10 ပတ်တာ · Project-based လက်တွေ့သင်ရိုး

Modern Software Engineer

လေ့လာရေးလမ်းညွှန် 2026

စက်တင်ဘာ 4၊ 2026 တွင် နောက်ဆုံးပြင်ဆင်ထားသည် · စစ်ဆေးပြီး လင့်ခ် 197 ခု

https://modern-swe.burmese.dev/my

Coding agents တွေကိုသုံးပြီး Software တည်ဆောက်နည်းကို project-based လက်တွေ့လေ့လာနိုင်မယ့် 10 ပတ်တာ လမ်းညွှန်ဖြစ်ပါတယ် - agents တွေရဲ့ အလုပ်လုပ်ပုံ၊ context engineering နဲ့ MCP၊ reusable skills၊ repository readiness၊ AI code review၊ security၊ background agents၊ Team တစ်ခုအတွင်းမှာ အသုံးပြုပုံ (team adoption) နဲ့ software factory အထိ ပါဝင်ပါတယ်။ တစ်ပတ်စီတိုင်းမှာ 4 နာရီဝန်းကျင် အချိန်ပေးရမယ့် အဓိက လေ့လာစရာများ (Core)၊ ကိုယ်တိုင် လက်တွေ့ တည်ဆောက်ရန် (Build)နဲ့ အမှန်တကယ် ပြီးစီးမှု ရှိမရှိ စစ်ဆေးနိုင်တဲ့ “ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)” စာရင်းတို့ တွဲဖက်ပါဝင်တာကြောင့် စာဖတ်မှတ်တမ်းတစ်ခု သက်သက်အစား အလုပ်လုပ်တဲ့ System တစ်ခုကို ကိုယ်တိုင် တည်ဆောက်သွားနိုင်မှာ ဖြစ်ပါတယ်။

Credit. 10 ပတ်တာ ဖွဲ့စည်းပုံနဲ့ အပတ်စဉ် ခေါင်းစဉ်တွေကို Stanford ရဲ့ CS146S: *The Modern Software Developer* (themodernsoftware.dev) Fall 2026 သင်ရိုးညွှန်းတမ်းအတိုင်း ယူထားပါတယ်။ ဒီလမ်းညွှန်ဟာ အဆိုပါသင်တန်းနဲ့ တိုက်ရိုက် မသက်ဆိုင်ပါဘူး။ ဖတ်စရာ Article များ၊ YouTube Video များ၊ လက်တွေ့ build တွေနဲ့ လေ့လာမှုကို စစ်ဆေးရန်စာရင်း တစ်ခုချင်းစီကို ကိုယ်ပိုင် ရွေးချယ်စုစည်းထားတာ ဖြစ်ပါတယ်။ Link အားလုံးကို September 5, 2026 မှာ စစ်ဆေးထားပြီးဖြစ်ပါတယ်။

10 ပတ်တာ သင်ရိုး အကျဉ်း

1 **The Internals of Coding Agents**
Tools 4 ခုနဲ့ logging အပြည့်အစုံပါတဲ့ lines 300–500 ရှိ terminal agent တစ်ခု တည်ဆောက်ရပါမယ်။

2 **Advanced Context Engineering**
1 မျက်နှာပါ spec တစ်ခု ရေးဆွဲခြင်း၊ RePPIT သုံးပြီး feature တစ်ခု အောင်မြင်စွာ ထုတ်လုပ်ခြင်းနဲ့ tools 2 ခု မှ 4 ခုပါ MCP server တစ်ခု တည်ဆောက်ခြင်း။

3 **Agent Skills and CLI**

Helper script ပါဝင်တဲ့ packaged skill တစ်ခုနဲ့ browser automation သုံးထားတဲ့ web skill တစ်ခု တည်ဆောက်ရပါမယ်။

4 **Customizing Your Agent and Repository**

Repo instructions ဖိုင်တစ်ခု၊ hooks 2 ခုနဲ့ planner / implementer / reviewer ခွဲထုတ်တာဝန်ပေးတဲ့ System တစ်ခု တည်ဆောက်ရပါမယ်။

5 **Agent-Ready Codebases**

Score သတ်မှတ်ထားတဲ့ readiness audit ပြုလုပ်ခြင်းနဲ့ agent အသစ်တစ်ခု ချက်ချင်းအလုပ်ဖြစ်စေမယ့် ပြုပြင်မှုများ ပြုလုပ်ခြင်း။

6 **Agentic Code Review**

Pull requests တွေနဲ့ ချိတ်ဆက်ထားပြီး PR 5 ခုပေါ်မှာ တိုင်းတာစမ်းသပ်ထားတဲ့ severity အလိုက် review စံသတ်မှတ်ချက် (rubric) တစ်ခု တည်ဆောက်ရပါမယ်။

7 **Security**

Threat model တစ်ခု၊ CI အတွင်း SAST / SCA / secret scans များ ထည့်သွင်းခြင်းနဲ့ prompt-injection စမ်းသပ်မှုတစ်ခု ပြုလုပ်ရပါမယ်။

8 **Background Agents**

Isolation၊ budgets၊ checkpoints တွေနဲ့ retries တွေ ပါဝင်တဲ့ issue-to-PR automated workflow တစ်ခု တည်ဆောက်ရပါမယ်။

9 **Building an AI-Native Team**

Model gateway တစ်ခု၊ MCP portal တစ်ခုနဲ့ စာမျက်နှာ 1 မျက်နှာပါ adoption policy တစ်ခု ရေးဆွဲရပါမယ်။

10 **The Software Factory + The Future**

Eval suite တစ်ခုနဲ့ စနစ်တကျ ထိန်းချုပ်ထားတဲ့ improvement loop ပါဝင်တဲ့ traced end-to-end factory System တစ်ခု တည်ဆောက်ရပါမယ်။

ဒီလမ်းညွှန်ကို ဘယ်လို လေ့လာမလဲ

တစ်ပတ်ကို 10–12 နာရီ အချိန်ပေးဖို့ လိုအပ်ပါတယ်။

3–4 နာရီ. သက်ဆိုင်ရာ Week တိုင်းအတွက် **အဓိက လေ့လာစရာများ (Core)** ကို ဖတ်ပါ။ (Core စာရင်းကို အများဆုံး 4 နာရီဝန်းကျင်ပဲ ကုန်အောင် ကန့်သတ်ထားပြီး ကြာချိန်တွေကိုလည်း ဘေးမှာ ရေးပေးထားပါတယ်)။

5–6 နာရီ. အပတ်စဉ် **လက်တွေ့တည်ဆောက်ရန် (Build)** ကို ကိုယ်တိုင် လုပ်ပါ။

1–2 နာရီ. ရလဒ်တွေ၊ မအောင်မြင်ခဲ့တာတွေ၊ ကုန်ကျစရိတ်နဲ့ ရခဲ့တဲ့ သင်ခန်းစာတွေကို မှတ်တမ်းတင်ပြီး **"ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)"** စာရင်းအတိုင်း စစ်ဆေးပါ။

ပိုလေ့လာချင်သူများအတွက် (Deeper material)၊ အပိုဆောင်း Video များ နဲ့ **Tools and references** တွေအောက်က အရာအားလုံးဟာ optional ဖြစ်ပါတယ်။ ကိုယ်စိတ်ဝင်စားတာကိုပဲ ရွေးဖတ်ပါ၊ အကုန်လုံးပြီးဖို့ မလိုပါ။

ဆယ်ပတ်လုံးလုံးမှာ **repository တစ်ခုတည်းကိုပဲ စွဲသုံးပါ**။ UI၊ tests နဲ့ CI အပြည့်ပါတဲ့ SaaS app အသေးစား၊ developer tool ဒါမှမဟုတ် API project တစ်ခုဟာ ဟိုစပ်စပ်ဒီစပ်စပ် ကစားစရာ project 10 ခုထက် အများကြီး ပိုထိရောက်ပါတယ်။ Week တိုင်းမှာ အဲဒီ project တစ်ခုတည်းကိုပဲ ပိုကောင်းအောင် ဆက်တိုက် လုပ်သွားရပါမယ်။

အကောင်းဆုံး အခမဲ့လေ့လာနိုင်တဲ့ အခြေခံကတော့ Hugging Face ရဲ့ **Context Course** ဖြစ်ပါတယ် (Agent Skills, MCP, Plugins, Sub-agents, Hooks နဲ့ Nano Harness စတဲ့ အခန်းတွေ ပါဝင်ပါတယ်)။ ဒီသင်တန်းဟာ ဒီလမ်းညွှန်ရဲ့ ပထမ 5 ပတ်နဲ့ အတော်လေးကို ကိုက်ညီမှုရှိတာကြောင့် အောက်ပါသင်ခန်းစာတွေနဲ့ တွဲပြီး ပိုပြီးနက်နက်နဲနဲ လေ့လာနိုင်ပါတယ်။

ဒီလမ်းညွှန်အတွက် ဘယ် programming language မဆို အသုံးပြုနိုင်ပါတယ် (ဥပမာအများစုက Python သို့မဟုတ် JavaScript ကို သုံးထားပါတယ်)။ ဒါပေမဲ့ programming အတွေ့အကြုံ ရှိဖို့တော့ လိုအပ်ပါမယ် (တက္ကသိုလ်အဆင့် programming course 2 ခု၊ 3 ခု ဒါမှမဟုတ် ညီမျှတဲ့ အတွေ့အကြုံ)။ Machine learning အတွေ့အကြုံရှိရင် ပိုကောင်းပေမဲ့ မဖြစ်မနေ မဟုတ်ပါဘူး။ တကယ်လို့ transformers သဘောတရားကို မသိသေးရင် Week 1 ရဲ့ optional foundation မှာ လေ့လာနိုင်ပါတယ်။

Tools တွေနဲ့ ကုန်ကျစရိတ် သတ်မှတ်ချက်

တချို့ tools တွေက subscription ဒါမှမဟုတ် API key လိုအပ်ပါတယ်။ ဒါကြောင့် Week 1 မစခင် အောက်ပါအချက်တွေကို ကြိုတင်ဆုံးဖြတ်ပါ -

ပင်မ coding agent တစ်ခုကို ရွေးပါ. Week 3–4 မှာ လုပ်မယ့် customization တွေ ပိုထိရောက်စေဖို့ အနည်းဆုံး 5 ပတ်လောက်တော့ ရွေးထားတဲ့ agent တစ်ခုတည်းကိုပဲ စွဲသုံးပါ။ သင့်တော်တဲ့ ရွေးချယ်မှုတွေကတော့ -

Claude Code (subscription သို့မဟုတ် API key)

OpenAI Codex CLI (open source၊ subscription သို့မဟုတ် API key)

Cursor

Gemini CLI (open source၊ free tier ပါဝင်သည်)

opencode (open source၊ ကိုယ်ပိုင် API key ထည့်သုံးရသည်)

မစတင်မီ လစဉ် သုံးစွဲမယ့် ကုန်ကျစရိတ် ကန့်သတ်ချက် (Spend Cap) ကို သေချာ သတ်မှတ်ပါ.
အပတ်စဉ် build မှတ်စုတွေနဲ့အတူ ကုန်ကျစရိတ်ကို မှတ်တမ်းတင်ပါ။ ဒီသင်ရိုးမှာ ကုန်ကျစရိတ်ကို သေချာ စောင့်ကြည့်တာဟာ ဦးစားပေး တိုင်းတာရမယ့် အချက် ဖြစ်ပါတယ်။

Prompt Caching ကို လေ့လာပြီး သုံးပါ
Subagents တွေနဲ့ review စစ်ဆေးတဲ့ အဆင့်တွေအတွက် ပိုဈေးသက်သာတဲ့ model တွေကို သုံးပါ။ ထိန်းမနိုင် သိမ်းမရ ကုန်သွားတဲ့ API စရိတ် အများစုဟာ turn တိုင်းမှာ context အရှည်ကြီးတွေကို ထပ်ခါထပ်ခါ ပြန်ပို့နေ တာကြောင့် ဖြစ်ပါတယ်။

အခမဲ့ လမ်းကြောင်း (Free Path). Hugging Face courses တွေ၊ အထက်ပါ free tier ပါတဲ့ open-source agents တွေနဲ့ ဒီလမ်းညွှန်ထဲက ဆောင်းပါးအားလုံးဟာ အခမဲ့ ဖြစ်ပါတယ်။ ကိုယ်တိုင် build လုပ်တဲ့အချိန်မှာ သုံးရတဲ့ API usage ကလွဲလို့ ကျန်တာ ဘာကုန်ကျစရိတ်မှ မရှိပါဘူး။

