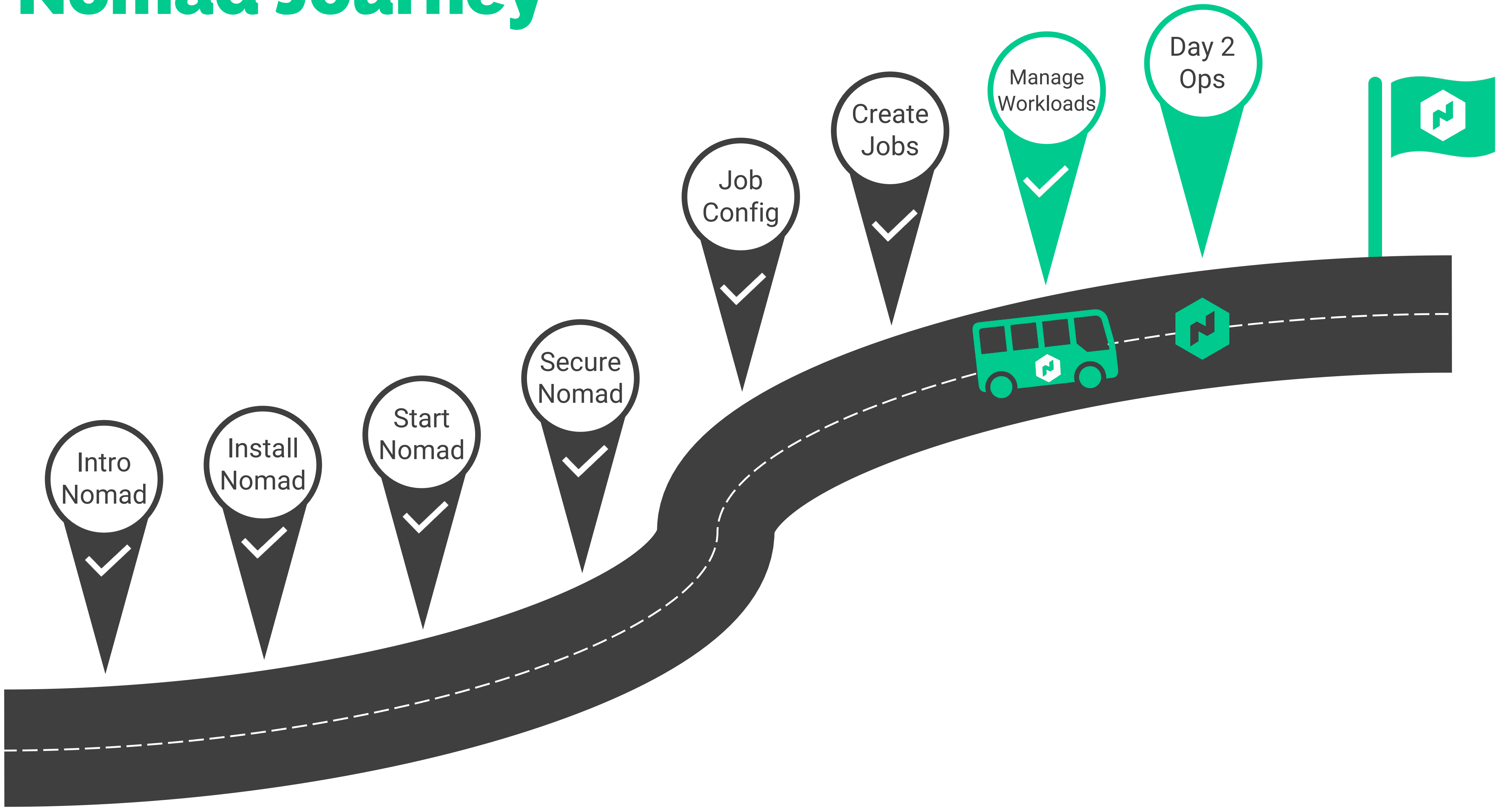
Managing Nomad Environments

Nomad Journey



Managing Nomad Environments

- Like any application or platform, Nomad requires operational monitoring, maintenance, upgrades, key rotations, and troubleshooting
- The same goes for the workloads running on Nomad – they need to be updated, monitored, and patched

Now that we know how to deploy Nomad and run jobs:

How do we manage Day 2 Operations of our Nomad cluster?

What are the ways we can monitor our environment and the applications running on Nomad?

