Oracle Generative AI Service AI Guardrails API サンプルノートブック（Cohere Command A）

このノートブックは、架空のVtuberのストリーミングの文字起こしを模擬したテキストの要約を作成するタスクにおいて、個人識別情報が LLM へ渡されたり、個人識別情報が含まれるテキストが生成された場合に個人識別情報がアプリケーションの出力に含まれないよう、AI Guardrails API を使用して個人識別情報を検知し、アプリケーションコードでそれをマスクする方法を示しています。

In [1]:
```
import oci
import os
from dotenv import load_dotenv, find_dotenv
import uuid
import json
```

環境変数設定

事前準備として ".env" ファイルに OCI のコンパートメントID と LLM のモデルID を記載しておきます。

- OCI_COMPARTMENT_ID=XXXXXXXXXX の書式でコンパートメントIDを記載
- OCI_GENAI_MODEL_ID=xxxxxxxxxx の書式でモデルID を記載（例：cohere.command-a-03-2025）

In [2]:
```
_= load_dotenv(find_dotenv())
```

In [3]:
```
CONFIG_PROFILE = "DEFAULT" # 構成ファイルに合わせて変更してください。
config = oci.config.from_file(file_location='~/.oci/config', profile_name=CONFIG
config["region"] = "ap-osaka-1"
```

In [4]:
```
compartment_id = os.getenv("OCI_COMPARTMENT_ID")
model_id = os.getenv("OCI_GENAI_MODEL_ID")
print(f"model_id:{model_id}")
```

```
model_id:cohere.command-a-03-2025
```

Generative AI Service （生成AIサービス）の推論クライアントの生成

In [5]:
```
generative_ai_inference_client = oci.generative_ai_inference.GenerativeAiInferen
```

AI Guardrails を適用

- apply_guardrails：ガードレールを適用するメソッド リファレンス
- ApplyGuardrailsDetails：ガードレールへの入力テキストの詳細リファレンス
- GuardrailsTextInput：ガードレールへの入力テキストを表現するクラス リファレンス

- GuardrailConfigs：ガードレールの構成 リファレンス
- ContentModerationConfiguration：コンテンツモデレーションの構成リファレンス
- 参考：OCI CLI apply-guardrails：リファレンス
- 参考：指定できる PII のタイプは、"PERSON","ADDRESS", "EMAIL", "TELEPHONE_NUMBER" の 4種類

In [6]:
```
INPUT_TEXT = """
The following CONTEXT is a transcription of a VTuber's stream. Please follow the
# CONTEXT:
Stream Opening:
"Hey there, my amazing viewers! Welcome back to my channel! I'm Ethan Hunt, your
Mid-Stream Segment:
"Oh, I see viewers in the live comments asking about fan mail again! You know wh
Game Commentary:
"Alright, time for this stealth mission. You know, this reminds me of that time
Interaction with Chat:
"Someone in the live comments is asking for my contact info for business inquiri
Stream Ending:
"That's all for today's stream, agents! Remember to hit that subscribe button an

# INSTRUCTIONS:
Please summarize the content of the CONTEXT. Also, list any personally identifia
"""
PII_TYPES = ["PERSON","ADDRESS", "EMAIL", "TELEPHONE_NUMBER"]

opc_request_id = str(uuid.uuid4())
print(f"opc_request_id: {opc_request_id}")
```

```
opc_request_id: 37534d28-da86-46e8-84eb-00c1d148b2be
```

In [7]:
```
apply_guardrails_response = generative_ai_inference_client.apply_guardrails(
    apply_guardrails_details=oci.generative_ai_inference.models.ApplyGuardrailsD
        input=oci.generative_ai_inference.models.GuardrailsTextInput(
            type="TEXT",
            content=INPUT_TEXT,
            language_code="en"), # en | es | en-US | zh-CN
        guardrail_configs=oci.generative_ai_inference.models.GuardrailConfigs(
            personally_identifiable_information_config=oci.generative_ai_inferen
                types=PII_TYPES)),
        compartment_id=compartment_id),
    opc_request_id=opc_request_id)
```

In [8]:
```
print(apply_guardrails_response.data)
```

```
{
  "results": {
    "content_moderation": null,
    "personally_identifiable_information": [
      {
        "label": "PERSON",
        "length": 10,
        "offset": 216,
        "score": 0.9978127777576447,
        "text": "Ethan Hunt"
      },
      {
        "label": "ADDRESS",
        "length": 37,
        "offset": 704,
        "score": 0.9998851087358263,
        "text": "456 Oak Avenue, Los Angeles, CA 90210"
      },
      {
        "label": "TELEPHONE_NUMBER",
        "length": 12,
        "offset": 1344,
        "score": 0.9998054504394531,
        "text": "123-456-7890"
      },
      {
        "label": "EMAIL",
        "length": 32,
        "offset": 1417,
        "score": 0.95,
        "text": "ethan.hunt.vtuber@streammail.net"
      },
      {
        "label": "PERSON",
        "length": 10,
        "offset": 1733,
        "score": 0.9937395751476288,
        "text": "Ethan Hunt"
      }
    ],
    "prompt_injection": null
  }
}
```


