



SPRING AI

THERE AND BACK AGAIN

Dan Vega · Spring Developer Advocate

Grab the code while you settle in.

ChatClient & Prompts

RIVENDELL
MCP

BREE



The demo repo

github.com/danvega/spring-ai-there-and-back-again

MORDOR
Agents



RIVENDELL
MCP

SPRING AI

THERE AND BACK AGAIN

BREE
Tools

Dan Vega

Spring Developer Advocate

THE SHIRE
ChatClient & Prompts

MORDOR
Agents



DAN VEGA

- ◆ Spring Developer Advocate
- ◆ Java Champion
- ◆ Author, *Fundamentals of Software Engineering*
- ◆ Cleveland, Ohio (*my Shire*)

Find out more at danvega.dev

SETTING THE SCOPE

WHEN WE SAY AI, WE MEAN TWO DIFFERENT THINGS

01

AI THAT WRITES THE CODE

- ◆ Assistants and agents in the IDE
- ◆ Writing, reviewing and debugging
- ◆ Reading unfamiliar codebases

A different talk

02

AI INSIDE THE CODE YOU SHIP

- ◆ Models called from your own services
- ◆ Features your users actually touch
- ◆ Your data, your rules, your uptime

This talk



**YOU DON'T NEED
TO LEAVE JAVA TO
BUILD AI APPS.**

WHY ARE WE TALKING ABOUT AI
ON THE JVM?

PYTHON EARNED ITS PLACE IN AI

THE TRAINING ERA BELONGED TO PYTHON

Python dominated AI research and model training. We are not here to debate that story.

TensorFlow

PyTorch

Scikit-Learn

Hugging Face

But the game has changed.

WHERE THE MONEY WENT

COMPUTE MOVED FROM TRAINING TO RUNNING MODELS

2/3

**of AI compute in 2026 is inference,
up from a third in 2023**

Deloitte, TMT Predictions 2026

55%

**of AI cloud spend is inference, past
training for the first time**

Gartner, 2026

74%

**of builders say most of their
workload is inference, up from 48%**

Menlo Ventures LLM market update

Running models in production is application work, and application work is where Java lives.

BUT WHAT ARE WE ACTUALLY DOING?

YOU ALREADY OWN THE TOOLS FOR THIS JOB

Training a model: a research problem

- Sourcing and cleaning a corpus
- Distributed training on a GPU cluster
- Weeks of runs, most of them discarded
- Evaluation and alignment before anyone can use it

A handful of labs do this

Most of us aren't training models. We're calling them.

Consuming a model: an integration problem

Calling the API	RestClient, provider SDKs
Credentials and config	Spring Boot configuration, Vault
Retries and timeouts	Spring Retry, Resilience4j
Observability and cost	Micrometer, OpenTelemetry

YOU'VE DONE THIS BEFORE

LLMS ARE JUST THE NEXT INTEGRATION

Databases → Message queues → Payment gateways → Cloud → **LLMs**

WHY THE JVM

JAVA IS BUILT FOR THIS

01

Enterprise-grade infrastructure

Spring, Kafka and observability tooling already running in production at scale

02

Type safety & structured output

Parse AI responses into real objects with compile-time guarantees, not loose strings

03

Battle-tested at scale

Millions of JVM applications serving billions of requests every single day

04

Your existing team

No need to hire a Python shop or replatform. Upskill the developers you already trust

BEFORE THE FRAMEWORK

NO FRAMEWORK

CALLING CLAUDE WITH CURL

```
#!/bin/bash
ANTHROPIC_API_KEY="YOUR_API_KEY_HERE"
PROMPT="Tell me an interesting fact about Java"

curl https://api.anthropic.com/v1/messages \
  -H "Content-Type: application/json" \
  -H "x-api-key: $ANTHROPIC_API_KEY" \
  -H "anthropic-version: 2023-06-01" \
  -d '{"model": "claude-haiku-4-5", "max_tokens": 1024,
      "messages": [{"role": "user", "content": "'${PROMPT}'"}]}'
```

WHAT COMES BACK

RAW JSON, AND YOUR OWN PARSING

```
{
  "id": "msg_01XFDUDYJgAACzvnptvVoYEL",
  "type": "message",
  "role": "assistant",
  "model": "claude-haiku-4-5",
  "content": [
    { "type": "text",
      "text": "Java was initially designed with interactive television in mind..." }
  ],
  "stop_reason": "end_turn",
  "usage": { "input_tokens": 14, "output_tokens": 90 }
}
```

NO FRAMEWORK

CALLING CLAUDE WITH PLAIN JAVA

```
var body = """
  { "model": "claude-haiku-4-5",
    "max_tokens": 1024,
    "messages": [{ "role": "user",
                    "content": "Tell me an interesting fact about Java" }] }
  """;

HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create("https://api.anthropic.com/v1/messages"))
    .header("Content-Type", "application/json")
    .header("x-api-key", apiKey)
    .header("anthropic-version", "2023-06-01")
    .POST(HttpRequest.BodyPublishers.ofString(body))
    .build();

var client = HttpClient.newHttpClient();
var response = client.send(request, HttpResponse.BodyHandlers.ofString());
System.out.println(response.body());
```

“

*Spring AI provides so much more than a facility for making
REST API calls.*

WHAT ARE THE CHALLENGES?

EVERYTHING YOU WOULD OTHERWISE BUILD YOURSELF

01

Talking to models

- ◆ Model abstraction and portability
- ◆ Unified API across providers
- ◆ Multimodality: text, image, audio
- ◆ Streaming and async responses

02

Running it in production

- ◆ Error handling and retry logic
- ◆ Observability and monitoring
- ◆ Token counts and cost control
- ◆ Evaluation and testing
- ◆ Security and compliance

03

Going past one call

- ◆ Structured output parsing
- ◆ Prompt template management
- ◆ Memory and conversation state
- ◆ RAG, embeddings, vector stores
- ◆ Tool calling and MCP

WHAT IS A TOKEN?

YOU PAY FOR TOKENS, NOT WORDS

100 tokens

is about 75 words. A token is roughly three quarters of a word.

Remember the usage block on the raw JSON slide? input_tokens and output_tokens are the bill.

TWO WAYS TO PAY

CHATGPT PRICING IS NOT API PRICING

\$/month

A subscription. Flat and predictable. ChatGPT Plus or Claude Pro is about \$20 a month for one person in a chat app.

What you pay when you chat

\$/token

The API. Metered, with no ceiling. Every request your app sends is billed by the token, in and out.

What your Spring Boot app pays

Your app uses the API. Tokens are your bill.

