

Mohamed Maktal · Solutions Engineer, Vultr



Agentic AI Foundation

AGNTCon + MCPCon
North America

Where the **agentic stack** is being built.

What Happens When You Integrate MCP and A2A at the Infrastructure Layer



Hi, I'm Mohamed Maktal



Solutions Engineer @ Vultr

I work with teams that run agents on cloud infrastructure, including sandbox providers.

WHAT I AM SHARING

What your agent is told when the sandbox under it fails, and how to make it the truth.

What does MCP tell your agent?

YOUR AGENT
via Playwright MCP

● **working...**



microVM sandbox

● ● ● shop.example / checkout

Signed in as demo-user

Name

Address

Card

Confirm

What does MCP tell your agent?

YOUR AGENT
via Playwright MCP

● **working...**



SANDBOX GONE

● ● ● shop.example / checkout

GONE

the browser, the pages, the sign-in

Nothing. About a minute later: “request timed out”.

YOUR AGENT
via Playwright MCP

- “request timed out”



SANDBOX GONE

● ● ● shop.example / checkout

GONE

the browser, the pages, the sign-in

At fleet scale, “rare” becomes
thousands a day

1 in 200 runs lost to the sandbox itself

3,000,000 sandboxes a day

15,000 FAIL

e.g. one fleet's audited week, 2026: 626 lost of 127,735 runs (1 in 204)

Where the **agentic stack** is being built.



At fleet scale, “rare” becomes thousands a day



Agentic AI Foundation

AGNTCon + MCPCon
North America

1 in 200

runs lost to the sandbox itself

Three ways your stack hides the truth, and an infrastructure solution for each

On a real Firecracker sandbox: 55 s of guessing → the true cause in 1 s

e.g. one fleet's audited week, 2026: 626 lost of 127,735 runs (1 in 204)

Where the **agentic stack** is being built.

What we'll cover

- 1 How a tool call reaches a sandbox
- 2 Three failure modes, in a real sandbox
- 3 The pattern, and A2A
- 4 One test to run this week

The agent harness

how it works **1** 2 3 4

To see why the agent was told nothing, let's follow one tool call through the stack.

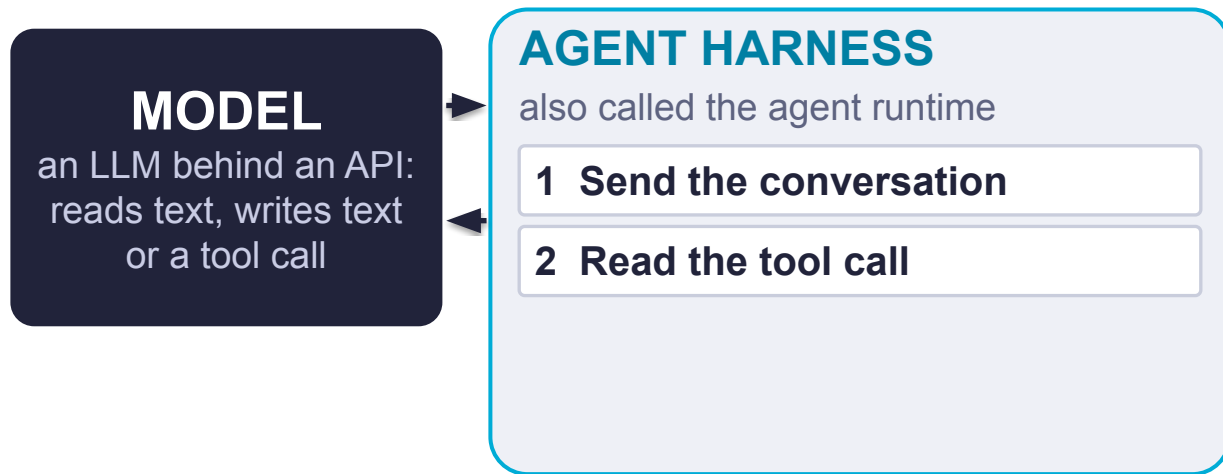
MODEL

an LLM behind an API:
reads text, writes text
or a tool call

The agent harness

how it works **1** 2 3 4

To see why the agent was told nothing, let's follow one tool call through the stack.



The agent harness

how it works **1** 2 3 4

To see why the agent was told nothing, let's follow one tool call through the stack.



The agent harness

how it works **1** 2 3 4

To see why the agent was told nothing, let's follow one tool call through the stack.



The model never sees the tool, the sandbox or the clock. **It sees only what the harness writes back in step 4.**

Why a sandbox, and why a microVM

how it works 1 2 3 4

The agent runs code and opens pages nobody has reviewed, so it runs in a sandbox: a microVM.

CONTAINER

untrusted code

the host's kernel, shared

FULL VM

untrusted code

its own kernel

big monitor, >1.4M lines

MICROVM

untrusted code

its own kernel

tiny monitor, ~50k lines

“Process isolation is the single highest-leverage mitigation in this guide.”

MCP's own security guide

Why a sandbox, and why a microVM

how it works 1 2 3 4

The agent runs code and opens pages nobody has reviewed, so it runs in a sandbox: a microVM.

