User Guide

Agent Toolkit for AWS

Agent Toolkit for AWS: User Guide

Table of Contents

What is the Agent Toolkit for AWS? 1
Components 1
What can I do with the Agent Toolkit for AWS? 2
How the Agent Toolkit for AWS works 3
Pricing 3
Getting started 4
Prerequisites 4
Step 1: Install 4
Step 2: Verify your connection 5
Step 3: Try it out 6
Additional plugins 6
Agent Toolkit for AWS components 7
AWS MCP Server 7
Setting up the AWS MCP Server 7
Understanding the MCP Server tools 20
Skills 21
What is a skill? 21
How agents discover and use skills 22
Types of skills 22
Available skills 23
Using skills with and without the AWS MCP Server 23
Plugins 24
Available plugins 24
Installing plugins 24
What a plugin contains 25
Updating plugins 25
Rules files 25
Recommended AWS rules file 26
Where to put the rules file 26
Customization 26
Security 27
Data protection 27
Encryption 29
File operations 30

Internetwork traffic privacy ... 31
Identity and access management ... 31
Audience ... 31
Authenticating with identities ... 32
Managing access using policies ... 33
How AWS MCP Server works with IAM ... 35
Identity-based policy examples ... 37
Troubleshooting ... 39
Compliance validation ... 40
Resilience ... 40
Monitoring ... 41
CloudWatch metrics ... 41
Metrics namespace ... 41
Available metrics ... 41
Metric dimensions ... 42
Using metrics ... 42
Example use cases ... 43
Usage metrics ... 43
CloudTrail logs ... 44
AWS MCP Server information in CloudTrail ... 45
Understanding AWS MCP Server log file entries ... 46
Quotas ... 48
Connection and session quotas ... 48
Throttling quotas ... 49
Session limits ... 49
Monitoring your quotas ... 49
Document history ... 51

What is the Agent Toolkit for AWS?

The Agent Toolkit for AWS gives AI coding agents the tools, knowledge, and guardrails they need to build, deploy, and manage applications on AWS. It works with the coding agents that developers already use — including Claude Code, Cursor, and Codex — without requiring you to switch tools or learn a new workflow.

AI coding agents can handle common AWS tasks like creating an Amazon S3 bucket or launching an Amazon EC2 instance, but they often struggle with complex, multi-step workflows. They choose the wrong service for a use case, misconfigure resources, or repeatedly retry operations on newer services they were not trained on. The Agent Toolkit for AWS addresses this by giving agents secure access to any AWS service, up-to-date documentation, and step-by-step guidance for the tasks where they need it most.

Topics

- Components
- What can I do with the Agent Toolkit for AWS?
- How the Agent Toolkit for AWS works
- Pricing

Components

The Agent Toolkit for AWS includes four components that work together:

- **AWS MCP Server** — A managed server that gives agents access to AWS through the Model Context Protocol (MCP). Agents can search AWS documentation and retrieve service information without authentication. To execute AWS API calls, run Python scripts in a sandboxed environment, or follow curated skills, agents authenticate through your existing IAM credentials. All of these capabilities are available through a single endpoint with CloudWatch metrics and IAM-based access controls.
- **Agent skills** — Curated packages of instructions, code scripts, and reference materials that help agents complete specific AWS tasks. Skills are loaded on demand — agents discover and retrieve only the skills relevant to the current task, so they do not consume unnecessary context. Skills cover service selection, step-by-step procedures, troubleshooting, and SDK best practices.

- **Plugins** — Single-install packages for Claude Code and Codex that bundle the AWS MCP Server configuration and a curated set of agent skills. After you install a plugin, your agent is ready to work with AWS.
- **Rules files** — Project-level configuration files that set guardrails and preferences for how agents work in your project. Rules files tell agents how to use AWS most effectively — for example, by using the AWS MCP Server, discovering available skills, or searching documentation before acting.

What can I do with the Agent Toolkit for AWS?

With the Agent Toolkit for AWS, your AI coding agent can:

- **Build and deploy applications on AWS** — Your agent creates AWS infrastructure, writes application code, and deploys to production. Skills from the toolkit guide your agent through choosing the right services, configuring them correctly, and following deployment best practices.
- **Stay up to date on the latest AWS services** — The AI models that power coding agents are trained on data that can be months or years old. Newer AWS services and recently launched features are often missing from an agent's knowledge. The toolkit gives your agent real-time access to current AWS documentation, API references, and service capabilities.
- **Follow tested procedures for complex workflows** — Instead of improvising from general knowledge, your agent follows tested procedures provided by the toolkit for workflows like configuring least-privilege IAM policies, setting up data pipelines, or deploying production-ready serverless applications.
- **Troubleshoot operational issues** — Point your agent at a failing deployment, a spike in error rates, or an unexpected cost increase. The toolkit provides skills that help your agent work with CloudWatch logs and metrics, CloudFormation stack status, and troubleshooting procedures.
- **Operate with security and visibility** — The AWS MCP Server provides CloudWatch metrics for monitoring agent activity, IAM-based access controls, and the ability to set enterprise guardrails — like restricting agents to read-only operations or blocking specific actions through MCP.
- **Work with any MCP-compatible agent** — The Agent Toolkit for AWS works with Claude Code, Cursor, Codex, Kiro, Windsurf, Cline, and any other agent that supports the Model Context Protocol.

How the Agent Toolkit for AWS works

The Agent Toolkit for AWS helps your AI coding agent build on AWS in three ways:

- **Skills provide structured guidance** — The agent uses skills that are installed locally via a plugin, or discovers them at runtime through the AWS MCP Server. Skills contain step-by-step instructions, decision guides, and reference materials that guide the agent through complex procedures while following AWS best practices.
- **Knowledge tools provide current information** — When the agent needs up-to-date information, it can search AWS documentation, retrieve API references, check regional availability, and access the latest AWS service information.
- **API tools execute authenticated actions** — The AWS MCP Server translates requests into properly formatted AWS API calls, handles authentication using your IAM credentials, and executes the commands with detailed feedback about results and any errors. For complex multi-step operations, agents can write and execute Python scripts in an isolated sandbox using the `run_script` tool.

The AWS MCP Server automatically adds two global condition context keys (`aws:ViaAWSMCPService` and `aws:CalledViaAWSMCP`) to all requests. These keys let you differentiate MCP-initiated actions from direct API calls in your IAM policies. CloudTrail logs all API calls for audit visibility.

Authentication and authorization use your existing AWS IAM roles and policies, so you maintain full control over what resources and actions are available.

> **Note**
>
> We recommend scoping down IAM roles to the minimum permissions that the agent needs to perform its task.

