User Guide

Claude Platform on AWS

Claude Platform on AWS: User Guide

Table of Contents

What is Claude Platform on AWS? 1
How the platform integration works 1
Features of Claude Platform on AWS 1
How this guide is organized 2
Claude Platform on AWS vs Amazon Bedrock 3
Set up your account 5
Step 1: Sign up in the AWS Console 5
Step 2: Set up your Anthropic organization 6
Step 3: Note your workspace ID 6
Step 4: Sign in to the Claude Console 6
Troubleshooting account setup 7
Prerequisites 8
Enable outbound web identity federation 8
Obtain your workspace ID 8
Authentication 10
SigV4 authentication 10
API key authentication 10
Short-term API keys 11
Credential precedence and region resolution 12
Workspace ID 12
Install an SDK 13
Available models 15
Model IDs 15
Making requests 16
Using the base Anthropic client 19
Feature support 21
Claude Managed Agents 21
Compliance 22
How workspace membership is modeled 22
Features not currently available 23
Data residency 24
Workspaces 28
Workspace scoping 28
Creating workspaces 28

Setting the workspace ID in your client ... 28
Tagging workspaces ... 29
Tag-based access control ... 29
Using the Claude Console ... 31
Signing in ... 31
Available pages ... 31
Switching organizations ... 33
Rate limits and quotas ... 34
Default limits ... 34
Rate limit headers ... 34
Requesting higher limits ... 35
Rate limit errors ... 35
Billing ... 36
Monitoring and logging ... 37
Request IDs ... 37
Usage metrics and dashboards ... 41
Cost allocation and monitoring ... 41
Migrating from Amazon Bedrock ... 42
What changes ... 42
What you gain ... 43
What stays the same ... 43
Migration pitfalls ... 43
Commercial considerations ... 44
IAM policies ... 45
Example: deny batch inference ... 45
Example: synchronous inference on a single workspace ... 46
Example: per-customer workspace isolation ... 47
Example: feature lockdown for a ZDR-sensitive workspace ... 47
Example: provisioning automation ... 48
Managed policies ... 49
Federating to the Claude Console ... 50
Calling with bearer tokens ... 51
Outbound web identity federation (required for console access) ... 51
IAM actions for Claude Platform on AWS ... 53
Service details ... 53
Actions ... 53

Inference ... 53
Batch processing ... 54
Models ... 54
Files ... 54
Skills ... 55
User profiles ... 55
Agents ... 56
Sessions ... 56
Environments ... 57
Vaults ... 57
Memory stores ... 58
Workspaces ... 59
Authentication ... 59
Console access ... 60
Route-to-action mapping ... 61
See also ... 63
Document history ... 64

What is Claude Platform on AWS?

Access Claude's full platform capabilities through AWS with Anthropic-managed infrastructure.

Claude Platform on AWS gives you the full Anthropic platform experience, including the Messages API, Agent Skills, code execution, and beta features, accessible through your AWS account. Unlike Amazon Bedrock, where AWS operates the inference stack, Anthropic operates Claude Platform on AWS. AWS provides the authentication layer (SigV4 or API key), IAM-based access control, and billing integration through AWS Marketplace.

> **Note**
>
> The Anthropic SDKs support Claude Platform on AWS. For per-language client availability, see the Anthropic Client SDKs documentation.

How the platform integration works

Claude models run on Anthropic-managed infrastructure. This is a commercial integration for billing and access through AWS. Both Anthropic and AWS act as independent data processors. Use of Claude Platform on AWS is governed by the AWS Service Terms and is subject to Anthropic's Commercial Terms of Service, Data Processing Addendum, Usage Policy, and other agreements with Anthropic governing your use of their services.

Note the following operational characteristics. Data may not reside in AWS. Inference may route to Anthropic's primary cloud. Subservices may change without notice. Set the `inference_geo` parameter per request to pin inference to a specific geography. See Data residency for details.

Claude Platform on AWS follows the same data retention policy as the first-party Claude API. Zero Data Retention (ZDR) is available on request. Contact your Anthropic account representative to enable it for your account.

Features of Claude Platform on AWS

Claude Platform on AWS provides the following capabilities:

- **Same-day model and feature access** — New Claude models and API features are available the same day they launch on the first-party Claude API.

- **Agent Skills** — Use pre-built Skills for document generation (PowerPoint, Excel, Word, PDF) and create custom Skills.
- **Code execution** — Run code in Anthropic's managed sandbox.
- **Extended thinking** — Enable extended thinking for complex reasoning tasks.
- **Streaming and batch processing** — Use real-time SSE streaming or submit batch requests for high-throughput workloads.
- **Prompt caching** — Cache tools, system prompts, and message history to reduce latency and cost.
- **Files API** — Upload and reference files across requests.
- **AWS IAM integration** — Control access with standard IAM policies and SigV4 authentication.
- **Unified AWS billing** — Pay through your existing AWS account with consumption-based metering (Marketplace as billing backend).

For the full list of supported features, see Feature support.

How this guide is organized

This guide covers the following topics:

- **Getting started** — Account setup, prerequisites, authentication, and SDK installation.
- **Using the API** — Available models, making requests, and feature support.
- **Managing your environment** — Data residency, workspaces, the Claude Console, rate limits, and billing.
- **Monitoring and operations** — CloudTrail logging and request ID tracking.
- **Migration** — Moving from Amazon Bedrock to Claude Platform on AWS.
- **Security and access control** — IAM policies and actions reference.

Claude Platform on AWS vs Amazon Bedrock

Both offerings let you use Claude through AWS, but they differ significantly in architecture, API surface, and feature availability.

	Claude Platform on AWS	Amazon Bedrock
Who operates the stack	Anthropic	AWS
API surface	Anthropic Messages API (`/v1/messages`)	Bedrock API (Converse / InvokeModel)
Feature availability	Same-day as Claude API	Depends on AWS integration timeline
Agent Skills	Available	Not available (requires code execution)
Beta features	Pass through with `anthropic-beta` headers (see Feature support)	Requires AWS support
Authentication	AWS IAM / SigV4 or API key	AWS IAM / SigV4 or bearer token (C#, Go, and Java SDKs only)
Base URL	`aws-external-anthropic.{region}.api.aws`	`bedrock-runtime.{region}.amazonaws.com`
SDK client	Platform-specific client class (for example, `AnthropicAWS` in Python), in beta	`AnthropicBedrock` / Bedrock SDK
Console	Claude Console (`platform.claude.com`, access through the AWS Console)	Bedrock Console
Rate limits and quotas	Managed by Anthropic	Managed by AWS

	Claude Platform on AWS	Amazon Bedrock
Data processors	Anthropic and AWS	AWS

If you need AWS-operated Claude with the Bedrock API, see Amazon Bedrock.

> **Important**
>
> Claude Platform on AWS is a third-party offering provided by Anthropic and is not included within the scope of standard AWS compliance programs, certifications, or audit reports (such as SOC, ISO, or HIPAA eligibility). Customers are solely responsible for performing their own due diligence to ensure that this third-party offering meets their regulatory, legal, and compliance requirements.

When to choose Bedrock: Choose Amazon Bedrock if your organization requires AWS to operate the inference stack, requires AWS to be the sole data processor, or requires coverage under AWS compliance programs, certifications, or audit reports (including FedRAMP High, IL4, IL5, SOC, ISO, and HIPAA eligibility). Bedrock runs entirely on AWS-controlled infrastructure with AWS as the operating party.

Set up your account

Setting up Claude Platform on AWS happens in four phases: sign up on the AWS Console service page, complete your Anthropic organization setup, note your workspace ID, and sign in to the Claude Console.

> **Note**
>
> Signing up through the AWS Console provisions a new Anthropic organization tied to your AWS account. This organization is separate from any existing organizations your company has with Anthropic, including Claude Enterprise organizations procured through AWS Marketplace. API keys, workspaces, and Claude Console settings from a first-party Anthropic organization don't carry over.
>
> If you have an existing Amazon Bedrock private offer, contact your Anthropic or AWS account executive before signing up so your discount applies from your first request. Discounts cannot be applied retroactively to usage incurred before your private offer is accepted. See Private offers.

Step 1: Sign up in the AWS Console

1. Open the AWS Console and navigate to the **Claude Platform on AWS** service page.
2. Choose **Sign up**.
3. Choose **Continue**.

The page shows a **Sign-up in progress** banner. Stay on the page. Sign-up takes a few minutes while AWS handles the AWS Marketplace subscription for you, then redirects you automatically.

If your organization has a private offer from Anthropic, the Console looks it up and prompts you to accept it in AWS Marketplace. See Private offers for details.

> **Note**
>
> If you use Claude Platform on AWS, your content (such as prompts and completions) is processed by Anthropic outside of AWS. See Anthropic's data use policies for details on how content and metadata are processed and stored.

Step 2: Set up your Anthropic organization

After sign-up completes, you're redirected to `platform.claude.com/partner-signup`.

1. Enter the email address of your organization's owner and choose **Get started**.
2. Check that email inbox for a setup link and follow it. If your browser shows a **Signed in as a different account** page, choose **Log out and continue**.
3. Complete the organization details form (organization name, entity type, country, intended use) and choose **Complete setup**.

