itential® AUTOCON 6

NAF AC6 WORKSHOP · WS:C1

LOOP ENGINEERING, SPEC-DRIVEN DEVELOPMENT & THE FUTURE OF VIBEOPS

Your Proctors

All three are on the floor during every lab: raise a hand any time.



John Capobianco

Head of AI & DevRel



Joksan Flores

Principal Solutions Engineer



Elliot Conner

Solutions Engineer

itential

© Itential Confidential and Proprietary

Four Hours

From an Idea to a Working Game on the Internet

Session	Time
Lecture: LLMs, Ralph loops, SDD and Spec Kit	9:00 – 9:35
Demo: Itential Builder Skills, with Joksan and Elliot	9:35 – 9:45
Lecture: loop engineering	9:45 – 9:50
Set up your project	9:50 – 10:00
Lecture: gamification of networking	10:00 – 10:15
Group activity: ideas and templates	10:15 – 10:45
Break	10:45 – 11:15
Group activity, continued	11:15 – 11:30
Bring your idea to life	11:30 – 12:30
Show your games	12:30 – 1:00

Agenda

01 The Journey So Far

02 Preface: LLM, RAG, MCP, Skills and the Open Agent Standards

03 Ralph Wiggum Loops

04 Spec-Driven Development

05 Spec Kit, and Running it as a Ralph Loop

06 Itential Builder Skills and SDD, with a Live Demo from Joksan and Elliot

07 Loop Engineering, and What Comes Next: Graph Engineering

08 Project Setup and the Gamification of Networking

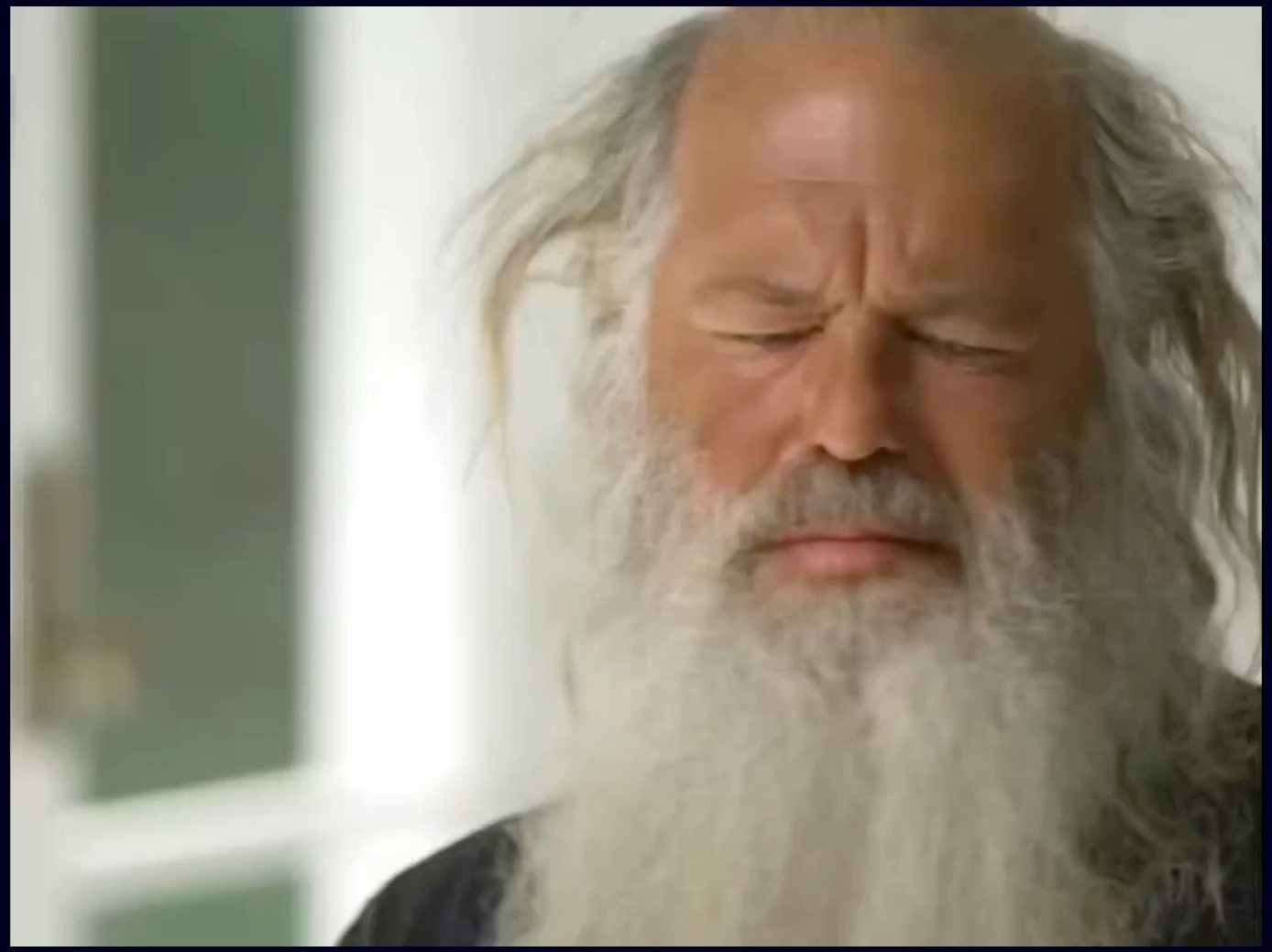
Build agents make the thing.
Run-time agents are the thing doing the work.

Build Agent The SDD and loop that builds the thing		Builds →	Run-Time Agent The operational thing that was built	
WHERE IT RUNS	Your laptop: VS Code with a terminal		WHERE IT RUNS	A server, a platform or the user's browser
HARNESS	Claude Code, Codex or GitHub Copilot in the IDE		CONNECTED TO	A model, and possibly MCPs, RAG and skills
LOADED WITH	Frontier model, skills, MCPs, RAG sources		LOADED WITH	Only the tools its job needs, with permissions
ITS JOB	Write specs, plan, generate and test code		ITS JOB	Operate: create, read, update and delete
WHAT IT PRODUCES	A game, a run-time agent, code, a website, even a SKILL.md or a spec		EXAMPLE	An agent that checks and changes network config

itential

© Itential Confidential and Proprietary

Preface – But I'm not a video game developer



© Itential Confidential and Proprietary

Democratization of a Technology



© Itential Confidential and Proprietary

Today your build harness is loaded with Spec Kit, Ralph and whatever helps you build.

YOUR BUILD HARNESS FOR TODAY

Model	Sonnet 5 or GPT-5.6 (frontier model)	Required
Harness	Claude Code or Codex, in VS Code with a terminal	Required
Spec Kit	/constitution through /implement	Required
Ralph loop	Plugin or bash loop to finish tasks.md	Optional
MCP servers	e.g. NetBox, Blender, pyATS	Optional
RAG and skills	RFCs, vendor docs, a Three.js skill	Optional

Builds →

YOUR RUN-TIME RESULT

A network-themed video game

index.html in a browser, built with Three.js, HTML and CSS

Pushed to your own GitHub repo

Load the build harness with what helps you build. None of it has to ship inside the game.

itential

© Itential Confidential and Proprietary

Should your game be agentic?
For you to decide; enforced by your spec.

Not Agentic: A Deterministic Game

How it works: all questions, answers and levels are written into the code and data

Needs: a browser, nothing else

Cost to play: zero

Risk: low; the constitution's accuracy rule is easy to check

Example: a subnetting quiz with a fixed question bank and written explanations

Recommended starting point for a 60-minute build.

Agentic: The Game Calls A Model

How it works: a model plays a tutor, generates new scenarios or grades free-text answers

Needs: an API key or a local model, maybe an MCP or RAG source

Cost to play: tokens on every turn, or free on a local open-weight model via Ollama (your mileage may vary)

Risk: the model can say something wrong; your spec must guard against it

Example: a BGP tutor character who explains, in its own words, why a route won

This makes your game a run-time agent built by your build agent.

Decide in /specify, settle it in /clarify, and write the rule into your constitution.

itential

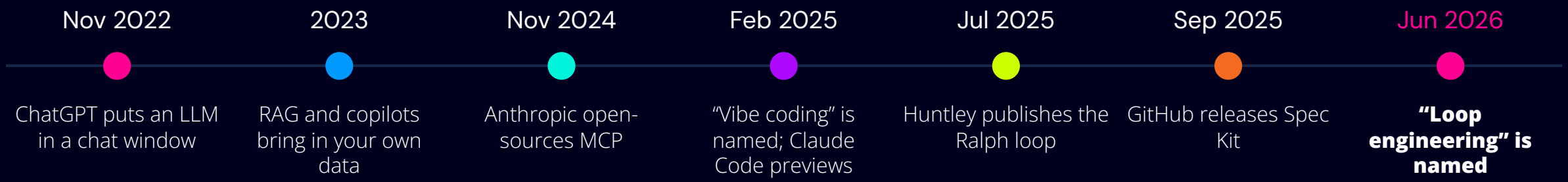
© Itential Confidential and Proprietary

01

THE JOURNEY

How we went from chatting with AI to designing loops that run it.

In four years, AI coding went from a chat window to loops that prompt themselves.



Each step made the model more useful by giving it more context, more tools, or more autonomy.

Each era moved the human one step further from the keystroke.

2022–23

Prompt engineering

You write the words

2024

Context engineering

You choose what the model sees

2025

Harness engineering

You give it tools and a loop

2026

Loop engineering

You design the system that prompts it

itential

VibeOps keeps the speed of vibe coding and adds engineering discipline.

VIBE CODING

Prompt and hope

- No spec: the model fills gaps with guesses
- The human acts as QA, finding every bug
- Fine for prototypes and throwaway demos

VIBEOPS

Intent written down as specs

- A constitution sets rules the agent can't break
- Tests and verifiers close the loop
- Output is deterministic, reviewed and versioned

© Itential Confidential and Proprietary

Production networks can't run on vibes.

