# Leverage deployment guardrails to update a SageMaker Inference endpoint using canary traffic shifting

# Contents



# Introduction

Deployment guardrails are a set of model deployment options in Amazon SageMaker Inference to update your machine learning models in production. Using the fully managed deployment guardrails, you can control the switch from the current model in production to a new one. Traffic shifting modes, such as canary and linear, give you granular control over the traffic shifting process from your current model to the new one during the course of the update. There are also built-in safeguards such as auto-rollbacks that help you catch issues early and take corrective action before they impact production.

We support blue-green deployment with multiple traffic shifting modes. A traffic shifting mode is a configuration that specifies how endpoint traffic is routed to a new fleet containing your updates. The following traffic shifting modes provide you with different levels of control over the endpoint update process:

- **All-At-Once Traffic Shifting** : shifts all of your endpoint traffic from the blue fleet to the green fleet. Once the traffic has shifted to the green fleet, your pre-specified Amazon CloudWatch alarms begin monitoring the green fleet for a set amount of time (the "baking period"). If no alarms are triggered during the baking period, then the blue fleet is terminated.
- **Canary Traffic Shifting** : lets you shift one small portion of your traffic (a "canary") to the green fleet and monitor it for a baking period. If the canary succeeds on the green fleet, then the rest of the traffic is shifted from the blue fleet to the green fleet before terminating the blue fleet.
- **Linear Traffic Shifting** : provides even more customization over how many traffic-shifting steps to make and what percentage of traffic to shift for each step. While canary

shifting lets you shift traffic in two steps, linear shifting extends this to n number of linearly spaced steps.

The Deployment guardrails for Amazon SageMaker Inference endpoints feature also allows customers to specify conditions/alarms based on Endpoint invocation metrics from CloudWatch to detect model performance regressions and trigger automatic rollback.

In this notebook we'll update endpoint with following deployment configurations:

- Blue/Green update policy with **Canary traffic shifting option**
- Configure CloudWatch alarms to monitor model performance and trigger auto-rollback action.

To demonstrate Canary deployments and the auto-rollback feature, we will update an Endpoint with an incompatible model version and deploy it as a Canary fleet, taking a small percentage of the traffic. Requests sent to this Canary fleet will result in errors, which will be used to trigger a rollback using pre-specified CloudWatch alarms. Finally, we will also demonstrate a success scenario where no alarms are tripped and the update succeeds.

This notebook is organized in 4 steps -

- Step 1 creates the models and Endpoint Configurations required for the 3 scenarios - the baseline, the update containing the incompatible model version and the update containing the correct model version.
- Step 2 invokes the baseline Endpoint prior to the update.
- Step 3 specifies the CloudWatch alarms used to trigger the rollbacks.
- Finally in step 4, we update the endpoint to trigger a rollback and demonstrate a successful update.

# Setup

Ensure that you have an updated version of boto3, which includes the latest SageMaker features:

```
!pip install -U awscli
!pip install sagemaker
```

```
Requirement already satisfied: awscli in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(1.31.4)
Collecting awscli
  Downloading awscli-1.31.13-py3-none-any.whl.metadata (11 kB)
Collecting botocore==1.33.13 (from awscli)
  Downloading botocore-1.33.13-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: docutils<0.17,>=0.10 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from awscli)
(0.16)
Requirement already satisfied: s3transfer<0.9.0,>=0.8.0 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from awscli)
(0.8.2)
```

```
Requirement already satisfied: PyYAML<6.1,>=3.10 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from awscli) (6.0.1)
Requirement already satisfied: colorama<0.4.5,>=0.2.5 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from awscli)
(0.4.4)
Requirement already satisfied: rsa<4.8,>=3.1.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from awscli) (4.7.2)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
botocore==1.33.13->awscli) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from botocore==1.33.13->awscli) (2.8.2)
Requirement already satisfied: urllib3<2.1,>=1.25.4 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
botocore==1.33.13->awscli) (1.26.18)
Requirement already satisfied: pyasn1>=0.1.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from rsa<4.8,>=3.1.2->awscli) (0.5.1)
Requirement already satisfied: six>=1.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from python-dateutil<3.0.0,>=2.1->botocore==1.33.13->awscli) (1.16.0)
Downloading awscli-1.31.13-py3-none-any.whl (4.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 23.2 MB/s eta
0:00:00:00:010:01
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.8/11.8 MB 41.7 MB/s eta
0:00:00:00:010:01
pting uninstall: botocore
    Found existing installation: botocore 1.33.4
    Uninstalling botocore-1.33.4:
      Successfully uninstalled botocore-1.33.4
  Attempting uninstall: awscli
    Found existing installation: awscli 1.31.4
    Uninstalling awscli-1.31.4:
      Successfully uninstalled awscli-1.31.4
Successfully installed awscli-1.31.13 botocore-1.33.13
Requirement already satisfied: sagemaker in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(2.199.0)
Requirement already satisfied: attrs<24,>=23.1.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (23.1.0)
Requirement already satisfied: boto3<2.0,>=1.33.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (1.33.4)
Requirement already satisfied: cloudpickle==2.2.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
```

```
(from sagemaker) (2.2.1)
Requirement already satisfied: google-pasta in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (0.2.0)
Requirement already satisfied: numpy<2.0,>=1.9.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (1.22.4)
Requirement already satisfied: protobuf<5.0,>=3.12 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (4.25.1)
Requirement already satisfied: smdebug-rulesconfig==1.0.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (1.0.1)
Requirement already satisfied: importlib-metadata<7.0,>=1.4.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (6.8.0)
Requirement already satisfied: packaging>=20.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (21.3)
Requirement already satisfied: pandas in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (2.1.1)
Requirement already satisfied: pathos in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (0.3.1)
Requirement already satisfied: schema in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (0.7.5)
Requirement already satisfied: PyYAML~=6.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (6.0.1)
Requirement already satisfied: jsonschema in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (4.19.1)
Requirement already satisfied: platformdirs in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (3.11.0)
Requirement already satisfied: tblib==1.7.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (1.7.0)
Requirement already satisfied: urllib3<1.27 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (1.26.18)
Requirement already satisfied: uvicorn==0.22.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (0.22.0)
Requirement already satisfied: fastapi==0.95.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (0.95.2)
```

```
Requirement already satisfied: requests in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (2.31.0)
Requirement already satisfied: docker in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (6.1.3)
Requirement already satisfied: tqdm in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (4.66.1)
Requirement already satisfied: psutil in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker) (5.9.5)
Requirement already satisfied: pydantic!=1.7,!=1.7.1,!=1.7.2,!=1.7.3,!
=1.8,!=1.8.1,<2.0.0,>=1.6.2 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from fastapi==0.95.2->sagemaker) (1.10.13)
Requirement already satisfied: starlette<0.28.0,>=0.27.0 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
fastapi==0.95.2->sagemaker) (0.27.0)
Requirement already satisfied: click>=7.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from uvicorn==0.22.0->sagemaker) (8.1.7)
Requirement already satisfied: h11>=0.8 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from uvicorn==0.22.0->sagemaker) (0.14.0)
Requirement already satisfied: botocore<1.34.0,>=1.33.4 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
boto3<2.0,>=1.33.3->sagemaker) (1.33.13)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
boto3<2.0,>=1.33.3->sagemaker) (1.0.1)
Requirement already satisfied: s3transfer<0.9.0,>=0.8.2 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
boto3<2.0,>=1.33.3->sagemaker) (0.8.2)
Requirement already satisfied: zipp>=0.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from importlib-metadata<7.0,>=1.4.0->sagemaker) (3.17.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
packaging>=20.0->sagemaker) (3.1.1)
Requirement already satisfied: websocket-client>=0.32.0 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from docker-
>sagemaker) (1.6.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests->sagemaker) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from requests->sagemaker) (3.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from requests->sagemaker) (2023.7.22)
Requirement already satisfied: six in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from google-pasta->sagemaker) (1.16.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from jsonschema->sagemaker) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from jsonschema->sagemaker) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from jsonschema->sagemaker) (0.10.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from pandas-
>sagemaker) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from pandas->sagemaker) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from pandas->sagemaker) (2023.3)
Requirement already satisfied: ppft>=1.7.6.7 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from pathos->sagemaker) (1.7.6.7)
Requirement already satisfied: dill>=0.3.7 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from pathos->sagemaker) (0.3.7)
Requirement already satisfied: pox>=0.3.3 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from pathos->sagemaker) (0.3.3)
Requirement already satisfied: multiprocess>=0.70.15 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from pathos-
>sagemaker) (0.70.15)
Requirement already satisfied: contextlib2>=0.5.5 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from schema->sagemaker) (21.6.0)
Requirement already satisfied: typing-extensions>=4.2.0 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
pydantic!=1.7,!=1.7.1,!=1.7.2,!=1.7.3,!=1.8,!=1.8.1,<2.0.0,>=1.6.2-
>fastapi==0.95.2->sagemaker) (4.8.0)
Requirement already satisfied: anyio<5,>=3.4.0 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from starlette<0.28.0,>=0.27.0->fastapi==0.95.2->sagemaker) (4.0.0)
Requirement already satisfied: sniffio>=1.1 in
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from anyio<5,>=3.4.0->starlette<0.28.0,>=0.27.0->fastapi==0.95.2-
```

```
>sagemaker) (1.3.0)
Requirement already satisfied: exceptiongroup>=1.0.2 in /home/ec2-
user/anaconda3/envs/python3/lib/python3.10/site-packages (from
anyio<5,>=3.4.0->starlette<0.28.0,>=0.27.0->fastapi==0.95.2-
>sagemaker) (1.1.3)
```

Setup some required imports and basic initial variables:

```
%matplotlib inline

import time
import os
import boto3
import botocore
import re
import json
from datetime import datetime, timedelta, timezone
from sagemaker import get_execution_role, session
from sagemaker.s3 import S3Downloader, S3Uploader

region = boto3.Session().region_name

# You can use a different IAM role with "SageMakerFullAccess" policy
for this notebook
role = get_execution_role()
print(f"Execution role: {role}")

sm_session = session.Session(boto3.Session())
sm = boto3.Session().client("sagemaker")
sm_runtime = boto3.Session().client("sagemaker-runtime")

# You can use a different bucket, but make sure the role you chose for
this notebook
# has the s3:PutObject permissions. This is the bucket into which the
model artifacts will be uploaded
bucket = "final-10lab"
prefix = "sagemaker/DEMO-Deployment-Guardrails-Canary"

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/
pandas/core/computation/expressions.py:21: UserWarning: Pandas
requires version '2.8.0' or newer of 'numexpr' (version '2.7.3'
currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED

sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/ec2-user/.config/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/home/ec2-user/.config/sagemaker/config.yaml
Execution role: arn:aws:iam::040700907151:role/LabRole
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/ec2-user/.config/sagemaker/config.yaml
```

Download the Input files and pre-trained model from S3 bucket

```
!aws s3 cp s3://final-10lab/model.tar.gz model/
!aws s3 cp s3://final-10lab/model2.tar.gz model/

!aws s3 cp s3://final-10lab/test1.csv test_data/
!aws s3 cp s3://final-10lab/test2.csv test_data/

download: s3://final-10lab/model.tar.gz to model/model.tar.gz
download: s3://final-10lab/model2.tar.gz to model/model2.tar.gz
download: s3://final-10lab/test1.csv to test_data/test1.csv
download: s3://final-10lab/test2.csv to test_data/test2.csv
```

# Step 1: Create and deploy the models

## First, we upload our pre-trained models to Amazon S3

This code uploads two pre-trained XGBoost models that are ready for you to deploy. These models were trained using the XGB Churn Prediction Notebook in SageMaker. You can also use your own pre-trained models in this step. If you already have a pretrained model in Amazon S3, you can add it by specifying the s3_key.

