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:

In [1]:
```
!pip install -U awscli
!pip install sagemaker
```

```
Requirement already satisfied: awscli in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-package
s (1.32.0)
Collecting botocore==1.34.0 (from awscli)
  Using cached botocore-1.34.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: docutils<0.17,>=0.10 in /home/ec2-user/anaconda3/envs/python3/lib/python3.1
0/site-packages (from awscli) (0.16)
Collecting s3transfer<0.10.0,>=0.9.0 (from awscli)
  Using cached s3transfer-0.9.0-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: PyYAML<6.1,>=3.10 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/s
ite-packages (from awscli) (6.0.1)
Requirement already satisfied: colorama<0.4.5,>=0.2.5 in /home/ec2-user/anaconda3/envs/python3/lib/python
3.10/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/sit
e-packages (from awscli) (4.7.2)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/ec2-user/anaconda3/envs/python3/lib/python
3.10/site-packages (from botocore==1.34.0->awscli) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/ec2-user/anaconda3/envs/python3/lib/py
thon3.10/site-packages (from botocore==1.34.0->awscli) (2.8.2)
Requirement already satisfied: urllib3<2.1,>=1.25.4 in /home/ec2-user/anaconda3/envs/python3/lib/python3.1
```

Setup some required imports and basic initial variables:

In [2]:
```
%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 = 'stroke-predection-project'
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::116732205680: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

In [31]:
```
!aws s3 cp s3://stroke-predection-project/sagemaker/stroke-predection/output/v5-stroke-predection-13-20-17-34-
!aws s3 cp s3://stroke-predection-project/sagemaker/stroke-predection/output/v5-stroke-predection-13-20-17-34-

!aws s3 cp s3://stroke-predection-project/sagemaker/stroke-predection/xg_test/xgboost_test_smote.csv test_data
```

```
download: s3://stroke-predection-project/sagemaker/stroke-predection/output/v5-stroke-predection-13-20-17-34-
002-a99300d4/output/model.tar.gz to model/model.tar.gz
download: s3://stroke-predection-project/sagemaker/stroke-predection/output/v5-stroke-predection-13-20-17-34-
006-4f8e01ed/output/model2.tar.gz to model/model2.tar.gz
download: s3://stroke-predection-project/sagemaker/stroke-predection/xg_test/xgboost_test_smote.csv to test_d
ata/xgboost_test_smote.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 (https://github.com/aws/amazon-sagemaker-examples/blob/master/introduction_to_applying_machine_learning/xgboost_customer_churn/xgboost_customer_churn.ipynb) 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

In [3]:
```
model_url = S3Uploader.upload(
    local_path="model/model2.tar.gz",
    desired_s3_uri=f"s3://{bucket}/{prefix}",
)
model_url2 = S3Uploader.upload(
    local_path="model/model.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://stroke-predection-project/sagemaker/DEMO-Deployment-Guardrails-Canary/model2.tar.gz
Model URI 2: s3://stroke-predection-project/sagemaker/DEMO-Deployment-Guardrails-Canary/model.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.

In [4]:
```
from sagemaker import image_uris

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

# using linear-learner 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, "0.90-1")

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:0.90-1-cpu-py3
```

In [5]:
```
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-2023-12-13-23-27-10
Model Name 2: DEMO-xgb-stroke-pred2-2023-12-13-23-27-10
Model Name 3: DEMO-xgb-stroke-pred3-2023-12-13-23-27-10
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-1:116732205680:model/demo-xgb-stroke-pred-2023-12-13-2
3-27-10', 'ResponseMetadata': {'RequestId': '144dedb7-bae1-4e57-817e-97862dc4f66d', 'HTTPStatusCode': 200, 'H
TTPHeaders': {'x-amzn-requestid': '144dedb7-bae1-4e57-817e-97862dc4f66d', 'content-type': 'application/x-amz-
json-1.1', 'content-length': '102', 'date': 'Wed, 13 Dec 2023 23:27:10 GMT'}, 'RetryAttempts': 0}}
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-1:116732205680:model/demo-xgb-stroke-pred2-2023-12-13-
23-27-10', 'ResponseMetadata': {'RequestId': 'a032cadb-3c98-4f8a-99b1-d55e257a3e5c', 'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amzn-requestid': 'a032cadb-3c98-4f8a-99b1-d55e257a3e5c', 'content-type': 'application/x-am
z-json-1.1', 'content-length': '103', 'date': 'Wed, 13 Dec 2023 23:27:13 GMT'}, 'RetryAttempts': 2}}
Created Model: {'ModelArn': 'arn:aws:sagemaker:us-east-1:116732205680:model/demo-xgb-stroke-pred3-2023-12-13-
23-27-10', 'ResponseMetadata': {'RequestId': '14f8d1c7-ee75-4cd6-8272-61f4a2d0a0cd', 'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amzn-requestid': '14f8d1c7-ee75-4cd6-8272-61f4a2d0a0cd', 'content-type': 'application/x-am
z-json-1.1', 'content-length': '103', 'date': 'Wed, 13 Dec 2023 23:27:14 GMT'}, 'RetryAttempts': 1}}
```

Create Endpoint Configs

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

In [6]:
```
ep_config_name = f"Stroke-EpConfig-1-{datetime.now():%Y-%m-%d-%H-%M-%S}"
ep_config_name2 = f"Stroke-EpConfig-2-{datetime.now():%Y-%m-%d-%H-%M-%S}"
ep_config_name3 = f"Stroke-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: Stroke-EpConfig-1-2023-12-13-23-27-22
Endpoint Config 2: Stroke-EpConfig-2-2023-12-13-23-27-22
Endpoint Config 3: Stroke-EpConfig-3-2023-12-13-23-27-22
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint-config/stro
ke-epconfig-1-2023-12-13-23-27-22', 'ResponseMetadata': {'RequestId': 'a24c632f-0c4f-4969-b833-0b5c71da0159',
'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'a24c632f-0c4f-4969-b833-0b5c71da0159', 'content-t
ype': 'application/x-amz-json-1.1', 'content-length': '118', 'date': 'Wed, 13 Dec 2023 23:27:22 GMT'}, 'Retry
Attempts': 0}}
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint-config/stro
ke-epconfig-2-2023-12-13-23-27-22', 'ResponseMetadata': {'RequestId': 'fc6dfbff-00c6-4c57-b209-7bda8f072946',
'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'fc6dfbff-00c6-4c57-b209-7bda8f072946', 'content-t
ype': 'application/x-amz-json-1.1', 'content-length': '118', 'date': 'Wed, 13 Dec 2023 23:27:27 GMT'}, 'Retry
Attempts': 0}}
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint-config/stro
ke-epconfig-3-2023-12-13-23-27-22', 'ResponseMetadata': {'RequestId': 'a127617f-4342-406b-aaca-f28d32dfc75c',
'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'a127617f-4342-406b-aaca-f28d32dfc75c', 'content-t
ype': 'application/x-amz-json-1.1', 'content-length': '118', 'date': 'Wed, 13 Dec 2023 23:27:32 GMT'}, 'Retry
Attempts': 0}}
```

Create Endpoint

Deploy the baseline model to a new SageMaker endpoint:

In [7]:
```
endpoint_name = f"Stroke-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: Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53

Created Endpoint: {'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardr
ails-canary-2023-12-13-23-27-53', 'ResponseMetadata': {'RequestId': '356c248b-6b9c-4dd4-8507-ff768d80781e',
'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '356c248b-6b9c-4dd4-8507-ff768d80781e', 'content-t
ype': 'application/x-amz-json-1.1', 'content-length': '123', 'date': 'Wed, 13 Dec 2023 23:27:53 GMT'}, 'Retry
Attempts': 0}}
```

Wait for the endpoint creation to complete.

In [8]:
```
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!
```

Out[8]:
```
{'EndpointName': 'Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-12-13-23-27-53',
 'EndpointConfigName': 'Stroke-EpConfig-1-2023-12-13-23-27-22',
 '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@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 13, 23, 27, 54, 508000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 13, 23, 27, 53, 473000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 23, 30, 8, 264000, tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': '7adcf65b-65ca-4766-85b5-5a3623b5cb4e',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '7adcf65b-65ca-4766-85b5-5a3623b5cb4e',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '801',
   'date': 'Wed, 13 Dec 2023 23:30:23 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.

