ACOT101

Build your generative AI application with Amazon Bedrock

Harshal Pimpalkhute

Sr. Manager Product, Amazon Bedrock

Building generative AI applications is challenging

Accessing multiple FMs and newer versions

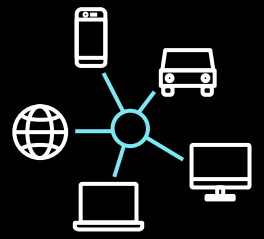
Customizing FMs is not easy

Data privacy and security

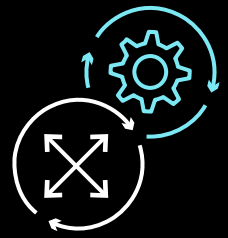
Getting FMs to execute tasks

Connecting to data sources

Difficult to manage infrastructure

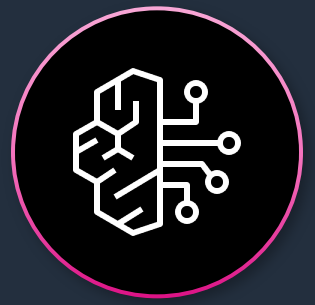


Which model should I use?



How can I move quickly?

How can I keep my data secure and private?



Amazon Bedrock

The easiest way to build and scale generative AI applications with foundation models

Choice of industry-leading FMs available via a single API

Customize your models using your organization’s data

Enterprise-grade security and privacy

How Amazon Bedrock works

Amazon Bedrock

Build generative AI applications using FMs through a serverless API service

Choose an FM

Use the playground to experiment with FMs and select the one that suits your needs

Use as is or customize

Fine-tune FMs as needed; Bedrock will automatically deploy the FM for inference

Send prompt

Use the Bedrock API to send your prompts to the model

Receive response

Receive the model response in your application

Amazon **Bedrock**

Broad choice of models

Jurassic

Contextual answers, summarization, paraphrasing

Amazon Titan

Text summarization, generation, Q&A, search

Claude

Summarization, complex reasoning, writing, coding

Command + Embed

Text generation, search, classification

Llama 2

Q&A and reading comprehension

Stable Diffusion

High-quality images and art

Enhance Customer Experiences

- CHATBOTS
- VIRTUAL ASSISTANTS
- CONVERSATION ANALYTICS
- PERSONALIZATION

Boost employee productivity & creativity

- CONVERSATIONAL SEARCH
- SUMMARIZATION
- CONTENT CREATION
- CODE GENERATION
- DATA TO INSIGHTS

Optimize business processes

- DOCUMENT PROCESSING
- DATA AUGMENTATION
- CYBERSECURITY
- PROCESS OPTIMIZATION

Model evaluation

PREVIEW

Playground

Evaluate directly in the playground as you try out different models; compare across cost latency and usage dimensions

Use playground as you narrow down on the use case and identify the FM

Programmatic

Evaluate as part of your application development lifecycle or model customization. Metrics include accuracy, toxicity, and robustness

Use programmatic evaluation as you iterate on the use case or the model

Human in the loop Bring your own team

Use the framework offered to organize your team for your evaluation. Better suited for subjective criteria such as brand voice, clarity, and tone

