Final exam

AWS Academy Virginia

Tasks

- In the exam you will use SageMaker Deployment Guardrail and Shadow Testing
- To perform this task, you will need to create two models first.
- There are a few rules to follow to solve this exam question. They are listed later. Read them carefully.

Deployment Guardrail

- Deployment guardrails are a set of model deployment options in Amazon SageMaker Inference to update your machine learning models in production.
- Through Gaurdrail 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 automatically take corrective action before they significantly impact production.

Different guardrails

- **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**: let 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.**

Notebook (1)

- Run the **Inference endpoint using the traffic shifting** notebook.
- That notebook simulates a successful deployment and a failure in an update that leads to a rollback and then a successful rollout.
- Two model artifacts will be downloaded as part of executing the notebook.
- Those two model artifacts will be uploaded to S3
- You will create three models from those two model artifacts:
  - Model Name 1: DEMO-xgb-churn-pred
  - Model Name 2: DEMO-xgb-churn-pred2
  - Model Name 3: DEMO-xgb-churn-pred3

- Model 2 in the above list is incompatible and we use it to simulate an error. It has an **incompatible algorithm version.** We deploy it as a Canary fleet, sending a small percentage of the traffic, and it will result in errors, which will be used to trigger a rollback using pre-specified CloudWatch alarms.

The image URIs we use in this test

- We use three image URIs
- The URI2 (version 1.2-1) is incompatible

```
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}")
```

Notebook (2)

- You will create one endpoint configuration for each of the models:
  - Endpoint Config 1: DEMO-EpConfig-1
  - Endpoint Config 2: DEMO-EpConfig-2
  - Endpoint Config 3: DEMO-EpConfig-3
- Each config has a different variant:

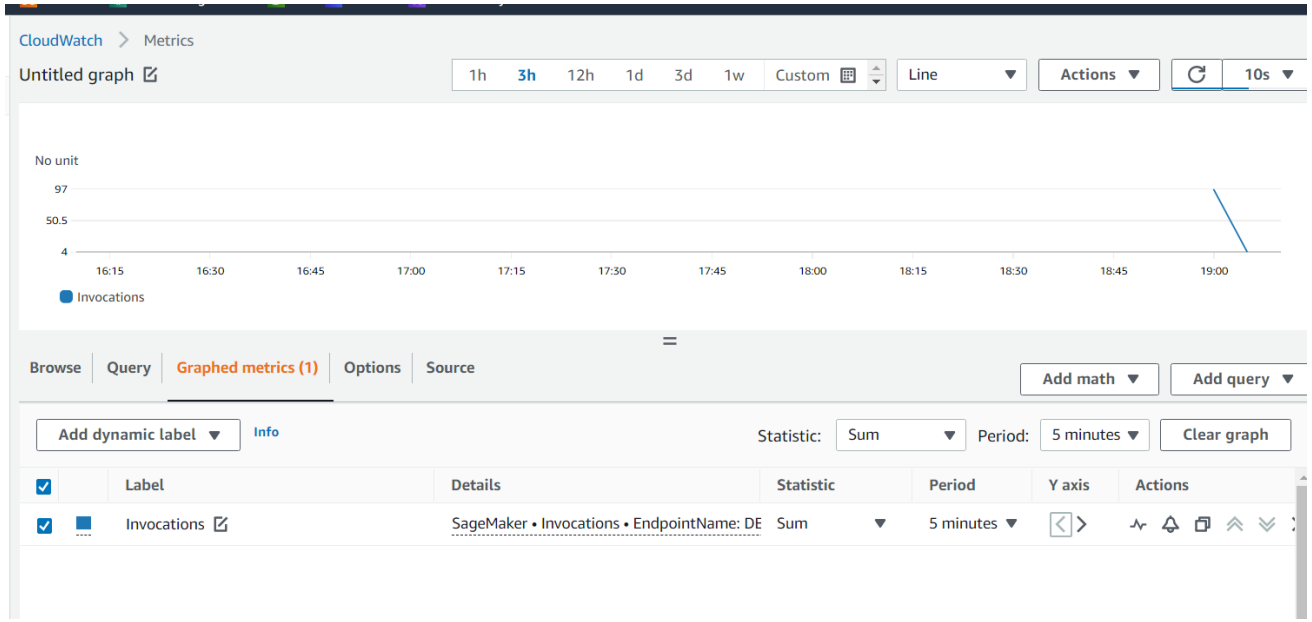
```
resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name,
            "InstanceType": "ml.m5.xlarge",
            "InitialInstanceCount": 3,
        }
    ],
)
print(f"Created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

