Unleash your feature flags!

@jhuspek, OpenAlt 2023-11-12

Jakub Huspek

Maker & Night owl **@baslirna**

IT Solution Designer **@T-Mobile**

bastlíři SH
MacGyver
macgyver.siliconhill.cz

What is not a feature flag?

```python
COOL_FEATURE = True


def do_something():
    if COOL_FEATURE:
        print("New feature is enabled!")
    else:
        print("New feature is disabled.")
```

Feature Flags (often also refered to as Feature Toggles) are a powerful technique, allowing teams to modify system behavior **without changing code.**

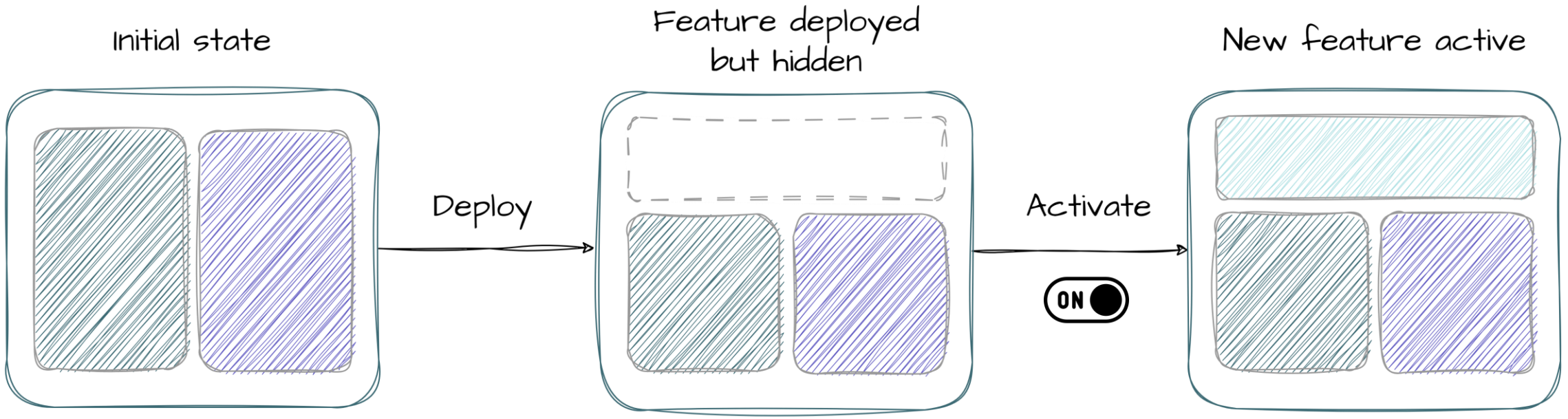
Martin Fowler, Feature Toggles

What is a feature flag?

```
client = FeatureFlagClient.initialize()


def do_something():
    if client.is_enabled("COOL_FEATURE")
        print("New feature is enabled!")
    else:
        print("New feature is disabled.")
```

What is a feature flag?



allow_darkmode

[on / off]

enable_customer_chat

[on / off]

[customer_type = B2B]

[language = cz]

A bit of general theory, but everyone can have a different motivation or different needs.

Why should you care?

From more practical cases:

- Independent (Quicker) release cycle
- Rollback / Kill-switch
- Testing in production
- Early / Block access
- Calendar driven launches