PRICING

WHAT THE MODELS COST, PER MILLION TOKENS

MODEL	PROVIDER	INPUT	OUTPUT
Claude Fable 5.1	Anthropic	\$10.00	\$50.00
Claude Opus 5.5	Anthropic	\$4.00	\$20.00
Claude Sonnet 5.5	Anthropic	\$2.00	\$10.00
Claude Haiku 5.5	Anthropic	\$0.10	\$0.50
Claude Haiku 4.5 today's demo	Anthropic	\$1.00	\$5.00
GPT-6 Astra	OpenAI	\$10.00	\$50.00
GPT-6.1 Sol	OpenAI	\$2.00	\$10.00
GPT-6 Luna	OpenAI	\$0.10	\$0.50
Gemini 4 Argon	Google · intro price	\$2.00	\$10.00
Gemini 3.8 Flash	Google · intro price	\$0.75	\$3.75
Gemini 3.5 Flash-Lite	Google	\$0.30	\$2.50

Standard rates for short prompts, checked October 13, 2026. Long prompts, batch and cached input price differently, and intro prices end soon. Check the provider's page before you budget.

CONTEXT WINDOW

ONE BUDGET, SPENT ON EVERY CALL

The model has no memory between calls.

Your instructions,
every call

Every tool you
register, every call

System
prompt

Conversation history

Tool
definitions

Retrieved documents

Room for
the answer

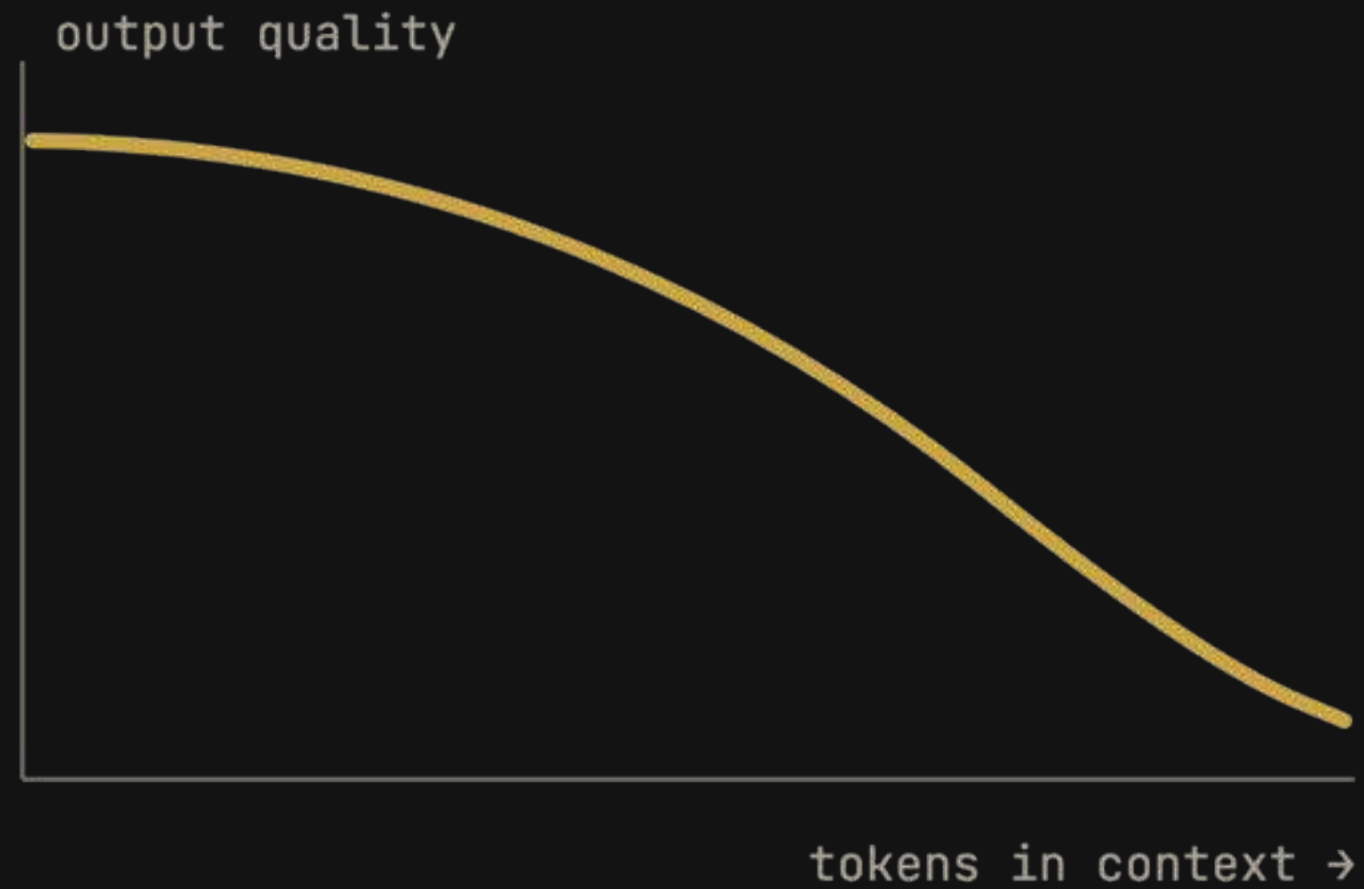
Grows with every turn you keep

Whatever you pulled in
for this question

Run out of room and the call fails. Fill it carelessly and you pay for it on every turn.

THE CATCH

CONTEXT ROT



More context is not better context.

Chroma Research, July 2025



RIVENDELL
MCP

BREE
Tools

THE SHIRE
ChatClient & Prompts

MORDOR
Agents

◆ **THE ROAD AHEAD** ◆

THE ROAD AHEAD

WHO DECIDES WHAT HAPPENS NEXT?

