KPLABS Course

HashiCorp Certified: Consul Associate

Security

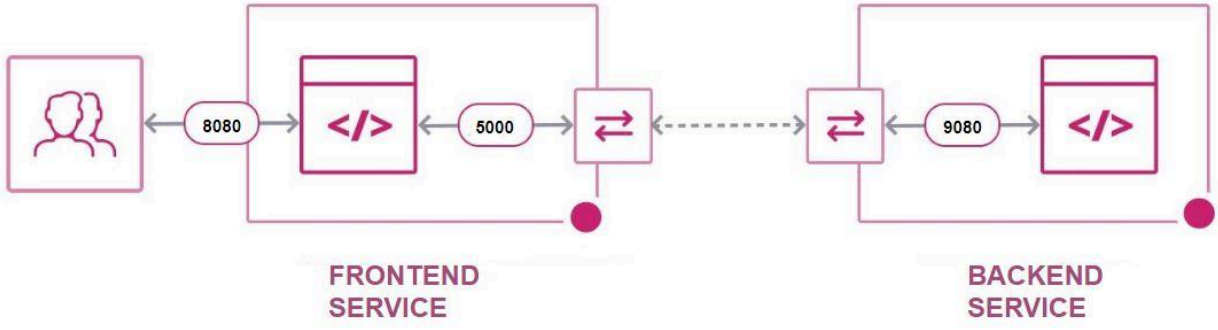
ISSUED BY
Zeal

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.

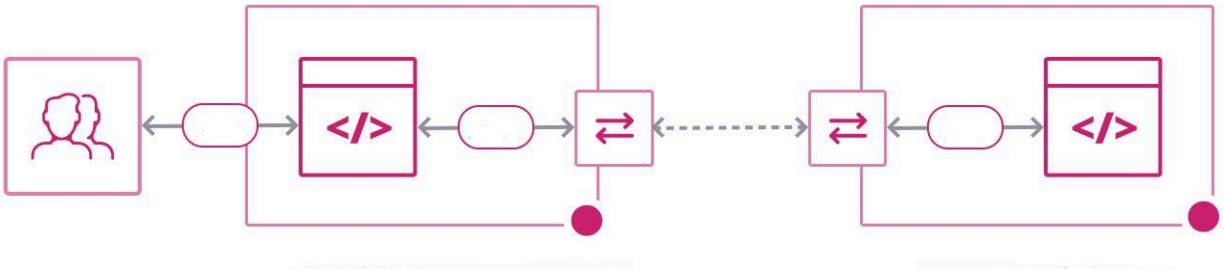
For any given request the first permission to match in the list is terminal and stops further evaluation

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.



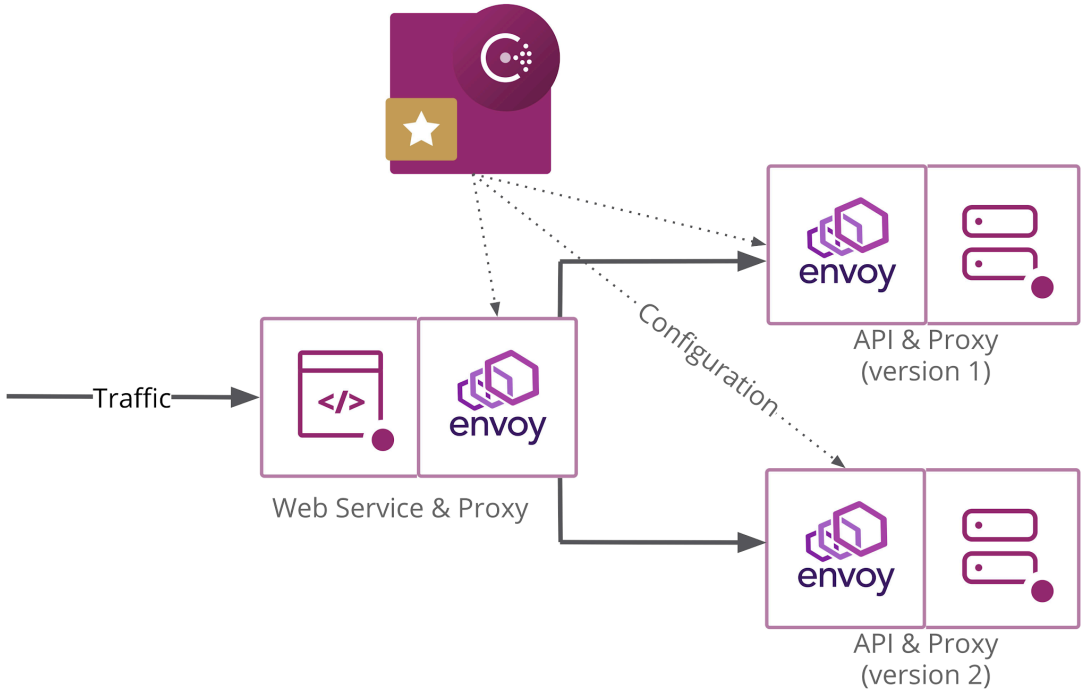
3.2 Overview of Envoy

Envoy is an open source edge and service proxy.

