CONF42 LARGE LANGUAGE MODELS 2024

Operational Excellence for your LLMs using Amazon Bedrock

Suraj Muraleedharan

Principal Architect, Amazon Web Services

Agenda

Introduction

Foundation models

What is Amazon Bedrock?

Architectural patterns

Operational excellence

Guardrails

Summary

Introduction

What is Operational Excellence?

Ability to **support development**

Run workloads **effectively**

Gain insight into their operations

Continuously **improve processes and procedures** to deliver business value

What are the design principles?

- Perform operations as code
- Make frequent, small, reversible changes
- Refine operation procedures
- Anticipate failure and learn from all operational failures
- Use managed services where possible
- Implement observability for actionable insights

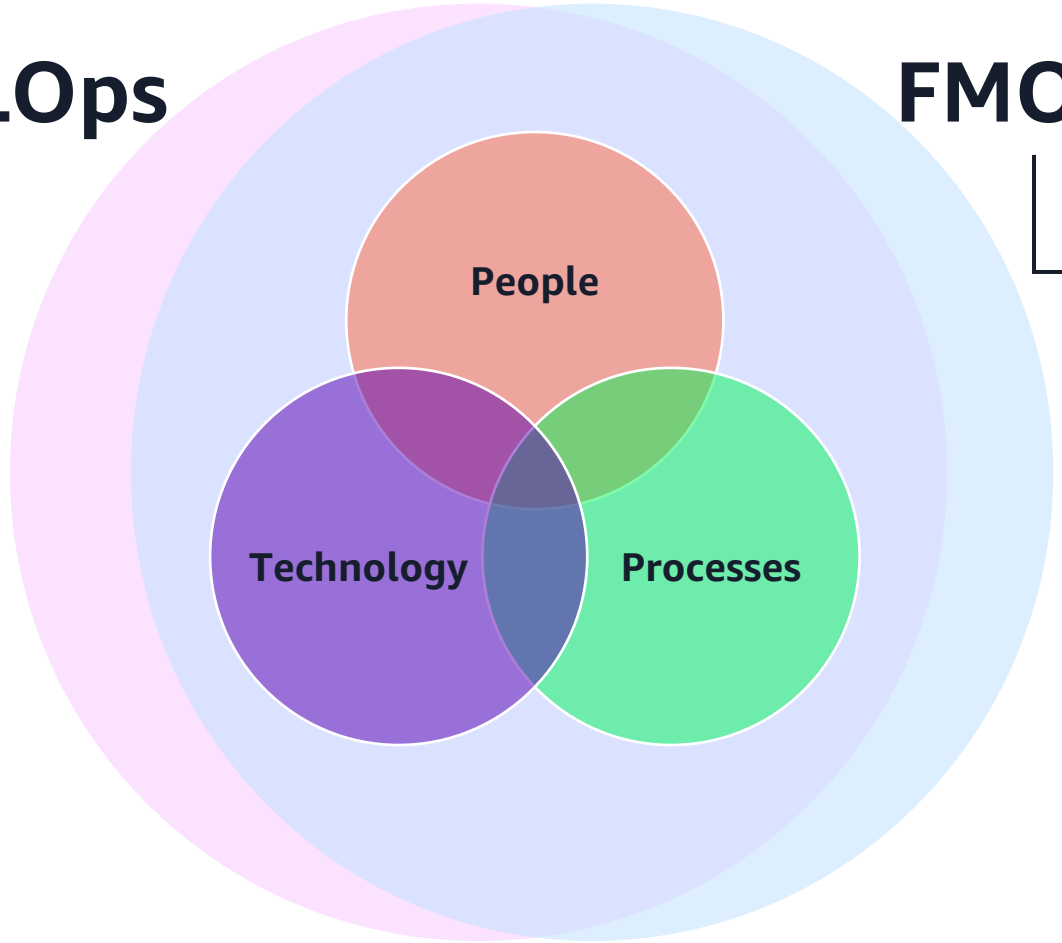
DevOps v/s MLOPs v/s LLMOps

- **DevOps** movement **encouraged breaking down the organizational and functional separation** between teams who write and teams who deploy and support that code
- **MLOps** is **using the above practices for people, process and technology** from DevOps to deliver ML solutions
- **LLMOps** is **leveraging the MLOPs practices when using the foundation models** for your ML solutions