Pricing

You can use the Agent Toolkit for AWS at no additional charge. You pay only for the AWS resources your agent provisions or interacts with, at standard AWS pricing. For more information about AWS pricing, see AWS Pricing. If you are new to AWS, you can get started with many services for free. For more information, see AWS Free Tier.

Getting started

The fastest way to get started with the Agent Toolkit for AWS is to install the plugin for your AI coding agent. The plugin bundles the AWS MCP Server configuration and a curated set of agent skills in a single install.

Prerequisites

- uv installed on your system (required for the MCP proxy).
- (Optional) An AWS account with IAM credentials set up on your local machine. Credentials are required for tools that execute AWS API calls and run scripts, but not for searching documentation or discovering skills. If you do not have credentials configured, see the section called "Setting up the AWS MCP Server" for detailed instructions.

Step 1: Install

Claude Code

In Claude Code:

```
/plugin marketplace add aws/agent-toolkit-for-aws
/plugin install aws-core@agent-toolkit-for-aws
/reload-plugins
```

Codex

In your terminal:

```
codex plugin marketplace add aws/agent-toolkit-for-aws
```

Then launch Codex and run `/plugins` to browse and install the **aws-core** plugin.

Kiro

Add the following to your MCP configuration file (for example, `~/.kiro/settings/mcp.json`):

```
{
  "mcpServers": {
    "aws-mcp": {
      "command": "uvx",
      "timeout": 100000,
      "transport": "stdio",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://aws-mcp.us-east-1.api.aws/mcp",
        "--metadata", "AWS_REGION=us-west-2"
      ]
    }
  }
}
```

Then install skills:

```
npx skills add aws/agent-toolkit-for-aws/skills
```

Other agents

If your agent supports MCP, you can configure the AWS MCP Server directly. See the section called "Setting up the AWS MCP Server" for instructions.

Then install skills:

```
npx skills add aws/agent-toolkit-for-aws/skills
```

Step 2: Verify your connection

After you install the plugin, verify that the AWS MCP Server is connected:

1. Start a new conversation with your agent.
2. Ask: *"What AWS Regions are available?"*

If the agent returns a list of AWS Regions, the connection is working. If you see an authentication error, see the section called "Troubleshooting authentication errors".

Step 3: Try it out

Ask your agent to perform an AWS task:

- "What AWS services should I use to build a serverless API?"
- "Create an Amazon S3 bucket with versioning enabled and a lifecycle policy that transitions objects to Glacier after 90 days."
- "Help me troubleshoot why my CloudFormation deployment failed."

The agent discovers and uses relevant skills automatically. You do not need to know which skills are available — the agent finds them based on your request.

Additional plugins

After you install aws-core, you can install additional plugins for specialized workflows:

- **aws-agents** — Skills for building AI agents on AWS with API Gateway and AgentCore.
- **aws-data-analytics** — Skills for data lake, analytics, and ETL workflows.

Install additional plugins using the same method as aws-core.

Agent Toolkit for AWS components

The Agent Toolkit for AWS includes four components that work together to help AI coding agents build, deploy, and manage applications on AWS.

- **the section called "AWS MCP Server"** — A managed server that gives agents access to AWS through the Model Context Protocol.
- **the section called "Skills"** — Curated packages of instructions and reference materials that help agents complete specific AWS tasks.
- **the section called "Plugins"** — Single-install packages that bundle the AWS MCP Server configuration and agent skills.
- **the section called "Rules files"** — Project-level configuration files that tell agents how to use AWS most effectively.

AWS MCP Server

The AWS MCP Server is a managed server that gives agents access to AWS through the Model Context Protocol (MCP). Agents can search AWS documentation and retrieve service information without authentication. To execute AWS API calls, run Python scripts in a sandboxed environment, or follow curated skills, agents authenticate through your existing IAM credentials.

All capabilities are available through a single endpoint with CloudWatch metrics and IAM-based access controls. CloudTrail logs all API calls for audit visibility.

Setting up the AWS MCP Server

This section outlines how you can set up your AWS MCP Server.

Topics

- Prerequisites
- Set up the AWS MCP Server
- Troubleshooting authentication errors

Prerequisites

Before you begin, you must ensure that you have set up an AWS account.

Sign up for an AWS account

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.
2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.
2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Set up the AWS MCP Server

To set up AWS MCP Server, use the steps in the following sections.

Topics

- Step 1: (If applicable) Remove conflicting MCP servers
- Step 2: Configure AWS credentials
- Step 3: Configure your MCP client

- Step 4: Understand IAM authorization
- Step 5: Test your connection

Step 1: (If applicable) Remove conflicting MCP servers

If you currently have AWS API MCP Server or AWS Knowledge MCP Server installed, we recommend removing them before setting up the AWS MCP Server to avoid tool conflicts that can confuse AI agents and reduce performance.

To remove existing AWS MCP servers:

1. Open your MCP client configuration file.
2. Remove any entries for these servers:
   - `aws-api-mcp-server`
   - `aws-knowledge-mcp-server`
3. Save the configuration file.
4. Restart your MCP client to apply the changes.

Step 2: Configure AWS credentials

Before connecting to AWS MCP Server, you need to configure AWS credentials on your local machine. The server uses these credentials to authenticate your requests.

You can use the SigV4 via Proxy to authenticate the AWS MCP Server. SigV4 via Proxy uses your available AWS credentials and requires the MCP Proxy for AWS. The MCP Proxy for AWS handles SigV4 signing and credential management between your MCP client and the AWS MCP Server. No separate installation is required — the `uvx mcp-proxy-for-aws@latest` command in your MCP client configuration automatically downloads and runs the latest version of the proxy each time your MCP client starts. For the full list of configuration options, see the MCP Proxy for AWS repository on GitHub.

For detailed information about configuring AWS credentials, see Sign in with the aws login command in the *AWS CLI User Guide*.

Important

AWS MCP Server requires valid AWS credentials for the duration of your development session. We strongly recommend using a credential method that supports automatic renewal, such as `aws login` or `aws configure sso`. Expired credentials would prevent the MCP server from being initialized, making all AWS tools unavailable until you restart your MCP client with valid AWS credentials.

Note

If your authentication step worked previously but you have encountered an authentication error, you might need to refresh your credentials and try again. See the section called "Troubleshooting authentication errors" for common authentication errors and how to resolve them.

1. Install the AWS CLI by following the instructions at Installing the AWS CLI.
2. Configure your AWS credentials using one of these methods:

For users with AWS Management Console credentials (Recommended)

From the AWS CLI, run the following command:

```
aws login
```

This is the recommended method because the AWS SDK automatically rotates your credentials every 15 minutes within the session, eliminating credential expiry during normal use. Your session remains valid for up to 12 hours without any manual intervention.

