Nomad Notes

Overview

Nomad is a cluster manager, designed for both long lived services and short lived batch processing workloads. Developers use a declarative job specification to submit work, and Nomad ensures constraints are satisfied and resource utilization is optimized by efficient task packing. Nomad supports all major operating systems and virtualized, containerized, or standalone applications.

Installation

https://www.nomadproject.io/downloads.html

```
wget
https://releases.hashicorp.com/nomad/0.5.2/nomad_0.5.2_linux_amd64.zip
unzip nomad_0.5.2_linux_amd64.zip
rm -f nomad_0.5.2_linux_amd64.zip
mv nomad /usr/bin/nomad
mkdir -p /etc/nomad
mkdir -p /opt/nomad
nomad agent -config /etc/nomad/server.hcl
```

Networking Information

Protocol	Port Number	Description	Role that uses this port
TCP	4646	The port used for internal **RPC** communication between agents and servers, and for inter-server traffic for the consensus algorithm (raft).	Server
TCP	4647	The port used to run the **HTTP** server. The command line client interacts with this to retrieve data.	Client, Server
TCP	4648	The port used for the **SERF** gossip protocol for cluster membership. Both TCP and UDP should be routable between the server nodes on this port.	Server
UDP	4648	The port used for the **SERF** gossip protocol for cluster membership. Both TCP and UDP should be routable between the server nodes on this port.	Server

Nomad Agent Daemon Configuration

Nomad Server v0.5.2

```
log_level  = "INFO"
data_dir   = "/opt/nomad/"
region     = "us-east-1"
bind_addr  = "192.168.33.10"
datacenter = "molecule"
name       = "centos-7-srv1"

# Takes precedence over bind_addr
addresses {
  http = "192.168.33.10"
  rpc  = "192.168.33.10"
  serf = "192.168.33.10"
}

# Takes precedence over bind_addr
advertise {
  # We need to specify our host's IP because we can't
  # advertise 0.0.0.0 to other nodes in our cluster.
  http = "192.168.33.10"
  rpc  = "192.168.33.10"
  serf = "192.168.33.10"
}

server {
    enabled             = true
    bootstrap_expect    = 1
    encrypt             = "mooleecuulmooleecuulmo=="
}

consul {
    address = "192.168.33.10:8500"
}

# Defaults
ports {
    http = "4646"
    rpc  = "4647"
    serf = "4648"
}
```

Nomad Client v0.5.2

https://www.nomadproject.io/docs/agent/configuration/client.html

```
log_level  = "INFO"
data_dir   = "/opt/nomad/"
region     = "us-east-1"
datacenter = "molecule"
name       = "centos-7-cl1"

# Takes precedence over bind_addr
addresses {
  http = "192.168.33.13"
  rpc  = "192.168.33.13"
  serf = "192.168.33.13"
}

client {
    enabled = true
    servers = ["192.168.33.10"]

    reserved {
        cpu             = 512
        memory          = 768
        disk            = 10240
        reserved_ports  = "22,80,443,4646-4648,8500-8600"
    }
}

consul {
    address = "192.168.33.13:8500"
}

# Defaults
ports {
    http = "4646"
    rpc  = "4647"
    serf = "4648"
}
```

Anatomy of a Nomad Job Specification

https://www.nomadproject.io/docs/job-specification/service.html

`job -> group -> task -> service`

Key	Description
Job	The `job` stanza is the top-most configuration option in the job specification. A job is a declarative specification of tasks that Nomad should run. Jobs have a globally unique name, one or many task groups, which are themselves collections of one or many tasks.
Group	The `group` stanza defines a series of tasks that should be co-located on the same Nomad client. Any task within a group will be placed on the same client.

Task	The `task` stanza creates an individual unit of work, such as a Docker container, web application, or batch processing.
Service	The `service` stanza instructs Nomad to register the task as a service using the service discovery integration.