Completing setup creates your Anthropic organization. Use of Claude Platform on AWS is governed by the AWS Service Terms and is subject to Anthropic's Commercial Terms of Service, Data Processing Addendum, Usage Policy, and other agreements with Anthropic governing your use of their services. The AWS Console service page now shows a left navigation with **Home**, **API keys**, **Quickstart**, and **Workspaces**.

Step 3: Note your workspace ID

A default workspace is provisioned automatically when you sign up. Additional workspaces can be created per region; see Workspaces for details on region binding, workspace management, and IAM resource scoping.

Find the workspace ID under **Workspaces** on the AWS Console **Claude Platform on AWS** service page or in the Claude Console (see Using the Claude Console). Workspace IDs use the format `wrkspc_` followed by an alphanumeric identifier.

Step 4: Sign in to the Claude Console

Access to the Claude Console is federated through AWS IAM:

1. Assume an IAM role with the `aws-external-anthropic:AssumeConsole` permission. See IAM actions.
2. From the **Claude Platform on AWS** service page, choose **Sign in**. The AWS Console issues a JWT and redirects you to `platform.claude.com`.
3. On first sign-in, you're prompted for an email address. Enter your work email. The platform provisions your Claude Console user just-in-time.

When you're signed in through the AWS Console, the Claude Console scopes to your Claude Platform on AWS organization. An **Account managed by AWS** indicator appears in the bottom-left of the Claude Console sidebar.

Troubleshooting account setup

- **"Sign-up failed: Failed to enable OutboundWebIdentityFederation":** If you see this red banner on first submit, choose **Continue** again. The IAM enablement can take a moment to propagate.
- **No progress indicator during sign-up:** Sign-up takes a few minutes. The page shows a static **Sign-up in progress** banner without a progress bar while AWS provisions your account.
- **"Signed in as a different account" after following the setup link:** Choose **Log out and continue**. The page reauthenticates you with the email address you entered.
- **"Not found" message during sign-in:** This message might appear briefly during redirect. You can dismiss it.
- **Usage page shows no data after your first API call:** Usage data can take a few minutes to appear in the Claude Console.

Prerequisites

Before making API calls, ensure you have:

1. An active AWS account with a subscription to Claude Platform on AWS (see Set up your account).
2. The AWS CLI installed and configured.
3. **Outbound web identity federation enabled** on your AWS account (a one-time setup step; see Enable outbound web identity federation).
4. Your workspace ID (see Obtain your workspace ID).

Enable outbound web identity federation

The Claude Platform on AWS gateway calls `sts:GetWebIdentityToken` server-side to mint a JWT it forwards to Anthropic. This STS capability is **disabled by default** on every AWS account. Enable it once per account:

```
aws iam enable-outbound-web-identity-federation
```

If the response is `[ERROR] (FeatureEnabled) … already enabled` , the setting is already on for your account and you can move on. Verify and retrieve your account's issuer URL:

```
aws iam get-outbound-web-identity-federation-info
```

> **Warning**
>
> Without this step, every request returns `"Outbound web identity federation is disabled for your account"`. This is the most common setup error.

Obtain your workspace ID

A default workspace is provisioned automatically in each region when you sign up (see Set up your account). Workspaces are bound to a single AWS region. You can find the workspace ID in the Claude Console under **Workspaces**, or in the **Workspaces** section of the AWS Console service page. See Workspaces for region binding and IAM resource scoping.

Set the ANTHROPIC_AWS_WORKSPACE_ID and AWS_REGION environment variables so the SDK clients read them automatically:

```
export ANTHROPIC_AWS_WORKSPACE_ID='wrkspc_01AbCdEf23GhIj'
export AWS_REGION='us-west-2'
```

The region is required. The SDK client throws if no region is set. Pass aws_region/awsRegion to the constructor, or set AWS_REGION (or AWS_DEFAULT_REGION). All AWS commercial regions are supported; opt-in regions require that you first opt the account in to the region.

Authentication

Claude Platform on AWS supports two authentication methods: AWS IAM with SigV4 request signing (primary) and API key authentication. Both use the same base URL and request format.

The Anthropic SDKs (see Install an SDK) implement the authentication flow and credential resolution described below. If you aren't using an Anthropic SDK, you must construct SigV4-signed requests (or present your API key as a bearer token) against the regional endpoint `https://aws-external-anthropic.<region>.api.aws`. For SDK-specific client configuration and the authoritative credential-resolution order, see the Claude on AWS setup guide in the Anthropic documentation.

SigV4 authentication

SigV4 is the enterprise-native path and integrates with your existing AWS IAM policies, roles, and auditing. Configure AWS credentials using any method supported by the AWS default credential provider chain:

- Environment variables (`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_SESSION_TOKEN`)
- Shared credentials file (`~/.aws/credentials`)
- Shared config file (`~/.aws/config`) including SSO and `credential_process`
- Web identity (`AWS_WEB_IDENTITY_TOKEN_FILE` and `AWS_ROLE_ARN`) for IRSA and GitHub Actions
- ECS container credentials
- EC2 instance metadata service (IMDS)

To verify the Anthropic SDK is picking up your credentials and resolving to the correct regional endpoint, make a test request following the examples in Making requests. A successful response confirms that credentials, region, base URL, and workspace ID are all correctly configured.

API key authentication

For simpler integration paths (local development and scripts), you can authenticate with an API key instead of SigV4. Set the `ANTHROPIC_AWS_API_KEY` environment variable or pass

`apiKey` to the SDK constructor. The Anthropic SDK sends the key as a bearer token to the Claude Platform on AWS endpoint; IAM authorizes the request through the `aws-external-anthropic:CallWithBearerToken` action (see IAM policies).

Generate API keys in the **AWS Console** under **Claude Platform on AWS** → **API keys**. Choose **Generate a key**, then copy the key value. Grant the `aws-external-anthropic:CallWithBearerToken` IAM action to the principals that should be allowed to use API key authentication.

> **Note**
>
> Only API keys created in the AWS Console under **Claude Platform on AWS** work with this service.
>
> - Keys created in the standard Claude Console for first-party API access do not work against the Claude Platform on AWS endpoint.
> - Amazon Bedrock API keys do not work either — Bedrock uses a separate endpoint, authentication flow, and IAM namespace.

Short-term API keys

For cases where you want API key authentication but without the lifetime of a long-lived key, Anthropic publishes token-generator libraries that mint short-term API keys from your AWS IAM credentials. Tokens default to a 12-hour lifetime and can be scoped to a specific workspace and to the `aws-external-anthropic:CallWithBearerToken` action. Install the generator that matches your language, then exchange IAM credentials for an API key and pass the result to the Anthropic SDK the same way you would a long-lived key.

- **Python:** aws/token-generator-for-aws-external-anthropic-python
- **JavaScript / TypeScript:** aws/token-generator-for-aws-external-anthropic-js
- **Java:** aws/token-generator-for-aws-external-anthropic-java

Short-term API keys still require the caller's IAM principal to hold `aws-external-anthropic:CallWithBearerToken` on the target workspace. They expire on their own and do not need to be rotated or revoked explicitly.

Credential precedence and region resolution

The Anthropic SDK's Claude Platform on AWS client resolves credentials and region using a defined precedence order. Argument names vary by language convention (TypeScript and PHP use camelCase; Python and Ruby use snake_case; Go uses Pascal case with capitalized acronyms; C# and Java use the language's property or builder idioms).

For the authoritative precedence order, supported environment variables, and constructor arguments for each SDK, see Claude on AWS setup in the Anthropic documentation. In general, explicit constructor arguments take precedence over environment variables, and `ANTHROPIC_AWS_API_KEY` takes precedence over the default AWS credential provider chain. Region is required; unlike `AnthropicBedrock` (which falls back to `us-east-1`), the Claude on AWS client throws if no region is supplied by the constructor or by `AWS_REGION` / `AWS_DEFAULT_REGION`.

Workspace ID

Every data plane request must include the workspace ID in the `anthropic-workspace-id` header. The Anthropic SDKs read this from the `ANTHROPIC_AWS_WORKSPACE_ID` environment variable by default, or you can pass `workspaceId` / `workspace_id` to the client constructor. If you use the base `Anthropic` client (not the Claude-on-AWS client), pass the header explicitly. See Making requests for per-language examples and Workspaces for how to locate your workspace ID.

Install an SDK

Anthropic's client SDKs support Claude Platform on AWS. All seven SDKs provide a platform-specific client class; alternatively, you can configure the base client with the Claude Platform on AWS base URL.

Python

```
pip install -U "anthropic[aws]"
```

> **Tip**
>
> On macOS with Homebrew Python or other externally managed Python environments, `pip install` may fail with a PEP 668 `externally-managed-environment` error. Create and activate a virtual environment first: `python3 -m venv .venv && source .venv/bin/activate`.

TypeScript

```
npm install @anthropic-ai/aws-sdk
```

C#

```
dotnet add package Anthropic.Aws
```

Go

```
go get github.com/anthropics/anthropic-sdk-go
```

Java

```
implementation("com.anthropic:anthropic-java-aws:2.27.0")
```

```
<dependency>
  <groupId>com.anthropic</groupId>
  <artifactId>anthropic-java-aws</artifactId>
```

```
  <version>2.27.0</version>
</dependency>
```

PHP

```
composer require anthropic-ai/sdk aws/aws-sdk-php
```

Ruby

```
gem install anthropic aws-sdk-core
```

> **Note**
>
> SDK clients for Claude Platform on AWS are in beta.

