Providers

Amazon Bedrock

OpenClaw can use **Amazon Bedrock** models via pi-ai's **Bedrock Converse** streaming provider. Bedrock auth uses the **AWS SDK default credential chain**, not an API key.

What pi-ai supports

Provider: `amazon-bedrock`

API: `bedrock-converse-stream`

Auth: AWS credentials (env vars, shared config, or instance role)

Region: `AWS_REGION` or `AWS_DEFAULT_REGION` (default: `us-east-1`)

Automatic model discovery

If AWS credentials are detected, OpenClaw can automatically discover Bedrock models that support **streaming** and **text output**. Discovery uses `bedrock:ListFoundationModels` and is cached (default: 1 hour).

Config options live under `models.bedrockDiscovery`:

```
{
  models: {
    bedrockDiscovery: {
      enabled: true,
      region: "us-east-1",
      providerFilter: ["anthropic", "amazon"],
      refreshInterval: 3600,
      defaultContextWindow: 32000,
      defaultMaxTokens: 4096,
    },
  },
}
```

Notes:

- `enabled` defaults to `true` when AWS credentials are present.
- `region` defaults to `AWS_REGION` or `AWS_DEFAULT_REGION`, then `us-east-1`.
- `providerFilter` matches Bedrock provider names (for example `anthropic`).
- `refreshInterval` is seconds; set to `0` to disable caching.
- `defaultContextWindow` (default: `32000`) and `defaultMaxTokens` (default: `4096`) are used for discovered models (override if you know your model limits).

Setup (manual)

1. Ensure AWS credentials are available on the **gateway host**:

```
export AWS_ACCESS_KEY_ID="AKIA..."
export AWS_SECRET_ACCESS_KEY="..."
export AWS_REGION="us-east-1"
# Optional:
export AWS_SESSION_TOKEN="..."
export AWS_PROFILE="your-profile"
# Optional (Bedrock API key/bearer token):
export AWS_BEARER_TOKEN_BEDROCK="..."
```

2. Add a Bedrock provider and model to your config (no `apiKey` required):

```
{
  models: {
    providers: {
      "amazon-bedrock": {
        baseUrl: "https://bedrock-runtime.us-east-1.amazonaws.com",
        api: "bedrock-converse-stream",
        auth: "aws-sdk",
        models: [
          {
            id: "us.anthropic.claude-opus-4-6-v1:0",
            name: "Claude Opus 4.6 (Bedrock)",
            reasoning: true,
            input: ["text", "image"],
            cost: { input: 0, output: 0, cacheRead: 0, cacheWrite: 0 },
            contextWindow: 200000,
            maxTokens: 8192,
          },
        ],
      },
    },
  },
  agents: {
    defaults: {
      model: { primary: "amazon-bedrock/us.anthropic.claude-opus-4-6-v1:0" },
    },
  },
}
```

EC2 Instance Roles

When running OpenClaw on an EC2 instance with an IAM role attached, the AWS SDK will automatically use the instance metadata service (IMDS) for authentication. However, OpenClaw's credential detection currently only checks for environment variables, not IMDS credentials.

Workaround: Set `AWS_PROFILE=default` to signal that AWS credentials are available. The actual authentication still uses the instance role via

IMDS.

```
# Add to ~/.bashrc or your shell profile
export AWS_PROFILE=default
export AWS_REGION=us-east-1
```

Required IAM permissions for the EC2 instance role:

- `bedrock:InvokeModel`
- `bedrock:InvokeModelWithResponseStream`
- `bedrock:ListFoundationModels` (for automatic discovery)

Or attach the managed policy `AmazonBedrockFullAccess`.

Quick setup:

```
# 1. Create IAM role and instance profile
aws iam create-role --role-name EC2-Bedrock-Access \
  --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [{
      "Effect": "Allow",
      "Principal": {"Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }]
  }'

aws iam attach-role-policy --role-name EC2-Bedrock-Access \
  --policy-arn arn:aws:iam::aws:policy/AmazonBedrockFullAccess

aws iam create-instance-profile --instance-profile-name EC2-Bedrock-Access
aws iam add-role-to-instance-profile \
  --instance-profile-name EC2-Bedrock-Access \
  --role-name EC2-Bedrock-Access

# 2. Attach to your EC2 instance
aws ec2 associate-iam-instance-profile \
  --instance-id i-xxxxx \
  --iam-instance-profile Name=EC2-Bedrock-Access

# 3. On the EC2 instance, enable discovery
openclaw config set models.bedrockDiscovery.enabled true
openclaw config set models.bedrockDiscovery.region us-east-1

# 4. Set the workaround env vars
echo 'export AWS_PROFILE=default' >> ~/.bashrc
echo 'export AWS_REGION=us-east-1' >> ~/.bashrc
source ~/.bashrc

# 5. Verify models are discovered
openclaw models list
```

Notes

Bedrock requires **model access** enabled in your AWS account/region.

Automatic discovery needs the `bedrock:ListFoundationModels` permission.

If you use profiles, set `AWS_PROFILE` on the gateway host.

OpenClaw surfaces the credential source in this order: `AWS_BEARER_TOKEN_BEDROCK`, then `AWS_ACCESS_KEY_ID` + `AWS_SECRET_ACCESS_KEY`, then `AWS_PROFILE`, then the default AWS SDK chain.

Reasoning support depends on the model; check the Bedrock model card for current capabilities.

If you prefer a managed key flow, you can also place an OpenAI-compatible proxy in front of Bedrock and configure it as an OpenAI provider instead.

‹ Litellm Vercel AI Gateway ›