Artificial Intelligence

Enforcing data residency with single-Region Claude Code on Amazon Bedrock

by Zaid Barkat | on 06 AUG 2026 | in Advanced (300), Amazon Bedrock, Technical How-to | Permalink | Share

A US-headquartered global organization recently came to us with a deceptively simple data-residency request: let their engineers use Claude Code. The requirement: Amazon Bedrock model inference had to be processed in London (the eu-west-2 AWS Region), not merely called from London. Their compliance team had drawn a hard line. Prompts, completions, and intermediate processing were required to stay inside a single AWS Region. Approximate compliance wasn't on the table.

We tried the newest path first (Anthropic's Mantle endpoint on Amazon Bedrock) and encountered a limitation. Then we went back to the classic Amazon Bedrock Invoke API with an application inference profile. That combination, plus an AWS Identity and Access Management (IAM) Region condition, satisfied the requirement.

In this post, we show you both paths, why the choice depends on your Region, the AWS Identity and Access Management (IAM) policy that enforces them, and how to verify compliance in AWS CloudTrail. The pattern applies to tools that target an Amazon Bedrock model ID.

Important context: for most Amazon Bedrock workloads, cross-Region inference (CRIS) is the right default. It smooths throughput, increases the capacity available to your application, and gives you access to newer models first. Use the single-Region patterns in this post only when your compliance requirement is a specific AWS Region rather than a geography. If "somewhere in the EU" satisfies your data residency need, an EU cross-Region profile is the more straightforward answer.

Prerequisites

Before you begin, confirm that you have:

- An AWS account with Amazon Bedrock model access enabled for the Claude models that you intend to use.
- IAM permissions for the identity Claude Code runs as, which differ by path:
  - Path 1 (Mantle): `bedrock-mantle:CreateInference`, `bedrock-mantle:Get*`, and `bedrock-mantle:List*`.
  - Path 2 (classic Amazon Bedrock): `bedrock:CreateInferenceProfile`, `bedrock:InvokeModel`, and `bedrock:InvokeModelWithResponseStream`.
  - Both paths: `iam:CreatePolicy`, `iam:AttachRolePolicy`, and `iam:PutRolePolicy` to attach the Region-scoped policy to that role or user.
- Claude Code v2.1.94 or later (Path 1 requires Mantle support, which was added in v2.1.94).
- AWS Command Line Interface (AWS CLI) installed and configured with credentials for the target Region.

Two endpoints, two paths

Amazon Bedrock exposes Claude models through two distinct endpoints. The one that you use determines how you enforce data residency. It also determines which Regions you can enforce it in.

Classic Amazon Bedrock (bedrock-runtime) is the original Amazon Bedrock Invoke API. Claude Code uses it when you set CLAUDE_CODE_USE_BEDROCK=1. Single-Region routing requires an inference profile. By default, the system-defined inference profiles are cross-Region. To keep a request in one Region, you create an application inference profile that points at the in-Region foundation model.

Mantle (bedrock-mantle) is a newer Amazon Bedrock endpoint that serves Claude through Anthropic's native API shape. Claude Code uses it when you set CLAUDE_CODE_USE_MANTLE=1. Single-Region routing is native. You set AWS_REGION, and Mantle resolves the endpoint to that Region directly with no application inference profile needed.

