HashiCorp Consul

Consul Deployment & Operations

Agenda

Consul Enterprise Path to Production

Week 1	Week 2	Week 3	Week 4	Week 5
Kickoff	**Consul Foundations**	**Consul Deployment & Operations**	**Office Hours**	**Advanced Concepts**
● Consul overview & Architectural deep dive	● Service Discovery, Health Checks & Consul DNS	● Deployment, Autopilot, Backup, Telemetry & Monitoring	● Open forum for Q&A	● Federation, Network Segments, Namespaces, & Runbooks ● Exit Ramp & Operational Readiness Checklist (asynchronous)

01

Deployment Patterns

Recommended Patterns

Immutable Builds

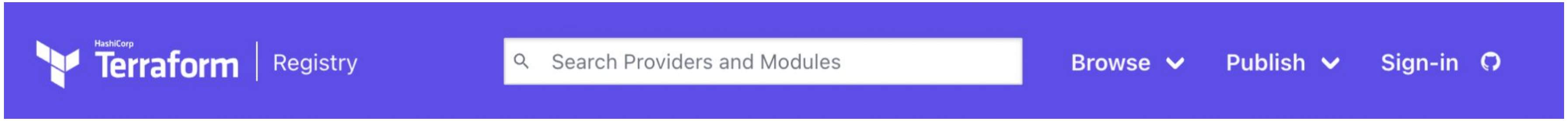
- Tools like Packer can be used to build immutable machine images for blue/green deployment using existing CI/CD orchestration
- This approach can streamline the lifecycle processes for managing Consul

Configuration Management

- Configuration Management tools and patterns can be used for installation, upgrade, and configuration of Consul
- Autopilot can be leveraged for in-place upgrades

Terraform Modules

Quickly deploy Consul cluster(s) based on reference architecture



Terraform modules provide an immutable foundation for deployment of Consul in Cloud Providers & Cloud Managed Kubernetes

- Consul Enterprise GCP Module (VM)
- Consul Enterprise Azure Module (VM)
- Consul Enterprise AWS Module (VM)
- Azure AKS Terraform Module (K8S)
- Google GKE Terraform Module (K8S)
- AWS EKS Terraform Module (K8S)

Migration from Consul OSS to Enterprise

- Once an instance has been upgraded to Consul Enterprise it cannot be downgraded to OSS
- Consul Enterprise 1.10.0+ requires license files be loaded from configuration or environment variables
- In-place migration via standard upgrade procedure
    a. Backup instance via Consul snapshot
    b. Identify Leader Node and leave for last
    c. Replace binary on follower node
    d. Add licensing configuration and cycle node
    e. Repeat on all follower nodes
    f. Replace binary and add licensing to leader node

Upgrades

- Major upgrades should occur **at least 2X per year** to stay within **N-2 major releases** version support window
- Automation of the update process is recommended to ensure ease of operations and keep Consul patched with current updates
- Prior to a production upgrade:
  a. Review version specific upgrade guide
  b. Review changelog
  c. Test version in QA environment
  d. Take a snapshot prior to any upgrade

Upgrading Consul on Kubernetes

- Consul on Kubernetes Upgrade Guide
- Review Helm Compatibility Matrix to see if a Helm chart upgrade is required
- Autopilot features are not available for Consul on Kubernetes
- If using Service Mesh (Consul Connect) do not restart all Consul clients at once

Consul Autopilot

Consul Autopilot

- Designed for automatic, operator-friendly management of Consul servers
- Functionality includes:
  - Dead server cleanup
  - Server Stabilization
  - Redundancy Zone Tags
  - Automated upgrades
- Enabled by default in Consul Enterprise

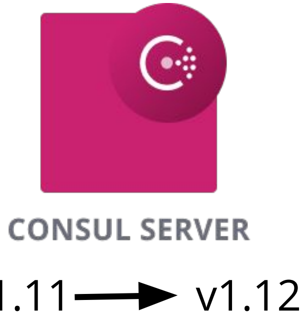
Autopilot Best Practices

- ☆ Autopilot configuration & on/off should be consistent across all cluster nodes
- ☆ Autopilot features can be configured independently, and can be enabled or disabled at any time
- ☆ Autopilot configuration is persisted in the Raft database and included in Consul snapshots
- ☆ Take a snapshot after any changes to the autopilot configuration