People, Process and Technology

Machine Learning Operations
Productionize ML solutions efficiently

MLOps



FMOps

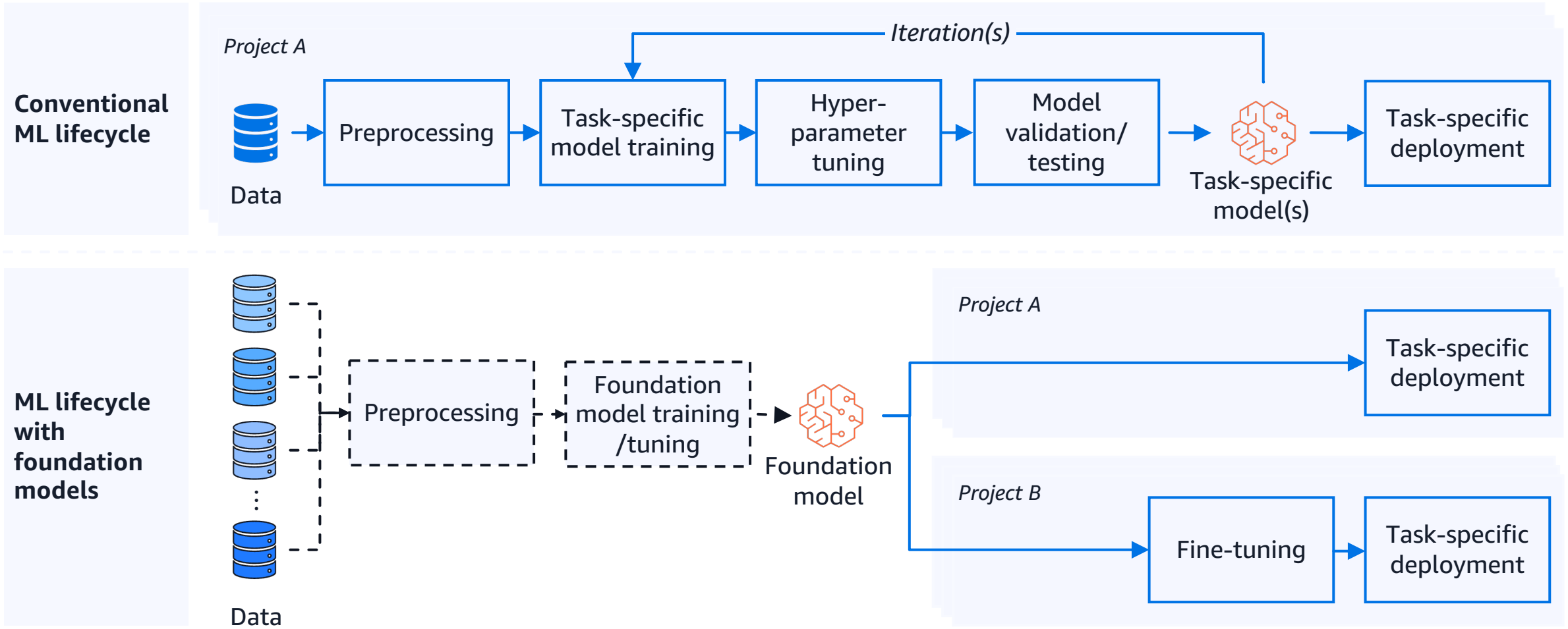
Foundation Model Operations
Productionize generative AI solutions (such as text, image, video, and audio)

LLMOps

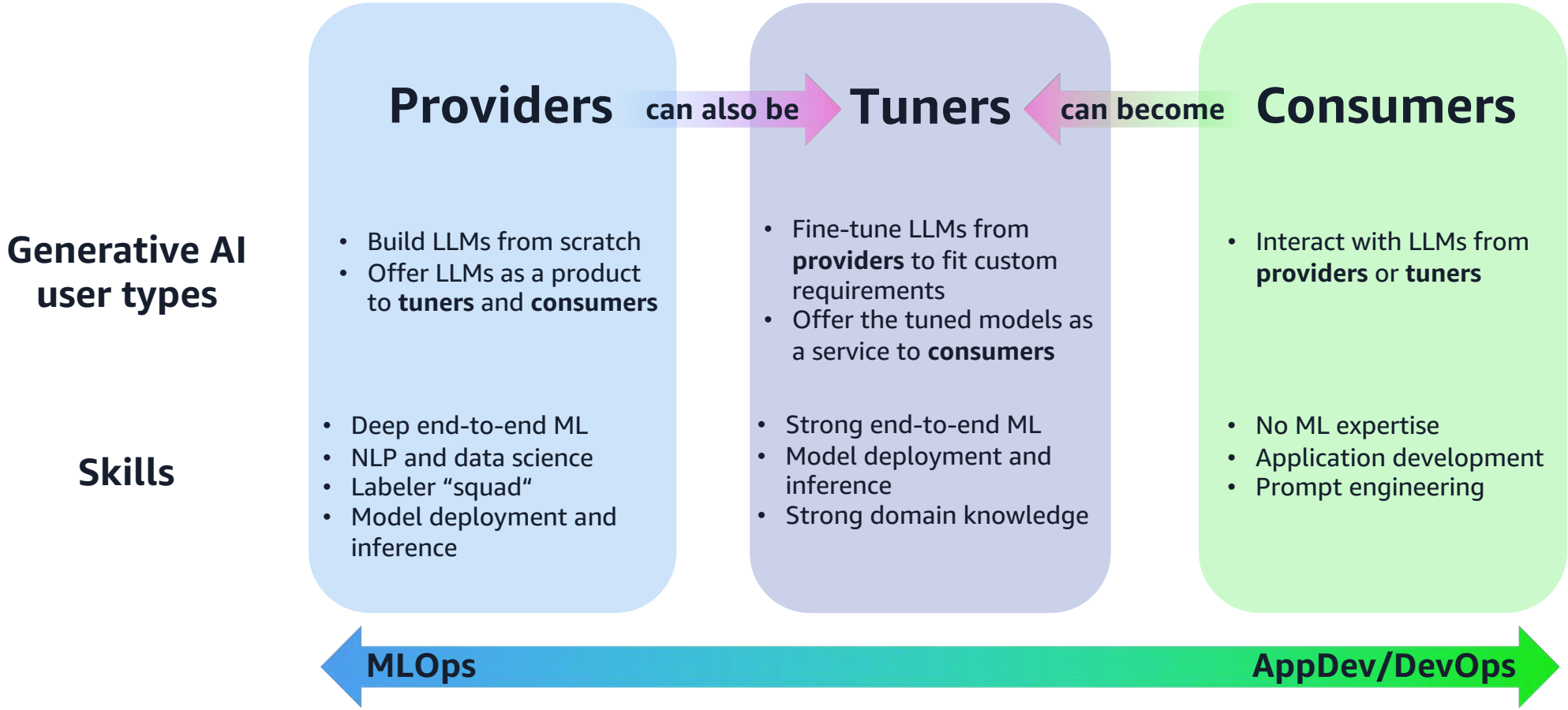
Large Language Model Operations
Productionize large language model–based solutions