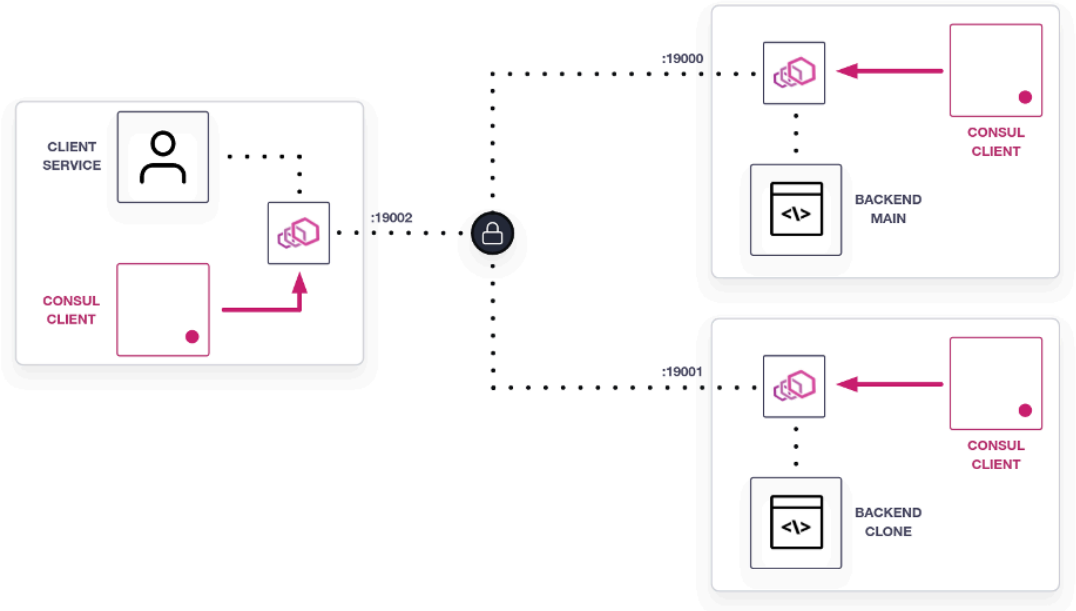
It comes with various additional features, some of these include:

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

Use-Case 1 - Traffic Splitting



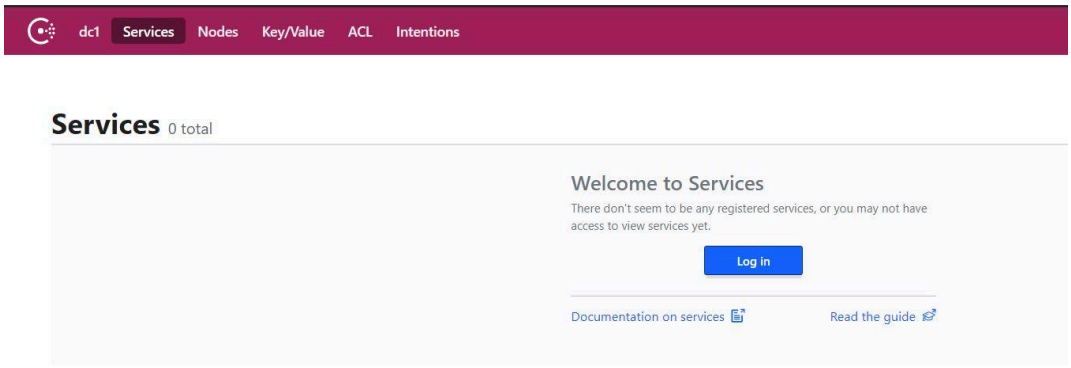
Use-Case 2 - Load Balancing



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.

