WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Agent Manager Tutorial - Session 2

Menaka Jayawardena
Head of Engineering, Agent Platform BU

Viggnah Selvaraj
Lead Sales Engineer

00 MODULE

Where We Left Off

A deployed agent operating without governance controls.

5 min

WSO2

SESSION 2

Four states, one agent

01

ON A LAPTOP

One process, one 200 OK per request, and standard output as the entire record.

02

IN THE PLATFORM

Built from a git commit, deployed, and reachable only through a gateway.

03

VISIBLE

A span tree per request, with tokens and tool arguments, and no code added to get it.

04

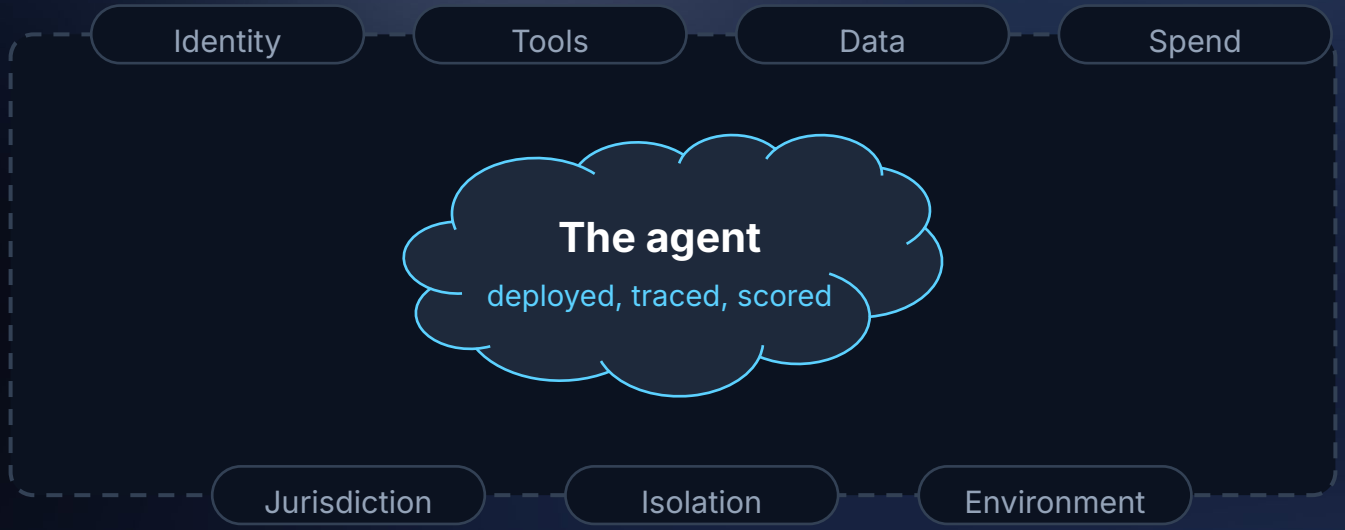
SCORED

Rule based and judge based evaluators over stored traces, including traffic served before the question existed.

And one the platform didn’t build. The same concierge, built differently and running on a laptop, registered and governed under the same control plane as everything else. The agent's code is the same code it was at the start of Session 1. Every state on this row was a change to what surrounds it.

SESSION 2

The boundary, half drawn



done **INSPECTABLE**

done **MANAGED**

open **ATTRIBUTABLE**

open **GOVERNED**

SESSION 2

Gaps, and the module that closes each

Nothing constraints which model it uses, what it spends, or what it is allowed to say.	MODULE 01
The agent endpoint still accepts unauthenticated requests.	MODULE 02
Nothing constraints which tools it may invoke, or on whose authority.	MODULE 03
No isolated environment for execution. No reusability across the organization.	MODULES 04, 05 AND 06

A note on what you are about to see. Module 01 stays with the concierge agent you already know. From module 02 onwards the domain changes, to a customer support agent and an account assistant in a banking scenario.

Different domain, different code, but every control applies unchanged.

What we are going to do

01 **LLM Governance** - Budgets and guardrails around a component that improvises.

02 **MCP Tool Governance** - Only the tools it needs.

03 **OAuth2 and Agent Identity**- Proving who is calling, and on whose behalf

04 **Agent Catalog** - Built once, found by everyone who needs it next

05 **Environment Management** - The same agent, with a different blast radius in each.

06 **Sandboxing** - Containing what the agent can reach when it misbehaves.

The Repository URL

REPO URL `https://github.com/wso2con/2026-NBO-AI-tutorial-4`

01 MODULE

LLM Governance

Guardrails and rate limiting at the gateway.

12 min

WSO2

MODULE 01

Uncontrolled data transfer

THE AGENT YOU JUST WATCHED

- a live provider key, in a file
- no cap on what it spends
- no policy over what it says
- no way to revoke it alone

WHAT IT SAYS

Unsafe, non compliant, or simply not in your voice.

A prompt is a request. The model is free to comply in a way you would never have approved.

WHAT IT COSTS

Billed per call, and per token inside the call.

The agent decides at runtime how many calls to make and how much context each one carries.

