Genkit 101

Learn the fundamentals of Genkit

Agenda

- What is Genkit?
  - Open-source framework for building, deploying, and monitoring AI-powered applications
- Core Building Blocks
  - Prompts: Reusable, structured templates for all your AI model requests
  - Flows: Functions that orchestrate AI calls with your business logic
  - Plugins: Easy integrations for models (Gemini, OpenAI), databases, and more.

What is Genkit?

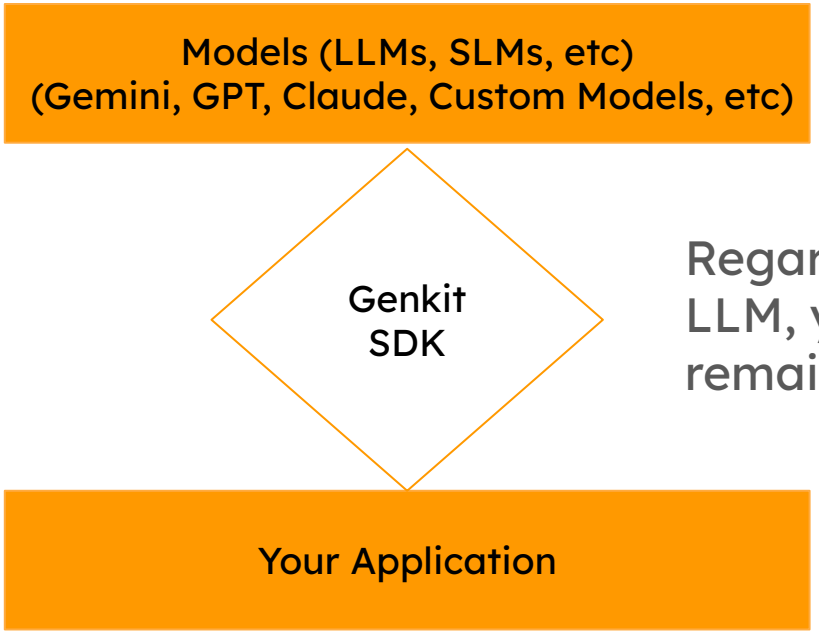
What is Genkit?

An open source framework by Google, for building AI solutions.

What is Genkit?

“A framework is a foundational blueprint of rules, tools, and best practices that provides a standardized structure for efficiently building something complex.”

What is Genkit?



Regardless of the LLM, your code remains consistent

What is Genkit?

A framework to power apps with AI.

Gemini | Imagen | OpenAI | Anthropic | xAI | DeepSeek | Ollama

```
import { genkit } from 'genkit';
import { googleAI } from '@genkit-ai/google-genai';

const ai = genkit({ plugins: [googleAI()] });

const { text } = await ai.generate({
    model: googleAI.model('gemini-2.5-flash'),
    prompt: 'Why is Genkit awesome?'
});
```

Genkit Features - Core Features

- Flows
- Prompts
- Plugins
- Tools
- Retrieval-Augmented Generation (RAG)
- Genkit Developer UI
- Built-in Observability and Traceability

Genkit Features - Developer Tools

- A CLI for command-line operations
- An optional local web app, called the Developer UI, that interfaces with your Genkit configuration for interactive testing and development

What is Genkit - Plugin?

Genkit’s capabilities are designed to be extended by plugins.