```
resp = sm.create_endpoint_config(
    EndpointConfigName=ep_config_name3,
    ProductionVariants=[
        {
            "VariantName": "AllTraffic",
            "ModelName": model_name3,
            "InstanceType": "ml.m5.xlarge",
            "InitialInstanceCount": 3,
        }
    ],
```

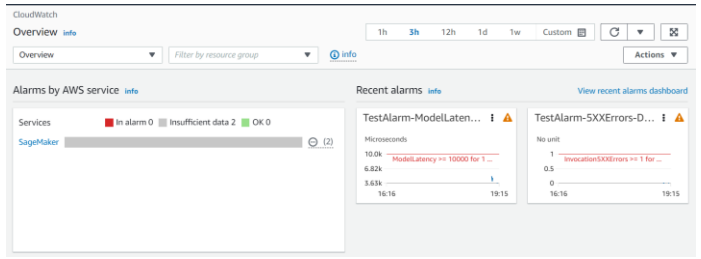
Create endpoint and check the metrics

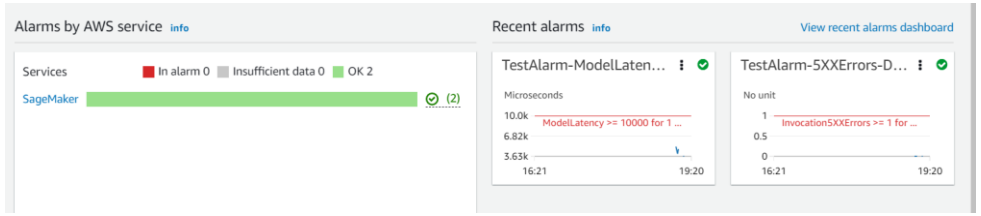
- You start with one **endpoint** first, using Endpoint Config 1: **DEMO-EpConfig-1**
- You will test that endpoint by invoking it by sending the test date set
- You will get some metrics. See those metrics also in the CloudWatch



Creating Alarms

- At this point if you go to CloudWatch you do not have an alarm
- Then You create **two** alarms in CloudWatch to activate them when you get Invocation5XXErrors and ModelLatency
- After you create the alarms, check the CloudWatch console to see them
- At the beginning, there is insufficient data. Wait until everything you see green line





Notebook (3)

- Then we update the endpoint to use a new **endpoint configuration** and as part of that, we also pass "BlueGreenUpdatePolicy“ to that endpoint.

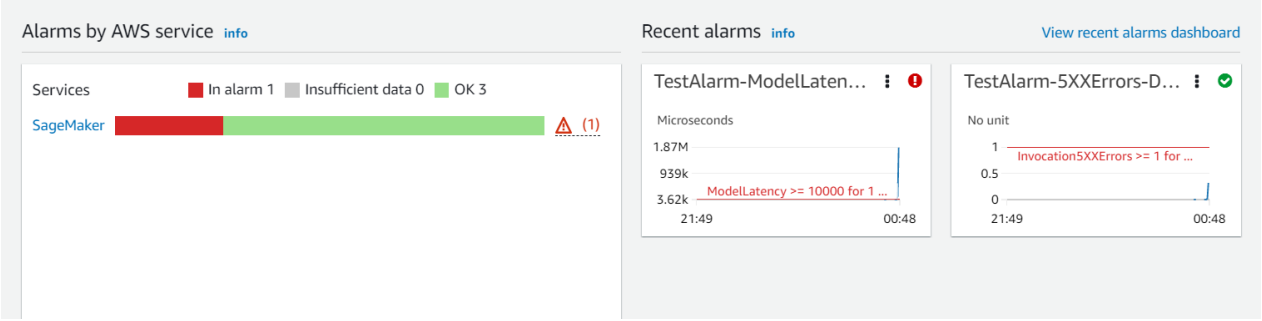
```
# update endpoint request with new DeploymentConfig parameter
sm.update_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=ep_config_name2,
    DeploymentConfig=canary_deployment_config,
)
```

- While the endpoint is updating, we start invoking the endpoint

DEMO-Deployment-Guardrails-Canary-2023-03-30-18-59-55	arn:aws:sagemaker:us-east-1:566462208046:endpoint/demo-deployment-guardrails-canary-2023-03-30-18-59-55	3/30/2023, 11:59:55 AM	Updating	3 1

- You will see that alarms will be triggered

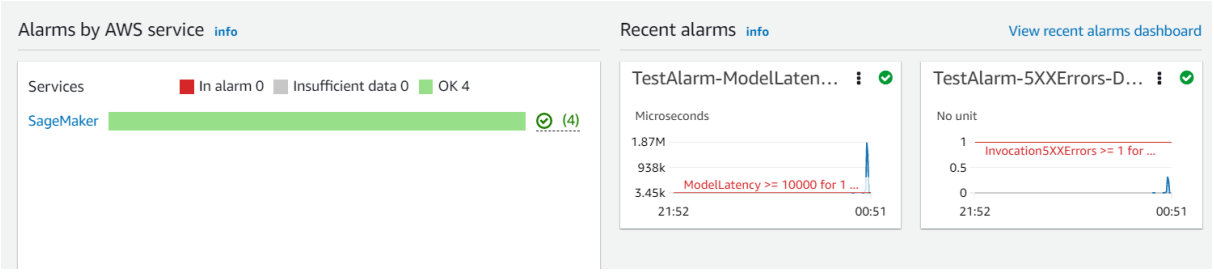
```
Sending test traffic to the endpoint DEMO-Deployment-Guardrails-Canary-2023-03-30-18-59-55.
Please wait...
.....................................................................................................
.....................................E..E.E.EE...EE.E.EE....E...
```



Notebook (4)

- And after that because of "AutoRollbackConfiguration“ in the canary_deployment_config, it rolls back automatically