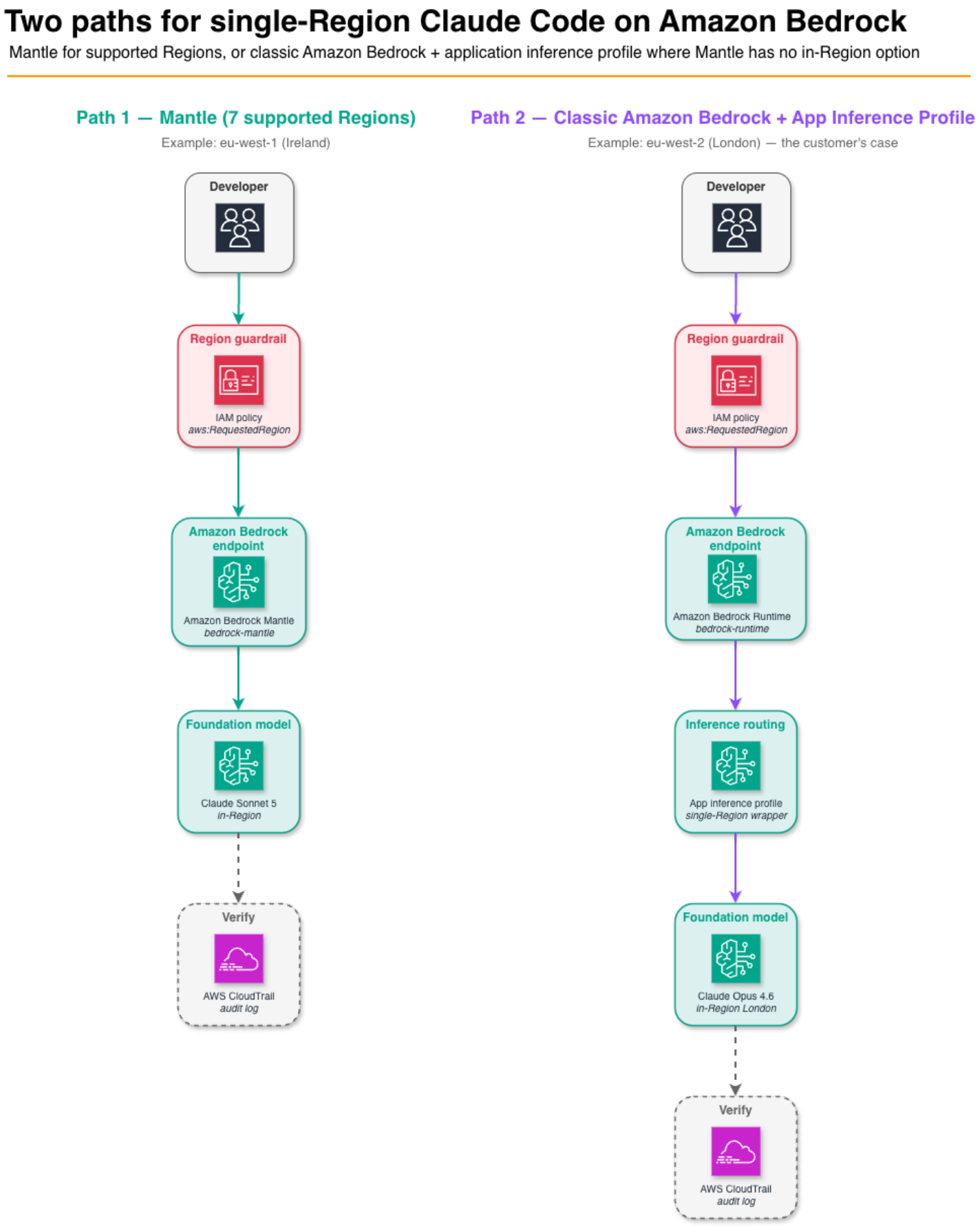

Figure 1: Two paths for single-Region Claude Code

However, each endpoint has its own Regional availability and model lineup:

	Classic Amazon Bedrock	Mantle
Regions with in-Region Claude support	**eu-west-2 (London) only** (for Claude Opus 4.6, Sonnet 4.6)	Ireland, Stockholm, Tokyo, Melbourne, US East (N. Virginia), US East (Ohio), US West (Oregon)
Model IDs	`anthropic.claude-opus-4-6-v1`, `anthropic.claude-sonnet-4-6`	`anthropic.claude-sonnet-5`, `anthropic.claude-opus-4-8`, `anthropic.claude-haiku-4-5`
Single-Region mechanism	Application inference profile + IAM condition	`AWS_REGION` + IAM condition

Claude Code can run both endpoints in the same session (set both env vars). Model IDs starting with `anthropic.` and no `us.` prefix route to Mantle. Everything else goes to classic Invoke.