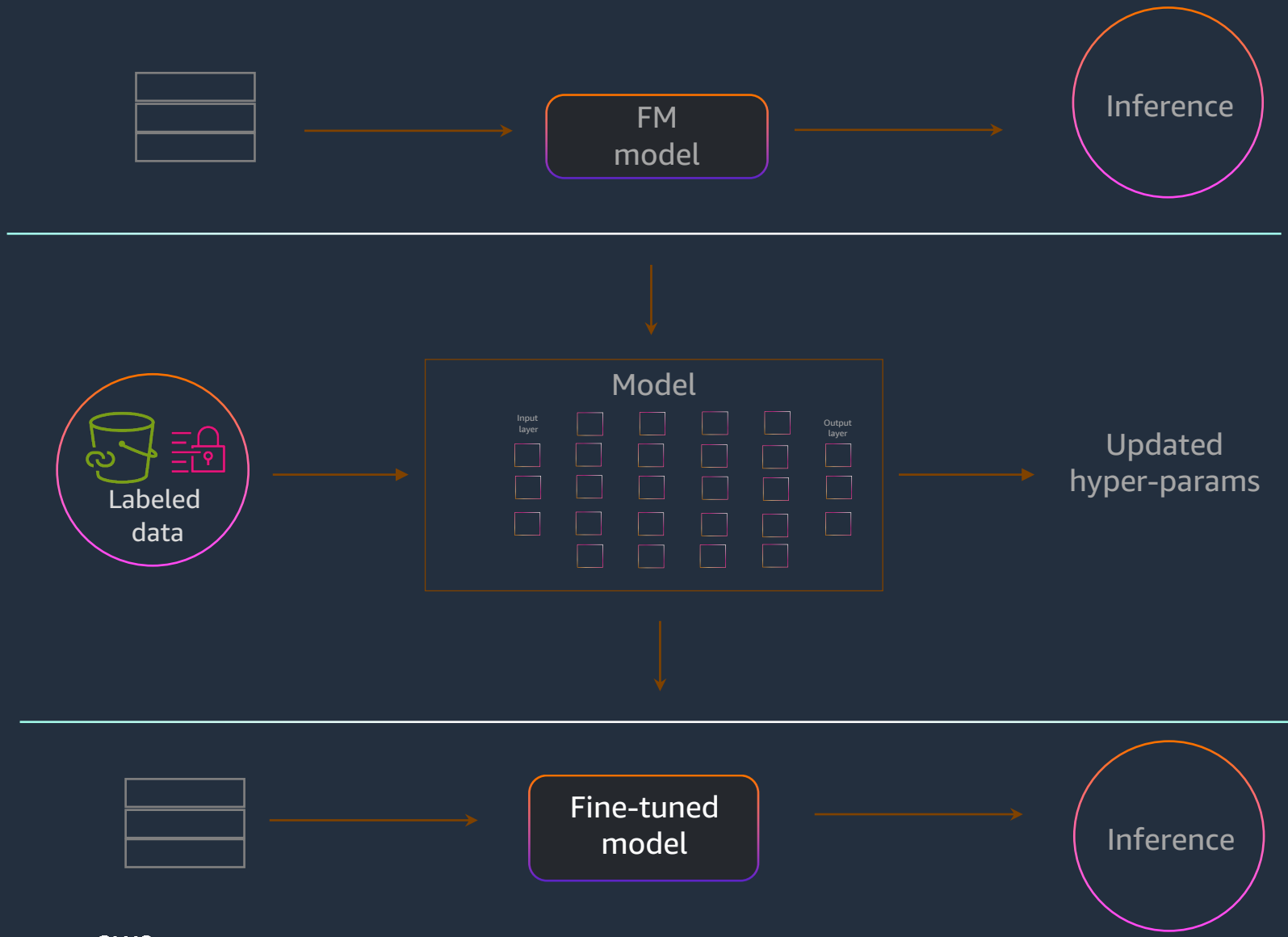
Bring your own team as you start testing your first prototype or get ready for pilot

Human in the loop AWS Managed team

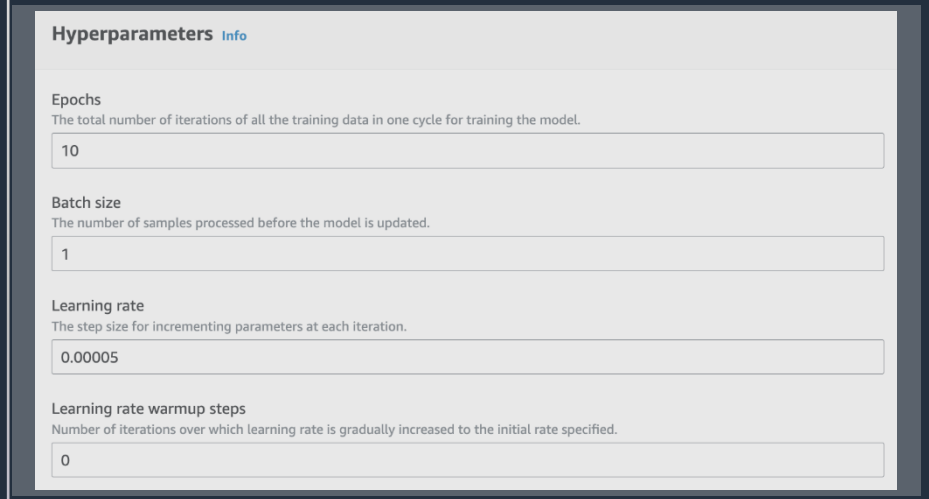
Leverage AWS expertise for an expert evaluation, in case you don't have the budget or expertise in-house

