FUN WITH FLAGS

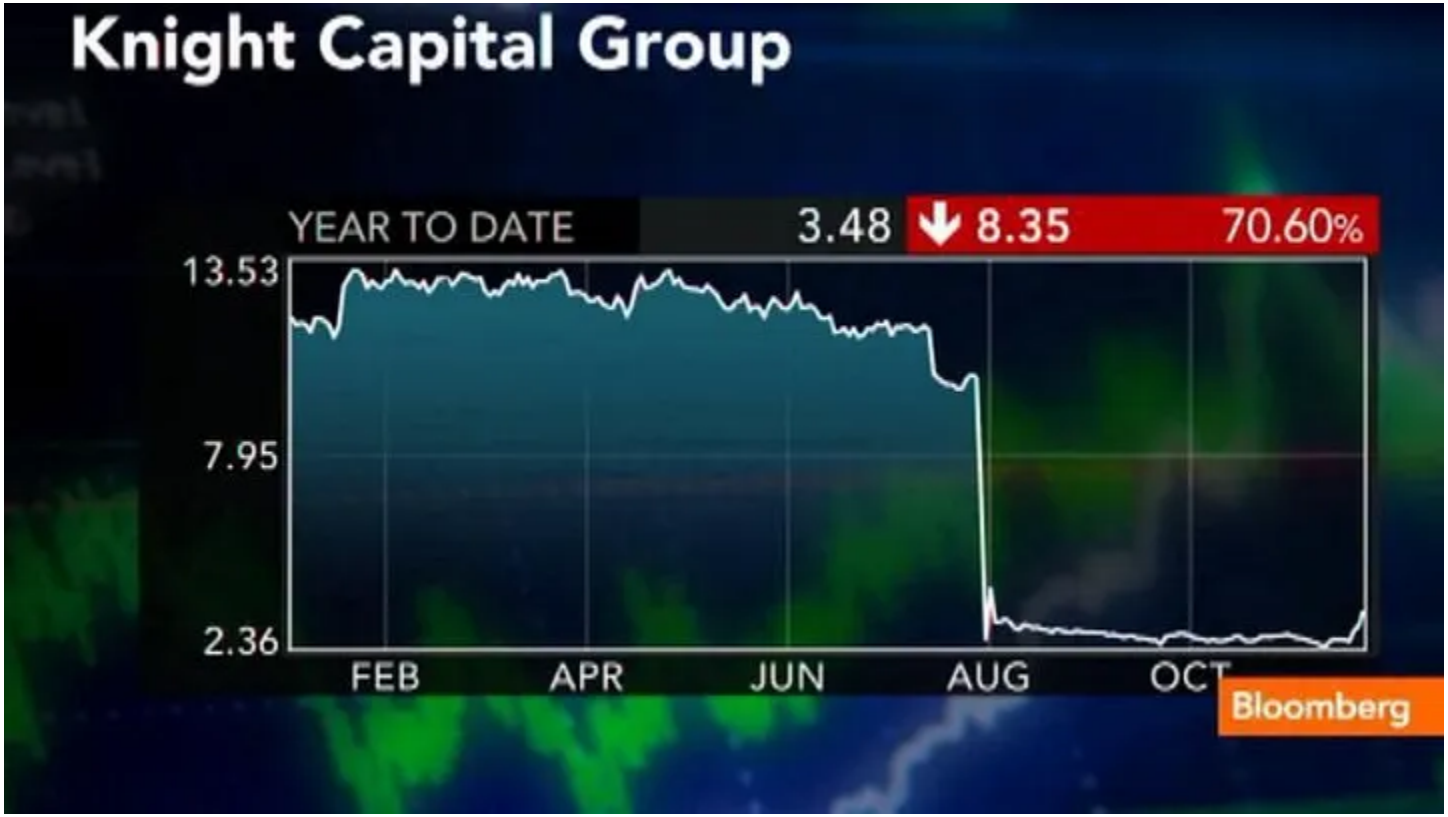
Bring the Fun Back into Feature Flagging with OpenFeature

✉: simon.schrottner@dynatrace.com
in 🦋: @aepfli(...)

Simon Schrottner

HOW NOT TO DO FEATURE FLAGS

based on Knight Capital Group



knight capital

- Lost half a billion USD
- … in one hour
- … due to a Feature Flag
- … they repurposed!

https://blog.statsig.com/how-to-lose-half-a-billion-dollars-with-bad-feature-flags-ccebb26adeec

FUN WITH FLAGS

Bring the Fun Back into Feature Flagging with OpenFeature

SIMON SCHROTTNER

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

✉: simon.schrottner@dynatrace.com
in 🦋: @aepfli(...)