Managing Nomad Environments

❑ Monitoring the Nomad Environment

❑ Monitoring Application Logs

❑ Rotating Gossip Encryption Key

❑ Upgrading Nomad to Newer Version

Monitoring the Nomad Environment

Nomad Monitoring

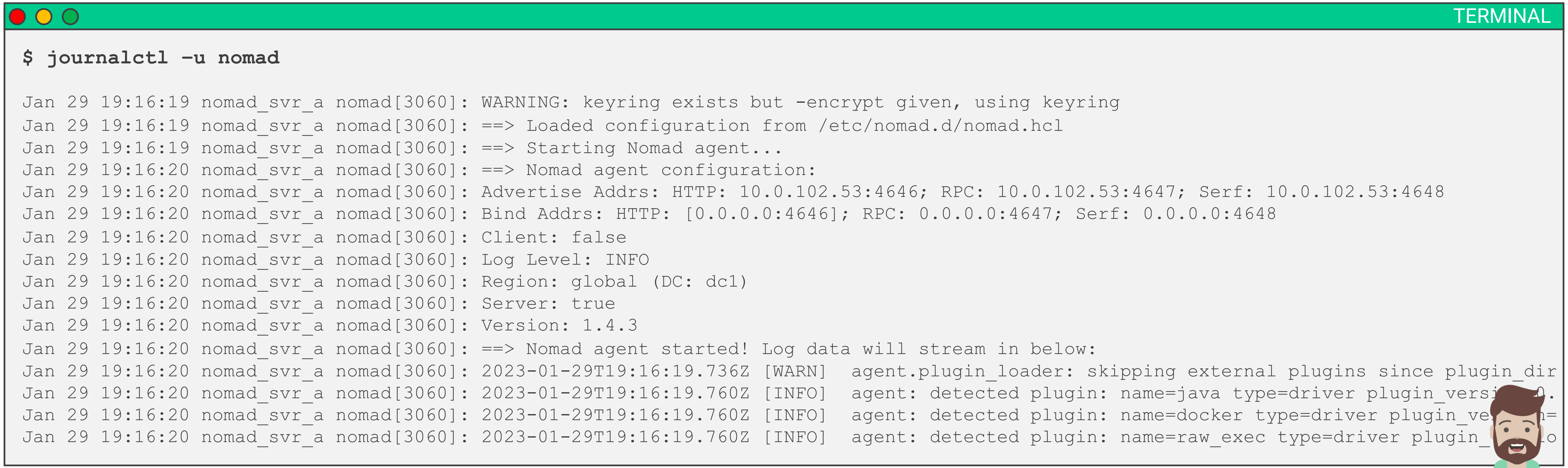
- Nomad offers multiple, integrated options to monitor the cluster for system events, audit logging, and events
- Nomad server and client agents collect runtime metrics that are useful for monitoring the health and performance of Nomad clusters
- The cluster leader also collects specific metrics as well
- Nomad clients have separate metrics for the host they are running on as well as each allocation being run
- Telemetry data can be sent to upstream systems at a 1 second interval (default), allowing you to leverage the capabilities of a monitoring provider to set up alerts and dashboards

Viewing System Logs

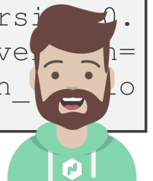
You can view journal logs on Linux operating systems to view important information about the Nomad service. This is great for troubleshooting issues if Nomad won't start or if you think you have misconfigurations in your Nomad agent configuration

```
$ journactl -b -u nomad
```

```
(-u = filter by unit)
(-b = filter by current boot)
```