ရက်သတ္တပတ် 10 ပတ်အနက် 1 ပတ်မြောက်

The Internals of Coding Agents

Core ≈ 3 h 20 min · မီဒီယို 6 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် LLM တစ်ခုရဲ့ အမှန်တကယ် အလုပ်လုပ်ပုံနဲ့ အောက်ခြေအဆင့်မှာ agent loop တစ်ခု ဘယ်လိုလည်ပတ်နေသလဲ ဆိုတာကို နားလည်စေရန်။ အဓိက tools တွေဖြစ်တဲ့ read, write, edit, bash တွေဆီ task တွေ ဘယ်လို ဖြတ်သန်းသွားသလဲ ဆိုတာနဲ့ production coding agents တွေမှာ system prompts တွေ၊ tool definitions တွေကို ဘယ်လို ဖွဲ့စည်းတည်ဆောက်ထားသလဲ ဆိုတာကို လေ့လာပါမယ်။

လက်တွေ့ တည်ဆောက်ရန် Tools 4 ခုနဲ့ logging အပြည့်အစုံပါတဲ့ lines 300–500 ရှိ terminal agent တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 20 min

Video · 59 min

▸ Intro to Large Language Models

Andrej Karpathy

LLM အယူအဆ အခြေခံအတွက် အကောင်းဆုံးနဲ့ အကျစ်လျစ်ဆုံး ရှင်းလင်းချက်။ အချိန်ရရင်တော့ ပိုပြည့်စုံတဲ့ ▸ *Deep Dive into LLMs like ChatGPT* (3 h 31 min) ကို ကြည့်ပါ။

ဆောင်းပါး + Code · Code ဖတ်ချိန်အပါ ≈ 60 min

How to Build an Agent

Thorsten Ball

မရှိမဖြစ် tool loop ပါဝင်တဲ့ coding agent အသေးစားလေးတစ်ခု။ ဒီအပတ်ရဲ့ build အတွက် နမူနာယူပြီး ရေးသားနိုင်ပါတယ်။

ဆောင်းပါး · 25 min

Building Effective AI Agents

Anthropic

Workflows နဲ့ agents ကွာခြားပုံ၊ augmented-LLM loop အကြောင်းကို အတိအကျဆုံး ရေးသားထားတဲ့ စာတမ်း။ အောက်က ▸ Barry Zhang ရဲ့ talk ရဲ့ talk ဟာ ဒီစာတမ်းကို Videoနဲ့ ရှင်းပြထားတာ ဖြစ်ပါတယ်။

အင်ဂျင်နီယာ ဆောင်းပါး 20 min

Unrolling the Codex agent loop

OpenAI

Codex ရဲ့ agent loop ဖွဲ့စည်းပုံနဲ့ design tradeoffs တွေကို production အမြင်နဲ့ ရှင်းပြထားချက်။

Talk 15 min

▸ How We Build Effective Agents

Barry Zhang of Anthropic

Agent ရဲ့ ကန့်သတ်ထားတဲ့ context အတွင်းကနေ loop အသေးဆုံး တည်ဆောက်ပုံ၊ tool အသုံးပြုပုံ၊ risk၊ verification နဲ့ debugging ပြုလုပ်ပုံများ။

Production Prompts နှင့် Tool Definitions

အကျဉ်းချုပ်တွေပဲ ဖတ်မနေဘဲ production မှာ တကယ်သုံးနေတဲ့ system prompts အစစ်တွေကို ဖတ်ကြည့်ပါ -

codex-rs/core

OpenAI Codex CLI ဟာ open source ဖြစ်ပါတယ်။ `gpt_5_2_prompt.md` ၊ `gpt_5_codex_prompt.md` စတဲ့ model prompts တွေနဲ့ `codex-rs/prompts/templates/compact/` အောက်က compaction prompt တွေကို ဖတ်ကြည့်ပါ။

Gemini CLI

Google ရဲ့ system prompt ကို `packages/core/src/core/prompts.ts` မှာ ဖတ်နိုင်ပြီး၊ MCP prompts တွေကို `packages/core/src/prompts/` မှာ ကြည့်နိုင်ပါတယ်။

opencode

အထက်က agent နှစ်ခုနဲ့ prompts တွေ၊ tool schemas တွေကို ယှဉ်ကြည့်လို့ရတဲ့ open-source coding agent ဖြစ်ပါတယ်။

Video 1 h 06 min

▸ How Claude Code Works

Prompt-driven architecture, tool calls, subagents, permissions နဲ့ evaluations တွေအကြောင်း လေ့လာနိုင်တဲ့ အကောင်းဆုံး workshop ဖြစ်ပါတယ်။

Optional Foundation (အခြေခံ ထပ်ဖြည့်ရန်)

Video 27 min

▸ Transformers, the tech behind LLMs

3Blue1Brown

Attention mechanism ကို အရုပ်တွေနဲ့ အရှင်းဆုံး ပြထားတဲ့ Video။

Jay Alammar

The Illustrated Transformer

အထက်ပါ Video နဲ့ တွဲဖတ်ရမယ့် နာမည်ကျော် ဆောင်းပါး။

Video 2 h 11 min

▸ How I use LLMs

Andrej Karpathy

Model အသုံးပြုပုံ လက်တွေ့၊ tools တွေနဲ့ အားနည်းချက်များ။

AI Engineering (Chip Huyen)

Foundation models, evaluation နဲ့ application architecture အခန်းတွေကို ဖတ်ပါ။

ပိုလေ့လာချင်သူများအတွက် (Deeper Material)

Geoffrey Huntley

How to build a coding agent: free workshop

သီးခြား design တစ်မျိုးနဲ့ အစကနေ တည်ဆောက်ပြထားတဲ့ workshop။

mini-swe-agent

SWE-bench Verified မှာ 74% ကျော် ရထားတဲ့ lines 100 သာရှိတဲ့ agent။ ကိုယ်တိုင် ရေးပြီးရင် ကိုယ့်Codeထဲ ဘာတွေ ပိုလျှံပြီး over-engineer ဖြစ်သွားလဲဆိုတာ ဒီ Code နဲ့ ယှဉ်ကြည့်ပါ။

PDF 30 min

A practical guide to building agents

OpenAI

Video 31 min

▸ Building Effective Agents with LangGraph

LangChain

Routing, parallelization, orchestrator-worker နဲ့ evaluator-optimizer patterns များ။

အပိုဆောင်း Video များ

Video 13 min

▸ What's next for AI agentic workflows

Andrew Ng

Reflection, tool use, planning နဲ့ multi-agent collaboration ဆိုတဲ့ အဓိက agentic patterns 4 ခု။

Video 17 min

▸ Software Development Agents: What Works and What Doesn't

Robert Brennan of OpenHands

Editor, terminal, browser, sandbox နဲ့ action loop အကြောင်း။

လက်တွေ့ တည်ဆောက်ရန် (Build)

Lines 300 ကနေ 500 ကြားရှိမယ့် terminal coding agent တစ်ခုကို ရေးပါ။

Tools 4 ခု ထည့်ပေးပါ - `list files`၊ `read a file`၊ `edit a file` နဲ့ `run a shell command`။

Model response တိုင်း၊ tool call တိုင်း၊ ရလဒ်တွေ၊ token count နဲ့ ဘာကြောင့် ရပ်သွားလဲဆိုတဲ့ stop condition အားလုံးကို log အပြည့်အစုံ မှတ်တမ်းတင်ပါ။

Repo အသေးစားလေးထဲက task အသေး 3 ခုပေါ်မှာ စမ်းသပ်ပါ။ ပြီးရင် ဖြစ်လာတဲ့ error တိုင်းကို model error လား၊ context ကြောင့်လား၊ tool ကြောင့်လား ဒါမှမဟုတ် control-loop ချို့ယွင်းချက်လားဆိုပြီး ခွဲခြားမှတ်တမ်းတင်ပါ။

နောက်ဆုံးအနေနဲ့ production system prompt တစ်ခု (Codex သို့မဟုတ် Gemini CLI) ကို အစအဆုံးဖတ်ပြီး note ထုတ်ပါ - ဘယ်နေရာတွေက persona သတ်မှတ်ထားသလဲ၊ ဘယ်နေရာက safety စည်းကမ်းတွေလဲ၊ ဘယ်နေရာက tool ရွေးချယ်မှုဆိုင်ရာလဲ၊ ဘယ်နေရာက ရပ်တန့်ဖို့ (stop) ညွှန်ကြားချက်လဲ ဆိုတာကို ကိုယ့် prompt နဲ့ ယှဉ်ပြီး အနည်းဆုံး 10 နေရာ မှတ်ချက်ရေးပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] ကိုယ်တိုင်ရေးထားတဲ့ agent ဟာ task 3 ခုထဲက အနည်းဆုံး 1 ခုကို tools 4 ခုလုံး အမှန်တကယ် သုံးပြီး အစအဆုံး အောင်မြင်စွာ လုပ်ဆောင်နိုင်ရမယ်။
- [ ] Run ထားတဲ့ log တိုင်းမှာ per-turn token counts တွေနဲ့ အဆုံးသတ် stop reason တွေ အပြည့်အစုံ ပါဝင်ရမယ်။
- [ ] Error 5 ခုထက်မနည်း ပါဝင်တဲ့ failure table တစ်ခု ပြုစုထားပြီး တစ်ခုချင်းစီကို model, context, tool ဒါမှမဟုတ် control loop ဆိုပြီး ခွဲခြားထားရမယ်။
- [ ] Production prompt တစ်ခုကို အနည်းဆုံး နေရာ 10 ခု အသေးစိတ် note ထုတ်ထားတဲ့ မှတ်စုရှိရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 2 ပတ်မြောက်

Advanced Context Engineering

Core ≈ 3 h 50 min · ဗီဒီယို 9 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် အဆင့်မြင့် prompting နည်းSystem တွေနဲ့ ဘယ်အချိန်မှာ ဘာကိုသုံးရမလဲဆိုတဲ့ အချက်များ၊ RePPIT (Research, Propose, Plan, Implement, Test) System နဲ့ spec-driven development အကြောင်း၊ MCP အခြေခံများ (servers, clients, tools, transports) နဲ့ agent တွေအတွက် သုံးရအဆင်ပြေမယ့် tool ဒီဇိုင်းဆွဲနည်းများ။

လက်တွေ့တည်ဆောက်ရန် 1 မျက်နှာပါ spec တစ်ခု ရေးဆွဲခြင်း၊ RePPIT သုံးပြီး feature တစ်ခု အောင်မြင်စွာ ထုတ်လုပ်ခြင်းနဲ့ tools 2 ခုမှ 4 ခုပါ MCP server တစ်ခု တည်ဆောက်ခြင်း။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 50 min

Podcast/Video · 1 h 33 min

▸ Context Engineering

The Pragmatic Engineer with Dex Horthy

Context, harnesses, loops, research/plan/implement workflows, compaction နဲ့ software factories အကြောင်း။ မှတ်စုများအတွက် article and transcript က အသုံးဝင်ပါတယ်။

ဆောင်းပါး · 25 min

Effective context engineering for AI agents

Anthropic

Context ကို အကန့်အသတ်ရှိတဲ့ အရင်းအမြစ်တစ်ခုအဖြစ် System တကျ သုံးစွဲပုံ - system prompts, tools, examples, retrieval နဲ့ compaction များ။

Video · 29 min

▸ Prompting for Agents

Anthropic

Model တစ်ခုဟာ tools တွေနဲ့ loop ပတ်ပြီး အလုပ်လုပ်တဲ့အခါ prompt ရေးသားပုံ ဘယ်လို ပြောင်းလဲသွားသလဲ ဆိုတာ။

ဆောင်းပါး 25 min

Writing effective tools for agents

Anthropic

Tool နာမည်များ၊ descriptions၊ သတ်မှတ်ချက် ဘောင်များ၊ ရလဒ် ထုတ်ပေးပုံနဲ့ evaluation လုပ်နည်းများ။

Spec စာတမ်း Architecture, transports နဲ့ tools အပိုင်းများအတွက် 30 min

Model Context Protocol (MCP) Specification

Tutorial တွေ မဖတ်ခင် မူရင်း အခြေခံ သတ်မှတ်ချက်ကို အရင် ဖတ်ပါ။

Documentation · GitHub

Spec Kit & Agentic SDD workflow

20 min

Spec-driven development ကို လက်တွေ့ အကောင်အထည်ဖော်ထားတဲ့ လုပ်ငန်းစဉ် လမ်းညွှန်။

ပိုလေ့လာချင်သူများအတွက် (Deeper Material)

