ה-Infra Stack של Software Factory לשנת 2026 — Zach Lloyd, Warp

איך מגדירים פקטורי של coding agents בקוד, מודדים אותו, ומורידים את העלות ל-PR מ-80 דולר ל-15–20 דולר

בשיחה עם Zach Lloyd (Warp) · 3 Latent Space · swyx בנובמבר 2026
משך: כ-35 דקות · YouTube
סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה פרק של הפודקאסט Latent Space, שהוקלט מרחוק. המארח swyx משוחח עם Zach Lloyd, המייסד וה-CEO של Warp, לקראת כנס AIE CODE בסן פרנסיסקו (10–12 בנובמבר 2026). Warp כבר לא מציגה את עצמה בעיקר כטרמינל. היא ממצבת את עצמה כ-open infrastructure ל-cloud software factories, כלומר מערכת שמקבלת משימה מ-Slack, מ-Linear או מ-GitHub ומחזירה pull request. ב-18 הדקות הראשונות Lloyd מראה את הפלטפורמה בדמו חי: הגדרת הפקטורי, traces של agents, מדדי עלות ו-throughput, scorers, self-improvement ו-benchmarks. אחר כך הוא עובר לשרטט stack ארכיטקטוני כללי, ובסוף מתווכחים על כלכלת ה-subscriptions ועל השאלה למי כל זה מתאים בכלל.

הטענה המרכזית: software factory הוא לא אוסף רופף של automations ו-agents אינטראקטיביים. זה דבר מוגדר שצריך לתאר באופן declarative בקוד, כמו Terraform. אחרי שכל העבודה עוברת דרך מערכת אחת, אפשר לאסוף את כל השיחות והתוצרים של ה-agents, למדוד אותם, להריץ עליהם evals ו-benchmarks על משימות אמיתיות של החברה, ולתת ל-agents עצמם להציע שינויים להגדרת הפקטורי. Lloyd קורא לזה מעבר מ-vibes ל-measurement ול-"factory engineering". את הטענה הוא מגבה בנתונים פנימיים של Warp, כמו העלות ל-PR, ה-throughput ו-revert rate.



פתיחת הפרק: swyx (בכיתוב: Host & Editor of Latent Space) משמאל, Zach Lloyd מ-Warp מימין. השיחה הוקלטה מרחוק

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם timestamps, ומה שלא נאמר או לא הוכח.
- **2. מה זה software factory:** ההגדרה של Lloyd, factory-as-code, אחד לכל product surface, ושתי הפרסונות.
- **3. מ-Slack ל-PR, ומדידה:** traces מרוכזים, מדדי DORA, אוטומציה, עלות ו-scorers מסוג LLM-as-judge.
- **4. Self-improvement ו-benchmarks:** agents שפותחים PR נגד הגדרת הפקטורי, והרצה חוזרת של משימות אמיתיות מול מטריצה של מודלים ו-harnesses.
- **5. ה-stack בשש שכבות:** מ-Factories-as-code ועד Access points, והמקרה של Warp: מ-80 דולר ל-15–20 דולר ל-PR.
- **6. למי זה מתאים:** סף של 5–10 מהנדסים, כלכלת ה-subs, ושני סוגי המהנדסים.
- **7–8. מה עושים עכשיו, ומילון מונחים:** צעדים מעשיים, טבלת שאלות פתוחות ומונחים.

הערת מקור המסמך נוצר אוטומטית מתמליל ההקלטה ומהשקופיות שהופיעו בה. התמליל הוא ASR, ולכן שמות ומונחים עלולים להיות משובשים. מקרים כאלה מסומנים במפורש, למשל Cloud Code שהוא כנראה Claude Code, ושמות המודלים בטבלת ה-benchmark. הציטוטים מועתקים מילה במילה מהתמליל ונבדקו מול ה-timestamp שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

