Nomad

- Suppress Nomad's logging overhead

Suppress Nomad's logging overhead

When managing Docker containers with Nomad, logs are handled in this way

- Docker capture stdout/stderr of the containers and write them to Nomad's FIFO
- Nomad spawns 2 processes (logmon and docker_logger) per container which read the logs sent by the docker container, and write them where Nomad expects them (for example in /opt/nomad/data/alloc/XXXXX/alloc/logs/task.stdout.0)
- When querying logs through the Nomad API (either using nomad alloc log or the web interface), the /opt/nomad/data/alloc/XXXXX/alloc/logs/task.stdout.0 is red/tailed

This is very inefficient, especially because of the logmon and docker_logger process : each of them consume arround 50 or 60MB, so a 100 to 120MB of overhead per container ! If you run a few big containers, it might not be a big problem, but when you use a lot of small sidecars (like I do, not only envoys for the service mesh, but also a lot of pgbouncer, ~~redis~~ valkey, memcached, nginx etc.), this is insane. On some of my nodes, this overhead represented ~30% of the total used RAM.

Nomad lets you disable log collection, either globaly, in the agent config

```
plugin "docker" {
  config {
    disable_log_collection = true
  }
}
```

Or for individual task, in the job file

```
task "metrics-proxy" {
      driver = "docker"

      # Reduce Docker logs collection (huge) overhead
      logs {
        disabled = true
      }
[...]
```

This removes the overhead, but also disable any log collection. You can collect logs through other means, but you won't be able to display them from the Nomad API or the web interface, which can

make debugging harder.

Here I'll present how I removed the overhead, while still being able to check logs.

- First step, is to disable Nomad's log collection as above, but also enable the fluentd log driver so all the Docker logs will be sent to a fluentd endpoint. To do this, add/set in your agent's config

```
plugin "docker" {
  config {
    disable_log_collection = true
    logging {
      type = "fluentd"
      config {
        # Send logs to a local fluentd service, running on port 4224
        fluentd-address = "127.0.0.1:4224"
        fluentd-async = true
        fluentd-buffer-limit = 10000
        # This is important for the fluentd service to access the metadata
        env =
"NOMAD_JOB_NAME,NOMAD_GROUP_NAME,NOMAD_DC,NOMAD_REGION,NOMAD_TASK_NAME,NOMAD_ALLOC_INDEX,NOMAD
_ALLOC_ID,NOMAD_NAMESPACE"
      }
    }
  }
}
```

> You will have to restart nomad agent and all running container for the change to take effect

- Run a local vector agent, which will listen on the port 4224 with a fluent source. There're lots of way to run vector (I do it with a system job, but it can also run natively on the host). The vector instance will receive structured logs directly from the Docker daemon, and can then write files where Nomad expects them. For example

```
sources:
  in_fluent:
    type: fluent
    address: 127.0.0.1:4224

transforms:
  transform_fluent:
```

```
    type: remap
    inputs: ["in_fluent"]
    # Map .log content in .message so we can use the text codec in sinks
    source: |
      .message = del(.log)

sinks:
  # Duplicate fluentd logs to files so Nomad API can read it
  out_nomad_files:
    type: file
    path: /opt/nomad/data/alloc/{{ .NOMAD_ALLOC_ID }}/alloc/logs/{{ .NOMAD_TASK_NAME }}.{{
.source }}.0
    inputs: ["transform_fluent"]
    encoding:
      codec: text
```

> You will have to handle log rotation yourself, for example with logrotate

With this in place, vector will create logs for Nomad, just like nomad's log collector would have done, but in a much more efficient way. You can query logs using nomad alloc logs, or from the web interface. And the nice thing is you can also send logs elsewhere (for example, a ES instance, or a Loki server, vector supports a lot of different sinks)