OpenShift CLI (oc) OpenShift cluster CLI

START HERE

`oc login <url>`
Log in to a cluster

`oc new-app <git-url>`
Build and deploy

`oc whoami`
Current user

`oc expose svc/<name>`
Public route

`oc projects`
List projects

`oc status`
Project overview

`oc project <name>`
Switch project

`oc rsh <pod>`
Shell in a pod

CONCEPTS

> oc contains kubectl: get, describe, apply, logs, exec, scale and every other kubectl verb work unchanged as oc <verb>. This sheet lists only what OpenShift adds.

Git source new-app / new-build
↓ start-build
Build BuildConfig
↓ push
ImageStream tag / import-image
↓ trigger
Deployment rollout
↓ expose
Route public hostname
Source-to-route pipeline

> A project is a namespace with extra metadata and access control. -n <project> works on every command.

LOGIN & SESSION

`oc login <url>`
Log in, prompting for credentials

`oc login --token=<token> --server=<url>`
Log in with a token
Copy it from the web console: user menu, Copy login command.

`oc login <url> --web`
Log in through a browser

`oc login -u <user> -p <password> <url>`
Non-interactive password login

`oc whoami`
Show logged-in user

`oc whoami --show-console`
Print web console URL

`oc whoami -t`
Print current token

`oc whoami --show-server`
Print API server URL

`oc logout`
End the session

PROJECTS

`oc projects`
List projects you can access

`oc project`
Show current project

`oc project <name>`
Switch project

`oc new-project <name>`
Create and switch to project

`oc delete project <name>`
Delete project and all in it

`oc status`
Overview of the current project

PROJECTS (cont.)

`oc status --suggest`
Overview with issue hints

APPS & BUILDS

`oc new-app <git-url>`
Detect, build and deploy source

`oc new-app --image=<img> --name=<name>`
Deploy an existing image

`oc new-app <builder>~<git-url>`
Build source with a builder image
Example: nodejs~https://github.com/<org>/<repo>

`oc new-app <img> -e <KEY>=<value>`
Set env vars on create

`oc new-app --template=<tpl> -p <KEY>=<value>`
Instantiate a template

`oc new-app --search <term>`
Search images and templates

`oc new-build <git-url>`
BuildConfig only, no deploy

`oc new-build --strategy=docker --binary --name=<name>`
Binary Dockerfile build

`oc start-build <bc>`
Start a build

`oc start-build <bc> --follow`
Start and stream build log

`oc start-build <bc> --from-dir=<dir> --follow`
Build from local files
Uploads the directory instead of cloning the repo.

`oc start-build <bc> --wait`
Wait; non-zero exit on failure

`oc logs -f bc/<bc>`
Follow latest build log

`oc get builds`
List builds

`oc process -f <template> -p <KEY>=<value> | oc apply -f -`
Render a template and apply

ROUTES & NETWORK

`oc expose svc/<name>`
Create a route for a service

`oc expose svc/<name> --hostname=<host>`
Route with a custom host

`oc get routes`
List routes and hosts

ROUTES & NETWORK (cont.)

`oc create route edge --service=<svc>`
TLS route, edge termination
Also passthrough and reencrypt.

`oc port-forward pod/<pod> <local>:<remote>`
Tunnel a local port

DEPLOY & ROLLOUT

`oc rollout status deploy/<name>`
Watch a rollout

`oc rollout restart deploy/<name>`
Restart all pods

`oc rollout undo deploy/<name>`
Roll back one revision

`oc rollout history deploy/<name>`
List revisions

`oc set env deploy/<name> <KEY>=<value>`
Set env var (triggers rollout)

`oc set env deploy/<name> --list`
Show env vars

`oc set env deploy/<name> --from=secret/<secret>`
Env from a secret

`oc set volume deploy/<name> --add -t pvc --claim-name=<pvc> -m <path>`
Mount a PVC

`oc set triggers deploy/<name> --from-image=<is>:<tag> -c <container>`
Redeploy on image change

IMAGES

`oc get is`
List image streams

`oc import-image <is>:<tag> --from=<img> --confirm`
Import an external image

`oc import-image <is>`
Re-check source for updates

`oc tag <is>:<tag> <is>:<newtag>`
Point a tag at an image
Promote dev to prod: workloads with an image trigger on the new tag redeploy.

`oc registry login`
Log in to the integrated registry

`oc secrets link <sa> <secret> --for=pull`
Use a pull secret for a SA

DEBUG & FILES

`oc rsh <pod>`
Shell in first container

`oc rsh -c <container> <pod>`
Shell in a named container

`oc rsh <pod> <cmd>`
Run one command

`oc rsync <dir>/ <pod>:<path>`
Copy local dir to pod

`oc rsync <pod>:<path> <dir>`
Copy from pod

`oc debug deploy/<name>`
Copy pod and open a shell
Probes are disabled and the command replaced, so crashing pods stay up.

`oc debug node/<node>`
Root shell on a node
Run chroot /host to use the node's own tools.

`oc extract secret/<name> --to=-`
Print secret values decoded

ADMIN (OC ADM)

`oc adm policy add-role-to-user <role> <user>`
Grant role in project

`oc adm policy add-role-to-user <role> -z <sa>`
Grant role to service account

`oc adm policy add-cluster-role-to-user <role> <user>`
Grant cluster-wide role

`oc adm policy add-scc-to-user <scc> -z <sa>`
Allow SA an SCC
Example SCCs: restricted-v2, anyuid, privileged.

`oc adm must-gather`
Collect cluster data for support
Writes ./must-gather.local.<random>.

`oc get co`
Cluster operator health

`oc get clusterversion`
Cluster version and update state

`oc adm top node`
Node CPU and memory

`oc adm drain <node> --ignore-daemonsets`
Evict pods for maintenance

RECIPES

Deploy from Git

```
oc new-project demo
oc new-app https://github.com/<org>/<repo> --name=web
oc logs -f bc/web
oc expose svc/web
oc get route web
```