USER INSTRUCTIONS / UBUNTU 26.04 LTS

Private MCP Gateway & AI Tool Governance

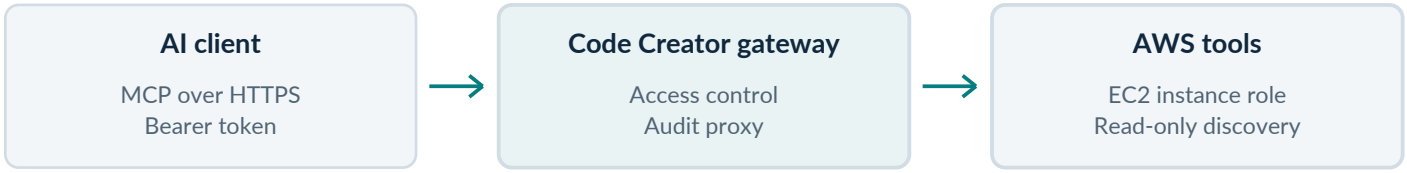
by Code Creator

Connect AI applications to approved AWS tools through a customer-owned HTTPS gateway with scoped access and audit visibility.

Govern access. Make AWS tools useful.

Deploy a Model Context Protocol (MCP) control layer in your own AWS account. Instead of distributing AWS access keys to AI applications, grant each MCP client a scoped token and let the EC2 instance role authorize the underlying AWS API calls.

How the gateway works



Audit records stay on the instance by default; optional export sends them to CloudWatch Logs in your AWS account. Tool results are returned to the connected AI client.

Five built-in AWS tools Discover S3 buckets and object metadata, EC2 instances, Lambda functions and CloudWatch alarms. One separate readiness tool is included.

Scoped client access Issue distinct tokens for different AI applications, restrict access to registered MCP servers and revoke access as needed.

Operational visibility Local JSONL audit records, customer-specific pseudonymous client IDs, automatic rotation and optional CloudWatch log shipping.

> **What this is not.** The browser is a readiness page, not a chatbot or graphical administration console. Use SSH to administer the gateway and an MCP-compatible application to call tools.

In this guide: Launch and connect (p. 2) · AWS permissions and tools (p. 3) · Client access (p. 4) · Audit logging (p. 5) · Troubleshooting (p. 6) · IP recovery (p. 7).

Launch and connect

First boot provisions the HTTPS endpoint, unique customer credentials, the restricted starter token, all six MCP tools and secure local auditing. No AWS access keys need to be entered into the gateway.

Before launching

Choose an x86-64 EC2 instance, an SSH key pair, and a network with public IPv4 connectivity, DNS and Internet access for HTTPS certificate issuance. For AWS tools or CloudWatch export, configure the optional IAM instance profile (pages 3 and 5).

> **Use a stable public IPv4 address.** The endpoint is based on your public IPv4 address and nip.io. Prefer an Elastic IP and make sure the intended address is in use when the endpoint is provisioned. If you associate a different IP later, or Stop/Start changes it, follow page 7. Ordinary EC2 reboot with the same IP was validated.

Inbound TCP port	Recommended security group access
22	Administrator IP addresses or approved management network only.
80 / 443	HTTPS clients and the access needed for public TLS certificate validation. Limit exposure where feasible without blocking certificate issuance or renewal.
Other ports	Do not expose internal PostgreSQL, MCP or audit service ports (5432, 8000, 8080, 8081, 2019).

Get connected

1. **Launch from AWS Marketplace.** Wait for first-boot provisioning and the HTTPS certificate to finish. Connect by SSH as **ubuntu** with your EC2 key.
2. **Check readiness.** Run the following command; continue when the status is Ready.

```
sudo codecreator-ai-tool-control-status
```

3. **Retrieve your private connection details.** Keep this file and its token confidential.

```
cat /home/ubuntu/FIRST_LOGIN.txt
```

MCP client setting	Value
Transport	Streamable HTTP
Server URL	https://<your-public-ip>.nip.io/mcp
Authentication	Bearer token from FIRST_LOGIN.txt
Test tool	`cc_readiness__get_control_center_status`
Expected result	**status: ready**

Browser check: Open the HTTPS root URL (without /mcp). The Ready page may display the original “Private AI Tool Control Center with MCP” name. Opening /mcp without a token should report a missing MCP client access token; this is expected.

