

Agentic Space Quest: בונים OmniApp, agent מולטימודלי עם Gemini ו-ADK

סדנה מעשית של Google Cloud: מאוואר עקבי, דרך מערכת multi-agent שמצביעה,
ועד agent שרואה ושומע בזמן אמת

Annie Wang (Google Cloud) · ובינר Cloud on Air של Google Cloud (לפי התמליל); פורסם מחדש ב-X על ידי
Morty · 3.10.2026 (מועד השידור המקורי לא צוין)

משך: כ-119 דקות · X

סיכום אוטומטי — (aibriefing.dev) AI Briefing

על המסמך הזה

זו הקלטה של סדנה מעשית בשידור חי בשם Agentic Space Quest, שהעבירה Annie Wang. לפי השקופית שלה היא AI Engineer ב-Google Cloud, ובכיתוב של הפוסט ב-X היא מוצגת כ-Senior Engineer. הסדנה רצה על Qwiklabs, בסביבת Google Cloud זמנית, ובנויה כמשחק: כל משתתף נוחת על כוכב לכת, יוצר לעצמו אוטאר, מזהה בעזרת צוות agents איפה נחת, ובסוף מדבר עם drone שמזהה אצבעות דרך המצלמה. לאורך כל הדרך המשתתפים מעתיקים קטעי קוד לתוך קבצי placeholder ב-Cloud Shell Editor.

הטענה המרכזית: רוב ה-agents היום עובדים בטקסט, נכנס טקסט ויוצא טקסט, וגם כשמכניסים להם תמונה או קול זה בדרך כלל עובר תמלול או כיתוב. OmniApp הוא האפליקציה שבאה אחרי זה. היא קולטת דרך כמה חושים (Perceive), חושבת (Reason), מגיבה בכמה ערוצים (Express), ובשיא שלה עושה את כל זה Live, במקביל ולא לפי תורות. חשוב לדעת: הכיתוב של הפוסט ב-X מתאר מדריך על AI Employee שעובד 24/7, עם loops ו-graphs. זה לא מה שיש בהקלטה. הנושאים האלה מוזכרים רק בחטף.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר.
- **2. מהו OmniApp:** ההגדרה, Reason, Perceive ו-Express, והמבנה של שלוש הרמות.
- **3. Level 0, Express:** אוטאר עקבי בעזרת chat session אחד, ואזהרת ה-prompt injection.
- **4. Level 1, Perceive + Reason:** שלושה agents מומחים, שרתי MCP בהתאמה אישית ומנוהלים, ParallelAgent, callbacks ו-state.
- **5. Level 3, Live:** Gemini Live API, full duplex, LiveRequestQueue ו-WebSocket.
- **6. מה עושים עכשיו:** צעדים מעשיים וטבלת שאלות פתוחות.
- **7. מילון מונחים:** המונחים באנגלית ומשמעותם.

הערת מקור המסמך הופק אוטומטית מהתמליל של ההקלטה ומהשקופיות שחולצו ממנה. שמות ומונחים עלולים להשתבש בתמלול האוטומטי, למשל Germany במקום Gemini או Jamnet ו-Drumline במקום Gemini Live, ובמקרים כאלה הם סומנו. הציטוטים מועתקים מילה במילה ונבדקו מול חותמת הזמן שלהם. לא נוסף שום מידע ממקורות חיצוניים. בתמליל חסרים כמה קטעים, בעיקר הפסקות וזמני המתנה לפריסה, וההקלטה נקטעת לפני הבדיקה הסופית של Level 3.

1. תקציר מנהלים

הסדנה מלמדת שלושה דפוסים שאפשר לקחת הלאה לכל פרויקט agents: עקביות בין יצירות בעזרת שיחה אחת, מערכת multi-agent מקבילית עם הצבעה וכלים שמחוברים דרך MCP, ו-agent שמזרים קול ווידאו בזמן אמת דרך ADK ו-Gemini Live API. אין כאן מספרי ביצועים או לקחים מ-production. זו הדגמה לימודית, עם כמה תובנות ארכיטקטוניות טובות בדרך.