Available models

Claude Platform on AWS offers the same set of Claude models as the first-party Claude API.

For the current list of models, model IDs, context window sizes, and capabilities, see Models overview in the Anthropic documentation.

Model IDs

Model IDs on Claude Platform on AWS are identical to those used on the first-party Claude API (for example, `claude-opus-4-6`, `claude-sonnet-4-6`, `claude-haiku-4-5`). There are no Bedrock-style ARNs or `anthropic.` prefixes — use the model ID exactly as Anthropic publishes it.

Making requests

Claude Platform on AWS uses the Anthropic Messages API (`/v1/messages`), the same API surface as the Claude first-party platform. The differences are the base URL, the authentication method, and a required `anthropic-workspace-id` header that identifies which workspace the request targets.

Shell

```
curl "https://aws-external-anthropic.us-west-2.api.aws/v1/messages" \
  --aws-sigv4 "aws:amz:us-west-2:aws-external-anthropic" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  -H "x-amz-security-token: $AWS_SESSION_TOKEN" \
  -H "content-type: application/json" \
  -H "anthropic-version: 2023-06-01" \
  -H "anthropic-workspace-id: $ANTHROPIC_AWS_WORKSPACE_ID" \
  -d '{
    "model": "claude-sonnet-4-6",
    "max_tokens": 1024,
    "messages": [
      {"role": "user", "content": "Hello!"}
    ]
  }'
```

Python

```
from anthropic import AnthropicAWS

client = AnthropicAWS()

message = client.messages.create(
    model="claude-sonnet-4-6",
    max_tokens=1024,
    messages=[{"role": "user", "content": "Hello!"}],
)
print(message)
```

TypeScript

```
import AnthropicAws from "@anthropic-ai/aws-sdk";
```

```
const client = new AnthropicAws();

const message = await client.messages.create({
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{ role: "user", content: "Hello!" }]
});
console.log(message);
```

C#

```
using Anthropic;
using Anthropic.Aws;

var client = new AnthropicAwsClient();

var message = await client.Messages.Create(new()
{
    Model = "claude-sonnet-4-6",
    MaxTokens = 1024,
    Messages = [new() { Role = "user", Content = "Hello!" }]
});

Console.WriteLine(message);
```

Go

```
client, err := anthropicaws.NewClient(context.Background(),
 anthropicaws.ClientConfig{})
if err != nil {
    panic(err)
}

message, err := client.Messages.New(context.Background(), anthropic.MessageNewParams{
    Model:     anthropic.ModelClaudeSonnet4_6,
    MaxTokens: 1024,
    Messages: []anthropic.MessageParam{
        anthropic.NewUserMessage(anthropic.NewTextBlock("Hello!")),
    },
})
if err != nil {
    panic(err)
```

```
}

fmt.Println(message.Content[0].Text)
```

Java

```
import com.anthropic.aws.backends.AwsBackend;
import com.anthropic.client.AnthropicClient;
import com.anthropic.client.okhttp.AnthropicOkHttpClient;
import com.anthropic.models.messages.Message;
import com.anthropic.models.messages.MessageCreateParams;
import com.anthropic.models.messages.Model;

AnthropicClient client = AnthropicOkHttpClient.builder()
    .backend(AwsBackend.fromEnv())
    .build();

Message message = client.messages().create(
    MessageCreateParams.builder()
        .model(Model.CLAUDE_SONNET_4_6)
        .maxTokens(1024)
        .addUserMessage("Hello!")
        .build()
);

IO.println(message);
```

PHP

```
use Anthropic\Aws\Client;

$client = new Client();

$message = $client->messages->create(
    model: 'claude-sonnet-4-6',
    maxTokens: 1024,
    messages: [['role' => 'user', 'content' => 'Hello!']],
);

echo $message;
```

Ruby

```
require "anthropic"

client = Anthropic::AWSClient.new

message = client.messages.create(
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{ role: "user", content: "Hello!" }]
)

puts message
```

The client reads `AWS_REGION` (or `AWS_DEFAULT_REGION`) and `ANTHROPIC_AWS_WORKSPACE_ID` from the environment. You can override either by passing `aws_region` / `awsRegion` or `workspace_id` / `workspaceId` to the constructor. Both region and workspace ID are required; the constructor throws if either cannot be resolved from these sources.

> **Note**
>
> The `x-amz-security-token` header (curl) is only required for temporary credentials such as IAM roles, SSO, or STS. Omit it when using long-term IAM user credentials. The SDK clients handle this automatically based on the credential source.

The `--aws-sigv4` value follows the format `aws:amz:<region>:<service>`. The SigV4 service name is `aws-external-anthropic`, and the region must match the region in your endpoint URL. A mismatch in either produces a generic signature-rejection error rather than a specific diagnostic.

Using the base Anthropic client

If you prefer not to add the AWS SDK dependency, you can use the base `Anthropic` client. This path uses the vanilla SDK environment variable names rather than the `ANTHROPIC_AWS_*` names the dedicated client reads:

```
export ANTHROPIC_API_KEY='your-api-key'
export ANTHROPIC_BASE_URL='https://aws-external-anthropic.us-west-2.api.aws'
export ANTHROPIC_WORKSPACE_ID='your-workspace-id'
```

The Python, TypeScript, and Go SDKs read `ANTHROPIC_API_KEY` and `ANTHROPIC_BASE_URL` automatically; for the other languages, pass them to the constructor. In every language, pass the workspace ID in the `anthropic-workspace-id` header.

Python

```
import os
from anthropic import Anthropic

client = Anthropic(
    # ANTHROPIC_API_KEY and ANTHROPIC_BASE_URL are read automatically
    default_headers={"anthropic-workspace-id": os.environ["ANTHROPIC_WORKSPACE_ID"]},
)

message = client.messages.create(
    model="claude-sonnet-4-6",
    max_tokens=1024,
    messages=[{"role": "user", "content": "Hello!"}],
)
print(message.content[0].text)
```

TypeScript

```
import Anthropic from "@anthropic-ai/sdk";

const client = new Anthropic({
  // ANTHROPIC_API_KEY and ANTHROPIC_BASE_URL are read automatically
  defaultHeaders: { "anthropic-workspace-id": process.env.ANTHROPIC_WORKSPACE_ID }
});

const message = await client.messages.create({
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{ role: "user", content: "Hello!" }]
});
console.log(message.content);
```

Feature support

Claude Platform on AWS uses the Anthropic Messages API directly, which means you get full Messages API feature parity with the first-party Claude API (except where noted in Features not currently available):

- **Beta features:** Pass the standard `anthropic-beta` header to access beta features, just as you would with the Claude API.
- **Agent Skills:** Use pre-built and custom Agent Skills with the same `container.skills` parameter and beta headers as the Claude API. All pre-built Skills (PowerPoint, Excel, Word, PDF) work out of the box.
- **Code execution:** Run code in Anthropic's managed sandbox using the code execution tool.
- **Tool use:** Computer use and all other tool use capabilities are available.
- **Extended thinking:** Enable extended thinking with the same parameters as the Claude API.
- **Streaming:** Full SSE streaming support for real-time responses.
- **Batch processing:** Submit batch requests for high-throughput workloads.
- **Prompt caching:** Cache tools, system prompts, and message history to reduce latency and cost. All prompt caching capabilities (5-minute TTL, 1-hour TTL, and automatic caching) are available.
- **Files API:** Upload and reference files across requests.
- **Workspace tags with IAM integration:** Workspaces can be tagged, and tags flow through to IAM for attribute-based access control and to AWS Cost and Usage Reports for cost allocation. Workspace tagging is a GA feature on Claude Platform on AWS (it is a beta feature on the first-party Claude API).

Claude Managed Agents

Claude Managed Agents are available on Claude Platform on AWS. Agents, sessions, environments (cloud container configurations), credential vaults, and memory stores are first-class IAM resources; see IAM actions for the action reference and Using the Claude Console for the corresponding Claude Console pages. The Anthropic Agent SDK works against Claude Platform on AWS without a separate integration step — point it at the Claude Platform on AWS base URL and authenticate with SigV4 or an API key.

Autonomous sessions on Claude Platform on AWS require re-authentication every 6 hours. Long-running agent runs must refresh their SigV4 credentials or API key within that window, or the session ends.

The following Claude Managed Agents capabilities are not currently available on Claude Platform on AWS:

- **Outcomes:** Outcome tracking for agent sessions is not available.
- **Multi-agent sessions:** Sessions with multiple interacting agents are not available.
- **Webhooks:** Webhook delivery of session events is not available.

Compliance

> **Important**
>
> This service is a third-party offering provided by Anthropic and is not included within the scope of standard AWS compliance programs, certifications, or audit reports (such as SOC, ISO, or HIPAA eligibility). Customers are solely responsible for performing their own due diligence to ensure that this third-party offering meets their regulatory, legal, and compliance requirements. Before processing sensitive or regulated data, you should review Anthropic's official compliance documentation via the Anthropic Trust Center. See the AWS Service Terms for more information.

