KPLABS Course

HashiCorp Certified: Consul Associate

Getting Started with Consul

ISSUED BY

Zeal

REPRESENTATIVE

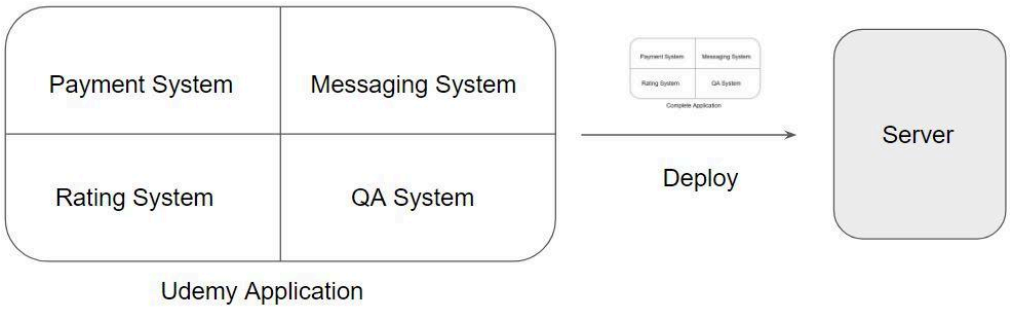
instructors@kplabs.in

Module 1: Introduction to Consul

1.1 Monolithic Approach

In the Monolithic approach, there is a single application that gets deployed.

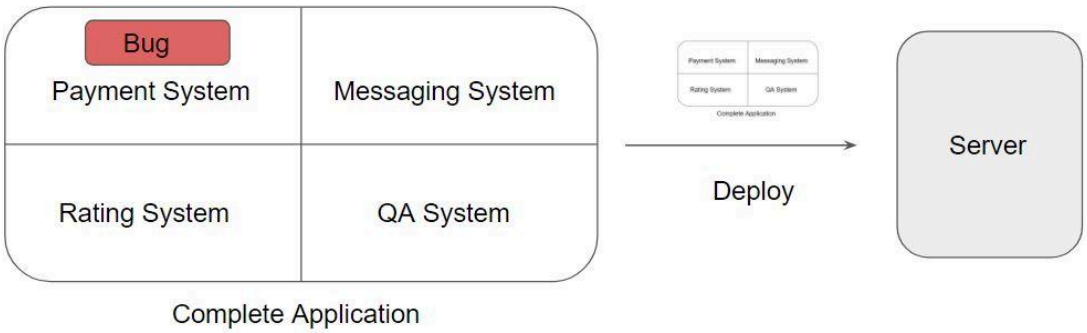
This single application in-turn has multiple sub-components.



1.2 Challenges with Monolithic Approach

If there is a bug in any one of the sub-components, we have to re-deploy the entire application.

We cannot just fix the bug in the “Payment system” and deploy it.



1.3 Disadvantage of Monolithic Based Architecture

The application and codebase size increases over time and it becomes difficult to manage.

The whole application needs to be deployed as a package even for small changes.

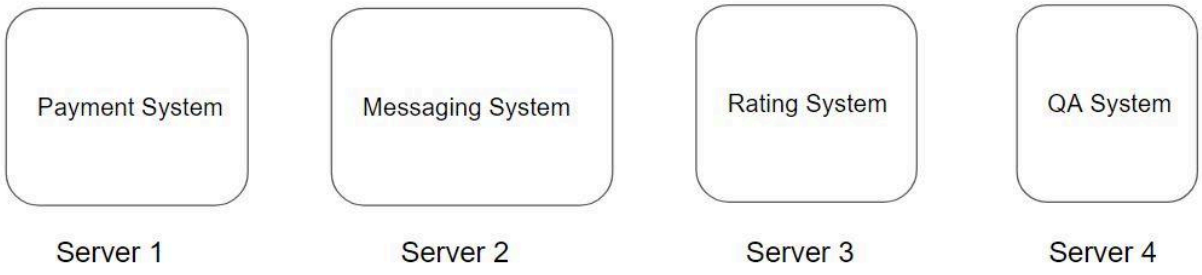
Even if a single part of the application is receiving huge traffic, we need to deploy the whole application.

A single bug can affect the whole system.

1.4 Microservice-based Architecture

In this architecture, “complete application” is divided into set of individual services.

Each service can independently be deployed as well as scaled.



1.5 Overview of Consul

Consul is a service mesh solution providing a full-featured control plane with service discovery, configuration, and segmentation functionality

1.6 Feature 1 - Service Discovery

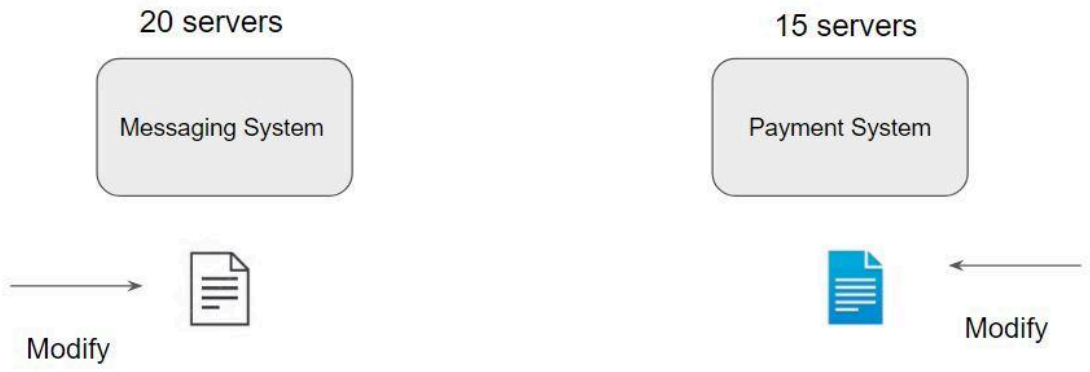
Service Discovery is the way in which microservices can locate each other on the network.

1.7 Feature 2 - Health Checks

Consul with health check functionality will only keep the healthy instances related data within its database.

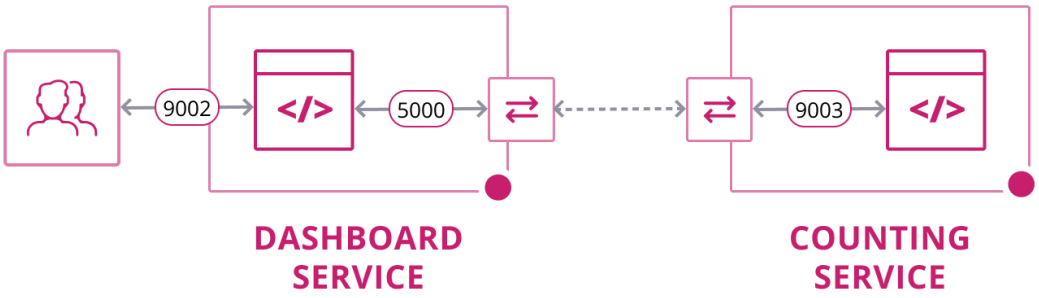
1.8 Feature 3 - KV Store

Key-Value store is generally used for storing the service configuration and other meta-data.



1.9 Feature 4 - Secure Service Communication

Sidecar proxies can be used to automatically establish TLS connections for inbound and outbound connections.



Module 2: Our Lab Architecture

2.1 Overview of Installation Process

Consul installation is very simple.

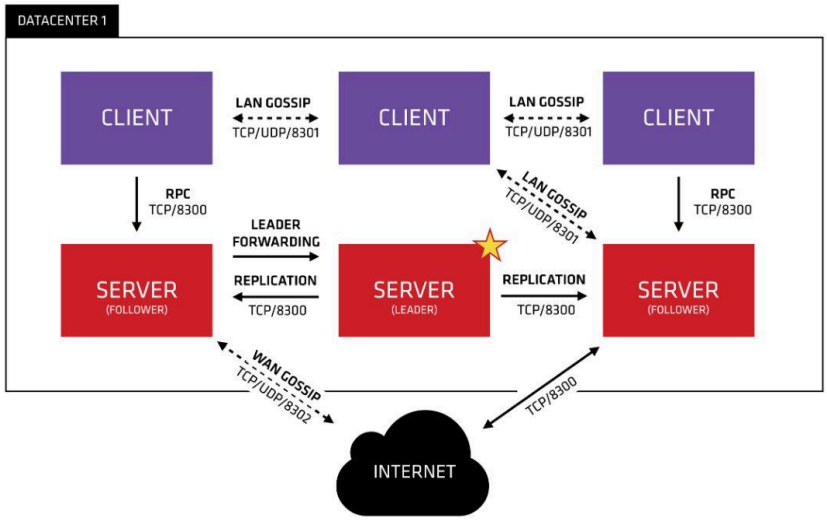
You have a single binary file, download and use it.

There are many amazing features that we need to practically try it out.

Some of these includes:

- Service Discovery
- Health Checks
- Service Mesh
- Secure Service Communication (Mutual TLS)

For these, we will need multiple instances of consul running.



2.2 Selecting a Provider

For our labs, we will be making use of the following configuration:

Operating System	Cloud Provider
Linux (CentOS 8)	Digital Ocean

You can use any Cloud provider as you like.

2.3 Why Digital Ocean for our Labs?

Following ar some of the benefits of the Digital Ocean:

Reasonably priced Servers ($5/month) - pay per hour

Good Amount of Credits for New Users (Referral) - $100

Keep it simple approach.

Module 4: Installing Consul In Linux

Consul installation is very simple.

You have a single binary file, download and use it.

Consul works on multiple platforms, primary ones include:

- Windows
- macOS
- Linux

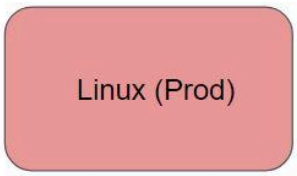
Module 4: Dev Agent Mode

3.1 Learning Approach

Whenever you learn any technology, you start from the very basics and keep it simple initially.

Once you are comfortable with the basics, you go more in-depth about aspects like security, high-availability, and others.

1. Basic Linux Commands
2. Installing Linux (basic level)
3. Understanding filesystem.



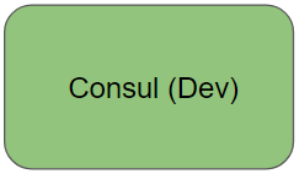
1. Server Hardening
2. Kernel Tuning
3. HIDS / IPS

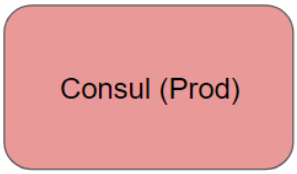
3.2 Overview of Consul Dev mode

The Dev agent mode in Consul is useful for local development, testing, and exploration.

Not very secure.

In-memory mode



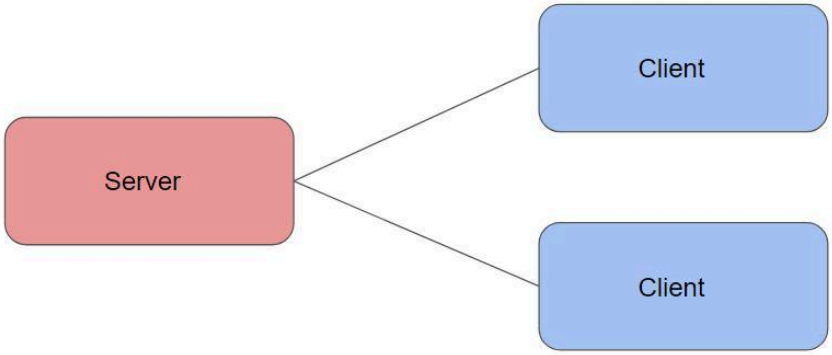


Module 5: Consul Architecture

5.1 10,000 Feet Overview

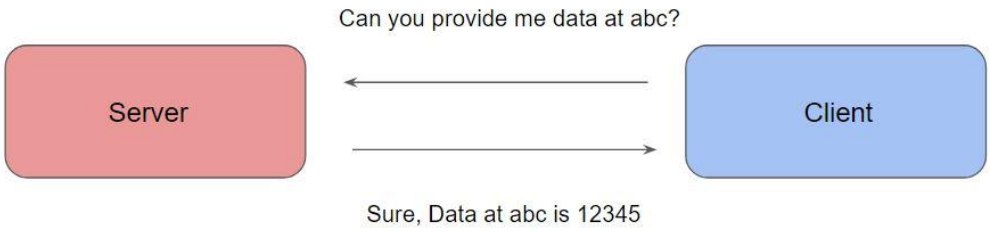
There are two primary components:

- Consul Server
- Consul Client



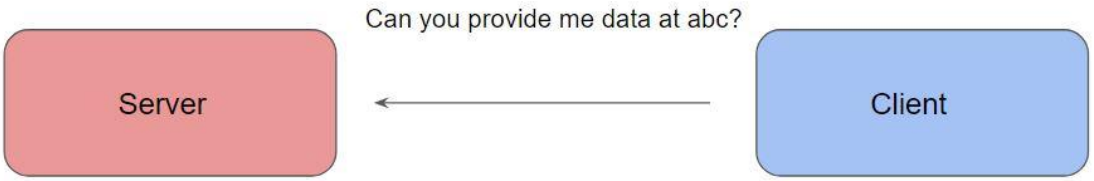
5.2 Consul Server

Consul Server is primarily responsible for maintaining the cluster state, as well as responding to queries received from clients.



5.3 Overview of Consul Client

The client is primarily responsible for making requests to the server and are also used for performing health check.



5.4 Overview of Consul Agent

An agent is a long-running daemon on every member of the Consul cluster.

The agent is able to run in either client or server mode.

Since all nodes must be running an agent, it is simpler to refer to the node as being either a client or server

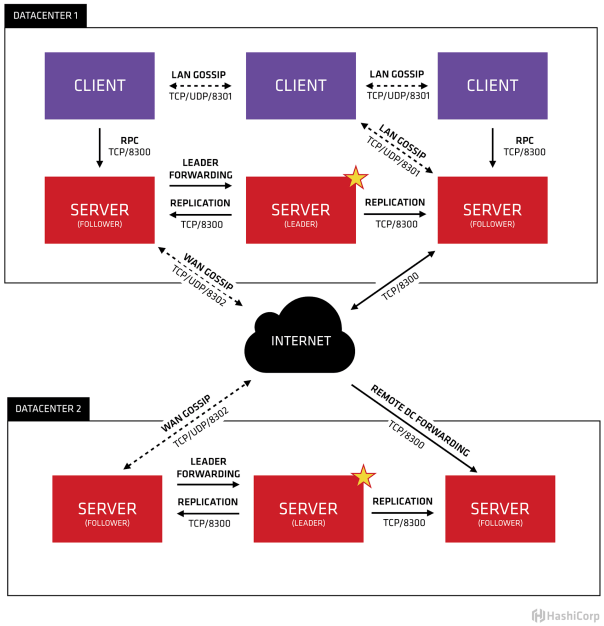
```
[root@consul-01 ~]# consul agent -dev
==> Starting Consul agent...
           Version: '1.8.5'
           Node ID: '71571abc-9474-9a9b-397c-1ff5acdfca72'
         Node name: 'consul-01'
        Datacenter: 'dc1' (Segment: '<all>')
            Server: true (Bootstrap: false)
       Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502, DNS: 8600)
      Cluster Addr: 127.0.0.1 (LAN: 8301, WAN: 8302)
           Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false

==> Log data will now stream in as it occurs:
```

5.5 Consul Datacenter

A datacenter is a networking environment that is private, low latency, and high bandwidth.

This excludes communication that would traverse the public internet.

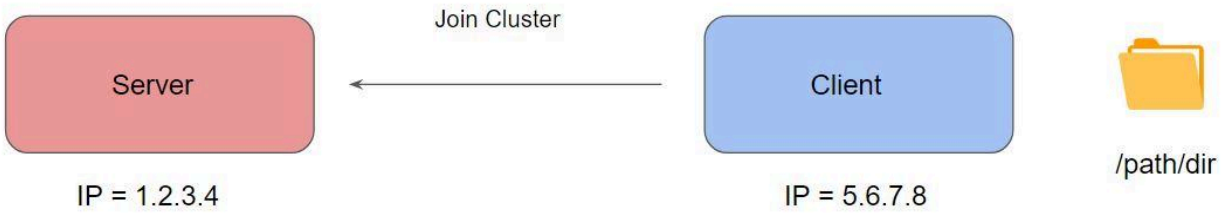
Multiple Availability Zone within a single AWS region would be considered part of a single datacenter.