A wrong ACL or route policy is an outage, not a bug report. Every change an AI produces has to meet the same bar a human change does.

Deterministic

The same intent produces the same change every time

Reviewed

A human approves the intent and the plan before anything touches a device

Auditable

Every change traces back to a spec, a test result and an approver

Specs, loops and verifiers let us keep these three guarantees at AI speed.

itential

02

PREFACE: LLM, RAG AND MCP

The three building blocks under every coding agent

An LLM predicts the next token from everything in its context window.



Stateless

Nothing carries over between calls unless you send it again

Finite Context

As the window fills, recall and reasoning degrade

Frozen Knowledge

It knows nothing after its training cutoff

The context window is the model's only working memory.

ONE CODING SESSION'S CONTEXT WINDOW (ILLUSTRATIVE)



As the session runs, history and tool output crowd out free space, and quality drops: context rot.

RALPH'S ANSWER

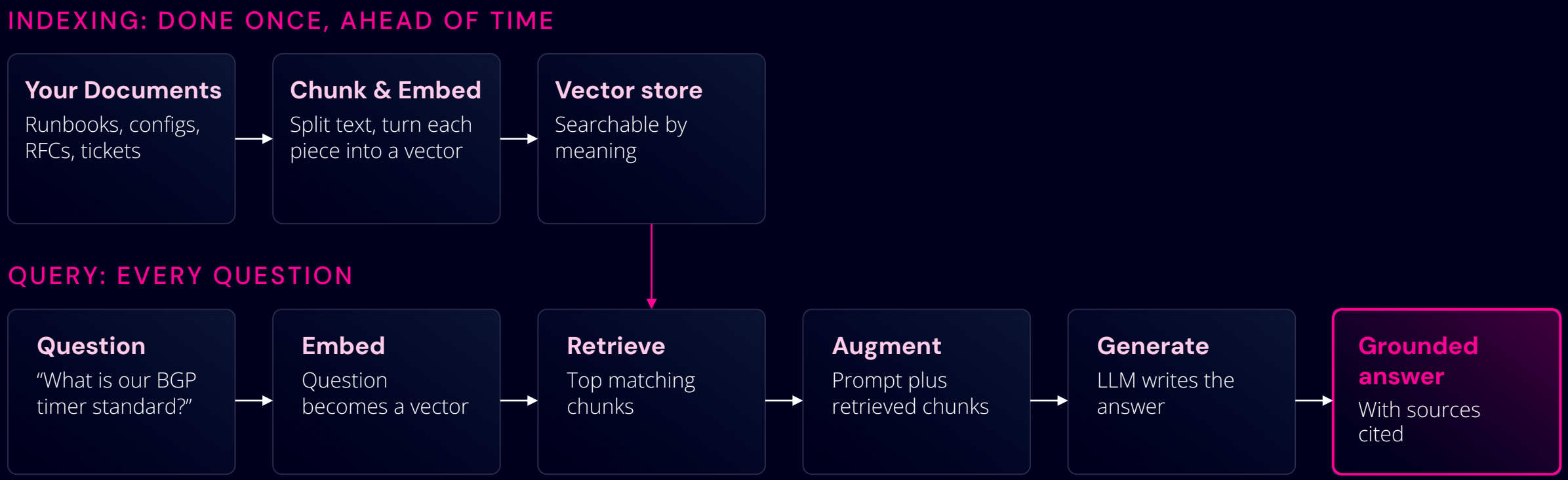
Throw the window away every iteration. Keep progress on disk where the next fresh agent can read it.

SDD'S ANSWER

Load only what the current step needs: the constitution, the spec, the plan and one task.

itential

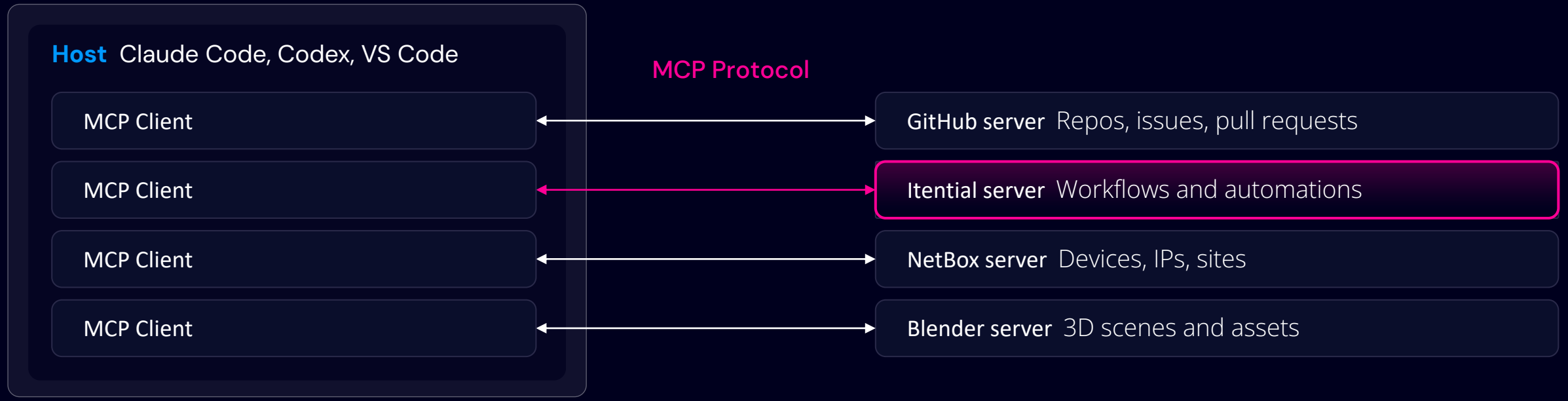
RAG grounds answers in your own documents instead of training data.



RAG fixes frozen knowledge without retraining: the model answers from your documents, and you can check its sources.

itential

MCP gives every model one standard connector for tools and data.



Tools
Actions the model can call

Resources
Data the model can read

Prompts
Reusable templates the server offers

Anthropic open-sourced MCP in November 2024. Build one server and every MCP-capable host can use it.

MCP servers turn network systems into tools an agent can call.

- Sources of truth such as NetBox or Nautobot become resources the agent can read
- Device actions such as show commands become tools, with approval on anything that writes
- Itential exposes workflows and automations to agents through MCP
- Today, Blender and Unreal Engine MCPs let your agent build 3D assets for your game

© Itential Confidential and Proprietary

78 MCP servers already cover 12 layers of the infrastructure stack.

01
Devices & platforms
Itential, pyATS, Arista CVP, ContainerLab

02
Cisco platform suite
ACI, ISE, Meraki, ThousandEyes

03
Source of truth & ITSM
NetBox, Nautobot, Infoblox, ServiceNow

04
Cloud infrastructure
AWS, Azure, Terraform, kubectl

05
Observability
Grafana, Prometheus, Splunk, SuzieQ

06
Incident management
PagerDuty, ServiceNow

07
Security & compliance
NVD CVE, PAN-OS, Zscaler, Prisma AIRS

08
Developer & GitOps
GitHub, GAIT, Ansible AAP

09
Reference & standards
RFC Lookup, Subnet Calculator, Wikipedia

10
Diagramming
Draw.io, Excalidraw, Markmap

11
Packet analysis & lab
Wireshark, Packet Buddy, Kubeshark

12
Assurance & digital twin
Batfish, IP Fabric

Source: Itential, The Ultimate MCP Guide for Network Automation, 2026. Examples shown; the guide lists all 78 servers.

itential

© Itential Confidential and Proprietary

Any MCP can feed the BUILD agent, the RUN-TIME agent, or both.

MCP SOURCE	BUILD TIME Pulled once by the coding agent	RUN TIME Called live by the game agent
Source of truth NetBox, Nautobot	Turn real sites and devices into level data	Load the player's own sites as a level
Devices and lab ContainerLab, CML, pyATS	Capture real show output for each puzzle	Live break-fix level; pyATS checks the fix
Assurance Batfish, IP Fabric	Validate every config before it is committed	Score the player's config against intent
Packets Wireshark, Packet Buddy	Generate real captures for decode levels	Explain the packet the player clicks on
Reference RFC Lookup, Subnet Calculator	Check every quiz answer against the source	Hints cite the RFC section by number
Itential Workflows, golden config	Pull real workflow and compliance examples	Final level runs a governed workflow

© Itential Confidential and Proprietary

Default to BUILD time, and go live only when the player needs live state.

CHOOSE BUILD TIME WHEN

- The data changes slowly: topologies, RFCs, configs
- Every player should see the same content
- The game must run offline, free and with no credentials

CHOOSE RUN TIME WHEN

- The player's own network is the level
- The right answer depends on live device state
- The agent has to react to what the player just did

RUN-TIME GUARDRAILS treat the game agent like a junior engineer with root access

Read-only by default
PagerDuty needs an explicit flag before it will write

Governed writes
Itential MCP exposes 65+ tools behind workflows and approvals

Git audit trail
Record every AI-generated change in Git

MCP gateway
One place for the allow-list, RBAC and logging

Source: Itential, The Ultimate MCP Guide for Network Automation, 2026.

© Itential Confidential and Proprietary

Skills package know-how that an agent loads only when a task needs it.

A SKILL IS A FOLDER

```
network-change/
├─ SKILL.md       name, description, steps
├─ scripts/       validate.py
├─ references/    golden-config.md
└─ assets/        templates
```

IT LOADS IN THREE STAGES

1. **At Startup**
   Only the name and description, about 100 tokens per skill
2. **When a Task Matches**
   The full SKILL.md instructions load
3. **Only if Needed**
   Scripts and reference files are read or run

Why it Matters Today

Spec Kit's commands and Itential Builder Skills both ship as skills. The format is an open standard (agentskills.io), so the same SKILL.md works in Claude Code, Codex, Copilot and Cursor.

itential

One foundation now hosts the open standards for tools, agents and skills.