```
        "MaximumExecutionTimeoutInSeconds": 1800,  # maximum timeout for deployment
    },
    "AutoRollbackConfiguration": {
        "Alarms": [{"AlarmName": error_alarm}, {"AlarmName": latency_alarm}],
    },
}
```



- Read more about that Alarm in this link: https://docs.aws.amazon.com/sagemaker/latest/dg/deployment-guardrails-configuration.html

Notebook (5)

- That was a simulation of failure scenarios in updates. Now we can test a successful deployment.
- This time we update the endpoint with endpoint configuration 3.
- This time we do not see any error and Alarms will not be triggered and the roll out is successful
- While this is running you can start working in shadow testing (next part of this exam)
- You need those **models** in the Shadow testing (next part). Do not remove those until you finish the shadow test
- Delete only the **endpoint (not models)**
- **NOTE:** The last cell in the notebook will be used later for shadow testing

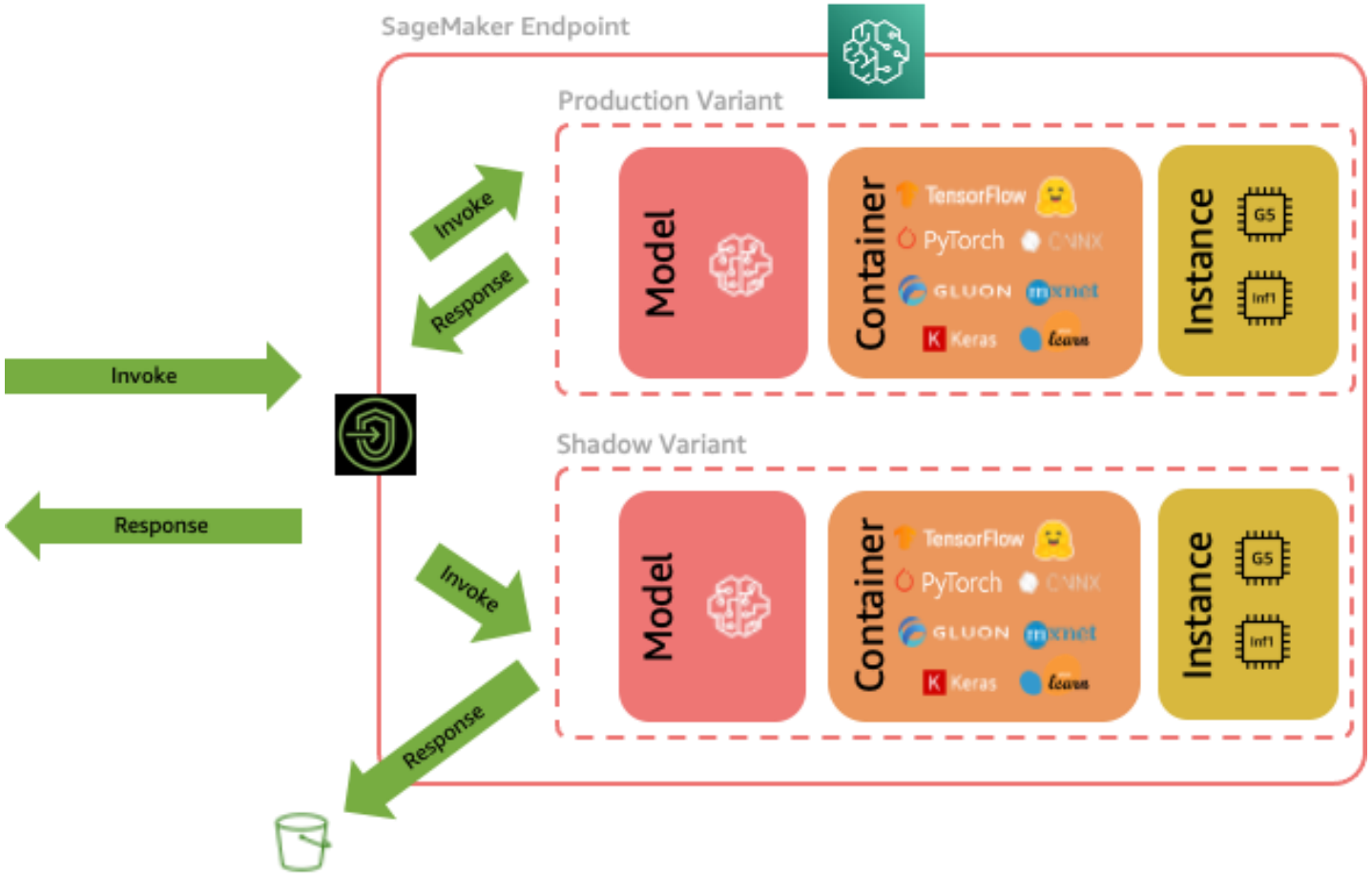
SageMaker shadow testing

- Amazon SageMaker now allows you to compare the performance of a new version of a model serving stack with the currently deployed version prior to a full production rollout using a deployment safety practice known as **shadow testing**.
- Shadow testing can help you identify potential configuration errors and performance issues before they impact end-users.
- SageMaker takes care of deploying the **new version** alongside the current version serving production requests, routing a portion of requests to the shadow version.

