



Inspecting Nomad Events

Using Vector and Grafana Loki





Part 1

- Understand Nomad Events
- How to connect to event stream

Part 2

- How Nomad Events Sink collector works
- Plumbing a pipeline with vector.dev + Grafana Loki

Part 3

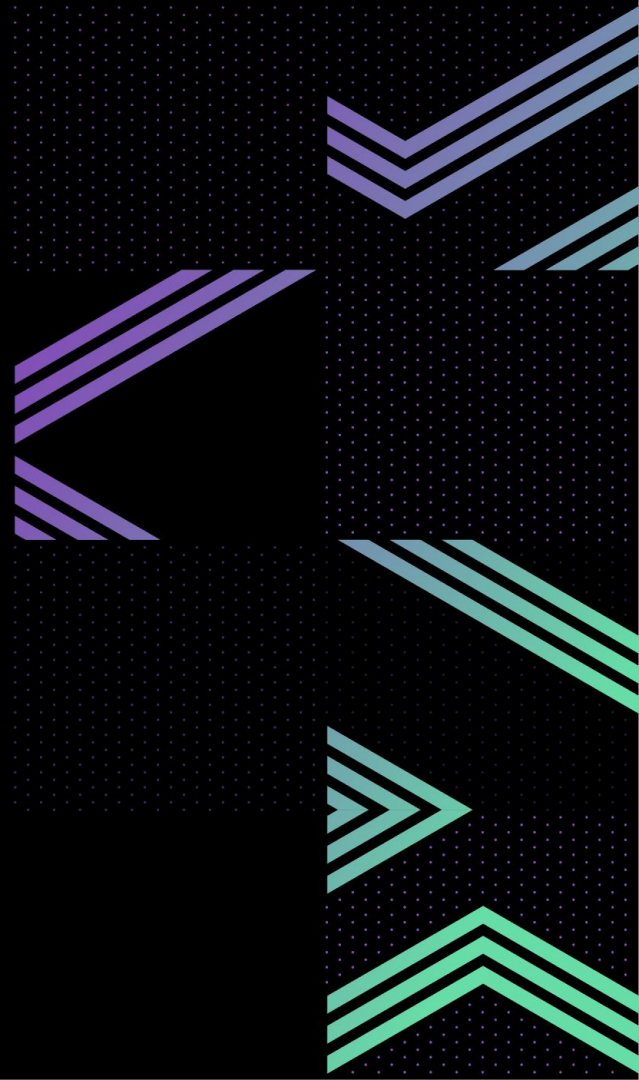
- Demo time!

Karan Sharma

(He/Him)

Software Engineer @ Zerodha

mrkaran.dev



What are Events?

(In the context of a Nomad cluster)



```
) nomad run example.nomad
==> 2022-02-08T19:58:23+05:30: Monitoring evaluation "8959c569"
    2022-02-08T19:58:23+05:30: Evaluation triggered by job "example"
    2022-02-08T19:58:23+05:30: Allocation "ffc7e269" created: node "edf80039", group "cache"
==> 2022-02-08T19:58:24+05:30: Monitoring evaluation "8959c569"
    2022-02-08T19:58:24+05:30: Evaluation within deployment: "92a79684"
    2022-02-08T19:58:24+05:30: Allocation "ffc7e269" status changed: "pending" -> "running" (Tasks are running)
    2022-02-08T19:58:24+05:30: Evaluation status changed: "pending" -> "complete"
==> 2022-02-08T19:58:24+05:30: Evaluation "8959c569" finished with status "complete"
==> 2022-02-08T19:58:24+05:30: Monitoring deployment "92a79684"
    ✓ Deployment "92a79684" successful

2022-02-08T19:58:35+05:30
ID           = 92a79684
Job ID       = example
Job Version  = 0
Status       = successful
Description  = Deployment completed successfully

Deployed
Task Group  Desired  Placed  Healthy  Unhealthy  Progress Deadline
cache       1        1       1        0          2022-02-08T20:08:33+05:30
```

Finite State Machine



Nomad applies all state changes via FSM.

