Introduction to Agents

Yogesh Haribhau Kulkarni

YHK

Outline

Introduction

What Makes an AI Agent Different?

- Unlike LLMs that just respond to prompts, agents are autonomous
- Can look at their environment and analyze the situation
- Make comprehensive plans to achieve specific goals
- Actually take action to execute those plans
- Agents bridge the gap between answering and doing

Welcome to AI Agents

- AI agents represent one of the most exciting frontiers in AI
- Not just everyday chatbots - systems that reason, plan, and take action
- Move beyond AI that just answers questions to AI that does things
- Can take on complex multi-step tasks autonomously
- The core promise: AI that accomplishes goals independently
- Technology is advancing rapidly from conversational to agentic AI

Meet Suresh - The Impossible Job

- Suresh's boss tasks him with planning a massive get-together
- Must research a huge guest list and plan fancy menu
- Needs to find entertainment for the event
- A simple chatbot cannot handle these complex requirements
- Suresh needs an AI agent - not just responses, but actions
- Perfect example of why we need more than conversational AI

Introduction to AI Agents

- 2025 is expected to be the year of AI agents.
- AI agents combine multiple components to solve complex problems.
- Shifting from monolithic models to compound AI systems.
- Compound AI systems use system design for better problem solving.
- AI agents improve with reasoning, acting, and memory components. (ReAct = Reasoning + Acting)

The Evolution of AI Capabilities

- **Traditional Programming:** Needed code to operate
- **Traditional ML:** Needed feature engineering
- **Deep Learning:** task-specific model
- **ChatGPT (2022):** Many tasks single model
  - Zero-shot learning (no examples needed)
  - In-context learning (understands from instructions)
- **Agents (2024 ...):** Can actually **do things**, not just talk

Why Does "Taking Action" Matter?

- In 2022, ChatGPT was revolutionary because AI felt conversational
- By 2024, people wanted more than conversation, they wanted **execution**
- Examples of what users now expect:
  - Instead of listing leads ? **email them directly**
  - Instead of summarizing docs ? **file and create workflow tasks**
  - Instead of suggesting products ? **customize landing pages**
- This shift from **information** to **action** defines the agent era

How Agents Work?

- Agent acts, take you from one state to the other state, provides value by workflow automation. (ReAct paper: Reasoning and Action), it can plan and make decisions.
- Agents have access to tools (ToolFormer paper) e.g. Search APIs, booking, send email etc.
- Interacting of external environment and other Agents, etc.
- Memory to keep the history of conversations/actions done so far.
- May have human-in-loop to keep it sane in the wild-world.
- Agents were there from 1950's but they are effective because of LLMs.
- Agents are systems where LLMs dynamically direct their own processes and tool usage
- Can operate autonomously over extended periods using various tools
- Distinct from workflows: agents have dynamic control vs. predefined code paths
- Essential component in modern AI systems with varying degrees of autonomy

The Agent's Fundamental Game Loop

- Not a one-and-done action, but a continuous reasoning loop
- Similar to a programming while loop that keeps iterating
- **Thought**: Analyzes situation and plans next step
- **Action**: Calls specific tools to execute the plan
- **Observation**: Examines results of the action taken
- Cycle repeats: Thought $\rightarrow$ Action $\rightarrow$ Observation
- Continues until the task is completely accomplished
- This loop enables continuous adaptation and problem-solving

YHK

Agent's Inner Monologue

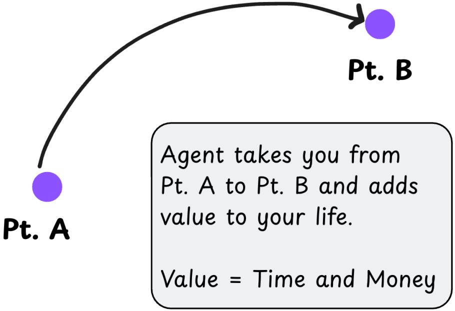
- Agents have visible thought processes before taking action
- Example: "User wants weather in New York. I have a tool for that."
- "My first move is to call the weather API"
- Internal planning step makes agents more than reactive programs
- Reasoning through problems before execution
- This deliberation distinguishes agents from simple scripts
- Shows intelligent decision-making rather than blind execution

How Do Agents Take Action?