SageMaker shadow testing benefits

- You can use shadow testing to validate changes to any component to your production variant, namely the **model**, the **container**, or the **instance**, without any end user impact.
- You are considering promoting a new model that has been validated **offline** to production but want to evaluate operational performance metrics such as **latency, error rate** before making this decision
- You are considering changes to your **serving infrastructure container**, such as patching vulnerabilities or upgrading to newer versions, and want to assess the impact of these changes prior to promotion
- You are considering changing your **ML instance** and want to evaluate how the new instance would perform with live inference requests.

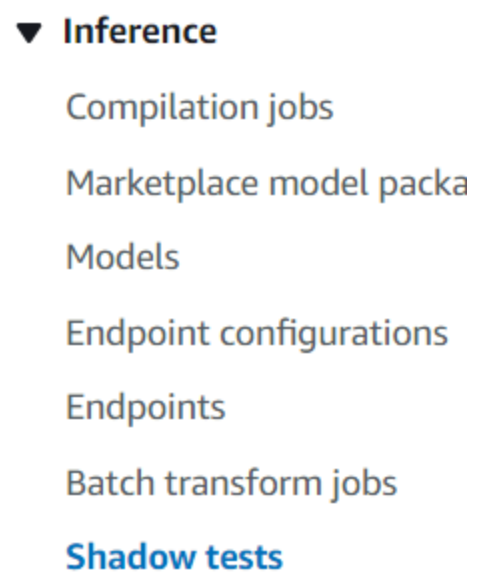
Production Variant vs Shadow Variant

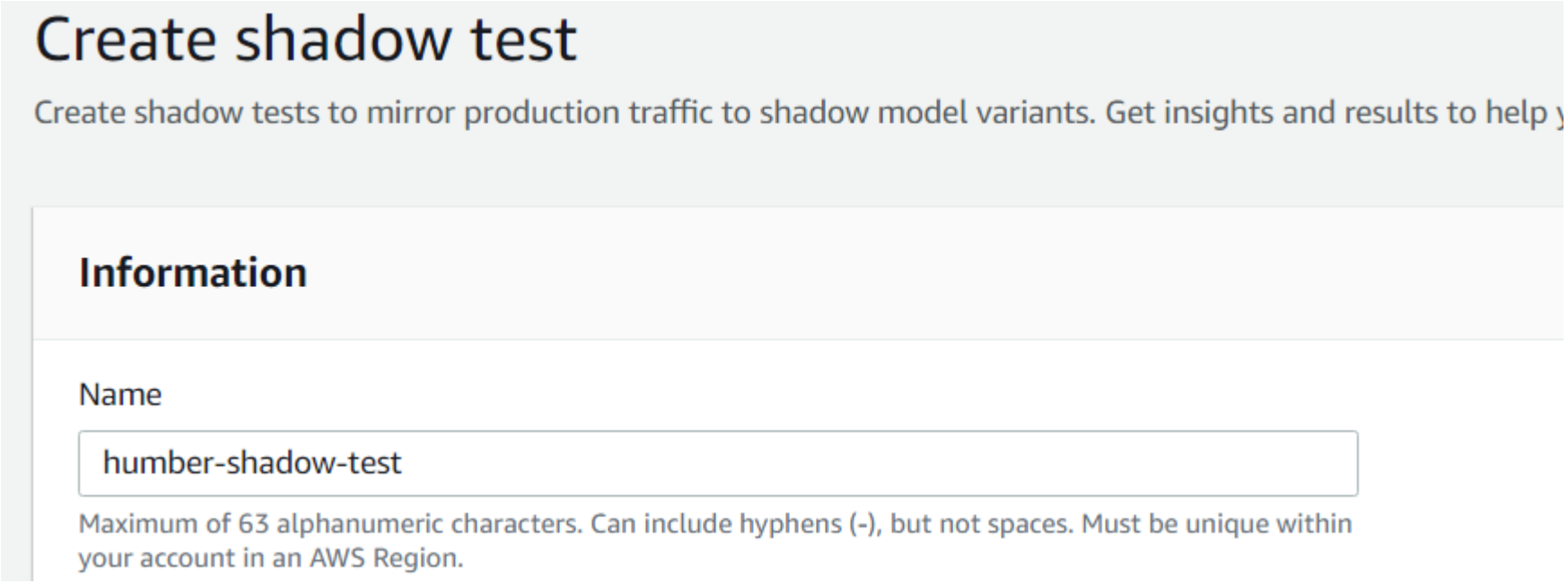


Creating shadow test

- In the Inference select: Shadow tests

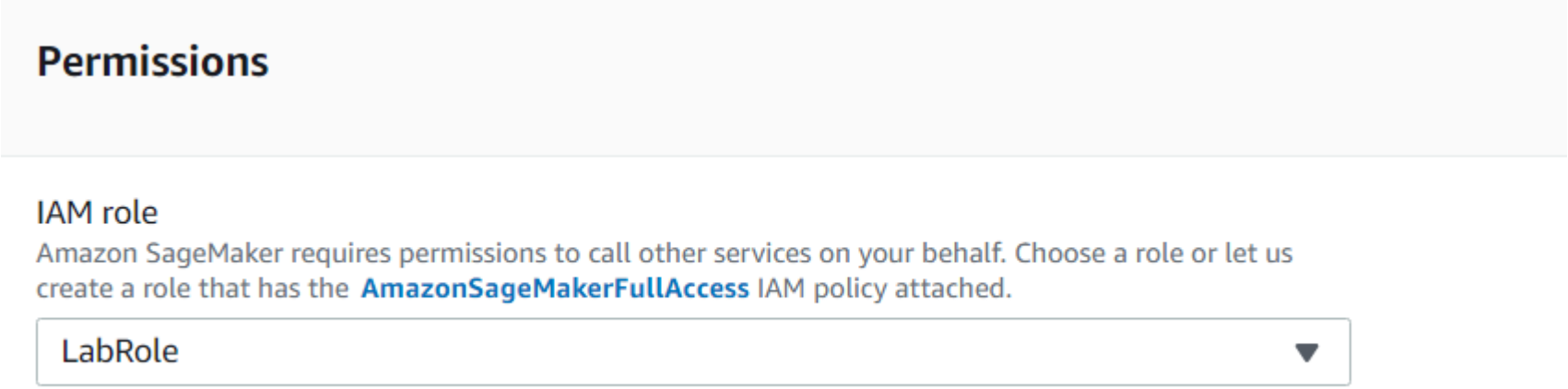
- Create a new test





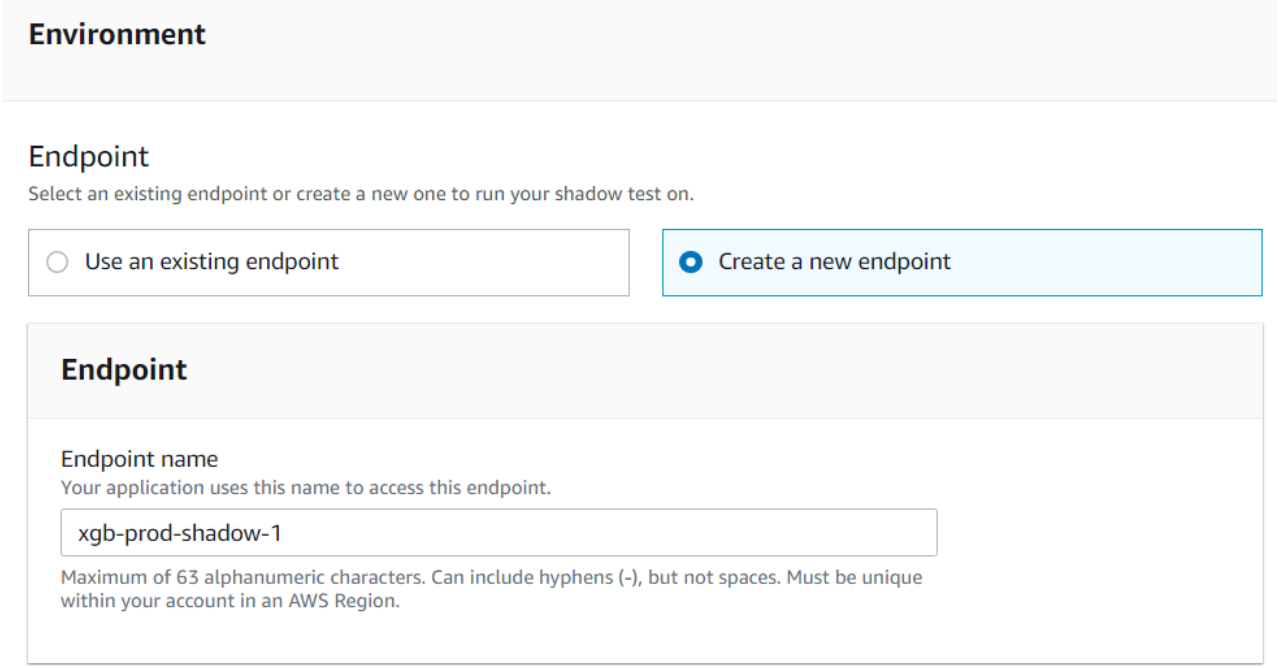
Configuring Shadow test

- IAM role: LabRole



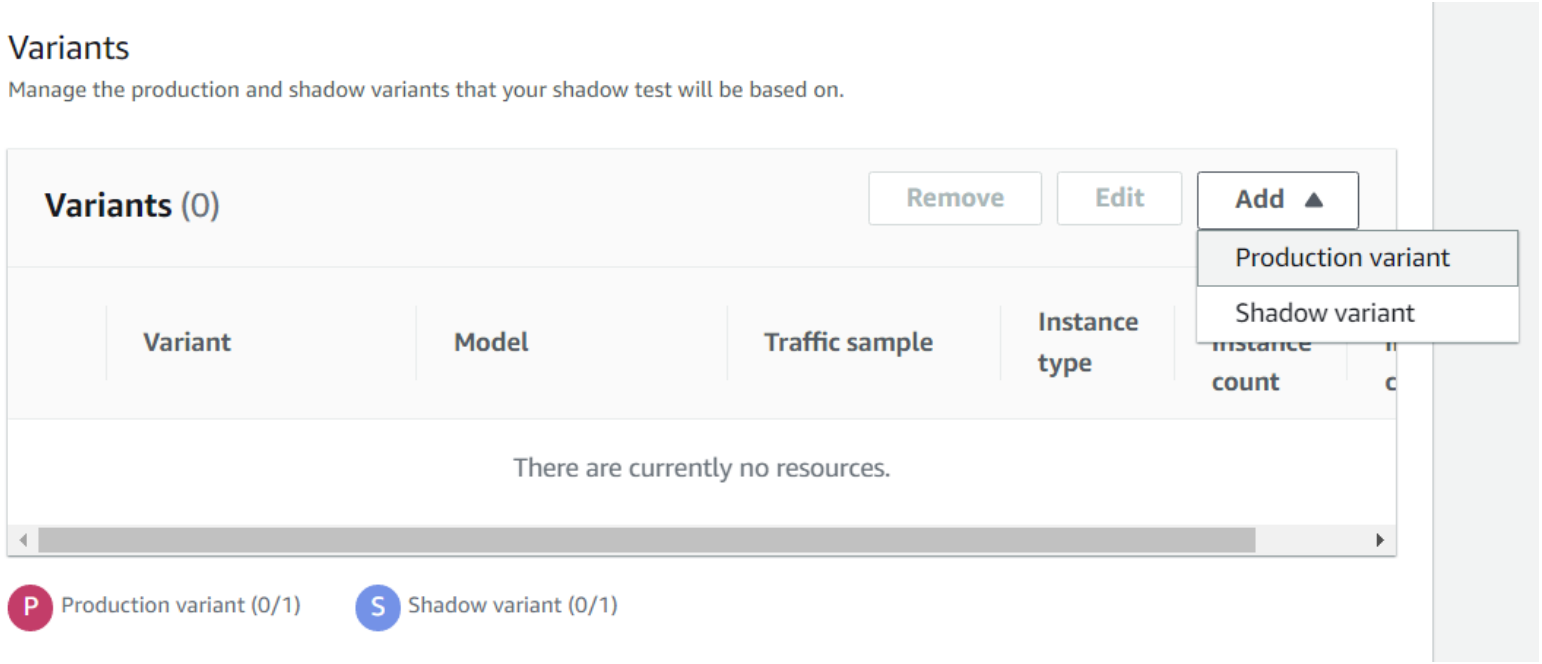
- Create a new endpoint name

(make sure no other endpoint exists. AWS Academy allows you to have only one endpoint)



Add Variants

- You can add one production and one shadow variant associated with this endpoint by clicking on '***Add***' in the Variants section.



