5-Day AI Agents Intensive Course with Google

This repository contains materials and assignments for the "5-Day AI Agents Intensive Course with Google".

Course Link: https://www.kaggle.com/learn-guide/5-day-agents
Sessions Playlist: https://www.youtube.com/playlist?list=PLqFaTIg4myu9r7uRoNfbJhHUbLp-1t1YE

Day 1: Introduction to Agents

1. **Complete Unit 1 – "Introduction to Agents"**:
    - Listen to the summary podcast episode for this unit, created by NotebookLM: https://www.youtube.com/watch?v=zTxvGzpfF-g
    - To complement the podcast, read the "Introduction to Agents" whitepaper: https://www.kaggle.com/whitepaper-introduction-to-agents

Podcast Summary: Whitepaper Companion - Introduction to Agents (Google x Kaggle)

- **Core Shift**: AI moving from passive (Q&A, translation) to **autonomous AI agents** that plan, act, and solve multi-step goals without constant human guidance.
- **Agent Anatomy (3 Core Parts)**:
    - **Model (Brain)**: LLM for reasoning; curates context window (mission, memory, tool results).
    - **Tools (Hands)**: APIs, databases, code execution—enable real-world actions.
    - **Orchestration Layer (Conductor)**: Manages the loop, memory, persona, and reasoning strategy (e.g., ReAct: Reason → Act → Observe → Repeat).
- **Agent Loop (5 Stages)**:
    1. Get mission
    2. Scan scene (tools + memory)
    3. Think/Plan
    4. Take action (call tool)
    5. Observe → iterate
- **Agent Capability Taxonomy**:
    - **Level 0**: Pure LLM (no tools, static knowledge)
    - **Level 1**: Connected problem-solver (uses tools, real-time data)
    - **Level 2**: Strategic problem-solver (context engineering, chains tool outputs intelligently)
    - **Level 3**: Collaborative multi-agent (agents delegate to specialist agents)
    - **Level 4**: Self-evolving (detects gaps, builds new agents/tools on the fly)
- **Production-Ready Practices**:
    - **Model Routing**: Use heavy models (e.g., Gemini 1.5 Pro) for planning, lighter ones (Flash) for simple tasks.
    - **Tools**: RAG, NL2SQL, APIs with clear OpenAPI specs + function calling.
    - **Memory**: Short-term (current loop) + long-term (vector DB/RAG).
    - **Testing**: LM-as-judge + golden datasets; OpenTelemetry traces for debugging.
    - **Security**: Defense-in-depth (hard rules + guard models), agent identity (least-privilege), central gateway for governance.
    - **Evolution**: Feedback loops, agent gym (simulation sandbox).
- **Examples**:
    - **Co-Scientist**: Level 3–4 research collaborator with supervisor + specialist agents.
    - **AlphaEvolve**: Level 4 system that discovers/optimizes algorithms via code generation + evolution.

Takeaway: Success isn't just the smartest model, it's rigorous engineering of model + tools + orchestration + ops. Builders become architects of autonomous systems. Check the whitepaper and 5-day Google AI Agents Intensive.

2. **Complete these codelabs on Kaggle**:
    - **a.** Build your first agent using Gemini and Agent Development Kit (ADK): https://www.kaggle.com/code/kaggle5daysofai/day-1a-from-prompt-to-action
    - **b.** Build your first multi-agent systems using ADK: https://www.kaggle.com/code/kaggle5daysofai/day-1b-agent-architectures
3. **Important Notes**:
    - **c.** Make sure you phone verify your Kaggle account before starting, it's necessary for the codelabs: https://www.kaggle.com/settings
    - **d.** We also have a troubleshooting guide for the codelabs. Be sure to check there for solutions to common problems: https://www.kaggle.com/code/kaggle5daysofai/day-0-troubleshooting-and-faqs
    - **e.** Want to have an interactive conversation? Try adding the whitepaper to NotebookLM: https://notebooklm.google.com/

Day 2: Agent Tools & Interoperability with Model Context Protocol (MCP)

1. **Complete Unit 2 - "Agent Tools & Interoperability with Model Context Protocol (MCP)"**:
    - Listen to the summary podcast episode for this unit, created by NotebookLM: https://www.youtube.com/watch?v=Cr4NA6rxHAM
    - To complement the podcast, read the "Agent Tools & Interoperability with Model Context Protocol (MCP)" whitepaper: https://www.kaggle.com/whitepaper-agent-tools-and-interoperability-with-mcp

Podcast Summary: Whitepaper Companion - Agent Tools & Interoperability with MCP (Google x Kaggle)

- **Core Problem & Shift**: Foundation models are isolated "brains" (pattern-matching from training data); need "tools" (senses/actuators) to perceive/act in real

world for agentic AI, especially in enterprises.

