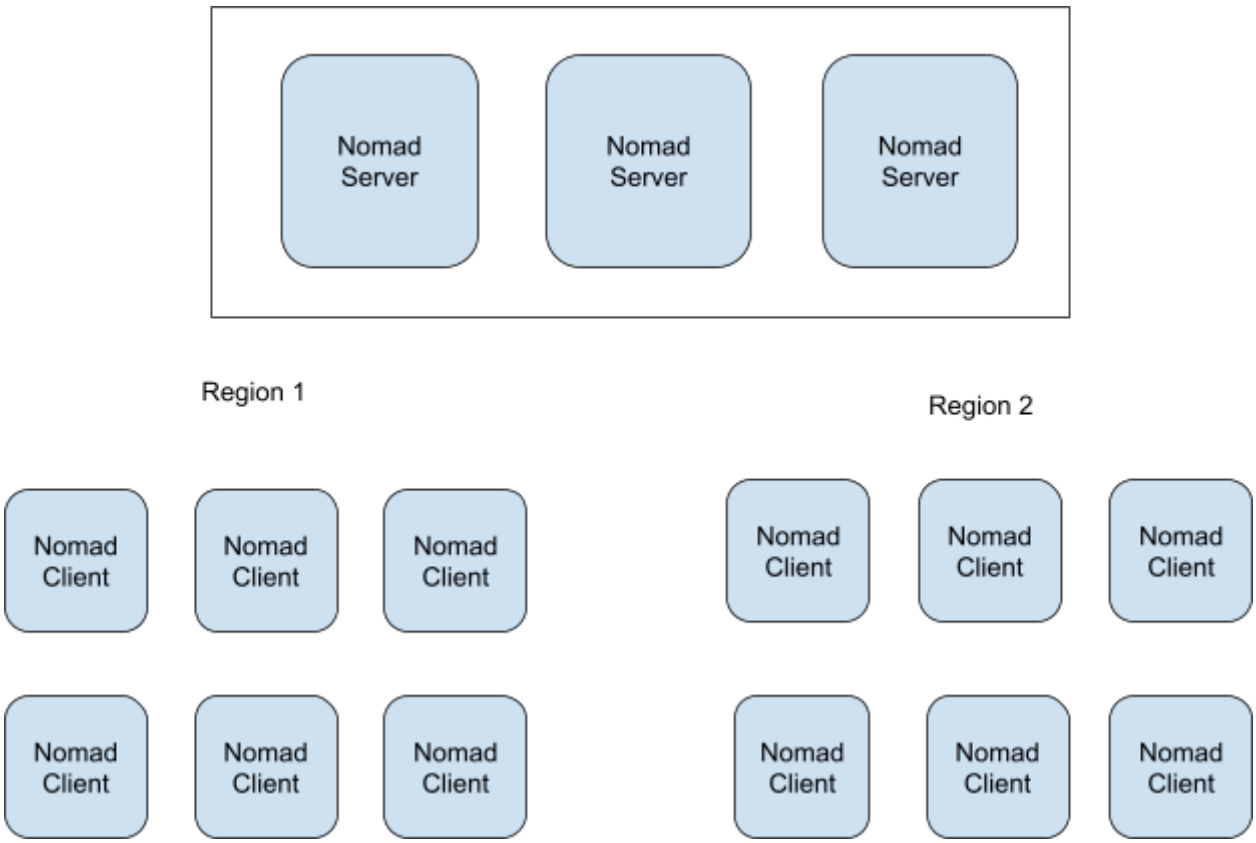
System Architecture



Multi Region Nomad Cluster

IO.net has decentralized compute suppliers. They can have following persona:

- Layman user - Gamer or Graphic designer who own GPU Laptops and PCs. They are not savvy in coding or any technical setups.
- Mining Data Center - web3 coin mining sites who own entire data centers of GPUs. They have technical capability to understand setups but need it to be scaled across their entire datacenter

In this condition we want the suppliers to run some software on their devices and that should initiate their devices to be used by io.net platform while they're being compensated for the jobs their devices were used for. IO.net is supposed to manage a Dynamic resource pool where computers join and leave the platform as they like.