CONTAINER

untrusted code

the host's kernel, shared

FULL VM

untrusted code

its own kernel

big monitor, >1.4M lines

MICROVM

untrusted code

its own kernel

tiny monitor, ~50k lines

FIRECRACKER

≤ 125 ms ≤ 5 MiB

to boot

overhead

2018

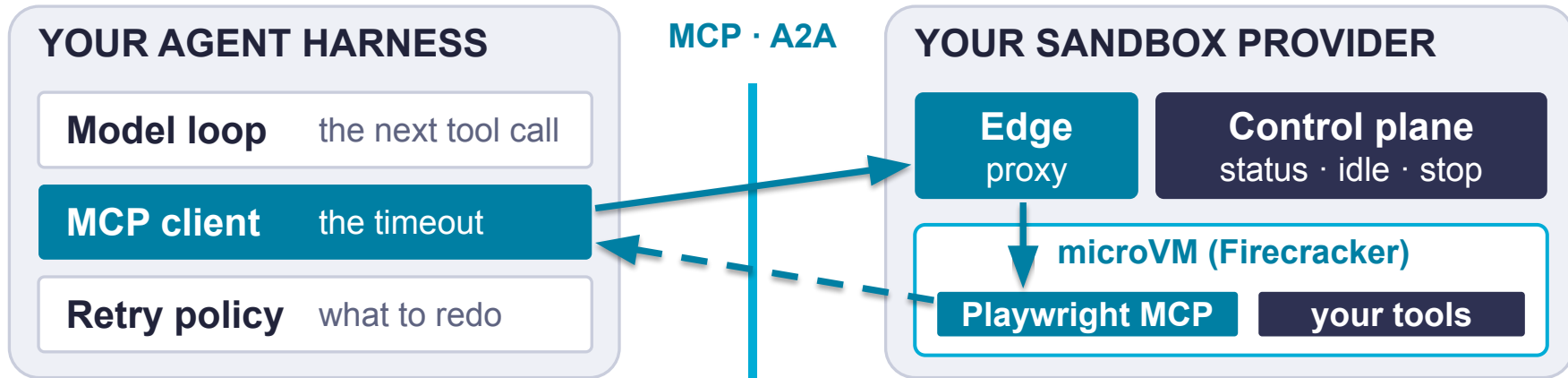
in production

One Firecracker process per microVM. **When the VM dies, the host knows at once.**

Your harness and your provider

how it works **1** 2 3 4

The tool call goes from your agent harness to your sandbox provider.



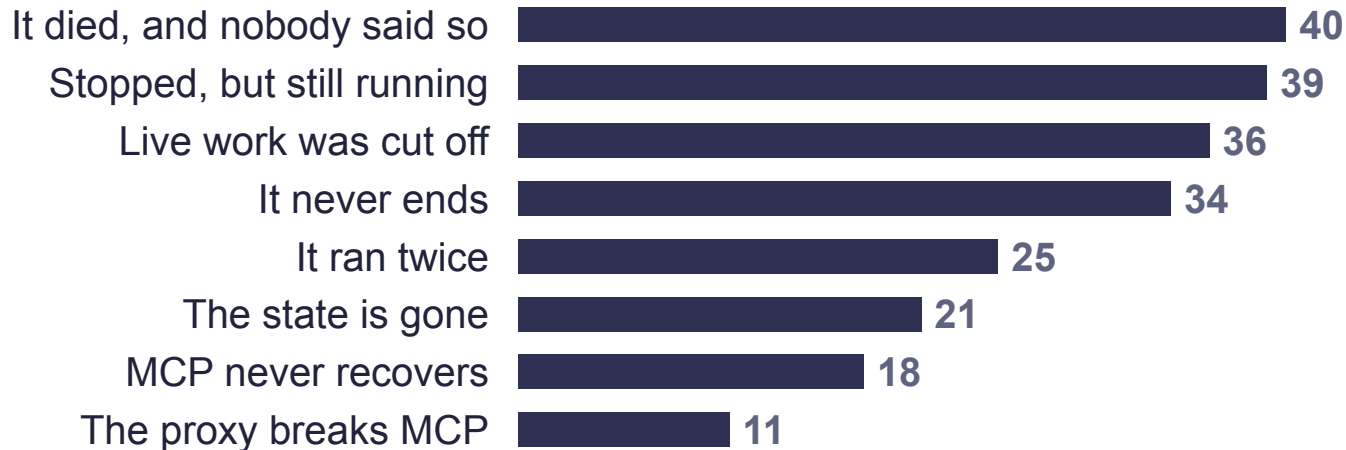
Each side learns what the other did only from what comes across.

Sometimes it isn't true.