© haraldtauderer.com

AGENDA

1. What are Feature Flags?
2. Open Feature
3. Feature Flagging Pitfalls

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

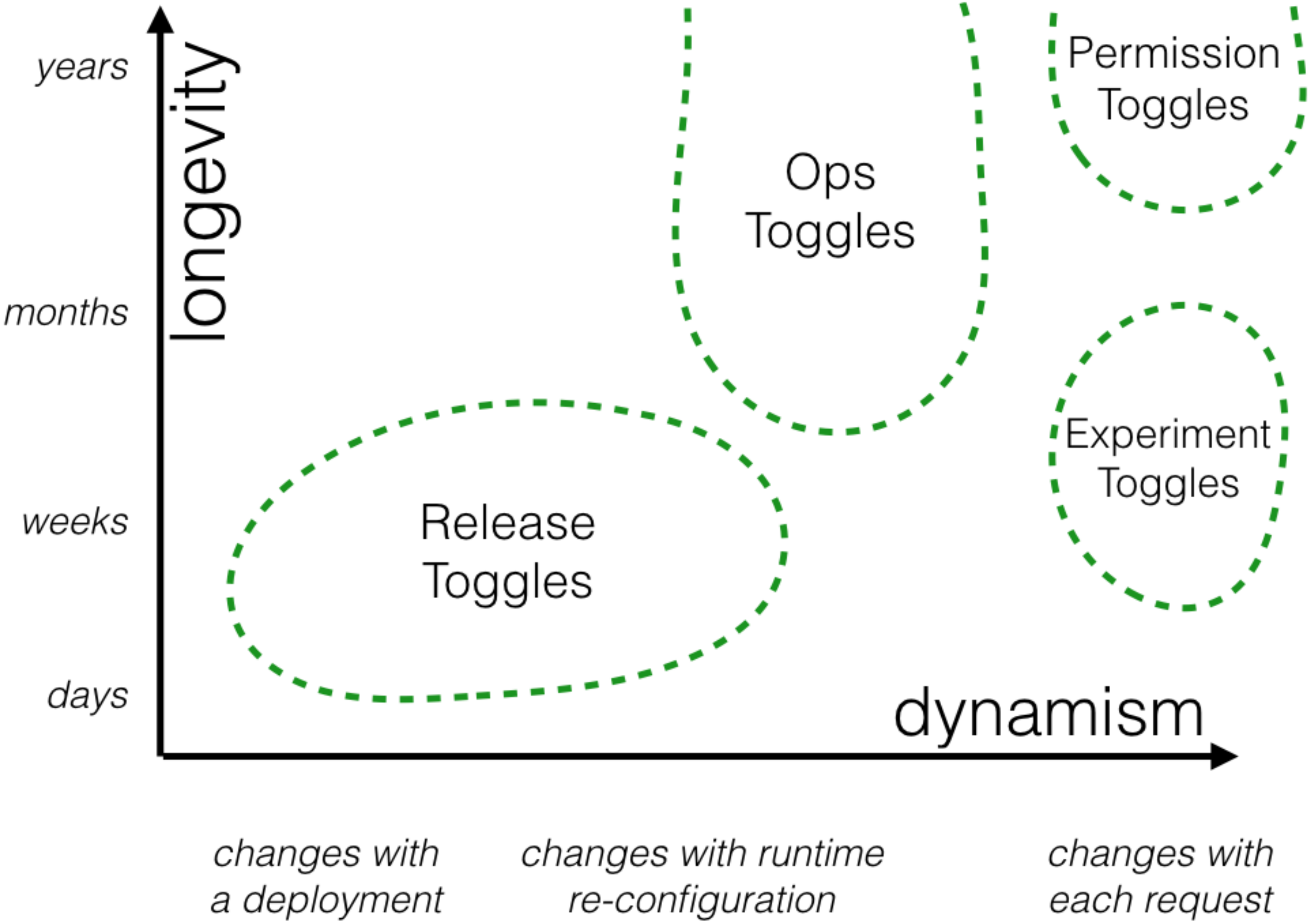
- Synchronized rollouts
- Experiments
- Usergroup specific features

REDUCE RISK

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

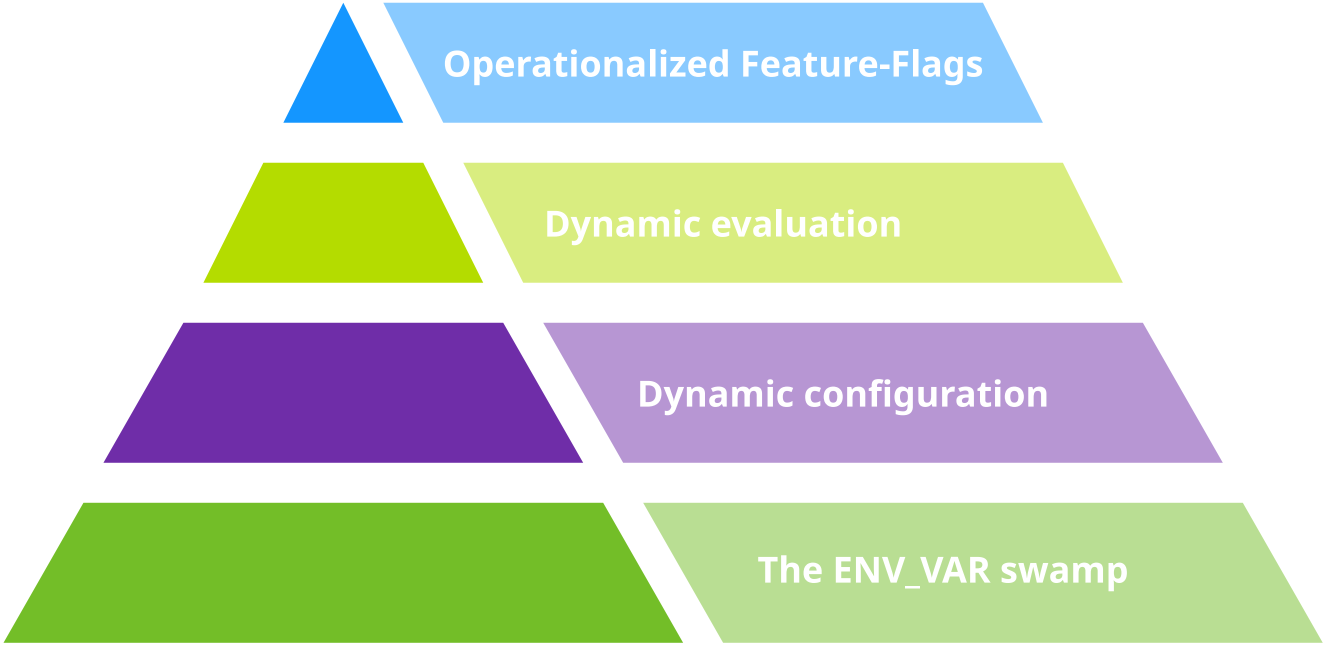
CATEGORIES OF FEATURE FLAGS

- differ in longevity and dynamism
- different needs



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

MATURITY MODEL



Operationalized Feature-Flags

Dynamic evaluation

Dynamic configuration

The ENV_VAR swamp

OpenFeature

Standardizing Feature Flagging for Everyone

CLOUD NATIVE
COMPUTING FOUNDATION
INCUBATING PROJECT

https://openfeature.dev

HISTORY

- Initialized by Dynatrace
- KubeCon Valencia 2022
- Collaborative Effort

REASONING

- Observability
- Insights
- Internal Pains

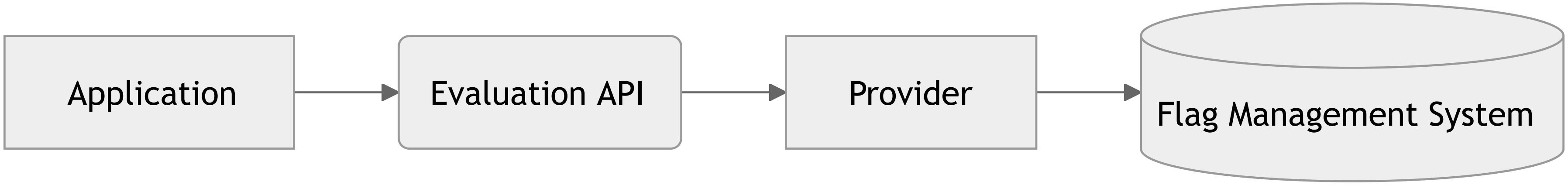
INTERNAL PAINS

- little granularity (off/on per tenant)
- baked into binary; can be overwritten at runtime
- inability to target specific users, run experiments, or do progressive roll-outs
- Java-only solution (really no support for the frontend, or other backend languages)