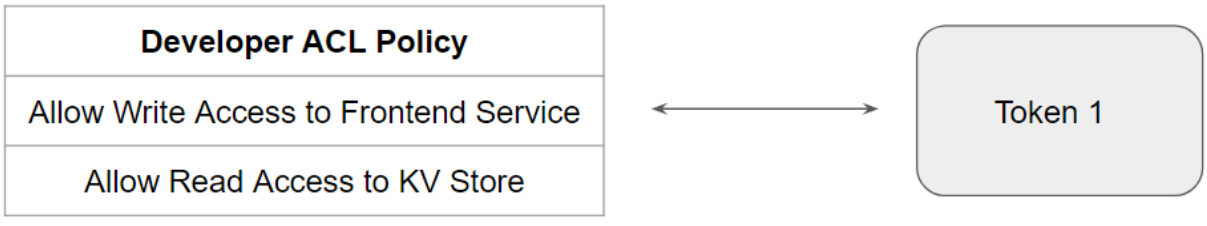
CONSUL_HTTP_TOKEN

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

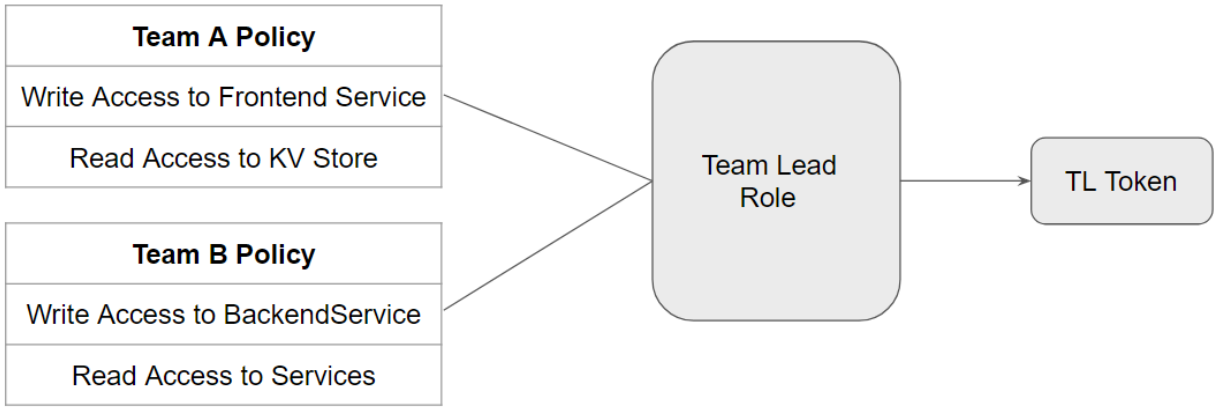
5.4 Scope for ACL Rules

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

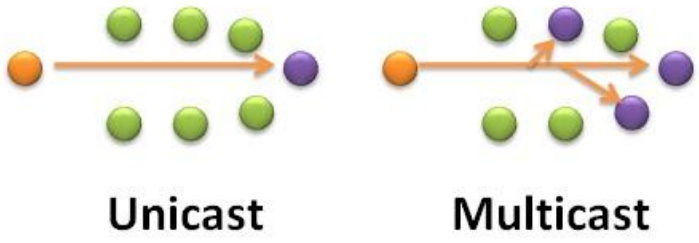
```
acl = {
  enabled = true
  default_policy = "deny"
  enable_token_persistence = true
  tokens {
    "agent" = "34c522d7-bce0-c27d-fc2e-69dc83e15487"
  }
}
```

Module 9: Overview of Gossip Protocol

9.1 Overview of Unicast and MultiCast

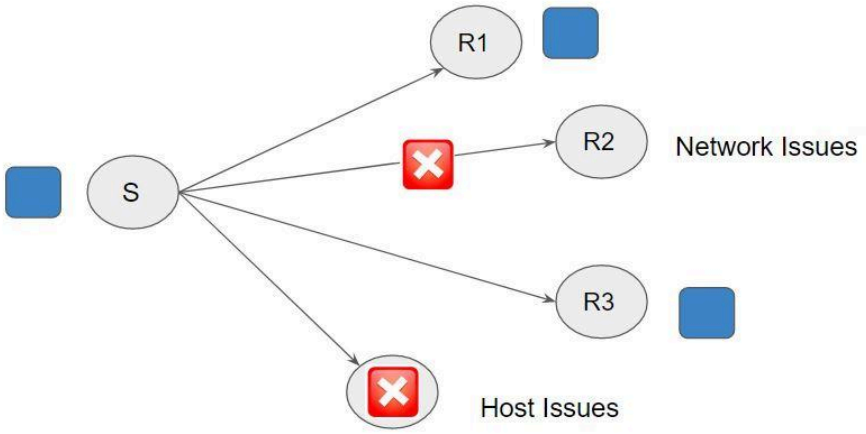
A Unicast transmission/stream sends IP packets to a single recipient on a network.