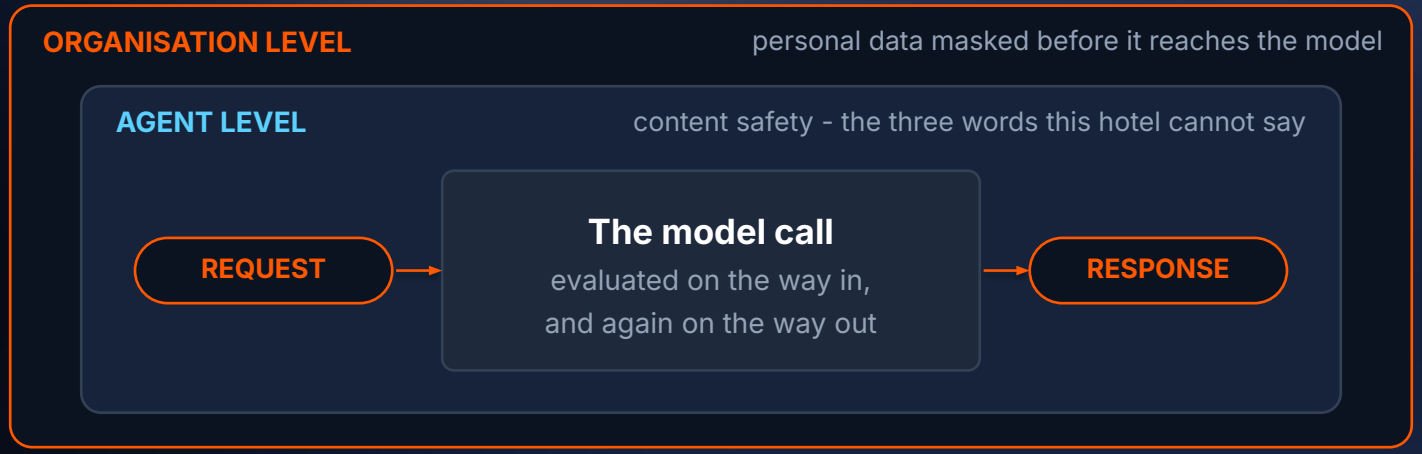
HOW OFTEN

One shared quota upstream, and many agents.

A single workload can exhaust it for everybody else without doing anything wrong.

MODULE 01

Where policy binds



Those words are the ones module 03's content safety evaluator was scoring. One scores it afterwards, one refuses it in the path.

PII masking defined outside the agent, so redeployment cannot remove them. Organisation level applies whether or not the team that built the agent knows it exists.

MODULE 01

Policies and Guardrails

Category	Policies	Notes
Block / allow	`regex`, `url`, `json-schema`, `content-length`, `word-count`, `sentence-count`	fail closed
Transform in flight	`pii-masking-regex`, `prompt-decorator`, `prompt-template`, `prompt-compressor`	change the payload, do not block
ML-backed safety	`AWS Bedrock`, `Azure Content Safety`, `Granite Guardian prompt-injection`, `NeMo Guard`, `semantic prompt guard`, `semantic tool filtering`	each needs its capability enabled on the install
Rate and cost	`basic-` / `advanced-` / `token-based-` / `llm-cost-based-ratelimit`, `llm-cost`	provider settings, not agent-level
Model routing	`intelligent`, `semantic`, `cost-based`, `time-based`, `round-robin`, `weighted`, `header router`	choose a model at the gateway
Provider translation	`OpenAI to Anthropic / Azure / Bedrock / Gemini / Mistral`	the agent speaks OpenAI; the gateway translates
Performance	`semantic-cache`	serves a similar earlier answer without calling upstream
Auth and transport	`api-key`, `JWT`, `basic`, `opaque token`, `OAuth2 generator`, `AWS SigV4`, `CORS`	inbound and upstream auth

MODULE 01

Shrink the door before you police it

WHAT AN ALLOW-EVERYTHING PROVIDER EXPOSES

organisation administration	threads	vector stores	fine tuning	images, audio, files, batches, uploads and the rest

95 operations reachable by any agent holding that key, including fourteen that administer the account

WHAT THE CONCIERGE NEEDS

`POST /chat/completions`

One operation. Set the provider to deny everything, then add that back.

This is a provider setting rather than a policy, so it costs nothing at runtime

MODULE 01

Prevention and enforcement

PREVENTION

Change what it does.

Instructions injected into every request before the model sees them, without touching the agent.

> Never promise a refund, upgrade or guarantee. Never quote a price that did not come from a tool result.

ENFORCEMENT

Change what it may return.

The guardrail reads the payload and refuses, and the refusal comes back as a policy decision rather than an outage.

> I am not able to put that in writing. Let me connect you with our duty manager, who can help.

You want both, because a model told not to say something still sometimes says it.

Nothing was rebuilt, redeployed or restarted, and nobody opened the agent's repository. For an agent another team owns, that is the difference between a policy you can state and a policy you can apply.

MODULE 01 · TAKEAWAY

Three instruments, three different jobs

TRACES

What the agent did.

WHEN after the fact

CAN IT STOP ANYTHING

no

EVALUATION

Whether it was any good.

WHEN after the fact

CAN IT STOP ANYTHING

no

GUARDRAILS

What the model may be asked, and may reply.

WHEN during the request

CAN IT STOP ANYTHING

yes

An organisation needs all three. Only guardrails are in the request path.

02 MODULE

MCP Tool Governance

Registered servers, scoped per tool.

10 min

WSO2

MODULE 02 · THE SCENARIO

Two agents, one bank

CUSTOMER SUPPORT AGENT

Answers customer questions.

Balances, account details, loan status.
It has no business moving money.

ACCOUNT ASSISTANT AGENT

Carries out account operations.

Opens accounts and transfers funds,
as well as everything support can do.

ACCOUNTS MCP SERVER · ONE SERVER, FIVE TOOLS · REACHED BY BOTH

READ

- list_accounts
- check_balance
- get_loan_status

WRITE

- open_account
- transfer_money

Today both agents can call all five. Nothing in the code, the prompt, or the platform tells them apart.

Everything from here closes the distance between what each agent should do, and what it can.

MODULE 02 · WHY

A dependency anyone can publish

WHO PUBLISHES

Any party can stand up a server.

There is no registry of record and no signing.
The endpoint you connect to is the one somebody handed you.

WHAT REACHES THE MODEL

Tool descriptions are prompt input.