How workspace membership is modeled

On the first-party Claude API, access is governed by users-to-workspaces membership — a user is added to a workspace and inherits the workspace's access rights. Claude Platform on AWS replaces that model with IAM authorization: there is no per-workspace user list. Instead, IAM principals (users or roles) hold policies that grant `aws-external-anthropic` actions against specific workspace ARNs. The IAM policy evaluation determines what each principal can do in each workspace.

This means workspace access is granted and revoked by modifying IAM policies — through AWS Organizations SCPs, permission boundaries, identity-attached policies, or resource-level conditions — not by managing per-workspace membership in a Claude Console UI. See IAM policies for examples.

Features not currently available

The following capabilities are not currently available on Claude Platform on AWS:

- **HIPAA readiness:** Anthropic's HIPAA-ready program is not available on Claude Platform on AWS. Customers with HIPAA requirements should evaluate Claude in Amazon Bedrock instead.
- **Service tiers beyond Standard and Batch:** Priority Tier is not available.
- **Admin API — organization member, workspace member, invite, API key, usage report, cost report, and rate limit report endpoints:** These Admin API endpoints are not currently available. Workspace CRUD and rename endpoints (`/v1/organizations/workspaces`) are available through the `aws-external-anthropic` namespace; see IAM actions. Membership is modeled through AWS IAM rather than through workspace membership lists (see preceding section). API key lifecycle is managed in the AWS Console under **Claude Platform on AWS** → **API keys**. For usage, cost, and rate limit data, use the Claude Console views (see Monitoring and logging) or the Anthropic Usage and Cost API.
- **Spend limits:** Not available. Rely on AWS billing controls instead.
- **Claude Code workspace and Analytics API:** The Claude Code workspace with automatic rate limits is not available. Claude Code usage appears in the general usage view rather than a dedicated screen.
- **OAuth authentication:** Not supported. Use SigV4 or API key authentication.
- **OpenAI-compatible API endpoints:** Not available on Claude Platform on AWS.
- **Workspace-level inference geography controls:** `allowed_inference_geos` and `default_inference_geo` are not available. Set `inference_geo` on each request instead.

Data residency

Claude Platform on AWS supports the following inference geographies:

- **US:** Inference stays within US data centers. A 1.1x pricing multiplier applies.
- **Global:** Inference may route to any Anthropic-operated data center worldwide. Standard pricing applies.

Set the inference geography per request with the `inference_geo` parameter:

> **Note**
>
> The `inference_geo` parameter is supported on Claude Opus 4.6, Claude Sonnet 4.6, and later models. Requests with `inference_geo` on Claude Opus 4.5, Claude Sonnet 4.5, or Claude Haiku 4.5 return a 400 error. See Data residency for model availability details.

Shell

```
curl "https://aws-external-anthropic.us-west-2.api.aws/v1/messages" \
  --aws-sigv4 "aws:amz:us-west-2:aws-external-anthropic" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  -H "x-amz-security-token: $AWS_SESSION_TOKEN" \
  -H "content-type: application/json" \
  -H "anthropic-version: 2023-06-01" \
  -H "anthropic-workspace-id: $ANTHROPIC_AWS_WORKSPACE_ID" \
  -d '{
    "model": "claude-sonnet-4-6",
    "max_tokens": 1024,
    "inference_geo": "us",
    "messages": [
      {"role": "user", "content": "Hello!"}
    ]
  }'
```

Python

```
from anthropic import AnthropicAWS
```

```
client = AnthropicAWS(aws_region="us-west-2")
message = client.messages.create(
    model="claude-sonnet-4-6",
    max_tokens=1024,
    inference_geo="us",
    messages=[{"role": "user", "content": "Hello!"}],
)
print(message)
```

TypeScript

```
import AnthropicAws from "@anthropic-ai/aws-sdk";
const client = new AnthropicAws({ awsRegion: "us-west-2" });
const message = await client.messages.create({
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  inference_geo: "us",
  messages: [{ role: "user", content: "Hello!" }]
});
console.log(message);
```

C#

```
using Anthropic;
using Anthropic.Aws;

var client = new AnthropicAwsClient(new() { AwsRegion = "us-west-2" });

var message = await client.Messages.Create(new()
{
    Model = "claude-sonnet-4-6",
    MaxTokens = 1024,
    InferenceGeo = "us",
    Messages = [new() { Role = "user", Content = "Hello!" }]
});

Console.WriteLine(message);
```

Go

```
client, err := anthropicaws.NewClient(context.Background(),
 anthropicaws.ClientConfig{})
```

```
if err != nil {
    panic(err)
}

message, err := client.Messages.New(context.Background(), anthropic.MessageNewParams{
    Model:        anthropic.ModelClaudeSonnet4_6,
    MaxTokens:    1024,
    InferenceGeo: anthropic.String("us"),
    Messages: []anthropic.MessageParam{
        anthropic.NewUserMessage(anthropic.NewTextBlock("Hello!")),
    },
})
if err != nil {
    panic(err)
}

fmt.Println(message)
```

Java

```
import com.anthropic.aws.backends.AwsBackend;
import com.anthropic.client.AnthropicClient;
import com.anthropic.client.okhttp.AnthropicOkHttpClient;
import com.anthropic.models.messages.Message;
import com.anthropic.models.messages.MessageCreateParams;
import com.anthropic.models.messages.Model;
import software.amazon.awssdk.regions.Region;

AnthropicClient client = AnthropicOkHttpClient.builder()
    .backend(AwsBackend.builder().region(Region.of("us-west-2")).build())
    .build();

Message message = client.messages().create(
    MessageCreateParams.builder()
        .model(Model.CLAUDE_SONNET_4_6)
        .maxTokens(1024)
        .inferenceGeo("us")
        .addUserMessage("Hello!")
        .build()
);

IO.println(message);
```

PHP

```
use Anthropic\Aws\Client;

$client = new Client(awsRegion: 'us-west-2');

$message = $client->messages->create(
    model: 'claude-sonnet-4-6',
    maxTokens: 1024,
    inferenceGeo: 'us',
    messages: [['role' => 'user', 'content' => 'Hello!']],
);

echo $message;
```

Ruby

```
require "anthropic"

client = Anthropic::AWSClient.new(aws_region: "us-west-2")

message = client.messages.create(
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  inference_geo: "us",
  messages: [{ role: "user", content: "Hello!" }]
)

puts message
```

If you omit `inference_geo`, the request defaults to `global`.

Workspace-level inference geography controls (`allowed_inference_geos` and `default_inference_geo`) are not available on Claude Platform on AWS. Set `inference_geo` on each request instead.

Workspaces

Inference and resource requests on Claude Platform on AWS target a workspace. You pass the workspace's ID in the `anthropic-workspace-id` header on these API calls. Workspace IDs use the tagged format `wrkspc_` followed by an alphanumeric identifier (for example, `wrkspc_01AbCdEf23GhIj`). See Obtain your workspace ID if you don't have it yet.

Workspace scoping

Workspaces are bound to a single AWS region. A workspace created in `us-west-2` can only be accessed through the `us-west-2` endpoint. Usage, quotas, cost, files, batches, and Skills all roll up per workspace, giving you per-region breakdowns in the Claude Console.

Workspaces also serve as the primary IAM resource for Claude Platform on AWS. You grant or deny access to specific workspaces through AWS IAM policies using the workspace ARN. The ARN's resource segment is the same `wrkspc_`-prefixed ID you pass in the `anthropic-workspace-id` header:

```
arn:aws:aws-external-anthropic:{region}:{account-id}:workspace/{workspace-id}
```

For example: `arn:aws:aws-external-anthropic:us-west-2:123456789012:workspace/wrkspc_01AbCdEf23GhIj`

See IAM policies for policy examples.

Creating workspaces

A default workspace is provisioned automatically at sign-up. Additional workspaces can be created, updated, renamed, and archived through the `/v1/organizations/workspaces` endpoints, authorized by the corresponding `aws-external-anthropic` actions (`CreateWorkspace`, `UpdateWorkspace`, `ArchiveWorkspace`); see IAM actions.

Setting the workspace ID in your client

Every data plane request must include the workspace ID in the `anthropic-workspace-id` header. When you use an Anthropic SDK's Claude-on-AWS client, there are three ways to supply it:

1. **Environment variable** — set `ANTHROPIC_AWS_WORKSPACE_ID`, and the client reads it automatically.

```
export ANTHROPIC_AWS_WORKSPACE_ID='wrkspc_01AbCdEf23GhIj'
```

2. **Constructor argument** — pass `workspaceId` / `workspace_id` (name follows the SDK's language convention) when instantiating the client.
3. **Per-request override** — pass the header directly on an individual request if you need to address multiple workspaces from the same client.

If you use the base `Anthropic` client (without the AWS backend), set the `anthropic-workspace-id` header explicitly on every request — see Making requests for per-language examples. For SDK-specific argument names, refer to the Anthropic Client SDKs documentation.

Tagging workspaces

Workspace tags are key-value pairs attached to a workspace. They integrate with AWS IAM for attribute-based access control, and with AWS Cost and Usage Reports for cost allocation (see Monitoring and logging for CUR details). Tagging is GA on Claude Platform on AWS.