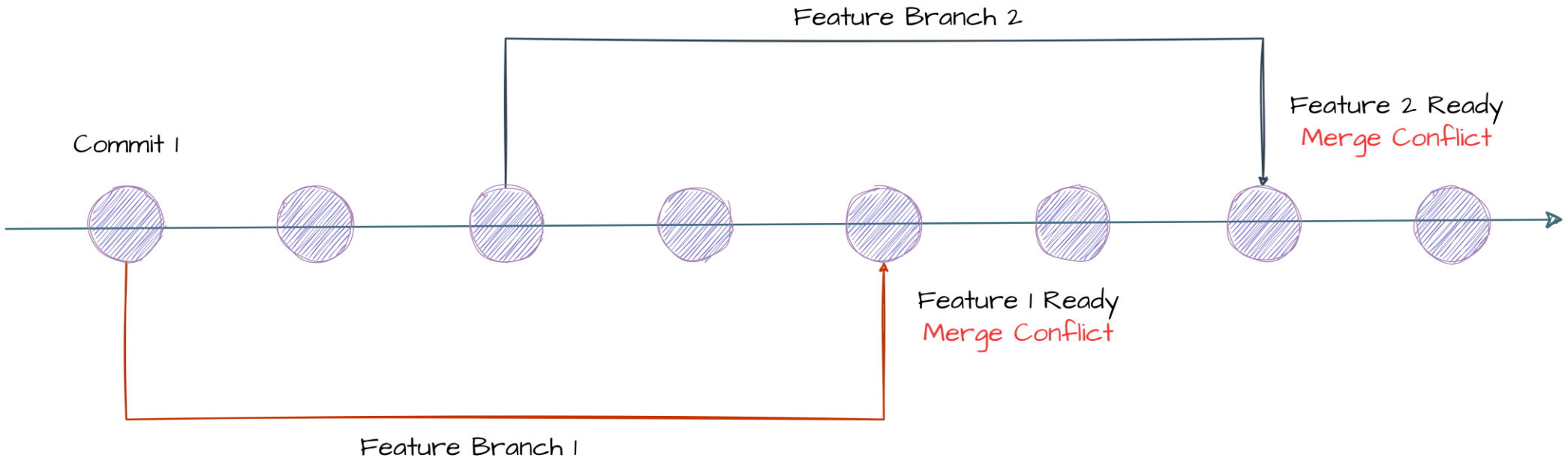
To more *theoretical ones:

- Maintenance
- Canary releases
- Incremental roll outs
- Hypothesis driven development (A/B)
- Newbie / Advanced users

*) My perspective only, as I have experience from not so much dynamic environment.

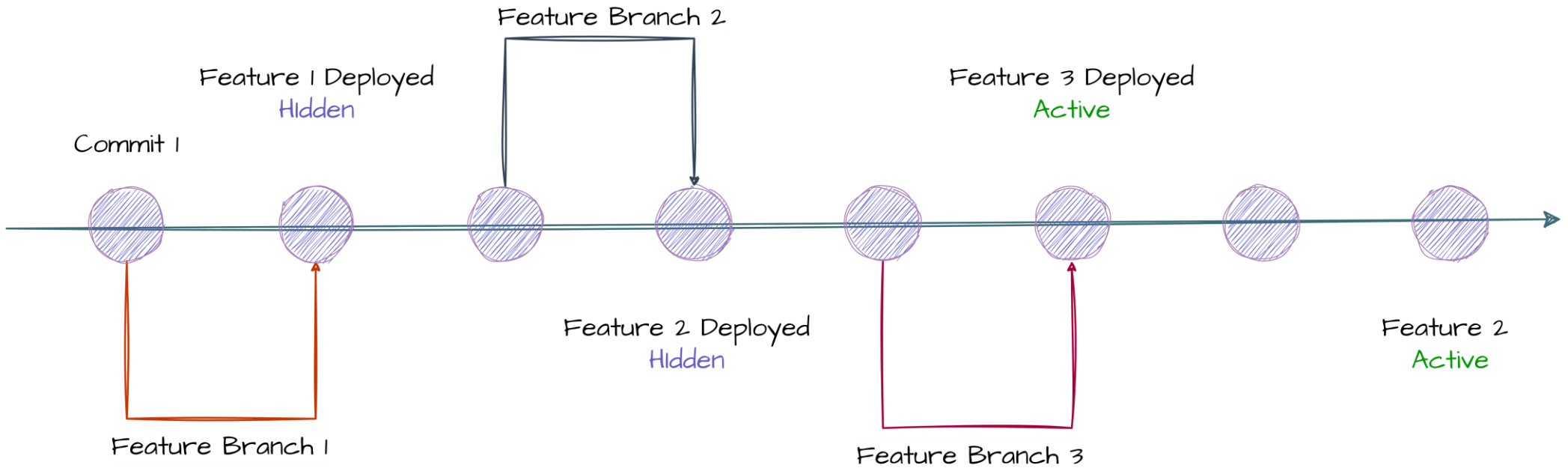
It is still one of many methods to deal with these situations. Sometimes you need to change the process around it as well for effective use.