Module 6: Joining Consul Clients

To join clients to consul cluster, the following command needs to be used:

consul agent -join 1.2.3.4 -bind 5.6.7.8 -data-dir /path/dir

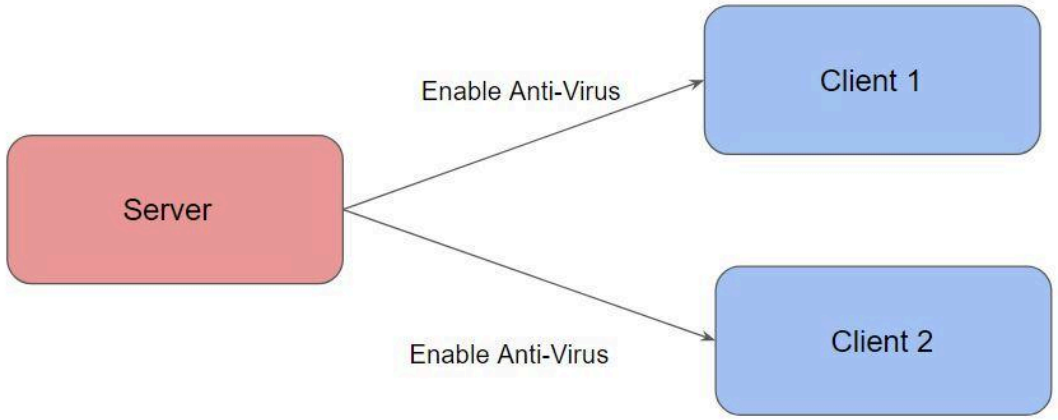


The following table shows some of the important flags along with their description.

Flag	Description
-join 1.2.3.4	Address of another agent to join upon starting up.
-bind 5.6.7.8	This is an IP address that should be reachable by all other nodes in the cluster
-data-dir /path/dir	This is required for all agents. The directory should be durable across reboots. All used for storing cluster state.

Module 7: Remote Execution Functionality

Remote Execution can be used to run a certain set of commands to perform the desired action.



The feature of remote execution is achieved with the consul exec command.

consul exec ping google.com

```
[root@consul-01 ~]# consul exec ping -c1 google.com
    consul-02: PING google.com (216.58.196.174) 56(84) bytes of data.
    consul-02: 64 bytes from maa03s31-in-f14.1e100.net (216.58.196.174):
    consul-02:
    consul-02: --- google.com ping statistics ---
    consul-02: 1 packets transmitted, 1 received, 0% packet loss, time 0m
    consul-02: rtt min/avg/max/mdev = 9.753/9.753/9.753/0.000 ms
    consul-02:
==> consul-02: finished with exit code 0
1 / 1 node(s) completed / acknowledged
```

Important Note:

Remote Execution is disabled by default.

You will need to explicitly enable it at the node level to make use of it.

Following is the command to Enable Remote Execution:

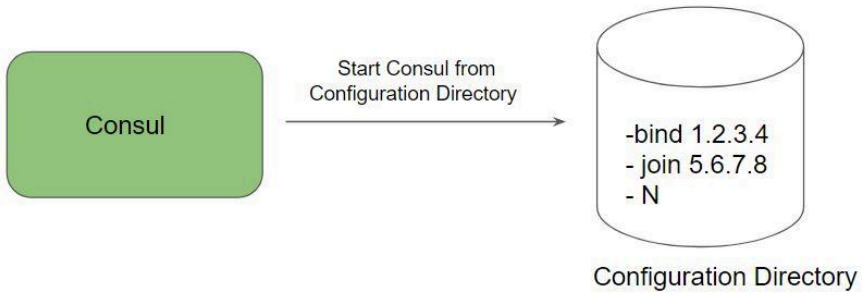
consul agent -hcl 'disable_remote_exec=false'

Module 8: Configuration Directory

8.1 Overview of Configuration Directory:

Running Consul from CLI with multiple command-line options is not a very scalable and efficient way of doing things.

consul agent -join 134.209.155.89 -bind 165.22.222.190 -data-dir /root/consul -hcl 'disable_remote_exec=false'



8.2 Using a Configuration Directory Approach

The agent has various configuration options that can be specified via the command-line or via configuration files.

Configuration can either be in JSON or HCL format.

```
[root@consul-02 consul-config]# consul agent -config-dir=/root/consul-config/
==> Starting Consul agent...
           Version: '1.8.5'
           Node ID: '952d3940-f447-3215-be8e-f20bf9b5e275'
         Node name: 'consul-02'
        Datacenter: 'dc1' (Segment: '')
            Server: false (Bootstrap: false)
       Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: -1, DNS: 8600)
      Cluster Addr: 165.22.222.190 (LAN: 8301, WAN: 8302)
           Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false

==> Log data will now stream in as it occurs:
```

8.3 Important Note - Configuration Directory

The CLI and Configuration File options are not always named the same.

CLI Command	Configuration File
join	start_join
bind	bind_addr
data-dir	data_dir

Module 9: Leave Behavior for Agents

Whenever we go somewhere, we generally inform parents/spouse about it.

If you do not inform, be prepared to sit on the couch for an hour or two listening to what could have been improved.

There are two primary leave behavior for an agent in Consul:

Options	Descriptions
Graceful Exit	It is used to ensure other nodes see the agent as "left" instead of "failed" When gracefully exiting, the agent first notifies the cluster it intends to leave the cluster.
Force Removal	When server simply fails (power/network cut). Datacenter will detect the failure and replication will continuously retry.

To gracefully halt an agent, send the process an interrupt signal (usually Ctrl-C from a terminal or running killall -s 2 consul

Forceful removal can be achieved by sending SIGKILL signal

killall -s 9 consul

Module 10: Consul Server Mode

Running consul in server mode (non-dev) allows customers to have flexibility related to the option sets that can be used.

Important Flags	Description
server	Providing this flag specifies that you want the agent to start in server mode.
-bootstrap-expect	This tells the Consul server how many servers the datacenter should have in total
-node	Each node in a datacenter must have a unique name. By default, Consul uses the hostname of the machine, but you'll manually override it, and set it to agent-one.
-bind and -data-dir	Address Agent will Listen & Storing state data.
config-dir	This flag tells consul where to look for its configuration Standard location is /etc/consul.d

Module 11: Systemd and Consul

11.1 Overview of Challenges

As of now, we have been deploying Consul either via CLI flags or via Configuration Directory.

Although this is a good approach for an initial start, but for production, it is not an ideal approach.

```
[root@consul-02 ~]# consul agent -join 134.209.155.89 -bind 165.22.222.190 -data-dir /root/consul
==> Starting Consul agent...
           Version: '1.8.5'
           Node ID: '952d3940-f447-3215-be8e-f20bf9b5e275'
         Node name: 'consul-02'
        Datacenter: 'dc1' (Segment: '')
            Server: false (Bootstrap: false)
       Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: -1, DNS: 8600)
      Cluster Addr: 165.22.222.190 (LAN: 8301, WAN: 8302)
           Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false
```

11.2 Systemd Based Approach

Systemd provides a system and service manager that runs as PID 1 and starts the rest of the system

You can specify a service file with all the configuration and settings and systemd will manage the consul process accordingly.

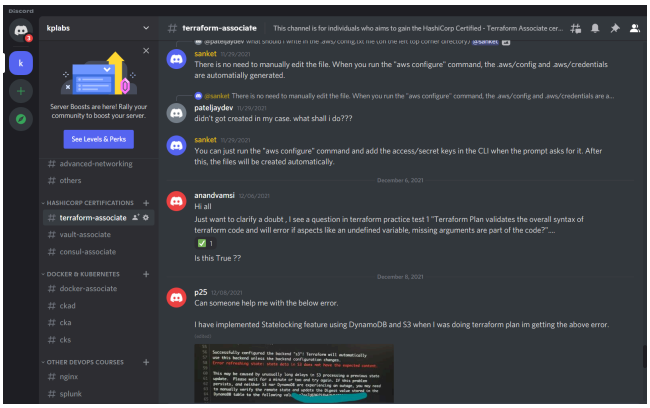
```
[root@consul-02 ~]# cat /usr/lib/systemd/system/consul.service
[Unit]
Description="HashiCorp Consul - A service mesh solution"
Documentation=https://www.consul.io/
Requires=network-online.target
After=network-online.target
ConditionFileNotEmpty=/etc/consul.d/consul.hcl

[Service]
User=consul
Group=consul
ExecStart=/usr/bin/consul agent -config-dir=/etc/consul.d/
ExecReload=/usr/bin/consul reload
ExecStop=/usr/bin/consul leave
KillMode=process
Restart=on-failure
LimitNOFILE=65536
```

Join Our Discord Community

We invite you to join our Discord community, where you can interact with our support team for any course-based technical queries and connect with other students who are doing the same course.

Joining URL:

http://kplabs.in/chat



KPLABS Course

HashiCorp Certified: Consul Associate

Service Discovery

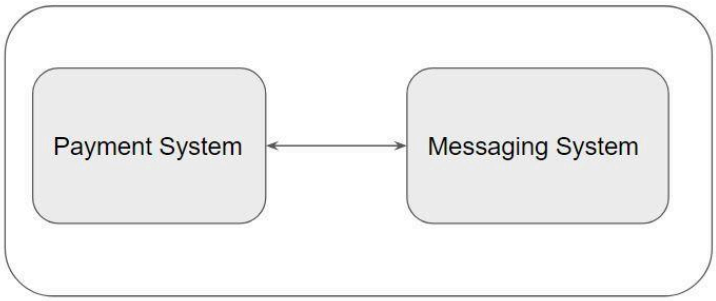
ISSUED BY
Zeal

REPRESENTATIVE
instructors@kplabs.in

Module 1: Overview of Service Discovery

1.1 Service Discovery in Monolithic Application

If two sub-systems in a monolithic application want to communicate with each other, it is very straight forward.



Complete Application

1.2 Service Discovery in Distributed Systems

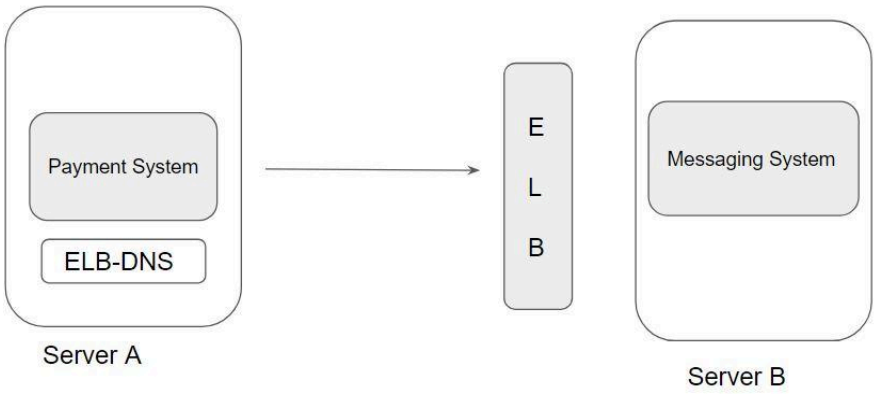
Since services are no longer part of the same application, there is a challenge involved in discovering other services.

There is also latency that is involved as networking comes into the picture.

1.3 Traditional Solution - Load Balancers

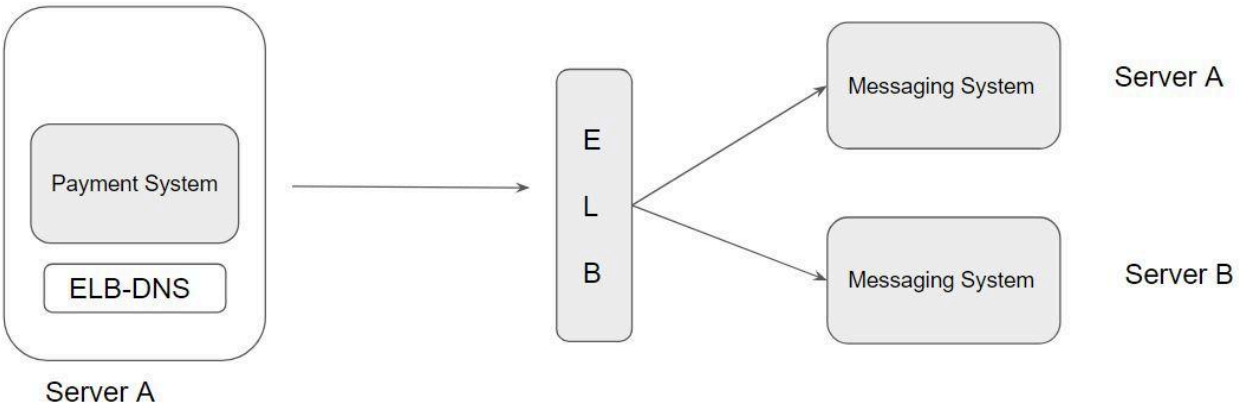
Load Balancers are used in-front of each service.

The caller service has details of the DNS of the Load balancer of the destination service.



1.4 Benefits of Load Balancers

Load Balancer based architecture also comes with various benefits like the ability to scale service, SSL termination at ELB, and others.



1.5 Challenges with Load Balancers

- Load Balancer needs to be managed for each individual service.
- Single Point of Failure
- Increased Latency

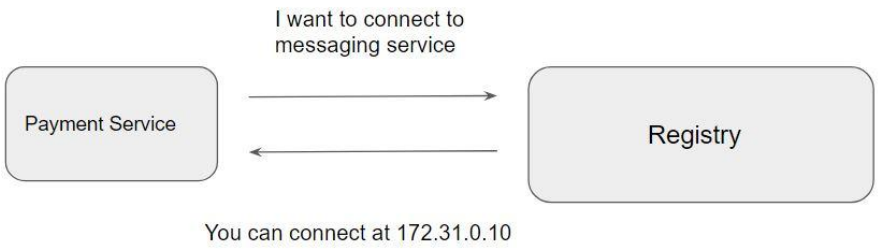
1.6 Consul for Service Discovery

With the Consul service discovery feature, there is a registry that is maintained.

This registry contains all the information about other services.

1.7 Payments Service to Messaging Communication

If a service wants to connect to a messaging service, it can query the registry and find the latest set of IP addresses for the messaging service.

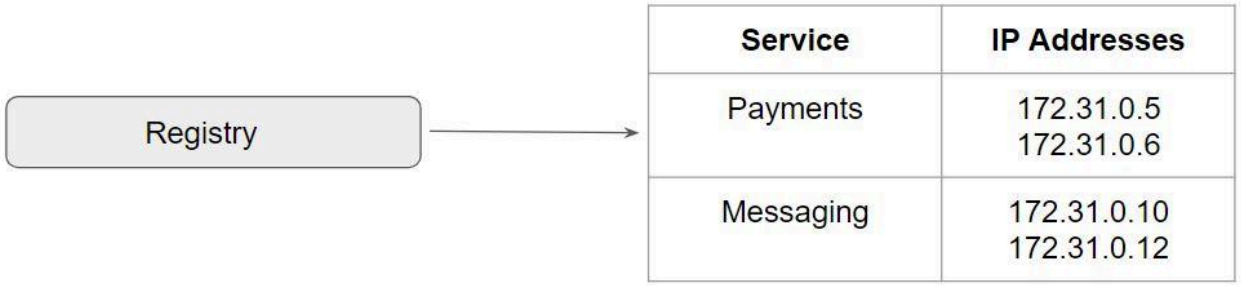


.

Module 2: Implementing Service Discovery

With the Consul service discovery feature, there is a registry that is maintained.

This registry contains all the information about other services.



2.1 2 Areas to Service Discovery

There are three key areas in the overall service discovery process:

- Registering a service
- Finding services
- Monitoring services.