- **OmniApp הוא agent עם גוף ולא רק עם מוח:** agent רגיל חווה את העולם. "as a transcript" [3:00]. OmniApp קולט, חושב ומגיב בכמה ערוצים, והגרסה החיה שלו עושה את זה במקביל [6:03].
- **עקביות של דמות מגיעה מהשיחה ולא מה-prompt:** קריאות API נפרדות ל-Nano Banana מחזירות כל פעם דמות אחרת. כשהפורטרט והאייקון נוצרים באותו chat session, התמונה הקודמת היא חלק מה-input של התור השני [19:50, 29:09].
- **קלט חופשי של המשתמש בתוך prompt הוא וקטור תקיפה:** הדוברת קוראת לזה variable injection ומודה שבמעבדה אין שכבת הגנה [32:11–33:44].
- **מכילים כלים פעם אחת ומחברים אותם דרך MCP:** ניתוח התמונה והווידאו נכתב כשרת FastMCP ונפרס ל-Cloud Run. ל-BigQuery משתמשים בשרת MCP מנוהל של Google, כדי לא לטפל בעצמכם באימות ובחיבור [51:22, 1:02:05].
- **שלושה agents מקבילים שמצביעים טובים מ-agent אחד:** בהרצה החיה שני מומחים זיהו אזור מאובנים (fossilized) ואחד bioluminescent. כלל רוב של 2 מתוך 3 הכריע [1:17:26].
- **state הוא הלוח המשותף, וה-callback הוא מי שכותב עליו:** ה-before_agent_callback טוען את נתוני המשתתף ל-session state, וה-agents וה-tools קוראים משם בזמן ריצה [1:12:51].
- **הכלי רואה את מה שהמודל לא רואה:** המודל רק בוחר biome וקורא לכלי האישור. ההשוואה מול הקואורדינטות האמיתיות נעשית בקוד דטרמיניסטי, דרך [1:12:51–1:14:24] ToolContext.
- **Live מחייב full duplex ואודיו טבעי:** pipeline של STT ואז LLM ואז TTS מוסיף השהיה בכל שלב. Gemini Live מעבד אודיו ישירות וקולט גם טון והיסוס [1:34:00–1:37:06].
- **הזרמה דו-כיוונית פירושה שני תהליכים שלא מחכים זה לזה:** LiveRequestQueue מפריד בין upstream ל-"downstream, sushi conveyor belt" בלשונה [1:40:11–1:43:14].

מה לא נאמר

- **אין AI Employee, loops או graphs:** הכיתוב ב-X מבטיח את כל אלה, אבל loop agent מוזכר רק כאחד מארבעה סוגי workflow agent ב-ADK 1.x, ו-graph workflow של ADK 2.0 מוזכר במשפט אחד [1:05:13].
- **אין מספרי latency או עלות ל-Live API:** המספר היחיד שמופיע הוא בשקופית, כ-\$0.08 לפורטרט ואייקון, וגם הוא לא נאמר בקול.
- **לא מוצג פתרון ל-prompt injection:** הדוברת מזכירה מוצר של Google לסינון הקלט, ששמו השתבש בתמלול (Model, כנראה Model Armor), ולא מדגימה אותו.
- **Level 2 לא מכוסה, ו-Level 3 לא מסתיים בהקלטה:** ההקלטה נקטעת אחרי הפעלת ADK Web, לפני שרואים את זיהוי האצבעות עובד מקצה לקצה.
- **אין הערכה (evaluation) של מערכת ההצבעה:** לא נאמר כמה פעמים שלושת המומחים טועים, ומה קורה כשאין רוב.

2. מהו OmniApp ולמה זה שונה מ-agent רגיל

WHY YOU SHOULD CARE

More than an AI agent.

You've seen AI agents that chat and call tools. An **omni app** is the next step — and the difference is bigger than it sounds.

A NORMAL AI AGENT

A brain in a text box

Reads text, reasons, calls a tool, writes text — turn by turn. Brilliant — but it experiences the world as a *transcript*.

AN OMNI APP

The same brain — with a body

Eyes, ears, a voice, and hands. It sees, hears, speaks, creates, and **acts** — natively, and in real time.

A normal agent right so that is usually how agent work worth your next three hours.

Google

השקופית "agent": More than an AI agent רגיל הוא מוח בתוך תיבת טקסט, ו-OmniApp הוא אותו מוח עם גוף, כלומר עיניים, אוזניים, קול וידיים, שפועל בזמן אמת

Wang פותחת בהדגמה חיה של המשימה הסופית: drone שמבקש "scan" ומזהה כמה אצבעות היא מרימה מול המצלמה [1:30]. אחר כך היא חוזרת לבסיס: agent ב-ADK מקבל prompt, המודל בוחר כלי, וה-framework מריץ אותו [3:00]. לטענתה, גם agent שנקרא "multi-modal" בדרך כלל רק הופך תמונה לכיתוב טקסטואלי וממשיך משם.