THE SHIRE

BREE

RIVENDELL

MORDOR



ChatClient

Tools

MCP

Agents

Your code decides

The model decides

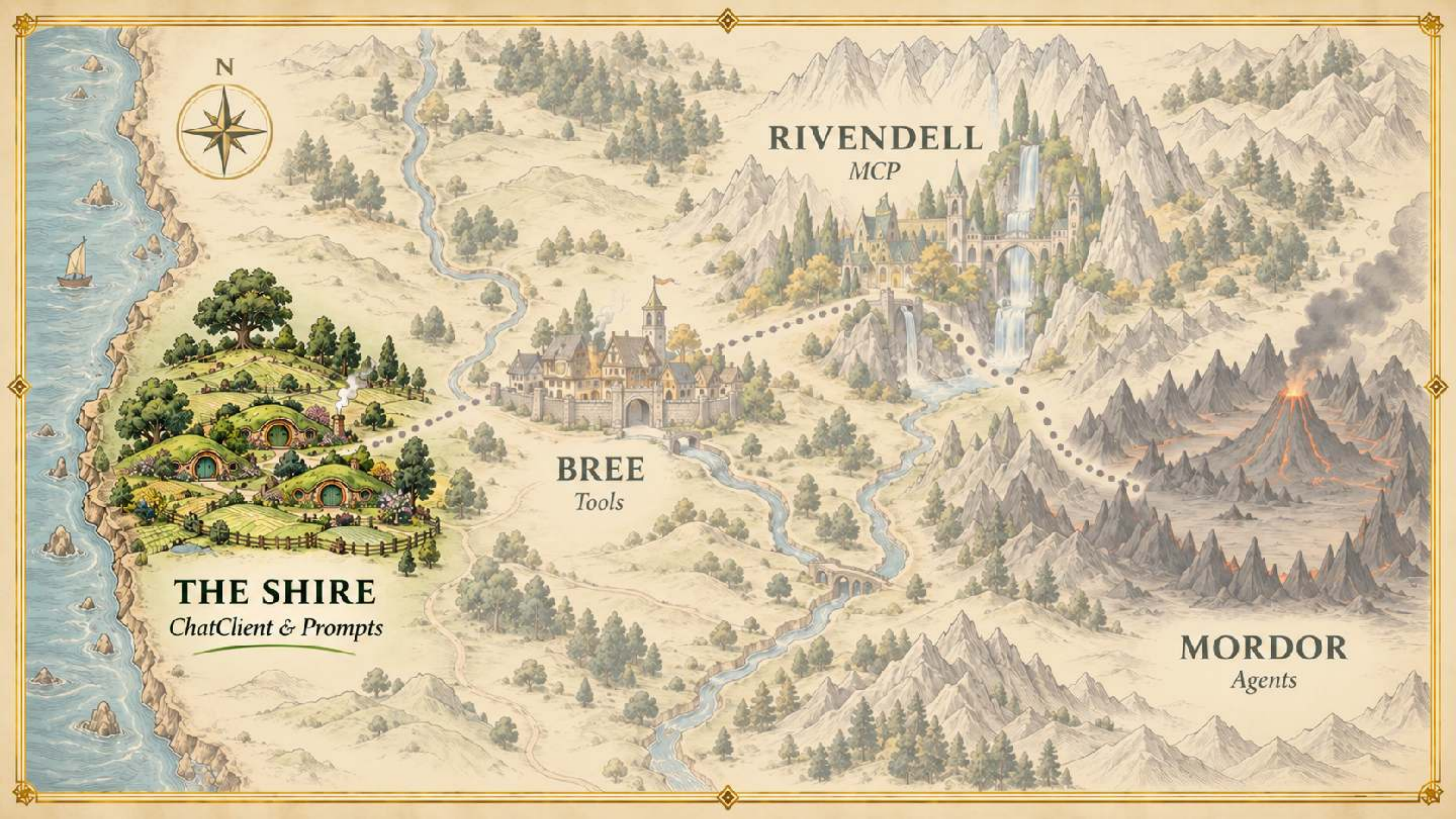


RIVENDELL
MCP

BREE
Tools

THE SHIRE
ChatClient & Prompts

MORDOR
Agents

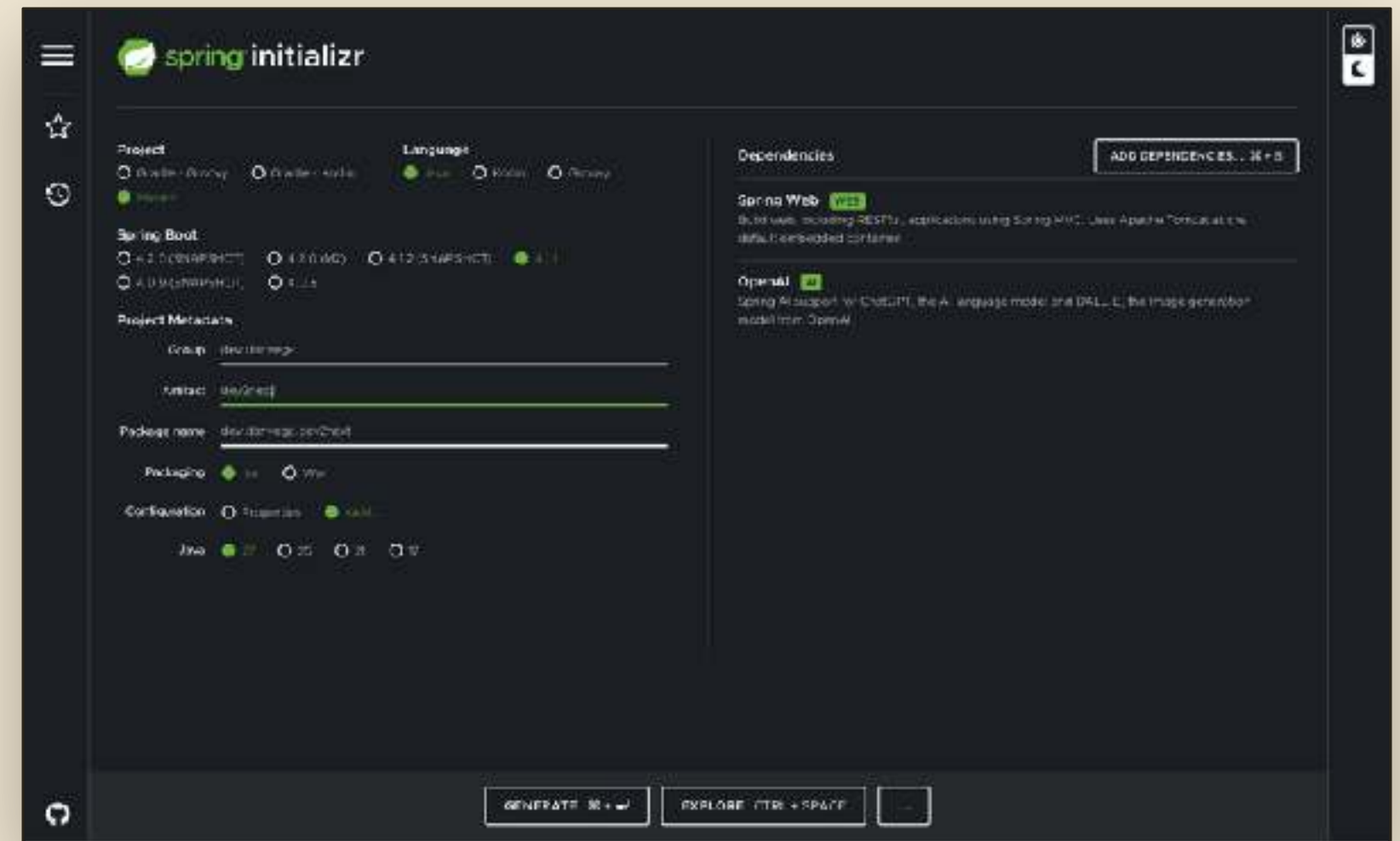


— ◆ —

*“It's a dangerous business,
going out your door.”*

BILBO BAGGINS

Step one: start.spring.io



HOW SPRING HANDS YOU A CHATCLIENT

```
pom.xml
<dependency>
  <groupId>org.springframework.ai</groupId>
  <artifactId>spring-ai-starter-model-anthropic</artifactId>
</dependency>
```

```
application.yml
spring:
  ai:
    anthropic:
      api-key: ${SPRING_AI_ANTHROPIC_API_KEY}
    chat:
      model: claude-haiku-4-5
```

```
ShireController.java
public ShireController(ChatClient.Builder builder) {
  this.chatClient = builder.build();
}
```

1 Add one starter.

2 Set your key and model.

At startup, Spring sees both and builds an AnthropicChatModel and a ChatClient.Builder for you.