MCP

How an agent reaches tools and data

Anthropic, Nov 2024
modelcontextprotocol.io

A2A

How agents find and delegate work to other agents

Google, Apr 2025
a2a-protocol.org

Agent Skills

How an agent learns a procedure

Anthropic, Dec 2025
agentskills.io

AGENTS.md

How a repo tells any agent its rules

OpenAI, Aug 2025
agents.md

Agentic AI Foundation (Linux Foundation)

Launched Dec 2025 with MCP, goose and AGENTS.md. Now hosts 10 projects, including A2A and Agent Skills (Oct 2026), under vendor-neutral governance. aaif.io

© Itential Confidential and Proprietary

A harness wraps the model in a loop with tools: reason, act, observe.



Claude Code
Anthropic's agentic coding harness, in the terminal and IDE

Codex
OpenAI's agentic coding harness, as a CLI, IDE extension and cloud agent

What both give the model
Read and edit files, run shell commands, call MCP servers

Today's models: Sonnet 5 or GPT-5.6

itential

03

RALPH WIGGUM LOOPS

Brute-force persistence with a fresh context every time.



- **“It’s better to fail predictably than succeed unpredictably.”**

— Geoffrey Huntley, creator of the Ralph loop

itential®

© Itential Confidential and Proprietary

“Ralph is a Bash loop.”

```
while :; do cat PROMPT.md | claude-code ; done
```

Geoffrey Huntley, July 2025

`while :; do … done`	Run forever, until a human or a stop rule ends it
`cat PROMPT.md`	Feed the same instructions every time; the plan on disk changes, the prompt doesn't
`| claude-code`	Start a fresh agent process with an empty context window

© Itential Confidential and Proprietary

Ralph works because the filesystem remembers what the model forgets.

01

Fresh context every loop

Each iteration starts a new agent process, so context rot never builds up

02

State lives on disk

Specs, the plan, the code and git history carry progress between iterations

03

One task per iteration

Small, finished steps instead of one long session that drifts

04

Backpressure from tests

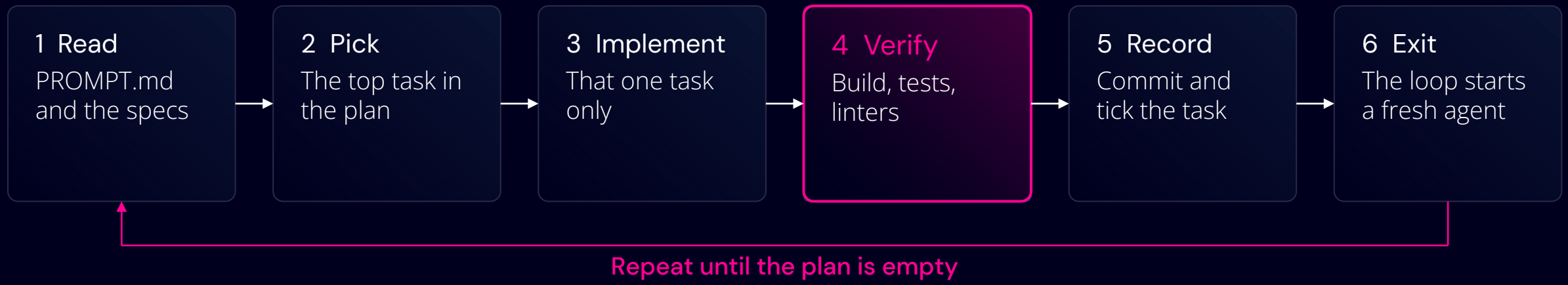
Builds, tests and linters reject bad work, and the failure output feeds the next loop

“LLMs are stateless. Filesystems aren't.”

itential

© Itential Confidential and Proprietary

Each Ralph iteration reads the plan, ships one task and exits.



ON DISK: THE MEMORY THAT SURVIVES EACH LOOP

`specs/*.md`	`IMPLEMENTATION_PLAN.md`	`src/ and tests/`	`git history`

Ralph runs in two modes: plan first, then build.

Plan Mode

- Compare the specs against the current code
- Write a prioritized IMPLEMENTATION_PLAN.md
- Change no code

Build Mode

- Take the top task from the plan
- Implement it and run the tests
- Commit, tick the task, exit

© Itential Confidential and Proprietary

Ralph needs guardrails because it never decides to stop on its own.

Iteration Cap
a hard maximum, such as 50 loops

Completion Promise
the agent prints DONE only when the tests pass

Sandbox
scoped permissions and no production credentials

Budget
Huntley measured about $10.42 an hour running Sonnet 4.5 in a loop

Review at the commit, not at every keystroke

© Itential Confidential and Proprietary

Within a year, the harnesses built Ralph in as a native command.

Jul 2025	Ralph is a hand-written Bash loop	`while :; do … done`
Mar 2026	Claude Code adds a recurring /loop command	`/loop`
Apr 2026	Codex adds goals with machine-checkable completion conditions	`goal objects`
May 2026	Claude Code and other harnesses add /goal	`/goal`
Jun 2026	**The practice gets a name: loop engineering**	`/loop + /goal + verifier`

Anthropic also packaged the technique as an official Claude Code plugin, ralph-loop.

© Itential Confidential and Proprietary

For networks, Ralph loops until the config passes validation.

```
# PROMPT.md
Study specs/bgp-peering.md.
Pick the highest-priority unchecked
item in IMPLEMENTATION_PLAN.md.
Generate the config for that item only.
Run: make validate
If it fails, read the errors and fix.
When it passes: commit, tick the item,
and exit.
```

WHAT MAKE VALIDATE CAN RUN

Syntax lint
Vendor parser rejects malformed config

Batfish
Reachability and policy checks on a model of the network

pyATS in a lab
Live state checks against virtual devices

Pre/post diff
Proves nothing changed that the spec didn't ask for

© Itential Confidential and Proprietary

04

SPEC-DRIVEN DEVELOPMENT

Write the intent down before anyone writes code.

A vague prompt makes the model guess, and every guess is a bug you find later.

THE WHOLE PROMPT

> “Make me a networking game.”

Six words. The model has to make every other decision itself, and you only see those decisions when you read the code.

WHAT THE MODEL HAS TO GUESS

- ? Which networking concepts to teach
- ? Who plays it and how hard it is
- ? Genre, controls and game loop
- ? Scoring and win conditions
- ? Tech stack and file layout
- ? What “done” looks like

© Itential Confidential and Proprietary

In SDD the spec is the source of truth and code is the output.

CODE-FIRST

Prompt, read the code, fix, prompt again

- Intent lives in chat history and is lost
- A change means rewriting code by hand
- Reviewers read diffs and guess the intent

SPEC-FIRST

Intent, then spec, plan, tasks and code

- Intent lives in versioned Markdown in the repo
- A change means editing the spec and regenerating
- Reviewers approve intent before code exists

itential

SDD splits intent into four files, each answering one question.

File	Question it answers	Example for a networking game
`constitution.md`	What rules never change?	Runs in a browser. No backend. Every level teaches one networking concept.
`spec.md`	What are we building, and why?	Players route packets across a topology to learn subnetting.
`plan.md`	How will we build it?	Three.js scene, plain HTML and CSS for the UI, levels stored as JSON.
`tasks.md`	In what order?	T001 scaffold the scene, T002 packet movement, T003 subnet puzzle…

You review each file before the agent generates the next one.

© Itential Confidential and Proprietary

A good spec describes behaviour and leaves out the tech stack.

SPEC.MD (EXCERPT)

User story 1: Subnet the packet home (P1)

As a new network engineer, I route a packet to the right host by choosing the correct subnet mask, so I learn CIDR by doing.

Acceptance criteria

Given a /24 and a host count, when I pick the right mask, then the packet reaches the host.

Given a wrong mask, then the packet drops and the game explains why.

Out of scope: multiplayer, user accounts

```
[NEEDS CLARIFICATION: how many levels ship in v1?]
```

User stories with priorities
Each one is a slice the agent can build and test alone

Testable acceptance criteria
Given, when, then: the loop's tests come from these

Explicit non-goals
Stop the agent building what nobody asked for

Unknowns marked, not guessed
Spec Kit tags them, and /clarify resolves them

itential

© Itential Confidential and Proprietary

A spec gives the loop its definition of done.

Spec
Says what done looks like

+

Loop
Keeps trying with fresh context

+

Verifier
Proves each step meets the spec

=

Convergence
Working code that matches intent

Without a spec

A Ralph loop keeps going until the budget runs out, because nothing tells it the work is finished.

With a spec

Each iteration checks acceptance criteria, and tests written from those criteria tell the loop when to stop.

itential

© Itential Confidential and Proprietary

05

SPEC KIT

GitHub's open-source toolkit for spec-driven development.

Spec Kit adds SDD commands to the coding agent you already use.

- The specify CLI scaffolds a project and installs the slash commands
- Works in Claude Code, Codex, Copilot, Cursor and 30+ other agents
- Every command is a Markdown prompt file you can read and edit
- Every command produces an artifact you review in the repo

© Itential Confidential and Proprietary

One command scaffolds the project and its folders.

```
uvx --from git+https://github.com/github/spec-kit.git specify init my-game
```

```
my-game/
├─ .specify/
│  ├─ memory/constitution.md
│  ├─ scripts/
│  └─ templates/
└─ specs/
   └─ 001-packet-game/
      ├─ spec.md
      ├─ plan.md
      └─ tasks.md
```

init asks which agent you use
Claude Code or Codex today; it installs the matching slash commands

memory/ holds the constitution
Every later command reads it and checks its output against it

Each feature gets its own folder
/specify creates the folder and a git branch for the feature

© Itential Confidential and Proprietary

Seven commands take an idea to working code.

