Consul Service discovery and KV

START HERE

`consul` **`agent -dev`**
Throwaway local cluster

`consul` **`members`**
List cluster members

`consul` **`catalog services`**
List registered services

`consul` **`services register`** *`<file>`*
Register a service

`consul` **`kv put`** *`<key>`* *`<value>`*
Write a key

`consul` **`kv get -recurse`** *`<prefix>`*
Read a subtree

`consul` **`snapshot save`** *`<file>`*
Back up server state

`consul` **`monitor`**
Stream agent logs

Service your app → `register` → **Client agent** on each node → `RPC` → **Server agents** Raft quorum → `query` → **Catalog** DNS and HTTP

CONCEPTS

Every command talks to a local agent over HTTP (default 127.0.0.1:8500). Point it elsewhere with -http-addr or CONSUL_HTTP_ADDR, and authenticate with CONSUL_HTTP_TOKEN.

Run 3 or 5 servers so Raft keeps a quorum. Losing the majority stops writes; -dev mode is one in-memory server and never for production.

AGENT

`consul` **`agent -dev`**
Run a single in-memory dev server
No persistence; UI available with -ui.

`consul` **`agent -server -bootstrap-expect=`***`<n>`* **`-data-dir=`***`<dir>`*
Start a server in a new cluster

`consul` **`agent -data-dir=`***`<dir>`* **`-retry-join=`***`<addr>`*
Start a client that joins a server

`consul` **`agent -config-dir=`***`<dir>`*
Load .json and .hcl config files

`consul` **`agent -node=`***`<name>`* **`-datacenter=`***`<dc>`*
Set node name and datacenter

AGENT (cont.)

`consul` **`agent -bind=`***`<ip>`* **`-client=`***`<ip>`*
Choose cluster and client addresses
-client defaults to loopback only.

`consul` **`agent -ui`**
Enable the web UI

`consul` **`validate`** *`<file-or-dir>`*
Check config before starting

CLUSTER

`consul` **`members`**
List members and their status

`consul` **`members -detailed`**
Include tags and versions

`consul` **`members -wan`**
Show the WAN pool (server members)

`consul` **`join`** *`<addr>`*
Join this agent to a cluster

`consul` **`leave`**
Leave gracefully and shut down
On a server this also removes it from the Raft peer set.

`consul` **`force-leave`** *`<node>`*
Mark a failed node as left

`consul` **`reload`**
Re-read config without a restart

`consul` **`monitor`**
Stream logs from the agent

CLUSTER (cont.)

`consul` **`monitor -log-level=debug`**
Raise the streamed log level

`consul` **`maint -enable -reason="`***`<why>`***`"`**
Put the node in maintenance mode

`consul` **`keygen`**
Generate a gossip encryption key

CATALOG AND SERVICES

`consul` **`catalog services`**
List services

`consul` **`catalog nodes`**
List nodes

`consul` **`catalog nodes -service=`***`<name>`*
Nodes that provide a service

`consul` **`catalog services -node=`***`<node>`*
Services on a node

`consul` **`services register`** *`<file>`*
Register services from HCL or JSON

`consul` **`services register -name=`***`<name>`* **`-port=`***`<port>`*
Register one service from flags

`consul` **`services register -tag=`***`<tag>`* **`-meta=`***`<k>=<v>`*
Add tags and metadata

CATALOG AND SERVICES (cont.)

`consul` **`services deregister -id=`***`<id>`*
Remove a service instance

`consul` **`services deregister`** *`<file>`*
Remove services defined in a file

KEY-VALUE STORE

`consul` **`kv put`** *`<key>`* *`<value>`*
Set or update a key

`consul` **`kv get`** *`<key>`*
Read a key

`consul` **`kv get -recurse`** *`<prefix>`***`/`**
Read all keys under a prefix

`consul` **`kv get -keys`** *`<prefix>`***`/`**
List key names only

`consul` **`kv get -detailed`** *`<key>`*
Show index, flags and session

`consul` **`kv delete`** *`<key>`*
Delete a key

`consul` **`kv delete -recurse`** *`<prefix>`***`/`**
Delete a whole subtree
Keep the trailing slash: foo also matches food.

KEY-VALUE STORE (cont.)

`consul` **`kv put`** *`<key>`* **`@`***`<file>`*
Take the value from a file

`consul` **`kv put -cas -modify-index=`***`<n>`* *`<key>`* *`<value>`*
Write only if unchanged
Fails if ModifyIndex no longer matches; not allowed with -recurse on delete.

`consul` **`kv export`** *`<prefix>`***`/`**
Dump a subtree as JSON

`consul` **`kv import @`***`<file>`*
Load a JSON export

SNAPSHOTS AND RAFT

`consul` **`snapshot save`** *`<file>`*
Save a point-in-time snapshot

`consul` **`snapshot restore`** *`<file>`*
Restore a snapshot to the servers

`consul` **`snapshot inspect`** *`<file>`*
Show snapshot metadata

`consul` **`operator raft list-peers`**
Show Raft servers and leader

`consul` **`operator raft remove-peer -id=`***`<id>`*
Remove a dead server

ACL

`consul` **`acl bootstrap`**
Create the first management token
Works once; save the SecretID.

`consul` **`acl policy create -name`** *`<name>`* **`-rules @`***`<file>`*
Create a policy from rules

`consul` **`acl policy list`**
List policies

`consul` **`acl token create -policy-name`** *`<name>`* **`-description`** *`<text>`*
Create a token for a policy

`consul` **`acl token list`**
List tokens

SERVICE MESH

`consul` **`intention create`** *`<src>`* *`<dst>`*
Allow src to reach dst
Deprecated since 1.9; prefer service-intentions config entries.

`consul` **`intention create -deny`** *`<src>`* *`<dst>`*
Deny src to dst

DNS

`dig @127.0.0.1 -p 8600` *`<name>`***`.service.consul`**
Resolve a service's addresses

`dig @127.0.0.1 -p 8600` *`<name>`***`.service.consul SRV`**
Get addresses and ports