The server decides the words the model reads before it picks a tool.
A downstream description change is a prompt change.

WHERE THE CONTROL IS

Registration, not the agent.

The platform pins what it registered, so an edit upstream does not silently reach the model later.

A tool description is untrusted text. It arrives from a third party and lands directly in the model's context.

MODULE 02 · THE CONTROL

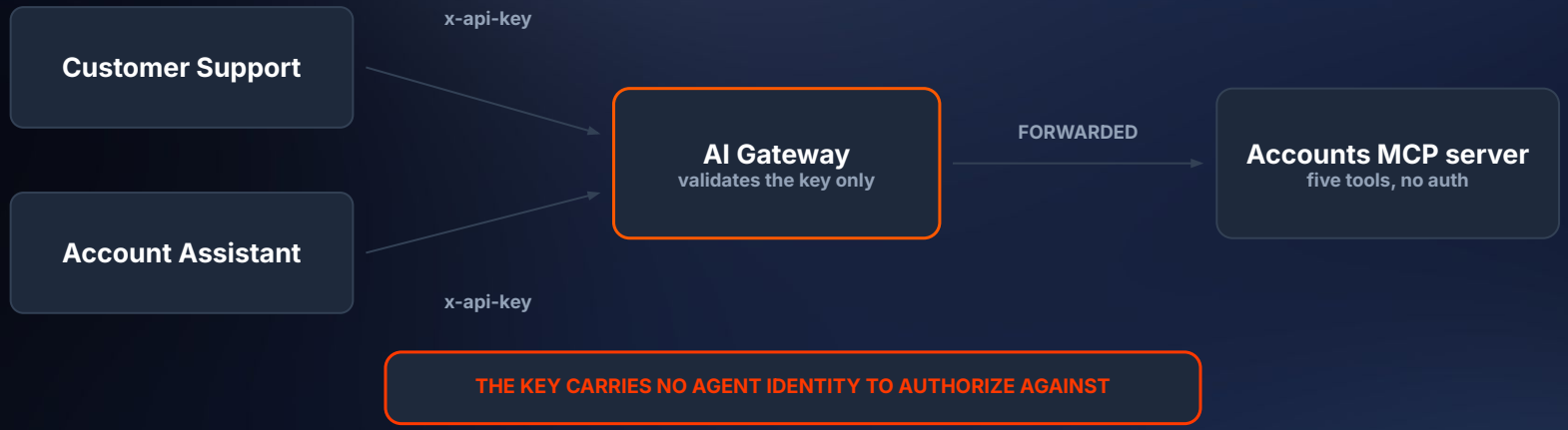
Registered, pinned, and keyed

Only registered servers are reachable from a governed agent	REACHABILITY
The endpoint, its auth scheme, and a pinned version are recorded	PROVENANCE
Tool descriptions are captured, so later edits do not reach the model	INTEGRITY
Each agent receives its own API key, sent as the x-api-key header	TRANSPORT

The key is not bound to Agent Identity. It proves the connection, not the caller. Per-tool policy is Module 03.

MODULE 02 · THE PATH

Routed, not yet restricted



Routing is centralized. Access is not. Necessary, but not sufficient.

MODULE 02 · CODE CHANGE

What has changed in the agent code?

BEFORE

```
mcp_gateway_url =
os.environ.get("MCP_SERVER_URL")
```

AFTER

```
mcp_gateway_url =
os.environ.get("MCP_GATEWAY_URL")
api_key =
   os.environ.get("MCP_GATEWAY_API_KEY")
```

SET API KEY HEADER

```
headers = (
        {"x-api-key": api_key}

    return MultiServerMCPClient(
        {
            "accounts": {
                "url": mcp_gateway_url,
                "transport": "streamable_http",
                "headers": headers,
            }
        }
    )
```

MODULE 02 · DEMO

Route the calls, then see what did not change

01 Expose the local Accounts MCP server through ngrok so Agent Manager can reach it

02 Register it at the organization level with API Key authentication

03 Bind both agents; each receives its own MCP_GATEWAY_URL and MCP_GATEWAY_API_KEY

04 Repeat the baseline prompt on Customer Support: open a savings account for cust-2

It still succeeds. Both agents hold a valid key, so both still reach every tool.

03 MODULE

OAuth2 and Agent Identity

Identity for the caller, and for the agent.

20 min

WSO2

03.1 MODULE

OAuth2

Authenticating the caller and the end user.

10 min

WSO2

MODULE 03.1 · WHY

An API returns data. An agent acts.

WHAT AN API NEEDS

Which application is calling.

An API returns data and stops. Whoever called it decides what to do next, so that is where the real decision gets made.

WHAT AN AGENT NEEDS

Which person it is acting for.

An agent does not return data and stop.
It opens the account.
So the bank has to know whose instruction that was.

WITH ONLY THE FIRST

Every action is recorded as the agent.

The transfer happened, and the log says the Account Assistant did it. Nobody can say who asked for it, or whether they were allowed to.

Two things have to be established at the door. Which client is calling, and which person they are calling for.

MODULE 03.1 · THE PIECES

Four parts, none of them in the agent

KEY MANAGER

Issues access tokens.

External to the platform or built-in.

Asgardeo in this session.

GATEWAY

Validates before the agent.

Signature and scopes are checked at the edge, not in the workload.

ASSERTION

Verified identity, forwarded.

`x-forwarded-authorization` carries the caller through to the agent.