- The magic lies in **tools** and **function calling**
- Agents are paired with APIs, plugins, or external systems
- Instead of just text responses, LLMs output structured commands:
  - "Call the send_email() function with these inputs..."
  - "Fetch records from CRM using this query..."
  - "Schedule a meeting for Tuesday at 2PM..."
- **Mental model:** LLM = brain, Tools = hands
- Without tools, agents just talk. With tools, they act.

Defining AI Agents with an example

- Planning a trip involves many complex tasks
- Point A: Just discussing the trip
- Point B: All bookings and itinerary ready
- AI Agents aim to take you from A to B
- First idea: Agent adds value by saving time/money

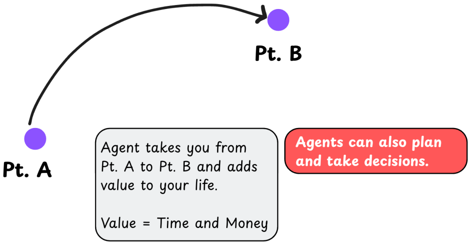


(Ref: Vizuara AI Agents Bootcamp)

YHK

Evolving Definition of Agents

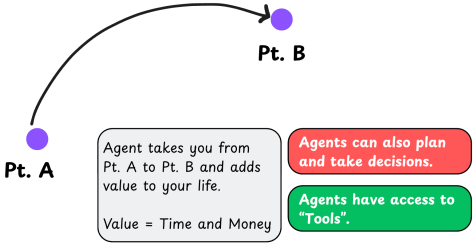
- Not all tools from A to B are agents (e.g., cars)
- Agents must plan and make decisions
- Second definition includes decision-making ability
- Example: Choosing flights based on budget
- Planning daily itinerary needs contextual judgment



(Ref: Vizuara AI Agents Bootcamp)

Agents Need Tools

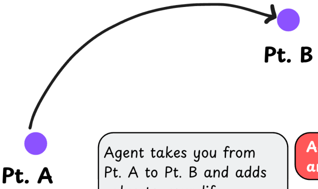
- Even self-driving cars plan but are not agents
- Agents need access to external tools
- Tools = Access to services (e.g., Gmail, Booking)
- Agents perform tasks using these tools
- Third definition adds tool access to capabilities



(Ref: Vizuara AI Agents Bootcamp)

Rise of LLMs in Agents

- Transformers (2017) enabled powerful LLMs
- LLMs understand and generate human language
- Agents use LLMs for reasoning and planning
- LLMs enable understanding of webpages and writing emails
- Fourth definition: Agents are LLMs with tools and planning ability



Agent takes you from Pt. A to Pt. B and adds value to your life.

Value = Time and Money

Agents can also plan and take decisions.

Agents have access to "Tools".

The brain or foundation of an agent is a Large Language Model (LLM).

(Ref: Vizuara AI Agents Bootcamp)

What Is an Agent? (Technical Definition)

- Agent acts and takes you from one state to another, providing value through workflow automation
- Based on ReAct paradigm: **Reasoning + Acting**
- Key capabilities:
  - Can plan and make decisions
  - Has access to tools (search APIs, booking, email, etc.)
  - Interacts with external environments and other agents
  - Maintains memory of conversations and actions
  - May include human-in-the-loop for safety
- Agents existed since the 1950s but are now effective because of LLMs

Two Ways to Define Agents

Technical View:

- LLM (brain)
- + Tools (hands)
- + Planning (strategy)
- + Memory (context)
- + State management

Business View:

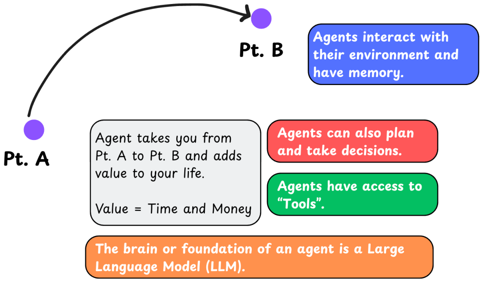
- Systems that complete tasks end-to-end
- Focus on outcomes, not components
- Solve real-world problems
- Provide measurable value

Important: Today's agents are **engineering wrappers** around AI models, the intelligence comes from the LLMs, agents help act on that intelligence.