The models in this example are used to predict the probability of a mobile customer leaving their current mobile operator. The dataset we use is publicly available and was mentioned in the book Discovering Knowledge in Data by Daniel T. Larose. It is attributed by the author to the University of California Irvine Repository of Machine Learning Datasets.

```
model_url = S3Uploader.upload(
    local_path="model/model.tar.gz",
    desired_s3_uri=f"s3://{bucket}/{prefix}",
)
model_url2 = S3Uploader.upload(
    local_path="model/model2.tar.gz",
    desired_s3_uri=f"s3://{bucket}/{prefix}",
)

print(f"Model URI 1: {model_url}")
print(f"Model URI 2: {model_url2}")
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/ec2-user/.config/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/ec2-user/.config/sagemaker/config.yaml
Model URI 1: s3://final-10lab/sagemaker/DEMO-Deployment-Guardrails-
Canary/model.tar.gz
Model URI 2: s3://final-10lab/sagemaker/DEMO-Deployment-Guardrails-
Canary/model2.tar.gz
```

## Next, we create our model definitions

Start with deploying the pre-trained churn prediction models. Here, you create the model objects with the image and model data. The three URIs correspond to the baseline version, the update containing the incompatible version, and the update containing the correct model version.

```
from sagemaker import image_uris

image_uri = image_uris.retrieve("xgboost",
boto3.Session().region_name, "0.90-1")

# using newer version of XGBoost which is incompatible, in order to
simulate model faults
image_uri2 = image_uris.retrieve("xgboost",
boto3.Session().region_name, "1.2-1")
image_uri3 = image_uris.retrieve("xgboost",
boto3.Session().region_name, "0.90-2")

print(f"Model Image 1: {image_uri}")
print(f"Model Image 2: {image_uri2}")
print(f"Model Image 3: {image_uri3}")
```

```
Model Image 1: 683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost:0.90-1-cpu-py3
Model Image 2: 683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost:1.2-1
Model Image 3: 683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost:0.90-2-cpu-py3
```

```
model_name = f"DEMO-xgb-stroke-pred-{datetime.now():%Y-%m-%d-%H-%M-
%S}"
model_name2 = f"DEMO-xgb-stroke-pred2-{datetime.now():%Y-%m-%d-%H-%M-
%S}"
model_name3 = f"DEMO-xgb-stroke-pred3-{datetime.now():%Y-%m-%d-%H-%M-
%S}"
```

```
print(f"Model Name 1: {model_name}")
print(f"Model Name 2: {model_name2}")
print(f"Model Name 3: {model_name3}")

resp = sm.create_model(
    ModelName=model_name,
    ExecutionRoleArn=role,
    Containers=[{"Image": image_uri, "ModelDataUrl": model_url}],
)
print(f"Created Model: {resp}")

resp = sm.create_model(
    ModelName=model_name2,
    ExecutionRoleArn=role,
    Containers=[{"Image": image_uri2, "ModelDataUrl": model_url2}],
)
print(f"Created Model: {resp}")

resp = sm.create_model(
    ModelName=model_name3,
    ExecutionRoleArn=role,
    Containers=[{"Image": image_uri3, "ModelDataUrl": model_url2}],
)
print(f"Created Model: {resp}")
```

```
Model Name 1: DEMO-xgb-stroke-pred-2023-12-12-23-44-21
Model Name 2: DEMO-xgb-stroke-pred2-2023-12-12-23-44-21
Model Name 3: DEMO-xgb-stroke-pred3-2023-12-12-23-44-21
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-
1:040700907151:model/demo-xgb-stroke-pred-2023-12-12-23-44-21',
'ResponseMetadata': {'RequestId': 'c31cd7a0-22a4-4ab1-8d35-
916dd11262ed', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': 'c31cd7a0-22a4-4ab1-8d35-916dd11262ed', 'content-type':
'application/x-amz-json-1.1', 'content-length': '102', 'date': 'Tue,
12 Dec 2023 23:44:20 GMT'}, 'RetryAttempts': 0}}
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-
1:040700907151:model/demo-xgb-stroke-pred2-2023-12-12-23-44-21',
'ResponseMetadata': {'RequestId': 'a8ce6c1c-fbac-45ba-aaf0-
f55220da8d10', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': 'a8ce6c1c-fbac-45ba-aaf0-f55220da8d10', 'content-type':
'application/x-amz-json-1.1', 'content-length': '103', 'date': 'Tue,
12 Dec 2023 23:44:23 GMT'}, 'RetryAttempts': 2}}
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-
1:040700907151:model/demo-xgb-stroke-pred3-2023-12-12-23-44-21',
'ResponseMetadata': {'RequestId': '492ee667-cc80-47b0-b0fe-
a620cb463096', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': '492ee667-cc80-47b0-b0fe-a620cb463096', 'content-type':
'application/x-amz-json-1.1', 'content-length': '103', 'date': 'Tue,
12 Dec 2023 23:44:24 GMT'}, 'RetryAttempts': 1}}
```

## Create Endpoint Configs

We now create three EndpointConfigs, corresponding to the three Models we created in the previous step.

```
ep_config_name = f"DEMO-EpConfig-1-{datetime.now():%Y-%m-%d-%H-%M-%S}"
ep_config_name2 = f"DEMO-EpConfig-2-{datetime.now():%Y-%m-%d-%H-%M-
%S}"
ep_config_name3 = f"DEMO-EpConfig-3-{datetime.now():%Y-%m-%d-%H-%M-
%S}"

print(f"Endpoint Config 1: {ep_config_name}")
print(f"Endpoint Config 2: {ep_config_name2}")
print(f"Endpoint Config 3: {ep_config_name3}")

resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name,
            "InstanceType": "ml.m5.xlarge",
            "InitialInstanceCount": 3,
        }
    ],
)
print(f"Created Endpoint Config: {resp}")
time.sleep(5)

resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name2,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name2,
            "InstanceType": "ml.m5.xlarge",
            "InitialInstanceCount": 3,
        }
    ],
)
print(f"Created Endpoint Config: {resp}")
time.sleep(5)

resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name3,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name3,
            "InstanceType": "ml.m5.xlarge",
```

```
            "InitialInstanceCount": 3,
        }
    ],
)
print(f"Created Endpoint Config: {resp}")
time.sleep(5)
```

```
Endpoint Config 1: DEMO-EpConfig-1-2023-12-12-23-44-25
Endpoint Config 2: DEMO-EpConfig-2-2023-12-12-23-44-25
Endpoint Config 3: DEMO-EpConfig-3-2023-12-12-23-44-25
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-
east-1:040700907151:endpoint-config/demo-epconfig-1-2023-12-12-23-44-
25', 'ResponseMetadata': {'RequestId': '617bc1f5-f819-4d45-ae39-
8d22b94fbcf2', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': '617bc1f5-f819-4d45-ae39-8d22b94fbcf2', 'content-type':
'application/x-amz-json-1.1', 'content-length': '116', 'date': 'Tue,
12 Dec 2023 23:44:25 GMT'}, 'RetryAttempts': 0}}
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-
east-1:040700907151:endpoint-config/demo-epconfig-2-2023-12-12-23-44-
25', 'ResponseMetadata': {'RequestId': '15374af1-a367-400e-99a4-
9812de5d355a', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': '15374af1-a367-400e-99a4-9812de5d355a', 'content-type':
'application/x-amz-json-1.1', 'content-length': '116', 'date': 'Tue,
12 Dec 2023 23:44:30 GMT'}, 'RetryAttempts': 0}}
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-
east-1:040700907151:endpoint-config/demo-epconfig-3-2023-12-12-23-44-
25', 'ResponseMetadata': {'RequestId': 'b044c2d0-afba-4ff5-a20f-
83c5ec9c1b9a', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': 'b044c2d0-afba-4ff5-a20f-83c5ec9c1b9a', 'content-type':
'application/x-amz-json-1.1', 'content-length': '116', 'date': 'Tue,
12 Dec 2023 23:44:35 GMT'}, 'RetryAttempts': 0}}
```

## Create Endpoint

Deploy the baseline model to a new SageMaker endpoint:

```
endpoint_name = f"DEMO-Deployment-Guardrails-Canary-{datetime.now():
%Y-%m-%d-%H-%M-%S}"
print(f"Endpoint Name: {endpoint_name}")

resp = sm.create_endpoint(EndpointName=endpoint_name,
EndpointConfigName=ep_config_name)
print(f"\nCreated Endpoint: {resp}")
```

```
Endpoint Name: DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-09

Created Endpoint: {'EndpointArn': 'arn:aws:sagemaker:us-east-
1:040700907151:endpoint/demo-deployment-guardrails-canary-2023-12-12-
23-48-09', 'ResponseMetadata': {'RequestId': 'eb087917-53a9-41e1-a54c-
```

```
6fd2e33674c4', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': 'eb087917-53a9-41e1-a54c-6fd2e33674c4', 'content-type':
'application/x-amz-json-1.1', 'content-length': '121', 'date': 'Tue,
12 Dec 2023 23:48:09 GMT'}, 'RetryAttempts': 0}}
```

Wait for the endpoint creation to complete.

```
def wait_for_endpoint_in_service(endpoint_name):
    print("Waiting for endpoint in service")
    while True:
        details = sm.describe_endpoint(EndpointName=endpoint_name)
        status = details["EndpointStatus"]
        if status in ["InService", "Failed"]:
            print("\nDone!")
            break
        print(".", end="", flush=True)
        time.sleep(30)


wait_for_endpoint_in_service(endpoint_name)

sm.describe_endpoint(EndpointName=endpoint_name)
```

```
Waiting for endpoint in service
.....
Done!

{'EndpointName': 'DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-
09',
 'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'EndpointConfigName': 'DEMO-EpConfig-1-2023-12-12-23-44-25',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-1-cpu-py3',
     'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff622841537
61d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 12, 23, 48, 11,
21000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 12, 23, 48, 10, 137000,
tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 12, 23, 50, 29,
```

```
725000, tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': 'e6838ccb-c3f7-47fe-b5be-
ebb7b8b9eae6',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': 'e6838ccb-c3f7-47fe-b5be-
ebb7b8b9eae6',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '795',
   'date': 'Tue, 12 Dec 2023 23:50:43 GMT'},
  'RetryAttempts': 0}}
```

# Step 2: Invoke Endpoint

You can now send data to this endpoint to get inferences in real time.

This step invokes the endpoint with included sample data with maximum invocations count and waiting intervals.

```
def invoke_endpoint(
    endpoint_name, max_invocations=600, wait_interval_sec=1,
should_raise_exp=False
):
    print(f"Sending test traffic to the endpoint {endpoint_name}. \
nPlease wait...")

    count = 0
    with open("test_data/test2.csv", "r") as f:
        for row in f:
            payload = row.rstrip("\n")
            try:
                response = sm_runtime.invoke_endpoint(
                    EndpointName=endpoint_name,
ContentType="text/csv", Body=payload
                )
                response["Body"].read()
                print(".", end="", flush=True)
            except Exception as e:
                print("E", end="", flush=True)
                if should_raise_exp:
                    raise e
            count += 1
            if count > max_invocations:
                break
            time.sleep(wait_interval_sec)

    print("\nDone!")
```

```
invoke_endpoint(endpoint_name, max_invocations=100)
```

```
Sending test traffic to the endpoint DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09.
Please wait...
.....................................................................
.............................
Done!
```

## Invocations Metrics

Amazon SageMaker emits metrics such as Latency and Invocations per variant/Endpoint Config (full list of metrics here) in Amazon CloudWatch.