היא מציגה שלושה תרחישי שימוש: עוזר שמלווה תיקון של מכונת כביסה דרך המצלמה, יצירה מכל קלט לכל פלט ("Sending to X creation" בתמלול, כנראה any-to-X), למשל מדידה וירטואלית של נעליים, וצוות agents שמטפל בתביעת ביטוח. אחד מסתכל על תמונת הנזק, אחד בודק את הפוליסה ואחד את ההיסטוריה [4:31]. לכולם יש אותו שלד: Perceive, Reason, Express.

Annie Wang [6:03] So when the perceive and express stop taking turn and run at the same time, — and then it feels like you're not using the tool. It feels like you're actually talking to the person

Google Cloud

YOUR QUEST = YOUR ROADMAP

Three levels, one omni app

We'll learn it as a game: be seen → be found → go live. Each level adds one omni-app power — and teaches one hard problem.

LEVEL 0 - be seen

Express

build: a consistent identity
learn: multimodal generation

LEVEL 1 - be found

Perceive + Reason

build: a parallel agent crew
learn: orchestration + tools (MCP)

LEVEL 3 - go live

Live

build: a real-time, interruptible app
learn: bidirectional streaming

you building a new like superpower with AI

מפת הדרכים "Express": Three levels, one omni app (יצירת זהות), Perceive + Reason (ניתוח ראיות על ידי צוות agents) ו-Live (ראייה ושמיעה בזמן אמת)

3. Level 0, Express: אווטאר עקבי בעזרת chat session

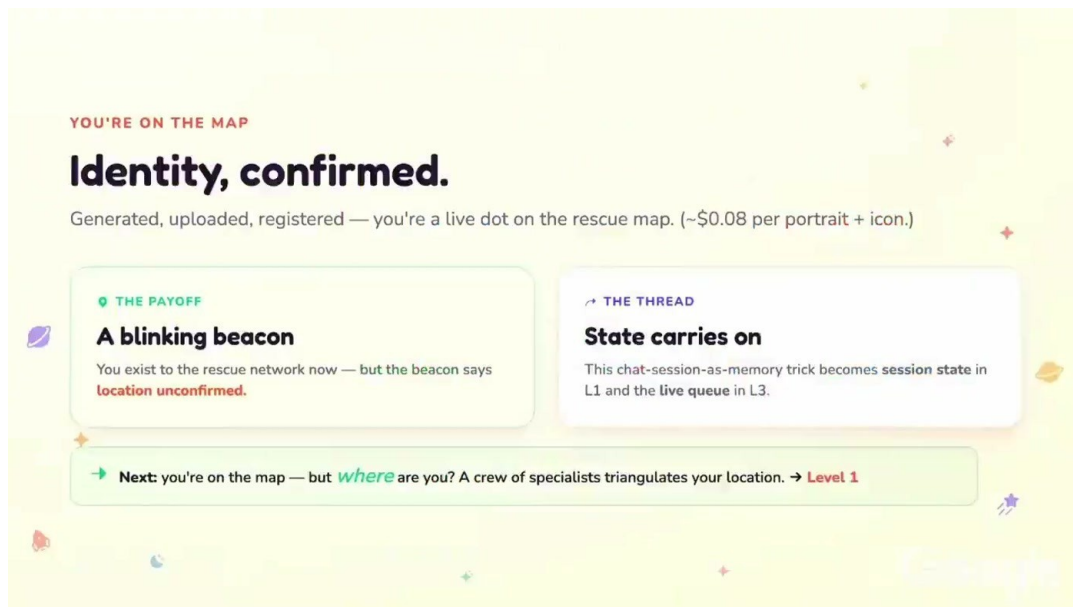
אחד

המשתתף מריץ setup.sh עם קוד האירוע space-quest, בוחר צבע חליפה ותיאור חופשי (Wang כתבה "cat hat", ומשלים ב-generator.py את יצירת השיחה. בתור הראשון נוצר פורטרט. בתור השני, באותה שיחה, נוצר אייקון עגול למפה, עם דרישות מפורשות כמו "friendly smile same person, same face, same expression, same suit" [24:30]. התוצאה עולה למפה משותפת ב-waybackhome.dev, שבה ראו בזמן הסדנה 15, אחר כך 21 ואחר כך 31 explorers [27:34, 43:36].

התובנה הטכנית: "If you're using a banana model and then if you just make independent API call, a model will treat each one as a brand new request." [19:50]. השם Nano Banana מופיע על המסך. בשיחה אחת, ה-input של התור השני כולל את הטקסט וגם את התמונה שכבר נוצרה [29:09]. כחלופות להמשך היא מזכירה fine-tuning ו-reference images, שדורש יותר משאבים אבל מניב תוצאה חזקה וקלה להרחבה [32:11].