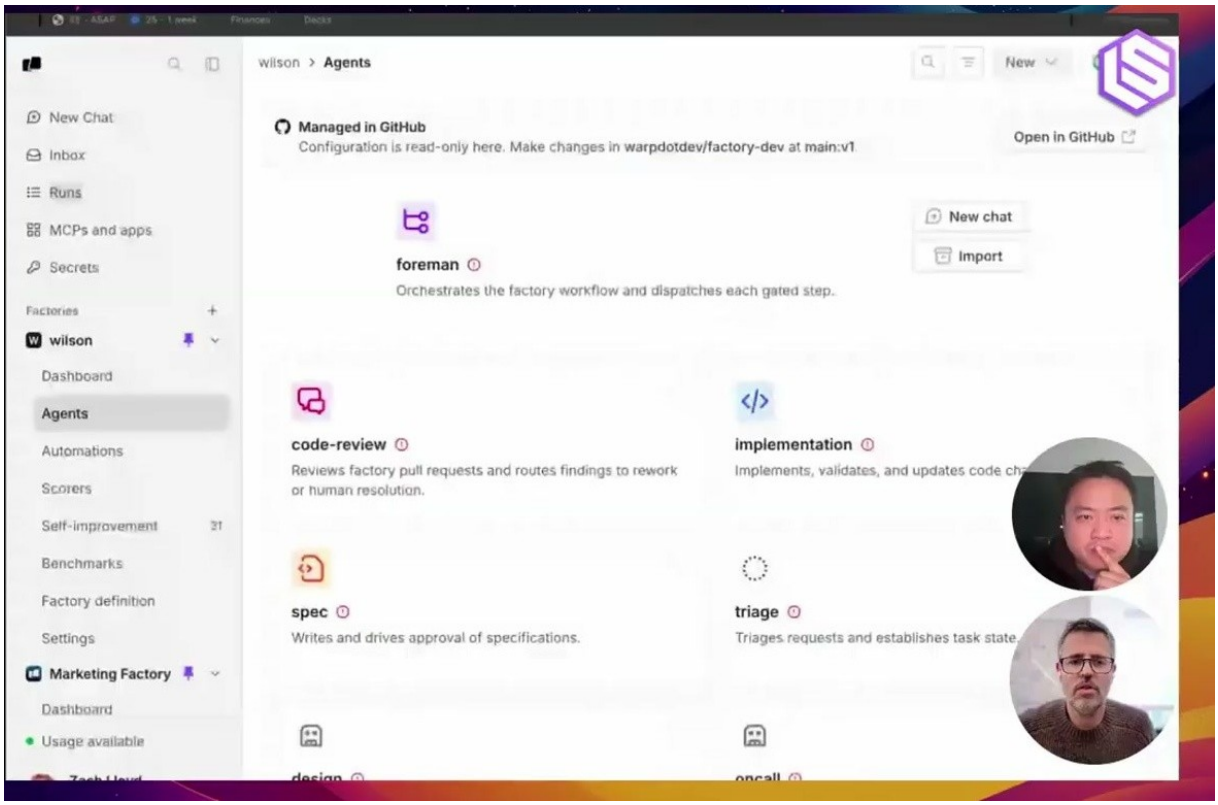
Lloyd מציג את ה-software factory כשכבה חדשה ב-"one level back ,developer infrastructure" מ-CI/CD. לדבריו, הערך לא מגיע מה-agent הבודד. הוא מגיע מכך שכל העבודה של ה-agents בארגון עוברת דרך מערכת אחת שמוגדרת בקוד, נמדדת ומשתפרת בלולאה סגורה. רוב הראיות הן דמו חי ונתונים פנימיים של Warp, ואין כאן מחקר בלתי תלוי.

- **Factory צריך להיות מוגדר בקוד:** קובץ YAML שמגדיר repos, skills, MCPs ו-workflow (triage, spec, implementation, code review). יש factory אחד לכל product surface, כמו mobile או server.
- **נקודת כניסה אחת היא התנאי לכל השאר:** משימה נכנסת מ-Slack, Linear, Jira, GitHub, API או MCP, ובכל המקרים עוברת דרך מערכת אחת. ב-Warp כ-60% מהעבודה מתחילה ב-Slack.
- **Traces מרוכזים הם חומר הגלם:** שיחה מלאה, צילומי computer use והפעולות של ה-orchestration agent, לכל משימה. רוב החברות לא מחזיקות דבר כזה כי כל מהנדס מריץ Claude Code או Codex מקומית.
- **מודדים עלות ל-PR, אוטומציה ואיכות:** מדדי DORA, כמה פעמים אדם נגע בכל PR שעבר merge, ועלות לפי אדם, לפי automation ולפי שלב ב-SDLC. האיכות נבדקת ב-scorers מסוג LLM-as-judge עם sampling rate.
- **הפקטורי משפר את עצמו:** agents מצאו 54 כשלים שבהם ה-triager היה צריך להריץ sub-agent, ופתחו PR נגד הגדרת הפקטורי.
- **Benchmark על משימות שלך ולא על SWE-bench:** מריצים מחדש משימות אמיתיות מול מטריצה של מודל ו-harness. בדוגמה, מודל אחד הגיע ל-95% נכונות ב-6 דולר בממוצע, ומודל אחר ל-93% ב-2 דולר.
- **Warp עצמה ירדה מ-80 דולר ל-15–20 דולר ל-PR:** בזכות פחות Opus במקומות הלא נכונים, מודל זול יותר ל-triage ופחות sub-agents. ה-throughput עלה ב-40% בחודש האחרון וה-revert rate לא עלה.
- **זה לא לכל אחד:** לצוות של כמה אנשים זה overkill, ועדיף subscription. הרף התחתון הוא בערך 5–10 מהנדסים, ולפי Lloyd תשלום לפי API יקר פי 5 עד 50 מ-subs.