TERMINAL

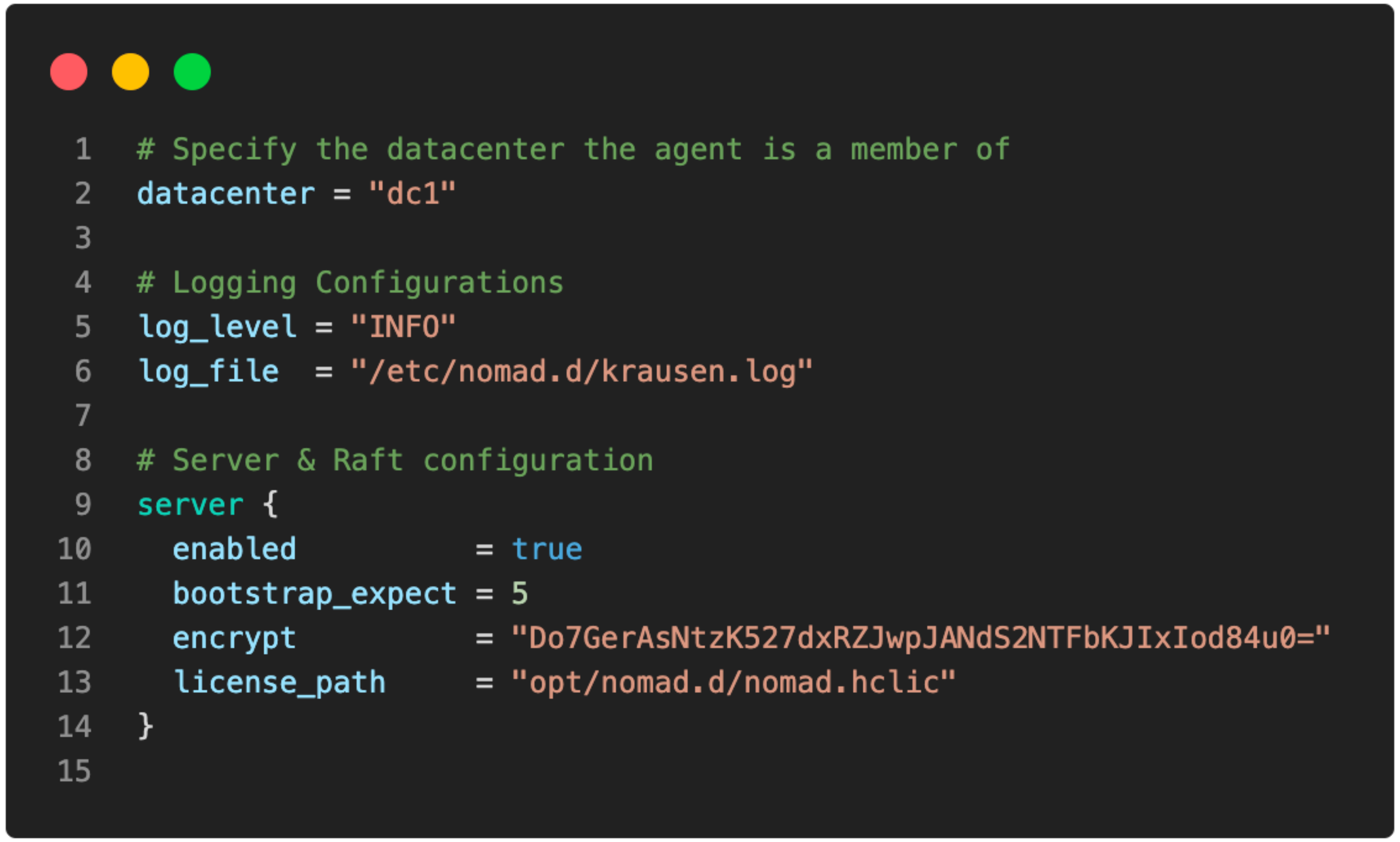
```
$ journalctl -u nomad

Jan 29 19:16:19 nomad_svr_a nomad[3060]: WARNING: keyring exists but -encrypt given, using keyring
Jan 29 19:16:19 nomad_svr_a nomad[3060]: ==> Loaded configuration from /etc/nomad.d/nomad.hcl
Jan 29 19:16:19 nomad_svr_a nomad[3060]: ==> Starting Nomad agent...
Jan 29 19:16:20 nomad_svr_a nomad[3060]: ==> Nomad agent configuration:
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Advertise Addrs: HTTP: 10.0.102.53:4646; RPC: 10.0.102.53:4647; Serf: 10.0.102.53:4648
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Bind Addrs: HTTP: [0.0.0.0:4646]; RPC: 0.0.0.0:4647; Serf: 0.0.0.0:4648
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Client: false
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Log Level: INFO
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Region: global (DC: dc1)
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Server: true
Jan 29 19:16:20 nomad_svr_a nomad[3060]: Version: 1.4.3
Jan 29 19:16:20 nomad_svr_a nomad[3060]: ==> Nomad agent started! Log data will stream in below:
Jan 29 19:16:20 nomad_svr_a nomad[3060]: 2023-01-29T19:16:19.736Z [WARN]  agent.plugin_loader: skipping external plugins since plugin_dir
Jan 29 19:16:20 nomad_svr_a nomad[3060]: 2023-01-29T19:16:19.760Z [INFO]  agent: detected plugin: name=java type=driver plugin_versi   0.
Jan 29 19:16:20 nomad_svr_a nomad[3060]: 2023-01-29T19:16:19.760Z [INFO]  agent: detected plugin: name=docker type=driver plugin_ve    n=
Jan 29 19:16:20 nomad_svr_a nomad[3060]: 2023-01-29T19:16:19.760Z [INFO]  agent: detected plugin: name=raw_exec type=driver plugin_    o
```



Viewing System Logs

Use the `log_level` and `log_file` parameters in the Nomad agent configuration file to output logs for Nomad.

