Skaffold Kubernetes dev loop

START HERE

`skaffold` **`init`**
Generate skaffold.yaml

`skaffold` **`dev -p`** *`<profile>`*
Use a profile

`skaffold` **`dev`**
Build, deploy, watch

`skaffold` **`diagnose`**
Show effective config

`skaffold` **`run`**
One-shot build and deploy

`skaffold` **`debug`**
Dev loop with debugger

`skaffold` **`render`**
Print final manifests

`skaffold` **`delete`**
Remove what was deployed

CONCEPTS

skaffold dev is the inner loop: it builds, deploys and tails logs, rebuilds on file change, and deletes what it deployed when you press Ctrl+C. skaffold run deploys once and leaves it.

- **build** images from artifacts
- **test** optional
- **render** manifests with new tags
- **deploy** kubectl / helm / kustomize
- **verify** optional checks

Skaffold only drives your existing tools: Docker or buildpacks to build, kubectl, Helm or Kustomize to deploy. Its own config is skaffold.yaml.

Image names in manifests are rewritten to the freshly built tag at deploy time; you never edit tags by hand.

DEV LOOP

`skaffold` **`dev`**
Build, deploy, watch, tail logs
Ctrl+C stops it and cleans up the deployment unless --cleanup=false.

`skaffold` **`dev --port-forward=services`**
Forward service ports locally
The dev default is user: only ports listed under portForward in skaffold.yaml.

`skaffold` **`dev --no-prune --cleanup=false`**
Keep resources after exit

DEV LOOP (cont.)

`skaffold` **`dev --auto-build=false`**
Rebuild only on request

`skaffold` **`dev --trigger=polling`**
Poll for changes
Values: notify (default), polling, manual. Use polling on network or container file systems.

`skaffold` **`dev -w`** *`<image>`*
Watch only matching images

`skaffold` **`dev --skip-tests`**
Skip tests after build

`skaffold` **`dev --keep-running-on-failure`**
Suspend instead of exit on error

`skaffold` **`dev --status-check=false`**
Do not wait for rollout

`skaffold` **`dev --tail=false`**
Do not stream logs

RUN, BUILD & DEPLOY

`skaffold` **`run`**
Build and deploy once

`skaffold` **`run --tail`**
Deploy and stream logs

`skaffold` **`build`**
Build images only

`skaffold` **`build -q`**
Print only built image names

`skaffold` **`build --push`**
Push built images

RUN, BUILD & DEPLOY (cont.)

`skaffold` **`build --file-output`** *`<file>`*
Save built tags as JSON

`skaffold` **`build --dry-run`**
Compute tags without building

`skaffold` **`deploy -a`** *`<file>`*
Deploy pre-built images
Pair with build --file-output for a build once, deploy many pipeline.

`skaffold` **`deploy -i`** *`<name>=<tag>`*
Deploy given image tag

`skaffold` **`run -b`** *`<substr>`*
Build only matching artifacts

`skaffold` **`run --force`**
Recreate resources if needed
May cause downtime.

`skaffold` **`test`**
Run tests on built images

`skaffold` **`apply`** *`<dir>`*
Apply hydrated manifests

DEBUG & VERIFY

`skaffold` **`debug`**
Dev loop with debug ports
Auto-build, auto-sync and auto-deploy are off by default here.

`skaffold` **`debug --port-forward=user,debug`**
Forward debug ports

`skaffold` **`verify`**
Run verify tests

`skaffold` **`verify -a`** *`<file>`*
Verify pre-built images

CONFIG & PROFILES

`skaffold` **`init`**
Generate skaffold.yaml
Detects Dockerfiles and manifests and asks which to use.

`skaffold` **`init --generate-manifests`**
Also create basic manifests

`skaffold` **`init -k`** *`<glob>`*
Pick Kubernetes manifests

`skaffold` **`init -a`** *`<Dockerfile>=<image>`*
Map Dockerfile to image

`skaffold` **`fix`**
Print config in newest schema

`skaffold` **`fix --overwrite`**
Upgrade skaffold.yaml in place

`skaffold` **`diagnose`**
Inspect effective config

`skaffold` **`diagnose --yaml-only`**
Print merged skaffold.yaml

`skaffold` **`dev -f`** *`<file>`*
Use another config file

`skaffold` **`dev -p`** *`<profile>`*
Activate a profile

`skaffold` **`dev -p`** *`<a>,-<b>`*
Enable a, disable b

`skaffold` **`dev -m`** *`<module>`*
Select config module

RENDER & DELETE

`skaffold` **`render`**
Print rendered manifests

`skaffold` **`render -o`** *`<file>`*
Write manifests to a file

RENDER & DELETE (cont.)

`skaffold` **`render --offline`**
Render without a cluster

`skaffold` **`delete`**
Remove deployed resources

`skaffold` **`delete --dry-run`**
List what would be deleted

TARGETING & IMAGES

`skaffold` **`dev -n`** *`<ns>`*
Deploy to a namespace

`skaffold` **`dev --kube-context`** *`<ctx>`*
Use a kube context

`skaffold` **`dev -d`** *`<repo>`*
Default image repository
Prefixes image names, e.g. gcr.io/<project>; use for remote clusters.

`skaffold` **`dev -t`** *`<tag>`*
Custom image tag

`skaffold` **`dev --platform`** *`<os/arch>`*
Target platform

`skaffold` **`dev --cache-artifacts=false`**
Disable build cache

HOUSEKEEPING

`skaffold` **`version`**
Print version

`skaffold` **`completion`** *`<shell>`*
Shell completion script
bash, fish or zsh.

HOUSEKEEPING (cont.)

`skaffold` **`dev -v`** *`<level>`*
Set log level
Levels: panic, fatal, error, warning, info, debug, trace.

SKAFFOLD.YAML

Minimal skaffold.yaml

```
apiVersion: skaffold/v4beta14
kind: Config
build:
  artifacts:
  - image: web
    docker: {dockerfile:
Dockerfile}
manifests:
  rawYaml: [k8s/*.yaml]
deploy:
  kubectl: {}
portForward:
- {resourceType: service,
resourceName: web, port: 80,
localPort: 8080}
```

RECIPES

Inner loop on local cluster

```
kubectl config use-context
kind-dev
skaffold dev --port-
forward=services -n dev
```

Build once, deploy per environment

```
skaffold build -d
registry.example/app --push --
file-output build.json
skaffold deploy -a build.json -
p staging -n staging
skaffold deploy -a build.json -
p prod -n prod
```

Review manifests in CI

```
skaffold render -a build.json -
p prod --offline -o
rendered.yaml
kubectl diff -f rendered.yaml
```