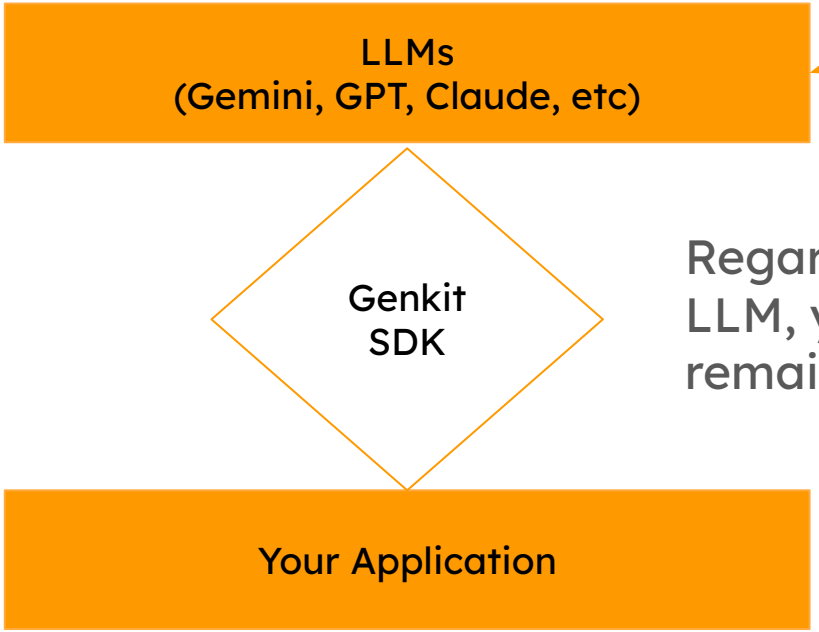
Providers for LLMs, Retrievers, Indexers, etc.

```
import { gemini20Flash, googleAI } from '@genkit-ai/googleai';
import { genkit, z } from 'genkit';

const ai = genkit({ plugins: [googleAI()], model: gemini20Flash });
```

codesnap.dev

What is Genkit - Plugin?



PLUGINS

Regardless of the LLM, your code remains consistent

What is Genkit - Plugin - Model Providers

1. Google AI Models - Gemini

```
import { genkit } from 'genkit';
import { googleAI } from '@genkit-ai/google-genai';

const ai = genkit({
  plugins: [googleAI()],
});
```

What is Genkit - Plugin - Model Providers

- OpenAI - GPT Models

```
import { genkit } from 'genkit';
import { openAI } from '@genkit-ai/compat-oai/openai';

export const ai = genkit({
  plugins: [openAI()],
});
```

Building with Genkit & Gemini

Building with Genkit & Gemini - Steps

1. Configure your model API Key

```
export GEMINI_API_KEY=<your API key>
```

Building with Genkit & Gemini - Steps

2. Initialize Genkit

```
import { googleAI } from '@genkit-ai/google-genai';
import { genkit, z } from 'genkit';

// Initialize Genkit with the Google AI plugin
const ai = genkit({
  plugins: [googleAI()],
  // default model to use
  model: googleAI.model('gemini-2.5-flash')
});
```

Building with Genkit & Gemini - Steps

3. Call Generate Command

```
async function run() {
  const response = await ai.generate(
        'Help me create a meal plan for weight loss, for 7 days'
  );

  console.log(response.text);
}

run();
```

Building with Genkit & Gemini

Flows

Building with Genkit & Gemini - Flows

A flow is a traceable function that orchestrates your entire AI-powered workflow, combining model calls with business logic into a single, deployable API endpoint.

Building with Genkit & Gemini - Flows

Orchestrate AI Workflows

- Perform steps before engaging LLMs
  - e.g. authentication, authorization, RAG, etc.

Building with Genkit & Gemini - Flows

Structured & Type-Safe

- Every flow defines a clear input and output schema (using Zod)

Building with Genkit & Gemini - Flows

Easy deployment as APIs

- Deploy Flows as HTTP endpoints

Building with Genkit & Gemini - Flows

Traceable for Easy Debugging

- Monitor deployed flows
- Debug locally with Genkit Developer UI

Building with Genkit & Gemini - Flows

Building with Genkit & Gemini - Flows

Deployable as API Endpoints. For example:

- Firebase Functions
- ExpressJS

Genkit Flows - Example

```
export const socialPostGeneratorFlow = defineFlow(
  {
    name: 'socialPostGeneratorFlow',
    // Input: Expects a simple string for the topic
    inputSchema: z.string().describe('The topic for the social media post'),

    // Output: Will return an object with the post and its character count
    outputSchema: z.object({
      postText: z.string(),
      characterCount: z.number(),
    }),
  },
  async (topic) => {
    // code to execute
  }
);
```

Genkit Flows - Example

```
export const socialPostGeneratorFlow = defineFlow(
  {
    name: 'socialPostGeneratorFlow',
    // Input: Expects a simple string for the topic
    inputSchema: z.string().describe('The topic for the social media post'),

    // Output: Will return an object with the post and its character count
    outputSchema: z.object({
      postText: z.string(),
      characterCount: z.number(),
    }),
  },
  async (topic) => {
    // Step 1: Call the AI model to generate content
    const llmResponse = await generate({
      prompt: `Generate a short, engaging social media post about "${topic}". Keep it under 280 characters.`,
      config: {
        temperature: 0.8, // Add some creativity
      },
    });

    const post = llmResponse.text();

    // Step 2: Apply some business logic
    const count = post.length;

    // Step 3: Return the final, structured object
    return {
      postText: post,
      characterCount: count,
    };
  }
);
```

Building with Genkit & Gemini

Prompts & User Inputs

Building with Genkit & Gemini - Prompts & User Inputs

- Genkit gives you two powerful ways to manage your prompts
- You can choose the best approach for your workflow.
- You can define them declaratively in simple .prompt
- or programmatically in your code for dynamic, tightly-coupled logic.

Building with Genkit & Gemini - Prompts & User Inputs

```
# /prompts/recipe.prompt
---
model: googleai/gemini-1.5-flash
input:
  schema: { type: 'string' }
output:
  format: json
  schema:
    type: 'object'
    properties:
      dishName: { type: 'string' }
      ingredients: { type: 'array', items: { type: 'string' } }
---
Generate a simple recipe for a dish that features {{input}} as the main ingredient.
```

Building with Genkit & Gemini - Prompts & User Inputs

```
import { definePrompt } from '@genkit-ai/prompt';
import { z } from 'zod';

export const recipePrompt = definePrompt({
  name: 'recipeGenerator',
  model: 'googleai/gemini-1.5-flash',
  inputSchema: z.string(),
  outputSchema: z.object({
    dishName: z.string(),
    ingredients: z.array(z.string()),
  }),
  messages: [
    {
      role: 'user',
      content: (mainIngredient) ⇒
        `Generate a simple recipe for a dish that features
            ${mainIngredient} as the main ingredient.`,
    },
  ],
});
```

Building with Genkit & Gemini - Prompts & User Inputs

```
import { generate } from
            '@genkit-ai/ai'; // 👈 Import the same function
import { sloganPrompt } from
            './prompts/slogan.prompt'; // 👈 Import the prompt from its file

// ▼▼▼ HERE IS HOW YOU "CALL" THE PROMPT ▼▼▼
async function run() {
  const response = await generate({
    prompt: sloganPrompt, // 1. Pass the imported prompt definition
    input: {              // 2. Provide the input data
      product: 'Zen Garden Kits',
      audience: 'apartment dwellers',
    },
  });

  // 3. Get the structured output
  const result = response.output();
  console.log(result?.slogan);
}

run();
```

What’s next?

- Chat Sessions & Context Propagation
- Genkit Development UI
- Retrieval Augmented Generation (RAG)
- Function/Tool Calling

Genkit Codelabs

Genkit Codelabs - Unstacked Labs

Google Codelabs

1. Automatically Deploy Generative AI Node.js Genkit Web Application from Version Control to Cloud Run | Google Codelabs
2. Automatically Deploy Generative AI Go with Genkit Web Application from Version Control to Cloud Run | Google Codelabs
3. Build gen AI features powered by your data with Genkit | Firebase Codelabs

Q&A

- The End -