Query CloudWatch to get number of Invocations and latency metrics per variant and endpoint configuration.

```
import pandas as pd

cw = boto3.Session().client("cloudwatch", region_name=region)


def get_sagemaker_metrics(
    endpoint_name,
    endpoint_config_name,
    variant_name,
    metric_name,
    statistic,
    start_time,
    end_time,
):
    dimensions = [
        {"Name": "EndpointName", "Value": endpoint_name},
        {"Name": "VariantName", "Value": variant_name},
    ]
    if endpoint_config_name is not None:
        dimensions.append({"Name": "EndpointConfigName", "Value":
endpoint_config_name})
    metrics = cw.get_metric_statistics(
        Namespace="AWS/SageMaker",
        MetricName=metric_name,
        StartTime=start_time,
        EndTime=end_time,
        Period=60,
        Statistics=[statistic],
        Dimensions=dimensions,
    )
    rename = endpoint_config_name if endpoint_config_name is not None
else "ALL"
```

```
    if len(metrics["Datapoints"]) == 0:
        return
    return (
        pd.DataFrame(metrics["Datapoints"])
        .sort_values("Timestamp")
        .set_index("Timestamp")
        .drop(["Unit"], axis=1)
        .rename(columns={statistic: rename})
    )


def plot_endpoint_invocation_metrics(
    endpoint_name,
    endpoint_config_name,
    variant_name,
    metric_name,
    statistic,
    start_time=None,
):
    start_time = start_time or datetime.now(timezone.utc) -
timedelta(minutes=60)
    end_time = datetime.now(timezone.utc)
    metrics_variants = get_sagemaker_metrics(
        endpoint_name,
        endpoint_config_name,
        variant_name,
        metric_name,
        statistic,
        start_time,
        end_time,
    )
    if metrics_variants is None:
        return
    metrics_variants.plot(title=f"{metric_name}-{statistic}")
    return metrics_variants
```

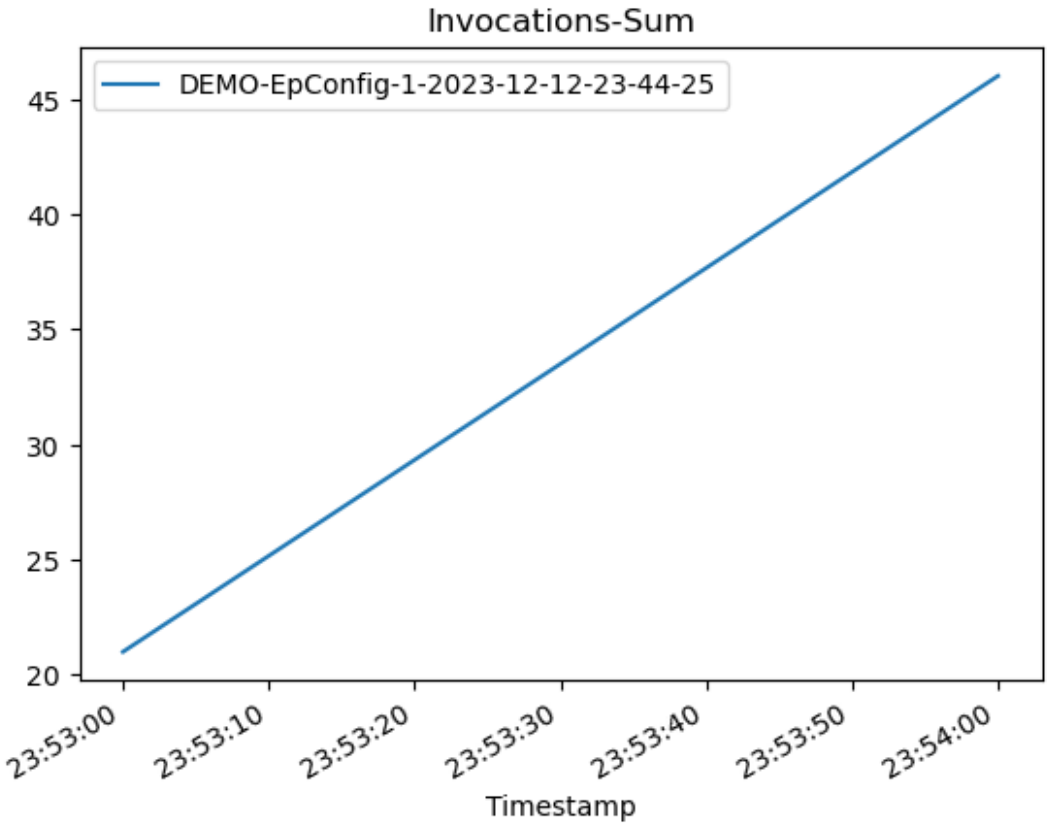
## Plot endpoint invocation metrics:

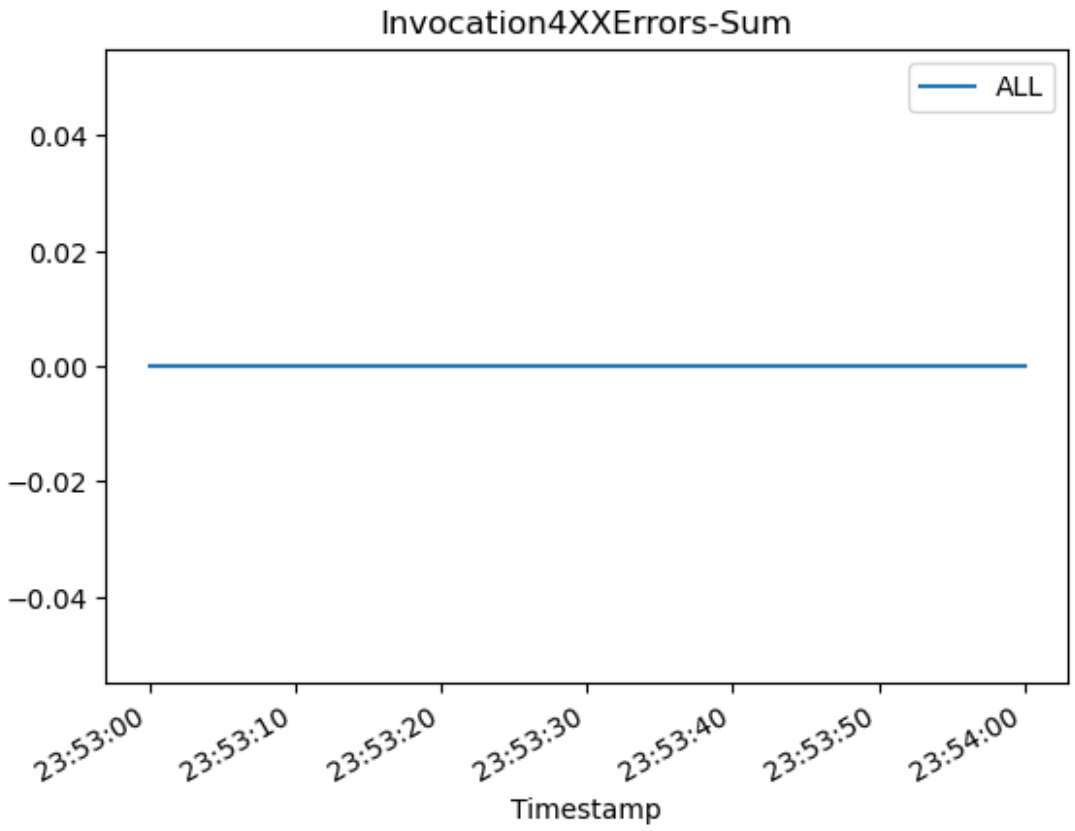
Below, we are going to plot graphs to show the Invocations,Invocation4XXErrors,Invocation5XXErrors,ModelLatency and OverheadLatency against the Endpoint.

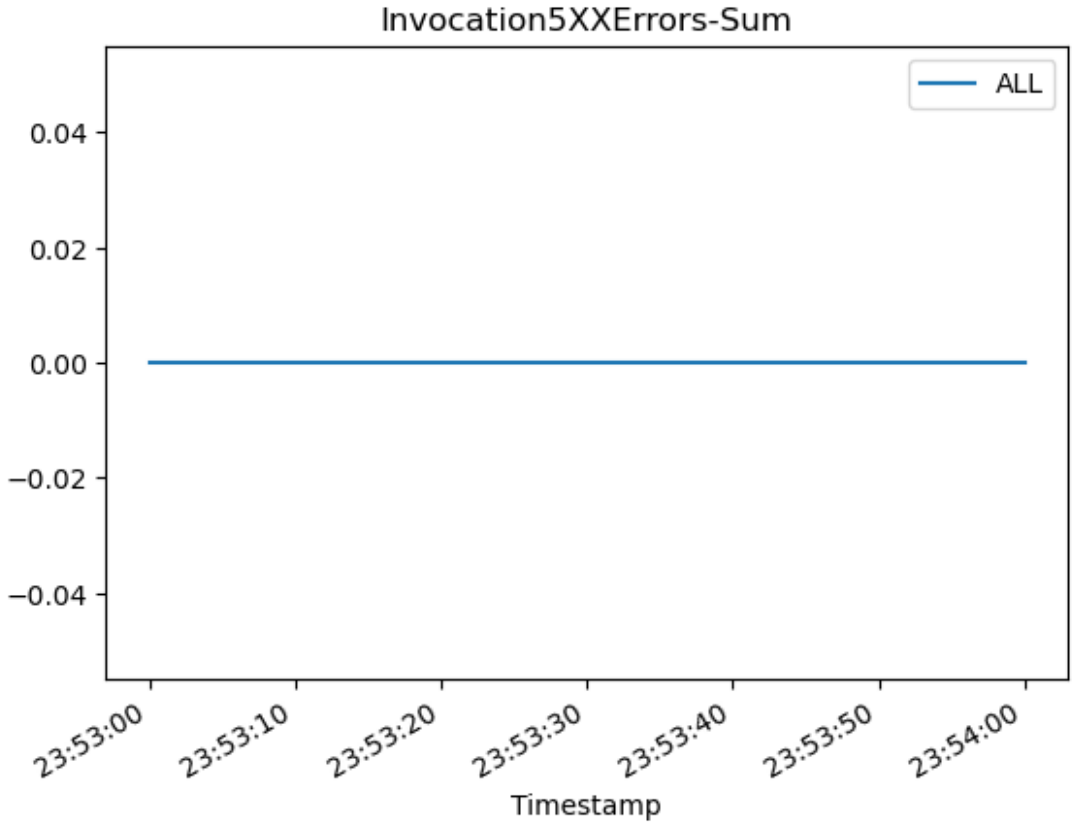
You will observe that there should be a flat line for Invocation4XXErrors and Invocation5XXErrors as we are using the correct model version and configs. Additionally, ModelLatency and OverheadLatency will start decreasing over time.

