FUN WITH FLAGS

Bring the Fun Back into Feature Flagging with OpenFeature

: simon.schrottner@dynatrace.com
: @aepfli(...)

Simon Schrottner

SIMON SCHROTTNER

- Senior Software Engineer @ Dynatrace
- OpenFeature maintainer
- Open Source Enthusiast

: simon.schrottner@dynatrace.com
: @aepfli(...)

© haraldtauderer.com

AGENDA

1. What are Feature Flags?
2. Open Feature
3. Feature Flagging Pitfalls
4. (Demo)

FEATURE FLAGS

Feature Flags enable, disable or change the behavior of certain features or code paths in a product or service, without modifying the source code.

WHY?

COORDINATE AND TARGET

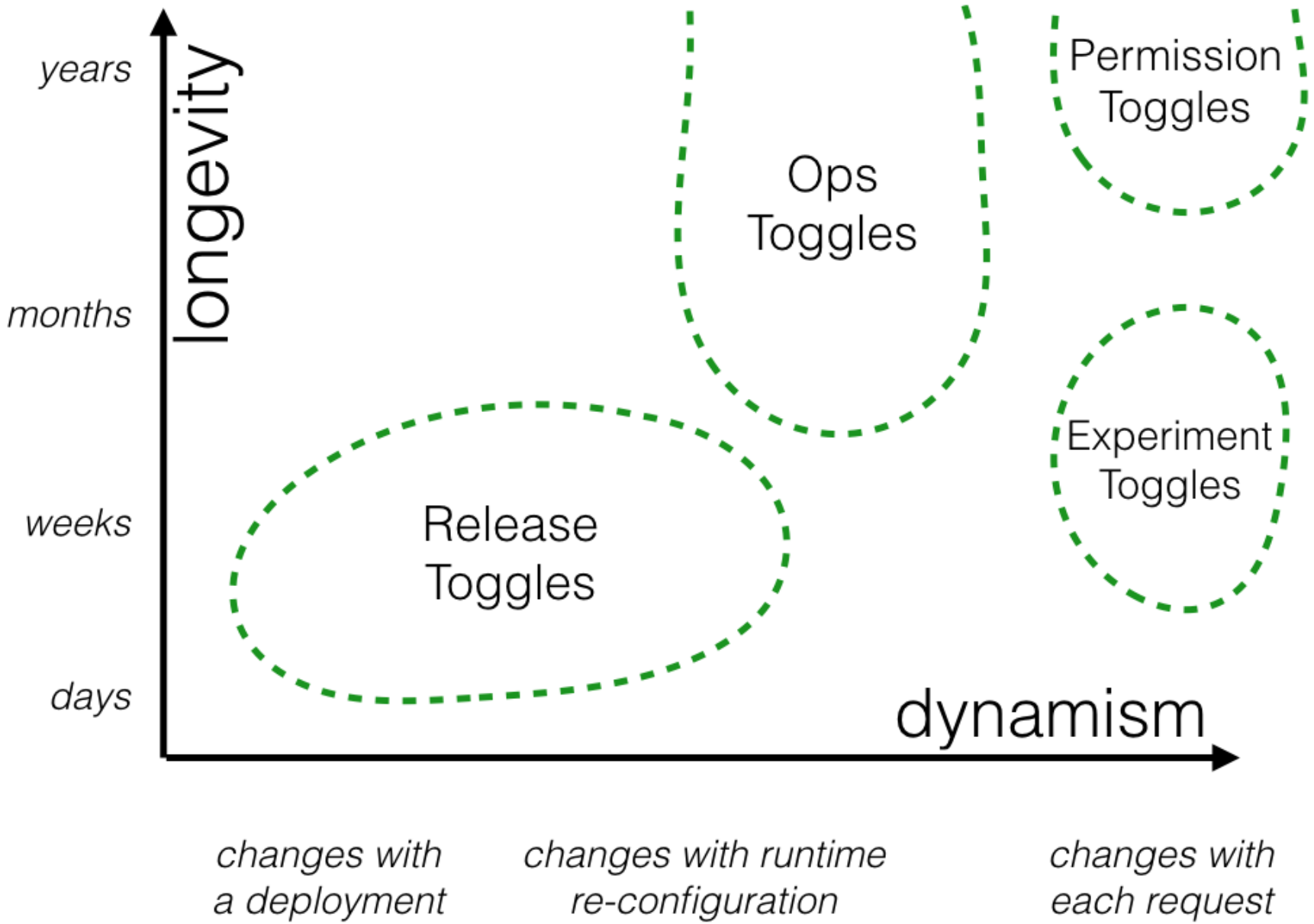
- Synchronized rollouts
- Experiments
- Usergroup specific features