Note

AWS Management Console credentials means that you have a username and password that allows you to sign in to the console. To use this method, you need the AWS CLI version `2.32.0` or later.

For SSO users (Recommended)

```
aws configure sso
```

Follow the prompts to set up your SSO configuration. SSO credentials support automatic renewal through cached refresh tokens. The SDK silently obtains new access tokens and role session credentials without requiring you to re-authenticate.

> **Tip**
>
> For administrators: The default SSO session duration is 1 hour. You can extend this up to 12 hours in your IAM Identity Center settings to reduce the frequency of re-authentication for your team.

For IAM users

```
aws configure
```

Enter your Access Key ID, Secret Access Key, default region, and output format.

> **Warning**
>
> Static IAM access keys do not support automatic credential renewal and are not recommended for use with AWS MCP Server. If you must use IAM access keys, be aware that AWS recommends rotating them every 90 days. Where possible, use `aws login` or `aws configure sso` instead.

For cross-account access (assume-role)

If you need to access AWS resources in a different account, you can configure an assume-role profile in your `~/.aws/config` file. The AWS SDK handles credential refresh transparently for this method.

```
[profile my-role]
role_arn = arn:aws:iam::123456789012:role/MyRole
source_profile = default
region = us-east-1
```

Then set the `AWS_PROFILE` environment variable or pass the profile name in your MCP client configuration.

3. Test your configuration:

```
aws sts get-caller-identity
```

4. Install uv (if not already installed)

On macOS and Linux

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Windows

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

Step 3: Configure your MCP client

AWS MCP Server is available in the following AWS Regions:

- US East (N. Virginia) – `us-east-1`: `https://aws-mcp.us-east-1.api.aws/mcp`
- Europe (Frankfurt) – `eu-central-1`: `https://aws-mcp.eu-central-1.api.aws/mcp`

Replace the endpoint URL in the configuration examples below with the endpoint for your preferred Region. When configuring the SigV4 proxy, append `/mcp` to the base endpoint URL as shown in the following examples.

The endpoint Region determines which MCP server you connect to, while the `AWS_REGION` metadata parameter sets the default Region for the AWS operations the server performs on your

behalf. These can be different — for example, you can connect to the `us-east-1` endpoint while operating on resources in `us-west-2`.

Set your default AWS Region by adding the `--metadata` parameter with `AWS_REGION`. Without this setting, all AWS operations default to `us-east-1`.

> **Note**
>
> Replace `us-west-2` with your preferred default AWS Region.

Region behavior:

- Without `--metadata` and `AWS_REGION`: Operations default to `us-east-1`
- With `--metadata` and `AWS_REGION`: Operations use your specified Region
- In queries: You can override by specifying a Region (example: "list my EC2 instances in eu-west-1")

Kiro CLI

```
{
  "mcpServers": {
    "aws-mcp": {
      "command": "uvx",
      "timeout": 100000,
      "transport": "stdio",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://aws-mcp.us-east-1.api.aws/mcp",
        "--metadata", "AWS_REGION=us-west-2"
      ]
    }
  }
}
```

Kiro IDE

```
{
  "mcpServers": {
```

```
    "aws-mcp": {
      "command": "uvx",
      "timeout": 100000,
      "transport": "stdio",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://aws-mcp.us-east-1.api.aws/mcp",
        "--metadata", "AWS_REGION=us-west-2"
      ]
    }
  }
}
```

Cursor IDE

```
{
  "mcpServers": {
    "aws-mcp": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://aws-mcp.us-east-1.api.aws/mcp",
        "--metadata", "AWS_REGION=us-west-2"
      ]
    }
  }
}
```

Claude Desktop

```
{
  "mcpServers": {
    "aws-mcp": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://aws-mcp.us-east-1.api.aws/mcp",
        "--metadata", "AWS_REGION=us-west-2"
      ]
    }
  }
}
```

Codex

```
[mcp_servers.aws_mcp]
command = "uvx"
args = [
  "mcp-proxy-for-aws@latest",
  "https://aws-mcp.us-east-1.api.aws/mcp",
  "--metadata", "AWS_REGION=us-west-2"
]
startup_timeout_sec = 60
```

Step 4: Understand IAM authorization

AWS MCP Server does not require separate IAM permissions to invoke the MCP server. Authorization occurs at the AWS service level using your existing IAM roles and policies. When an AI agent calls the AWS MCP Server, the server authenticates your request and forwards it to the appropriate AWS service. The AWS service then authorizes the request using your existing IAM permissions, just as it would for any direct API call.

Two global condition context keys are automatically added to all requests made through the AWS MCP Server:

- `aws:ViaAWSMCPService` – Set to `true` for any request that passes through an AWS managed MCP server.
- `aws:CalledViaAWSMCP` – Contains the service principal of the specific AWS managed MCP server (for example, `aws-mcp.amazonaws.com`).

You can use these context keys in your IAM policies to allow or deny actions initiated through any AWS managed MCP server. For more information about these context keys, see IAM condition context keys in the *IAM User Guide*.

The following examples show how to restrict access using these context keys.

Deny all operations via any AWS managed MCP server

```
{
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
```

```
    "Condition": {
        "Bool": {
            "aws:ViaAWSMCPService": "true"
        }
    }
}
```

Deny specific operations via a specific AWS managed MCP server

```
{
    "Effect": "Deny",
    "Action": ["s3:DeleteBucket", "s3:DeleteObject"],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaAWSMCP": "aws-mcp.amazonaws.com"
        }
    }
}
```

> **Note**
>
> If you previously configured IAM permissions using `aws-mcp:InvokeMcp`, `aws-mcp:CallReadOnlyTool`, or `aws-mcp:CallReadWriteTool`, we recommend that you remove these permissions from your policies as they no longer have any effect. If you used these permissions to deny access to AWS MCP Server, you must update your policies to use the context keys shown above.

Step 5: Test your connection

1. Start your MCP client (Kiro CLI, Cursor, Claude Desktop, etc.).
2. Wait for the MCP server to initialize (this may take a few minutes on first connection).
3. Test the connection by asking your AI assistant:

   Example: What AWS Regions are available?
4. Verify that tools are loaded by running (in Kiro CLI):

   `/tools`

Or to see installed MCP servers:

```
/mcp
```

You should see tools like `aws___search_documentation` and `aws___retrieve_skill` listed. For more information about the tools, see Understanding the MCP Server tools.

