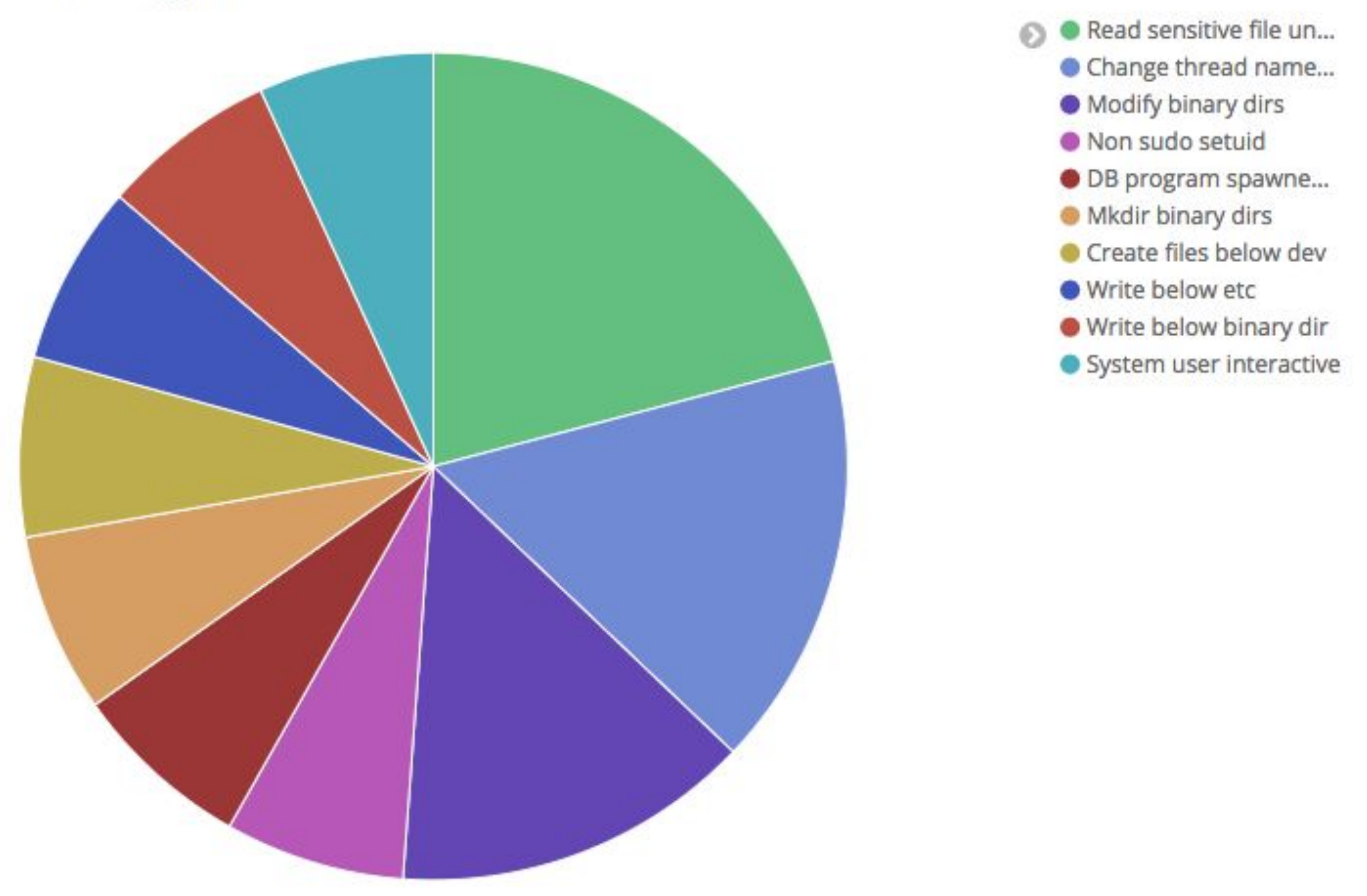
Kibana dashboards can be used to aggregate, filter, and report on alerts.

SIEM with EFK.