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]

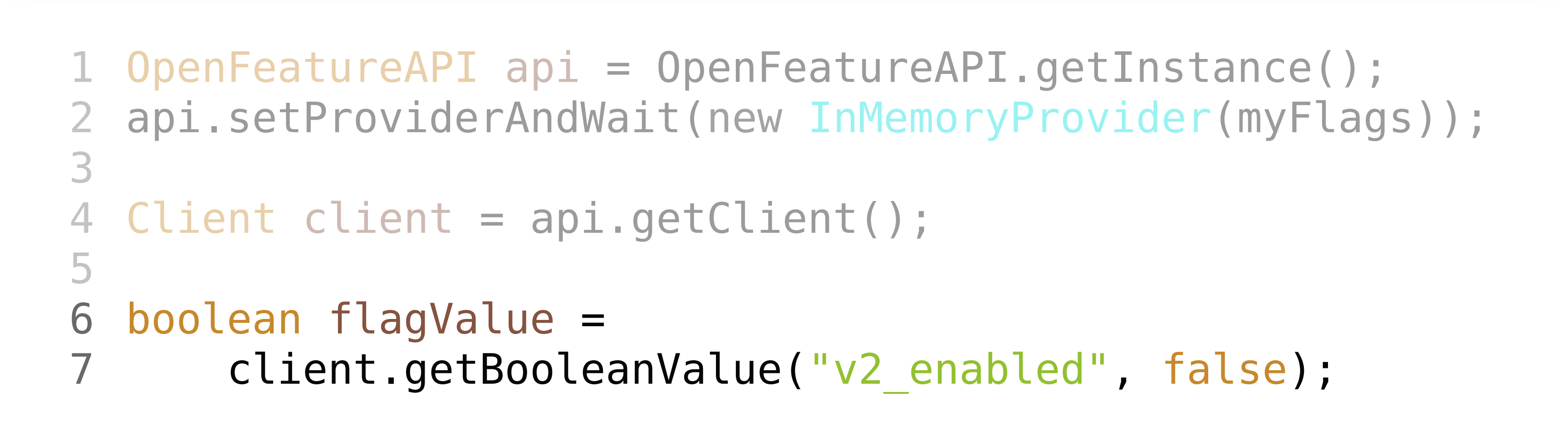
[1] https://openfeature.dev/docs/reference/intro