In [9]:
```
def invoke_endpoint(
    endpoint_name, max_invocations=50, 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/xgboost_test_smote4.csv", "r") as f:
        for row in f:
            payload = row.rstrip("\n")

            # Convert 'FALSE' and 'TRUE' to 0 and 1 respectively
            payload = payload.replace('FALSE', '0').replace('TRUE', '1')

            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 Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53.
Please wait...
....................................................................................................
Done!
```

Invocations Metrics

Amazon SageMaker emits metrics such as Latency and Invocations per variant/Endpoint Config (full list of metrics here (https://docs.aws.amazon.com/sagemaker/latest/dg/monitoring-cloudwatch.html)) in Amazon CloudWatch.

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

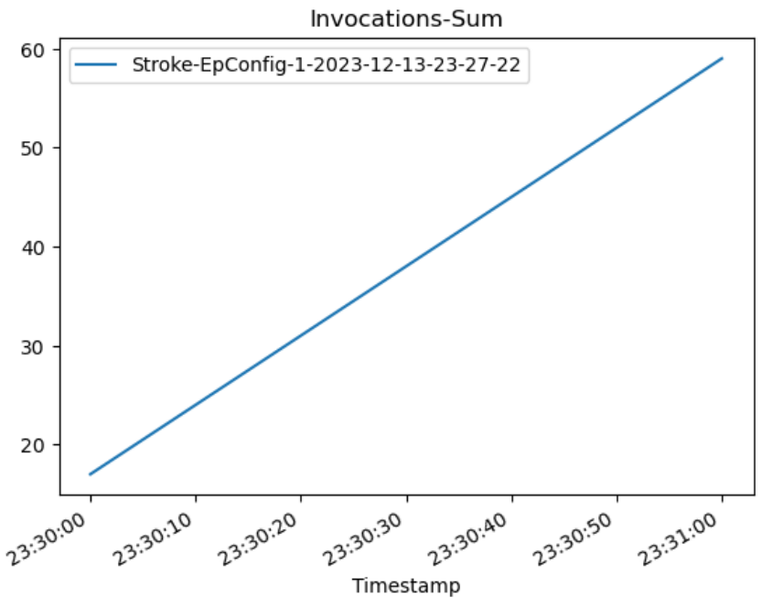
In [10]:
```
import pandas as pd

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
```

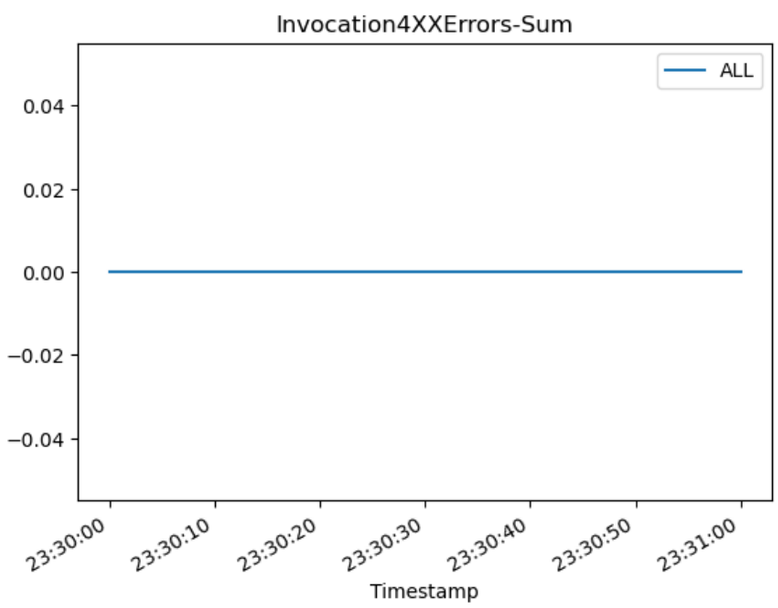
Plot endpoint invocation metrics:

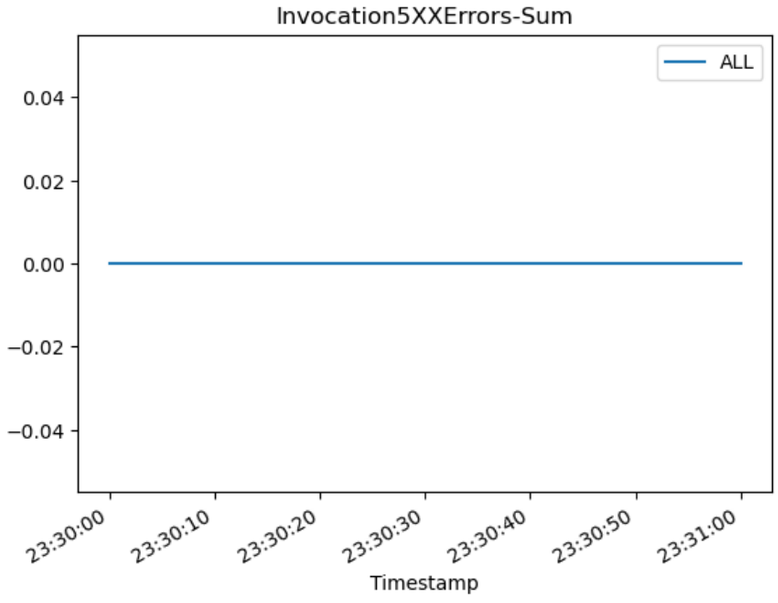
Below, we are going to plot graphs to show the Invocations,Invocation4XXErrors,Invocation5XXErrors,ModelLatency and OverheadLatency against the Endpoint.

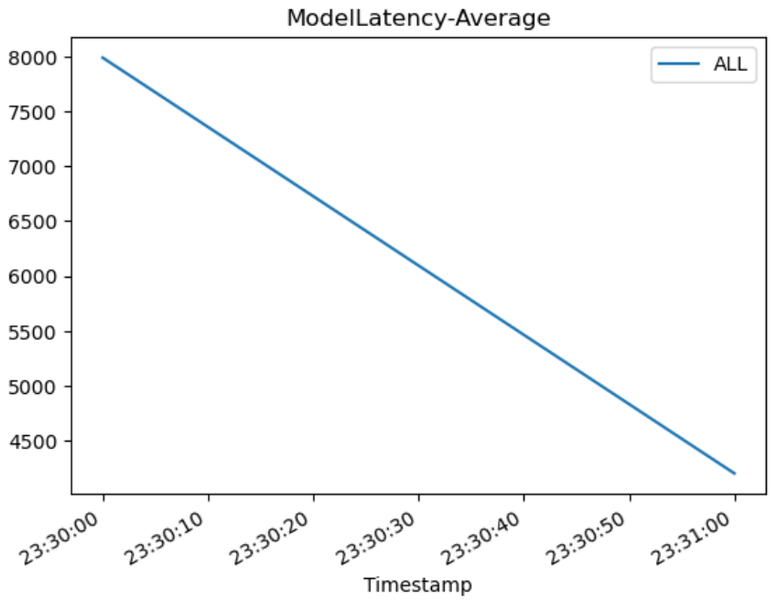
You will observe that there should be a flat line for Invocation4XXErrors and Invocation5XXErrors as we are using the correct model version and configs. Additionally, ModelLatency and OverheadLatency will start decreasing over time.