- State changes are logged as “events” and pushed to an event broker
- Event broker ensures each server gets identical set of events


```
{
  "Events": [
    {
      "FilterKeys": [
        "example",
        "02a7efac-11f2-6084-5cb2-b7812f2d9442"
      ],
      "Index": 19,
      "Key": "ac89cbcd-f5df-3659-b63f-0fe377ecc3ce",
      "Namespace": "default",
      "Payload": {
        "Evaluation": {
          "CreateIndex": 17,
          "CreateTime": 1644325944257324800,
          "DeploymentID": "02a7efac-11f2-6084-5cb2-b7812f2d9442",
          "ID": "ac89cbcd-f5df-3659-b63f-0fe377ecc3ce",
          "JobID": "example",
          "ModifyIndex": 19,
          "ModifyTime": 1644325944510045400,
          "Namespace": "default",
          "Priority": 50,
          "QueuedAllocations": {
            "cache": 0
          },
          "SnapshotIndex": 17,
          "Status": "complete",
          "TriggeredBy": "deployment-watcher",
          "Type": "service"
        }
      },
      "Topic": "Evaluation",
      "Type": "EvaluationUpdated"
    }
  ],
  "Index": 19
}
```

Streaming Events

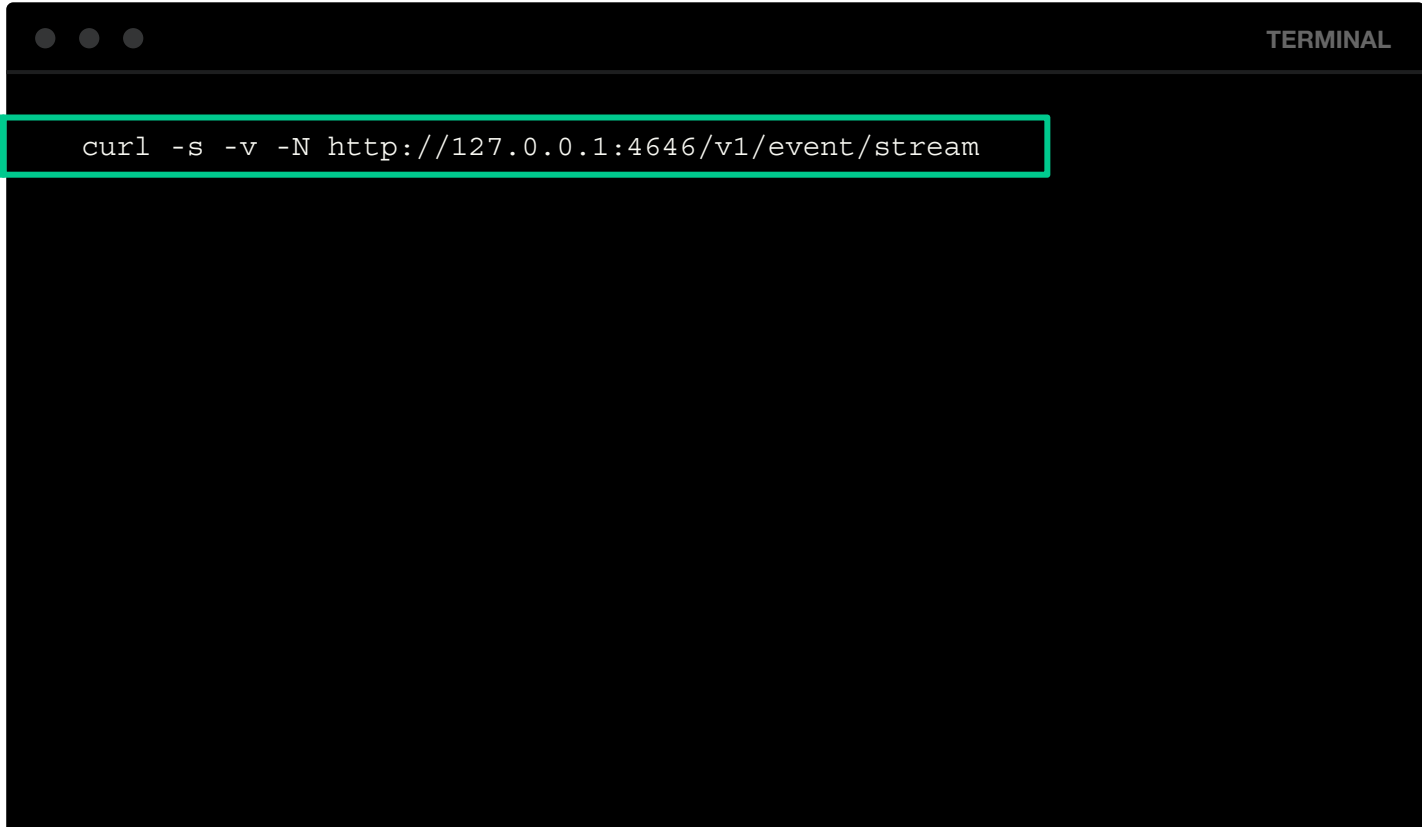


- Event Stream connects to the broker and streams events from memory.
- Event broker can handle max `n` events in memory.
- Incoming stream contains an “index”
- Events can be filtered by Topic, Namespace etc