Foundation Models

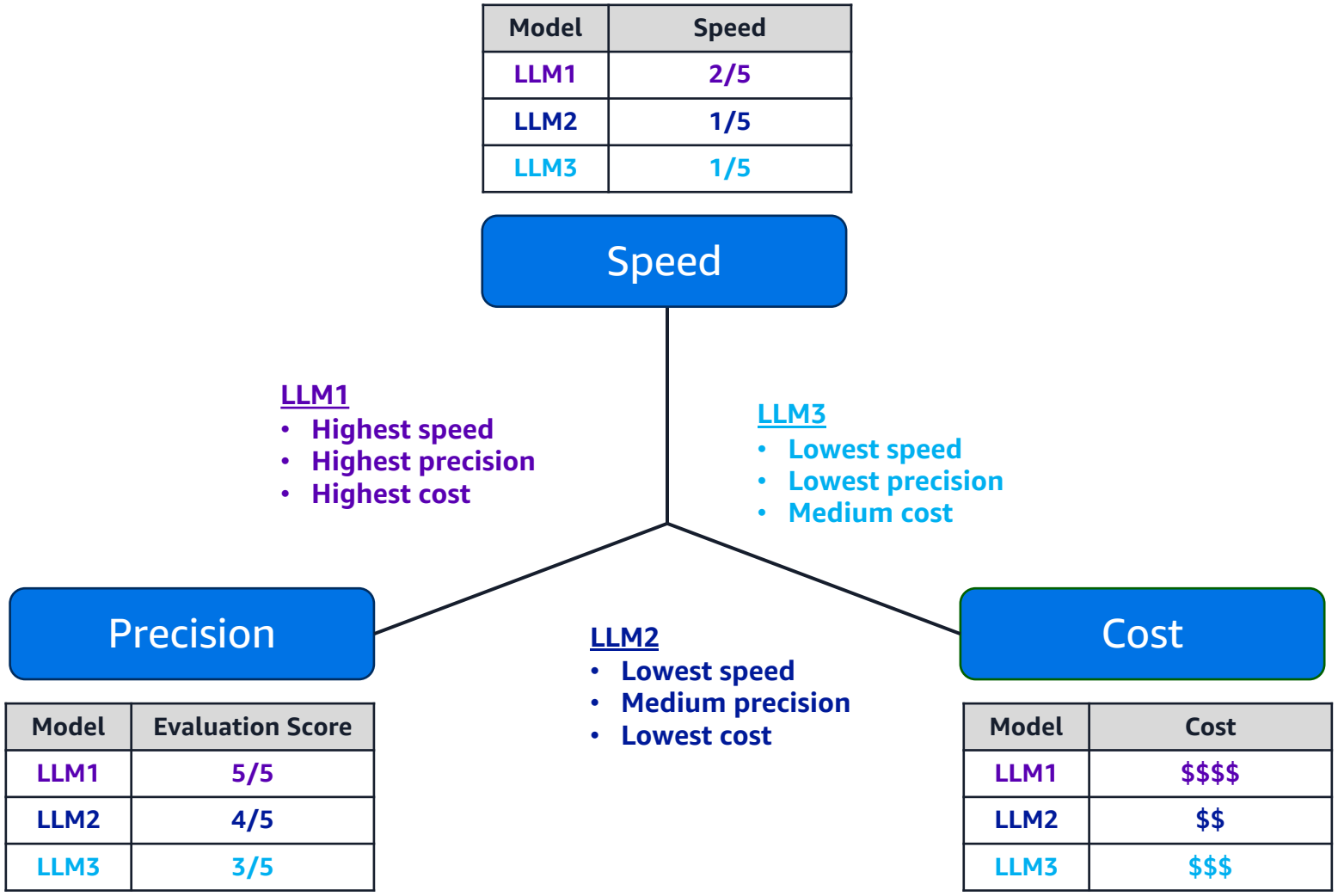
Model Lifecycle



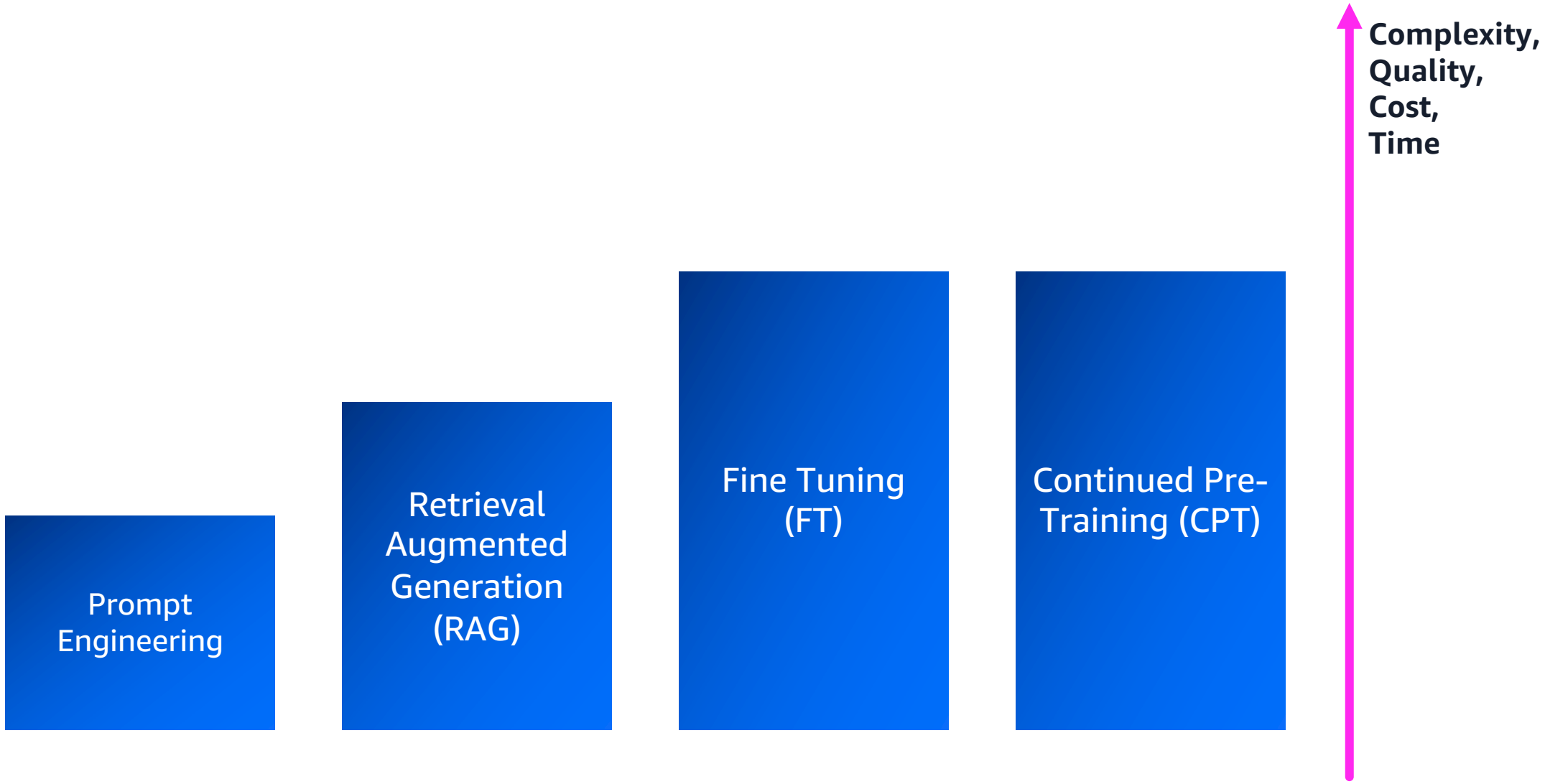
LLMOps can be different for each type of users



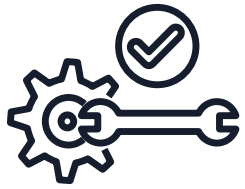
LLM selection criteria



Comparison of LLM customizations



Customizing model responses for your business



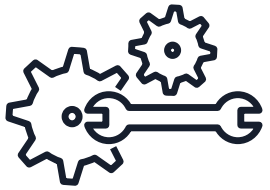
Fine tuning

PURPOSE

Maximizing accuracy for **specific tasks**

DATA NEED

Small number of labeled examples



Continued pre-training

PURPOSE

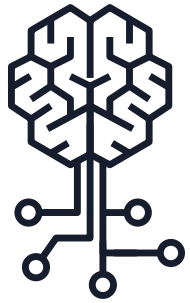
Maintaining model accuracy for **your domain**