Troubleshooting authentication errors

Authentication errors can prevent the MCP server from initializing, which results in AWS MCP tools not being available to AI agents. If your AI agent is not using AWS MCP tools, an expired or missing credential is the most likely cause.

Use the following table to identify and resolve common authentication errors.

Common authentication errors

Error	Cause	Resolution
`ExpiredTokenException`: Your AWS session token has expired.	Your temporary AWS credentials have expired. This is the most common authentication error, typically caused by short-lived session tokens (default 1 hour) expiring during development.	Refresh your credentials based on your authentication method: • `aws login` users: Run `aws login` again to start a new session. Consider switching to this method if you experience frequent expiry, as it auto-rotates credentials every 15 minutes. • SSO users: Run `aws sso login --profile your-profile-name` to refresh your SSO session. • Assume-role users: Refresh the source profile credentials, then the SDK will automatically re-assume the role. After refreshing, restart your MCP client to re-initialize the server.

Error	Cause	Resolution
`UnrecognizedClientException` : The security token included in the request is not recognized.	Your credentials are invalid. This can happen when credentials have been revoked, are from a different AWS partition, are malformed, or belong to a deleted IAM user or role.	Verify your credentials are valid: 1. Run `aws sts get-caller-identity` to check if your credentials are recognized. 2. If the command fails, reconfigure your credentials using `aws login` or `aws configure sso`. 3. Ensure you are using credentials for the correct AWS partition (for example, `aws` vs. `aws-cn`).
`InvalidSignatureException` : The request signature we calculated does not match the signature you provided.	The SigV4 signature does not match. Common causes include credentials scoped to the wrong service or Region, clock skew on your machine, or a request body that was modified after signing.	Try the following steps: 1. Verify your system clock is accurate. Run `date` and compare with an authoritative time source. SigV4 requires your clock to be within 5 minutes of AWS servers. 2. Ensure your credentials are configured for the correct AWS Region and service. 3. Reconfigure your credentials and restart your MCP client.
No AWS credentials found.	AWS credentials are not configured on your machine, or the credential provider chain cannot locate them.	Configure your credentials by following the section called “Step 2: Configure AWS credentials”. We recommend using `aws login` for the simplest setup with automatic credential renewal.

Tip

To avoid credential expiry issues, use `aws login` (preferred) or `aws configure sso`. Both methods support automatic credential renewal, which prevents the most common authentication failures. For more information, see the section called “Step 2: Configure AWS credentials”.

Understanding the MCP Server tools

AWS MCP Server provides the following tools to help you complete AWS tasks through natural language interactions.

AWS Knowledge Tools

- `aws___retrieve_skill` - Retrieve domain-specific expertise for a particular AWS domain. Skills provide workflows, context, best practices, decision frameworks, and step-by-step procedures. When called with a skill name, returns the full skill content including reference materials. Use `aws___search_documentation` to discover available skills.
- `aws___search_documentation` - Search across all AWS documentation, including API references, best practices, service guides, and skills (formerly Agent SOPs). Use the topic filter to search skills exclusively, or see skills alongside general knowledge search results. Find relevant information from multiple AWS knowledge sources.
- `aws___read_documentation` - Retrieve and convert AWS documentation pages to markdown format for easy consumption by AI assistants.
- `aws___recommend` - Get content recommendations for AWS documentation pages based on related topics and commonly viewed content.
- `aws___list_regions` - Retrieve a list of all AWS regions, including their identifiers and names.
- `aws___get_regional_availability` - Check AWS regional availability information for services, features, SDK APIs, and CloudFormation resources.

AWS API Tools

- `aws___call_aws` - Execute authenticated AWS API calls with proper syntax validation and error handling. Supports most of the 15,000+ AWS APIs with automatic credential management.

- `aws___suggest_aws_commands` - Get descriptions and syntax help for relevant AWS APIs, including newly released APIs that may not be in the AI model's training data.
- `aws___run_script` - Execute Python code in a sandboxed environment with AWS API access. Use for tasks that involve listing resources and checking their properties, parallel API calls, multi-step workflows, cross-service checks, and retry logic.
- `aws___get_presigned_url` - Generate pre-signed Amazon S3 URLs for uploading or downloading files. Use this tool for direct Amazon S3 uploads and downloads, or when an AWS CLI command requires a local file path.
- `aws___get_tasks` - Poll the status of long-running tasks started by `aws___call_aws` or `aws___run_script`. Use when a previous tool call returns a task ID with a working status.

These tools work together to provide comprehensive AWS task completion: skills guide the workflow, knowledge tools provide current information and best practices, and API tools execute the actual AWS operations with proper authentication and authorization.

Skills

Agent skills are curated packages of instructions, code scripts, and reference materials that help AI coding agents complete specific AWS tasks. Skills bridge the gap between what AI models know from their training data and what is needed to work effectively with AWS — especially for newer services, complex multi-service workflows, and tasks where best practices matter.

Topics

- What is a skill?
- How agents discover and use skills
- Types of skills
- Available skills
- Using skills with and without the AWS MCP Server

What is a skill?

A skill is a directory containing a `SKILL.md` file and optional supporting files. The `SKILL.md` file includes a brief description and a set of instructions that tell the agent how to complete a task — which steps to follow, which AWS APIs to call, which mistakes to avoid, and how to verify the result.

Skills can also include reference files with deeper guidance on specific subtasks, code scripts for deterministic operations, and slash commands that let you invoke the skill directly.

Skills are designed to be lightweight and modular. Each skill focuses on a single task or domain, and agents load only the skills they need for the current task. A skill typically consumes a few thousand tokens when loaded — far less than the equivalent documentation — because it contains only the information the agent needs to act, not background context the agent already has.

How agents discover and use skills

There are three ways agents get access to skills:

- **Bundled with a plugin** — Each plugin includes a curated set of skills that are available to the agent immediately after installation. The agent can use these skills without any network calls or additional setup.
- **Installed locally** — You can download individual skills from the Agent Toolkit for AWS repository on GitHub and add them to your agent's skills directory.
- **Discovered at runtime via the AWS MCP Server** — Agents can search for and retrieve skills on demand through the AWS MCP Server, without any local installation. The agent uses the `search_documentation` tool to find relevant skills and the `retrieve_skill` tool to load them into context.

When an agent uses a skill — regardless of how it was installed — the process is the same:

1. The agent reads the skill's description to determine if it is relevant to the current task.
2. If relevant, the agent loads the full instructions from `SKILL.md`.
3. The agent follows the skill's procedures, loading reference files only as needed.
4. After the task is complete, the agent releases the skill content from context.