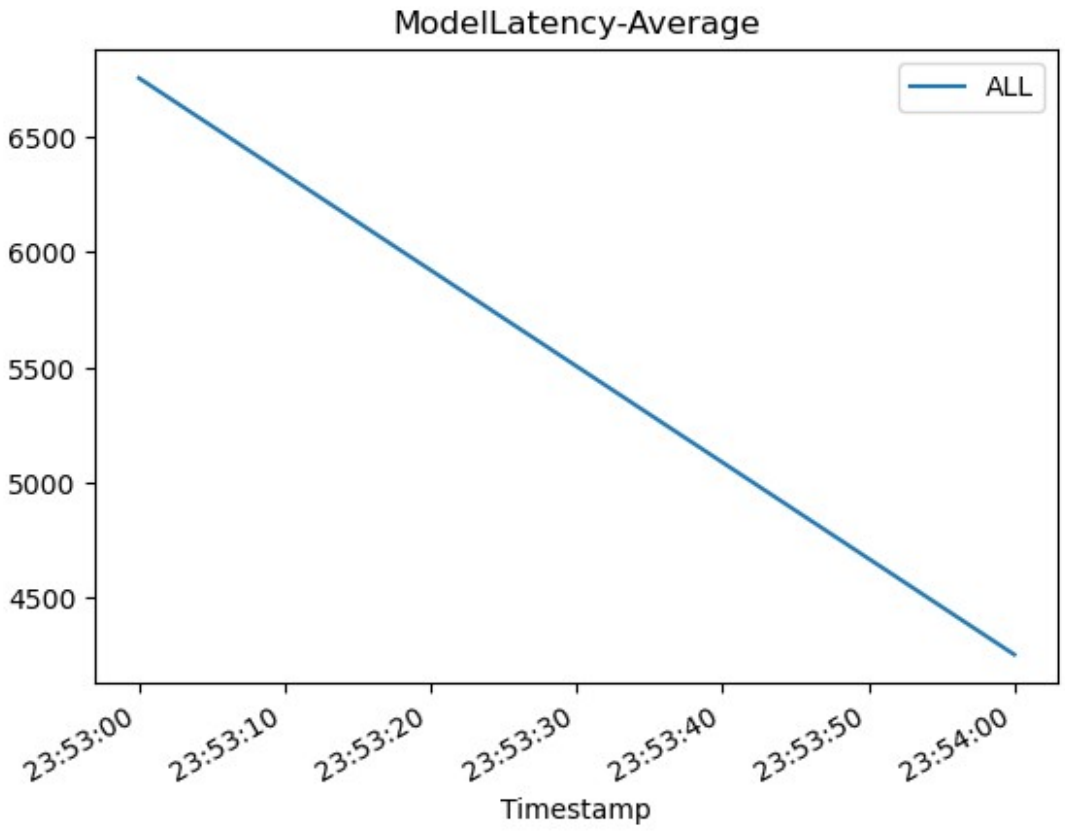
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
```

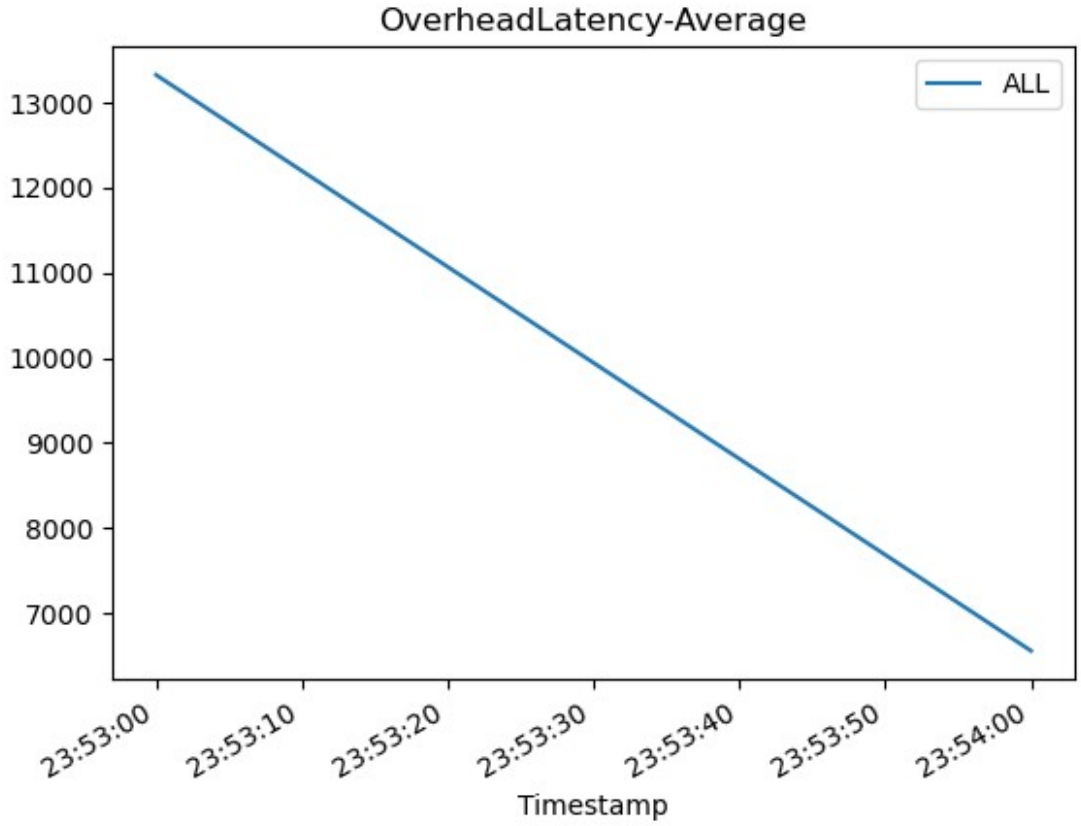
```
invocation_4xx_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocation4XXErrors", "Sum"
)
invocation_5xx_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocation5XXErrors", "Sum"
)
model_latency_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "ModelLatency", "Average"
)
overhead_latency_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "OverheadLatency", "Average"
)
```




Invocation4XXErrors-Sum
ALL
0.04
0.02
0.00
−0.02
−0.04
23:53:00
23:53:10
23:53:20
23:53:30
23:53:40
23:53:50
23:54:00
Timestamp


Invocation5XXErrors-Sum
ALL
0.04
0.02
0.00
−0.02
−0.04
23:53:00
23:53:10
23:53:20
23:53:30
23:53:40
23:53:50
23:54:00
Timestamp


ModelLatency-Average
ALL
6500
6000
5500
5000
4500
23:53:00
23:53:10
23:53:20
23:53:30
23:53:40
23:53:50
23:54:00
Timestamp


OverheadLatency-Average
ALL
13000
12000
11000
10000
9000
8000
7000
23:53:00
23:53:10
23:53:20
23:53:30
23:53:40
23:53:50
23:54:00
Timestamp

# Step 3: Create CloudWatch alarms to monitor Endpoint performance

Create CloudWatch alarms to monitor Endpoint performance with following metrics:

- Invocation5XXErrors
- ModelLatency

Following metric dimensions are used to select the metric per Endpoint config and variant:

- EndpointName
- VariantName

```
def create_auto_rollback_alarm(
    alarm_name, endpoint_name, variant_name, metric_name, statistic,
threshold
):
    cw.put_metric_alarm(
        AlarmName=alarm_name,
        AlarmDescription="Test SageMaker endpoint deployment auto-
rollback alarm",
        ActionsEnabled=False,
        Namespace="AWS/SageMaker",
        MetricName=metric_name,
        Statistic=statistic,
        Dimensions=[
            {"Name": "EndpointName", "Value": endpoint_name},
            {"Name": "VariantName", "Value": variant_name},
        ],
        Period=60,
        EvaluationPeriods=1,
        Threshold=threshold,
        ComparisonOperator="GreaterThanOrEqualToThreshold",
        TreatMissingData="notBreaching",
    )

error_alarm = f"TestAlarm-5XXErrors-{endpoint_name}"
latency_alarm = f"TestAlarm-ModelLatency-{endpoint_name}"

# alarm on 1% 5xx error rate for 1 minute
create_auto_rollback_alarm(
    error_alarm, endpoint_name, "AllTraffic", "Invocation5XXErrors",
"Average", 1
)
# alarm on model latency >= 10 ms for 1 minute
create_auto_rollback_alarm(
    latency_alarm, endpoint_name, "AllTraffic", "ModelLatency",
"Average", 10000
)
```

```
cw.describe_alarms(AlarmNames=[error_alarm, latency_alarm])
time.sleep(60)
```

# Step 4: Update Endpoint with deployment configurations

Update the endpoint with deployment configurations and monitor the performance from CloudWatch metrics.

## BlueGreen update policy with Canary traffic shifting

We define the following deployment configuration to perform Blue/Green update strategy with Canary traffic shifting from old to new stack. The Canary traffic shifting option can reduce the blast ratio of a regressive update to the endpoint. In contrast, for the All-At-Once traffic shifting option, the invocation requests start failing at 100% after flipping the traffic. In the Canary mode, invocation requests are shifted to the new version of model gradually, preventing errors from impacting 100% of your traffic. Additionally, the auto-rollback alarms monitor the metrics during the canary stage.

## Rollback Case

Rollback case

Update the Endpoint with an incompatible model version to simulate errors and trigger a rollback.

```
canary_deployment_config = {
    "BlueGreenUpdatePolicy": {
        "TrafficRoutingConfiguration": {
            "Type": "CANARY",
            "CanarySize": {
                "Type": "INSTANCE_COUNT",  # or use "CAPACITY_PERCENT"
as 30%, 50%
                "Value": 1,
            },
            "WaitIntervalInSeconds": 300,  # wait for 5 minutes before
enabling traffic on the rest of fleet
        },
        "TerminationWaitInSeconds": 120,  # wait for 2 minutes before
terminating the old stack
        "MaximumExecutionTimeoutInSeconds": 1800,  # maximum timeout
for deployment
    },
    "AutoRollbackConfiguration": {
        "Alarms": [{"AlarmName": error_alarm}, {"AlarmName":
latency_alarm}],
```

```
    },
}

# update endpoint request with new DeploymentConfig parameter
sm.update_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=ep_config_name2,
    DeploymentConfig=canary_deployment_config,
)

{'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'ResponseMetadata': {'RequestId': '18640bf5-3940-4e56-a4b4-
5f498ecdaa3c',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '18640bf5-3940-4e56-a4b4-
5f498ecdaa3c',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '121',
   'date': 'Tue, 12 Dec 2023 23:57:21 GMT'},
  'RetryAttempts': 0}}

sm.describe_endpoint(EndpointName=endpoint_name)

{'EndpointName': 'DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-
09',
 'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'EndpointConfigName': 'DEMO-EpConfig-1-2023-12-12-23-44-25',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-1-cpu-py3',
     'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff622841537
61d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 12, 23, 48, 11,
21000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'Updating',
 'CreationTime': datetime.datetime(2023, 12, 12, 23, 48, 10, 137000,
tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 12, 23, 57, 22,
870000, tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy':
```

```
{'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-
5XXErrors-DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-09'},
    {'AlarmName': 'TestAlarm-ModelLatency-DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09'}]}},
 'ResponseMetadata': {'RequestId': 'f0e9aefe-43f3-4d71-a159-
a0ff9dee2c46',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': 'f0e9aefe-43f3-4d71-a159-
a0ff9dee2c46',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1264',
   'date': 'Tue, 12 Dec 2023 23:57:23 GMT'},
  'RetryAttempts': 0}}
```

## We invoke the endpoint during the update operation is in progress.

**Note : Invoke endpoint in this notebook is in single thread mode, to stop the invoke requests please stop the cell execution**

The E's denote the errors generated from the incompatible model version in the canary fleet.

The purpose of the below cell is to simulate errors in the canary fleet. Since the nature of traffic shifting to the canary fleet is probabilistic, you should wait until you start seeing errors. Then, you may proceed to stop the execution of the below cell. If not aborted, cell will run for 600 invocations.

```
invoke_endpoint(endpoint_name)
```

```
Sending test traffic to the endpoint DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09.
Please wait...
................................................................
................................................................
.....................................E..E.....EE..E.......EE....EE.
........E.........E..............................................
................................................
Done!
```

Wait for the update operation to complete and verify the automatic rollback.