MCP Course (Hugging Face)

Units 0–2 (6–8 နာရီခန့်၊ Week 2 နဲ့ 3 မှာ ခွဲပြီး လေ့လာပါ) - အစအဆုံး လက်တွေ့ပါဝင်တဲ့ အခမဲ့ MCP သင်တန်း။ end-to-end MCP application unit အထိ ဆက်လက် လေ့လာသွားပါ။

Context Engineering for AI Agents: Lessons from Building Manus

KV-cache hit rate, file system ကို context အဖြစ် သုံးပုံနဲ့ error တွေကို trace ထဲမှာ သိမ်းဆည်းပုံ။

How Long Contexts Fail (Drew Breunig) & Context Rot (Chroma)

Context ပိုရှည်တိုင်း အလကားမရဘူးဆိုတဲ့ အထောက်အထားများ။

Advanced Context Engineering (HumanLayer)

12-Factor Agents မူဝါဒများ။

Specs Are the New Source Code (Ravi Mehta)

Spec-driven development ရဲ့ product အမြင်။ Al Harris ရဲ့ talk နဲ့ တွဲဖက်လေ့လာနိုင်တဲ့ Kiro specs docs ကိုလည်း ကြည့်ပါ။

MCP Inspector

MCP servers တွေကို စမ်းသပ်စစ်ဆေးဖို့ tooling System ။ MCP Registry ကတော့ MCP servers တွေကို ထုတ်ဝေမျှဝေဖို့နဲ့ ရှာဖွေဖို့အတွက် ဖြစ်ပါတယ်။

Video 10 min

▸ How I use Claude Code for real engineering

Matt Pocock

Planning, မရှင်းတာတွေ မေးမြန်းခြင်း၊ အဆင့်လိုက် အကောင်အထည်ဖော်ခြင်းနဲ့ context window စီမံခန့်ခွဲပုံ။

အပိုဆောင်း Video များ

ဗီဒီယို 1 h 16 min

▸ AI Prompt Engineering: A Deep Dive

Anthropic

နှင့် ▸ Prompting 101 *(24 min)*

ဗီဒီယို 1 h 37 min

▸ AI prompt engineering in 2025: What works and what doesn't

Sander Schulhoff on Lenny's Podcast

ဗီဒီယို 21 min

▸ No Vibes Allowed: Solving Hard Problems in Complex Codebases

Dex Horthy

Research, compaction, plan နဲ့ human review လုပ်ငန်းစဉ်များ။

ဗီဒီယို 39 min

▸ Context Engineering Our Way to Long-Horizon Agents

Harrison Chase of LangChain

ဗီဒီယို 1 h 04 min

▸ Spec-Driven Development: Agentic Coding at FAANG Scale and Quality

Al Harris of Amazon Kiro

ဗီဒီယို 1 h 44 min

▸ Building Agents with Model Context Protocol

Mahesh Murag of Anthropic

Videoသင်တန်း ≈ 1 h 30 min

MCP: Build Rich-Context AI Apps

DeepLearning.AI & Anthropic

ဗီဒီယို 9 min

▸ Model Context Protocol: A Deep Dive

Gaurav Sen

လက်တွေ့ တည်ဆောက်ရန် (Build)

ကိုယ့် repo အတွက် တကယ့် feature အစစ်တစ်ခုကို ရွေးပါ။

Objective, constraints, out-of-scope, acceptance checks, edge cases တွေနဲ့ test plan ပါတဲ့ **1 မျက်နှာပါ spec** တစ်ခု ရေးပါ။

Research → Propose → Plan → Implement → Test (RePPIT). အဆင့်တိုင်းကို ကျော်မသွားဘဲ တစ်ဆင့်ချင်း လုပ်ဆောင်ပြီး ထွက်လာတဲ့ စာရွက်စာတမ်း/ဖိုင် (artifacts) တွေကို သိမ်းထားပါ။

ပြီးရင် သီးသန့် အလုပ်လုပ်မယ့် tools 2 ခုကနေ 4 ခုပါတဲ့ **MCP server တစ်ခု တည်ဆောက်ပါ**။

အဲဒီ server ကို MCP Inspector မှာ အရင်စမ်းပါ၊ ပြီးမှ ကိုယ့် agent နဲ့ ချိတ်စမ်းပါ။ ပုံမှန်အလုပ်လုပ်ပုံ (happy paths) သာမက မှားယွင်းတဲ့ arguments တွေ၊ ရလဒ် အများကြီး ထွက်တာတွေ၊ timeouts တွေနဲ့ permission failures တွေကိုပါ စမ်းသပ်ပါ။

Agent အသစ်တစ်ခုက သင့်တော်တဲ့ tool ကို အမြဲတမ်း မှန်မှန်ကန်ကန် ရွေးချယ်နိုင်ပြီး parameter အမှန်တွေကို ထည့်ပေးနိုင်တဲ့အထိ tool descriptions တွေကို ပြင်ဆင်ရေးသားပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Spec အပြည့်အစုံ ရှိရမယ်၊ RePPIT အဆင့်တစ်ခုချင်းစီက ထွက်လာတဲ့ artifacts တွေ (research notes, proposal, plan, git diff, test results) အကုန် သိမ်းဆည်းထားပြီး ဖြစ်ရမယ်။
- [ ] တည်ဆောက်ထားတဲ့ MCP server ဟာ Inspector ထဲမှာ tools အားလုံးအတွက် valid ရော invalid inputs တွေကိုပါ စမ်းသပ်အောင်မြင်ရမယ်။
- [ ] Scripted requests 5 ခု စမ်းသပ်တဲ့အခါ agent အသစ်တစ်ခုက 5 ကြိမ်စလုံး tool အမှန်ကို ရွေးချယ်နိုင်ရမယ်။
- [ ] အဲဒီ feature တည်ဆောက်ရာမှာ ကုန်ကျခဲ့တဲ့ token counts နဲ့ API စရိတ်ကို notes ထဲမှာ မှတ်တမ်းတင်ထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 3 ပတ်မြောက်

Agent Skills and CLI

Core ≈ 3 h 40 min · မီဒီယို 3 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် Skills ဆိုတာ ဘာလဲ၊ `SKILL.md` နဲ့ scripts တွေက workflow တစ်ခုကို ဘယ်လို code အဖြစ် ပြောင်းလဲပေးသလဲ ဆိုတာကို နားလည်စေရန်။ Web skills သုံးပြီး agent တွေရဲ့ စွမ်းဆောင်ရည်ကို repo အပြင်ဘက်အထိ ဘယ်လို ချဲ့ထွင်မလဲနဲ့ CLI ကနေ ထိရောက်စွာ အလုပ်လုပ်ပုံများ။

လက်တွေ့ တည်ဆောက်ရန် Helper script ပါဝင်တဲ့ packaged skill တစ်ခုနဲ့ browser automation သုံးထားတဲ့ web skill တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 40 min

Course unit · Hugging Face

Agent Skills

60 minခန့်

လွှဲပြောင်းရလွယ်ပြီး အဆင့်အလိုက် ဖွင့်ပြပေးတဲ့ progressive disclosure skills တွေအကြောင်း အကောင်းဆုံး သင်ခန်းစာ။

ဆောင်းပါး · Anthropic

Equipping agents for the real world with Agent Skills

20 min

Public skills repository နမူနာများနှင့်တကွ ရှင်းပြချက်။

Spec & Tutorial · 30 min

Agent Skills Quickstart

Video · 45 min

▸ The Beginner's Guide to Coding with Cursor

Lee Robinson of Cursor

Typed languages, linting, formatting, tests, branch review နဲ့ parallel background work များ။

MIT CLI Foundation · လေ့ကျင့်ခန်းများအပါ ≈ 60 min

The Missing Semester 2026: Shell Overview & Intro

Shell navigation, scripting, streams, permissions နဲ့ safe bash practices များ။

10 min

AGENTS.md Standard

နမူနာတွေနဲ့ စည်းမျဉ်းတွေကို အခု ဖတ်ထားပါ (Repo instructions အကြောင်းကို Week 4 မှာ အဓိက လေ့လာပါမယ်)။

Tools and References

Command Line Interface Guidelines

Agent တွေရော လူတွေပါ စိတ်ချလက်ချ ခိုင်းစေနိုင်မယ့် CLI ဒီဇိုင်းဆွဲနည်း။ ကိုယ့် skill ရဲ့ helper script မှာ ဒီ စည်းကမ်းတွေကို အသုံးချပါ။

Web Skills Tooling. Playwright MCP, Chrome DevTools MCP, နှင့် `browser-use` ။

Common Workflows (Claude Code)

Git worktrees, parallel sessions နဲ့ CLI ကနေ conversations တွေကို ပြန်လည် ဆက်လုပ်ပုံ (resuming) များ။

အပိုဆောင်း Video များ

Video · 1 h 19 min

▸ Skill Issue: How We Used AI to Make Agents Actually Good at Supabase

Pedro Rodrigues of Supabase

`SKILL.md` တည်ဆောက်ပုံ၊ supporting scripts များ၊ progressive disclosure နဲ့ routing အလုပ်လုပ်ပုံ။

Video · 18 min

▸ Bringing Agents onto the World Wide Web

Paul Klein of Browserbase

Browser-agent harnesses, network interception, WebMCP, Playwright CLI နဲ့ reusable website skills များ။

Video 35 min

▸ How AI agents & Claude skills work (Clearly Explained)

Greg Isenberg

Video 2 h 29 min

▸ Future of Programming with AI

Lex Fridman with the Cursor team

Editor design, speculative edits နဲ့ Cursor တည်ထောင်သူတွေရဲ့ agent ဆိုင်ရာ အတွေးအခေါ်များ။

လက်တွေ့ တည်ဆောက်ရန် (Build)

ကိုယ့်အလုပ်ထဲမှာ အနည်းဆုံး 3 ကြိမ်ထက်မနည်း ထပ်ခါထပ်ခါ လုပ်ခဲ့ရတဲ့ workflow တစ်ခုကို ရှာပါ - ဥပမာ release notes ထုတ်တာ၊ DB migration စစ်တာ၊ API endpoint တည်ဆောက်တာ၊ dependencies အဆင့်မြှင့်တာ ဒါမှမဟုတ် UI regression စစ်ဆေးတာမျိုး ဖြစ်နိုင်ပါတယ်။

အဲဒီ workflow ကို အောက်ပါအချက်တွေပါဝင်တဲ့ **skill တစ်ခုအဖြစ် packaged လုပ်ပါ** -

Agent ဘယ်အချိန်မှာ ဒီ skill ကို သုံးရမလဲဆိုတာ ညွှန်ပြထားတဲ့ တိုတိုရှင်းရှင်း `SKILL.md` ဖိုင်။

အသေးစိတ် အချက်အလက်တွေကို context မပြည့်စေဖို့အတွက် linked references အနေနဲ့ သီးခြားထားပေးတဲ့ progressive disclosure ပုံစံ။

CLI Guidelines အတိုင်း ရေးထားတဲ့ deterministic helper script တစ်ခု။

စမ်းသပ်ဖို့ fixture သို့မဟုတ် sample input တစ်ခု။

အလုပ်ဖြစ်မဖြစ် စစ်ဆေးပေးမယ့် pass/fail verification command တစ်ခု။

ဒီ skill ကို agent session အသစ်တစ်ခုဖွင့်ပြီး မတူညီတဲ့ inputs 2 ခုနဲ့ စမ်းသပ်ပါ။ ညွှန်ကြားချက် မရှင်းလင်းတဲ့ နေရာတွေနဲ့ context အရမ်းများသွားစေတဲ့ နေရာတွေကို မှတ်ထားပါ။

