1. Big picture

You'll do this:

1. **Create / use a SageMaker-managed MLflow tracking server** (with Model Registry).
2. **Write a conversion script** that:
   - Downloads and untars your existing `model.tar.gz` from S3.
   - Logs it as an **MLflow Transformers model** using `mlflow.transformers.log_model`.
   - Registers it in the **MLflow Model Registry** (backed by SageMaker).
   - Optionally downloads the MLflow model artifacts and re-packages them as `mlflow-model.tar.gz` and uploads that to S3.
3. **Wrap that script in a SageMaker Pipelines `ProcessingStep`** so you can run it as part of an ML pipeline.

Because you're on SageMaker with managed MLflow, the AWS MLflow plugin lets you connect to the MLflow tracking server by just using its **ARN as the tracking URI**. (AWS Documentation)

---

2. One-time setup

2.1. Create / identify your SageMaker MLflow tracking server

In SageMaker Studio (Unified Studio):

- Go to **Build** → **MLflow** and create a **Tracking Server** (if one doesn't exist).
- Configure:
  - An **S3 artifact bucket** for MLflow.
  - An **execution IAM role**.
- Grab the **Tracking Server ARN**, e.g.:

```
arn:aws:sagemaker:<region>:<account-id>:mlflow-app/app-xxxxxxxx
```

This ARN will be your `mlflow_tracking_arn`.

AWS's docs show that with the AWS MLflow plugin (`sagemaker-mlflow`), you can do:

```
mlflow.set_tracking_uri("arn:aws:sagemaker:...:mlflow-app/app-...")
```

(AWS Documentation)

2.2. IAM permissions for the Pipeline role

The SageMaker Pipeline execution role must be able to:

- **Read** your *existing* model artifact:
  - `s3:GetObject` on `s3://<your-training-bucket>/<path-to-model.tar.gz>`.
- **Write** the MLflow packaged model tarball:
  - `s3:PutObject` on `s3://<your-mlflow-model-bucket>/<prefix>/`.
- **Call MLflow**:
  - The role needs `sagemaker:InvokeEndpoint` and the MLflow-related IAM permissions for your MLflow app as documented by AWS (create experiments, runs, registered models, etc.). (AWS Documentation)

2.3. Container image for the Processing step

Use a custom ECR image (or a SageMaker base Python image) with:

```
python >= 3.10
mlflow == <version that matches your tracking server>   # e.g. 2.13.2 or
3.0.0
sagemaker-mlflow
transformers
torch
boto3
```

Per AWS docs, `sagemaker-mlflow` + a compatible `mlflow` version is required. (AWS Documentation)

Example `requirements.txt` (used when building your container):

```
mlflow==3.0.0        # or 2.13.2 / 2.16.2 depending on your tracking server
sagemaker-mlflow
transformers
torch
boto3
```

---

3. Confirm your starting artifact layout

Your `model.tar.gz` contains:

- `model.safetensors` (weights)
- `config.json`
- tokenizer files: `tokenizer.json`, `tokenizer_config.json`, `special_tokens_map.json`, `processor_config.json`
- `training_args.bin`

Once untarred into a directory, that's a valid **local HuggingFace checkpoint**.

MLflow's Transformers flavor can log a model directly **from a local checkpoint path** (directory with `config.json` + weights + tokenizer files) using `mlflow.transformers.log_model(transformers_model="<local_path>", task="...", ...)`. (MLflow)

So we **don't need to load the model into memory**; MLflow reads metadata and weights from that directory.

---

4. Standalone conversion script: HF checkpoint → MLflow model + tarball

Below is a self-contained script that:

1. Downloads `model.tar.gz` from S3 & extracts it.
2. Logs it as an MLflow Transformers model.
3. Registers it to the MLflow Model Registry.
4. Optionally downloads the MLflow model artifacts and uploads `mlflow-model.tar.gz` to S3.

