➡️ Next Steps for Execution

To move from this proposed stack to an actionable plan, consider structuring your agent development into four phases:

1. Discovery & Design (Gemini & NotebookLM)

- **Goal:** Define the problem, target user, and agent functionality.
- **Action:**
    - Use **Gemini** to analyze existing data, user feedback, or market trends to **synthesize the core value proposition** and **identify necessary tools/APIs**.
    - Create a **PRD-lite** in **NotebookLM**, citing all source documents (e.g., Gemini's output, design docs) to solidify requirements and tool mandates.

2. Prototyping & Composition (AgentSpace)

- **Goal:** Quickly build and validate the agent's core logic.
- **Action:**
    - Use **AgentSpace** to model the **autocomposed taskflow** based on the PRD-lite (from NotebookLM).
    - Integrate and test the specific tool choices identified by the initial Gemini analysis within AgentSpace's testing framework.

3. Asset Management & Centralization (Drive/Docs)

- **Goal:** Establish a single source of truth for all project materials.
- **Action:**
    - Store all configuration files, non-code assets, test data, and the final PRD/documentation in a dedicated **Drive** folder. Use **Docs** for human-readable guides and handover materials.

4. Monitoring & Iteration (Looker Studio)

- **Goal:** Ensure the agent is meeting its performance and adoption goals.
- **Action:**
    - Set up a **Looker Studio** dashboard connected to the agent's logs or usage data (e.g., from Google Analytics or internal databases) to track: **start-up time**, **tool selection accuracy**, and **user success rates**. This provides the necessary feedback to improve consistency.

That's an excellent question! The "how to do it" depends on which part of the solution you are trying to implement, as it involves multiple products.

Here is a step-by-step guide on how to implement your proposed solution stack for developing an AI agent, focusing on the integration of these tools.

The Agent Development Workflow: Step-by-Step

Phase 1: Discovery, Synthesis, and Requirements (Gemini & NotebookLM)

Goal: Define *what* the agent needs to do and *how* it should do it.

1. **Use Gemini for Discovery and Pattern Synthesis:**
    - **Input:** Provide Gemini with raw inputs (e.g., existing user feedback, operational logs, a description of the desired outcome, and internal documents about the business process).
    - **Prompt:** "Analyze this data to identify the top 5 pain points in our current process. Based on these, synthesize the optimal task flow for a new AI agent and recommend the specific external tools or APIs it would need to use."
    - **Output:** Get a clear, prioritized list of agent capabilities and a suggested **tool list/pattern blueprint**.
2. **Use NotebookLM for PRD-lite Citation:**
    - **Create Source:** Upload the detailed output from Gemini (the blueprint and tool list) into **NotebookLM** as a source document.
    - **Generate PRD-lite:** Ask NotebookLM to generate a "Product Requirements Document (PRD)-lite" for the agent, using the Gemini output as the core source.
    - **Refine & Cite:** As you refine the requirements, use NotebookLM's citation feature to ensure every requirement for the agent's logic or tool use is directly traceable back to the initial synthesis from Gemini. **This solves the "unclear tool choice" problem with clear documentation.**

Phase 2: Rapid Composition and Testing (AgentSpace)

Goal: Quickly build and validate the agent's logic and tool chaining.

1. **Design the Taskflow:**
    - In **AgentSpace** (or the equivalent Google environment for agent orchestration), start a new project.
    - **Input Requirements:** Import or reference the PRD-lite created in NotebookLM.
    - **Autocompose Taskflow:** Use AgentSpace's capabilities to map the required steps (defined in the PRD-lite) into a sequential or branched workflow. This is

    where you connect the recommended external tools/APIs from the Gemini analysis.
  - *Example Taskflow:* `[Trigger] -> [Analyze Input (Gemini)] -> [Decide Tool (AgentSpace)] -> [Call External Tool 1] -> [Format Output] -> [Complete]`.
2. **Develop and Automate Tests:**
   - **Write Tests:** Create unit and integration tests within **AgentSpace** that validate the **consistency** of the tool choice and the **accuracy** of the task flow.
   - *Focus:* Ensure that for a given input, the agent consistently chooses the correct tool and completes the task in the designed time. **This directly solves the "slow and inconsistent start-up" problem by providing immediate testing feedback.**

Phase 3: Asset Management and Deployment (Drive/Docs)

Goal: Centralize and secure all project artifacts.