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

```javascript
import { OpenFeature } from '@openfeature/server-sdk';
await OpenFeature.setProviderAndWait(new YourProviderOfCh

const client = OpenFeature.getClient();

const v2Enabled =
    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 fallback/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]

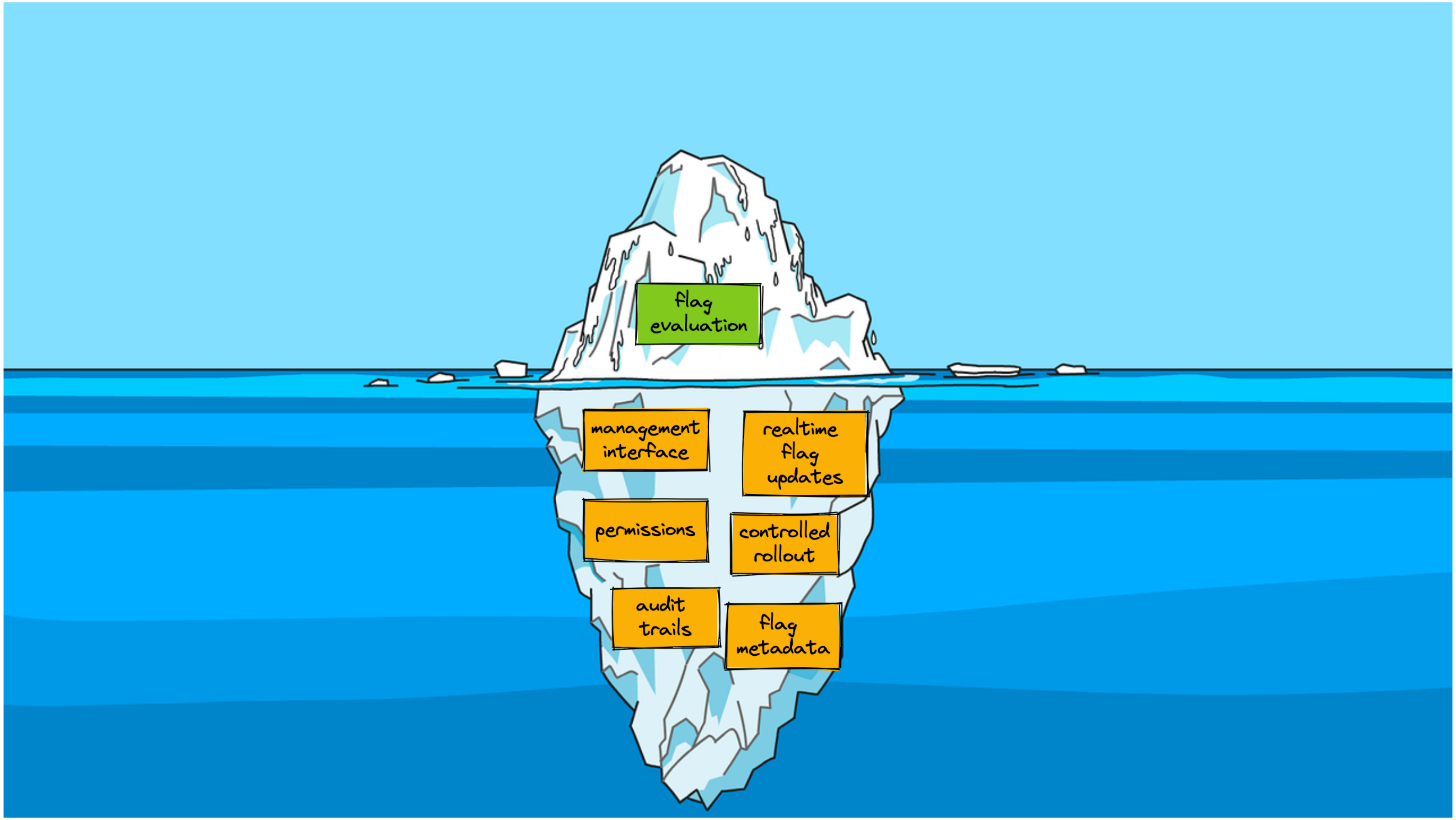
[1] https://openfeature.dev/docs/reference/concepts/evaluation-api

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
- Obsolete Feature Flags

LOCK INS

WHY?

- Homegrown solutions
- Vendor specific SDKs

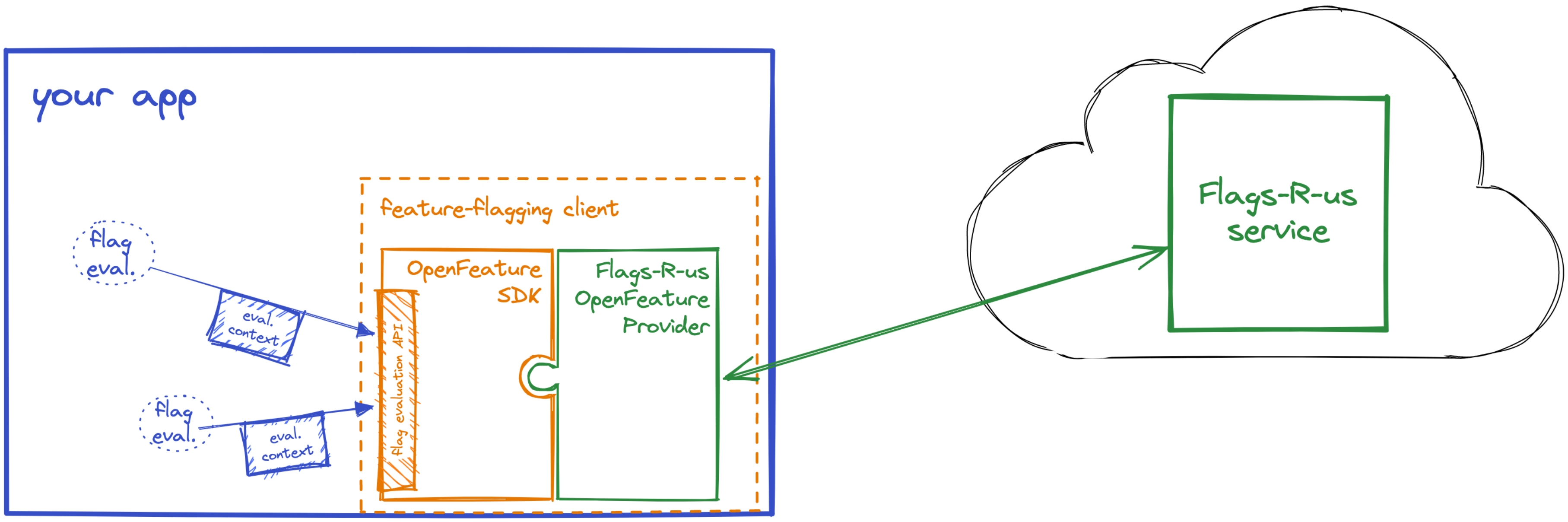
HOMEGROWN SOLUTION

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

VENDORS

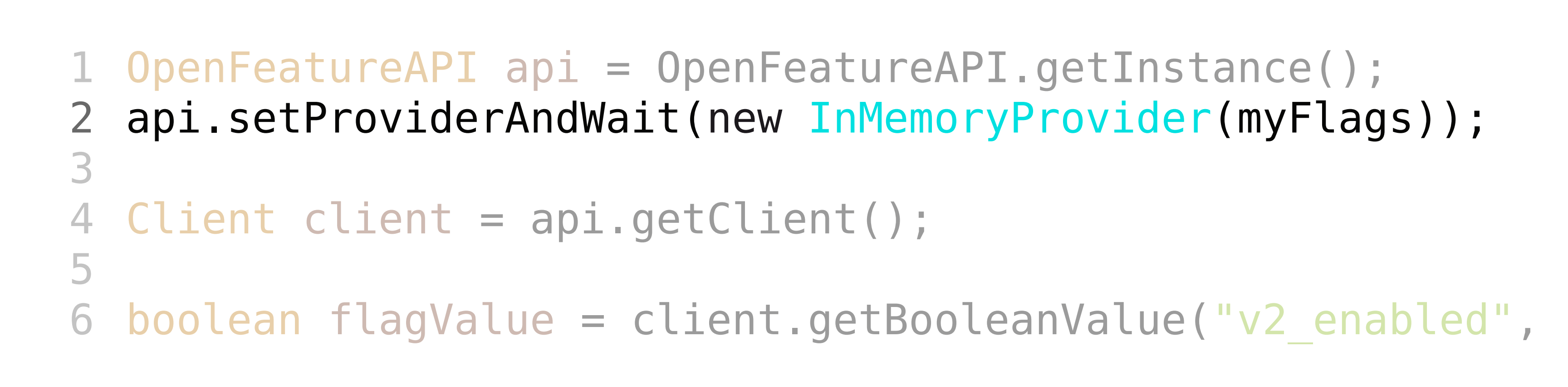
- Specific SDK
- Migration pain

ARCHITECTURE



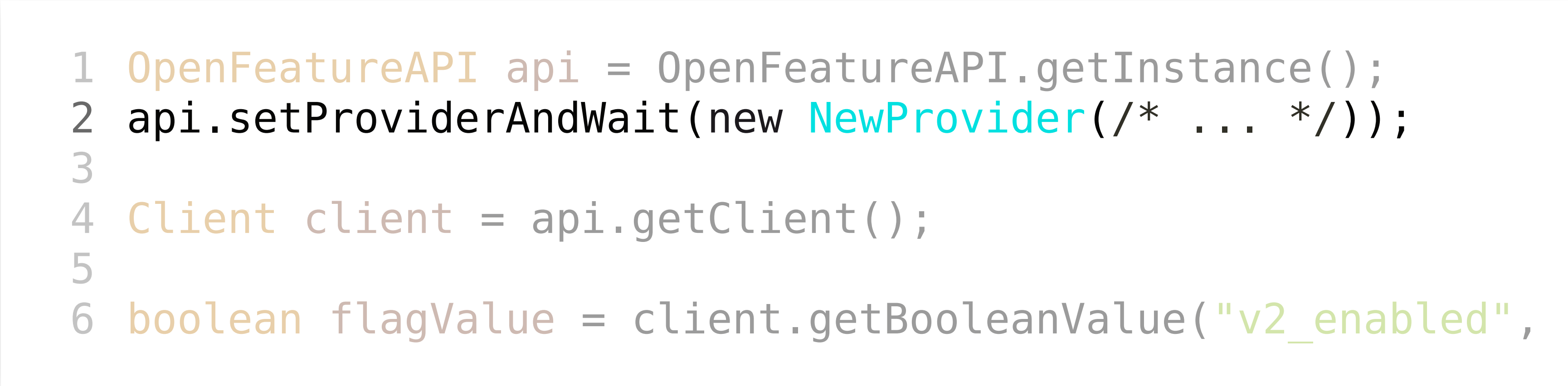
OpenFeature Provider Architecture

EXAMPLE



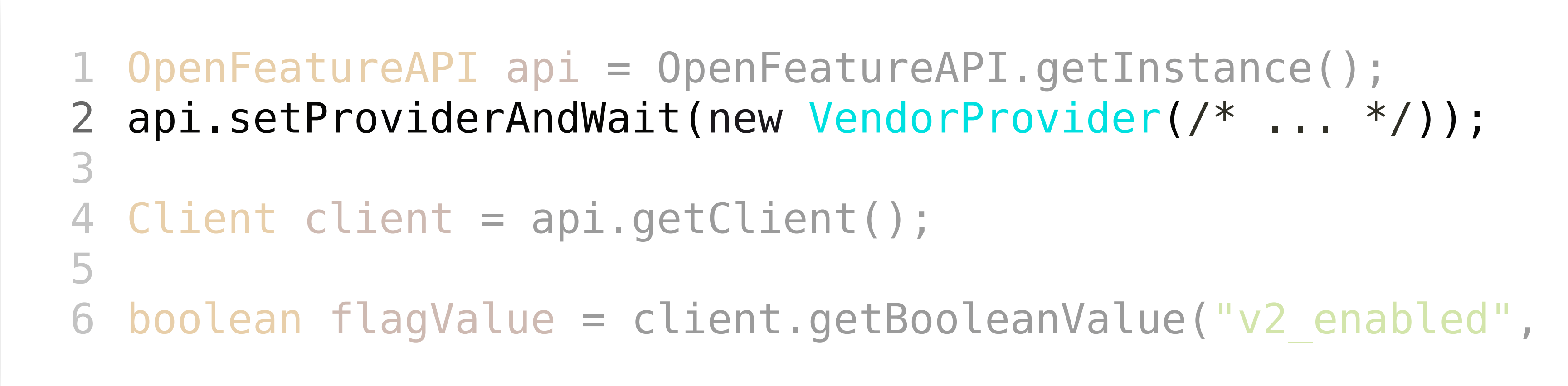
```java
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new InMemoryProvider(myFlags));