Requirements

- Regions should be fault isolated
- Fast Scheduling
- Single region deployments should function
- Minimal additional overhead

System Components

Nomad Cluster

- No cluster of clusters
- One cluster must be able to talk to another cluster (federation)
- Data is not replicated across clusters (loose federation)
- Use consul for service mesh

Nomad Servers

Server Configuration

- Security : Enable TLS
- Machine Requirements : 4-8+ cores, 16-32 GB+ of memory, 40-80 GB+ of fast disk and significant network bandwidth
- High Availability : Should have 3 or 5 nomad servers

Why choose 3 or 5 Nomad servers and not 7,9,11 or more ?

Nomad servers use rat protocol for consensus . In raft all communications have to be serialized through the leader . Nomad follows optimistic concurrency . Leader has to replicate the raft logs to a quorum of followers .Every log entry needs to be replicated to a quorum of follower nodes . Hence the network bandwidth requirements will increase linearly with the number of nomad servers .
Terminology :

Broadcast time → Time it takes to commit log entries to a quorum of servers
Election timeout → TIme required to select a new leader
MTBF -> Mean time between failures

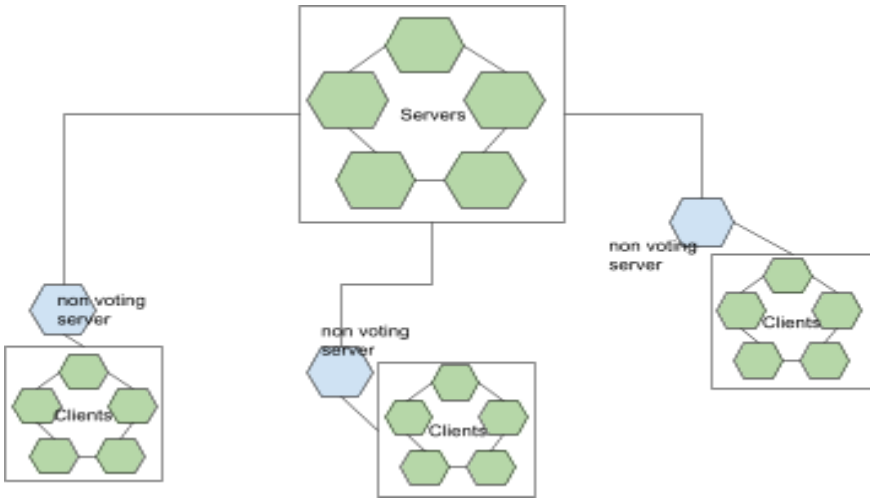
Timeout requirements for RAFT (settings)

- **Broadcast time << Election timeout << MTBF**

Nomad servers in a region need minimal latency between each other , hence the servers in a particular region need to be close to each other .

Scenario 1 (Non voting servers can be placed close to clients)