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

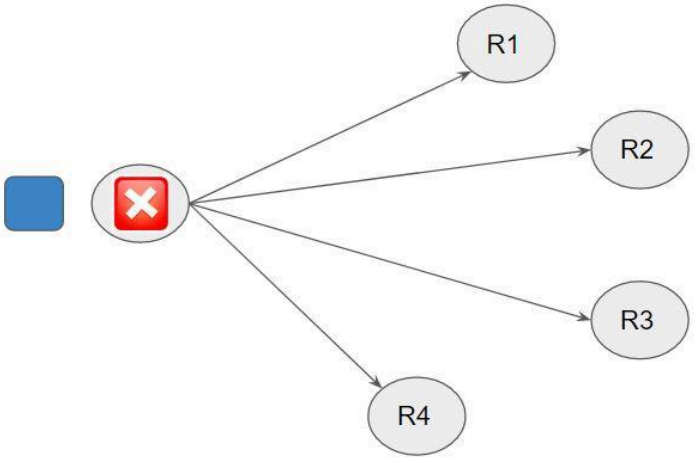


9.2 Multi-Cast Challenges

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.

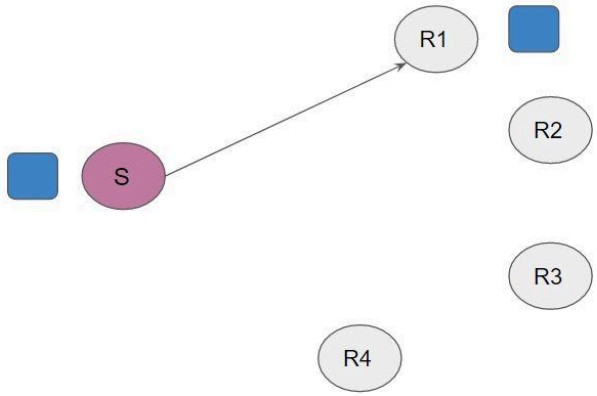


It can also be possible that the sender of the message went down.

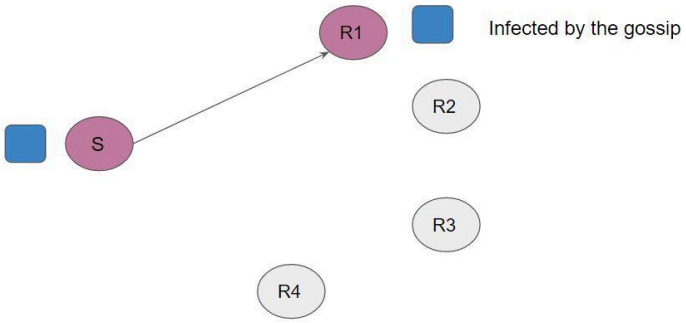


9.3 Gossip Protocol

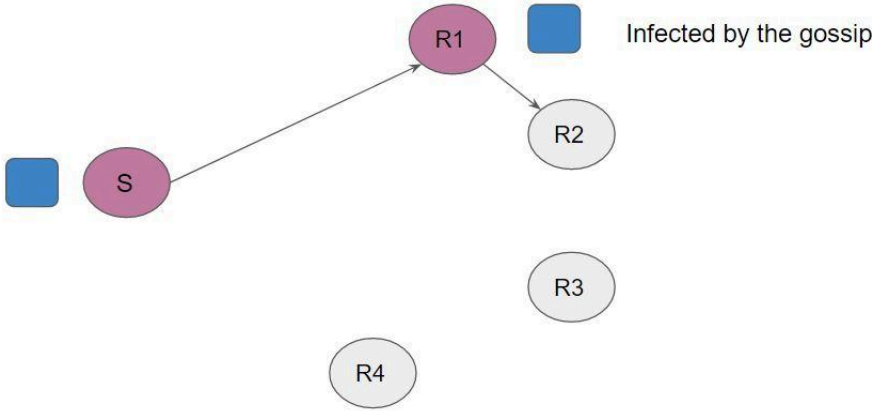
Data is periodically transmitted to random targets. In the below case, the sender has sent the message to R1.