Save this as `convert_hf_to_mlflow.py` and include it in your Processing image or in a code S3 bucket.

```python
# convert_hf_to_mlflow.py

import argparse
import os
import tarfile
from urllib.parse import urlparse

import boto3
import mlflow
import mlflow.transformers
from mlflow.tracking import MlflowClient
```

```python
import mlflow.artifacts as mlflow_artifacts


def parse_s3_uri(s3_uri: str):
    if not s3_uri.startswith("s3://"):
        raise ValueError(f"Invalid S3 URI: {s3_uri}")
    parsed = urlparse(s3_uri)
    bucket = parsed.netloc
    key = parsed.path.lstrip("/")
    return bucket, key


def download_s3_object(s3_uri: str, local_path: str):
    bucket, key = parse_s3_uri(s3_uri)
    s3 = boto3.client("s3")
    os.makedirs(os.path.dirname(local_path), exist_ok=True)
    s3.download_file(bucket, key, local_path)


def upload_s3_object(local_path: str, s3_uri: str):
    bucket, key = parse_s3_uri(s3_uri)
    s3 = boto3.client("s3")
    s3.upload_file(local_path, bucket, key)


def extract_tar_gz(tar_path: str, dst_dir: str):
    os.makedirs(dst_dir, exist_ok=True)
    with tarfile.open(tar_path, "r:gz") as tar:
        tar.extractall(dst_dir)


def main():
    parser = argparse.ArgumentParser()

    # Where the original HF model.tar.gz lives
    parser.add_argument("--s3-model-artifact", required=True)

    # HuggingFace task, e.g. "text-generation", "text-classification", etc.
    parser.add_argument("--hf-task", required=True)

    # MLflow config
    parser.add_argument("--mlflow-tracking-arn", required=True)
    parser.add_argument("--mlflow-experiment-name", required=True)
    parser.add_argument("--mlflow-registered-model-name", required=True)

    # Optional: place to upload mlflow-model.tar.gz
    parser.add_argument("--mlflow-model-tar-s3-uri", required=False)

    # Optional: tag with original SageMaker model package ARN / name
    parser.add_argument("--source-model-package-arn", required=False)

    args = parser.parse_args()

    # 1) Configure MLflow to talk to SageMaker managed MLflow
    # ARN is used as tracking URI thanks to sagemaker-mlflow plugin
    mlflow.set_tracking_uri(args.mlflow_tracking_arn)
    mlflow.set_experiment(args.mlflow_experiment_name)

    # 2) Download and untar the original HF artifact
    local_artifacts_root = "/opt/ml/processing"
    local_tar_path = os.path.join(local_artifacts_root, "model.tar.gz")
    local_hf_dir = os.path.join(local_artifacts_root, "hf_model")

    print(f"Downloading {args.s3_model_artifact} to {local_tar_path}")
    download_s3_object(args.s3_model_artifact, local_tar_path)
    print(f"Extracting {local_tar_path} to {local_hf_dir}")
    extract_tar_gz(local_tar_path, local_hf_dir)

    # At this point local_hf_dir contains config.json, model.safetensors,
tokenizer files, etc.

    # 3) Start an MLflow run and log the HF checkpoint as an MLflow
Transformers model
    with mlflow.start_run(run_name="import-hf-model") as run:
        run_id = run.info.run_id
        print(f"MLflow run_id: {run_id}")

        # Helpful metadata
        mlflow.set_tag("source", "sagemaker-model-registry")
        mlflow.set_tag("source_s3_artifact", args.s3_model_artifact)
        if args.source_model_package_arn:
            mlflow.set_tag("sagemaker_model_package_arn",
args.source_model_package_arn)

        mlflow.log_param("hf_task", args.hf_task)

        # Log as Transformers model (from local checkpoint path)
        # For large models, this avoids loading into memory; MLflow inspects
the checkpoint.
        # See "Storage-Efficient Model Logging" in the MLflow transformers
docs.
        model_info = mlflow.transformers.log_model(
            transformers_model=local_hf_dir,
            task=args.hf_task,
            artifact_path="model",  # subdir in the run's artifact store
            registered_model_name=args.mlflow_registered_model_name,
        )

        print(f"Logged MLflow model at: {model_info.model_uri}")

        # 4) Optionally: download the MLflow model and repack as mlflow-
model.tar.gz
        if args.mlflow_model_tar_s3_uri:
            local_mlflow_model_dir = os.path.join(local_artifacts_root,
"mlflow_model")
            print(f"Downloading MLflow model artifacts from
{model_info.model_uri} ...")
            local_model_path = mlflow_artifacts.download_artifacts(
                artifact_uri=model_info.model_uri,
                dst_path=local_mlflow_model_dir,
            )
            print(f"MLflow model downloaded to: {local_model_path}")

            # Create tar.gz
            tar_output_path = os.path.join(local_artifacts_root, "mlflow-
model.tar.gz")
            print(f"Creating tarball {tar_output_path} ...")
            with tarfile.open(tar_output_path, "w:gz") as tar:
                # We want a nice top-level directory name in the tarball
                tar.add(local_model_path, arcname="mlflow-model")

            print(f"Uploading tarball to {args.mlflow_model_tar_s3_uri}
...")
            upload_s3_object(tar_output_path, args.mlflow_model_tar_s3_uri)

    print("Conversion to MLflow model complete.")


if __name__ == "__main__":
    main()
```