Usage

Display version

```
nomad --version
```

Start up nomad with a config

> The config determines whether or not the node acts as a client or server despite the command being **nomad agent**

```
nomad agent -config /etc/nomad/server.hcl
```

Get status of a node running as a client

```
nomad node-status -self
```

From a client

```
[root@ip-252-25-69-190 nomad]# nomad node-status -self
ID     = 74c4ef9c
Name   = ip-252-25-69-190.internal
Class  = <none>
DC     = dc1
Drain  = false
Status = ready
Uptime = 2158h35m31s

Allocated Resources
CPU         Memory       Disk        IOPS
0/4988 MHz  0 B/3.9 GiB  0 B/38 GiB  0/0

Allocation Resource Utilization
CPU         Memory
0/4988 MHz  0 B/3.9 GiB

Host Resource Utilization
CPU         Memory           Disk
0/4988 MHz  1.2 GiB/3.9 GiB  12 GiB/50 GiB
```

From a server

```
[root@ip-252-25-5-53 nomad]# nomad node-status
-address=http://172.25.5.53:4646
ID        DC   Name                                             Class
Drain  Status
74c4ef9c  dc1  ip-252-25-69-190.internal  <none>  false  ready

[root@centos-7-srv1 ~]# nomad node-status
ID        DC        Name          Class   Drain  Status
96df05b8  molecule  centos-7-cl1  <none>  false  ready
```

Show server-members

```
nomad server-members -address=http://172.25.5.53:4646
nomad server-members
```

```
[root@ip-252-25-5-53 nomad]# nomad server-members
-address=http://172.25.5.53:4646
Name                                                       Address
Port  Status  Leader  Protocol  Build  Datacenter  Region
ip-252-25-5-53.internal.us-east-1  172.25.5.53  4648  alive   true    2
0.4.1  dc1        us-east-1

[root@centos-7-cl1 ~]# nomad server-members
Name                     Address         Port  Status  Leader  Protocol
Build  Datacenter  Region
centos-7-srv1.us-east-1  192.168.33.10  4648  alive   true    2
0.5.1  molecule    us-east-1
```

Testing out nomad from the cli

```
nohup nomad agent -config /etc/nomad/server.hcl &>nomad.log &
tailf nomad.log
```

Validate a nomad job

You should check out the anatomy of a nomad job! https://www.nomadproject.io/docs/jobspec/

```
nomad validate $JOBNAME.nomad
```

```
[root@ip-252-25-5-53 ~]# nomad validate sleep.nomad
Job validation successful
```

This particular job file looks like

```
job "sleepy-job" {
    region = "us-east-1"
    datacenters = ["dc1"]
    type = "batch"

    constraint {
        attribute = "${attr.kernel.name}"
        value = "linux"
    }

    task "gotosleep" {
        driver = "exec"
        config {
            command = "/bin/sleep"
            args = ["1"]
        }
        resources {
            cpu = 200
        }
    }
}
```

On a receiving client, the executing job from above will look as follows. Pay attention to the task name **gotosleep**

```
[root@ip-252-25-69-190 ~]# ps aux | egrep '(sleep|nomad)'
root     12250 59.7  0.5 390768 22164 ?        Ssl  21:07   0:02
/usr/bin/nomad executor
/opt/nomad/alloc/40b04879-5eb0-62b8-088d-fdfb8cdb3b4d/gotosleep/gotosleep-
executor.out
nobody   12273  0.0  0.0 107908   676 ?        S    21:07   0:00 bin/sleep
20
```

Plan and diff a job

This is a very similar concept to Terraform. You'll want to do this prior to running the actual job.

```
[root@ip-252-25-5-53 ~]# nomad plan -diff
-address=http://172.25.5.53:4646 sleep.nomad
```

```
[root@ip-252-25-5-53 ~]# nomad plan -diff -address=http://172.25.5.53:4646
sleep.nomad
[root@centos-7-srv1 ~]# nomad plan example.nomad
+ Job: "sleepy-job"
+ Task Group: "gotosleep" (1 create)
  + Task: "gotosleep" (forces create)

Scheduler dry-run:
- All tasks successfully allocated.

Job Modify Index: 0
To submit the job with version verification run:

nomad run -check-index 0 sleep.nomad

When running the job with the check-index flag, the job will only be run if
the
server side version matches the job modify index returned. If the index has
changed, another user has modified the job and the plan's results are
potentially invalid.
```

