Overview

- Provider Categories
- Provider Namespace
- Supported Providers
- Referencing a Traefik Dynamic Configuration Object from Another Provider
- Restrict the Scope of Service Discovery
- Providers Precedence
  - providers.precedence
    - Default precedence

Providers are infrastructure components, whether orchestrators, container engines, cloud providers, or key-value stores. The idea is that Traefik queries the provider APIs in order to find relevant information about routing, and when Traefik detects a change, it dynamically updates the routes.

Provider Categories

While each provider is different, you can think of each as belonging to one of four categories:

- Label-based: each deployed container has a set of labels attached to it
- Key-Value-based: each deployed container updates a key-value store with relevant information
- Annotation-based: a separate object, with annotations, defines the characteristics of the container
- File-based: uses files to define configuration

Provider Namespace

When you declare certain objects in the Traefik dynamic configuration, such as middleware, services, TLS options or server transports, they reside in their provider's namespace. For example, if you declare a middleware using a Docker label, it resides in the Docker provider namespace.

If you use multiple providers and wish to reference such an object declared in another provider (e.g. referencing a cross-provider object like middleware), then the object name should be suffixed by the `@` separator, and the provider name.

For the list of the providers names, see the supported providers table below.

```
<resource-name>@<provider-name>
```

!!! important "Kubernetes Namespace vs Traefik Namespace"

```
As Kubernetes also has its own notion of namespace,
one should not confuse the _provider namespace_ with the _Kubernetes Namespace_ of a resource
when in the context of cross-provider usage.

In this case, since the definition of a Traefik dynamic configuration object is not in Kubernetes,
specifying a Kubernetes Namespace when referring to the resource does not make any sense.

On the other hand, if you were to declare a middleware as a Custom Resource in Kubernetes and use the non-CRD Ingress objects,
you would have to add the Kubernetes Namespace of the middleware to the annotation like this `<middleware-namespace>-<middleware-name>@kubernetescrd`.

When [`safeNaming`](./kubernetes/kubernetes-crd.md#safenaming) is enabled, the value becomes `<middleware-namespace>_<middleware-name>@kubernetescrd`, using the same `_` separator as the other names generated by the provider.
```

Supported Providers

Below is the list of the currently supported providers in Traefik.

Provider	Type	Configuration Type	Provider Name
Docker	Orchestrator	Label	`docker`
Docker Swarm	Orchestrator	Label	`swarm`
Kubernetes IngressRoute	Orchestrator	Custom Resource	`kubernetescrd`
Kubernetes Ingress	Orchestrator	Ingress	`kubernetes`
Kubernetes Ingress NGINX	Orchestrator	Ingress-NGINX	`kubernetesIngressNGINX`

Provider	Type	Configuration Type	Provider Name
Kubernetes Gateway API	Orchestrator	Gateway API Resource	`kubernetesgateway`
Consul Catalog	Orchestrator	Label	`consulcatalog`
Nomad	Orchestrator	Label	`nomad`
ECS	Orchestrator	Label	`ecs`
File	Manual	YAML/TOML format	`file`
Consul	KV	KV	`consul`
Etcd	KV	KV	`etcd`
ZooKeeper	KV	KV	`zookeeper`
Redis	KV	KV	`redis`
HTTP	Manual	JSON/YAML format	`http`

!!! info "More Providers"

```
The current version of Traefik does not yet support every provider that Traefik v2.11 did.
See the [previous version (v2.11)](https://doc.traefik.io/traefik/v2.11/) for more
information.
```

Referencing a Traefik Dynamic Configuration Object from Another Provider

Declaring the add-foo-prefix in the file provider.

```
http:
  middlewares:
    add-foo-prefix:
      addPrefix:
        prefix: "/foo"
```

```
[http.middlewares]
  [http.middlewares.add-foo-prefix.addPrefix]
    prefix = "/foo"
```

Using the add-foo-prefix middleware from other providers:

```
your-container:
  image: your-docker-image

  labels:
    # Attach add-foo-prefix@file middleware (declared in file)
    - "traefik.http.routers.my-container.middlewares=add-foo-prefix@file"
```

```
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: ingressroutestripprefix

spec:
  entryPoints:
    - web
  routes:
    - match: Host(`example.com`)
      kind: Rule
      services:
        - name: whoami
          port: 80
      middlewares:
        - name: add-foo-prefix@file
        # namespace: bar
        # A namespace specification such as above is ignored
        # when the cross-provider syntax is used.
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: appspace
  annotations:
```

```
    "traefik.ingress.kubernetes.io/router.middlewares": add-foo-prefix@file
spec:
```

Restrict the Scope of Service Discovery

By default, Traefik creates routes for all detected containers.

If you want to limit the scope of the Traefik service discovery, i.e. disallow route creation for some containers, you can do so in two different ways:

1. With Consul Catalog, Docker, ECS, Nomad and Swarm providers, you can set `exposedByDefault` to `false` and add a label `traefik.enable=true` on containers you want to expose
2. Use a finer-grained mechanism based on label selector or constraints.

!!! info "The following providers support constraints"

```
- [Consul Catalog](./hashicorp/consul-catalog.md#constraints)
- [Docker](./docker.md#constraints)
- [ECS](./others/ecs.md#constraints)
- [Nomad](./hashicorp/nomad.md#constraints)
- [Swarm](./swarm.md#constraints)
```

!!! info "The following providers support label selectors"

```
- [Kubernetes CRD](./kubernetes/kubernetes-crd.md#opt-providers-kubernetesCRD-labelselector)
- [Kubernetes Gateway API](./kubernetes/kubernetes-gateway.md#opt-providers-kubernetesGateway-labelselector)
- [Kubernetes Ingress](./kubernetes/kubernetes-ingress.md#opt-providers-kubernetesIngress-labelselector)
```

Providers Precedence

`providers.precedence`

Optional

When two routers from **different providers** define the same rule with equal numeric priority, the `precedence` option determines which provider's route takes precedence.

The list is ordered from highest to lowest precedence: a provider listed first wins over providers listed later.

```yaml
providers:
  precedence:
    - kubernetescrd
    - kubernetes
    - file
```

```toml
[providers]
  precedence = ["kubernetescrd", "kubernetes", "file"]
```

```
--providers.precedence=kubernetescrd,kubernetes,file
```

Default precedence

When `precedence` is not set, Traefik uses the following default order (highest precedence first):

Position	Provider name
1	`kubernetesgateway`
2	`kubernetescrd`
3	`kubernetes`
4	`kubernetesingressnginx`
5	`swarm`
6	`docker`
7	`file`
8	`redis`
9	`knative`
10	`consul`
11	`consulcatalog`

Position	Provider name
12	`nomad`
13	`etcd`
14	`ecs`
15	`http`
16	`zookeeper`
17	`rest`

!!! note

```
- `precedence` only acts as a **tiebreaker**: it is applied only when two routes from different providers share the same numeric `priority` value. An explicit router priority always takes precedence.
- A provider absent from `precedence` loses to any listed provider.
- Provider names are case-insensitive.
```

{% include-markdown "includes/traefik-for-business-applications.md" %}

□□□ #2
□ □□□ □□□ 2026-10-31 14:01:31 UTC
□ □□□ □□□ 2026-10-31 14:01:32 UTC