מה לא נאמר

- **אין baseline לאיכות:** הירידה בעלות וה-40% ב-throughput מוצגים בלי נתוני איכות מעבר ל-revert rate, ובלי מספרים מוחלטים, למשל כמה PRs בחודש.
- **ה-scorers לא אומתו:** לא הוצג איך יודעים שה-LLM judge מדויק, ולא כמה עולה הרצה שלו בפועל.
- **לולאת השיפור עוד לא סגורה:** Lloyd אומר במפורש שהרצה חוזרת של runs כדי לבדוק שינוי שהוצע עוד לא קיימת. כרגע ה-PRs של self-improvement לא עוברים בדיקה כזו.
- **אין מחיר:** לא נאמר כמה עולה Warp Factories, ולא מה מתוך ה-stack הוא open source בפועל, מעבר ל-H1 בדף הנחיתה.
- **תוצאות ה-benchmark ישנות:** Lloyd עצמו אומר שהמודלים בגרף כבר מעט ישנים, ושמות המודלים משובשים בתמליל.

2. מה זה software factory, ולמה להגדיר אותו בקוד



דמו: מסך Agents בפלטפורמת Warp. ההגדרה מסומנת Managed in GitHub ומוצגת לקריאה בלבד, ושינויים נעשים ב-repo של הגדרת הפקטורי. מוצגים agents בתפקידים foreman (orchestration), code review, implementation, spec ו-triage. בתפריט הצד: Automations, Scorers, Self-improvement, Benchmarks ו-Factory definition, ופקטורי נוסף בשם Marketing Factory

ההגדרה של Lloyd: מערכת agentic בענן, שבה מגדירים את התפקידים של agents שונים לאורך בניית התוכנה. יש לה מידה של אוטונומיה, והיא צריכה להיות "very measurable, very closed loop". הוא אומר שמה שחברות מכנות היום "הפקטורי הפנימי שלנו" הוא בדרך כלל שרשור של automations, או סתם כינוי לכך שמשתמשים הרבה ב-agents. העמדה של Warp היא שפקטורי הוא דבר קונקרטי שאפשר להריץ ולמדוד בכל רגע נתון, ושצריך פקטורי אחד לכל product surface.

כל פקטורי עובד על סט של repos ויש לו skills, MCPs ו-workflow. ה-workflow אמור לשקף את מה שהצוות כבר עושה היום: triage, specking, implementation ו-code review. ב-Warp כל זה מוגדר בקובץ YAML. חשוב לשים לב לשתי הפרסונות. מי שעושה product engineering לא נכנס בכלל לממשק של הפקטורי. הוא "מכניס work item לפקטורי" מתוך הכלים שהוא כבר עובד איתם, וזה מפעיל workflow מוגדר מראש.

> **Zach Lloyd** — Like you should define your factory in code. You should make sure that the factory has all the workflows, skills, MCPs, everything is like, it's like a concrete thing that you should be able to run at some point in time and measure how it's doing.

- **Product engineer:** מכניס עבודה דרך Slack, Linear, Jira, GitHub, API או MCP, ולא צריך לדעת איך הפקטורי בנוי.
- **Platform / DevX team, EM ו-CTO:** אלה המשתמשים של ה-dashboard. לדברי Lloyd זה בדרך כלל "not your day-to-day engineer", אלא הצוות שמנסה להבין איך להפיק את המקסימום מ-Claude Code או מ-Codex בחברה.
- **למה YAML ולא UI:** מקבלים versioning, branches לבדיקה ו-revert, וגם את היכולת של coding agents להציע שינויים לפקטורי עצמו. פירוט בפרק 5.