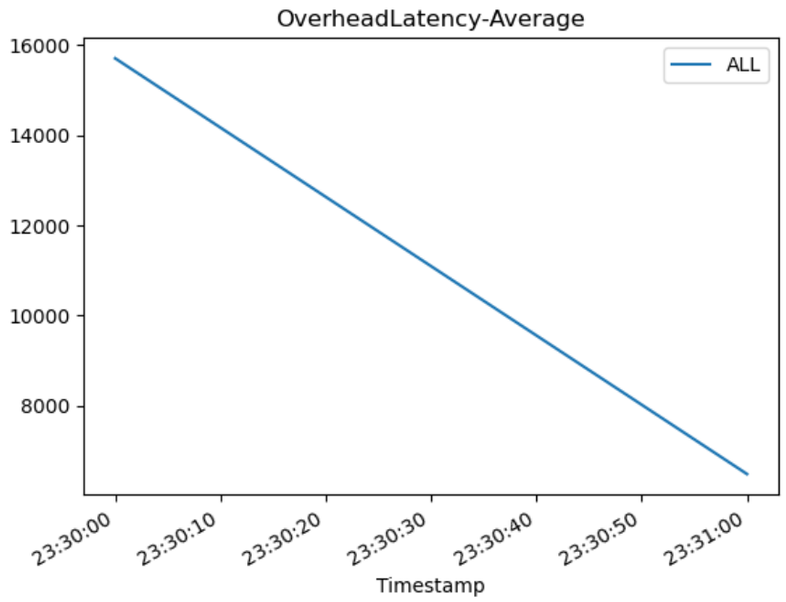
In [11]:
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
invocation_4xx_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocation4XXErrors", "Sum"
)
invocation_5xx_metrics = plot_endpoint_invocation_metrics(
    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"
)
```











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

In [12]:
```
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=True,
        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",
    )
```

In [13]:
```
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
)
```

In [14]:
```
cw.describe_alarms(AlarmNames=[error_alarm, latency_alarm])
time.sleep(60)
```

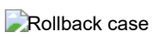
Step 4: Update Endpoint with deployment configurations

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



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

In [15]:
```
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,
)
```

Out[15]:
```
{'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-12-13-23-27-53',
 'ResponseMetadata': {'RequestId': 'abfd1b16-2b7c-4334-8627-91b8d0efea00',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': 'abfd1b16-2b7c-4334-8627-91b8d0efea00',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '123',
   'date': 'Wed, 13 Dec 2023 23:40:25 GMT'},
  'RetryAttempts': 0}}
```

In [20]:
```
sm.describe_endpoint(EndpointName=endpoint_name)
```

Out[20]:

```
{'EndpointName': 'Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-1
2-13-23-27-53',
 'EndpointConfigName': 'Stroke-EpConfig-2-2023-12-13-23-27-22',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost:1.2-
1',
     'ResolvedImage': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost@sha256:1a15d9dc919fe19c
6c8b432772cc4680eddf305e63f5f3df94caf642589df257',
     'ResolutionTime': datetime.datetime(2023, 12, 13, 23, 40, 26, 872000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 13, 23, 27, 53, 473000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 23, 50, 28, 514000, tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy': {'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-5XXErrors-Stroke-Deployment-Guardrails-Ca
nary-2023-12-13-23-27-53'},
    {'AlarmName': 'TestAlarm-ModelLatency-Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53'}]}},
 'ResponseMetadata': {'RequestId': '7d0bfe2a-5349-4d0c-9972-9cba31407162',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '7d0bfe2a-5349-4d0c-9972-9cba31407162',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1267',
   'date': 'Thu, 14 Dec 2023 00:03:44 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.

In [21]:
```
invoke_endpoint(endpoint_name)
```

```
Sending test traffic to the endpoint Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53.
Please wait...
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
Done!
```

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

In [22]:
```
wait_for_endpoint_in_service(endpoint_name)

sm.describe_endpoint(EndpointName=endpoint_name)
```

```
Waiting for endpoint in service

Done!
```

Out[22]:

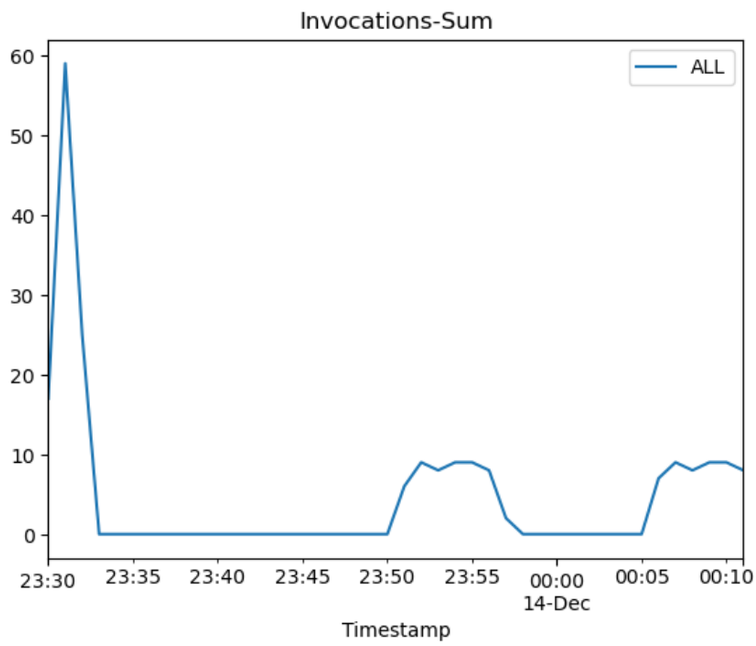
```
{'EndpointName': 'Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-12-13-23-27-53',
 'EndpointConfigName': 'Stroke-EpConfig-2-2023-12-13-23-27-22',
 'ProductionVariants': [{'VariantName': 'AllTraffic',
   'DeployedImages': [{'SpecifiedImage': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost:1.2-1',
     'ResolvedImage': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost@sha256:1a15d9dc919fe19c6c8b432772cc4680eddf305e63f5f3df94caf642589df257',
     'ResolutionTime': datetime.datetime(2023, 12, 13, 23, 40, 26, 872000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 13, 23, 27, 53, 473000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 13, 23, 50, 28, 514000, tzinfo=tzlocal()),
 'LastDeploymentConfig': {'BlueGreenUpdatePolicy': {'TrafficRoutingConfiguration': {'Type': 'CANARY',
    'WaitIntervalInSeconds': 300,
    'CanarySize': {'Type': 'INSTANCE_COUNT', 'Value': 1}},
   'TerminationWaitInSeconds': 120,
   'MaximumExecutionTimeoutInSeconds': 1800},
  'AutoRollbackConfiguration': {'Alarms': [{'AlarmName': 'TestAlarm-5XXErrors-Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53'},
    {'AlarmName': 'TestAlarm-ModelLatency-Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53'}]}},
 'ResponseMetadata': {'RequestId': '31e62ded-a195-441a-ba61-db263e00ee18',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '31e62ded-a195-441a-ba61-db263e00ee18',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1267',
   'date': 'Thu, 14 Dec 2023 00:12:04 GMT'},
  'RetryAttempts': 0}}
```

