istioctl Service mesh CLI

START HERE

`istioctl install --set profile=demo`
Install Istio

`istioctl version`
Client and control plane

`istioctl analyze -A`
Lint config everywhere

`istioctl proxy-status`
Is every proxy synced?

`istioctl proxy-config listeners <pod>`
Inspect a sidecar

`istioctl x describe pod <pod>`
Why does this pod behave so?

`istioctl dashboard kiali`
Open the mesh UI

`istioctl uninstall --purge`
Remove Istio

istiod control plane — `xDS push` → — **Envoy / ztunnel** data plane — `proxy-status` → — **proxy-config** what it received

CONCEPTS

Run analyze after every config change: it catches missing gateways, bad hosts and unlabelled namespaces before traffic breaks.

A pod is <name>.<namespace>; use deployment/<name> in place of a pod name where the command accepts it.

proxy-config prints the xDS config a proxy holds; proxy-status tells you whether it is current.

INSTALL

`istioctl install --set profile=<name>`
Install a profile

`istioctl install -f <iop.yaml>`
Install from an IstioOperator file

`istioctl install --set <path>=<value>`
Override a single value

`istioctl x precheck`
Check the cluster before install

`istioctl manifest generate -f <iop.yaml>`
Render manifests to stdout

INSTALL (cont.)

`istioctl install -r <rev>`
Install a revision alongside others

`istioctl install --dry-run`
Preview without applying

`istioctl uninstall --revision <rev>`
Remove one revision

`istioctl uninstall --purge`
Remove every Istio resource

STATUS & VERSION

`istioctl version`
Show client and control plane versions

`istioctl version --remote=false`
Client version only

`istioctl proxy-status`
Sync state of every proxy

`istioctl ps <pod>.<ns>`
Sync state of one proxy

ANALYZE & VALIDATE

`istioctl analyze`
Analyze the current namespace

`istioctl analyze -A`
Analyze all namespaces

ANALYZE & VALIDATE (cont.)

`istioctl analyze <file.yaml>`
Check a file before applying

`istioctl analyze -n <ns>`
Analyze one namespace

`istioctl analyze --use-kube=false <file.yaml>`
Files only, ignore the cluster

`istioctl validate -f <file.yaml>`
Validate resource syntax

`istioctl bug-report`
Collect logs and state into an archive

PROXY CONFIG

`istioctl proxy-config all <pod>`
Dump all Envoy config

`istioctl pc cluster <pod>`
Upstream clusters of a proxy

`istioctl pc listener <pod>`
Listeners of a proxy

`istioctl pc route <pod>`
Route tables of a proxy

`istioctl pc endpoint <pod>`
Endpoints the proxy knows

`istioctl pc secret <pod>`
Certificates loaded in the proxy

PROXY CONFIG (cont.)

`istioctl pc listener <pod> --port <n>`
Filter by port

`istioctl pc cluster <pod> --fqdn <host>`
Filter by service host

`istioctl pc endpoint <pod> --cluster <name>`
Endpoints of one cluster

`istioctl pc route <pod> --name <n> -o json`
Full JSON of one route

`istioctl pc cluster deployment/<name>`
Use a deployment instead of a pod

`istioctl pc log <pod> --level <lvl>`
Change Envoy log level

INJECT & DESCRIBE

`istioctl x describe pod <pod>`
Explain routing and policy for a pod

`istioctl x describe svc <svc>`
Explain config for a service

`istioctl x check-inject <pod>`
Why was sidecar injected or not?

INJECT & DESCRIBE (cont.)

`istioctl kube-inject -f <file.yaml>`
Inject the sidecar into a manifest

`istioctl kube-inject -f <file.yaml> | kubectl apply -f -`
Inject and apply

DASHBOARDS

`istioctl dashboard kiali`
Open Kiali

`istioctl dashboard grafana`
Open Grafana

`istioctl dashboard prometheus`
Open Prometheus

`istioctl dashboard jaeger`
Open Jaeger

`istioctl dashboard proxy <pod>.<ns>`
Envoy admin UI of a pod

REVISIONS & TAGS

`istioctl tag list`
List revision tags

`istioctl tag set <tag> --revision <rev>`
Point a tag at a revision

REVISIONS & TAGS (cont.)

`istioctl tag set <tag> -r <rev> --overwrite`
Move a tag to a new revision

AMBIENT

`istioctl waypoint apply -n <ns>`
Create a waypoint proxy

`istioctl waypoint apply --enroll-namespace`
Create and use for the namespace

`istioctl waypoint list -A`
List waypoints

`istioctl waypoint status`
Is the waypoint ready?

`istioctl ztunnel-config workload`
Workloads known to ztunnel

`istioctl ztunnel-config service`
Services known to ztunnel

GLOBAL FLAGS

`-n <ns>`
Target namespace

`--context <name>`
Kubeconfig context to use

`-i <ns>`
Istio system namespace