Welcome Back to AWS CERTIFIED AI PRACTITIONER DAY 2

Michael Shannon
and Eian Clair

Class will begin at 10:00 am
Central Standard Time

BUILDING GENERATIVE AI APPLICATIONS WITH AWS

Objectives

- Describe how to develop generative AI applications with Amazon SageMaker JumpStart, Amazon Bedrock, Amazon Q, and PartyRock
- List the advantages of using AWS generative AI services for building applications
- Explore the benefits of AWS infrastructure for generative AI applications

DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON SAGEMAKER JUMPSTART

In this demo…

We will explore how to develop generative AI applications with Amazon SageMaker JumpStart.

DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON BEDROCK

- Developing generative AI applications with Amazon Bedrock is a streamlined process thanks to its comprehensive suite of tools and models
- In the AWS account, navigate to the Amazon Bedrock section in the AWS Management Console

CHOOSE YOUR FOUNDATION MODELS IN BEDROCK

- **Model selection:** Bedrock offers a variety of high-performing foundation models from leading AI companies, including AI21 Labs, Anthropic, Cohere, Meta, Mistral AI, and Stability AI
- **Customization:** Customers can fine-tune these models using their proprietary data to improve accuracy and relevance

DEVELOP THE BEDROCK APPLICATION

- **Amazon Bedrock IDE:** Use the Amazon Bedrock integrated development environment (IDE) within Amazon SageMaker Unified Studio
  - This environment provides an intuitive interface for building and customizing generative AI applications
- **Knowledge bases:** Implement Retrieval-Augmented Generation (RAG) workflows to connect the data to the models, ensuring responses are contextual and relevant

BUILD AND TEST IN BEDROCK

- **Prototyping:** Use the IDE to prototype the application, leveraging tools like prompt engineering, functions, and flows for rapid iteration
- **Testing:** Evaluate the models using built-in tools to ensure they meet performance, quality, and safety metrics

DEPLOY AND SCALE BEDROCK

- **Deploy and scale deployment:** Once the application is ready, deploy it using Bedrock's managed services
  - This allows one to scale the application as needed without worrying about infrastructure management
- **Monitoring and optimization:** Continuously monitor the application's performance and make necessary adjustments to optimize its efficiency and effectiveness

DEVELOPING GENERATIVE AI APPLICATIONS WITH PARTYROCK BEDROCK PLAYGROUNDS

In this demo…

We will explore how to develop generative AI applications with PartyRock Bedrock Playgrounds.

DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON Q

- **Set up the environment:** Install the necessary tools and libraries
  - Amazon Q integrates well with various IDEs like Visual Studio Code and JetBrains
  - Make sure to have an AWS account and the required permissions to access Amazon Q services
- **Data preparation:** Collect and preprocess the data
  - This might involve cleaning, labeling, and organizing the data to ensure it’s ready for training
  - Use Amazon Q’s data labeling services to create high-quality training datasets

DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON Q

- **Model development:** Utilize Amazon Q's integrated development environment to build and train the generative AI models
  - You can leverage built-in algorithms and frameworks like TensorFlow, PyTorch, and Apache MXNet
  - Experiment with different model architectures and hyperparameters to optimize performance

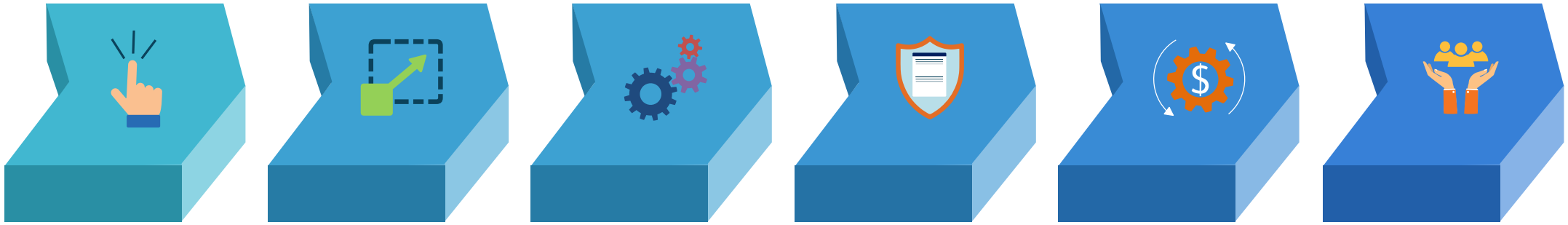
DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON Q

- **Model deployment:** Deploy the trained models using Amazon Q's managed services
  - This includes setting up endpoints for real-time inference and batch processing
  - Monitor and manage the deployed models to ensure they perform as expected

DEVELOPING GENERATIVE AI APPLICATIONS WITH AMAZON Q

- **Integration and testing:** Integrate the generative AI models into your applications
  - Q provides APIs and SDKs to facilitate this process
  - Test your applications thoroughly to ensure they meet your requirements and perform well under different scenarios
- **Maintenance and updates:** Regularly update your models with new data to improve accuracy and performance

THE "ACCESSIBILITY" ADVANTAGE OF USING AWS GENERATIVE AI TO BUILD APPLICATIONS



Ease of use

Scalability

Integration

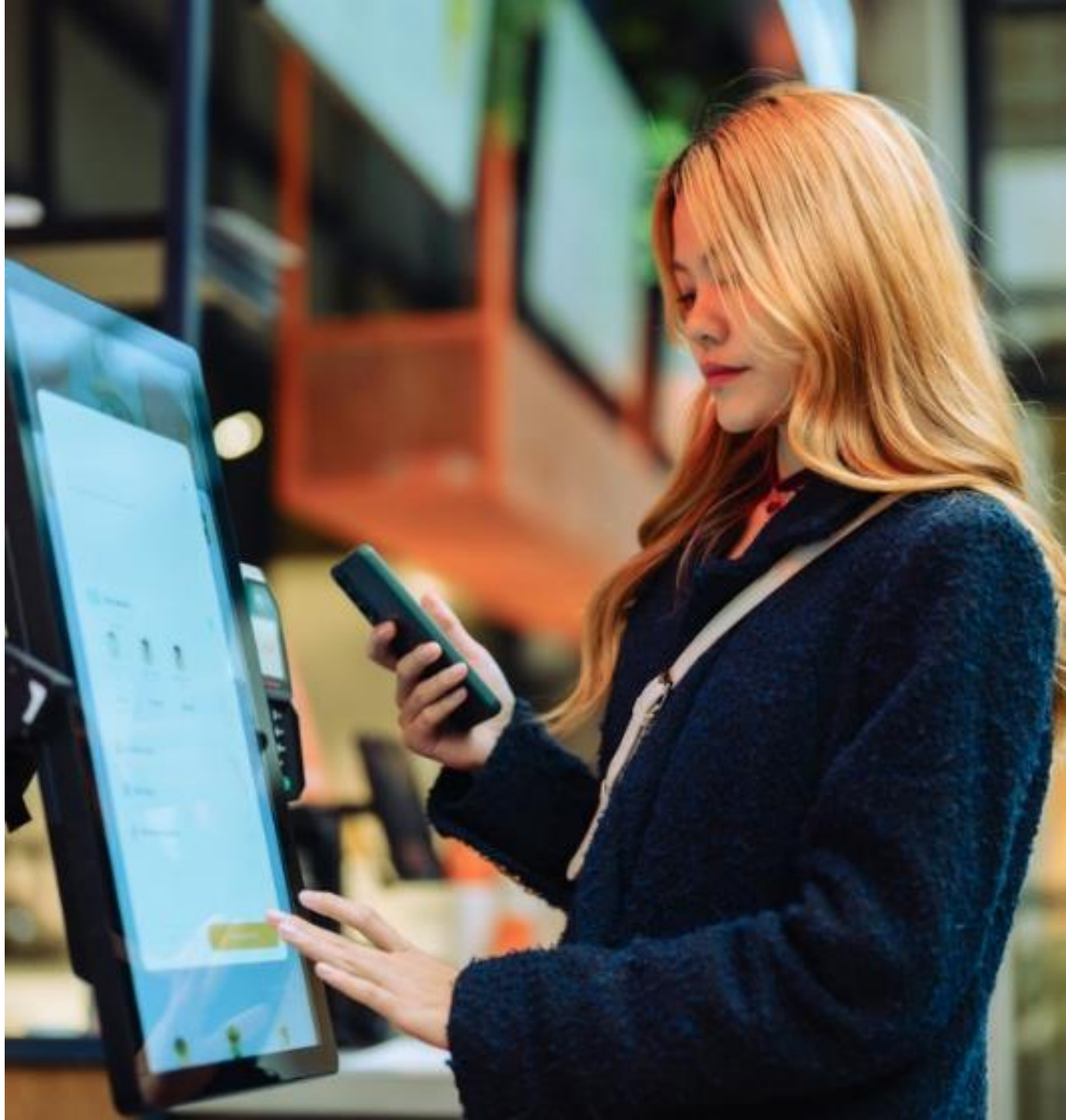
Security and compliance

Cost effectiveness

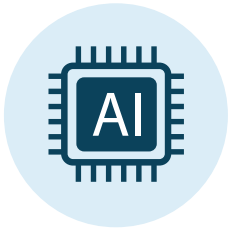
Support and community

THE "EFFICIENCY" ADVANTAGE OF USING AWS GENERATIVE AI

- The "efficiency" advantage of using AWS generative AI to build applications lies in its ability to streamline and accelerate the development process
- Here are some key points:
  - Automated processes
  - Optimized performance
  - Resource management
  - Integration with the AWS ecosystem
  - Continuous improvement

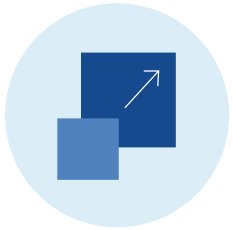


THE "LOW BARRIER TO ENTRY" ADVANTAGE OF USING AWS GENERATIVE AI TO BUILD APPLICATIONS



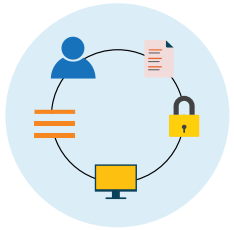
Pre-trained models

Integration with existing data



Scalability

Comprehensive tools and services



Support and resources

THE "SPEED TO MARKET" ADVANTAGE OF USING AWS GENERATIVE AI

- The "speed to market" advantage of using AWS generative AI is all about accelerating the development and deployment of applications
- Here are some key points:
  - Rapid prototyping
  - Pre-built models and services
  - Seamless integration
  - Scalable infrastructure
  - Comprehensive support

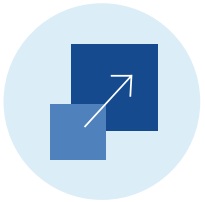
THE "COST-EFFECTIVENESS" ADVANTAGE OF USING AWS GENERATIVE AI TO BUILD APPLICATIONS