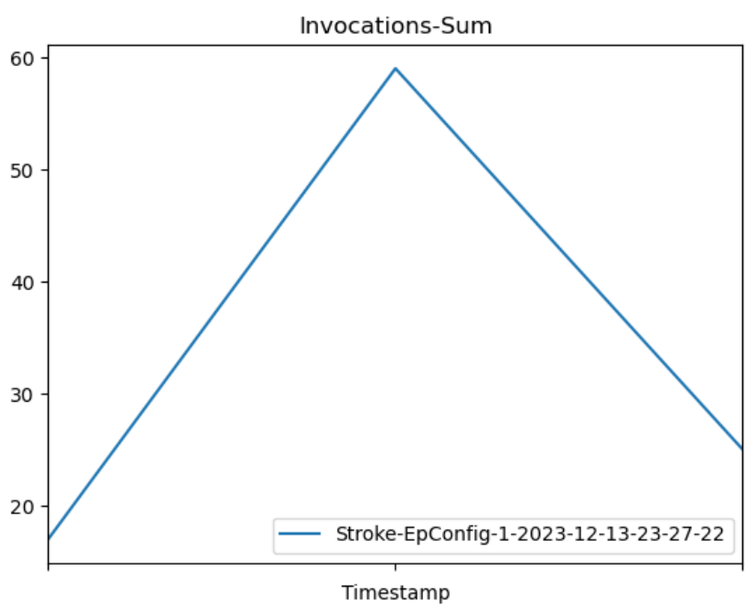

Collect the endpoint metrics during the deployment:

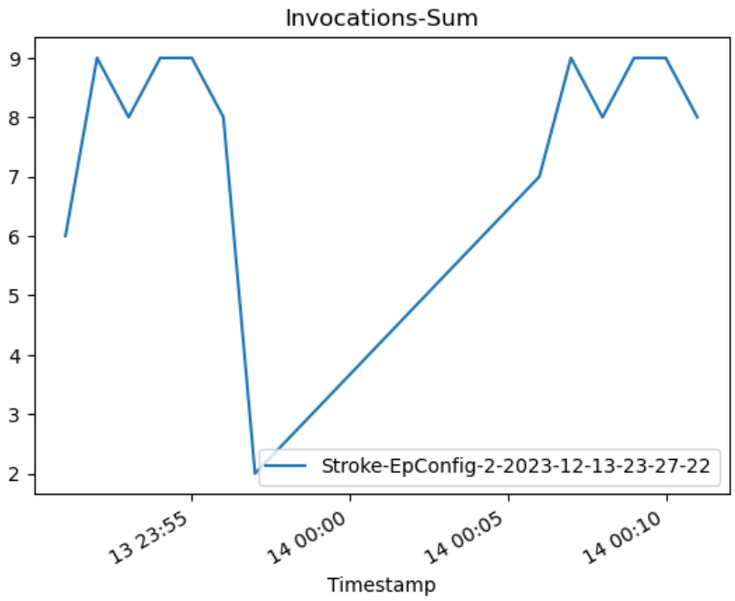
Below, we are going to plot graphs to show the Invocations,Invocation5XXErrors and ModelLatency against the Endpoint.

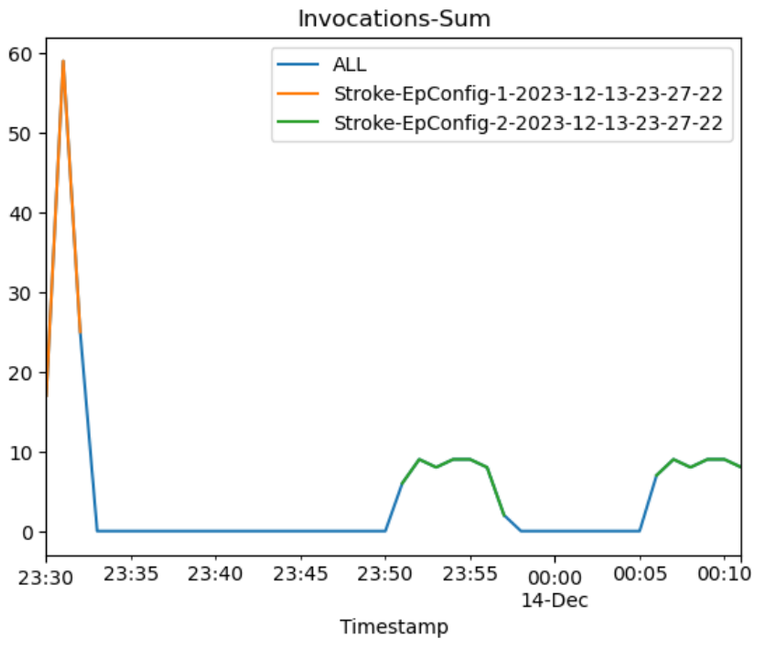
You can expect to see as the new endpoint config-2 (erroneous due to model version) starts getting deployed, it encounters failure and leads to the rollback to endpoint config-1. This can be seen in the graphs below as the Invocation5XXErrors and ModelLatency increases during this rollback phase

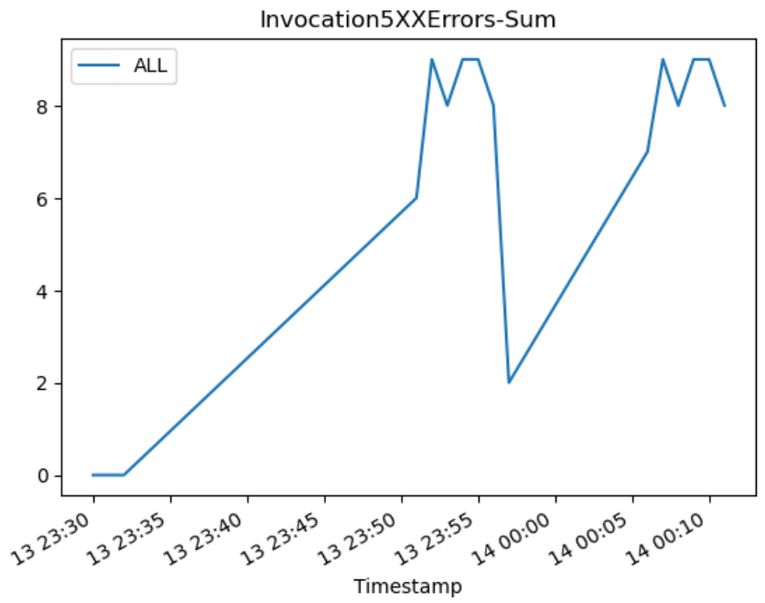
In [23]:
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_1 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_2 = plot_endpoint_invocation_metrics(
    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"
)
```

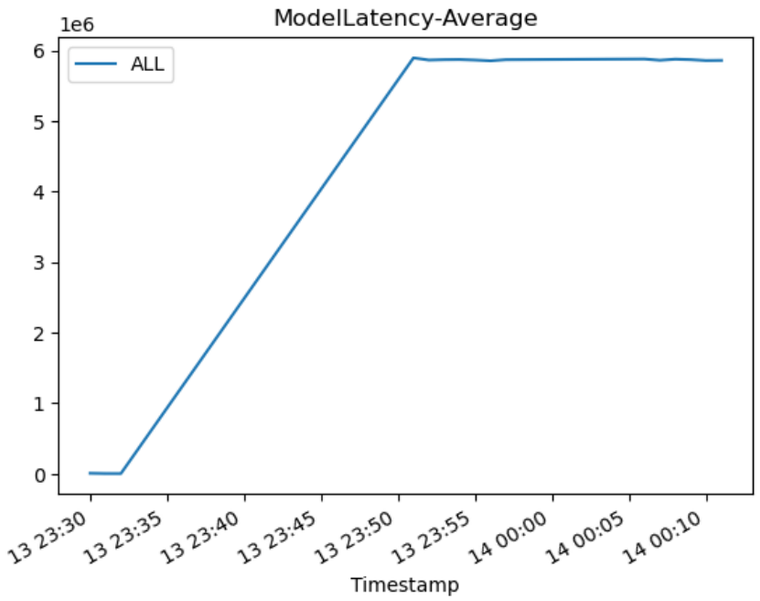





Invocations-Sum
Stroke-EpConfig-2-2023-12-13-23-27-22
13 23:55
14 00:00
14 00:05
14 00:10
Timestamp


Invocations-Sum
ALL
Stroke-EpConfig-1-2023-12-13-23-27-22
Stroke-EpConfig-2-2023-12-13-23-27-22
23:30
23:35
23:40
23:45
23:50
23:55
00:00
14-Dec
00:05
00:10
Timestamp


Invocation5XXErrors-Sum
ALL
13 23:30
13 23:35
13 23:40
13 23:45
13 23:50
13 23:55
14 00:00
14 00:05
14 00:10
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

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:

In [28]:
```
# update endpoint with a valid version of DeploymentConfig