"Chat Sessions Unlock Character Consistency": קריאות נפרדות מייצרות דמויות אקראיות, ו-chat session משותף שומר על זהות אחת. בתחתית ההגדרה הקריטית ["TEXT", "IMAGE"] response_modalities =



"Identity, confirmed": סיכום Level 0 עם העלות המשוערת (כ-0.08\$ לפורטרט ואייקון) והחוט המקשר: הטריק של זיכרון בתוך session הופך ל-session state ב-Level 1 ול-live queue ב-Level 3

Annie Wang [32:11] However, there is a safety concern in this whole behavior, and it has a name — called variable injection

4. Level 1, Perceive + Reason: שלושה מומחים, MCP והצבעה



"The multi-agent system": MissionAnalysisAI כ-root, מתחתיו שלושה מומחים (Geological, Botanical, Astronomical), ובשכבת הכלים שרת FastMCP מותאם, function tool לחילוף מאפייני כוכבים ו-BigQuery MCP

התרחיש: על הכוכב יש ארבעה אזורים, cryo, volcanic, bioluminescent ו-fossilized. כל משתתף מקבל ראיות שנוצרו בהתאם לקואורדינטות שלו: תמונת קרקע, וידאו של צמחייה עם אודיו, ושדה כוכבים [40:29, 46:46]. שלושה מומחים מנתחים את הראיות במקביל ומצביעים.

הכלים בנויים בשלוש דרכים. analyze_geological ו-analyze_botanical נכתבו כשרת FastMCP עם decorator של tool, ושניהם קוראים ל-Gemini 2.5 Flash על התמונה או הווידאו [52:55]. השרת נבדק מקומית ונפרס ל-Cloud Run דרך קובץ cloudbuild.yaml. לשרת הזה אין UI, רק URL לחיבור [54:26, 59:02]. המומחה

האסטרונמי משתמש ב-function tool מקומי ובשרת MCP מנוהל של Google ל-BigQuery, כי בגישה ישירה "you".need to worry about how to connect, how to authenticate the security, and more" [1:02:05]

DECISION GUIDE

Custom vs. managed vs. MCP Toolbox

Three MCP patterns — when to use each and how to combine them in production

DIMENSION	CUSTOM (FASTMCP/CLOUD RUN)	MANAGED (GOOGLE)	MCP TOOLBOX
Who builds	You	Google	Open source (pre-built)
Who hosts	Your Cloud Run service	Google's infra	You (self-hosted)
Auth	Any (API key, IAM...)	Google OAuth2 / ADC	Your config
Best for	Custom AI logic domain-specific models	Standard GCP services BigQuery, GCS, Spanner	Databases & APIs Postgres, MySQL, Redis
Level 1 example	analyze_geological, analyze_botanical	BigQuery execute_query	—

Real production systems use all three: managed MCP for cloud services, MCP Toolbox for databases, custom MCP for your unique AI-powered tools.

מדריך החלטה "Custom vs. managed vs. MCP Toolbox": מי בונה, מי מארח, איך מתבצע האימות ולמה כל אפשרות מתאימה, עם ההערה ש-production אמיתי משלב את שלושתן

התזמור: "1:05:13] ב-ADK 1.x. This lab currently is written with ADK 1.x יש ארבעה workflow agents: Sequential, Parallel, Loop, ו-agent LLM שמאציל משימות ל-sub-agents. ב-ADK 2.0 יש graph workflow, אבל הוא לא חלק מהמעבדה. שלושת המומחים נעטפים ב-ParallelAgent בשם EvidenceAnalysisCrew. מעליו יושב root orchestrator, שהוא LLM agent רגיל עם confirm_location_tool [1:06:43–1:08:15] ו-befor_agent_callback=setup_participant_context

ה-callback רץ פעם אחת, שולף את נתוני המשתתף וכותב אותם ל-state. ה-agents משתמשים בהם בהוראות שלהם דרך placeholders כמו {soil_url}. Wang מסבירה גם למה עדיין מעתיקים קוד ידנית בעידן שבו AI כותב קוד: "I think we're still important, good to know all those mental model and why and how so that you can write better code and help design a better system." [1:08:15]