Add the parameters to your agent configuration file

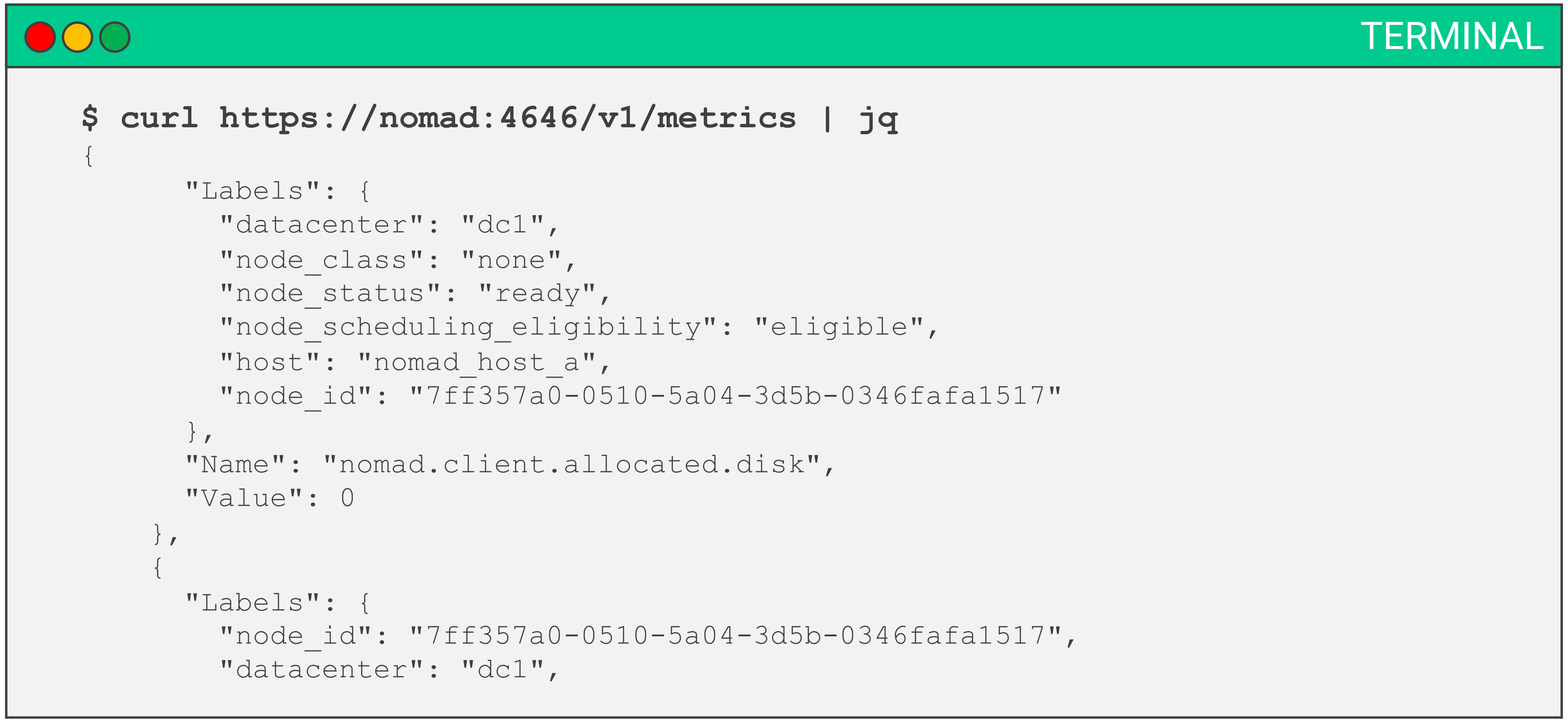


```
# Specify the datacenter the agent is a member of
datacenter = "dc1"

# Logging Configurations
log_level = "INFO"
log_file  = "/etc/nomad.d/krausen.log"

# Server & Raft configuration
server {
  enabled          = true
  bootstrap_expect = 5
  encrypt          = "Do7GerAsNtzK527dxRZJwpJANdS2NTFbKJIxIod84u0="
  license_path     = "opt/nomad.d/nomad.hclic"
}