To branch or not to branch?



“If you merge every day, suddenly you never get to the point where you have huge merge conflicts that are hard to resolve.” — Linus Torvalds

To branch or not to branch?



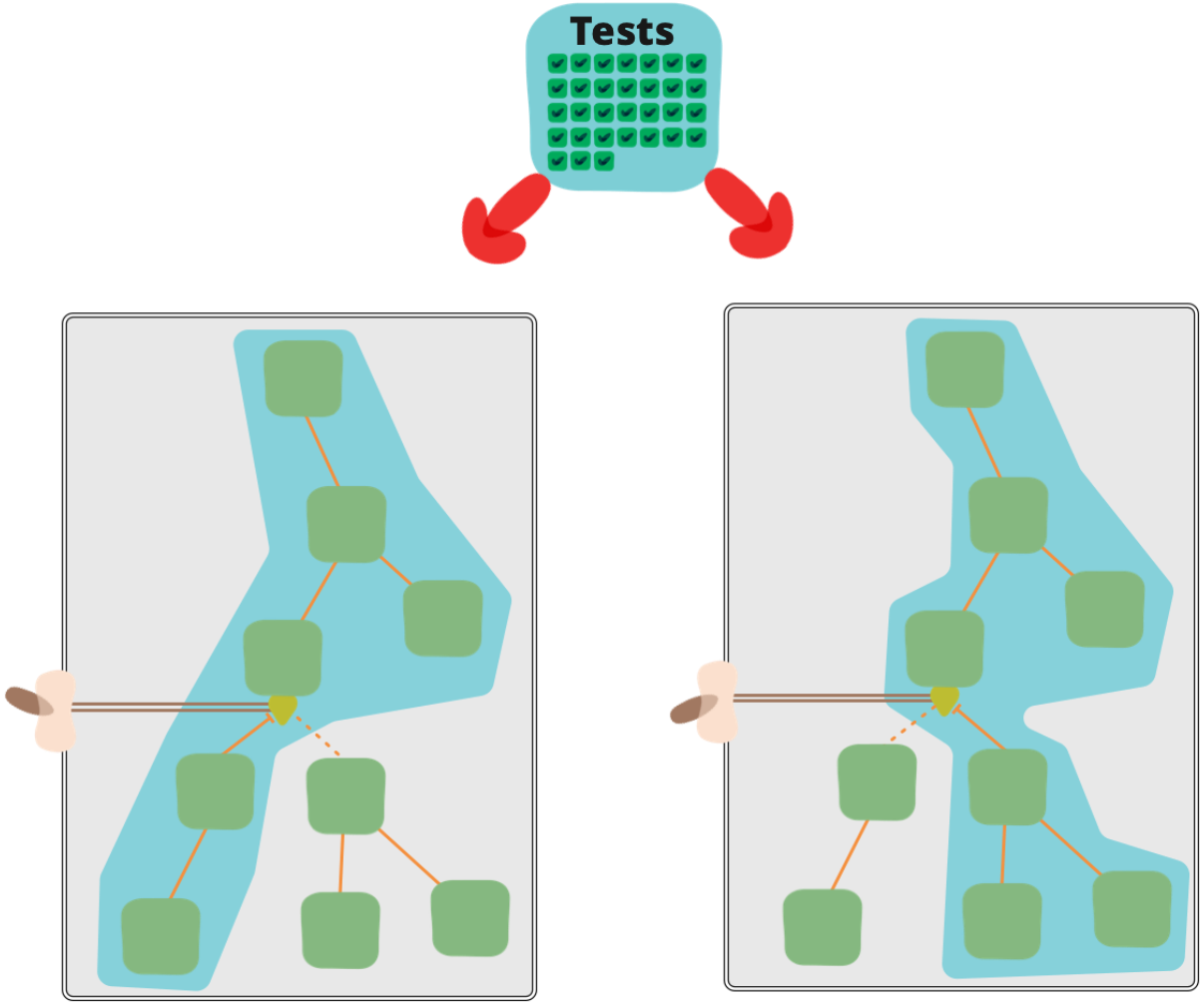
Feature flags does not replace branching, it is complementary.

