Reliably deliver ML models at scale with Amazon SageMaker MLOps

MLOps Overview & Strategy

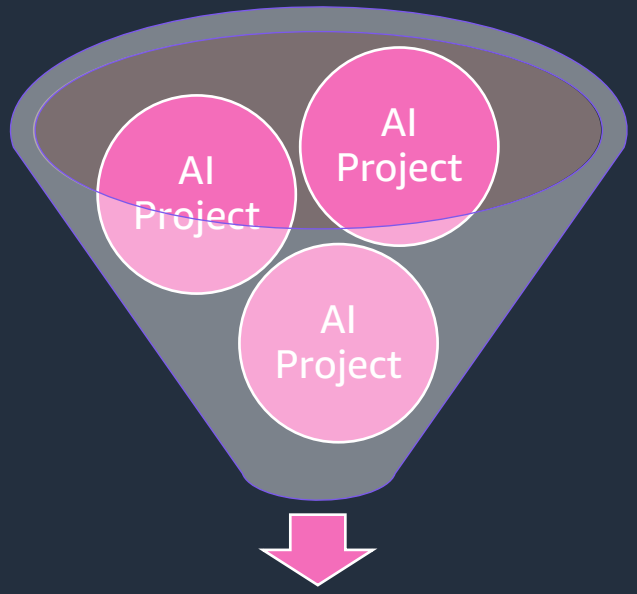
Shelbee Eigenbrode

Principal AI/ML Specialist Solutions Architect

MLOps Drivers

Creating a repeatable, reliable, and scalable path to production

Creating a repeatable path to Production



54%

of AI projects make it from pilot to production

Source: *Gartner AI in Organizations Survey, 2022*

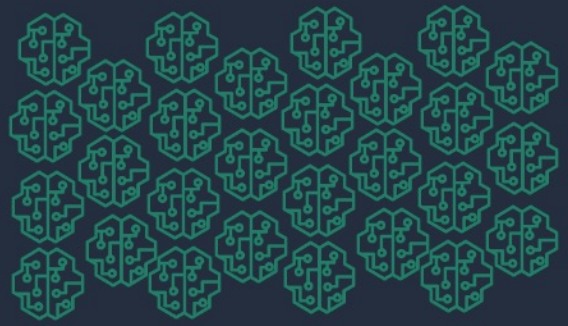
MLOps Drivers

Creating a repeatable, reliable, and scalable path to production

Reliably manage models at scale

Going From…

To…



MLOps is required to scale the adoption of ML

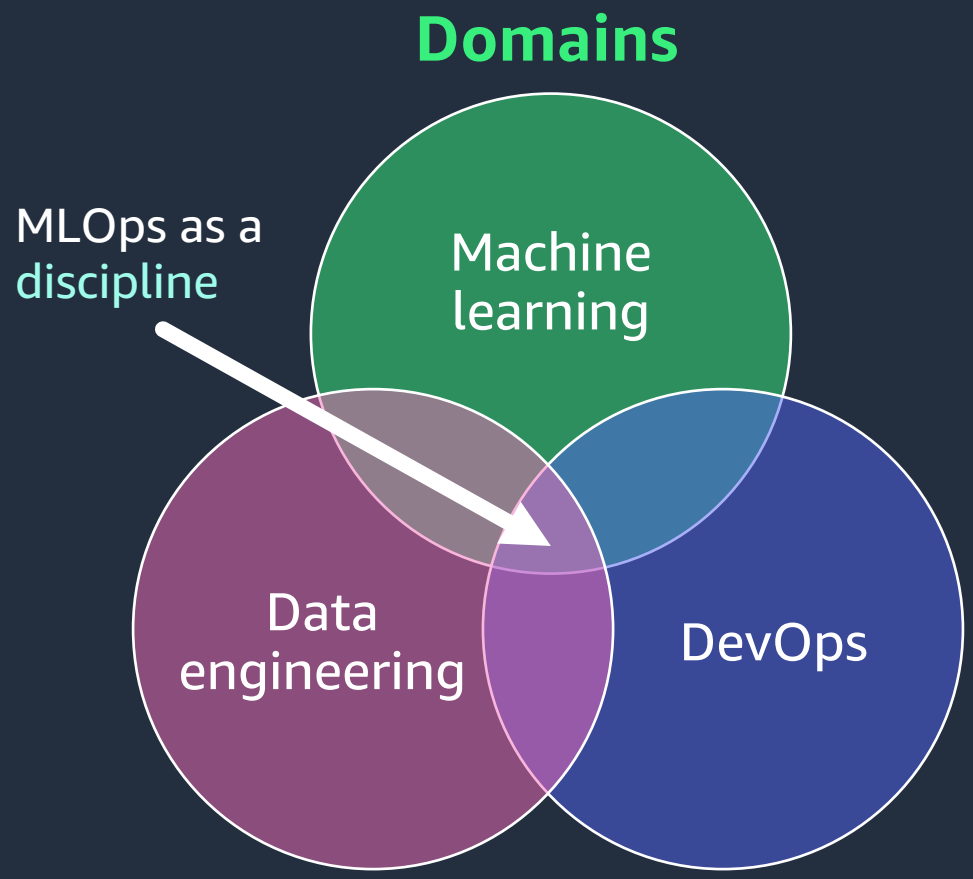
MLOps: Recommended → MLOps: Required

40%

of organizations reported having thousands, or hundreds of thousands of models deployed

