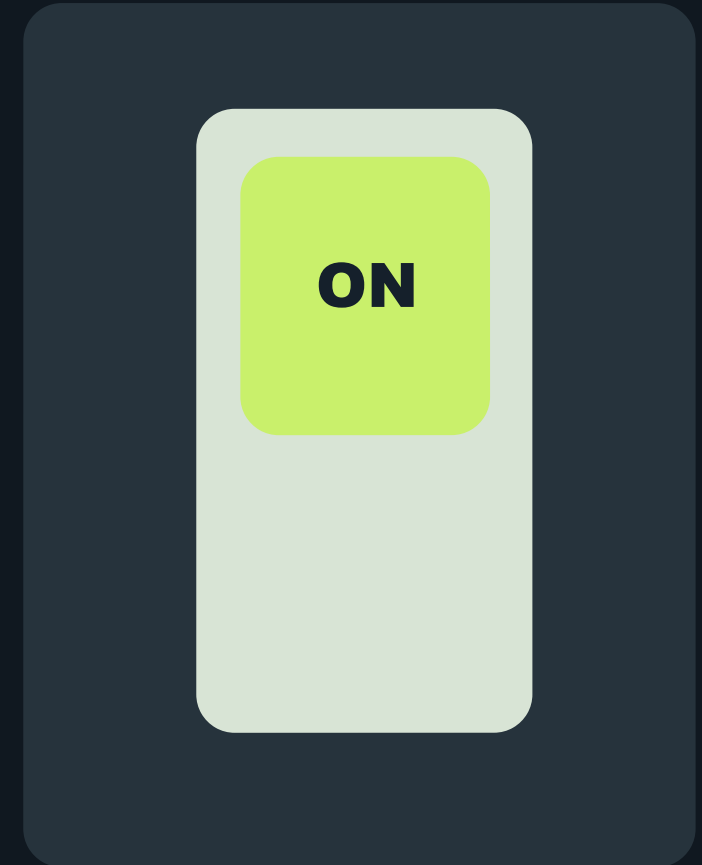


Beyond On and Off

Using Feature Flags for Safer Cloud Deployments

A practical guide to progressive delivery



A feature flag is a light switch for code

Ship the code now. Choose when people can use it.

CODE DEPLOYED

The feature exists in production, but can remain hidden.

FEATURE RELEASED

Flip the flag to make it available to selected users.

The important part is not just ON or OFF — it is control.

Deploying is not releasing

Two separate decisions. Two separate moments.

DEPLOY

Code reaches production.

RELEASE

Users actually see the feature.

Think: stocking a shelf before opening the store.

Why big-bang releases hurt

One switch. Every user. All at once.

100% OF USERS EXPOSED

A bug can affect everyone — and recovery may require another deployment.

Flags put you in control

Change exposure without shipping new code.

TARGET

Only selected users see it

STOP

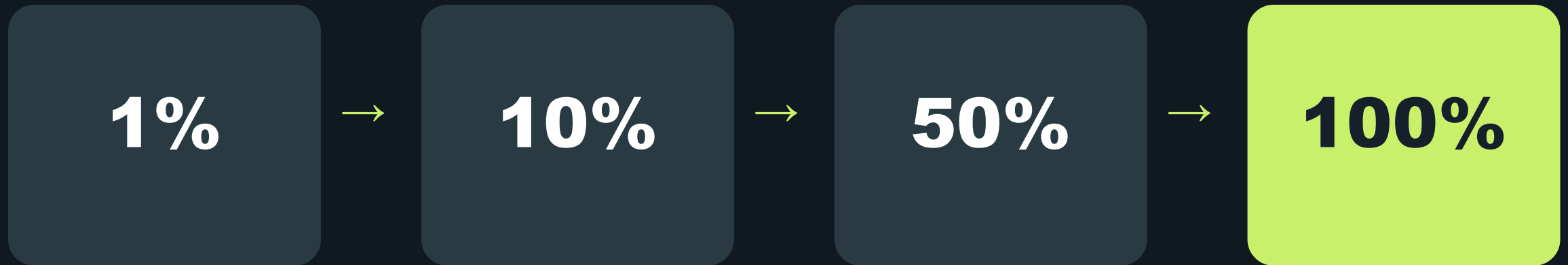
Switch off in seconds

RECOVER

No redeploy required

Progressive rollouts

Open the door slowly.



At every stage: check signals, then expand.

Canary releases

Test with a few before rolling out to many.

SMALL CANARY GROUP

Release to a limited audience first.