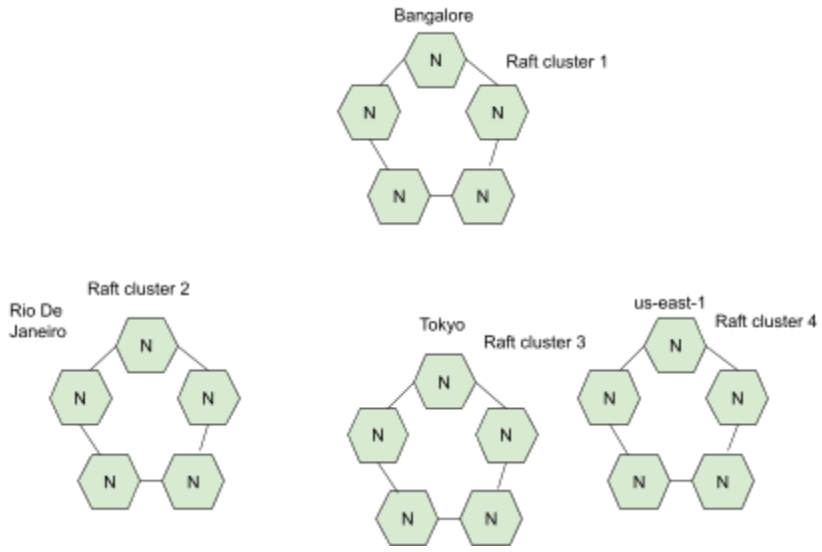
Non voting servers can be placed away from the voting servers . Non Voting servers can act as local proxy to clients in their network and propagate information to the voting servers . The voting servers in one region as mentioned need to be close to each other but that could be a cause for a single point of failure .



Scenario 2 (HA availability of Servers, with region isolation)

To avoid single point of failures isolated regions with each having its own set of nomad servers . Since each cluster will have its own raft cluster if something goes wrong in one region it does not affect the other .

Disadvantages of using this approach is maintenance and administration overhead . Secure (mtls) connections needed for each region as well as for CICD systems . Also it requires Separate ACL’s .



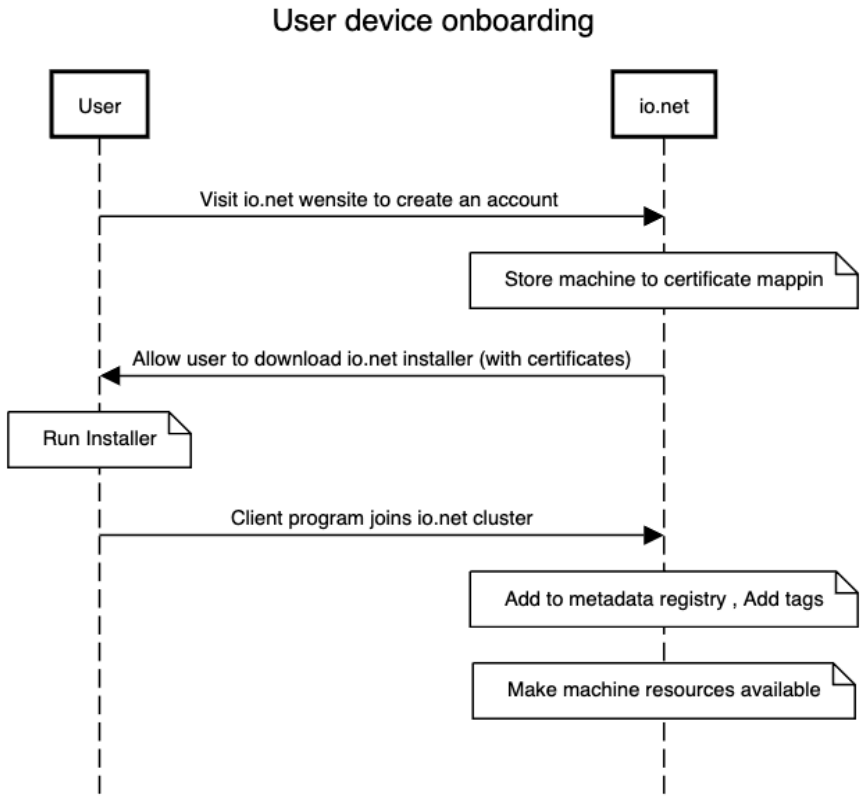
Scenario 2 (Federated clusters with region isolation)

Message passing enabled between clusters , user can interact with one cluster and corresponding messages intended for clusters will be relayed by that cluster . One set of ACL's and one secure connection to a cluster needs to be setup