3 Ask for it in your constructor.

*No **new**. No setup code. That's auto-configuration.*

MAKE CAMP · CHAPTER ONE

CAMP 1: THE SHIRE

WHAT WE'LL BUILD

- ◆ A new project from start.spring.io
- ◆ Your first prompt, then stream it
- ◆ A messy email into a Java record

```
git checkout 01-shire
```



WHAT WE JUST BUILT · THE SHIRE

YOUR FIRST PROMPT

ShireController.java

```
String answer = chatClient.prompt()  
    .system("You are a hobbit who has never left the Shire.")  
    .user("Plan a perfect day in the Shire, starting with second breakfast.")  
    .call()  
    .content();
```

WHAT WE JUST BUILT · THE SHIRE

STREAMING

ShireController.java

Wait for the whole answer

```
.call().content() // String
```

Watch it arrive

```
.stream().content() // Flux<String>
```

Same prompt. One word changes.

WHAT WE JUST BUILT · THE SHIRE

STRUCTURED OUTPUT

Journey.java · ShireController.java

```
record Journey(String traveler, LocalDate departure,  
               List<Stop> stops) {}
```

```
record Stop(String place, LocalDate arrival,  
            String lodging, String confirmationCode) {}
```

```
Journey journey = chatClient.prompt()  
    .system("Extract the journey details from this email.")  
    .user(email)  
    .call()  
    .entity(Journey.class);
```

**MOST AI FEATURES NEVER NEED TO
LEAVE THE SHIRE.**



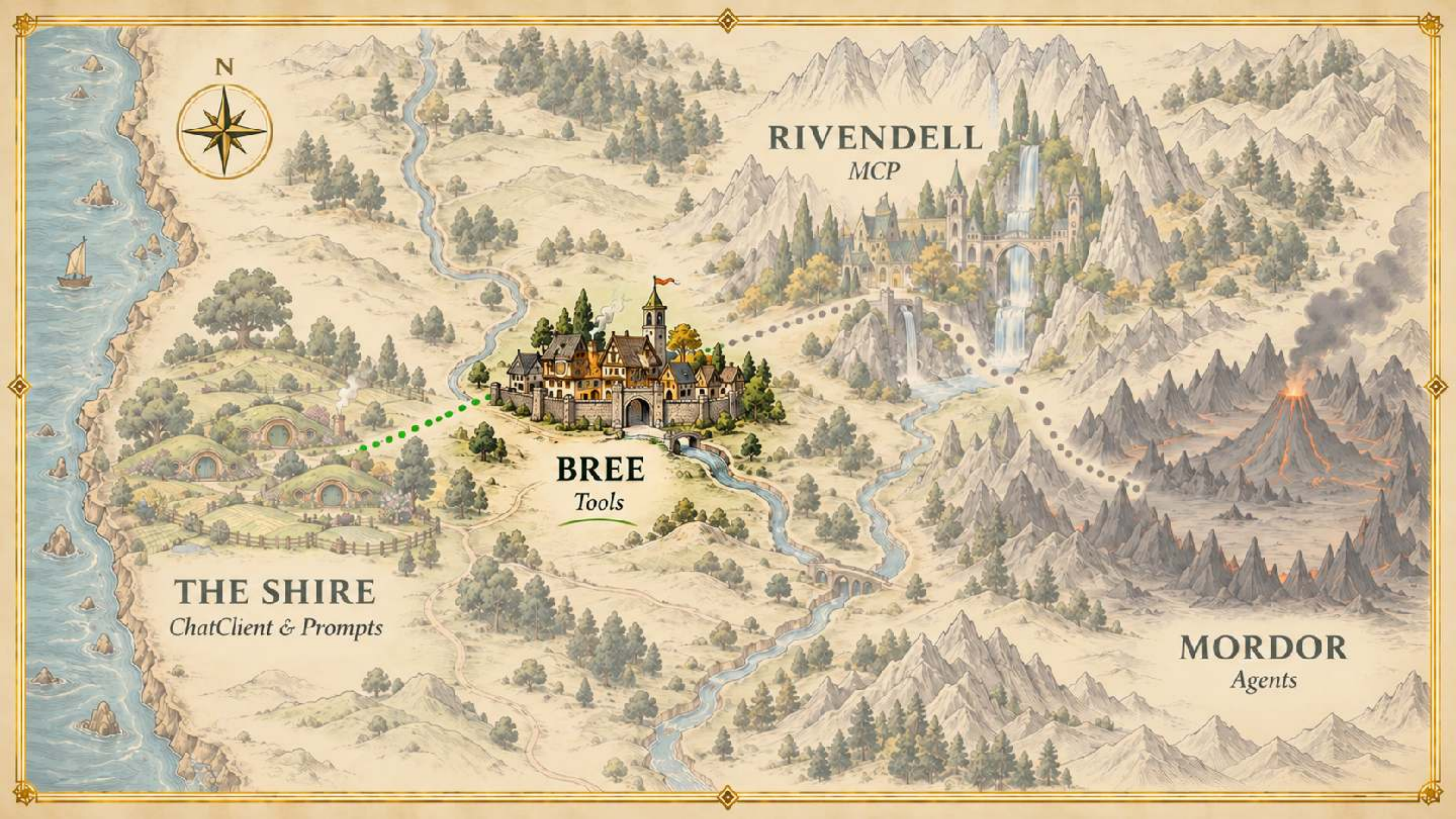


RIVENDELL
MCP

BREE
Tools

THE SHIRE
ChatClient & Prompts

MORDOR
Agents



HOW A TOOL CALL WORKS

Your code runs on this side only.



YOU

What should I pack
for Rivendell?

YOUR SPRING BOOT APP

Spring AI manages the loop

- 1 Sends your question and the tool list to Claude
- 3 Runs `getWeather("Rivendell")` in your app
- 4 Sends back: *Cold and misty. Rain by evening.*
- 6 Hands the final answer back to you

Your @Tool methods

```
getCurrentDate()
```

```
getWeather(place)
```

CLAUDE

The model

- 2 `getWeather("Rivendell")`
Not an answer. A request.
- 5 Pack a warm cloak and
waterproof boots.

