Flux CLI GitOps for Kubernetes

CONCEPTS

> Flux pulls: a Source (Git, OCI, Helm repo, bucket) is fetched, then a Kustomization or HelmRelease applies it. The CLI mostly inspects and nudges; Git stays the source of truth.

Source GitRepository, OCI, HelmRepository
↓ artifact
Kustomization / HelmRelease what to apply, how often
↓ apply
Cluster live objects, drift corrected

> Commands default to the flux-system namespace. Add -n <ns> or -A, or --ns-follows-kube-context to use the kubeconfig context's namespace.

INSTALL & BOOTSTRAP

`flux check --pre`
Check cluster meets prerequisites

`flux bootstrap github --owner=<org> --repository=<repo> --path=<dir>`
Install Flux and sync from GitHub

INSPECT

`flux get all -A`
Everything in all namespaces

`flux get kustomizations`
List Kustomizations and readiness

`flux get helmreleases -A`
List Helm releases

`flux get all -A --status-selector ready=false`
Only the broken ones

RECONCILE & SUSPEND

`flux reconcile kustomization <name>`
Apply now, skip the interval

RECONCILE & SUSPEND (cont.)

`flux reconcile kustomization <name> --with-source`
Refetch source, then apply

`flux suspend kustomization <name>`
Pause one Kustomization

`flux resume kustomization <name>`
Resume and reconcile

CREATE RESOURCES

`flux create source git <name> --url=<url> --branch=<b>`
Define a Git source

`flux create kustomization <name> --source=<src> --path=<dir> --prune`
Apply a path from a source

`flux create helmrelease <name> --source=HelmRepository/<repo> --chart=<chart>`
Install a chart

`flux create source git <name> ... --export > <file>.yaml`
Write YAML instead of applying

DEBUG

`flux logs -A`
Controller logs, all namespaces

`flux events --for Kustomization/<name>`
Events for one object

`flux trace <kind> <name> -n <ns>`
Which source manages an object

BUILD & DIFF

`flux diff kustomization <name> --path <dir>`
Local changes vs the cluster

GLOBAL FLAGS

`-n <ns>` / `-A`
Namespace or all namespaces