# ML with Amazon SageMaker

## From idea to production

Yevgeniy Ilyin

Senior Solutions Architect
AWS

“Insanity is doing the same thing over and over and expecting different results.”

**Albert Einstein**

# Challenges with ML development

## Elasticity

- Different steps in the ML lifecycle require different compute resources
- Lack of elasticity leads to over- or under-provisioning of resources, poor cost-efficiency
- Need for decoupling the UI from kernel compute

## Scalability

- Management overhead to patch, secure, and maintain all ML environments being used by a large team
- Moving to production cycle typically requires switching between different tools and environments
- Processing of large datasets in a notebook is a challenge and requires specialized frameworks
- Challenge with distributed processing

## Collaboration & Access

- Development environment isolation is required for teams as each data scientist might be working on their ML problem
- Need for teams to share their work and artifacts while also preserving the state
- Need to support different types of teams
- How to keep people away from data

# Amazon SageMaker

# Amazon SageMaker Studio Notebooks

## IDE for ML

- All ML tools and features in a single, web-based interface
- One-click sharing of notebooks and state
- Full isolation of data science environments for each users
- Amazon Elastic File System (EFS) private home directory for each user

## Elastic notebooks

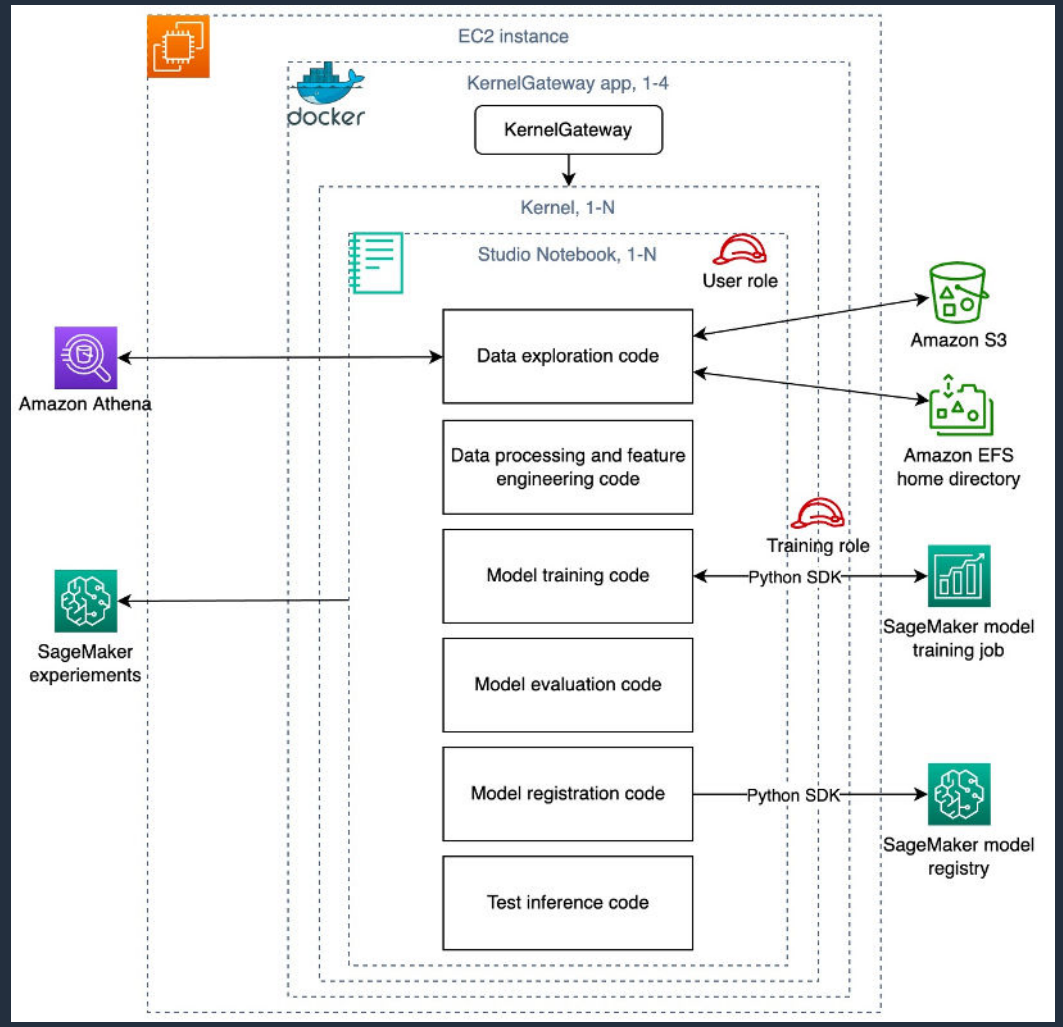
- One-click provisioning of elastic underlying compute resources, easy switching between GPU and CPU instances
- Built on Docker containers and EC2
- Decoupled Jupyter server from kernels
- Decoupled storage from kernel compute
- Full isolation of compute resources for each user