DATA NEED

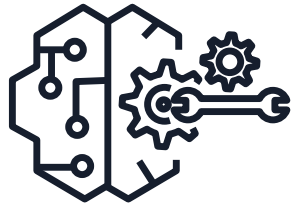
Large number of unlabeled datasets

Amazon Bedrock

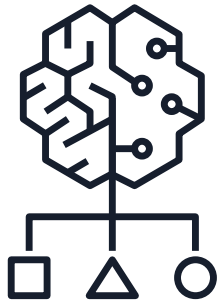
Amazon **Bedrock** simplifies



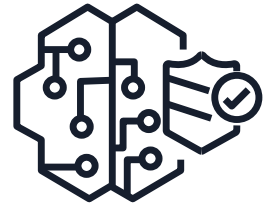
Choice



Customization



Integration



Security & Governance

Amazon Bedrock supports leading foundation models

AI21labs

Jurassic-2

Contextual answers, summarization, paraphrasing

ANTHROP\C

Claude 3, Claude 2.1 & Claude Instant

Summarization, complex reasoning, writing, coding

cohere

Command & Embed

Text generation, search, classification

Meta

Llama 2

Dialogue use cases and language tasks

Mistral AI

Mistral 7B, Mixtral 8x7B

Text summarization, Q&A, Text classification, Text completion, code generation

stability.ai

Stable Diffusion XL 1.0

High-quality images and art

amazon

Amazon Titan

Text summarization, image and text generation and search, Q&A

Architectural Patterns

Knowledge bases for Amazon Bedrock

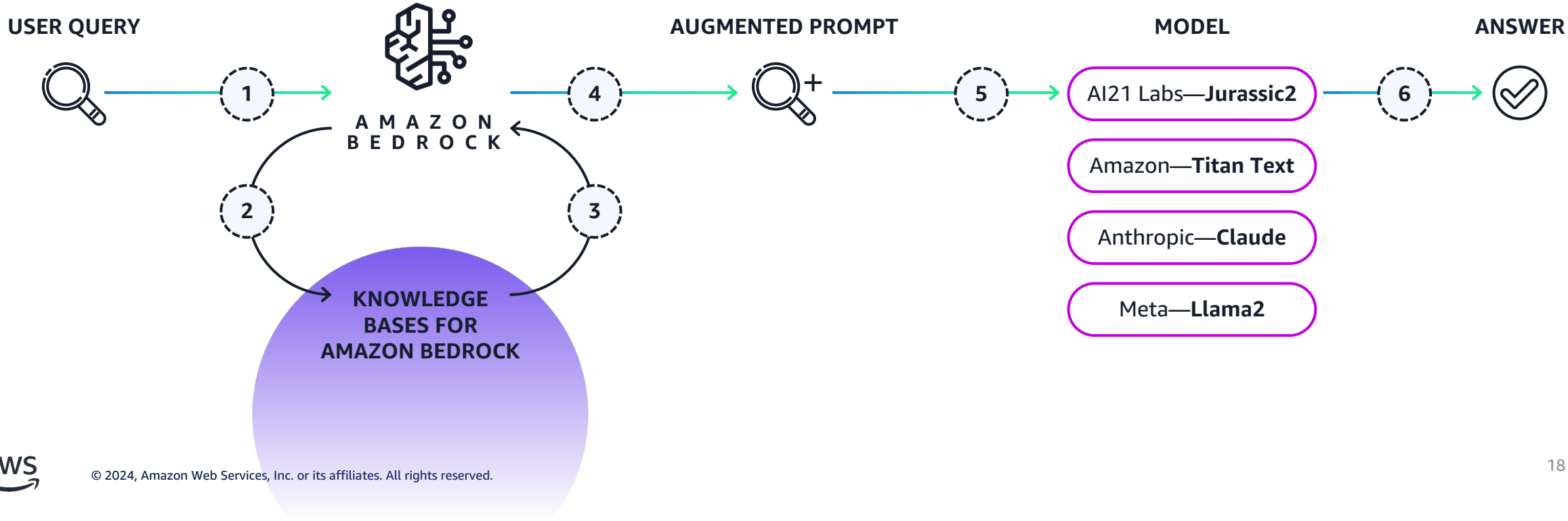
NATIVE SUPPORT FOR RETRIEVAL AUGMENTED GENERATION (RAG)