This progressive disclosure approach means that having many skills available does not slow your agent down or consume unnecessary context. Only the skills relevant to the current task are loaded.

Types of skills

The Agent Toolkit for AWS includes several types of skills:

- **Service decision guides** help agents choose the right AWS service for a use case. For example, a database decision guide helps agents recommend DynamoDB, Aurora, or DSQL based on the workload requirements.
- **Step-by-step procedures** provide tested workflows for common tasks like creating S3 Tables, setting up AWS Glue ETL pipelines, configuring IAM policies, and deploying serverless applications.
- **Troubleshooting guides** provide diagnostic procedures for common errors, with steps to identify the cause and resolve the issue. For example, a CloudFormation deployment troubleshooting skill covers the top failure patterns and how to fix them.
- **SDK usage guides** provide language-specific best practices for the AWS SDKs, covering common mistakes that models consistently get wrong (like DynamoDB marshalling in JavaScript or pagination patterns in Python).

Available skills

Each plugin includes a curated set of skills covering the most common workflows for that domain. The full set of skills — including domain-specific skills for individual AWS services — is available on GitHub and discoverable at runtime through the AWS MCP Server.

To see what skills are available from within your agent, ask: *"What AWS skills do you have available?"* or *"Search for AWS skills related to databases."*

You can also browse and install skills from the command line:

```
npx skills add aws/agent-toolkit-for-aws/skills
```

Using skills with and without the AWS MCP Server

Skills work best with the AWS MCP Server, which provides authenticated API access, sandboxed script execution, and enterprise controls like CloudWatch metrics and IAM context keys. For production workflows, use the AWS MCP Server.

Skills also work without the AWS MCP Server. When agents do not have access to the MCP Server, they can run the same AWS operations using the AWS CLI directly. Each skill includes instructions that work with both approaches.

Plugins

Plugins for the Agent Toolkit for AWS are single-install packages that bundle the AWS MCP Server configuration and a curated set of agent skills. Instead of manually configuring MCP endpoints and installing skills individually, you install one plugin and your agent is ready to work with AWS.

Plugins are available for Claude Code and Codex. They are open source and published on GitHub at aws/agent-toolkit-for-aws.

Topics

- Available plugins
- Installing plugins
- What a plugin contains
- Updating plugins

Available plugins

- **aws-core** — The primary plugin for the Agent Toolkit for AWS. Includes the AWS MCP Server configuration and default skills covering the most common AWS workflows: service selection, infrastructure as code (CDK and CloudFormation), serverless, containers, storage, observability, billing and cost management, SDK usage, and deployment. This plugin is recommended for all AWS developers.
- **aws-agents** — Skills for building AI agents on AWS using API Gateway, Bedrock AgentCore, and related services.
- **aws-data-analytics** — Skills for data lake, analytics, and ETL workflows using Amazon Simple Storage Service Tables, AWS Glue, Amazon Athena, and related services.

Installing plugins

Claude Code

In Claude Code:

```
/plugin marketplace add aws/agent-toolkit-for-aws
/plugin install aws-core@agent-toolkit-for-aws
/reload-plugins
```

Codex

In your terminal:

```
codex plugin marketplace add aws/agent-toolkit-for-aws
```

Then launch Codex and run `/plugins` to browse and install the **aws-core** plugin.

Agents that do not support plugins

If your agent does not support plugins, you can configure the AWS MCP Server directly. See the section called “Setting up the AWS MCP Server” for instructions.

What a plugin contains

Each plugin contains the following:

- **Plugin manifest** (`.claude-plugin/plugin.json` or `.codex-plugin/plugin.json`) — The plugin metadata including name, description, and version.
- **Skills directory** (`skills/`) — Agent skills included in the plugin.
- **MCP configuration** (`.mcp.json`) — AWS MCP Server endpoint configuration.

Updating plugins

Plugins update automatically when your agent client refreshes its marketplace data. You can also update manually by running the update command in your agent client.

Rules files

Most AI coding agents support project-level configuration files — often called rules files — that provide persistent instructions the agent follows in every session. For example, you might have a rules file that tells your agent to always write TypeScript in strict mode, or to use a specific testing framework.

The Agent Toolkit for AWS includes a recommended rules file that tells your agent how to work with AWS — for example, to use the AWS MCP Server for API calls, to search for available skills before starting a task, and to prefer infrastructure-as-code over direct CLI commands.

Topics

- Recommended AWS rules file
- Where to put the rules file
- Customization

Recommended AWS rules file

The Agent Toolkit for AWS includes a recommended rules file that covers using the AWS MCP Server, discovering skills, verifying against documentation, and following infrastructure-as-code best practices. You can find the latest version of this file in the Agent Toolkit for AWS repository on GitHub. Copy the content into the appropriate file for your agent.

Where to put the rules file

The file name and location depend on your agent:

Rules file locations by agent

Agent	Project rules	Location
Claude Code	`CLAUDE.md`	Project root
Codex	`AGENTS.md`	Project root
Cursor	`.cursor/rules/*.mdc`	`.cursor/rules/` directory
Kiro	`.kiro/steering/*.md`	`.kiro/steering/` directory

For Claude Code and Codex, add the content above to your existing `CLAUDE.md` or `AGENTS.md` file, or create a new one in your project root. For Cursor, create a new `.mdc` file in the `.cursor/rules/` directory (for example, `.cursor/rules/aws.mdc`). The legacy `.cursorrules` file is still supported but deprecated.

Customization

The recommended rules file is a starting point. Customize it for your project — for example, specifying which AWS Region to use, which VPC to deploy into, or which naming conventions to follow for resources.

Security in AWS MCP Server

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS MCP Server, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS MCP Server. The following topics show you how to configure AWS MCP Server to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS MCP Server resources.

Topics

- Data protection in AWS MCP Server
- Identity and access management for AWS MCP Server
- Compliance validation for AWS MCP Server
- Resilience in AWS MCP Server

Data protection in AWS MCP Server

The AWS shared responsibility model applies to data protection in Agent Toolkit for AWS. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks

for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Agent Toolkit for AWS or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

> **Note**
>
> The AWS MCP Server doesn't support FIPS endpoints.

Encryption

AWS MCP Server is a stateless proxy that executes AWS API calls on your behalf. The service does not use any persistent storage services such as Amazon S3, DynamoDB, Amazon EBS, or Amazon SQS to store customer content. When file processing is required (for example, deploying an application package), customer content exists only transiently on ephemeral compute storage during active processing.