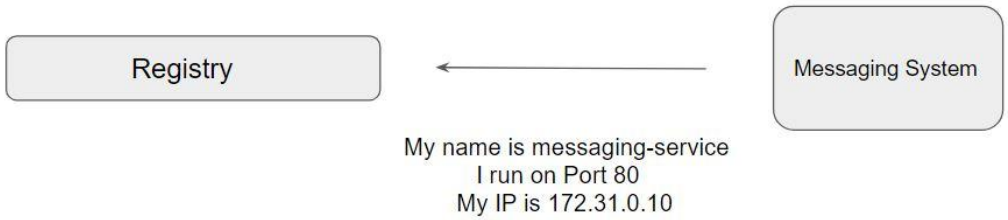
We need to get all these areas right for a perfect service discovery feature. And, it's easy :)

Step 1: Registering A Service

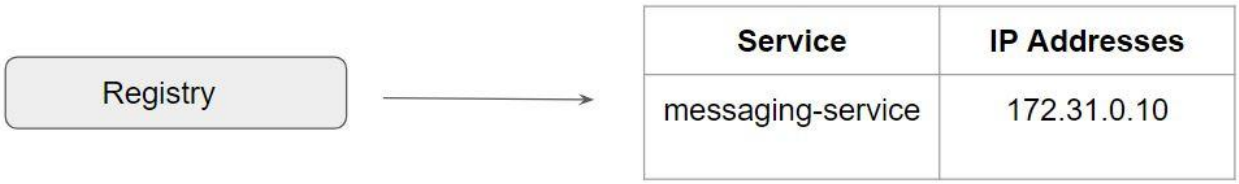
One of the main goals of service discovery is to provide a catalog of available services.

We have to write a simple service definition file to declare the availability of a service.

Consul Agent will forward the information to the Server.



After you have written a service definition file, consul agent will communicate with the cluster so that the service is registered.



Step 2: Finding Services

One of the primary query interfaces for Consul is DNS.

```
[root@consul-01 ~]# dig @localhost -p 8600 messaging-service.service.consul

; <<>> DiG 9.11.13-RedHat-9.11.13-6.el8_2.1 <<>> @localhost -p 8600 messaging-service.service.consul
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58801
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;messaging-service.service.consul. IN   A

;; ANSWER SECTION:
messaging-service.service.consul. 0 IN  A       206.189.135.92
```

Module 3: Service Health Checks

One of the primary roles of the agent is the management of system-level and application-level health checks.

A health check is considered to be application-level if it is associated with a service

Health Check Types	Description
Script + Interval	These checks depend on invoking an external application that performs the health check, exits with an appropriate exit code, and potentially generates some output.
HTTP + Interval	These checks make an HTTP GET request to the specified URL, waiting the specified interval amount of time between requests The status of the service depends on the HTTP response code: any 2xx code is considered passing
TCP + Interval	These checks make a TCP connection attempt to the specified IP/hostname and port, waiting interval amount of time between attempts

3.1 Script Checks

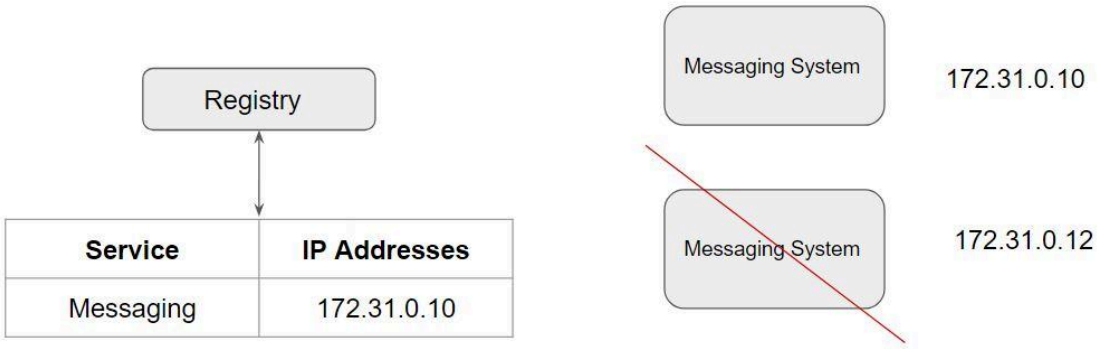
A check script is generally free to do anything to determine the status of the check.

The only limitations placed are that the exit codes must obey this convention:

Exit Code	Description
0	Check is passing
1	Check is warning
Any other code	Check is failing

3.2 Important Pointer - Service Health Checks

Consul will only return hosts that are healthy.

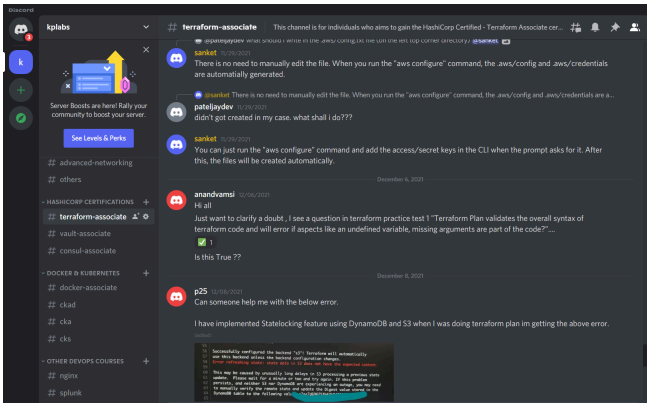


Join Our Discord Community

We invite you to join our Discord community, where you can interact with our support team for any course-based technical queries and connect with other students who are doing the same course.

Joining URL:

http://kplabs.in/chat



KPLABS Course

HashiCorp Certified: Consul Associate

Dynamic Application Configuration

ISSUED BY
Zeal

REPRESENTATIVE
instructors@kplabs.in

Module 1: Overview of the Key-Value Store

Consul's simple key/value store, useful for storing service configuration or other metadata.

Key	Value
timeout	500
max_memory	128M
max_cpu	2

Important Pointers for KV Store:

The KV store can be accessed by the consul kv CLI subcommands, HTTP API, and Consul UI.

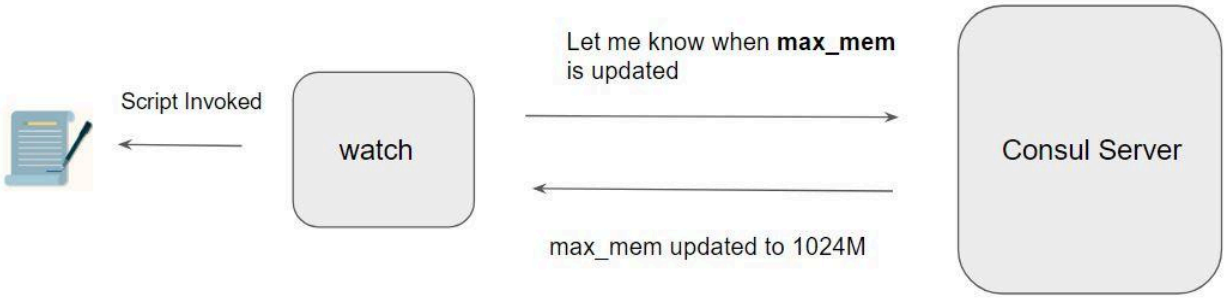
Objects are opaque to Consul, meaning there are no restrictions on the type of object stored in a key/value entry. The main restriction on an object is size - the maximum is 512 KB.

```
[root@consul-01 ~]# consul kv put max_memory 512M
Success! Data written to: max_memory
[root@consul-01 ~]# consul kv get max_memory
512M
```

Module 2: Understanding Watches Functionality

Watches feature in Consul checks for changes made to the Consul key-value store.

When an update is detected, an external handler is invoked.

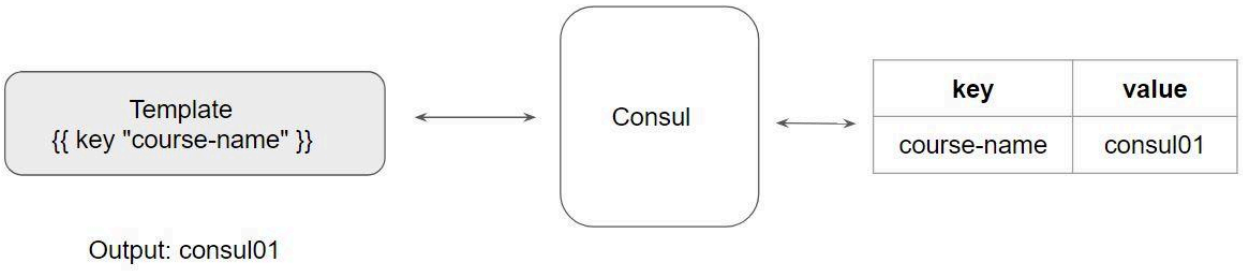


Watch functionality can watch for a wide variety of resources, these include:

- key
- keyprefix
- services
- nodes
- checks
- events

Module 3: Consul Template

Consul Template queries a Consul instance and updates any number of specified templates on the filesystem



Module 4: envconsul

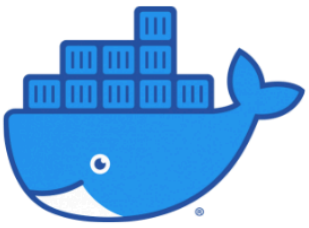
Envconsul provides a convenient way to launch a subprocess with environment variables populated from HashiCorp Consul and Vault.

key	value
address	1.2.3.4
max_conns	5
port	80

```
[root@ip-172-31-87-159 ~]# envconsul -prefix my-app env | egrep 'address|port|max_conns'
address=1.2.3.4
max_conns=5
port=80
```

The following diagram illustrates the high level workflow

key	value
address	1.2.3.4
max_conns	5
port	80



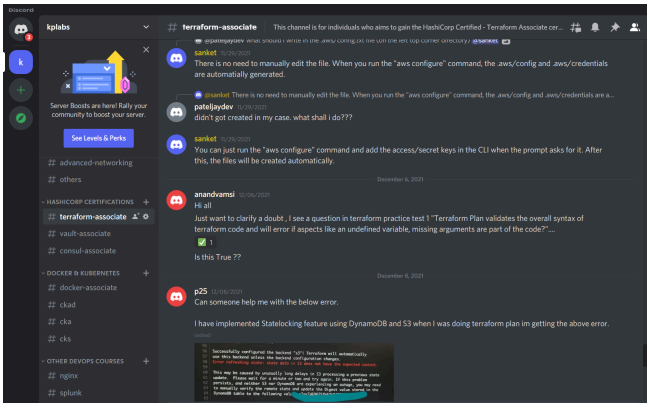

```
[root@ip-172-31-87-159 ~]# envconsul -prefix my-app env | egrep 'address|port|max_conns'
address=1.2.3.4
max_conns=5
port=80
```

Join Our Discord Community

We invite you to join our Discord community, where you can interact with our support team for any course-based technical queries and also connect with other students who are doing the same course.

Joining URL:

http://kplabs.in/chat



KPLABS Course

HashiCorp Certified: Consul Associate

Security

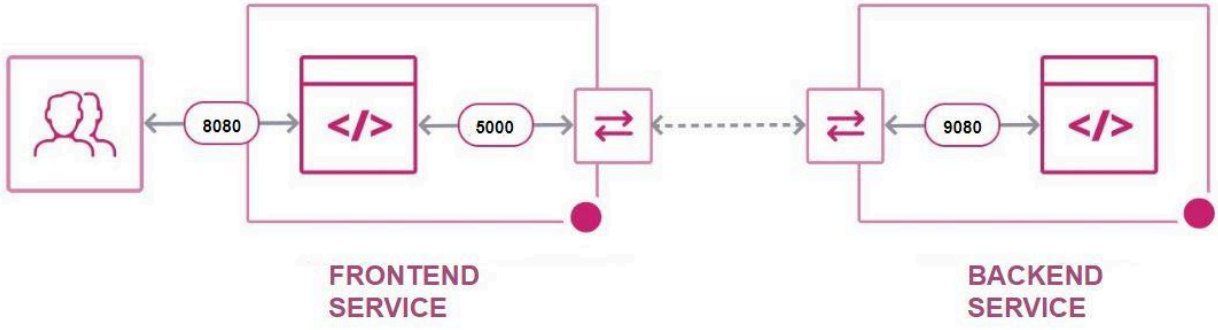
ISSUED BY
Zeal

REPRESENTATIVE
instructors@kplabs.in

Module 1: Overview of Consul Connect

1.1 Overview of Consul Connect

Consul Connect provides service-to-service connection authorization and encryption using mutual Transport Layer Security (TLS).



1.2 Sample Use-Cae

Frontend Service wants to communicate with Backend Service.

Additional Requirements:

- Should be over TLS (encrypted communication).
- Should have required level of authorization.

Module 2: Intentions and Precedence

2.1 Intentions

Intentions define access control for services via Connect and are used to control which services may establish connections or make requests

CLI Command	Description
consul intention create web db	Allow web to talk to db.
consul intention create -deny db '*'	Deny db from initiating connection to any service.
consul intention check web db	Checks whether a connection attempt between two services would be authorized given the current set of intentions and Consul configuration.
consul intention match db	Find all intentions for communicating to the "db" service:

2.2 Precedence

Permission precedence is applied top to bottom.

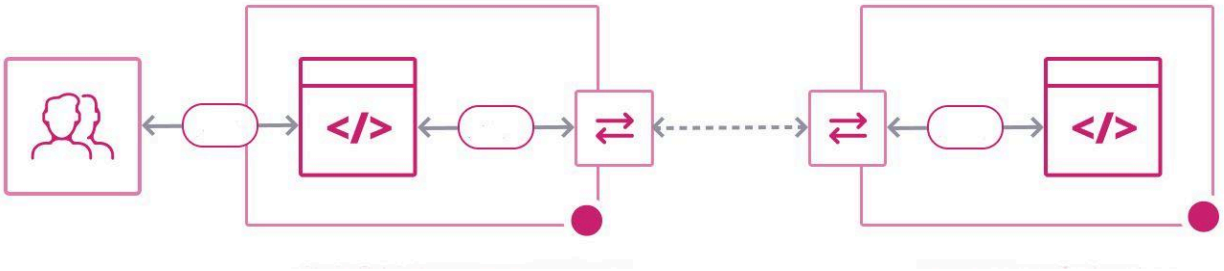
For any given request the first permission to match in the list is terminal and stops further evaluation

Source Namespace	Source Name	Destination Namespace	Destination Name	Precedence
Exact	Exact	Exact	Exact	9
Exact	*	Exact	Exact	8
*	*	Exact	Exact	7
Exact	Exact	Exact	*	6
Exact	*	Exact	*	5
*	*	Exact	*	4
Exact	Exact	*	*	3
Exact	*	*	*	2
*	*	*	*	1

Module 3: Intentions and Precedence

3.1 Supports for Multiple Proxy

Consul includes its own built-in Layer 4 (L4) proxy for testing and development but also offers first class support for Envoy as a sidecar proxy.



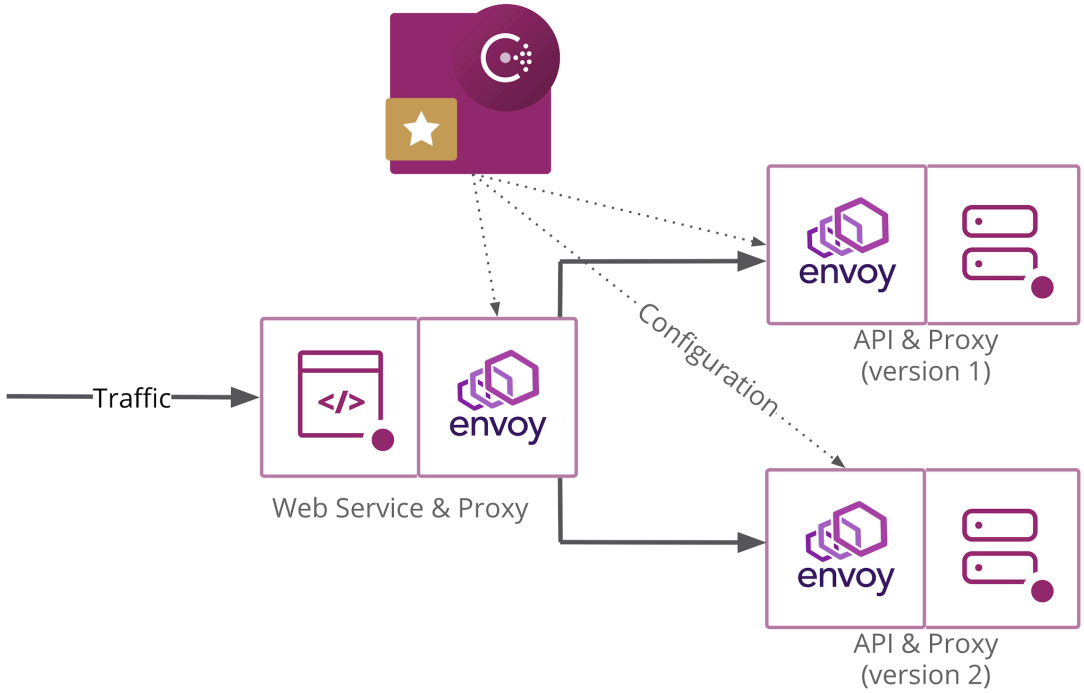
3.2 Overview of Envoy

Envoy is an open source edge and service proxy.