Nomad Clients

Nomad clients can be geographically distributed . Nomad client machines can have their own certificates which they can use to authenticate themselves with the nomad servers

Provisioning a VM to the Nomad io.net cloud



The VM owner needs to have a certificate from io.net to be able to run a program on its own device to add itself to the io.net cloud . When a new node comes online fingerprint them to possibly roll out drivers if they are attached

1) Add necessary TAGS
    a) Tags specific to hardware
    b) Tags specific to OS
    c) etc etc

- BareMetal OnPrem
    - Install Nomad Binary
    - Allow ports on firewall
    - Enable TLS (client certificate)
    - Create appropriate folders/structures necessary for nomad
    - Also ensure file/folder permissions are as per desired user and group

- Windows VM

```
# Set-NetFirewallRule -Name
#Create Directory for Nomad
#Download Nomad
#Extract File
#Nomad Configuration File
New-Item C:\WorkSpace\Nomad\client.hcl
#Create Service for Nomad
#Start Nomad Services
```

- Linux VM

```
# Set-NetFirewallRule -Name
#Create Directory for Nomad
#Download Nomad
#Extract File
#Nomad Configuration File
New-Item C:\WorkSpace\Nomad\client.hcl
#Create Service for Nomad
#Start Nomad Services
```

Networking

RPC connections in Nomad

Within region

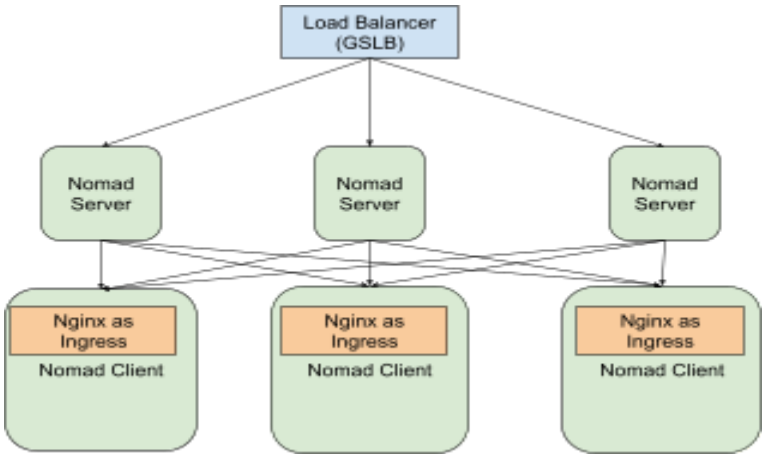
- Server Follower to Server leader
- Server to Client

Across region

WAN gossip protocol used in large multi region setup

- Region to region

External Endpoints



External Load Balancer (Traffic to Nomad Servers)

Nomad Servers fronted by GSLB (global scale load balancer) , having IP addresses of all the nomad server nodes . Traffic intended for master nodes (Ex: job submission, cluster status) can be routed through the GSLB load balancer . Traffic for **nomad services** can be routed through the Nginx proxy running as a task on the clients .

Internal Load Balancer (Traffic to Nomad Services i.e Ingress)

Nginx writes a new configuration dynamically when a Nomad service changes its config . The Nginx service can then use the new config to route traffic to new services .
Sample Nginx Config :

```
job "nginx" {
  datacenters = ["dc1"]

  group "nginx" {
    count = 1

    network {
      port "http" {
        static = 8080
      }
    }

    service {
      name = "nginx"
      port = "http"
    }

    task "nginx" {
      driver = "docker"

      config {
        image = "nginx"

        ports = ["http"]

        volumes = [
          "local:/etc/nginx/conf.d",
```

```
      ]
    }

    template {
      data = <<EOF
upstream backend {
{{ range service "demo-webapp" }}
  server {{ .Address }}:{{ .Port }};
{{ else }}server 127.0.0.1:65535; # force a 502
{{ end }}
}

server {
  listen 8080;

  location / {
    proxy_pass http://backend;
  }
}
EOF

      destination   = "local/load-balancer.conf"
      change_mode   = "signal"
      change_signal = "SIGHUP"
    }
  }
 }
}
```

