Preparing data and training at scale

Generative AI Foundations on AWS

Emily Webber, Principal ML Specialist SA at AWS

Lesson 5 – Level 400

Today’s activities

- Preparing data at scale on AWS
- Making CPUs work for you
- Hydrating your training loop
- Distributed file systems
- Warm pools and troubleshooting
- Hands-on walk through: SageMaker warm pools with FSx for Lustre

Reminder – everything we discuss today
is possible on AWS and SageMaker!

Ideally data preparation should be ….

Low cost

Simple

Scalable

Reliable

Usable

Easy to manage

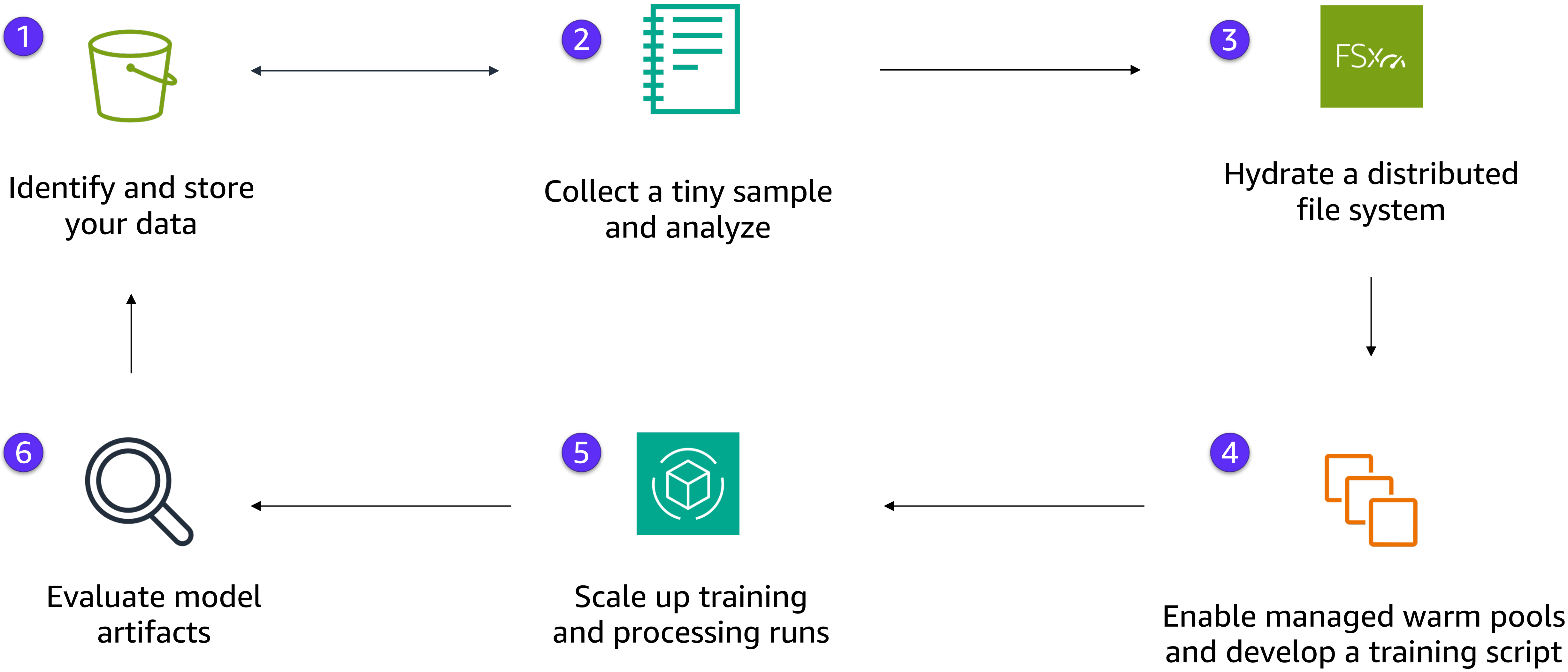
Today we'll focus on these

- Amazon SageMaker
- Amazon FSx for Lustre
- Amazon Simple Storage Service (Amazon S3)
- Amazon Elastic Container Registry (Amazon ECR)