Creating the Variants

- Select **pred** model for production and **pred2** for shadow
- Optionally, you can change the instance type and count associated with each variant.

Variants

Manage the production and shadow variants that your shadow test will be based on.

Variants (2)

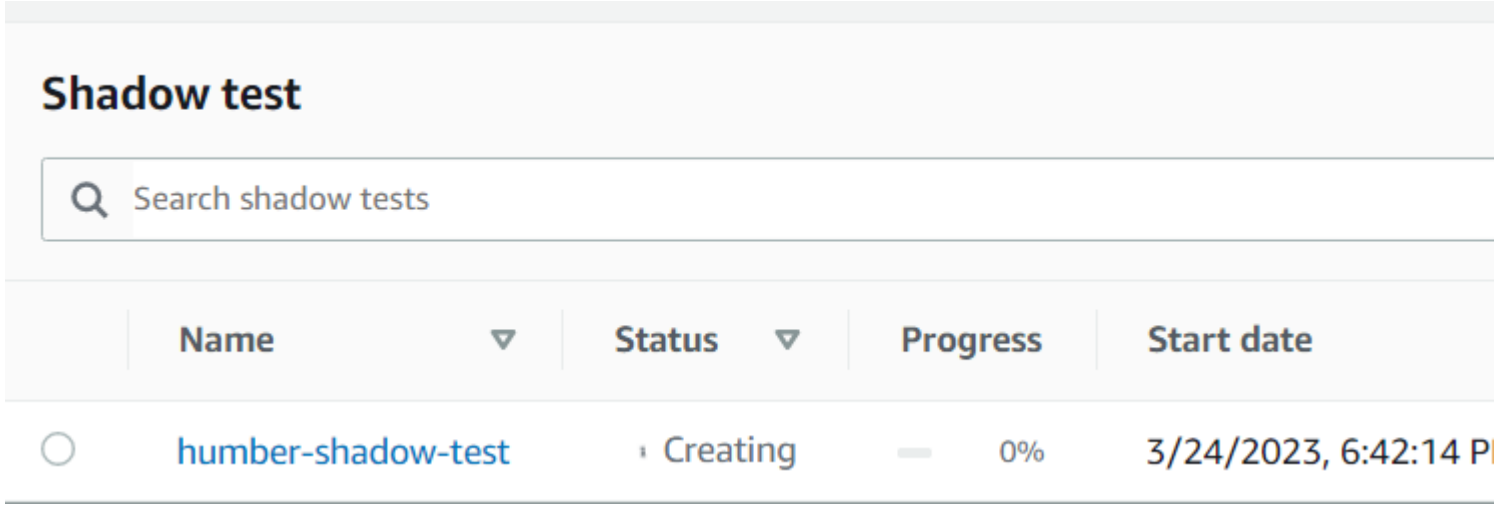
Remove | Edit | Add

	Variant	Model	Traffic sample	Instance type	Initial instance count
○	P Production-01	DEMO-xgb-churn-pred-2023-03-24-10-58-38	100%	ml.m5.xlarge	1
○	S Shadow-01	DEMO-xgb-churn-pred2-2023-03-24-10-58-38	100%	ml.m5.xlarge	1

P Production variant (1/1) S Shadow variant (1/1)

Create Shadow test

- You do not need to change the schedule, leave it as-is
- You can control the duration of the test from one hour to 30 days. If unspecified, it defaults to 7 days. After this period, the test is marked complete. If you are running a test on an existing endpoint, it will be rolled back to the state prior to starting the test upon completion.
- Click on **create shadow test**



After creation

- New endpoint is created

Amazon SageMaker > Endpoints

Endpoints

Update endpoint | Actions | Create endpoint

Search endpoints

1

Name	ARN	Creation time	Status	Last updated
xgb-prod-shadow1	arn:aws:sagemaker:us-east-1:906486081684:endpoint/xgb-prod-shadow1	3/24/2023, 6:44:22 PM	Creating	3/24/2023, 6:44:22 PM

- It has two variants

Endpoint runtime settings

Update weights | Update instance count | Configure auto scaling