Update tags on the existing workspace with `TagResource` and `UntagResource` in the `aws-external-anthropic` namespace. The same tag keys can drive IAM conditions and cost allocation without additional configuration.

Tag-based access control

You can gate `aws-external-anthropic` actions on workspace tag values using the `aws:ResourceTag/<key>` condition context key. The following policy allows inference only on workspaces tagged `Environment=production`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-external-anthropic:CreateInference",
        "aws-external-anthropic:CreateBatchInference",
        "aws-external-anthropic:CountTokens"
```

```
      ],
      "Resource": "arn:aws:aws-external-anthropic:*:*:workspace/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "production"
        }
      }
    }
  ]
}
```

Use `aws:RequestTag/<key>` and `aws:TagKeys` condition keys on `TagResource` to enforce tagging policies (for example, requiring the workspace to carry `CostCenter` and `Owner` tags). See IAM policies for more examples.

Using the Claude Console

Claude Platform on AWS uses the standard Claude Console at platform.claude.com. When you sign in from the AWS Console, an **Account managed by AWS** indicator appears in the bottom-left of the Claude Console sidebar and the Console scopes to your Claude Platform on AWS organization. It provides usage analytics, cost breakdowns, rate limit visibility, workspace visibility, and pages for managing files, Agent Skills, batch jobs, agents, sessions, environments, credential vaults, and memory stores.

Signing in

Access to the Claude Console is federated through AWS IAM. See Set up your account for the full first-time sign-in flow. In short:

1. Assume an IAM role with the `aws-external-anthropic:AssumeConsole` permission. See IAM actions.
2. Navigate to the Claude Platform on AWS page in the AWS Console.
3. Choose **Open Claude Console**. The AWS Console issues a JWT and redirects you to `platform.claude.com`.
4. On first sign-in, you're prompted for an email address; enter your work email. The platform provisions your Claude Console user just-in-time.

Two Claude Console roles are available: **Admin** and **Developer**. The Admin role grants access to all Claude Console pages and settings available for Claude Platform on AWS. The Developer role grants read access to usage, cost, rate limit, and workspace information. Contact your Anthropic account representative to assign the Admin or Developer role to a principal.

Available pages

The **Via AWS gateway** column indicates whether the page reads and writes data through the AWS gateway (and is therefore governed by IAM actions). Pages marked **No** read organization-level metadata directly through Anthropic APIs and bypass IAM action checks.

Page	Available	Via AWS gateway	Notes
Usage	Yes	No	View token usage by model, workspace, and dimension. Data may take a few minutes to appear after a request.
Cost	Yes	No	View cost breakdowns by model and workspace. AWS Cost Explorer shows the aggregated CCU line item.
Limits	Yes	No	View rate limits (read-only).
Workspaces	Yes	No	View per-region workspaces (read-only).
Files	Yes	Yes	View and manage uploaded files.
Skills	Yes	Yes	View and manage Agent Skills.
Batches	Yes	Yes	View and manage batch processing jobs.
Agents	Yes	Yes	View and manage agent definitions.
Sessions	Yes	Yes	View agent sessions and event history.
Environments	Yes	Yes	View and manage cloud container configurations for sessions.
Credential vaults	Yes	Yes	View and manage credential vaults for session authentication.
Memory stores	Yes	Yes	View and manage persistent agent memory.
API keys	No	N/A	Manage API keys in the AWS Console (**Claude Platform on AWS** → **API keys**). See Authentication.
Members	No	N/A	Not applicable. AWS IAM manages access.

Page	Available	Via AWS gateway	Notes
Billing	No	N/A	Not applicable. AWS Marketplace manages billing and invoicing. View cost breakdowns on the Cost page.
Claude Code	No	N/A	View Claude Code usage on the Usage page.

Switching organizations

The Claude Console does not support organization switching for Claude Platform on AWS. To access a different organization, sign out and reauthenticate through the AWS Console using the IAM role for that organization's AWS account.

Rate limits and quotas

Claude Platform on AWS assigns Tier 1 rate limits when you subscribe. Anthropic manages rate limits directly, not through AWS quota systems.

Default limits

Claude Platform on AWS uses Anthropic's standard tier schedule, identical to the first-party Claude API. Tier 1 limits apply per workspace. Limits are pooled by model family (for example, one combined limit covers Claude Opus 4.7, 4.6, 4.5, and earlier Opus models; Sonnet models share a separate combined limit; Haiku models share another).

For the current Tier 1 values — requests per minute (RPM), input tokens per minute (ITPM), output tokens per minute (OTPM) — and for higher-tier thresholds, see Rate limits on the Anthropic documentation website. The Anthropic page is the source of truth and is updated when limits change.

Rate limit headers

Every response includes headers that report your current rate limit status. Key headers:

- `anthropic-ratelimit-requests-limit` — Maximum requests per minute
- `anthropic-ratelimit-requests-remaining` — Requests remaining in the current window
- `anthropic-ratelimit-requests-reset` — Time when the request limit resets (RFC 3339)
- `anthropic-ratelimit-tokens-limit` — Maximum combined tokens (input + output) per minute
- `anthropic-ratelimit-tokens-remaining` — Combined tokens remaining in the current window
- `anthropic-ratelimit-tokens-reset` — Time when the combined token limit resets (RFC 3339)
- `anthropic-ratelimit-input-tokens-limit` / `-remaining` / `-reset` — Input-token-specific headers
- `anthropic-ratelimit-output-tokens-limit` / `-remaining` / `-reset` — Output-token-specific headers
- `retry-after` — On a 429 response, the number of seconds to wait before retrying

See Response headers on the Anthropic documentation website for the complete set.

Requesting higher limits

Unlike the first-party Claude API, automatic tier advancement does not apply on Claude Platform on AWS. To request higher limits, contact your Anthropic account representative with your workspace ID and desired throughput. For tier thresholds and other details, see Rate limits on the Anthropic documentation website.

Rate limit errors

When you exceed a rate limit, the API returns HTTP 429 with a `rate_limit_error` type. Implement exponential backoff with jitter in your retry logic. The `retry-after` header indicates how many seconds to wait before retrying.

Billing

Claude Platform on AWS uses AWS Marketplace as a billing backend for consumption-based metering. Anthropic meters usage hourly through AWS Marketplace, and AWS issues monthly invoices on your existing AWS bill.

Billing is **arrears-only** (you pay for what you used) and **pre-tax** (AWS applies your account's tax settings).

Usage is denominated in Claude Consumption Units (CCUs) at $0.01 USD per CCU. The CCU price is fixed and never discounted. Anthropic rates your token usage in USD at standard per-model, per-feature rates, applies any negotiated discount, then converts the result to CCUs at $0.01 per CCU. Discounts result in fewer CCUs metered, not a lower CCU price. CCUs are not prepaid credits; there is no CCU balance or commitment. See Pricing for the CCU definition and per-model token rates.

Monitoring and logging

AWS CloudTrail can capture all requests to Claude Platform on AWS. Workspace and vault operations are logged as Management events by default. Inference, batch, file, skill, model, user profile, and Claude Managed Agents operations (other than vaults) are classified as Data events and require explicit data event logging configuration, which incurs additional CloudTrail charges. See IAM actions for the full event type classification and the AWS CloudTrail documentation for configuration details.

When configuring CloudTrail data event logging, select the resource type `AWS::AWSExternalAnthropic::Workspace`. This is the CloudTrail resource type for Claude Platform on AWS workspaces and is required to capture data plane events (inference, batch, file, and other per-workspace operations). Scope data event selectors to specific workspace ARNs if you want to log activity for a subset of workspaces rather than the whole account.

Request IDs

Each response includes two request IDs in the response headers:

- **AWS request ID (`x-amzn-requestid`):** The primary ID, indexed in CloudTrail. Use this when investigating requests through AWS tooling or when contacting AWS support.
- **Anthropic request ID (`request-id`):** The secondary ID. Use this when contacting Anthropic support.

Python

```
from anthropic import AnthropicAWS

client = AnthropicAWS(aws_region="us-west-2")

response = client.messages.with_raw_response.create(
    model="claude-sonnet-4-6",
    max_tokens=1024,
    messages=[{"role": "user", "content": "Hello!"}],
)

print(response.headers.get("x-amzn-requestid"))  # AWS request ID
```

```
print(response.headers.get("request-id"))  # Anthropic request ID

message = response.parse()
print(message.content)
```

TypeScript

```
import AnthropicAws from "@anthropic-ai/aws-sdk";

const client = new AnthropicAws({ awsRegion: "us-west-2" });

const { data: message, response } = await client.messages
  .create({
    model: "claude-sonnet-4-6",
    max_tokens: 1024,
    messages: [{ role: "user", content: "Hello!" }]
  })
  .withResponse();

console.log(response.headers.get("x-amzn-requestid")); // AWS request ID
console.log(response.headers.get("request-id")); // Anthropic request ID
console.log(message.content);
```

C#

```
using Anthropic;
using Anthropic.Aws;

var client = new AnthropicAwsClient(new() { AwsRegion = "us-west-2" });

var response = await client.WithRawResponse.Messages.Create(new()
{
    Model = "claude-sonnet-4-6",
    MaxTokens = 1024,
    Messages = [new() { Role = "user", Content = "Hello!" }]
});