Many options for data preparation on AWS

- Amazon EMR
- AWS Glue
- Amazon Redshift
- Amazon QuickSight
- Amazon Athena
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon DynamoDB
- Amazon Aurora

Steps to prepare and train at scale on AWS



1 Identify and store your data

2 Collect a tiny sample and analyze

3 Hydrate a distributed file system

4 Enable managed warm pools and develop a training script

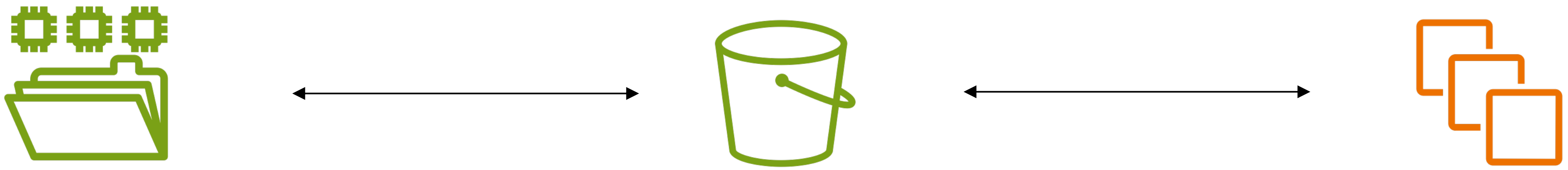
5 Scale up training and processing runs

6 Evaluate model artifacts

1. Identify and store your data

Buckets are great!

- Can upload with a button, SDK, CLI, API
- Can use as storage for downloading and scraping
- 100’s of features for management, security, access control
- Serves as point of entry and exit for rest of services in the lifecycle
- Reliable: 99.999999999% of durability throughout the year
- Scalable: zero provisioning needed



Mountable distributed file system

Long-term object storage

Process

Common data analysis and transformation steps

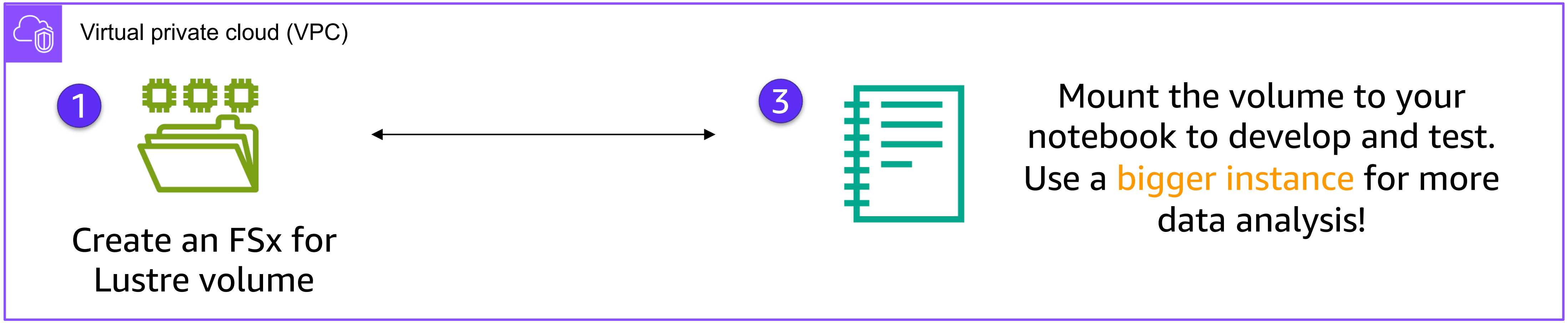
2. Sample and analyze

Bucket
Notebook

Inspect a small data sample

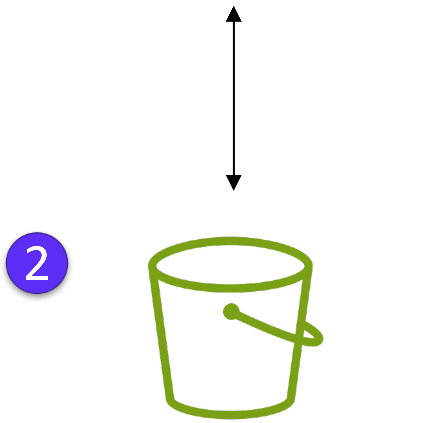
- Grab ~1% of your data
- Pull to a notebook
  - On your laptop
  - On SageMaker
  - On any instance
- Run Python code to understand and inspect
- Detect bias
- Test small model versions
- Prompt engineering
- Lightweight fine-tuning
- Use @remote decorator to scale to single-node

3. Create a distributed file system



Create an FSx for Lustre volume

Mount the volume to your notebook to develop and test. Use a bigger instance for more data analysis!