THE "MEETING BUSINESS GOALS" ADVANTAGES OF USING AWS GENERATIVE AI SERVICES

Customization

Enhanced decision-making

Increased productivity



Scalability

Competitive advantage

THE "SECURITY" BENEFITS OF AWS INFRASTRUCTURE FOR GENERATIVE AI APPLICATIONS

Built-in security

AWS infrastructure is designed with security at its core

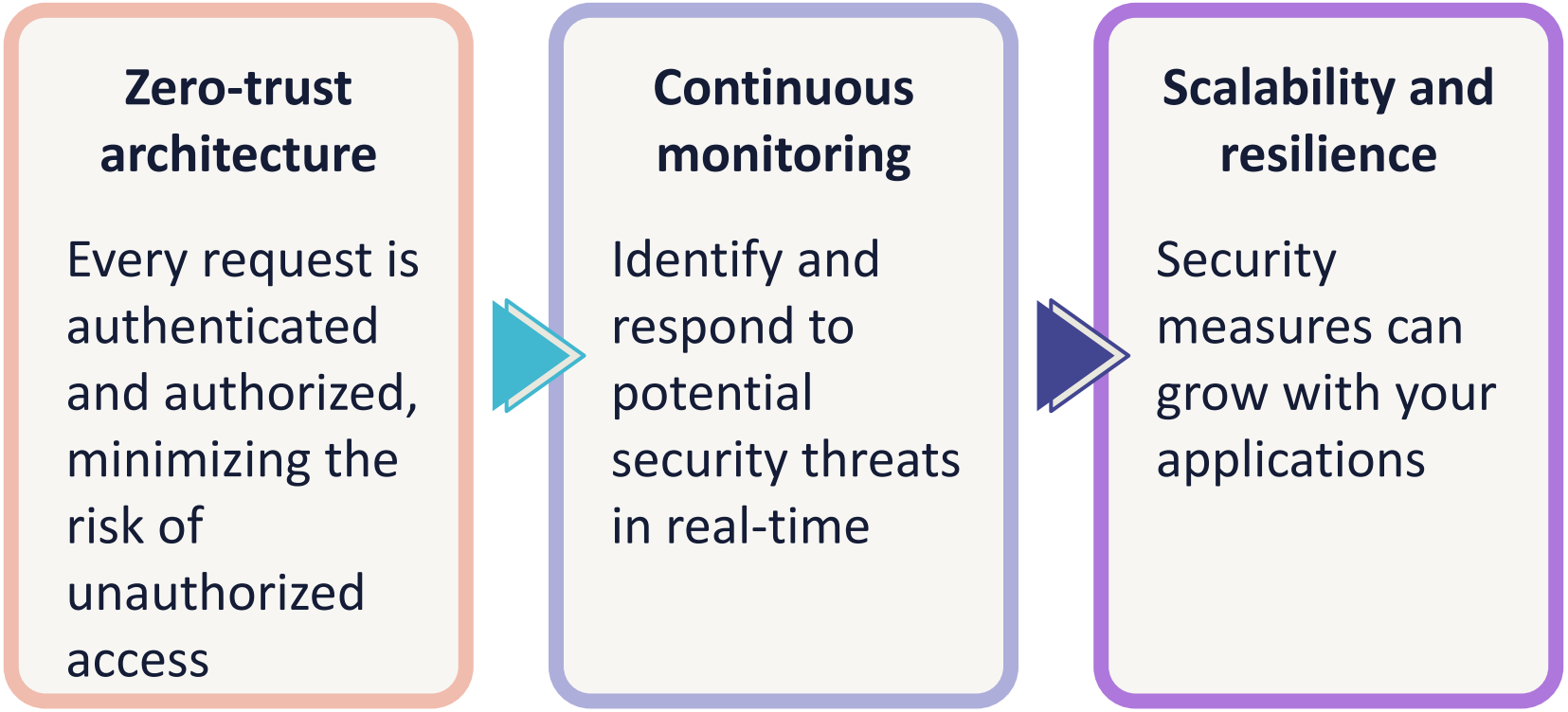
Data protection

AWS ensures data confidentiality, integrity, and availability

Compliance and governance

AWS offers integrated tools for compliance and governance

THE "SECURITY" BENEFITS OF AWS INFRASTRUCTURE FOR GENERATIVE AI APPLICATIONS

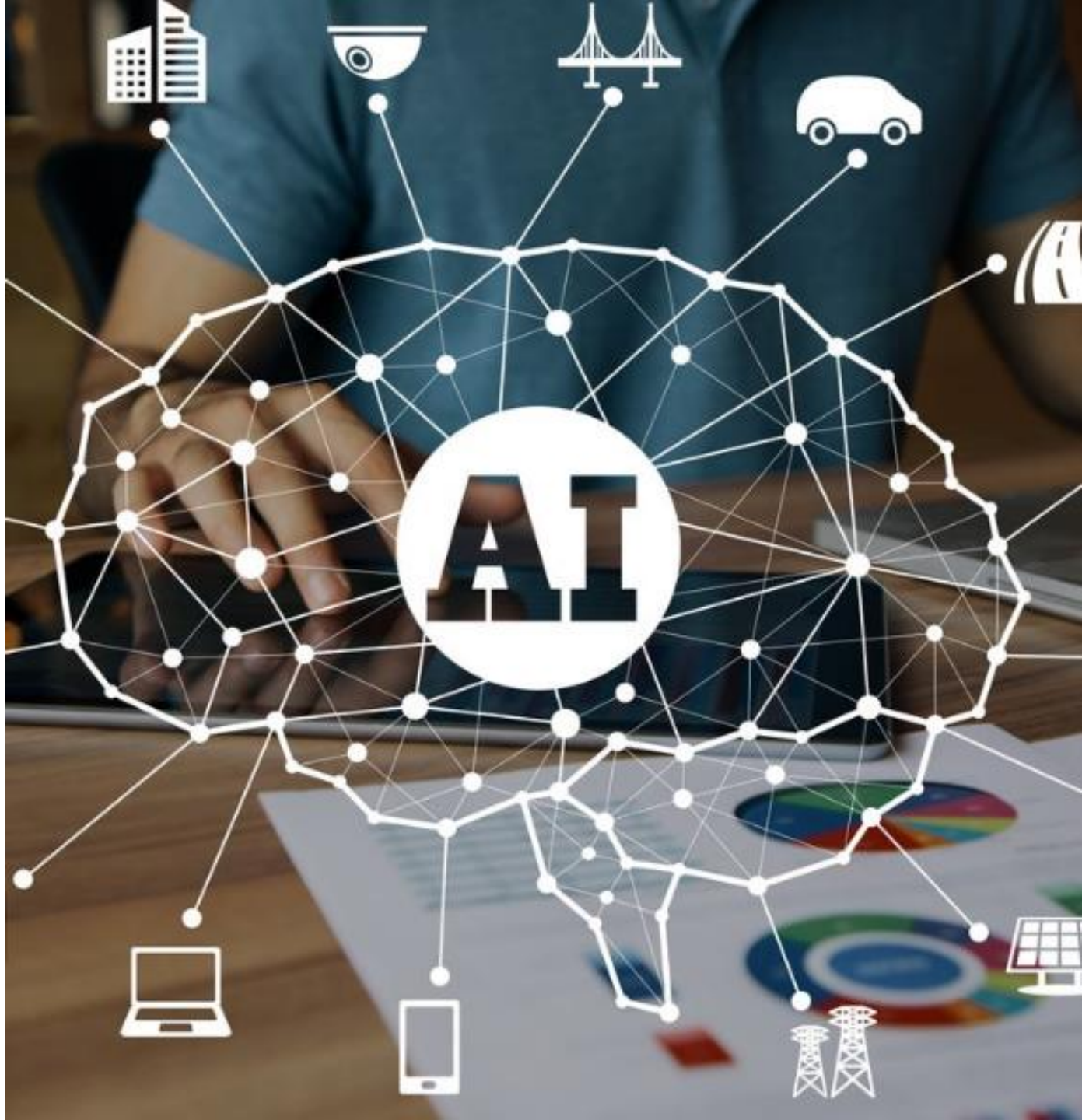


THE "COMPLIANCE" BENEFITS OF AWS INFRASTRUCTURE FOR GENERATIVE AI APPLICATIONS

- Comprehensive compliance programs
- Audit and reporting tools
- Data residency and sovereignty
- Security certifications
- Automated compliance checks

THE "RESPONSIBILITY" BENEFITS OF AWS INFRASTRUCTURE

- The "responsibility" benefits of using AWS infrastructure for generative AI applications focus on ethical AI practices and accountability
- Here are some key points:
  - Ethical AI development
  - Bias mitigation
  - Data privacy
  - Accountability
  - Sustainability



THE "SAFETY" BENEFITS OF AWS INFRASTRUCTURE FOR GENERATIVE AI APPLICATIONS

Robust security measures

Compliance with security standards

Data isolation

Continuous monitoring and threat detection

Disaster recovery and backup

THE "RESPONSIVENESS" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- The "responsiveness" cost tradeoff of AWS generative AI services involves balancing the speed and efficiency of AI operations with the associated costs
- Choosing the right model is crucial
  - High-performance models may offer faster response times but can be more expensive to run
  - Conversely, smaller models might be more cost-effective but could have slower response times

THE "RESPONSIVENESS" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- AWS allows for allocating resources dynamically
  - While allocating more resources can improve responsiveness, it also increases costs
  - Efficient resource management is essential to balance performance and cost
- The cost of using generative AI models depends on the number of tokens processed
  - Optimizing token usage can help reduce costs while maintaining acceptable response times

THE "RESPONSIVENESS" COST TRADEOFF

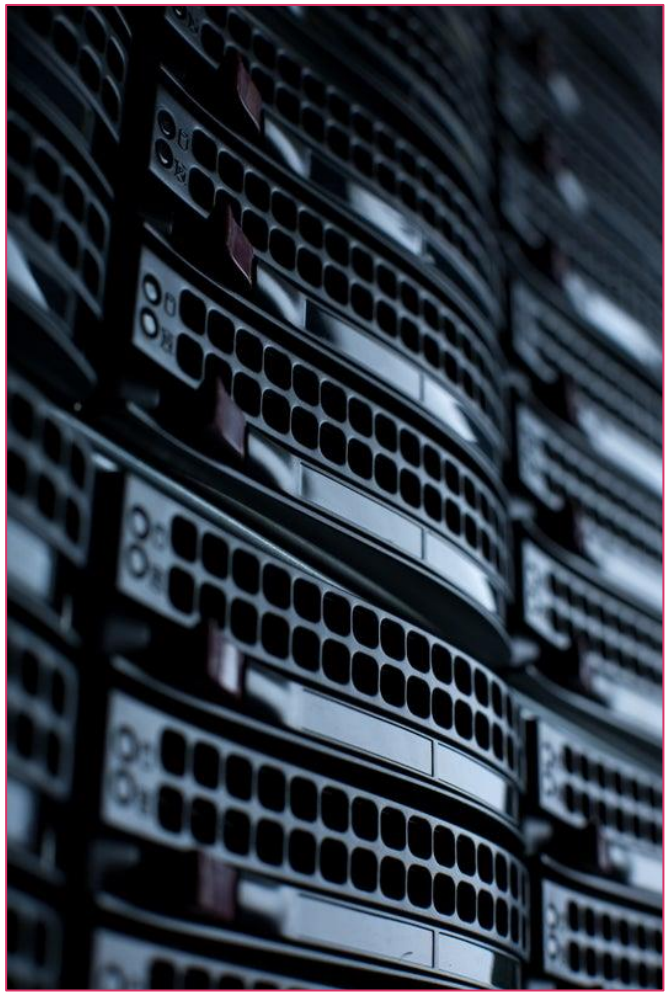
- Implementing caching strategies can improve responsiveness by reducing the need for repeated computations
  - However, setting up and maintaining caches can add to the overall cost
- AWS's scalable infrastructure allows you to handle increased demand without compromising responsiveness
  - However, scaling up resources to meet high demand can lead to higher costs

THE "AVAILABILITY" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- Allocating resources to maintain high availability, such as additional compute instances and storage, can lead to higher operational expenses
  - AWS provides tools like AWS Cost Explorer and AWS Budgets to help you monitor and manage costs associated with maintaining high availability
  - By setting budgets and alerts, you can optimize resource usage and control expenses

THE "REDUNDANCY" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- The "redundancy" cost tradeoff of AWS generative AI services involves balancing the need for high availability and fault tolerance with the associated costs
- Maintaining redundant systems can increase costs due to the need for additional resources

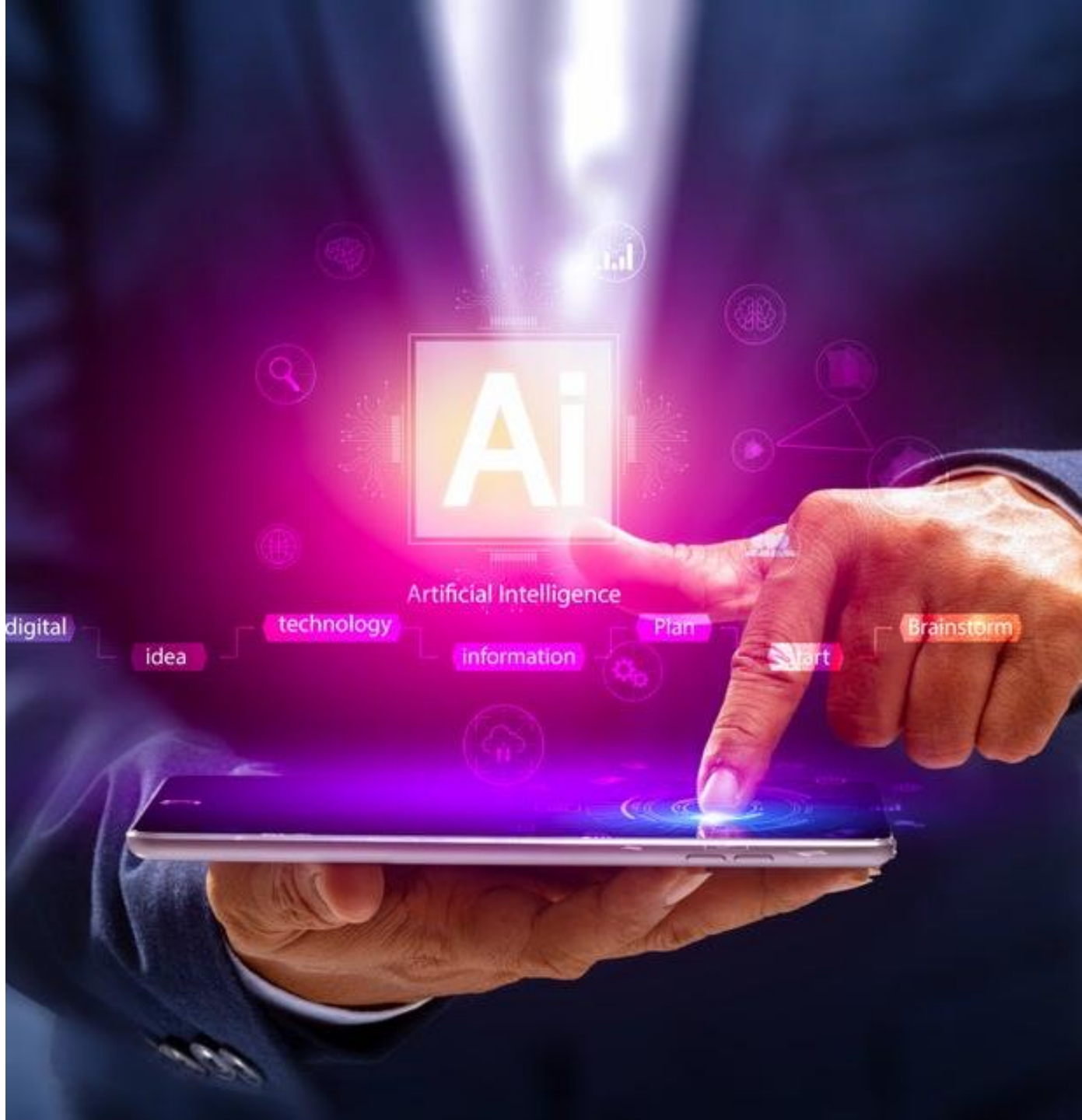


THE "REDUNDANCY" COST TRADEOFF

- Geographic distribution of applications across multiple geographic regions can enhance redundancy and reduce the risk of downtime, but it can also lead to higher costs due to data transfer and storage across regions
- While backup and recovery measures protect your data and applications, they also require additional storage and resources, contributing to higher costs

