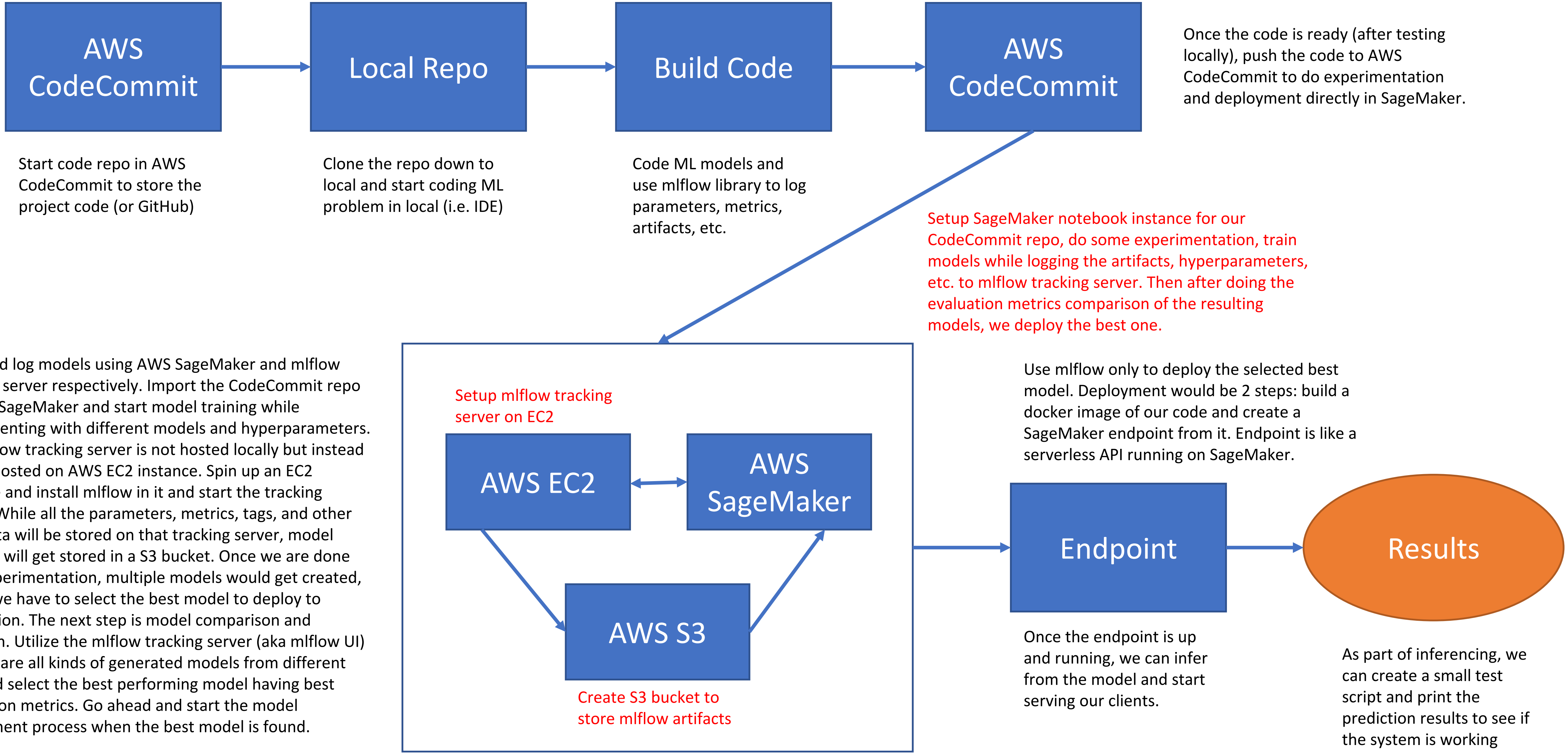
Subset of MLOps Pipeline in AWS Cloud



Create empty CodeCommit Repo to store code

Start code repo in AWS CodeCommit to store the project code (or GitHub)

Clone the repo down to local and start coding ML problem in local (i.e. IDE)

Build ML solution, run it locally, and push to CodeCommit repo

Code ML models and use mlflow library to log parameters, metrics, artifacts, etc.

Use CodeCommit repo to start experimentation, model trainings, etc. using AWS SageMaker while logging the entities to mlflow tracking server on EC2 instance.

Once the code is ready (after testing locally), push the code to AWS CodeCommit to do experimentation and deployment directly in SageMaker.

Setup SageMaker notebook instance for our CodeCommit repo, do some experimentation, train models while logging the artifacts, hyperparameters, etc. to mlflow tracking server. Then after doing the evaluation metrics comparison of the resulting models, we deploy the best one.

Train and log models using AWS SageMaker and mlflow tracking server respectively. Import the CodeCommit repo to AWS SageMaker and start model training while experimenting with different models and hyperparameters. The mlflow tracking server is not hosted locally but instead will be hosted on AWS EC2 instance. Spin up an EC2 instance and install mlflow in it and start the tracking server. While all the parameters, metrics, tags, and other metadata will be stored on that tracking server, model artifacts will get stored in a S3 bucket. Once we are done with experimentation, multiple models would get created, where we have to select the best model to deploy to production. The next step is model comparison and selection. Utilize the mlflow tracking server (aka mlflow UI) to compare all kinds of generated models from different runs and select the best performing model having best evaluation metrics. Go ahead and start the model deployment process when the best model is found.

Setup mlflow tracking server on EC2

Create S3 bucket to store mlflow artifacts

Use mlflow only to deploy the selected best model. Deployment would be 2 steps: build a docker image of our code and create a SageMaker endpoint from it. Endpoint is like a serverless API running on SageMaker.

Once the endpoint is up and running, we can infer from the model and start serving our clients.

As part of inferencing, we can create a small test script and print the prediction results to see if the system is working properly.

Model Evaluation & Comparison

- Each experiment trains a single type of model (Xgb, ridge, elasticnet)
  - 3 experiments total
- Multiple runs inside each experiment testing different combinations of parameters based on specified dictionary
- How to pick best model?
  - For each experiment, pick the best run. Register the model under same registry.
  - Compare best runs from each experiment.
  - Get the best model.

Model Deployment on AWS SageMaker

- mlflow sagemaker build-and-push-container
  - This command is responsible for building and pushing a docker container image that encapsulates the mlflow model along with its dependencies and configurations suitable for deployment on AWS SageMaker.
  - Mlflow prepares a docker image that includes the necessary dependencies and configurations specified for the model deployment
  - Once the image is built, it is pushed to Amazon ECR, which is AWS managed docker container registry. This makes the container image accessible and ready for deployment on AWS SageMaker.

Deployment (cont.)

- Once the docker image is created of our code and pushed it to ECR, we can deploy it and create an endpoint out of it which can be then used to serve our model.
- 2 ways to create the endpoint: CLI command vs Python code
- Get predictions → can be passed to downstream applications to create dashboard or trigger a new pipeline

Wrap up

- Delete the endpoint
- Stop SageMaker notebook instance
- Stop EC2 mlflow tracking server
- Etc.