Traefik & Consul Catalog

- Configuration Example
- Configuration Options
  - requireConsistent
  - defaultRule
  - constraints
  - namespaces
- Routing Configuration

Configuration Example

You can enable the Consul Catalog provider as detailed below:

```
providers:
  consulCatalog: {}
```

```
[providers.consulCatalog]
```

```
--providers.consulcatalog=true
```

Attaching tags to services:

```
- traefik.http.routers.my-router.rule=Host(`example.com`)
```

Configuration Options

Field	Description	Default	Required

`providers.providersThrottleDuration`	Minimum amount of time to wait for, after a configuration reload, before taking into account any new configuration refresh event. If multiple events occur within this time, only the most recent one is taken into account, and all others are discarded. **This option cannot be set per provider, but the throttling algorithm applies to each of them independently.**	2s	No
`providers.consulCatalog.refreshInterval`	Defines the polling interval.	15s	No
`providers.consulCatalog.prefix`	Defines the prefix for Consul Catalog tags defining Traefik labels.	traefik	No
`providers.consulCatalog.requireConsistent`	Forces the read to be fully consistent. See here for more information.	false	No
`providers.consulCatalog.exposedByDefault`	Expose Consul Catalog services by default through Traefik. If set to *false*, services that do not have a `traefik.enable=true` tag are ignored from the resulting routing configuration. See here for additional information.	true	no
`providers.consulCatalog.defaultRule`	The Default Host rule for all services. See here for more information.	`"Host(`{{ normalize .Name }}`)"`	No
`providers.consulCatalog.connectAware`	Enable Consul Connect support. If set to `true`, Traefik will be enabled to communicate with Connect services.	false	No

`providers.consulCatalog.connectByDefault`	Consider every service as Connect capable by default. If set to true, Traefik will consider every Consul Catalog service to be Connect capable by default. The option can be overridden on an instance basis with the traefik.consulcatalog.connect tag.	false	No
`providers.consulCatalog.serviceName`	Defines the name of the Traefik service in Consul Catalog.	"traefik"	No
`providers.consulCatalog.constraints`	Defines an expression that Traefik matches against the container labels to determine whether to create any route for that container. See here for more information.	""	No
`providers.consulCatalog.namespaces`	Defines the namespaces to query. See here for more information.	""	no
`providers.consulCatalog.stale`	Instruct Traefik to use stale consistency for catalog reads.	false	no
`providers.consulCatalog.cache`	Instruct Traefik to use local agent caching for catalog reads.	false	no
`providers.consulCatalog.endpoint`	Defines the Consul server endpoint.	-	No
`providers.consulCatalog.endpoint.address`	Defines the address of the Consul server.	127.0.0.1:8500	no
`providers.consulCatalog.endpoint.scheme`	Defines the URI scheme for the Consul server.	""	no
`providers.consulCatalog.endpoint.datacenter`	Defines the datacenter to use. If not provided in Traefik, Consul uses the default agent datacenter.	""	no
`providers.consulCatalog.endpoint.token`	Defines a per-request ACL token which overwrites the agent's default token.	""	no
`providers.consulCatalog.endpoint.endpointWaitTime`	Defines a duration for which a `watch` can block. If not provided, the agent default values will be used.	""	no

`providers.consulCatalog.endpoint.httpAuth`	Defines authentication settings for the HTTP client using HTTP Basic Authentication.	N/A	no
`providers.consulCatalog.endpoint.httpAuth.username`	Defines the username to use for HTTP Basic Authentication.	""	no
`providers.consulCatalog.endpoint.httpAuth.password`	Defines the password to use for HTTP Basic Authentication.	""	no
`providers.consulCatalog.strictChecks`	Define which Consul Service health checks are allowed to take on traffic.	["passing", "warning"]	no
`providers.consulCatalog.endpoint.tls.ca`	Defines the path to the certificate authority used for the secure connection to Consul Calatog, it defaults to the system bundle.	""	No
`providers.consulCatalog.endpoint.tls.cert`	Defines the path to the public certificate used for the secure connection to Consul Calatog. When using this option, setting the `key` option is required.	""	Yes
`providers.consulCatalog.endpoint.tls.key`	Defines the path to the private key used for the secure connection to Consul Catalog. When using this option, setting the `cert` option is required.	""	Yes
`providers.consulCatalog.endpoint.tls.insecureSkipVerify`	Instructs the provider to accept any certificate presented by Consul Catalog when establishing a TLS connection, regardless of the hostnames the certificate covers.	false	No
`providers.consulCatalog.watch`	When set to `true`, watches for Consul changes (Consul watches checks).	false	No

`requireConsistent`

Forces the read to be fully consistent. Setting this option can be expensive due to an extra round-trip but prevents ever performing a stale read.