1. **Centralize Assets in Drive:**
   - Create a single Google Drive folder for the agent project.
   - **Store Non-Code Assets:** Place training data, configuration files (JSON, YAML), sensitive access keys (if applicable and securely managed), and documentation here.
   - **Centralization:** Store the final, approved PRD (exported from NotebookLM) and the AgentSpace composition log in this folder. **This solves the "scattered pattern" problem.**
2. **Create Documentation in Docs:**
   - Develop a **handover/onboarding document** in Google Docs.
   - Include links to the AgentSpace project, the Looker Studio dashboard (once built), and the key assets in Drive.

Phase 4: Monitoring and Iteration (Looker Studio)

Goal: Measure performance and consistency for continuous improvement.

1. **Connect Data Sources:**
   - Ensure your deployed agent logs relevant data (e.g., task duration, tool selected, success/failure status) to a database or Google Sheet/BigQuery.
   - **Connect:** Open **Looker Studio** and connect it to the agent's logging database.
2. **Build the Adoption Metrics Dashboard:**
   - **Key Charts:** Create visualizations that directly address your initial problems:

        - **Consistency:** Chart the *latency* and *variance* (inconsistency) of the agent's completion time.
        - **Tool Choice:** Use a pie chart or bar graph to track which tools are being selected and the *success rate* per tool.
        - **Adoption:** Track the number of successful tasks per day/week.
    - **Feedback Loop:** Use this dashboard to identify new inconsistencies or underperforming tool choices, and feed that insight back into Phase 1 (Gemini) for a new cycle of synthesis and improvement. **This ensures ongoing consistency and clarity.**

Usecase 1: Automated Customer Support Triage Agent

Initial Problem	How the Solution Stack Solves It
Slow/Inconsistent Start	AgentSpace quickly composes and tests the triage logic.
Scattered Pattern	Docs/Drive hold the NLP models and conversation flows.
Unclear Tool Choice	Gemini analyzes ticket data to dictate the exact API (Tool) needed for each category (e.g., Jira vs. Salesforce).

Export to Sheets

The Workflow

1. **Phase 1: Discovery (Gemini & NotebookLM)**
    - **Gemini:** Analyzes thousands of historical support tickets. It synthesizes a pattern: **30% are billing issues, 60% are technical bugs, and 10% are feature requests.** It recommends the agent use the **Stripe API** for billing, the **Jira API** for bugs, and the **Salesforce Service Cloud API** for feature requests.

    - **NotebookLM:** A "Triage Agent PRD-lite" is created, citing Gemini's analysis as the source requirement for using the three specific APIs.
2. **Phase 2: Composition (AgentSpace)**
    - **AgentSpace:** The taskflow is autocomposed:
        - Receive Support Chat/Email.
        - Call Gemini (Internal Tool) for initial categorization (Billing, Bug, or Feature).
        - **Conditional Tool Choice (Solves "Unclear Tool Choice"):** If Billing, call Stripe API; If Bug, call Jira API; If Feature, call Salesforce API.
        - Create and tag the ticket in the relevant system.
    - Tests are run to ensure that 100 sample billing queries consistently result in the agent choosing the **Stripe API** and creating a tagged ticket.
3. **Phase 3: Monitoring (Looker Studio)**
    - **Looker Studio:** A dashboard tracks the agent's performance.
        - **Metric:** "Average time to tag ticket" (focusing on consistency).
        - **Insight:** If the agent's time to tag a "Bug" ticket is spiking, the team investigates the Jira API connection, not the agent's logic, leading to a focused fix.

That's a great request! To make the proposed solution stack more concrete, here are two sample use cases that demonstrate how the combination of **Gemini, NotebookLM, AgentSpace, Drive/Docs, and Looker Studio** addresses the initial problems (slow start, scattered process, unclear tools).