AWS managed team as you get ready for production launch of your application

Fine-tuning



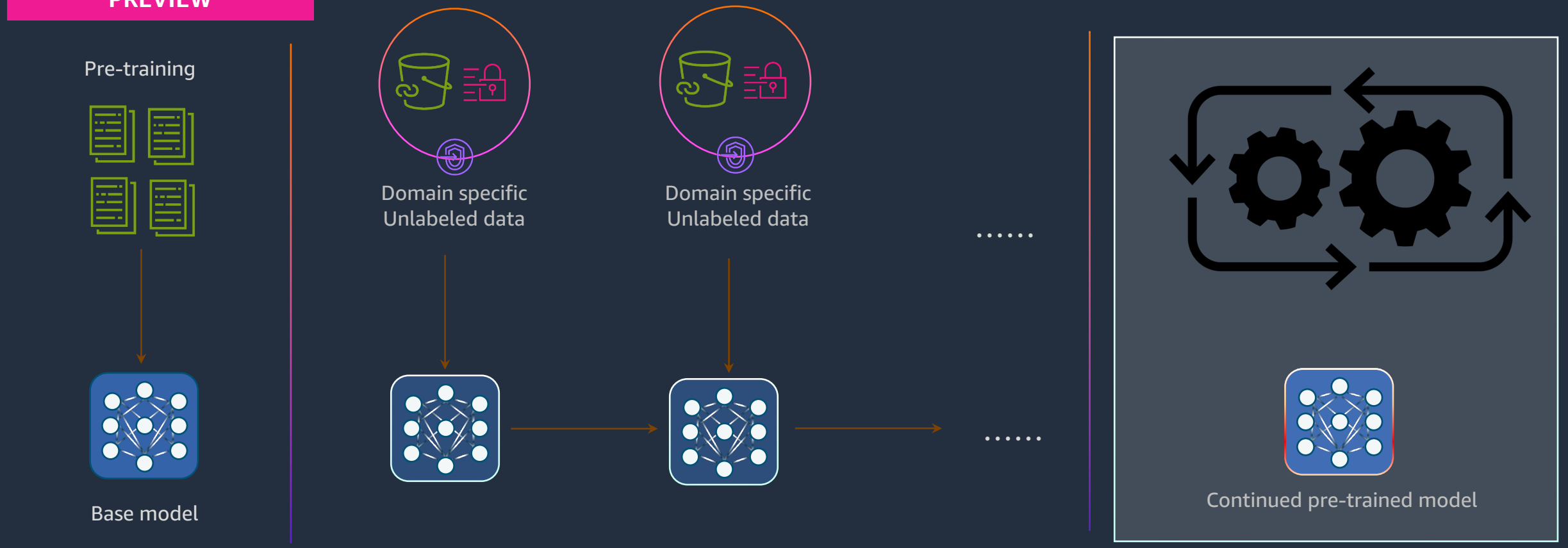
Customize models with simple configuration
Epochs, Batch size, Learning rate, Warmup steps



- **Easy** — Simplified hyper params update
- **Secure** — Data encrypted with CMK
- **Private** — Access to Amazon S3 bucket via PrivateLink