```

Getting Metrics from the API

Nomad offers an API endpoint to return metrics for the current Nomad process

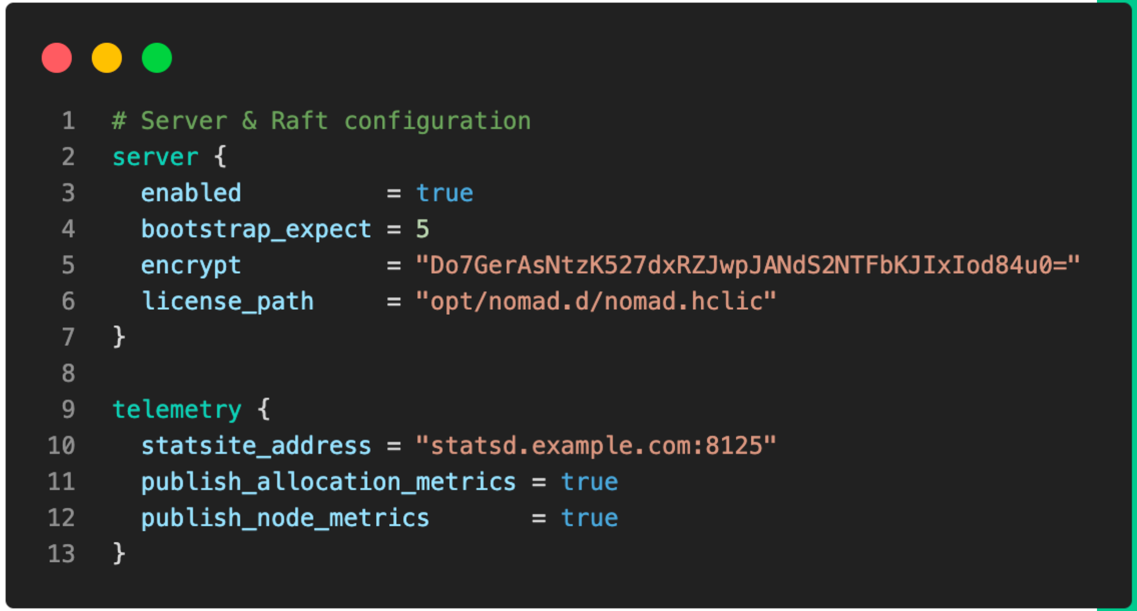
`https://nomad.example.com:4646/v1/metrics`



```
$ curl https://nomad:4646/v1/metrics | jq
{
      "Labels": {
        "datacenter": "dc1",
        "node_class": "none",
        "node_status": "ready",
        "node_scheduling_eligibility": "eligible",
        "host": "nomad_host_a",
        "node_id": "7ff357a0-0510-5a04-3d5b-0346fafa1517"
      },
      "Name": "nomad.client.allocated.disk",
      "Value": 0
    },
    {
      "Labels": {
        "node_id": "7ff357a0-0510-5a04-3d5b-0346fafa1517",
        "datacenter": "dc1",
```

Configuring Telemetry

- Telemetry logs provide metrics about the health of our cluster along with key performance metrics we can use to validate or forecast the resources of our cluster
- To enable and use telemetry metrics, add the stanza to Nomad client and server configuration files to send metrics to Prometheus/Graphana, Splunk, DataDog, etc.
- Examples of key performance indicators (KPIs) include:
  - ➢ Raft
  - ➢ Scheduling
  - ➢ Capacity
  - ➢ Job and Task Status
  - ➢ Runtime Metrics



```
# Server & Raft configuration
server {
  enabled          = true
  bootstrap_expect = 5
  encrypt          = "Do7GerAsNtzK527dxRZJwpJANdS2NTFbKJIxIod84u0="
  license_path     = "opt/nomad.d/nomad.hclic"
}

telemetry {
  statsite_address = "statsd.example.com:8125"
  publish_allocation_metrics = true
  publish_node_metrics       = true
}
```

DEMO | Nomad Monitoring

Monitoring Applications

Monitoring Applications

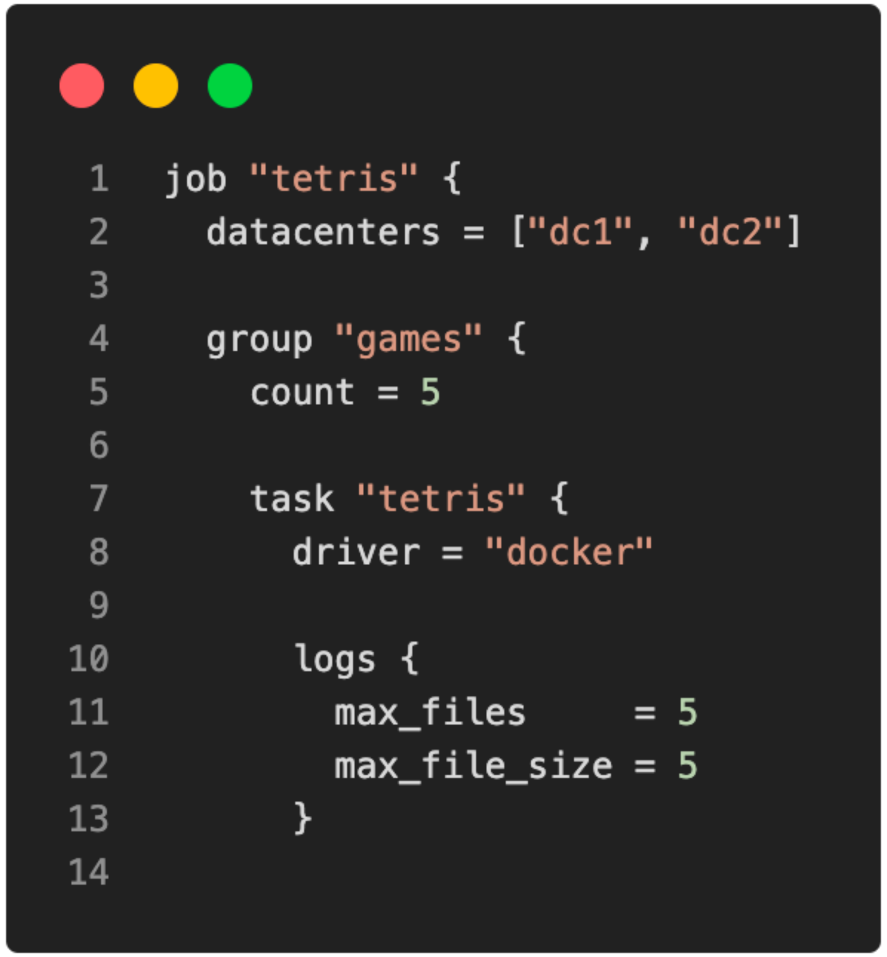
- Beyond Nomad cluster logs, you might need to gather or monitor application logs for security, performance, or troubleshooting purposes