REDUCE RISK

- Deployment != Release
- Risk-averse releases
- Progressive rollouts

CATEGORIES OF FEATURE FLAGS

- differ in longevity and dynamism
- different needs



https://martinfowler.com/articles/feature-toggles.html - Pete Hodgson's Feature Flag Types

Standardizing Feature Flagging for Everyone

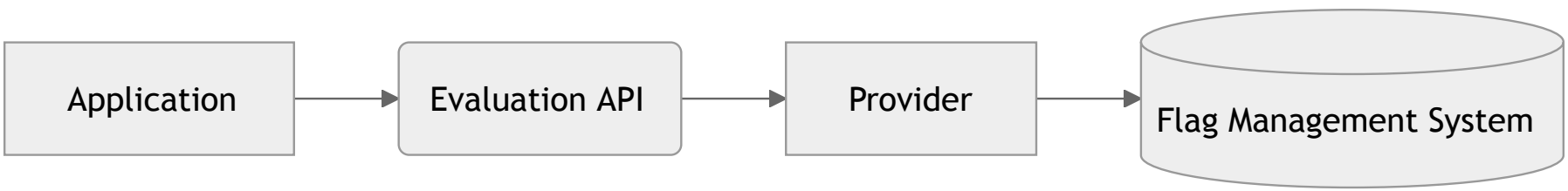
https://openfeature.dev

OpenFeature is an open specification that provides a vendor-agnostic, community-driven API for feature flagging that works with your favorite feature flag management tool. [1]

---

1. https://openfeature.dev/docs/reference/intro

FLOW


Application
Evaluation API
Provider
Flag Management System

BASIC USAGE - JAVA

```java
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new InMemoryProvider(myFlags));

Client client = api.getClient();

boolean flagValue =
    client.getBooleanValue("v2_enabled", false);
```

[1]

---

1. https://openfeature.dev/docs/reference/technologies/server/java

BASIC USAGE - NODE.JS

```
1 import { OpenFeature } from '@openfeature/server-sdk';
2 await OpenFeature.setProviderAndWait(new YourProviderOfCh
3
4 const client = OpenFeature.getClient();
5
6 const v2Enabled =
7     await client.getBooleanValue('v2_enabled', false);
```

[1]

---

1. https://openfeature.dev/docs/reference/technologies/server/javascript

BASIC USAGE - GOLANG

```
openfeature.SetProvider(openfeature.NoopProvider{})

client := openfeature.NewClient()

v2Enabled, _ := client.BooleanValue(
    context.Background(),
    "v2_enabled",
    true,
    openfeature.EvaluationContext{},
)
```

[1]

1. https://openfeature.dev/docs/reference/technologies/server/go

CONSIDERATIONS

- never breaks your code
- good default values

SUPPORTED TYPES - BOOLEAN

```
client.getBooleanValue("v2_enabled", false);
```

SUPPORTED TYPES - STRING

```
client.getStringValue("v2_enabled", "fallback");
```

SUPPORTED TYPES - NUMBER

```
client.getIntegerValue("v2_enabled", 0);
client.getDoubleValue("v2_enabled", 0d);
```

SUPPORTED TYPES - OBJECT

```
client.getObjectValue("v2_enabled", new Value());
```

EVALUATION API

The Evaluation API is the primary component of OpenFeature that application authors interact with. The Evaluation API allows developers to evaluate feature flags to alter control flow and application characteristics. [1]