*The model never runs your code. It only asks.
Spring AI runs this loop for you.*

MAKE CAMP · CHAPTER TWO

CAMP 2: BREE

WHAT WE'LL BUILD

- ◆ A tool that knows today's date
- ◆ A tool that checks the weather
- ◆ Watch the model choose

```
git checkout 02-bree
```



WHAT WE JUST BUILT · BREE

DEFINING A TOOL

TravelTools.java

```
@Tool(description = "Get today's date")
String getCurrentDate() {
    return LocalDate.now().toString();
}

@Tool(description = "Get the current weather for a place in Middle-earth")
String getWeather(@ToolParam(description = "The place, for example Bree or Rivendell")
                  String place) {
    // Sunny in the Shire, misty in Rivendell, ash in Mordor
}
```

BreeController.java

```
chatClient.prompt()
    .user(prompt)
    .tools(new TravelTools())
    .call()
    .content();
```

WHAT DO YOU TRUST THE MODEL TO CARRY?

READ TOOLS

Look things up.

Weather, search, database queries

SAFE TO CARRY

WRITE TOOLS

Change things.

Send email, update records, place orders

CARRY WITH CARE



RIVENDELL

MCP

BREE

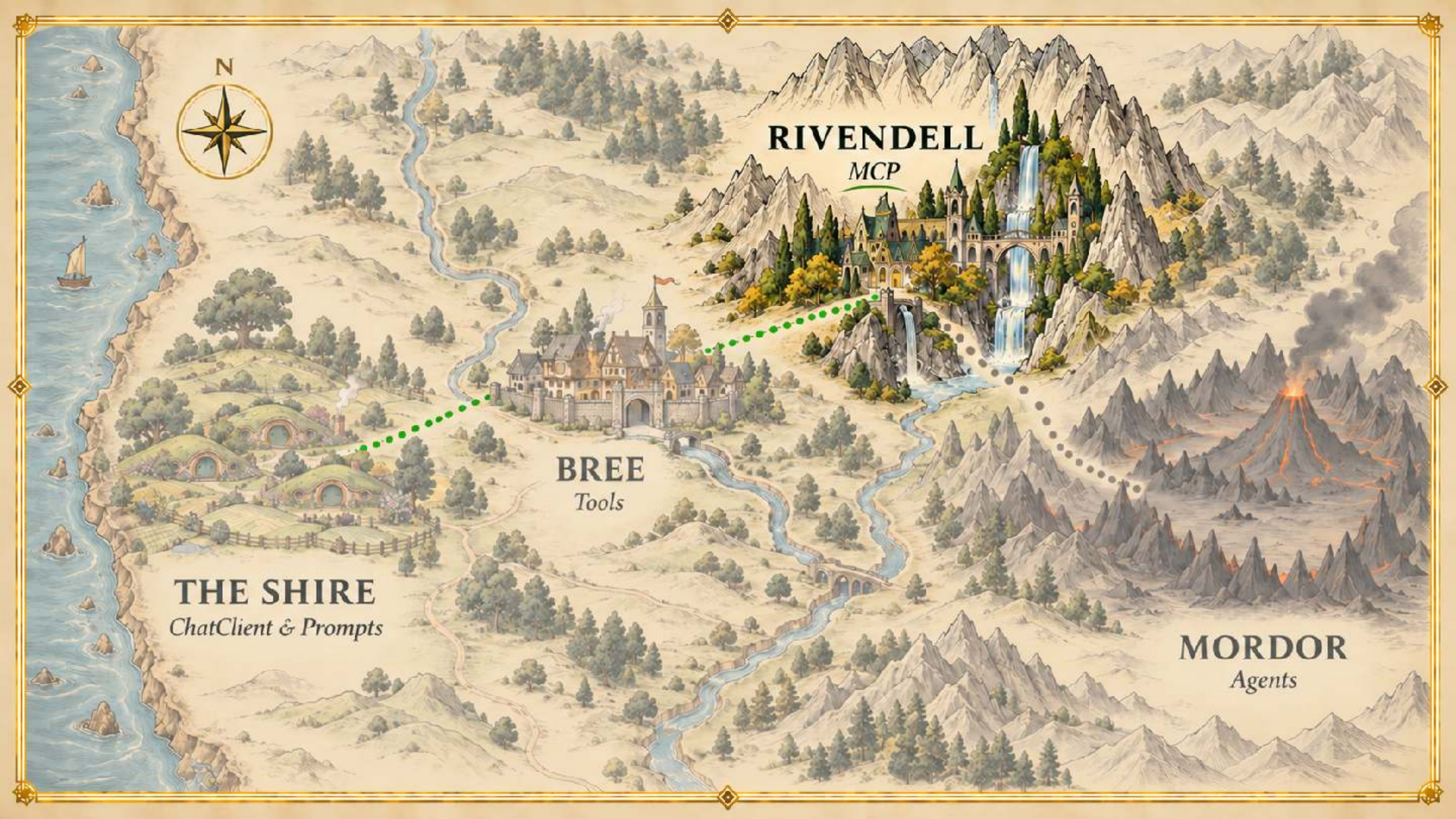
Tools

THE SHIRE

ChatClient & Prompts

MORDOR

Agents





ONE PROTOCOL TO CONNECT THEM ALL.



`modelcontextprotocol.io`

MCP BY THE NUMBERS

EVERYONE SPEAKS MCP NOW

400M+

SDK downloads a month, up 4x this year

Anthropic, July 2026

DECEMBER 2025

Anthropic donates MCP to the

AGENTIC AI FOUNDATION

A directed fund under the Linux Foundation. Neutral governance.



MADE IN SPRING

JAVA'S MCP SDK CAME FROM THE SPRING TEAM

Nov 2024

Spring AI starts an experimental MCP project for Java

Feb 2025

It becomes the official MCP Java SDK, built with Anthropic

Spring blog, February 2025

Today

Spring AI 2.0 builds on it. A starter and a few lines of YAML.

When you use MCP from Java, you're using code the Spring team wrote.

TWO WAYS TO CONNECT

LOCAL OR REMOTE, SAME PROTOCOL

stdio

The server runs on your machine. Your app starts it as a process and talks to it over standard input and output.

In our demo, the app starts the filesystem server

Streamable HTTP

The server runs anywhere on the network. Clients connect to it with a URL.