Consul Upgrade Patterns

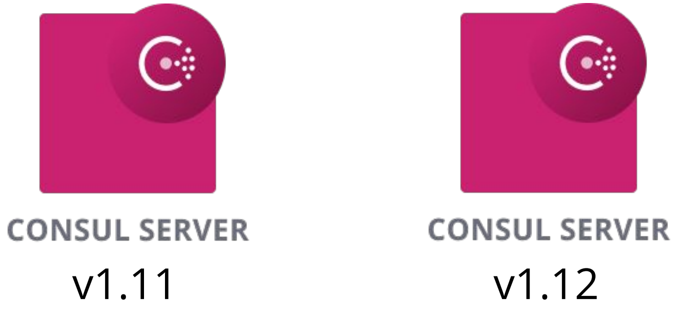
Rolling Restart

1. Replace the old binary
2. Rolling restart of the server
3. Check health
   a. `consul version`
   b. `consul members`
   c. `consul operator raft peer-list`
4. Repeat

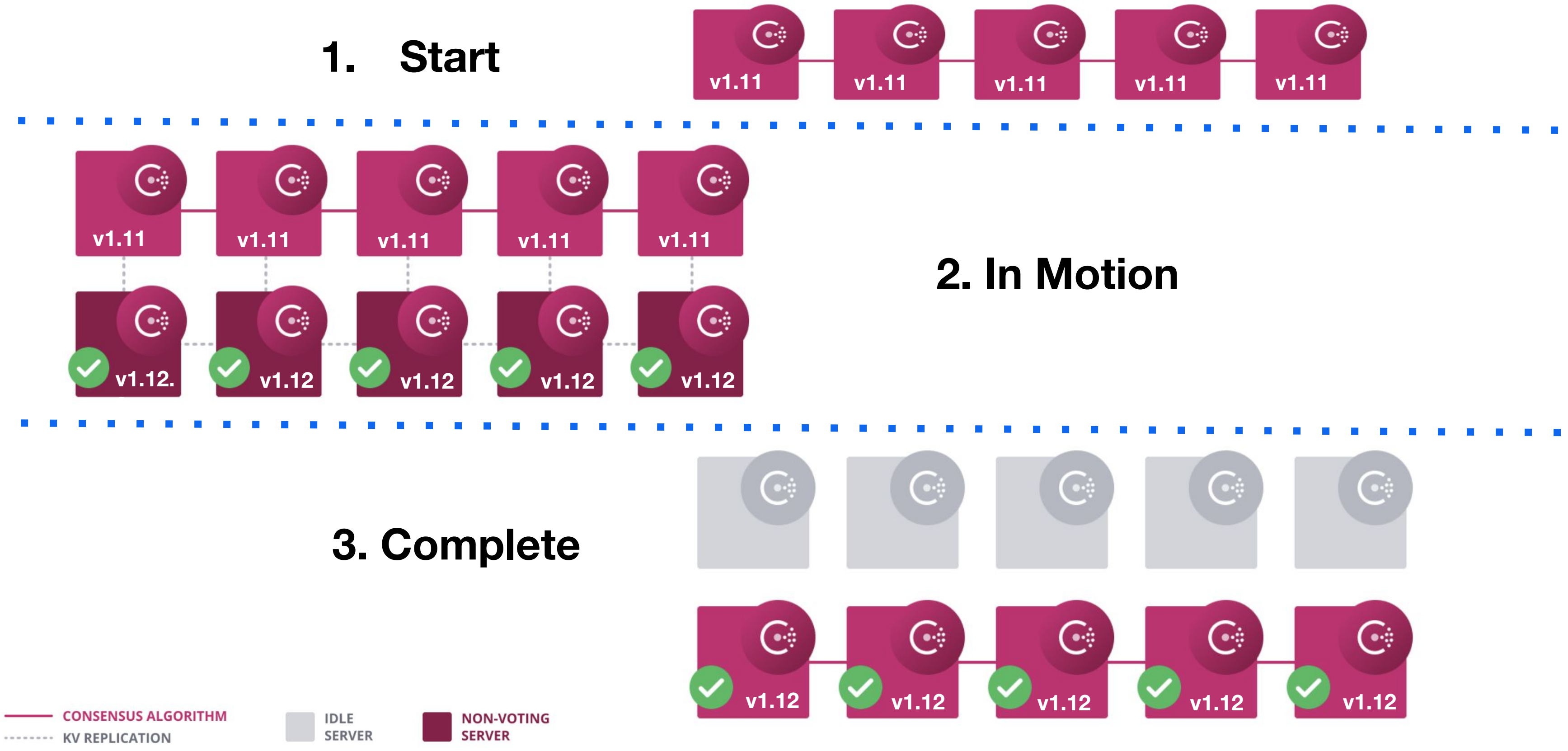


Add & Remove Servers

1. Add a new server to the existing peer set