## Operationalization

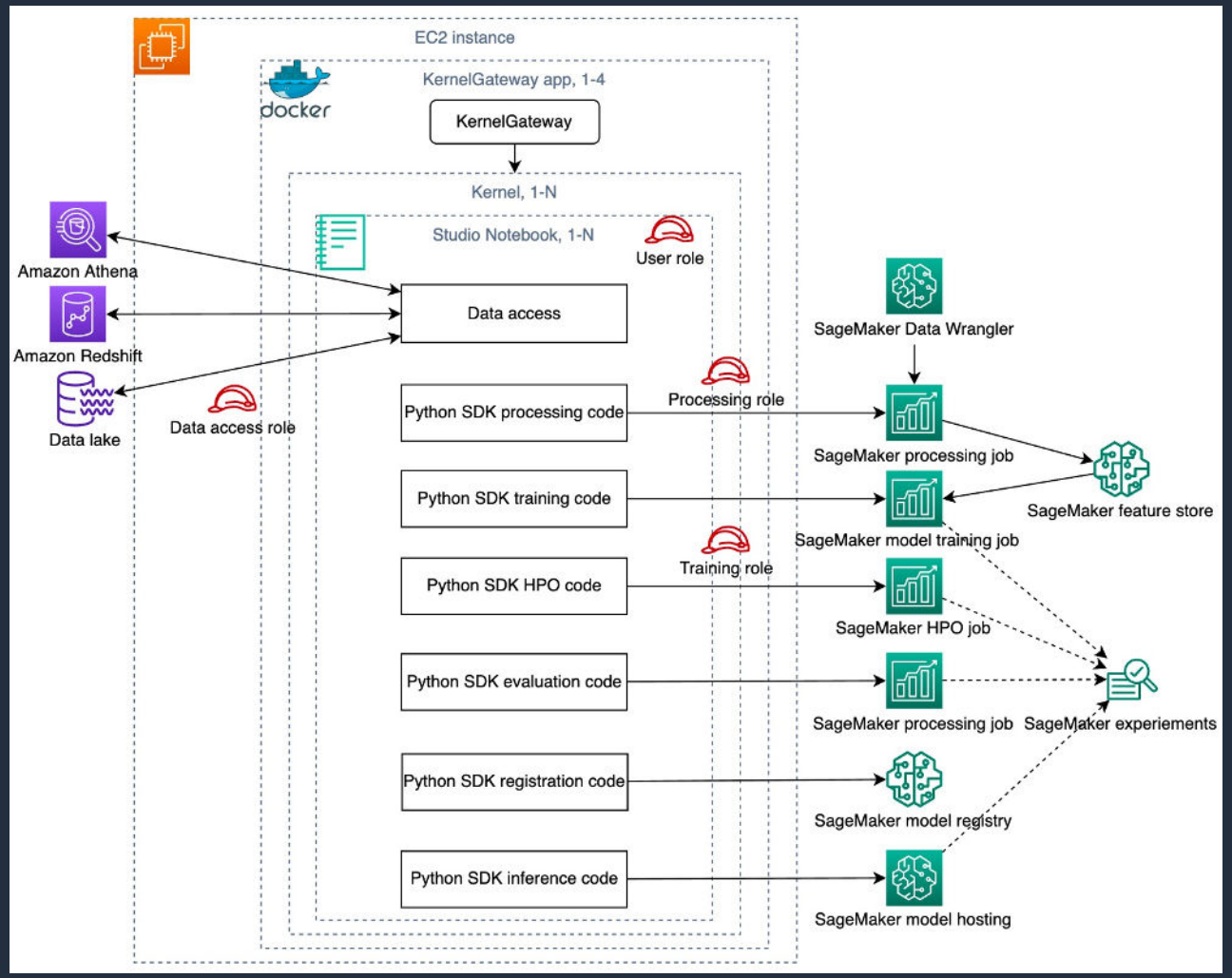
- Purpose-built functionalities to support and simplify production deployment and monitoring
- Enterprise-grade security
- Model deployment, inference endpoint management and monitoring from Studio UX
- Visual data processing flows and orchestration pipelines from Studio UX
- Data and resource access isolation