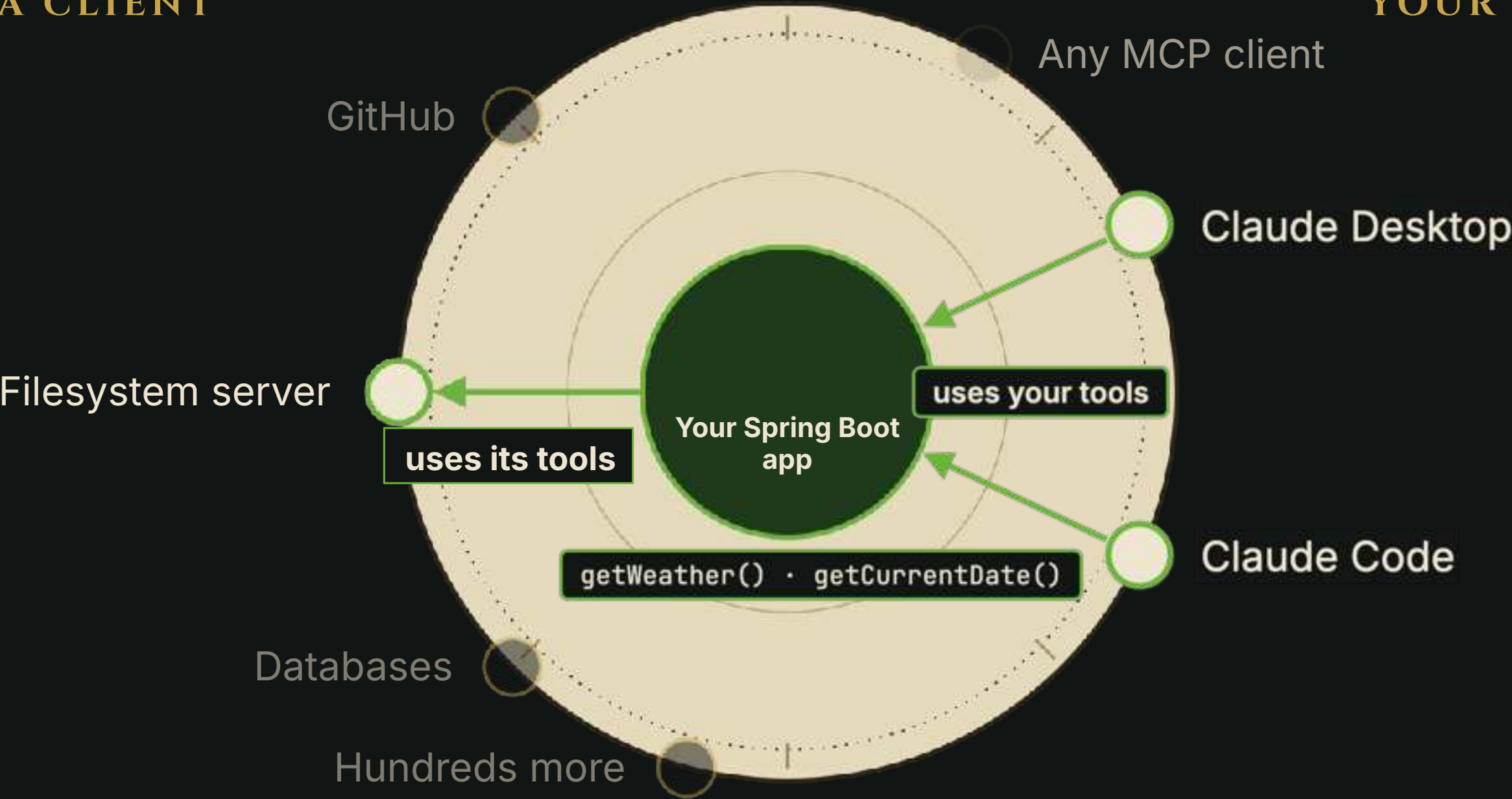
In our demo, the MCP Inspector connects to /mcp

Same tools, same messages. Only the connection changes.

CLIENT AND SERVER

YOUR APP AS A CLIENT

YOUR APP AS A SERVER



Your app uses tools other people built.
Other AI apps use the tools you built.

MAKE CAMP · CHAPTER THREE

CAMP 3: RIVENDELL

WHAT WE'LL BUILD

- ◆ Use an existing MCP server
- ◆ Expose our tools as an MCP server
- ◆ Call them from Claude

```
git checkout 03-rivendell
```



WHAT WE JUST BUILT · RIVENDELL

YOUR APP AS A CLIENT

application.yaml

```
spring:
  ai:
    mcp:
      client:
        stdio:
          connections:
            filesystem:
              command: npx
              args: ["-y", "@modelcontextprotocol/server-filesystem" , "./journeys"]
```

it can only touch this
one folder

RivendellController.java

```
chatClient.prompt()
    .user(prompt)
    .tools(new TravelTools(), mcpTools)
    .call()
    .content();
```

WHAT WE JUST BUILT · RIVENDELL

YOUR APP AS A SERVER

```
application.yaml
spring:
  ai:
    mcp:
      server:
        name: journey
        protocol: STREAMABLE #required: without it you get the older SSE transport
```

```
McpServerConfig.java
@Bean
ToolCallbackProvider travelTools() {
    return MethodToolCallbackProvider.builder()
        .toolObjects(new TravelTools())
        .build();
}
```

Building an MCP server from scratch? Use @McpTool.