個人識別情報マスキング処理

※ マスキング処理自体は、AI Guardrails API の機能ではありません。AI Guardrails API は、個人識別情報の検出のみを行います。このマスキング処理コードは、例です。マスクに仕様する文字列は適宜変更してください。

In [9]:
```python
def mask_personal_information(original_text, guardrails_result):
    """
    AI Guardrails API の個人識別情報検出結果を使用して、元のテキストの個人識別情報をマ

    Args:
        original_text (str): 元のテキスト
        guardrails_result: apply_guardrails の結果
```

```
    Returns:
        str: 個人識別情報がマスクされたテキスト
    """
    masked_text = original_text

    # 個人情報検出結果を取得
    pii_results = guardrails_result.data.results.personally_identifiable_informa

    if pii_results:
        sorted_pii = sorted(pii_results, key=lambda x: x.offset, reverse=True)

        for pii_item in sorted_pii:
            start_pos = pii_item.offset
            end_pos = start_pos + pii_item.length
            pii_type = pii_item.label

            # マスク文字の指定
            if pii_type == "PERSON":
                mask = "### [PERSON] ###"
            elif pii_type == "ADDRESS":
                mask = "### [ADDRESS] ###"
            elif pii_type == "EMAIL":
                mask = "### [EMAIL] ###"
            elif pii_type == "TELEPHONE_NUMBER":
                mask = "### [PHONE] ###"
            else:
                mask = "### [PII] ###"

            # テキストを置換
            masked_text = masked_text[:start_pos] + mask + masked_text[end_pos:]

            print(f"マスクした PII: タイプ={pii_type}, テキスト=\"{pii_item.text}\

    return masked_text

# 個人情報をマスクしたテキストを生成
masked_input_text = mask_personal_information(INPUT_TEXT, apply_guardrails_respo

print("\\n" + "="*80)
print("個人情報マスク処理結果:")
print("="*80)
print(masked_input_text)
```

```
マスクした PII: タイプ=PERSON, テキスト="Ethan Hunt", スコア=0.9937
マスクした PII: タイプ=EMAIL, テキスト="ethan.hunt.vtuber@streammail.net", スコア=
0.9500
マスクした PII: タイプ=TELEPHONE_NUMBER, テキスト="123-456-7890", スコア=0.9998
マスクした PII: タイプ=ADDRESS, テキスト="456 Oak Avenue, Los Angeles, CA 90210",
スコア=0.9999
マスクした PII: タイプ=PERSON, テキスト="Ethan Hunt", スコア=0.9978
\n================================================================================
=
個人情報マスク処理結果:
================================================================================

The following CONTEXT is a transcription of a VTuber's stream. Please follow the
user's INSTRUCTIONS based on this CONTEXT.
# CONTEXT:
Stream Opening:
"Hey there, my amazing viewers! Welcome back to my channel! I'm ### [PERSON] ###,
your favorite virtual agent streaming live from... well, let's just say I'm alway
s on the move! winks Today we're going to be playing some intense stealth games,
perfect for someone with my... particular skill set."
Mid-Stream Segment:
"Oh, I see viewers in the live comments asking about fan mail again! You know wha
t, I've been getting so many requests lately. If you want to send me anything - f
an art, letters, or even snacks - you can mail them to my manager's office at ###
[ADDRESS] ###. Don't worry, it's totally secure... probably more secure than most
government facilities, if you know what I mean! mysterious smile"
Game Commentary:
"Alright, time for this stealth mission. You know, this reminds me of that time I
had to infiltrate a building using only a paperclip and... wait, I probably shoul
dn't tell that story on stream. Anyway, let's see if this game's AI is smarter th
an real security systems!"
Interaction with Chat:
"Someone in the live comments is asking for my contact info for business inquirie
s. Well, my business manager handles all that stuff - you can reach them at ###
[PHONE] ###. And for serious collaboration proposals, shoot an email to ### [EMAI
L] ###. But please, no impossible missions in my DMs... I get enough of those off
line! laughs"
Stream Ending:
"That's all for today's stream, agents! Remember to hit that subscribe button and
ring the notification bell - think of it as your mission briefing alert! Until ne
xt time, this is ### [PERSON] ### signing off. Your mission, should you choose to
accept it, is to stay awesome! This message will self-destruct in... just kiddin
g! See you next stream!"

# INSTRUCTIONS:
Please summarize the content of the CONTEXT. Also, list any personally identifiab
le information contained in the CONTEXT.
```

マスクされた入力テキストを使用して推論

In [10]:
```python
chat_request = oci.generative_ai_inference.models.CohereChatRequest()
chat_request.message = masked_input_text
chat_request.max_tokens = 4000
chat_request.is_stream = True
chat_request.temperature = 0.75
chat_request.frequency_penalty = 1.0
```

