Introduction to Cluster Scheduler Nomad

Easily Deploy Applications at Any Scale

2018

SHLUG

Me

- Su Yan
- Pure FP, Type Theory
- Infrastructure

Goal

Goal

- Cluster Scheduler Concepts
- Nomad as an alternative to k8s/mesos

Overview & History

What's Cluster Scheduler

- a tool for managing a cluster of machines and running applications on them

Why use it

- Hard to change things in production
- Snowflakes in production
    - run 30 instances of app Foo
    - run only 1 instance of app Bar (problematic otherwise)
    - run X & Y together
    - ad-hoc scripts (rendering configs, monitoring)
- Resource Utilization - multiple apps per server

Hype Driven Architecture

- Monoliths - high application complexity
- Microservice - high operational complexity

History

- Google Borg (Omega)
- Tupperware (Facebook)
- Apache Mesos (2010) (UC Berkely, Twitter)
- Apache Aurora (2010 , OSS 2013) (Twitter)
- Mesosphere,YARN, ECS, Rancher, Nomad, Kubernetes...

Nomad

Nomad

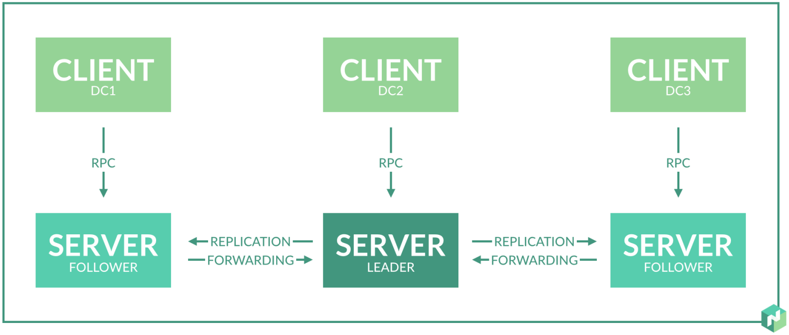
- Delcarative jobs
- Infrasturcture as Code
- Consul & Vault Integration
- ACL
- Operationally Simple & Scalable
    - Single Binary
    - No Dependencies
    - Highly Avaiable
    - Multi-DC/Region Support

Features

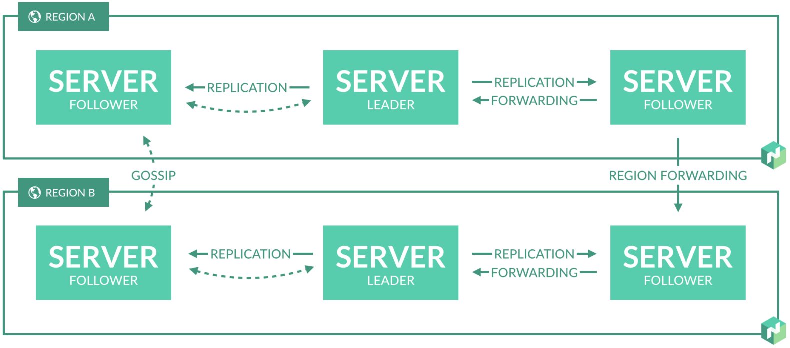
- Rolling updates, rollbacks
- Blue/Green depoyments
- Automatic failure handling
- Fexible workloads
    - linux, windows, mac
    - container (docker, rkt, lxc), VM, raw process
    - service, batch (spark)

Architecture

Single Region



Global


REGION A
SERVER
FOLLOWER
REPLICATION
SERVER
LEADER
REPLICATION
FORWARDING
SERVER
FOLLOWER
GOSSIP
REGION FORWARDING
REGION B
SERVER
FOLLOWER
REPLICATION
SERVER
LEADER
REPLICATION
FORWARDING
SERVER
FOLLOWER

Job specification

Job Spec

```
job
  \_ group[]
        \_ task[]
```

- HCL or JSON
- Job $->$ Task Group[] $->$ Task[]
- Job: datacenters, region, type(service/batch/system), update/deployment strategy, priority, constraints, meta, ...
- Task Group: count, meta, ...
- Task: driver, config, resources, logs, meta, ...
- Resources: cpu, disk, iops, memory, network (ports, mbits)

Job Spec - Job & Group

```
job "docs" {
  region = "us"
  datacenters = ["us-west-1", "us-east-1"]
  type = "service"
  update {
    stagger      = "30s"
    max_parallel = 2
  }
  group "webs" {
    count = 5
    task "frontend" {...}
  }
}
```