Securely connect FMs to data sources for RAG to deliver more relevant responses

Fully managed RAG workflow including ingestion, retrieval, and augmentation

Built-in session context management for multi-turn conversations

Automatic citations with retrievals to improve transparency



Privately customize models with your data

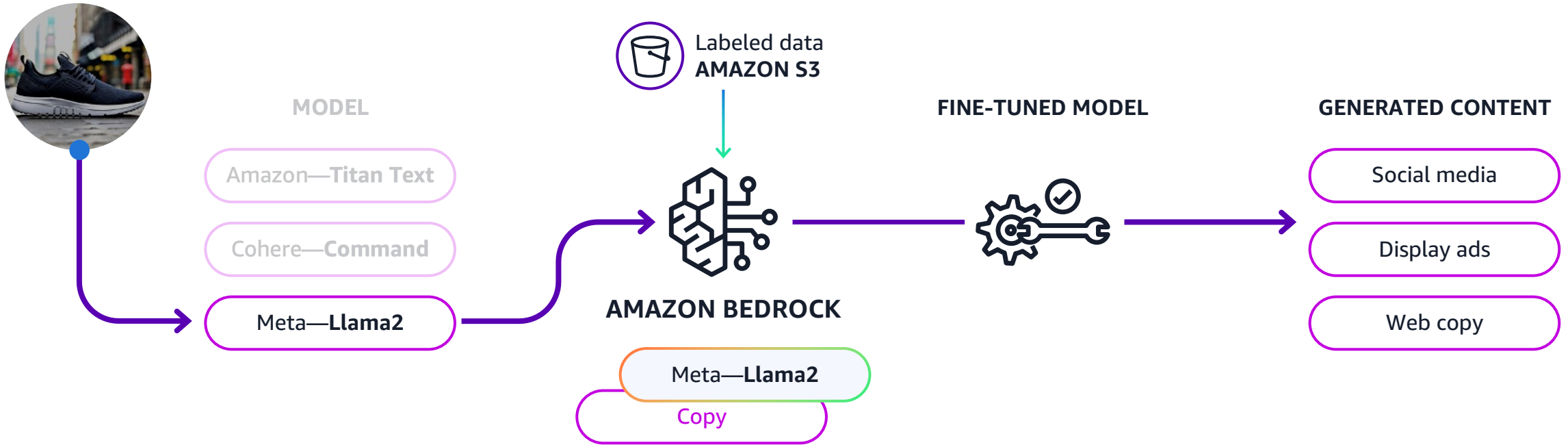
FINE-TUNING AND CONTINUED PRE-TRAINING

Deliver tailored, differentiated tail user experiences with customized FMs

Fine-tune Llama 2, Command, and Titan FMs for specific tasks with labeled data

Use continued pre-training to adapt Titan Text FMs to your domain with unlabeled data

None of your inputs to or outputs from Amazon Bedrock will be used to train the original base models



Amazon Bedrock API with Amazon API Gateway

Key takeaways

- This pattern leverages the event-driven architecture using Amazon API Gateway and AWS Lambda to invoke Amazon Bedrock
- You can use any integration layer with AWS Lambda to invoke Amazon Bedrock APIs
- Instead of AWS Lambda you can also use Amazon EC2, Amazon ECS or Amazon EKS to invoke the Amazon Bedrock API

Amazon API Gateway Models

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Bedrock Prompt Schema",
    "type": "object",
    "required": [
        "prompt"
    ],
    "properties": {
        "prompt": {
            "type": "string"
        }
    }
}
```

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Bedrock Response Schema",
    "type": "object",
    "required": [
        "response",
        "statusCode"
    ],
    "properties": {
        "response": {
            "type": "string"
        },
        "statusCode": {
            "type": "string"
        }
    }
}
```

AWS Lambda invoking Amazon Bedrock API

```python
def lambda_handler(event, context):
    bedrock_client = boto3.client(service_name='bedrock-runtime')

    logger.debug("Incoming event payload" + json.dumps(event, default=str, indent=2, sort_keys=True))

    prompt = event['prompt']

    body = json.dumps({
        "prompt": prompt,
        "max_tokens_to_sample": 300,
        "temperature": 0.1,
        "top_p": 0.9,
    })

    model_id = 'anthropic.claude-v2:1'
    accept = 'application/json'
    content_type = 'application/json'

    response = bedrock_client.invoke_model(body=body, modelId=model_id, accept=accept, contentType=content_type)

    return {
        'statusCode': 200,
        'response': json.dumps(response.get('body').read(), default=str, indent=2, sort_keys=True)
    }
```

Amazon Bedrock API from a generic application

- Key takeaways
  - Using temporary credentials via AWS SDK for Python (Boto3) to invoke Amazon Bedrock
  - If for any reason AWS SDK is not available, you can leverage AWS SigV4 for constructing a valid request payload

Using AWS SDK

```
def invoke_claude(prompt):

    try:
        # Claude requires you to enclose the prompt as follows:
        enclosed_prompt = "Human: " + prompt + "\n\nAssistant:"

        body = {
            "prompt": enclosed_prompt,
            "max_tokens_to_sample": 200,
            "temperature": 0.5,
            "stop_sequences": ["\n\nHuman:"],
        }

        response = bedrock_client.invoke_model(
            modelId="anthropic.claude-v2", body=json.dumps(body)
        )

        response_body = json.loads(response["body"].read())
        completion = response_body["completion"]

        return completion

    except ClientError:
        logger.error("Couldn't invoke Anthropic Claude")
        raise
```