THE "PERFORMANCE" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- Utilizing high-performance models will often require more computational resources, leading to higher costs
- Allocating more resources, such as GPU instances, can enhance performance but also increase operational expenses
- Leveraging optimized algorithms can help achieve a balance between performance and cost-effectiveness



THE "PERFORMANCE" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- While scaling up can improve performance, it also leads to higher costs
  - Efficient scalability management is essential to balance these factors
- AWS offers tools like AWS Cost Explorer and AWS Budgets to help monitor and manage costs associated with high-performance requirements
  - By setting budgets and optimizing resource usage, you can achieve a balance between performance and cost

THE "TOKEN-BASED PRICING" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- AWS charges for every input token processed and every output token generated by the AI models, so this means that the more tokens the application uses, the higher the cost
- Choosing efficient models that require fewer tokens to achieve the desired results can help reduce costs
- Optimizing prompts and responses to minimize token usage is crucial

THE "TOKEN-BASED PRICING" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- Implementing caching strategies can reduce the number of tokens processed by reusing previous responses for similar queries
- AWS offers batch processing options that allow you to process multiple prompts in a single request, which can be more cost-effective than processing each prompt individually

THE "PROVISION THROUGHPUT" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- Provisioned throughput is a time-based commitment that ensures that the application meets performance requirements consistently
- It is typically lower per unit of usage compared to on-demand, but it requires a commitment
  - If consistent usage is expected, then provisioned throughput can save costs
- Choosing between these options depends on the application's usage patterns and performance requirements

THE "CUSTOM MODEL" COST TRADEOFF OF AWS GENERATIVE AI SERVICES

- Custom models offer improved performance and relevance for your specific tasks, but at a higher cost
  - Pre-trained models are more cost-effective but may not perform as well for specialized tasks
- Training custom models requires significant computational resources, which can increase costs
  - However, the improved performance can lead to better outcomes and efficiency in your applications

DESIGN CONSIDERATIONS FOR APPLICATIONS USING FOUNDATION MODELS

Objectives

- Select the criteria to choose pre-trained models
- Explore the effect of inference parameters on model responses
- Define Retrieval-Augmented Generation (RAG)
- Learn how to store embeddings within vector databases
- Understand cost tradeoffs with foundation model customization
- Compare the role of agents in multi-step tasks

SELECTION CRITERIA TO CHOOSE PRE-TRAINED MODELS

The "cost" selection criteria for choosing pre-trained models typically involves several factors to ensure the best model is selected for a specific task

- **Computational cost**: Includes the resources needed to train and fine-tune the model, such as GPU/CPU time and memory usage
- **Performance**: The accuracy and effectiveness of the model on the target task are crucial
  - Models that perform better on similar tasks or datasets are often selected

SELECTION CRITERIA TO CHOOSE PRE-TRAINED MODELS

- **Scalability**: The ability of the model to scale with larger datasets and more complex tasks is important
  - Models that can handle increased data and complexity without significant performance degradation are preferred
- **Transfer learning capability**: The ability of the model to transfer knowledge from one task to another is a key cost factor
- **Historical performance data**: Past performance data on similar tasks can guide the selection process

PRE-TRAINED MODEL MODALITIES

Object detection

Natural language processing

Data extraction

Feature engineering

Image classification

MULTI-LINGUAL CRITERIA

- Models that support multiple languages are preferred, especially for applications that need to cater to a global audience
- The model should maintain high accuracy across different languages, ensuring consistent results
- Training on diverse datasets that include multiple languages will likely perform well in multi-lingual scenario

MODEL SIZE AND COMPLEXITY CRITERIA

- **Number of parameters**: Larger models with more parameters can capture more complex patterns in data but require more computational resources and memory
- **Resource requirements**: Larger models need more powerful hardware and incur higher costs for hosting and inference

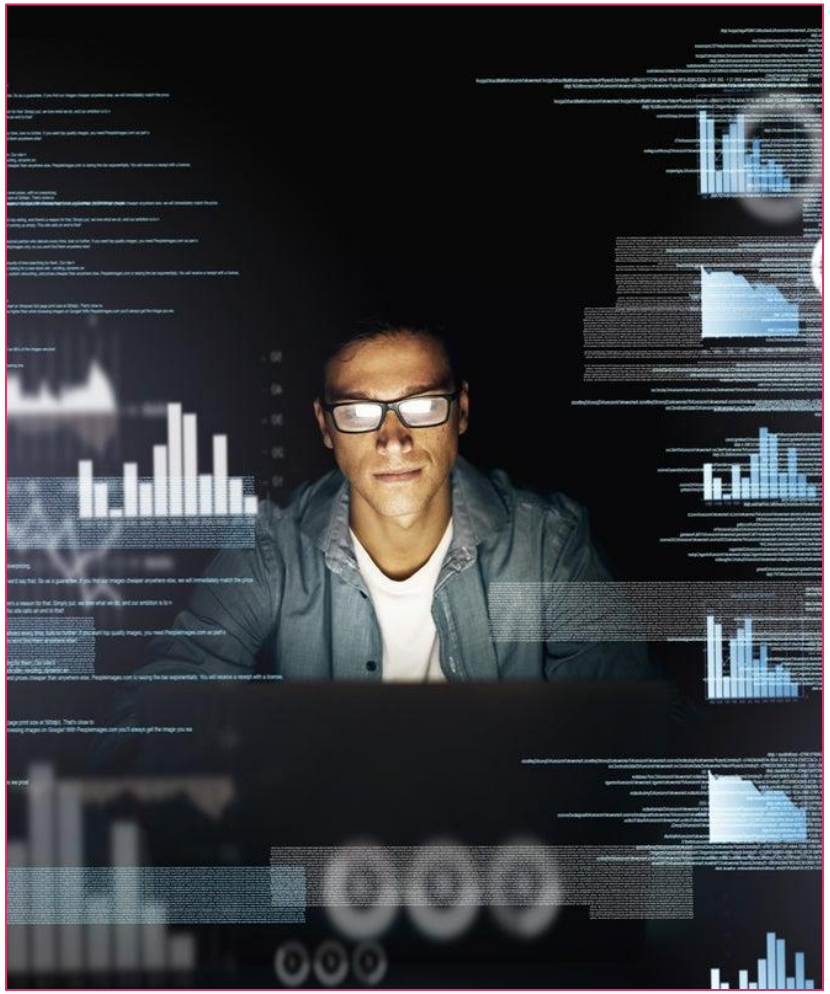
MODEL SIZE AND COMPLEXITY CRITERIA

- **Task requirements**: Not all tasks require the largest models
  - Smaller models can be just as effective for certain tasks, offering a balance between performance and cost
- **Efficiency**: Smaller models are more efficient to train and deploy, saving on hardware costs and reducing deployment time

LATENCY CRITERIA

- Models optimized for low-latency performance are preferred for real-time applications, such as customer service chatbots and interactive coding assistants
- AWS offers latency-optimized inference options for certain models, which deliver faster response times without compromising accuracy
- Users can set the "latency" parameter to "optimized" while calling the Bedrock runtime API to access latency-optimized inference

CUSTOMIZATION CRITERIA



Fine-tuning

Adjusting model parameters to better fit the new data model

Continued pre-training

Using unlabeled data to further train a foundation model, familiarizing it with specific types of inputs

Data preparation

Preparing a training dataset and, if applicable, a validation dataset is crucial

Security configurations

Allowing for additional security measures, such as encrypting input and output data

INPUT/OUTPUT LENGTH CRITERIA

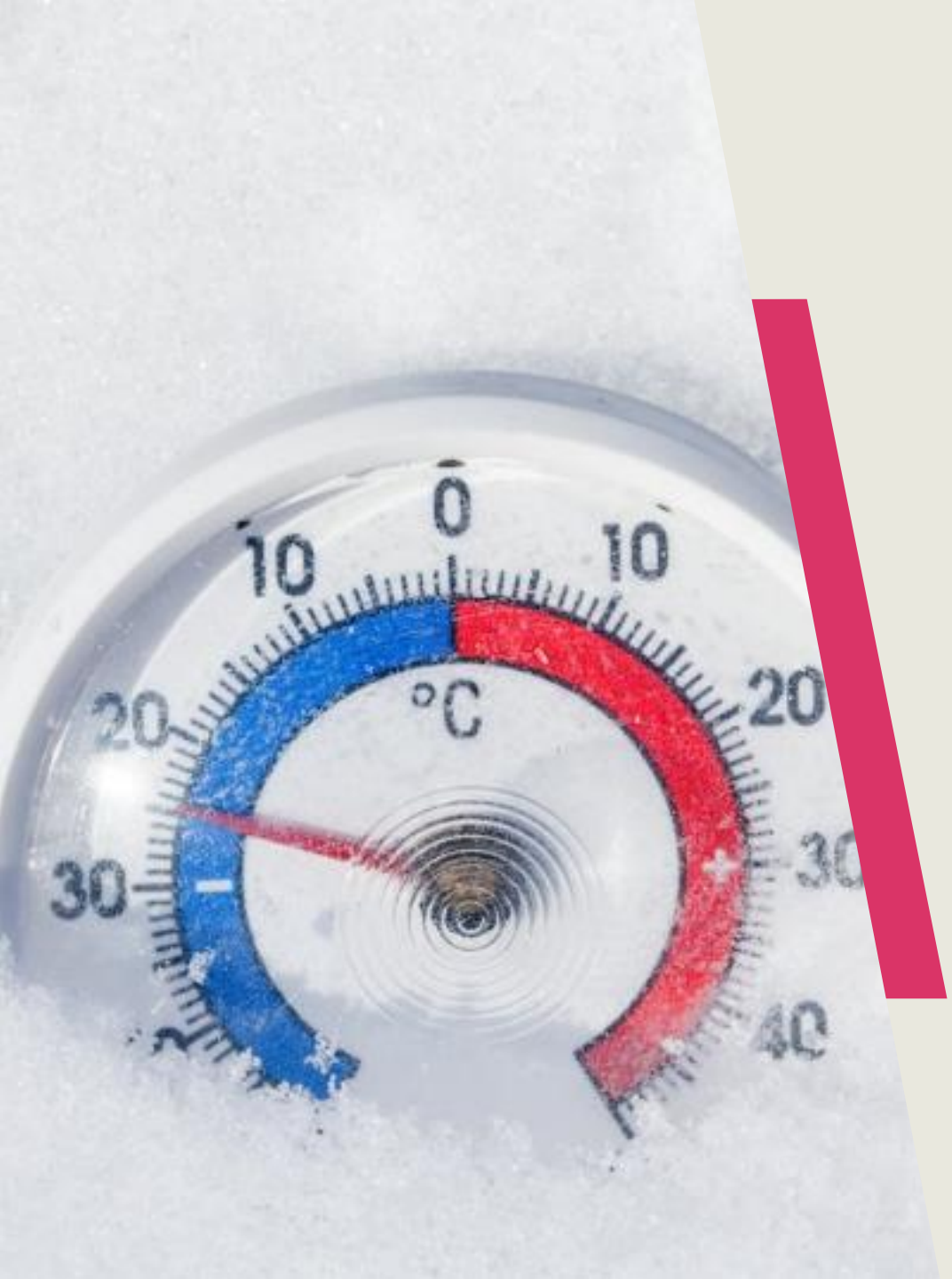
- Input length is the maximum length of the input that the model can handle
  - This is crucial for tasks that involve long documents or sequences, such as text summarization or translation
- Output length is the maximum length of the output that the model can generate
  - This is important for tasks that require detailed responses, such as content generation or question answering

INPUT/OUTPUT LENGTH CRITERIA

- **Task requirements**: Different tasks may have different input and output length requirements
  - For example, a chatbot may need to handle shorter inputs and outputs, while a document summarization model may need to handle longer texts
- **Model limitations**: Some models may have limitations on the maximum input and output lengths they can handle
  - It is important to choose a model that can meet the specific requirements of your task