Encryption at rest

When AWS MCP Server temporarily processes customer files, the data resides only on ephemeral compute storage (Lambda `/tmp` storage and AgentCore Runtime ephemeral storage). This ephemeral storage is:

- Automatically encrypted at rest by the compute platform using AWS managed keys.
- Automatically destroyed when ephemeral storage is reclaimed (maximum ephemeral storage retention of 8 hours). Most operations complete in seconds.
- Not accessible through any external API or operator tooling.
- Fully isolated per customer through hardware-backed tenant isolation. For more information, see Security overview for Lambda in the *Lambda Developer Guide*.

AWS MCP Server does not currently offer customer managed KMS keys for encrypting ephemeral compute storage. While Lambda supports customer managed keys for `/tmp` storage, the service uses AWS managed encryption across all compute platforms for consistency and operational simplicity. However, your source data in Amazon S3 remains protected by whatever encryption you have configured on your buckets, including customer managed keys. If your Amazon S3 objects are encrypted with a customer managed key, AWS MCP Server requires `kms:Decrypt` permission (provided through your credentials) to access those objects. Revoking this permission prevents the service from accessing your content.

Encryption in transit

All communication between your MCP client and AWS MCP Server is encrypted using TLS. All downstream AWS API calls made by the service on your behalf also use TLS encryption.

Key management

AWS MCP Server uses AWS managed keys to encrypt ephemeral compute storage. You do not need to manage or configure these keys.

For your source data stored in AWS services such as Amazon S3, you retain full control over your encryption keys. AWS MCP Server accesses your data using credentials derived from your own IAM session. These credentials are scoped to your authenticated session and respect all AWS KMS key policies you have configured.

File operations

AWS MCP Server runs in a cloud environment that cannot access your local file system directly. When an operation requires a file (for example, deploying a package or uploading an object to Amazon S3), the service uses pre-signed Amazon S3 URLs to stage files through your own Amazon S3 bucket.

The file operation workflow is:

1. The service generates a pre-signed Amazon S3 URL pointing to your Amazon S3 bucket.
2. You upload your file to your own bucket using the pre-signed URL.
3. The service references the staged file when calling the target AWS service.

When the target AWS service accepts Amazon S3 locations natively, the service passes the Amazon S3 reference directly and your file is never downloaded to the server. When the target service requires file content directly, the service temporarily downloads the file to ephemeral compute storage, executes the operation, and immediately deletes the temporary copy.

Billing considerations

Because file staging uses your own Amazon S3 bucket, standard Amazon S3 charges apply to your account:

- Amazon S3 PUT and GET request charges for each file upload and download.
- Amazon S3 storage charges for staged files that remain in your bucket.

The AWS MCP Server service itself does not incur additional storage charges for file operations. For current pricing, see Amazon S3 pricing.

Limits

File staging has the following limits:

- Pre-signed URLs expire after a maximum of 15 minutes (900 seconds).
- The maximum size for a single staged file that the service downloads for processing is 125 MB.
- The maximum size for a staged zip archive (used for directory uploads such as `deploy push --source` or `gamelift upload-build --build-root`) is 1 GB.
- The maximum total size across all staged files on the runtime is 4 GB. This budget is shared across all concurrent operations.

Internetwork traffic privacy

AWS MCP Server communicates with AWS services over the AWS network using TLS-encrypted connections. Your requests to the AWS MCP Server endpoint are encrypted in transit using TLS 1.2 or later.

Identity and access management for AWS MCP Server

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Agent Toolkit for AWS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS MCP Server works with IAM
- Identity-based policy examples for AWS MCP Server
- Troubleshooting AWS MCP Server identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS MCP Server identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS MCP Server works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS MCP Server)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS MCP Server works with IAM

AWS MCP Server uses a simplified authorization model that works like the AWS Command Line Interface (AWS CLI) and AWS SDKs. The server does not define its own IAM actions, resources, or service-specific condition keys. Instead, it authenticates your request using SigV4, adds standardized condition context keys, and forwards the request to the downstream AWS service. The downstream service performs the authorization check using your existing IAM policies. This means your AI agents work with your existing AWS credentials and service-level permissions, and you do not need to configure separate MCP-specific IAM actions.

Authorization flow

When an AI agent calls the AWS MCP Server, the following authorization flow occurs:

1. Your agent's request is authenticated with your AWS credentials using SigV4. The MCP Proxy for AWS handles this signing automatically between your host application and the server.
2. AWS MCP Server authenticates the request and adds the MCP condition context keys (`aws:ViaAWSMCPService` and `aws:CalledViaAWSMCP`).
3. AWS MCP Server forwards the request to the target AWS service.
4. The target AWS service authorizes the request using your existing IAM policies, which can reference the MCP condition context keys for fine-grained control.

MCP condition context keys

AWS MCP Server automatically adds the following global condition context keys to all requests it forwards to downstream AWS services. You can use these keys in IAM policies and service control policies (SCPs) to differentiate between requests made through an AWS managed MCP server and direct API calls.

`aws:ViaAWSMCPService`

A Boolean key set to `true` for any request that passes through an AWS managed MCP server. Use this key to allow or deny all actions initiated through any AWS managed MCP server.

Type: Boolean

`aws:CalledViaAWSMCP`

A single-valued string key containing the service principal of the specific AWS managed MCP server that initiated the request. Use this key to apply controls for a specific MCP server.

Type: String

Example values:

- `aws-mcp.amazonaws.com` – AWS MCP Server
- `eks-mcp.amazonaws.com` – Amazon EKS MCP Server
- `ecs-mcp.amazonaws.com` – Amazon ECS MCP Server

For more information about global condition context keys, see IAM condition context keys in the *IAM User Guide*.

Identity-based policies

Supports identity-based policies: Yes

Because AWS MCP Server forwards requests to downstream AWS services using your credentials, the IAM policies attached to your IAM user or role determine what actions the MCP server can perform on your behalf. No additional IAM configuration is required to use AWS MCP Server beyond the permissions you already grant for direct API access.

You can use the MCP condition context keys in your existing policies to apply different permissions when actions are initiated through an MCP server. For examples, see Identity-based policy examples for AWS MCP Server.

Using temporary credentials with AWS MCP Server

Supports temporary credentials: Yes

AWS MCP Server works with temporary credentials obtained through AWS STS. When you authenticate with AWS MCP Server, you can use temporary credentials from IAM roles, federated identities, or assumed roles. The server forwards these credentials to downstream AWS services, which honor the same session policies and permission boundaries as direct API calls.