Important Metrics

- Ingress response grouped by response codes
- Ingress log files
- Number of instances
- Load on each ingress (request/second)

Config

Config can be stored in lower cost storages like S3 or Google/Azure Cloud buckets

Jobs

Sequence diagram for submitting a job

Security

Require TLS

```
tls {
 http = true
 rpc  = true

 ca_file   = "nomad-agent-ca.pem"
 cert_file = "global-server-nomad.pem"
```

```
 key_file  = "global-server-nomad-key.pem"
 verify_server_hostname = true
 verify_https_client    = true
}
```

verify_server_hostname = true
verify_https_client   = true

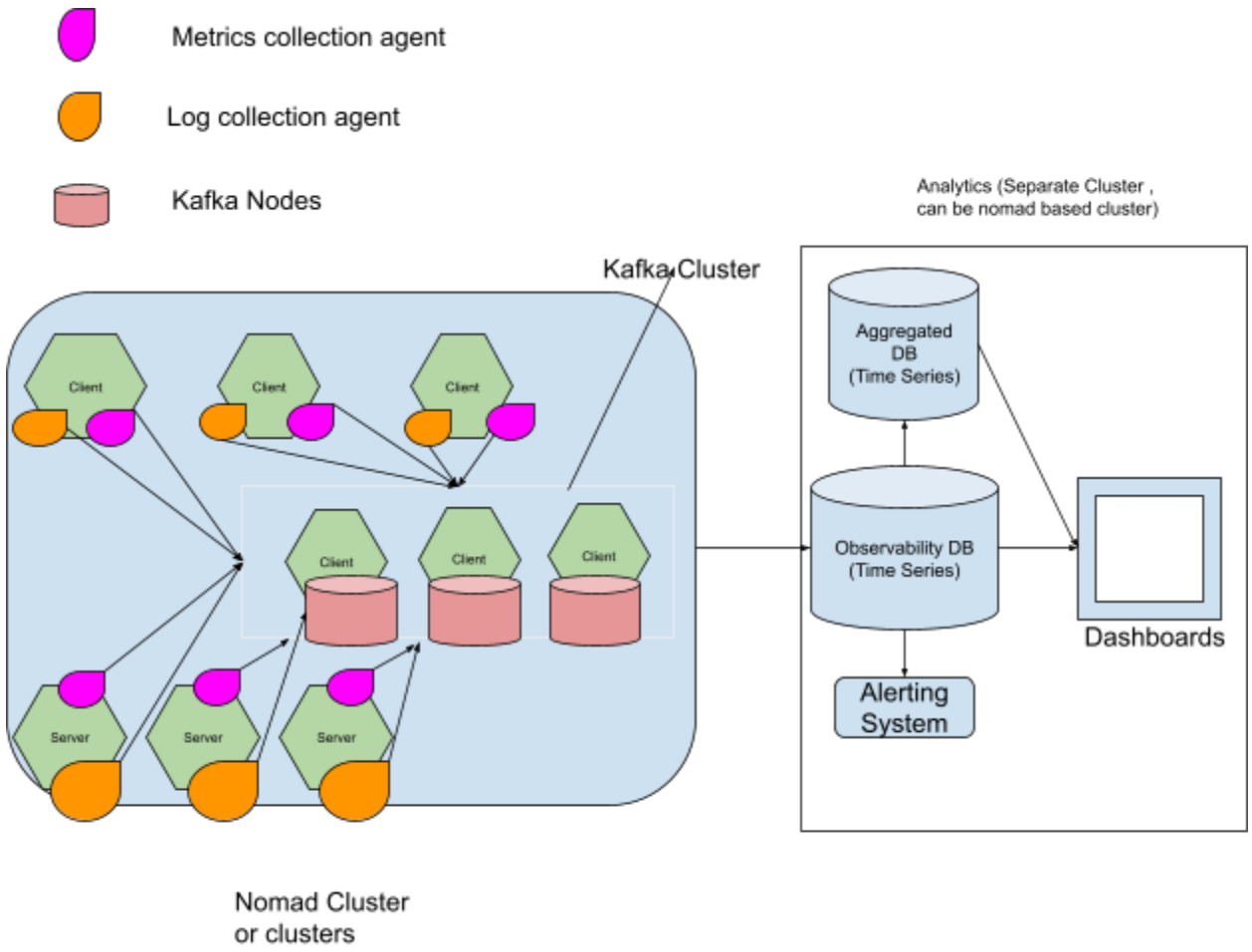
These two settings are important for ensuring all of Nomad's mTLS security properties are met. If verify_server_hostname is set to false the node's certificate will be checked to ensure it is signed by the same CA, but its role and region will not be verified. This means any service with a certificate signed by same CA as Nomad can act as a client or server of any region.