Using the Events HTTP API



Subscribe to Events Stream over HTTP



Nomad Events Sink



Listen to Events

- Use Golang client for the Nomad HTTP API.
- Subscribe to an events stream

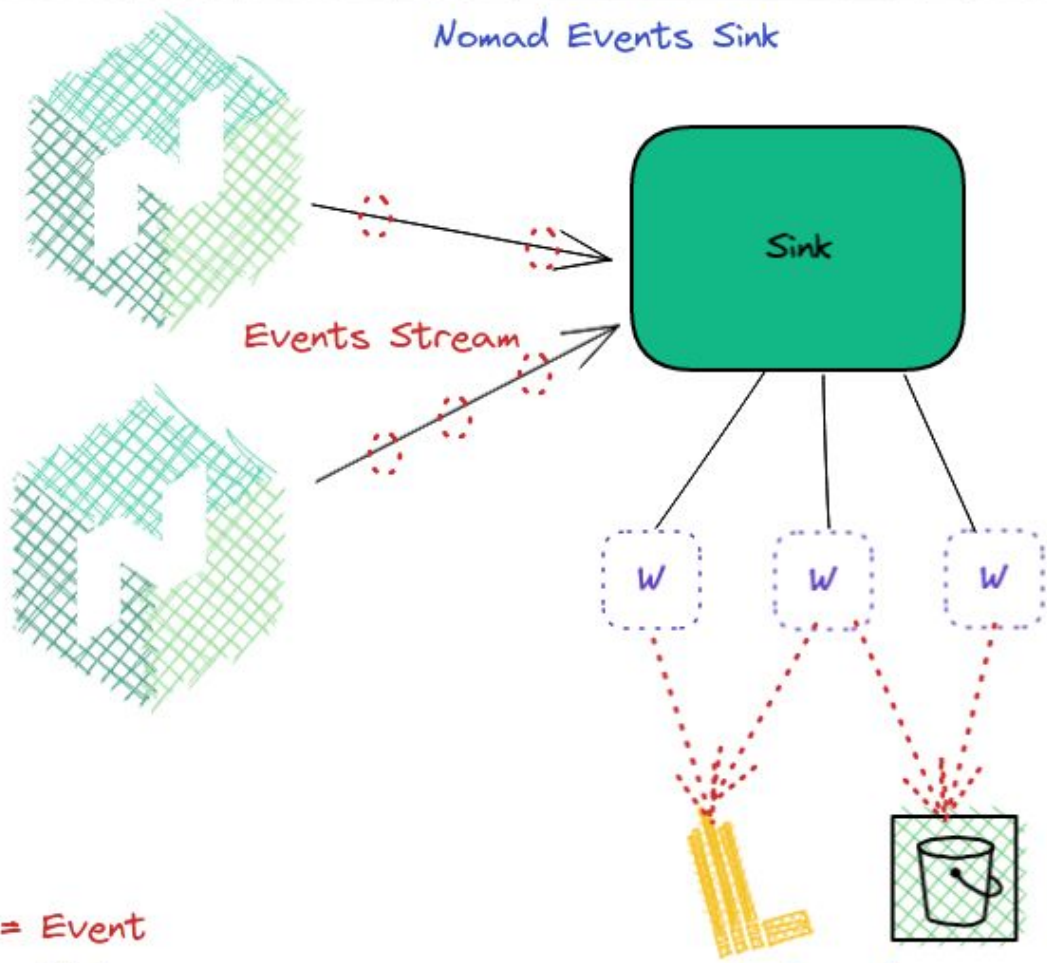
Collect Events

- Push incoming events to a queue
- Start background workers to consume events from the queue
- Flush the events to a sink

Process Events

- Implement a `Provider` interface for sinks.
- HTTP Sink Provider flushes batch of events to an upstream API endpoint
- Can implement other providers

Nomad Events Sink



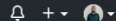
⊙ = Event
W = Worker

Providers



Search or jump to...

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[mr-karan / nomad-events-sink](#) (Public)

[Unwatch](#) 2 [Fork](#) 0 [Star](#) 13

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

main 1 branch 1 tag

[Go to file](#) [Add file](#) [Code](#)

mr-karan fix: image name	✓ 77753cd on 10 Nov 2021	🕒 2 commits
.github/workflows	init project	3 months ago
cmd/app	init project	3 months ago
deploy	fix: image name	3 months ago
docs	init project	3 months ago
examples	init project	3 months ago
internal	init project	3 months ago
.gitignore	init project	3 months ago
.goreleaser.yml	fix: image name	3 months ago
Dockerfile	init project	3 months ago
LICENSE	init project	3 months ago
Makefile	init project	3 months ago
README.md	fix: image name	3 months ago
config_sample.toml	init project	3 months ago
go.mod	init project	3 months ago
go.sum	init project	3 months ago

About

An events collection agent which processes Nomad Events and dumps to external sink providers like HTTP

nomad

- Readme
- MIT License
- 13 stars
- 2 watching
- 0 forks

Releases 1

v0.1.0 Latest
on 10 Nov 2021

Packages 1

nomad-events-sink

Languages




Code Walkthrough



Plumbing different pieces together

Using vector to push events to Grafana
Loki





vector

Use vector to build an end to end pipeline for collecting and storing events

Vector supports multiple sources and sinks. It can even transform JSON events if you wish to.