Google Cloud



STATE

{state} is a contract – not an f-string.

```
agent.py
# middleware: runs once, BEFORE the LLM - deterministic fetch
def setup_participant_context(ctx):
    ctx.state["soil_url"] = fetch_evidence(participant) # HTTP GET, no LLM
    return None

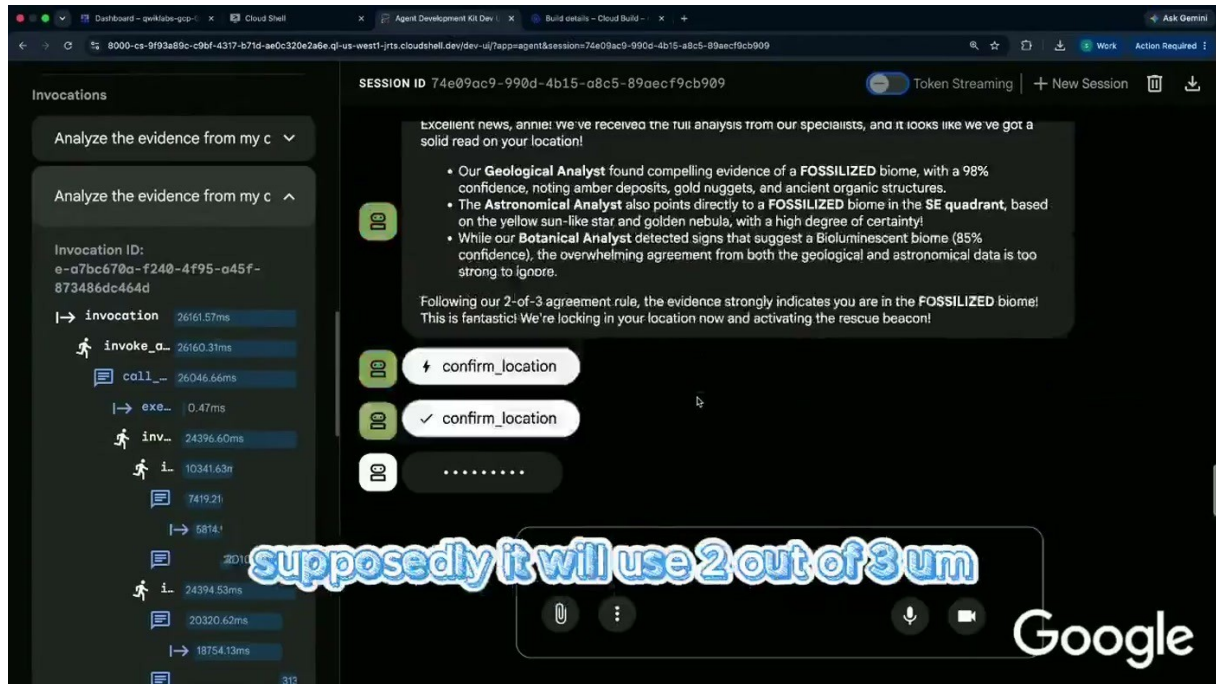
# the specialist's instruction - looks like an f-string, ISN'T
instruction = "Analyze the soil at {soil_url}." # ADK fills {soil_url} at INVOCATION
```

The gotcha: {soil_url} is resolved by ADK when the agent runs — not at Python import. The callback guarantees the state shape; the agent reads it. That's the contract.

"{state} is a contract, not an f-string {state}" ה-placeholder {soil_url} נפתר על ידי ADK בזמן ריצת ה-agent ולא בזמן ה-import של Python, וה-callback מבטיח שהערכים יהיו ב-state לפני כן

Annie Wang [1:12:51] So the order is the callback writes a state first, and then the agent reads a **—**.state at the runtime

נקודת התכנון העדינה ביותר בסדנה: המודל רק מחליט על ה-biome וקורא לכלי האישור, ו-"the model is not able to see anything below it" [1:14:24]. הכלי ניגש דרך ToolContext ל-state ומשווה את התשובה לקואורדינטות האמיתיות, שהמודל לא רואה אף פעם. השקופית קוראת לזה Verifiable AI: החלטה הסתברותית שנבדקת מול ground truth דטרמיניסטי. באגים מהשטח: ההרצה הראשונה ב-ADK Web נכשלה ועבדה בניסיון השני [1:15:56], ואם MCP_SERVER_URL ריק, כנראה שכחתם לעשות לו [1:14:24].export

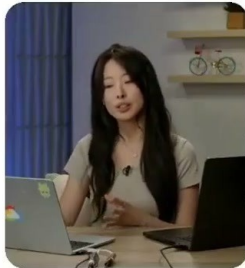


ממשק ADK Web בזמן ההרצה החיה: כל מומחה מחזיר ממצא ורמת ביטחון (fossilized 98% Geological), ה-root מיישם את כלל 2 מתוך 3, קורא ל-confirm_location ומפעיל את משואת החילוף