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

Out[28]:
```
{'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-12-13-23-27-53',
 'ResponseMetadata': {'RequestId': '7bbb9c6b-9963-4c25-90fd-263ab932d8be',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '7bbb9c6b-9963-4c25-90fd-263ab932d8be',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '123',
   'date': 'Thu, 14 Dec 2023 00:20:22 GMT'},
  'RetryAttempts': 0}}
```

In [30]:
```
sm.describe_endpoint(EndpointName=endpoint_name)
```

Out[30]:
```
{'EndpointName': 'Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-12-13-23-27-53',
 'EndpointConfigName': 'Stroke-EpConfig-3-2023-12-13-23-27-22',
 '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@sha256:4814427c3e0a6cf99e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 0, 23, 50, 753000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 13, 23, 27, 53, 473000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 14, 0, 26, 9, 846000, tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': '8b1eea22-13b9-4c8c-8376-6f95363bb9c3',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '8b1eea22-13b9-4c8c-8376-6f95363bb9c3',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '801',
   'date': 'Thu, 14 Dec 2023 00:27:34 GMT'},
  'RetryAttempts': 0}}
```

Invoke the endpoint during the update operation is in progress:

In [31]:
```
invoke_endpoint(endpoint_name, max_invocations=50)
```

```
Sending test traffic to the endpoint Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53.
Please wait...
..................................................
Done!
```

Wait for the update operation to complete:

In [32]:
```
#wait_for_endpoint_in_service(endpoint_name)

sm.describe_endpoint(EndpointName=endpoint_name)
```

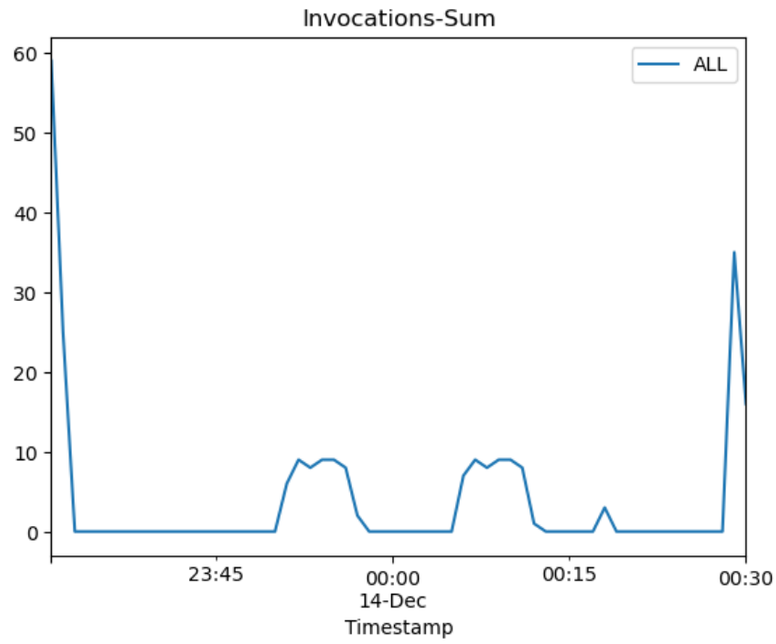
Out[32]:
```
{'EndpointName': 'Stroke-Deployment-Guardrails-Canary-2023-12-13-23-27-53',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-deployment-guardrails-canary-2023-1
2-13-23-27-53',
 'EndpointConfigName': 'Stroke-EpConfig-3-2023-12-13-23-27-22',
 'ProductionVariants': [{'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@sha256:4814427c3e0a6cf9
9e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 0, 23, 50, 753000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 13, 23, 27, 53, 473000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 14, 0, 26, 9, 846000, tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': '7c92e750-69a5-434b-9861-a039323c3cfa',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '7c92e750-69a5-434b-9861-a039323c3cfa',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '801',
   'date': 'Thu, 14 Dec 2023 00:31:30 GMT'},
  'RetryAttempts': 0}}
```

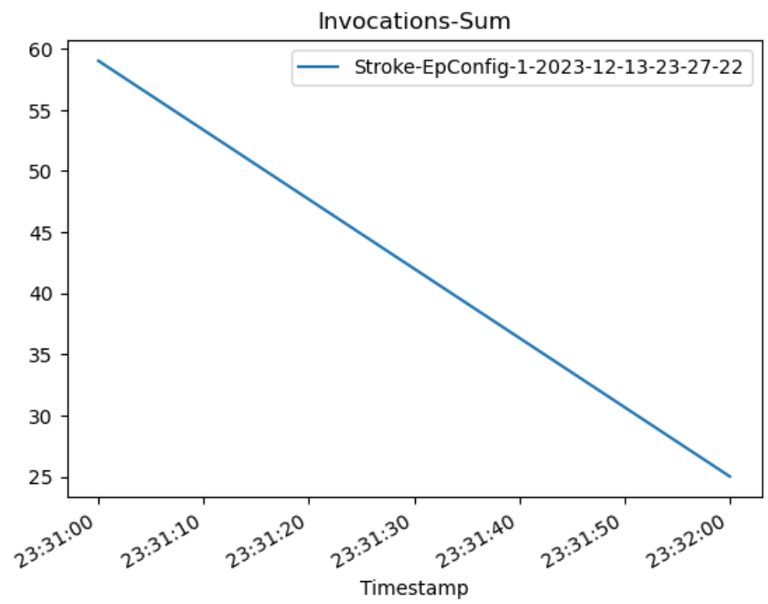
Collect the endpoint metrics during the deployment:

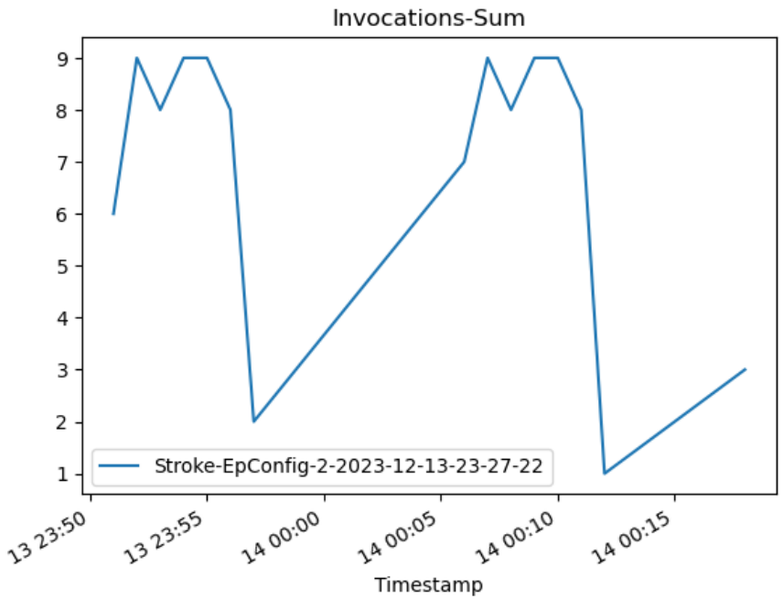
Below, we are going to plot graphs to show the Invocations,Invocation5XXErrors and ModelLatency against the Endpoint.