YHK

Final Definition of Agents

- Agents can learn from feedback and environment
- Agents interact with tools, humans, and websites
- They improve with experience (memory)
- Fifth definition: LLMs + Tools + Planning + Learning
- Agents evolve over time via memory and feedback



(Ref: Vizuara AI Agents Bootcamp)

Understanding Agency

- Agency = Level of autonomy an agent has
- Low agency → less value
- High agency → high value
- More autonomous agents can handle complex tasks
- Agency is key to measuring agent usefulness

Agency Level	Description	Name	Example
●○○○○	Agent does not influence what happens next	Simple Processor	Grammar checker that rewrites sentences
●●○○○	Agent determines basic control flow	Router	Customer Query → Tech Support or Sales
●●●○○	Agent determines function execution	Tool Caller	A smart calendar assistant that spots “let's meet on Tuesday” and books the meeting
●●●●○	Lays out a short plan and carries it step by step	Multi-step Agent	Personal travel planner that gathers flight options, hotels, local activities
●●●●●	One agentic workflow starts another agentic workflow	Multi-agent	Travel planner agent → Booking agent ← Email agent

(Ref: Vizuara AI Agents Bootcamp)

YHK

Tools: The Agent's Hands

- LLM is the agent's brain; tools are its hands
- Tools are functions agents call to interact with the world
- Can search web, run calculations, or query databases
- Bridge between thinking and doing in the real world
- Enable agents to move from planning to execution
- Without tools, agent thoughts would be useless
- Tools provide the interface to external systems and data

Two Ways Agents Use Tools

- **JSON Agent**: Writes structured work orders for other systems
- JSON approach requires external system to read and execute
- **Code Agent**: Directly writes and runs code blocks
- Code approach is more direct and powerful
- Code is naturally more expressive than JSON
- Can handle complex logic like loops and conditionals
- Modular, easier to debug, and taps into existing libraries
- Code agents can access thousands of APIs directly

Code Agent in Action

- Alfred needs a gala menu - agent has "suggest_menu" tool
- Agent doesn't just make up suggestions randomly
- Generates and runs actual code to call the specific tool
- Gets real results from the tool execution
- Super direct, efficient, and powerful way to take action
- Code generation enables precise tool interaction
- Results are based on actual tool capabilities, not hallucination

Advanced Pattern: Agentic RAG

- Traditional RAG: Retrieval Augmented Generation fetches info before answering
- Agentic RAG supercharges this with intelligent multi-step processes
- Turns retrieval itself into an agent-driven task
- Like having a master researcher on staff
- Doesn't just do one search - runs complete research processes
- Rewrites queries for better results and runs multiple searches
- Uses findings to inform next searches and validates accuracy
- Pulls from both private data and public web sources

Multi-Agent Systems: Digital Teams

- Complex problems like finding the missing Batmobile need teams
- Single agents can't handle web searches, calculations, and visualization
- Solution: Build teams of specialized agents
- Manager agent acts as project lead breaking down big tasks
- Delegates work to specialist agents with specific skills
- Web agent handles online searching while manager coordinates
- Manager focuses on big picture and final integration
- Digital division of labor for complex problem solving

The GAIA Benchmark Reality Check

- GAIA benchmark tests real-world multi-step problems
- Measures how well systems handle tricky, complex tasks
- Results are eye-opening and show current limitations
- Humans solve these tasks with 92% accuracy
- Today's most advanced AI models: only 15% accuracy
- Massive 77% gap between human and AI performance
- This gap is exactly what agentic systems aim to close
- Shows the enormous potential for improvement

The Fundamental Shift

- Moving from conversational AI to agentic AI era
- Old paradigm: Ask questions, get answers
- New paradigm: State goals, systems plan and accomplish them
- Represents fundamental change in human-computer interaction
- Technology for building personal AI assistants advancing rapidly
- Not a question of "if" but "when" this becomes reality
- The future Alfred is closer than we think
- Prepare for AI that can handle "impossible" complex tasks