STEP	COMMAND	WHAT IT DOES	PRODUCES
1	/constitution	Project rules, set once	`constitution.md`
2	/specify	What and why	`spec.md and a feature branch`
3	/clarify	Removes ambiguity	`Answers written into spec.md`
4	/plan	How: stack and design	`plan.md, research.md, data-model.md`
5	/tasks	Ordered work items	`tasks.md`
6	/analyze	Finds gaps and conflicts	`Read-only report`
7	/implement	Builds from tasks.md	`Source code`

Quality gates: optional in Spec Kit, required checkpoints in today's workshop

Spec Kit also ships /checklist, which writes quality checks for the requirements, and /converge, which verifies the code against the spec. Command names vary by agent: /speckit.specify, /speckit-specify or $speckit-specify.

Clarify and analyze catch mistakes before any code is written.

/clarify

- Asks targeted questions about underspecified areas
- Writes your answers back into spec.md
- Costs minutes now instead of hours mid-build

/analyze

- Read-only check across spec, plan and tasks
- Flags constitution violations as CRITICAL
- Fix the source file, not the output

itential

Today you run the full pipeline with two room-wide checkpoints.

Step	Command	What to do
1	/constitution	Point to the constitution your group personalized
2	/specify	Point to your group's personalized specification
3	/clarify	Room checkpoint: compare the questions everyone is asked
4	/plan	Ask for the plan in Markdown
5	/tasks	No prompt needed
6	/analyze	Room checkpoint: review the findings together
7	/implement	Build your game

itential

© Itential Confidential and Proprietary

SDD defines what done looks like; Ralph keeps going until the work gets there.

SDD WRITES THE FILES (YOU REVIEW THEM)

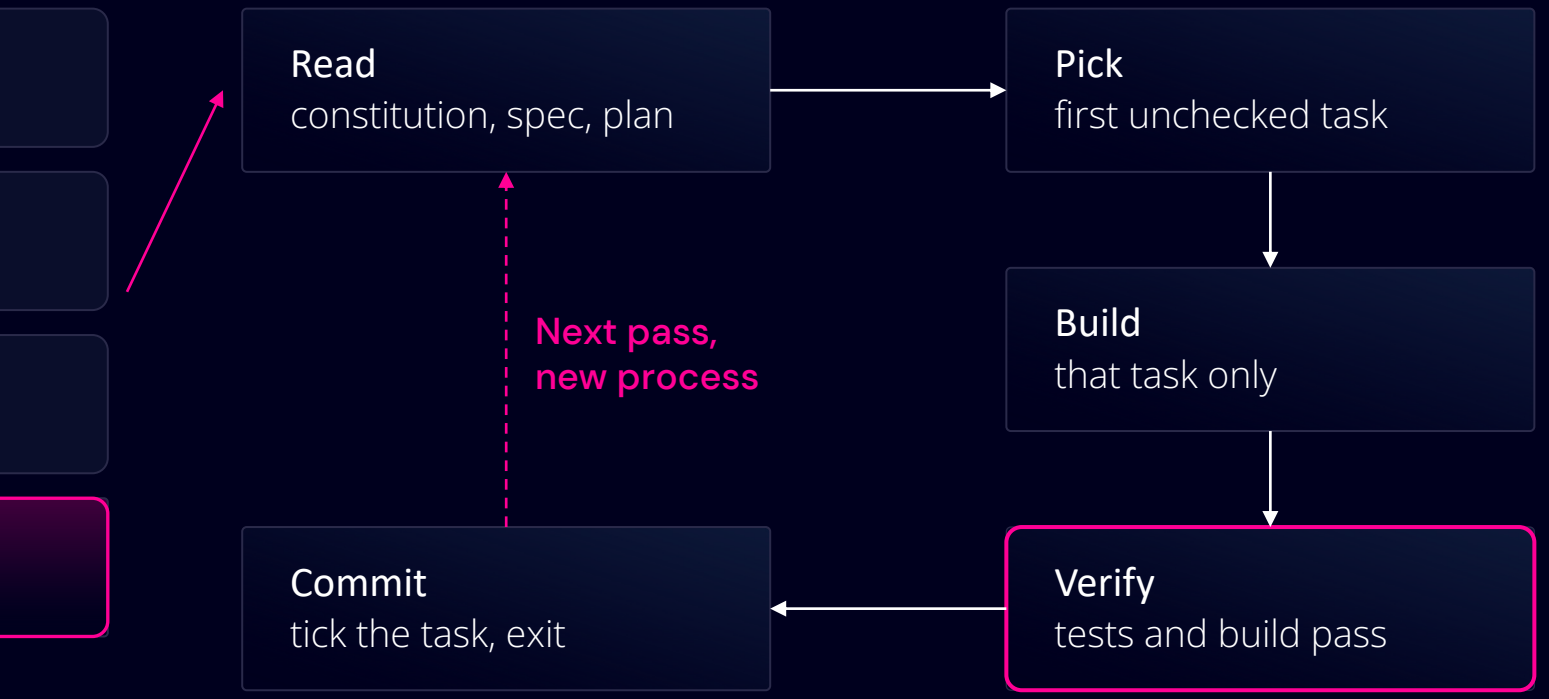
constitution.md
rules that never change

spec.md
acceptance criteria = done

plan.md
stack and architecture

tasks.md
ordered, checkable tasks

RALPH RUNS THE LOOP (FRESH CONTEXT EACH PASS)



Specs without a loop stop after one pass. A loop without specs never knows it has finished.

itential

Run /implement as a Ralph loop: one task from tasks.md per fresh context.

PROMPT.MD

```
Read .specify/memory/constitution.md and
specs/001-*/spec.md.
Open specs/001-*/tasks.md; take the first
unchecked task. Implement only that task,
then run the tests.
If they pass: tick the task, commit, exit.
If every task is ticked: print ALL_TASKS_DONE.
```

RALPH.SH

```
for i in $(seq 1 30); do
  cat PROMPT.md | claude -p | tee -a ralph.log
  grep -q ALL_TASKS_DONE ralph.log && break
done
```

Why it helps

Each task gets a clean context window, so task 40 is built as carefully as task 1.

Two stop rules

The completion word ends it when done; the 30-pass cap ends it if it never converges.

Other harnesses

Codex: codex exec

Claude Code: the ralph-loop plugin

itential

© Itential Confidential and Proprietary

06

ITENTIAL BUILDER SKILLS AND SDD

Spec-driven development for infrastructure automation.

Builder Skills give a coding agent the domain knowledge to build on Itential.

- Install from the Anthropic marketplace in one command
- Runs in Claude Code, Codex CLI, Copilot CLI, Copilot in VS Code and Cursor
- Builds workflows, command templates, golden config, Lifecycle Manager models, IAG services, FlowAgents and JSON forms

17 Total Skills

8 Delivery Skills

9 Platform Skills

Six stages, each producing an artifact the engineer must approve.

Stage	Agent	Artifact	Engineer gate
Requirements	spec-agent	customer-spec.md	Approves the spec
Feasibility	solution-arch-agent	feasibility.md	Approves the feasibility decision
Design	solution-arch-agent	solution-design.md	Approves the design
Build	builder-agent	Deployed assets	Reviews and accepts the delivery
Test	qa-agent	test-plan.md, test-report.md	Approves the test plan before anything runs live
As-Built	qa-agent	as-built.md	Signs off

Builder Skills follow the same SDD pattern as Spec Kit, tuned for infrastructure.

Spec Kit command	Builder Skills equivalent	Artifact
/constitution	Constitution	High-level guardrails
/specify + /clarify	spec-agent	customer-spec.md
/analyze	solution-arch-agent: feasibility	feasibility.md
/plan	solution-arch-agent: design	solution-design.md
/tasks + /implement	builder-agent	Deployed assets
No direct equivalent	qa-agent	test-plan.md, test-report.md, as-built.md

© Itential Confidential and Proprietary

The agent that builds never certifies its own work.

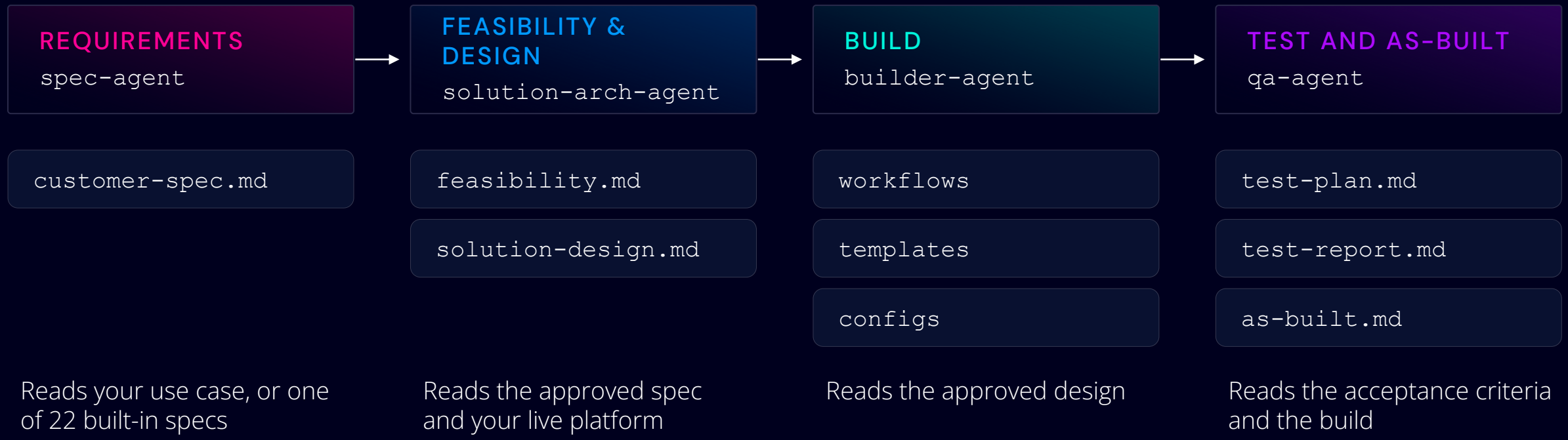
- In the original five-stage model, builder-agent built, tested and documented
- Version 2 hands Test and As-Built to a separate qa-agent
- qa-agent writes the test plan from the approved acceptance criteria
- The engineer approves the test plan before anything runs live

Explore with agents first, then ship a deterministic build.

