Cilium CLI Kubernetes networking CLI

START HERE

`cilium install`
Install Cilium

`cilium hubble port-forward`
Reach Hubble Relay

`cilium status --wait`
Wait until ready

`hubble observe -P`
Watch live flows

`cilium connectivity test`
Verify networking

`cilium clustermesh status`
Check cluster mesh

`cilium hubble enable`
Turn on flow visibility

`cilium sysdump`
Collect a debug bundle

cilium CLI install and manage → `deploys` → **Agent + operator** per node, eBPF → `flows to` → **Hubble Relay** cluster-wide → `queried by` → **hubble CLI / UI** observe flows

CONCEPTS

The cilium CLI talks to the cluster in your kubeconfig and manages the install; hubble is a separate binary for reading flows.

Every command takes --context to target another cluster in your kubeconfig.

INSTALL

`cilium install`
Install Cilium with detected defaults

`cilium install --version <ver>`
Install a specific release

`cilium install --set <key>=<value>`
Pass a Helm value

`cilium install --dry-run`
Show resources without applying

`cilium install --dry-run-helm-values`
Show non-default Helm values

`cilium install --encryption=ipsec`
Enable IPsec encryption

`cilium upgrade --version <ver>`
Upgrade to a release

`cilium upgrade --reuse-values --set <key>=<value>`
Change one value on upgrade

`cilium upgrade --dry-run | kubectl diff -f -`
Preview an upgrade

`cilium uninstall`
Remove Cilium

STATUS AND CONFIG

`cilium status`
Show component health

`cilium status --wait`
Block until Cilium is ready

`cilium context`
Show the cluster being targeted

`cilium config view`
Print the agent configuration

`cilium sysdump`
Collect a debug archive

`cilium sysdump --node-list <nodes>`
Dump only chosen nodes

`cilium sysdump --logs-since-time <dur>`
Limit logs to a time window

`kubectl -n kube-system get cm cilium-config -o yaml`
Read the raw agent ConfigMap

CONNECTIVITY

`cilium connectivity test`
Run the end-to-end test suite

`cilium connectivity test --single-node`
Test on a one-node cluster

`cilium connectivity test --context <a> --multi-cluster <b>`
Test across a cluster mesh

`cilium connectivity perf`
Run network performance tests

The test deploys into the cilium-test namespace; give it a few minutes and a cluster with outbound access.

HUBBLE

`cilium hubble enable`
Deploy Hubble and Relay

`cilium hubble port-forward`
Forward Relay to localhost:4245

`hubble status`
Check Relay and flow counts

`hubble observe`
Stream recent flows

`hubble observe -P`
Auto port-forward Relay first

`hubble observe --pod <name>`
Flows for one pod

`hubble observe --verdict DROPPED`
Only dropped traffic

`hubble observe --protocol http`
Only L7 HTTP flows

`hubble observe --reply / --not-reply`
Pick reply or request flows

`hubble observe --encrypted`
Only encrypted flows

`cilium hubble ui`
Open the Hubble UI

`hubble help observe`
List every filter flag

CLUSTER MESH

`cilium install --set cluster.name=<name> --set cluster.id=<n>`
Give each cluster a unique name and ID

`cilium clustermesh enable --context <ctx>`
Deploy the mesh API server

CLUSTER MESH (cont.)

`cilium clustermesh status --wait`
Wait for the mesh to be ready

`cilium clustermesh connect --destination-context <ctx>`
Connect two clusters
One direction is enough; the link is created both ways.

`cilium clustermesh disconnect --destination-context <ctx>`
Remove a cluster link

`cilium clustermesh disable`
Remove the mesh API server

Cluster names and IDs must be unique, and pod CIDRs must not overlap.

RECIPES

Fresh cluster, verified

```
cilium install --version <ver>
cilium status --wait
cilium connectivity test
```

Find why traffic is dropped

```
cilium hubble enable
cilium hubble port-forward &
hubble observe --pod <name> --
verdict DROPPED
```

Join two clusters

```
cilium clustermesh enable --
context $C1
cilium clustermesh enable --
context $C2
cilium clustermesh connect --
context $C1 --destination-
context $C2
cilium clustermesh status --
context $C1 --wait
```

Safe upgrade

```
cilium upgrade --version <ver>
--dry-run | kubectl diff -f -
cilium upgrade --version <ver>
cilium status --wait
```