Your Region drives the choice. Targeting London (eu-west-2)? Use Path 2. Mantle offers no in-Region routing there, and classic Amazon Bedrock supports in-Region inference for Claude Opus 4.6 and Sonnet 4.6. Targeting one of Ireland, Stockholm, Tokyo, Melbourne, US East (N. Virginia), US East (Ohio), or US West (Oregon)? Use Path 1. Mantle's native routing works there and gives you access to newer models with less setup. Targeting a different Region? Neither pattern works today. Escalate to your AWS account team.

For model availability by Region, refer to Supported models by AWS Region in Amazon Bedrock.

Path 1: Mantle for a supported single Region

If your compliance requirement points at one of Mantle's seven in-Region Regions (Ireland, Stockholm, Tokyo, Melbourne, US East (N. Virginia), US East (Ohio), or US West (Oregon)), Mantle is the more straightforward path. You get direct single-Region routing without creating any inference profiles, and access to newer models including Claude Sonnet 5.

Configure Claude Code to use Mantle and pin it to your target Region:

```
# Route Claude Code through the Mantle endpoint

# Ireland, an in-Region Mantle Region
export CLAUDE_CODE_USE_MANTLE=1
export AWS_REGION=eu-west-1
# Pin the model family aliases to Mantle model IDs
export ANTHROPIC_DEFAULT_OPUS_MODEL='anthropic.claude-opus-4-8'
export ANTHROPIC_DEFAULT_SONNET_MODEL='anthropic.claude-sonnet-5'
export ANTHROPIC_DEFAULT_HAIKU_MODEL='anthropic.claude-haiku-4-5'
```

Run the `claude` command and verify the `/status` output. The provider should read `Amazon Bedrock (Mantle)` and the Region should match. Mantle is available in Claude Code v2.1.94 and later.

Setup summary: three environment variables, no additional AWS resources to provision beyond the IAM policy.

Claude Code has no credentials of its own. Instead, it signs Amazon Bedrock calls with whatever AWS credentials are already in the developer's environment. Attach this policy to that IAM principal: the role your developers assume (for example, through AWS IAM Identity Center) or the IAM user behind their local profile. It permits Amazon Bedrock calls only in your target Region:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowClaudeCodeMantleInIrelandOnly",
      "Effect": "Allow",
      "Action": [
        "bedrock-mantle:CreateInference",
        "bedrock-mantle:Get*",
        "bedrock-mantle:List*"
      ],
      "Resource": "arn:aws:bedrock-mantle:eu-west-1:111122223333:project/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "eu-west-1"
        }
      }
    }
  ]
```

Replace `111122223333` with your account ID. The `aws:RequestedRegion` condition rejects any Mantle call not made against the Ireland endpoint, so even if a developer misconfigures `AWS_REGION`, IAM refuses the call. That is the Region guardrail this post is about. Note that model pinning works differently on Mantle: the resource is a project, not a model, so the identity policy controls only Region. Your model choice comes from the `ANTHROPIC_DEFAULT_*_MODEL` variables. To enforce an approved model list, use a service control policy. That is the asymmetry with Path 2, where the foundation-model Amazon Resource Name (ARN) lets one identity policy pin both Region and models.

Path 2: Application inference profile for Regions Mantle doesn't cover

Our customer's compliance requirement was London specifically, and Mantle's Regional availability table shows why that matters. **eu-west-2 offers only Global and EU endpoints on Mantle, with no in-Region-only option.** The EU endpoint routes across the whole EU geography, so a request from London can be processed in Frankfurt, Ireland, or Paris. That is exactly what the customer needed to prevent.

Classic Amazon Bedrock is the only option for London. Additionally, there is a constraint: in-Region availability on classic Amazon Bedrock is narrow.

Per the Claude Opus 4.6 and Claude Sonnet 4.6 model cards, eu-west-2 is currently the only Region where these models offer in-Region inference. Opus 4.7 and 4.8 don't. They are Geo-only on classic Amazon Bedrock. So, the London deployment is anchored on Claude Opus 4.6 and Sonnet 4.6.

Some foundation models can't be called directly by their model ID because Amazon Bedrock requires you to invoke them through an inference profile instead. So, when you point Claude Code straight at one of these model IDs, the call fails. Amazon Bedrock returns an error saying on-demand throughput isn't supported and asks you to retry with the ID or ARN of an inference profile. But the system-defined inference profiles are the cross-Region ones (`eu.` and `global.` prefixes), which is exactly what we are trying to avoid.

The solution is an application inference profile. This is a profile you create with a foundation-model ARN in one Region as its model source. Point it at the London foundation model ARN and you get a profile-shaped ARN that Claude Code accepts and that resolves to a single-Region model.

Create one profile per model family:

```
aws bedrock create-inference-profile \
  --region eu-west-2 \
  --inference-profile-name "claude-code-opus-london" \
  --model-source copyFrom="arn:aws:bedrock:eu-west-2::foundation-model/anthropic.claude-opus-4-6-v1"
aws bedrock create-inference-profile \
  --region eu-west-2 \
  --inference-profile-name "claude-code-sonnet-london" \
  --model-source copyFrom="arn:aws:bedrock:eu-west-2::foundation-model/anthropic.claude-sonnet-4-6"
```

Each call returns an `inferenceProfileArn` like `arn:aws:bedrock:eu-west-2:<account-id>:application-inference-profile/<id>` . Save both. Claude Code uses them as models. Application inference profiles also give you per-profile cost and usage tracking in your AWS bill.

Wire Claude Code to the profile ARNs:

```
# Route Claude Code through classic Bedrock in London
export CLAUDE_CODE_USE_BEDROCK=1
export AWS_REGION=eu-west-2
# Map the model aliases to the London application inference profiles
export ANTHROPIC_DEFAULT_OPUS_MODEL='arn:aws:bedrock:eu-west-2:<account-id>:application-inference
export ANTHROPIC_DEFAULT_SONNET_MODEL='arn:aws:bedrock:eu-west-2:<account-id>:application-inferer
```

For a team rollout, set these in a managed settings file rather than each engineer's shell, so the configuration is consistent and centrally controlled.

Setup summary: 2 application inference profile creations (one per model family), four environment variables (`CLAUDE_CODE_USE_BEDROCK` , `AWS_REGION` , and the two `ANTHROPIC_DEFAULT_*_MODEL` variables), plus the IAM policy. Roughly 5 minutes of additional provisioning per model family versus Path 1.

The IAM policy looks similar to Path 1, but the `Resource` list now includes the application inference profile ARNs, their backing foundation-model ARNs, and the IAM actions for Amazon Bedrock:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowClaudeCodeInLondonOnly",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream",
        "bedrock:GetInferenceProfile",
        "bedrock:ListInferenceProfiles"
      ],
      "Resource": [
        "arn:aws:bedrock:eu-west-2:<account-id>:application-inference-profile/<opus-id>",
        "arn:aws:bedrock:eu-west-2:<account-id>:application-inference-profile/<sonnet-id>",
        "arn:aws:bedrock:eu-west-2::foundation-model/anthropic.claude-opus-4-6-v1",
        "arn:aws:bedrock:eu-west-2::foundation-model/anthropic.claude-sonnet-4-6"
      ],
      "Condition": {
```

Verify end to end. Run `/status` in Claude Code and confirm the provider reads `Amazon Bedrock` and the Region is London.

Verifying single-Region compliance

Every Claude Code Amazon Bedrock call must appear in the target Region's AWS CloudTrail, and nowhere else. How you query for it depends on which path you deployed, because the two endpoints log under different event names, different event sources, and different CloudTrail event types.

Path 2 (classic Amazon Bedrock): Model invocations log as `InvokeModel` under the `bedrock.amazonaws.com` event source. Look them up in the event history:

```
aws cloudtrail lookup-events \
  --region eu-west-2 \
  --lookup-attributes AttributeKey=EventName,AttributeValue=InvokeModel
```

A compliant event contains three fields that matter:

```
{
  "eventName": "InvokeModel",
  "eventSource": "bedrock.amazonaws.com",
  "awsRegion": "eu-west-2",
  "requestParameters": {
    "modelId": "arn:aws:bedrock:eu-west-2:111122223333:application-inference-profile/opus-london"
  }
}
```

If no `InvokeModel` events appear, model-invocation logging might be off. Enable Amazon Bedrock data events on your trail and retry.