3. מ-Slack ל-PR: traces, מדדים ו-scorers

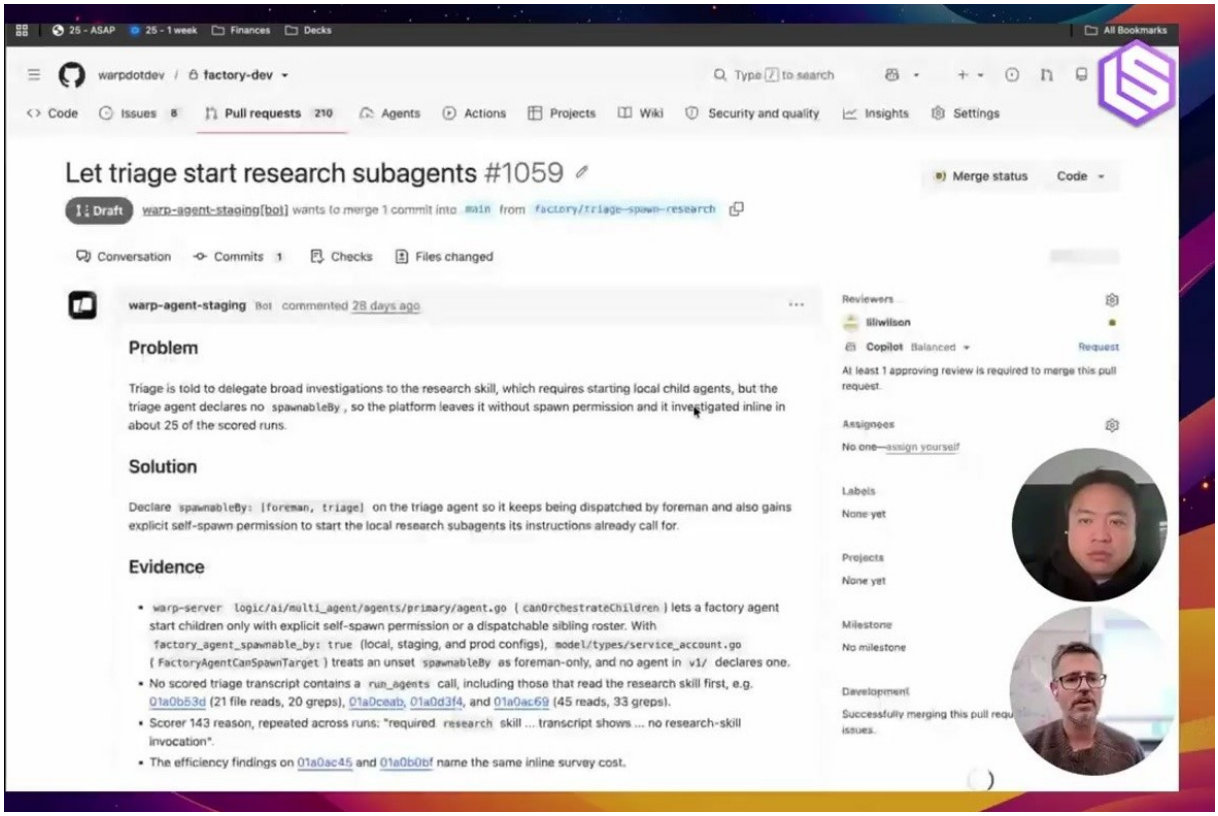
הדמו עוקב אחרי משימה של מהנדס ב-Warp (בתמליל Sofik או Safik, השם לא ודאי) על אפליקציית המובייל. המשימה התחילה ב-Slack, וה-trace מציג את השיחה המלאה, צילומי computer use שנוצרו תוך כדי הבנייה, ורשומה של מה שכל חלק של ה-orchestration agent עשה. Lloyd מדגיש שהערך לא בצפייה במשימה אחת. הערך הוא שיש רשומה כזו לכל משימה בארגון. לדבריו, רוב החברות שהוא מדבר איתן לא מגיעות לזה כי המהנדסים מריצים agents מקומית, כל אחד לבד.

