@VIBE_CODE • FREE GUIDE • APR 2026

7 AI AGENT TYPES IN CLAUDE MANAGE

Launched April 2026 — Anthropic's hosted agent platform. Build all 7 types. No infrastructure required.

7	£0	90s
Agent Types	Infra Cost	First Build

SETUP (ONE TIME — WORKS FOR ALL 7 TYPES)

```
1. Get a free API key at platform.claude.com/settings/keys
2. Install the SDK:  pip install anthropic
3. Set your key:     export ANTHROPIC_API_KEY='your-key-here'
4. All requests need the beta header: managed-agents-2026-04-01
   (The SDK sets this automatically — no extra work needed)
```

AGENT TYPE 01 • BEGINNER FRIENDLY

01

BASIC AGENT WITH TOOLS

Create an agent with the built-in toolset, then start a session and give it a task. Anthropic runs the whole thing in a secure cloud container — no server setup needed.

THINK OF IT AS

"A personal assistant with access to the web, a terminal and your files, running 24/7 in the cloud."

USE CASES

→ Run scripts and shell commands autonomously in a managed container
→ Search the web, fetch pages, read and write files without you watching
→ Any task that needs a terminal, a browser and a file system

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# 1. Create the agent (do this once, save the returned agent.id)
ant beta:agents create \
  --name 'My Assistant' \
  --model '{id: claude-opus-4-7}' \
  --system 'You are a helpful assistant.' \
  --tool '{type: agent_toolset_20260401}'

# 2. Create an environment (a managed cloud container)
ant beta:environments create \
  --name 'my-env' \
  --config '{type: cloud, networking: {type: unrestricted}}'

# 3. Start a session and send your task
session = client.beta.sessions.create(
    agent=agent.id, environment_id=environment.id)
client.beta.sessions.events.send(session.id,
    events=[{type: 'user.message', content: 'Do the task'}])
```

AGENT TYPE 02 • THIS IS WHERE IT GETS POWERFUL

02

MCP SERVER AGENT

Add MCP server URLs when defining your agent. The agent gains read/write access to those external services and can interact with everything stored there, automatically.

THINK OF IT AS

"Giving your agent a keycard to every tool in your business, not just its own container."

USE CASES

→ Connect to Notion, update pages and databases automatically
→ Read Jira boards, create tickets and update statuses hands-free
→ Plug into any service that has a public MCP server URL

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# When creating your agent, add mcp_servers to the config
ant beta:agents create \
  --name 'Notion Agent' \
  --model '{id: claude-opus-4-7}' \
  --system 'You can read and write Notion pages.' \
  --tool '{type: agent_toolset_20260401}' \
  --mcp-server '{type: url, url: https://mcp.notion.com/sse}'

# Then start a session exactly as in Type 01.
# The agent now has live access to your Notion workspace.
# Add multiple --mcp-server flags for multiple services.
```

AGENT TYPE 03 • AI ASSEMBLY LINE

03

SEQUENTIAL AGENTS

Create separate agents for each stage of your pipeline. Start each session in order, passing the output of the previous session as the input to the next. Each agent is a specialist.

THINK OF IT AS

"A factory production line — each station does one job perfectly before handing off to the next."

USE CASES

- → Research -> Summarise -> Write -> Publish, each step a different agent
- → Prospect -> Qualify -> Draft outreach -> Send, fully automated
- → Ingest raw data -> Clean it -> Analyse -> Generate report

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# Create three specialist agents (each with its own system prompt)
agent_a = create_agent('You read and summarise contacts.')
agent_b = create_agent('You write personalised emails.')
agent_c = create_agent('You send emails via the mail API.')

# Run them in sequence — output of each becomes input of next
session_a = start_session(agent_a)
result_a  = send_and_wait(session_a, original_task)

session_b = start_session(agent_b)
result_b  = send_and_wait(session_b, result_a)

session_c = start_session(agent_c)
result_c  = send_and_wait(session_c, result_b)
```

AGENT TYPE 04 • SPEED MULTIPLIER

04

PARALLEL EXECUTION AGENTS

Start multiple sessions at the same time — each runs independently in Anthropic's cloud. Use Python asyncio to fire them all simultaneously and gather results when they're done.

THINK OF IT AS

"Three researchers working at once instead of waiting for one to finish before the next starts."

USE CASES

→ Research three competitors simultaneously and merge the findings
→ Write five email variants at once and pick the best one
→ Analyse separate datasets in parallel and combine the insights

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
import asyncio

# Create one agent, start multiple sessions at the same time
async def run_session(task):
    session = client.beta.sessions.create(
        agent=agent.id, environment_id=env.id)
    return await send_and_wait_async(session, task)

# Fire all sessions simultaneously
tasks = [
    run_session('Research competitor A'),
    run_session('Research competitor B'),
    run_session('Research competitor C'),
]
results = await asyncio.gather(*tasks)
merged  = combine(results)  # merge however you need
```

AGENT TYPE 05 • AI TRAFFIC CONTROLLER

05

AGENTS WITH ROUTERS

Create a lightweight router agent whose only job is to classify an incoming task and output which workflow should handle it. Your code then starts the right specialist session.