# Demo

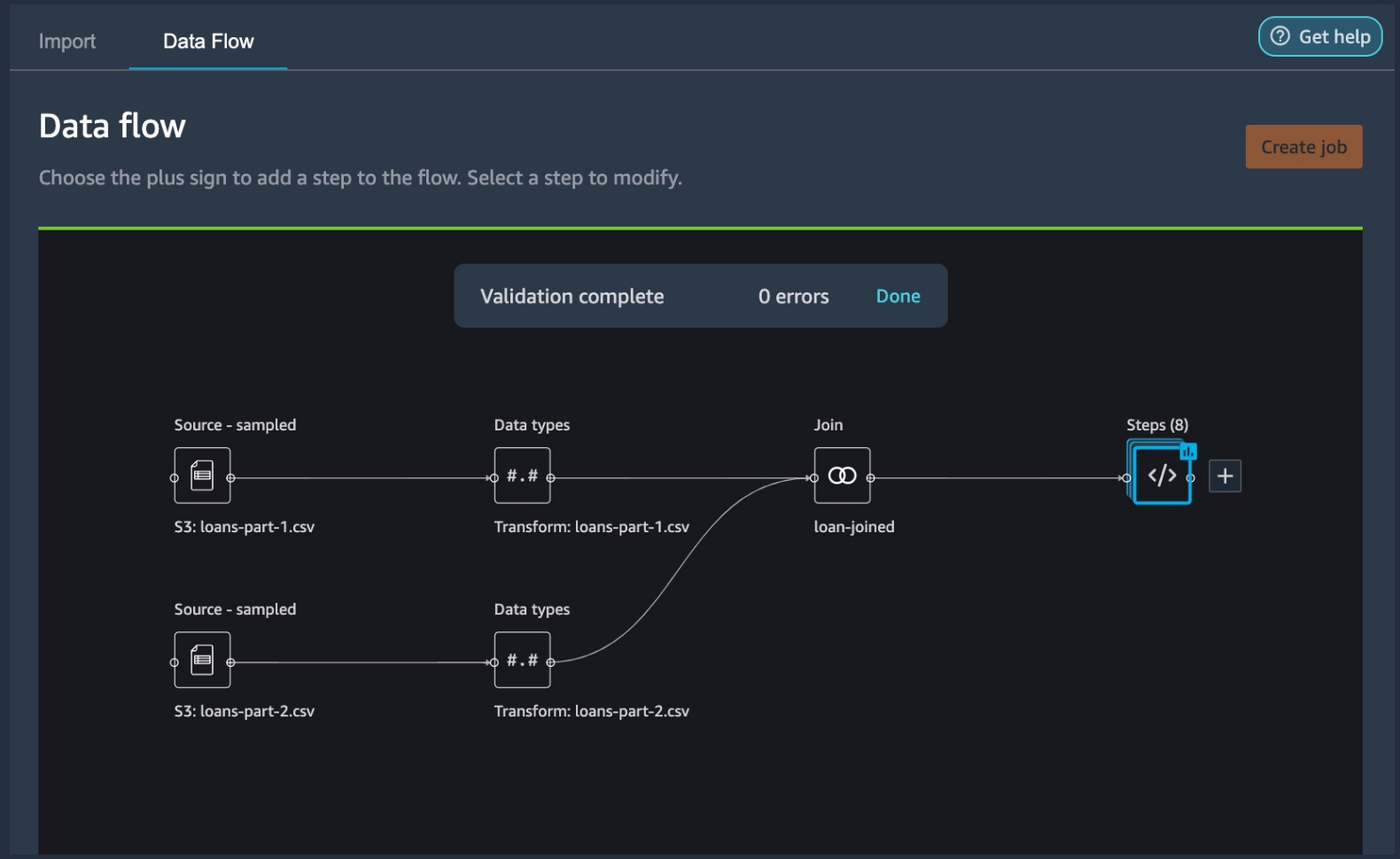
# Ideation and experimentation flow

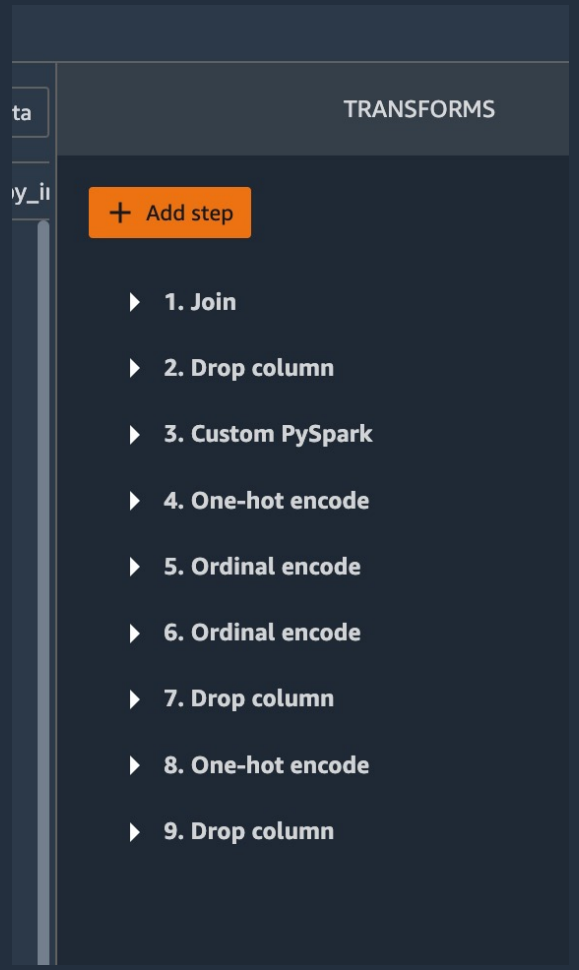


# Productization flow



# Visual data preparation flow



# Model training Python SDK code

```
[82]: xgb = sagemaker.estimator.Estimator(container,
                                          role,
                                          instance_count=1,
                                          instance_type='ml.m4.xlarge',
                                          output_path=f's3://{default_bucket}/{project_prefix}/mod
                                          sagemaker_session=session,
                                          base_job_name=project_prefix)
      xgb.set_hyperparameters(max_depth=10,
                              eta=0.2,
                              gamma=4,
                              min_child_weight=6,
                              subsample=0.8,
                              silent=0,
                              objective=objective,
                              num_class=num_class if num_class > 2 else None,
                              num_round=100)
```

```
[*]: xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

```
2022-04-25 09:39:53 Starting - Starting the training job...
2022-04-25 09:40:21 Starting - Preparing the instances for trainingProfilerReport-165087959
3: InProgress
.............
2022-04-25 09:42:19 Downloading - Downloading input data..
```

# Run batch inference Python SDK code

[39]:
```
transformer = xgb.transformer(
    instance_count=1,
    instance_type="ml.m5.4xlarge",
    accept="text/csv",
    role=role,
    output_path=transform_output_path,
    model_name=model_name
)
```

## Run transform job

[40]:
```
transformer.transform(
    data=s3_test,
    content_type="text/csv",
    split_type="Line",
    job_name=transform_job_name
)

transformer.wait()
```

# Pipeline Python SDK code

```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        p_processing_instance_type,
        p_processing_instance_count,
        p_processing_volume_size,
        p_flow_output_name,
        p_input_flow,
        p_input_data,
        p_output_prefix
    ],
    steps=[step_process],
    sagemaker_session=sagemaker_session
)