Console.WriteLine(response.Headers.GetValues("x-amzn-requestid").First()); // AWS
 request ID
Console.WriteLine(response.Headers.GetValues("request-id").First()); // Anthropic
 request ID
Console.WriteLine(response.Value.Content);
```

Go

```
client, err := anthropicaws.NewClient(context.Background(),
 anthropicaws.ClientConfig{})
if err != nil {
    panic(err)
}

var response *http.Response
message, err := client.Messages.New(
    context.Background(),
    anthropic.MessageNewParams{
        Model:     anthropic.ModelClaudeSonnet4_6,
        MaxTokens: 1024,
        Messages: []anthropic.MessageParam{
            anthropic.NewUserMessage(anthropic.NewTextBlock("Hello!")),
        },
    },
    option.WithResponseInto(&response),
)
if err != nil {
    panic(err)
}

fmt.Println(response.Header.Get("x-amzn-requestid")) // AWS request ID
fmt.Println(response.Header.Get("request-id"))       // Anthropic request ID
fmt.Println(message.Content[0].Text)
```

Java

```
import com.anthropic.aws.backends.AwsBackend;
import com.anthropic.client.AnthropicClient;
import com.anthropic.client.okhttp.AnthropicOkHttpClient;
import com.anthropic.core.http.HttpResponseFor;
import com.anthropic.models.messages.Message;
import com.anthropic.models.messages.MessageCreateParams;
import com.anthropic.models.messages.Model;

AnthropicClient client = AnthropicOkHttpClient.builder()
    .backend(AwsBackend.fromEnv())
    .build();

HttpResponseFor<Message> response = client.messages().withRawResponse().create(
```

```
    MessageCreateParams.builder()
        .model(Model.CLAUDE_SONNET_4_6)
        .maxTokens(1024)
        .addUserMessage("Hello!")
        .build()
);

IO.println(response.headers().values("x-amzn-requestid")); // AWS request ID
IO.println(response.requestId()); // Anthropic request ID
IO.println(response.parse().content());
```

PHP

```
use Anthropic\Aws\Client;

$client = new Client();

$response = $client->messages->raw->create(
    model: 'claude-sonnet-4-6',
    maxTokens: 1024,
    messages: [['role' => 'user', 'content' => 'Hello!']],
);

echo $response->getHeaderLine('x-amzn-requestid') . "\n"; // AWS request ID
echo $response->getHeaderLine('request-id') . "\n"; // Anthropic request ID
echo $response->parse();
```

Ruby

The Ruby SDK does not currently expose a raw-response accessor, so request IDs cannot be read from the response headers in Ruby. If you need request-ID logging, use any of the other languages above, or capture the IDs at the HTTP proxy layer.

Anthropic recommends logging your activity on at least a 30-day rolling basis to understand usage patterns and investigate any potential issues.

> **Note**
>
> AWS CloudTrail is configured within your AWS account. Enabling logging does not provide AWS or Anthropic access to your content beyond what is necessary for billing and service operation.

Usage metrics and dashboards

Claude Platform on AWS does not publish per-workspace usage metrics to Amazon CloudWatch. Token counts, request volume, and per-model usage are instead available through Anthropic's usage surfaces:

- **Claude Console usage views** — when a principal federates into the Claude Console with `aws-external-anthropic:AssumeConsole` (see IAM policies), the usage dashboards show request volume, token counts, and per-model breakdowns for the workspaces the principal has access to. Account-wide usage views require admin console capability.
- **Anthropic Usage and Cost API** — for programmatic access to usage and cost data, including per-workspace and per-model aggregation, see Usage and Cost API in the Anthropic documentation.

For operational monitoring (error rates, latency, rate limit headers), use the response headers returned on every request (see Rate limits and quotas) along with the AWS request IDs captured in CloudTrail.

Cost allocation and monitoring

Claude Platform on AWS charges appear on your AWS bill under the **Claude Platform on AWS** service, with per-model usage dimensions. Use the AWS Cost and Usage Report (CUR) combined with workspace tags for fine-grained cost allocation:

1. Tag your workspaces with the allocation dimensions you care about (team, application, environment, cost center). See Workspaces for tagging details.
2. In the AWS Billing console, activate each tag key as a **cost allocation tag**. After activation, usage incurred on or after the activation date is split by tag in the CUR.
3. In CUR or AWS Cost Explorer, group by the `resourceTags/user:<tag-key>` column to break down spend by workspace tag.

Workspace tags flow through to CUR line items for all Claude Platform on AWS usage, so the same tag keys can drive both IAM-based access control and cost allocation without additional configuration. See Cost allocation tags in the AWS Billing documentation for setup steps and activation delays.

Migrating from Amazon Bedrock

If you currently use Claude on Bedrock, migrating to Claude Platform on AWS requires changes throughout your integration. SigV4 signing remains supported, but the signing context, base URL, API format, model IDs, SDK client and package, streaming format, request headers, and region availability all change. The following table summarizes the differences.

What changes

	Amazon Bedrock	Claude Platform on AWS
Base URL	`bedrock-runtime.{region}.amazonaws.com`	`aws-external-anthropic.{region}.api.aws`
API format	Bedrock Converse / InvokeModel	Anthropic Messages API (`/v1/messages`)
Model IDs	`anthropic.claude-opus-4-7-v1`	`claude-opus-4-7`
SDK client	`AnthropicBedrock` / Bedrock SDK	Platform-specific client (`AnthropicAWS`, `AnthropicAws`, `AnthropicAwsClient`, etc.), in beta
SDK package	`anthropic[bedrock]`, `@anthropic-ai/bedrock-sdk`, etc.	`anthropic[aws]`, `@anthropic-ai/aws-sdk`, etc. (see Install an SDK)
SigV4 service name	`bedrock`	`aws-external-anthropic`
Streaming format	AWS EventStream	SSE (same as Claude API)
Workspace header	Not applicable	`anthropic-workspace-id` required

	Amazon Bedrock	Claude Platform on AWS
Region availability	Multiple commercial regions	All AWS commercial regions (opt-in regions require account opt-in)

What you gain

- Typically same-day access to new models and features, without a separate partner integration step
- Agent Skills for document generation (PowerPoint, Excel, Word, PDF)
- Code execution in Anthropic's managed sandbox
- Beta features through the `anthropic-beta` header (see Features not currently available)
- Claude Console for quota visibility and usage analytics
- Direct Anthropic support
- API key authentication as an alternative to SigV4 (see Authentication)

What stays the same

- AWS IAM authentication (SigV4)
- Billing through your AWS account
- AWS commitment retirement

Migration pitfalls

Warning

Enable outbound web identity federation first. If your AWS account has not previously used Claude Platform on AWS, you must enable outbound web identity federation once per account before making requests. Without this step, all requests fail with a federation error. This step is not required for Bedrock.

Warning

Zero Data Retention (ZDR) is opt-in on Claude Platform on AWS. On Bedrock, AWS is the data processor and Anthropic does not retain inference inputs or outputs; Anthropic's ZDR program does not apply there. On Claude Platform on AWS, Anthropic is the data processor, and ZDR follows the first-party Claude API model: it is available on request through your Anthropic account representative. Confirm ZDR enrollment before migrating production workloads that depend on data-retention guarantees.

Commercial considerations

- **Terms of service:** Use of Claude Platform on AWS is governed by the AWS Service Terms and is subject to Anthropic's Commercial Terms of Service, Data Processing Addendum, Usage Policy, and other agreements with Anthropic governing your use of their services.
- **Discounts and private offers:** Negotiated discounts and AWS Marketplace private offers don't transfer automatically between Bedrock and Claude Platform on AWS. Work with your Anthropic account representative to set up commercial terms for Claude Platform on AWS.

IAM policies

Claude Platform on AWS integrates with AWS IAM for access control. You grant or deny access to specific API actions on specific workspaces using standard IAM policy syntax.

The SigV4 service name and IAM action namespace is `aws-external-anthropic`. Actions follow the pattern `aws-external-anthropic:<Action>` (for example, `aws-external-anthropic:CreateInference`).

Example: deny batch inference

The following policy allows real-time inference while blocking batch processing, a common requirement for ZDR-sensitive workloads:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-external-anthropic:CreateInference",
        "aws-external-anthropic:CountTokens",
        "aws-external-anthropic:GetModel",
        "aws-external-anthropic:ListModels",
        "aws-external-anthropic:GetWorkspace",
        "aws-external-anthropic:ListWorkspaces"
      ],
      "Resource": "arn:aws:aws-external-anthropic:*:*:workspace/*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "aws-external-anthropic:CreateBatchInference",
        "aws-external-anthropic:GetBatchInference",
        "aws-external-anthropic:ListBatchInferences"
      ],
      "Resource": "*"
    }
  ]
}
```

The `GetBatchInference` action authorizes both the batch metadata route and the batch results route. Denying it, alongside `ListBatchInferences`, blocks both reads and batch enumeration.

The `Allow` statement enumerates specific `Get*` and `List*` actions rather than using wildcards. Wildcards would grant `GetFile` (which downloads file bytes) and other reads you may not intend; `Deny` overrides `Allow` regardless, but the explicit form is the safer pattern to model.