And how to benefit?

- Flexibility in feature release
- Reduced risk of deploying new features
- Separation of deployment from release
- Ability to perform closed testing and experimentation, even in production
- Shorter development cycle
- Simplified version control
- One of the gates to continues deployments

But it’s not all roses.

Validation complexity



Source: martinfowler.com/articles/feature-toggles.html

Any other pitfalls?

- Testing and validation complexity
- Tech. depts if not managed correctly and FF accumulate
- Flags proliferation can clutter the codebase
- Badly selected level of flagging
- Carrying costs of feature flags
- Insufficient management and monitoring

So there must be some recommendations?

How to lose half a billion dollars with bad feature flags

The demise of Knight Capital

Vineeth Madhusudanan · Follow

Published in Statsig · 2 min read · Jul 13, 2021

Knight Capital was the largest trader in US equities in 2012 (~$21b/day) thanks to their high frequency trading algorithms. They also executed trades on behalf of retail brokers like TD Ameritrade and ETrade.

Their demise came in 2012 when they developed a new feature in their Smart Market Access Routing system to handle transactions for a new NYSE program.

PowerPeg

For testing purposes only

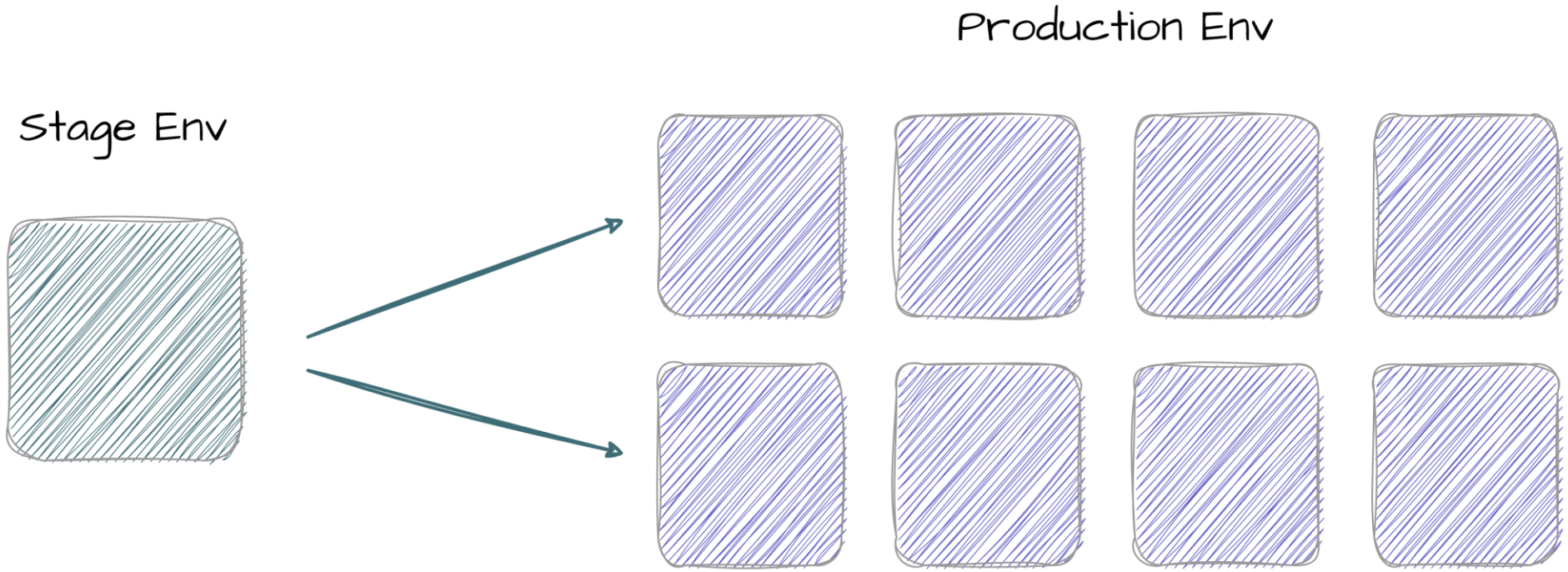
Unused since 2003 (8 years)