Show previously run jobs and their status

```
nomad status
nomad status $JOBID
nomad inspect $JOBID
```

```
[root@ip-252-25-5-53 ~]# nomad status
ID          Type   Priority  Status
sleepy-job  batch  50        dead
writer      batch  50        dead

[root@ip-252-25-5-53 ~]# nomad status writer
ID          = writer
Name        = writer
Type        = batch
Priority    = 50
Datacenters = dc1
Status      = dead
Periodic    = false

Summary
Task Group  Queued  Starting  Running  Failed  Complete  Lost
echo        0       0         0        1       1         0

Allocations
ID        Eval ID   Node ID   Task Group  Desired  Status    Created At
2e7cd21f  f60c1d1f  74c4ef9c  echo        run      failed    10/18/16
21:57:01 EDT
188def87  742d43aa  74c4ef9c  echo        stop     complete  10/18/16
21:54:28 EDT
[root@ip-252-25-5-53 ~]# nomad status sleepy-job
ID          = sleepy-job
Name        = sleepy-job
Type        = batch
Priority    = 50
Datacenters = dc1
Status      = dead
Periodic    = false

Summary
Task Group  Queued  Starting  Running  Failed  Complete  Lost
gotosleep   0       0         0        0       3         0

Allocations
ID        Eval ID   Node ID   Task Group  Desired  Status    Created At
40b04879  e5ccc8c8  74c4ef9c  gotosleep   run      complete  10/18/16
21:07:40 EDT
15752502  ff78cb88  74c4ef9c  gotosleep   stop     complete  10/18/16
21:06:41 EDT
d31db0b0  ecc0aa67  74c4ef9c  gotosleep   stop     complete  10/18/16
21:05:29 EDT
```

Troubleshooting

Resource pool members "Down"

If you are seeing the following message on a client

```
$ cat /var/log/nomad/nomad.log
==> Error starting agent: client setup failed: failed to restore state: 2
error(s) occurred:

* failed to decode state: unexpected end of JSON input
* failed to decode state: unexpected end of JSON input
    Loaded configuration from /etc/nomad/client.hcl
==> Starting Nomad agent...
==> Error starting agent: client setup failed: failed to restore state: 2
error(s) occurred:

* failed to decode state: unexpected end of JSON input
* failed to decode state: unexpected end of JSON input
    Loaded configuration from /etc/nomad/client.hcl
==> Starting Nomad agent...
```

and you are seeing the following message on a server

```
[root@centos-7-srv1 jobs]# nomad node-status
ID        DC        Name          Class   Drain  Status
73cf3fc7  molecule  centos-7-cl3  <none>  false  down
e3496454  molecule  centos-7-cl1  <none>  false  down
a42fd16a  molecule  centos-7-cl2  <none>  false  ready
```

You should then run the following commands on the affected clients

```
systemctl stop nomad
rm -rf /opt/nomad/client/alloc/*
rm -rf /opt/nomad/alloc/*
systemctl restart nomad
```

and watch the server for the updated node-status

```
[root@centos-7-srv1 jobs]# nomad node-status
ID        DC        Name          Class   Drain  Status
73cf3fc7  molecule  centos-7-cl3  <none>  false  down
e3496454  molecule  centos-7-cl1  <none>  false  down
a42fd16a  molecule  centos-7-cl2  <none>  false  ready

[root@centos-7-srv1 jobs]# nomad node-status
ID        DC        Name          Class   Drain  Status
73cf3fc7  molecule  centos-7-cl3  <none>  false  initializing
e3496454  molecule  centos-7-cl1  <none>  false  initializing
a42fd16a  molecule  centos-7-cl2  <none>  false  ready

[root@centos-7-srv1 jobs]# nomad node-status
ID        DC        Name          Class   Drain  Status
73cf3fc7  molecule  centos-7-cl3  <none>  false  ready
e3496454  molecule  centos-7-cl1  <none>  false  ready
a42fd16a  molecule  centos-7-cl2  <none>  false  ready
```

References

https://github.com/hashicorp/nomad

https://www.nomadproject.io/

https://www.nomadproject.io/docs/job-specification/service.html