response = pipeline.upsert(
    role_arn=execution_role,
    description=pipeline_description,
    tags=[
    {'Key': 'sagemaker:project-name', 'Value': project_name },
    {'Key': 'sagemaker:project-id', 'Value': project_id }
],
)
```

## Start pipeline

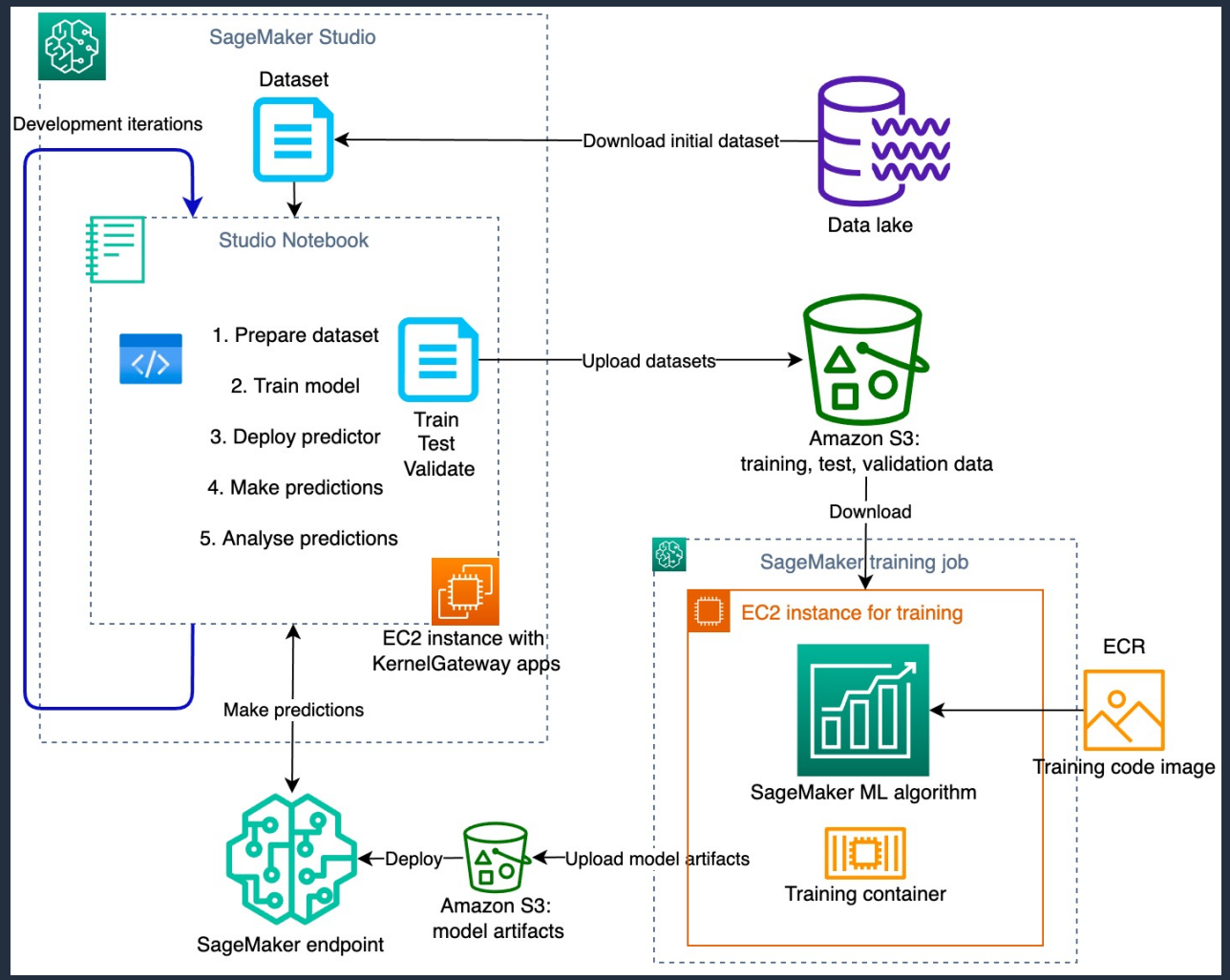
```
[16]: pipeline.upsert(role_arn=execution_role)
      execution = pipeline.start()

[ ]: execution.wait()