Path 1 (Mantle): Mantle logs inference as `CreateInference` under the `bedrock-mantle.amazonaws.com` event source, and it is a CloudTrail data event. Before you can verify anything, enable `bedrock-mantle` data events with an advanced event selector on a trail or event data store. Once logging is on, query the event data store in AWS CloudTrail Lake:

```
SELECT eventName, awsRegion, element_at(requestParameters, 'model') AS model
FROM <event-data-store-id>
WHERE eventSource = 'bedrock-mantle.amazonaws.com'
AND eventName = 'CreateInference'
```

A compliant Mantle event carries the target Region in `awsRegion` and the pinned model in `requestParameters.model`:

```
{
  "eventName": "CreateInference",
  "eventSource": "bedrock-mantle.amazonaws.com",
  "awsRegion": "eu-west-1",
  "requestParameters": {
    "model": "anthropic.claude-sonnet-5"
  }
}
```

Regardless of path, three checks prove compliance. First, every event's `awsRegion` must match your target Region. Second, rerunning the same query in every other Region should return zero results. Third, any `errorCode: AccessDenied` event should trace back to a deliberate test rather than a real client.

Choosing the right path

Compare the two paths:

	Path 1 (Mantle)	Path 2 (Classic Amazon Bedrock)
Regions with in-Region Claude support	7	1 (London only)
Models with in-Region support	Sonnet 5, Opus 4.8, Haiku 4.5	Opus 4.6, Sonnet 4.6
AWS resources to create	0	2 application inference profiles
Environment variables to set	3	4
Setup steps beyond IAM	Env vars only	2 CLI calls + env vars

Verify the current in-Region status on the Amazon Bedrock model cards before committing. Availability moves as new models land.

Clean up

If you want to remove the application inference profiles you created, delete them with the AWS CLI:

```
aws bedrock delete-inference-profile \
  --region eu-west-2 \
  --inference-profile-identifier <opus-profile-arn>
aws bedrock delete-inference-profile \
  --region eu-west-2 \
  --inference-profile-identifier <sonnet-profile-arn>
```

Application inference profiles carry no ongoing charge on their own. Model invocation is the source of cost. Removing unused profiles keeps your account organized and reduces the surface area for accidental invocation. Path 1 requires no cleanup beyond detaching the IAM policy, since it creates no additional AWS resources.

Considerations and next steps

Treat this as a point-in-time snapshot. Regional availability changes as Anthropic and AWS bring new models to more Regions on both endpoints. Before you roll out, confirm the current in-Region status and exact model IDs for your chosen models on their model cards.

Claude Code can run both endpoints in the same session. Set `CLAUDE_CODE_USE_BEDROCK=1` and `CLAUDE_CODE_USE_MANTLE=1` together, and model IDs matching the Mantle format route to Mantle while everything else routes to classic Invoke. That is useful during a migration or when different model families live on different endpoints.

Single-Region data residency for an agentic coding tool might appear to require custom infrastructure. It does not. Depending on your Region, choose either Mantle's native single-Region routing or a classic Amazon Bedrock application inference profile. Pair either with an IAM Region condition for a clean, auditable deployment of Claude Code.

To get started, open the Amazon Bedrock console, check the Regional availability tables for the models you need on both endpoints, and pick the path that fits your Region. Learn more about inference profiles, cross-Region inference, and Claude Code on Amazon Bedrock in their respective documentation.

Related content

- For enterprise-scale Claude Code deployment with OpenID Connect (OIDC) federation, IAM Identity Center integration, and observability, see the Claude Apps Gateway repository.
- For reference architectural patterns, look at the Guidance for Claude Code on Amazon Bedrock in the AWS Solutions Library, which includes an accompanying GitHub repository.
- For optimizing Claude Code cost and latency through prompt caching, see Supercharge your development with Claude Code and Amazon Bedrock prompt caching.

About the author

Zaid Barkat

Zaid is an Enterprise Solutions Architect on the AWS Government Technology team. He is a Claude Champion and an active member of the AI/ML Technical Field Community, focused on architectural patterns for agentic tools on AWS. Outside of work, he enjoys swimming and trying out new restaurants.