Argo CD GitOps for Kubernetes

START HERE

`argocd login <server>`
Sign in to Argo CD

`argocd app list`
List applications

`argocd app get <app>`
Status and resources

`argocd app diff <app>`
Live vs Git

`argocd app sync <app>`
Deploy from Git

`argocd app wait <app> --health`
Block until healthy

`argocd app history <app>`
Past deployments

`argocd app rollback <app> <id>`
Back to a version

Git repo desired state → `render` → **Application** repo + path + target → `sync` → **Cluster** live state

CONCEPTS

Argo CD compares Git (desired) with the cluster (live). Out of Sync means they differ; Healthy is separate and describes the running resources.

With automated sync, a merge to Git deploys by itself; manual sync and rollback are for apps without it.

LOGIN & CONTEXT

`argocd login <server>`
Log in with username and password
Add --sso for single sign-on, --grpc-web behind proxies without HTTP/2.

`argocd login <server> --sso`
Log in through your SSO provider

`argocd login --core`
Talk straight to the cluster API
Uses your kubeconfig; no Argo CD API server needed.

`argocd context`
List saved server contexts

`argocd context <name>`
Switch to another server

`argocd logout <server>`
Remove saved login

APPLICATIONS

`argocd app list`
List applications
-o name prints names only; -l <label> filters.

`argocd app get <app>`
Show status, sources and resources

`argocd app get <app> --refresh`
Re-read Git before showing status
--hard-refresh also drops the manifest cache.

`argocd app manifests <app>`
Print the rendered manifests

`argocd app logs <app>`
Show logs from the app's pods

`argocd app history <app>`
Show deployment history and IDs

`argocd app delete <app>`
Delete an application
Cascades to its resources by default; --cascade=false keeps them.

CREATE & CONFIGURE

`argocd app create <app> --repo <url> --path <dir> --dest-server https://kubernetes.default.svc --dest-namespace <ns>`
Create an app from a Git path

CREATE & CONFIGURE (cont.)

`argocd app create -f <file.yaml>`
Create from an Application manifest
Needed for multi-source apps.

`argocd app create <app> ... --project <proj>`
Place the app in a project

`argocd app create <app> ... --sync-policy automated`
Deploy automatically on Git change

`argocd app create <app> ... --auto-prune --self-heal`
Prune deletions, revert manual drift

`argocd app create <app> ... --helm-set <key>=<val>`
Set a Helm value

`argocd app set <app> --revision <rev>`
Point at another branch, tag or commit

`argocd app set <app> --helm-set <key>=<val>`
Change a Helm value

`argocd app set <app> --sync-policy automated`
Turn on automated sync

CREATE & CONFIGURE (cont.)

`argocd app set <app> --sync-policy none`
Turn off automated sync

SYNC & DIFF

`argocd app diff <app>`
Show live vs target differences
Exits 1 when there is a diff; useful in scripts.

`argocd app sync <app>`
Sync to the target state

`argocd app sync <app> --dry-run`
Preview without changing the cluster

`argocd app sync <app> --prune`
Also delete removed resources

`argocd app sync <app> --preview-changes`
Show the diff, then confirm

`argocd app sync <app> --resource <group>:<kind>:<name>`
Sync only one resource
Repeatable; a leading ! excludes matches.

`argocd app sync -l <label>=<val>`
Sync apps matching a label

SYNC & DIFF (cont.)

`argocd app sync <app> --async`
Return without waiting

WAIT & ROLLBACK

`argocd app wait <app> --health`
Wait until the app is healthy

`argocd app wait <app> --sync --health`
Wait for synced and healthy

`argocd app wait <app> --timeout <sec>`
Give up after a time

`argocd app rollback <app> <id>`
Roll back to a history ID
IDs come from argocd app history. Without an ID it goes back one version.

REPOSITORIES

`argocd repo list`
List registered repositories

`argocd repo add <url> --username <user> --password <pw>`
Add a private HTTPS Git repo

`argocd repo add <git@host:org/repo> --ssh-private-key-path <key>`
Add a Git repo over SSH

REPOSITORIES (cont.)

`argocd repo add <url> --type helm --name <name>`
Add a Helm chart repository
--name is required for Helm repos.

CLUSTERS

`argocd cluster list`
List registered clusters

`argocd cluster add <kube-context>`
Register a cluster from kubeconfig
Creates an argocd-manager service account in the target cluster (kube-system by default).

`argocd cluster add <kube-context> --name <name>`
Register under a custom name

PROJECTS

`argocd proj list`
List projects

`argocd proj get <proj>`
Show a project's rules

`argocd proj create <proj>`
Create a project

HELP

`argocd <command> --help`
Usage and flags for a command