על הנתונים המרוכזים בונים שלושה סוגי מבט: output של הצוות, עלות, ואיכות שנבדקת באמצעות scorers (online evals). Warp מספקת scorers כלליים מוכנים, למשל verbose מדי, יעילות ואיכות קוד. לקוחות כותבים scorers משלהם לכשלים ספציפיים, למשל שינוי שמשבר את ה-build. scorer כזה הוא agent שעובר בדיעבד על שיחות היסטוריות ונותן להן ציון. הדוגמה בדמו היא redundant tests: agents כותבים טסטים בזמן שהם מנסים להבין איך לממש משהו, ואז משאירים אותם בקוד. זה מנפח את ה-codebase ומבזבז tokens.

ממד	מה מודדים	פרטים מהשיחה
Output	מדדי DORA, זמן עד ship, throughput	"How long does it take to ship something?"
אוטומציה	כמה פעמים אדם נגע בכל PR שעבר merge	מדד ייעודי של Warp
עלות	עלות ל-PR, מפולחת לפי אדם, לפי automation ולפי שלב ב-SDLC	"very, very important" ללקוחות. משמשת לבחירת מודלים ו-skills
איכות	Scorers מסוג LLM-as-judge: מודל שופט, prompt, היכן רץ, sampling rate	ה-sampling rate קיים כי ה-judges הם בעצמם agents שעולים כסף

> **Zach Lloyd** — what the world sort of needs is like out of the realm of vibes for understanding these coding agents and actually into the world of measurement.

4. פקטורי שמשפר את עצמו, ו-benchmark על משימות אמיתיות



PR שפתח agent של self-improvement נגד הגדרת הפקטורי: "Let triage start research subagents #1059", עם הסעיפים Problem, Solution ו-Evidence. ה-agent מבקש לאפשר ל-triage להפעיל research sub-agents, ומפנה להרצות שבהן זה היה עוזר

מעל ה-scorers רצה לולאה נוספת של agents שמחפשים כשלים בציונים ומציעים שינויים למערכת. Warp קוראת לזה self-improvement. הדוגמה מהדמו: ה-agents בדקו את שלב ה-triage ומצאו "54 failure cases of this particular form where it would have been better if our triager had been able to spawn a sub-agent to do its investigation". הפקטורי מוגדר בקוד וה-agents הם coding agents, ולכן הם יכולים לפתוח PR ישירות נגד ההגדרה. זה ה-PR שבשקופית.

כדאי לשים לב להסתייגות של Lloyd עצמו: "like we don't have this yet, but the next thing that we want to do with this is like, actually for any suggested change like this, you could go back, replay the runs and make sure that this actually would have improved them". כלומר, ה-PRs המוצעים לא עוברים היום הרצה חוזרת שמוכיחה שהשינוי עוזר. אדם צריך להחליט אם למזג.

> **Zach Lloyd** — they can essentially make a pull request against the factory definition to try to improve the factory itself. And so this is like a, like a full learning loop that's built into the system to try to actually self optimize.