THE EFFECT OF INFERENCE PARAMETERS ON MODEL RESPONSES

- The **temperature** inference parameter in AWS affects the randomness and diversity of model responses
- It modulates the probability distribution for the next token in the sequence



LOWER TEMPERATURE

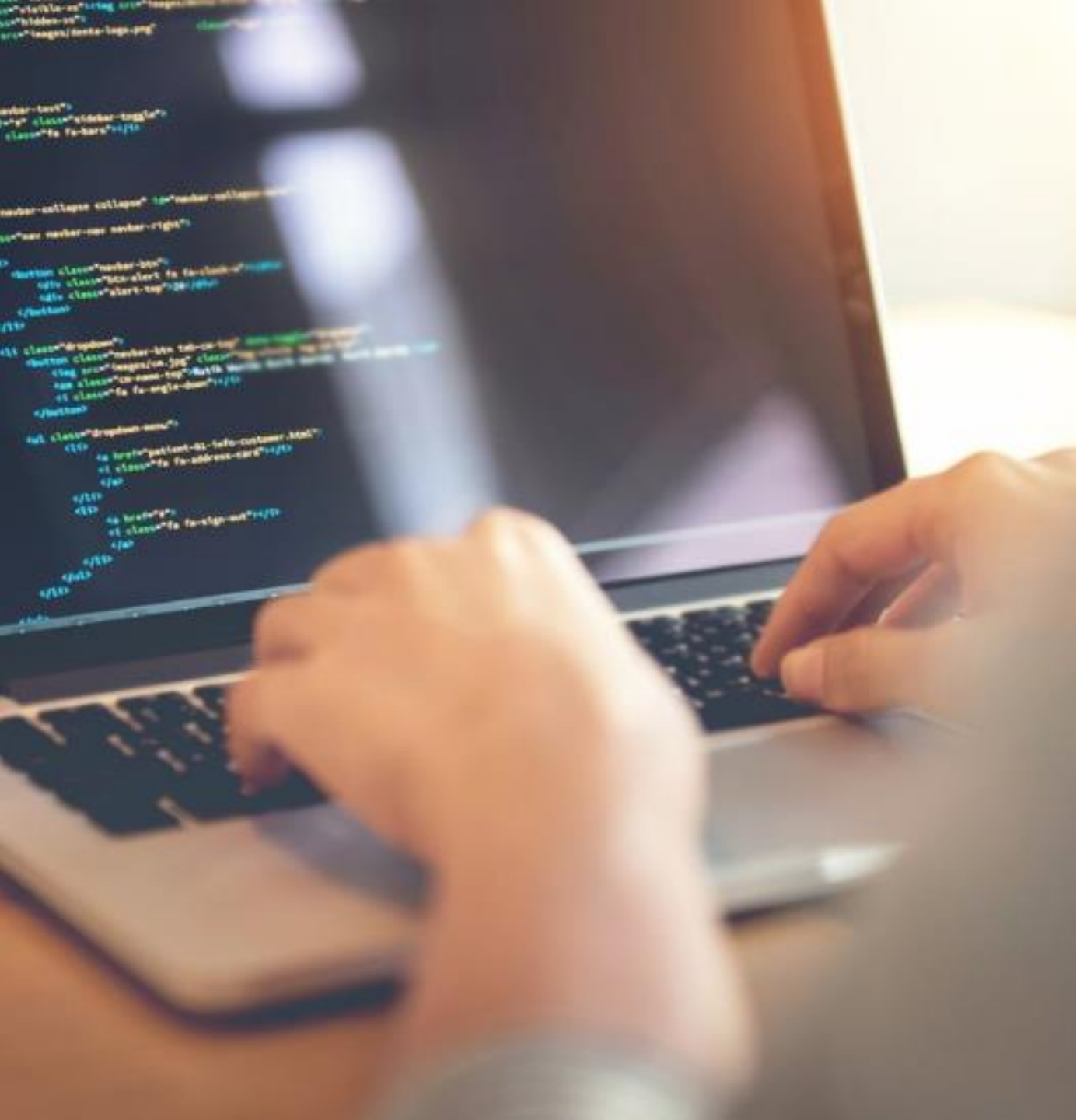
- In technical terms, a lower temperature value makes the model's responses more focused and consistent
- Lower temperature steepens the probability distribution, making the model more likely to select higher-probability outputs
- As a result, the responses are more deterministic and predictable

HIGHER TEMPERATURE

- A higher temperature value introduces more variation and creativity
- Higher temperature flattens the probability distribution, increasing the likelihood of the model selecting lower-probability outputs
- This leads to more random and diverse responses

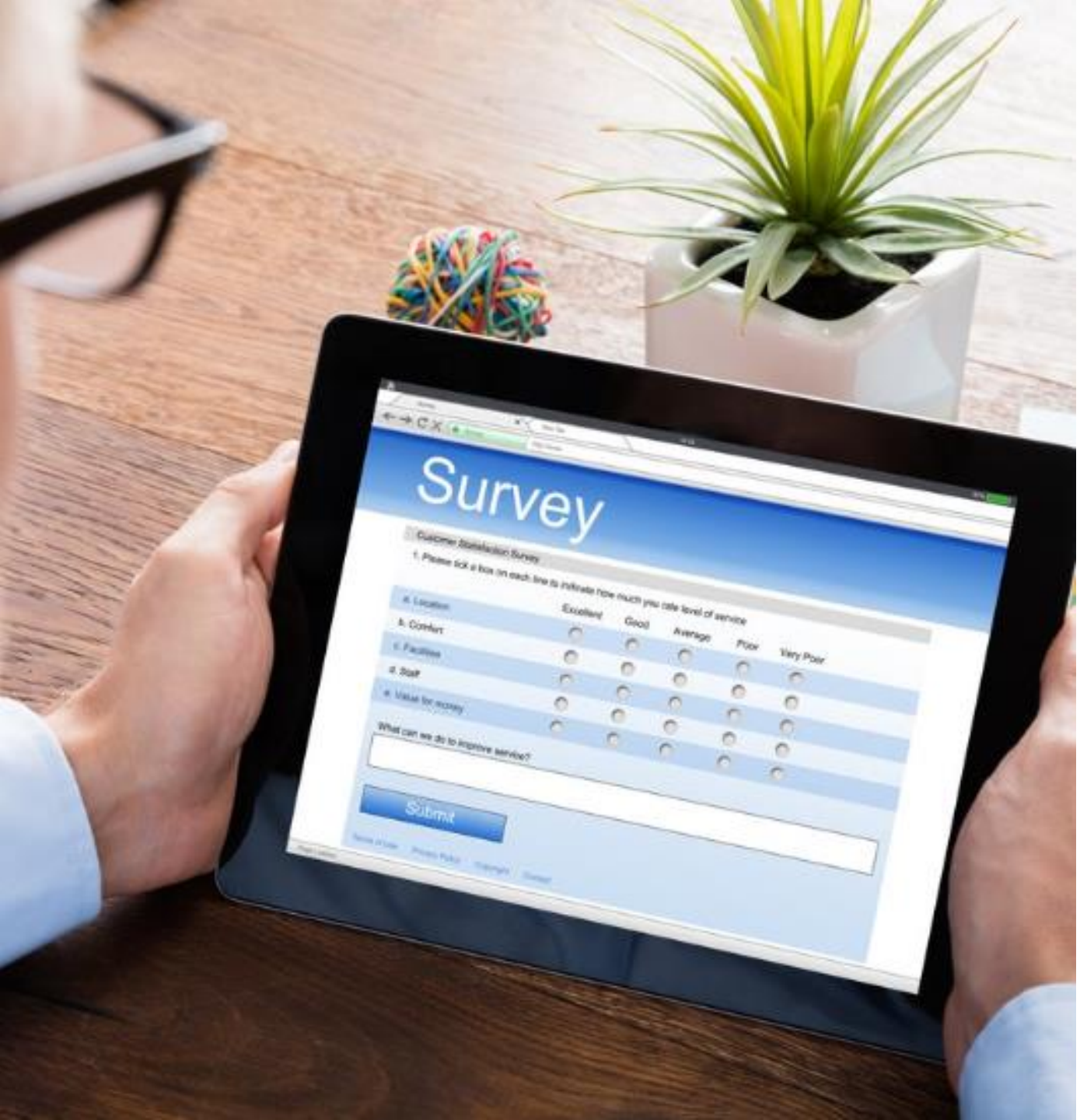
INPUT/OUTPUT LENGTH

- In a nutshell, the input/output length parameter helps manage the complexity and verbosity of the model's responses
- It can help to ensure that they are appropriate for the given context



INPUT LENGTH

- The input length parameter determines the maximum length of the input sequence that the model can process
- If the input exceeds this length, it will be truncated
  - This ensures that the model can handle the input efficiently without running into memory or processing constraints