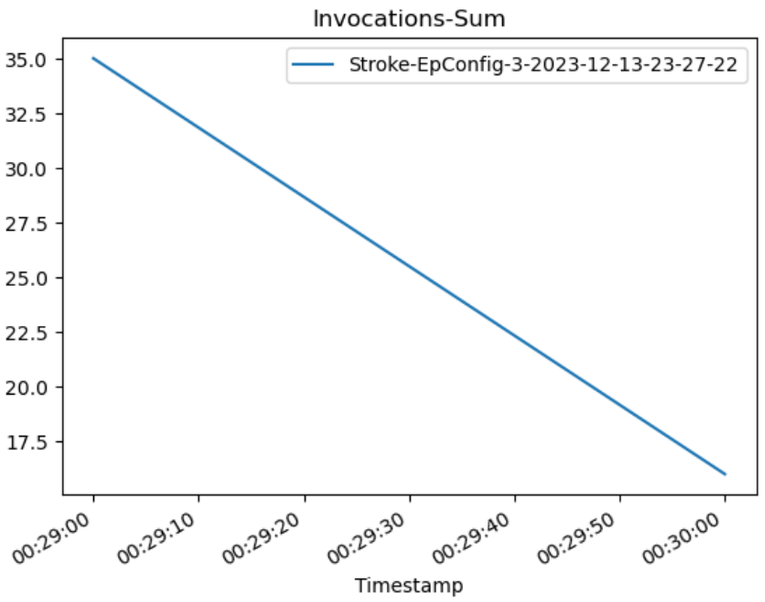
You can expect to see that, as the new endpoint config-3 (correct model version) starts getting deployed, it takes over endpoint config-2 (incompatible due to model version) without any errors. This can be seen in the graphs below as the Invocation5XXErrors and ModelLatency decreases during this transition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

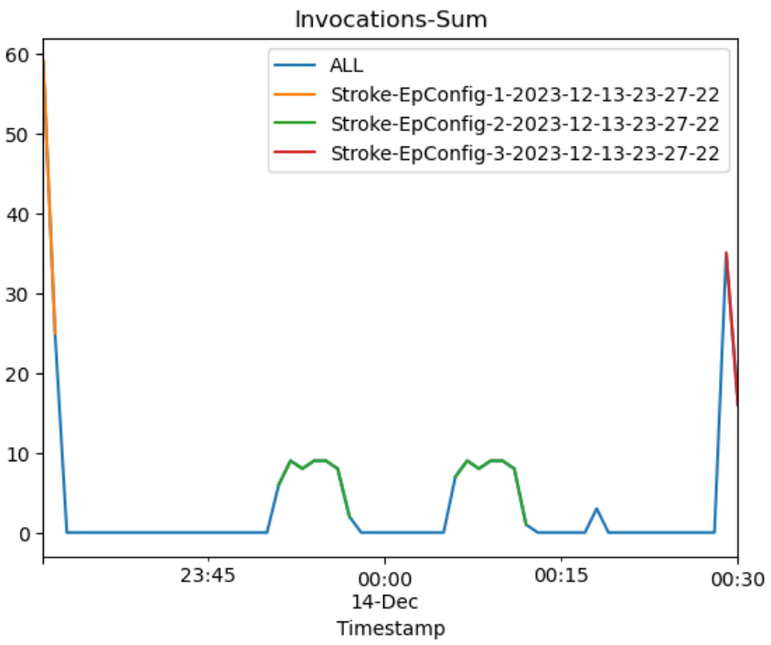
In [33]:
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name, None, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_1 = plot_endpoint_invocation_metrics(
    endpoint_name, ep_config_name, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_2 = plot_endpoint_invocation_metrics(
    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"
)
```

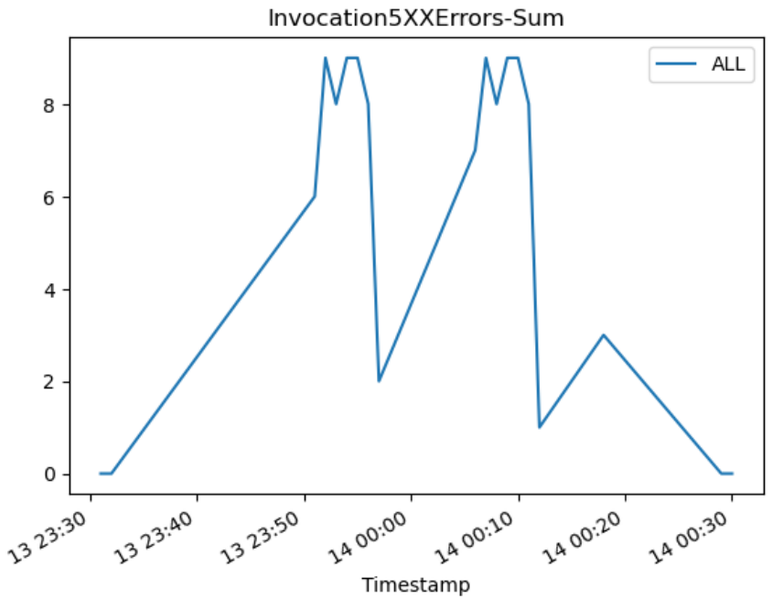


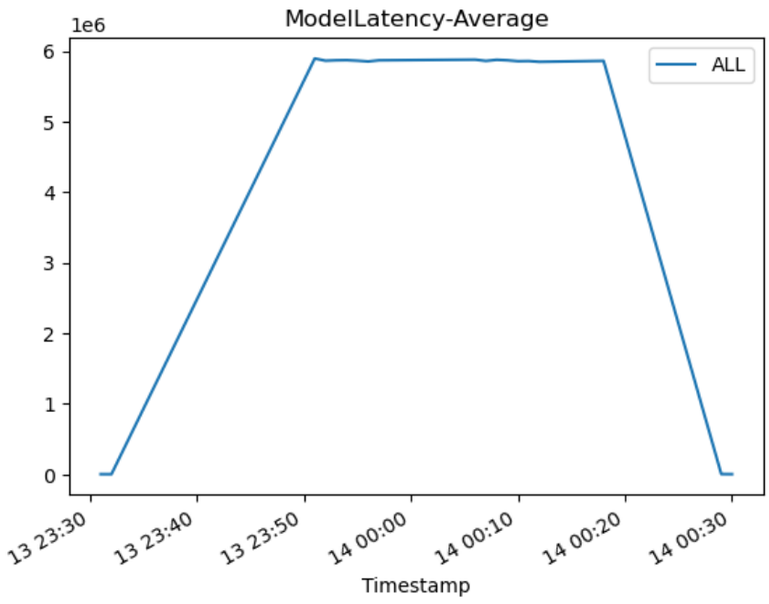



Invocations-Sum
Stroke-EpConfig-2-2023-12-13-23-27-22
13 23:50
13 23:55
14 00:00
14 00:05
14 00:10
14 00:15
Timestamp


Invocations-Sum
Stroke-EpConfig-3-2023-12-13-23-27-22
00:29:00
00:29:10
00:29:20
00:29:30
00:29:40
00:29:50
00:30:00
Timestamp


Invocations-Sum
ALL
Stroke-EpConfig-1-2023-12-13-23-27-22
Stroke-EpConfig-2-2023-12-13-23-27-22
Stroke-EpConfig-3-2023-12-13-23-27-22
23:45
00:00
14-Dec
00:15
00:30
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.

In [34]:
```
sm.delete_endpoint(EndpointName=endpoint_name)
```

Out[34]:
```
{'ResponseMetadata': {'RequestId': '8e0279e1-3af5-4c02-aa14-6a878fabb1d4',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '8e0279e1-3af5-4c02-aa14-6a878fabb1d4',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '0',
   'date': 'Thu, 14 Dec 2023 00:32:40 GMT'},
  'RetryAttempts': 0}}
```

In [ ]:
```
sm.delete_endpoint_config(EndpointConfigName=ep_config_name)
sm.delete_endpoint_config(EndpointConfigName=ep_config_name2)
sm.delete_endpoint_config(EndpointConfigName=ep_config_name3)
```

In [ ]:
```
sm.delete_model(ModelName=model_name)
sm.delete_model(ModelName=model_name2)
sm.delete_model(ModelName=model_name3)
```

In [35]:
```
cw.delete_alarms(AlarmNames=[error_alarm, latency_alarm])
```

Out[35]:
```
{'ResponseMetadata': {'RequestId': '3de67ff0-eb7e-4bc1-8891-8188592f7f68',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '3de67ff0-eb7e-4bc1-8891-8188592f7f68',
   'content-type': 'text/xml',
   'content-length': '210',
   'date': 'Thu, 14 Dec 2023 00:33:44 GMT'},
  'RetryAttempts': 0}}
```

Creating configuration for Shadow Endpoint

In [38]:
```
ep_config_name_shadow = f"Stroke-Shadow-EpConfig-1-{datetime.now():%Y-%m-%d-%H-%M-%S}"

