Agents in AI Configs

Overview

This topic explains how to create and use agent-based AI Configs to support multi-step AI workflows with shared tools and real-time observability.

Use agent-based AI Configs to define, monitor, and adapt structured model workflows in real time. Agents help teams standardize complex logic, reuse tool definitions, and track performance across environments and projects.

Agents expand the capabilities of AI Configs by supporting multi-step, tool-driven workflows. Instead of responding to a single prompt, an agent guides a model through a structured plan and selects tools to complete each step. Agents use tools from a shared library that you define at the project level and attach to each config. This approach promotes consistency across steps, encourages reuse, and lets teams update behavior without redeploying code.

Agent support is designed for teams adopting orchestration patterns that require transparency and control. Agent configs integrate with LaunchDarkly's monitoring and targeting features, so platform teams can track usage, evaluate performance, and make informed decisions across workflows.

Capabilities and use cases

Agent configs support structured workflows that require tool selection, multi-step orchestration, and real-time monitoring. Each config defines the model's behavior through a single instructions field and can include a shared set of tools.

Coordinate multi-step large language model (LLM) workflows by guiding the model through structured plans

- Monitor inputs, tool usage, latency, cost, and satisfaction across workflows
- Define tools once and reuse them across multiple configs and projects
- Update behavior or tool access without redeploying code

Agent configs are useful when teams need to evaluate tool-driven reasoning, compare performance across agents, or standardize orchestration logic in production environments.

Set up an agent config

To create an agent-based AI Config:

1. In the left navigation, select **AI Configs**, then click **Create AI Config**.
2. Under **AI behavior type**, select **Agent-based**.
3. Enter a name that describes how your application uses generative AI. For example, “Email summarizer.”
4. (Optional) Click **Edit key** to update the config’s **key**. You use the key to reference the config in your code. LaunchDarkly auto-generates a key from the name, but you can change it.
5. (Optional) Select a maintainer.
6. Click **Create**. The configuration panel appears.
7. Select a provider and enter the model ID.
    - The model ID must match the model provider’s identifier exactly.
    - To define a role that allows or denies access to this config, use the exact config name and model ID in the role configuration.
8. Enter instructions that describe how the agent should respond to each request.
9. Save the configuration.

Retrieve agent configs using the SDK

To retrieve agent configurations using the SDK:

```
const { agents } = aiClient.agents([
  "my-agent-key",
  "backup-agent",
], context, {}, { parameters });

const primaryAgent = agents["my-agent-key"];
console.log(primaryAgent.description);
```

Agent configs include:

```
type LDAgent = {
  model?: LDModelConfig;
  description: string;
  instructions: string;
  provider?: LDProviderConfig;
  tracker: LDAIConfigTracker;
  enabled: boolean;
}
```

Use the `.agents()` method to retrieve agent configs. Agent configs are not included in `.configs()` responses. This separation ensures each agent is resolved with its own model and tool context, which supports safe composition of multi-agent workflows.

Agent config structure and behavior

Agent configs have a different structure than completion-based AI Configs. Each config includes a `mode` property, which determines which fields are required, how targeting and evaluation work, and which metrics LaunchDarkly collects.

message history or roles.

The table below summarizes the structural and behavioral differences between agent and completion modes:

Mode	Fields required for targeting and evaluation	Observability support
`completion`	`role`, `messages`	Event tracking per message
`agent`	`description`, `instructions`	Invocation metrics

Agents can use tools to support multi-step workflows. Tools are defined at the project level and attached to each config. At runtime, the agent can select tools based on the request context and the instructions you provide.

Agent configs include built-in metrics that track usage and performance. These include:

- Invocation latency
- Response cost

To measure additional signals, such as satisfaction, use external instrumentation.

The LaunchDarkly UI displays different fields depending on the selected mode:

- Completion mode includes fields for prompt history, role, and messages.
- Agent mode includes fields for description, instructions, and attached tools.

You must select the mode when you create the config. The mode cannot be changed after the config is saved.

Compatibility and rollout

Agent support is available in the TypeScript and Python SDKs.

Was this page helpful? Yes No

Previous Contexts Next