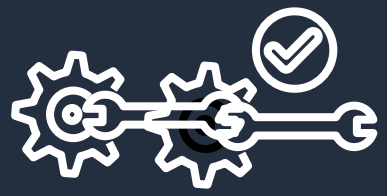
Continued pre-training

PREVIEW



Adapt model responses to the vocabulary and terminology specific to a domain

Customizing Amazon Titan models



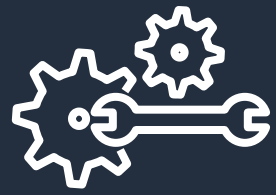
Fine tuning

PURPOSE

Maximizing accuracy
for **specific tasks**

DATA NEED

Small number
of labeled examples



Continued pre-training

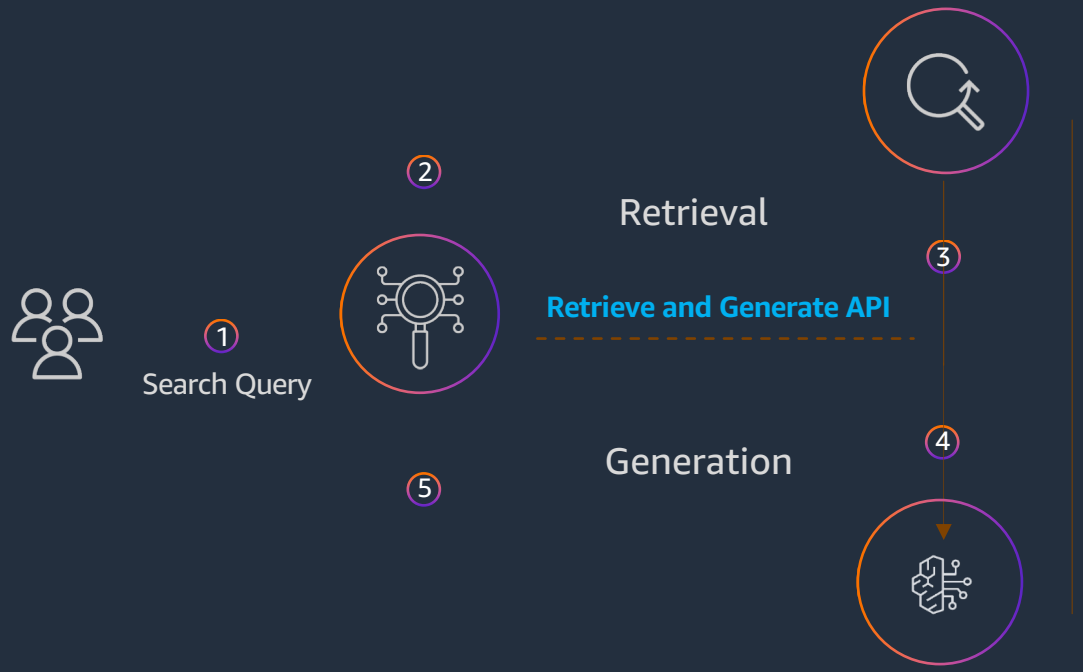
PURPOSE

Maintaining model
accuracy for **your domain**

DATA NEED

Large number
of unlabeled datasets

Architecture 1: RAG



- Pinecone
- redis
- OpenSearch
- Aurora — *Coming soon*
- MongoDB — *Coming soon*

Single API	One API for RAG
Comprehensive	Multiple Vector DBs
Session	Built-in session management