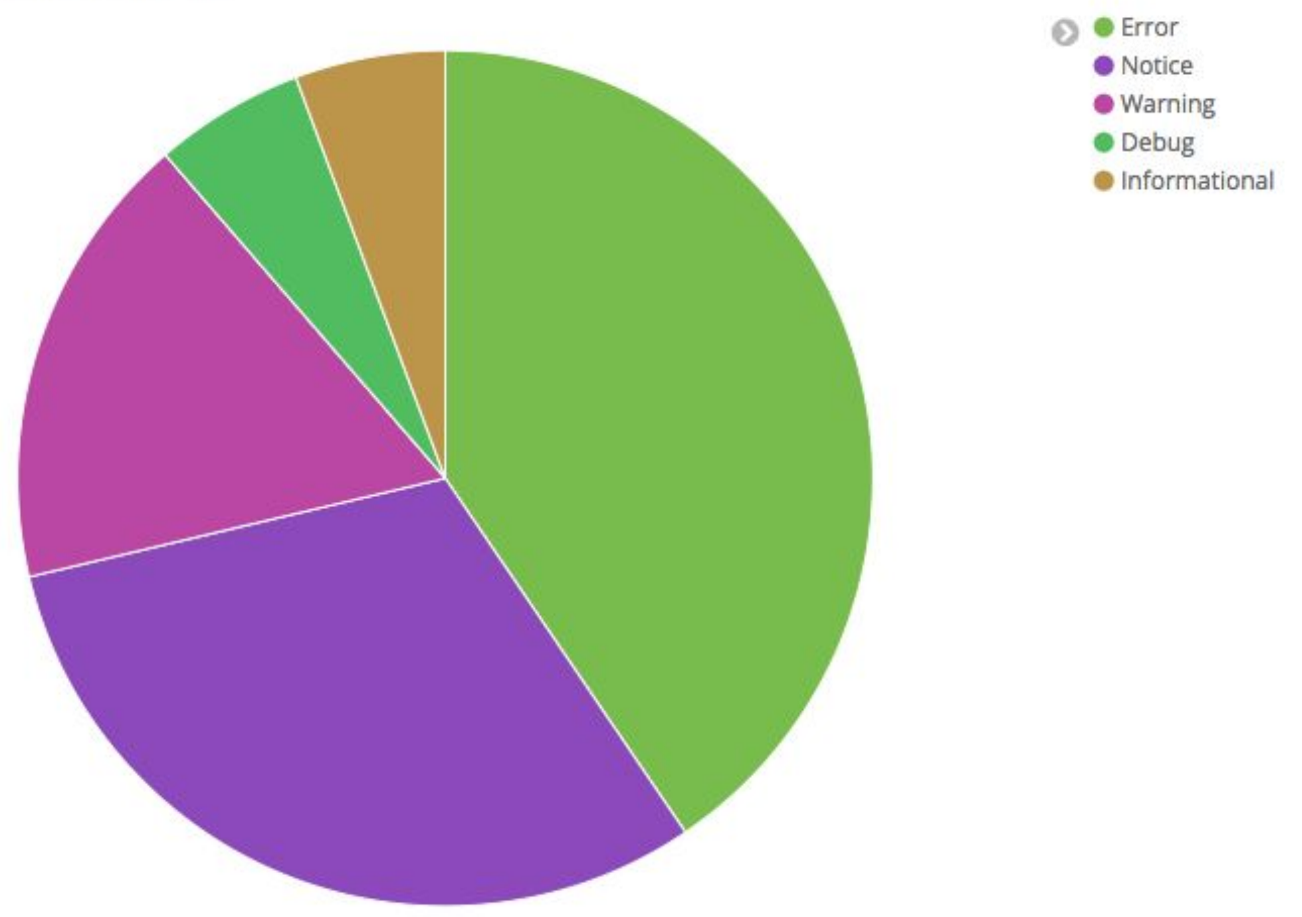
Falco - Alerts over Time



Falco - Top 10 Rules Triggered



Falco - Alerts by Priority - Pie



Falco - Top 20 Rules Triggered - Table

Top 20 Falco Rules Triggered	Count
Change thread namespace	90
Create files below dev	39
DB program spawned process	39
Mkdir binary dirs	39
Modify binary dirs	78
Non sudo setuid	39
Read sensitive file untrusted	116
Run shell untrusted	38
System procs network activity	38
System user interactive	38

Export: Raw Formatted

1 2 »

@ckranz

Build.