2. Gracefully remove one of the followers
3. Repeat



Automated Upgrades with Autopilot



03

Consul Agent

Consul Agent

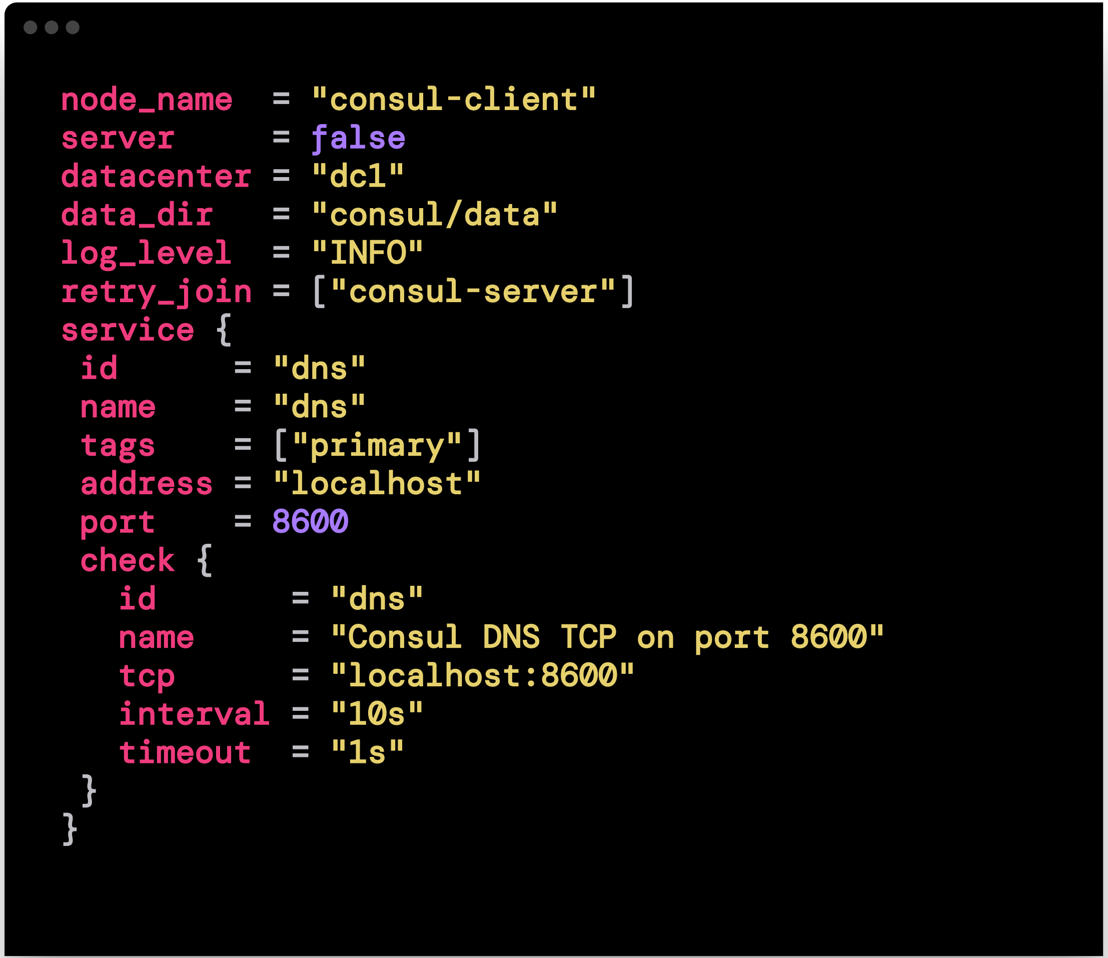
- Core process of Consul that runs on every node that is part of a Consul cluster
- Maintains membership information, registers services, runs checks, & responds to queries
- Runs in either client or server mode