SCOPES

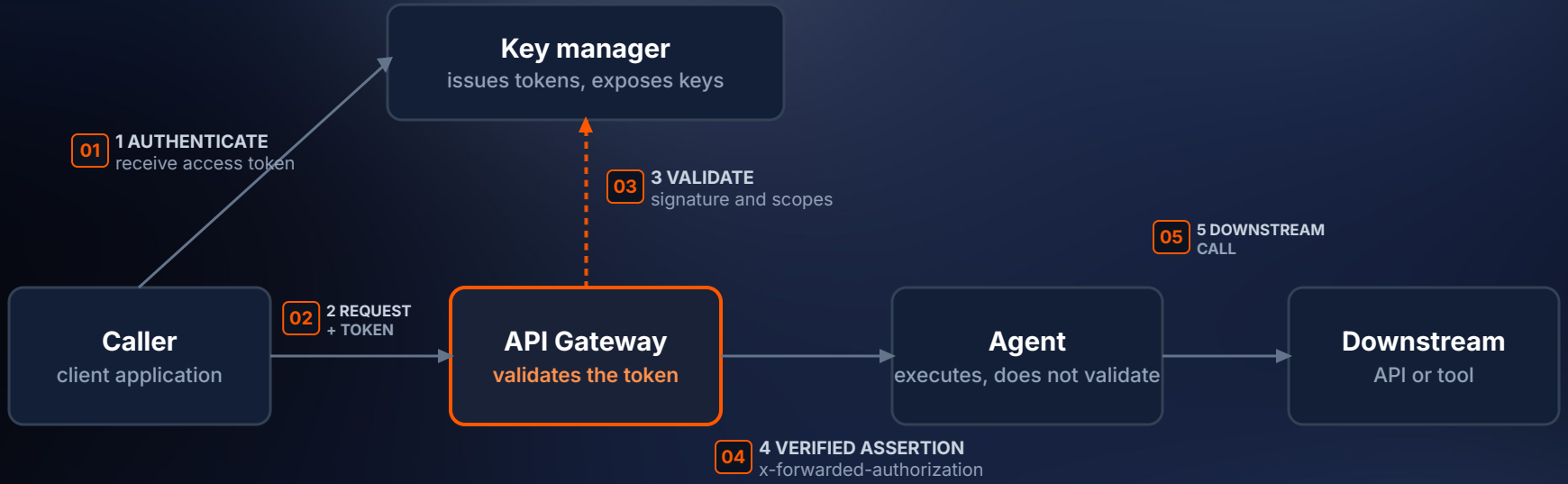
What the token authorizes.

Declared on the token, enforced by the gateway.

Configuration, not code. All four live outside the agent and survive a redeploy.

MODULE 03.1 · THE FLOW

The gateway validates, the agent trusts



The gateway validates the token; the agent receives a verified assertion.

MODULE 03.1 · DEMO

Close the endpoint, then open it properly

01 Create an Asgardeo application and register it as a key manager on Agent Manager's default gateway

02 Confirm the Customer Support Agent's /chat endpoint accepts unauthenticated requests today

03 Apply OAuth2 security to the deployed Customer Support Agent

04 Retry the same request and confirm it now fails with 401

05 Sign in through the Asgardeo-integrated web client and verify the authenticated request succeeds

Same endpoint, same agent. The only thing that changed is who the gateway will let through.

03.2 MODULE

Agent Identity

A non-human principal the platform can authorize.

10 min

WSO2

MODULE 03.2 · WHY

The agent needs its own identity

HUMAN IDENTITY

A person signs in.

Permissions follow the person. When they log out, the session ends.
This is what every access control system was designed for.

SERVICE IDENTITY

A fixed credential in config.

It does not expire and it is not tied to anyone.
Whoever holds it gets everything it can do, and the logs only ever show the service.

AGENT IDENTITY

The agent has an account of its own.

It can be given specific permissions, its actions are recorded against it, and it can be revoked in one place without touching anything else.

An agent acts on its own, after the person who asked has gone A user session needs that person present. A service account cannot record who asked or be granted less than everything.

MODULE 03.2 · THE PROBLEM

The confused deputy

WITHOUT AGENT IDENTITY

Caller is authorized for: **account read**

Agent credential permits: **account read** | **open account** | **transfer funds**

OPERATIONS THE CALLER CANNOT PERFORM DIRECTLY

WITH AGENT IDENTITY AND ON-BEHALF-OF DELEGATION

Effective permission: **account read** — intersection of caller and agent, not the union

Audit records the agent, the delegating user, and the tool invoked.

MODULE 03.2 · THE RULE

Connecting is not the same as being allowed

The Accounts MCP server exposes five tools; reading a balance and moving money are not one decision	PER-TOOL
Reading a balance, opening an account, and moving money are three different risks. One permission cannot cover all three.	RISK TIER
The grant binds to the agent asking, not to the connection it happens to hold	PRINCIPAL
If the smallest grant you can make is the whole server, least privilege is only a phrase	PRIVILEGE

A **Permission has to be smaller than the connection.** Per tool, bound to the Agent Identity, checked on every call.

One role per agent, evaluated per call. The gateway resolves the Agent Identity, then checks the scope for that tool.

MODULE 03.2 · THE SWAP

What the platform injects

MODULE 02 · API KEY

Customer Support

MCP_GATEWAY_API_KEY
its own value
no identity attached

Account Assistant

MCP_GATEWAY_API_KEY
its own value
no identity attached

NOT BOUND TO AGENT IDENTITY - EVERY TOOL STAYS REACHABLE

MODULE 03 · AGENT IDENTITY

Customer Support

AMP_AGENTID_*
client id · secret
token endpoint · scopes

Account Assistant

AMP_AGENTID_*
client id · secret
token endpoint · scopes

BOUND TO AGENT IDENTITY - AUTHORIZED PER TOOL

An API key proves the connection. It does not say which agent is calling.
Agent Manager swaps it for per-agent credentials.

What has changed in the agent code?