Reported MCP problems

failure modes 1 2 3 4

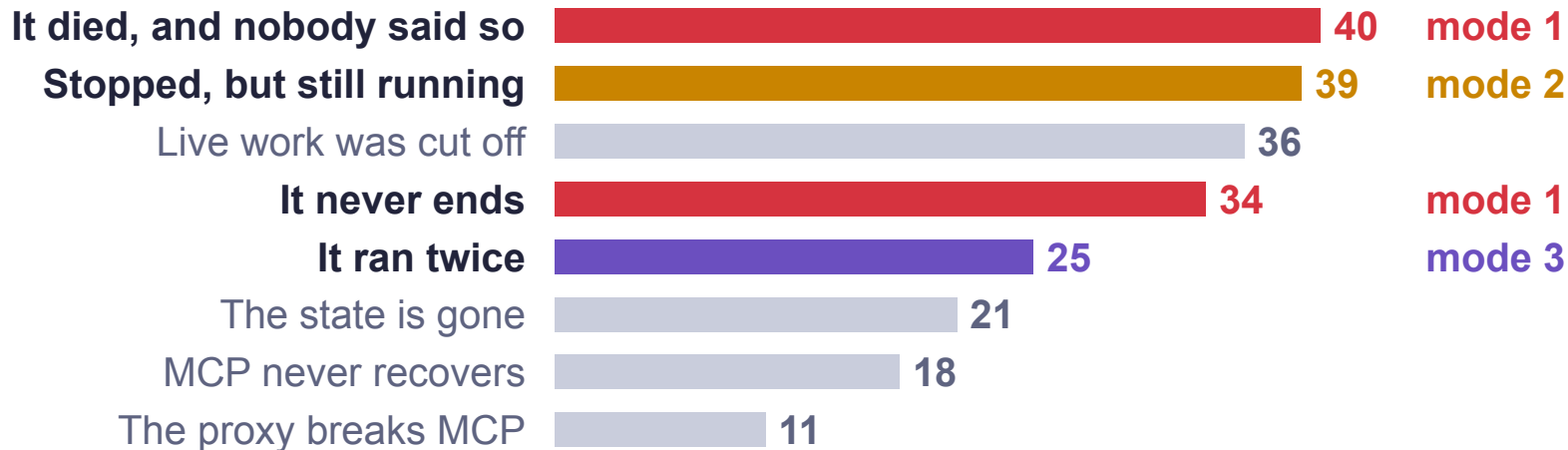
GitHub issues linked to projects forked from MCP. Three failures come out on top.



Reported MCP problems

failure modes 1 2 3 4

GitHub issues linked to projects forked from MCP. Three failures come out on top.



Failure mode 1 · “Still working.”

The sandbox dies in the middle of a tool call. The agent is told it's still working.

74 GitHub issues across 11 of 14 products

Failure mode 1 · “Still working.”

failure modes 1 2 3 4

The agent only has a timer; the host saw the VM die at once.

WHO KNEW, AND WHEN · our recording, one tool call

HOST SUPERVISOR

the Firecracker process



AGENT'S MCP CLIENT

a timer



0 s 10 20 30 40 50 60



55 s blind, and no cause

"Still working."

A browser agent asks Playwright MCP for a 20 s report. Five seconds in, the sandbox's VM dies.

RECORDING

STOCK · REAL AGENT

00:11.0

WHAT THE AGENT IS TOLD

real agent · Pi, open-weights model

WORKING

browser_navigate /reports/generate: the server renders the report for 20 s

WHAT IS ACTUALLY TRUE

host supervisor · Firecracker

SANDBOX RUNNING

Firecracker microVM running; Playwright MCP and Chromium inside

1895084
VM PROCESS

running
STATE

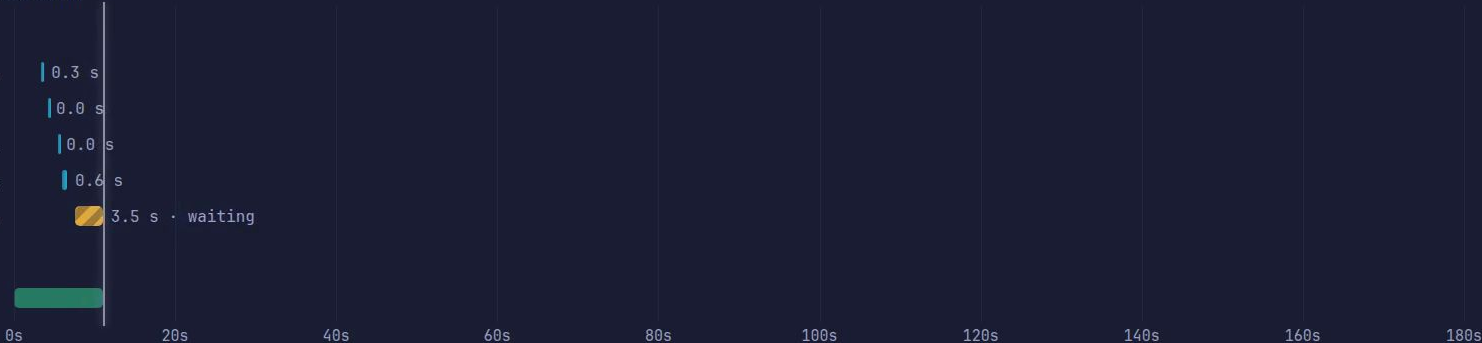
10.9
UP (S)

AGENT · tool calls, as the runtime sees them

browser_navigate {"url": "http://1... 0.3 s
browser_snapshot {} 0.0 s
browser_fill_form {"fields": [{"na... 0.0 s
browser_click {"element": "Sign in ... 0.6 s
browser_navigate {"url": "http://1... 3.5 s · waiting

TRUTH · VM · host supervisor

VM process



Stock, real agent: Pi with an open-weights model; Pi's MCP timeout is 60 s by default.