Consul Agent

- Consul agent gets deployed on **every** Consul **server node**
- Consul agent gets deployed on **every client node** that participates in service discovery, service mesh, and/or active health checks
- Is a daemon-set thats gets deployed on **every** Kubernetes **worker node**
- Only non-default values must be set in agent configuration file
- Configuration can be read from multiple files

Consul Agent Configuration

- Client Node
- Service Registration
- Health Checks



```
node_name  = "consul-client"
server     = false
datacenter = "dc1"
data_dir   = "consul/data"
log_level  = "INFO"
retry_join = ["consul-server"]
service {
 id      = "dns"
 name    = "dns"
 tags    = ["primary"]
 address = "localhost"
 port    = 8600
 check {
   id       = "dns"
   name     = "Consul DNS TCP on port 8600"
   tcp      = "localhost:8600"
   interval = "10s"
   timeout  = "1s"
 }
}
```

Kubernetes Pod Resource Manifest

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dashboard
spec:
  serviceAccountName: dashboard # Authenticate Kube workload
with Consul
  containers:
    - name: dashboard
      image: hashicorp/dashboard-service:0.0.4
      ports:
        - containerPort: 9002
      env:
        - name: COUNTING_SERVICE_URL
          value: "http://counting:9001" # Transparent Proxy
automatically configures mesh routing
---
apiVersion: v1
kind: ServiceAccount
metadata:
  # Service Account used to authenticate with Consul ACL system
  # Service Account name becomes Consul service name (unless
otherwise annotated)
   name: dashboard
```

Enable Gossip Traffic Encryption

The Consul agent **needs** an encryption key when starting

- Key can be set with the encrypt parameter in agent config
- Key can also be placed in a separate config file with only the encrypt field, Consul agent can merge multiple config files
- Keys must be 32-bytes, Base64 encoded
- Consul `keyring` is used for rotation and lifecycle management of encryption keys

04

ACL Configuration

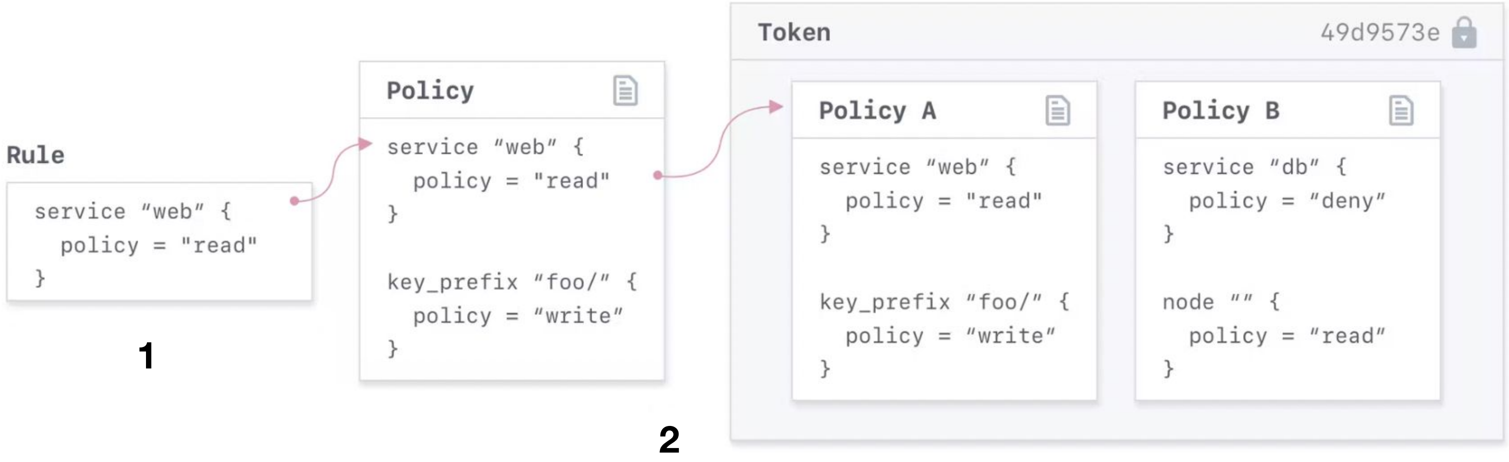
Consul ACLs

- Consul ACLs are disabled by default
- Consul ACLs control access to all Consul components (UI, API, CLI) & authorize service-to-service and agent-to-agent communication
- ACL System Components:
    - *Token*: Bearer token used during the UI, CLI and API request
    - *Policy*: Grouping of rules that determine fine-grained rules to be applied to token
    - *Roles*: A collection of policies and/or service identities applied to token(s)
    - *Service Identities / Node Identities*: Shorthand terms for appropriate minimum ACL policy for a service or node

ACL Creation

1. Authentication rules are specified to define a policy

2. ACL Administrator generates and links a token to a policy