Validation algorithm for another component, buy high sells low

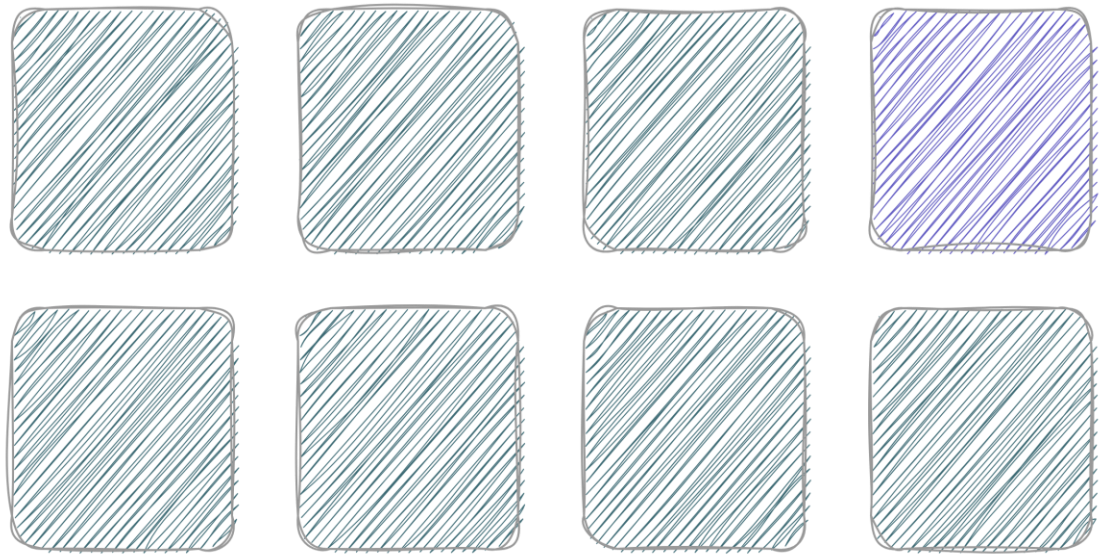
SMARS

Newly build core algorithm for routing the orders

Uses same FF as PowerPeg


Production Env
Stage Env
Current Code
New Code

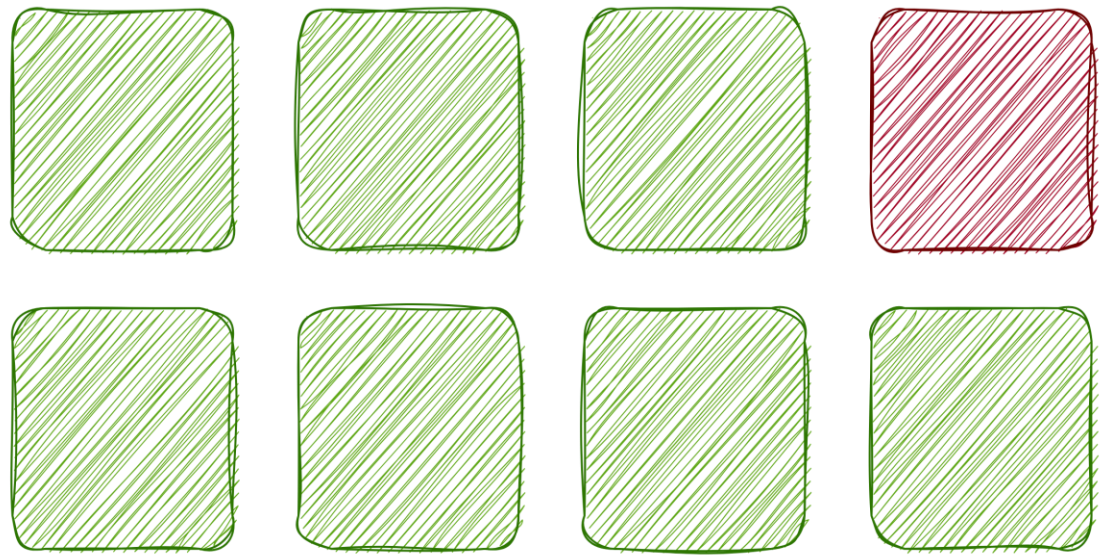
Production Env
(FF OFF)