Enable 2-way data repository with an S3 path

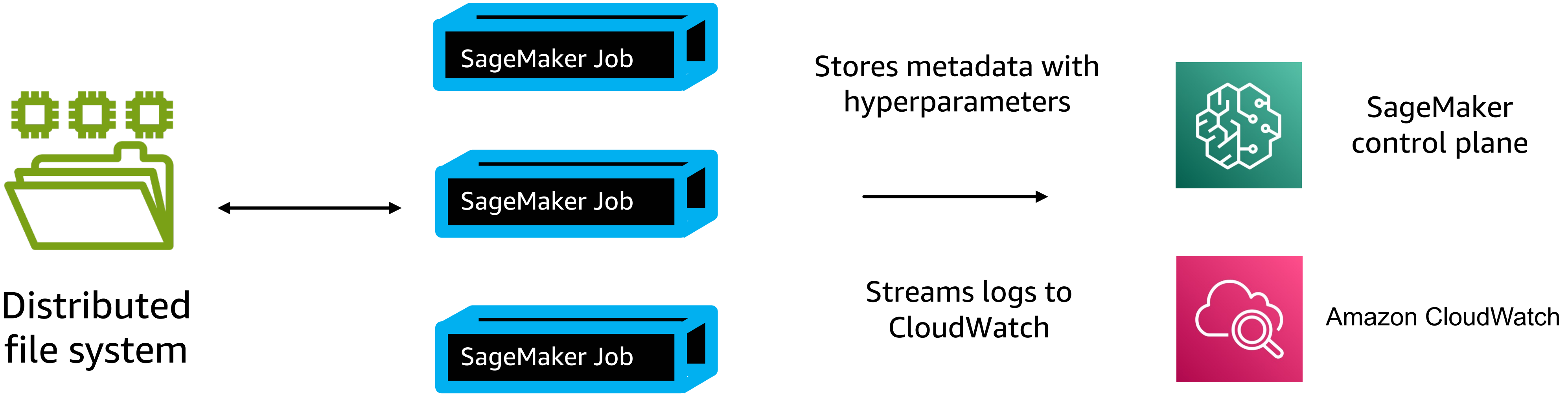
Pro tips with FSx for Lustre

1. Takes a few minutes to create in the AWS console.
2. Follow guides to do this in your preferred VPC, including SGs and ACLs.
3. Lustre copies all the *file names* on to the mount. You need to run a CPU-based job to *read* all of these files, which moves the files onto Lustre.
4. Mount the volume from a notebook in the same VPC to test the data and develop your scripts

4. Run data processing at scale with CPUs

1. Each job can train as many models as you need, or process as much data as you need.
2. You can use ***warm pools*** to reuse the instances
3. Use CPUs to keep costs low!

```
for job in list_of_jobs:
    s3_input = get_data(job)
    s3_output = get_location(job)
    estimator = get_estimator(job, s3_output)
    estimator.fit(s3_input, wait=False)
```