```
service "web" {
  policy = "read"
}
```

```
service "web" {
  policy = "read"
}

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

```
service "web" {
  policy = "read"
}

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

```
service "db" {
  policy = "deny"
}

node "" {
  policy = "read"
}
```

ACL Implementation

1. Authentication rules are specified to define a policy
2. ACL Administrator generates and links a token to a policy
3. Tokens are distributed and incorporated into services
4. Agents and services present tokens when making requests
5. Consul evaluates token for valid permissions

Bootstrapping Consul ACLs

1. Enable ACLs on agent config file and restart Consul

```
$ cat agent.hcl

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

2. Create the initial bootstrap token

```
$ consul acl bootstrap
```

3. Apply individual tokens to agents

    a. Create the agent policy

    b. Create the token with the newly created agent policy

    c. Add the token to the agent

Token Example

Token for DNS

Example policy provides read privileges for all services, nodes, and prepared queries

```
$ cat dns-request-policy.hcl

namespace_prefix "" {
  policy = "write"
 node_prefix "" {
   policy = "read"
 }
service_prefix "" {
  policy = "read"
}
# only needed if using prepared queries
 query_prefix "" {
   policy = "read"
 }
}

$ consul acl policy create -name "dns-requests" -rules
@dns-request-policy.hcl

$ consul acl token create -description "Token for DNS
Requests" -policy-name dns-requests

$ consul acl set-agent-token default "<dns token>"
```

ACLs Best Practices

- Implement least privileges policy for every ACL token generated
- Use exact match resource rules to achieve least privilege patterns
- Don’t reuse tokens, generate a unique token per service
- Use allowlist (default deny) to force explicit anonymous access or token usage for all requests
- Reuse policies and roles for similar environments & services
- Don't use bootstrap token or a global token for management
- Rotate tokens on a regular basis

Important Token Considerations

Federation - Global vs. Local Tokens

- Global tokens are the default type
- Local tokens are not available or valid in federated datacenters
- Certain Consul features require global token, such as Mesh Gateway service tokens and replication tokens

The anonymous token policy is NOT additive

- ACLs are not additive with the anonymous token policy
- If read catalog permissions is granted to the anonymous token, read capabilities must be explicitly granted to other tokens

05

Backup & Disaster Recovery

Consul Backups

- Snapshots are Consul's primary backup & DR solution
- Snapshots are atomic, point-in-time, datacenter specific copies of Consul state
- Consul Snapshot Agent allows for scheduled automatic process
- Configure the Consul Snapshot Agent interval to meet desired RPO
- By default, snapshots run from the cluster leader

Snapshot Restore

- Is a turbulent process, all communication with Consul halts until snapshot is restored
- Is not selective to a feature or data element, is an all-or-nothing process
- Only needs to be run once, from cluster leader node
- Before performing make sure cluster is stable and has a leader
- Is not designed to handle server failures when process is running

Consul on Kubernetes

- Consul on Kubernetes requires backing up 4 essential secrets:
    - The last active Consul ACL bootstrap token
    - The last active Consul CA cert
    - The last active Consul CA key
    - The last active gossip encryption key
- Without these 4 secrets you **cannot recover** from a disaster
- These secrets need to be secure and stored **outside** the Kubernetes secrets engine