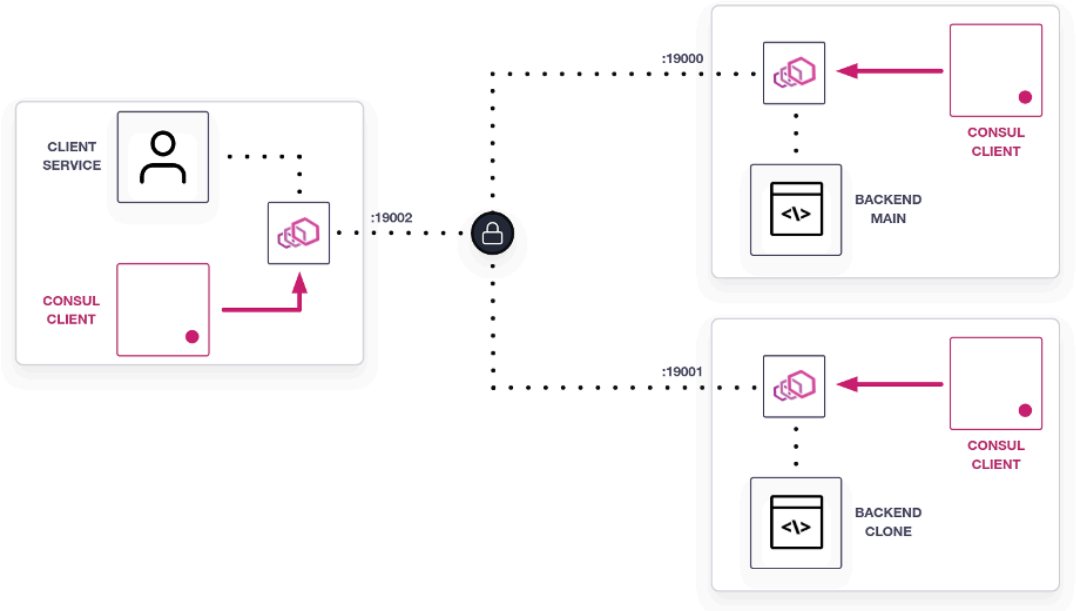
It comes with various additional features, some of these include:

- Support for HTTP/2 and gRPC
- Advanced Load Balancing
- Deep observability of L7 traffic

Use-Case 1 - Traffic Splitting



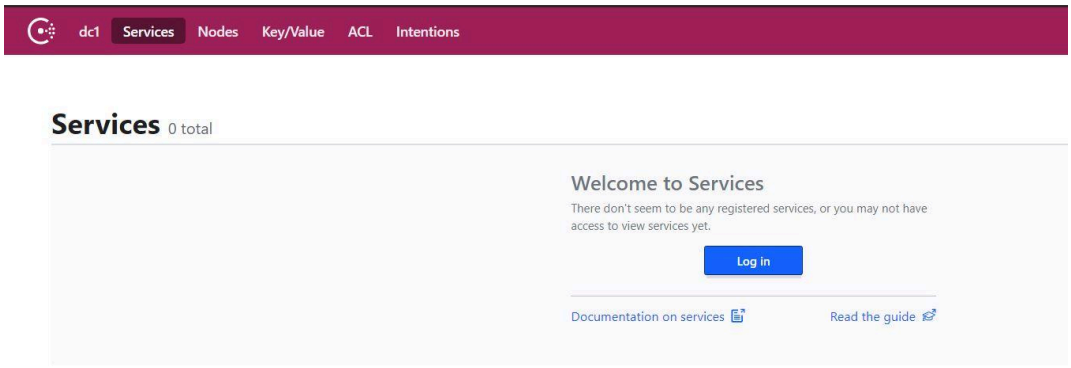
Use-Case 2 - Load Balancing



Module 4: Consul ACLs

One of the biggest challenges as of now in Consul is the lack of Authentication.

To overcome this, Consul uses Access Control Lists (ACLs) to secure access to the UI, API, CLI, service communications, and agent communications.



Step 1: Enable ACL in Consul

To enable ACLs, add the following ACL parameters to the agent's configuration file and then restart the Consul service.

```
acl = {
  enabled = true
  default_policy = "deny"
  enable_token_persistence = true
}
```

Step 2: Create a Bootstrap Token

It is important to have one token with unrestricted privileges in case of emergencies.

This will also allow you to quickly get started.

```
[root@consul-01 consul.d]# consul acl bootstrap
AccessorID:       c9e3a6dd-5b4a-de5f-fe6b-dfe8e01bb031
SecretID:         f23a368f-5b75-b832-7733-591b7d8eef6c
Description:      Bootstrap Token (Global Management)
Local:            false
Create Time:      2020-11-14 12:58:55.315441221 +0000 UTC
Policies:
   00000000-0000-0000-0000-000000000001 - global-management
```

Important Note

Using the token on the command line with the -token flag is not recommended, instead, you can set it as an environment variable once.

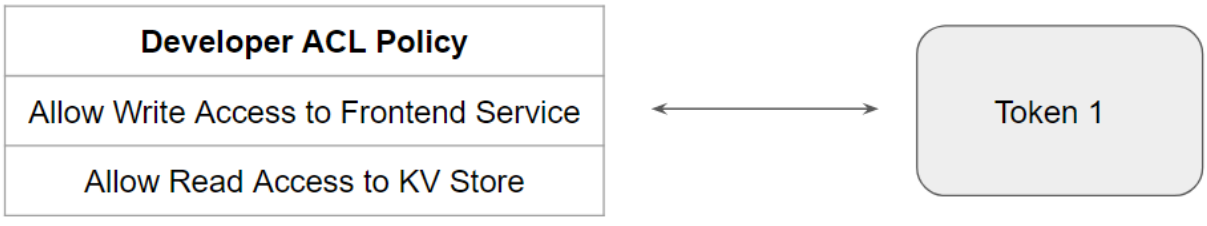
CONSUL_HTTP_TOKEN

Module 5: Understanding ACL Rules

5.1 ACL System in Consul

The ACL is Capability-based, relying on tokens which are associated with policies to determine which fine grained rules can be applied.

There are two primary components of ACL system: ACL Policies & ACL Tokens



5.2 Overview of Rules

Rules are composed of a resource, a segment (for some resource areas) and a policy disposition.

```
<resource> "<segment>" {
  policy = "<policy disposition>"
}
```

Write access on key-value store.

```
key_prefix "mobiles/" {
  policy = "write"
}
```

5.3 Actions for Rules

Following actions can be determined while writing a rule:

Action	Description
read	allow the resource to be read but not modified.
write	allow the resource to be read and modified.
deny	do not allow the resource to be read or modified.
list	allows access to all the keys under a segment in the Consul KV

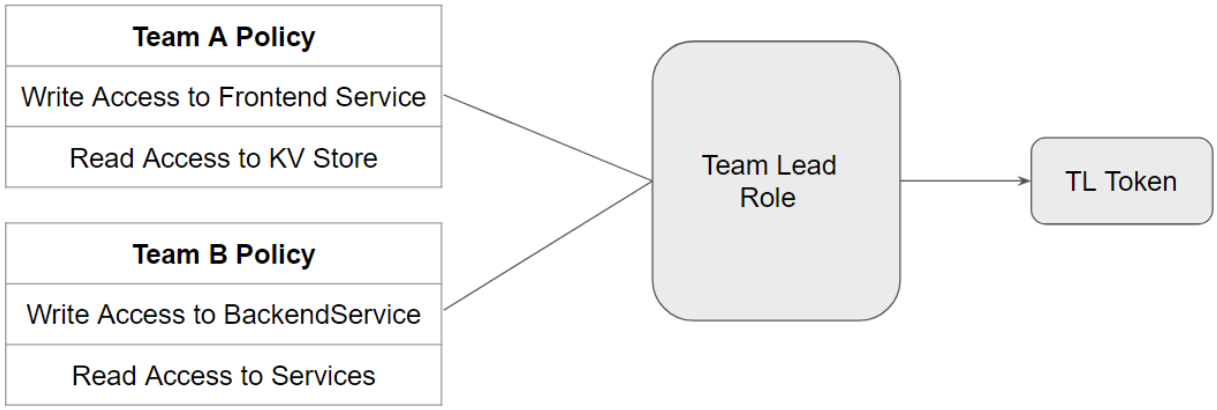
5.4 Scope for ACL Rules

Following resources are available for constructing rules:

Resource	Description
acl	Operations for managing the ACL system ACL API
agent	Utility operations in the Agent API, other than service and check registration
event	Listing and firing events in the Event API
key	Key/value store operations in the KV Store API
keyring	Keyring operations in the Keyring API
node	Node-level catalog operations in the Catalog API, Health API, Prepared Query API, Network Coordinate API, and Agent API
operator	Cluster-level operations in the Operator API, other than the Keyring API
etc	Includes query, service and session

Module 6: Understanding ACL Roles

Roles allow for the grouping of a set of policies into a reusable higher-level entity that can be applied to many tokens.



Module 7: Anonymous Tokens

The anonymous token is used when a request is made to Consul without specifying a bearer token.

The anonymous token's description and policies may be updated but Consul will prevent this token's deletion.

.

Module 8: Enabling ACLs on Agent

When you enable ACLs with a “deny” based approach, by default requests will be denied.

This applies even at the agent level.

```
consul[71676]:     2020-11-16T07:43:49.921Z [WARN]  agent: Coordinate update blocked by A
consul[71676]:     2020-11-16T07:44:09.118Z [WARN]  agent: Coordinate update blocked by A
consul[71676]:     2020-11-16T07:44:21.626Z [WARN]  agent: Node info update blocked by AC
consul[71676]:     2020-11-16T07:44:29.209Z [WARN]  agent: Coordinate update blocked by A
consul[71676]:     2020-11-16T07:44:51.352Z [WARN]  agent: Coordinate update blocked by A
consul[71676]:     2020-11-16T07:45:13.343Z [WARN]  agent: Coordinate update blocked by A
```

Step 1: Create Policy for Agent Token

Create the following policy for agent token

```
node_prefix "" {
   policy = "write"
}
service_prefix "" {
   policy = "read"
}
```

Step 2: Add token in Configuration

Add the agent token within the configuration

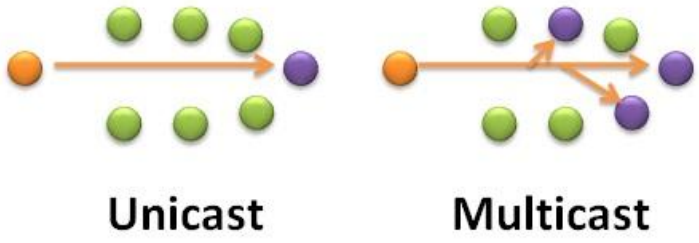
```
acl = {
  enabled = true
  default_policy = "deny"
  enable_token_persistence = true
  tokens {
    "agent" = "34c522d7-bce0-c27d-fc2e-69dc83e15487"
  }
}
```

Module 9: Overview of Gossip Protocol

9.1 Overview of Unicast and MultiCast

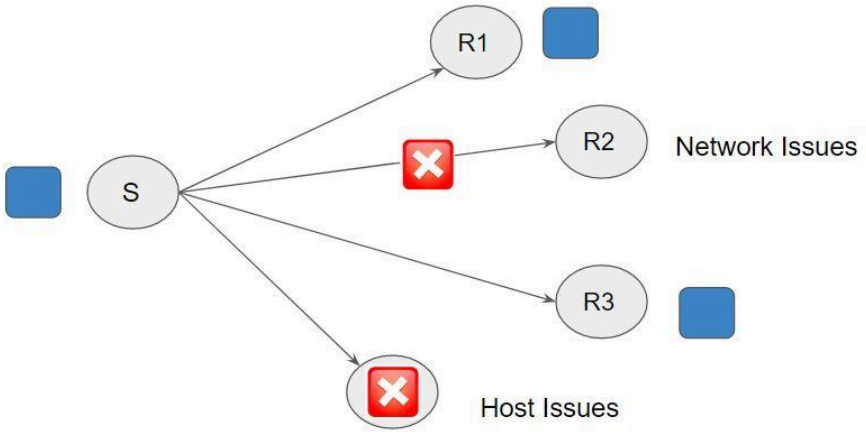
A Unicast transmission/stream sends IP packets to a single recipient on a network.

Multicast transmission sends IP packets to a group of hosts on a network

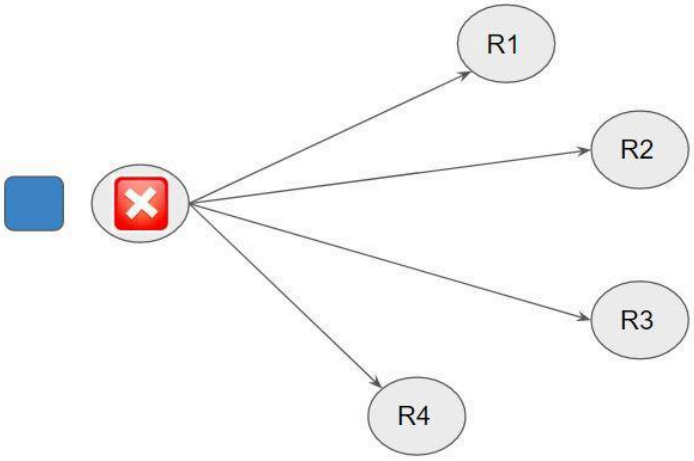


9.2 Multi-Cast Challenges

Suppose a sender wants to send a message to a group of hosts. The hosts might not receive the message due to various issues like network connectivity, the host being down, and so on.

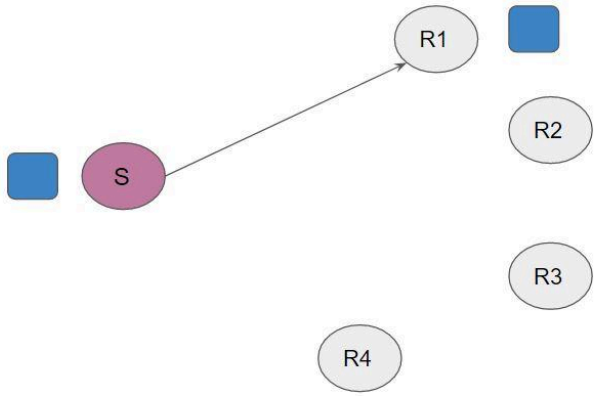


It can also be possible that the sender of the message went down.