ပြီးနောက် **web skill တစ်ခု ထပ်ပေါင်းထည့်ပါ** - ကိုယ့် app ပေါ်မှာ တကယ် run နေတဲ့ အခြေအနေတစ်ခုခုကို စစ်ဆေးနိုင်မယ့် browser-driven automation (Playwright MCP သို့မဟုတ် Chrome DevTools MCP) တစ်ခု ဖြစ်ရပါမယ်။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] ဖန်တီးထားတဲ့ skill ဟာ session အသစ်တစ်ခုမှာ အပိုရှင်းပြချက်တွေ ထပ်ပြောစရာမလိုဘဲ inputs 2 ခုလုံးအတွက် မှန်မှန်ကန်ကန် အလိုအလျောက် trigger ဖြစ်ရမယ်။
- [ ] Helper script မှာ ပြန်ထွက်လာမယ့် exit code တွေကို စနစ်တကျ ရေးသားထားပြီး run ချိန် 10 စက္ကန့်အောက်သာ ကြာမြင့်ရမယ်။
- [ ] Web skill ဟာ browser အစစ်ကို မောင်းနှင်ပြီး ကိုယ့် app ပေါ်မှာ pass/fail ရလဒ်ကို အမှန်တကယ် ပြသနိုင်ရမယ်။
- [ ] `SKILL.md` ထဲမှာ မရှင်းလင်းလို့ ပြင်ဆင်ခဲ့ရတဲ့ အချက်တွေနဲ့ ဘာကြောင့် ပြင်ရတယ်ဆိုတဲ့ အကြောင်းပြချက်တွေကို notes ထဲမှာ စာရင်းပြုစုထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 4 ပတ်မြောက်

Customizing Your Agent and Repository

Core ≈ 3 h 40 min · မီဒီယို 6 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် CLAUDE.md နဲ့ AGENTS.md ထဲ ဘာတွေထည့်ရမလဲ၊ ဘယ်အချက်ကို ဘယ်နေရာမှာ ထားရမလဲဆိုတာ နားလည်စေရန်။ Lint gates၊ tests တွေနဲ့ guardrails တွေအတွက် hooks သုံးပုံ၊ subagent patterns တွေဖြစ်တဲ့ planner / implementer / reviewer အခန်းကဏ္ဍ ခွဲဝေပုံများ။

လက်တွေ့တည်ဆောက်ရန် Repo instructions ဖိုင်တစ်ခု၊ hooks 2 ခုနဲ့ planner / implementer / reviewer ခွဲထုတ်တာဝန်ပေးတဲ့ System တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 40 min

ဆောင်းပါး · Anthropic

Claude Code best practices

30 min

`CLAUDE.md`၊ tool allowlists၊ workflows နဲ့ multi-Claude patterns တွေအတွက် အဓိက လမ်းညွှန်ချက်။

ဆောင်းပါး · Anthropic

Steering Claude Code: skills, hooks, rules, subagents, and more

15 min

ယန္တရားတစ်ခုချင်းစီကို ဘယ်အချိန်မှာ သုံးရမလဲဆိုတဲ့ အချက်များ။

Course units · Hugging Face ≈ 90 min

Sub-agents & Hooks

Sub-agents များနှင့် hooks များ အခန်း။

Video · Anthropic

▸ Mastering Claude Code in 30 minutes

28 min

Boris Cherny ကိုယ်တိုင် configuration ပြင်ဆင်ပုံတစ်ခုလုံးကို ရှင်းပြထားချက်။

Video · Boris Cherny

▸ Inside Claude Code With Its Creator

50 min

Terminal design၊ `CLAUDE.md` ၊ teams၊ subagents၊ plan mode နဲ့ coding လောကရဲ့ အနာဂတ်။

Tools and References

Hooks Reference & Create Custom Subagents (Claude Code docs)

Build မှာ သုံးရမယ့် event names၊ matchers နဲ့ frontmatter သတ်မှတ်ချက်များ။

Repo Instructions အကြောင်း လေ့လာရန်. GitHub ရဲ့ *How to write a great agents.md: lessons from over 2,500 repositories*၊ Real Python ရဲ့ *How to Write an AGENTS.md File*၊ Agentic AI Foundation ရဲ့ *Writing an Effective AGENTS.md*။

Subagent အခြေအတင် ဆွေးနွေးချက်များ. Cognition ရဲ့ *Don't Build Multi-Agents* နဲ့ Anthropic ရဲ့ *How we built our multi-agent research system*။ Role 3 ခု မခွဲခင် ဒီဆောင်းပါး 2 ခုလုံးကို အရင် ဖတ်ထားပါ။

How Anthropic teams use Claude Code

Anthropic အဖွဲ့တွင်း လက်တွေ့အသုံးချနေတဲ့ ပုံစံများ။

Anthropic Academy

Claude Code in Action

နှင့် **Claude Code: A Highly Agentic Coding Assistant** *(DeepLearning.AI):* အခမဲ့ သင်တန်းများ။

အပိုဆောင်း Video များ

ဗီဒီယို · 1 h 37 min

▸ AI Coding Workflow: From Product Idea to Tested Implementation

Matt Pocock

PRDs၊ vertical slices၊ unattended agents၊ QA၊ review နဲ့ parallelization များ။

မီဒီယို 18 min

▸ Claude Code & the evolution of agentic coding

Boris Cherny

မီဒီယို 1 h 10 min

▸ The Secrets of Claude Code From the Engineers Who Built It

Every

Boris Cherny နဲ့ Cat Wu တို့ ပါဝင်ထားတဲ့ Latent Space အင်တာဗျူးဖြစ်တဲ့ Claude Code: Anthropic's Agent in Your Terminal ကိုလည်း ကြည့်ရှုနိုင်ပါတယ်။

မီဒီယို 30 min

▸ I'm HOOKED on Claude Code Hooks: Advanced Agentic Coding

IndyDevDan

Pre/post tool hooks၊ logging နဲ့ အန္တရာယ်ရှိတဲ့ command တွေကို ရပ်တန့်ပုံ။

Anthropic

Claude Code Advanced Patterns: Subagents, MCP, and Scaling to Real Codebases

DHH with Lex Fridman

▸ Future of Programming, AI, Modern Software Engineering, Vibe Coding and Linux

Programming-with-agents၊ vibe-coding-versus-engineering၊ agent setup၊ model နဲ့ harness အခန်းတွေကိုပဲ ရွေးကြည့်ပါ။ timestamped transcript ကြောင့် စိတ်ကြိုက် ရွေးချယ်ကြည့်ရှုရ လွယ်ကူစေပါတယ်။

လက်တွေ့ တည်ဆောက်ရန် (Build)

ကိုယ့် repo ထဲမှာ တိုတိုရှင်းရှင်း instruction file တစ်ခု ထည့်ပါ။ စွယ်စုံကျမ်းလို အကုန်လျှောက်မရေးဘဲ လမ်းပြမြေပုံ (map) လိုမျိုး အဓိက အချက်တွေကိုပဲ ထည့်ပါ - architecture ဝင်ပေါက်များ၊ အဓိက သုံးရမယ့် commands၊ စည်းမျဉ်းများ (conventions)၊ safety ဘောင်များနဲ့ အသေးစိတ် စာရွက်စာတမ်းတွေဆီ သွားတဲ့ links များ။

အမြန်စစ်ဆေးပေးမယ့် lint သို့မဟုတ် test gate အတွက် deterministic hook တစ်ခု ရေးပါ။

အန္တရာယ်ရှိတဲ့ command တွေကို တားဆီးပေးမယ့် safety hook တစ်ခု ထည့်ပါ။

အခန်းကဏ္ဍ 3 ခု ခွဲပါ - **planner**၊ **implementer** နဲ့ သီးခြားစစ်ဆေးမယ့် **reviewer**။ တစ်ခုချင်းစီအတွက် ရှင်းလင်းတိကျတဲ့ တာဝန်သတ်မှတ်ချက် (contract) ပေးပါ။

Feature တစ်ခုတည်းကို agent တစ်ခုတည်းနဲ့ တစ်ကြိမ်၊ role 3 ခု ခွဲထားတဲ့ workflow နဲ့ တစ်ကြိမ် စမ်းသပ် run ကြည့်ပါ။ ကုန်သွားတဲ့ အချိန်၊ tokens ပမာဏ၊ တွေ့ရှိတဲ့ ချို့ယွင်းချက် (defects) နဲ့ လူကိုယ်တိုင် ဝင်ပါဖြေရှင်းပေးရတဲ့ အကြိမ်ရေတို့ကို နှိုင်းယှဉ်ပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Instruction file ဟာ စာကြောင်းရေ 150 အောက် ဖြစ်ရမယ်။ Agent အသစ်တစ်ခုက ဒီဖိုင် တစ်ခုတည်းကို ဖတ်ပြီး setup လုပ်တာ၊ tests တွေနဲ့ lint စစ်တာကို လုပ်ဆောင်နိုင်ရမယ်။
- [ ] Lint/test hook ဟာ သက်သက်မှားရေးထားတဲ့ commit ကို တားဆီးနိုင်ရမယ်။ Safety hook ဟာ အန္တရာယ်ရှိအောင် သက်သက်ခိုင်းတဲ့ command ကို ပိတ်ပင်နိုင်ရမယ်။
- [ ] Single-agent နဲ့ three-role workflow နှိုင်းယှဉ်ချက်ဇယား (အချိန်၊ tokens၊ တွေ့တဲ့ ချို့ယွင်းချက်၊ လူဝင်ပါရမှု) အပြည့်အစုံ ရှိရမယ်။
- [ ] ကိန်းဂဏန်းအထောက်အထားတွေကို ကိုးကားပြီး subagent 3 ခု ခွဲသုံးတာဟာ ကုန်ကျစရိတ်/အချိန်နဲ့ တန်သလားဆိုတဲ့ သုံးသပ်ချက် 1 ပိုဒ် ရေးသားထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 5 ပတ်မြောက်

Agent-Ready Codebases

Core ≈ 3 h 30 min · ဗီဒီယို 3 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် Codebase တစ်ခုကို agent-ready ဖြစ်စေတဲ့ အချက်များ (structure၊ docs၊ tests နဲ့ checks)။ အသင့်ဖြစ်မှုကို အကဲဖြတ်စစ်ဆေးပြီး score သတ်မှတ်ပုံ။ တကယ့် codebase တွေမှာ agent တွေကို ပိတ်ဆို့လေ့ရှိတဲ့ အဖြစ်များသော အားနည်းချက်များ။

လက်တွေ့တည်ဆောက်ရန် Score သတ်မှတ်ထားတဲ့ readiness audit ပြုလုပ်ခြင်းနဲ့ agent အသစ်တစ်ခု ချက်ချင်းအလုပ်ဖြစ်စေမယ့် ပြုပြင်မှုများ ပြုလုပ်ခြင်း။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 30 min

Case study · OpenAI

Harness engineering: leveraging Codex in an agent-first world

30 min

ဒီအပတ်ရဲ့ အရေးကြီးဆုံး ဖတ်စရာ - repo ရှင်းလင်းမှု၊ စနစ်တကျ ရေးထားတဲ့ docs၊ စက်က အလိုအလျောက် ထိန်းကွပ်ထားတဲ့ invariants တွေနဲ့ progressive disclosure အကြောင်း။

လက်တွေ့လမ်းညွှန် · Adobe ≈ 60 min

Repository Harnesses for AI Coding Agents

Agent တွေ အလွယ်တကူ နားလည်နိုင်ပြီး verify လုပ်နိုင်မယ့် repo ပြင်ဆင်နည်း အခမဲ့ လမ်းညွှန်။

ဆောင်းပါး · Martin Fowler

Harness Engineering

20 min

Video များ · Factory

▸ Making Codebases Agent-Ready & Building Reliable Agentic Systems

34 min

Mechanical verification၊ linters၊ end-to-end tests၊ interface docs၊ planning၊ grounding နဲ့ လူကိုယ်တိုင် ကြီးကြပ်မှုများ။

ဆောင်းပါး · Hamel Husain

Your AI Product Needs Evals

20 min

Evaluation လုပ်တဲ့ အလေ့အကျင့်ကို ဒီကတည်းက စတင်ပါ (Week 10 ဟာ ဒီအပေါ်မှာ အခြေခံပါလိမ့်မယ်)။

ဆောင်းပါး · Beyang Liu of Sourcegraph

▸ The ROI of AI: Why You Need Eval Frameworks

25 min

တိကျသေချာတဲ့ evalsı repo contextı engineering KPIs နဲ့ အပေါ်ယံ အလုပ်ဖြစ်ပြရုံ ကုန်ထုတ်စွမ်းအားတု (productivity theater) ကို ရှောင်ရှားပုံ။

ပိုလေ့လာချင်သူများအတွက် (Deeper Material)

Unlocking the Codex harness (OpenAI)

မီဒီယို

▸ Context Engineering (Dex Horthy)