Papers that Shaped AI Agents

- Core research papers laid the foundation
- Introduced key frameworks and architectures
- Sparked recent boom in agent development
- Include Transformer and Agentic frameworks
- Major driving force in LLM-based agent systems

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou
Google Research, Brain Team
{jasonwei,dennyzhou}@google.com

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]
[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom
Meta AI Research [†]Universitat Pompeu Fabra

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park, Stanford University, Stanford, USA, joonspk@stanford.edu
Joseph C. O'Brien, Stanford University, Stanford, USA, jobrien3@stanford.edu
Carrie J. Cai, Google Research, Mountain View, CA, USA, cjcai@google.com
Meredith Ringel Morris, Google DeepMind, Seattle, WA, USA, merrie@google.com
Percy Liang, Stanford University, Stanford, USA, pliang@cs.stanford.edu
Michael S. Bernstein, Stanford University, Stanford, USA, msb@cs.stanford.edu

(Ref: Vizuara AI Agents Bootcamp)

When to Use Agents?

- Best suited for tasks requiring flexibility and model-driven decision-making
- Consider tradeoffs: agents increase latency and cost for better task performance
- Recommended for open-ended problems with unpredictable steps
- Simple solutions preferred - single LLM calls with retrieval often sufficient

Future AI Applications

- What are future AI applications like?
  - **Generative:** Generate content like text and images
  - **Agentic:** Execute complex tasks on behalf of humans
- How do we empower every developer to build them?
  - **Co-Pilots:** Human-AI collaboration
  - **Autonomous:** Independent task execution
- 2024 is expected to be the year of AI agents

The Big Question

- Agentic AI technology is moving incredibly fast
- Personal AI assistants will soon be capable of complex tasks
- Think beyond simple queries to multi-step accomplishments
- Consider what "impossible" tasks you want to delegate
- What complex, party-of-the-century level challenge will you tackle?
- The era of AI that truly does rather than just discusses
- Prepare for AI assistants that can handle your biggest challenges

Implementations

Agno

Introduction to Agno Framework

- Agno (erstwhile phidata) is a open-source, light-weight framework for building Multi-Agent Systems with memory, knowledge and reasoning
- Official documentation available at: `https://docs.agno.com/examples/introduction`
- Designed for for speed and efficiency and to build production-ready agentic systems with minimal boilerplate
- Agno exposes LLMs as a unified API and gives them superpowers like memory, knowledge, tools and reasoning.
- Supports 5 progressive levels of agentic system complexity
- Framework focuses on performance, reliability, and ease of use
- Built for both individual agents and complex multi-agent workflows

Key Components

- Agents: Think in terms of agency and autonomy:
- Tools: These are like plugins that give agents extraordinary abilities.
- Memory and Knowledge: Remember past interactions and store relevant information, which allows them to have context and build on previous conversations Agno also supports connecting to knowledge stores, like Vector databases, to enable retrieval augmented generation (RAG)
- Multi-Agent Orchestration: to create teams of agents that can work together

The 5 Levels of Agentic Systems

- **Level 1:** Agents with tools and instructions - Basic autonomous task execution
- **Level 2:** Agents with knowledge and storage - Persistent data management
- **Level 3:** Agents with memory and reasoning - Context-aware decision making
- **Level 4:** Agent Teams that can reason and collaborate - Multi-agent coordination
- **Level 5:** Agentic Workflows with state and determinism - Complex orchestrated processes
- Each level builds upon the previous, enabling progressively sophisticated AI systems

Key Features - Model Support & Performance

- **Model Agnostic:** Unified interface to 23+ model providers with no vendor lock-in
- **High Performance:** Agents instantiate in approximately 3 microseconds
- **Memory Efficient:** Uses only 6.5KB memory on average per agent
- **Scalable Architecture:** Designed for production workloads and high throughput
- **Zero Dependencies Overhead:** Minimal resource footprint for deployment
- **Production Ready:** Built-in FastAPI routes for immediate deployment

Advanced Capabilities - Reasoning & Multi-Modal

- **Reasoning First-Class:** Three approaches - Reasoning Models, ReasoningTools, custom chain-of-thought
- **Multi-Modal Native:** Accepts text, image, audio, and video inputs
- **Multi-Modal Output:** Generates text, image, audio, and video responses
- **Reliability Focus:** Reasoning improves system reliability for autonomous operations
- **Complex Task Support:** Essential for sophisticated autonomous agent workflows
- **Flexible Implementation:** Choose reasoning approach based on use case requirements