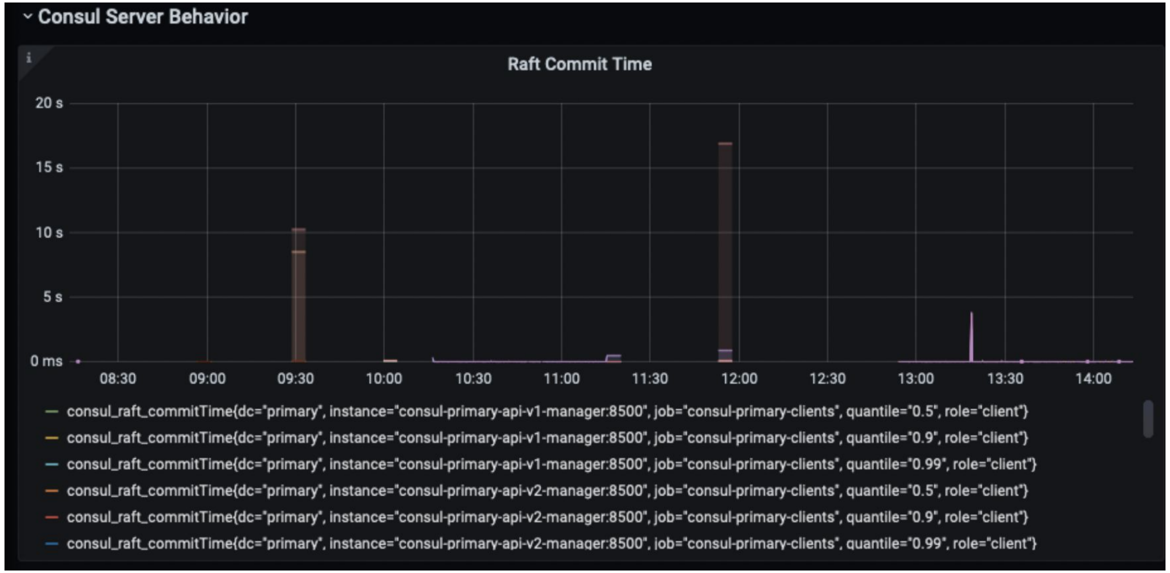
06

Telemetry & Monitoring

Monitoring Consul

Use a multi-layered approach to monitor the state and health of Consul datacenter(s)

- Consul CLI and API for initial and manual use
- Visualize metrics for real-time monitoring
- Collect and store metrics for comparison over time

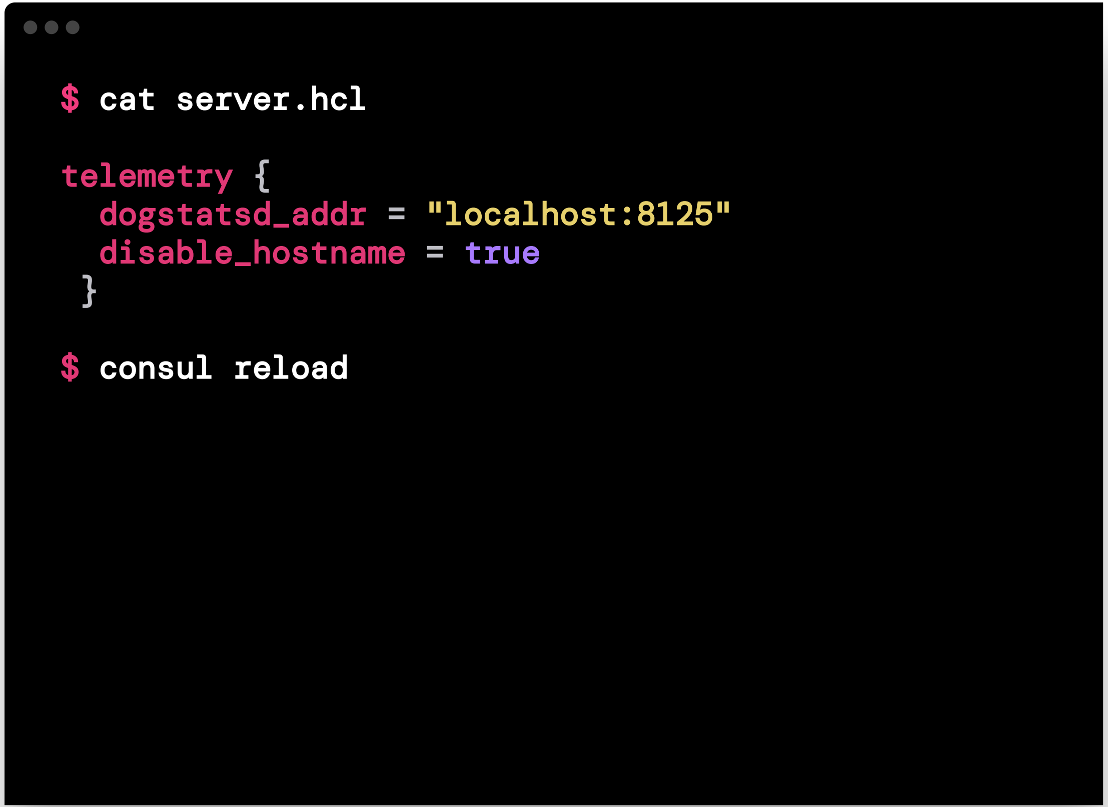


```
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v1-manager:8500", job="consul-primary-clients", quantile="0.5", role="client"}
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v1-manager:8500", job="consul-primary-clients", quantile="0.9", role="client"}
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v1-manager:8500", job="consul-primary-clients", quantile="0.99", role="client"}
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v2-manager:8500", job="consul-primary-clients", quantile="0.5", role="client"}
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v2-manager:8500", job="consul-primary-clients", quantile="0.9", role="client"}
— consul_raft_commitTime{dc="primary", instance="consul-primary-api-v2-manager:8500", job="consul-primary-clients", quantile="0.99", role="client"}
```