READ THE ENVIRONMENT VARIABLES

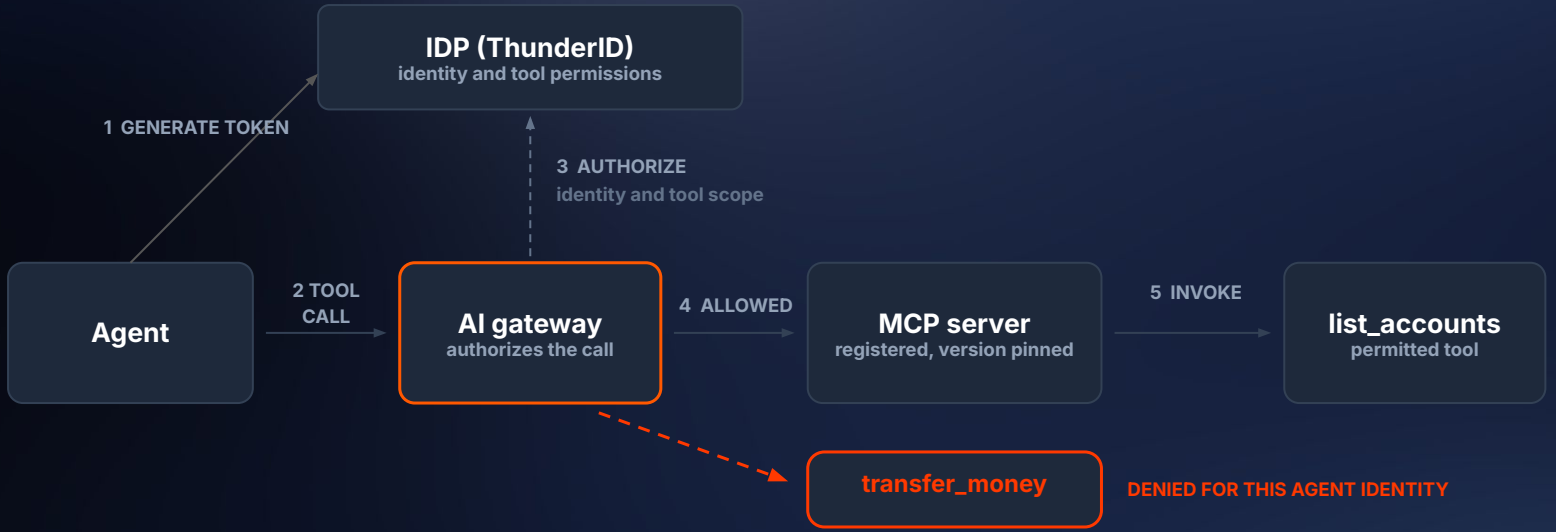
```
  mcp_gateway_url =
  os.environ.get("MCP_GATEWAY_URL")
  client_id =
os.environ["AMP_AGENTID_CLIENT_ID"]
    client_secret =
os.environ["AMP_AGENTID_CLIENT_SECRET"]
    token_endpoint =
os.environ["AMP_AGENTID_TOKEN_ENDPOINT"]
    scopes =
os.environ.get("AMP_AGENTID_SCOPES")

data = {"grant_type": "client_credentials",
"resource": mcp_gateway_url}
    if scopes:
        data["scope"] = scopes

    token_response = requests.post(
        token_endpoint,
        auth=(client_id, client_secret),
        data=data,
        timeout=30,
    )
```

SET AUTHORIZATION HEADER

```
headers = (
        {"Authorization": f"Bearer
{_mint_agentid_token(mcp_gateway_url)}"}
    )
```

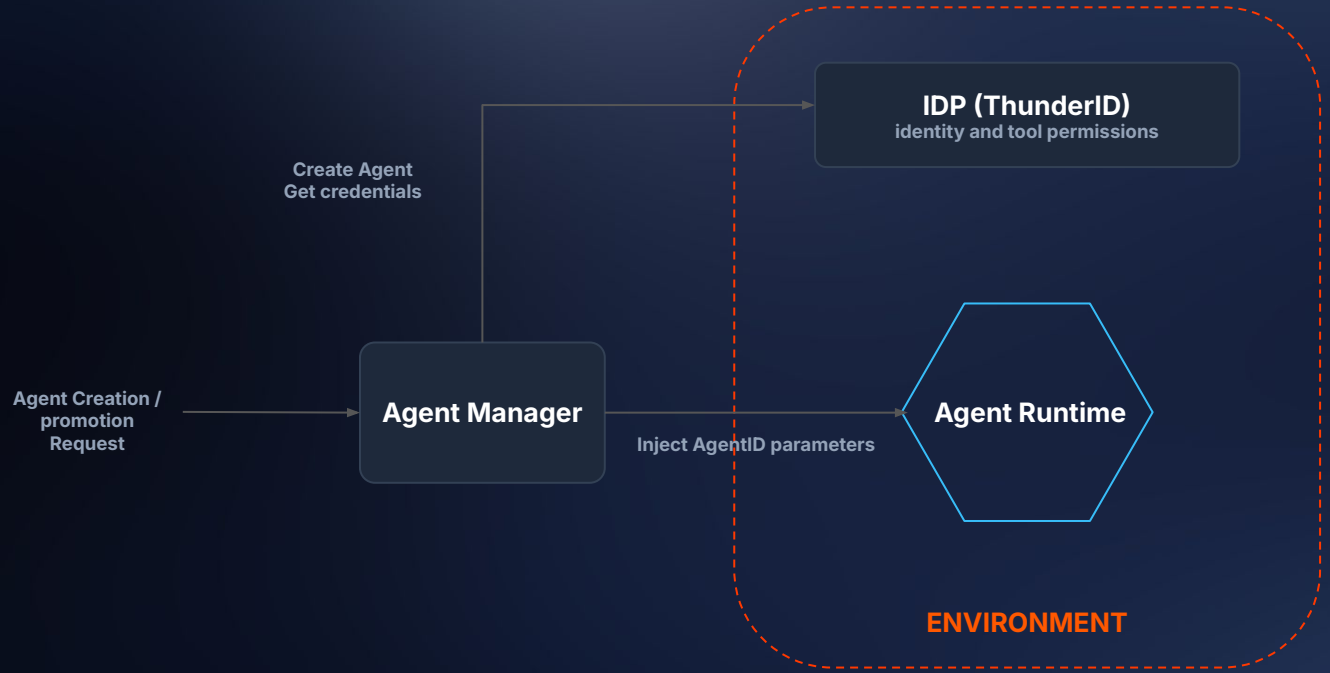
MODULE 03.2 · THE PATH

The tool call path



Permissions bind to Agent Identity and are evaluated per tool, not per server.

