Linkerd Service mesh CLI

START HERE

`linkerd check --pre`
Can it install here?

`linkerd install --crds | kubectl apply -f -`
Install CRDs

`linkerd install | kubectl apply -f -`
Install control plane

`linkerd check`
Is it healthy?

`linkerd viz install | kubectl apply -f -`
Add metrics

`kubectl get deploy -n <ns> -o yaml | linkerd inject - | kubectl apply -f -`
Mesh a namespace

`linkerd viz stat deploy -n <ns>`
Success, RPS, latency

`linkerd viz dashboard`
Open the web UI

linkerd install CRDs, then control plane → `linkerd check` → **Control plane** identity, destination, policy → `inject` → **Meshed pods** proxy sidecar per pod → `viz install` → **Viz extension** stat, tap, top, dashboard

CONCEPTS

The CLI only prints YAML: pipe it to kubectl apply -f - to change the cluster.

Install order matters: CRDs first, then the control plane, then extensions such as viz.

Meshing is done at pod creation: restart workloads after annotating them.

INSTALL & UPGRADE

`linkerd check --pre`
Pre-flight cluster checks

`linkerd install --crds | kubectl apply -f -`
Install the CRDs

`linkerd install | kubectl apply -f -`
Install the control plane

`linkerd viz install | kubectl apply -f -`
Install the viz extension

`linkerd install --ha`
High-availability control plane

`linkerd install --set <key>=<val>`
Override a Helm chart value

`linkerd install -f <values.yaml>`
Install with a values file

INSTALL & UPGRADE (cont.)

`linkerd install --identity-trust-anchors-file <ca.crt>`
Supply your own trust anchor
Needs the issuer cert and key too (--identity-issuer-certificate-file, --identity-issuer-key-file).

`linkerd install --linkerd-cni-enabled`
Use the CNI plugin, not init containers

`linkerd upgrade | kubectl apply -f -`
Render an upgrade for the control plane

CHECK & VERSION

`linkerd check`
Verify the control plane

`linkerd check --proxy -n <ns>`
Verify data-plane proxies

`linkerd check --crds`
Only check the CRDs

`linkerd viz check`
Verify the viz extension

`linkerd version`
Client and server versions

`linkerd version --client`
Client version only

INJECT

`linkerd inject <file|dir|URL⟶>`
Annotate manifests for meshing

`kubectl get deploy <name> -o yaml | linkerd inject - | kubectl apply -f -`
Mesh a live deployment

`kubectl annotate ns <ns> linkerd.io/inject=enabled`
Auto-inject a whole namespace
Only pods created afterwards get a proxy: run kubectl rollout restart.

`linkerd uninject <file⟶>`
Remove the injection annotation

`linkerd inject --skip-outbound-ports <ports>`
Bypass the proxy for ports

DASHBOARD

`linkerd viz dashboard`
Open the web dashboard

`linkerd viz dashboard --port <n>`
Pick the local port

`linkerd viz dashboard --show url`
Print the URL only

METRICS

`linkerd viz stat deploy -n <ns>`
Success rate, RPS, latency

`linkerd viz stat <kind>/<name> -n <ns>`
Stats for one resource

`linkerd viz stat ns`
Stats per namespace

`linkerd viz stat deploy -A`
All namespaces

`linkerd viz stat deploy/<name> --from deploy/<src>`
Traffic from a source

`linkerd viz stat deploy --to deploy/<dst>`
Traffic to a destination

`linkerd viz stat deploy -t <window>`
Time window, 15s minimum

`linkerd viz edges deploy -n <ns>`
Show who talks to whom (mTLS)

`linkerd viz routes svc/<name> -n <ns>`
Per-route metrics
Needs a ServiceProfile for the service; see linkerd viz profile.

`linkerd viz authz deploy/<name> -n <ns>`
Traffic per authorization policy

LIVE TRAFFIC

`linkerd viz tap deploy/<name> -n <ns>`
Stream live requests

`linkerd viz tap deploy/<name> --path <prefix>`
Filter by path prefix

`linkerd viz tap deploy/<name> --method <verb>`
Filter by HTTP method

`linkerd viz top deploy/<name> -n <ns>`
Top-like live table of requests

DEBUG & IDENTITY

`linkerd identity -n <ns> <pod>`
Show a pod's mTLS certificate

`linkerd diagnostics proxy-metrics -n <ns> <kind>/<name>`
Raw proxy metrics

UNINSTALL

`linkerd viz uninstall | kubectl delete -f -`
Remove the viz extension

`linkerd uninstall | kubectl delete -f -`
Remove the control plane
Uninstall extensions first; add --force if injected pods remain.

GLOBAL FLAGS

`-n, --namespace <ns>`
Namespace of the resource

`--context <name>`
Kubeconfig context

`--kubeconfig <path>`
Kubeconfig file

RECIPES

Fresh install with viz

```
linkerd check --pre
linkerd install --crds |
kubectl apply -f -
linkerd install | kubectl apply
-f -
linkerd check
linkerd viz install | kubectl
apply -f -
linkerd viz check
```

Mesh a namespace and verify

```
kubectl annotate ns <ns>
linkerd.io/inject=enabled
kubectl rollout restart deploy
-n <ns>
linkerd check --proxy -n <ns>
linkerd viz stat deploy -n <ns>
```

Find a failing dependency

```
linkerd viz stat deploy -n <ns>
linkerd viz edges deploy -n
<ns>
linkerd viz tap deploy/<name> -
n <ns> --to deploy/<dst>
```