What this script gives you

- A **Model in the MLflow Model Registry** under `mlflow-registered-model-name`.
  - Because you used `registered_model_name=...` in `log_model`, MLflow automatically creates/updates a registered model and version. (MLflow)

- The registered model (in SageMaker managed MLflow) can then be **automatically mirrored into SageMaker Model Registry**, thanks to AWS' integration. (AWS Documentation)
- If `--mlflow-model-tar-s3-uri` is provided, you also get a **`mlflow-model.tar.gz`** in that S3 location containing the MLflow model directory:
  - `MLmodel`
  - `conda.yaml` / `requirements.txt`
  - underlying HF checkpoint & tokenizer
  - transformers & pyfunc flavors

---

5. Building the SageMaker Pipeline

Now you wrap that script into a **SageMaker Pipeline** so the whole process can run as a step.

5.1. Pipeline parameters

In your pipeline definition (e.g. `hf_to_mlflow_pipeline.py`):

```python
from sagemaker.workflow.parameters import ParameterString
from sagemaker.workflow.pipeline import Pipeline
from sagemaker.workflow.steps import ProcessingStep
from sagemaker.processing import ScriptProcessor
import sagemaker

region = sagemaker.Session().boto_region_name
role = "<your-sagemaker-pipeline-execution-role>"  # IAM role ARN

# Parameters so you can reuse the pipeline for different models
s3_model_artifact = ParameterString(
    name="S3ModelArtifact",
    default_value="s3://your-bucket/path/to/model.tar.gz",
)

mlflow_tracking_arn = ParameterString(
    name="MLflowTrackingArn",
    default_value="arn:aws:sagemaker:region:account-id:mlflow-app/app-xxxx",
)

mlflow_experiment_name = ParameterString(
    name="MLflowExperimentName",
    default_value="hf-import-experiments",
)
```

```
mlflow_registered_model_name = ParameterString(
    name="MLflowRegisteredModelName",
    default_value="my-hf-model-mlflow",
)

hf_task = ParameterString(
    name="HFTask",
    default_value="text-generation",  # adjust for your use-case
)

mlflow_model_tar_s3_uri = ParameterString(
    name="MlflowModelTarS3Uri",
    default_value="s3://your-bucket/mlflow-models/my-hf-model/mlflow-
model.tar.gz",
)

source_model_package_arn = ParameterString(
    name="SourceModelPackageArn",
    default_value="",  # optional; populate with SageMaker Model Package ARN
if you want
)
```