Example: synchronous inference on a single workspace

Grants the minimal permissions for an IAM principal that runs inference against one production workspace:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-external-anthropic:CreateInference",
        "aws-external-anthropic:CountTokens",
        "aws-external-anthropic:Get*",
        "aws-external-anthropic:List*"
      ],
      "Resource": "arn:aws:aws-external-anthropic:us-west-2:123456789012:workspace/
wrkspc_01AbCdEf23GhIj"
    }
  ]
}
```

Note

The `List*` wildcard in this policy also matches `ListWorkspaces`, which is account-scoped. The workspace ARN constraint silently filters it out, so this policy does not authorize listing workspaces. If your service account needs to enumerate workspaces, add a separate `Allow` statement for `ListWorkspaces` with `Resource: "*"`.
This policy assumes AWS SigV4 authentication. If the principal authenticates with an API key, also grant `aws-external-anthropic:CallWithBearerToken` (see Authentication).

Example: per-customer workspace isolation

Restricts a role to a single workspace:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "aws-external-anthropic:*",
      "Resource": "arn:aws:aws-external-anthropic:us-west-2:123456789012:workspace/
wrkspc_01AbCdEf23GhIj"
    },
    {
      "Effect": "Allow",
      "Action": [
        "aws-external-anthropic:CallWithBearerToken",
        "aws-external-anthropic:AssumeConsole"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

The `aws-external-anthropic:*` wildcard in the first statement includes account-scoped actions (`CreateWorkspace`, `ListWorkspaces`) that the workspace ARN constraint silently filters out. This is consistent with the "isolation" intent — the role cannot create or enumerate workspaces — but the policy contains permissions that have no effect. See Provisioning automation for the account-scoped pattern.

The second statement grants `CallWithBearerToken` and `AssumeConsole` on all resources because both are route-less actions that don't bind to a workspace ARN. Omit the second statement if the role uses SigV4 only and never federates to the Claude Console.

Example: feature lockdown for a ZDR-sensitive workspace

Blocks batch processing and file upload on a specific workspace while leaving synchronous inference available. Useful when a workspace handles Zero Data Retention (ZDR) data

that must not persist server-side. Attach this policy alongside an Allow policy such as `AnthropicLimitedAccess` or the single-workspace example above; on its own, a Deny-only policy grants no permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "aws-external-anthropic:CreateBatchInference",
        "aws-external-anthropic:CreateFile"
      ],
      "Resource": "arn:aws:aws-external-anthropic:us-west-2:123456789012:workspace/
wrkspc_01AbCdEf23GhIj"
    }
  ]
}
```

Note

This deny blocks creation only. Other file and batch actions are not denied unless you list them as well. For a complete lockdown where the workspace must never hold files or batches, also deny `aws-external-anthropic:GetFile`, `aws-external-anthropic:ListFiles`, `aws-external-anthropic:DeleteFile`, `aws-external-anthropic:GetBatchInference`, `aws-external-anthropic:ListBatchInferences`, `aws-external-anthropic:CancelBatchInference`, and `aws-external-anthropic:DeleteBatchInference`.

Example: provisioning automation

Grants a CI/CD role the actions needed to create and manage workspaces, without any inference permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "aws-external-anthropic:CreateWorkspace",
        "aws-external-anthropic:GetWorkspace",
        "aws-external-anthropic:ListWorkspaces",
        "aws-external-anthropic:UpdateWorkspace",
        "aws-external-anthropic:ArchiveWorkspace"
      ],
      "Resource": "*"
    }
  ]
}
```

`CreateWorkspace` and `ListWorkspaces` are account-scoped operations. Specifying a workspace ARN on these actions has no effect; use `Resource: "*"`.

Managed policies

AWS provides managed policies for common access patterns:

- **`AnthropicFullAccess`:** Grants `aws-external-anthropic:*` on all resources.
- **`AnthropicReadOnlyAccess`:** Grants `Get*`, `List*`, and `CallWithBearerToken` on all resources.
- **`AnthropicInferenceAccess`:** Grants the ReadOnly actions plus the inference actions (`CreateInference`, `CreateBatchInference`, `CancelBatchInference`, `DeleteBatchInference`, `CountTokens`) on all resources.
- **`AnthropicLimitedAccess`:** Grants the `AnthropicInferenceAccess` actions plus all Claude Managed Agents actions (agents, sessions, environments, vaults, memory stores) on all resources.

`AnthropicInferenceAccess` is the narrowest managed policy sufficient to run inference. Through the `Get*` and `List*` wildcards it grants read access to every API resource in the namespace, including file content download through `GetFile` and memory contents through `GetMemoryStore`. It does not grant file or skill creation or deletion, user profile management, workspace mutation, or console federation.

Note

`AnthropicReadOnlyAccess`, `AnthropicInferenceAccess`, and `AnthropicLimitedAccess` do not grant `AssumeConsole`. Principals who need to federate to the Claude Console require a separate grant for `aws-external-anthropic:AssumeConsole` — either through `AnthropicFullAccess` or a custom policy. See Federating to the Claude Console.

Note

`CreateInference` and `CreateBatchInference` are separate actions. Denying one does not block the other. If you intend to prevent all model calls, deny both.

Federating to the Claude Console

`aws-external-anthropic:AssumeConsole` lets an IAM principal federate into the Anthropic-operated Claude Console. Access within the console is still governed by IAM for most operations, but a subset of admin operations — primarily usage views that have no corresponding IAM API — are gated on the capability the principal federated with.

Two capabilities exist:

- **`developer`** — permits the operations a Claude Platform on AWS developer needs for day-to-day work: running inference from the console, reading workspace data, viewing personal usage.
- **`admin`** — additionally permits admin-only console operations, including account-wide usage views and administrative settings that are not surfaced through IAM actions.

Control which capability a principal can request with the `aws-external-anthropic:Capability` condition key on the `AssumeConsole` action:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "aws-external-anthropic:AssumeConsole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws-external-anthropic:Capability": "admin"
        }
      }
    }
  ]
}
```

`AnthropicFullAccess` grants `AssumeConsole` with no capability restriction. For any narrower grant — developer-only console access, or admin-only access — attach a custom policy that scopes `AssumeConsole` with the `aws-external-anthropic:Capability` condition as shown above.

Calling with bearer tokens

`aws-external-anthropic:CallWithBearerToken` authorizes the SigV4-free request path used when an API key is presented as a bearer token. Any principal that will authenticate with an API key — whether through the Anthropic SDK's `ANTHROPIC_AWS_API_KEY` environment variable or by setting the `Authorization: Bearer` header directly — needs this action on the target workspace.

This is required for API key callers in addition to the inference actions (`CreateInference`, `CreateBatchInference`, etc.). Without `CallWithBearerToken`, API key requests are rejected before reaching the inference authorization check. SigV4 callers do not need this action.

`AnthropicReadOnlyAccess`, `AnthropicInferenceAccess`, `AnthropicLimitedAccess`, and `AnthropicFullAccess` all include `CallWithBearerToken`. If you write a custom policy for API key access, add it explicitly.

Outbound web identity federation (required for console access)

The Claude Console runs in Anthropic infrastructure, not AWS. When an IAM principal calls `AssumeConsole`, AWS STS issues a web identity token scoped to the Anthropic audience; the Anthropic console then accepts that token and establishes the federated session. For this to work, the AWS account must allow outbound web identity federation to the `aws-external-anthropic` audience.

Principals that call `AssumeConsole` need the following STS permissions in addition to the `aws-external-anthropic:AssumeConsole` action:

- **`sts:GetWebIdentityToken`** — permits issuing the web identity token that the Anthropic console consumes.
- **`sts:TagGetWebIdentityToken`** — permits attaching session tags to the web identity token. Claude Platform on AWS uses these tags to convey the principal's capability and workspace context to the console.

Include both in the trust policy of any role that console users assume, or in the inline/managed policy attached to user identities that call `AssumeConsole` directly. `AnthropicFullAccess` includes both STS actions. Environments that deny `sts:*` at the SCP or permission boundary level must explicitly allow these two actions for console federation to succeed.

IAM actions for Claude Platform on AWS

IAM action reference for controlling access to Claude Platform on AWS through AWS policies.

Claude Platform on AWS uses AWS IAM for access control. Every API route maps to an IAM action in the `aws-external-anthropic` namespace. This page lists all actions, the routes each action authorizes, and the managed policies available for common access patterns. For platform setup and authentication, see What is Claude Platform on AWS?.

Service details

IAM service prefix	`aws-external-anthropic`
Resource types	`workspace`
Workspace ARN	`arn:aws:aws-external-anthropic:{region}:{account-id}:workspace/{workspace-id}`

The ARN region is always populated and matches the region the workspace is bound to. The resource segment is the tagged workspace ID (`wrkspc_...`), the same value you pass in the `anthropic-workspace-id` header.

Actions

The service defines 58 actions across 14 groups. Actions follow the AWS `VerbNoun` convention and use verb discipline so that `Get*` and `List*` wildcards produce a clean read-only boundary.