print(f"Endpoint Config Shadow: {ep_config_name_shadow}")

resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name_shadow,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name3,
            "InstanceType": "ml.m5.xlarge",
            "InitialInstanceCount": 3,
        }
    ],
    # Type: Array of ShadowProductionVariants
     ShadowProductionVariants = [
         {
             "ModelName": model_name,
             'VariantName':"shadow",
             "InitialInstanceCount": 1,
             "InitialVariantWeight": 0.5,
             "InstanceType": "ml.m5.xlarge"
         }
     ],
    DataCaptureConfig = {
         'EnableCapture': True,
         'InitialSamplingPercentage': 100,
         'DestinationS3Uri': f"s3://{bucket}/{prefix}",
         'CaptureOptions': [{'CaptureMode': 'Input'}, {'CaptureMode': 'Output'}],
         'CaptureContentTypeHeader': {'JsonContentTypes': ['application/json']}
        }

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

```
Endpoint Config Shadow: Stroke-Shadow-EpConfig-1-2023-12-14-00-50-09
Created Endpoint Config: {'EndpointConfigArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint-config/stro
ke-shadow-epconfig-1-2023-12-14-00-50-09', 'ResponseMetadata': {'RequestId': '6ad19700-75e4-4f1e-b012-8ad51b7
2cdbc', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '6ad19700-75e4-4f1e-b012-8ad51b72cdbc', 'c
ontent-type': 'application/x-amz-json-1.1', 'content-length': '125', 'date': 'Thu, 14 Dec 2023 00:50:09 GM
T'}, 'RetryAttempts': 0}}
```

Creating an endpoint from the shadow configurations.

In [39]:
```
endpoint_name_shadow = f"Stroke-Shadow-Deployment-Guardrails-Canary-{datetime.now():%Y-%m-%d-%H-%M-%S}"
print(f"Endpoint Name: {endpoint_name_shadow}")

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

```
Endpoint Name: Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44

Created Endpoint: {'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-shadow-deployment
-guardrails-canary-2023-12-14-00-51-44', 'ResponseMetadata': {'RequestId': '8ff7b525-5a9a-4168-bb23-97792042e
482', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '8ff7b525-5a9a-4168-bb23-97792042e482', 'con
tent-type': 'application/x-amz-json-1.1', 'content-length': '130', 'date': 'Thu, 14 Dec 2023 00:51:45 GMT'},
'RetryAttempts': 0}}
```

Check the status of the endpoint!

In [40]:
```
def wait_for_endpoint_in_service(endpoint_name_shadow):
    print("Waiting for endpoint in service")
    while True:
        details = sm.describe_endpoint(EndpointName=endpoint_name_shadow)
        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_shadow)

