# Consul Cheatsheet

A quick reference for Consul, covering key concepts, CLI commands, configuration, and best practices for service discovery, health checking, and configuration management.

## Core Concepts

### Service Discovery

**Service Discovery** enables applications to find and connect to each other dynamically. Consul maintains a catalog of available services and their locations.

**Key Features:**

- **Service Registration:** Services register themselves with Consul.
- **Health Checking:** Consul monitors the health of registered services.
- **DNS and HTTP Interface:** Applications can query Consul for service locations using DNS or HTTP.

**Benefits:**

- Improved application resilience.
- Simplified service configuration.
- Dynamic scaling and deployment.

### Health Checking

**Health Checks** ensure that only healthy services are used. Consul supports various health check types.

**Types of Health Checks:**

- **HTTP:** Checks if an HTTP endpoint returns a 2xx or 3xx status code.
- **TCP:** Checks if a TCP connection can be established.
- **Script:** Executes a script and checks its exit code.
- **TTL:** Requires services to periodically update their status.

**Health Check States:**

- **passing:** The service is healthy.
- **warning:** The service is experiencing issues but is still functional.
- **critical:** The service is unhealthy.

### Key/Value Store

**Key/Value (KV) Store** provides a hierarchical storage system for configuration data and other metadata. It is commonly used for centralized configuration management.

**Key Features:**

- Hierarchical key structure.
- Support for atomic operations.
- Change notification via blocking queries.

**Use Cases:**

- Storing application configuration.
- Feature toggles.
- Leader election.

## CLI Commands

### Basic Commands

| | |
|---|---|
| `consul members` | Lists the members of the Consul cluster. |
| `consul info` | Displays information about the Consul agent. |
| `consul catalog services` | Lists all registered services. |
| `consul catalog nodes` | Lists all registered nodes. |

### KV Store Commands

| | |
|---|---|
| `consul kv put <key> <value>` | Sets a key/value pair in the KV store.<br><br>**Example:**<br><br>`consul kv put myapp/config/version 1.0` |
| `consul kv get <key>` | Retrieves the value for a given key.<br><br>**Example:**<br><br>`consul kv get myapp/config/version` |
| `consul kv delete <key>` | Deletes a key from the KV store.<br><br>**Example:**<br><br>`consul kv delete myapp/config/version` |
| `consul kv export <prefix>` | Exports keys with specified prefix.<br><br>**Example:**<br><br>`consul kv export myapp/config/` |

### Agent Commands

| | |
|---|---|
| `consul agent -dev` | Starts a Consul agent in development mode (not for production). |
| `consul agent -server -bootstrap-expect=1 -data-dir=/tmp/consul -node=server-1 -client=0.0.0.0` | Starts a Consul server agent. (example) |
| `consul agent -data-dir=/tmp/consul -node=client-1 -client=0.0.0.0 -join=<server_ip>` | Starts a Consul client agent and joins an existing cluster. (example) |

## Configuration

### Agent Configuration

Consul agent configuration is typically done via JSON files. Key parameters include `data_dir`, `node_name`, `server`, `bootstrap_expect`, and `client_addr`.

Example configuration file (`agent.json`):

```
{
  "data_dir": "/opt/consul",
  "node_name": "consul-server-0",
  "server": true,
  "bootstrap_expect": 3,
  "client_addr": "0.0.0.0",
  "advertise_addr": "192.168.1.10",
  "ports": {
    "dns": 8600,
    "http": 8500,
    "https": 8501
  }
}
```

Start the agent with the configuration file:

```
consul agent -config-
file=/path/to/agent.json
```

### Service Definitions

Services are defined in JSON files and placed in the Consul configuration directory or registered via the HTTP API.

Example service definition (`web.json`):

```
{
  "id": "web-1",
  "name": "web",
  "tags": ["rails", "load-balanced"],
  "port": 80,
  "address": "192.168.1.20",
  "check": {
    "http":
"http://192.168.1.20:80/health",
    "interval": "10s",
    "timeout": "5s"
  }
}
```

Place the `web.json` file in the Consul configuration directory (e.g., `/etc/consul.d/`) or register it using the HTTP API.

### Health Check Definitions

Health checks can be defined alongside service definitions or separately.

Example health check definition (`check_web.json`):

```
{
  "id": "web-check",
  "name": "Web Health Check",
  "service_id": "web-1",
  "http":
"http://192.168.1.20:80/health",
  "interval": "10s",
  "timeout": "5s"
}
```

Place the `check_web.json` file in the Consul configuration directory or register it using the HTTP API.

## API Endpoints

### Service Catalog

| | |
|---|---|
| `GET /v1/catalog/services` | Lists all services in the catalog.<br>**Example:**<br>`curl http://localhost:8500/v1/catalog/services` |
| `GET /v1/catalog/service/<service>` | Lists instances of a specific service.<br>**Example:**<br>`curl http://localhost:8500/v1/catalog/service/web` |
| `GET /v1/health/service/<service>` | Lists healthy instances of a specific service.<br>**Example:**<br>`curl http://localhost:8500/v1/health/service/web` |

### Key/Value Store

| | |
|---|---|
| `GET /v1/kv/<key>` | Retrieves the value for a given key.<br>**Example:**<br>`curl http://localhost:8500/v1/kv/myapp/config/version` |
| `PUT /v1/kv/<key>` | Sets a key/value pair.<br>**Example:**<br>`curl -X PUT -d '1.0' http://localhost:8500/v1/kv/myapp/config/version` |
| `DELETE /v1/kv/<key>` | Deletes a key.<br>**Example:**<br>`curl -X DELETE http://localhost:8500/v1/kv/myapp/config/version` |

### Agent API

| | |
|---|---|
| `PUT /v1/agent/service/register` | Registers a service.<br>**Example:**<br>`curl -X PUT -d @service.json http://localhost:8500/v1/agent/service/register` |
| `PUT /v1/agent/service/deregister/<service_id>` | Deregisters a service.<br>**Example:**<br>`curl -X PUT http://localhost:8500/v1/agent/service/deregister/web-1` |
| `PUT /v1/agent/check/register` | Registers a check.<br>**Example:**<br>`curl -X PUT -d @check.json http://localhost:8500/v1/agent/check/register` |