Annie Wang [1:17:26] that if you only rely on one agent in my making some mistakes so have the **—** parallel agent you have many people voting so it's more likely to make a better decision right

5. Gemini Live API דו-כיווני עם Level 3, Live: agent

Google Cloud



Traditional Architecture: The blocking Pipeline

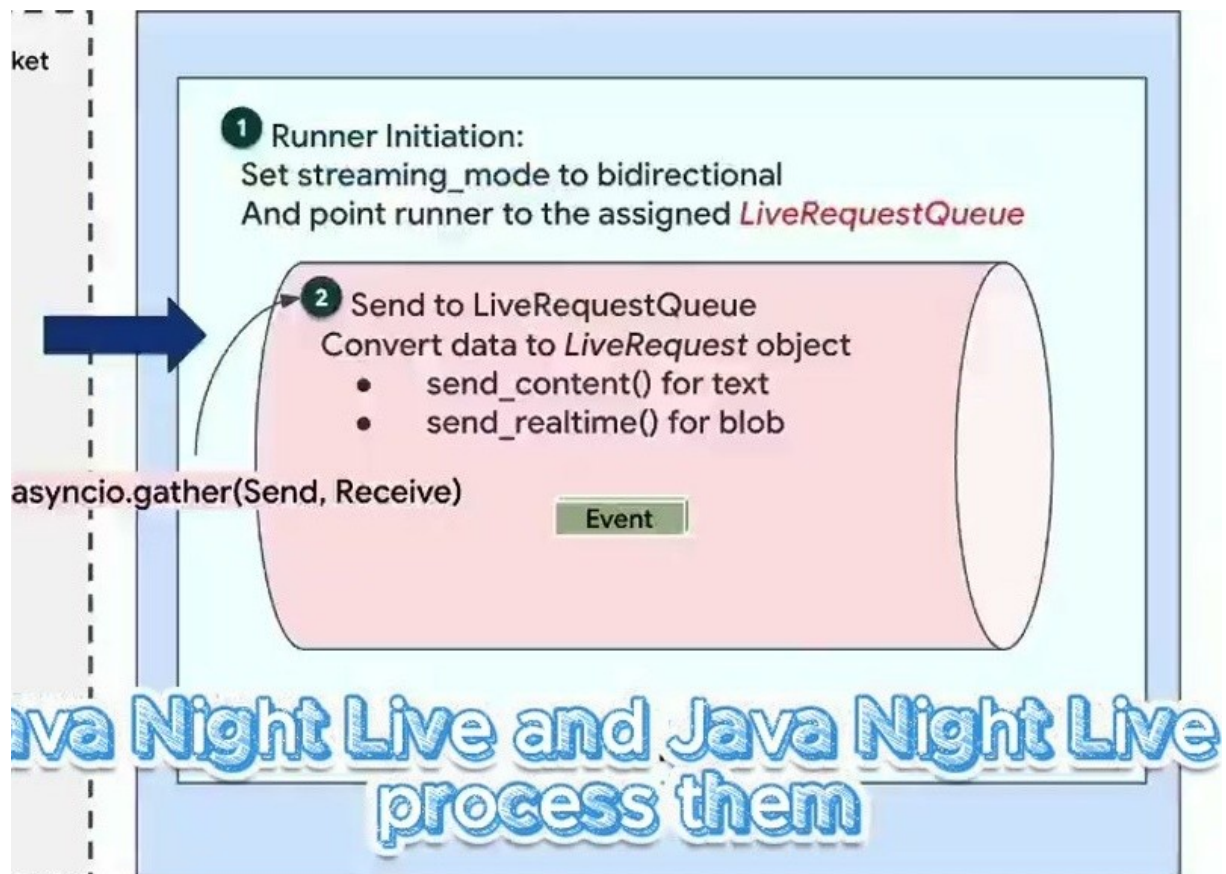


very simple concept like a voice simple voice agent

"Traditional Architecture: The Blocking Pipeline": אודיו מהמשתמש, אחריו STT, אחריו LLM ואחריו TTS, וכל שלב מוסיף עוד השהיה עד שהתשובה מגיעה

הארכיטקטורה: frontend ב-React, שהוא ה-cockpit, לוכד וידאו מהמצלמה ואודיו מהמיקרופון. backend ב-FastAPI ו-ADK מנהל את הלוגיקה, ו-Gemini Live API הוא המוח. ביניהם יש חיבור WebSocket דו-כיווני [1:32:26]. ה-pipeline המסורתי " [1:34:00] each step is adding time, it's adding latency". בטקסט לא מרגישים את זה, בשיחה קולית כן.