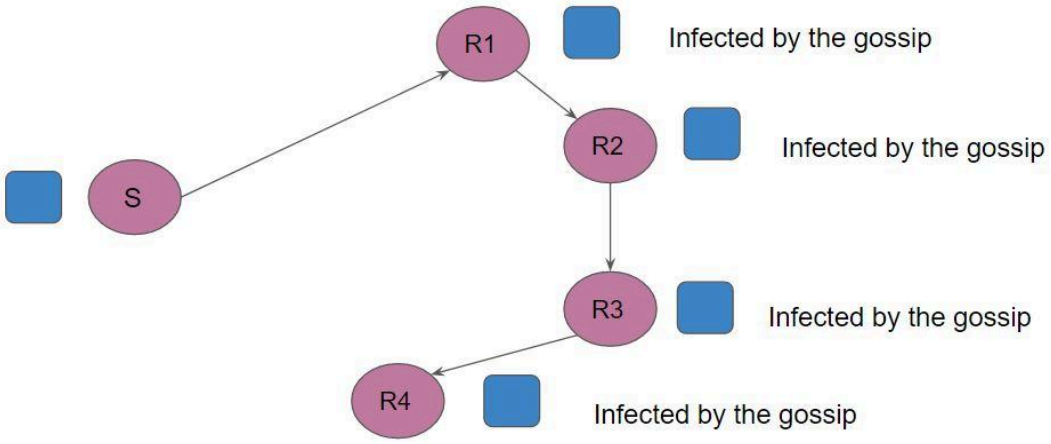
Once R1 receives the message, it is referred to as infected by the gossip.



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.

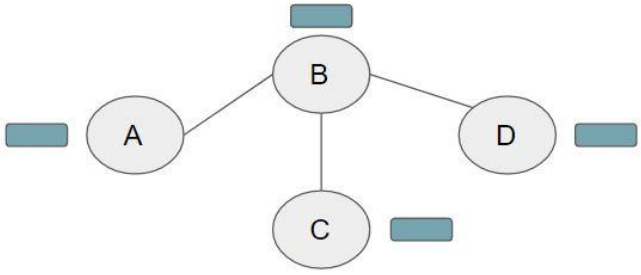


9.4 Understanding Gossip Protocol

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.

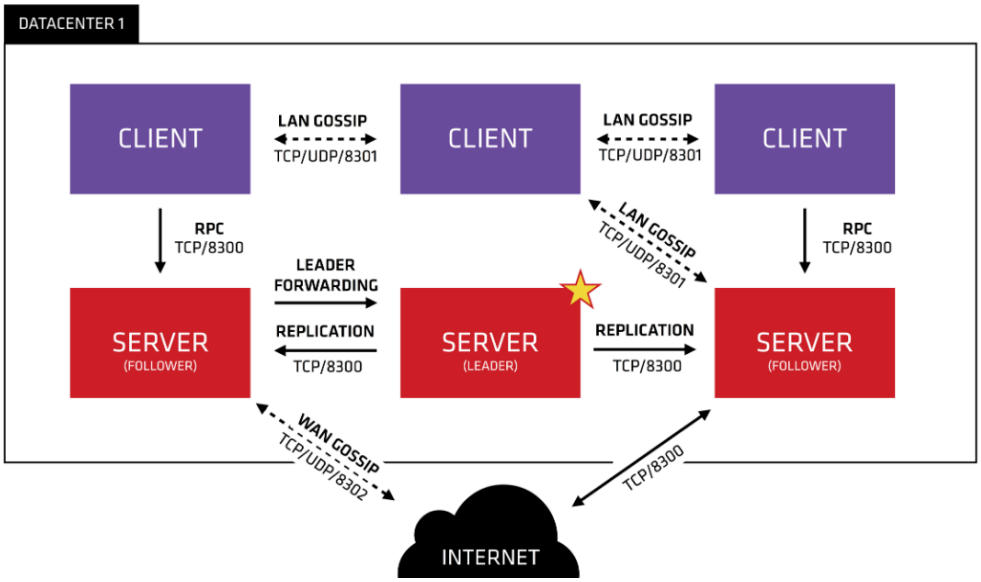


.

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

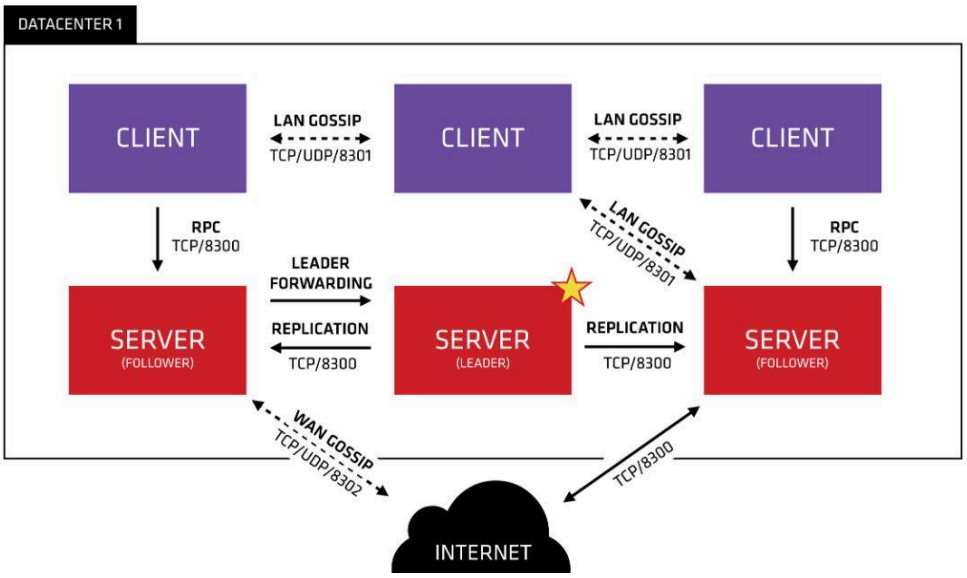
Gossip encryption can also be enabled on existing data centers but requires several extra steps.