How an Agent gets identity?



MODULE 03.2 · DEMO

The same prompt, two different answers

01 Send the same prompt to both agents at once in the side-by-side console

02 Note each agent's Agent Identity in the console

03 Change the MCP server's auth type from API Key to OAuth2

04 Define scopes per tool: accounts:read, loans:read, accounts:write

05 Create customer-support-role and account-assistant-role, and assign each to the matching Agent Identity

06 Resend the identical prompt: Account Assistant succeeds, Customer Support is denied

07 Confirm read still works by asking Customer Support for the loan status of cust-2

04 MODULE

Agent Catalog

Publishing governed agents for controlled reuse.

10 min

WSO2

MODULE 04 · WHY

The same agent, six times over

BUILT AGAIN

Every team writes its own.

Functionally equivalent agents, each with its own prompt and its own bugs.

NEVER FOUND

No discovery, no shared review.

Nobody can see what already exists, so nobody reuses any of it.

RISK MULTIPLIED

Duplicated effort is the small problem.

Every copy carries its own unreviewed access to the very same systems.

The catalog makes reuse the reviewed path. Publication becomes the gate, rather than a copied repository.

MODULE 04 · WHAT TRAVELS

The build travels. Nothing else does.

The agent's build, its code, at a captured version	CARRIES
Build name, release date, and source	CARRIES
LLM providers, MCP servers, environment variables, secrets	DOES NOT
Agent Identity, and any role assigned to it	DOES NOT

A new agent starts with no tool access at all. Until a role is assigned to its fresh Agent Identity, it can reach nothing.

Unpublishing a Kind removes the catalog entry only. Agents already deployed from it keep running.

MODULE 04 · THE FLOW

Publish once, instantiate clean

01 RUNNING
as deployed

Agent

built, Module 01

Active running instance configured and executing in its environment.

→

02 PUBLISH
as a Kind

Kind

captures the build

Immutable, versioned blueprint containing the code definition.

→

03 CATALOG
new entry

Agent Catalog

stores published Kinds

Central registry holding versioned entries ready for reuse across teams.

→

04 DEPLOY
new identity

New Agent

own config, own ID

Instantiated clean with brand new permissions, identity, and env vars.

The build is inherited. LLM providers, MCP servers, env vars, and Agent Identity are not.

MODULE 04 · DEMO

Publish once, instantiate clean

01 Publish the deployed Customer Support Agent as a new Agent Kind from its Publish tab

02 Add a New Agent, choose Platform-Hosted, then Agent Catalog as the source, and pick the new Kind

03 Fill in the new agent's own LLM Providers, MCP Servers, and environment variables; none are pre-filled

04 Deploy, then check Security and Agent Identity to confirm a fresh identity with no role assigned

Reuse the build, not the trust. The new agent inherits code, and nothing else.

05 MODULE

Environment Management

One artifact, a distinct policy set per environment.

10 min

WSO2

MODULE 05 · WHY

The same agent should not be able to do the same things everywhere

MISTAKES COST DIFFERENT AMOUNTS

A wrong answer in testing is free.

The same wrong answer in production opens an account or moves money. The agent cannot tell the difference, so the platform has to.

YOU CANNOT TEST IT ONCE

The agent answers differently every time.

You have to run it again and again to trust it, which means you need somewhere that running it repeatedly costs nothing.