```
wait_for_endpoint_in_service(endpoint_name)

sm.describe_endpoint(EndpointName=endpoint_name)
```

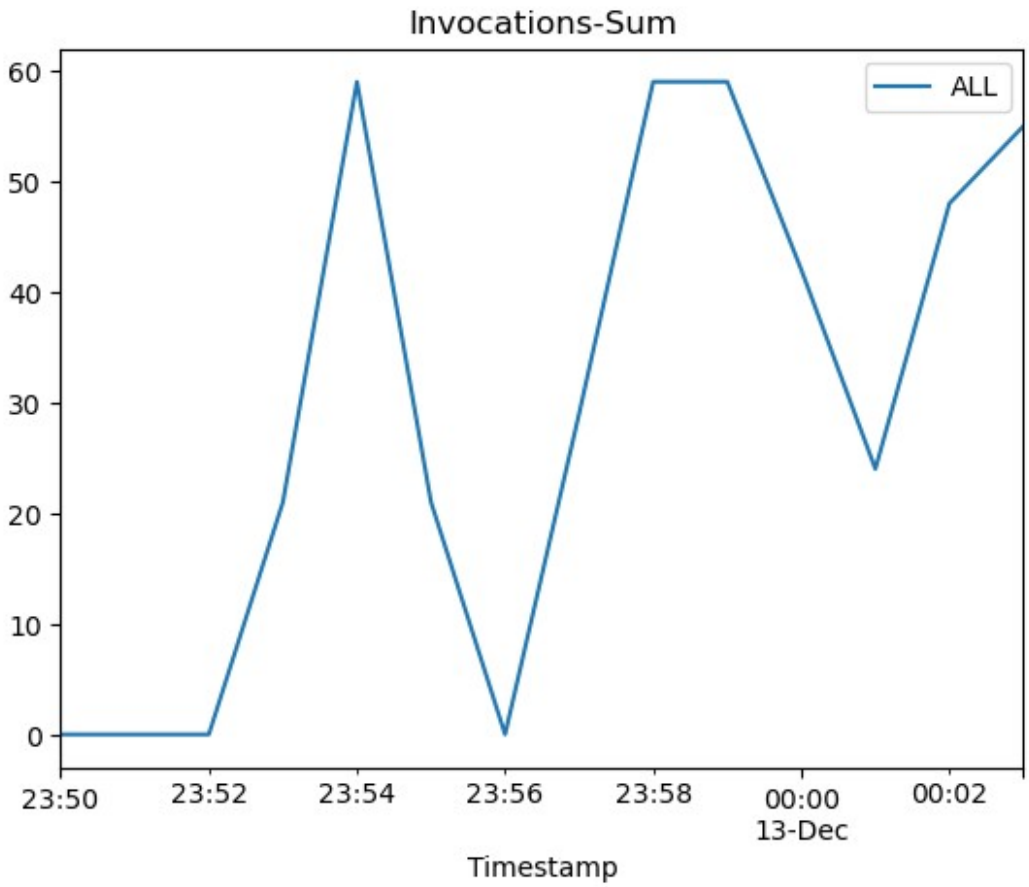
```
Waiting for endpoint in service

Done!

{'EndpointName': 'DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-
09',
 'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'EndpointConfigName': 'DEMO-EpConfig-1-2023-12-12-23-44-25',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-1-cpu-py3',
     'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff622841537
61d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 12, 23, 48, 11,
21000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'FailureReason': 'One or more configured alarm for automatic rollback
deployment is in ALARM state: [TestAlarm-ModelLatency-DEMO-Deployment-
Guardrails-Canary-2023-12-12-23-48-09].',
 'CreationTime': datetime.datetime(2023, 12, 12, 23, 48, 10, 137000,
tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 0, 1, 43, 69000,
tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy':
{'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-
5XXErrors-DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-09'},
    {'AlarmName': 'TestAlarm-ModelLatency-DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09'}]}},
 'ResponseMetadata': {'RequestId': '751ca514-b778-4580-91fb-
3d10b4f3a8ba',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '751ca514-b778-4580-91fb-
3d10b4f3a8ba',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1446',
   'date': 'Wed, 13 Dec 2023 00:04:19 GMT'},
  'RetryAttempts': 0}}
```

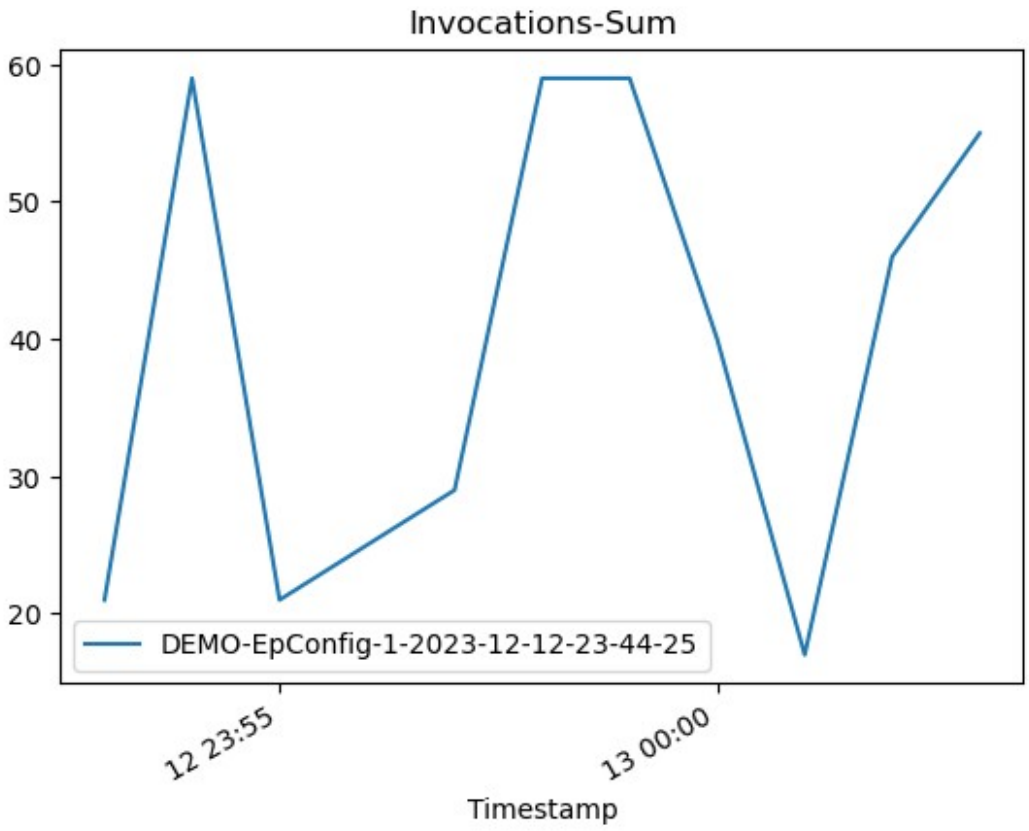
Collect the endpoint metrics during the deployment:

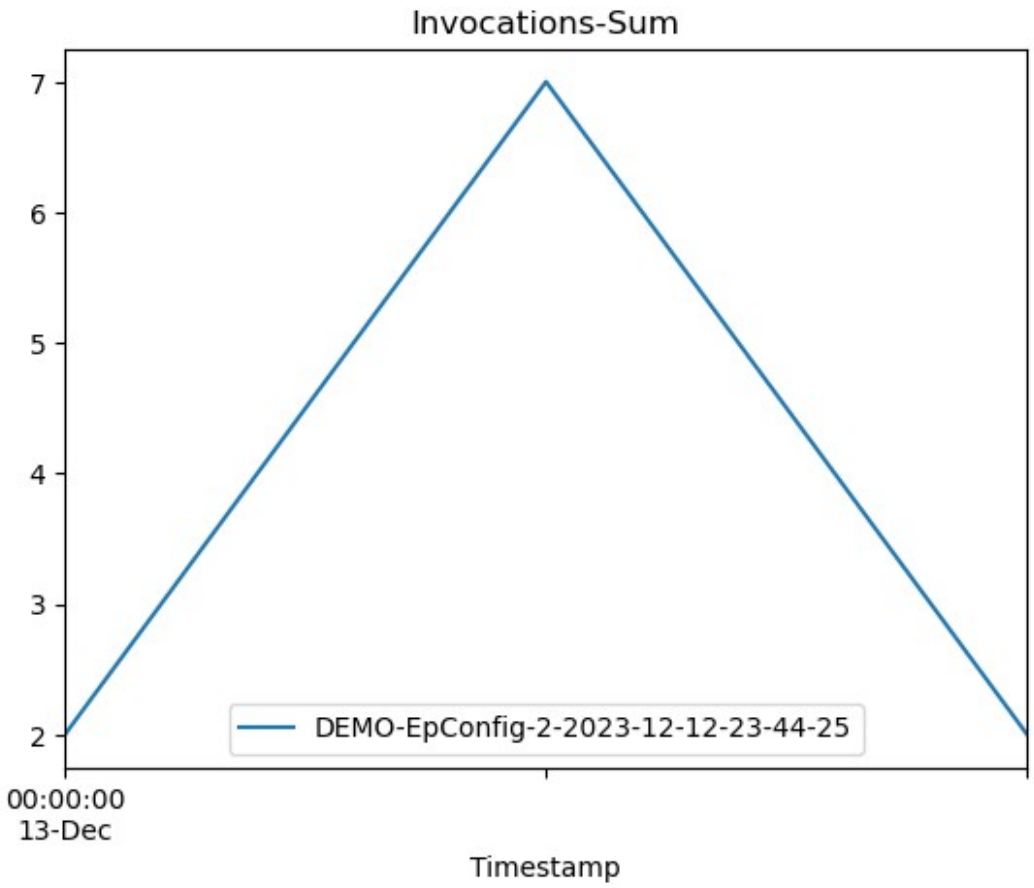
Below, we are going to plot graphs to show the Invocations,Invocation5XXErrors and ModelLatency against the Endpoint.

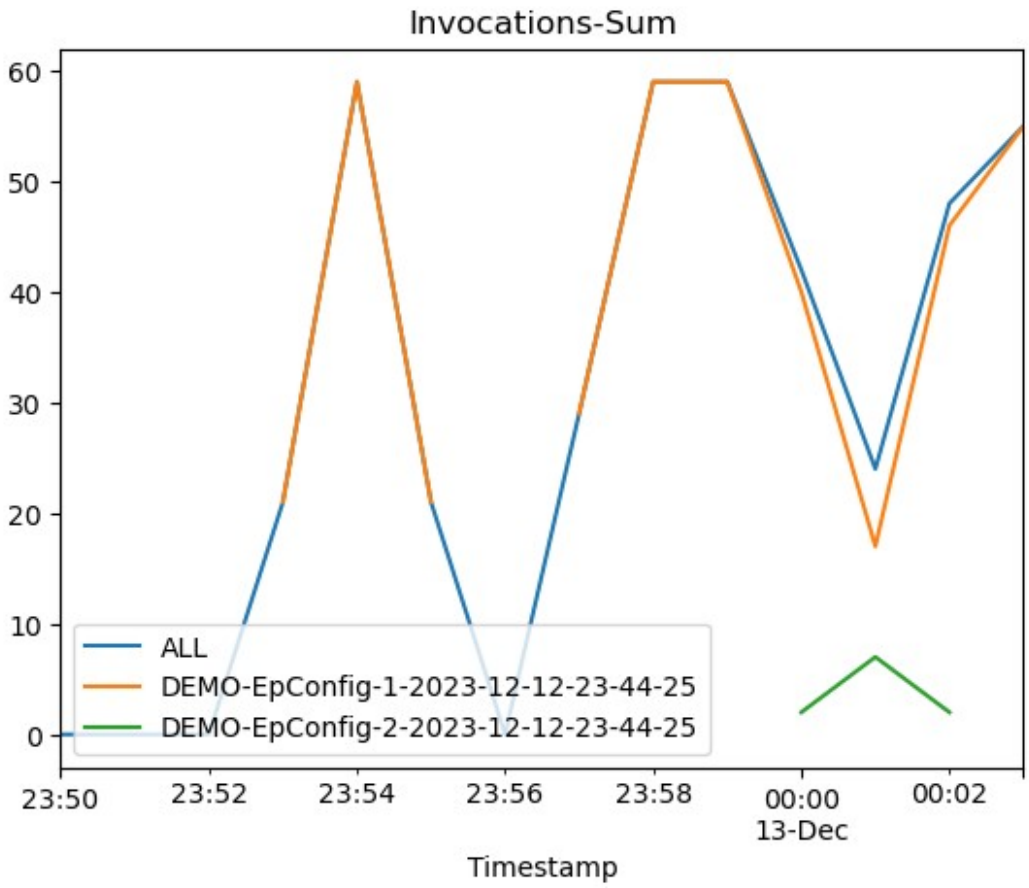
You can expect to see as the new endpoint config-2 (erroneous due to model version) starts getting deployed, it encounters failure and leads to the rollback to endpoint config-1. This can be seen in the graphs below as the Invocation5XXErrors and ModelLatency increases during this rollback phase

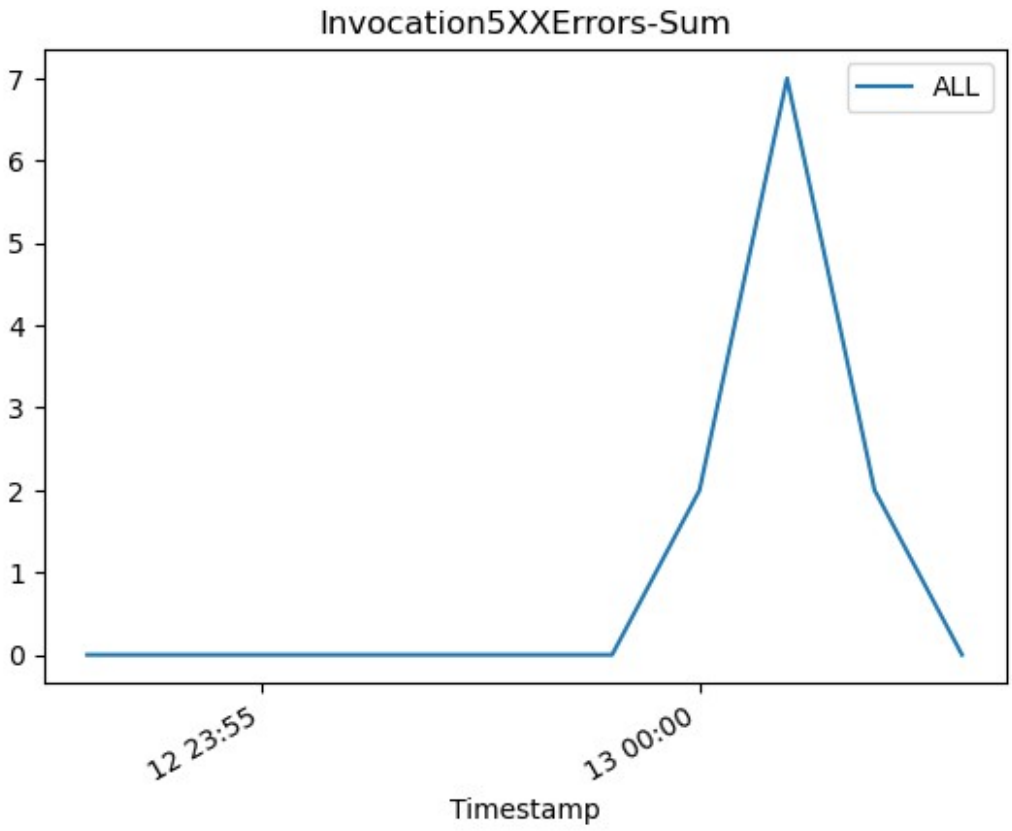
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_1 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_2 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name2, "AllTraffic", "Invocations", "Sum"
)

metrics_all = invocation_metrics.join([metrics_epc_1, metrics_epc_2],
how="outer")
metrics_all.plot(title="Invocations-Sum")

invocation_5xx_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocation5XXErrors", "Sum"
)
model_latency_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "ModelLatency", "Average"
)
```