AGENT RUNTIME · what the agent does and hears

```
"type": "textbox", "target": "e5", "value": "demo-  
5.5s ← ok (0.0s)  
6.0s → browser_click {"element": "Sign in button",  
  "target": "e8"}  
6.6s ← ok (0.6s)  
7.6s agent says: Signed in successfully. Now navigating to  
  the report generation page.  
7.6s → browser_navigate {"url":  
  "http://172.16.0.1:8080/reports/generate"}
```

the slow call starts

SANDBOX HOST · what is actually happening

HOST · Firecracker supervisor 20:08:29

VM process 1895084 RUNNING

```
PID ELAPSED RSS COMMAND  
1895084 00:14 734152 firecracker --api-sock /tmp/fc.sock --config-file /opt/zt/vm.j  
provider status API (what a runtime can ask):  
GET /api/sandbox → {"state":"running","reason":"","at":""}
```

FAILURE MODE 1 OF 3

"Still working."

A browser agent asks Playwright MCP for a 20 s report. Five seconds in, the sandbox's VM dies.

RECORDING

STOCK · REAL AGENT

01:11.5

WHAT THE AGENT IS TOLD

real agent · Pi, open-weights model

AGENT SAYS

"The navigation timed out — the server is taking a while to render, as expected. Let me wait and check the page state."

WHAT IS ACTUALLY TRUE

host supervisor · Firecracker

SANDBOX DEAD

The host's supervisor recorded it at 20:08:31.250: the VM process was killed (signal 9, exit 137).

1895084

VM PROCESS

137

EXIT CODE

58.8

S DEAD

AGENT · tool calls, as the runtime sees them

browser_snapshot {}

browser_fill_form {"fields": [{"na...

browser_click {"element": "Sign in ...

browser_navigate {"url": "http://1...

browser_wait_for {"time": 30}

TRUTH · VM · host supervisor

VM process

BLIND 55.0 S · THEN GUESSED A CAUSE

0.0 s

0.0 s

0.6 s

60.0 s · error

1.8 s · waiting

killed · exit 137 · the host knew at once

+1 EARLIER CALLS

Stock, real agent: Pi with an open-weights model; Pi's MCP timeout is 60 s by default.

AGENT RUNTIME · what the agent does and hears

```
the report generation page.
7.6s → browser_navigate {"url":
"http://172.16.0.1:8080/reports/generate"}
67.6s ← ERROR (60.0s) MCP request timed out after 60000ms
69.7s agent says: The navigation timed out – the server is
taking a while to render, as expected. Let me wait
and check the page state.
69.7s → browser_wait_for {"time": 30}
71.3s ← ERROR (1.6s) fetch failed
```

the slow call starts

60 s, then: no cause

it guesses a cause

SANDBOX HOST · what is actually happening

HOST · Firecracker supervisor 20:09:29

VM process 1895084 GONE

supervisor exit record: {"pid": 1895084, "rc": 137, "t": 1792181311.250}

provider status API (what a runtime can ask):

GET /api/sandbox → {"state":"dead","reason":"the VM process was killed (signal 9, exit 137)","at":"20:08:31"}

Infrastructure solution - failure mode 1



Agentic AI Foundation

AGNTCon + MCPCon
North America

failure modes 1 2 3 4

Ask the provider if the sandbox is alive.
The agent knows in 1s, with the cause.

WITH knew in 1 s, with the cause

without: blind 55 s

```
4.2s    provider status: running
9.2s    provider status: dead
9.2s    ← STOPPED WAITING after 6.0s:
        Sandbox died at 22:27:37: the
        VM process was killed (signal
        9, exit 137). Its browser,
        pages and sign-in are gone.
```

```
# asks provider every 1 s
# the provider says: dead
# knows, with the cause
```

Failure mode 2 - “Stopped.”

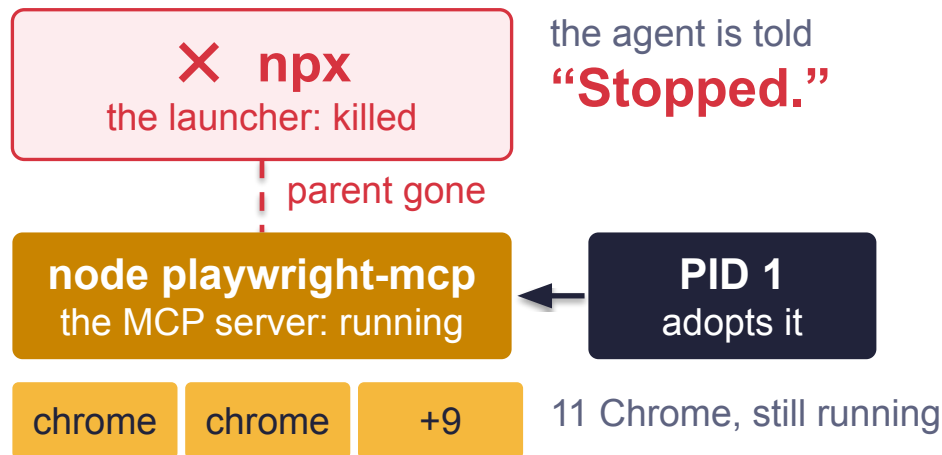
The runtime stops the MCP server. Its processes keep running.