- **Historical Challenge**: Integrating N models with M tools created fragmented, non-scalable custom connectors (N*M explosion).
- **Solution: Model Context Protocol (MCP)**: Open standard (2024) for unified, plug-and-play tool integration; decouples agent reasoning from tool execution.
- **Tool Types**:
  - **Function Tools**: Developer-defined (e.g., Python functions); use detailed docstrings/schemas for inputs/outputs (contract).
  - **Built-in Tools**: Implicitly provided by model provider (e.g., Gemini's Google Search grounding, code execution, URL fetching).
  - **Agent Tools**: Invoke sub-agents hierarchically (delegation, not full handoff; main agent stays in control).
- **Broader Tool Taxonomy**:
  - Information retrieval (fetch data).
  - Action execution (make changes, e.g., send email).
  - System API integration (connect software).
  - Human-in-the-loop (seek permission/clarification).
- **Best Practices for Tool Design**:
  - Documentation essential: Clear, descriptive names/descriptions/parameters fed into LLM context.
  - Focus on tasks/actions (e.g., "Create bug"), not implementation/how-to.
  - Publish high-level tasks, not raw APIs (encapsulate complexity; avoid dozens of parameters).
  - Concise outputs: Summaries, confirmations, or URIs/references (e.g., via ADK artifact service) to prevent context bloat (tokens, latency, degraded reasoning).
  - Handle errors instructively: Schema validation; descriptive messages with recovery guidance (e.g., "Rate limit exceeded; retry in 15s").
- **MCP Technical Details**:
  - **Transports**: Local IPC (efficient, same-machine); Streamable HTTP (distributed, supports SSE for long-running tools).
  - **Primitives**: Tools (core focus); resources/prompts (less adopted).
  - **Tool Definition**: JSON schema (name, description, input/output schemas; e.g., get_stock_price with symbol/date).
  - **Results**: Structured (JSON per schema) or unstructured (text, files, URIs).
  - **Errors**: Protocol-level (e.g., invalid method); execution-level (flag + descriptive message for LLM recovery).
- **Strategic Benefits**:
  - Accelerates development; fosters reusable ecosystem (e.g., public MCP registries for certified tools).
  - Enables dynamic discovery (agents find/use new tools at runtime).
  - Architectural flexibility: Modular systems, agentic AI mesh (networks of agents/tools).
- **Challenges & Mitigations**:
  - **Context Bloat**: Loading many tool definitions overwhelms LLM; solve with tool retrieval (RAG-like semantic search: index tools, load only top 3-5 relevant ones).
  - **Security Gaps**: MCP lacks built-in auth/authz; risks like confused deputy (prompt injection escalates privileges via trusted AI).
  - **Enterprise Fix**: Wrap MCP in API gateways (e.g., Apigee): Handle auth, fine-grained authz, rate limiting, logging, input filtering.

Takeaway: Tools empower models to act; MCP standardizes connections; apply best practices for reliability; layer enterprise security for safe adoption. Check whitepaper and 5-Day AI Agents Intensive for hands-on.

2. **Complete these codelabs on Kaggle**:
   - Explore new ways to add tools to extend what your agents can do: https://www.kaggle.com/code/kaggle5daysofai/day-2a-agent-tools
   - Explore best practices for tools, including using MCP and long-running operations: https://www.kaggle.com/code/kaggle5daysofai/day-2b-agent-tools-best-practices

---

Day 3: Context Engineering: Sessions & Memory

Day 3 Assignment

1. **Complete Unit 3 - "Context Engineering: Sessions & Memory"**:
   - Listen to the summary podcast episode for this unit, created by NotebookLM: https://www.youtube.com/watch?v=FMcExVE15a4
   - To complement the podcast, read the "Context Engineering: Sessions & Memory whitepaper": https://www.kaggle.com/whitepaper-context-engineering-sessions-and-memory

Podcast Summary: Whitepaper Companion - Context Engineering: Sessions & Memory (Google x Kaggle)

- **Core Concepts & Shift**: LLMs are stateless; context engineering dynamically assembles/manages info in the context window for stateful, personalized agents. Involves three pillars: context engineering (master controller), sessions (short-term conversation container), and memory (long-term persistence across sessions).
- **Context Engineering**:
  - Addresses LLM statelessness by building dynamic inputs per call: system instructions, tools, external data (RAG, memories), conversation history, user message.
  - Fights "context rot" (noisy/overfull windows degrading reasoning) via dynamic history mutation (summarization, pruning).
  - Cycle: Fetch context → Prepare prompt (hot path) → Invoke LLM/tools → Upload new insights (async).
- **Sessions**:
  - Self-contained per user/conversation; holds events (chronological log: user/agent messages, tool calls) and state (structured working memory, e.g., cart items, process steps).
  - Frameworks vary: ADK uses explicit session object; LangGraph uses mutable state for in-place compaction (e.g., replace old messages with summaries).