အင်တာဗျူးထဲက harnessı loop နဲ့ software-factory အခန်းများကို ပြန်လည်ကြည့်ရှုပါ။

Diátaxis Framework

Agent တွေ သွားလာဖတ်ရှုရ လွယ်ကူစေမဲ့ Folder Structure and System (tutorials, how-tos, reference, explanation)။

Context Rot (Chroma)

Context ထဲ အကုန်ပုံထည့်နေတာထက် ကျစ်လျစ်ပြီး index သေချာလုပ်ထားတဲ့ repo က ဘာကြောင့် ပိုသာလဲ ဆိုတဲ့ အချက်။

အပိုဆောင်း Video များ

မီဒီယို 1 h 15 min

▸ TDD, AI agents and coding

Kent Beck with The Pragmatic Engineer

Agent တွေ Code ရေးတဲ့အခါ test-driven development ဟာ ဘာကြောင့် စွမ်းအားတစ်ခု ဖြစ်လာသလဲ ဆိုတာ။

ဗီဒီယို 23 min

▸ **When to Build Your Own Agent Harness**

Harrison Chase of LangChain

လက်တွေ့ တည်ဆောက်ရန် (Build)

ကိုယ့် repo ကို အချက် 5 ချက်နဲ့ audit စစ်ဆေးပါ -

1. **Discoverability.** (အချက်အလက် ရှာဖွေရ လွယ်ကူမှု)
2. **Setup.** (စတင် run ရ လွယ်ကူမှု)
3. **Feedback speed.** (စစ်ဆေးမှု ရလဒ်ထွက်နှုန်း မြန်ဆန်မှု)
4. **Architecture enforcement.** (Architecture ဘောင်ကို ထိန်းကွပ်ထားနိုင်မှု)
5. **Recovery from failure.** (မှားယွင်းမှုကနေ အမြန်ပြန်လည် ပြင်ဆင်နိုင်မှု)

တစ်ချက်ချင်းစီကို အထောက်အထားနဲ့တကွ 1 မှ 5 အထိ score သတ်မှတ်ပါ။ ပြင်ပက လူတစ်ယောက်/စက်တစ်လုံး အနေနဲ့ repo ကို clone ဆွဲတာနဲ့ တစ်ဆင့်စကားမေးစရာမလိုဘဲ နားလည်နိုင်၊ install လုပ်နိုင်၊ test စစ်နိုင်ပြီး code ပြင်နိုင်တဲ့အထိ တိုးတက်အောင် ပြုပြင်ပါ။

အနည်းဆုံးအားဖြင့် - Command တစ်ခုတည်းနဲ့ setup ဖြစ်ရမယ်၊ အမြန်စစ်ဆေးပေးမယ့် deterministic test path တစ်ခု ပါရမယ်၊ architecture မြေပုံတိုလေးတစ်ခု ပါရမယ်၊ စည်းမျဉ်းတွေကို သက်ဆိုင်ရာ code တွေအနီးမှာ ထားပေးရမယ်၊ ချိုးဖောက်ရင် အလိုအလျောက် ဖမ်းပေးမယ့် architectural boundary check တစ်ခု ထည့်ရပါမယ်။

ပြီးနောက် agent အသစ်တစ်ခုဆီကို ဘာအပိုအကူအညီမှ မပေးဘဲ issue တစ်ခု ဖြေရှင်းခိုင်းကြည့်ပါ။ အဲဒီ agent မျက်စိလည်သွားတဲ့ နေရာတိုင်းကို မှတ်တမ်းတင်ပြီး ပြန်လည်ပြင်ဆင်ပါ။ အားလုံးပြီးရင် score ပြန်တွက်ပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] အချက် 5 ချက်စလုံးအတွက် အထောက်အထား အပြည့်အစုံနဲ့တကွ ပြုပြင်မွမ်းမံမှု မတိုင်မီနဲ့ နောက်ပိုင်း readiness scores များ ရှိရမယ်။
- [ ] Command တစ်ခုတည်းနဲ့ clone အသစ်ကို setup လုပ်နိုင်ရမယ်။ Command တစ်ခုတည်းနဲ့ deterministic test တွေကို 2 minအောက်အတွင်း run နိုင်ရမယ်။
- [ ] Architectural boundary တစ်ခုကို စစ်ဆေးတဲ့ check ပါဝင်ရမယ် (ချိုးဖောက်ပါက CI မှာ fail ဖြစ်ရမယ်)။
- [ ] Agent အသစ် စမ်းသပ်မှုအတွင်း မျက်စိလည်သွားတဲ့ အချက်တိုင်းနဲ့ ကိုယ်တိုင် ပြန်လည်ဖြေရှင်းပေးခဲ့ရတဲ့ ပြင်ဆင်ချက်တွေကို log မှတ်တမ်းတင်ထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 6 ပတ်မြောက်

Agentic Code Review

Core ≈ 3 h 30 min · မီဒီယို 1 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် AI review က ဘာတွေကို ကောင်းကောင်း ဖမ်းမိပြီး ဘာတွေကို လွတ်သွားလေ့ရှိသလဲ။ Review architectures နဲ့ စိတ်ကြိုက် rules တွေ ရေးဆွဲပုံ။ အသင်းအဖွဲ့ရဲ့ pull request workflow ထဲ AI review ကို အံဝင်ခွင်ကျ ထည့်သွင်းနည်းများ။

လက်တွေ့ တည်ဆောက်ရန် Pull requests တွေနဲ့ ချိတ်ဆက်ထားပြီး PR 5 ခုပေါ်မှာ တိုင်းတာစမ်းသပ်ထားတဲ့ severity အလိုက် review စံသတ်မှတ်ချက် (rubric) တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 30 min

Google

Engineering Practices: Code Review & Reviewer Guide

60 min

Google ရဲ့ စံပြ code review လမ်းညွှန်ချက်များ။

အခမဲ့ စာအုပ်အခန်း · 45 min

Software Engineering at Google (Chapter 9: Code Review)

Talk · Tomas Reimers of Graphite

▸ AI-powered entomology: lessons from millions of AI code reviews

30 min

Graphite ရဲ့ ရေးသားထားတဲ့ လမ်းညွှန်ဖြစ်တဲ့ AI code review implementation and best practices နဲ့ တွဲဖက်ဖတ်ရှုပါ။

Video · Ankit Jain

▸ How to Kill the Code Review

16 min

Specs၊ reusable guardrails၊ deterministic checks၊ test plans၊ previews နဲ့ human alignment တွေ ပေါင်းစပ်ထားတဲ့ အလွှာ 5 လွှာပါ trust model။

ဆောင်းပါး · Geoffrey Litt

▸ Understanding Is the New Bottleneck

20 min

Review ဆိုတာ အမှားစစ်ရုံသက်သက် မဟုတ်ဘဲ architecture ကို နားလည်စေခြင်း၊ အချင်းချင်း လမ်းပြပေးခြင်းနဲ့ ညှိနှိုင်းဆောင်ရွက်ခြင်းဖြစ်ကြောင်း၊ automation ဖြင့် အစားမထိုးနိုင်တဲ့ အချက်များ။

Integrations · 30 min

GitHub Copilot code review & Claude Code GitHub Actions

PR ပေါ်မှာ reviewer agent ထားရှိနိုင်တဲ့ အသုံးအများဆုံး နည်းလမ်း 2 ခု။

From Cognition & ပိုလေ့လာချင်သူများအတွက်

Don't Build Multi-Agents (Cognition)

Context မျှဝေသုံးစွဲပုံနဲ့ reviewer agent တစ်ခုအနေနဲ့ trace အပြည့်အစုံ ဘာကြောင့် လိုအပ်သလဲဆိုတဲ့ အချက်။

Video · Latent Space with Cognition

▸ DeepWiki: The GitHub Encyclopedia

32 min

SWE-bench technical report & Devin: Coding Agents 101 (Cognition)

သုတေသနစာတမ်း) နှင့် [Google Research: Resolving code review comments with ML] (https://research.google/blog/resolving-code-review-comments-with-ml/

AI-Assisted Assessment of Coding Practices in Modern Code Review

မီဒီယို · 10 min

▸ Your Coding Agent Doesn't Always Follow Your Rules

Hooksı deterministic checksı asynchronous verificationı reviewer agents နဲ့ LLM-as-judge tradeoffs များ။

လက်တွေ့တည်ဆောက်ရန် (Build)

Severity (ပြင်းထန်မှုအဆင့်) အလိုက် စီထားတဲ့ review rubric တစ်ခု ရေးဆွဲပါ -

1. **Correctness.** (မှန်ကန်မှု)
2. **Security.** (လုံခြုံရေး)
3. **Data loss.** (ဒေတာ ဆုံးရှုံးနိုင်ခြေ)
4. **Concurrency.** (ပြိုင်တူလုပ်ဆောင်မှုဆိုင်ရာ ပြဿနာများ)
5. **Compatibility.** (ကိုက်ညီမှု)
6. **Tests.** (စမ်းသပ်ချက်များ ပြည့်စုံမှု)
7. **Maintainability.** (ထိန်းသိမ်းရ လွယ်ကူမှု)
8. **Style.** (Code ရေးဟန်)

Reviewer agent ကို line အတိအကျ ထောက်ပြခိုင်းပါ၊ ဖြစ်လာနိုင်တဲ့ failure scenario ကို ရှင်းပြခိုင်းပါ၊ အသေးငယ်ဆုံး ပြင်ဆင်ရမယ့် fix ကို အဆိုပြုခိုင်းပါ။ Code ကို ရေးခဲ့တဲ့ agent/session မဟုတ်တဲ့ သီးခြား model/session တစ်ခုကို သုံးပြီး review စစ်ခိုင်းပါ။ အထက်ပါ integrations တစ်ခုခုကို သုံးပြီး ကိုယ့်ရဲ့ pull requests တွေနဲ့ ချိတ်ဆက်ပါ။

အနည်းဆုံး PR 5 ခုပေါ်မှာ စမ်းကြည့်ပါ (bug တွေကို တမင်ထည့်ထားတဲ့ PR 1 ခု ပါဝင်ရမယ်)။ မှတ်ချက်တစ်ခုချင်းစီကို **true positive** (အမှား အစစ်)၊ **false positive** (မှားယွင်း ထောက်ပြမှု)၊ **duplicate** (ထပ်နေမှု) သို့မဟုတ် **low-value** (တန်ဖိုးမရှိသော မှတ်ချက်) ဆိုပြီး label တပ်ပါ။ Acceptance rate နဲ့ လွတ်သွားတဲ့ bugs တွေကို မှတ်တမ်းတင်ပါ။ အတည်ပြု merge လုပ်ပိုင်ခွင့်ကိုတော့ လူကသာ ဆက်လက် ကိုင်တွယ်ရပါမယ်။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Review rubric ကို repo ထဲ ထည့်သွင်းထားပြီး ဖြစ်ရမယ်။ PR တင်တာနဲ့ reviewer agent က အလိုအလျောက် စစ်ဆေးပေးရမယ်။
- [ ] စစ်ဆေးထားတဲ့ PR 5 ခုစလုံးရဲ့ comment တိုင်းကို label တပ်ထားပြီး precision ရာခိုင်နှုန်းကို တွက်ချက်ထားရမယ်။
- [ ] Bug သက်သက်ထည့်ထားတဲ့ PR အစီရင်ခံစာထဲမှာ ဘယ် bugs တွေကို ဖမ်းမိပြီး ဘယ် bugs တွေ လွတ်သွားလဲဆိုတာ ဖော်ပြထားရမယ်။
- [ ] Reviewer အနေနဲ့ အမြဲတစေ လွတ်သွားလေ့ရှိတဲ့ အချက်တွေနဲ့ အဲဒါတွေကို ကာကွယ်ဖို့ ဘယ် deterministic check တွေ ထည့်သွင်းလိုက်တယ်ဆိုတဲ့ သုံးသပ်ချက် 1 ပိုဒ် ရေးသားထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 7 ပတ်မြောက်

Security

Core ≈ 3 h 50 min · မီဒီယို 2 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် SAST/SCA၊ dependencies နဲ့ secret-leak အားနည်းချက်များ။ Prompt injection နဲ့ agent တွေမှာ သီးသန့်ကြုံရတဲ့ attack surfaces များ။ Agent အကူအညီနဲ့ vulnerability တွေကို စစ်ဆေးခွဲခြားခြင်း (triage) နဲ့ ဖြေရှင်းခြင်း (remediation)။