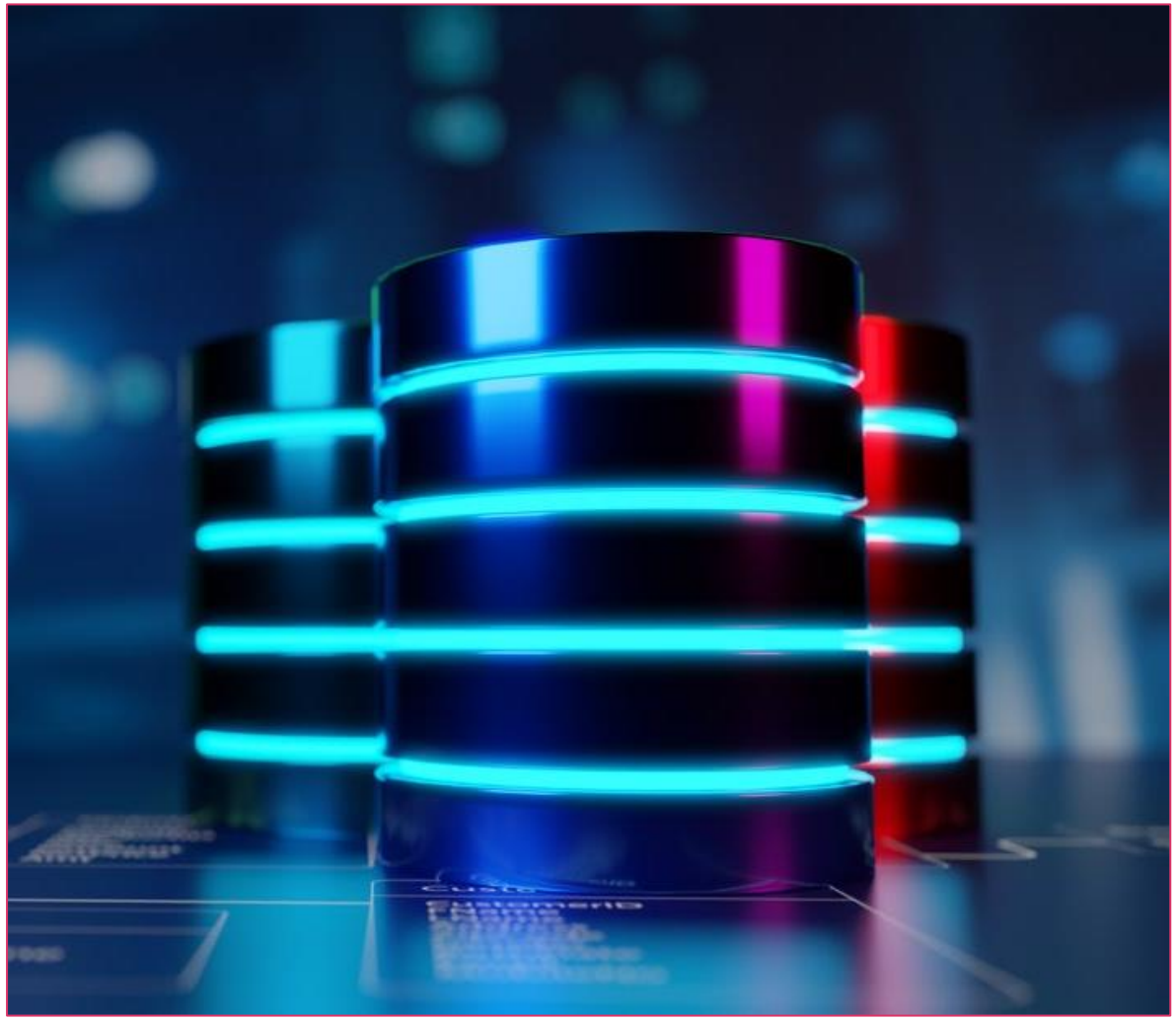


OUTPUT LENGTH

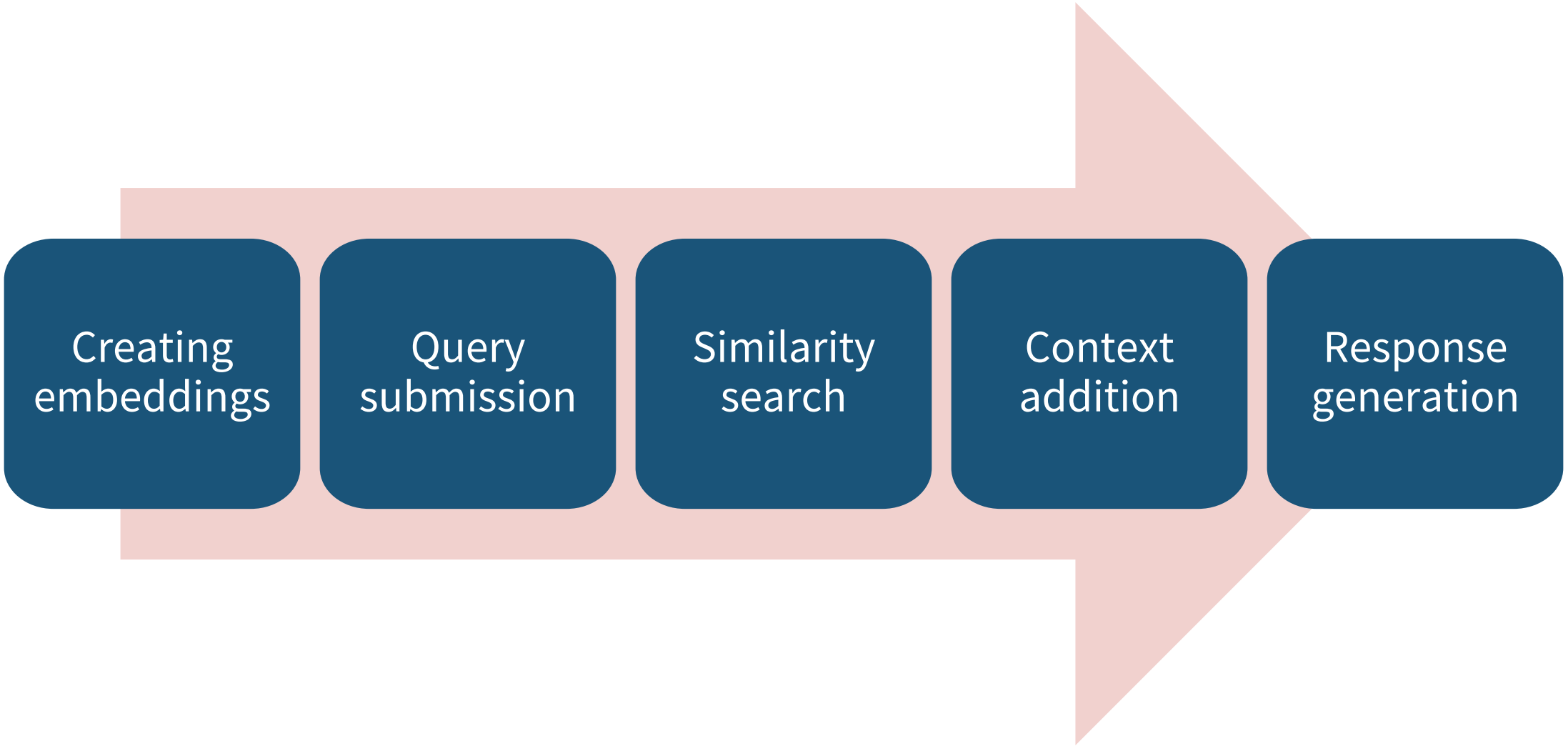
- The output length parameter sets the maximum length of the output sequence that the model can generate
- By adjusting this parameter, you can control how verbose or concise the model's responses are
- A shorter output length will result in more concise responses, while a longer output length allows for more detailed and elaborate responses

RETRIEVAL-AUGMENTED GENERATION (RAG)

- Retrieval-Augmented Generation (RAG) is a method that improves the performance of a large language model by integrating external data sources, like a company's internal documents or trusted knowledge bases
- RAG helps ensure that the model's responses are accurate, relevant, and specific to the given context, without the need to retrain the model



RETRIEVAL-AUGMENTED GENERATION PROCESS



RAG APPLICATIONS WITH AMAZON BEDROCK

- **Question answering**: RAG can be used to build systems that provide accurate and contextually relevant answers to user queries by retrieving information from a company's internal documents or knowledge bases
- **Contextual chatbots**: Businesses can develop chatbots that offer personalized and context-aware responses, improving customer service and user experience

RAG APPLICATIONS WITH AMAZON BEDROCK

- **Personalized search**: RAG enables the creation of search engines that deliver highly relevant search results by incorporating user-specific data and preferences
- **Content generation**: Companies can use RAG to generate content that is tailored to specific audiences, leveraging proprietary data to ensure accuracy and relevance

RAG APPLICATIONS WITH AMAZON BEDROCK

- **Multi-tenant applications**: For software as a service (SaaS) providers, RAG can help deliver personalized experiences for each customer by using tenant-specific data while ensuring data isolation and security
- **Metadata filtering**: RAG applications can benefit from intelligent metadata filtering, which refines search results based on custom metadata attributes, reducing noise and irrelevant information

STORING EMBEDDINGS WITHIN VECTOR DATABASES

- **Amazon OpenSearch Service** supports vector search capabilities, allowing customers to store and search vector embeddings efficiently
- One can create a vector search collection, upload embeddings, and perform similarity searches using k-nearest neighbor (k-NN) algorithms

AMAZON OPENSEARCH PROCESS

Configure permissions

Create a collection

Upload data

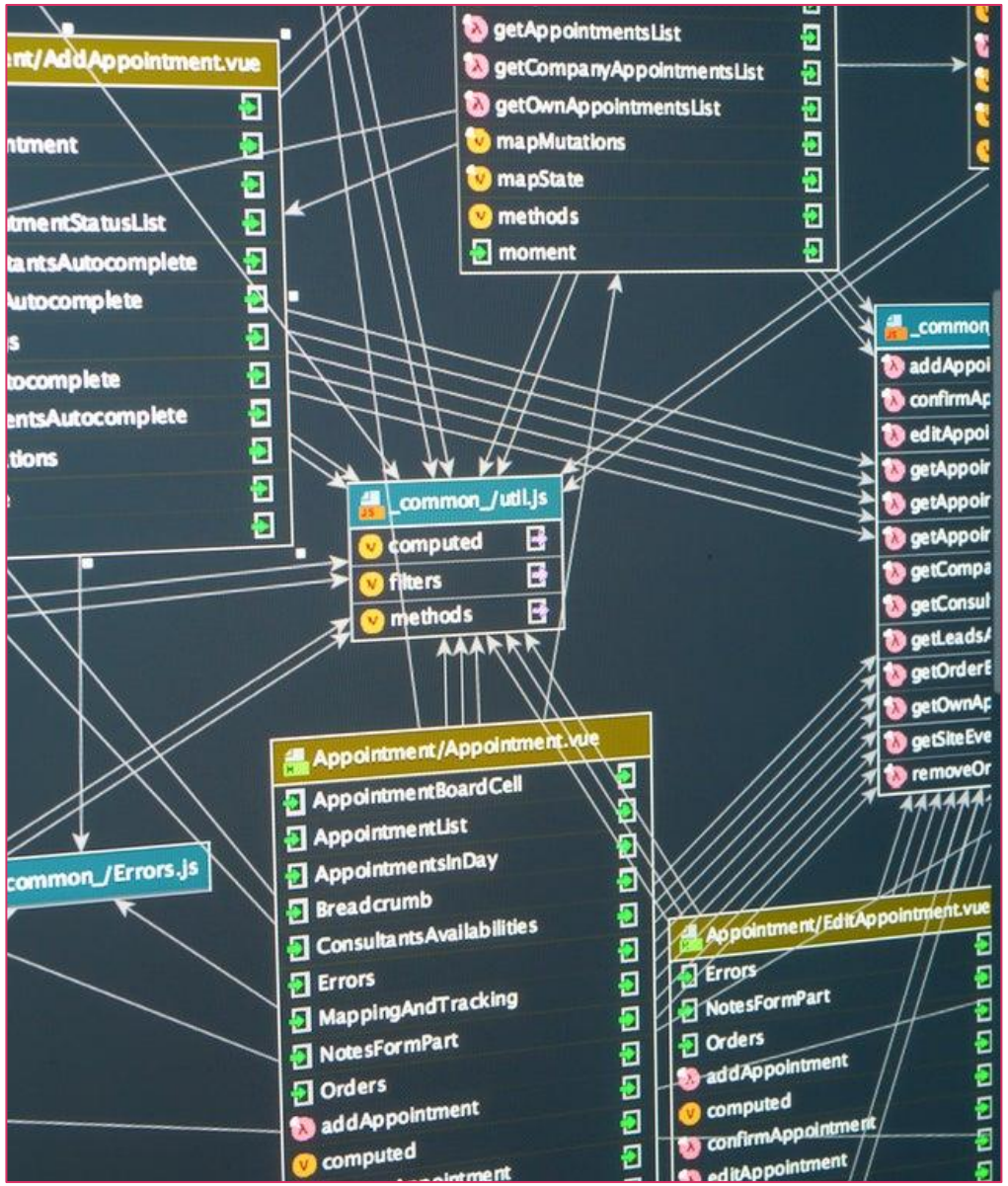
Search data

AMAZON AURORA

- Amazon Aurora PostgreSQL-Compatible Edition supports the **pgvector** extension, which allows customers to store and manipulate vector embeddings directly within the relational database
- The steps include:
  1. Set up Aurora: Create an Aurora PostgreSQL cluster
  2. Enable **pgvector**: Install and enable the pgvector extension
  3. Store embeddings: Insert the vector embeddings into the database
  4. Perform searches: Use SQL queries to perform similarity searches on the stored embeddings

AMAZON RDS FOR POSTGRESQL

- RDS for PostgreSQL also supports the **pgvector** extension, enabling one to store and search vector embeddings within your relational database
- The process is like Amazon Aurora explained on the previous slide



AMAZON NEPTUNE

- Amazon Neptune supports vector indexing within Neptune Analytics, enabling customers to store and search vector embeddings in a graph database
- The process involves:
  1. Create a vector index: Specify the index dimension when creating a Neptune Analytics graph
  2. Load vectors: Load the vector embeddings from graph data files or Amazon S3
  3. Search vectors: Perform similarity searches using vector search algorithms

AMAZON DOCUMENTDB

- DocumentDB (with MongoDB compatibility) supports vector search and lets customers store, index, and search vector embeddings within a document database
- The steps include:
  1. Create a collection: Set up a collection in DocumentDB
  2. Insert vectors: Insert the vector embeddings into the collection
  3. Create a vector index: Create an index on the vector field
  4. Search vectors: Perform similarity searches using the vector index

COST TRADEOFFS WITH FOUNDATION MODEL CUSTOMIZATION

- **Pre-training** foundation models is a significant investment, but it offers unique advantages and tradeoffs
- Pre-training requires substantial computational resources and time
- Pre-trained models can be scaled across various applications leading to cost savings in the long run
- Utilizing pre-trained models can optimize resource usage which can be more efficient and cost-effective

COST TRADEOFFS OF FINE-TUNING IN FOUNDATION MODEL CUSTOMIZATION

- **Fine-tuning** requires a significant investment in computational resources and labeled data which can be expensive, especially for large-scale models
- Fine-tuned models can offer improved performance and lower latency during inference, as there is no additional retrieval step involved
- Fine-tuned models may need periodic retraining to incorporate new data and maintain performance
  - This ongoing maintenance can add to the overall cost

COST TRADEOFFS OF IN-CONTEXT LEARNING IN FOUNDATION MODEL CUSTOMIZATION

- **In-context learning** typically requires less computational power and resources compared to pre-training or fine-tuning, which can result in lower initial costs
- In-context learning reduces the need for ongoing maintenance and retraining, which can further lower costs over time
  - However, it may require continuous updates to the context data to ensure accuracy and relevance