ההבחנה המרכזית היא בין half duplex, מכשיר קשר שבו מדברים לפי תורות, לבין full duplex: "Full duplex is like a phone call." [1:35:33]. Gemini Live It will process audio" [1:37:06]. לכן הוא קולט גם טון, היסוס והתלהבות, ולא רק מילים. הקלט נכנס בשני סוגים: send_realtime (אודיו בערך כל 16 מילישניות, ופריים מהמצלמה כל חצי שנייה) ו-send_content. לפעולות בדידות כמו טקסט או snapshot, שמפעילות reasoning מיד [1:40:11–1:41:43, 1:52:23].

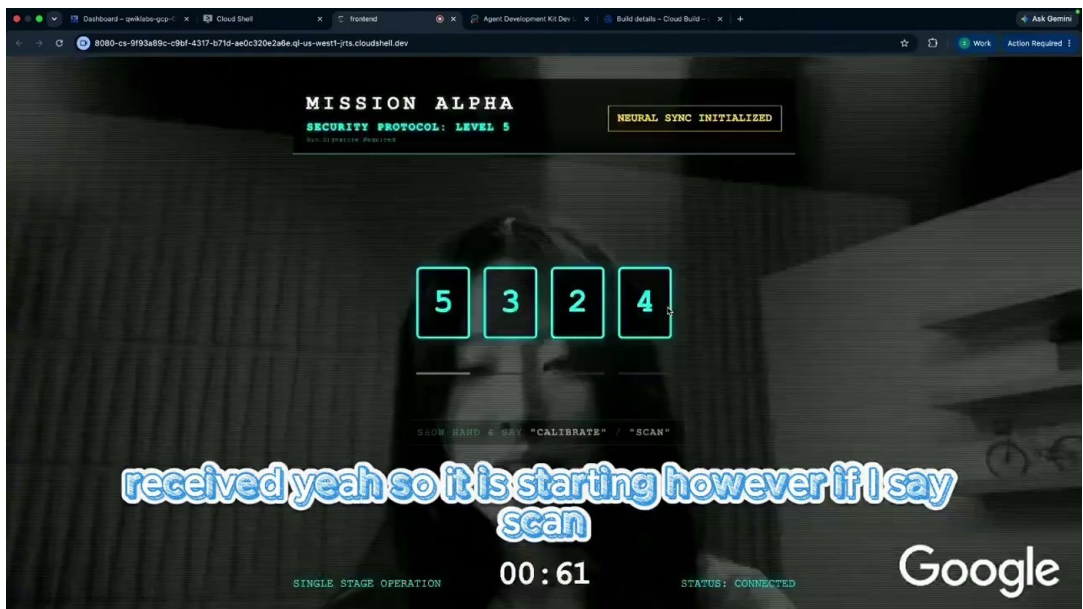


The Execution Logic": WebSocket" מקבל נתונים, הם מומרים לאובייקט LiveRequest (send_content לטקסט, send_realtime ל-blob) ונכנסים ל-LiveRequestQueue, ה-Runner רץ במצב streaming דו-כיווני, ושליחה וקבלה רצות במקביל דרך asyncio.gather

Annie Wang [1:43:14] So the upstream never wait for the downstream and the downstream never —
.wait for the upstream. That is a bi-directional streaming in action

החלק המעשי: ב-useGeminiSocket.js משלימים handler שמבחין בין function call, שמעדכן את המנעול לפי מספר האצבעות, לבין אודיו שמושמע. אחר כך משלימים את לכידת המצלמה והמיקרופון. בדיקה מול mock server מודאת שה-WebSocket עובד לפני שיש agent. בצד ה-agent, biometric_agent, backend מקבל הוראות ("wait until this voice trigger"), מודל Live ממשפחת Gemini 2.5 Flash עם native audio, כפי שמופיע על המסך, וכלי .report_digit [1:55:27–1:56:57]

כמה דברים מהשטח: אחרי כשעה ה-token של Cloud Shell פג ונדרש רענון [1:44:44]. ב-pyproject יש websocket כפול שצריך להסיר [1:47:49]. המודל במעבדה מיושן, "but you can using latest model too" [1:55:27]. ההקלטה נקטעת בזמן הפעלת ADK Web, ולכן ההדגמה המלאה של Level 3 לא מופיעה בה.