Implementation of RAG – LLM

Considerations

Use case

Context length

Hosting

Training data, customization

License agreements

Options

Amazon Titan

A121 Jurassic

Anthropic Claude

Cohere Command

Meta Llama

Implementation of RAG – Embedding model

Considerations

Max input size

Latency

Output embedding size

Ease of hosting

Accuracy

Options

Amazon Titan Embeddings

Cohere Embed

Other embedding model

Implementation of RAG – Vector store

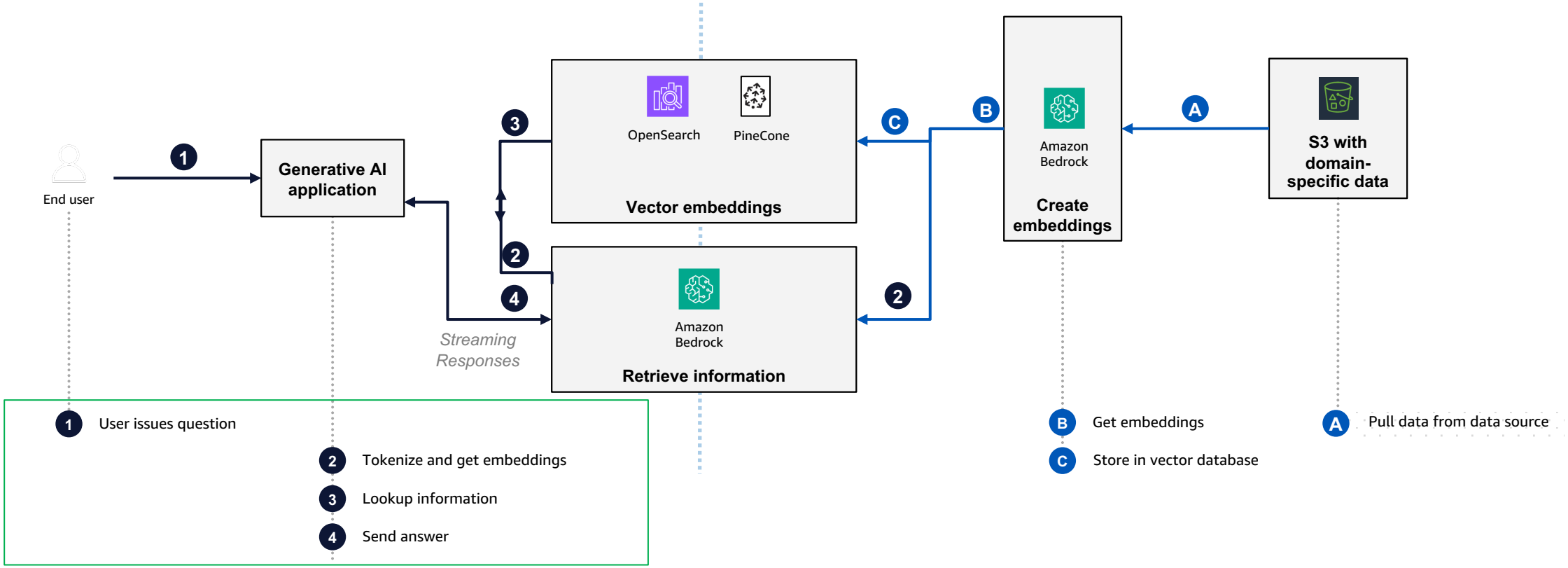
Considerations

- Nature of data sources and formats
- Dimensions
- Fully Managed x DIY
- Vector Evaluation Metrics -- RECALL
- Latency of response
- Development complexity
- Scalability
- Flexibility