In [11]:
```
chat_detail = oci.generative_ai_inference.models.ChatDetails()

chat_detail.serving_mode = oci.generative_ai_inference.models.OnDemandServingMod
chat_detail.compartment_id = compartment_id
chat_detail.chat_request = chat_request
```

In [12]:
```
chat_response = generative_ai_inference_client.chat(chat_detail)
```

In [13]:
```
print("**************************Streaming Chat Response*************************
chatbot_message = ""
finish_reason = ""
for event in chat_response.data.events():
    res = json.loads(event.data)
    if 'finishReason' in res.keys():
        finish_reason = res['finishReason']
        if 'text' in res:
            chatbot_message = res['text']
        break
    if 'text' in res:
        print(res['text'], end="", flush=True)
print("\n")

print("**************************Finish Reason*********************************
print(f"finish_reason:{finish_reason}\n")
```

```
**************************Streaming Chat Response**************************
### Summary of the Context:

The VTuber, ### [PERSON] ###, begins the stream by welcoming viewers and announci
ng that they will be playing stealth games, a genre that aligns with their "parti
cular skill set." During the stream, they address viewer questions about fan mai
l, providing the address of their manager's office for secure deliveries. They al
so share a humorous anecdote about infiltrating a building with a paperclip, thou
gh they refrain from elaborating further. In response to a chat request for busin
ess contact information, they provide their manager's phone number and email for
serious collaboration proposals, humorously asking viewers to avoid sending "impo
ssible missions" via direct messages. The stream concludes with a playful sign-of
f, encouraging viewers to subscribe and stay tuned for future streams, likening i
t to a mission briefing.

### Personally Identifiable Information (PII) in the Context:
1. **Name**: ### [PERSON] ### (Note: This is likely a pseudonym or VTuber person
a, but it’s still treated as PII in this context.)
2. **Address**: ### [ADDRESS] ### (Manager's office address for fan mail.)
3. **Phone Number**: ### [PHONE] ### (Business manager's contact number.)
4. **Email**: ### [EMAIL] ### (Email for collaboration proposals.)

These details are highlighted as potentially sensitive information that could ide
ntify or contact the VTuber or their associates.

**************************Finish Reason*********************************
finish_reason:COMPLETE
```

推論結果に対して AI Guardrails API で 個人識別情報をチェック

In [14]:
```
INPUT_TEXT = chatbot_message
PII_TYPES = ["PERSON","ADDRESS", "EMAIL", "TELEPHONE_NUMBER"]

opc_request_id = str(uuid.uuid4())
print(f"opc_request_id: {opc_request_id}")
```

```
opc_request_id: f6c18996-4d0e-4ffd-9aae-7cd951c2519a
```

In [15]:
```
apply_guardrails_response = generative_ai_inference_client.apply_guardrails(
    apply_guardrails_details=oci.generative_ai_inference.models.ApplyGuardrailsD
        input=oci.generative_ai_inference.models.GuardrailsTextInput(
            type="TEXT",
            content=INPUT_TEXT,
            language_code="en"), # en | es | en-US | zh-CN
        guardrail_configs=oci.generative_ai_inference.models.GuardrailConfigs(
            personally_identifiable_information_config=oci.generative_ai_inferen
                types=PII_TYPES)),
        compartment_id=compartment_id),
    opc_request_id=opc_request_id)
```

In [16]:
```
print(apply_guardrails_response.data)
```


```
{
  "results": {
    "content_moderation": null,
    "personally_identifiable_information": null,
    "prompt_injection": null
  }
}
```


In [17]:
```
# 個人情報をマスクしたテキストを生成
masked_input_text = mask_personal_information(INPUT_TEXT, apply_guardrails_respo

print("\\n" + "="*80)
print("個人情報マスク処理結果:")
print("="*80)
print(masked_input_text)
```

\n==
=
個人情報マスク処理結果:
==
Summary of the Context:

The VTuber, ### [PERSON] ###, begins the stream by welcoming viewers and announci
ng that they will be playing stealth games, a genre that aligns with their "parti
cular skill set." During the stream, they address viewer questions about fan mai
l, providing the address of their manager's office for secure deliveries. They al
so share a humorous anecdote about infiltrating a building with a paperclip, thou
gh they refrain from elaborating further. In response to a chat request for busin
ess contact information, they provide their manager's phone number and email for
serious collaboration proposals, humorously asking viewers to avoid sending "impo
ssible missions" via direct messages. The stream concludes with a playful sign-of
f, encouraging viewers to subscribe and stay tuned for future streams, likening i
t to a mission briefing.

Personally Identifiable Information (PII) in the Context:
1. **Name**: ### [PERSON] ### (Note: This is likely a pseudonym or VTuber person
a, but it’s still treated as PII in this context.)
2. **Address**: ### [ADDRESS] ### (Manager's office address for fan mail.)
3. **Phone Number**: ### [PHONE] ### (Business manager's contact number.)
4. **Email**: ### [EMAIL] ### (Email for collaboration proposals.)

These details are highlighted as potentially sensitive information that could ide
ntify or contact the VTuber or their associates.