Distributed file system

SageMaker Job

SageMaker Job

SageMaker Job

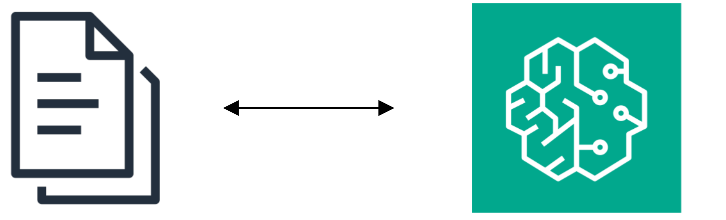
Stores metadata with hyperparameters

Streams logs to CloudWatch

SageMaker control plane

Amazon CloudWatch

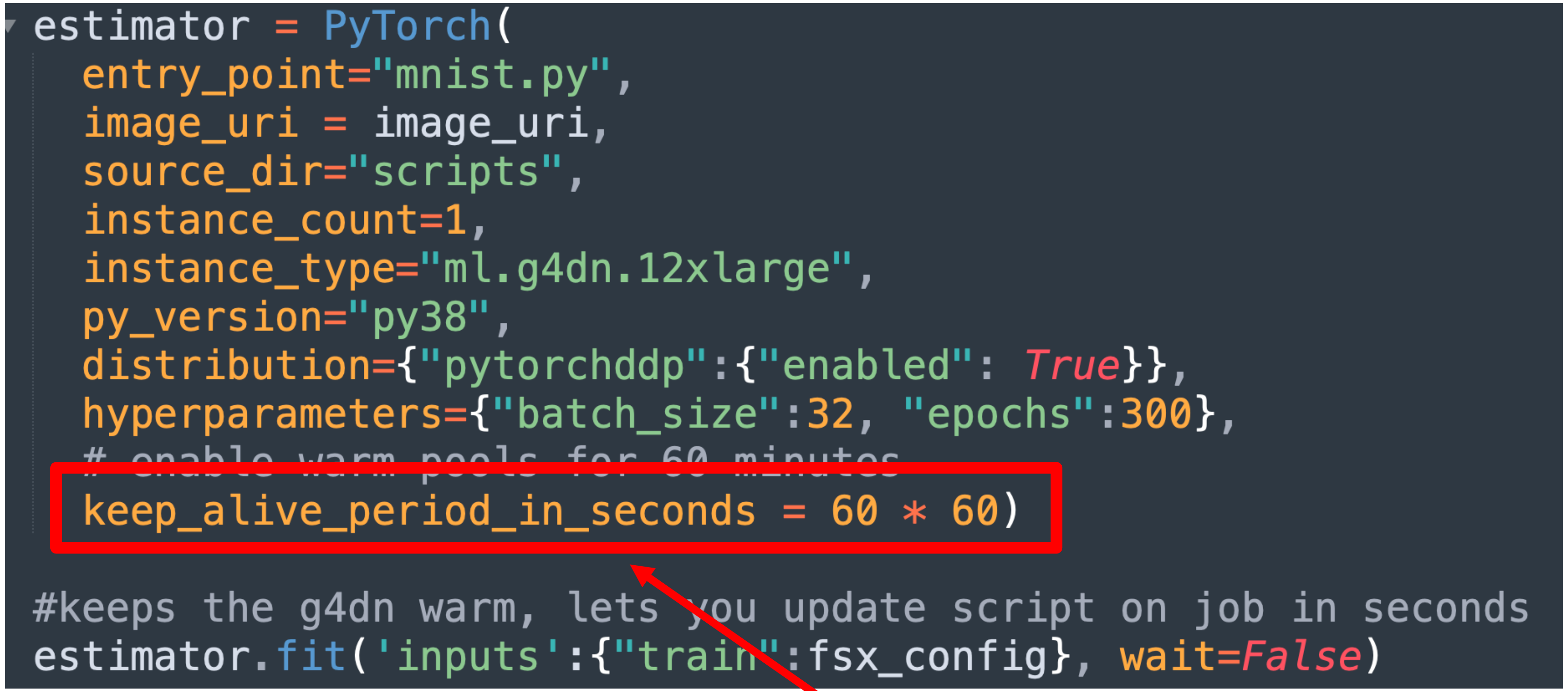
5. Enable SageMaker warm pools for rapid development



Local scripts

SageMaker training instances

Instances stay warm, enabling new script updates in seconds



```
estimator = PyTorch(
    entry_point="mnist.py",
    image_uri = image_uri,
    source_dir="scripts",
    instance_count=1,
    instance_type="ml.g4dn.12xlarge",
    py_version="py38",
    distribution={"pytorchddp":{"enabled": True}},
    hyperparameters={"batch_size":32, "epochs":300},
    # enable warm pools for 60 minutes
    keep_alive_period_in_seconds = 60 * 60)

#keeps the g4dn warm, lets you update script on job in seconds
estimator.fit('inputs':{"train":fsx_config}, wait=False)
```