Methods for Collecting Metrics

- Enable Telemetry (recommended solution)
  - Send telemetry to a remote monitoring solution to gather data over time & spot trends
  - Metrics are aggregated on a 10s interval and retained for 1 minute
  - Supported telemetry agents:
    - Circonus
    - DataDog (via dogstatsd)
    - StatsD (via statsd, statsite, telegraf, etc.)
- API GET Request
  - Curl can be used to collect metrics via HTTP API
  - Example command [“curl http://127.0.0.1:8500/v1/agent/metrics”](http://127.0.0.1:8500/v1/agent/metrics)
  - Can be added to a script for monitoring agents like Prometheus via HTTP scraping
  - In production secure traffic via ACL token(s) & enabling TLS

Example DataDog Configuration



```
$ cat server.hcl

telemetry {
  dogstatsd_addr = "localhost:8125"
  disable_hostname = true
 }

$ consul reload
```

Monitoring Strategy

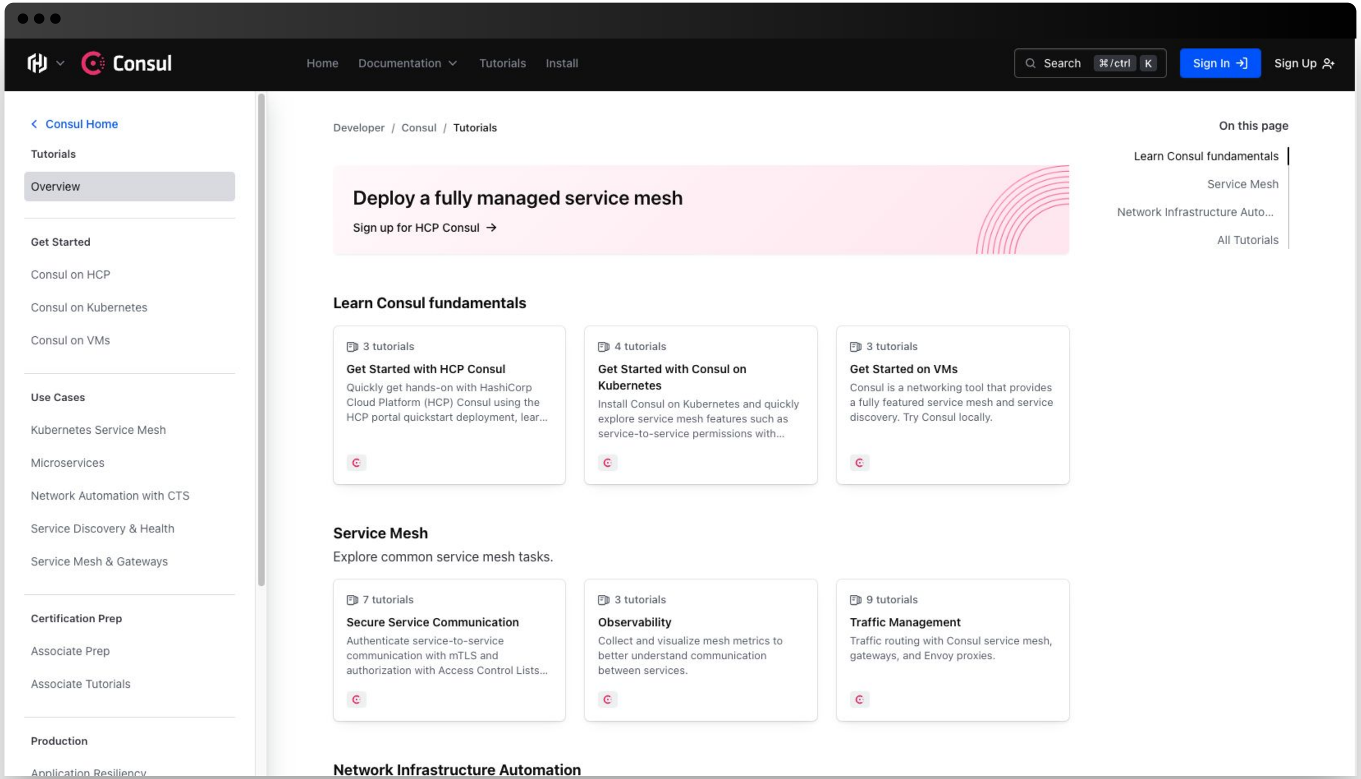
- **Consul Datacenter Health** - information about the Consul datacenter
  - Transaction timing
  - Leadership changes
  - Autopilot status
  - Garbage collection
- **Server Health** - information about each server node in the cluster
  - File handles
  - CPU usage
  - Network activity
  - Disk activity
  - Memory usage
- **Establish a baseline from a healthy cluster** for comparison purposes