Client client = api.getClient();

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

EXAMPLE



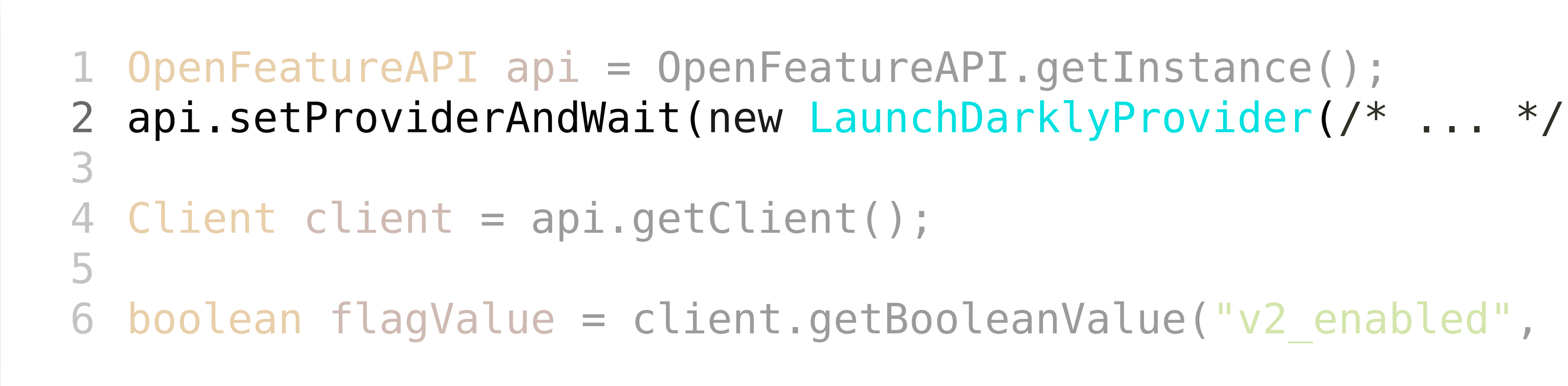
```
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new NewProvider(/* ... */));

Client client = api.getClient();

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

EXAMPLE



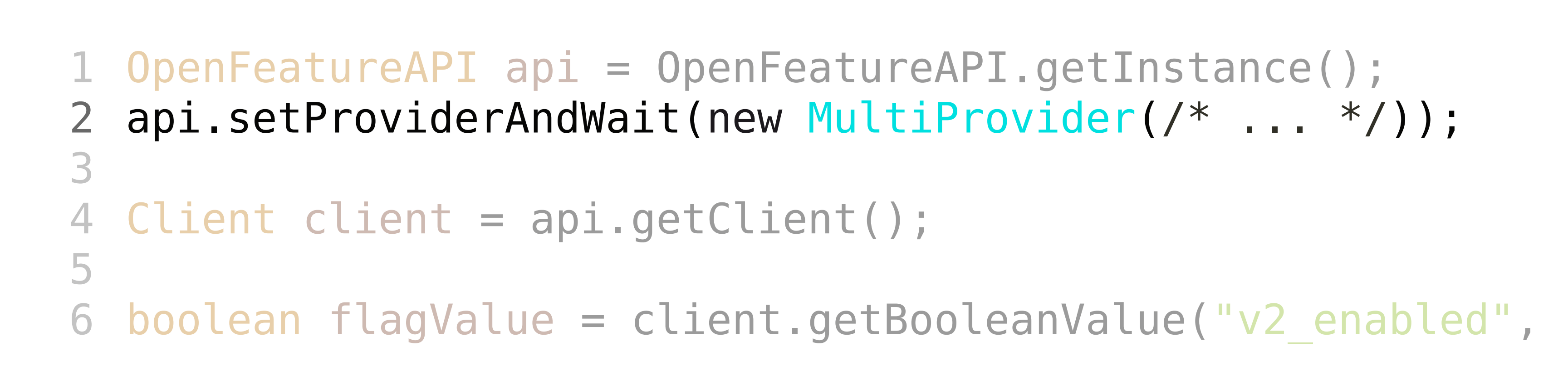
```
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new VendorProvider(/* ... */));