39 GitHub issues across 11 of 14 products

Failure mode 2 · “Stopped.”

The runtime killed one process; the kernel can still see the rest running.

OUR RUN · one session's processes after “stop”



AFTER THREE STOPS

36 processes

3 MCP servers + 33 Chrome
920 of 2,048 MB in use

FAILURE MODE 2 OF 3

"Stopped."

Playwright MCP starts in the sandbox, the agent signs in, the user cancels, the runtime stops it. Three times.

RECORDING

STOCK STACK

00:06.5

WHAT THE AGENT IS TOLD

agent runtime · official MCP Python SDK

WORKING

Session 1: the agent uses the browser

WHAT IS ACTUALLY TRUE

inside the sandbox · ps, free

RUNNING

1 Playwright MCP server and 11 Chrome processes, 3 adopted by PID 1

1

MCP SERVERS

11

CHROME PROCESSES

479

MB IN USE

AGENT · tool calls, as the runtime sees them

session 1 Playwright MCP, session 1

TRUTH · processes in the sandbox

processes

VM memory (MB)



Stock: to stop a server, the runtime kills the process it started.

AGENT RUNTIME · what the agent does and hears

SANDBOX HOST · what is actually happening

SANDBOX · processes inside the microVM 22:27:55

PID	PARENT	MB	UNIT	PROCESS
1257	1	80	-	npm exec playwright-mcp :8932
1281	1257	1	-	sh -c playwright-mcp :8932
1282	1281	134	-	node playwright-mcp :8932 ← the MCP server
1337	1282	176	-	chrome ← the browser
1339	1	3	-	chrome crash handler
1342	1	3	-	chrome crash handler
1344	1337	70	-	chrome --type=zygote

FAILURE MODE 2 OF 3

"Stopped."

Playwright MCP starts in the sandbox, the agent signs in, the user cancels, the runtime stops it. Three times.

RECORDING

STOCK STACK

00:33.5

WHAT THE AGENT IS TOLD

agent runtime · official MCP Python SDK

ALL STOPPED

The runtime reports all three sessions stopped

WHAT IS ACTUALLY TRUE

inside the sandbox · ps, free

STILL RUNNING

3 Playwright MCP servers and 33 Chrome processes, 9 adopted by PID 1

3

MCP SERVERS

33

CHROME PROCESSES

920

MB IN USE

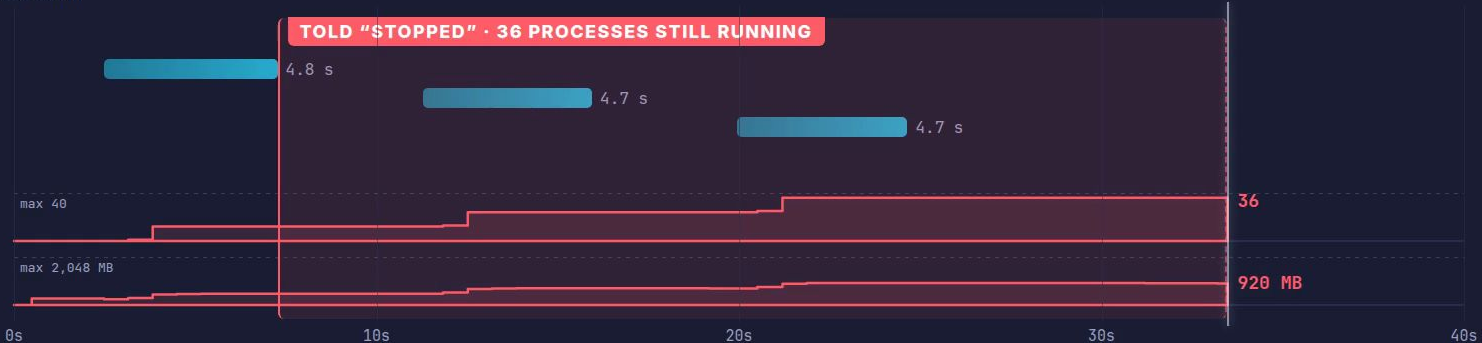
AGENT · tool calls, as the runtime sees them

session 1 Playwright MCP, session 1
session 2 Playwright MCP, session 2
session 3 Playwright MCP, session 3

TRUTH · processes in the sandbox

processes

VM memory (MB)



Stock: to stop a server, the runtime kills the process it started.