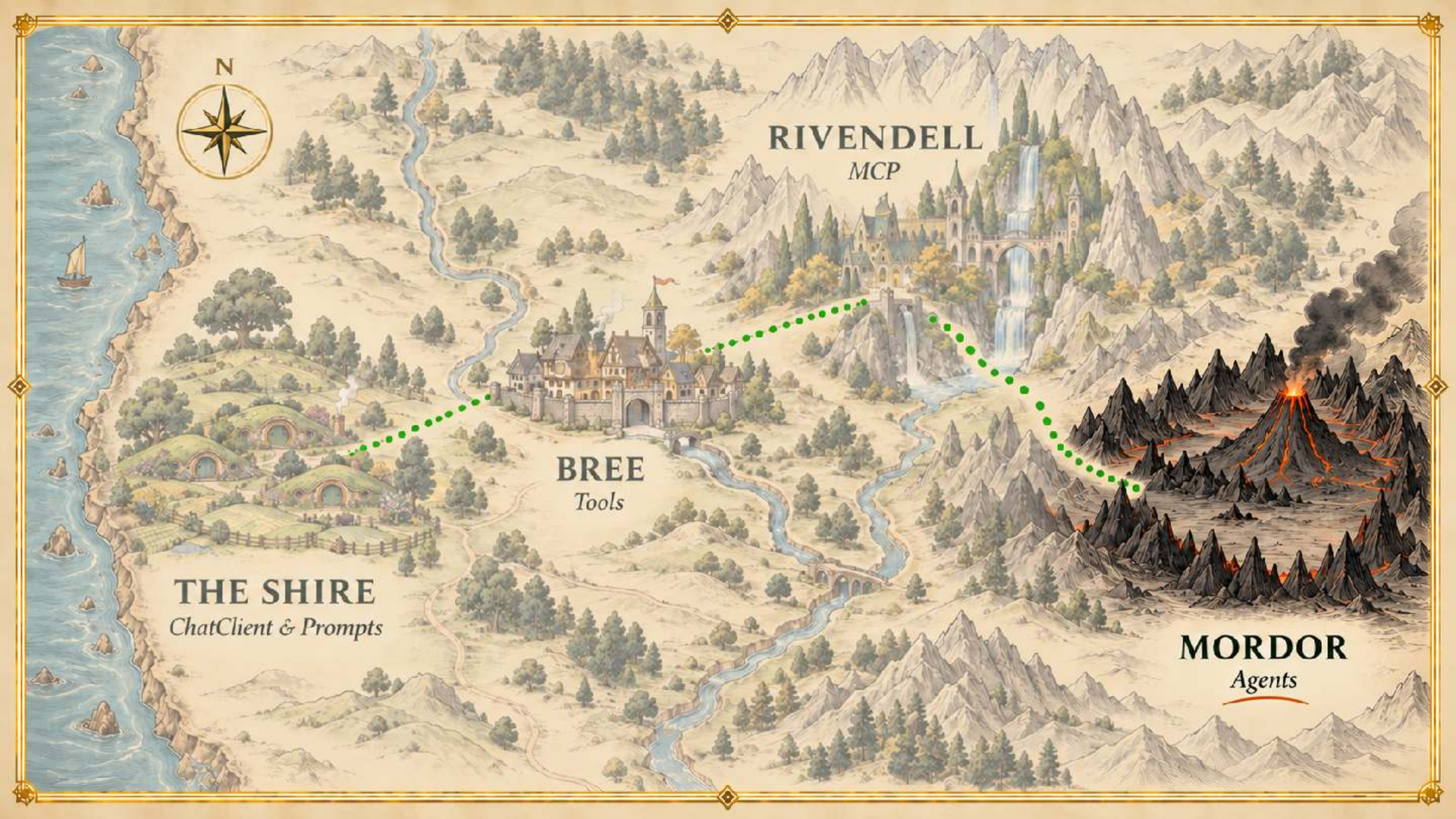


RIVENDELL
MCP

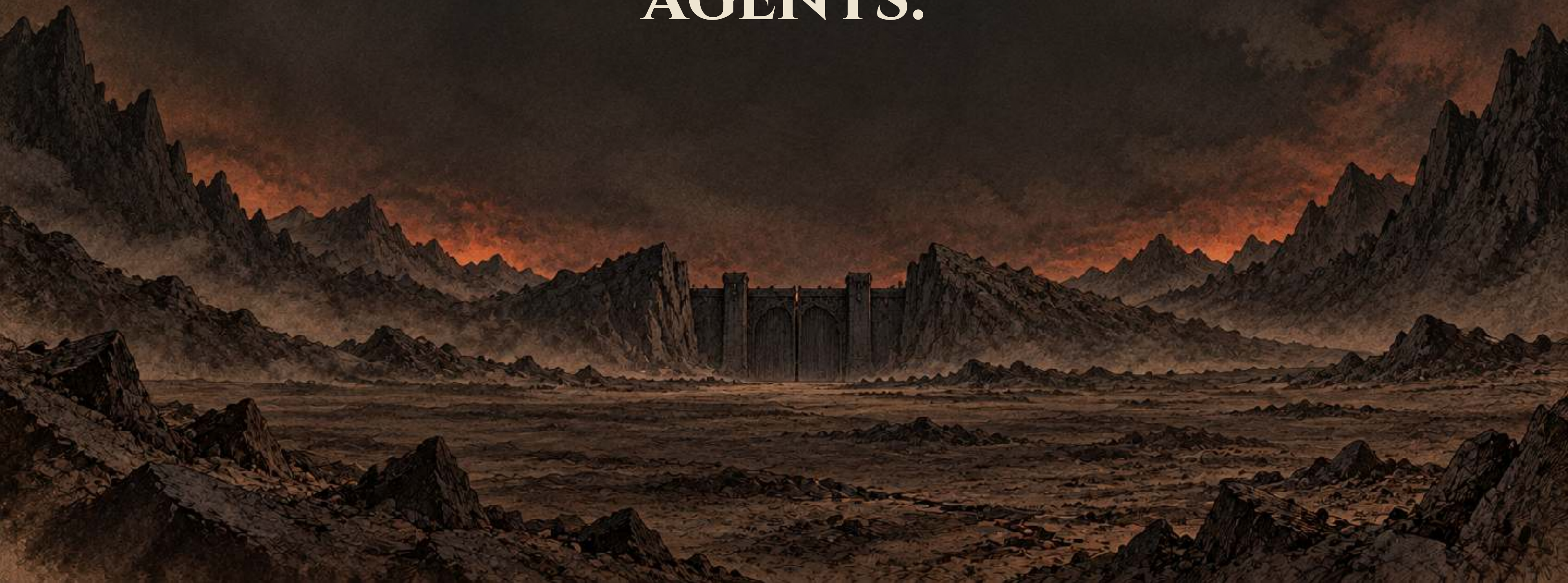
BREE
Tools

THE SHIRE
ChatClient & Prompts

MORDOR
Agents



ONE DOES NOT SIMPLY WALK INTO
AGENTS.



WHY DIDN'T THEY JUST FLY THE EAGLES?

Planning is the hard part.



HOW AN AGENT WORKS

GOAL Plan the fastest safe route from Rivendell to Mount Doom.

ACT · what it called	→ CHECK · what came back	→ PLAN · what it decides next
getRoutes(Rivendell)	Caradhras 2 days, Moria 4 days	Caradhras is faster. Check the weather first.
getWeather(Caradhras)	Blizzard. Pass blocked.	Re-plan: go through Moria.
getRoutes(Moria)	Lothlorien, 2 days. "Something stirs in the deep."	It's open. Keep going.
... a few more routes ...		
requestEagles(Mount Doom)	Declined. Walk.	Keep walking.
getRoutes(Dead Marshes)	Black Gate closed. Cirith Ungol open.	Take Cirith Ungol.
write_file(route-to-mordor.md)	Saved.	Done. 22 days.

An agent is the tool loop from Bree, given a goal instead of a question.

MAKE CAMP · CHAPTER FOUR

CAMP 4: MORDOR

WHAT WE'LL BUILD

- ◆ Give it a goal, not a question
- ◆ Watch it plan and act
- ◆ See where it goes wrong

```
git checkout 04-mordor
```



WHAT WE JUST BUILT · MORDOR

WHAT WE JUST BUILT

```
MordorController.java
```

```
builder.defaultSystem("""  
    You are a route planner for Middle-earth. Reach the goal using your tools.  
    Explore the routes one step at a time. Never invent a route.  
    Check the weather before crossing any mountain pass.  
    """)
```

```
chatClient.prompt()  
    .user(goal)  
    .tools(new TravelTools(), new RouteTools(), mcpTools)  
    .call()  
    .content();
```

Same loop as Bree. A goal, more tools, and instructions. The model did the planning.