Add this parameter to use warm pools

6. Mount a distributed file system for SageMaker Training

1. Point to the file system settings:
   File system ID
   Security group ID
   Subnet
   Mount name

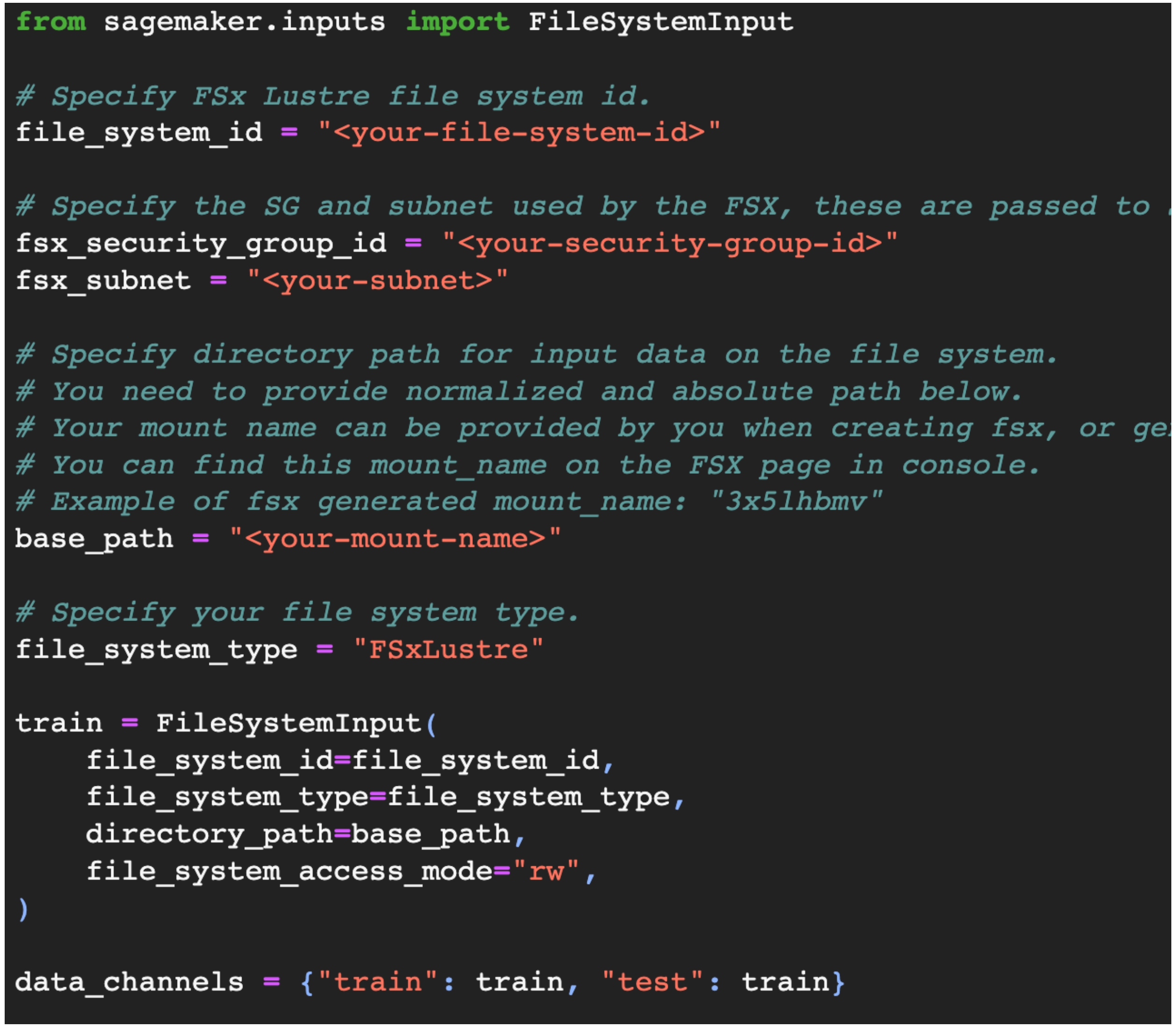
2. Point to your VPC settings:
   Subnet
   Security group