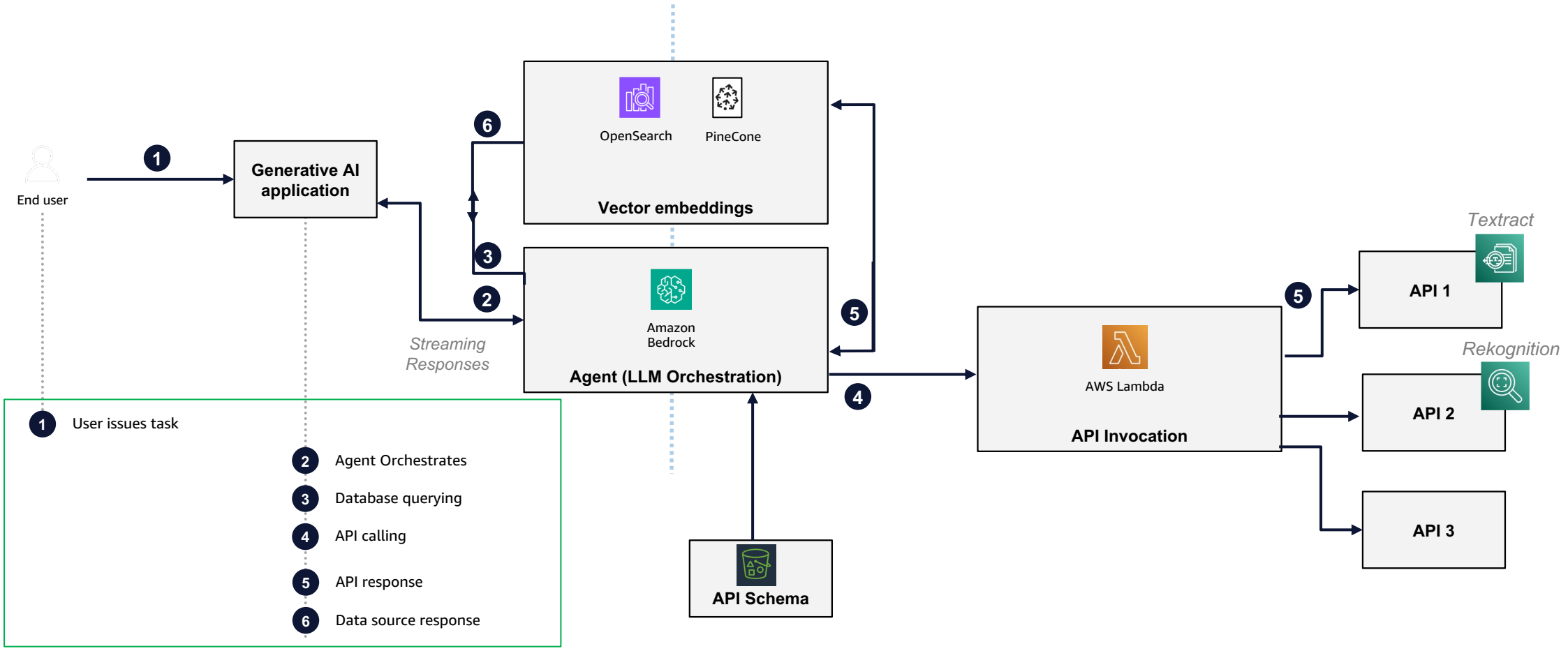
Options

- OpenSearch
- AWS Kendra
- Aurora PgVector Store
- DIY

Architecture 2: Search

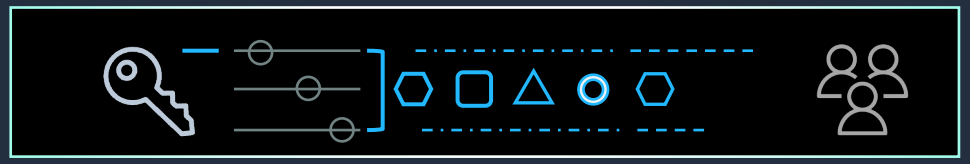


Architecture 3: Agents



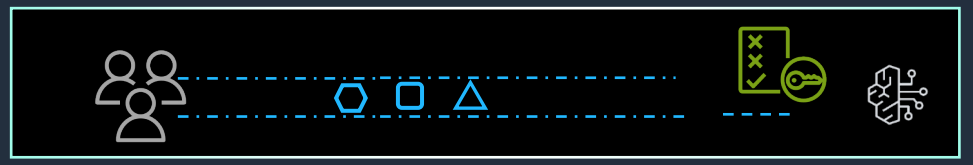
Data privacy & security

AWS IAM



Customer Managed Keys

PrivateLink



Resource based policy

Coming soon

Tagging, CloudTrail, CloudWatch

Tagging

Use tags to track model consumption and usage

CloudTrail

Track API calls across your account to base and fine tune models

CloudWatch

Log inputs, model responses and metadata for all FMs

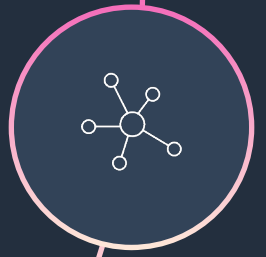
PREVIEW

Guardrails for Amazon Bedrock

Implement safeguards customized to your application requirements and responsible AI policies