Activate AWS discovery tools

The AWS Starter Bundle is pre-registered. To use its five read-only operations, attach an appropriately scoped EC2 IAM role and create an AWS-authorized MCP client (page 4). The readiness-only starter works without an IAM role.

1. Review the available tools

Registered tool	Purpose and required IAM action
`cc_aws__s3_list_buckets`	Lists bucket names. **s3:ListAllMyBuckets**
`cc_aws__s3_list_objects`	Lists object keys and sizes, not file contents. **s3:ListBucket** on selected buckets.
`cc_aws__ec2_list_instances`	Lists instance IDs, types and states. **ec2:DescribeInstances**
`cc_aws__lambda_list_functions`	Lists function names and runtimes. **lambda:ListFunctions**
`cc_aws__cloudwatch_list_alarms`	Lists alarm names and states. **cloudwatch:DescribeAlarms**

The regional EC2, Lambda and CloudWatch operations use the instance Region. The tools return bounded discovery results, not complete inventories; a truncated result may omit additional resources. S3 object listing requires a bucket name, with optional prefix and a maximum of 25 keys per call.

2. Configure the EC2 instance role

1. **Create a dedicated role in IAM.** Select AWS service / EC2 as the trusted service (ec2.amazonaws.com). Add only the permissions required by your deployment. Avoid administrator or full-access policies.
2. **Apply least-privilege permissions.** In IAM, create a custom policy with the JSON editor, using the example below for the four account/Region discovery APIs. Remove actions you do not need and attach the policy to the EC2 role.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:ListAllMyBuckets", "ec2:DescribeInstances",
      "lambda:ListFunctions", "cloudwatch:DescribeAlarms"],
    "Resource": "*"
  }]
}
```

For optional object listing, add a separate Allow statement for **s3:ListBucket** limited to approved bucket ARNs, such as `arn:aws:s3:::your-approved-bucket`. No s3:GetObject grant is needed.

3. **Attach the role.** At launch select it under Advanced details > IAM instance profile. For an existing instance, use EC2 > Actions > Security > Modify IAM role.

Keep IMDSv2 required. For container access to instance-role credentials, a metadata response hop limit of 2 may be needed. Ensure the instance can reach the required AWS API endpoints.

AWS reference: Attach an IAM role to an EC2 instance.

Control what each AI client can use

Use separate MCP client tokens for each AI application. The starter token is intentionally limited to the readiness server, so it cannot discover or invoke the AWS Starter Bundle.

Create an AWS-authorized client

From an administrator SSH session, run:

```
sudo codecreator-ai-tool-control-admin create mcp-client \
  aws-reader --allow "cc_aws"
```

Securely copy the returned client token into your MCP-compatible application. Use the same HTTPS /mcp endpoint and Streamable HTTP transport as the readiness test, but replace the starter token with this new token.

Scope	Tool access
`cc_readiness`	Readiness test only; no AWS tools.
`cc_aws`	All five built-in AWS tools; no readiness tool. AWS actions remain subject to the EC2 role.

Two layers of authorization. MCP tokens restrict access by registered server or bundle; the five AWS tools share the instance role. This is not per-tool IAM or a separate AWS role for every client. Grant only the AWS permissions you intend the bundle to use.

Try a real operational question

In your connected AI client, ask “Which EC2 instances are running?” or “Show my Lambda functions and runtimes.” For S3 object listing, specify an approved bucket and optional prefix. A successful connection alone does not guarantee IAM access. Review tool errors and any truncated results.

Manage and revoke access

List the registered tools, review clients, or revoke a named client. Choose a unique name when creating each new application token.