AGENT RUNTIME · what the agent does and hears

```
21.3s → tools/call browser_type User = demo-user
21.3s → tools/call browser_type Password = *****
21.4s → tools/call browser_click Sign in
21.9s → tools/call browser_navigate http://172.16.0.1:8080/reports/new
22.0s ← ok: signed in, report form open
24.5s session 3 is cancelled (the user pressed Stop): the runtime stops the server
24.6s → stop: kill 2258 (the process the runtime started) # kills only the launcher
```

(scene finished)

SANDBOX HOST · what is actually happening

SANDBOX · processes inside the microVM 22:28:23

PID	PARENT	MB	UNIT	PROCESS
1281	1	1	-	sh -c playwright-mcp :8932
1282	1281	116	-	node playwright-mcp :8932 ← the MCP server
1337	1282	174	-	chrome ← the browser
1339	1	3	-	chrome crash handler
1342	1	3	-	chrome crash handler
1344	1337	70	-	chrome --type=zygote
1345	1337	70	-	chrome --type=zygote

Infrastructure solution - failure mode 2



Agentic AI Foundation

AGNTCon + MCPCon
North America

failure modes 1 2 3 4

Run each server as one unit.
Stop the unit, then check.
Nothing left running.

WITH

0 left, 240 MB

without: 36 left, 920 MB

```
21.9s → exec in sandbox:
      systemd-run
      --unit=pw-session-3 npx
      playwright-mcp --port 8934
27.3s → stop: systemctl stop
      pw-session-3; verify: unit
      inactive, 0 processes left
```

own unit (one cgroup)

stops unit, then checks

Failure mode 3 - “Failed.”

A call times out after its work already happened. The retry does it again.

25 GitHub issues across 7 of 14 products

Failure mode 3 · “Failed.”

failure modes 1 2 3 4

The call timed out, the charge went through.
The retry charged again.

THE RACE · our recording, seconds from the first call

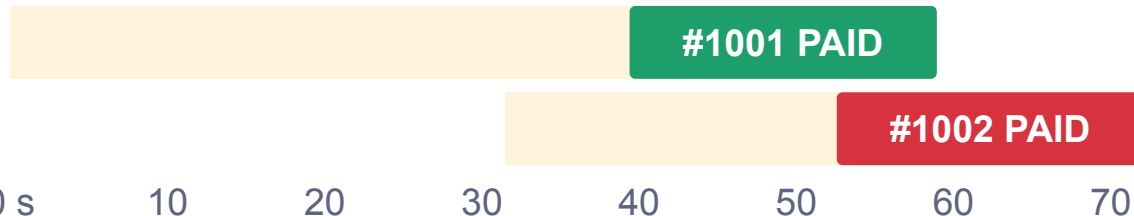
AGENT'S RUNTIME

waits 30 s, retries once



THE SHOP'S LEDGER

payment takes 40 s



Charged twice for one order. Told “failed”.

FAILURE MODE 3 OF 3

"Failed."

An MCP tool places one \$42 order. The payment takes 40 s; the client waits at most 30 s.

RECORDING

STOCK STACK

00:20.0

WHAT THE AGENT IS TOLD

agent runtime · official MCP Python SDK

WORKING

place_order Annual plan \$42.00 (attempt 1); the client waits at most 30 s

WHAT IS ACTUALLY TRUE

the shop's payment ledger

PAYMENT PROCESSING

1
ORDERS

0
CHARGES

0
CHARGES

1 order for one request · 0 paid

AGENT · tool calls, as the runtime sees them

place_order {"item": "Annual plan", ...} 17.7 s · waiting

TRUTH · orders · shop ledger

order #1001 \$42.00 payment processing

0s

10s

20s

30s

40s

50s

60s

70s

80s

Stock: the runtime's retry policy tries again after a timeout.

AGENT RUNTIME · what the agent does and hears

AGENT RUNTIME · official MCP Python SDK · scene s3 · STOCK

```
2.3s → initialize http://172.16.0.2:8000/mcp
2.3s ← connected: orders-mcp 1.0
2.3s   tool place_order: readOnlyHint=False idempotentHint=False
2.3s   tool list_orders: readOnlyHint=True idempotentHint=None
2.3s → tools/call place_order {"item": "Annual plan", # one $42 order
                                "amount": 42.0}
```

SANDBOX HOST · what is actually happening

SANDBOX · the orders MCP server log, inside the microVM

```
22:29:10 orders-mcp place_order Annual plan $42.0 key=None: calling payments API
[]
```

FAILURE MODE 3 OF 3

"Failed."

An MCP tool places one \$42 order. The payment takes 40 s; the client waits at most 30 s.