Client client = api.getClient();

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

EXAMPLE



```
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new LaunchDarklyProvider(/* ... */

Client client = api.getClient();

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

EXAMPLE



```java
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setProviderAndWait(new MultiProvider(/* ... */));

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

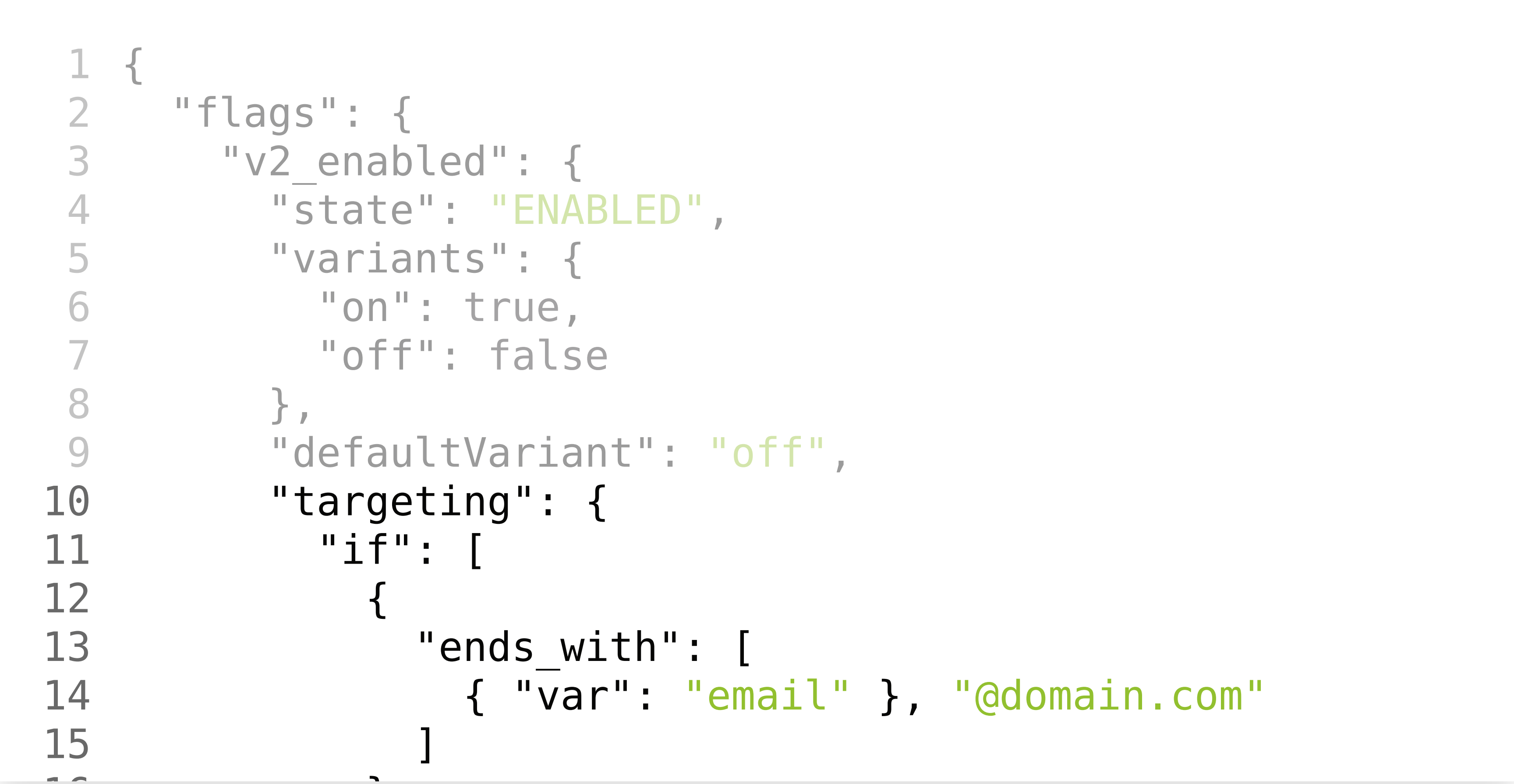
DYNAMIC EVALUATION

Changing evaluation based on rulesets.

WHY?