THE RULES LIVE OUTSIDE THE CODE

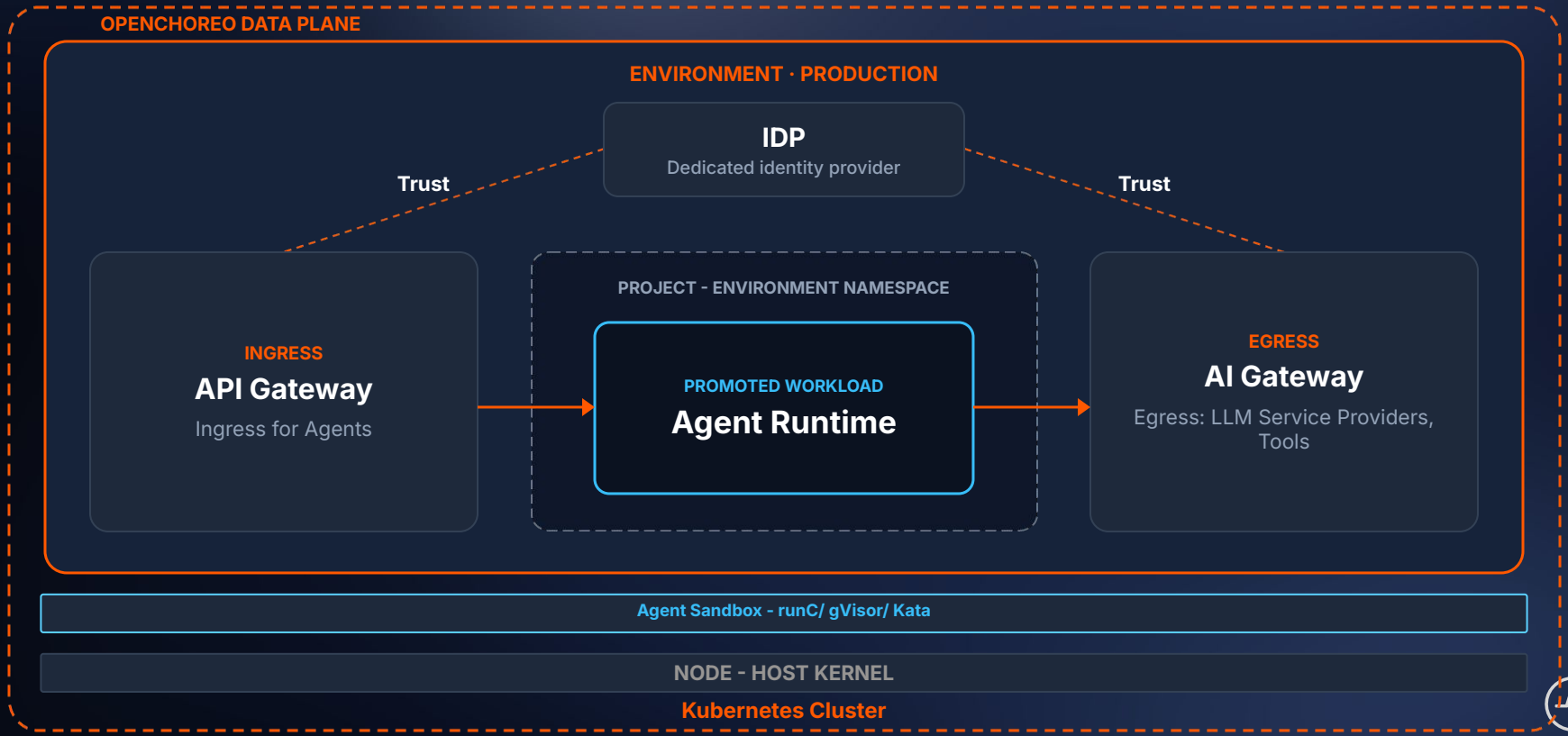
Providers, tools, scopes and limits are all configuration.

Since none of it is in the build, it can differ from place to place. An environment is where one set of those rules is held.

An environment is a set of permissions the agent inherits by being there. The build is the same everywhere. What it is allowed to touch is not.

MODULE 05 · ANATOMY

What an environment actually contains



MODULE 05 · PROMOTION

The artifact moves. The policy does not.

WHAT MOVES

The agent build.

Promoted from one environment to the next, byte for byte unchanged.

WHAT STAYS

The policy envelope.

Providers, tool allowlists, limits and credentials belong to the environment, not the agent.

WHAT MUST NOT

Escalation.

Promotion is not a route to permissions the agent did not already hold.

Development is permissive by design. That is exactly why its configuration must not travel with the build.

MODULE 05 · COMPARED

Same build, two different blast radii

	DEVELOPMENT	PRODUCTION
Model access	any registered provider	approved providers only
Tools	mock or sandboxed servers	live registered servers
Limits	permissive, for iteration	restrictive and cost-bounded
Identity	shared test IDP	dedicated identity provider
Data	synthetic records	live customer data

06 MODULE

Sandboxing

Constraining what the agent runtime can reach.

10 min

WSO2

MODULE 06 · WHY

Assume it misbehaves

IT ACTS ALONE

Code runs and tools fire with no human in the loop.

The runtime decides at execution time what to call.

THE QUESTION

Not whether, but how far.

Containment is about reach during failure or compromise, not about preventing it.

WHERE IT BINDS

The runtime, not the prompt.

An instruction is a request. A sandbox is a boundary.

Every other control in this session assumed good faith. This one does not.

MODULE 06 · BOUNDARIES

Four things to fence

NETWORK

Egress destinations.

Which hosts the runtime is permitted to reach at all.

FILESYSTEM

Persistent or ephemeral.

What survives the invocation, and what is discarded with it.

LIFETIME

Per-invocation or long-running.

How long a compromised runtime stays available to an attacker.

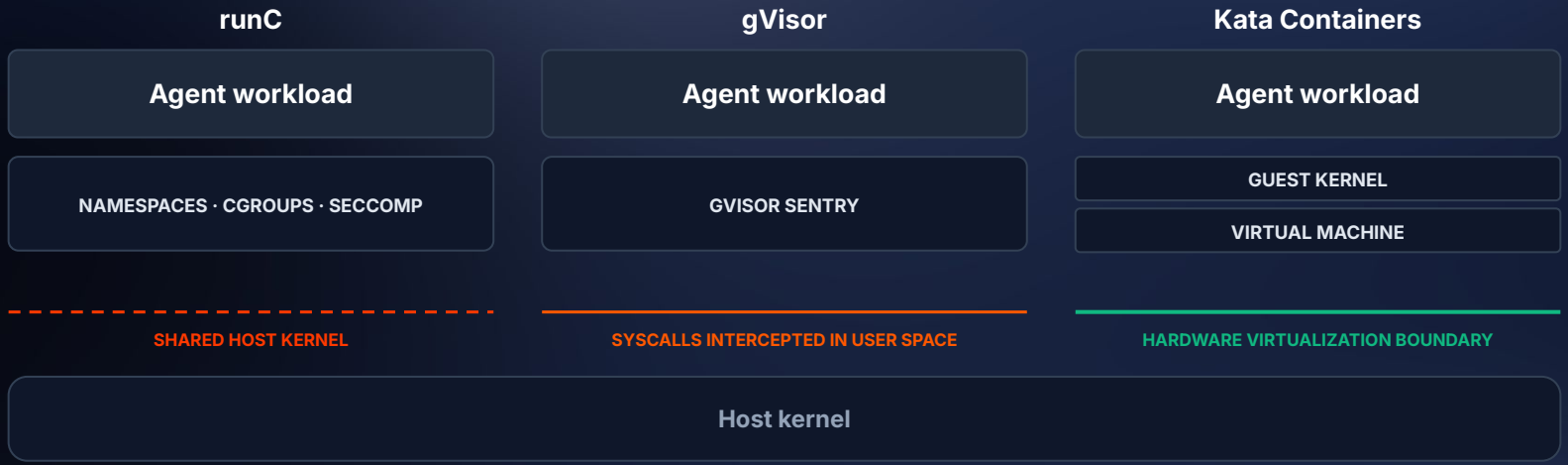
RESOURCES

CPU, memory, time.