sm.describe_endpoint(EndpointName=endpoint_name_shadow)
```

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

Out[40]:
```
{'EndpointName': 'Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-shadow-deployment-guardrails-canary
-2023-12-14-00-51-44',
 'EndpointConfigName': 'Stroke-Shadow-EpConfig-1-2023-12-14-00-50-09',
 '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@sha256:4814427c3e0a6cf9
9e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 0, 51, 45, 845000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'DataCaptureConfig': {'EnableCapture': True,
  'CaptureStatus': 'Started',
  'CurrentSamplingPercentage': 100,
  'DestinationS3Uri': 's3://stroke-predection-project/sagemaker/DEMO-Deployment-Guardrails-Canary'},
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 14, 0, 51, 45, 242000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 14, 0, 54, 5, 101000, tzinfo=tzlocal()),
 'ShadowProductionVariants': [{'VariantName': 'shadow',
   '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@sha256:4814427c3e0a6cf9
9e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 0, 51, 45, 921000, tzinfo=tzlocal())}],
   'CurrentWeight': 0.5,
   'DesiredWeight': 0.5,
   'CurrentInstanceCount': 1,
   'DesiredInstanceCount': 1}],
 'ResponseMetadata': {'RequestId': '57214aa7-0f6c-4459-8bf2-d8d6fa71f9f0',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '57214aa7-0f6c-4459-8bf2-d8d6fa71f9f0',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1468',
   'date': 'Thu, 14 Dec 2023 00:54:08 GMT'},
  'RetryAttempts': 0}}
```

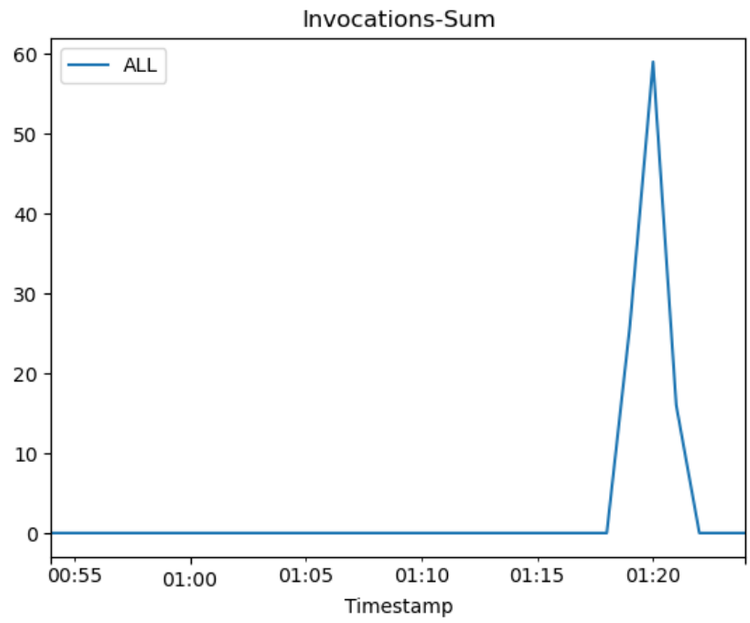
Invoke Endpoint on the shadow instance

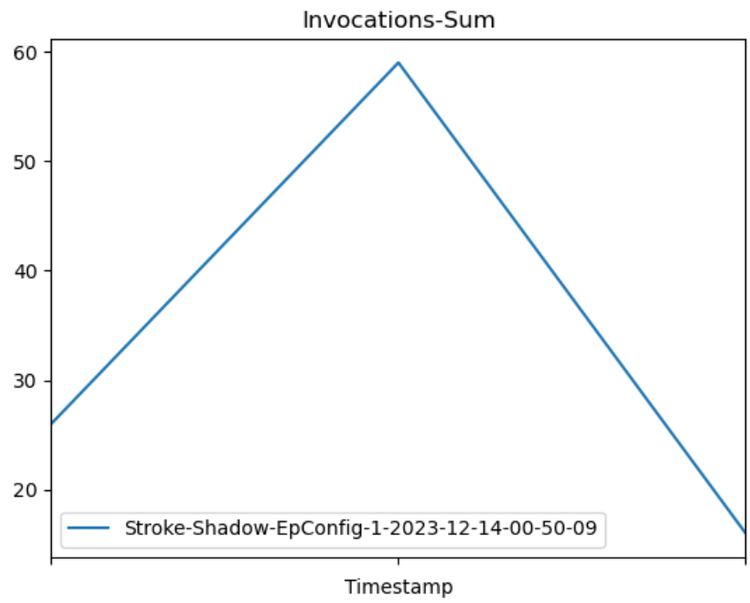
In [42]:
```
invoke_endpoint(endpoint_name_shadow, max_invocations=100)
```

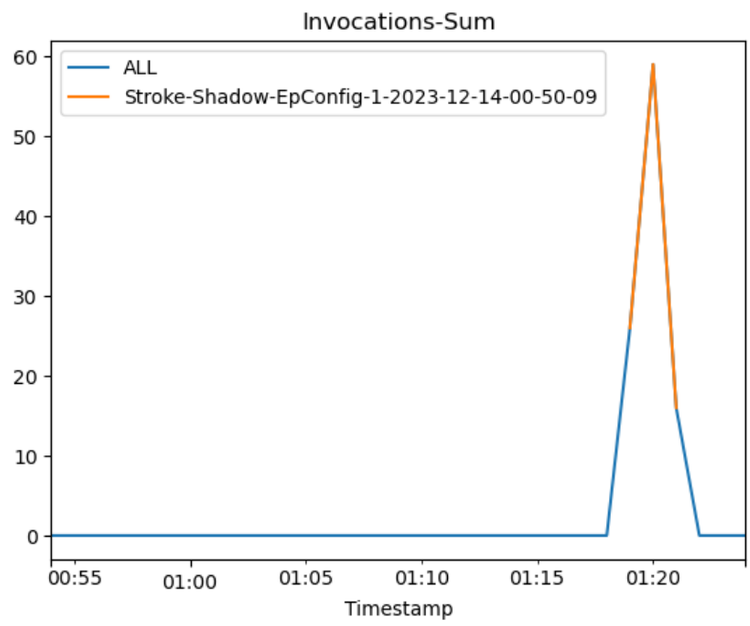
```
Sending test traffic to the endpoint Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44.
Please wait...
.....................................................................................................
Done!
```

In [43]:
```
invocation_metrics = plot_endpoint_invocation_metrics(
    endpoint_name_shadow, None, "AllTraffic", "Invocations", "Sum"
)
metrics_epc_1 = plot_endpoint_invocation_metrics(
    endpoint_name_shadow, ep_config_name_shadow, "AllTraffic", "Invocations", "Sum"
)

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

Out[43]: <Axes: title={'center': 'Invocations-Sum'}, xlabel='Timestamp'>







Update from the endpoint from production variant to the shadow variant

In [ ]:

In [55]:
```
# Specify the endpoint name and the variant names for the production and shadow variants
endpoint_name = endpoint_name_shadow
production_variant_name = "AllTraffic"
shadow_variant_name = "shadow"

# Get the current endpoint configuration to obtain the existing variant weights
response = sm.describe_endpoint(EndpointName=endpoint_name_shadow)
current_config = response["EndpointConfigName"]

# Get the current production variant weight
current_production_weight = next(
    (variant["CurrentWeight"] for variant in response["ProductionVariants"] if variant["VariantName"] == produ
    0.0  # Default weight if not found
)

# Get the current shadow variant weight
current_shadow_weight = next(
    (variant["CurrentWeight"] for variant in response["ProductionVariants"] if variant["VariantName"] == shado
    0.0  # Default weight if not found
)

# Update the weights to replace the shadow variant with the production variant
sm.update_endpoint_weights_and_capacities(
    EndpointName=endpoint_name,
    DesiredWeightsAndCapacities=[
        {"VariantName": production_variant_name, "DesiredWeight": 1.0},
        {"VariantName": shadow_variant_name, "DesiredWeight": 0.0},
    ],
)

print(f"Updated endpoint {endpoint_name} weights to replace shadow with production variant.")
```

```
Updated endpoint Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44 weights to replace shadow wit
h production variant.
```

In [57]:
```
sm.describe_endpoint(EndpointName=endpoint_name_shadow)
```

Out[57]:
```
{'EndpointName': 'Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44',
 'EndpointArn': 'arn:aws:sagemaker:us-east-1:116732205680:endpoint/stroke-shadow-deployment-guardrails-canary
-2023-12-14-00-51-44',
 'EndpointConfigName': 'Stroke-Shadow-EpConfig-1-2023-12-14-00-50-09',
 '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@sha256:4814427c3e0a6cf9
9e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 2, 5, 33, 63000, tzinfo=tzlocal())}],
   'CurrentWeight': 1.0,
   'DesiredWeight': 1.0,
   'CurrentInstanceCount': 3,
   'DesiredInstanceCount': 3}],
 'DataCaptureConfig': {'EnableCapture': True,
  'CaptureStatus': 'Started',
  'CurrentSamplingPercentage': 100,
  'DestinationS3Uri': 's3://stroke-predection-project/sagemaker/DEMO-Deployment-Guardrails-Canary'},
 'EndpointStatus': 'InService',
 'CreationTime': datetime.datetime(2023, 12, 14, 0, 51, 45, 242000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2023, 12, 14, 2, 6, 49, 240000, tzinfo=tzlocal()),
 'ShadowProductionVariants': [{'VariantName': 'shadow',
   '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@sha256:4814427c3e0a6cf9
9e637704da3ada04219ac7cd5727ff62284153761d36d7d3',
     'ResolutionTime': datetime.datetime(2023, 12, 14, 2, 5, 33, 136000, tzinfo=tzlocal())}],
   'CurrentWeight': 0.0,
   'DesiredWeight': 0.0,
   'CurrentInstanceCount': 1,
   'DesiredInstanceCount': 1}],
 'ResponseMetadata': {'RequestId': '6596b0c0-0f5c-49cb-a48c-47470cdb6163',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '6596b0c0-0f5c-49cb-a48c-47470cdb6163',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '1467',
   'date': 'Thu, 14 Dec 2023 02:08:03 GMT'},
  'RetryAttempts': 0}}
```

In [58]:
```
invoke_endpoint(endpoint_name_shadow, max_invocations=100)
```

```
Sending test traffic to the endpoint Stroke-Shadow-Deployment-Guardrails-Canary-2023-12-14-00-51-44.
Please wait...
.....................................................................................................
Done!
```

In [ ]:

In [60]:
```
sm.delete_endpoint(EndpointName=endpoint_name_shadow)
```

Out[60]:
```
{'ResponseMetadata': {'RequestId': '79ddf9ac-c211-4d88-a5fc-045508dc1623',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '79ddf9ac-c211-4d88-a5fc-045508dc1623',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '0',
   'date': 'Thu, 14 Dec 2023 02:16:13 GMT'},
  'RetryAttempts': 0}}
```

In [ ]:
```

```