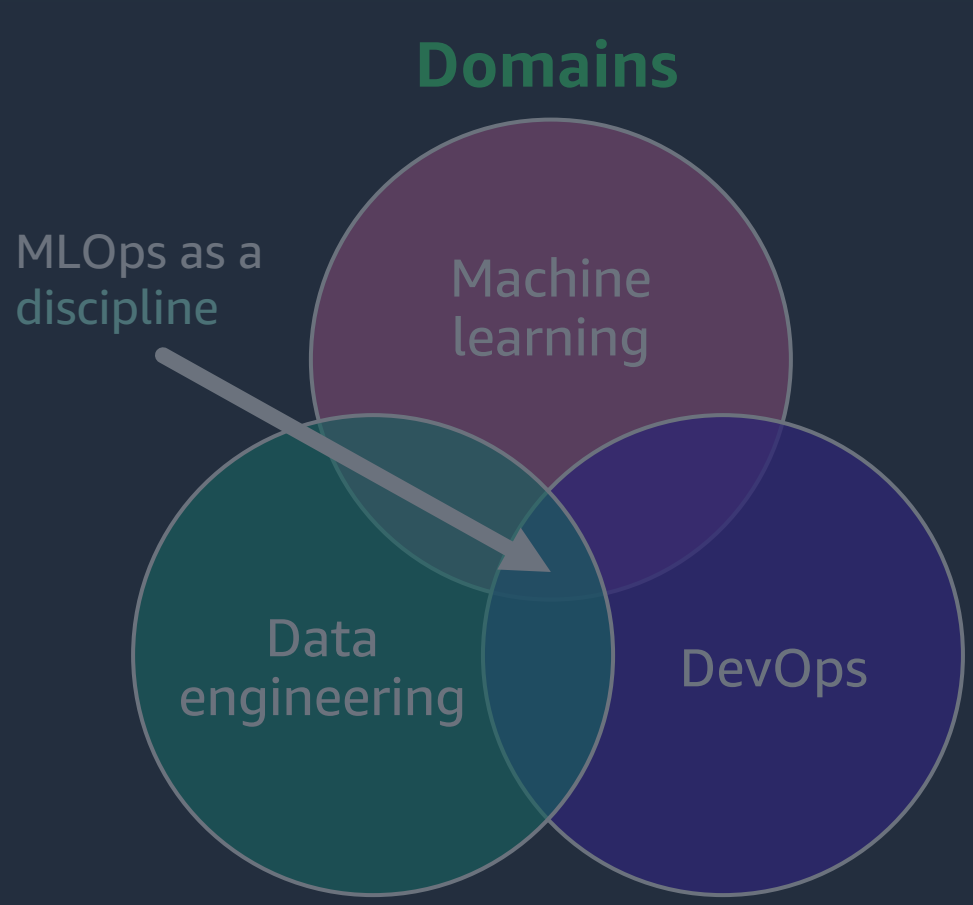
Source: Gartner AI in Organizations Survey, 2022