RECORDING

STOCK STACK

01:16.0

WHAT THE AGENT IS TOLD

agent runtime · official MCP Python SDK

● **FAILED**

place_order failed after 2 attempts: "MCPErrors: Request 'tools/call' timed out"

WHAT IS ACTUALLY TRUE

the shop's payment ledger

● **CHARGED \$84.00**

2 orders for one request · 2 paid

2

ORDERS

2

CHARGES

84.00

CHARGED (\$)

AGENT · tool calls, as the runtime sees them

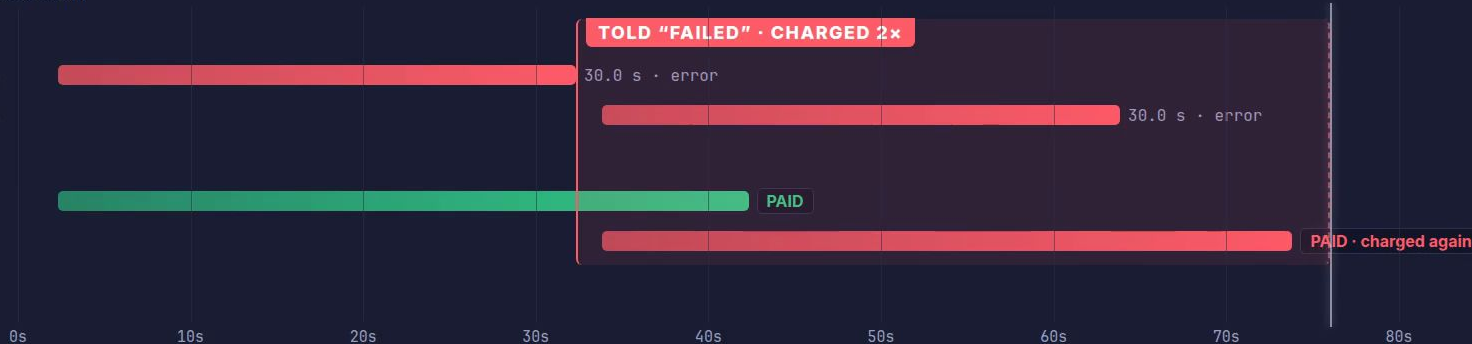
place_order {"item": "Annual plan",...

place_order {"item": "Annual plan",...

TRUTH · orders · shop ledger

order #1001 \$42.00

order #1002 \$42.00



Stock: the runtime's retry policy tries again after a timeout.

AGENT RUNTIME · what the agent does and hears

```
"amount": 42.0}
32.3s ← ERROR after 30.0s: MCPErrors: Request 'tools/call'
      timed out
33.8s → tools/call place_order {"item": "Annual plan",
      "amount": 42.0}
63.8s ← ERROR after 30.0s: MCPErrors: Request 'tools/call'
      timed out
```

shop still charging

retry: a second order

#1001 was already paid

(scene finished)

SANDBOX HOST · what is actually happening

SANDBOX · the orders MCP server log, inside the microVM

```
22:29:10 orders-mcp place_order Annual plan $42.0 key=None: calling payments API
22:29:42 orders-mcp place_order Annual plan $42.0 key=None: calling payments API
22:29:50 orders-mcp place_order → order #1001 paid
22:30:22 orders-mcp place_order → order #1002 paid
```

Infrastructure solution - failure mode 3



Agentic AI Foundation

AGNTCon + MCPCon
North America

failure modes 1 2 3 4

A timeout means “unknown”: ask before redoing.

WITH charged 1×, asked first

without: charged 2×

```
2.3s    → tools/call place_order {"item":  
    "Annual plan", "amount": 42.0,  
    "idempotency_key": "order-2e31c1b89c"}  
32.4s   ← ERROR after 30.0s: MCPError: Request  
    'tools/call' timed out  
32.4s   → timeout on a write: outcome UNKNOWN.  
    Not retrying. Asking the system that  
    knows: list_orders  
44.4s   ← the shop says: #1001 Annual plan  
    $42.00 paid key=order-2e31c1b89c
```

one \$42 order + a key

shop still charging

unknown: ask, don't redo

charged once

The pattern

Every infrastructure solution is the same move: stop guessing from a timer, and ask the part that knows.

THE GUESS

ASK THE PART THAT KNOWS

a timer



The host's supervisor
ask the provider's status

55 s → 1 s

The pattern

Every infrastructure solution is the same move: stop guessing from a timer, and ask the part that knows.

THE GUESS

ASK THE PART THAT KNOWS

a timer



The host's supervisor
ask the provider's status

55 s → 1 s

a PID



The kernel's cgroup
stop the unit, then check

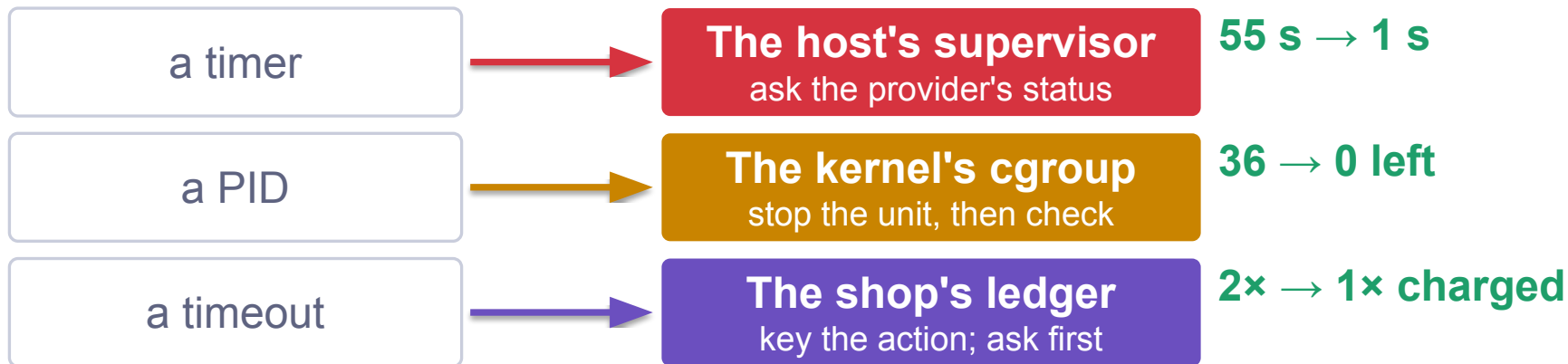
36 → 0 left

The pattern

Every infrastructure solution is the same move: stop guessing from a timer, and ask the part that knows.