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

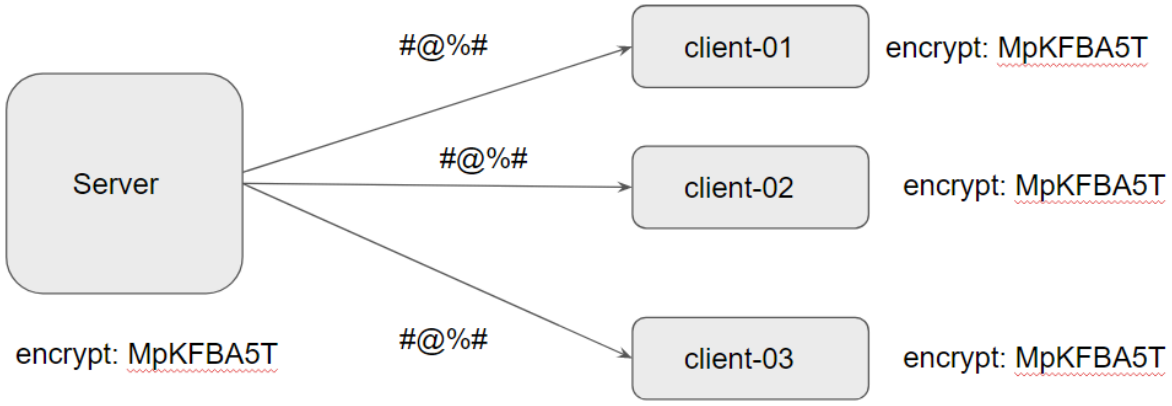


.