Ceilings that bound a runaway loop before it bounds your budget.

MODULE 06 · TIERS

Where the isolation boundary sits



MODULE 06 · TIERS

Three tiers, three prices

runC

Shared host kernel.

Isolation from namespaces, cgroups and seccomp.
Lowest overhead, and a kernel vulnerability permits escape.

gVisor

A kernel in user space.

Syscalls are intercepted by the Sentry rather than passed to the host.
Smaller attack surface, and syscall-heavy code pays for it.

Kata Containers

A virtual machine per workload.

A dedicated guest kernel behind a hardware boundary.
Strongest isolation, highest startup latency and memory footprint.

The boundary moves down the stack. Namespaces, then a user-space kernel, then hardware.

MODULE 06 · CHOOSING

Pick by how far you trust the code

An agent your team built and reviewed	runC
An agent from the catalog, or a third party	gVisor
An agent that writes and runs its own code	KATA

Agent Manager (default - runC): Configurable per Environment.

Increased isolation is always paid for in startup latency and density. Choose the weakest tier the code justifies.

SESSION 2 · TAKEAWAY

What changed

INSPECTABLE
traces, and scores over stored traces

MANAGED
promote, roll back, suspend, revoke an identity

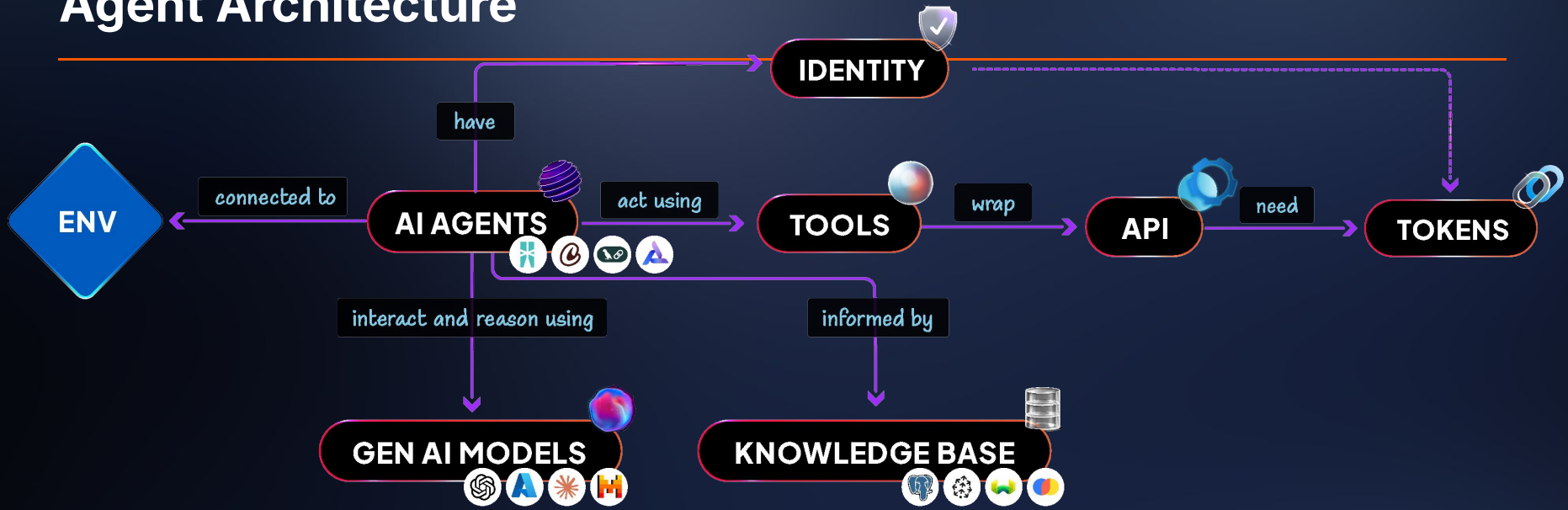
ATTRIBUTABLE
agent identity, delegation, authorisation per tool

GOVERNED
guardrails, access control, budgets, environments

The agent's code only changed minimally in this session. Everything happened around the core agent logic.

SESSION 2 · TAKEAWAY

Agent Architecture



WHAT YOU NEED TO BUILD & OPERATE

1 Build & Integrate
Agent, MCP building

2 Govern and Guardrail
Gateway

3 Define Identity and Access
Agent ID

4 Observe and Evaluate

5 Manage the Lifecycle

Agent Manager - govern and scale Agentic apps in the enterprise

SESSION 2 · WHERE TO START

Two things

01 THIS WEEK

Run one.

02 IN A MONTH

Register one you already have, or pick a process to make agentic

START HERE

`console.agent-manager.cloud.wso2.com`

THE TUTORIAL

`wso2.github.io/agent-manager/docs/v1.0.0/tutorials/create-your-first-agent`

WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Thank You!

Menaka Jayawardena

Technical Lead & Head of Engineering, Agent Platform BU

Viggnah Selvaraj

Lead Sales Engineer