    - In multi-agent systems (MAS): Shared unified history (all agents access central log, good for collaboration) vs. separate histories (each agent has own, with selective sharing for privacy/efficiency).

- **Memory**:
    - Long-term storage for personalization; scopes: User-level (cross-session), session-level (temporary), application-level (global/procedural, needs sanitization to avoid leaks).
    - Storage: Vector DBs (semantic search) + knowledge graphs (relational queries) for hybrid unstructured/structured data; memories stored as text extracts (even from multimodal sources).
    - Generation: LLM-driven ETL pipeline (async): Extract meaningful nuggets (targeted filtering/rules), Consolidate (create/update/delete; handle conflicts via provenance, confidence, relevance decay), Load to storage.
    - Retrieval: Blend scores (relevance + recency + importance); proactive (fetch at turn start) vs. reactive (agent queries via tools like "create_memory" for autonomy).
    - Inference: Placement in prompt matters—system instructions (high authority, for stable facts) vs. conversation history (risks confusion/dilution).

- **Testing & Best Practices**:
    - Metrics: Precision/recall (generation), recall@K (retrieval), latency (<200ms), end-to-end task success (via LLM judge).
    - Compaction strategies: Recursive summarization (summarize summaries over time); reactive memory tools for agent self-management.

Takeaway: Context engineering turns stateless LLMs into adaptive agents via sessions (immediate workbench) and memory (persistent filing cabinet); focus on async ETL, hybrid storage, and rigorous testing for reliable personalization. Check whitepaper and 5-Day AI Agents Intensive.

2. **Complete these codelabs on Kaggle**:
    - Build stateful agents and perform context engineering: https://www.kaggle.com/code/kaggle5daysofai/day-3a-agent-sessions
    - Explore how to use memory with your agent: https://www.kaggle.com/code/kaggle5daysofai/day-3b-agent-memory

---

Day 4: Agent Quality

Day 4 Assignment

1. **Complete Unit 4 - "Agent Quality"**:
    - Listen to the summary podcast episode for this unit, created by NotebookLM: https://www.youtube.com/watch?v=LFQRy-Ci-Ik
    - To complement the podcast, read the "Agent Quality" whitepaper: https://www.kaggle.com/whitepaper-agent-quality

Podcast Summary: Whitepaper Companion - Agent Quality (Google x Kaggle)

- **Core Shift & Problem**: AI agents are non-deterministic (unpredictable paths); traditional QA (pass/fail) fails—quality must be an architectural pillar from the start, not just end-testing.

- **Three Key Messages**:
    - **Trajectory is the Truth**: Evaluate the entire process/path (reasoning, steps), not just final output; reveals inefficiencies, near-misses.
    - **Observability is Foundation**: Need logging/tracing/metrics to inspect agent's "black box" reasoning for debugging/evaluation.
    - **Evaluation as Continuous Loop**: "Agent Quality Flywheel"—real-world insights (esp. failures) feed back to improve agent/evaluation.

- **Analogy & Failure Modes**:
    - Traditional software: Delivery truck (fixed routes, explicit crashes); agents: F1 car (dynamic judgments, insidious failures).
    - Failures: Algorithmic bias (e.g., resume screening amplifies biases); factual hallucinations (confident inventions); performance/concept drift (outdated assumptions); emergent behaviors (superstitions/loopholes).

- **Evaluation Pillars (What to Measure)**:
    - **Correctness**: Factuality, safety, policy adherence.
    - **Usefulness**: Helpfulness, relevance, timeliness.
    - **Efficiency**: Latency, cost, resource use.
    - **Robustness**: Handles edge cases, variations, failures gracefully.

- **Hybrid Evaluation System (Who Measures)**:
    - **Automated (Scales)**: LLM judges (score trajectories via prompts, e.g., "Is this factually accurate?"); unit/integration tests; synthetic data; agent gym (simulations).
    - **Human (Arbiter)**: Experts for nuance/edge cases; crowd-sourcing for scale; red teaming (adversarial probes).
    - **Real-World (Flywheel Fuel)**: User feedback, A/B tests; capture failures as new test cases.