What is MLOps?
It’s not just technology

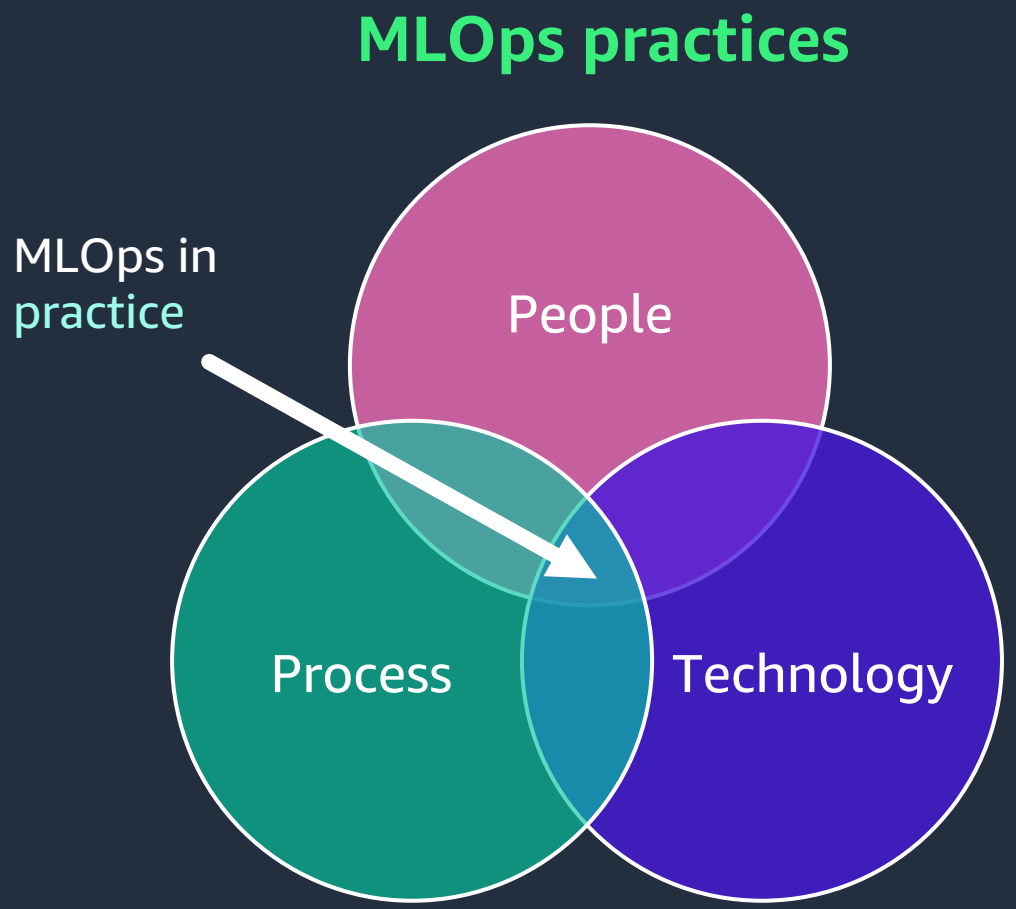


MLOps is a discipline that sits at the intersection of the domains of data science, data engineering, and DevOps.

What is MLOps?
It's not just technology



MLOps is a discipline that sits at the intersection of the domains of data science, data engineering, and DevOps.



MLOps as a discipline is enabled by a core set of practices that span People, Process, and Technology across the Machine Learning Development Lifecycle.

MLOps vs DevOps

MLOps builds on DevOps practices with unique considerations for ML

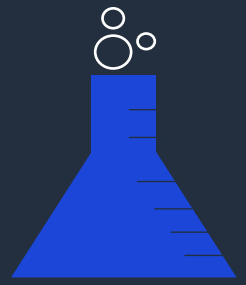
Feature	DevOps	MLOps
Code versioning	✅	✅
Artifact versioning	✅	✅
Continuous integration/continuous delivery (CI/CD)	✅	✅
Continuous Monitoring	✅	✅
Data lineage, governance, & management		✅
Experiment tracking		✅
Model lineage, governance, & management		✅
Model re-training strategies		✅
Model build & deployment workflows		✅

MLOps vs DevOps

Machine Learning Development Lifecycle != Software Development Lifecycle

1

Experimentation



Model Development is unique…

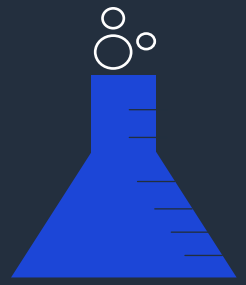
- *Requires a data strategy that includes ML & often requires access to larger compute/storage needs*
- *Enable rapid experimentation while balancing the needs of IT*
- *Experiment tracking is key for model development activities*

MLOps vs DevOps

Machine Learning Development Lifecycle != Software Development Lifecycle

1

Experimentation

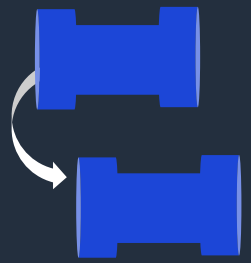


Model Development is unique…

- *Requires a data strategy that includes ML & often requires access to larger compute/storage needs*
- *Enable rapid experimentation while balancing the needs of IT*
- *Experiment tracking is key for model development activities*

2

Pipelines



Multiple disparate pipelines…

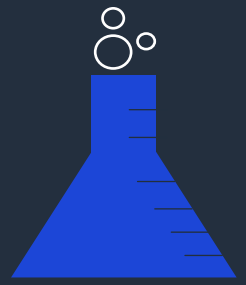
- *Not always built during model development*
- *Pipelines can have different lifecycles, dependencies, and triggers*
- *End-to-end traceability expands in scope*