3. Create an S3 VPC endpoint:
   To write to S3 from the VPC

4. Update VPC rules to ensure access:
   NAT Gateway
   ACLs
   Route table

5. Create a data channel:
   For the training files
   For the base model

6. Send all the configs to the training job and run!

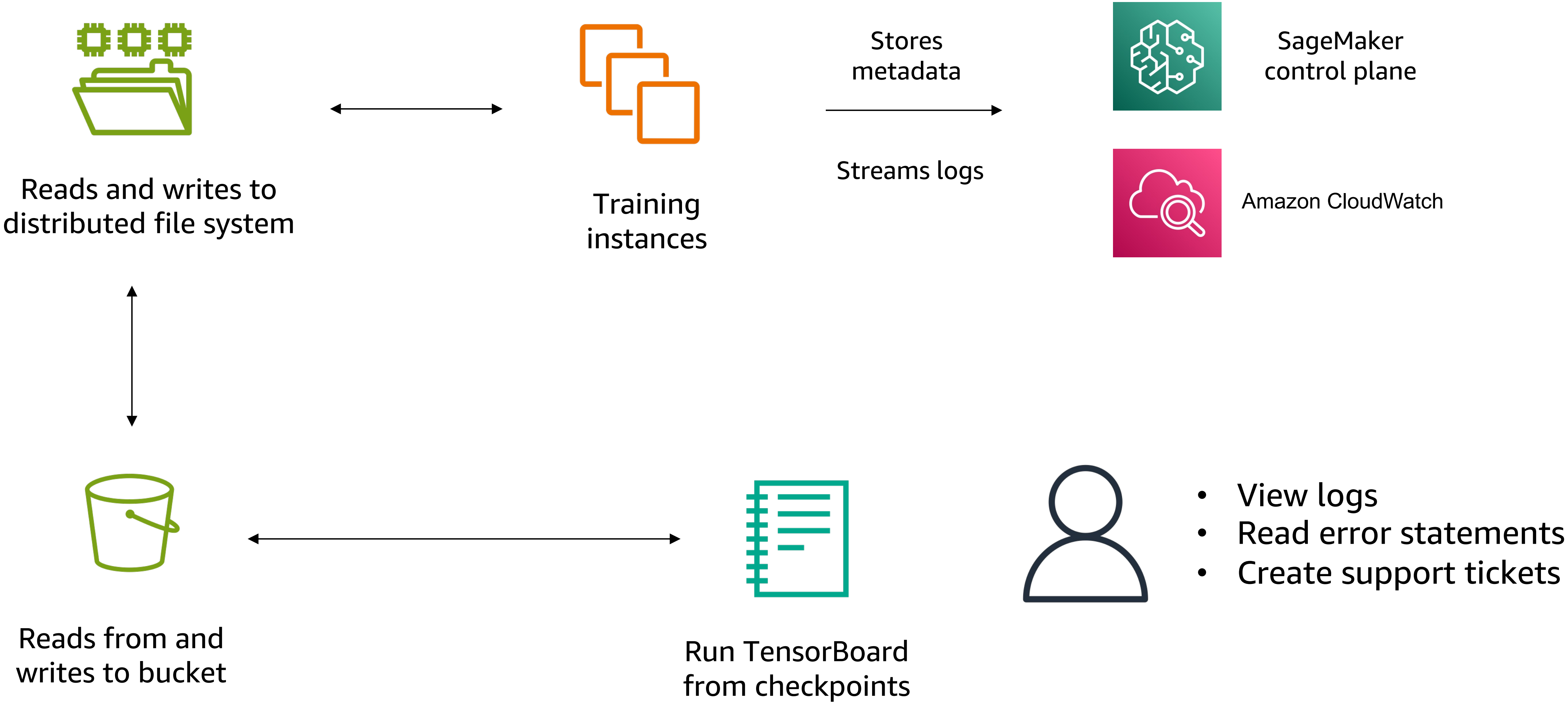


```
from sagemaker.inputs import FileSystemInput

# Specify FSx Lustre file system id.
file_system_id = "<your-file-system-id>"

# Specify the SG and subnet used by the FSX, these are passed to
fsx_security_group_id = "<your-security-group-id>"
fsx_subnet = "<your-subnet>"

# Specify directory path for input data on the file system.
# You need to provide normalized and absolute path below.
# Your mount name can be provided by you when creating fsx, or ge
# You can find this mount_name on the FSX page in console.
# Example of fsx generated mount_name: "3x5lhbmv"
base_path = "<your-mount-name>"

# Specify your file system type.
file_system_type = "FSxLustre"

train = FileSystemInput(
    file_system_id=file_system_id,
    file_system_type=file_system_type,
    directory_path=base_path,
    file_system_access_mode="rw",
)

data_channels = {"train": train, "test": train}
```