5.2. ScriptProcessor step

Use a `ScriptProcessor` to run the conversion script inside a container that has `mlflow`, `sagemaker-mlflow`, `transformers`, etc. (Amazon SageMaker Examples)

```
from sagemaker.processing import ScriptProcessor
from sagemaker.workflow.steps import ProcessingStep

script_processor = ScriptProcessor(
    image_uri="<your-ecr-image-with-mlflow-and-transformers>",
    command=["python3"],
    instance_type="ml.m5.xlarge",
    instance_count=1,
    role=role,
)

convert_step = ProcessingStep(
    name="ConvertHuggingFaceToMLflow",
    processor=script_processor,
    code="convert_hf_to_mlflow.py",
    job_arguments=[
        "--s3-model-artifact", s3_model_artifact,
        "--hf-task", hf_task,
        "--mlflow-tracking-arn", mlflow_tracking_arn,
        "--mlflow-experiment-name", mlflow_experiment_name,
        "--mlflow-registered-model-name", mlflow_registered_model_name,
        "--mlflow-model-tar-s3-uri", mlflow_model_tar_s3_uri,
        "--source-model-package-arn", source_model_package_arn,
    ],
)
```

No explicit `ProcessingInput` / `ProcessingOutput` is strictly needed here since the script pulls from and pushes to S3 directly.

5.3. Assemble the pipeline

```
def get_pipeline():
    pipeline = Pipeline(
        name="HFToMLflowRegistrationPipeline",
        parameters=[
            s3_model_artifact,
            mlflow_tracking_arn,
            mlflow_experiment_name,
            mlflow_registered_model_name,
            hf_task,
            mlflow_model_tar_s3_uri,
            source_model_package_arn,
        ],
        steps=[convert_step],
    )
    return pipeline
```

Then from a notebook or CI job:

```
from sagemaker.workflow.pipeline import Pipeline

pipeline = get_pipeline()
pipeline.upsert(role_arn=role)
execution = pipeline.start()
```

That single step will:

1. Pull your **existing SageMaker/HF `model.tar.gz`** from S3.
2. Create an **MLflow model (Transformers flavor + pyfunc)** from it.
3. **Register** it into the MLflow Model Registry (inside your SageMaker-managed MLflow).

4. Save a `mlflow-model.tar.gz` in S3 (if the parameter is set).

---

6. Getting the S3 path from SageMaker Model Registry (optional helper)

If instead of hard-coding the `S3ModelArtifact` parameter you want to derive it **from an existing SageMaker Model Package**, you can do:

```
import boto3

sm = boto3.client("sagemaker")

def get_model_data_from_package(model_package_arn: str) -> str:
    resp = sm.describe_model_package(ModelPackageName=model_package_arn)
    return resp["InferenceSpecification"]["Containers"][0]["ModelDataUrl"]

# Then pass that value to the pipeline parameter S3ModelArtifact
```

The returned `ModelDataUrl` is the `s3://.../model.tar.gz` you're already seeing in the console. (AWS Documentation)

---

7. What you get at the end

After a pipeline run completes:

- In the **MLflow UI** (connected to your tracking server):
  - You'll see a **Run** under `HFTask` /experiment.
  - In the **Models** tab, a registered model named `my-hf-model-mlflow` with a new version.
- In **SageMaker Studio → Models / Model Registry**:
  - Because of the SageMaker–MLflow integration, models registered in MLflow Model Registry are mirrored into SageMaker's Model Registry as Model Package Groups / Versions. (AWS Documentation)
- In **S3**:
  - MLflow artifacts for the run (the usual `MLmodel`, environment, HF checkpoint directory).
  - Your explicit `mlflow-model.tar.gz` at whatever `MlflowModelTarS3Uri` you configured.

From here you can:

- Use **ModelBuilder** to deploy the MLflow model to SageMaker endpoints by pointing it to the MLflow artifact S3 path. (AWS Documentation)
- Or build additional SageMaker Pipeline steps (approval, testing, deployment) that depend on the new MLflow registered model.