	Variant name	Current weight	Desired weight	Elastic Inference	Instance type	Current instance count	Desired instance count	Instance min - max	Automatic scaling
P	Production-01	100	100	-	ml.m5.xlarge	1	1	-	No
S	Shadow-01	100	100	-	ml.m5.xlarge	1	1	-	No

Invoke endpoint

- Now adjust the last cell of the notebook I have given you. You need to change the very end of the "**Inference endpoint using-canary traffic shifting"** notebook. Change the endpoint name accordingly and **run** that cell.
- Now come back to shadow tests and see different metrics

Observe different metrics

Select metric

View the selected metric summary and statistics from the start of experiment to present.

ModelLatency

A lower value of the latency metric usually indicates a faster model. For more information about the metric, please visit Monitor Amazon SageMaker with Amazon CloudWatch.

Variant name	Sample count	Average (Microseconds)	Maximum (Microseconds)
P Production-01	352	3181.18	13090.00
S Shadow-01	237	4079087.65 +128125.54%	21006543.00 +160377.79%

Select metric

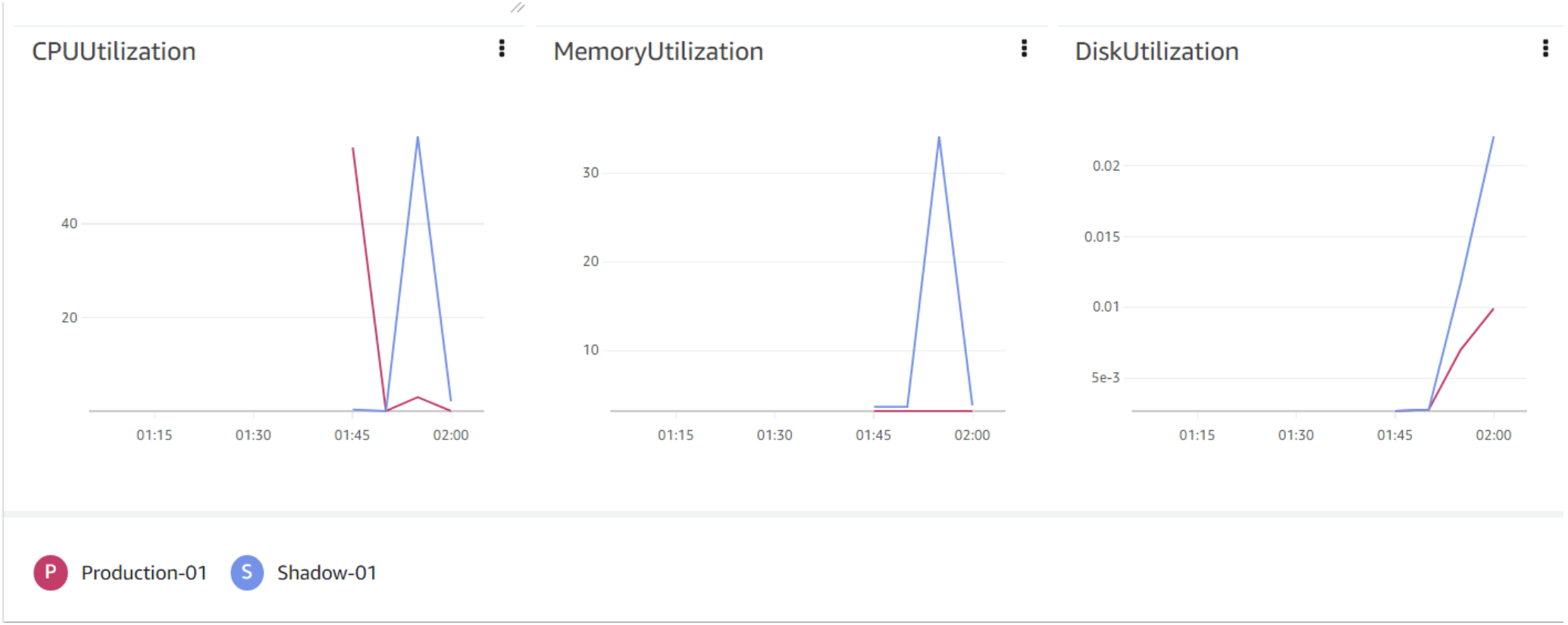
View the selected metric summary and statistics from the start of experiment to present.

Invocation5XXErrors

A lower value of the error metric usually indicates a more stable model. For more information about the metric, please visit Monitor Amazon SageMaker with Amazon CloudWatch.

Variant name	Sample count	Average (Percent)	Sum (Count)
P Production-01	3113	0	0