Mounts in seconds!

Pro tip – remember to hydrate your Lustre volume with the actual data and use CPUs

7. Monitor performance and troubleshoot



Reads and writes to distributed file system

Training instances

Stores metadata

Streams logs

SageMaker control plane

Amazon CloudWatch

Reads from and writes to bucket

Run TensorBoard from checkpoints

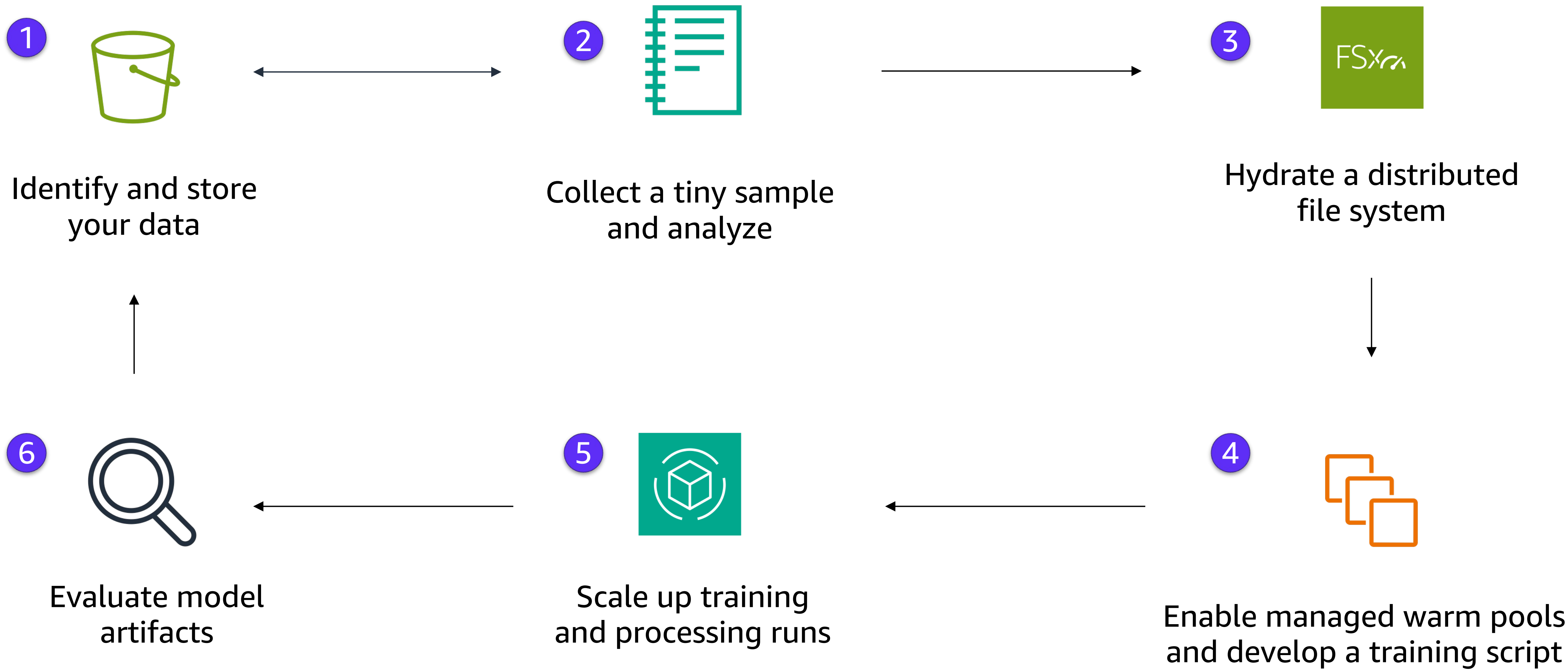
- View logs
- Read error statements
- Create support tickets