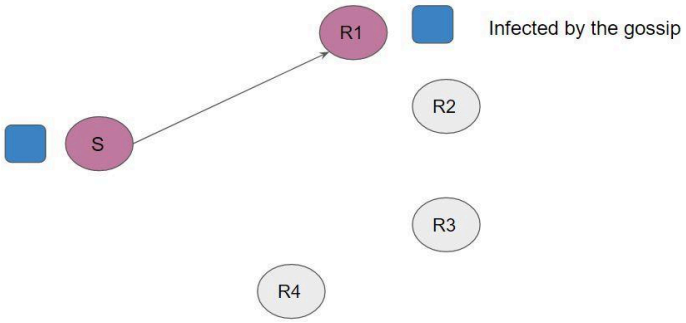


9.3 Gossip Protocol

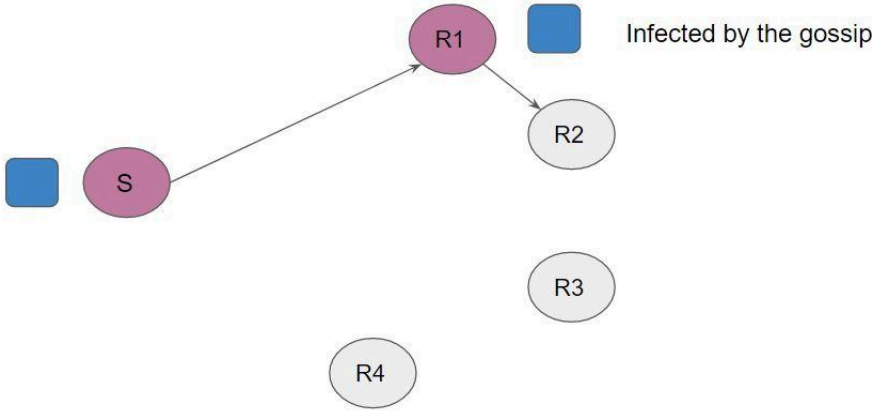
Data is periodically transmitted to random targets. In the below case, the sender has sent the message to R1.



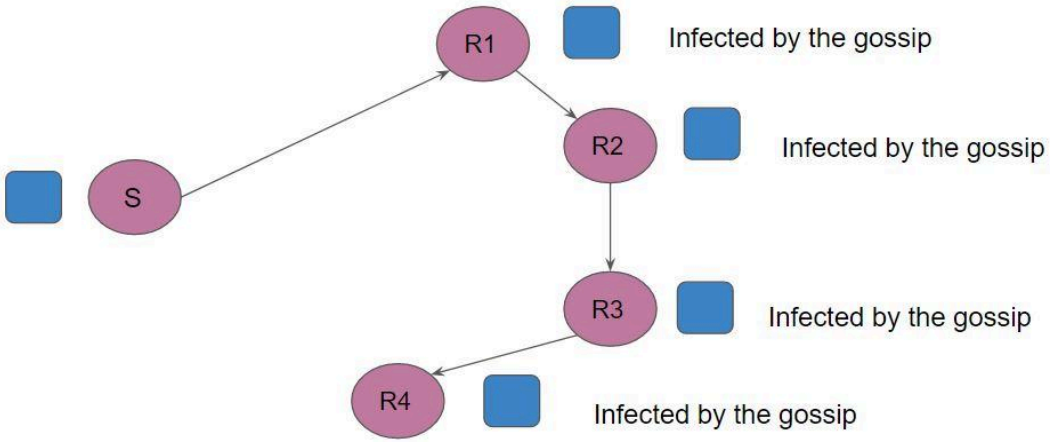
Once R1 receives the message, it is referred to as infected by the gossip.



Once R1 receives the message, it chooses random targets and sends out copies of the message.



In the final stage, all the hosts will have the data.

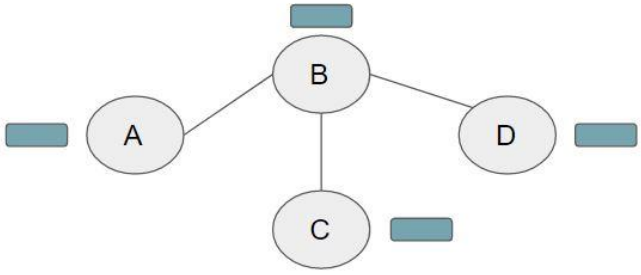


9.4 Understanding Gossip Protocol

A gossip protocol is a procedure or process of computer peer-to-peer communication

The amount of overhead involved is not that high when compared to a non-gossip scenario.

Some ad-hoc networks have no central registry and the only way to spread common data is to rely on each member to pass it along to their neighbors.

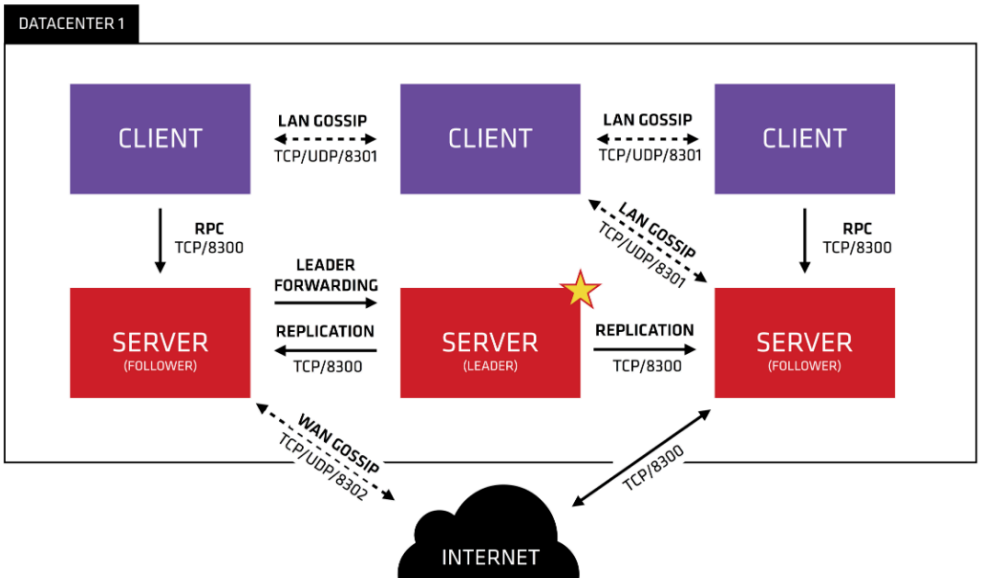


.

Module 10: Gossip Encryption in Consul

10.1 Consul and Gossip Protocol

Consul uses a gossip protocol to manage membership and broadcast messages to the cluster



10.2 Challenge with Plain Text Data

By default, the data would be in plaintext and it is possible to capture the network packets and retrieve the information.

```
23:54:56.295177 IP (tos 0x0, ttl 62, id 21957, offset 0, flags [DF], proto UDP (17), length 180)
    134.209.154.246.amberon > consul-01.amberon: [udp sum ok] UDP, length 152
        0x0000:  4500 00b4 55c5 4000 3e11 93ca 86d1 9af6  E...U.@.>.......
        0x0010:  9f41 91a0 206d 206d 00a0 e9b7 0282 a750  .A...m.m.......P
        0x0020:  6179 6c6f 6164 da00 8201 84aa 4164 6a75  ayload......Adju
        0x0030:  7374 6d65 6e74 cbbf 1c6a 34cb fc78 dba5  stment...j4..x..
        0x0040:  4572 726f 72cb 3fb2 6863 53ca 9a97 a648  Error.?.hcS....H
        0x0050:  6569 6768 74cb 3f1f 3860 b1cf e438 a356  eight.?.8`...8.V
        0x0060:  6563 98cb bf0d cd0a 4ee7 df3f cb3f 1460  ec......N..?.?.`
        0x0070:  6fea 7b72 24cb bf2e 6d61 3029 00b0 cbbf  o.{r$...ma0)....
        0x0080:  058f 3c60 5d43 94cb 3f12 a934 98b5 9dc3  ..<`]C..?..4....
        0x0090:  cbbf 31f6 ddb9 045e e6cb bf2b 9cda 8468  ..1....^...+...h
        0x00a0:  2181 cb3f 0fb7 332f 826e c8a5 5365 714e  !..?..3/.n..SeqN
        0x00b0:  6fcd 031d                                o...
```

10.3 Enabling Encryption

As part of security, it is important to enable gossip encryption.

Two steps:

- Generate a cryptographic key.
- Add key within the configuration file.

Step 1: Generate Cryptographic Key

We can easily generate a key with the consul keygen command

```
[root@consul-01 ~]# consul keygen
6k527qHgwLNpdai7Yuu9nmKrl1Z6je+4H6JQ7NZwjYg=
```

Step 2: Add Key in Configuration File

Add the encryption key parameter to the agent configuration file

```
[root@consul-01 ~]# cat consul.hcl
data_dir =  "/etc/consul.d/consul-dir"
bind_addr = "159.65.145.160"
client_addr = "0.0.0.0"
bootstrap_expect = 1
node_name = "consul-server"
ui = true
server = true
encrypt = "tpnV5YZa56x9GcOnC4Ob/nPdcG2pus3xZei/oMhFHow="
```

10.4 Configuring Gossip for Existing Datacenter

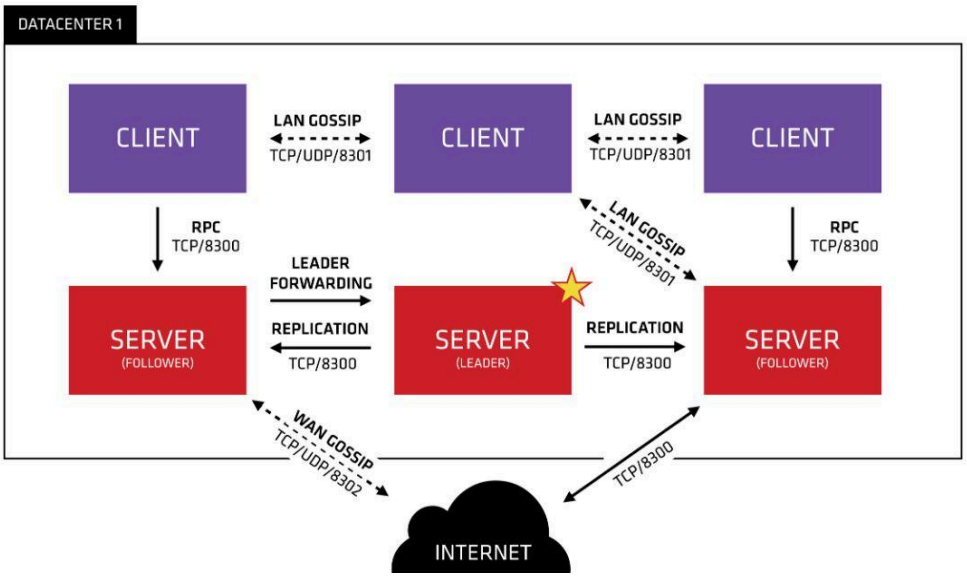
Gossip encryption can also be enabled on existing data centers but requires several extra steps.

The additional configuration of the agent configuration parameters, encrypt_verify_incoming and encrypt_verify_outgoing is necessary.

```
data_dir =  "/etc/consul.d/consul-dir"
bind_addr = "159.65.145.160"
client_addr = "0.0.0.0"
bootstrap_expect = 1
node_name = "consul-server"
ui = true
server = true
encrypt = "tpnV5YZa56x9Gc0nC4Ob/nPdcG2pus3xZei/oMhFHow="
encrypt_verify_incoming = true,
encrypt_verify_outgoing = true
```

10.5 Important Note

- TCP and UDP protocol can be used for Gossip.
- Port Number: 8301

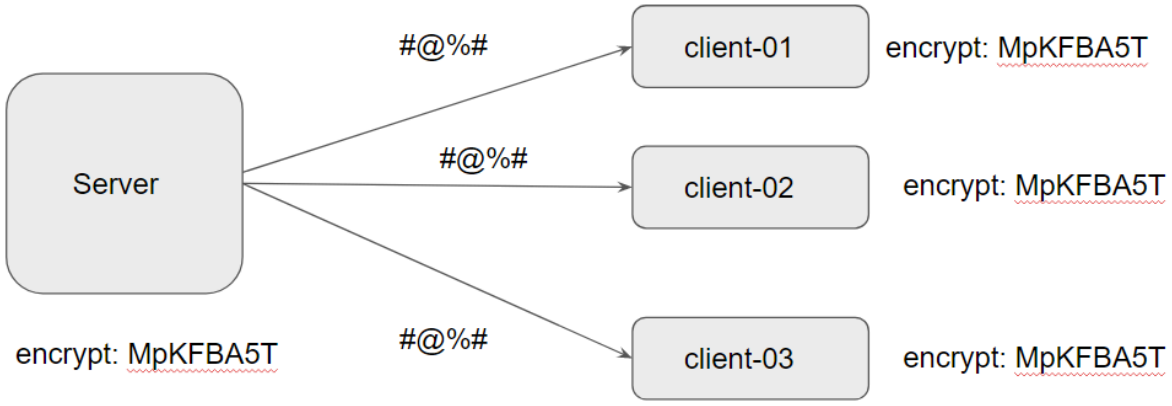


.

Module 11: Gossip Encryption on Existing DC

11.1 Understanding the Challenge

The steps to enable gossip encryption on a new and an existing datacenter differs.



11.2 Steps Involved

To enable Gossip encryption on existing datacenters, there are several additional steps involved.

The additional configuration of the agent configuration parameters, encrypt_verify_incoming and encrypt_verify_outgoing is necessary.

- Step 1: Generate the keygen.
- Step 2: Set encrypt_verify_incoming and encrypt_verify_outgoing to false & restart.
- Step 3: Set encrypt_verify_outgoing to true and restart.
- Step 4: Set encrypt_verify_incoming to true and restart.

11.3 Final Configuration

```
data_dir =  "/etc/consul.d/consul-dir"
bind_addr = "134.209.154.246"
node_name = "consul-02"
start_join = ["159.65.145.160"]
enable_local_script_checks = true
encrypt = "ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE=",
encrypt_verify_incoming = true,
encrypt_verify_outgoing = true
```

11.4 Importance of 2 Parameters

Configuration Parameter	Description
encrypt: MpKFBA5T encrypt_verify_incoming = false, encrypt_verify_outgoing = false	Agents will be able to decrypt gossip but will not yet be able to send encrypted traffic.
encrypt: MpKFBA5T encrypt_verify_incoming = false, encrypt_verify_outgoing = true	The agents will now be sending encrypted gossip but will still allow incoming unencrypted traffic.
encrypt: MpKFBA5T encrypt_verify_incoming = true, encrypt_verify_outgoing = true	All the agents will now be strictly enforcing encrypted gossip

Module 13: Rotating Gossip Encryption Key

13.1 Overview of Challenge

As part of compliance and security best practises , it is important to regularly rotate the encryption keys.

13.2 Rotating Gossip Encryption Keys

The consul keyring command is used to examine and modify the encryption keys used in Consul's gossip pools.

It is capable of distributing new encryption keys to the agents, retiring old encryption keys, and changing the keys used by the agents to encrypt messages.

```
[root@consul-01 ~]# consul keyring -list
==> Gathering installed encryption keys...

WAN:
  ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE= [1/1]

dc1 (LAN):
  ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE= [2/2]
```

13.3 4 steps

There are four primary steps for the entire process of gossip key rotation.

- Generate a new encryption key.
- Add new key to the keyring
- Promote new key to primary
- Remove the old key from the keyring.

Step 1: Create New Encryption Key

We can easily generate key with consul keygen command

```
[root@consul-01 ~]# consul keygen
NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M=
```

Step 2: Add New Key to the Keyring

Add your newly generated key to the keyring.

```
[root@consul-01 ~]# consul keyring -install NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M=
==> Installing new gossip encryption key...
[root@consul-01 ~]# consul keyring -list
==> Gathering installed encryption keys...

WAN:
  ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE= [1/1]
  NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M= [1/1]

dc1 (LAN):
  ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE= [2/2]
  NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M= [2/2]
```

Step 3: Promote New Key to Primary

Once all agents have received the key and are able to use it as the primary encryption key, it is possible to promote the new key to primary.

```
[root@consul-01 ~]# consul keyring -use NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M=
==> Changing primary gossip encryption key...
```

Step 4: Remove Old Key from Keyring

To avoid unused keys remaining in the keyring, we recommend you remove the old primary from the keyring once a new key is installed.

```
[root@consul-01 ~]# consul keyring -remove ER5awGvrbkd25fO67Q4SktzZwSwR/F2SFQMExXmFlUE=
==> Removing gossip encryption key...
```

Important Pointers

The encrypt parameter is needed only at join time and after that the agent will persist whatever data it needs in the local keyring.

