Package in a Dockerfile

So far we have been training our models in Sagemaker before deploying them to Sagemaker endpoints. All of this was happening inside Docker Containers, however, Sagemaker had abstracted away from this process by using estimators to create prebuilt docker images.

But how do you deploy a model in Sagemaker using a container that you have built locally? On this page, we will learn more about how Sagemaker uses Docker for training and inference processes and how you can "Bring Your Own Model" to Sagemaker and deploy it.

Prebuilt Docker Images

So far you have been using Prebuilt Docker Images when creating Pytorch or TensorFlow estimators using the SageMaker Python SDK.

Such prebuilt estimators are available for deep learning frameworks like TensorFlow, HuggingFace, and Pytorch, machine learning frameworks like scikit-learn and XGBBoost, and other frameworks like Apache MXNet, Chainer, and SparkML. These are really powerful and should cater to most of your training. A list of the available images can be found here. You can also fetch these images without using the SageMaker Python API and instructions for that can be found here.
Extending Prebuilt Images: In case the prebuilt images are not enough and you want some added functionalities or settings, you can extend them without having them create an image from scratch. You will need to set the `SAGEMAKER_SUBMIT_DIRECTORY` and `SAGEMAKER_PROGRAM` environment variables to extend the container. More information about these variables can be found here. To extend a prebuilt container, you need to do the following steps:

- **Step 1**: Create a new SageMaker Notebook.
- **Step 2**: Create your updated Dockerfile. To utilize previously built images, you will need to download one of the SageMaker images. You can find a list of them here. Here is what a sample docker file might look like:

```
#SageMaker PyTorch image
FROM 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.5.1-cpu-p

ENV PATH="/opt/ml/code:${PATH}"

#this environment variable is used by the SageMaker PyTorch container to deter

ENV SAGEMAKER_SUBMIT_DIRECTORY /opt/ml/code

#/opt/ml and all subdirectories are utilized by SageMaker, use the /code subdi

COPY cifar10.py /opt/ml/code/cifar10.py

#Defines cifar10.py as script entrypoint

ENV SAGEMAKER_PROGRAM cifar10.py
```

- **Step 3**: Build the docker container locally. To do that, you can use the following command:

```
docker build -t <your container here> .
```

- **Step 4**: You can now test your built docker container locally or upload it to Amazon Elastic Container Registry (Amazon ECR).
- **Step 5**: Cleanup Resources that you used.

You can get a more detailed tutorial on extending your container here

Using your own Models

So far we have seen how you can use prebuilt images to train models in Sagemaker and deploy them. We also saw how you can extend the images in case you want to include some additional functionalities.

However, what happens if you do not want to use the prebuilt docker containers and instead want to use your own custom containers? How can you deploy a model trained in such a container in SageMaker?

SageMaker allows you to "Bring Your Own Model" and train and deploy them. By packaging an algorithm in a container, you can bring almost any code to the Amazon SageMaker environment, regardless of programming language, environment, framework, or dependencies.

How it works

SageMaker uses the same container for training and deploying. It runs the container with either a `train` or `serve` argument depending on what it is doing.

Training your model is easy. You can use the same training scripts that we have been using in this course. However, when you are Deploying your model, the container should create two endpoints:

- `/ping`: Which is called to check whether the container is up or not. It should return a `200` response if the container is running. and,
- `/invocations`: Which is the endpoint that receives the query data.

Once you have built the container and tested it locally you need to register it to ECR.

Finally, you can use your own container for training and deploying your model inside SageMaker with an estimator:

```
from sagemaker.estimator import Estimator

hyperparameters = {"train-steps": 100}

instance_type = "ml.m4.xlarge"

estimator = Estimator(
    role=role,
    instance_count=1,
    instance_type=instance_type,
    image_uri=ecr_image,
    hyperparameters=hyperparameters,
)

estimator.fit(data_location)

predictor = estimator.deploy(1, instance_type)
```

where the `image_uri` is the path to your image in ecr.

Additional Resources

- You can read more about using Docker Containers with SageMaker here

- "Bring Your Own" TensorFlow Model: link
- Extending Pytorch Container: link
- Example Notebooks for Docker and SageMaker: link
- Other Advanced Functionality in SageMaker: link