Build your own Developer Advocate with DeepAgents and Elasticsearch

github.com/justincastilla/DevRel-DeepAgent

Justin Castilla

Elasticsearch

Talk Overview

🎓 What You'll Learn

- **LangChain's DeepAgents Framework -** Hierarchical multi-agent orchestration
- **Elastic Agent Builder -** Remote agent tooling for intelligent data operations
- **Live Demo -** DevRel Research Agent walkthrough and demo

PART 1

LangChain's DeepAgents Framework

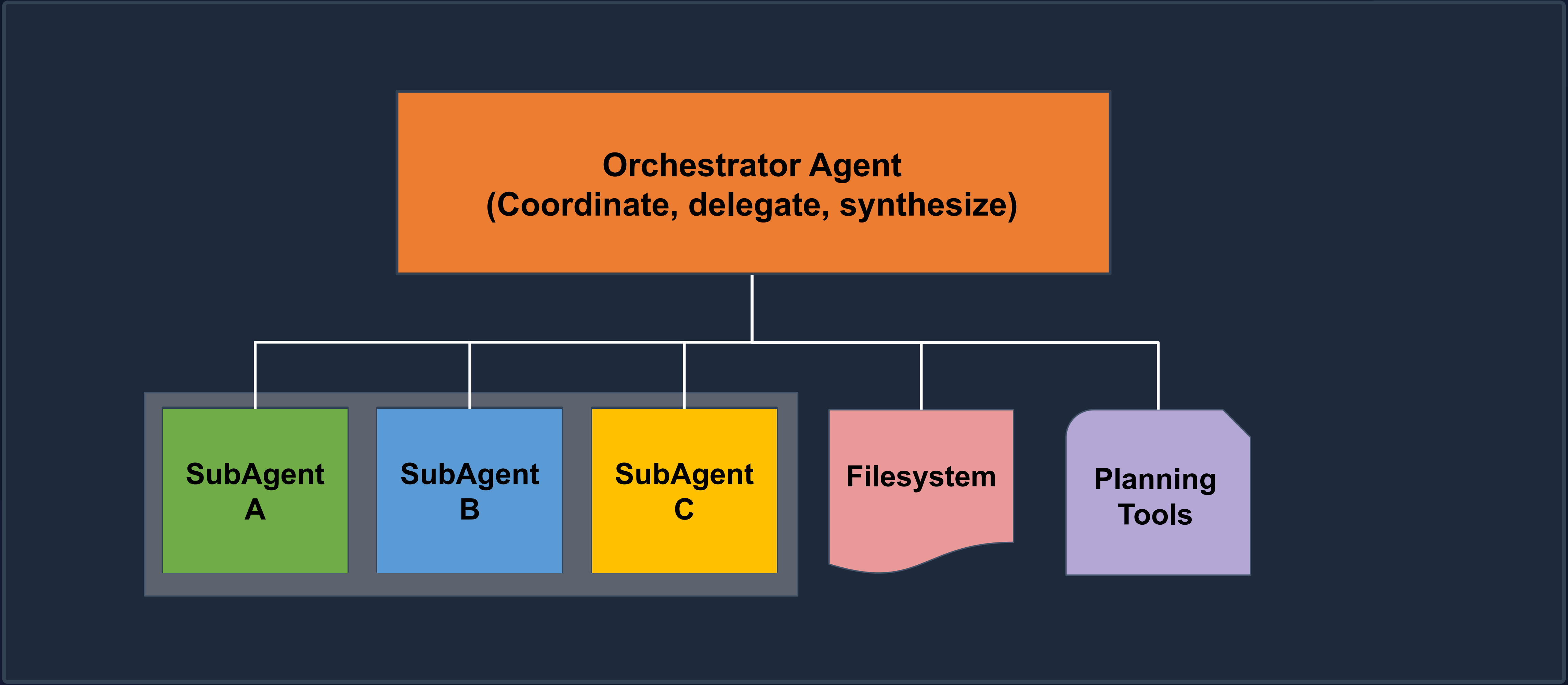
github.com/langchain-ai/deepagents

DeepAgents - The Single Agent Problem

Traditional single-agent systems struggle with:

- Complex multi-step workflows
- Specialized domain knowledge
- Parallel task execution
- Context management at scale