Next Steps

Tutorials

https://developer.hashicorp.com/consul/tutorials

Step-by-step guides to accelerate deployment of Consul



Consul Home

Tutorials

Overview

Get Started

Consul on HCP

Consul on Kubernetes

Consul on VMs

Use Cases

Kubernetes Service Mesh

Microservices

Network Automation with CTS

Service Discovery & Health

Service Mesh & Gateways

Certification Prep

Associate Prep

Associate Tutorials

Production

Application Resiliency

Developer / Consul / Tutorials

Deploy a fully managed service mesh

Sign up for HCP Consul →

Learn Consul fundamentals

3 tutorials
Get Started with HCP Consul
Quickly get hands-on with HashiCorp Cloud Platform (HCP) Consul using the HCP portal quickstart deployment, lear...

4 tutorials
Get Started with Consul on Kubernetes
Install Consul on Kubernetes and quickly explore service mesh features such as service-to-service permissions with...

3 tutorials
Get Started on VMs
Consul is a networking tool that provides a fully featured service mesh and service discovery. Try Consul locally.

Service Mesh

Explore common service mesh tasks.

7 tutorials
Secure Service Communication
Authenticate service-to-service communication with mTLS and authorization with Access Control Lists...

3 tutorials
Observability
Collect and visualize mesh metrics to better understand communication between services.

9 tutorials
Traffic Management
Traffic routing with Consul service mesh, gateways, and Envoy proxies.

Network Infrastructure Automation

On this page

Learn Consul fundamentals

Service Mesh

Network Infrastructure Auto...

All Tutorials

Additional Resources

- Upgrading Consul
- Consul Enterprise Licenses
- Consul Version Upgrade Guide
- Consul Agents
- Enabling Gossip Encryption
- Consul Snapshot Restore
- Secure Consul with ACLs
- Consul Metrics
- Grafana Dashboard for Consul

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered.

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a ticket for your issue at:

support.hashicorp.com

Discuss

Engage with the HashiCorp Cloud community including HashiCorp Architects and Engineers

discuss.hashicorp.com

Upcoming Webinars

Office Hours

An open forum with Consul Subject Matter Experts to answer questions that have arisen during the program and your deployment

Advanced Concepts

A detailed examination of Consul Federation, Namespaces & Admin Partitions, content also cover cluster operations and runbooks along with managing geographic failover and prepared queries

Action Items

- If not done, please share to customer.success@hashicorp.com
  - Authorized technical contacts for support
  - Stakeholders contact information (name and email addresses)
- Review planned/configured telemetry & monitoring strategy, platform, and dashboard(s)
- Prepare any questions for Office Hours next week

Q&A

Thank you

customer.success@hashicorp.com

www.hashicorp.com/customer-success