For more information, see the Consul documentation on consistency.

`defaultRule`

The default host rule for all services.

For a given service, if no routing rule was defined by a tag, it is defined by this `defaultRule` instead. The `defaultRule` must be set to a valid Go template, and can include sprig template functions. The service name can be accessed with the `Name` identifier, and the template has access to all the labels (i.e. tags beginning with the `prefix`) defined on this service.

The option can be overridden on an instance basis with the `traefik.http.routers.{name-of-your-choice}.rule` tag.

```
providers:
  consulCatalog:
    defaultRule: "Host(`{{ .Name }}.{{ index .Labels \"customLabel\"}}`)"
    # ...
```

```
[providers.consulCatalog]
  defaultRule = "Host(`{{ .Name }}.{{ index .Labels \"customLabel\"}}`)"
  # ...
```

```
--providers.consulcatalog.defaultRule="Host(`{{ .Name }}.{{ index .Labels \"customLabel\"}}`)"
```

??? info "Default rule and Traefik service"

```
The exposure of the Traefik container, combined with the default rule mechanism,
can lead to create a router targeting itself in a loop.
In this case, to prevent an infinite loop,
Traefik adds an internal middleware to refuse the request if it comes from the same router.
```

`constraints`

The `constraints` option can be set to an expression that Traefik matches against the service tags to determine whether to create any route for that service. If none of the service tags match the expression, no route for that service is created. If the expression is empty, all detected services are included.

The expression syntax is based on the `Tag(`tag`)`, and `TagRegex(`tag`)` functions, as well as the usual boolean logic, as shown in examples below.

!!! tip "Constraints key limitations"

```
Note that `traefik.*` is a reserved label namespace for configuration and can not be used as a key for custom constraints.
```

??? example "Constraints Expression Examples"

````
```toml
# Includes only services having the tag `a.tag.name=foo`
constraints = "Tag(`a.tag.name=foo`)"
```

```toml
# Excludes services having any tag `a.tag.name=foo`
constraints = "!Tag(`a.tag.name=foo`)"
```

```toml
# With logical AND.
constraints = "Tag(`a.tag.name`) && Tag(`another.tag.name`)"
```

```toml
# With logical OR.
constraints = "Tag(`a.tag.name`) || Tag(`another.tag.name`)"
```

```toml
# With logical AND and OR, with precedence set by parentheses.
constraints = "Tag(`a.tag.name`) && (Tag(`another.tag.name`) || Tag(`yet.another.tag.name`))"
```

```toml
# Includes only services having a tag matching the `a\.tag\.t.+` regular expression.
constraints = "TagRegex(`a\.tag\.t.+`)"
```
````

```
providers:
  consulCatalog:
    constraints: "Tag(`a.tag.name`)"
```

```
    # ...
```

```
[providers.consulCatalog]
  constraints = "Tag(`a.tag.name`)"
  # ...
```

```
--providers.consulcatalog.constraints="Tag(`a.tag.name`)"
# ...
```

For additional information, refer to Restrict the Scope of Service Discovery.

namespaces

The `namespaces` option defines the namespaces in which the consul catalog services will be discovered. When using the `namespaces` option, the discovered configuration object names will be suffixed as shown below:

```
<resource-name>@consulcatalog-<namespace>
```

!!! warning

```
- The namespaces option only works with [Consul
Enterprise](https://www.consul.io/docs/enterprise),
  which provides the [Namespaces](https://www.consul.io/docs/enterprise/namespaces) feature.

- One should only define either the `namespaces` option or the `namespace` option.
```

```
providers:
  consulCatalog:
    namespaces:
      - "ns1"
      - "ns2"
    # ...
```

```
[providers.consulCatalog]
  namespaces = ["ns1", "ns2"]
  # ...
```

```
--providers.consulcatalog.namespaces=ns1,ns2
# ...
```

Routing Configuration

See the dedicated section in routing.

□□□ #2
□ □□□ □□□ 2026-10-31 14:00:40 UTC
□ □□□ □□□ 2026-10-31 14:00:41 UTC