- **flowagent-to-spec:** reads a FlowAgent's mission history and writes the spec for a deterministic equivalent
- **project-to-spec:** reverse-engineers an existing Itential project into a spec and design
- In a live session, John and Ankit Bansali built VPN tunnel, DNS A record and BGP peer workflows from specs

© Itential Confidential and Proprietary

Each stage hands the next one a document, not a chat history.



Between every pair of agents, an engineer approves the document before the next agent starts.

Watch three moves in the demo: write the spec, check feasibility, build.

01 Write the spec
`spec-agent`

- Turns a use case into customer-spec.md through a back-and-forth dialogue
- Can start from one of 22 built-in specs
- Engineer approves the spec

02 Check feasibility
`solution-arch-agent`

- Connects to your platform and finds the adapters and integrations you have
- Decides whether the spec can be built here, before anything is built
- Designs what to build or reuse, and in what order

03 Build
`builder-agent`

- Builds the approved design in dependency order
- Tests each component, then hands off to qa-agent
- Builds only what the design approved

Question to hold during the demo: what did the feasibility check catch that a single prompt would have missed?

itential

© Itential Confidential and Proprietary

LIVE DEMO

ITENTIAL BUILDER SKILLS, LIVE

Joksan Flores and Elliot Connor show how they use Builder Skills, and how the artifacts each stage generates become the supporting documents the agent follows.

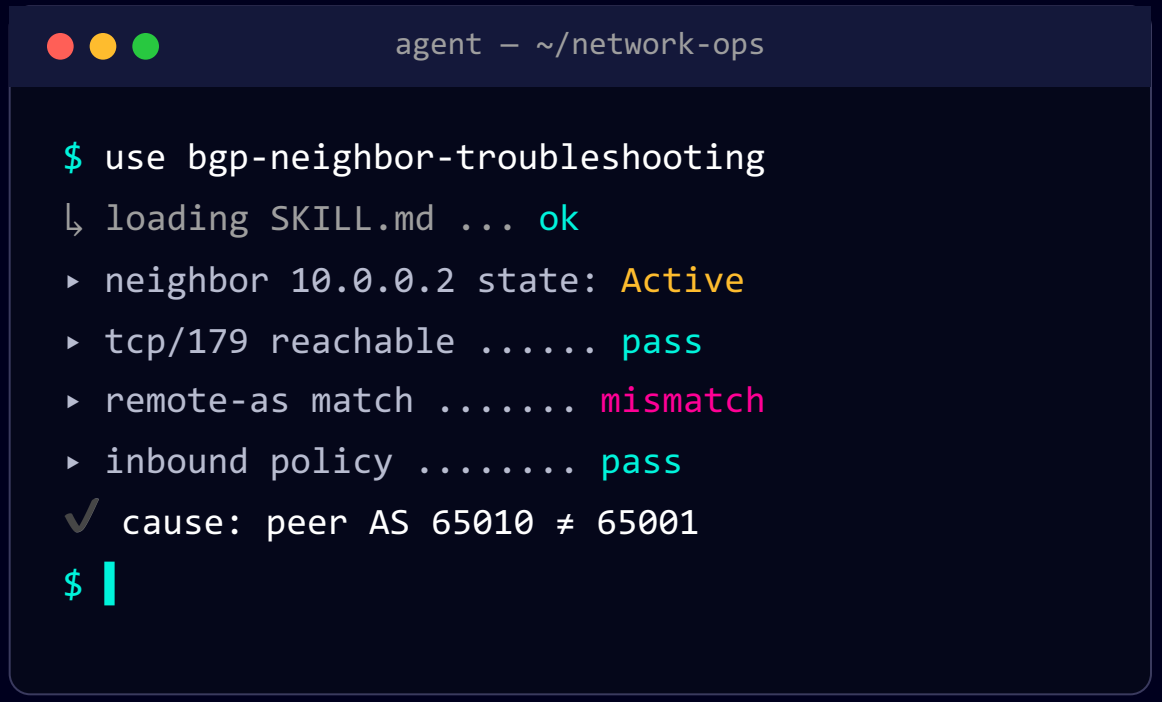
itential

Itential opened a public directory of 336 skills for infrastructure agents.

ANNOUNCING infraagentskills.ai

336	83	32
skills	vendors	contributors

- **Open GitHub repo:** github.com/itential/infra-agent-skills
- **Vendor and community submissions** through pull requests, with issues for fixes and requests
- **A front end to browse it all:** filter by category, vendor, contributor and source
- **Works in any harness** that reads SKILL.md: Claude Code, Codex, Copilot, Cursor, FlowAgents

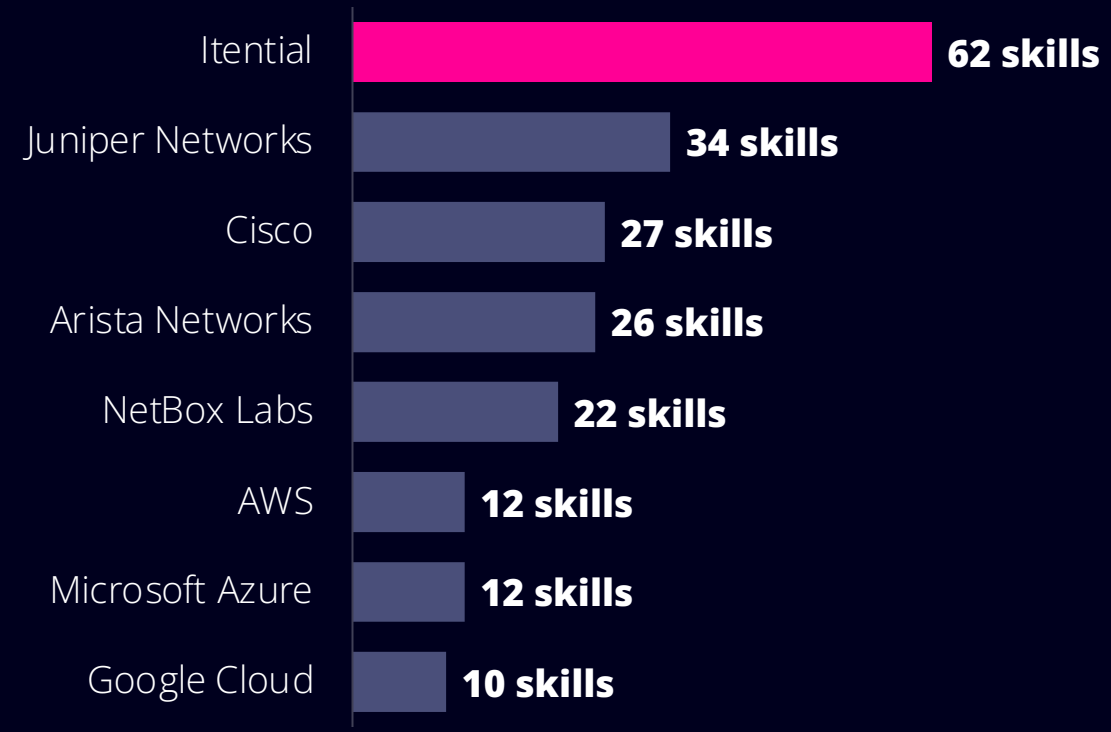


Recreated from the infraagentskills.ai home page, November 2026.

© Itential Confidential and Proprietary

Vendors and the community contribute through GitHub, and maintainers review every change.

Skills by contributor, top 8 of 32



A ADD TO THE REPOSITORY

1. Fork and branch from main
2. Write a short SKILL.md; vendor steps go in references/
3. Check every step against vendor docs
4. Open a PR and sign the CLA; a maintainer reviews it

B LIST YOUR OWN SKILL

Add one entry to CATALOG.yaml. The skill stays in your repo under your license, and the directory links to it.

Not ready to write one? Open an issue to report a problem or propose a new skill or vendor.

Each skill tells an agent what to check, in what order, and what it has proved.

oob-console-access/SKILL.md

```
---
name: oob-console-access
description: Reach a device through
  its console server when SSH is down
---
## Tools
- console server, device CLI
## Method
1. Find the console server and port
2. Confirm hostname and serial match
## Guardrails
- Read-only; writes need approval
## Output
- Say what the evidence can't prove
```

1 **Purpose**
The question it answers and when to use it

2 **Tools**
Which tools or MCP servers it may call

3 **Method**
The order of checks an expert would run

4 **Guardrails**
Read-only by default; writes need approval

5 **Output**
What the evidence does and does not establish

BUILD Copy into .claude/skills for Claude Code

RUN-TIME Load the same skills into Itential FlowAgents

itential

© Itential Confidential and Proprietary

07

LOOP ENGINEERING

Stop prompting the agent. Design the loop that prompts it.

“Design loops that prompt your agents.”