Take control of your observability data

Collect, transform, and route *all* your logs and metrics with *one* simple tool.

Sources | **Transforms** | **Sinks**

The diagram illustrates a data pipeline. On the left, under 'Sources', are 'File', 'Kafka', and 'Statsd'. Arrows from these sources point to a central 'Transforms' section. The 'Transforms' section consists of three stacked blocks: 'Structure' (top, cyan), 'Sample' (middle, blue), and 'Aggregate' (bottom, pink). A large blue 'V' logo is above the 'Structure' block. On the right, under 'Sinks', are 'S3', 'Elastic search', and 'Prometheus'. Dashed arrows point from the 'Structure' block to 'S3', from the 'Sample' block to 'Elastic search', and from the 'Aggregate' block to 'Prometheus'. A vertical line separates the sources from the transforms, and another vertical line separates the transforms from the sinks.

HTTP Source



```
1 # Ingest logs from HTTP server.  
2 [sources.nomad_events]  
3 type = "http"  
4 address = "0.0.0.0:3333"  
5 decoding.codec = "json"
```

Transform the JSON event



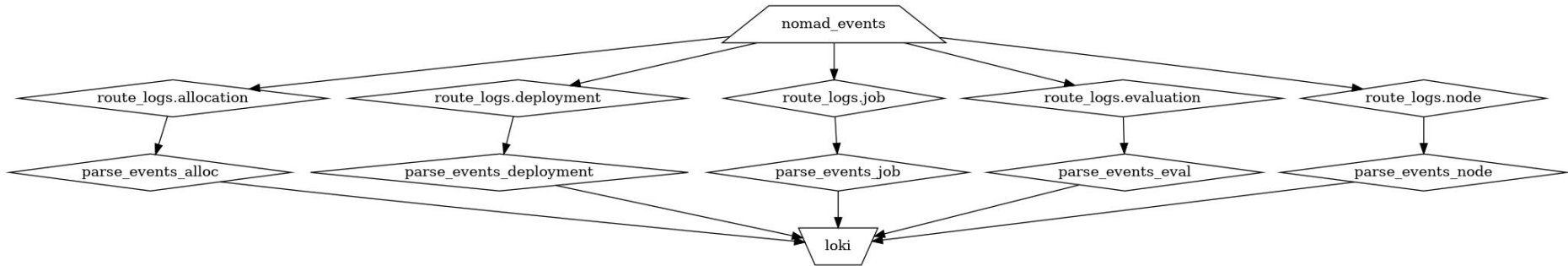
```
1 [transforms.parse_events_alloc]
2 type = "remap"
3 inputs = ["route_logs.allocation"]
4 source = ''
5 # Parse the Allocation Event.
6
7 # Prepare the full event payload with details that are needed.
8 . = {
9   "timestamp": .timestamp,
10  "topic": .Topic,
11  "type": .Type,
12  "client_description": .Payload.Allocation.ClientDescription,
13  "client_status": .Payload.Allocation.ClientStatus,
14  "desired_status": .Payload.Allocation.DesiredStatus,
15  "job_id": .Payload.Allocation.JobID,
16  "name": .Payload.Allocation.Name,
17  "namespace": .Payload.Allocation.Namespace,
18  "node_name": .Payload.Allocation.NodeName,
19  "task_group": .Payload.Allocation.TaskGroup
20 }
21 ''
```



Output events to Sink

```
1 # Output to Loki
2 [sinks.loki]
3 type = "loki"
4 inputs = ["parse_events_*"]
5 endpoint = "http://loki:3100"
6 encoding.codec = "json"
7 encoding.timestamp_format = "rfc3339"
8 healthcheck.enabled = true
9
10 # Add labels to identify a log stream.
11 labels.namespace = "{{ namespace }}"
12 labels.topic = "{{ topic }}"
13 labels.type = "{{ type }}"
14
15 # Remove fields that have been converted to labels to avoid having them twice.
16 remove_label_fields = false
17
18 out_of_order_action="rewrite_timestamp"
19 remove_timestamp=true
20
```

Graphviz representation



Demo Time



Possibilities with Events Stream



- Overview of all components in a single place
- Log all “drained” node updates
- Alerts on failed allocations
- When a container image version has changed, send it to Slack.



Thank You

twitter.com/mrkaran_ | github.com/mr-karan | mrkaran.dev