- Nomad offers multiple options for collecting logs directly from an application:
  - All tasks will write logs to the Nomad client under the `alloc/logs` directory
  - Use the `nomad alloc logs` CLI command to view logs of an allocation
  - Stream logs via the API using the `/v1/client/fs/logs/<alloc>` endpoint

Allocation Logs

All tasks in Nomad will automatically write logs when running on Nomad clients

- **Currently, there is no way to disable logging for tasks**
- Tasks will write logs to a file in the directory: `/data/alloc/<task>/alloc/logs/<stdout/stderr>.index`
- You can change the maximum number of files Nomad will retain and the maximum size of those files



```
job "tetris" {
  datacenters = ["dc1", "dc2"]

  group "games" {
    count = 5

    task "tetris" {
      driver = "docker"

      logs {
        max_files     = 5
        max_file_size = 5
      }

```

Allocation Logs

Path on a Nomad client for the `tetris` job:

`/etc/nomad.d/data/alloc/003b26e5-8f6c-af88-e5e7-a07a1792cbd9/alloc/logs`

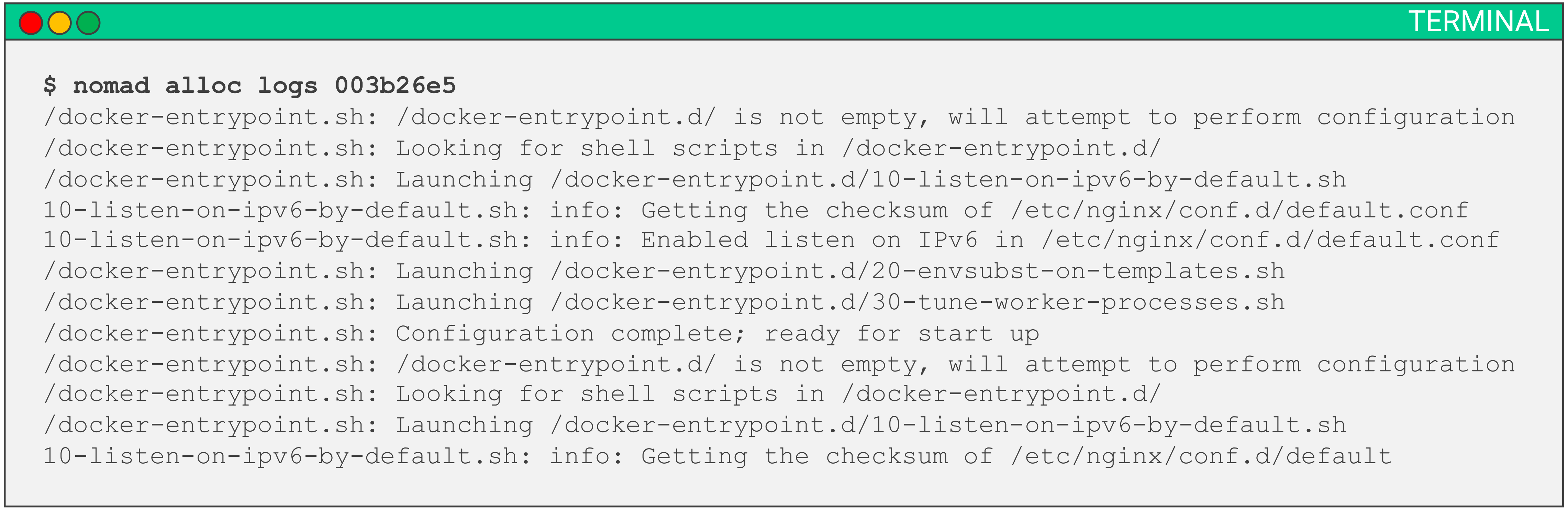
TERMINAL

```
$ ls
tetris.stderr.0  tetris.stdout.0

$ cat tetris.stdout.0
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
```