If you later remove the encrypt parameter from configuration file, it will not have an impact.

```
[root@consul-01 serf]# pwd
/etc/consul.d/consul-dir/serf
[root@consul-01 serf]# cat local.keyring
[
  "NKLJr8MU3zIiC15EK9SVWtibNynArFa/wHo71nzPf0M="
```

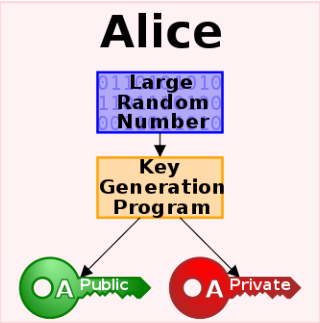
Module 14: Introduction to Asymmetric Key Encryption

Asymmetric cryptography uses public and private keys to encrypt and decrypt data.

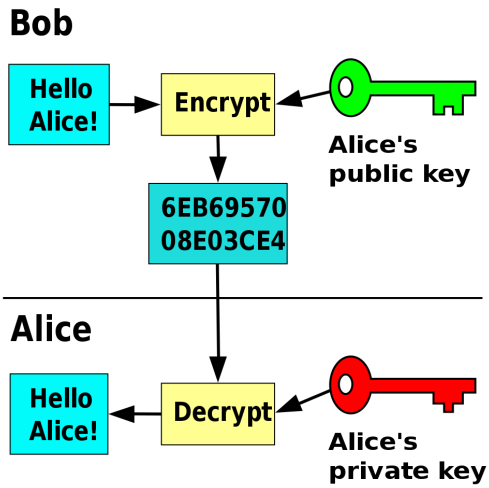
One key in the pair can be shared with everyone; it is called the public key. The other key in the pair is kept secret; it is called the private key.

Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption.

Step 1: Generation of Keys



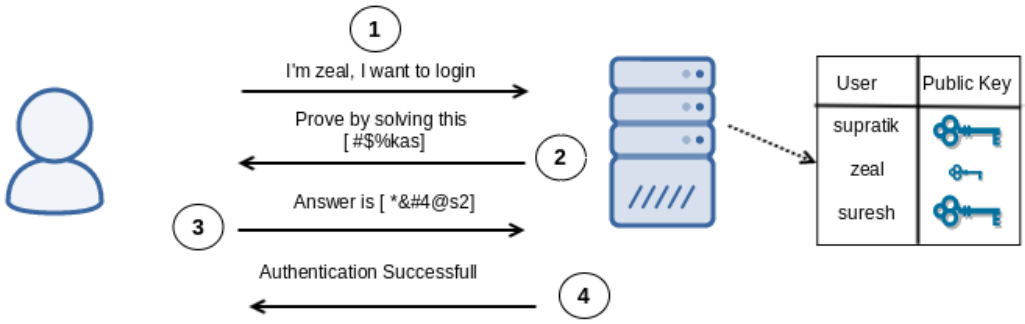
Step 2: Encryption and Decryption



14.1 Use-Case of Asymmetric Key Encryption

Step 1

User zeal wants to log in to the server. Since the server uses a public key authentication, instead of taking the password from the user, the server will verify if the User claiming to be zeal actually holds the right private key.



Step 2

The server creates a simple challenge, 2+3=? and encrypts this challenge with the Public Key of the User and sends it back to the User. The challenge is sent in an encrypted format.

Step 3:

Since the user zeal holds the associated private key, he will be able to decrypt the message and compute the answer, which would be 5. Then, he will encrypt the message with the private key and send it back to the server.

Step 4:

The server decrypts the message with the user's Public Key and checks if the answer is correct. If yes, then the server will send an Authentication Successful message and the user will be able to log in.

14.2 Protocols

Because of the advantage that it offers, Asymmetric key encryption is used by a variety of protocols.

Some of these include:

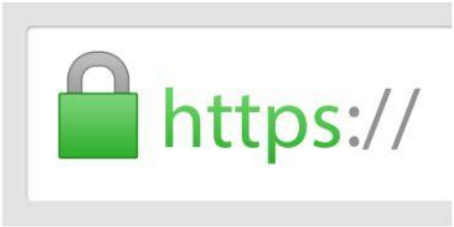
- PGP
- SSH
- Bitcoin
- TLS
- S/MIME

Module 15: Understanding SSL/TLS

HTTPS is an extension of HTTP.

In HTTPS, the communication is encrypted using Transport Layer Security (TLS)

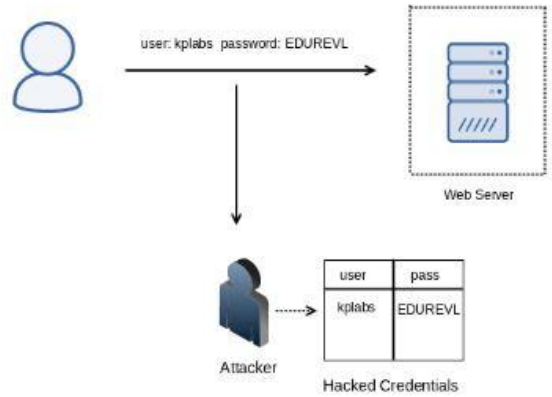
The protocol is therefore also often referred to as HTTP over TLS or HTTP over SSL.



Scenario 1: MITM Attacks

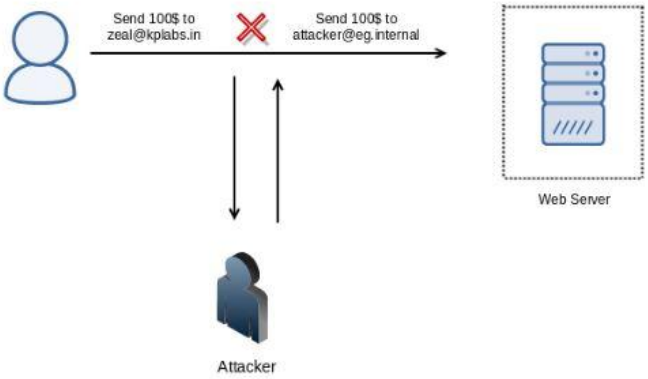
User is sending their username and password in plaintext to a Web Server for authentication over a network.

There is an Attacker sitting between them doing a MITM attack and storing all the credentials he finds over the network to a file:



Scenario 2: MITM & Integrity Attacks

Attackers changing the payment details while packets are in transit.



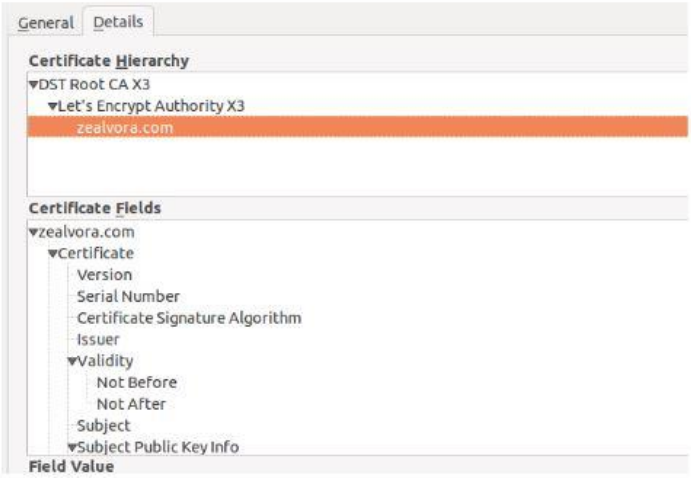
Introduction to SSL/TLS

To avoid the previous two scenarios (and many more), various cryptographic standards were clubbed together to establish secure communication over an untrusted network and they were known as SSL/TLS.

Protocol	Year
SSL 2.0	1995
SSL 3.0	1996
TLS 1.0	1999
TLS 1.1	2006
TLS 1.2	2008
TLS 1.3	2018

Every website has a certificate (like a passport that is issued by a trusted entity).

The certificate has a lot of details like the domain name it is valid for, the public key, validity, and others.



The browser (clients) verifies if it trusts the certificate issuer.

It will verify all the details of the certificate.

It will take the public key and initiate a negotiation.

Asymmetric key encryption is used to generate a new temporary

Symmetric key which will be used for secure communication.

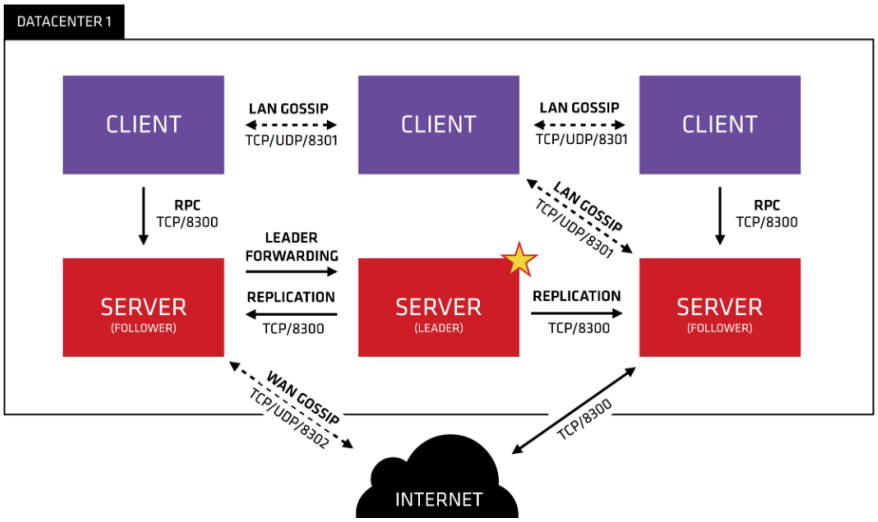
The following snapshot shows sample web-server configuration:

```
server {
       listen         80;
       server_name    zealvora.com;
       return         301 https://$server_name$request_uri;
}


server {
  server_name zealvora.com;
  listen 443 default ssl;
  server_name zealvora.com;
  ssl_certificate /etc/letsencrypt/archive/zealvora.com/fullchain1.pem;
  ssl_certificate_key /etc/letsencrypt/archive/zealvora.com/privkey1.pem;

  location / {
    root /websites/zealvora/;
    include location-php;
    index index.php;
 }
 location ~ /.well-known {
        allow all;
 }
}
```

Module 16: RPC Encryption with TLS

16.1 Consul & RPC

Consul Client and Server communicate over RPC on port 8300.



16.2 Challenges with Plain Text Data

By default, the data would be in plaintext and it is possible to capture the network packets and retrieve the information.

```
10:51:50.059116 IP (tos 0x0, ttl 64, id 17638, offset 0, flags [DF], proto TCP (6),
    consul-01.tmi > consul-01.55355: Flags [P.], cksum 0x623a (incorrect -> 0x185c),
ions [nop,nop,TS val 3504529536 ecr 3504529536], length 80
        0x0000:  4500 0084 44e6 4000 4006 93ca 9f41 91a0  E...D.@.@....A..
        0x0010:  9f41 91a0 206c d83b 1405 92eb 7efa 9eef  .A...l.;....~...
        0x0020:  8018 0156 623a 0000 0101 080a d0e2 e080  ...Vb:..........
        0x0030:  d0e2 e080 83a5 4572 726f 72a0 a353 6571  ......Error..Seq
        0x0040:  12ad 5365 7276 6963 654d 6574 686f 64b0  ..ServiceMethod.
        0x0050:  5374 6174 7573 2e52 6166 7453 7461 7473  Status.RaftStats
        0x0060:  83ab 4c61 7374 436f 6e74 6163 74a1 30a9  ..LastContact.0.
        0x0070:  4c61 7374 496e 6465 7867 a84c 6173 7454  LastIndexg.LastT
```

16.3 Enabling Encryption

As part of security, it is important to enable RPC encryption.

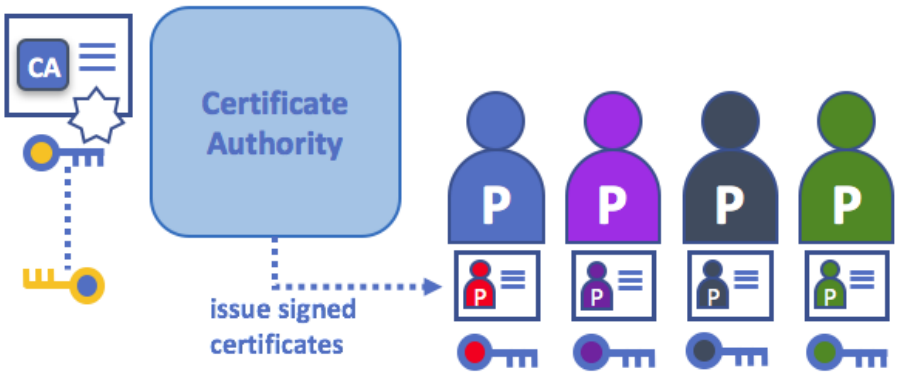
Three steps:

- Initialize In-Built CA
- Create Server Certificates.
- Configure Servers and Clients.

16.4 Certificate Authority

Certificate Authority is an entity that issues digital certificates.

The key part is that both the receiver and the sender trusts the CA.



16.5 Overview Steps Involved

Step 1: Initialize Built-In CA

You can use the in-built CA provided by the Consul or make use of 3rd party private CAs.

```
[root@consul-01 consul.d]# consul tls ca create
==> Saved consul-agent-ca.pem
==> Saved consul-agent-ca-key.pem
```

Step 2: Create Server Certificates

You can create the server certificates easily using in-built CA.

```
[root@consul-01 consul.d]# consul tls cert create -server
==> WARNING: Server Certificates grants authority to become a
    server and access all state in the cluster including root keys
    and all ACL tokens. Do not distribute them to production hosts
    that are not server nodes. Store them as securely as CA keys.
==> Using consul-agent-ca.pem and consul-agent-ca-key.pem
==> Saved dc1-server-consul-0.pem
==> Saved dc1-server-consul-0-key.pem
```

Step 3: Configure Server & Clients

Add appropriate configuration parameters within the configuration file to make use of certificates.

```
verify_incoming = true,
verify_outgoing = true,
verify_server_hostname = true,
ca_file = "consul-agent-ca.pem",
cert_file = "/etc/consul.d/dc1-server-consul-0.pem",
key_file = "/etc/consul.d/dc1-server-consul-0-key.pem",
auto_encrypt {
  allow_tls = true
}
```

Server Configuration

```
verify_incoming = false,
verify_outgoing = true,
verify_server_hostname = true,
ca_file = "consul-agent-ca.pem",
auto_encrypt = {
  tls = true
}
```

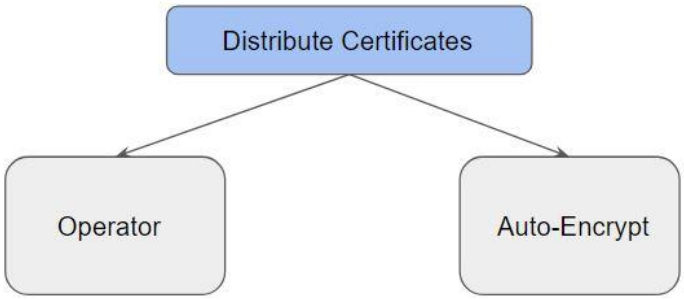
Client Configuration

16.6 Methods for Distributing Certificates

There are two methods for distributing client certificates: operator and auto encryption

With auto-encryption, you can configure the Consul servers to automatically distribute certificates to the clients.

The operator method is recommended if you need to use a third-party CA



Module 17: Overview of API

API stands for an application programming interface.

It is generally used for inter-communication between multiple software.

With the API, the exact structure of request and response is documented upfront and is likely to remain the same throughout time.

Use-Case

James wants to build a weather report application. Since it needs a weather report for all countries, he wonders where he can get all the data from. OpenWeatherMap has all this data and thus James decides to integrate his application and fetch data from OpenWeatherMap database.

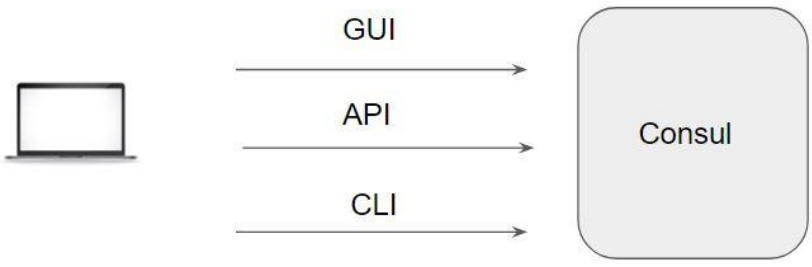
Now question is, how will he parse all this data?