What if my instances go down?

- Accelerators are themselves extremely complex distributed compute systems
- Using hundreds and thousands of accelerators *increases the likelihood* of even one of these breaking
- Managed services offer enhanced reliability
  - Replacing instances during the training process
  - Running GPU-health checks ahead of launching the instances
  - Automatic job retries to get more instances and counter errors
  - Add model checkpoints every few hours, and continue training from them
- You can build your own Lambda functions to monitor your jobs and restart them on other conditions, like script, loss, and data errors

Lambda functions can do a lot for you!

Steps to prepare and train at scale on AWS



1 Identify and store your data

2 Collect a tiny sample and analyze

3 Hydrate a distributed file system

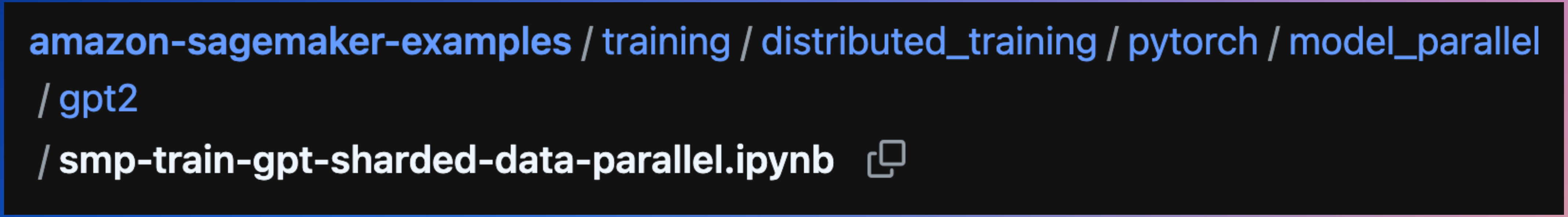
4 Enable managed warm pools and develop a training script

5 Scale up training and processing runs

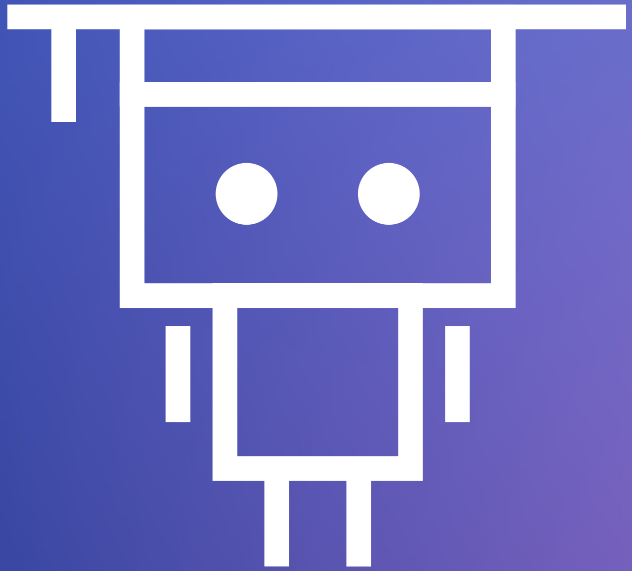
6 Evaluate model artifacts

Hands-on demo

https://bit.ly/sm-nb-4



amazon-sagemaker-examples / training / distributed_training / pytorch / model_parallel / gpt2 / smp-train-gpt-sharded-data-parallel.ipynb



Thank you!

Type: Corrections, feedback, or other questions?
Contact us at https://support.awsamazon.com/#/contacts/aws-academy.
All trademarks are the property of their owners.