COST TRADEOFFS OF RAG IN FOUNDATION MODEL CUSTOMIZATION

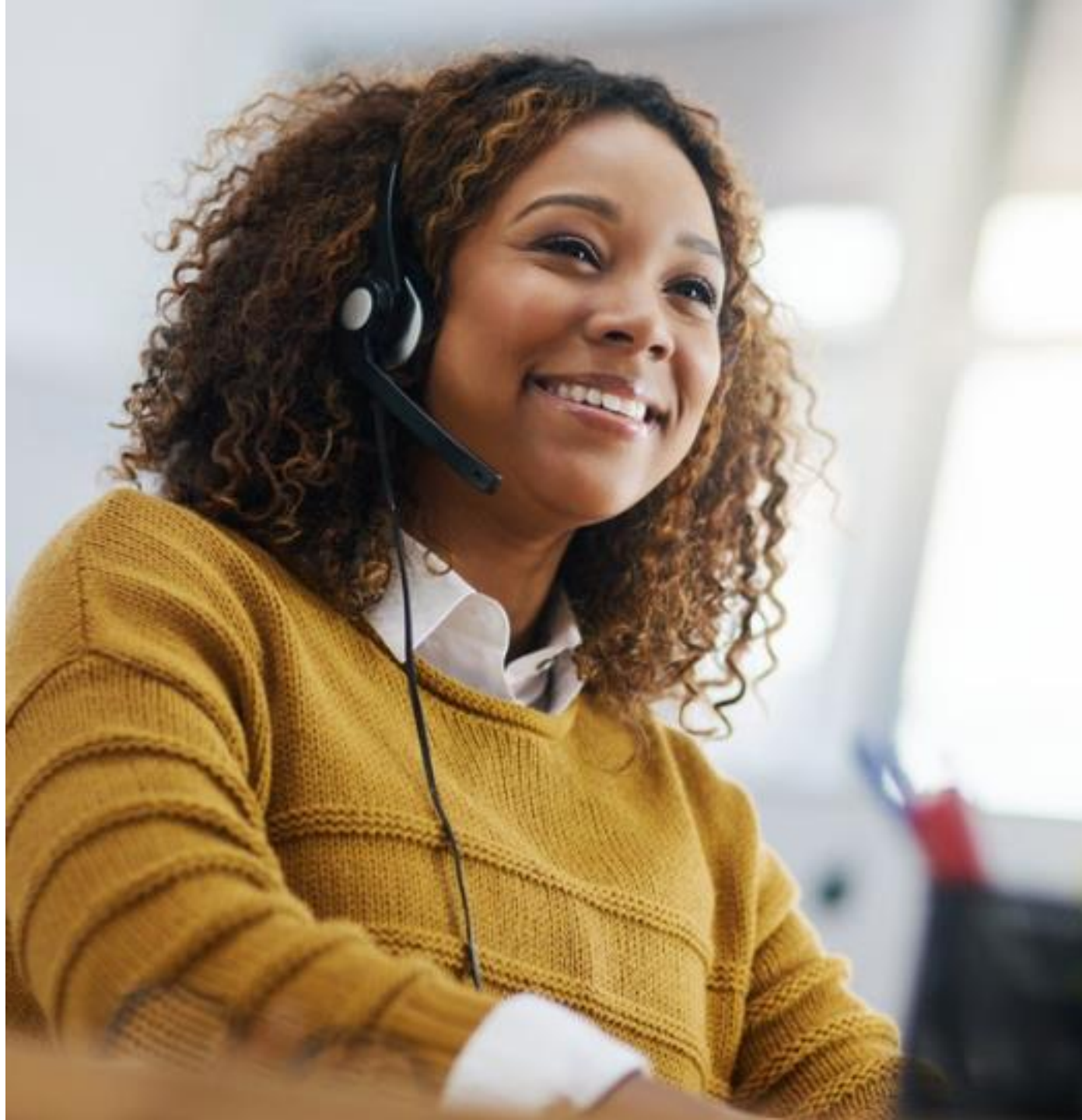
- Implementing **RAG** requires setting up infrastructure to manage external knowledge sources and retrieval mechanisms
  - This can involve costs related to data storage, indexing, and retrieval systems
- RAG involves ongoing costs for maintaining and updating the external knowledge sources
  - This includes costs for data ingestion, processing, and ensuring the data remains current and relevant

COST TRADEOFFS OF RAG IN FOUNDATION MODEL CUSTOMIZATION

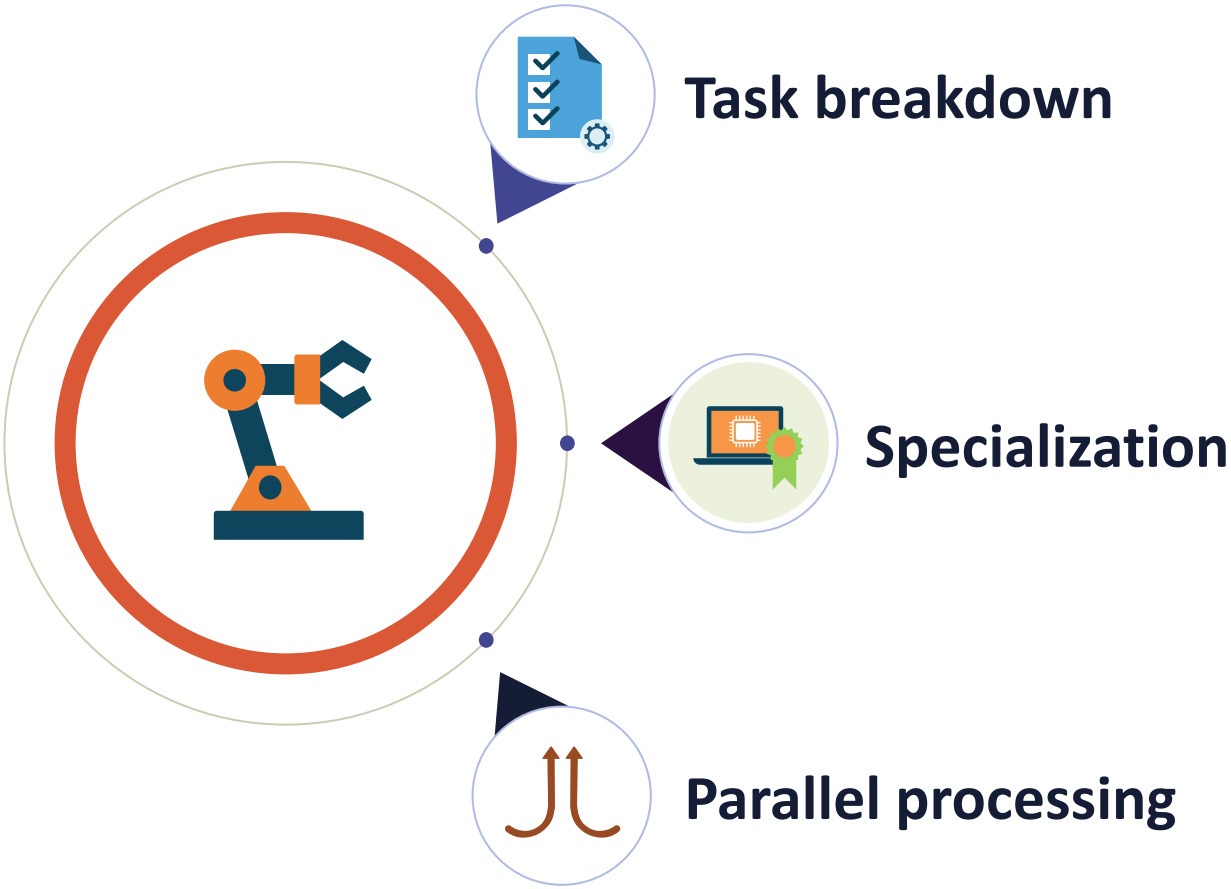
- Integrating RAG with existing systems and workflows can incur additional costs
  - This includes costs for developing and maintaining the integration, as well as potential costs for training staff to use the new system effectively
- Organizations also need to consider the trade-offs between scalability/performance and cost
  - AWS provides managed services like Amazon Bedrock Knowledge Bases to help manage these costs

THE ROLE OF AGENTS IN MULTI-STEP TASKS

- In multi-step tasks, agents play a crucial role in efficiently managing and executing complex processes
- Agents in multi-step tasks enhance efficiency, accuracy, and flexibility by breaking down tasks, specializing in specific roles, processing tasks in parallel, coordinating efforts, and dynamically adapting to changes



THE ROLE OF AGENTS IN MULTI-STEP TASKS



THE ROLE OF AGENTS IN MULTI-STEP TASKS

Coordination and orchestration

Dynamic adaptation

Integration with Amazon Bedrock

EFFECTIVE PROMPT ENGINEERING TECHNIQUES

Objectives

- Compare chain-of-thought, zero-shot, single-shot, and few shot prompt engineering
- Explore prompt engineering with templates
- Learn the benefits and best practices for prompt engineering
- Describe the potential risks and limitations of prompt engineering

PROMPT ENGINEERING CONTEXT

- According to AWS, prompt engineering involves crafting and optimizing input prompts to guide generative AI solutions to produce desired outputs
- The "**context**" concept in prompt engineering refers to providing detailed information and background within the prompt to help the AI generate accurate and relevant response

CONCEPTS AND CONSTRUCTS OF PROMPT ENGINEERING

- This context can include specific instructions, relevant data, and any other information that can help the AI understand the task better
- Systematically designing prompts with the right context helps achieve more meaningful and usable outputs from the AI system

PROMPT ENGINEERING INSTRUCTION

- The "**instruction**" concept in prompt engineering involves providing clear and specific directives within the prompt to guide the AI's response
  - These instructions help the AI understand the task at hand and generate outputs that align with the user's expectations
- This is particularly important when dealing with complex tasks or when the AI needs to follow a specific format or style

PROMPT ENGINEERING NEGATIVE PROMPTS

- **Negative prompts** involve specifying elements that should be excluded from the AI's output
- This is done using the **negativeText** parameter
- By including negative prompts, one can guide the AI to avoid certain objects, styles, or characteristics that might otherwise appear in the generated output
  - For example, if a graphic of a beach without waves is desired, a negative prompt can be used to exclude that element

MODEL LATENT SPACE

- The "**model latent space**" concept in prompt engineering refers to the high-dimensional space where the AI model represents and processes the input data
- This space is where the model captures and organizes the underlying patterns and relationships within the data

MODEL LATENT SPACE

- By navigating this latent space, the AI can generate meaningful and relevant outputs based on the input prompts
- The quality and structure of the prompts can significantly influence how effectively the model explores and utilizes this latent space to produce desired results

CHAIN-OF-THOUGHT PROMPT ENGINEERING

- **Chain of Thought** prompt engineering is a technique used to guide AI models to generate more coherent and logical responses by providing a sequence of thoughts or steps in the prompt.
- This approach helps the model understand the context and reasoning behind the task, leading to more accurate and relevant outputs.

CHAIN-OF-THOUGHT PROMPT ENGINEERING EXAMPLE

Chain-of-Thought Prompt: “Write an engaging story, following these steps:

First, introduce the main character and set the scene. Next, present a problem or challenge that the character must face.

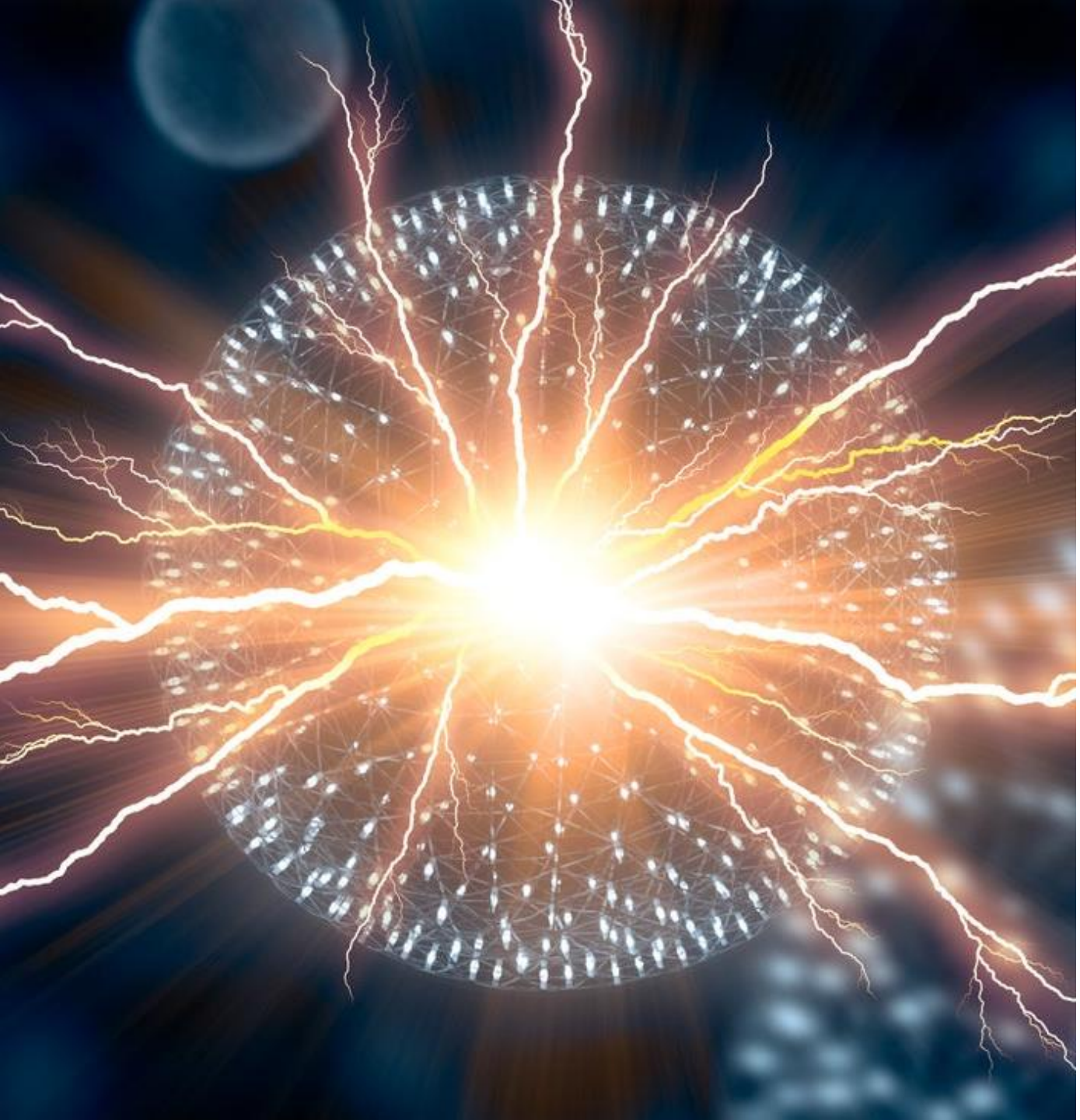
Then, describe the character's journey to overcome the challenge, including any obstacles they encounter along the way.