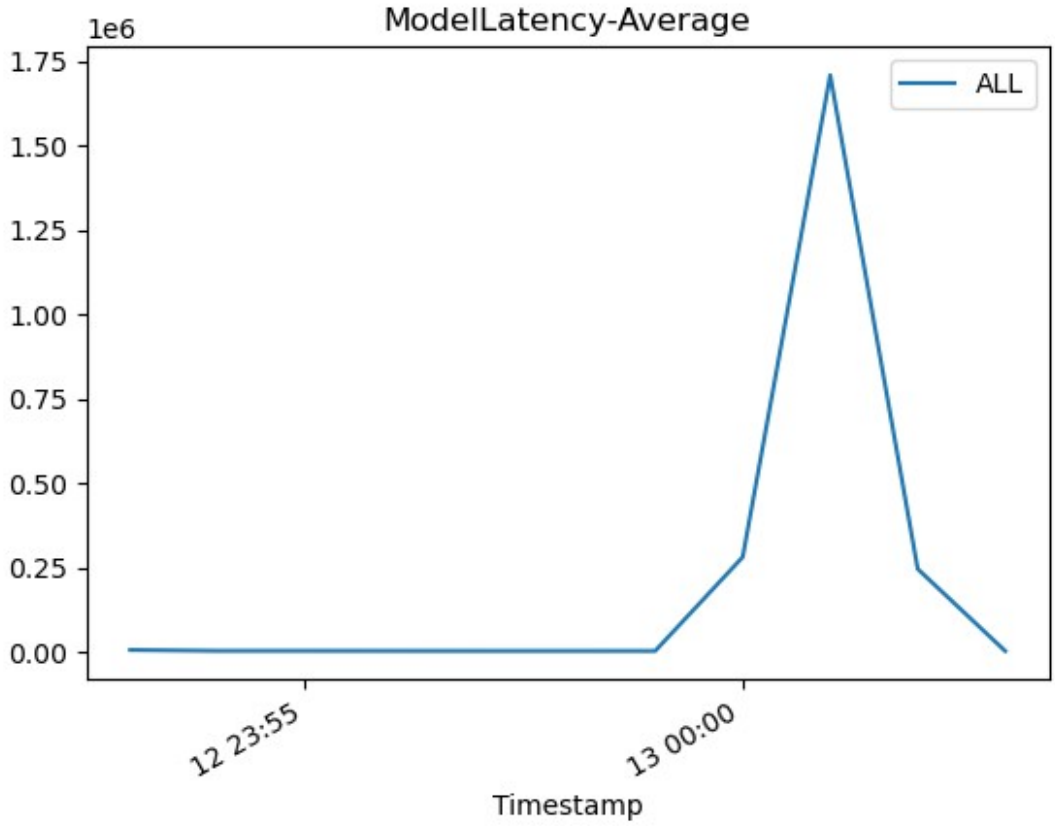

Invocations-Sum
ALL
60
50
40
30
20
10
0
23:50
23:52
23:54
23:56
23:58
00:00
13-Dec
00:02
Timestamp


Invocations-Sum
60
50
40
30
20
DEMO-EpConfig-1-2023-12-12-23-44-25
12 23:55
13 00:00
Timestamp


Invocations-Sum
7
6
5
4
3
2
DEMO-EpConfig-2-2023-12-12-23-44-25
00:00:00
13-Dec
Timestamp


Invocations-Sum
60
50
40
30
20
10
0
ALL
DEMO-EpConfig-1-2023-12-12-23-44-25
DEMO-EpConfig-2-2023-12-12-23-44-25
23:50
23:52
23:54
23:56
23:58
00:00
13-Dec
00:02
Timestamp


Invocation5XXErrors-Sum
ALL
7
6
5
4
3
2
1
0
12 23:55
13 00:00
Timestamp


ModelLatency-Average
1e6
ALL
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
12 23:55
13 00:00
Timestamp

Let's take a look at the Success case where we use the same Canary deployment configuration but a valid endpoint configuration.

## Success Case

Success case

Now we show the success case where the Endpoint Configuration is updated to a valid version (using the same Canary deployment config as the rollback case).

Update the endpoint with the same Canary deployment configuration:

```
# update endpoint with a valid version of DeploymentConfig

sm.update_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=ep_config_name3,
    RetainDeploymentConfig=True,
)
```

```
{'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'ResponseMetadata': {'RequestId': '0b741906-a034-405a-836d-
7facde723aa3',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '0b741906-a034-405a-836d-
7facde723aa3',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '121',
   'date': 'Wed, 13 Dec 2023 00:04:48 GMT'},
  'RetryAttempts': 0}}
```

```
sm.describe_endpoint(EndpointName=endpoint_name)
```

```
{'EndpointName': 'DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-
09',
 'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'EndpointConfigName': 'DEMO-EpConfig-1-2023-12-12-23-44-25',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-1-cpu-py3',
     'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff622841537
61d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 12, 23, 48, 11,
21000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
```

```
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'Updating',
 'FailureReason': 'One or more configured alarm for automatic rollback
deployment is in ALARM state: [TestAlarm-ModelLatency-DEMO-Deployment-
Guardrails-Canary-2023-12-12-23-48-09].',
 'CreationTime': datetime.datetime(2023, 12, 12, 23, 48, 10, 137000,
tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 0, 4, 49, 351000,
tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy':
{'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-
5XXErrors-DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-09'},
    {'AlarmName': 'TestAlarm-ModelLatency-DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09'}]}},
 'PendingDeploymentSummary': {'EndpointConfigName': 'DEMO-EpConfig-3-
2023-12-12-23-44-25',
  'ProductionVariants': [{'VariantName': 'AllTraffic',
    'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-2-cpu-py3',
      'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:0d098653ff2915993d61180da0cde0ed982805093463d40f30212b8
050486f18',
      'ResolutionTime': datetime.datetime(2023, 12, 13, 0, 4, 49,
592000, tzinfo=tzlocal())}],
    'CurrentWeight': 0.0,
    'DesiredWeight': 1.0,
    'CurrentInstanceCount': 0,
    'DesiredInstanceCount': 3,
    'InstanceType': 'm5.xlarge'}],
  'StartTime': datetime.datetime(2023, 12, 13, 0, 4, 52,
tzinfo=tzlocal())},
 'ResponseMetadata': {'RequestId': '6f2e98cf-3209-4b33-b661-
712693b3275b',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '6f2e98cf-3209-4b33-b661-
712693b3275b',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '2033',
   'date': 'Wed, 13 Dec 2023 00:04:52 GMT'},
  'RetryAttempts': 0}}
```

Invoke the endpoint during the update operation is in progress:

```
invoke_endpoint(endpoint_name, max_invocations=500)
```

```
Sending test traffic to the endpoint DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09.
Please wait...
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................
Done!
```

Wait for the update operation to complete:

```
#wait_for_endpoint_in_service(endpoint_name)

sm.describe_endpoint(EndpointName=endpoint_name)
```

```
{'EndpointName': 'DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-
09',
 'EndpointArn':
'arn:aws:sagemaker:us-east-1:040700907151:endpoint/demo-deployment-
guardrails-canary-2023-12-12-23-48-09',
 'EndpointConfigName': 'DEMO-EpConfig-3-2023-12-12-23-44-25',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-
east-1.amazonaws.com/sagemaker-xgboost:0.90-2-cpu-py3',
     'ResolvedImage':
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
xgboost@sha256:0d098653ff2915993d61180da0cde0ed982805093463d40f30212b8
050486f18',
     'ResolutionTime': datetime.datetime(2023, 12, 13, 0, 4, 49,
592000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 12, 23, 48, 10, 137000,
tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 0, 14, 29,
672000, tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy':
{'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
```