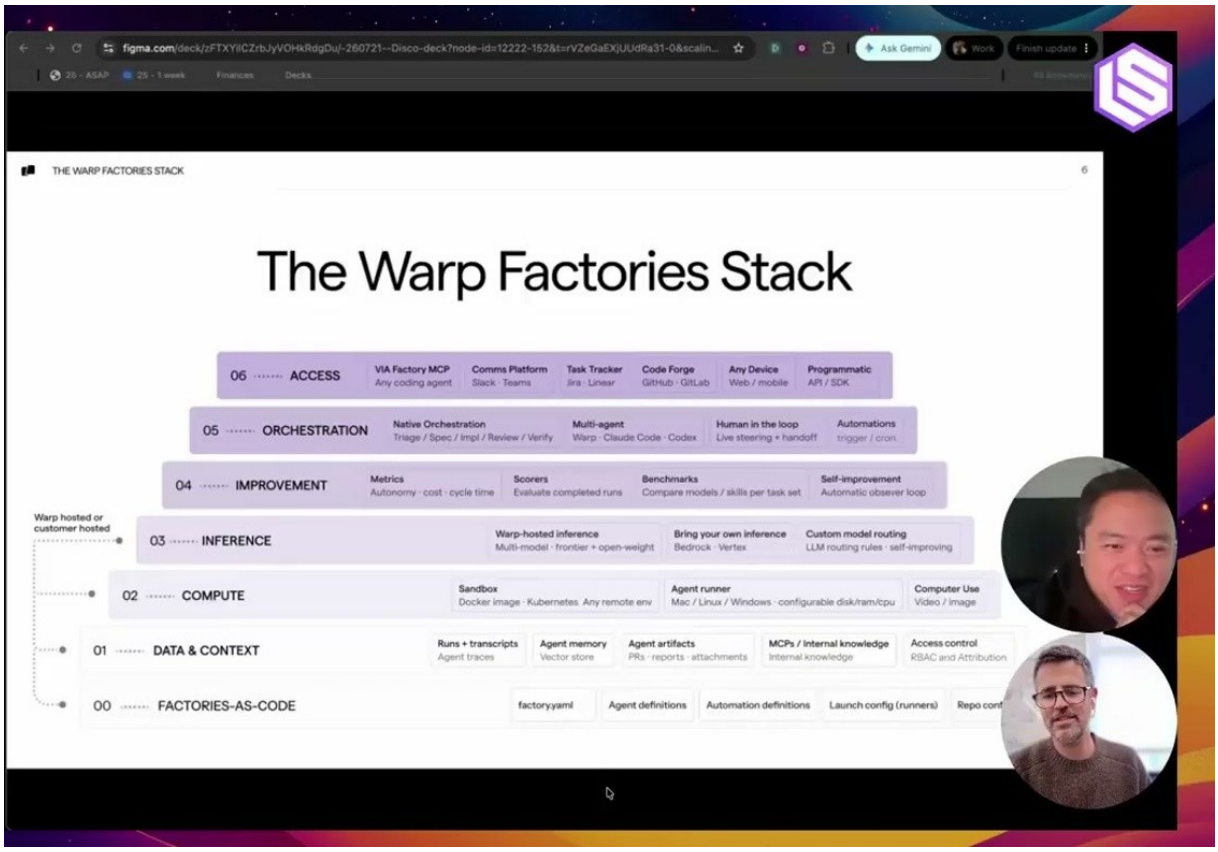
Benchmarking: SWE-bench פרטי על ה-runs של החברה

כל run שעבר בפקטורי אפשר להכניס ל-benchmark suite, למשל אוסף של משימות frontend. את המשימות האלה מריצים מחדש מול מטריצה של קונפיגורציות. Lloyd מדגיש שצריך לשנות לא רק את המודל אלא גם את ה-harness: מול ה-harness של Warp אפשר להשוות Claude Code (בתמליל "Cloud Code"), Codex ו-Pi. התוצאה דומה לגרפים שמתפרסמים ל-SWE-bench או ל-Terminal-Bench, רק על המשימות של החברה עצמה. בגרף cost מול correctness רוצים להיות בפינה השמאלית העליונה: עלות נמוכה ונכונות גבוהה.

המספרים מהדמו: מודל שבתמליל נקרא "GPT-56 Sol" הגיע ל-95% נכונות בעלות ממוצעת של 6 דולר, ומודל שנקרא "GLM-53 Flash" הגיע ל-93% ב-2 דולר. השמות כנראה שובשו ב-ASR. סביר שמדובר בגרסאות

GPT-5.x ו-GLM-5.x, אבל זה לא מאומת. המסקנה של Lloyd: את התוצאות מזינים ל-model routing, או אפילו ל-harness routing, שמוגדר בקוד לפי ה-workflows וה-codebase הספציפיים שלך. הוא גם מציין שהמודלים בגרף כבר "slightly older".

5. ה-stack בשש שכבות, ומ-80 דולר ל-15–20 דולר ל-PR



השקופית "The Warp Factories Stack": פירמידה של שכבות ממוספרות, מ-00 Factories-as-code בבסיס, דרך Data & Context, Compute, Inference ו-Improvement, ועד Orchestration ו-Access בראש. לכל שכבה רשומים הרכיבים שלה

אחרי הדמו מכליל וממליץ על ה-stack גם למי שבונה לבד: "whether it's warp or not". swyx מעיר שהמהלך כאן הוא להוביל את השיח עד כדי כך שכולם יבנו Warp גם בלי להשתמש ב-Warp. Lloyd לא מתווכח עם זה. הטבלה מסכמת את השכבות לפי הסדר שבו הוא מציג אותן, מלמטה למעלה, יחד עם ההמלצות הקונקרטיות.