Finally, conclude the story with a resolution that shows how the character has grown or changed from their experiences.

For example, if the main character is a young girl who wants to save her village from a dragon, start by describing her life in the village, then introduce the dragon's threat, and follow her adventure as she ultimately defeats the dragon."

ZERO-SHOT PROMPT ENGINEERING

- Zero-shot prompt engineering is a technique where the AI model is prompted to perform a task without being provided with any specific examples of the desired output in the initial prompt
- This approach relies on the model's ability to generalize from the instructions given in the prompt to generate the appropriate response

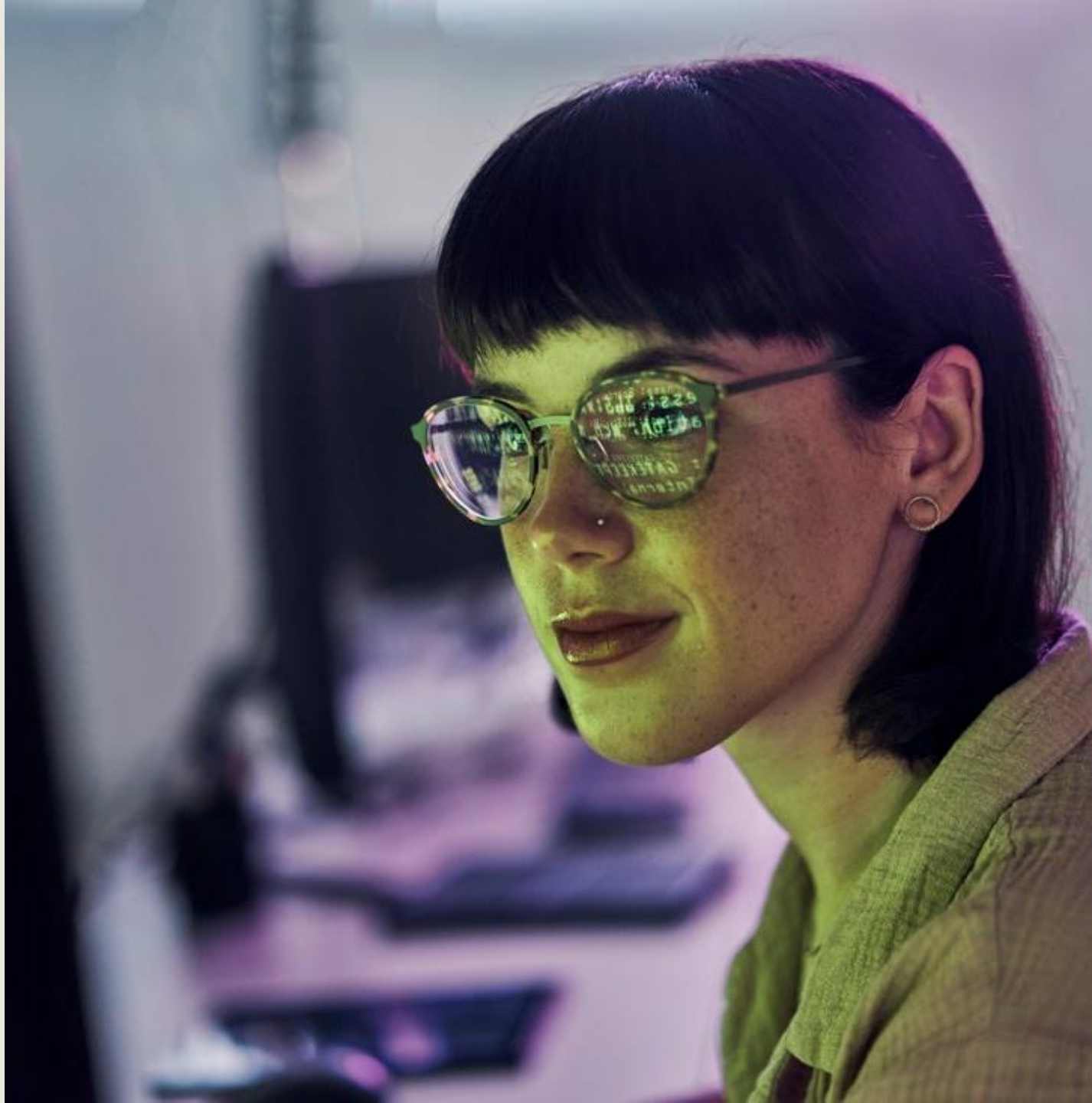


ZERO-SHOT PROMPT ENGINEERING

- **An example of NOT using this technique would be if the model were prompted to classify text into categories like sports, blockchain, or politics**
- One would construct a prompt that explicitly provides information about the target task and the desired output format, such as: "Classify the following text as either sports, blockchain, or politics: [input text]"

ZERO-SHOT PROMPT ENGINEERING

- This method allows the model to make accurate predictions even for tasks it hasn't seen before
- Zero-shot prompting is particularly useful for enabling the model to handle new tasks or generate responses to novel prompts without requiring extensive fine-tuning or additional training data



SINGLE-SHOT PROMPT ENGINEERING

- According to AWS, single-shot prompt engineering involves providing the AI model with one example of the desired output along with the input prompt
- This example helps the model understand the task better and generate more accurate responses
- Including a single example helps guide the AI to produce outputs that closely match the provided example, thereby improving the overall quality and relevance of the generated content

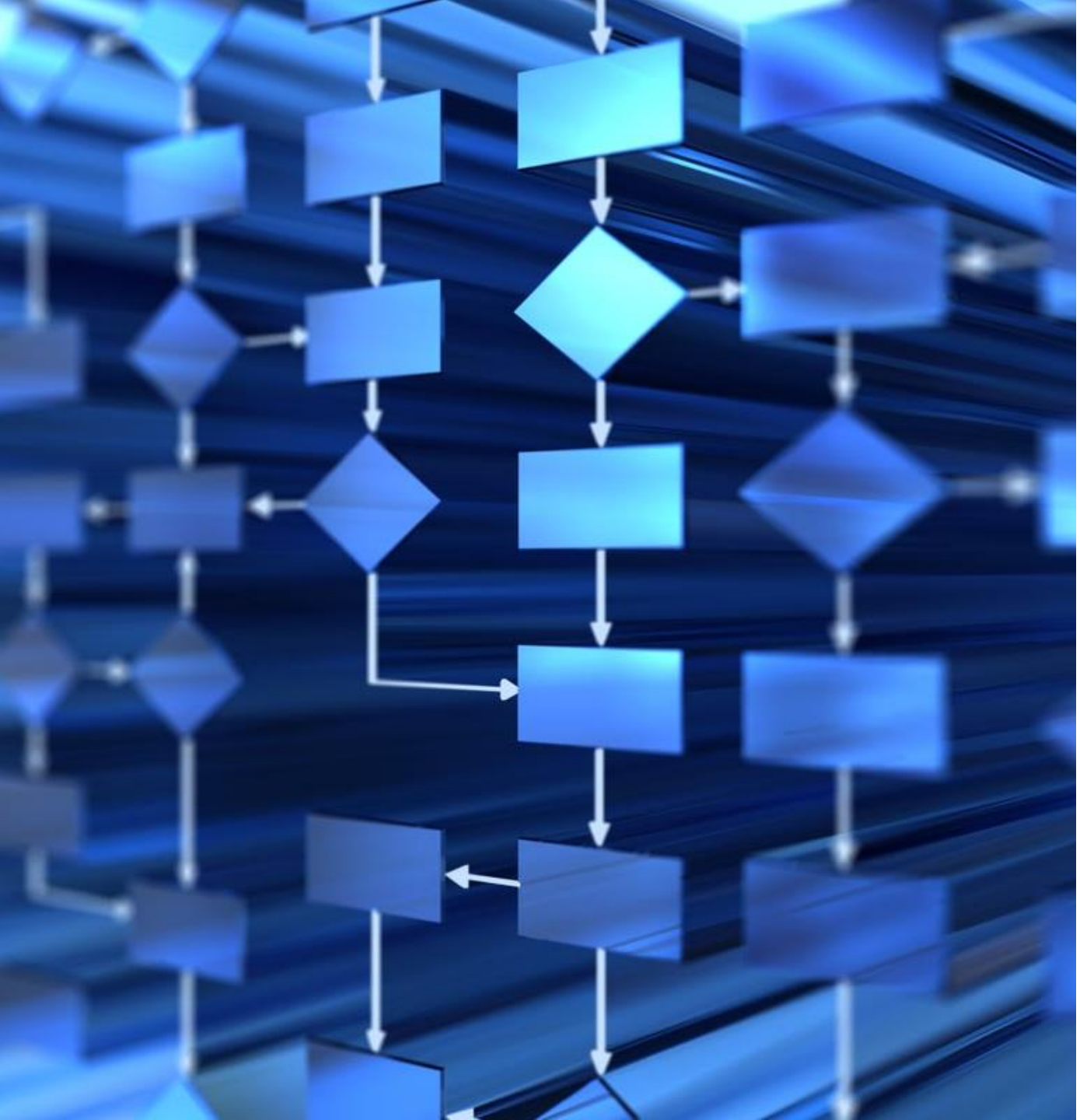
SINGLE-SHOT PROMPT ENGINEERING

Here is a simple example of single-shot prompt engineering:

- **Task**: Classify the following text as either "positive" or "negative."
- **Prompt**: "The movie was great! I loved every moment of it."
- **Example**: "Positive"
- **Input**: "The food was terrible, and the service was slow."
- **Output**: "Negative"

FEW-SHOT PROMPT ENGINEERING

- Few-shot prompt engineering involves providing the AI model with a few examples of the desired output along with the input prompt
- This technique helps the model understand the task better and generate more accurate responses by leveraging the limited examples provided



FEW-SHOT PROMPT ENGINEERING

- Few-shot prompting is particularly useful when there is a small amount of labeled data available for a new task or class
- By including a few examples, the AI can generalize from these examples to produce relevant and accurate outputs for new inputs

FEW-SHOT PROMPT ENGINEERING

Here is an example of few-shot prompt engineering:

- **Task**: Translate the following sentences from English to French.
- **Prompt:**
  1. "The cat is on the roof." -> "Le chat est sur le toit."
  2. "She is reading a book." -> "Elle lit un livre."
  3. "They are playing soccer." -> "Ils jouent au football."
- **Input**: "He is eating an apple."
- **Output**: "Il mange une pomme."

USING PROMPT ENGINEERING PROMPT TEMPLATES

In this demo…

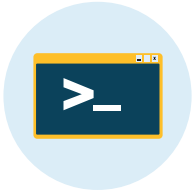
You will show how to use AWS prompt engineering prompt templates.