Job Spec - Task (Docker)

```
task "frontend" {
  driver = "docker"
  config {
    image = "hashicorp/web-frontend"
  }
  ...
}
```

Job Spec - Service Registration (Consul)

```
task "frontend" {
  ...
  service {
    port = "http"
    check {
      type     = "http"
      path     = "/health"
      interval = "10s"
      timeout  = "2s"
    }
  }
}
```

Job Spec - Environment Variable

```
task "frontend" {
  ...
  env {
    "DB_HOST" = "db01.example.com"
    "DB_USER" = "web"
    "DB_PASS" = "loremipsum"
  }
  ...
}
```

Job Spec - Resource

```
task "frontend" {
  ...
  resources {
    network {
      mbits = 100
      port "http" {}
      port "https" {
        static = 443
      }
    }
  }
  ...
}
```

Runtime Environment Variable

- `NOMAD_META_{key} = {value}`
- `NOMAD_CPU_LIMIT`
- `NOMAD_MEMORY_LIMIT`
- `NOMAD_IP`
- `NOMAD_PORT_{label}`

Job Spec

- artifact
- check_restart
- constraint
- dispatch_payload
- env
- ephemeral_disk
- group
- job
- logs
- meta
- network
- parameterized
- periodic
- resources
- restart
- service
- task
- template
- update
- vault

Security

- TLS
- Vault integration
- ACL
- Sentinel (policy as code) (paid)

API

- /v1/acl
- /v1/jobs
- /v1/nodes
- /v1/allocations
- /v1/evaluations
- /v1/deployments
- /v1/agent/...
- /v1/status

Future

- Stateful Application*
- Volume Plugin (CSI)
- Network Plugin (CNI)
- Priorities (preemption)
- Quotas

Web UI

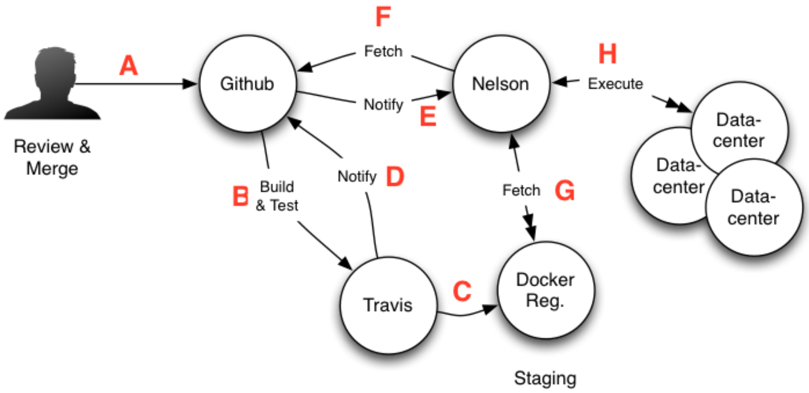
built-in (last year)

demo: https://demo.nomadproject.io/

Integration

- Serverless (OpenFaaS plugin)
- Spark (fork), Apache Heron
- Nelson (workflow system) (getnelson.github.io/nelson)
  - Fully integrated with GitHub or GitHub Enterprise.
  - Developer-driven, automated build & release workflow revisioned as code.
  - Can deploy applications to any number of datacenters.
  - State of the art runtime routing via Envoy.
  - Integrated support for alert definition and propagation via Prometheus.
  - Utilizes secure introduction for safe distribution of credentials from Vault.

Nelson


F
Fetch
A
Github
H
Nelson
Execute
Notify
E
Review &
Merge
Data-
center
Data-
center
Data-
center
Build
& Test
B
Notify
D
Fetch
G
C
Travis
Docker
Reg.
Staging

Experiences

Start small/simple

Servers

- 3 node
- run nomad, consul & vault server

Workers

- provision nomad and consul agent
- consul agent need 1 consul server address to bootstrap
- nomad connect to servers via consul

Experiences

Consul DNS

- do use for trivial use cases
- more sophisticated solutions
  - istio/envoy/linkerd
  - fat client libraries

Server upgrade

- rolling upgrade
- make sure server stand up before starting next one

Security

Use ACL, TLS, Vault from the very beginning

Experiences

- Run bigger and less servers, easier to do experimentation
- Code review!
- Survive mandatory security update in public cloud
- Portable (k8s, mesos, etc.)

Thanks

Questions?