လက်တွေ့တည်ဆောက်ရန် Threat model တစ်ခု၊ CI အတွင်း SAST / SCA / secret scans များ ထည့်သွင်းခြင်းနဲ့ prompt-injection စမ်းသပ်မှုတစ်ခု ပြုလုပ်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 50 min

ခြိမ်းခြောက်မှု စာရင်းများ · 60 min

OWASP MCP Top 10 & OWASP Top 10 for LLM Applications

Simon Willison

The lethal trifecta for AI agents & Prompt injection explained

25 min

Embrace The Red

GitHub Copilot: Remote Code Execution via Prompt Injection (CVE-2025-53773)

15 min

Coding agent တစ်ခုကို အစအဆုံး တိုက်ခိုက်ပြထားတဲ့ attack chain လက်တွေ့ဖြစ်ရပ်။

Video · Isaac Evans of [Semgrep](https://semgrep.dev/docs/)

▸ When AI Writes Code: Rethinking App Security

45 min

AI ရေးတဲ့ Code တွေမှာ ပါလာတတ်တဲ့ အားနည်းချက်များ၊ SAST၊ security assistants၊ feedback loops နဲ့ coding agents တွေကြောင့် ဖြစ်လာတဲ့ အန္တရာယ်များ။

Semgrep

Finding vulnerabilities in modern web apps using Claude Code and OpenAI Codex

20 min

Agent အကူအညီနဲ့ triage ပြုလုပ်ပုံ လက်တွေ့။

Fouad Matin of OpenAI

▸ Safety and Security for Code-Executing Agents

14 min

Remote code execution၊ prompt injection၊ ဒေတာ ခိုးထုတ်မှု (exfiltration)၊ containers၊ network ကန့်သတ်ချက်များ၊ approvals နဲ့ OS-level sandboxing။

PortSwigger [Web Security Academy](https://portswigger.net/web-security) ≈ 60 min

Web LLM attacks labs

Indirect prompt injection ပါဝင်တဲ့ လက်တွေ့ lab 2 ခုကို စမ်းသပ်ပါ။

Tools and References

Scanners. SAST အတွက် Semgrep သို့မဟုတ် CodeQL၊ secrets အတွက် `gitleaks`၊ dependencies/SCA အတွက် `OSV-Scanner`။ (Build မှာ ဒီ 3 မျိုးစလုံး ထည့်သွင်းရပါမယ်)။

Sandboxing & Rules. Anthropic ရဲ့ sandboxing လမ်းညွှန်နဲ့ `claude-code-security-review` GitHub Action။

OWASP စာတမ်းများ. MCP Tool Poisoning၊ Agentic AI Threats and Mitigations နဲ့ OWASP GenAI Security Project။ Invariant Labs ရဲ့ မူရင်း ထုတ်ပြန်ချက်မှာ MCP Security Notification: Tool Poisoning Attacks ဖြစ်ပါတယ်။

သုတေသနများ. Design Patterns for Securing LLM Agents against Prompt Injections၊ NIST ရဲ့ Strengthening AI Agent Hijacking Evaluations။

စာအုပ်. Adam Shostack ရဲ့ Threat Modeling: Designing for Security — ပထမအကြိမ် ထုတ်ဝေမှုကို သုံးပါ၊ AI ကမ္ဘာအတွက် အမည်ပြောင်းထားတဲ့ second edition ကိုတော့ 2027 ဖေဖော်ဝါရီမှာ ထွက်ရှိမယ်လို့ ကြေညာထားပါတယ်။

အပိုဆောင်း Video များ

Video 38 min

▸ Why and How You Need to Sandbox AI-Generated Code

Harshil Agrawal of Cloudflare

capabilities, secrets, networking, cleanup, isolation surfaces နဲ့ indirect prompt injection အကြောင်း။

Video 12 min

▸ Web Security Academy series introduction

Rana Khalil

PortSwigger labs အတွက် အထောက်အကူပြု Video။

လက်တွေ့ တည်ဆောက်ရန် (Build)

Agent workflow တစ်ခုလုံးကို threat-model ရေးဆွဲပါ - User inputs၊ repo အချက်အလက်များ၊ ဆွဲယူလာတဲ့ web pages၊ tools၊ credentials၊ MCP servers၊ ထွက်လာတဲ့ commands၊ logs နဲ့ deployment အထိ အကုန်ပါဝင်ရပါမယ်။

Least privilege (အနည်းဆုံး လုပ်ပိုင်ခွင့်ပေးခြင်း)၊ စိတ်ချရတဲ့ allowlist များ၊ secrets တွေကို သီးခြားခွဲထားခြင်း၊ sandboxing နဲ့ ပြန်ပြင်မရနိုင်တဲ့ လုပ်ဆောင်ချက်တွေအတွက် လူကိုယ်တိုင် အတည်ပြုချက် (human approval) ရယူတာမျိုးတွေကို ထည့်သွင်းပါ။

CI ထဲမှာ **SAST** (Semgrep/CodeQL)၊ **dependency/SCA** (OSV-Scanner) နဲ့ **secret scans** (gitleaks) တွေကို run ပါ။

စိတ်မချရတဲ့ fixture ဖိုင်တစ်ခုထဲမှာ အန္တရာယ်မရှိတဲ့ **indirect prompt injection စာသားတစ်ခုကို သက်သက် ထည့်သွင်းထားပါ**။ Agent က အဲဒီစာသားကို ညွှန်ကြားချက်အဖြစ် မယူဘဲ data သက်သက် အဖြစ် မှန်မှန်ကန်ကန် သဘောထားကြောင်း စစ်ဆေးပြပါ။

Credential ပေါက်ကြားမှု၊ မသမာတဲ့ tool output ထွက်ပေါ်မှုနဲ့ ငွေကုန်ကြေးသွားတဲ့ အခြေအနေတွေအတွက် ကိုင်တွယ်ဖြေရှင်းမယ့် **incident playbook အတိုတစ်ခု** ရေးဆွဲပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Agent workflow ရဲ့ trust boundary တစ်ခုချင်းစီနဲ့ ၎င်းအတွက် ကာကွယ်မှု နည်းလမ်းတွေ ပါဝင်တဲ့ threat model စာတမ်း ရှိရမယ်။
- [ ] CI ထဲမှာ SAST၊ SCA နဲ့ secret scanning တွေ run နေရမယ်။ High-severity တွေ့တာနဲ့ အလိုအလျောက် block လုပ်ရမယ်။
- [ ] စမ်းသပ်ထည့်သွင်းထားတဲ့ injection စာသားကို agent က အမိန့်အဖြစ် လက်မခံဘဲ လျစ်လျူရှုနိုင်ကြောင်း သက်သေပြတဲ့ transcript မှတ်တမ်း ရှိရမယ်။
- [ ] Incident playbook ထဲမှာ အထက်ပါ အရေးပေါ် အခြေအနေ 3 ခုအတွက် တာဝန်ခံ (owner) နဲ့ လုပ်ဆောင်ရမယ့် ပထမဆုံး အဆင့် 3 ဆင့်ကို တိကျစွာ ရေးသားထားရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 8 ပတ်မြောက်

Background Agents

Core ≈ 3 h 30 min · မီဒီယို 4 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် Asynchronous ဖြစ်ပြီး cloud ပေါ် လွှဲအပ်ခိုင်းစေနိုင်တဲ့ agents များ။ ပြိုင်တူအလုပ်လုပ်တဲ့ parallel agents အုပ်စုကြီး (fleets) ကို စီမံခန့်ခွဲပုံ။ Slack၊ Linear နဲ့ GitHub တို့ကနေ trigger လုပ်တဲ့ issue-to-PR pipelines များ တည်ဆောက်ပုံ။

လက်တွေ့ တည်ဆောက်ရန် Isolation၊ budgets၊ checkpoints တွေနဲ့ retries တွေ ပါဝင်တဲ့ issue-to-PR automated workflow တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 30 min

တစ်ခုလျှင် ≈ 20 min

Cloud Agent Products အကြောင်း

OpenAI ရဲ့ Codex cloud၊ Anthropic ရဲ့ Claude Code on the web၊ Cursor Cloud Agents၊ GitHub Copilot coding agent။ (၎င်းတို့ရဲ့ trigger၊ isolation နဲ့ approval ပုံစံတွေကို နှိုင်းယှဉ်လေ့လာပါ)။

Case study · OpenAI ≈ 60 min

Open-sourcing Symphony & Specification

Issue-to-PR orchestration ကို specification အဆင့်အထိ စနစ်တကျ ရေးဆွဲထားချက်။

Video · Rajesh Bhatia

▸ Cloudflare's AI Engineering Stack: Assisted to Delegated

37 min

Cloudflare အနေနဲ့ လူကူတဲ့ အဆင့်ကနေ agent ဆီ အပြည့်အဝ လွှဲအပ်တဲ့အဆင့်ကို သူတို့ platform ပေါ်မှာ ဘယ်လို ပြောင်းလဲခဲ့သလဲ ဆိုတာ။

Video · Preeti Somal of Temporal

▸ Scaling AI Agents Without Breaking Reliability

15 min

Persistent state၊ orchestration၊ စောင့်ကြည့်နိုင်စွမ်း၊ အလိုအလျောက် ပြန်လည်ကောင်းမွန်မှု (automatic recovery)၊ လူနှင့် ချိတ်ဆက်မှုနဲ့ parallel work များ။

Video · Kyle Jaejun Lee

▸ I Run a Fleet of AI Agents Across Three Machines: Here's What Broke

9 min

စက် 3 လုံးပေါ် agent အုပ်စုကြီး ပြုံင်တူလွှတ်တဲ့အခါ ကြုံရတဲ့ attention limits၊ ပျက်စီးမှုများနှင့် ပြန်လည်ပြင်ဆင်ပုံ လက်တွေ့။

Tools, References & အပိုဆောင်း Video များ

Cloudflare Agents SDK & Sandbox SDK

Stateful agents တွေနဲ့ သီးခြားခွဲထုတ်ထားတဲ့ environment အတွက် platform။

Temporal Durable AI Agent Tutorial

OpenAI Agents integration သဘောတရားများ။

Code Mode (Cloudflare)

Fleets တွေအတွက် tokens 1000 တည်းနဲ့ API တစ်ခုလုံးကို ပေးစွမ်းနိုင်တဲ့ tool-efficiency Technique။

Jules (Google)

နှိုင်းယှဉ်လေ့လာနိုင်တဲ့ တတိယမြောက် cloud agent။

Common Workflows (Claude Code)

local agent အများအပြားကို ပြုံင်တူ run ရန်အတွက် git worktrees အသုံးပြုပုံ။

Video · 1 h 52 min

▸ Claude Agent SDK: Full Workshop

Thariq Shihipar of Anthropic

Claude Code ကိုယ်တိုင် အခြေခံထားတဲ့ SDK အသုံးပြုပုံ။

Video · 22 min

▸ Building the future of agents with Claude

Anthropic

agent တွေ ရှေ့ဆက်သွားမယ့် ဦးတည်ချက်။

Video · Neil Movva

▸ Why AI Is About to Get 1000x Cheaper

Long-running agents (`05:32`), inference stack (`15:12`), throughput vs latency (`20:03`) နဲ့ transformer hardware (`33:19`) ခေါင်းစဉ်များ။ “1000x” ဆိုတာကို လက်တွေ့ပြသထားတဲ့ ခန့်မှန်းချက်တစ်ခု မဟုတ်ဘဲ ရည်မှန်းချက်တစ်ခုအဖြစ်သာ မှတ်ယူပါ။

လက်တွေ့ တည်ဆောက်ရန် (Build)

Issue တစ်ခုကနေ pull request အထိ နောက်ကွယ်ကနေ အလိုအလျောက် အလုပ်လုပ်ပေးမယ့် background flow တစ်ခုကို တည်ဆောက်ပါ။

Job တစ်ခုချင်းစီအတွက် သီးခြားခွဲထုတ်ထားတဲ့ git worktree (သို့မဟုတ် container)၊ တင်းကျပ်တဲ့ budget သတ်မှတ်ချက်၊ ရပ်နားသိမ်းဆည်းနိုင်တဲ့ checkpoint၊ စနစ်တကျ စစ်ဆေးမှု (deterministic validation) နဲ့ ပြန်လည်စတင်နိုင်တဲ့ resumable state တွေ ထည့်သွင်းပါ။

Bounded backoff ပါတဲ့ retries System ထည့်ပါ၊ agent နှစ်ခုက file တစ်ခုတည်းကို တစ်ပြိုင်နက် ဝင်ပြင်တာမျိုး မဖြစ်အောင် တားဆီးပါ။