Apply guardrails to multiple foundation models and Agents for Amazon Bedrock

Configure harmful content filtering based on your responsible AI policies

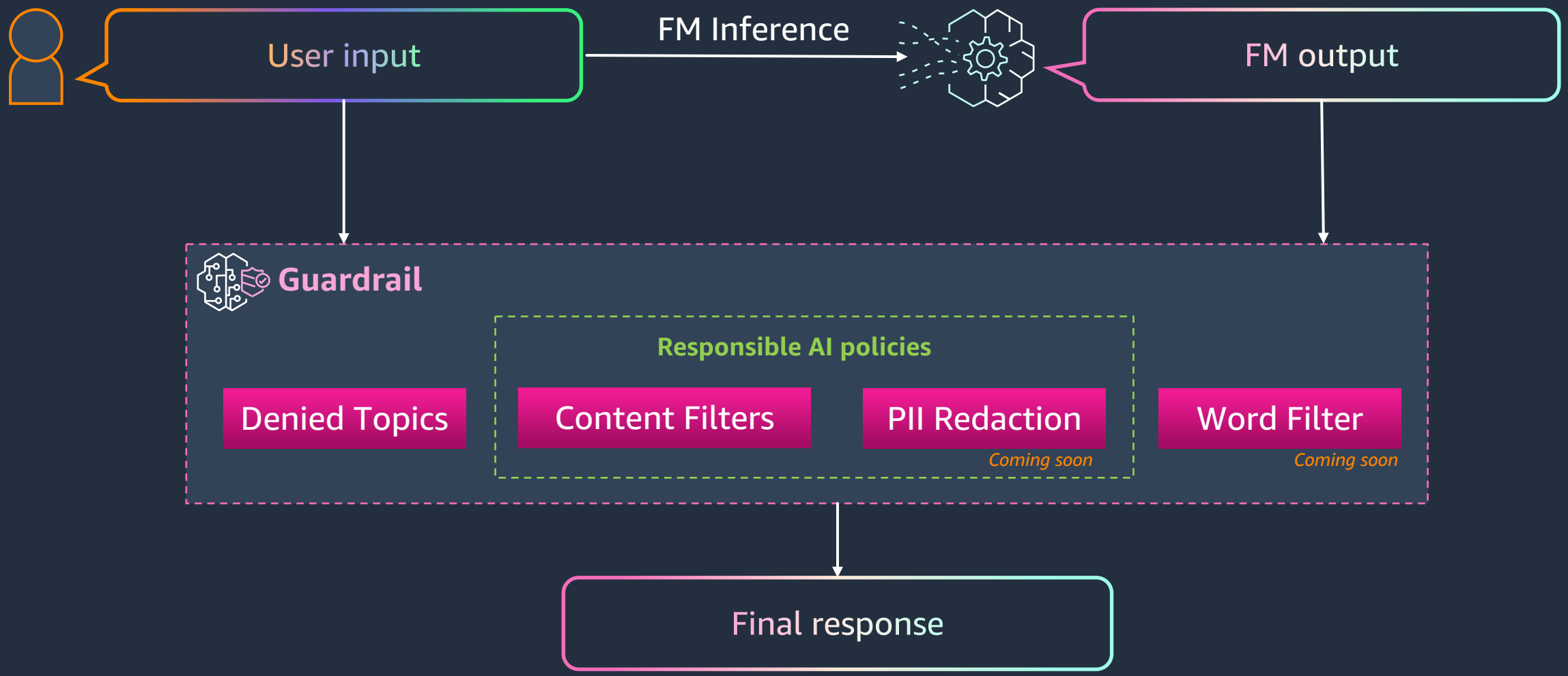


Define and disallow denied topics with short natural language descriptions