שכבה	מה היא עושה	המלצות ופרטים מהשיחה
Factories-as-code	הגדרה declarative של הפקטורי	the analogy is like Terraform": versions, branches, revert, ו-agents שמציעים שינויים
Data & Context	איסוף כל השיחות והתוצרים של agents, וחיבור MCPs	שרשרת שלמה: Slack, Linear, GitHub, code review, merge ואפילו server crash אחרי ship. הרשאות MCP לפי מחלקה (marketing, sales, engineering)
Compute	Sandboxes להרצת agents	צריך להיות pluggable. Warp משתמשת ב-Namespace, ואפשר גם Daytona, Modal או E2B. ל-iOS ול-computer use צריך Mac VM עם simulator
Inference	גישה למודלים	BYO inference כדי לנצל commits קיימים מול Anthropic, OpenAI או Google, או רכישה דרך Warp. אפשר גם BYO harness, כל harness תואם ACP, למשל Pi
Improvement	Scorers, self-improvement ו-benchmarks	"the real power of this whole system"

שכבה	מה היא עושה	המלצות ופרטים מהשיחה
Orchestration	מריץ את ההגדרה: מעביר עבודה בין שלבים, טוען skills ומפעיל automations	"Sometimes people talk about the orchestrator like it's the whole platform. I think it's a piece of it."
Access points	Slack, task tracker, terminal, API ו-MCP	הפקטורי הוא headless infrastructure. כ-60% מהעבודה ב-Warp מתחילה ב-Slack

> **Zach Lloyd** — when we first started using the factory approach, it was costing us about $80 per PR because we were using like Opus where we shouldn't have. Our triage agent was using like a really high powered model.

מה תיקנו: הורידו את Opus מהמקומות שבהם לא היה בו צורך, החליפו את המודל החזק ב-triage, וקיצצו עבודה כפולה שנבעה מהמבנה של ה-agents. היו יותר מדי sub-agents ביחס למצב שבו עובדים עם פחות agents. Lloyd אומר שכל זה נעשה "using these built-in features", כלומר עם הכלים שהציג בדמו. התוצאה: "we were able to get the cost per PR down to like 20 bucks or 15 bucks, something like that", ובמקביל "Over the last month, it's gone up by 40%. We measure like the revert rate and it hasn't gone up.". המסקנה שלו היא observe, measure, tune. מה שחסר בנתונים: גודל המדגם, מספר ה-PRs, והשאלה אם גודל ה-PR הממוצע השתנה בתקופה הזו.

6. למי זה מתאים, כלכלת ה-subs, ושני סוגי מהנדסים

swyx שואל, בכובע של CTO של חברת B2B SaaS, אם באמת צריך את כל שש השכבות. Lloyd מציב רף תחתון: "if you're like a few people, honestly, this is, this is overkill". במקרה כזה עדיף subscription של Claude Code או Codex. לדבריו, זה מתחיל להשתלם סביב 5–10 מהנדסים. בטווח של 10–100 מהנדסים, בחברת growth שעובדת עם מה שהוא מכנה "happy stack" (Slack, Linear או Jira), ההטמעה כמעט self-serve. בחברות גדולות יותר נדרשת עבודת הטמעה. מי שמתאים פחות: חברות עם תשתית bespoke מאוד, וחברות עם צוותי פיתוח פנימיים ענקיים שבונות לעצמן, והוא מזכיר בשמן את Uber ו-Ramp.

החלק המעניין ביותר בשיחה הוא הכלכלה. Lloyd מספר על חברות קטנות שעוברות בין "collection of $200 subs", ואומר שהוא לא יודע אם תנאי השימוש מתירים את זה. swyx מוסיף שתוכנית של 200 דולר שווה בערך 8,000 דולר של שימוש ב-API. מכאן Lloyd מסביר למה Warp לא רוצה להיות בעסק של coding agents טהורים: אי אפשר להתחרות על בסיס עלות API שגבוהה כל כך. לדבריו, Cursor הבינה את זה והפכה ל-model lab, ו-Warp Factories ממוצבת כ-"coding agent optimization platform" ולא כמי שמוכרת tokens מחדש.

> **Zach Lloyd** — a lot of people go straight to API, pay by API. And that is between five to 50 times more expensive than just getting a whole bunch of subs.

בסגירה Lloyd מקדים את הנושא של ה-keynote שלו ב-AIE CODE: שני סוגי הנדסה. מצד אחד product engineering, ומצד שני factory engineering, דיסציפלינה חדשה שבה "you're engineering the system that builds the software and not just building the software directly". Warp מגייסת מהנדסים, אנשי go-to-market, אנשי data ומעצב.