סך Mission Alpha, ממשק ה-cockpit שהשתתפים בו: רצף הספרות שהמערכת מבקשת לאשר בתנועות יד מול המצלמה, ומונה זמן

6. מה עושים עכשיו

- אם אתם מייצרים תמונות סדרתיות: העבירו את כל היצירות של אותה ישות לתוך chat session אחד עם response_modalities של טקסט ותמונה, במקום קריאות API נפרדות.
- אם קלט חופשי של משתמש נכנס ל-prompt: הוסיפו שכבת סינון לפני המודל. במעבדה הזו אין שכבה כזו, והדוברת אומרת את זה במפורש.
- בנו כלים כשרתי MCP ולא כקוד שמוטמע ב-FastMCP: agent ל-logic שלכם, ושרת MCP מנוהל לשירותי ענן כמו BigQuery.
- כשהחלטה חשובה, הריצו כמה מומחים במקביל: ParallelAgent ו-aggregator עם כלל רוב מפורש. בדקו מה קורה כשאין רוב, כי זה לא הודגם.
- השאירו את ה-ground truth מחוץ למודל: המודל מחליט, וכלי דטרמיניסטי עם גישה ל-state מאמת. זה דפוס שכדאי להעתיק.
- ל-agent קולי, התחילו מ-full duplex: WebSocket, LiveRequestQueue ו-Runner במצב BIDI, עם בדיקה מול mock server לפני שמחברים מודל.
- כדי להתנסות בעצמכם: המעבדה זמינה כ-Google Cloud Labs: Agentic Space Quest, עם קוד אירוע space-quest או sandbox ללמידה עצמית.

שאלה פתוחה	למה זה חשוב
מה קורה במערכת ההצבעה כששלושה מומחים חלוקים לגמרי?	כלל 2 מתוך 3 לא מגדיר מה עושים כשכל אחד אומר משהו אחר
מה ה-latency בפועל של Gemini Live לעומת pipeline של STT, LLM ו-TTS?	הטענה המרכזית של Level 3 נשענת על השהיה, ולא הוצג אף מספר
איך משרדים את המעבדה ל-ADK 2.0 graph workflow?	הוזכר שזו הגרסה החדשה, אבל המעבדה כתובה ב-ADK 1.x
איך מגינים בפועל מפני variable injection?	מוצר הסינון הוזכר בשם משובש ולא הודגם
כמה עולה הרצה של Level 1 ושל Level 3?	מספר העלות היחיד הוא בשקופית של Level 0, בערך \$0.08

7. מילון מונחים

מונח	משמעות
OmniApp	השם שהדוברת נותנת לאפליקציה שקולטת, חושבת ומגיבה בכמה ערוצים, ובגרסה החיה עושה את זה במקביל
ADK (Agent Development Kit)	ה-framework של Google לבניית agents. המעבדה כתובה ב-ADK 1.x

מונח	משמעות
MCP / FastMCP	פרוטוקול שמחבר agents לכלים. FastMCP היא ספריית Python לבניית שרת MCP
Google-managed MCP	שרת MCP ש-Google מנהלת עבור שירותי הענן שלה, כמו BigQuery, בלי שתצטרכו לטפל באימות
ParallelAgent	workflow agent שמריץ כמה sub-agents במקביל
before_agent_callback	hook שרץ לפני ה-agent ומאפשר להכין context, למשל לכתוב ל-state
Session state	לוח משותף שה-agents וה-tools קוראים ממנו וכותבים אליו
ToolContext	האובייקט שדרכו כלי ניגש ל-state, כולל מידע שהמודל לא רואה
Cloud Run / Cloud Build	שירות הריצה המנוהל לקונטיינרים, ושירות הבנייה שפורס אליו
Nano Banana	השם המסחרי של מודל יצירת התמונות של Gemini, שהיה בשימוש ביצירת האווטאר
Gemini Live API	API להזרמה דו-כיוונית של אודיו ווידאו, עם עיבוד אודיו ישיר (native audio)
Half / Full duplex	תקשורת לפי תורות, כמו מכשיר קשר, לעומת תקשורת שבה שני הצדדים מדברים בו בזמן, כמו שיחת טלפון
WebSocket	חיבור דפדפן-שרת שנשאר פתוח לתנועה דו-כיוונית רציפה
LiveRequestQueue / Runner	תור FIFO שמפריד בין קלט לפלט, והמנוע של ADK שמנהל את מחזור החיים של האירועים
send_realtime / send_content	שליחת זרם רציף (אודיו או וידאו) לעומת שליחת פעולה בדידה שמפעילה reasoning מיד
Variable / prompt injection	החדרת הוראות זדוניות דרך קלט של המשתמש שמשולב ב-prompt