Deprecated MCP-specific IAM actions

During the preview period, AWS MCP Server required the following service-specific IAM actions:

- `aws-mcp:InvokeMcp`
- `aws-mcp:CallReadOnlyTool`
- `aws-mcp:CallReadWriteTool`

These actions are no longer required and have no effect. If you previously configured IAM permissions using these actions, we recommend that you remove them from your policies. If you used these actions in Deny statements to block access to AWS MCP Server, you must update your policies to use the `aws:ViaAWSMCPService` or `aws:CalledViaAWSMCP` condition context keys instead.

Identity-based policy examples for AWS MCP Server

The following examples show how to use IAM policies with the MCP condition context keys to control access through AWS MCP Server. Each example shows the policy statement to include within your IAM policy document.

Topics

- Deny all actions through any AWS managed MCP server
- Deny destructive actions through AWS MCP Server
- Restrict actions to a specific MCP server

Deny all actions through any AWS managed MCP server

The following SCP or IAM policy denies all actions when the request originates from any AWS managed MCP server. Use this to completely block MCP server access across an organization or for specific principals.

```
{
    "Sid": "DenyAllActionsViaMCP",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "Bool": {
            "aws:ViaAWSMCPService": "true"
        }
    }
}
```

Deny destructive actions through AWS MCP Server

The following policy allows read operations but denies destructive actions when the request comes through AWS MCP Server. This lets AI agents inspect resources without being able to delete them.

```
[
    {
        "Sid": "AllowS3ReadOperations",
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Resource": "*"
    },
    {
        "Sid": "DenyDeleteWhenAccessedViaMCP",
        "Effect": "Deny",
        "Action": [
            "s3:DeleteObject",
            "s3:DeleteBucket"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaAWSMCP": "aws-mcp.amazonaws.com"
            }
        }
    }
]
```

Restrict actions to a specific MCP server

The following policy denies all actions when the request comes specifically through AWS MCP Server, while allowing requests from other AWS managed MCP servers or direct API calls.

```
{
    "Sid": "DenyActionsViaAWSMCPServer",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaAWSMCP": "aws-mcp.amazonaws.com"
        }
    }
}
```

Troubleshooting AWS MCP Server identity and access

Use the following information to help you diagnose and fix common issues when working with AWS MCP Server and IAM.

Topics

- I get an access denied error when using AWS MCP Server
- My Deny policy using aws-mcp actions no longer blocks access

I get an access denied error when using AWS MCP Server

If you receive an `AccessDenied` error when AWS MCP Server calls a downstream AWS service on your behalf, check the following:

- Verify that your IAM role or user has the required permissions for the target AWS service action. AWS MCP Server uses your credentials, so you need the same permissions as you would for a direct API call.
- Check whether any SCPs or permission boundaries include Deny statements that use `aws:ViaAWSMCPService` or `aws:CalledViaAWSMCP` conditions that block MCP server access.
- If you previously used `aws-mcp:InvokeMcp` in Allow statements, note that these actions no longer have any effect. Your permissions for the downstream service are what matter.

My Deny policy using aws-mcp actions no longer blocks access

If you previously used Deny statements with `aws-mcp:InvokeMcp`, `aws-mcp:CallReadOnlyTool`, or `aws-mcp:CallReadWriteTool` to block access to AWS MCP Server, these actions no longer have any effect. Update your policies to use the condition context keys instead:

```
{
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "Bool": {
            "aws:ViaAWSMCPService": "true"
        }
    }
```

```
}
```

Compliance validation for AWS MCP Server

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in AWS MCP Server

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Agent Toolkit for AWS offers several features to help support your data resiliency and backup needs.

Monitoring AWS MCP Server

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS MCP Server and your other AWS solutions. AWS provides the following monitoring tools to watch AWS MCP Server, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. AWS MCP Server automatically publishes metrics to CloudWatch at no additional cost. For more information, see AWS MCP Server CloudWatch metrics.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

AWS MCP Server CloudWatch metrics

AWS MCP Server automatically publishes metrics to Amazon CloudWatch at no additional cost. You can use these metrics to monitor usage patterns, track success rates, identify errors, and set up alarms for your AWS MCP Server operations.

Metrics namespace

All AWS MCP Server metrics are published under the `AWS-MCP` namespace in CloudWatch. Metrics are organized by tool name, allowing you to monitor which MCP tools you use most frequently (such as `aws___call_aws` or `aws___list_regions`) and track their success rates.

Available metrics

The following metrics are available for AWS MCP Server. All metrics use standard resolution (1-minute granularity).

Metric	Description	Unit
`Invocation`	The number of times a tool was called, regardless of the response status.	Count

Metric	Description	Unit
`Success`	The number of successful requests that returned a 200 response.	Count
`UserError`	The number of requests that failed with a 4XX client error (excluding throttles).	Count
`SystemError`	The number of requests that failed with a 5XX server error.	Count
`Throttle`	The number of requests that were throttled with a 429 response.	Count

Metric dimensions

Metrics include the following dimensions to help you filter and analyze your data:

Tool Name

The name of the specific MCP tool that was invoked. For example, `aws___call_aws`, `aws___list_regions`, or `aws___retrieve_skill`. For a complete list of available tools, see Understanding the MCP Server tools.

Using metrics

You can use AWS MCP Server metrics to:

- **Monitor usage patterns** – Track which tools you use most frequently to understand your interaction patterns with AWS services.
- **Identify errors** – Monitor `UserError` and `SystemError` metrics to quickly detect and troubleshoot issues such as permission problems or service disruptions.
- **Track success rates** – Calculate success rates by comparing `Success` counts to `Invocation` counts to ensure your operations are completing successfully.
- **Set up alarms** – Create CloudWatch alarms to notify you when error rates exceed thresholds or when usage patterns change unexpectedly.

- **Optimize costs** – Monitor invocation counts to identify inefficient usage patterns that might be increasing your AWS costs.

For more information about working with CloudWatch metrics, see Using Amazon CloudWatch metrics in the *CloudWatch User Guide*.

Example use cases

The following examples show how you can use AWS MCP Server metrics:

Calculate success rate for API calls

Filter metrics by `Tool Name = aws___call_aws` and compare `Success` to `Invocation` to calculate your API call success rate. Set up an alarm to notify you if the success rate drops below 95%.

Detect permission issues

Monitor `UserError` metrics for specific tools. A spike in user errors often indicates IAM permission issues or incorrect API parameters.

Track tool usage trends

Compare `Invocation` counts across different tools over time to understand which AWS services and operations you interact with most frequently.

Monitor system health

Set up alarms for `SystemError` metrics to be notified of service disruptions or infrastructure issues that might affect your operations.

Usage metrics

AWS MCP Server publishes usage metrics to the `AWS/Usage` namespace in CloudWatch. These metrics help you monitor your resource consumption relative to your account quotas. You can use these metrics with the Service Quotas console to measure utilization and create alarms as you approach a quota.