Current Code

New Code

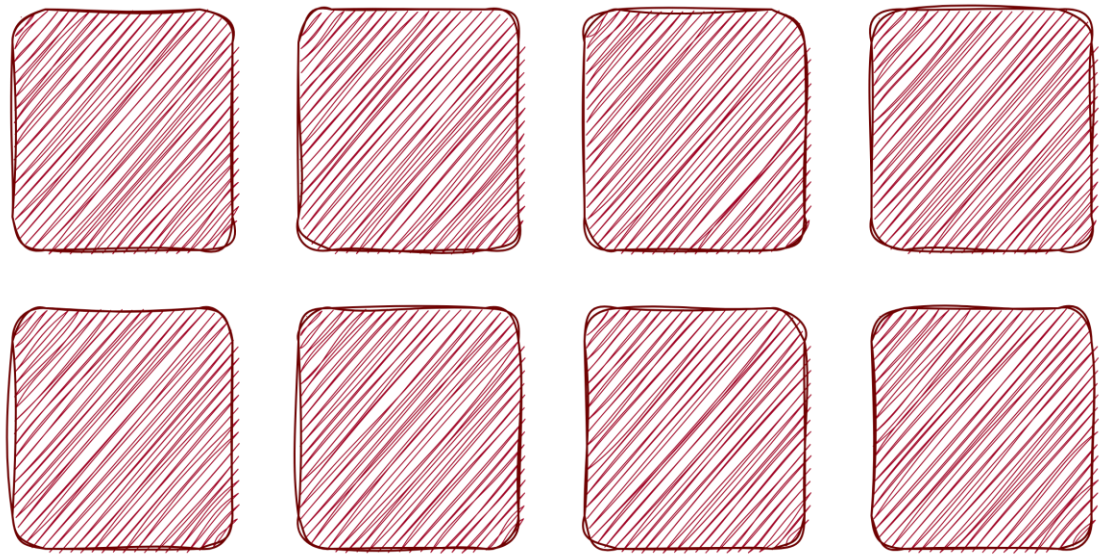
Production Env
(FF ON)



Old Code Activated

New Code Activated

Production Env -
Rollback (FF ON)



Old Code Activated

New Code Activated

Recommendation No. 1

Never reuse old feature flags.

Recommendation No. 2

Be proactive in removing feature flags that are no longer needed.

Recommendation No. 3

Choose descriptive names for your flags.

Recommendation No. 3

Choose descriptive names for your flags.

enable_power_peg

activate_smars_algorithm

feature_test_8

feature_jira_1867

And now together

- Ensure consistency (especially data) by destructive changes
- Be proactive in removing old flags
- All new features must be tested
- Choose right level of flagging
- Use them with measure, can get out of control
- Keep lifespan of flags short (weeks)
- Choose descriptive names
- Setup proper logging and monitoring

Why we started to implement

Before Q1/2023:

- **more frequent releases** of new changes that we can enable/disable without outage
- allow other teams to **independently develop & test** our applications
- **mitigate the risk associated** with releases by delivering small changes we can turn off
- allow **testing** the changes on **selected set of users** on the production environment

After Q1/2023:

- fully support the **trunk-based development** on multiple environments
- fully support **short-lived** feature branches approach

Before you start to implement

- identify your needs, you can benefit even from just of few use cases
- identify the right places and sets of the application/components where it makes sense to implement it
- find the right contexts/strategies that will be interesting in context of the whole company (area)
- materialize the benefit (effort to implementation / usage of the flags / code changes / flag type)
- keep it in mind while creating new applications from the beginning, difficult to change existing
- research a build versus buy solution to help you with management of flags
- do not forget the processes associated with feature flags (introduction, cleanup, production enablement)

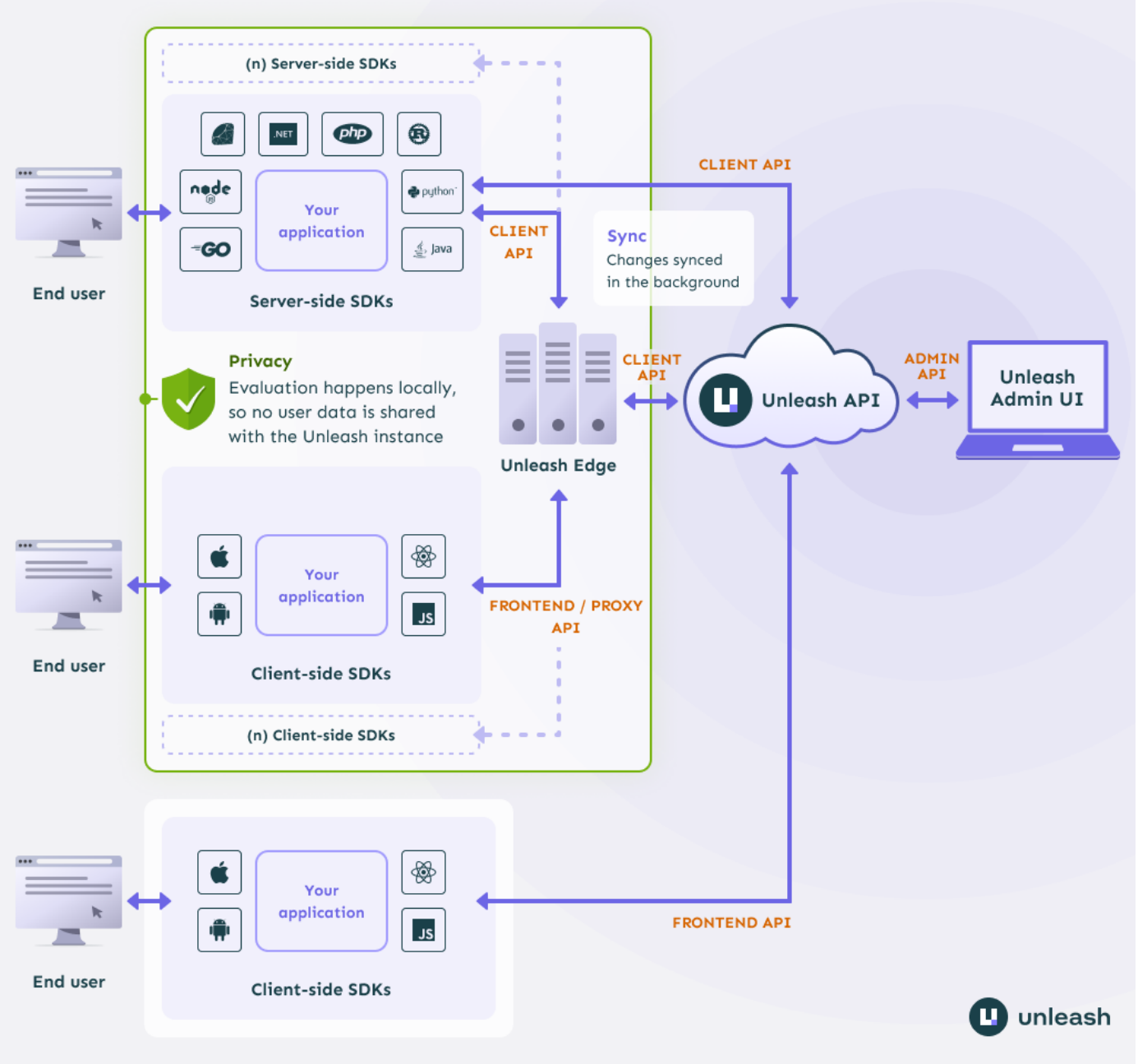
Toggles introduce complexity.