Module 18: HTTP API in Consul

11.1 Overview of Consul Interface

There are multiple ways to connect with Consul:

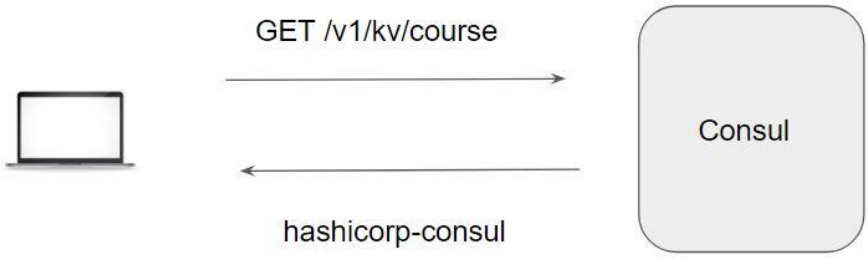
GUI, API as well as CLI



18.2 HTTP API

The main interface to Consul is a RESTful HTTP API.

All API routes are prefixed with /v1/



18.3 Important Pointers

Depending on the resource on which operation needs to be performed, the endpoint changes.

The API documentation provides extensive information about the same.

By default, the output of all HTTP API requests is minimized JSON. If the client passes pretty on the query string, formatted JSON will be returned.

18.4 Authentication

When authentication is enabled, a Consul token should be provided to API requests using the X-Consul-Token header or with the Bearer scheme in the authorization header.

```
curl \
  --header "X-Consul-Token: <consul token>" \
  http://127.0.0.1:8500/v1/agent/members
```

```
curl \
  --header "Authorization: Bearer <consul token>" \
  http://127.0.0.1:8500/v1/agent/members
```

KPLABS Course

HashiCorp Certified: Consul Associate

Infrastructure & High-Availability

ISSUED BY
Zeal

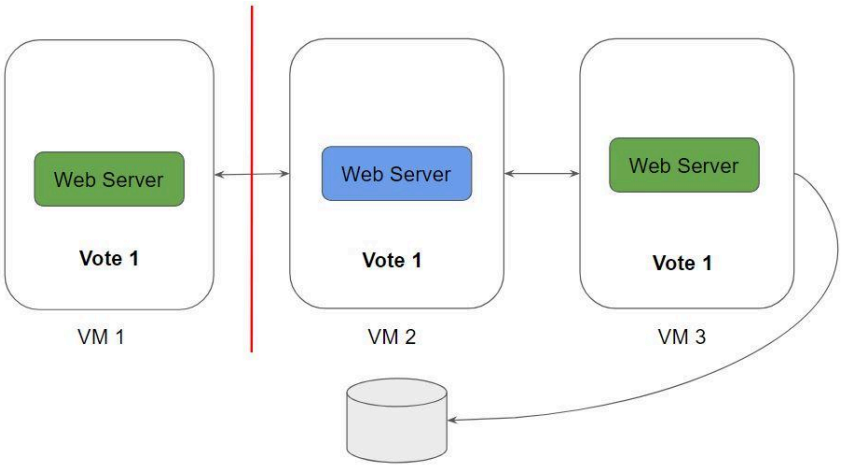
REPRESENTATIVE
instructors@kplabs.in

Module 1: Split-Brain & Importance of Quorum

Swarm manager nodes use the Raft Consensus Algorithm to manage the swarm state. You only need to understand some general concepts of Raft in order to manage a swarm.

There is no limit on the number of manager nodes. The decision about how many manager nodes to implement is a trade-off between performance and fault-tolerance.

Adding manager nodes to a swarm makes the swarm more fault-tolerant.



,

We should maintain an odd number of nodes within the cluster.

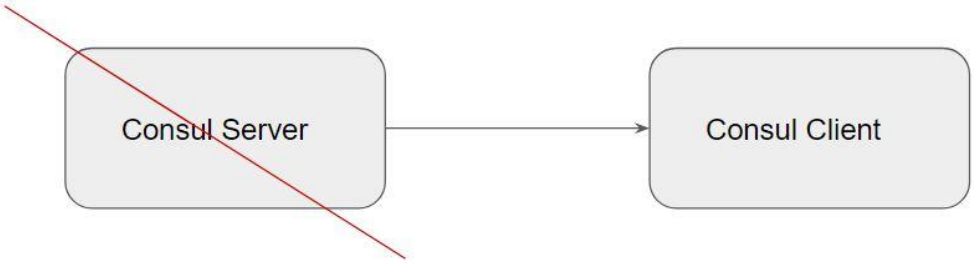
Cluster Size	Majority	Fault Tolerance
1	0	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

Module 2: Implementing High-Availability in Consul

2.1 Understanding the Challenge

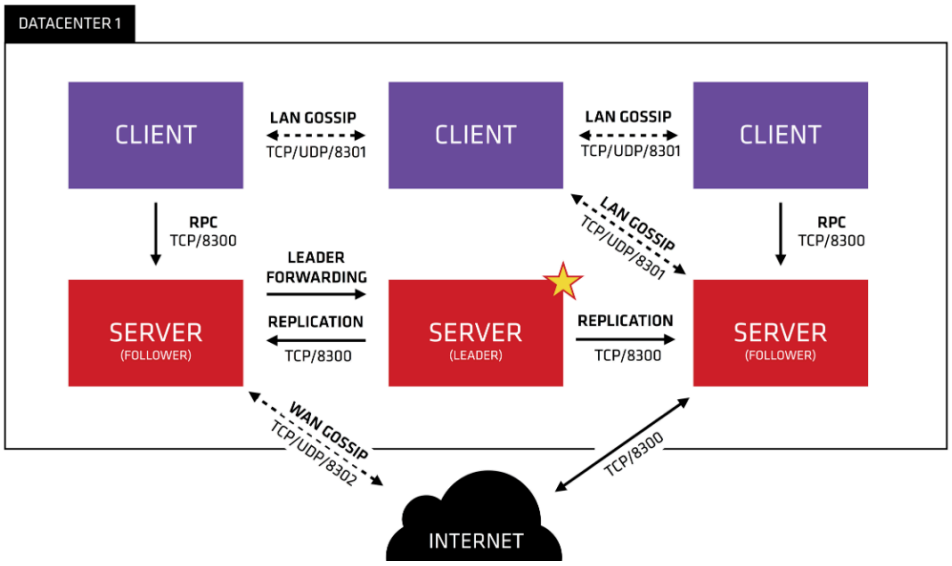
As of now, we have been making use of a single server and a client for our learning.

If the server goes down, then all of your data will be inaccessible and requests would fail.



2.2 Ideal Setup

In a production environment, you should have multiple sets of servers for high-availability.



2.3 Overview of Bootstrap Expect

Bootstrap Expect flag informs Consul of the expected number of server nodes and automatically bootstraps when that many servers are available.

If you have 3 servers that will be part of the cluster, bootstrap-expect should be 3.

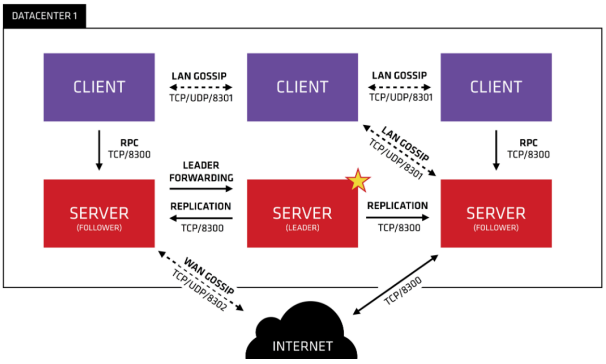
Module 3: Multiple Datacenter in Consul

3.1 Revising Concept of Datacenter

A datacenter is a networking environment that is private, low latency, and high bandwidth.

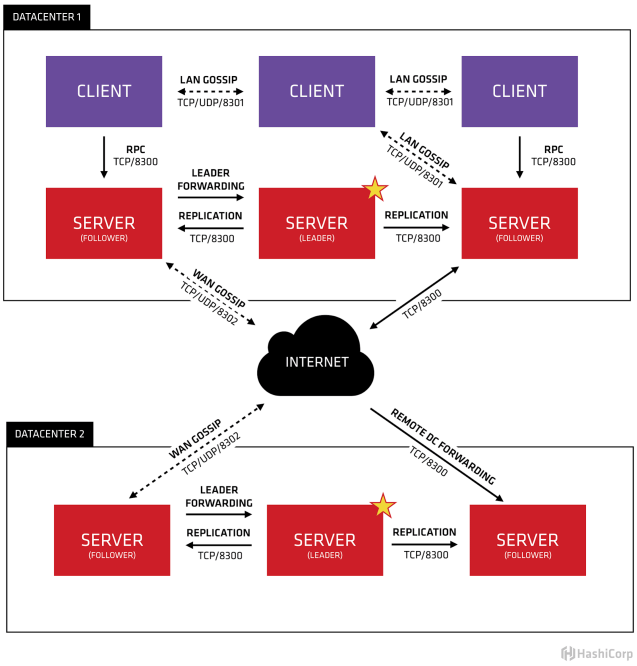
This excludes communication that would traverse the public internet.

Multiple Availability Zone within a single AWS region would be considered part of a single datacenter.



3.2 Multiple Datacenter approach

A large scale organization can be hosting their infrastructure across multiple cloud platforms and even on-premise.



3.3 Important Pointers

All server nodes must be able to talk to each other. Otherwise, the gossip protocol as well as RPC forwarding will not work.

In general, data is not replicated between different Consul datacenters. When a request is made for a resource in another datacenter, the local Consul servers forward an RPC request to the remote Consul servers for that resource and return the results.

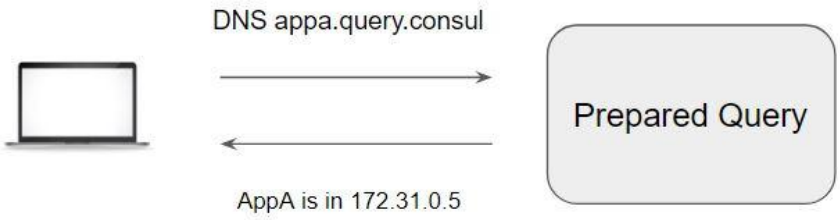
.

Module 4: Overview of Prepared Query

4.1 Understanding Prepared Query

Prepared queries are objects that are defined at the datacenter level.

Once created, prepared queries can then be invoked by applications to perform the query and get the latest results.

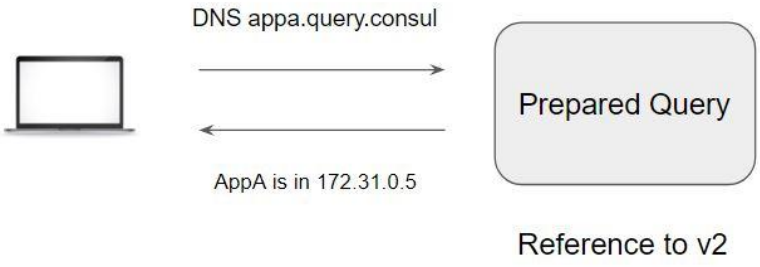


4.2 Use Case 1 - Multiple Versions

Let's assume there are multiple versions of AppA: v1 and v2

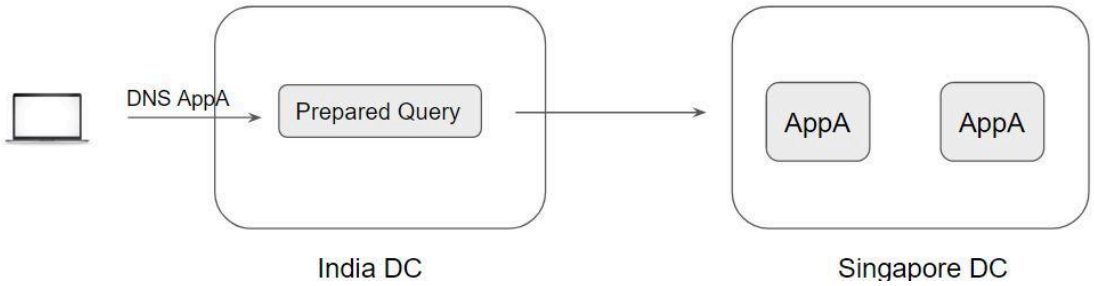
Within the prepared query, we can explicitly specify a version that needs to be returned to the client.

There are no configuration changes required at the client-side.



4.3 Use Case 2 - Failover Policy

You can contact other data centers once there are no healthy instances in the local datacenter.



4.4 Steps Involved in Failover Policy

Consul servers in the local datacenter will attempt to find healthy instances of the "AppA" service within the dc1

If none are available locally, the Consul servers will make an RPC request to the Consul servers in "dc2" to perform the query there.

Finally, an error will be returned if none of these datacenters had any instances available.

Module 5: Backup & Restore

All servers write to the -data-dir before commit on write requests

Consul provides the snapshot command that saves a point-in-time snapshot of the state of the Consul servers. Some of the important data includes:

- Key-Value entries
- the service catalog
- prepared queries
- sessions
- ACLs

Following commands can be used to take backup and restore data from backup.

Commands	Description
consul snapshot save backup.snap	Takes backup and stores it to backup.snap file.
consul snapshot restore backup.snap	Restore data from the backup.

Important Note:

Snapshots will not be saved if the datacenter is degraded or if no leader is available.

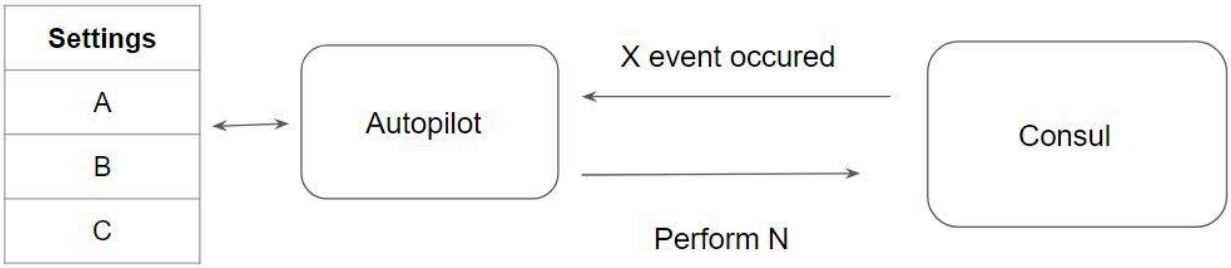
It is possible to run the snapshot on any non-leader server using stale consistency mode. This means that a very small number of recent writes may not be included.

It is recommended to regularly take a backup in an automated way.

Module 6: Overview of AutoPilot

6.1 Overview of AutoPilot Pattern

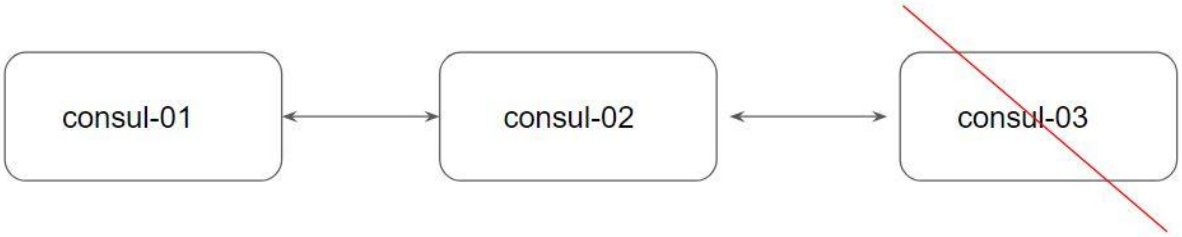
The autopilot pattern automates in code the repetitive and boring operational tasks of an application, including startup, shutdown, scaling, and recovery from anticipated failure conditions for reliability, ease of use, and improved productivity.