All usage metrics include the following dimensions: `Type`, `Resource`, `Service`, and `Class`. The `Service` dimension is always `AWS MCP` and the `Class` dimension is always `None`.

Metric name	Type	Resource	Description	Unit
`CallCount`	API	Request	The number of requests made to the AWS MCP Server in this account in the current Region.	Count
`ResourceCount`	Resource	ConcurrentConnection	The number of concurrent connections per AWS account in the current Region.	Count
`ResourceCount`	Resource	AccountSessionCount	The number of concurrent active sessions per AWS account in the current Region.	Count
`ResourceCount`	Resource	UserSessionCount	The number of concurrent active sessions per user in this AWS account in the current Region.	Count

Note

Requests that are throttled before reaching the AWS MCP Server are not reflected in the `CallCount` metric. As a result, the metric may not reliably report values above the throttling limit.

For more information about monitoring your usage and setting up alarms, see AWS usage metrics in the *CloudWatch User Guide* and Service Quotas and Amazon CloudWatch in the *Service Quotas User Guide*.

Logging AWS MCP Server API calls using AWS CloudTrail

AWS MCP Server is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS MCP Server. CloudTrail captures all API calls for AWS MCP Server as events. The calls captured include calls from the AWS MCP Server console and code calls to the AWS MCP Server API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS MCP Server. If you

don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS MCP Server, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS MCP Server information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS MCP Server, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for AWS MCP Server, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All AWS MCP Server actions are logged by CloudTrail and are documented in the Agent Toolkit for AWS API Reference. For example, calls to the `ACTION_1`, `ACTION_2` and `ACTION_3` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding AWS MCP Server log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

> **Important**
>
> Tool names in CloudTrail logs may not match exactly the tools shown in your MCP client. For example:
>
> - MCP client shows: `aws___retrieve_skill`
> - CloudTrail logs show: `retrieve_skill`
>
> This occurs because CloudTrail logs the tool name without the namespace prefix used by MCP clients.

The following example shows a CloudTrail log entry that demonstrates the `CallTool` action.

```
{
  "eventVersion": "1.08",
  "eventCategory": "Data",
  "eventType": "AwsMcpEvent",
  "userIdentity": {
      ...
  },
  "eventTime": "...",
  "eventSource": "aws-mcp.us-east-1.api.aws",
  "eventName": "CallTool",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "...",
  "delegatedViaAWS": "...",
  "requestParameters": {
    "method": "call_aws",
    "params": {
```

```
        // Exact copy of MCP request params
    },
    "id": "request-id"
  },
  "responseElements": {
    "content": [
      {
        "type": "text",
        "text": "example"
      }
    ],
    "isError": false
  },
  "requestID": "12345678-1234-1234-1234-123456789012",
  "eventID": "87654321-4321-4321-4321-210987654321",
  "readOnly": true,
  "recipientAccountId": "123456789012",
  "resources": [
    {
      "type": "AWS::S3::Bucket",
      "ARN": "arn:aws:s3:::example-bucket-1",
      "accountId": "123456789012"
    }
  ],
  "mcpEventDetails": {
    "sessionId": "sess_xyz789_YXJuOmF3czppYW06OjEyMzQ1Njc4OTAxMjpkZXZlbG9wZXI=",
    "mcpProtocolVersion": "2024-11-05",
    "serverVersion": "1.0.0",
    "mcpServerName": "aws-mcp.us-east-1.api.aws",
    "executionTimeMs": 250,
    ...
  }
}
```

Quotas for AWS MCP Server

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for AWS MCP Server, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **AWS MCP Server**.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the limit increase form.

Topics

- Connection and session quotas
- Throttling quotas
- Session limits
- Monitoring your quotas

Connection and session quotas

The following quotas apply to connections and sessions for AWS MCP Server.

Quota	Default value	Adjustable	Description
Concurrent connections per account per Region	27	No	The maximum number of concurrent MCP connections per AWS account in a single Region.
Concurrent active sessions per account per Region	180	Yes	The maximum number of concurrent active sessions per AWS account in a single Region.
Concurrent active sessions per user per Region	90	Yes	The maximum number of concurrent active sessions per IAM user or role in a single Region.

Throttling quotas

The following quotas apply to request rates for AWS MCP Server. Requests that exceed these limits are throttled with a 429 response.

Quota	Default value	Adjustable	Description
Requests per account per Region	3 per second (sustained)	No	The maximum number of requests per second to the AWS MCP Server per AWS account in a single Region.

Session limits

The following limits apply to individual sessions with AWS MCP Server. These limits are not adjustable.

Limit	Value	Adjustable	Description
Maximum ephemeral storage retention	8 hours	No	The maximum duration that ephemeral compute storage is retained for a session. After this time, ephemeral storage is reclaimed.

Monitoring your quotas

AWS MCP Server publishes usage metrics to the `AWS/Usage` namespace in CloudWatch. You can use these metrics with the Service Quotas console to measure your utilization and create alarms as you approach a quota. The following usage metrics are available:

- `CallCount` (Resource: `Request`) – The number of requests made to the AWS MCP Server
- `ResourceCount` (Resource: `ConcurrentConnection`) – The number of concurrent connections
- `ResourceCount` (Resource: `AccountSessionCount`) – The number of concurrent active sessions per account

- `ResourceCount` (Resource: `UserSessionCount`) – The number of concurrent active sessions per user

For more information about these metrics, see Usage metrics in the AWS MCP Server CloudWatch metrics section.

> **Note**
>
> Requests that are throttled before reaching the AWS MCP Server are not reflected in the `CallCount` metric. For more information about this limitation, see Usage metrics.

For more information about monitoring your usage and setting up alarms, see AWS usage metrics in the *CloudWatch User Guide* and Service Quotas and Amazon CloudWatch in the *Service Quotas User Guide*.

Document history for the Agent Toolkit for AWS User Guide

The following table describes the documentation releases for Agent Toolkit for AWS.

Change	Description	Date
General availability of the Agent Toolkit for AWS	The Agent Toolkit for AWS is now generally available, with agent skills, plugins, and rules files. The user guide has been restructured to cover all toolkit components.	April 30, 2026
Deployment SOPs updates	Updated Deployment SOPs documentation with `deploy-supabase-app` guidance, and overall improved guidance.	February 18, 2026
Initial release	Initial release of the Agent Toolkit for AWS User Guide	November 30, 2025
Deployment SOPs	Added content describing the Deployment SOPs now available in Agent Toolkit for AWS	January 26, 2025