Using the Nomad CLI to View Logs

Use the `nomad alloc logs <alloc_id>` to view logs directly from the CLI

- **Requires a valid ACL token**



```
$ nomad alloc logs 003b26e5
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default
```

Stream Logs via API

- Using the API, you can stream the contents of a file in an allocation directory, such as the `stdout` or `stderr` logs
- Uses the full allocation ID (not the short one) as part of the API endpoint path

TERMINAL

```
$ curl \
  http://localhost:4646/v1/client/fs/stream/003b26e5-8f6c-af88-e5e7-a07a1792cbd9
     ?path=/alloc/logs/tetris.stdout.0
{"Data":"L2RvY2tlci1lbnRyeXBvaW50LnNoOiAvZG9ja2VyLWVudHJ5cG9pbnQuZC8gaXMgbm90IGVtcHR5
LCB3aWxsIGF0dGVtcHQgdG8gcGVyZm9ybSBjb25maWd1cmF0aW9uCi9kb2NrZXItZW50cnlwb2ludC5zaDogT
G9va2luZyBmb3Igc2hlbGwgc2NyaXB0cyBpbiAvZG9ja2VyLWVudHJ5cG9pbnQuZC8KL2RvY2tlci1lbnRyeX
BvaW50LnNoOiBMYXVuY2hpbmcgL2RvY2tlci1lbnRyeXBvaW50LmQvMTAtbGlzdGVuLW9uLWlwdjYtYnktZGV
mYXVsdC5zaAoxMC1saXN0ZW4tb24taXB2Ni1ieS1kZWZhdWx0LnNoOiBpbmZvOiBHZXR0aW5nIHRoZSBjaGVj
a3N1bSBvZiAvZXRjL25naW54L2NvbmYuZC9kZWZhdWx0LmNvbmYKMTAtbGlzdGVuLW9uLWlwdjYtYnktZGVmY
XVsdC5zaDogaW5mbzogRW5hYmxlZCBsaXN0ZW4gb24gSVB2NiBpbiAvZXRjL25naW54L2NvbmYuZC9kZWZhdW
```

DEMO

Monitoring Applications

Rotating the Gossip Encryption Key

Rotate the Gossip Encryption Key

- Many organizations have security policies that require encryption keys to be rotated at least once a year, sometimes for often
- Nomad has built-in features that allow you to easily generate, install, use, and remove an encryption key from the keyring

Required Steps for Rotating the Gossip Encryption Key

1. Generate a new key using `keyring generate`
2. Install the new key to all members in the cluster
3. Change the encryption key used for gossip
4. Remove the old key from the cluster

Generate a New Gossip Encryption Key

- Use the `key generate` command included in Nomad to create a new key
- This is the same command we used when talking about securing Nomad earlier in the course and you could use it the same way
- Note: You'll get a unique key each time you run this command

TERMINAL

```
$ nomad operator gossip keyring generate
WHvDRJKnCA3UyCUQv0wclfZM5XABgXXnn4qYxQhQBJE=
```

Install the New Gossip Encryption Key

- Install the new gossip encryption key. This will broadcast the key to all members in the cluster – only need to run it on one node, and all the other nodes will receive it.

Use the `nomad operator gossip keyring install <key>` command

TERMINAL

```
$ nomad operator gossip keyring install WHvDRJKnCA3UyCUQv0wclfZM5XABgXXnn4qYxQhQBJE=
Installing new gossip encryption key...
```