6.2 Use-Case 1 - Dead Servers

It will take 72 hours for dead servers to be automatically reaped or an operator must write a script to consul force-leave

Autopilot helps prevent these kinds of outages by quickly removing failed servers as soon as a replacement Consul server comes online.



6.3 Use-Case 2 - Server Stabilization Time

When a new server is added to the data center, there is a waiting period where it must be healthy and stable for a certain amount of time before being promoted to a full, voting member.

This is defined by the ServerStabilizationTime autopilot's parameter and by default is 10 seconds.

In case your configuration requires a different amount of time for the node to get ready, you can tune the parameter and assign it to a different duration.

6.4 Additional Features for Enterprise

There are some additional features of autopilot that are available in Consul Enterprise

- Redundancy Zones
- Automated upgrades

Module 7: Automatically Joining Servers

7.1 Approaches to Join Servers

There are multiple options for joining the servers.

- Specify a list of servers with -join and start_join options.
- Specify a list of servers with -retry-join option.
- Use automatic joining by tag for supported cloud environments with the -retry-join.

You can choose the method which best suits your environment and specific use case.

7.2 Approaches to Join Servers

We have been using this approach for our practicals.

consul join SERVER-NODE

If the server is not available, this command will fail and you will see your cluster is not created.

7.3 Approach 2 - Retry Join

Retry Join is similar to -join but allows retrying a join until it is successful.

consul agent -retry-join "10.0.4.67"

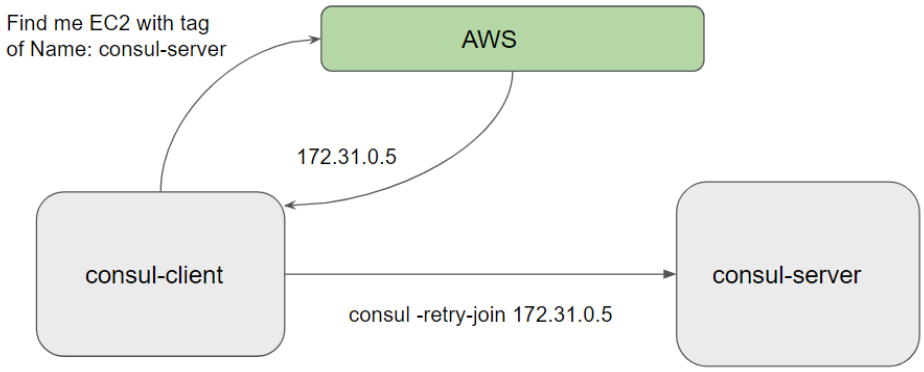
This is useful for cases where you know the address will eventually be available

retry_join could be more appropriate to help mitigate node startup race conditions when automating a Consul cluster deployment.

.

Module 8: Cloud Auto-Join

retry-join accepts a unified interface using the go-discover library for automatically joining a Consul datacenter using cloud metadata.



Working Steps:

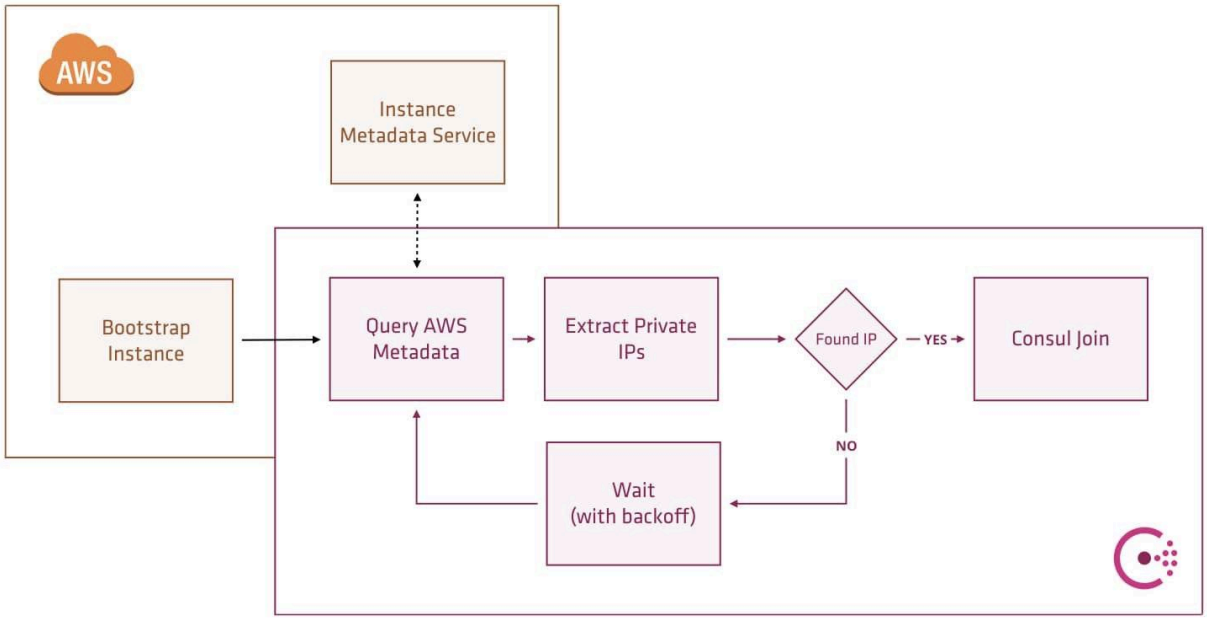
The instance bootstraps and installs consul

Init system starts consul with the configuration to join via EC2 metadata

On start, consul queries the EC2 metadata service with ec2:DescribeInstances to list all instance tags

Consul extracts the private IP addresses of other EC2 instances which have the configured tag name and tag value from the metadata

Consul runs consul join on those private IP addresses



Module 9: Cloud Logs

9.1 Consul Monitor

The monitor command is used to connect and follow the logs of a running Consul agent.

The amount of logged data depends on the overall logging level.

Available log levels are:

- Trace
- Debug
- Info (default)
- Warn
- Err

9.2 Consul Debug

The consul debug command monitors a Consul agent for the specified period of time, recording information about the agent, cluster, and environment to an archive written to the current directory.

```
[root@consul-01 consul.d]# consul debug
[WARN] Unable to capture pprof. Set enable_debug to true on target agent to enable profiling.
==> Starting debugger and capturing static information...
     Agent Version: '1.9.0'
          Interval: '30s'
          Duration: '2m0s'
            Output: 'consul-debug-1606483957.tar.gz'
           Capture: 'metrics, logs, host, agent, cluster'
==> Beginning capture interval 2020-11-27 13:32:37.086429377 +0000 UTC (0)
==> Capture successful 2020-11-27 13:32:37 +0000 UTC (1)
```

Module 10: Reloadable Configuration

Reloading configuration does not reload all configuration items. The items which are reloaded include:

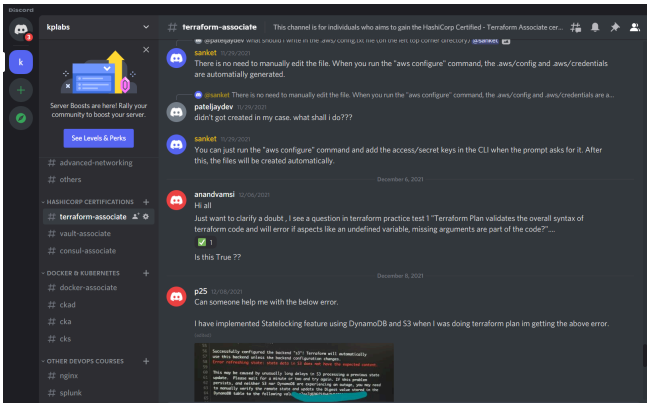
- Services
- ACL Tokens
- Configuration Entry Bootstrap
- Checks
- Discard Check Output
- HTTP Client Address
- Log level
- Node Metadata
- Watches

Join Our Discord Community

We invite you to join our Discord community, where you can interact with our support team for any course-based technical queries and connect with other students who are doing the same course.

Joining URL:

http://kplabs.in/chat



KPLABS Course

HashiCorp Certified: Consul Associate

Consul Enterprise

ISSUED BY
Zeal

REPRESENTATIVE
instructors@kplabs.in

Module 1: Overview of Consul Enterprise

Consul Enterprise includes capabilities that improve failure resilience, read scalability, and managing access in accordance with organizational policies.

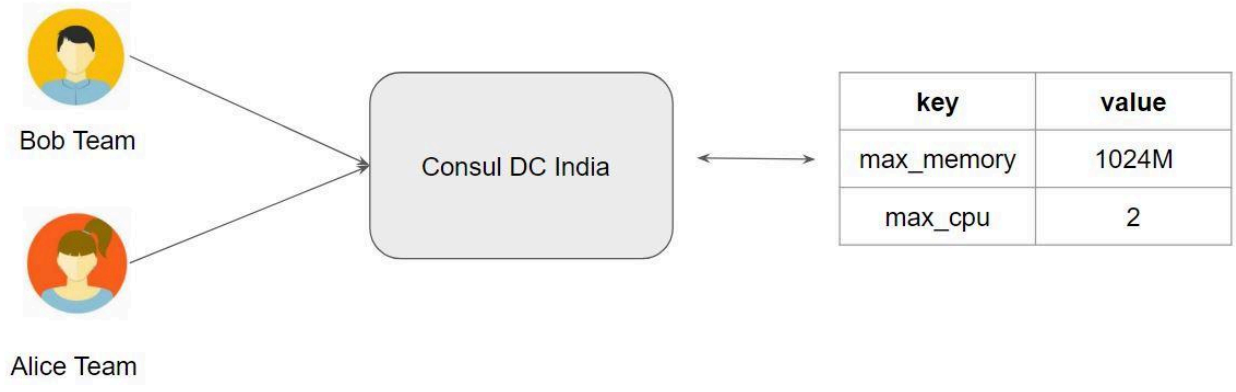
Some of the unique features include:

- Audit Logging
- Automated Backups
- Namespaces
- Redundancy Zones
- Automated Upgrades
- Enhanced Read Scalability

Module 2: Consul Namespaces

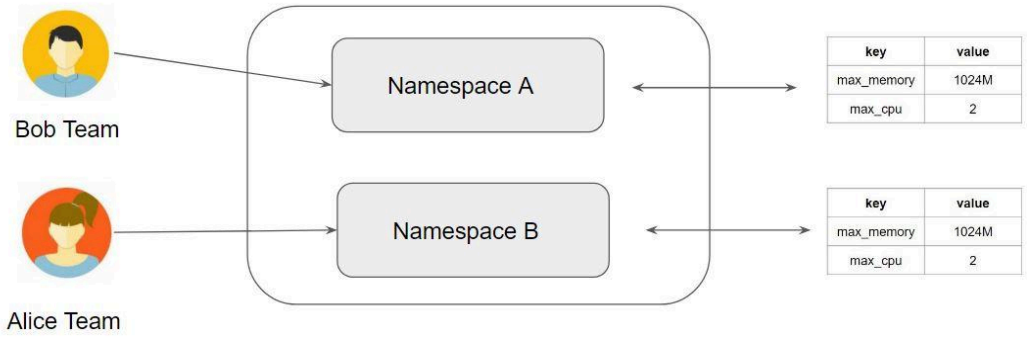
2.1 Understanding the Challenge

Namespaces provide separation for teams within a single organization enabling them to share access to one or more Consul datacenters without conflict.



2.2 Overview of Namespaces

Namespaces provide separation for teams within a single organization enabling them to share access to one or more Consul datacenters without conflict.



Module 3: Automated Backups in Enterprise

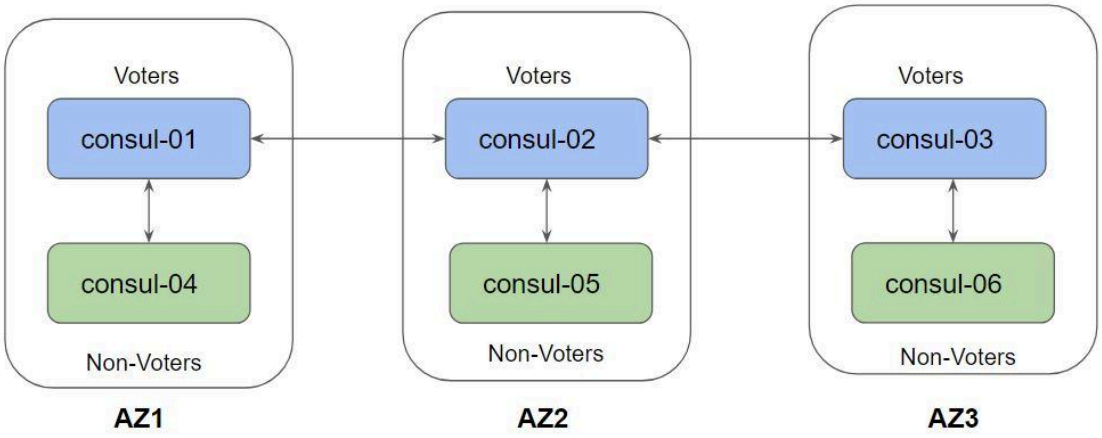
Consul Snapshot Agent is an enterprise-only feature that automatically manages on taking snapshots, backup rotation, and sending backup files offsite to Amazon S3

Consul Enterprise enables you to run the snapshot agent within your environment as a service like systemd.

```
[root@consul-01 ~]# consul snapshot agent
==> Consul snapshot agent running!
               Version: 1.8.6+ent
            Datacenter: (default)
              Interval: "1h0m0s"
                Retain: 30
                 Stale: false
       Local Scratch: /root
                  Mode: daemon
               Service: "consul-snapshot"
    Deregister After: "72h0m0s"
             Lock Key: "consul-snapshot/lock"
          Max Failures: 3
    Snapshot Storage: Local -> Path: "/root"
```

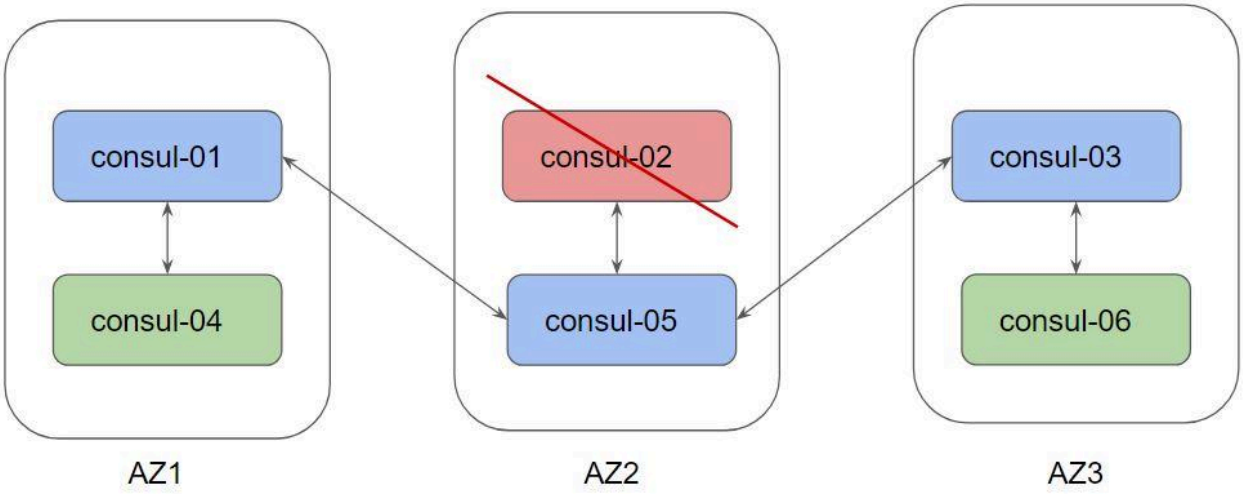
Module 4: Overview of Redundancy Zones

4.1 Overview of Redundancy Zones
Redundancy zones is a Consul autopilot feature that makes it possible to run one voter and any number of non-voters in each defined zone.



4.2 Use-Case

If only the voter is lost in an availability zone, the autopilot will promote the non-voter to voter automatically, putting the hot standby server into service quickly.



4.3 Important Note

Non-voting servers still receive data from the cluster replication, however, they do not take part in quorum election operations.