- A/B testing
- Quality Assurance
- Premium users
- Dogfooding
- Compliance

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

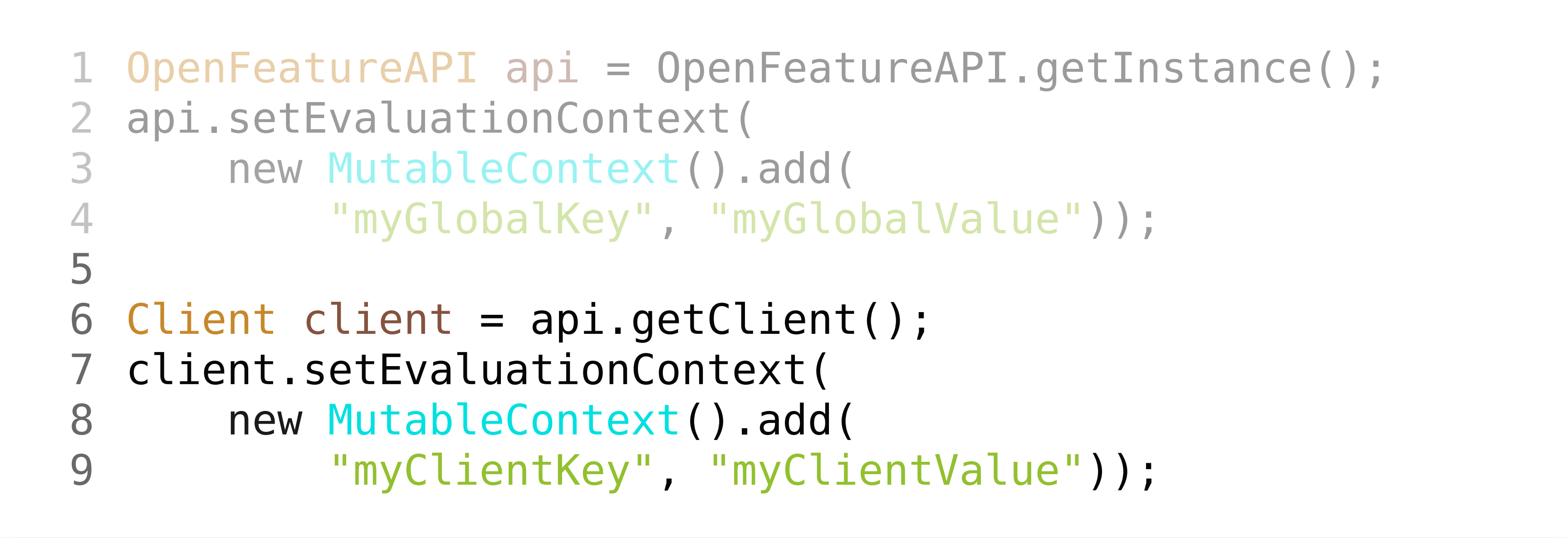
```java
Map<String, Value> requestAttrs = new HashMap<>();
requestAttrs.put("email",
    new Value(session.getAttribute("email")));
requestAttrs.put("product",
    new Value("productId"));
EvaluationContext reqCtx =
    new ImmutableContext(requestAttrs);

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

Changing evaluation based on rulesets for other use-cases?

OPERATIONAL INFORMATION

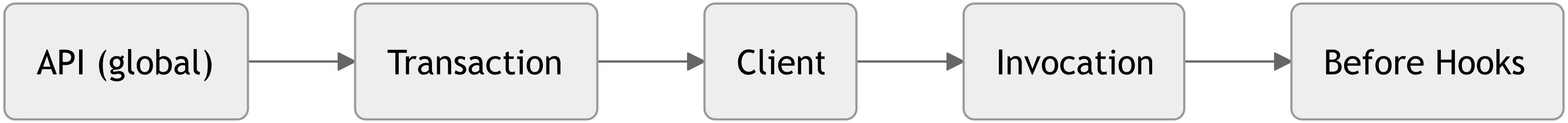
- Application information
- Hyperscalers
- Operating Systems
- Environmental information

STATIC CONTEXT



```
OpenFeatureAPI api = OpenFeatureAPI.getInstance();
api.setEvaluationContext(
    new MutableContext().add(
        "myGlobalKey", "myGlobalValue"));