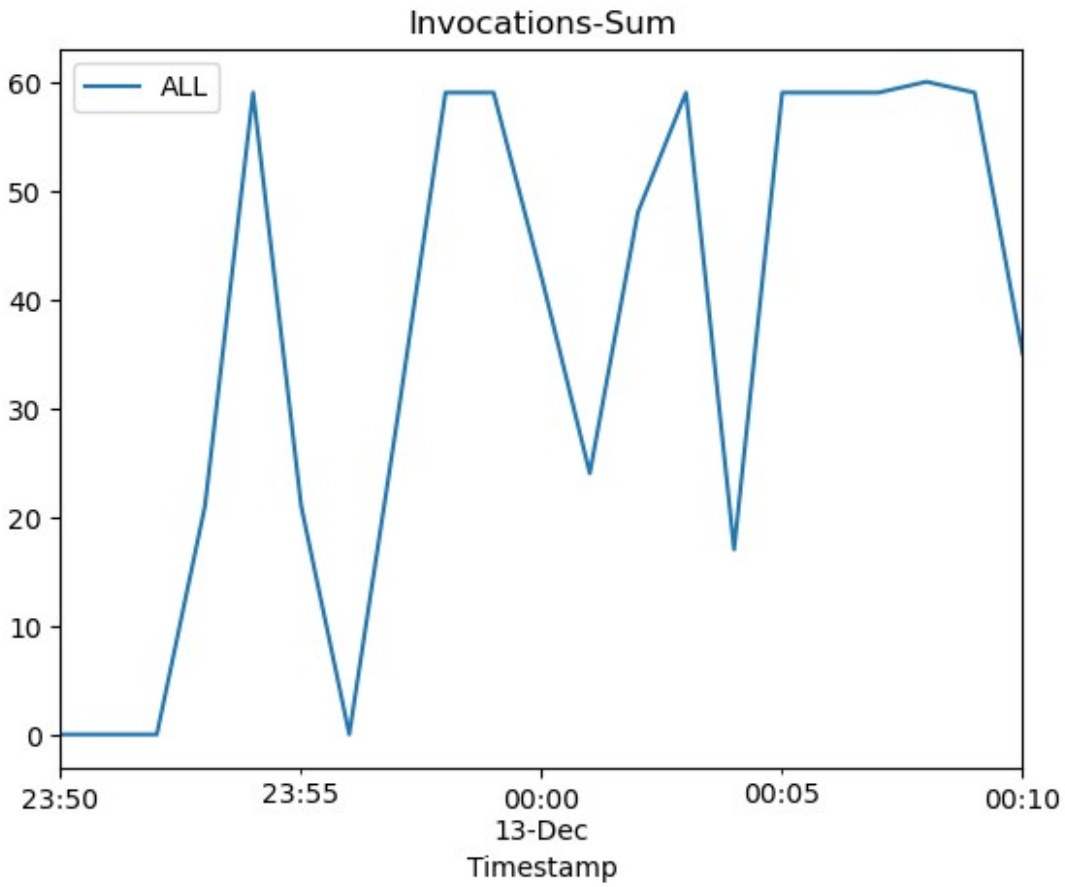
```
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-
5XXErrors-DEMO-Deployment-Guardrails-Canary-2023-12-12-23-48-09'},
    {'AlarmName': 'TestAlarm-ModelLatency-DEMO-Deployment-Guardrails-
Canary-2023-12-12-23-48-09'}]}},
 'ResponseMetadata': {'RequestId': '631485a5-8051-45da-ad4b-
4263cab8d576',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '631485a5-8051-45da-ad4b-
4263cab8d576',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1266',
   'date': 'Wed, 13 Dec 2023 00:15:32 GMT'},
  'RetryAttempts': 0}}
```

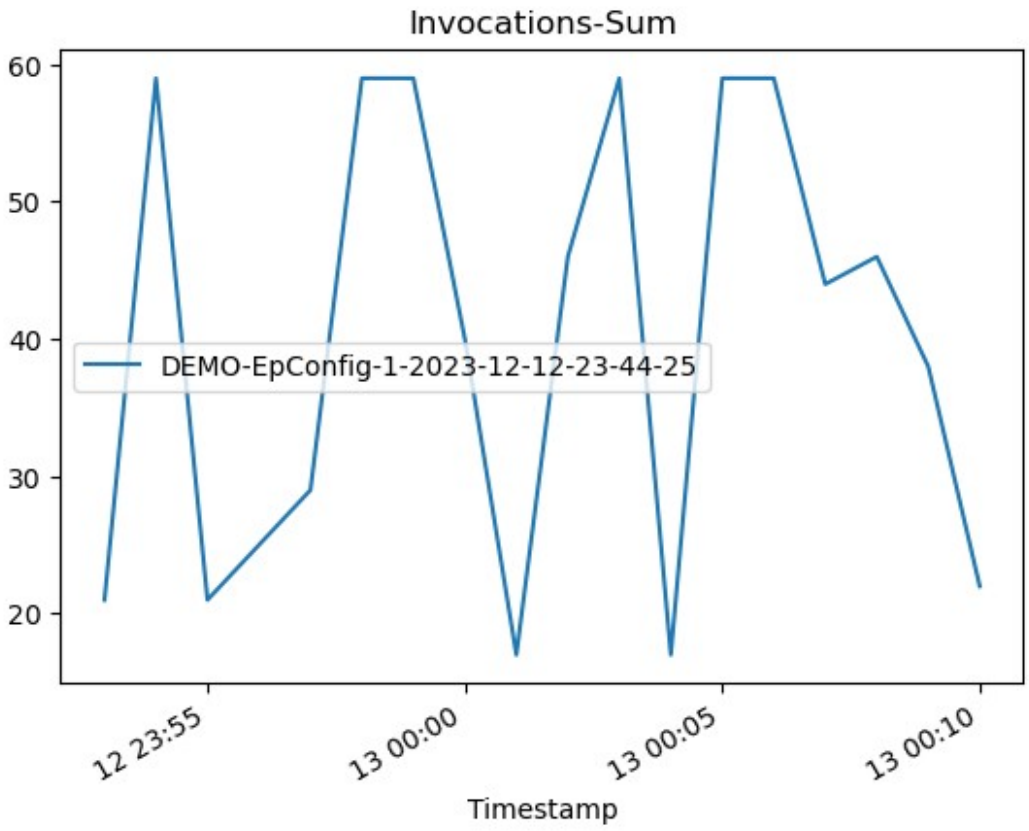
Collect the endpoint metrics during the deployment:

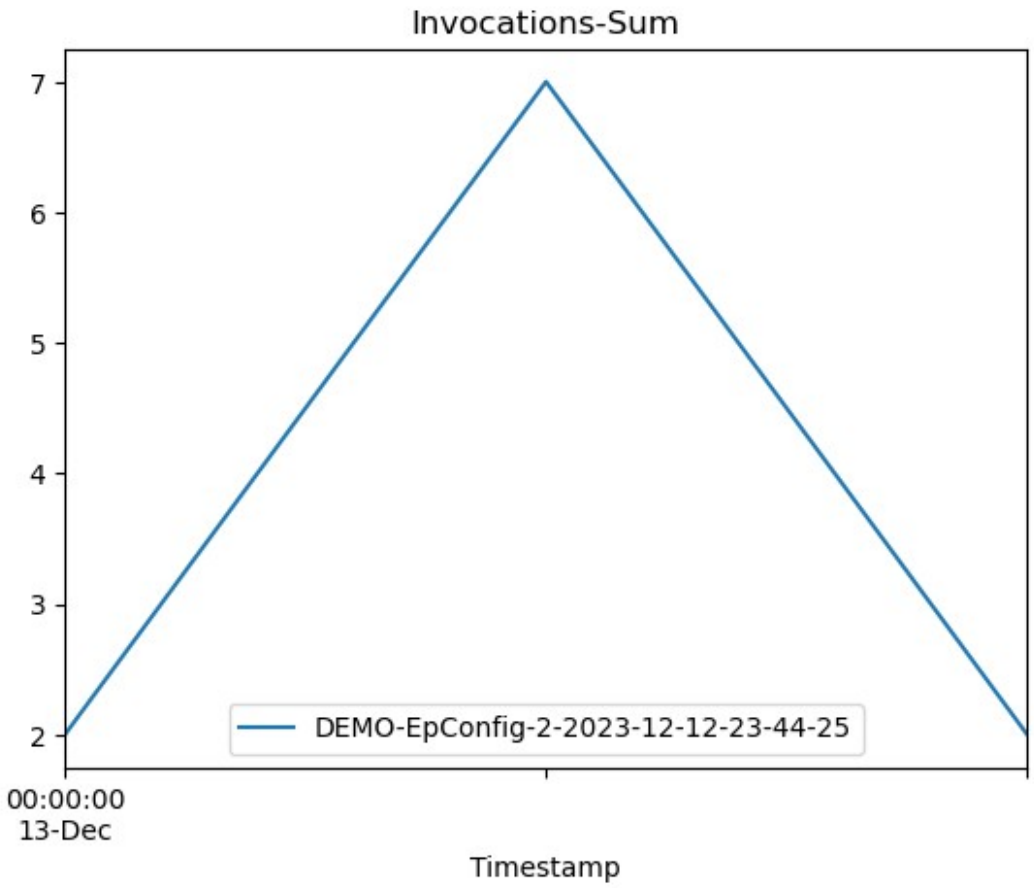
Below, we are going to plot graphs to show the Invocations,Invocation5XXErrors and ModelLatency against the Endpoint.

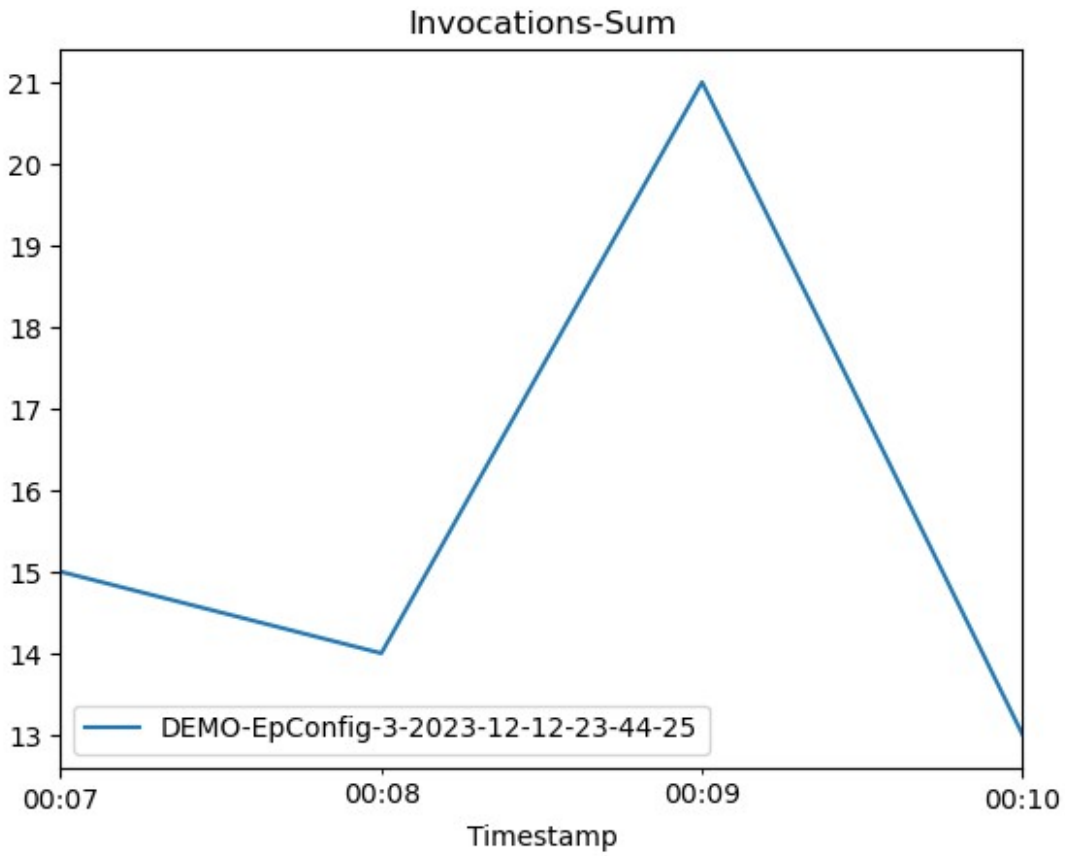
You can expect to see that, as the new endpoint config-3 (correct model version) starts getting deployed, it takes over endpoint config-2 (incompatible due to model version) without any errors. This can be seen in the graphs below as the Invocation5XXErrors and ModelLatency decreases during this transition phase