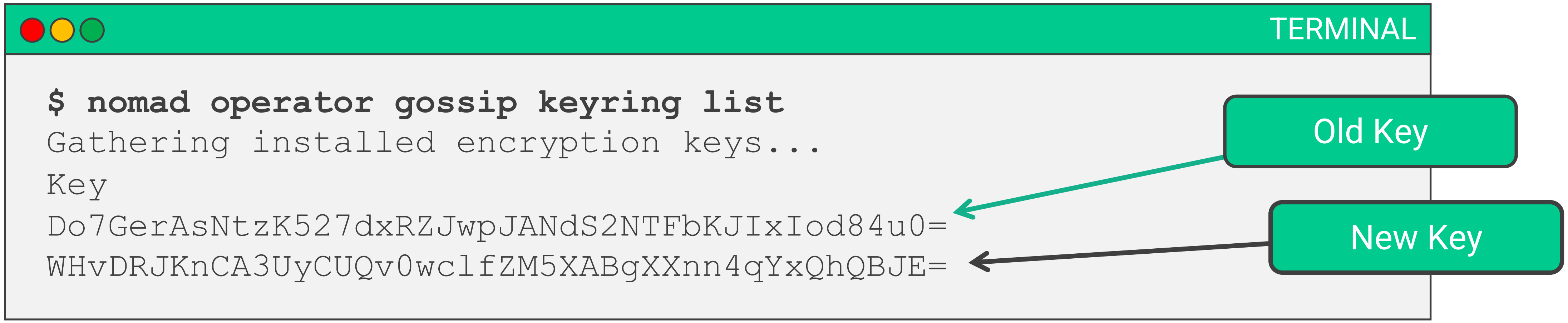
Validate key change in logs

TERMINAL

```
$ journalctl -u nomad
Jan 30 19:31:01 nomad_svr_a nomad[3060]: 2023-01-30T19:31:01.482Z [INFO]  nomad: serf:
Received install-key query
```

View the Gossip Encryption Keys

- View both keys in Nomad



```
$ nomad operator gossip keyring list
Gathering installed encryption keys...
Key
Do7GerAsNtzK527dxRZJwpJANdS2NTFbKJIxIod84u0=
WHvDRJKnCA3UyCUQv0wclfZM5XABgXXnn4qYxQhQBJE=
```

Change the Encryption Key

- Once the key has been installed, you can instruct Nomad to use the key for gossip encryption

Use the `nomad operator gossip keyring use <key>` command

TERMINAL

```
$ nomad operator gossip keyring use WHvDRJKnCA3UyCUQv0wclfZM5XABgXXnn4qYxQhQBJE=
Changing primary gossip encryption key...
```

Validate key change in logs

TERMINAL

```
$ journalctl -u nomad
Jan 30 19:46:44 nomad_svr_a nomad[3060]: 2023-01-30T19:46:44.730Z [INFO]  nomad:
serf: Received use-key query
```

Remove the Old Gossip Encryption Key

- Once the new key is installed, we can easily remove the old key, leaving us with only the new key on the keyring to be used for gossip encryption

Use the **`nomad operator gossip keyring remove <key>`** command

TERMINAL

```
$ nomad operator gossip keyring remove Do7GerAsNtzK527dxRZJwpJANdS2NTFbKJIxIod84u0=
Removing gossip encryption key...
```

DEMO

Rotating the Gossip Encryption Key

Upgrading Nomad to a Newer Version

Upgrading Nomad to a Newer Version

- Nothing says "Day 2 Operations" more than upgrading the cluster ☺
- Nomad supports both **in-place upgrades** by replacing the binary or **rolling updates** by adding new hosts to the cluster and removing the old ones
- Nomad is backward compatible for at least one point release but doesn't test or support upgrades beyond that
- Nomad does NOT support downgrading either

Critical Items for Upgrading

Servers

- When upgrading server nodes, you **MUST** maintain a quorum to ensure the Nomad service is available. Quorum size is based on the number of server agents in the cluster. Upgrade incrementally and verify cluster health at each step.

Clients

- Clients run jobs. Jobs are applications, and applications are important to the business and its customers. Client node upgrades should likely involve disabling allocation eligibility and draining any existing allocations

Order of Operations (Rolling)

- HashiCorp recommends to upgrade servers first since many new client features will not work until servers are upgraded
- Any new features (server and client) are unlikely to work correctly until all nodes in the cluster have been upgraded

1 Add New Servers to the Cluster → **2** Validate Cluster Health → **3** Remove Older Servers from Cluster *gracefully → **4** Validate Cluster Health → **5** Migrate Workloads and Upgrade Clients

Order of Operations (In-Place)

1
Stop Nomad Service
Replace Binary
Server E
(Follower)

→

2
Stop Nomad Service
Replace Binary
Server D
(Follower)

→

3
Stop Nomad Service
Replace Binary
Server C
(Follower)

→

4
Stop Nomad Service
Replace Binary
Server B
(Follower)

→

5
Stop Nomad Service
Replace Binary
Server A
(Leader)

Nomad doesn't currently have an option to tell the cluster leader to gracefully give up leadership, so just stop the service on the Leader node and another node will be elected Leader

Other Notes for Upgrades

Check the Upgrade Guides:

- HashiCorp publishes version-specific upgrade guides that will call out any "gotchas" or issues that you might run into during the upgrade process.

Read the Release Notes

- Make sure to checkout the release notes for each version before upgrading to ensure there aren't any deprecated features or commands that you might be using in your environment

DEMO

Upgrading Nomad to a Newer Version