We can keep that complexity in check by using smart toggle implementation practices and appropriate tools to manage our toggle configuration.


unleash

Unleash

- Feature flag management tool
- Open-source, no vendor lock in
- Fully transparent lifecycle, communication, open to contributions
- Free and Enterprise plan available
- On-premise and hosted solution possible
- Very active community and development
- A lot of languages already supported by SDKs
- Several deployment methods available



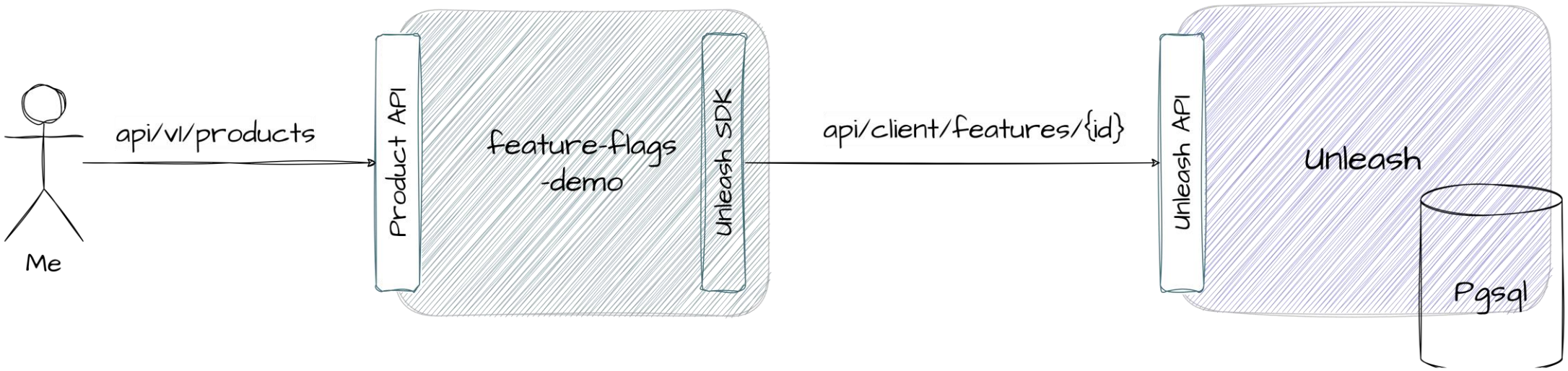
Server-side SDKs:

- Go SDK
- Java SDK
- Node.js SDK
- PHP SDK
- Python SDK
- Ruby SDK
- Rust SDK
- .NET SDK

Client-side SDKs:

- Android SDK
- Flutter Proxy SDK
- iOS Proxy SDK
- Javascript SDK
- React Proxy SDK
- Svelte Proxy SDK
- Vue Proxy SDK

Demo components



`GET /api/v1/products`

```
[
  {
    "id": "P-123",
    "status": "Active",
    "price": null
  },
  {
    "id": "P-456",
    "status": "Inactive",
    "price": null
  }
]
```

Feature Flags Demo

`GET /api/client/features/product_with_price`

```
{
  "name": "product_with_price",
  "type": "release",
  "enabled": false,
  "project": "default",
  "stale": false,
  "strategies": [
    {
      "name": "default",
      "constraints": [],
      "parameters": {},
      "variants": []
    }
  ],
  "variants": [],
  "description": null,
  "impressionData": false
}
```

Unleash

What else to read?

- Martin Fowler – Feature Toggles
- FeatureFlags.io
- getunleash.io