THINK OF IT AS

"An air traffic controller who instantly decides which runway every inbound flight goes to."

USE CASES

- → Urgent tasks go to priority queue, normal tasks to standard flow
- → Support tickets classified by complexity and routed to the right agent
- → Incoming leads sorted and sent to different sales workflows automatically

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# Router agent: classify input, return a routing decision
router = create_agent(
  system='Read the task. Reply with ONLY one word: '
         'PRIORITY, STANDARD, or ESCALATE.')

# Specialist agents for each route
priority_agent  = create_agent('Handle urgent tasks fast.')
standard_agent  = create_agent('Handle normal tasks carefully.')
escalate_agent  = create_agent('Handle complex edge cases.')

# Route the task
route  = send_and_wait(router_session, incoming_task)
agents = {'PRIORITY': priority_agent,
          'STANDARD': standard_agent,
          'ESCALATE': escalate_agent}
result = send_and_wait(start_session(agents[route]), incoming_task)
```

AGENT TYPE 06 • PERFECT FOR HIGH-STAKES TASKS

06

HUMAN IN THE LOOP

Set your tools to always_ask confirmation mode. Before the agent executes any action, you receive a confirmation event. You approve or deny — then it fires (or doesn't).

THINK OF IT AS

"A surgeon's assistant who preps everything perfectly, then waits for the surgeon to say go."

USE CASES

→ Sending large payments or invoices after you review the draft
→ Mass email sends where a human must sign off before firing
→ Deleting, archiving or modifying critical data with your approval

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# Create agent with always_ask tool confirmation
agent = create_agent(
  tools=[{type: 'agent_toolset_20260401',
          confirmation: 'always_ask'}])

# Stream events — intercept confirmation requests
for event in stream:
    if event.type == 'agent.tool_use':
        # Agent wants to run a tool — you decide
        print(f'Agent wants to run: {event.name}')
        approval = input('Allow? (y/n): ')
        result   = 'allow' if approval == 'y' else 'deny'
        client.beta.sessions.events.send(session.id, events=[
            {type: 'user.tool_confirmation',
             tool_use_id: event.id, result: result}])
```

AGENT TYPE 07 • THE MOST POWERFUL — RESEARCH PREVIEW

07

ORCHESTRATOR AGENT

Create specialist sub-agents, then create an orchestrator that lists them as callable_agents. The orchestrator dynamically spawns threads to delegate work — it decides who it needs at runtime.

THINK OF IT AS

"A CEO who instantly knows to hire a lawyer, accountant and designer for a problem and does it on the spot."

USE CASES

- → Complex tasks where the required sub-tasks are unknown until mid-execution
- → Business analysis requiring a research agent, legal agent and comms agent
- → Anything too complex to pre-plan where the system self-organises

HOW TO BUILD IT — CLAUDE MANAGED AGENTS

```
# Note: Multiagent is in Research Preview.
# Request access at claude.com/form/claude-managed-agents

# 1. Create your specialist sub-agents
researcher = create_agent('You research topics deeply.')
writer     = create_agent('You write clear summaries.')

# 2. Create the orchestrator, listing sub-agents it can call
orchestrator = client.beta.agents.create(
  name='Orchestrator',
  model='claude-opus-4-7',
  system='Delegate research to the researcher.',
  callable_agents=[
    {type: 'agent', id: researcher.id},
    {type: 'agent', id: writer.id}])

# 3. Start a single session — it spawns threads automatically
session = start_session(orchestrator)
```

MASTER ALL 7. BUILD REAL THINGS.

#	Type	Best For	Level
01	**Basic Agent With Tools**	Built-in tools: bash, web, files	Easy
02	**MCP Server Agent**	External databases via MCP URLs	Easy
03	**Sequential Agents**	Multi-step ordered pipelines	Medium
04	**Parallel Execution**	Same task, multiple sessions at once	Medium
05	**Router Agent**	Classify then route to right workflow	Medium
06	**Human in the Loop**	always_ask tool confirmation	Medium
07	**Orchestrator Agent**	callable_agents + dynamic threads	Research Preview

YOUR NEXT STEPS

1. Get your API key at platform.claude.com/settings/keys — it's free to start
2. Build Type 01 today. Takes 10 minutes. Proves the concept.
3. Add an MCP server (Type 02). Notion is the easiest first connection.
4. Follow @vibe_code for daily builds, teardowns and new agent patterns.