Inference

Action	Routes authorized
`CreateInference`	`POST /v1/messages`
`CountTokens`	`POST /v1/messages/count_tokens`

Batch processing

Action	Routes authorized
CreateBatchInference	POST /v1/messages/batches
GetBatchInference	GET /v1/messages/batches/{id} GET /v1/messages/batches/{id}/results
ListBatchInferences	GET /v1/messages/batches
CancelBatchInference	POST /v1/messages/batches/{id}/cancel
DeleteBatchInference	DELETE /v1/messages/batches/{id}

Models

Action	Routes authorized
GetModel	GET /v1/models/{id}
ListModels	GET /v1/models

Files

Action	Routes authorized
CreateFile	POST /v1/files
GetFile	GET /v1/files/{id} GET /v1/files/{id}/content
ListFiles	GET /v1/files
DeleteFile	DELETE /v1/files/{id}

Note

`GetFile` authorizes both metadata and content download. A principal with read-only access can download file bytes, not just list files.

Skills

Action	Routes authorized
`CreateSkill`	`POST /v1/skills`
`GetSkill`	`GET /v1/skills/{id}` `GET /v1/skills/{id}/versions` `GET /v1/skills/{id}/versions/{version}`
`ListSkills`	`GET /v1/skills`
`UpdateSkill`	`POST /v1/skills/{id}/versions` `DELETE /v1/skills/{id}/versions/{version}`
`DeleteSkill`	`DELETE /v1/skills/{id}`

Note

Deleting an individual skill version maps to `UpdateSkill`, not `DeleteSkill`. A policy that denies `aws-external-anthropic:Delete*` still permits version deletion. Deny `UpdateSkill` and `CreateSkill` as well if you need to prevent any skill mutation.

User profiles

Action	Routes authorized
`CreateUserProfile`	`POST /v1/user_profiles`
`GetUserProfile`	`GET /v1/user_profiles/{id}`

Action	Routes authorized
`ListUserProfiles`	`GET /v1/user_profiles`
`UpdateUserProfile`	`POST /v1/user_profiles/{id}`

> **Warning**
>
> IAM action matching is case-insensitive. The wildcard `aws-external-anthropic:*File` matches `CreateFile`, `GetFile`, and `DeleteFile`, but does not match `ListFiles` (which ends in "files", not "file"). It also over-matches `CreateUserProfile`, `GetUserProfile`, and `UpdateUserProfile` because "Profile" ends in "file". If you intend to grant or deny only Files API actions, enumerate them explicitly (`CreateFile`, `GetFile`, `ListFiles`, `DeleteFile`) rather than using a `*File` suffix pattern.

Agents

Agent definitions for Claude Managed Agents.

Action	Routes authorized
`CreateAgent`	Agent create routes
`GetAgent`	Agent read routes
`ListAgents`	Agent list routes
`UpdateAgent`	Agent update routes
`ArchiveAgent`	Agent archive routes

Sessions

Agent sessions and event history.

Action	Routes authorized
`CreateSession`	Session create routes
`GetSession`	Session read routes
`ListSessions`	Session list routes
`UpdateSession`	Session update routes
`ArchiveSession`	Session archive routes
`DeleteSession`	Session delete routes

Environments

Cloud container configurations for sessions.

Action	Routes authorized
`CreateEnvironment`	Environment create routes
`GetEnvironment`	Environment read routes
`ListEnvironments`	Environment list routes
`UpdateEnvironment`	Environment update routes
`ArchiveEnvironment`	Environment archive routes
`DeleteEnvironment`	Environment delete routes

Vaults

Credential vaults for session authentication.

Action	Routes authorized
`CreateVault`	Vault create routes
`GetVault`	Vault read routes
`ListVaults`	Vault list routes
`UpdateVault`	Vault update routes
`ArchiveVault`	Vault archive routes
`DeleteVault`	Vault delete routes

> **Note**
>
> Vault operations are classified as CloudTrail Management events (rather than Data events) because vaults hold credentials and benefit from default-on audit logging. Other Claude Managed Agents operations (agents, sessions, environments, memory stores) are Data events.

Memory stores

Persistent agent memory.

Action	Routes authorized
`CreateMemoryStore`	Memory store create routes
`GetMemoryStore`	Memory store read routes
`ListMemoryStores`	Memory store list routes
`UpdateMemoryStore`	Memory store update routes
`ArchiveMemoryStore`	Memory store archive routes

Action	Routes authorized
DeleteMemoryStore	Memory store delete routes

> **Note**
>
> GetMemoryStore reads memory contents. The Get* wildcard in a managed or custom policy therefore grants memory read access. Scope policies explicitly if memory contents must be restricted.

Workspaces

Action	Routes authorized
CreateWorkspace	POST /v1/organizations/workspaces
GetWorkspace	GET /v1/organizations/workspaces/{id}
ListWorkspaces	GET /v1/organizations/workspaces
UpdateWorkspace	POST /v1/organizations/workspaces/{id}
ArchiveWorkspace	POST /v1/organizations/workspaces/{id}/archive

A default workspace is provisioned at sign-up; see Workspaces.

Authentication

Action	Routes authorized
CallWithBearerToken	(none)

`CallWithBearerToken` is an auth-layer permission that authorizes a principal to authenticate through an API key (bearer token) rather than AWS SigV4. It does not map to a route. Grant it alongside the route-mapped actions you want the API key holder to perform.

Console access

Action	Routes authorized
`AssumeConsole`	(none)

`AssumeConsole` authorizes a principal to open the Claude Console for a Claude Platform on AWS workspace through the AWS Console federation flow. It does not map to a route. Grant it to principals who should be able to click **Open Claude Console** on the Claude Platform on AWS service page in the AWS Console. The console capability (developer or admin) is controlled by the `aws-external-anthropic:Capability` condition key on the `AssumeConsole` action; see IAM policies for details on the capability model and Using the Claude Console for the sign-in flow.

Warning

`aws-external-anthropic:AssumeConsole` issues a Claude Console session that lasts up to 12 hours, independent of the caller's own session duration. A caller whose IAM session is shorter than 12 hours can still obtain a console session that outlives their source credentials. Restrict this permission to principals who require Claude Console access, and revoke it promptly when no longer needed.

Note

Actions performed inside the Claude Console after federation are not recorded in AWS CloudTrail or attributable through IAM. This includes global, workspace-scoped activities such as viewing usage reports. If you need an audit trail for console activity, use the audit logs available in the Claude Console rather than CloudTrail.

Route-to-action mapping

The following table lists every route on Claude Platform on AWS and the IAM action required to call it. The stable route's IAM action also authorizes requests that use the `anthropic-beta` header. CloudTrail classifies each action as either a Data event (high-volume, data-plane operations) or a Management event (control-plane operations).

Method	Route	IAM action	CloudTrail event type
POST	/v1/messages	CreateInference	Data
POST	/v1/messages/count_tokens	CountTokens	Data
POST	/v1/messages/batches	CreateBatchInference	Data
GET	/v1/messages/batches	ListBatchInferences	Data
GET	/v1/messages/batches/{id}	GetBatchInference	Data
GET	/v1/messages/batches/{id}/results	GetBatchInference	Data
POST	/v1/messages/batches/{id}/cancel	CancelBatchInference	Data
DELETE	/v1/messages/batches/{id}	DeleteBatchInference	Data
GET	/v1/models	ListModels	Data
GET	/v1/models/{id}	GetModel	Data
POST	/v1/files	CreateFile	Data
GET	/v1/files	ListFiles	Data
GET	/v1/files/{id}	GetFile	Data

Method	Route	IAM action	CloudTrail event type
GET	/v1/files/{id}/content	GetFile	Data
DELETE	/v1/files/{id}	DeleteFile	Data
POST	/v1/skills	CreateSkill	Data
GET	/v1/skills	ListSkills	Data
GET	/v1/skills/{id}	GetSkill	Data
DELETE	/v1/skills/{id}	DeleteSkill	Data
POST	/v1/skills/{id}/versions	UpdateSkill	Data
GET	/v1/skills/{id}/versions	GetSkill	Data
GET	/v1/skills/{id}/versions/{version}	GetSkill	Data
DELETE	/v1/skills/{id}/versions/{version}	UpdateSkill	Data
POST	/v1/user_profiles	CreateUserProfile	Data
GET	/v1/user_profiles	ListUserProfiles	Data
GET	/v1/user_profiles/{id}	GetUserProfile	Data
POST	/v1/user_profiles/{id}	UpdateUserProfile	Data
POST	/v1/organizations/workspaces	CreateWorkspace	Management
GET	/v1/organizations/workspaces	ListWorkspaces	Management
GET	/v1/organizations/workspaces/{id}	GetWorkspace	Management

Method	Route	IAM action	CloudTrail event type
POST	/v1/organizations/workspaces/{id}	UpdateWorkspace	Management
POST	/v1/organizations/workspaces/{id}/archive	ArchiveWorkspace	Management

Claude Managed Agents routes (agents, sessions, environments, vaults, memory stores) follow the same route-to-action pattern; for the complete per-route mapping, see the Anthropic IAM actions reference. Vault routes are Management events; agent, session, environment, and memory store routes are Data events.

Routes not on Claude Platform on AWS are denied at the gateway by default.

See also

- IAM policies for example policies and managed policies
- AWS IAM User Guide for IAM policy syntax and evaluation logic
- AWS CloudTrail User Guide for audit logging configuration

Document history

The following table describes important changes to the *Claude Platform on AWS User Guide*.

Change	Description	Date
Initial publication	Published the Claude Platform on AWS User Guide.	May 7, 2026