Client client = api.getClient();
client.setEvaluationContext(
    new MutableContext().add(
        "myClientKey", "myClientValue"));
```

MERGE ORDER



SPECIAL TARGETING CASES

- Fractional Evaluations
- Percentage-based Evaluations

DETERMINISTIC PROBLEM

- Ensure same Result for User
- Flacky behavior

TARGETING KEY

- 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 all the time or are never evaluated.

WHY?

- Dead code
- Technical debt
- Increased complexity

DYNATRACE INSIGHTS

😯 2247 flags

775 not evaluated in last 6 months 😳

😭 1184 not evaluated in 2 years

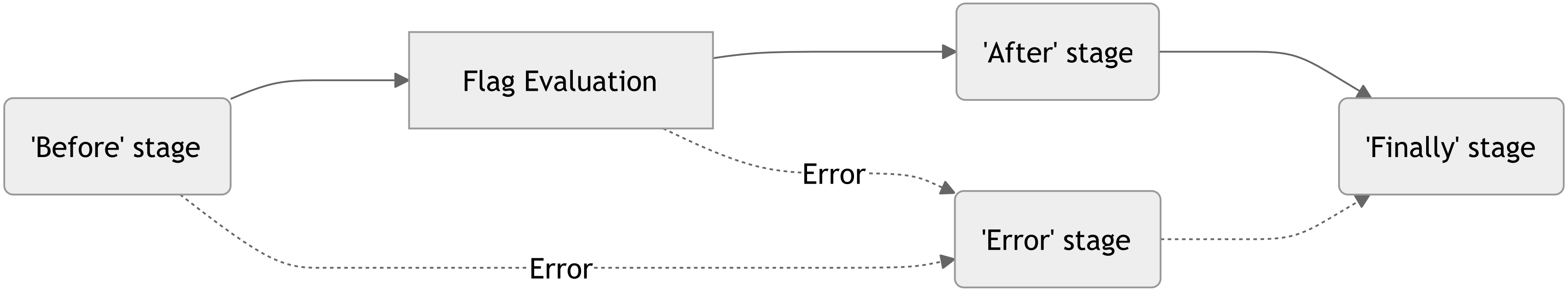
oldest from 2018 💀

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

OpenFeature

WE COVERED MANY CONCEPTS

(Evaluation API, Providers, Evaluation Context, Hooks)

OpenFeature

… THERE ARE MORE …

(Events, Tracking)

OpenFeature

... AND MORE TO COME!

We grow with your problems, share your experiences with the community!

OpenFeature

SUPPORTS *EVERYONE* WITHIN THE SOFTWARE DELIVERY LIFE-CYCLE

OpenFeature

BRINGS CONFIDENCE TO *EVERYONE!*

OpenFeature

https://openfeature.dev

GETTING STARTED

https://openfeature.dev - Official Documentation
https://flagd.dev - Cloud Native Reference Implementation
https://flagd.dev/playground - Playground for targeting rules
https://github.com/aepfli/Fun-With-Flags-Demo-Java - Java Spring Boot Demo

Q&A

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

✉: simon.schrottner@dynatrace.com
in 🐙 🦋 🐘: @aepfli(...)

ADOPTION STORY

WHO IS *EVERYONE*?

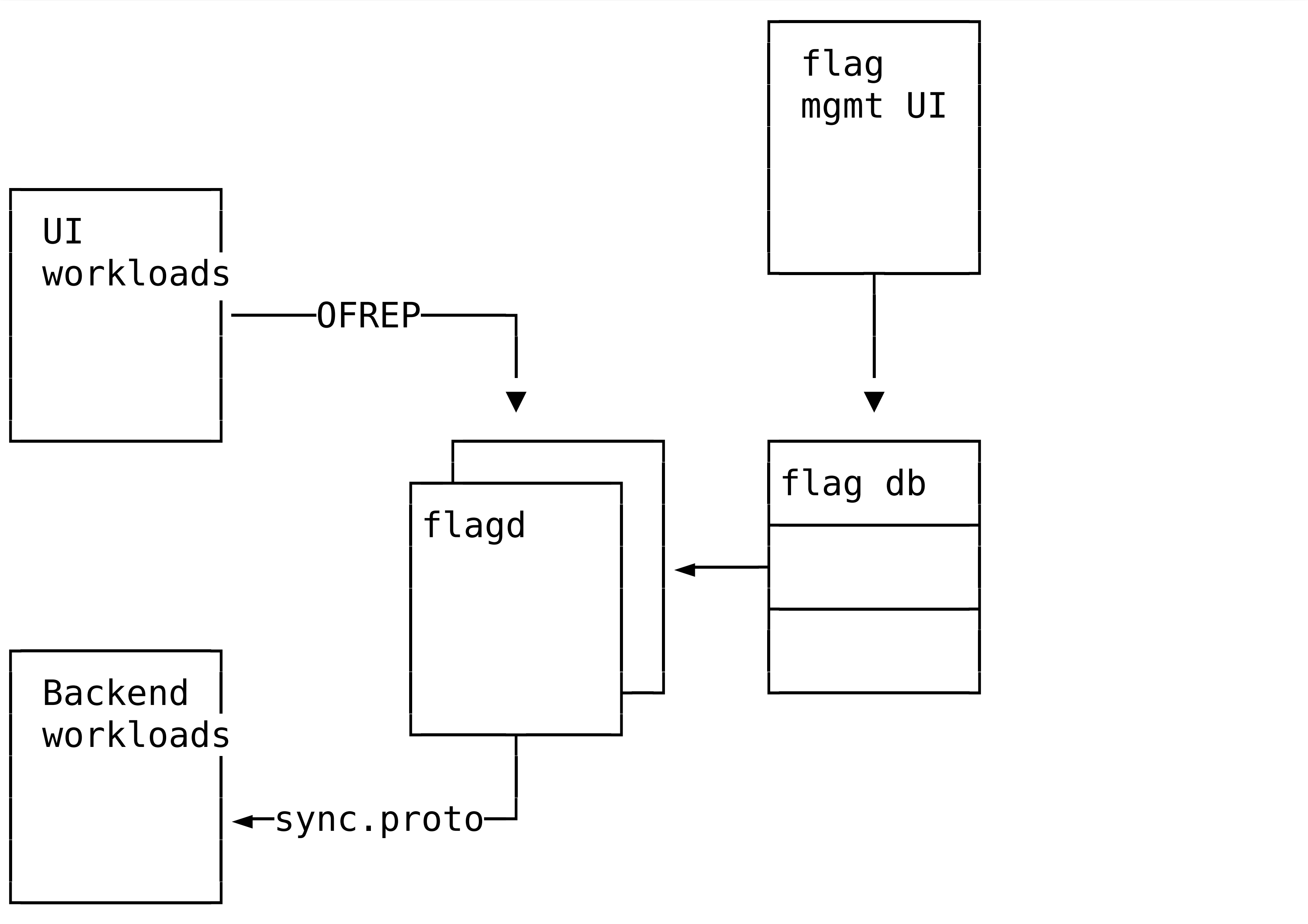
- Increasing
- OpenFeature-Compliance

ADOPTERS

- Dynatrace
- Otto GmbH
- Ebay
- Ford
- Spotify
- Google
- Octopusdeploy
- ...

INTERNAL ADOPTION

ARCHITECTURE OVERVIEW



WHAT WENT WELL

- Our initial research and efforts to understand the existing system paid off; our MVP covered the basic use-cases and did so in a more user-friendly way.
- Our UI was well-received and has been a key factor in the adoption of our solution.

WHAT WENT WELL

- We "pre-seeded" some standard context attributes for teams
- "Dynatrace-flavoured" OpenSource components
- Flexible migration through *Provider* concept

WHAT WENT WELL

- Our evaluation response times are very fast (~20ms web evaluation, <1ms service evaluation)

WHAT SURPRISED US

- Our adoption rate was initially slow but then picked up quickly…maybe too quickly.
- We were surprised by the number of flags and projects that were created in a short amount of time (80 dev teams, and 100s of flags since August).
- We didn’t anticipate some use cases…

UNEXPECTED USE-CASES

synchronized roll-outs ⌚

💼 object flags

server-side lambdas ➰

🐘 large numbers

Think about the unexpected!

OVERALL FEEDBACK

- Devs: Easy to adopt, simple, love it
- SRE: Advocating for the solution
- Bottlenecked for production changes (SRE-only)
- Processes slowly adopting

CLI

FIGHTING COMMON PITFALLS

- brittle API
- possible inconsistencies

BRITTLE API - EXAMPLE

```
client.getBooleanValue("v2_enabled", false);
// typo in keys
client.getBooleanValue("v2_enbld", false);
// different fallbacks
client.getBooleanValue("v2_enabled", true);
```

EASY FIXES

- Abstraction
- Constants
- …

…. BUT WE WANT TO BE THE WRAPPER!!!

CLI TO THE RESCUE

generating code based on a manifest*

*still experimental

MANIFEST

```
{
  "$schema": "https://raw.githubusercontent.com/open-feature/
  "flags": {
    "<name>": {
      "flagType": "<type>",
      "defaultValue": "<value>",
      "description": "<description>"
    }
  }
}
```

EXECUTION

```
openfeature generate <language>
```