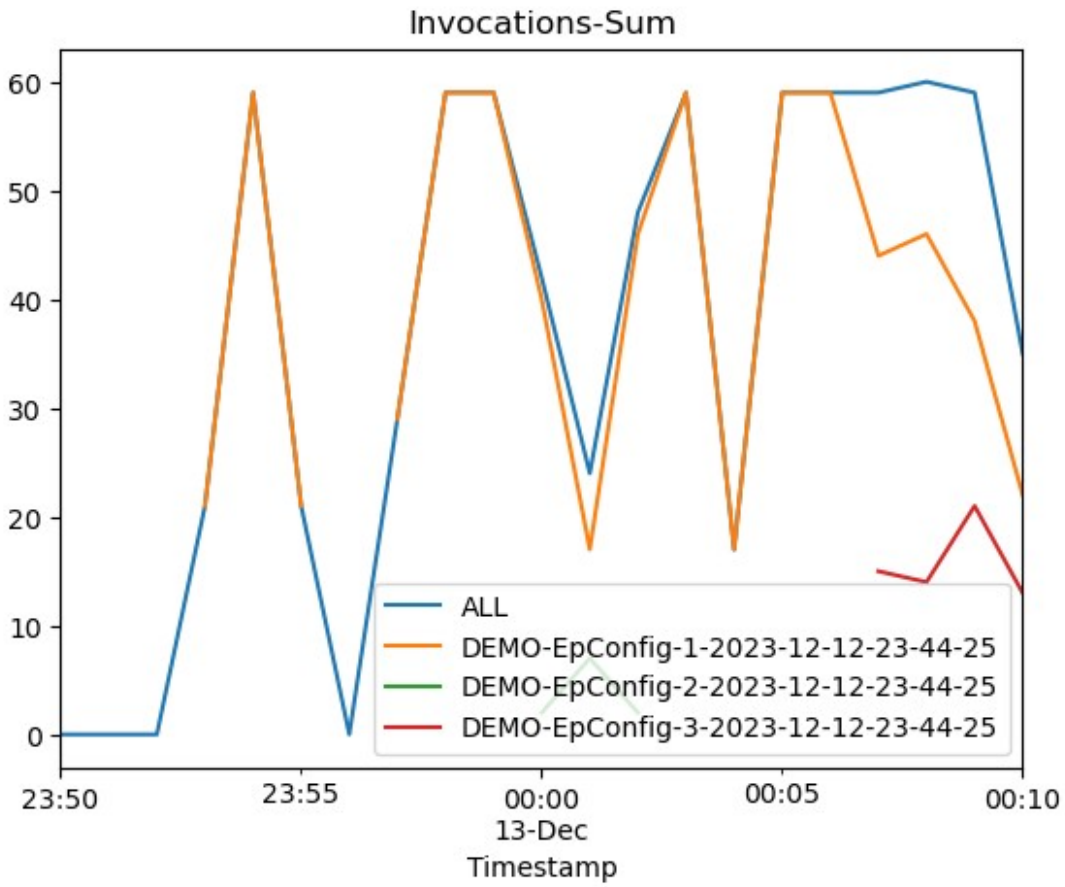
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_1 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_2 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name2, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_3 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name3, "AllTraffic", "Invocations", "Sum"
)

metrics_all = invocation_metrics.join([metrics_epc_1, metrics_epc_2,
metrics_epc_3], how="outer")
metrics_all.plot(title="Invocations-Sum")

invocation_5xx_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocation5XXErrors", "Sum"
)
model_latency_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "ModelLatency", "Average"
)
```

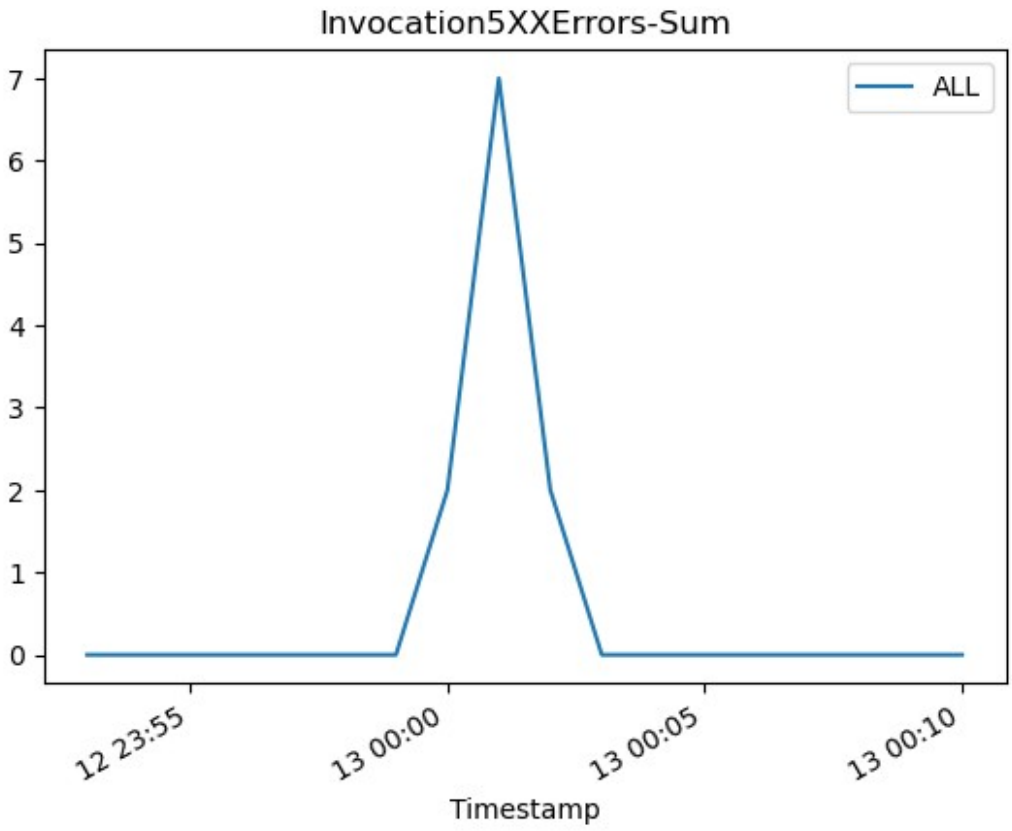

Invocations-Sum
ALL
60
50
40
30
20
10
0
23:50
23:55
00:00
13-Dec
00:05
00:10
Timestamp

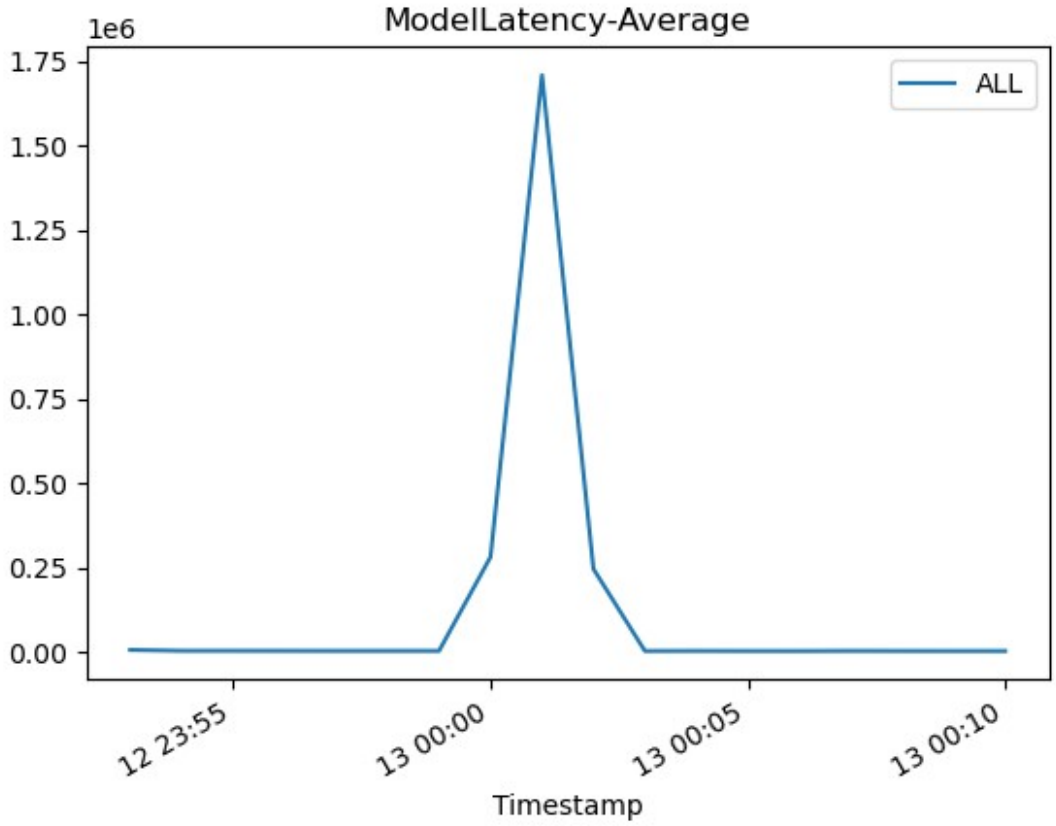

Invocations-Sum
60
50
40
30
20
DEMO-EpConfig-1-2023-12-12-23-44-25
12 23:55
13 00:00
13 00:05
13 00:10
Timestamp


Invocations-Sum
7
6
5
4
3
2
DEMO-EpConfig-2-2023-12-12-23-44-25
00:00:00
13-Dec
Timestamp


Invocations-Sum
21
20
19
18
17
16
15
14
13
DEMO-EpConfig-3-2023-12-12-23-44-25
00:07
00:08
00:09
00:10
Timestamp


Invocations-Sum
60
50
40
30
20
10
0
23:50
23:55
00:00
13-Dec
00:05
00:10
Timestamp
ALL
DEMO-EpConfig-1-2023-12-12-23-44-25
DEMO-EpConfig-2-2023-12-12-23-44-25
DEMO-EpConfig-3-2023-12-12-23-44-25


Invocation5XXErrors-Sum
ALL
7
6
5
4
3
2
1
0
12 23:55
13 00:00
13 00:05
13 00:10
Timestamp


ModelLatency-Average
1e6
ALL
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
12 23:55
13 00:00
13 00:05
13 00:10
Timestamp

The Amazon CloudWatch metrics for the total invocations for each endpoint config shows how invocation requests are shifted from the old version to the new version during deployment.

You can now safely update your endpoint and monitor model regressions during deployment and trigger auto-rollback action.

## NOTE: You need the models (Not endpoint) for Shadow Testing. Do not clean them now, until you are done with next section

# Cleanup

If you do not plan to use this endpoint further, you should delete the endpoint to avoid incurring additional charges and clean up other resources created in this notebook.

```
sm.delete_endpoint(EndpointName=endpoint_name)
```

```
{'ResponseMetadata': {'RequestId': '81970658-a280-47da-b91a-
14606972cdda',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '81970658-a280-47da-b91a-
14606972cdda',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '0',
   'date': 'Wed, 13 Dec 2023 00:16:15 GMT'},
  'RetryAttempts': 0}}
```

```
sm.delete_endpoint_config(EndpointConfigName=ep_config_name)
sm.delete_endpoint_config(EndpointConfigName=ep_config_name2)
sm.delete_endpoint_config(EndpointConfigName=ep_config_name3)
```

```
{'ResponseMetadata': {'RequestId': '106a71c5-137c-4f2a-896b-
746184dcaf26',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '106a71c5-137c-4f2a-896b-
746184dcaf26',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '0',
   'date': 'Tue, 12 Dec 2023 23:13:02 GMT'},
  'RetryAttempts': 1}}
```

```
sm.delete_model(ModelName=model_name)
sm.delete_model(ModelName=model_name2)
sm.delete_model(ModelName=model_name3)
```

```
{'ResponseMetadata': {'RequestId': '6670fb92-16ac-473b-9c28-
2684379abad1',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '6670fb92-16ac-473b-9c28-
2684379abad1',
```

```
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '0',
   'date': 'Tue, 12 Dec 2023 23:13:11 GMT'},
  'RetryAttempts': 1}}

cw.delete_alarms(AlarmNames=[error_alarm, latency_alarm])

{'ResponseMetadata': {'RequestId': '93b0bfdc-ceea-4f62-b109-
d5717dc4e5c3',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '93b0bfdc-ceea-4f62-b109-
d5717dc4e5c3',
   'content-type': 'text/xml',
   'content-length': '210',
   'date': 'Tue, 12 Dec 2023 23:13:14 GMT'},
  'RetryAttempts': 0}}
```

---

# NOTE: The following cell is for Shadow Testing.