S Shadow-01	3019	0.96 +Infinity%	2885 +Infinity%

Analyze the metrics



Mark the test as complete

- Mark the Shadow test as complete and choose what you want to do with shadow variant

Select No, ...

Important: Remove the end point and delete Shadow test

humber-shadow-test

Overview | Settings | Details

Summary

Status	Progress	Type
Stopping	Mar 25, 2023 01:47 UTC - Apr 01, 2023 01:42 UTC 100%	Shadow mode

Reason
-

Mark shadow test as complete

Select the action to be performed on the shadow variant once the shadow test is marked as complete. A shadow test that is marked complete cannot be resumed and this action cannot be undone.

Deploy shadow variant to production endpoint?

- Yes, deploy shadow variant
  The current production variant will be removed from the endpoint configuration. The shadow variant will become the new production variant and will begin to receive 100% of the invocation traffic. You will be taken to the deployment configuration and the shadow test will be marked as complete.
- No, remove shadow variant
  The shadow variant will be removed from the endpoint configuration. The production variant will remain the same, and continue to receive 100% of innovation traffic.

Comment - *Optional*
Leave a comment or reason why you are completing the shadow test.

Enter text

Cancel | Confirm

Rules and Conditions (1)

- Use the rules we set for the project. Use the same data set you used and the models you created.

Rules and Conditions (2)

- You will simulate a deployment to production through **SageMaker guardrail** and SageMaker **Shadow testing**
- You will start from a data set and prepare the data for your project algorithm **(2 marks).** You should show me the data set and algorithm you have used.
- You train **two** models. Both model artifacts have to be prepared in notebook. You can use SageMaker Hyperparameter tuning job to create a set of models and you pick the best two (Do not use web console). Make sure one model has a better metric than the other one **(4 marks)**
- Use the notebook I gave as a starting point for simulating a guardrail. You are expected to use parts of that code in your newly created notebook.
- The first guardrail is a **failed** deployment that the guardrail rollbacks automatically **(5 marks).** You must show me the "E" letters in the notebook.
- The second guardrail is a successful deployment, and you will show that the guardrail allows it to be deployed successfully **(5 marks)**

Rules and Conditions (3)

- I showed you how to use the Shadow test in the console. You can do the same through code. Here is the link: https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-shadow-variant/Shadow_variant.html **(5 marks)**
- You will use the same models you have created in the guardrail. You will design the test in a way that the shadow variant looks better than production variant and you allow that to replace the production variant. **(4 marks)**
- The