Jobs 3 ခုကို တစ်ပြိုင်နက် (parallel) run ပြပါ. - feature အသေးတစ်ခု၊ bug fix တစ်ခုနဲ့ documentation task တစ်ခု။

တစ်ခုကို လမ်းတစ်ဝက်မှာ တမင်တကာ ဖြတ်တောက် (interrupt) ကြည့်ပြီး checkpoint ကနေ ပြန်စနိုင်သလား ဒါမှမဟုတ် clean ဖြစ်စွာ fail ပြနိုင်သလားဆိုတာ စမ်းသပ်ပါ။

Agent တွေက PR ကို အလိုအလျောက် ဖွင့်ခွင့်ရှိပေမဲ့ merge လုပ်တာကိုတော့ လူကပဲ ခွင့်ပြုရပါမယ်။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Issues 3 ခုကနေ စတင်တဲ့ parallel jobs 3 ခု အောင်မြင်စွာ run နိုင်ခဲ့ပြီး တစ်ခုရဲ့ files တွေကို နောက်တစ်ခုက မထိခိုက်စေဘဲ သီးခြား PR ဖွင့်ပေးနိုင်ရမယ်။
- [ ] လမ်းတစ်ဝက်မှာ ဖြတ်တောက်ခံရတဲ့ job ဟာ checkpoint ကနေ ပြန်စနိုင်ခဲ့သလား၊ ဒါမှမဟုတ် သန့်ရှင်းစွာ fail သွားသလားဆိုတာ သက်သေပြနိုင်တဲ့ log ရှိရမယ်။
- [ ] Job တိုင်းဟာ သတ်မှတ် budget အတွင်းမှာပဲ ရှိရမယ်။ Budget ကျော်သွားတဲ့အခါ အလိုအလျောက် ရပ်တန့်တဲ့လမ်းကြောင်းကို အနည်းဆုံး 1 ကြိမ် စမ်းသပ်ထားရမယ်။
- [ ] အထက်ဖော်ပြပါ hosted cloud products တွေကို trigger၊ isolation နဲ့ approval ပုံစံတွေ အပေါ် အခြေခံပြီး နှိုင်းယှဉ်သုံးသပ်ထားတဲ့ မှတ်စု ရှိရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 9 ပတ်မြောက်

Building an AI-Native Team

Core ≈ 3 h 40 min | ဗီဒီယို 1 ခု | စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် MCP portals နဲ့ ခွင့်ပြုချက် အဆင့်ဆင့်ပါဝင်တဲ့ ဗဟို tool access System ။ LLM gateways၊ model routing နဲ့ ကုန်ကျစရိတ် ခွဲဝေတာနည်းများ။ ကုမ္ပဏီ/အဖွဲ့အစည်းတစ်ခုလုံး အတိုင်းအတာနဲ့ လက်ခံအသုံးချမယ့် အလေ့အထများ။

လက်တွေ့တည်ဆောက်ရန် Model gateway တစ်ခု၊ MCP portal တစ်ခုနဲ့ စာမျက်နှာ 1 မျက်နှာပါ adoption policy တစ်ခု ရေးဆွဲရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 40 min

Talk · Uber

▸ Agentic SDLC: Building Blocks for Uber's Software Factory

18 min

Model gateways၊ agent identity၊ PII အချက်အလက်များ ဖျောက်ဖျက်ခြင်း (redaction)၊ MCP gateways၊ tool access၊ context graphs နဲ့ pre-CI validation များ။

Anthropic & Tobin South

▸ Gateways Are All You Need & Enterprise-Ready MCP

32 min

Identity၊ access policy၊ စောင့်ကြည့်နိုင်စွမ်း၊ oversight နဲ့ data-loss prevention များ။

Cloudflare

Scaling MCP adoption: reference architecture for enterprise deployments

25 min

MCP Spec & [Technical specification](https://modelcontextprotocol.io/extensions/auth/enterprise-managed-authorization)

Enterprise-Managed Authorization

20 min

DORA

2025 State of AI-assisted Software Development

60 min

AI ဟာ ရှိရင်းစွဲ System ရဲ့ အားသာချက်ရော အားနည်းချက်ကိုပါ ပိုကြီးသွားစေကြောင်း သုတေသန။ (METR ရဲ့ *Measuring the Impact of Early-2025 AI on Experienced Developer Productivity* ဆောင်းပါးနဲ့ တွဲဖတ်ပါ)။

Video · Amjad Masad of Replit

▸ The Future of Software Creation

42 min

Software ဖန်တီးရတာ ပိုမိုလွယ်ကူ သက်သာလာတဲ့အခါ အဖွဲ့အစည်းတွေ ဘယ်လို ပြောင်းလဲသွားမလဲ ဆိုတာ။

Tools, References & Book

Gateways. LiteLLM AI Gateway (open source) သို့မဟုတ် Cloudflare AI Gateway။ (Routing အတွက် Vercel ရဲ့ *Six LLM routing strategies* ကို ဖတ်ပါ)။

Internal Registries. MCP Registry။

Adoption အထောက်အထားများ. DORA AI Capabilities Model၊ Stack Overflow 2025 Survey၊ How OpenAI uses Codex (PDF)၊ How Anthropic teams use Claude Code။

Video 18 min

▸ Building an Autonomous Engineering Org

Champions၊ repo readiness၊ delegation နဲ့ organizational impact ဆိုင်ရာ maturity model။

Accelerate (စာအုပ်)

Nicole Forsgren, Jez Humble, and Gene Kim။ (AI ကြောင့် output အရေအတွက် သက်သက်များလာတာလား၊ ဒါမှမဟုတ် delivery တကယ်တိုးတက်လာတာလားဆိုတာ တိုင်းတာဖို့ အသုံးဝင်ပါတယ်)။

လက်တွေ့တည်ဆောက်ရန် (Build)

ကိုယ့် System ထဲက model calls တွေအားလုံးကို gateway သို့မဟုတ် proxy တစ်ခုခုရဲ့ နောက်မှာ ထားပါ (LiteLLM သို့မဟုတ် Cloudflare AI Gateway)။

Agent တစ်ခုချင်းစီအတွက် identity၊ budgets၊ model routing၊ fallback အစီအစဉ်၊ cost/latency logs နဲ့ မလိုလားအပ်တဲ့ အချက်အလက်တွေ ဖျောက်ပေးမယ့် redaction rules တွေ ထည့်ပါ။

စိတ်ချရတဲ့ allowlisted tools တွေ၊ သတ်မှတ်ထားတဲ့ scopes၊ စစ်ဆေးနိုင်မယ့် audit logs နဲ့ အချိန်မရွေး ပြန်လည်ရုပ်သိမ်းနိုင်တဲ့ credentials တွေပါဝင်တဲ့ **MCP portal အသေးစားတစ်ခု တည်ဆောက်ပါ**။

1 မျက်နှာပါ အဖွဲ့တွင်း အသုံးပြုမှု မူဝါဒ (team adoption policy). တစ်ခုကို ရေးဆွဲပါ - ခွင့်ပြုထားတဲ့ data အမျိုးအစား၊ လူကိုယ်တိုင် မဖြစ်မနေ ဆုံးဖြတ်ရမယ့် အချက်များ၊ ပြဿနာတက်ရင် တာဝန်ယူရမယ့်သူ၊ review စည်းမျဉ်းများနှင့် တိုင်းတာမယ့် metrics များ။ (Lines of code အရေအတွက်ထက် change-failure rate၊ lead time၊ review burden၊ ကုန်ကျစရိတ်နဲ့ အောင်မြင်စွာ ပြီးမြောက်မှုနှုန်းတို့ကို ဦးစားပေးတိုင်းတာပါ)။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] System ထဲက model call တိုင်း gateway ကနေ ဖြတ်သွားရမယ်။ တိုက်ရိုက်ခေါ်ယူမှုတွေကို block လုပ်ထားရမယ် (သို့မဟုတ်) ချိုးဖောက်မှုအဖြစ် log မှတ်တမ်းတင်ရမယ်။
- [ ] Request အုပ်စု အနည်းဆုံး 1 ခုကို ဈေးသက်သာတဲ့ model ဆီ routing လမ်းကြောင်းလွဲပေးနိုင်ရမယ်။ သက်သာသွားတဲ့ ကုန်ကျစရိတ်ကို log မှာ ပြသနိုင်ရမယ်။
- [ ] MCP portal ကနေ allowlist ထဲက tools တွေကိုပဲ ဖွင့်ပေးထားရမယ်။ Credential တစ်ခုကို ရုပ်သိမ်းလိုက်တာနဲ့ တစ်ကြိမ်အတွင်း access ပြတ်တောက်သွားရမယ်။
- [ ] Adoption policy ဟာ စာမျက်နှာ 1 မျက်နှာတည်းနဲ့ ပြည့်စုံရမယ်၊ ပြဿနာဖြစ်လာပါက တာဝန်ယူဖြေရှင်းမယ့်သူ (owner) အမည် ပါဝင်ရမယ်။

ရက်သတ္တပတ် 10 ပတ်အနက် 10 ပတ်မြောက်

The Software Factory + The Future

Core ≈ 3 h 30 min · မီဒီယို 5 ခု · စံသတ်မှတ်ချက် 4 ခု

အဓိက ဦးတည်ချက် အလိုအလျောက် လည်ပတ်ပြီး ကိုယ့်ကိုယ်ကိုယ် ပြန်လည်ကောင်းမွန်အောင် လုပ်ဆောင်နိုင်တဲ့ software factory System များ။ Deployment ပြုလုပ်ပြီးနောက်ပိုင်း agents များကို စောင့်ကြည့်ခြင်းနှင့် လုံခြုံရေး ထိန်းသိမ်းခြင်း။ AI software engineering ရဲ့ အနာဂတ် ဦးတည်ရာများ။

လက်တွေ့တည်ဆောက်ရန် Eval suite တစ်ခုနဲ့ စနစ်တကျ ထိန်းချုပ်ထားတဲ့ improvement loop ပါဝင်တဲ့ traced end-to-end factory System တစ်ခု တည်ဆောက်ရပါမယ်။

အဓိက လေ့လာစရာများ (Core) ≈ 3 h 30 min

Keynote · Andrej Karpathy

▸ Software Is Changing (Again)

39 min

Case study · Warp

The self-improvement loop in a software factory

20 min

Dex Horthy

▸ Harness Engineering Is Not Enough: Why Software Factories Fail

19 min

Brownfield System တွေမှာ ထိန်းသိမ်းရခက်ခဲမှု၊ အားနည်းတဲ့ review များ၊ agent တွေ ထုတ်ပေးတဲ့ Code အမြောက်အများကြောင့် architecture ပျက်စီးယိုယွင်းလာမှုများ။

မီဒီယို

25 min) နှင့် [Google SRE: Postmortem Culture](https://sre.google/sre-book/postmortem-culture/) (30 min

▸ Always-on agents run production without the on-call tax

Agent တွေ အသုံးပြုတဲ့ System တွေမှာ မရှိမဖြစ် လိုအပ်တဲ့ စည်းကမ်းချက်များ။

Arize AI & [Phoenix](https://arize.com/docs/phoenix/tracing/tutorial/your-first-traces)

▸ How to Debug AI Agents: Tracing, Observability & Evals

49 min

OpenAI

Symphony Specification

20 min

Factory build ကို စိတ်ထဲထားပြီး Symphony ရဲ့ system design ကို ပြန်လည်လေ့လာပါ။

Tools, References & အပိုဆောင်း Video များ

DeepLearning.AI ≈ 2 h

Evaluating AI Agents

Tracing၊ trajectory evaluation၊ LLM judges နဲ့ production monitoring။

Langfuse & OpenTelemetry GenAI Conventions

Phoenix အစား သုံးနိုင်တဲ့ open-source စောင့်ကြည့်ရေး System ။

DSPy

Prompts တွေကို လက်နဲ့ လိုက်မပြင်ဘဲ eval set အပေါ် မူတည်ပြီး အလိုအလျောက် optimize လုပ်ပေးတဲ့ System ။

Benchmarks. SWE-bench နဲ့ Terminal-Bench။

Google SRE: Introduction

agent တွေနဲ့ လည်ပတ်တဲ့ system တွေမှာ မရှိမဖြစ် လိုအပ်နေဆဲဖြစ်တဲ့ လုပ်ငန်းလည်ပတ်မှုဆိုင်ရာ စည်းကမ်းများ။