Enterprise Features - Search, Memory & Storage

- **Built-in Agentic Search:** Runtime information retrieval using 20+ vector databases
- **State-of-the-art RAG:** Fully asynchronous and highly performant retrieval
- **Long-term Memory:** Built-in Storage & Memory drivers for persistent context
- **Session Storage:** Maintain conversation state across interactions
- **Structured Outputs:** Fully-typed responses using model structured outputs or JSON mode
- **Real-time Monitoring:** Track agent sessions and performance on agno.com

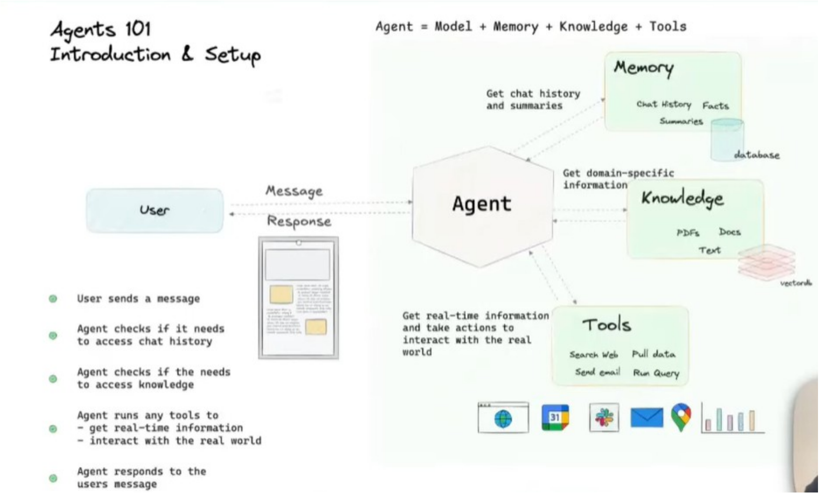
Installation & Quick Start

- **Simple Installation:** Single pip command for complete setup
- **No Complex Dependencies:** Minimal requirements for basic functionality
- **Quick Deployment:** 0 to production in minutes with pre-built routes
- **Extensible Design:** Add tools and capabilities as needed
- **Documentation:** Comprehensive examples and guides available

```
pip install -U agno
```

YHK

Recap: What is an Agent?



(Ref: Agents 101 - Agno channel)

Level 1 Agent Example - Basic Setup

- Basic reasoning agent with YFinance API integration
- Uses Claude Sonnet 4 model for advanced reasoning capabilities
- Incorporates ReasoningTools for structured thinking process
- YFinanceTools provide stock market data access
- Markdown output for formatted responses
- Table formatting for clear data presentation

```python
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.reasoning import ReasoningTools
from agno.tools.yfinance import YFinanceTools

reasoning_agent = Agent(
    model=Claude(id="claude-sonnet-4-20250514"),
    tools=[
        ReasoningTools(add_instructions=True),
        YFinanceTools(stock_price=True,
                      analyst_recommendations=True,
                      company_info=True, company_news=True),
    ],
    instructions="Use tables to display data.",
    markdown=True,
)
agent.print_response("How NVIDIA stock performed in last two years?",
                     stream=True, show_full_reasoning=True,stream_intermediate_steps=True)
```

Complete Reasoning Agent Implementation

- Full implementation showing agent creation and execution
- Multiple instruction types for behavior control
- Stream processing with intermediate step visibility
- Full reasoning transparency for debugging and understanding
- Real-time output streaming for user engagement
- Clean report generation without extraneous text

```python
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.reasoning import ReasoningTools
from agno.tools.yfinance import YFinanceTools

agent = Agent(model=Claude(id="claude-sonnet-4-20250514"),
    tools=[ReasoningTools(add_instructions=True),
           YFinanceTools(stock_price=True, analyst_recommendations=True,
                         company_info=True, company_news=True)],
    instructions=["Use tables to display data",
                  "Only output the report, no other text"],
    markdown=True,)

agent.print_response("Write a report on NVIDIA",
                     stream=True, show_full_reasoning=True,stream_intermediate_steps=True)
```