7. מה עושים עכשיו

- **אם הצוות קטן מ-5–10 מהנדסים:** להישאר עם subscriptions של Claude Code או Codex. לפי Lloyd, factory בגודל הזה הוא overkill.
- **להתחיל מנקודת כניסה אחת:** לנתב משימות של agents דרך ערוץ אחד, למשל Slack או Linear, שיוצר trace מרכזי. בלי זה אין נתונים למדוד.
- **לכתוב את ה-workflow כקוד:** גם בלי Warp, קובץ declarative של repos, skills, MCPs ושלבים שמנוהל ב-Git. זה הבסיס לכל השאר.
- **למדוד עלות ל-PR ו-human touches ל-PR:** עם פילוח לפי שלב ב-SDLC. אצל Warp רוב החיסכון הגיע מבחירת מודל לפי שלב, ובמיוחד ל-triage.

- **Scorer ראשון ל-redundant tests:** כשל נפוץ וזול לזיהוי. להריץ LLM-as-judge עם sampling rate נמוך.
- **Benchmark פנימי לפני החלפת מודל או harness:** לאסוף 20–50 runs אמיתיים מתחום אחד ולהריץ אותם מחדש. זו הצעת עבודה שלנו, לא מספר שנאמר בשיחה.
- **להשאיר את ה-compute וה-inference כרכיבים pluggable:** כדי לנצל commits קיימים מול ספקים ולהחליף sandbox provider בלי לשכתב.

שאלה פתוחה	למה זה חשוב
איך מאמתים שה-LLM judge של ה-scorer צודק?	ה-PRs של self-improvement נשענים על הציונים האלה. judge שגוי יוביל לשינויים שגויים בפקטורי
האם הירידה מ-80 דולר ל-15–20 דולר נשמרה באיכות?	הנתון היחיד שנמסר הוא revert rate. אין נתונים על באגים בייצור או על גודל PR
מתי replay של runs יאמת שינויים מוצעים?	עד אז הלולאה מוצעת ולא מוכחת. Lloyd אמר "we don't have this yet"
מה עולה Warp Factories ומה בה open source?	הכותרת בדף הנחיתה מדברת על open infrastructure, אבל בשיחה לא נמסרו פרטים
האם rotation של subs מותר?	Lloyd אמר "I just don't really know". זה סיכון תפעולי לחברות קטנות

8. מילון מונחים

מונח	פירוש
Software factory	מערכת agentic בענן שמקבלת work items ומריצה workflow מוגדר של agents עד ל-PR
Factory engineering	הנדסה של המערכת שבונה את התוכנה, בניגוד ל-product engineering
Product surface	תחום מוצר, למשל אפליקציית mobile או שרת. יש factory אחד לכל אחד
Factory-as-code	הגדרה declarative של הפקטורי (repos, skills, MCPs, workflow) בקובץ YAML שמנוהל ב-Git
Trace	רשומה מלאה של שיחת agent, הפעולות שלו והתוצרים שלו
Scorer / online eval	Agent שנותן ציון בדיעבד לשיחות היסטוריות לפי קריטריון מוגדר
LLM-as-judge	שימוש במודל שפה כשופט שמעריך תוצר של מודל אחר
Sampling rate	אחוז ההרצות שנשלחות לבדיקה, כדי לשלוט בעלות
Self-improvement	Agents שמזהים כשלים ופותחים PR נגד הגדרת הפקטורי
Benchmark suite	אוסף runs אמיתיים שמריצים מחדש מול קונפיגורציות שונות
Harness	שכבת ה-agent שעוטפת את המודל, כמו Claude Code, Codex, Pi או ה-harness של Warp
Model / harness routing	בחירה אוטומטית של מודל או harness לכל משימה, לפי תוצאות benchmark
Sandbox / compute layer	סביבת הרצה מבודדת ל-agents, למשל Namespace, Daytona, Modal או E2B
ACP	פרוטוקול שדרכו Warp מתחברת ל-harnesses חיצוניים (בתמליל: "ACP compatible harness")
DORA metrics	מדדי ביצוע של צוותי תוכנה, כמו זמן עד ship ו-throughput
Revert rate	שיעור השינויים שבוטלו אחרי merge, ומשמש כאינדיקציה לאיכות
Subs	מנויים חודשיים (למשל 200 דולר) ל-Claude Code או Codex, לעומת תשלום לפי API