Using federated ACL's for multi region deployment .
One authoritative region and this region's ACLs get replicated to other regions .

Using Hashicorp Vault .

Observability

Collecting Observability Data for analysis



Agents —--> Kafka Cluster (Part of Nomad Cluster) —--> Time Series Store (Out Of nomad cluster for which data is collected)

Metrics

The Nomad log collection jobs must be run as **system services** .
Possible agents are Prometheus , metricbeat, telegraf etc .
Metrics must be shipped to cluster outside of Nomad (ELK, Grafana, SumoLogic)

Logs

The Nomad log collection jobs must be run as **system services** .
Possible agents are vector , filebeat , fluent bit , fluentd etc .
Logs must be shipped to cluster outside of Nomad (ELK, Grafana, SumoLogic)

Alerting Systems

Can be used to self heal the nomad cluster and for sending notifications , emails and alerts

Collecting Copy of Observability Data to store logs and metrics in S3

This way if the agents fail to send data or collect then a copy can be retrieved if necessary

- Linux : - **logrotate** can be used on machines to zip logs and send to s3 or low cost storage periodically
- Windows - **LogRotateWin** can be used on machines to zip logs and send to s3 or low cost storage periodically
- Mac : - **logrotate** can be installed via homebrew

Important Metrics

Nomad Jobs/Tasks application Health

- Log Events per job (count (filter by job name))
- Allocations per job (count)
- Total jobs running (unique job name)
- Nomad Job log errors (nomad task logs ex: stdout/stderr task logs , grep for ERROR (type) fields)

Nomad Service Health

- Log events with time (count (log entries))
- Nomad service log errors (service logs ex: journald service logs , /var/log etc)
- Memory Utilization (directly proportional to runtime_alloc_bytes metric)
- Load Pressure (directly proportional to number of goroutines metric)
- Memory Pressure (directly proportional to number of heap objects metric)

Deployment

- Update and version deployments
- Use blue green , rolling or canary deployment strategies

Remote Management & Dynamic Resource Pools

Task Drivers can be written for specific systems and their services . For example, to manage a Linux systemd driven machine , a systemd driver can be used . Similarly to remotely manage a windows machine a windows driver can be used . A task driver for OS type and devices will be necessary to manage and manipulate a wide variety of machine types and devices .

List of Machine Types List of package manager

Machine Type	OS type	Package manager	Device Type	Nomad Driver

Resource Pools need to be properly tagged . So that a correct match for the submitted jobs may be used optimally . Bridged mode networks can only be placed in Linux machines as all other OS systems only support host mode networking . Proper bifurcation among resources available is required for efficient and correct scheduling .

Testing

- Unit Testing (autopush)
- Functional Testing (autopush)
- Integration Testing (staging)
- Load Testing (staging)

Privacy

Costs and Estimates

Rollout Plan