PROMPT ENGINEERING RESPONSE QUALITY IMPROVEMENT

- One of the key benefits of prompt engineering is the improvement in response quality
- Carefully crafting and optimizing prompts helps guide the AI model to produce more accurate, relevant, and high-quality outputs
- This is particularly important for tasks that require precise and contextually appropriate responses, such as classification, question answering, and creative writing

BEST PRACTICES FOR RESPONSE QUALITY IMPROVEMENT

Use clear and specific prompts



Provide context

Iterative refinement

Use examples

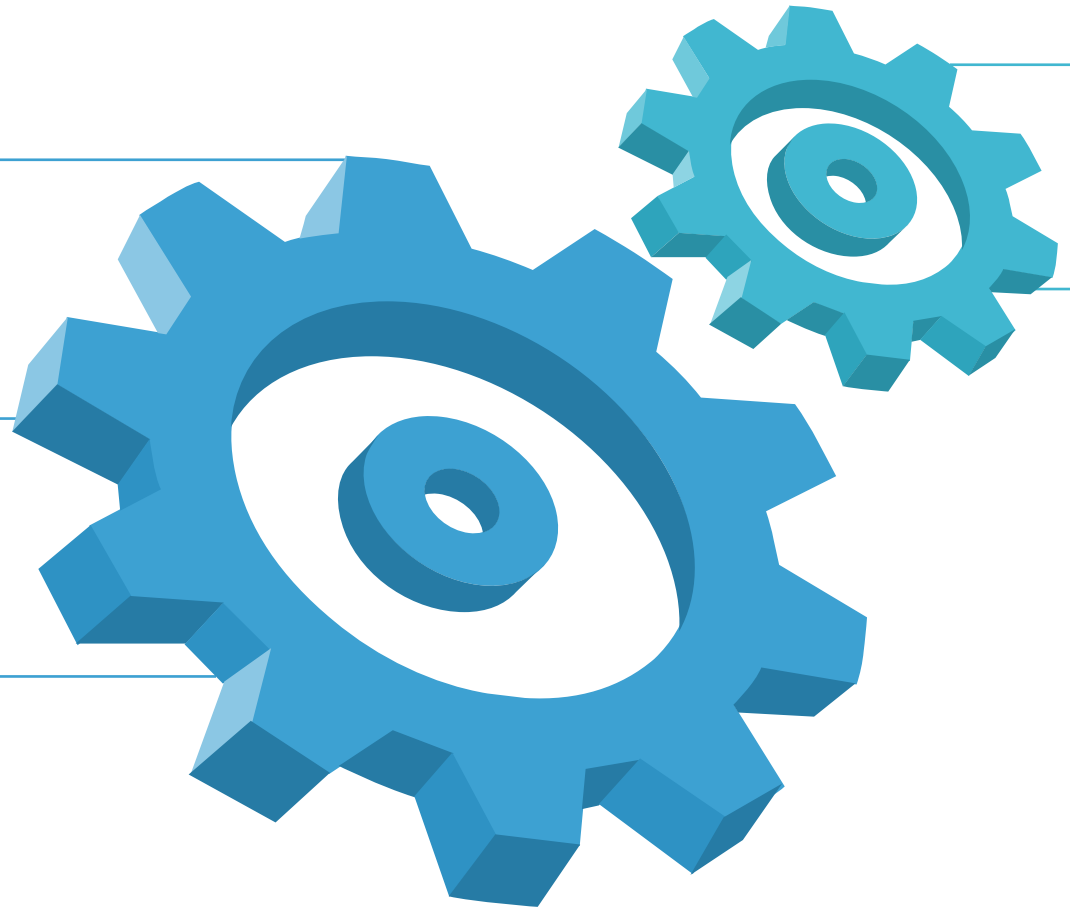
Evaluate and adjust

BENEFITS OF PROMPT ENGINEERING EXPERIMENTATION AND DISCOVERY

Improved response quality

Enhanced flexibility

Increased efficiency



Better understanding of AI behavior

Tailored customization and personalization

BENEFITS OF USING PROMPT ENGINEERING GUARDRAILS

- Guardrails help mitigate risks such as prompt injection attacks, data leaks, and other security vulnerabilities
- Implementing robust security measures ensures that the AI model operates within safe boundaries
- Guardrails also help ensure that the AI model adheres to relevant regulations and industry standards

BENEFITS OF USING PROMPT ENGINEERING GUARDRAILS

- Guardrails can guide the AI model to produce more accurate and reliable outputs by preventing it from generating harmful or biased content
- This helps maintain the quality and trustworthiness of the AI-generated responses
- By using guardrails, organizations can promote responsible AI practices

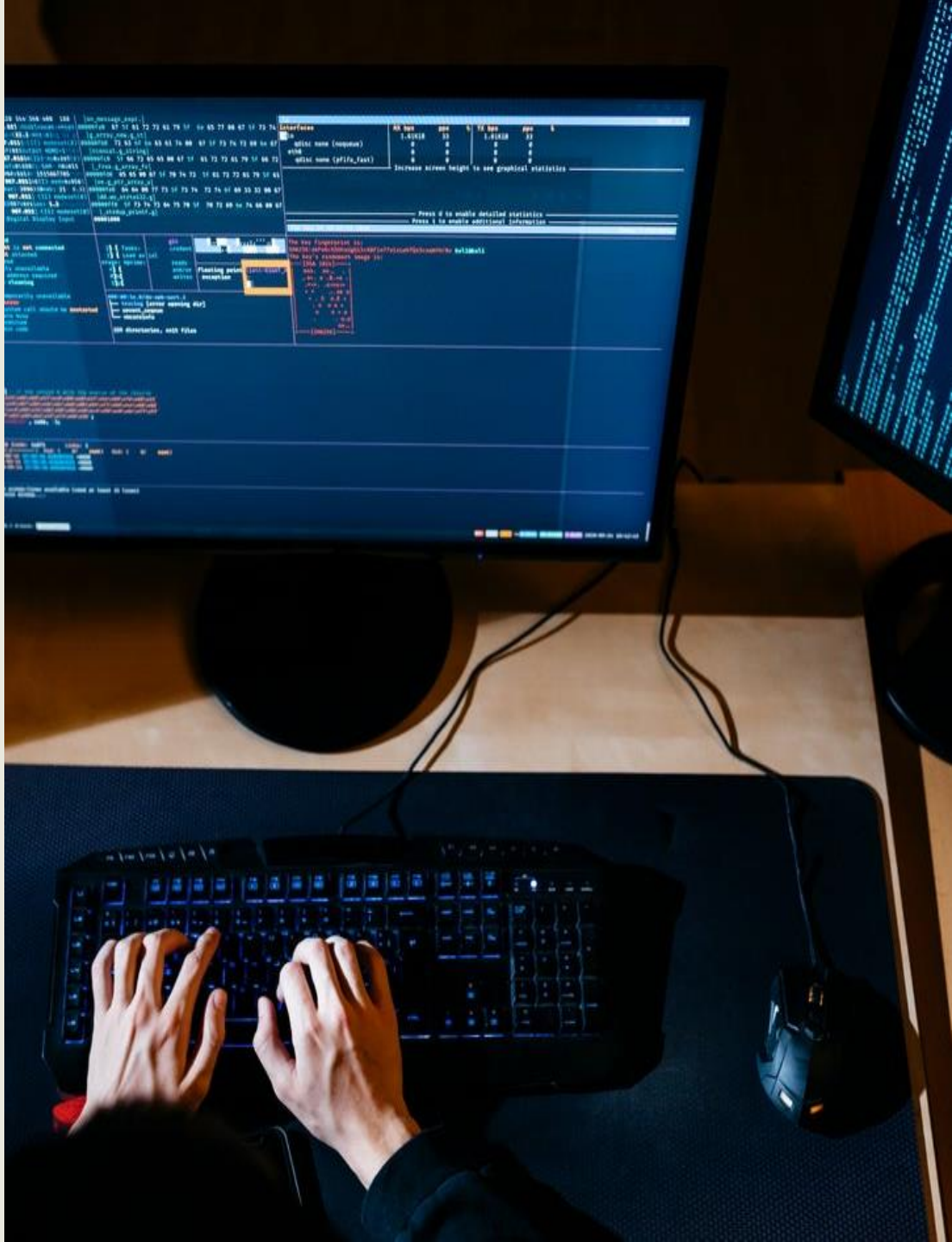
PROMPT ENGINEERING GUARDRAILS

Here is a simple example of a prompt engineering guardrail:

- **Task**: Generate a story about a day at the beach.
- **Prompt:** "Write a story about a day at the beach. Do not include any references to drugs, alcohol, or violence."

PROMPT ENGINEERING SPECIFICITY AND CONCISION

- When it comes to prompt engineering, "specificity and concision" are crucial elements
- **Specificity** involves providing detailed and precise instructions to the AI model
  - This helps in reducing ambiguity and ensures that the AI generates the desired output
- **Concision**, on the other hand, means being clear and to the point
  - It involves using the fewest words necessary to convey the instructions effectively



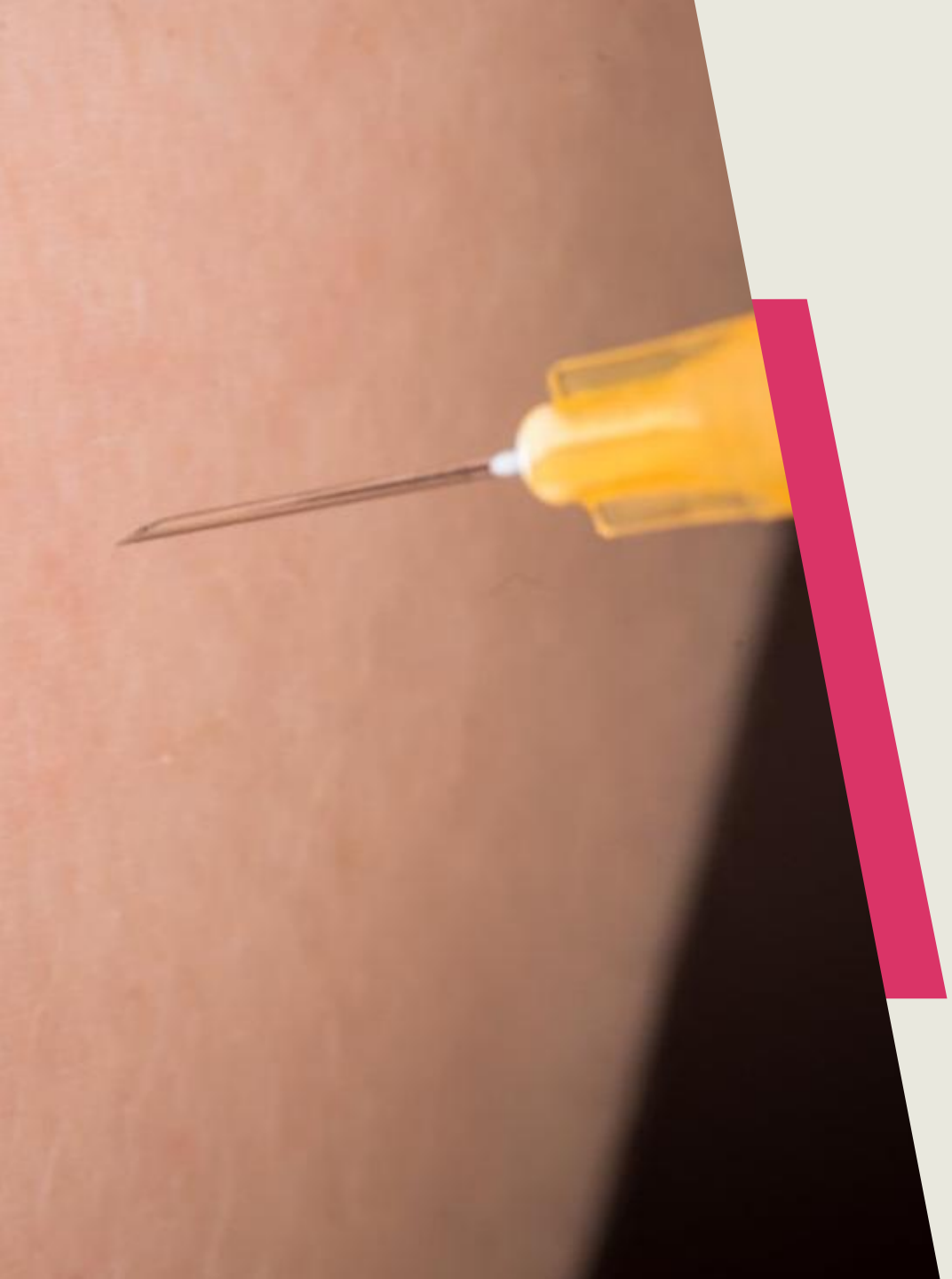
BENEFITS OF USING MULTIPLE COMMENTS WITH PROMPT ENGINEERING

Clarity and precision

Contextual guidance

Error reduction

Flexibility



RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- According to AWS, the "**exposure**" risk in prompt engineering primarily refers to the potential for prompt injection attacks
- These attacks involve manipulating prompts to influence the outputs of large language models (LLMs) in a harmful or biased manner

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- This can lead to the exposure of sensitive information, biased outputs, and other security vulnerabilities
- To mitigate these risks, AWS recommends implementing robust security measures, such as using guardrails and specific tagging structures, to ensure the integrity and security of AI-generated outputs

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- The "**poisoning**" risk in prompt engineering refers to the deliberate manipulation of training data or prompts to introduce harmful or biased outputs
- This can occur when malicious actors inject misleading or harmful data into the training set or prompt templates, causing the AI model to generate incorrect or harmful responses

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- This risk can compromise the integrity and reliability of AI systems, leading to potential security vulnerabilities and ethical concerns
- To address this risk, AWS advises adopting strong security protocols, including the use of protective barriers and precise tagging systems, to maintain the integrity and safety of AI-generated results

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- The "**hijacking**" risk in prompt engineering refers to the potential for malicious actors to manipulate prompts in a way that causes the AI model to adopt a new, unintended persona or behavior
- This can lead to the AI generating harmful or biased outputs, compromising the integrity and reliability of the system

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- The "j**ailbreaking**" risk in prompt engineering refers to the potential for users to manipulate prompts in a way that bypasses the intended restrictions and controls of the AI model
- This can lead to the AI generating outputs that are harmful, inappropriate, or otherwise unintended

RISKS AND LIMITATIONS OF PROMPT ENGINEERING

- Jailbreaking can compromise the integrity and security of AI systems, making them vulnerable to misuse
- To mitigate **hijacking and jailbreaking** risk, AWS recommends implementing robust security measures, such as using guardrails and detailed tagging structures, to ensure the integrity and security of AI-generated outputs