DeepAgents - The Orchestrator Solution



Core Concepts — Orchestrator Agent

The main coordinator

- Receives user requests
- Decides which subagents to invoke
- Runs subagents in parallel when possible
- Synthesizes results into a unified response

Core Concepts — SubAgents and Tools

SubAgents

- **Specialized workers**
  - Focus on a single domain
  - May call other subagents
  - Disappear once their task is done

Tools

- **Atomic operations**
  - Connect to external APIs
  - Perform calculations
  - Store/retrieve data

Agent Creation Pattern

```
# Create the main orchestrator
agent = create_deep_agent(
    model="anthropic:claude-sonnet-4-5-20250929",
    system_prompt=system_prompt,
    tools=[                         # Tools for THIS agent
        find_similar,
        calculate_score,
    ],
    subagents=[                     # Agents to delegate to
        metrics_subagent,
        sentiment_subagent,
        web_subagent,
    ],
)
```

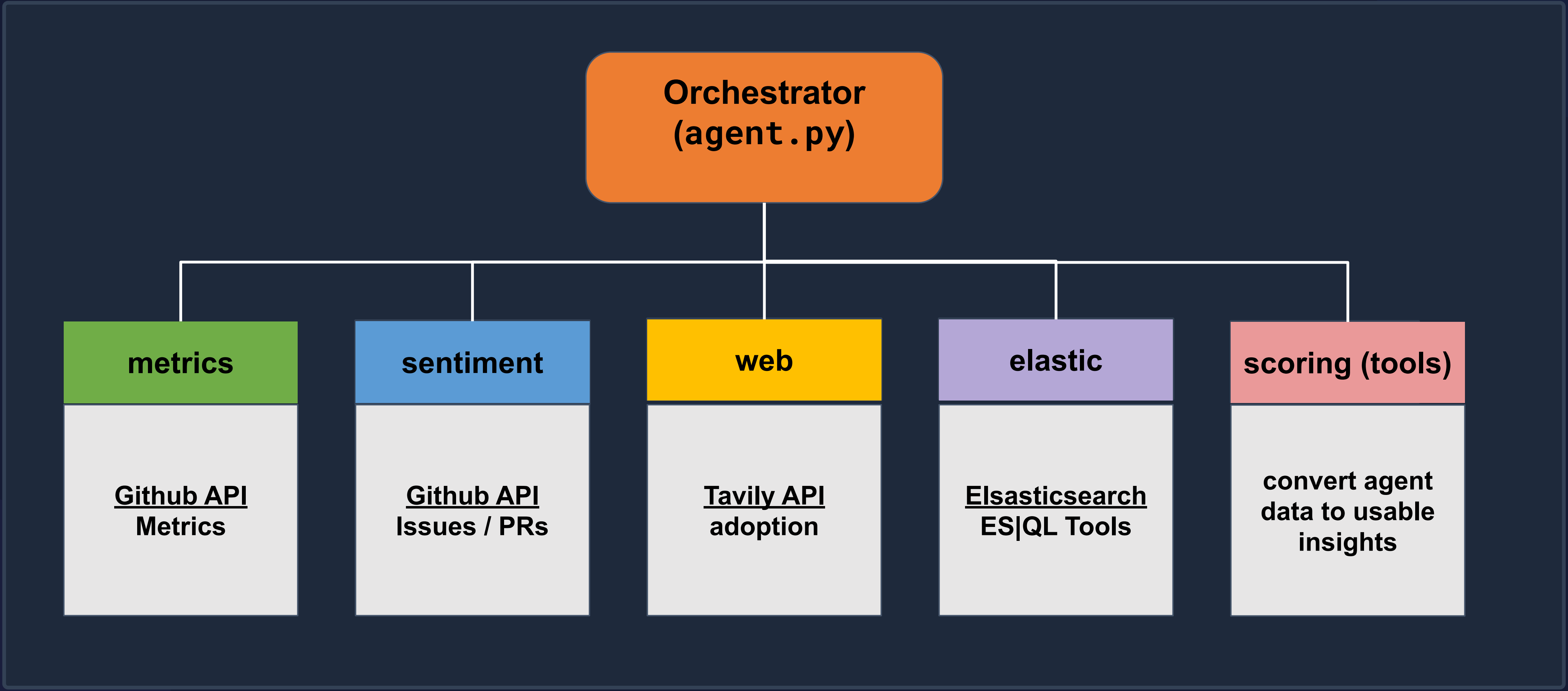
Orchestrator Agent Parallel Execution

```
# In the system prompt, guide parallel execution:
system_prompt = """
## Delegation Rules


When evaluating a repository:
1. Delegate to metrics-agent, sentiment-agent, and web-agent
   IN PARALLEL (they don't depend on each other)
2. Wait for all results
3. Use elastic-agent to check historical data
4. Synthesize findings into final report
"""
```