Peter Steinberger, June 2026

Boris Cherny
CREATOR OF CLAUDE CODE

Said his job is now writing the loops that drive Claude

Addy Osmani
NAMED THE PRACTICE DAYS LATER

“A recursive goal where you define a purpose and the AI iterates until complete”

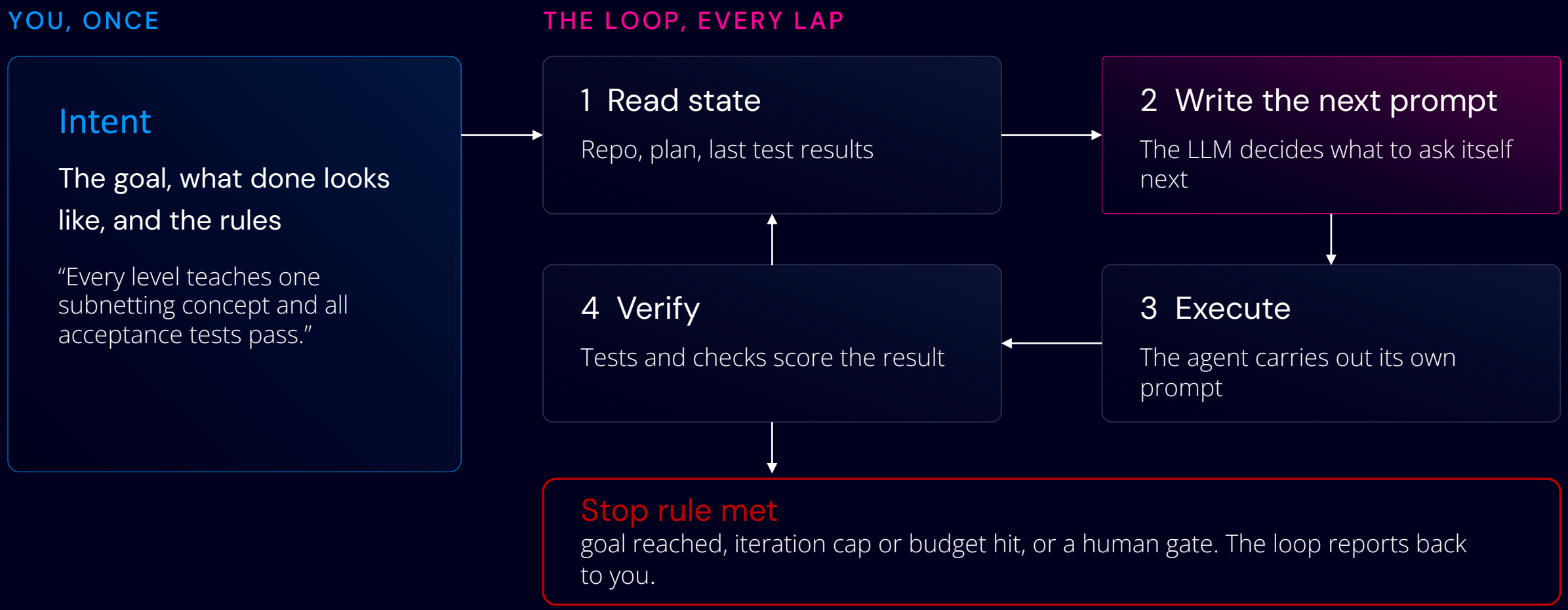
Andrew Ng
LATER THAT MONTH

Framed it as three loops: agentic coding, developer feedback and external feedback

The skill moved from writing good prompts to designing the system that writes them.

© Itential Confidential and Proprietary

You write the intent; the LLM writes its own next prompt.



Every loop has four parts: trigger, topology, verifier and stop rules.

Trigger

What starts a lap

A schedule, a commit, an alert or a ticket

IN A NETWORK LOOP

A config drift alert

Topology

How agents are arranged

One agent, a planner with workers, or a builder with a reviewer

IN A NETWORK LOOP

Builder agent plus a separate QA agent

Verifier

What proves progress

Tests, linters, a proof checker or a second model

IN A NETWORK LOOP

Batfish and pyATS checks

Stop rules

When it quits

Goal met, iteration cap, spend cap or a human gate

IN A NETWORK LOOP

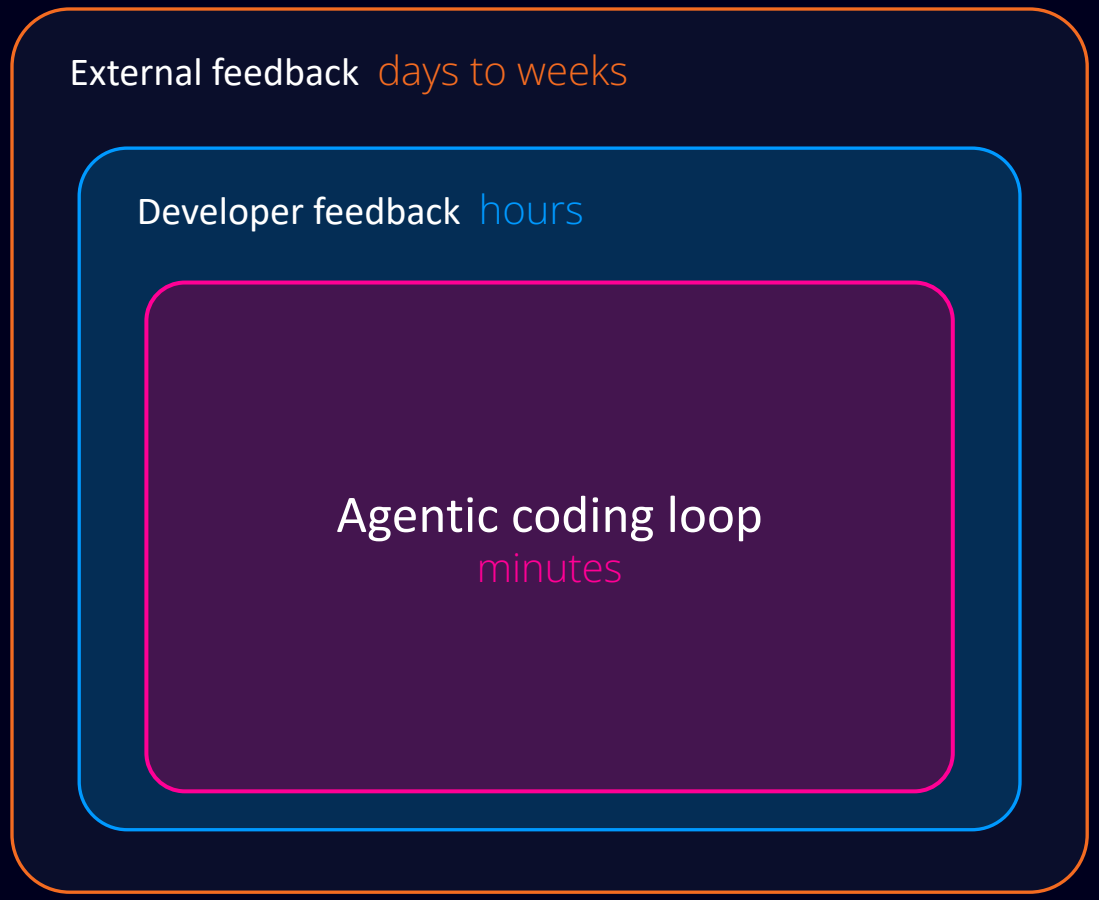
Engineer approval before deploy

Anatomy from Addy Osmani, June 2026

itential

© Itential Confidential and Proprietary

Andrew Ng describes three nested loops, each slower and more human.



Agentic coding loop

Given a spec and optional evals, the agent writes code, tests it and keeps iterating. Ng's agent worked about an hour on its own, checking its work in a browser.

Developer feedback loop

You review what was built and steer the next round. This is where today's /clarify and /analyze checkpoints sit.

External feedback loop

Users react, and their response sets product direction. The agent can't see this unless you feed it in.

© Itential Confidential and Proprietary

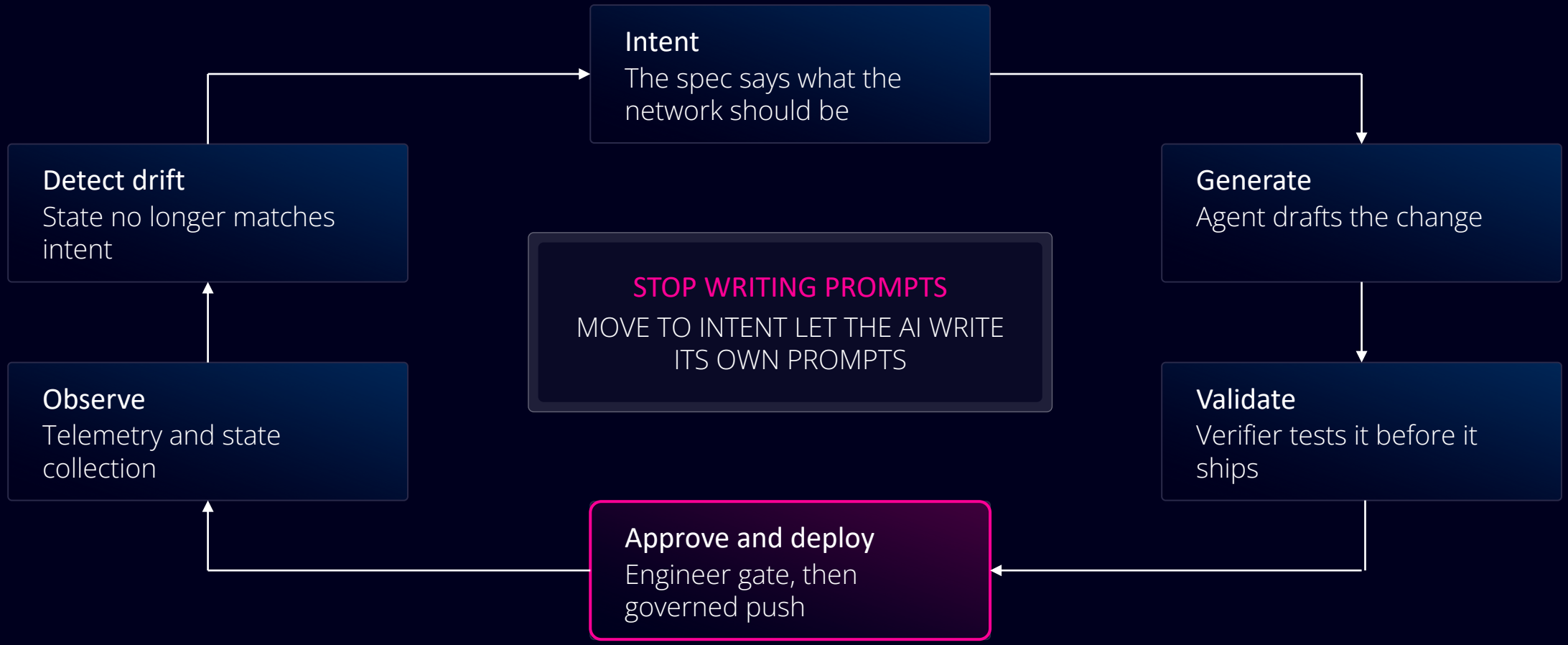
The verifier, not the model, decides how far a loop can go.