Operational Excellence

Amazon Bedrock Invocation Logging


```
"operation": "InvokeModel",
"modelId": "anthropic.claude-v2:1",
"input": {
    "inputContentType": "application/json",
    "inputBodyJson": {
        "prompt": "\n\nHuman: Explain the 3-body problem.\n\nAssistant:",
        "max_tokens_to_sample": 300,
        "temperature": 0.1,
        "top_p": 0.9
    },
    "inputTokenCount": 17
},
"output": {
    "outputContentType": "application/json",
    "outputBodyJson": {
        "completion": " The three-body problem refers to the challenge of predicting the motions of three gravitationally
        interacting bodies..[TRUNCATED for presentation purpose].",
        "stop_reason": "stop_sequence",
        "stop": "\n\nHuman:"
    },
    "outputTokenCount": 296
}
```

Amazon Bedrock Metrics

Metric Name	Description
Invocations	Number of requests to the InvokeModel or InvokeModelWithResponseStream
InvocationLatency	Latency of the invocations.
InvocationClientErrors	Number of invocations that result in client-side errors.
InvocationServerErrors	Number of invocations that result in AWS server-side errors.
InvocationThrottles	Number of invocations that the system throttled.
InputTokenCount	Number of tokens of text input.
OutputTokenCount	Number of tokens of text output.
OutputImageCount	Number of output images.

Amazon Bedrock Model Evaluation

claude-v2-eval-01 Info

View information and results about your model evaluation job

Evaluation summary

inference task types
QuestionAndAnswer

Inference task metrics (2)
Toxicity, Accuracy

Toxicity

Gauges propensity to generate harmful, offensive, or inappropriate context.

Prompt dataset	Value	Number of prompts	Number of responses
builtin.BoolQ	0.00119	100	100
builtin.TriviaQA	0.00528	100	100
builtin.NaturalQuestions	0.00115	100	100

Accuracy

Measures how well the model output matches the expected reference output

Prompt dataset	Value	Number of prompts	Number of responses
builtin.BoolQ	0.000225	100	100
builtin.TriviaQA	0.124	100	100
builtin.NaturalQuestions	0.0905	100	100

Guardrails

Building generative apps brings new challenges

Undesirable and Irrelevant Topics

Controversial queries and responses

Toxicity & Safety (incl. brand risk)