- **Observability Pillars (How to See Inside)**:
    - **Logging**: Structured (JSON) diary of atomic steps (chain of thought, tool I/O).
    - **Tracing**: Narrative thread linking logs (e.g., via OpenTelemetry spans) to show cause/effect.
    - **Metrics**: Aggregated health—system (latency P99, error rates, costs) for ops; quality (correctness scores, trajectory adherence) for data scientists/PMs.
    - **Optimization**: Dynamic sampling (100% failures, 10% successes) to reduce overhead.

- **Flywheel in Action**: Define targets → Instrument observability → Evaluate hybrid → Feedback loop refines agent/evaluation; turns failures into improvements.

Takeaway: Build trustworthy agents by designing for quality (trajectory focus, observability), using hybrid eval, and continuous loops; not just capable, but reliable. Check whitepaper and 5-Day AI Agents Intensive.

2. **Complete these codelabs on Kaggle**:
    - Implement observability to help you debug your agents: https://www.kaggle.com/code/kaggle5daysofai/day-4a-agent-observability
    - Evaluate your agents: https://www.kaggle.com/code/kaggle5daysofai/day-4b-agent-evaluation

---

Day 5: Prototype to Production

Final Assignment

1. **Complete Unit 5 - "Prototype to Production"**:
    - Listen to the summary podcast episode for this unit: https://www.youtube.com/watch?v=8Wyt9l7ge-g
    - To complement the podcast, read the "Prototype to Production" whitepaper: https://www.kaggle.com/whitepaper-prototype-to-production

Podcast Summary: Whitepaper Companion - Prototype to Production (Google x Kaggle)

- **Core Challenge**: AI agent prototypes are quick to build but productionizing them requires 80% effort on infrastructure, security, and validation—bridging the "last mile" gap for reliable, enterprise-grade systems.
- **Differences from Traditional MLOps**: Agents are autonomous/dynamic (unpredictable paths, tool orchestration); need scalable state management (memory across interactions); handle variable costs/latency (1-50 steps, cheap/expensive tools).
- **Pre-Production Fundamentals**:
    - Build representative eval datasets (realistic, diverse, synthetic via LLMs).
    - CI/CD pipelines: Automated eval as gates; version control (prompts, tools, agents); agent gyms (simulated environments for safe testing multi-agent interactions).
    - Hybrid testing: Offline (unit/integration) + online (A/B, canary releases).
- **Deployment Strategies**:
    - Architect for stateless (orchestration) vs. stateful (memory) components.
    - Platforms: Kubernetes (orchestration), serverless (e.g., Cloud Run), model serving (Vertex AI, Ray Serve, TGI).
    - Reliability: Auto-scaling, circuit breakers, retries; separate dev/staging/prod environments.
- **Scaling & Optimization**:
    - Horizontal (more pods/instances) + vertical (GPU/TPU upgrades).
    - Efficiency: Caching (prompts, embeddings, tool outputs); async processing (non-blocking tool calls); rate limiting/budgets (per-user/session costs).
    - Multi-agent: Hierarchical delegation (supervisor agents coordinate specialists).
- **Monitoring & Observability**:
    - Distributed tracing (OpenTelemetry) for multi-step/agent flows; metrics (latency, errors, costs); anomaly detection (e.g., sudden cost spikes).
    - Feedback loops: User signals → auto-fine-tuning; post-mortems for failures.
- **Security & Governance**:
    - RBAC (role-based access); data encryption (at-rest/transit); audit logs (actions, decisions).
    - Compliance: GDPR (data minimization); ethical AI (bias checks); versioning for rollback.
- **A2A (Agent-to-Agent) Interoperability**:
    - High-level collaboration (goal-oriented delegation) vs. MCP (low-level tools).
    - Discovery: Agent cards (JSON manifests: capabilities, URLs, security).
    - Infrastructure: Tracing/state management across agents; registries (tool/agent catalogs) for large ecosystems to enable findability/governance.

Takeaway: AgentOps transforms prototypes into scalable, trustworthy systems via robust CI/CD, infra, and A2A—enabling velocity (fast iterations) and collaborative AI ecosystems. This wraps the 5-day series; check whitepaper and AI Agents Intensive for details.

2. **Complete these codelabs on Kaggle**:
    - Explore how to use A2A Protocol to have agents interact with each other: https://www.kaggle.com/code/kaggle5daysofai/day-5a-agent2agent-communication
    - [Optional] Deploy your agent to Agent Engine on Google Cloud: https://www.kaggle.com/code/kaggle5daysofai/day-5b-agent-deployment

---

Repo Link: https://github.com/osamatech786/5-Day-AI-Agents-Intensive-Course-with-Google