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="
```

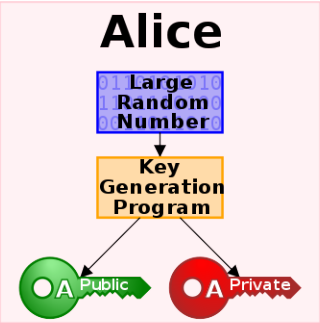
Module 14: Introduction to Asymmetric Key Encryption

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

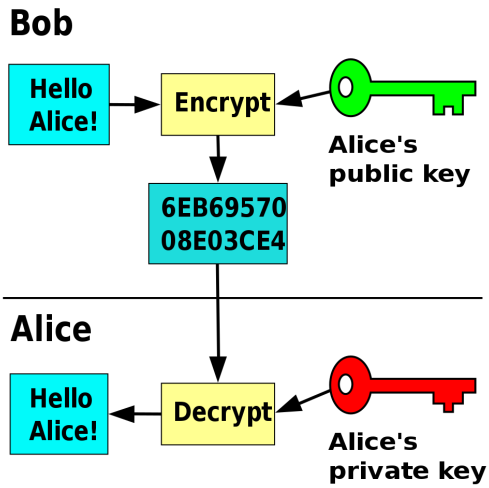
One key in the pair can be shared with everyone; it is called the public key. The other key in the pair is kept secret; it is called the private key.

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



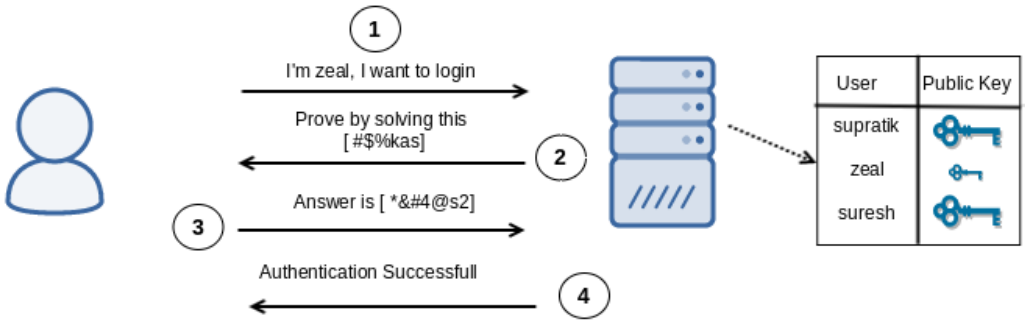
Step 2: Encryption and Decryption



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:

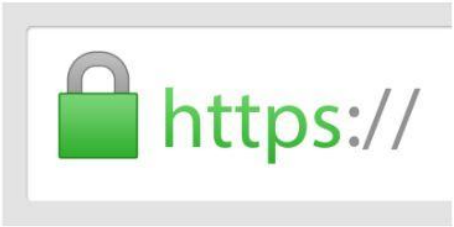
- PGP
- SSH
- Bitcoin
- TLS
- S/MIME

Module 15: Understanding SSL/TLS

HTTPS is an extension of HTTP.

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

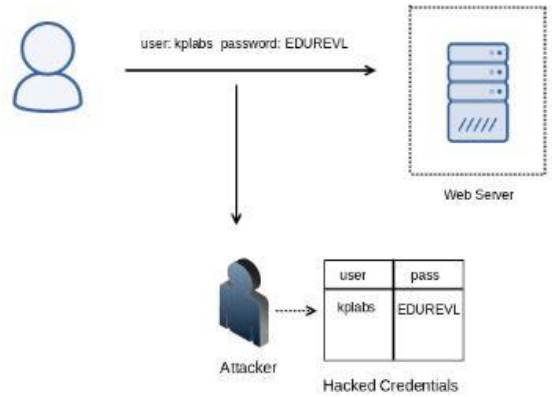
The protocol is therefore also often referred to as HTTP over TLS or HTTP over SSL.



Scenario 1: MITM Attacks

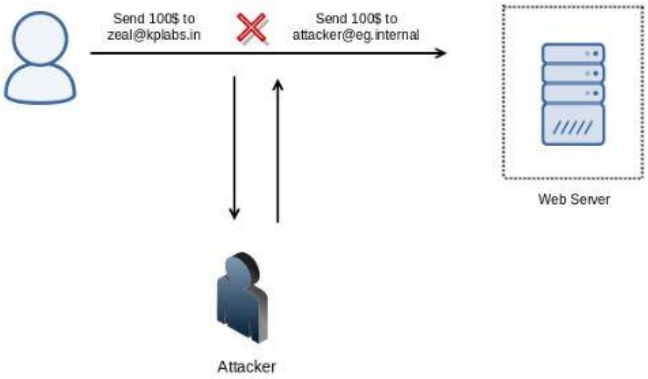
User is sending their username and password in plaintext to a Web Server for authentication over a network.

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.



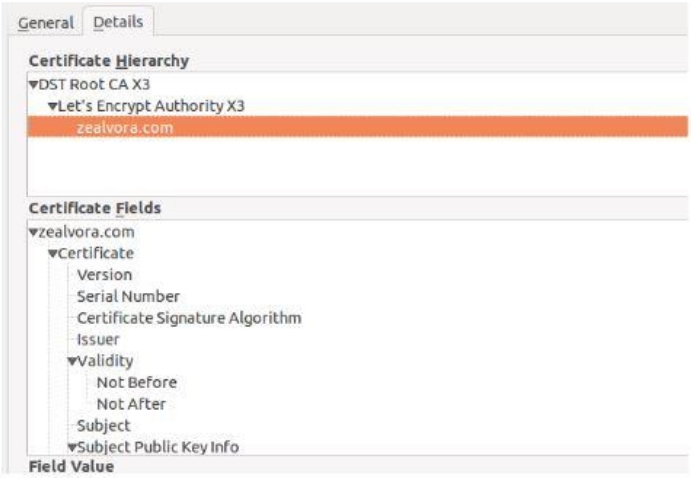
Introduction to SSL/TLS

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.

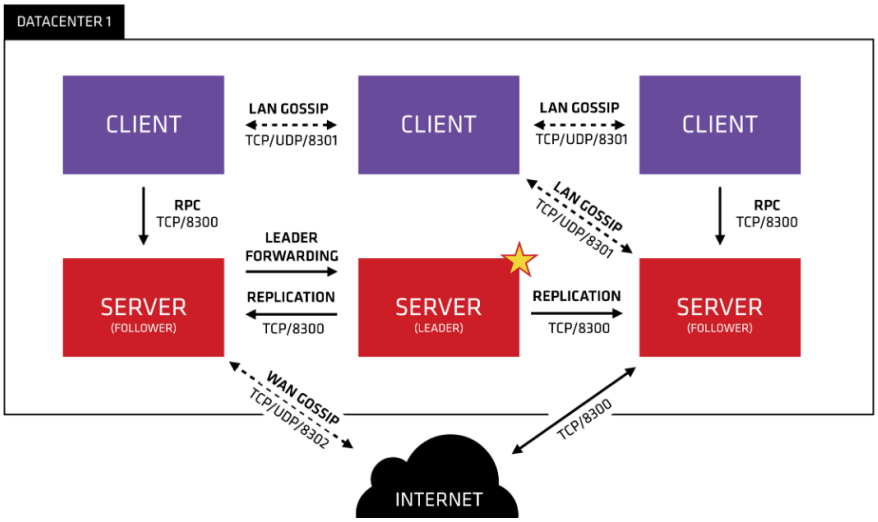
The following snapshot shows sample web-server configuration:

