# Flux CLI GitOps for Kubernetes

**START HERE**

`flux check --pre`
Check cluster prerequisites

`flux bootstrap github --owner=<org> --repository=<repo> --path=<dir>`
Install Flux, wire to Git

`flux get all -A`
Status of everything

`flux reconcile kustomization <name> --with-source`
Sync now

`flux diff kustomization <name> --path <dir>`
Preview local changes

`flux logs -A --level=error`
Controller errors

`flux events --for Kustomization/<name>`
Why it is not ready

`flux suspend kustomization <name>`
Pause reconciling

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

## 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.

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

Anything that changes the cluster directly is overwritten by the next sync of the Kustomization that owns it. Change Git instead, and use create --export to generate the YAML.

## INSTALL & BOOTSTRAP

`flux check --pre`
Check cluster meets prerequisites

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

*Needs GITHUB_TOKEN exported. Re-running is safe and upgrades the controllers.*

`flux check`
Verify controllers are healthy

Bootstrap commits the controller manifests and a sync config under --path, so Flux then manages its own upgrades from Git.

## INSPECT

`flux get all`
Everything in one namespace

`flux get all -A`
Everything in all namespaces

## INSPECT (cont.)

`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

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

`flux reconcile helmrelease <name>`
Re-run a Helm release

`flux suspend kustomization <name>`
Pause one Kustomization

*Suspended objects keep their resources but stop being corrected.*

`flux resume kustomization <name>`
Resume and reconcile

`flux suspend helmrelease <name>`
Pause a Helm release

`flux resume helmrelease <name>`
Resume a Helm release

## 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

## CREATE RESOURCES (cont.)

`flux create helmrelease <name> ... --values=<file>`
Pass local values files

`flux create helmrelease <name> ... --create-target-namespace`
Install into a new namespace

`flux create secret git <name> --url=ssh://git@<host>/<repo>`
Generate an SSH deploy key secret

*Prints the public key; add it to the repository as a deploy key.*

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

*--export works on every create command and is the way to put the result into Git.*

`flux export kustomization --all`
Dump all Kustomizations as YAML

## DEBUG

`flux logs -A`
Controller logs, all namespaces

`flux logs -A --level=error`
Only errors

`flux logs --kind=Kustomization --name=<name>`
Logs for one object

`flux events`
Events for Flux objects

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

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

`flux debug kustomization <name> --show-vars`
Post-build substitution variables

*Output can contain secret values.*

## BUILD & DIFF

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

*Exit code 1 means differences, above 1 means error; usable in CI.*

`flux diff kustomization <name> --path <dir> --recursive`
Diff nested Kustomizations

## GLOBAL FLAGS

`--context <name>` / `--kubeconfig <file>`
Choose cluster or kubeconfig

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

`--timeout <dur>`
Operation timeout (default 5m)

`--export`
Print YAML instead of applying

## RECIPES

**Bootstrap a cluster**

```
export GITHUB_TOKEN=<token>
flux check --pre
flux bootstrap github --
owner=<org> --repository=<repo>
\
  --path=clusters/prod --
personal=false
flux get all -A
```

**Deploy an app from Git**

```
flux create source git podinfo
--url=https://github.com/<org>/
podinfo \
  --branch=main --export >
clusters/prod/podinfo-
source.yaml
flux create kustomization
podinfo --source=GitRepository/
podinfo \
  --path=./kustomize --
prune=true --interval=5m \
  --export > clusters/prod/
podinfo-ks.yaml
git add . && git commit -m 'Add
podinfo' && git push
```

**Why is it not ready**

```
flux get all -A --status-
selector ready=false
flux events --for
Kustomization/<name>
flux logs --kind=Kustomization
--name=<name> --level=error
flux reconcile kustomization
<name> --with-source
```