QCLUG
Consul by HashiCorp

Service Mesh (AKA Service Discovery)

Made faithfully in
LibreOffice and
Presented by
Aaron Johnson

Introduction

- What is Consul?
  - Consul is a distributed, highly available, and data center aware solution to connect and configure applications across dynamic, distributed infrastructure.
  - Consul is a tool for service discovery and configuration. Consul is distributed, highly available, and extremely scalable.
  - Consul is a distributed service mesh to connect, secure, and configure services across any runtime platform and public or private cloud

Introduction continued...

- What does that mean?
  - Simply put it means that you can put service information into consul and get service information out of consul
  - It also has a distributed key/value store that you can use as well but this is not the primary purpose of consul
  - Also IMO consul exists primarily as a supporting tool for other HashiCorp tools such as Nomad and Vault

Installation

- Note: The consul specific steps are *mostly* taken from the getting started installation steps located here:

  https://learn.hashicorp.com/consul/getting-started/install

- Create 3 VMs

  ```
  # yum update -y && yum install -y bind-utils unzip wget vim
  ```

  ```
  # cd && wget
  https://releases.hashicorp.com/consul/1.5.0/consul_1.5.0_linux_amd64
  .zip
  ```

- Since consul is a single binary simply download and extract the zip file into /usr/local/bin

  ```
  # cd /usr/local/bin && unzip ~/consul*.zip
  ```

consul agent

- The Consul agent can be ran in either server or client mode.
  - For production 3-5 servers is recommended to avoid dataloss/outage
  - All other agents will run in client mode which will:
    - Register "services"
    - Run health checks
    - Forward queries to consul servers
  - For testing purposes you can start the consul agent "server" in development mode (no redundancy and less configuration required)

```
# consul agent -dev
```

  - If you run `consul members` to see a list of members of the cluster only one will be displayed currently

consul agent with systemd

- The default installation does not define how to start/stop consul.
- If deploying consul on traditional VMs one option would be to use systemd

```
[Unit]
Description=Consul Daemon
After=network.target

[Service]
User=root
Group=root
ExecStart=/usr/local/bin/consul agent -config-dir=/etc/consul.d

[Install]
WantedBy=multi-user.target
```

Interfaces

- HTTP API
  - https://www.consul.io/api/index.html
  - Note: The output of the `consul members` command is eventually consistent due to it using the gossip protocol...
  - https://www.consul.io/docs/internals/gossip.html
  - You can actually get a strongly consistent view of the cluster members using the HTTP API instead

```
# curl localhost:8500/v1/catalog/nodes
```

- DNS interface
  - https://www.consul.io/docs/agent/dns.html
  - Consul can be queried using the DNS protocol on port 8600

```
# dig @127.0.0.1 -p 8600 consul-2gb-nbg1-1
```

- Web UI
  - Listens on port 8500 if you pass `-ui` via the cli or `ui: true` in consul.json

Defining a Service

```
# mkdir /etc/consul.d
# vim /etc/consul.d/web.json
{
  "service": {
    "name": "web",
    "tags": ["rails"],
    "port": 80
  }
}
```

- Restart the development agent like this:

  ```
  # /usr/local/bin/consul agent -dev -config-dir=/etc/consul.d
  ```

- Query the service using the DNS interface (Both A and SRV records are available)

  ```
  # dig @127.0.0.1 -p 8600 web.service.consul
  # dig @127.0.0.1 -p 8600 web.service.consul SRV
  ```

- The SRV record tells you what port the service is listening on (80 in this case)
- You can even use the DNS interface to filter by tags:

  ```
  # dig @127.0.0.1 -p 8600 rails.web.service.consul
  ```

- Or you can query the service using the HTTP API

  ```
  # curl http://localhost:8500/v1/catalog/service/web
  ```

consul server + client

- Skip the section on consul Connect and Intentions...
- Configure the first node as a server:

```
# mkdir -p /var/lib/consul/data
# vim /etc/consul.d/consul.json
```

```
{
  "datacenter": "production",
  "server": true,
  "ui": true,
  "bootstrap_expect": 1,
  "bind_addr": "159.69.4.237",
  "enable_script_checks": true,
  "data_dir": "/var/lib/consul/data"
}
```

- Start the consul agent in server mode

```
# consul agent -config-dir=/etc/consul.d
```

consul server + client

- Configure the second node as a client:
  - Install consul
  - Ensure the second node hostname is unique or override with `node_name` in consul.json

```
  # mkdir /etc/consul.d
  # mkdir -p /var/lib/consul/data
  # vim /etc/consul.d/consul.json
{
  "datacenter": "production",
  "server": false,
  "bind_addr": "195.201.118.249",
  "enable_script_checks": true,
  "data_dir": "/var/lib/consul/data",
  "retry_join": ["159.69.4.237"]
}
```

- Start the consul agent in client mode (Note: `retry_join` will make the agent auto join)

```
  # consul agent -config-dir=/etc/consul.d
```

- Running `consul members` should now show both nodes when ran from either node

multi-server consul cluster

- If you value your data you should follow these steps, but rather than 2 servers deploy at least 3...
- Set up a server identical to the first server except set `bootstrap_expect` to `2` on both servers (or for however many servers as you expect to have...)
- Also you should add `retry_join` to each server as well

```
# vim /etc/consul.d/consul.json

{
  "datacenter": "production",
  "server": true,
  "ui": true,
  "bootstrap_expect": 2,
  "bind_addr": "159.69.4.237",
  "enable_script_checks": true,
  "data_dir": "/var/lib/consul/data",
  "retry_join": ["159.69.4.237"]
}
```

- Start the consul agent in server mode

```
# consul agent -config-dir=/etc/consul.d
```

- Now all three nodes should be listed in `consul members` command output

Next Steps

- Service discovery is great but how should I use this?
  - Monitoring (prometheus – future meeting topic?)
  - Service publishing via kubernetes or HashiCorp Nomad?
  - Elimination of application DNS dependency, however if you still need DNS for your application consul has you covered.
  - Also HashiCorp vault uses consul’s key/value store so if you plan to use vault you may want consul.

Questions