- **Ralph loops:** unit tests, builds and linters push back on bad code
- **DeepMind's AlphaProof Nexus:** ran Ralph-style loops with a formal proof checker and solved 9 open Erdős problems
- **Networks:** Batfish models, pyATS tests and pre/post state checks

Without a verifier, a loop only produces wrong output faster.

© Itential Confidential and Proprietary

In operations, loop engineering closes the gap between intent and network state.



itential

Loops multiply cost and risk as fast as they multiply output.

Cost

Send scanning and drafting to small models; cap iterations and spend

Runaway loops

Every loop needs a hard stop rule

Permissions

Least privilege, and no production credentials in build loops

Review gates

Humans approve specs, plans and deployments

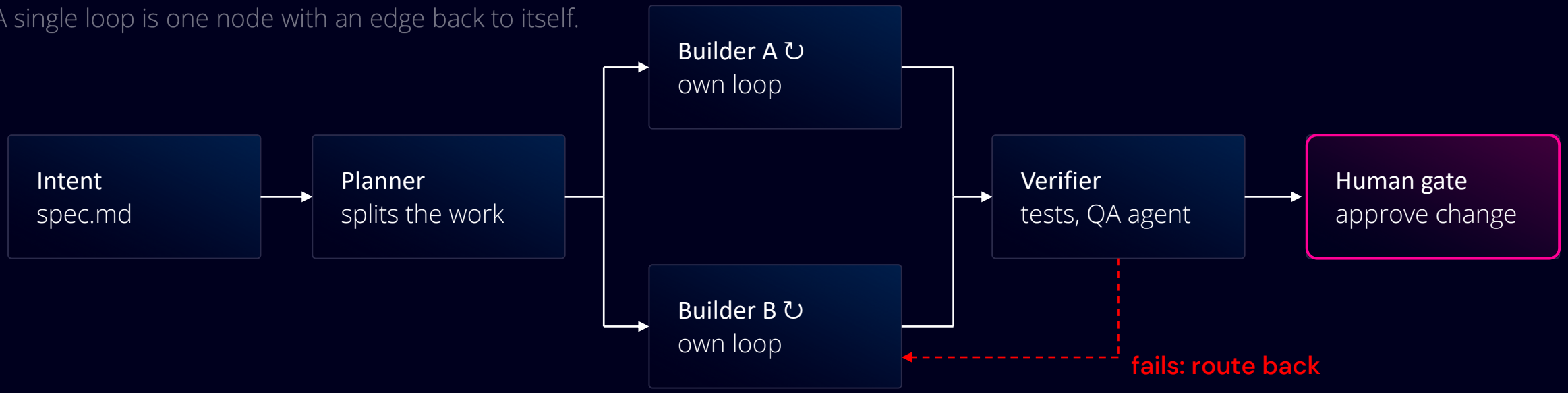
© Itential Confidential and Proprietary

Next comes graph engineering: many loops wired into one governed system.

Nodes do the work.

Edges route it, including branches and retries.

A single loop is one node with an edge back to itself.



Shared state on every edge: specs, artifacts, test results, approvals

The term surfaced in July 2026 and is still contested; the practice (graph-based orchestration) is older.

itential

A loop repeats one agent's cycle; a graph decides which agent works next.

	Loop engineering	Graph engineering
Unit of design	One agent's repeated cycle	A network of agents, tools, verifiers and people
Who picks the next step	The same agent, every pass	Routing rules on the edges between nodes
State	Files and git for one task	Shared, versioned state across all nodes
Verification	Tests at the end of each pass	Dedicated verifier nodes and approval gates
Network example	Loop a config until Batfish passes	Spec, design, parallel builds, QA, change approval

For network teams: Itential workflows already model work as a graph of tasks with approval gates. Graph engineering puts agents on those nodes.

itential

© Itential Confidential and Proprietary

Stacked together, these pieces are VibeOps.

- **Loop Engineering** Loops that write their own next prompt
- **SDD & Spec Kit** Written intent and review gates
- **Ralph Loop** Persistence with a fresh context each lap
- **Harness** Claude Code or Codex: the agent loop with tools
- **RAG & MCP** Your data and your systems
- **LLM** Predicts the next token

VibeOps

AI speed with engineering discipline

Each layer fixes a limit of the one below it.

08

GAMIFICATION OF NETWORKING

10:00 – 10:15 lecture, then the group activity

Next: set up your project (10 minutes)

1. Open VS Code
2. Install your harness: Claude Code or Codex
3. Choose your model: GPT 6 Astra; Opus 5-5; Fable 5.2 best results most costly context windows
4. Install the Ralph loop plugin (in Claude Code): /plugin install ralph-loop@claude-plugins-official
5. Install Spec Kit: uv tool install specify-cli --from git+https://github.com/github/spec-kit.git
6. Create your GitHub repository – Using https://github.com/automateyournetwork/NAF_AC6 as a starting point

Room-wide check: raise a hand if anything fails

© Itential Confidential and Proprietary

Then: write your constitution and spec

7 **Clone your repo** `git clone <your-repo-url>` then `cd` into it

8 **Add Spec Kit** `specify init . --integration claude` (or codex, copilot)

9 **Copy John's starter files** CONSTITUTION.md, SPEC.md and PLAN.md into the project root

10 **Fill them in as a team** replace the [BRACKETED] placeholders with your topic, rules and stack

11 `/speckit.constitution` "Please refer to the CONSTITUTION.md file and build the constitution"

12 `/speckit.specify` "Please refer to the SPEC.md file and build the specification"

Room-wide check: everyone should have a constitution and a spec before we clarify together

© Itential Confidential and Proprietary

Then: clarify, plan, analyze and build

13	**GROUP BREAK** `/speckit.clarify`	answer its questions, then we catch up and share ours
14	`/speckit.plan`	"Please refer to the PLAN.md file and build the plan"
15	`/speckit.tasks`	break the plan into small, ordered tasks
16	**GROUP BREAK** `/speckit.analyze`	check spec, plan and tasks agree, then we compare findings
17	`/speckit.implement`	build it task by task, as a Ralph loop
18	**Play it**	open index.html in a browser and test against your spec

At each break: stop typing, and bring one question or finding to share with the room

© Itential Confidential and Proprietary

John's starter kit is three Markdown templates you fill in.

`CONSTITUTION.md`

The rules that must never break

Feeds `/speckit.constitution`

→

`SPEC.md`

What you're building, for whom and why

Feeds `/speckit.specify`

→

`PLAN.md`

How: tech stack, architecture and data schema

Feeds `/speckit.plan`

Each file arrives pre-loaded with the defaults that made the 3 instructor example games work, plus [BRACKETED] placeholders for your topic and idea.

https://github.com/automateyournetwork/NAF_AC6

© Itential Confidential and Proprietary

The constitution sets the rules your game can never break.

Accuracy
the game must never present incorrect information, even to be more fun

Feedback
every wrong answer explains why

Difficulty
driven by data, not hardcoded

Your edits
replace the [BRACKETED] placeholders and add your own non-negotiables

© Itential Confidential and Proprietary

The specification says what you're building, for whom and why.

Topic
the networking concept you know cold

Learning outcomes
what a player can do after playing

Gameplay loop
the action the player repeats, and how it teaches

No idea yet
use the Inspiration section at the bottom of SPEC.md

© Itential Confidential and Proprietary

The plan says how: tech stack, architecture and data schema.

Suggested stack
runs in a browser with Three.js, HTML and optional CSS

Data schema
questions, levels and difficulty live in data files, not code

Done means
open index.html in a browser and play

Want more?
Add tools to your build harness to build the game better, and to the game itself to make it better to play (next two slides)

© Itential Confidential and Proprietary

Augment your BUILD harness so it builds a better game.

BUILD Used while you build. Nothing here ships inside the game or costs players anything.

MCP servers

Blender or Unreal Engine MCP to make 3D models and scenes. An RFC or CVE MCP so questions match the source. GitHub MCP. Your favourite MCP.

RAG sources

Point the build agent at vendor docs, RFC PDFs or HTML pages so it writes accurate questions and explanations.

AI-generated assets

Images: OpenAI Image 2.5, Nano Banana. Voice: ElevenLabs. Video: Veo 3. Generate once, commit the files.

Skills and loops

A Three.js or front-end skill for better code. The Ralph loop to work through tasks.md unattended.

© Itential Confidential and Proprietary

Augment your RUN-TIME game so it plays better.

RUN-TIME Used while someone plays. Each addition makes the game an agent, with a cost and a risk.

A model in the game

A tutor or opponent that explains answers. Free and local with Ollama, or through an API key the player supplies.

RAG at play time

The game looks up the RFC or vendor page behind each answer and shows the player the source.

MCP at play time

The game calls an RFC or CVE MCP to pull fresh scenarios, so no two rounds are the same.

Live generated media

ElevenLabs narrates hints, or an image model draws each new level. Costs per play, so cap it.

Whatever you add, write it into your spec and keep the constitution rule: the game never presents incorrect information.