[ ]: execution.list_steps()
```

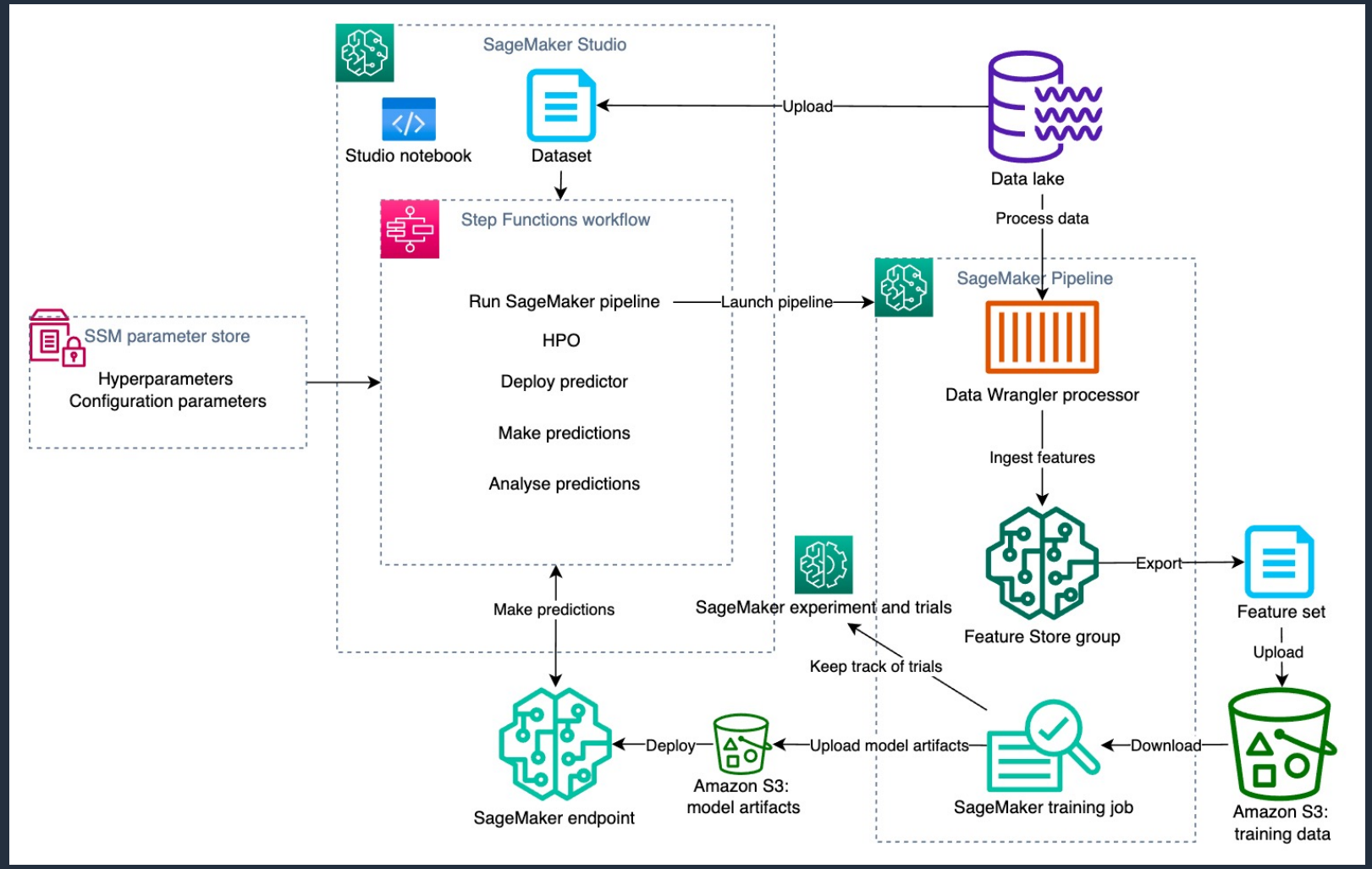
# An ML use case: from idea to production

# Step 1: Explore



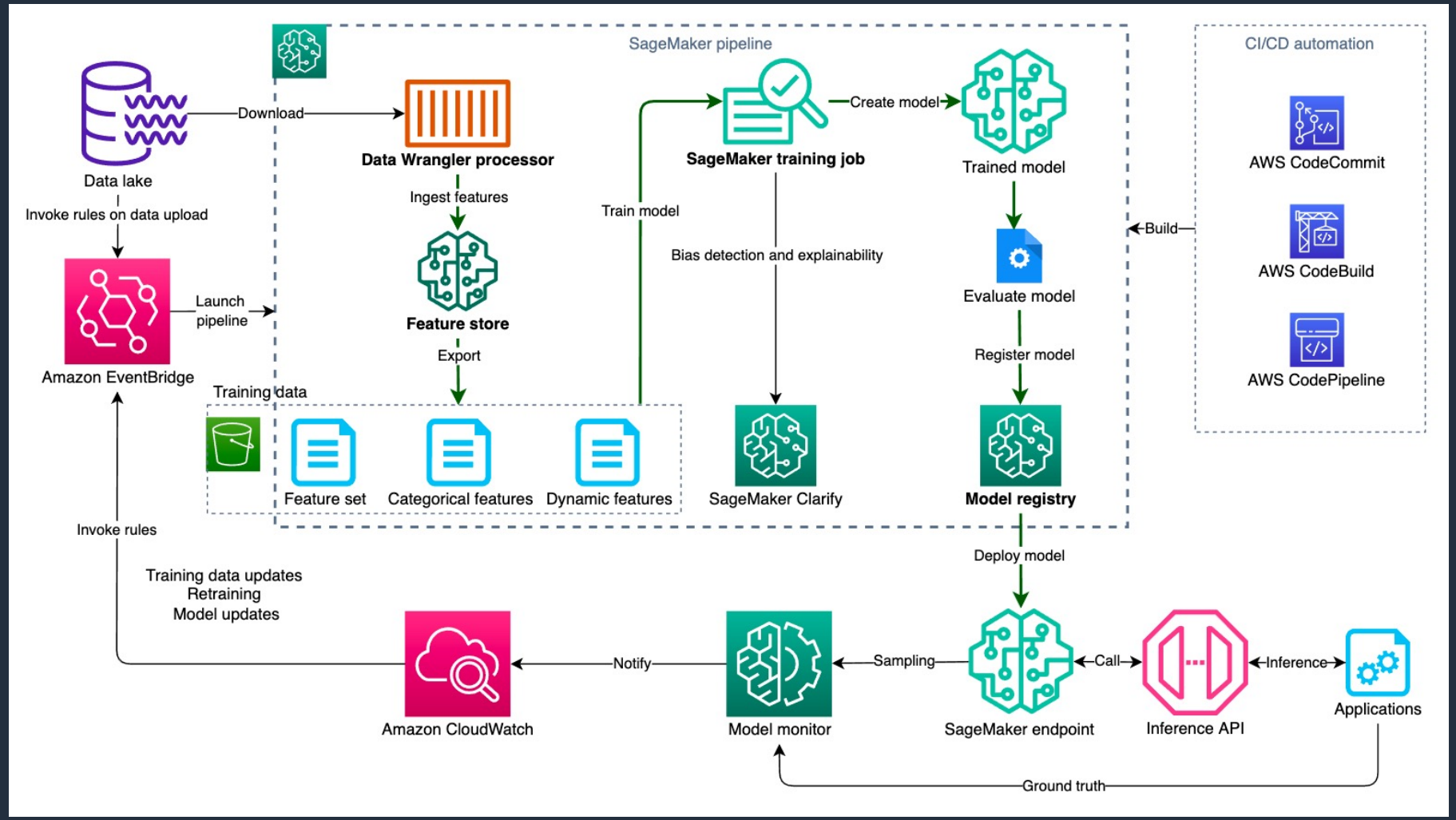

- Initial interactive data exploration, processing, and model training
- Experiment in a managed, elastic, and shareable Studio notebook
- Train a pre-built SageMaker algorithm on an compute instance outside the experimentation notebook
- User isolation in Studio: each Studio user has own dedicated resources
- One-click notebook sharing with other Studio users

# Step 2: Orchestrate and automate




- Agility and interaction of notebook
- Data processing with Data Wrangler or processing jobs
- Orchestration with AWS Step Functions and SageMaker Pipelines
- Store features in a centralized feature store

# Step 3: Move to production




- Production workflow
- Detect bias
- Model registry
- Model monitor
- Event-based workflows
- CI/CD automation

# Demo

# Productivity:
# Low code no code ML tools

# Amazon SageMaker low code no code tools

**Canvas**

…generate, use, and share ML models in a dedicated no-code workspace

**Autopilot**

…use AutoML to automatically build, train, and tune the best machine learning pipelines for your tabular datasets

**Data Wrangler**

…do exploratory data analysis, data preparation and feature engineering with a simple drag-and-drop UI

**JumpStart**

…use pre-trained state-of-the-art models like ResNet, Hugging Face BERT and GPT-2 for your Computer Vision, and Natural Language Processing

SageMaker Studio

Business leads

Domain Experts

Business Analysts

Data Engineers and Data Scientists

# Additional resources

Amazon SageMaker
Machine learning for every data scientist and developer

Amazon SageMaker Studio Lab
Learn and experiment with machine learning

Amazon SageMaker Studio Notebooks
Deep dive into Studio notebooks architecture

Why use Docker containers for ML development?
A case for considering using containers for machine learning development

# Thank you!

Yevgeniy Ilyin