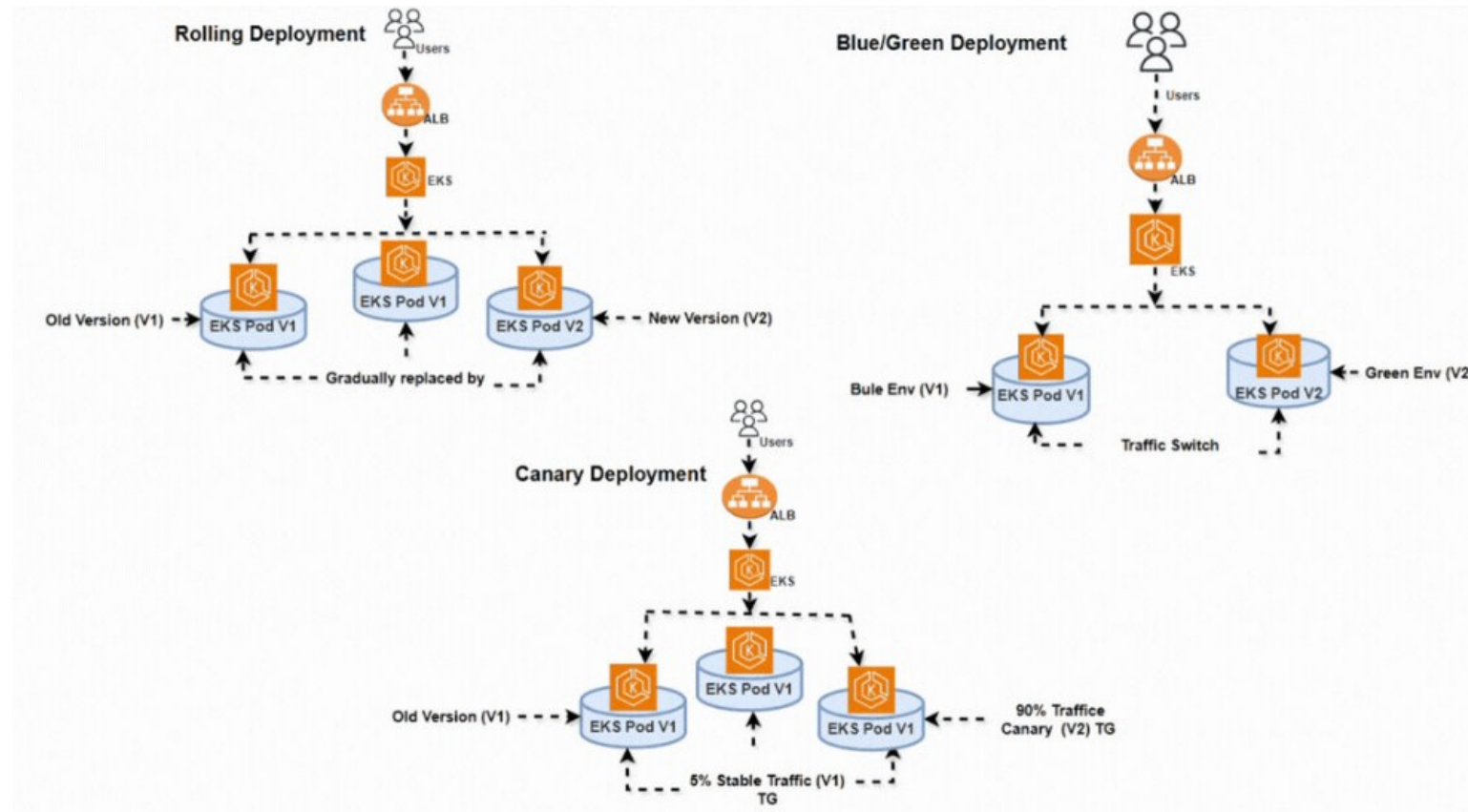
WATCH THE SIGNALS

Track errors, latency and user impact.

Expand only when the canary behaves as expected.

Deployment strategies in Kubernetes

Rolling, blue/green and canary deployments — architecture comparison



Safe experiments with A/B tests

Make decisions using actual outcomes.

VERSION A

Existing experience

VERSION B

New experience

Compare results → keep what works

Rollback is a flip

Not a fire drill.

01

Spot a problem

02

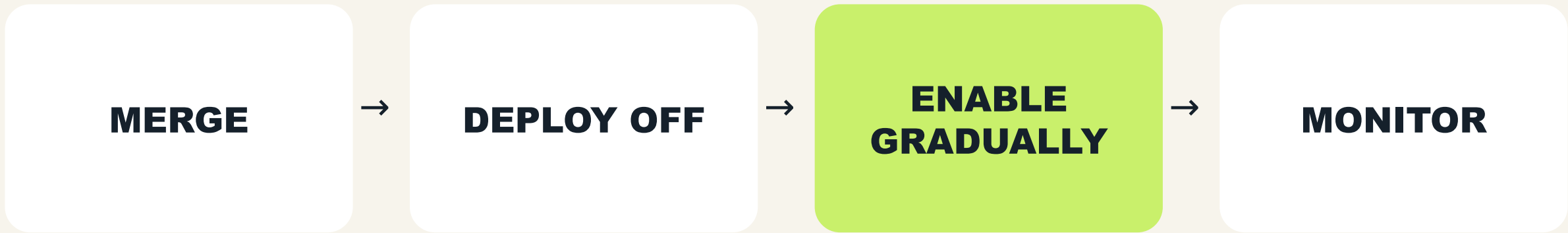
**Turn the flag
OFF**

03

Fix safely

Fit flags into your cloud workflow

Progressive delivery becomes part of your pipeline.



Small changes. Reversible exposure. Observable outcomes.

Getting started

Keep it small. Make it repeatable.

- 01 Choose one risky feature**
- 02 Use a flag service or simple config**
- 03 Give the flag a clear name**

Mistake #1: flags that never go away

Old flags become technical debt.

FLAG CLUTTER

THE FIX

Set a removal date when you create the flag.

Temporary controls should have an expiry plan.

Mistake #2: too many flags

More flags create more paths to test.

2^n

n independent binary flags can create up to 2^n combinations.

Avoid tangled dependencies between flags.

Mistake #3: no ownership or rules

Production switches need governance.

OWNER

Who is accountable?

ACCESS

Who can change it?

AUDIT

What changed, and when?

A simple flag checklist

Before the flag goes live.

- ✓ **Clear name and owner**
- ✓ **Safe default value**
- ✓ **Planned cleanup date**
- ✓ **Monitoring in place**

SHIP OFTEN. RELEASE CALMLY.

Separate deploy from release. Roll out gradually. Start with one flag this week.

ONE FEATURE · ONE FLAG · ONE SAFER RELEASE