Local evaluation

Copy page

Local evaluation runs evaluation logic in the SDK, saving you the overhead incurred by making a network request per user evaluation. The sub-millisecond evaluation is perfect for latency-minded systems which need to be performant at scale.

Targeting capabilities

Local evaluation happens outside of Amplitude, which means advanced targeting and identity resolution powered by Amplitude Analytics isn't supported. That said, local evaluation allows you to perform consistent bucketing with target segments, which is often enough.

Client-side local evaluation

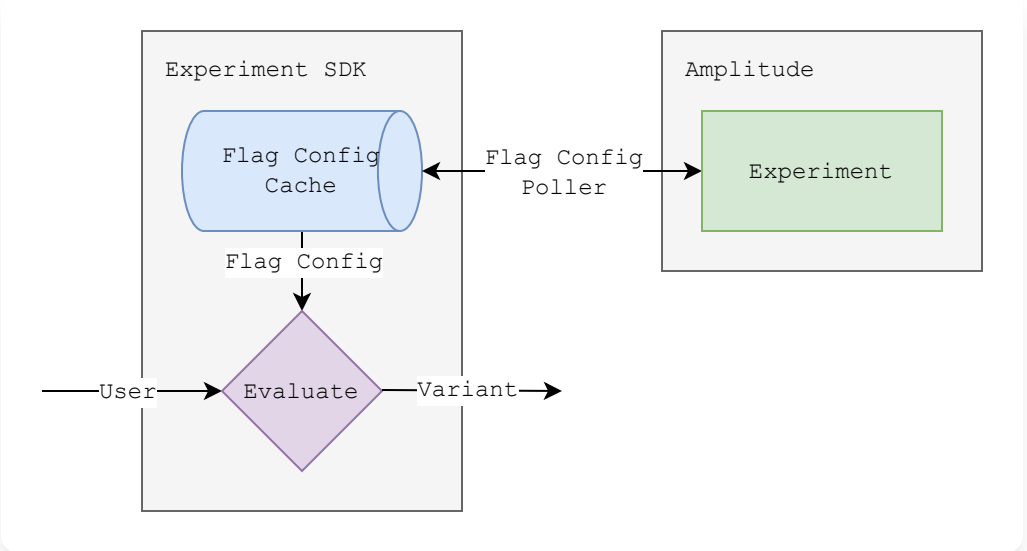
When using client-side local evaluation it is important to note that all data used in targeting is included in the flag configuration loaded on the client-side. For example, if you are targeting a specific user by their email, that email has effectively been leaked to all clients, regardless of user.

Feature	Remote Evaluation	Local Evaluation
Consistent bucketing	✅	✅
Individual inclusions	✅	✅
Targeting segments	✅	✅
Amplitude ID resolution	✅	❌
User enrichment	✅	❌
Sticky bucketing	✅	❌

SDK	Cohort Targeting	Version
Node.js	✅	`1.10.0+`
Ruby	✅	`1.5.0+`
JVM	✅	`1.4.0+`
Go	✅	`1.6.0+`
Python	✅	`1.4.0+`
PHP	❌	N/A

Implementation

Local evaluation is just evaluation--a function which takes a user and a flag as input, and outputs a variant.



Edge evaluation

The local evaluation Node.js SDK can be run in edge worker/functions which support JavaScript and a distributed store. Contact your representative or email experiment@amplitude.com to learn more.

Exposure and assignment tracking

Local evaluation SDKs track evaluations differently on the client-side vs on the server-side.

- Client-side SDKs track an **exposure event** when the user is evaluated due to a variant being accessed from the SDK.
- Server-side SDKs track an **assignment event** (if configured to do so) when a user is evaluated.

Server-side local evaluation experiments often set the Assignment event as a heuristic for Exposure.

Performance

The following results are for **a single flag evaluation**, and were collected over 10 executions of 10,000 iterations of evaluation with randomized user inputs evaluated for 1 flag configuration, selected at random out of 3 possible flag configurations.

SDK	Average	Median	Cold Start
Node.js	0.025ms	0.018ms	3ms
Go	0.098ms	0.071ms	0.7ms
JVM	0.007ms	0.005ms	6ms