---

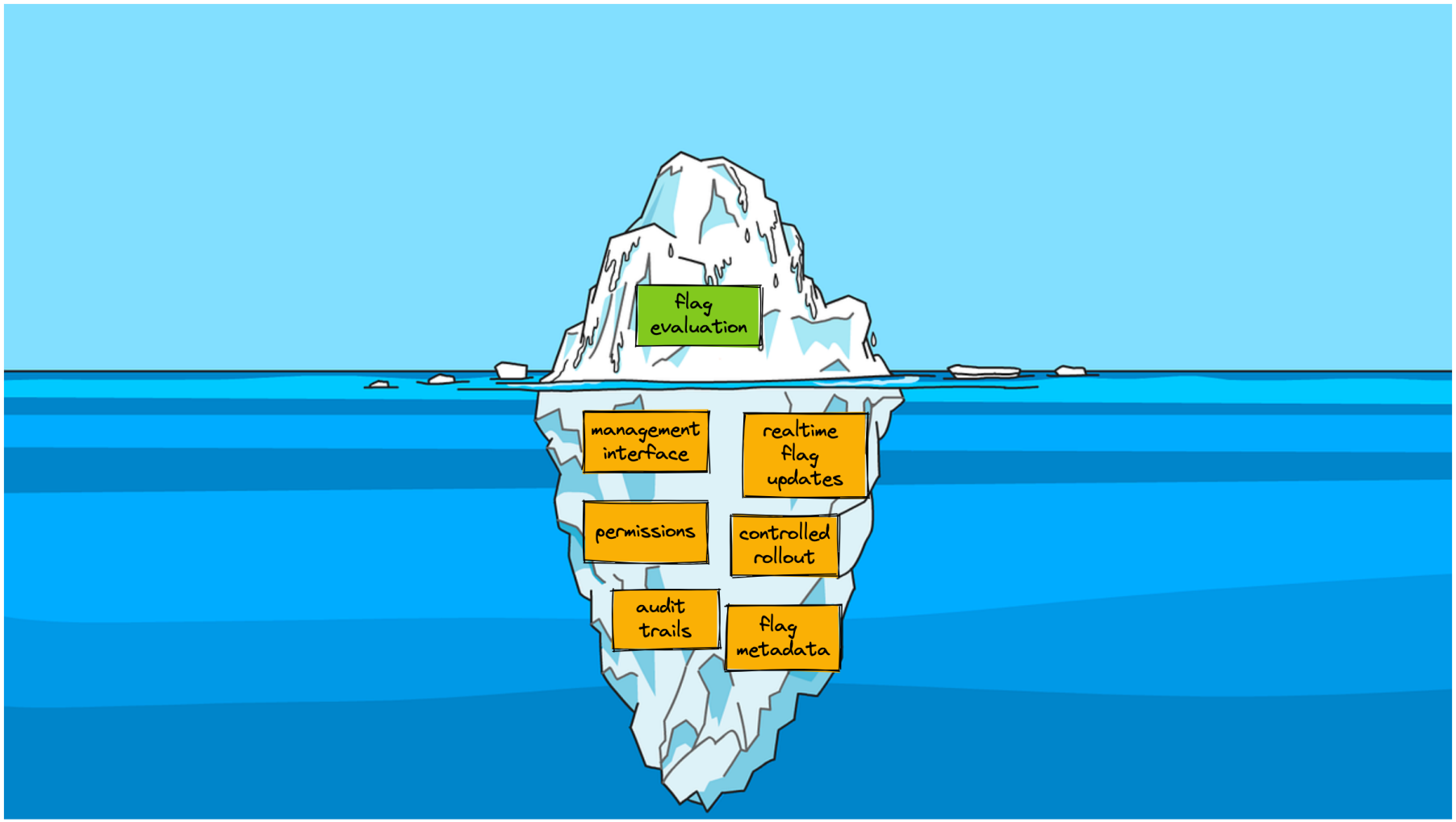
1. https://openfeature.dev/docs/reference/concepts/evaluation-api

EVALUATION API

- Easy to use API
- Multiple Languages
- Similar Interfaces

PROBLEMS WITH FEATURE FLAGS

FEATURE FLAGGING ICEBERG



https://openfeature.dev/blog/openfeature-a-standard-for-feature-flagging

TOPICS WE WILL COVER

- Vendor lock ins
- Dynamic Evaluation
- Obsolote Feature Flags

LOCK INS

WHY?