THE GUESS

ASK THE PART THAT KNOWS

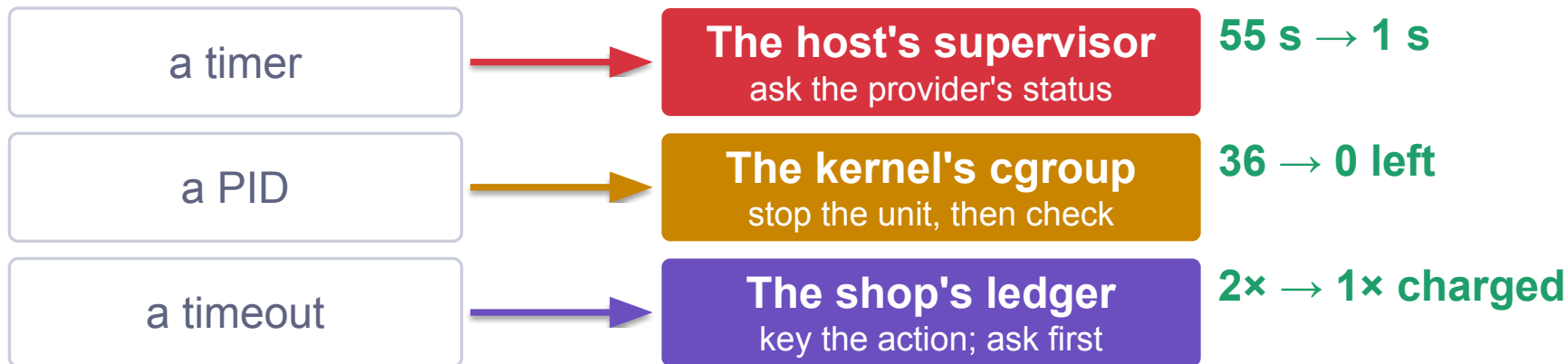


The pattern

Every infrastructure solution is the same move: stop guessing from a timer, and ask the part that knows.

THE GUESS

ASK THE PART THAT KNOWS



No new protocol. No SDK inside the agent. It all exists today.

A2A has the same three failure modes, and the infrastructure solution is already built in.

Failure mode	What went wrong (2026 GitHub issues)	What A2A already has
“Still working.”	A stream ended early, looked complete; the task stayed “working” forever	No final state? Call <code>GetTask</code>

A2A has the same three failure modes, and the infrastructure solution is already built in.

Failure mode	What went wrong (2026 GitHub issues)	What A2A already has
“Still working.”	A stream ended early, looked complete; the task stayed “working” forever	No final state? Call <code>GetTask</code>
“Stopped.”	A task waiting for input never went idle: billed to the 8-hour cap	Count input-required as idle

A2A has the same three failure modes, and the infrastructure solution is already built in.

Failure mode	What went wrong (2026 GitHub issues)	What A2A already has
“Still working.”	A stream ended early, looked complete; the task stayed “working” forever	No final state? Call <code>GetTask</code>
“Stopped.”	A task waiting for input never went idle: billed to the 8-hour cap	Count input-required as idle
“Failed.”	A redelivered message ran the whole agent again	Key side effects by task or message ID

Pick one test and run it on your stack this week.

FAILURE MODE 1

Kill the sandbox
during a long tool call.

→ Ask the provider's
status during calls.

Pick one test and run it on your stack this week.

FAILURE MODE 1

Kill the sandbox during a long tool call.

→ Ask the provider's status during calls.

FAILURE MODE 2

Stop a session, then list what's running.

→ Stop the unit, not the PID.

Pick one test and run it on your stack this week.

FAILURE MODE 1

Kill the sandbox during a long tool call.

→ **Ask the provider's status during calls.**

FAILURE MODE 2

Stop a session, then list what's running.

→ **Stop the unit, not the PID.**

FAILURE MODE 3

Make a write time out after it lands.

→ **Key the action; ask first.**

Pick one test and run it on your stack this week.

FAILURE MODE 1

Kill the sandbox during a long tool call.

→ Ask the provider's status during calls.

FAILURE MODE 2

Stop a session, then list what's running.

→ Stop the unit, not the PID.

FAILURE MODE 3

Make a write time out after it lands.

→ Key the action; ask first.

Your stack already knows the truth.
Stop letting your agent be the last to know.

Mohamed Maktal · Solutions Engineer, Vultr