```
sudo codecreator-ai-tool-control-admin list tools
sudo codecreator-ai-tool-control-admin list mcp-clients
sudo codecreator-ai-tool-control-admin delete mcp-client \
  aws-reader
```

Deleting a client revokes its token. This is the tested removal path; invalid and revoked tokens are rejected. To replace a token, create a new limited client, update your application, then delete the old client.

Extend the gateway

Register additional compatible MCP servers through the SSH administration helper. Review the supported registration options first:

```
sudo codecreator-ai-tool-control-admin register --help
```

Grant access only after reviewing a server and its permissions. The included integrations are AWS discovery tools, not a general integration marketplace.

Monitor AI tool activity

The dedicated audit proxy records recognized MCP tool invocation attempts and supported response classifications through the HTTPS /mcp route. Local records are available even when no IAM instance profile is attached.

Local audit history

```
sudo tail -n 20 /var/log/codecreator-ai-tool-control/audit.jsonl
```

Records contain timestamps, event IDs, pseudonymous client IDs, tool names, phases and outcome labels, with status codes or durations where available. The proxy does not record bearer tokens, tool arguments or result payloads. The local log is restricted and configured for daily rotation, with a 32 MiB size threshold and up to 30 rotations retained.

> **Interpret outcomes carefully.** A JSON-RPC result is not proof of AWS API success; inspect the client result. Streaming (SSE) and batch response final-outcome classification has not been fully verified. This is tool-call auditing, not a record of every network request or a compliance certification.

Optional CloudWatch Logs export

The CloudWatch agent is preinstalled. With an instance profile, first boot attempts to start export; without one, local auditing continues. To prepare export, create the following log group in your instance Region using the AWS console and choose a retention period appropriate for your organization (for example, 30 days).

```
/codecreator/ai-tool-control/audit
```

IAM permission	Resource scope
logs:DescribeLogGroups	*
logs:DescribeLogStreams	Audit log group ARN
logs:CreateLogStream logs:PutLogEvents	Audit log stream ARN pattern

Replace REGION and ACCOUNT_ID in the ARN templates for your account (commercial AWS partition). Keep the log group pre-created so the instance role need not create arbitrary groups.

```
arn:aws:logs:REGION:ACCOUNT_ID:log-group:/codecreator/ai-tool-control/audit
arn:aws:logs:REGION:ACCOUNT_ID:log-group:/codecreator/ai-tool-control/audit:log-stream:*
```

Attach these permissions to the EC2 role. Then use the installed helper:

```
sudo codecreator-ai-tool-control-cloudwatch enable
sudo codecreator-ai-tool-control-cloudwatch status
```

Verify delivery: Invoke the readiness tool. In CloudWatch > Logs > Log groups, open the audit group and select the stream named for your EC2 instance ID. Confirm fresh records are present. An active agent alone does not prove ingestion. AWS logging and infrastructure charges may apply.

AWS reference: CloudWatch Logs permissions.

Operate and troubleshoot

Symptom	Action
Setup not ready	Run **sudo codecreator-ai-tool-control-status**. Wait for first boot and check the service journal if setup fails.
HTTPS fails	Confirm public IP, DNS, security group, certificate connectivity and the stored URL. If the IP changed, follow the recovery steps on page 7.
Missing token at /mcp	Expected in a normal browser. Configure a bearer token in a Streamable HTTP MCP client.
AWS access denied / unavailable	Check the attached role, IAM action and resource permissions, AWS API connectivity, and IMDSv2 hop limit for containers.
CloudWatch logs missing	Check role, log group, Region, agent status and actual log ingestion. Local logs remain available.