BEYOND MORDOR

GOING FURTHER: EMBABEL

An agent framework for the JVM, from the creator of Spring

- ◆ Plans with an algorithm (GOAP, from game AI), not the LLM
- ◆ Re-plans after every step
- ◆ Typed Java: `@Agent`, `@Action`, `@AchievesGoal`

“Spring AI is the Servlet API. Embabel is Spring MVC.”

the Embabel README

github.com/embabel/embabel-agent



CHAPTER FOUR · MORDOR

DON'T GO FURTHER THAN YOU NEED TO.

WORKFLOW: THE ROAD

You choose the path.



Predictable. ◆ Cheaper. ◆ Easy to test.

AGENT: THE WILDERNESS

The model chooses the path.



Flexible. ◆ Slower and costlier. ◆ Might walk into Moria.

Most features are roads. Save agents for the wilderness.



RIVENDELL
MCP

BREE
Tools

THE SHIRE
ChatClient & Prompts

MORDOR
Agents

◆ **THERE AND BACK AGAIN** ◆

THERE AND BACK AGAIN

WHICH STOP DO YOU NEED?



THE SHIRE

ChatClient & Prompts

Text goes in and text or data comes out. Summaries, extraction, classification.



BREE

Tools

The answer needs live data, or the app needs to act in your system.



RIVENDELL

MCP

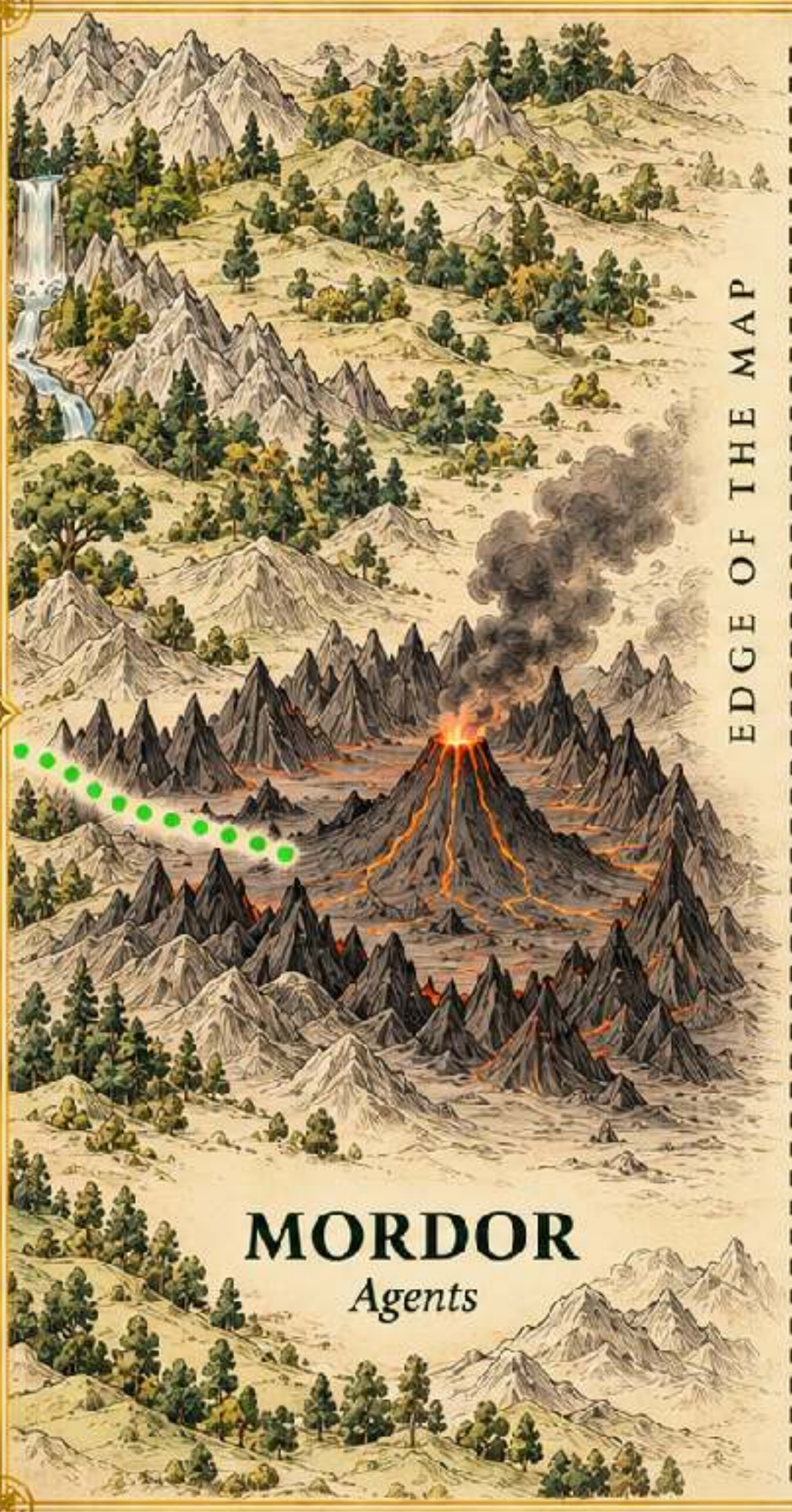
The tools already exist somewhere else, or other AI apps should use yours.



MORDOR

Agents

You can't know the steps ahead of time, and the task is worth the cost and risk.



MORDOR
Agents

EDGE OF THE MAP



MAPS FOR THE NEXT JOURNEY

RAG

Vector Stores

Chat Memory

Evals

Observability

COMING IN SPRING AI 2.1

THE ROAD GOES EVER ON

THE SHIRE **Session management.** Conversation history is coming to Spring AI's core.

BREE **Message parts.** Replies keep their order: reasoning, tool calls, text, and images.

RIVENDELL The newest MCP spec (2026-07-28) is in progress.

MORDOR **Spring AI Agents.** A new experimental project, planned for November 2026.

2.1.0-M1 is out now. Milestone APIs may still change.

THERE AND BACK AGAIN

YOUR NEXT STEPS

- 1 Spring AI reference
docs.spring.io/spring-ai/reference
- 2 Spring AI Community
github.com/spring-ai-community
Community projects that extend Spring AI
- 3 The demo repo
github.com/danvega/spring-ai-there-and-back-again



THANK YOU

Dan Vega

danvega.dev

THE SHIRE
ChatClient & Prompts

BREE
Tools

RIVENDELL
MCP



The demo repo
github.com/danvega/spring-ai-there-and-back-again

MORDOR
Agents