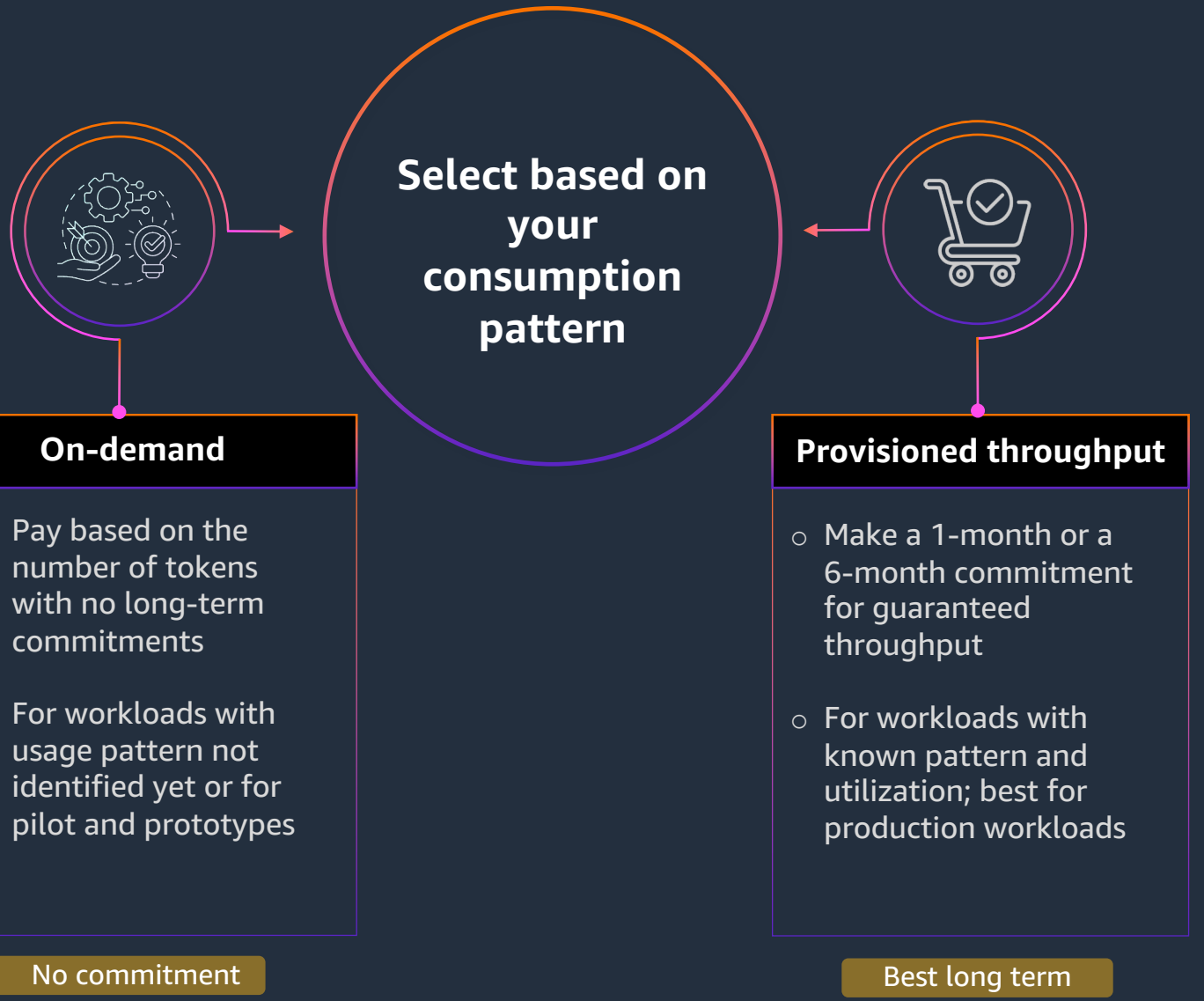
COMING SOON

Redact sensitive PII information in FM responses

How it works: Guardrails for Amazon Bedrock



Consumption model



Thank you!

Harshal Pimpalkhute

Sr. Manager Product,
Amazon Bedrock

AWS