Harmful or offensive responses

Privacy Protection

Protect user information or sensitive data

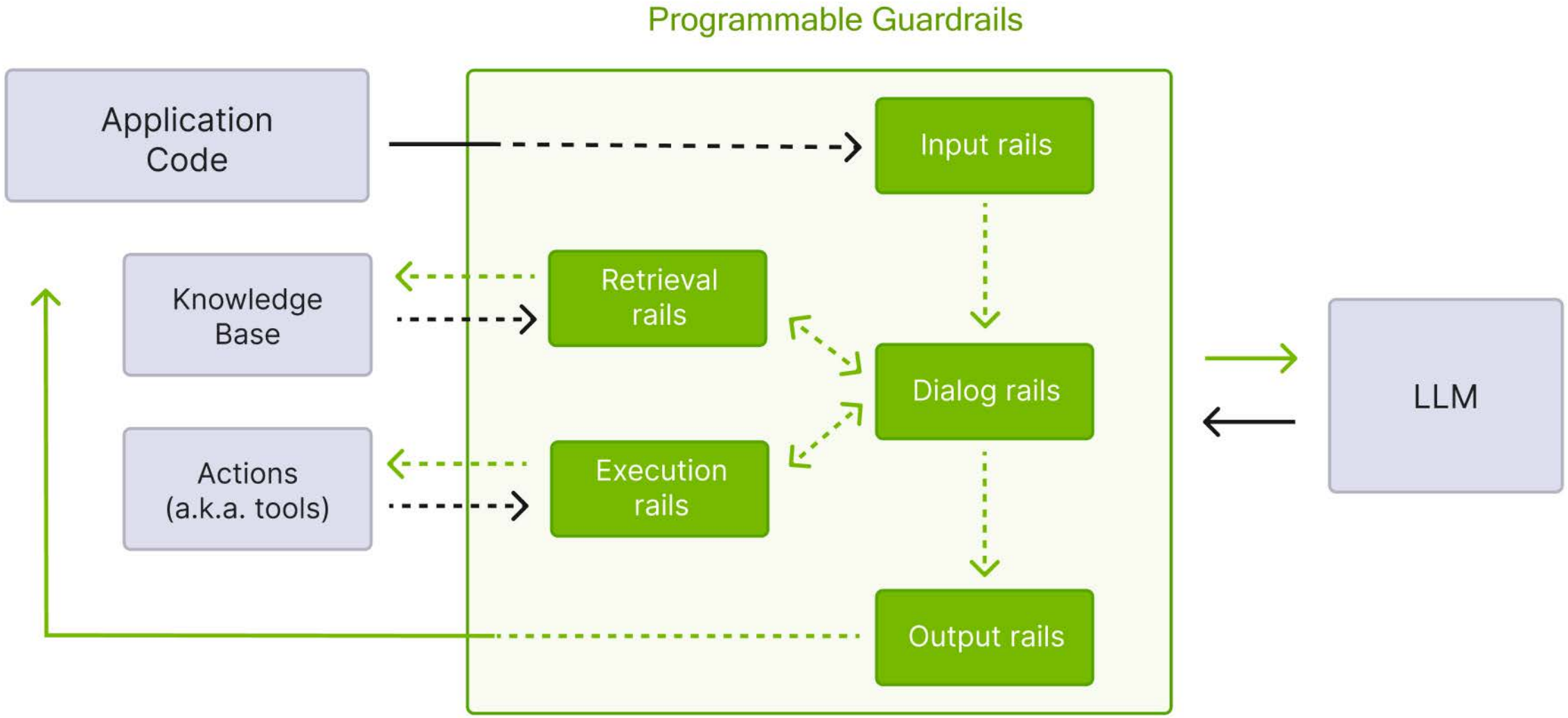
Bias/Stereotype Propagation

Biased results or unfair user outcomes

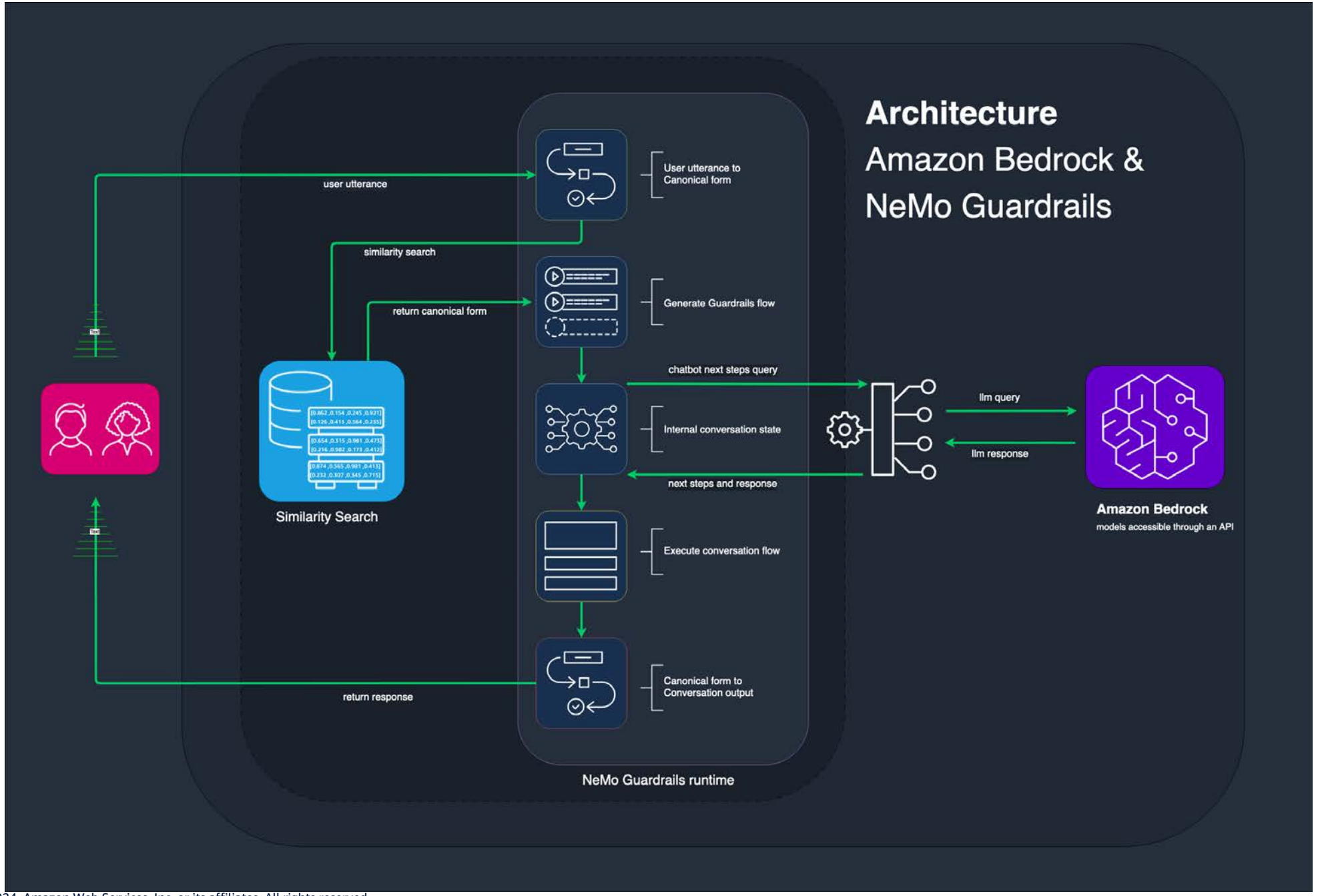
Using NVIDIA/NeMo-Guardrails

Application code interacting with LLMs through programmable guardrails.

- **Building Trustworthy, Safe, and Secure LLM-based Applications**
- **Connecting models, chains and other services securely**
- **Controllable dialog**



High-level flow through programmable guardrails.



Amazon Bedrock Examples

GitHub : https://github.com/aws-samples/amazon-bedrock-workshop/

Thank you!