Sound, art and scores make a game feel finished.

BUILD used while you build RUN-TIME used while someone plays BOTH either one

BOTH

Text to speech

Free: gTTS, generated once and committed as MP3s. Paid: ElevenLabs for richer voices, or live narration at play time.

BUILD

Soundtracks and audio

Ask your agent to install local audio tools and make MP3 tracks. Tested on Mac, not yet on Windows or Linux.

BUILD

Images

AI image tools such as ChatGPT image generation or Google Gemini. Generate once and commit the files.

BUILD

Video cutscenes

Intro, cutscene and ending clips from Google Veo 3.1, Kling or Runway. Paid per second ($$$) for ~7-second clips; compress MP4s for cloud hosting.

BUILD

Abandonware and homage

Start from the code of a legacy game, or build a homage to one people already know. Check the license first.

RUN-TIME

Leaderboards and statistics

High scores and per-player stats. Browser storage covers one player; a shared board needs a small backend.

RUN-TIME

Multiplayer

Players race or compete on the same network puzzle. A server has to sync state, so scope it in the spec.

Most of these are BUILD-time: generate once, commit the files, and the game stays free to play.

itential®

© Itential Confidential and Proprietary

Borrow from the lab to make both agents better.

BUILD used while you build RUN-TIME used while someone plays BOTH either one

BUILD

Lab as source of truth

A containerlab or CML MCP lets the agent capture real show output instead of inventing it.

BUILD

Batfish on every config

Any config the game shows is parsed and validated before the loop commits it.

BUILD

A QA agent that plays

Playwright MCP plays every level and logs errors, so the builder never grades its own work.

BUILD

Real packets via Scapy

Generate real PCAPs at build time; the game decodes the actual headers, Wireshark style.

RUN-TIME

Break-fix final level

The player fixes a broken router in a live containerlab; pyATS checks the state and scores it.

RUN-TIME

Routers as NPC agents

Each router is a small agent with its own table. Poison a route and watch the others react.

BOTH

Adaptive difficulty

Tag every question by concept; the game serves more of what the player misses. No model needed.

BOTH

A NetBox-shaped world

Build levels from NetBox demo data, then let players load their own sites and devices.

Bonus level: route the final change through an Itential workflow with an approval gate, the way production does.

itential

© Itential Confidential and Proprietary

Classic game genres already have a mechanic for most networking concepts.

DDoS mitigation

× **Tower defense:** place firewalls and rate limiters before each wave of traffic lands.

Spanning tree

× **Snake:** the snake is a broadcast storm; block the right port before it eats itself.

VLANs and trunks

× **Pipe Dream:** lay access and trunk links before the traffic starts to flow.

OSI troubleshooting

× **Roguelike:** each dungeon floor is a layer; clear Layer 1 to unlock Layer 2.

Stateful firewalls

× **Papers, Please:** inspect each packet at the border and remember the sessions you let in.

TCP handshake

× **Rhythm game:** hit SYN, SYN-ACK and ACK on the beat; miss a keepalive and the session drops.

Network design

× **City builder:** build a campus on a budget; outages strike wherever you skipped redundancy.

ARP spoofing

× **Social deduction:** one player is the rogue device; the rest catch them from the ARP tables.

More pairings: Wordle × subnet masks, Typing of the Dead × CLI, Tetris × MTU, Frogger × collisions.

© Itential Confidential and Proprietary

Pair a networking concept you know with a game mechanic people already love.

BUILT LIVE: PACKET QUEST

Routing

× Play as the packet

Subnetting

× Daily Duolingo-style lessons

Load balancing

× Duolingo for F5

QoS queuing

× A racing game

ACLs

× Arkanoid: block or permit what bounces in

BGP

× A path-hijack thriller

Ideas from the Inspiration section of SPEC.md. This list is a starting point, not the ceiling.

© Itential Confidential and Proprietary

09

GROUP ACTIVITY

45 minutes · join the group that matches the number you were given.

The group shapes the ideas; you build solo or as a team.

10:15 | 10:30 | 10:45 | 11:15 | 11:30

Block 1 · 15 min
Introductions and game ideas

Block 2 · 15 min
Read aloud and personalize the templates

Break · 30 min

Block 3 · 15 min
Finish personalizing the templates

Ideate as a group

Everyone shares a networking-themed game idea; the group helps refine each one

Build solo, or as a team

Each person, pair or team leaves with its own constitution, spec and plan (next slide)

Draw from everywhere

Networking, network automation, video games and the templates

Templates: about 10 minutes per file. One person reads aloud; the group reviews and adjusts.

© Itential Confidential and Proprietary

Ideas overlap? Pair up or form a small team, with one person driving.

Recommended: one driver

1. **The driver creates the repo**
   on their own GitHub account (previous steps)
2. **Add everyone as collaborators**
   Repo Settings, then Collaborators, then Add people
3. **One person types, everyone steers**
   Answer /clarify together and review /analyze as a group

Advanced: split the work another layer to manage

Agree on one shared constitution first, then each person owns one feature:

Alex	`/specify levels → 001-levels branch`
Sam	`/specify scoring → 002-scoring branch`
Priya	`/specify visuals → 003-visuals branch`

Merge each branch with a pull request. Separate specs keep each agent inside its own files, so you don't overwrite each other.

Short on time? Stay with one driver. Splitting the work pays off on a longer build, not in a 60-minute block.

© Itential Confidential and Proprietary

Bring your idea to life: run these seven commands in order.

Step	Command	What to type or do
1	`/speckit.constitution`	Please reference the StarterKit/CONSTITUTION.md file and start the SDD process for my network-themed educational game.
2	`/speckit.specify`	Please reference the StarterKit/SPEC.md file for your specifications.
3	`/speckit.clarify`	Answer the questions. Room checkpoint: compare what everyone is being asked.
4	`/speckit.plan`	Please reference the StarterKit/PLAN.md file and start planning. Use Markdown.
5	`/speckit.tasks`	No prompt needed.
6	`/speckit.analyze`	Room checkpoint: review findings, then ask the agent to fix everything found in your analysis.
7	`/speckit.implement`	Build the game, then open index.html in a browser.

© Itential Confidential and Proprietary

Show your games.

Each group picks two to share with the room.

12:30

Return to your group and play each other's games

Pick two

Each group chooses two games to show the room

Push

Commit your work to your GitHub repository

Share

Open an issue on the starter kit repo or tag @automateyournetwork

© Itential Confidential and Proprietary

BRING YOUR GAME TO THE INTERNET

Inside the repo there is an ARCADE.md template

Have your agent complete the template

When you are happy and totally optionally send me your GitHub URL

I will add the game to the automateyournetwork.com/arcade

© Itential Confidential and Proprietary

Links to everything you need today.

- Starter kit: github.com/automateyournetwork/NetworkingVideoGameStarterKit
- Spec Kit: github.com/github/spec-kit
- uv (installs the specify CLI): docs.astral.sh/uv
- Claude Code: claude.com/claude-code
- Codex: openai.com/codex
- Ralph loop plugin for Claude Code: `/plugin install ralph-loop@claude-plugins-official`
- VS Code: code.visualstudio.com
- Itential Builder Skills: github.com/itential/builder-skills
- Guide to spec-driven development: itential.com/resource/guide/spec-driven-development
- AONE episode, Stop Vibe Coding: youtube.com/watch?v=9iWDteEhYk0

© Itential Confidential and Proprietary

Standards and guides for going further.

- Agentic AI Foundation (MCP, A2A, Agent Skills, AGENTS.md): aaif.io
- Model Context Protocol: modelcontextprotocol.io
- Agent2Agent (A2A) Protocol: a2a-protocol.org
- Agent Skills open standard: agentskills.io
- AGENTS.md: agents.md
- Ralph, by Geoffrey Huntley (the original post): ghuntley.com/ralph
- Ralph explained, plan and build modes: ralph-wiggum.ai
- Ralph loop plugin for Claude Code: github.com/anthropics/claude-plugins-official
- Loop Engineering, by Addy Osmani: addyosmani.com/blog/loop-engineering
- Graph Engineering in the Era of LLM Agents (survey): arxiv.org/abs/2608.21156

© Itential Confidential and Proprietary

Sources

- Geoffrey Huntley, “Ralph Wiggum as a ‘software engineer’,” July 2025: ghuntley.com/ralph
- LinearB Dev Interrupted, “Inventing the Ralph Wiggum Loop,” January 2026: linearb.io
- Anthropic, ralph-loop plugin for Claude Code: github.com/anthropics/claude-plugins-official
- GitHub, Spec Kit repository and documentation: github.com/github/spec-kit
- Anthropic, Model Context Protocol, November 2024: modelcontextprotocol.io
- Itential, “Bring Your Own Coding Agent: Itential Builder Skills,” October 2026: itential.com
- Itential, “Getting Started with Itential Builder Skills in Claude Code” (demo): itential.com
- “Loop Engineering: Building Blocks, Adoption, and Impact,” 2026: arXiv:2608.21884
- ADTmag, “Loop Engineering Emerges as Developers Put AI Coding Agents on Repeat,” July 2026: adtmag.com
- Google DeepMind, AlphaProof Nexus preprint, 2026: arXiv:2605.22763
- John Capobianco, Networking Video Game Starter Kit: github.com/automateyournetwork/NetworkingVideoGameStarterKit
- The Art of Network Engineering, “Stop Vibe Coding: Start Spec-Driven Development,” August 2026: youtube.com/watch?v=9iWDteEhYk0

© Itential Confidential and Proprietary


Join the
VIBEOPS
FORUM

FPO

Register Now

Live Virtual Workshop

Build & Orchestrate Infrastructure AI Agents

Dec. 18, 2026

12–3 p.m. ET

Virtual

Free to Attend



John Capobianco
Head of AI & DevRel



Joksan Flores
Principal Solutions Engineer



Elliot Conner
Solutions Engineer

itential

THANK YOU