Usecase 1: Automated Customer Support Triage Agent

Initial Problem	How the Solution Stack Solves It
Slow/Inconsistent Start	AgentSpace quickly composes and tests the triage logic.
Scattered Pattern	Docs/Drive hold the NLP models and conversation flows.
Unclear Tool Choice	Gemini analyzes ticket data to dictate the exact API (Tool) needed for each category (e.g., Jira vs. Salesforce).

Export to Sheets

The Workflow

1. **Phase 1: Discovery (Gemini & NotebookLM)**

    - **Gemini:** Analyzes thousands of historical support tickets. It synthesizes a pattern: **30% are billing issues, 60% are technical bugs, and 10% are feature requests.** It recommends the agent use the **Stripe API** for billing, the **Jira API** for bugs, and the **Salesforce Service Cloud API** for feature requests.
    - **NotebookLM:** A "Triage Agent PRD-lite" is created, citing Gemini's analysis as the source requirement for using the three specific APIs.
2. **Phase 2: Composition (AgentSpace)**
    - **AgentSpace:** The taskflow is autocomposed:
        - Receive Support Chat/Email.
        - Call Gemini (Internal Tool) for initial categorization (Billing, Bug, or Feature).
        - **Conditional Tool Choice (Solves "Unclear Tool Choice"):** If Billing, call Stripe API; If Bug, call Jira API; If Feature, call Salesforce API.
        - Create and tag the ticket in the relevant system.
    - Tests are run to ensure that 100 sample billing queries consistently result in the agent choosing the **Stripe API** and creating a tagged ticket.
3. **Phase 3: Monitoring (Looker Studio)**
    - **Looker Studio:** A dashboard tracks the agent's performance.
        - **Metric:** "Average time to tag ticket" (focusing on consistency).
        - **Insight:** If the agent's time to tag a "Bug" ticket is spiking, the team investigates the Jira API connection, not the agent's logic, leading to a focused fix.

Usecase 2: Internal Data Reporting Agent

Initial Problem	How the Solution Stack Solves It
Slow/Inconsistent Start	AgentSpace allows rapid testing of the data retrieval and aggregation sequence.
Scattered Pattern	Drive/Docs stores all the necessary SQL query templates and schema documents.
Unclear Tool Choice	Gemini defines the exact data sources (Tools) and the required sequence of aggregation steps.

Export to Sheets

The Workflow

1. **Phase 1: Discovery (Gemini & NotebookLM)**
    - **Gemini:** Is fed documentation about internal data warehouses (e.g., BigQuery, internal SQL servers) and common reporting requests (e.g., "monthly sales

    figures per region"). It synthesizes a clear pattern: To answer a specific report, the agent **must** query **BigQuery** first, then join the results with data from the **Internal MySQL DB**, and finally use a **Python notebook API** for visualization.
    - **NotebookLM:** The "Reporting Agent PRD-lite" mandates the sequence and the specific API tools based on Gemini's synthesis.
2. **Phase 2: Composition (AgentSpace)**
    - **AgentSpace:** The taskflow is built to reflect the sequential tool use:
        - Receive Report Request.
        - Call **BigQuery API** (Tool 1).
        - Wait for results.
        - Call **MySQL API** (Tool 2) with the results from Tool 1.
        - Call **Python Notebook Execution API** (Tool 3) to generate the final chart.
    - **Drive/Docs:** The standardized SQL templates and Python scripts required for these tools are stored in a central Drive folder, linked directly into the AgentSpace configuration (**Solves "Scattered Pattern"**).
3. **Phase 3: Monitoring (Looker Studio)**
    - **Looker Studio:** The dashboard focuses on the agent's efficiency and reliability.
        - **Metric:** "Average step duration" (e.g., how long the BigQuery step takes vs. the MySQL step).
        - **Insight:** If the MySQL step is identified as the bottleneck, the team knows to optimize the query template in **Drive**, rather than overhauling the entire agent workflow. This provides a surgical, **consistent** improvement process.