SubAgent Definition + Description Field

```
{
    "name": "metrics-agent",

    "description": """Fetches GitHub repository metrics including stars, commits,
        contributors, and issue close rates. Use this agent when you need quantitative
        health data about a repository.""",

    "system_prompt": """You are a GitHub Metrics Specialist.

        Your workflow:
        1. Use fetch_repo_metrics to get data
        2. Analyze patterns and trends
        3. Store snapshot for historical tracking
        4. Report findings with specific numbers""",

    "tools": [
        fetch_repo_metrics,
        store_research_snapshot,
    ],
}
```

DevRel Research Agent - Architecture



Orchestrator Tools vs SubAgent Tools

Tool Location	Purpose	Example
Orchestrator	Cross-cutting concerns	`calculate_viability_score`
SubAgent	Domain-specific operations	`fetch_repo_metrics`

Orchestrator-level tools (synthesis, scoring)

```
orchestrator_tools = [
    # Needs data from all agents
    find_similar_technologies,
    # Combines all metrics
    calculate_viability_score,
    # Final output
    store_research_report,
]
```

SubAgent-level tools (domain-specific)

```
metrics_tools = [
    # GitHub API calls
    fetch_repo_metrics,
    # Raw data storage
    store_research_snapshot,
]
```

PART 2

Elastic Agent Builder

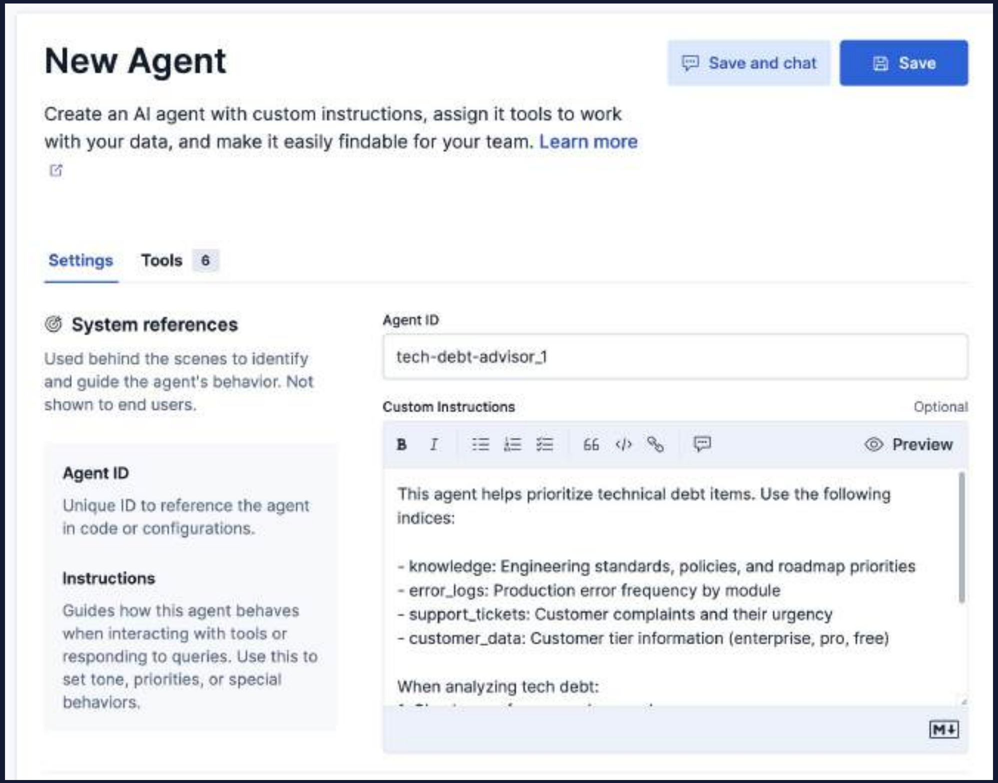
What is Elastic Agent Builder?