- Homegrown solutions
- Vendor specific SDKs

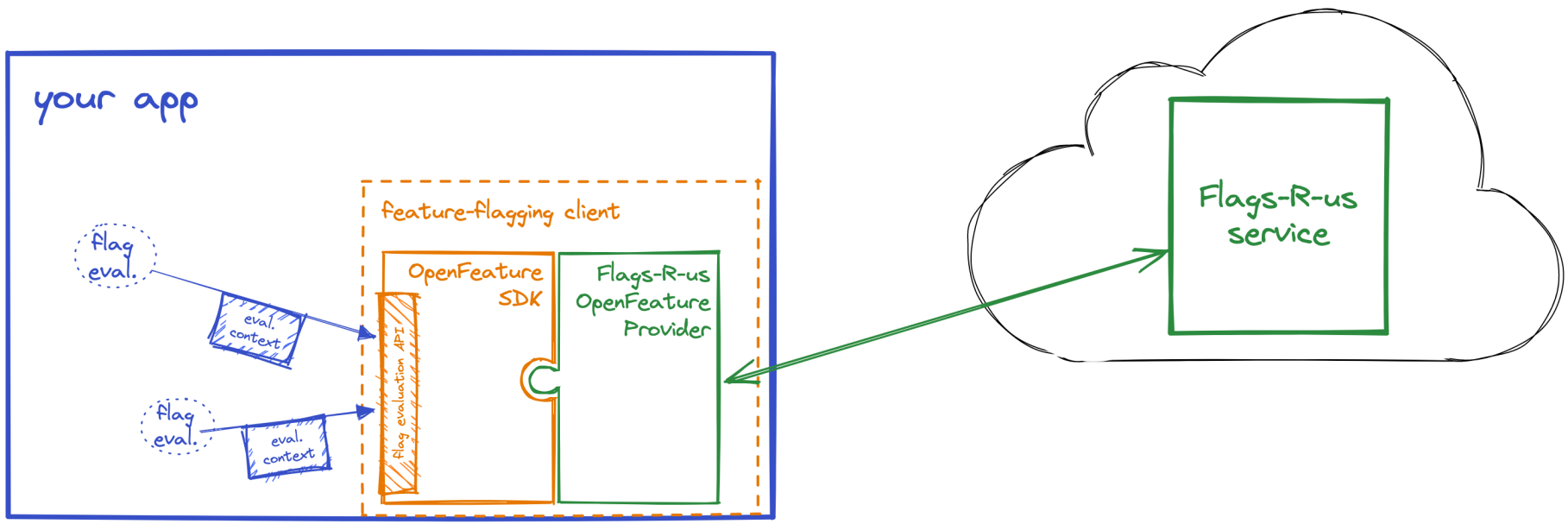
HOMEGROWN SOLUTION

- High Effort
- Limited functionality
- Hard to support additional technologies

VENDORS

- Specific SDK
- Migration pain

ARCHITECTURE



OpenFeature Provider Architecture

EXAMPLE

```
1 OpenFeatureAPI api = OpenFeatureAPI.getInstance();
2 api.setProviderAndWait(new InMemoryProvider(myFlags));
3
4 Client client = api.getClient();
5
6 boolean flagValue = client.getBooleanValue("v2_enabled",
```

EXAMPLE

```
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new NewProvider());

Client client = api.getClient();

boolean flagValue = client.getBooleanValue("v2_enabled",
```

PROVIDERS[1]

- Encapsulate feature flag management tool
- Reduces migration pains

---

1. https://openfeature.dev/docs/reference/concepts/provider

DYNAMIC EVALUATION

Changing evaluation based on rulesets.

WHY?

- A/B testing
- SDQA
- premium users
- dogfooding

FLAGD - TARGETING EXAMPLE

```
{
  "flags": {
    "v2_enabled": {
      "state": "ENABLED",
      "variants": {
        "on": true,
        "off": false
      },
      "defaultVariant": "off",
      "targeting": {
        "if": [
          {
            "ends_with": [
              { "var": "email" }, "@domain.com"
            ]
```

a simple flag config with targeting

DYNAMIC CONTEXT

```
1  Map<String, Value> requestAttrs = new HashMap<>();
2  requestAttrs.put("email",
3      new Value(session.getAttribute("email")));
4  requestAttrs.put("product",
5      new Value("productId"));
6  EvaluationContext reqCtx =
7      new ImmutableContext(requestAttrs);
8
9  boolean flagValue =
10     client.getBooleanValue("v2_enabled", false, reqCtx);
```

Changing evaluation based on rulesets for other use-cases?

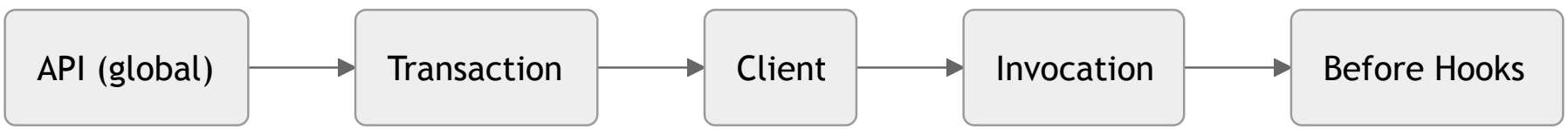
OPERATIONAL INFORMATION

- Application information
- Hyperscalers
- Operating Systems
- Environmental information

STATIC CONTEXT

```
1 OpenFeatureAPI api = OpenFeatureAPI.getInstance();
2 api.setEvaluationContext(
3     new MutableContext().add(
4         "myGlobalKey", "myGlobalValue"));
5
6 Client client = api.getClient();
7 client.setEvaluationContext(
8     new MutableContext().add(
9         "myClientKey", "myClientValue"));
```