မီဒီယို 1 h 46 min

▸ Why AI evals are the hottest new skill for product builders

Hamel Husain & Shreya Shankar

Eno Reyes

The Software Factory

Deterministic systems၊ harnesses၊ feedback loops အကြောင်း။

ဗီဒီယို 2 h 26 min

▸ **“We're summoning ghosts, not building animals”**

Andrej Karpathy with Dwarkesh Patel

Agent တွေမှာ လက်ရှိအချိန်အထိ ဘာကြောင့် အစီအစဉ်ချနိုင်စွမ်းနဲ့ မှတ်ဉာဏ် မရှိသေးတာလဲ၊ နောက် 10 နှစ်မှာ ဘာတွေ ဖြစ်လာမလဲ ဆိုတာ။

ဗီဒီယို

▸ **Future of Programming (DHH)**

နည်းပညာအမြင်အတွက် အထူးသဖြင့် `03:59:24` ကို ကြည့်ပါ။

လက်တွေ့ တည်ဆောက်ရန် (Build)

အရင်အပတ်တွေက တည်ဆောက်ခဲ့တဲ့ အစိတ်အပိုင်းတွေကို စုစည်းပြီး အခြေခံ software factory တစ်ခု အဖြစ် ချိတ်ဆက်ပါ -

```
issue → aligned spec → isolated agent run → tests/checks → independent review → human merge
```

Run တိုင်းကို **trace လိုက်ပါ** (Phoenix သို့မဟုတ် Langfuse သုံးပါ)။

ကိုယ်စားပြု task 10 ခုမှ 20 ခုပါဝင်တဲ့ ပုံသေ **evaluation set တစ်ခု ဖန်တီးပါ**။ အောင်မြင်မှုနှုန်း၊ regressions၊ စရိတ်၊ ကြာချိန်၊ retries နဲ့ လူဝင်ပါရတဲ့ အကြိမ်ရေတို့ကို မှတ်တမ်းတင်ပါ။

Prompts၊ skills သို့မဟုတ် tools တွေကို ပြင်ဆင်ဖို့ အကြံပြုနိုင်တဲ့ **controlled improvement loop တစ်ခု ထည့်ပါ**။ သို့သော်လည်း အကဲဖြတ်စစ်ဆေးမှု (evaluation) နဲ့ လူကိုယ်တိုင် ခွင့်ပြုချက် (human approval) မပါဘဲ အဲဒီ အပြောင်းအလဲတွေကို deploy လုပ်ခွင့် မရှိစေရပါဘူး။ (“ကိုယ့်ကိုယ်ကိုယ် တိုးတက်စေခြင်း” ဆိုတာ “ကိုယ့်ရဲ့ ထိန်းချုပ်မှု ဘောင်တွေကို တိတ်တဆိတ် ပြန်ပြင်ခွင့်ပေးခြင်း” မဟုတ်ပါ)။

ဒီ 10 ပတ်အတွင်း အဲဒီ factory ကနေ ကြုံတွေ့ခဲ့ရတဲ့ အဆိုးရွားဆုံး failure အတွက် အပြစ်တင်မှုမပါတဲ့ **blameless postmortem တစ်ခု** ရေးသားပါ။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When)

- [ ] Factory လည်ပတ်မှုတိုင်းကို issue ကနေ merge ဆုံးဖြတ်ချက်အထိ Phoenix သို့မဟုတ် Langfuse ထဲမှာ အစအဆုံး trace လိုက်ကြည့်နိုင်ရမယ်။
- [ ] သတ်မှတ်ထားတဲ့ evaluation set ထဲက အနည်းဆုံး task 10 ခုအတွက် မူလရလဒ် (baseline) နဲ့ နောက်ဆုံးရလဒ် (final) တွေ ရှိရမယ်။
- [ ] Improvement loop ကနေ အနည်းဆုံး အပြောင်းအလဲ 1 ခုကို အဆိုပြုခဲ့ရမယ်။ အဲဒီ အပြောင်းအလဲကို လူက လက်ခံ/ငြင်းပယ်ခြင်း မပြုမီ eval နဲ့ အရင် စစ်ဆေးပြနိုင်ခဲ့ရမယ်။
- [ ] အချိန်ဇယား၊ အဓိက အကြောင်းရင်း (root cause) နဲ့ နောင်မဖြစ်အောင် ကာကွယ်မယ့် ပြင်ဆင်ချက် ပါဝင်တဲ့ postmortem စာတမ်း 1 စောင် ရှိရမယ်။

Capstone Project

တကယ့် codebase အစစ်တစ်ခုအတွက် **agentic maintenance system တစ်ခုကို ကိုယ်တိုင် တည်ဆောက်ပါ**။

အဲဒီ System ဟာ တိကျသေချာတဲ့ issue တစ်ခုကို လက်ခံနိုင်ရမယ်၊ သီးခြားခွဲထားတဲ့ workspace တစ်ခုကို ဖန်တီးနိုင်ရမယ်၊ ပြင်ဆင်မယ့်အချက်ကို လေ့လာပြီး အစီအစဉ်ဆွဲရမယ်၊ Code ကို လက်တွေ့ အကောင်အထည်ဖော် ရေးသားရမယ်၊ repo စစ်ဆေးမှုတွေကို run ရမယ်၊ သီးခြား AI review တစ်ခု စစ်ဆေးပေးရမယ်၊ ပြီးရင် လူက စစ်ဆေးအတည်ပြုနိုင်ဖို့ pull request တစ်ခု ဖွင့်ပေးရပါမယ်။

အဓိက တင်ပြရမည့် အရာများ (Deliverables)

- Command တစ်ခုတည်းနဲ့ setup လုပ်နိုင်ပြီး ရှင်းလင်းတဲ့ architecture မြေပုံပါဝင်တဲ့ repository တစ်ခု။
- တိုတိုရှင်းရှင်း ရေးထားတဲ့ `AGENTS.md` (သို့မဟုတ် ၎င်းနှင့်ညီမျှသော repo guide)။
- ပြန်လည်အသုံးပြုနိုင်တဲ့ reusable agent skill တစ်ခုနဲ့ MCP integration တစ်ခု။
- Deterministic hooks၊ tests တွေနဲ့ CI gates များ။
- Threat model တစ်ခုနဲ့ least-privilege permission ဒီဇိုင်း။
- ရပ်တန့်သွားရင် ပြန်စနိုင်တဲ့ durable issue-to-PR background workflow (retries mechanism ပါဝင်ရမယ်)။
- ကုန်ကျစရိတ်၊ latency၊ routing နဲ့ errors တွေကို စောင့်ကြည့်နိုင်မယ့် model gateway telemetry။
- ပုံသေ evaluation suite တစ်ခုနဲ့ အကျဉ်းချုပ် results dashboard သို့မဟုတ် report တစ်ခု။
- 5 minမှ 10 minစာ demo video တစ်ခုနဲ့ ကြုံတွေ့ခဲ့ရတဲ့ အမှားများ၊ တိုးတက်မှုများကို ရှင်းပြထားတဲ့ retrospective သုံးသပ်ချက်တစ်ခု။

ပြီးစီးကြောင်း စစ်ဆေးရန် (Done When / Completion Gates)

Clone အသစ်တစ်ခုကို လမ်းညွှန်ချက်ပါ command တစ်ခုတည်းနဲ့ setup လုပ်နိုင်ရမယ်၊ အမှန်တကယ် အလုပ်ဖြစ်ကြောင်း အတည်ပြုနိုင်ရမယ်။

ပုံသေ evaluation tasks အနည်းဆုံး 10 ခုအတွက် baseline ရော final results ပါ အပြည့်အစုံ ရှိရမယ်။

လမ်းတစ်ဝက်မှာ ဖြတ်တောက်ခံရတဲ့ background job တစ်ခုဟာ ပြန်လည်စတင်နိုင်ခြင်း သို့မဟုတ် သန့်ရှင်းစွာ ရပ်တန့်နိုင်ခြင်းကို သက်သေပြနိုင်ရမယ်။

System က ထုတ်ပေးလိုက်တဲ့ အပြောင်းအလဲတွေဟာ tests တွေ၊ security checks တွေနဲ့ လူကိုယ်တိုင် merge အတည်ပြုချက်တွေကို ကျော်လွှားခွင့် မရှိစေရဘူး။

Run တိုင်းကို သက်ဆိုင်ရာ issue၊ spec၊ prompts/context၊ tool calls၊ outputs၊ costs နဲ့ နောက်ဆုံး review ဆုံးဖြတ်ချက်အထိ အစအဆုံး ပြန်လည် trace လိုက်နိုင်ရမယ်။

အကြံပြု စာအုပ်စင် (The Short Bookshelf)

စာအုပ် ကေရီ အများကြီးကို အစအဆုံး ကုန်အောင် ဖတ်ဖို့ မကြိုးစားပါနဲ့။ ဒီစာအုပ် 5 အုပ်ဟာ ဒီလမ်းညွှန်ရဲ့ နောက်ကွယ်က ခိုင်မာတဲ့ အယူအဆတွေကို လွှမ်းခြုံထားပါတယ် -

1. Chip Huyen, ***AI Engineering*** — foundation model တွေနဲ့ application တွေ တည်ဆောက်ပုံနဲ့ ဆန်းစစ်ပုံ။
2. Jay Alammar နှင့် Maarten Grootendorst, ***Hands-On Large Language Models*** — Code repository နဲ့အတူ လေ့လာနိုင်တဲ့ လက်တွေ့ကျ LLM အခြေခံများ။
3. Titus Winters, Tom Manshreck, နှင့် Hyrum Wright, ***Software Engineering at Google*** — အွန်လိုင်းမှာ အခမဲ့ ဖတ်ရှုနိုင်ပါတယ်၊ အထူးသဖြင့် code review၊ testing၊ dependency management နဲ့ large-scale change အခန်းတွေကို လေ့လာပါ။
4. Adam Shostack, ***Threat Modeling: Designing for Security*** — attack surface မှာ agents နဲ့ tools တွေ ပါဝင်လာချိန်မှာပါ အသုံးဝင်ဆဲဖြစ်တဲ့ security တွေးခေါ်ပုံများ။
5. Nicole Forsgren, Jez Humble, နှင့် Gene Kim, ***Accelerate*** — Agentic System တစ်ခုက တကယ်ပဲ တိုးတက်မှုရှိမရှိ တိုင်းတာခြင်း။

အချိန်တစ်ဝက်သာ ရရှိပါက (The Minimum Path)

လက်တွေ့ တည်ဆောက်မှု (Builds) တွေကို စွဲစွဲမြဲမြဲလုပ်ပြီး ဖတ်စရာ/ကြည့်စရာတွေကို လျှော့ချပါ။ ဒီအနည်းဆုံး လေ့လာရန် လမ်းကြောင်းကို အသုံးပြုပါ -

1. Thorsten Ball ရဲ့ **How to Build an Agent**၊ ပြီးရင် Codex repository ထဲက production prompt file တစ်ခု။
2. Anthropic ရဲ့ **Effective context engineering** နဲ့ **Writing effective tools**၊ ထို့ပြင် MCP specification။
3. Dex Horthy ရဲ့ **Context Engineering** အင်တာဗျူး။
4. Anthropic ရဲ့ **Claude Code best practices** နဲ့ hooks reference။
5. OpenAI ရဲ့ **Harness Engineering** case study။
6. Google ရဲ့ **Code Review** လမ်းညွှန်။
7. OWASP ရဲ့ **MCP Top 10**၊ **lethal trifecta** နဲ့ PortSwigger labs နှစ်ခု။
8. OpenAI ရဲ့ **Symphony** နဲ့ Uber ရဲ့ **Agentic SDLC** talk။
9. DeepLearning.AI ရဲ့ **Evaluating AI Agents** သင်တန်း။

လက်တွေ့ တည်ဆောက်ရတဲ့ အလုပ်ကသာလျှင် အဓိက လမ်းညွှန်ဖြစ်ပါတယ်။ System ကို ကိုယ်တိုင် မတည်ဆောက်ဘဲ အကုန်လုံးကို လိုက်ကြည့်ရုံနဲ့ ဝေါဟာရတွေကိုပဲ သိသွားမှာဖြစ်ပြီး၊ လက်တွေ့ ရေးသားတည်ဆောက်ခြင်းကသာ အင်ဂျင်နီယာ အဆုံးအဖြတ် သုံးသပ်နိုင်စွမ်းကို သင်ယူစေမှာ ဖြစ်ပါတယ်။