kubeconform Manifest schema validator

START HERE

`kubeconform <file>`
Validate one manifest

`kubeconform <dir>`
Validate a folder

`kubeconform -summary <dir>`
Add totals

`kubeconform -strict <dir>`
Reject unknown fields

`kubeconform -kubernetes-version 1.30.0 <dir>`
Pick a cluster version

`kubeconform -ignore-missing-schemas <dir>`
Skip unknown kinds

`kubeconform -output pretty <dir>`
Readable results

`helm template <chart> | kubeconform -`
Check rendered output

Manifests files, dirs, stdin → parse → **Schema lookup** kind + apiVersion → validate → **Result** valid / invalid / skipped

CONCEPTS

kubeconform checks manifests offline against JSON schemas; it never talks to a cluster, so admission rules and server-side defaults are not tested.

A kind with no schema is an error by default; use -ignore-missing-schemas or -skip, or add a CRD schema location.

Exit code is 0 when all resources pass and 1 when any is invalid or errors.

RUNNING

`kubeconform <file>`
Validate one manifest

`kubeconform <dir> <file>`
Validate several paths at once

`kubeconform -`
Read manifests from stdin

`kubeconform -summary <dir>`
Print totals at the end

`kubeconform -verbose <dir>`
Show valid resources too

`kubeconform -exit-on-error <dir>`
Stop at the first failure

`kubeconform -ignore-filename-pattern '<regex>' <dir>`
Skip matching paths

`kubeconform -n 16 <dir>`
Use more parallel workers

`kubeconform -v`
Print the version

`kubeconform -h`
Show all flags

RUNNING (cont.)

`kubeconform -debug <dir>`
Print debug information

STRICTNESS AND VERSIONS

`kubeconform -strict <dir>`
Fail on unknown or duplicate keys

`kubeconform -kubernetes-version 1.30.0 <dir>`
Validate against that Kubernetes version

Default is master, which tracks the newest schemas. Pin it to your cluster's version.

`kubeconform -ignore-missing-schemas <dir>`
Skip kinds without a schema

`kubeconform -skip <kind>,<kind> <dir>`
Ignore kinds, all API versions

`kubeconform -skip v1/<Kind> <dir>`
Ignore one group/version of a kind

`kubeconform -reject <kind> <dir>`
Fail on listed kinds or GVKs

SCHEMAS AND CRDS

`kubeconform -schema-location default <dir>`
Use the built-in schema source

`kubeconform -schema-location '<url-or-path-template>' <dir>`
Add a custom schema location

SCHEMAS AND CRDS (cont.)

`kubeconform -cache <dir> <manifests>`
Cache downloaded schemas on disk

`kubeconform -insecure-skip-tls-verify <dir>`
Skip TLS checks on schema hosts

Template variables for -schema-location

Variable	Expands to
`{{.ResourceKind}}`	Kind of the resource
`{{.ResourceAPIVersion}}`	Version part of apiVersion
`{{.Group}}`	API group
`{{.NormalizedKubernetesVersion}}`	Version with a v prefix - strict when -strict is set
`{{.StrictSuffix}}`	Group and version suffix
`{{.KindSuffix}}`	

Repeat -schema-location to try several sources in order; keep 'default' first so built-in kinds still resolve.

OUTPUT

`kubeconform -output pretty <dir>`
Coloured, human-readable output

OUTPUT (cont.)

`kubeconform -output json <dir>`
Machine-readable result

`kubeconform -output junit <dir>`
JUnit XML for CI reports

`kubeconform -output tap <dir>`
TAP format

`kubeconform -output text <dir>`
Plain text, the default

-summary is ignored with junit output, and -verbose with tap and junit.

RECIPES

Validate a Helm chart

```
helm template <release> <chart> -f values.yaml \
  | kubeconform -strict -summary -kubernetes-version 1.30.0 -
```

Validate a kustomize overlay

```
kustomize build overlays/prod \
  | kubeconform -strict -ignore-missing-schemas -summary -
```

Include CRDs from the Datree catalog

```
kubeconform -strict -summary -schema-location default \
  -schema-location 'https://raw.githubusercontent.com/datreeio/CRDs-catalog/main/{{.Group}}/{{.ResourceKind}}_{{.ResourceAPIVersion}}.json' \
  manifests/
```

CI check with a schema cache

```
mkdir -p /tmp/schemas
kubeconform -cache /tmp/schemas -n 8 -output junit manifests/ > report.xml
```