The Concept

- A no-code/low-code way to create AI agent tools:
- Execute ES|QL queries against Elasticsearch
- Are callable via the Kibana API
- Support parameterized queries for security
- Enable semantic search with vector embeddings

Why Use It?

- Centralize data access logic in Elasticsearch
- Share tools across multiple agents
- Leverage ES|QL's powerful analytics
- Built-in security and rate limiting



New Agent

Create an AI agent with custom instructions, assign it tools to work with your data, and make it easily findable for your team. Learn more

System references

Used behind the scenes to identify and guide the agent's behavior. Not shown to end users.

Agent ID

Unique ID to reference the agent in code or configurations.

Instructions

Guides how this agent behaves when interacting with tools or responding to queries. Use this to set tone, priorities, or special behaviors.

Agent ID: tech-debt-advisor_1

Custom Instructions (Optional)

This agent helps prioritize technical debt items. Use the following indices:

- knowledge: Engineering standards, policies, and roadmap priorities
- error_logs: Production error frequency by module
- support_tickets: Customer complaints and their urgency
- customer_data: Customer tier information (enterprise, pro, free)

When analyzing tech debt:

ES|QL Primer + Semantic Search

ES|QL - A pipe-based query language

```
FROM technology-research
| WHERE repo == "langchain-ai/langgraph"
| WHERE timestamp > "2025-01-01"
| SORT timestamp DESC
| KEEP repo, timestamp, analysis.viability_score
| LIMIT 10
```

ES|QL Semantic Search

```
FROM technology-research METADATA _score
| WHERE semantic_content:"AI agent frameworks Python"
| SORT _score DESC
| KEEP repo, analysis.summary, _score
| LIMIT 5
```

Key Operators

- **FROM** - Source index
- **WHERE** - Filter conditions
- **SORT** - Ordering
- **KEEP** - Select fields (like SQL SELECT)
- **STATS** - Aggregations

Key Requirements

- ✓ **METADATA _score** — Include relevance scores
- ✓ **Field mapped as semantic_text type**
- ✓ **SORT _score DESC** — Rank by relevance

Creating Tools Via Kibana or API

API Endpoint

```
POST kbn:/api/agent_builder/tools
```

Headers Required

```
kbn-xsrf: true
Authorization: ApiKey ${API_KEY}
Content-Type: application/json
```

Tool Definition Structure

```
{
  "id": "find-similar-technologies",
  "type": "esql",
  "description": "Semantic search for similar technologies",
  "tags": ["search", "semantic"],
  "configuration": {
  "query": "FROM technology-research METADATA _score
          | WHERE semantic_content:?description
          | SORT _score DESC
          | KEEP repo, analysis.summary, _score
          | LIMIT 10”,
  "params": {
    "description": {
      "type": "text",
      "description": "What you're looking for",
      "required": true
    }
  }
  }
}
```

Elastic SubAgent Design

`tools/elastic_agent_client.py`

```python
class ElasticAgentClient:
    def __init__(self, kibana_url: str, api_key: str):
        self.base_url = kibana_url
        self.client = httpx.Client(
            headers={
                "kbn-xsrf": "true",
                "Authorization": f"ApiKey {api_key}",
            },
        )

    def invoke_tool(self, tool_id: str, params: dict) -> dict:
        response = self.client.post(
            f"{self.base_url}/api/agent_builder/tools/{tool_id}/invoke",
            json={"params": params},
        )
        return response.json()
```

Elastic SubAgent Design

subagents/elastic_agent.py

```
elastic_subagent = {
    "name": "elastic-agent",

    "description": """Interfaces with Elasticsearch for all data
    operations. Use when you need to:
    - Search for similar technologies
    - Retrieve historical trends
    - Check caches before API calls
    - Get adoption signals and reports""",

    "system_prompt": """You are an Elastic Data Specialist.

    ## Best Practices You Demonstrate
    1. ES|QL queries with parameterized inputs
    2. Semantic search with METADATA _score
    3. Check caches before expensive API calls
    4. Use STATS for aggregations""",

    "tools": ELASTIC_SUBAGENT_TOOLS,  # 16 tools
}
```

PART 3

Putting It Together

Thank You!

github.com/justincastilla/DevRel-DeepAgent