Multi-Agent Teams - Architecture Principles

- **Atomic Agents:** Individual agents work best with narrow scope and limited tools
- **Specialization:** Each agent focuses on specific domain expertise
- **Load Distribution:** Teams spread cognitive load across multiple agents
- **Scalable Design:** Handle multiple concepts through agent collaboration
- **Tool Management:** Prevent tool overload by distributing capabilities
- **Coordinated Execution:** Team-level orchestration for complex tasks

Multi-Agent Team Implementation - Part 1

- Web Agent specializes in information search and sourcing
- Finance Agent focuses on financial data retrieval and analysis
- Each agent has dedicated tools for their domain
- Domain-specific instructions optimize agent behavior

```python
from agno.agent import Agent
from agno.models.openai import OpenAIChat
from agno.tools.duckduckgo import DuckDuckGoTools
from agno.tools.yfinance import YFinanceTools

web_agent = Agent(name="Web Agent",
    role="Search the web for information",
    model=OpenAIChat(id="gpt-4o"),
    tools=[DuckDuckGoTools()],
    instructions="Always include sources",
    show_tool_calls=True, markdown=True,)

finance_agent = Agent(name="Finance Agent", role="Get financial data",
    model=OpenAIChat(id="gpt-4o"),
    tools=[YFinanceTools(stock_price=True,
                         analyst_recommendations=True,
                         company_info=True)],
    instructions="Use tables to display data",
    show_tool_calls=True, markdown=True,)
```

Multi-Agent Team Implementation - Part 2

- Team coordination mode enables collaborative agent interaction
- Stream processing provides real-time team collaboration visibility
- Comprehensive reporting combines multiple agent capabilities
- `pip install ddgs yfinance`

```python
from agno.team import Team

agent_team = Team(
    mode="coordinate",
    members=[web_agent, finance_agent],
    model=OpenAIChat(id="gpt-4o"),
    success_criteria="A comprehensive financial news report with clear sections and data-driven insights.",
    instructions=["Always include sources",
                  "Use tables to display data"],
    show_tool_calls=True,
    markdown=True,)

agent_team.print_response(
    "What's the market outlook and financial performance of AI semiconductor companies?",
    stream=True)
```

YHK

Performance Philosophy & Benchmarking

- **Performance by Design:** Agents optimized for speed and efficiency from ground up
- **Accuracy Priority:** Reliability and correctness more important than raw speed
- **Fair Benchmarking:** Framework differences make direct comparisons challenging
- **Self-Comparison Focus:** Future benchmarks will compare against previous Agno versions
- **Continuous Optimization:** Performance tuning is ongoing development priority
- **Production Metrics:** Real-world performance measurement over synthetic benchmarks

Getting Started - Next Steps

- **Start Simple:** Begin with Level 1 agents to understand core concepts
- **Progressive Complexity:** Advance through levels as requirements grow
- **Documentation:** Explore comprehensive examples at docs.agno.com
- **Community Support:** Active community for questions and best practices
- **Production Deployment:** Built-in FastAPI routes for immediate scaling
- **Monitoring Integration:** Use agno.com for real-time system insights

References

- CS 194/294-196 (LLM Agents) - Lecture 3, Chi Wang and Jerry Liu
- LLM Powered Autonomous Agents Lil'Log
- Power of Autonomous AI Agents - Yogesh Kulkarni
- Microsoft AutoGen- Yogesh Kulkarni
- Microsoft AutoGen using Open Source Models- Yogesh Kulkarni
- A CAMEL ride - Yogesh Kulkarni
- Autonomous AI Agents (LLM, VLM, VLA) - Code Your Own AI
- Awesome LLM-Powered Agent https://github.com/hyp1231/awesome-llm-powered-agent
- Autonomous Agents (LLMs). Updated daily https://github.com/tmgthb/Autonomous-Agents

Thanks ...

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 3 pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

(https://www.linkedin.com/in/yogeshkulkarni/)

(https://medium.com/@yogeshharibhaukulkarni)

(https://www.github.com/yogeshhk/)