```
sudo journalctl -u codecreator-ai-tool-control-firstboot.service \
  --no-pager -l
sudo journalctl -u amazon-cloudwatch-agent.service --no-pager -n 50
```

Security essentials

Keep all client tokens private. Do not include them in chats, tickets, screenshots or source code. Use a separate limited client for each application and revoke unused tokens. Keep internal services unexposed and protect the SSH key, audit key and local log files.

If the browser root ever displays an actual token instead of SSH retrieval instructions, restrict access immediately and investigate. The Ready page should not reveal credentials. Do not use the readiness indicator alone as proof that IAM-backed features are operational.

Additional readiness-only clients

For safe integration testing without AWS tool access, create a separate restricted client:

```
sudo codecreator-ai-tool-control-admin create mcp-client \
  readiness-test --allow "cc_readiness"
```

Check your deployment

A production handoff should verify certificate-validated HTTPS from your client network, the readiness call, expected tool scopes, local audit records and actual CloudWatch ingestion if export is enabled. Use EC2 Reboot for a normal reboot, not Stop/Start, to avoid an unexpected public address change.

> **Acceptance scope.** Fresh launch, readiness, scoped access, local auditing and reboot persistence passed; four live AWS discovery operations and CloudWatch ingestion passed in build validation. Live bucket-specific S3 object listing remains unverified. Test the IAM-dependent tools and log delivery in your own environment.

Use the gateway with a supported MCP-compatible application. It does not host an AI model, provide a graphical management console, or include an AgentCore integration or multi-tenant workspace system.

Recover HTTPS after a public IP change

The generated nip.io hostname does not automatically refresh after a changed public IP. Use a stable public IP whenever possible. If a change is unavoidable, an administrator can recover the endpoint without resetting credentials or rerunning first boot.

1. **Identify the old and new public addresses.** Run the URL helper to see the stored endpoint, and compare it with the current public IPv4 shown in the EC2 console.

```
sudo codecreator-ai-tool-control-url
```

2. **Back up and update these three files.** Use a private root-only backup directory, then edit the files via SSH. Replace only the old nip.io hostname and endpoint-related old IP values with the new address; preserve all tokens, other settings, and the audit route.

```
BACKUP=$(sudo mktemp -d /root/mcp-endpoint.XXXXXX)
sudo cp -a --parents \
  /opt/codecreator-ai-tool-control/Caddyfile \
  /var/lib/codecreator-ai-tool-control/runtime.env \
  /home/ubuntu/FIRST_LOGIN.txt "$BACKUP" && \
  sudoedit /opt/codecreator-ai-tool-control/Caddyfile \
  /var/lib/codecreator-ai-tool-control/runtime.env \
  /home/ubuntu/FIRST_LOGIN.txt
```

3. **Validate and restart only Caddy.** Confirm the URL helper shows the new address and the configuration still routes /mcp to audit-proxy:8081. Run validation first; restart only if it succeeds.

```
sudo docker exec cc-ai-tool-control-caddy caddy validate \
  --config /etc/caddy/Caddyfile --adapter caddyfile && \
  sudo docker restart cc-ai-tool-control-caddy
```

Open the new HTTPS root in your browser (certificate issuance may take time). Require valid TLS, then update MCP clients and run the readiness tool. If validation fails, restore all three files from the private backup before restarting Caddy. Never bypass TLS verification or expose token files.

Rollback if validation fails

In the same SSH session, restore the saved files from the BACKUP directory. If reconnecting, first set BACKUP to the exact private backup directory created above. Restore only if the endpoint edit failed; validate the restored Caddyfile before restarting.

```
sudo cp -a "$BACKUP/opt/codecreator-ai-tool-control/Caddyfile" \
  /opt/codecreator-ai-tool-control/Caddyfile
sudo cp -a "$BACKUP/var/lib/codecreator-ai-tool-control/runtime.env" \
  /var/lib/codecreator-ai-tool-control/runtime.env
sudo cp -a "$BACKUP/home/ubuntu/FIRST_LOGIN.txt" \
  /home/ubuntu/FIRST_LOGIN.txt
```

> **Do not reset the instance.** The endpoint correction is an administrative operation, not a first-boot reset. Keep the same database, credentials and audit identity. Retain private backups according to your security policy and avoid including them in any shared image.