Image/Software Provenance

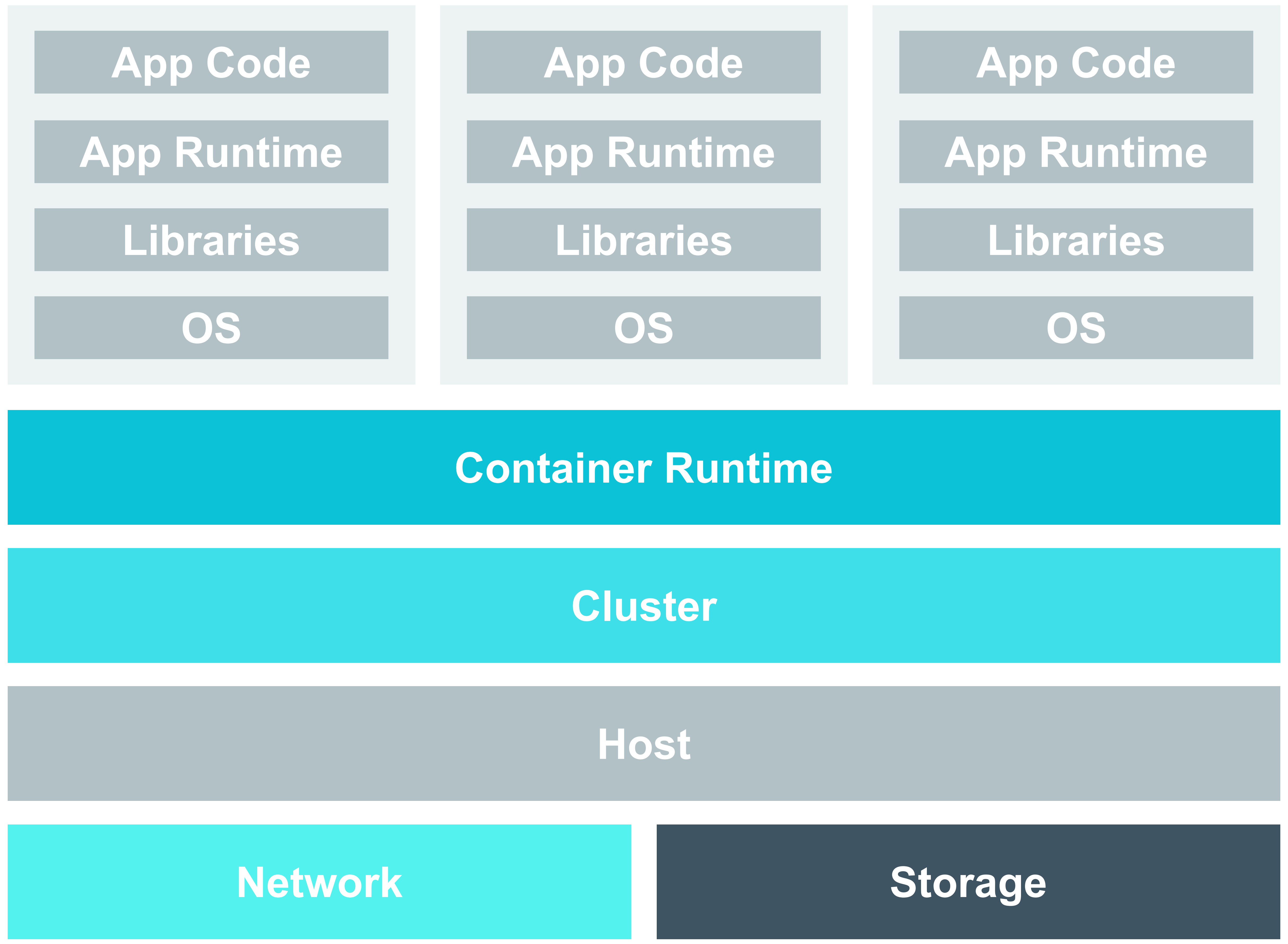
- Signed Images/Layers
- Artifact Signing

Vulnerability Management

- Upstream OS
- Application Vulnerabilities

ck

Runtime security.



@ckranz

Runtime.

Service/Container Admittance

Secure Secrets

Anomaly Detection

Forensics

ck