MERGE ORDER



DETERMINISM PROBLEM

- fractional Evaluations
- percentage-based Evaluations
- ensure same Result for User

TARGETING KEY

- ensure determinism
- unique subject identifier
- optional for evaluation context

[1]

---

1. https://openfeature.dev/docs/reference/concepts/evaluation-context#targeting-key

TARGETING KEY - EXAMPLE

```
String targetingKey = session.getId();
EvaluationContext reqCtx =
    new ImmutableContext(targetingKey, requestAttrs);
```

EVALUATION CONTEXT[1]

The evaluation context is a container for arbitrary contextual data that can be used as a basis for dynamic evaluation.

---

1. https://openfeature.dev/docs/reference/concepts/evaluation-context

EVALUATION CONTEXT

- Experiment
- Reduce impact
- Increase Flexibility
- Provide Determinism

OBSOLETE FEATURE FLAGS

Feature Flags that only evaluate to the same value anymore.

WHY?

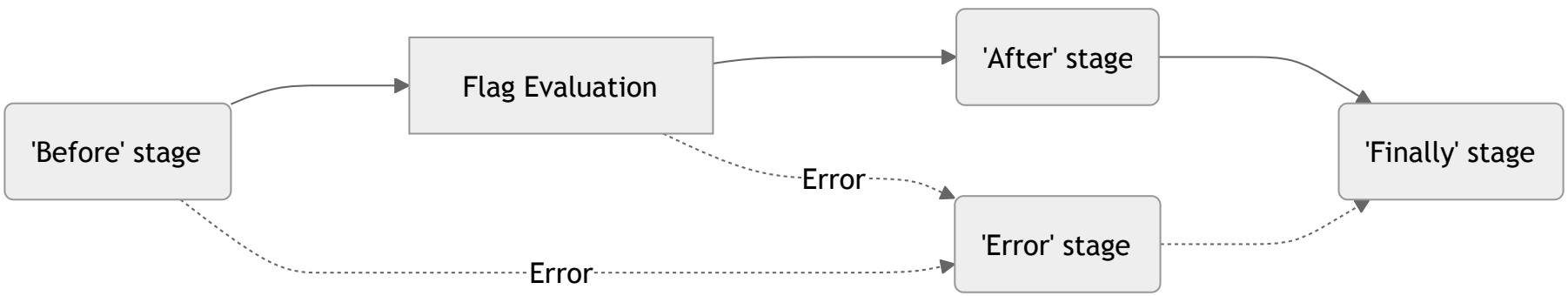
- Dead code
- Technical debt
- Increased complexity

DECOMMISSIONING

- Remove outdated features
- Remove obsolete behaviour
- Remove complexity

…BUT WHEN IS IT SAFE?

FLAG EVALUATION LIFE-CYCLE



IMPLEMENTATION - DYNAMIC

```
Boolean value = client.getBooleanValue(
    "key",
    false,
    null,
    FlagEvaluationOptions
        .builder()
        .hook(new ExampleInvocationHook())
        .build()
    );
```

IMPLEMENTATION - CLIENT

```
Client client = api.getClient();
client.addHooks(new ExampleClientHook());
```

IMPLEMENTATION - GLOBAL

```
OpenFeatureAPI.getInstance().addHooks(new ExampleGlobalHook()
```

OPENTELEMETRY

- Traces
- Metrics
- https://github.com/open-feature/java-sdk-contrib/tree/main/hooks/open-telemetry

TRACES

- After and Error stage
- Evaluation Details:
  - Key
  - Provider name
  - Variant

METRICS

- Number of evaluation requests
- Successful flag evaluations
- Errornous flag evaluations
- Active flag evaluations counter

OTHER USE-CASES?

- Logging
- Validation
- Enhancing context

HOOKS[1]

Hooks are a mechanism that allow for the addition of arbitrary behavior at well-defined points of the flag evaluation life-cycle.

---

1. https://openfeature.dev/docs/reference/concepts/hooks

HOOKS

- OpenTelemetry out of the box
- Enhance existing providers

TAKE AWAYS

MANY CONCEPTS

SUPPORTS *EVERYONE* WITHIN THE SOFTWARE DELIVERY LIFE-CYCLE

BRINGS CONFIDENCE TO *EVERYONE*!

https://openfeature.dev

Q&A

1. What are Feature Flags?
2. Open Feature
3. Providers
4. Dynamic Evaluations
5. Hooks
6. (Demo)

: simon.schrottner@dynatrace.com
: @aepfli(...)

DEMO