```
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.

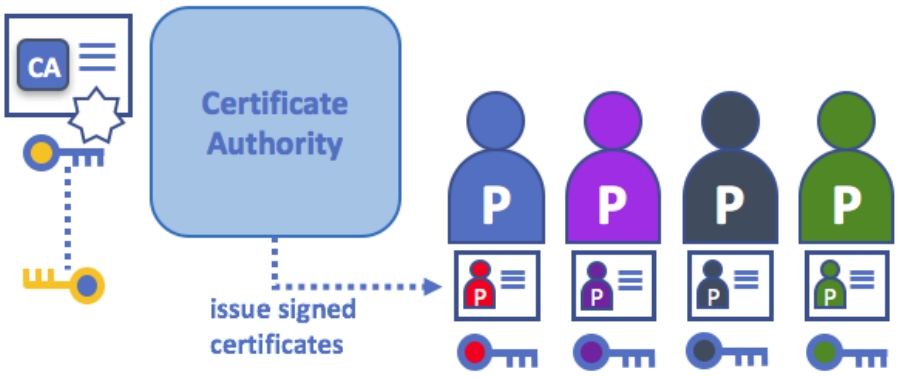
Three steps:

- 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
}
```

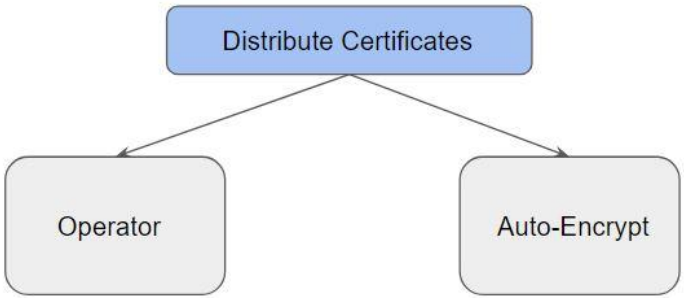
Client Configuration

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.

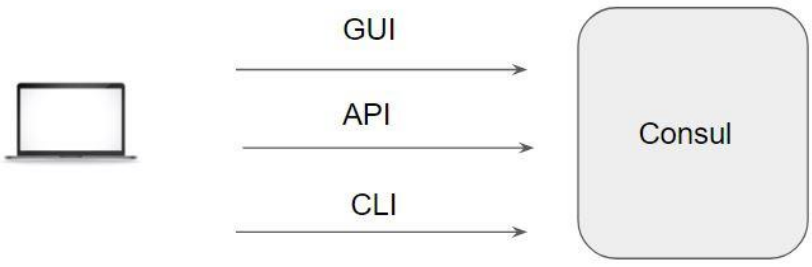
Now question is, how will he parse all this data?

Module 18: HTTP API in Consul

11.1 Overview of Consul Interface

There are multiple ways to connect with Consul:

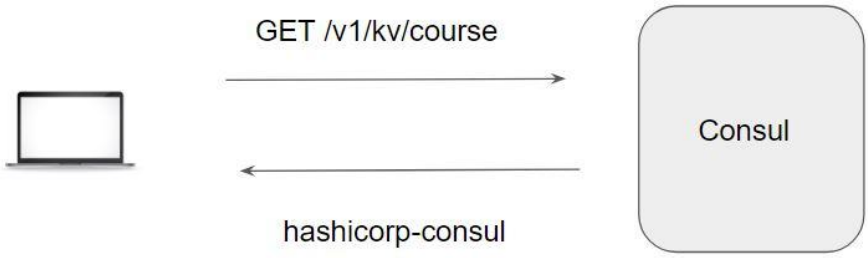
GUI, API as well as CLI



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
```