MLOps vs DevOps

Machine Learning Development Lifecycle != Software Development Lifecycle

1

Experimentation

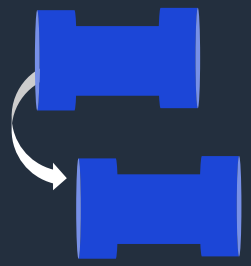


Model Development is unique…

- *Requires a data strategy that includes ML & often requires access to larger compute/storage needs*
- *Enable rapid experimentation while balancing the needs of IT*
- *Experiment tracking is key for model development activities*

2

Pipelines



Multiple disparate pipelines…

- *Not always built during model development*
- *Pipelines can have different lifecycles, dependencies, and triggers*
- *End-to-end traceability expands in scope*

3

Model Lineage & Governance

Expanded scope for CI/CD…

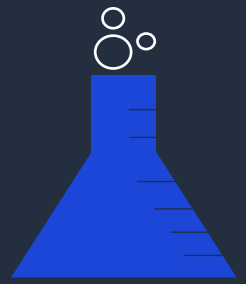
- *Source control includes multiple code sources*
- *Version control includes data, features, pipeline outputs and model artifacts*
- *Governance includes data related to model performance and explainability*

aws

MLOps vs DevOps

Machine Learning Development Lifecycle != Software Development Lifecycle

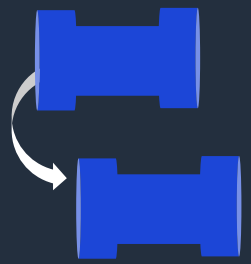
1 Experimentation



Model Development is unique...

- *Requires a data strategy that includes ML & often requires access to larger compute/storage needs*
- *Enable rapid experimentation while balancing the needs of IT*
- *Experiment tracking is key for model development activities*

2 Pipelines



Multiple disparate pipelines...

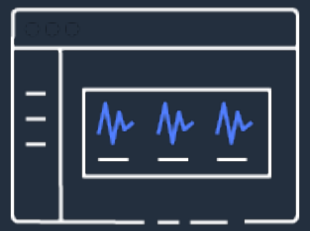
- *Not always built during model development*
- *Pipelines can have different lifecycles, dependencies, and triggers*
- *End-to-end traceability expands in scope*

3 Model Lineage & Governance

Expanded scope for CI/CD...

- *Source control includes multiple code sources*
- *Version control includes data, features, pipeline outputs and model artifacts*
- *Governance includes data related to model performance and explainability*

4 Monitoring



Monitoring includes models...

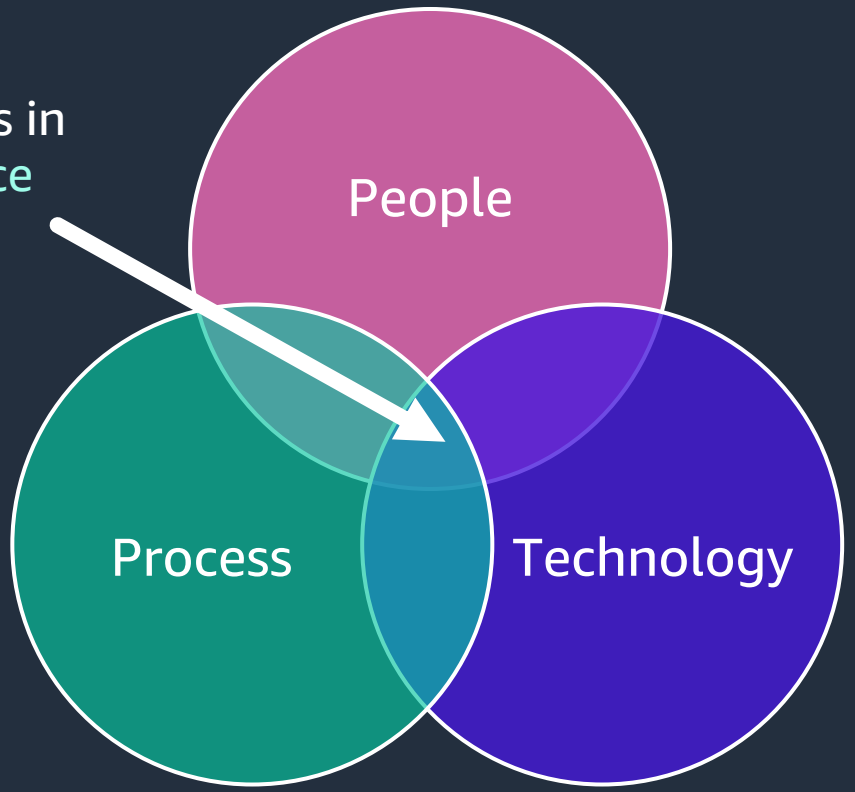
- *Continuous monitoring includes system PLUS model specific monitoring (data/concept drift)*
- *Interpreting/responding to alerts may require multiple personas*
- *Continuous retraining is not always the goal!*

aws

Why MLOps?

MLOps drivers

MLOps in practice



Economy
Improved time-to-market and reusability

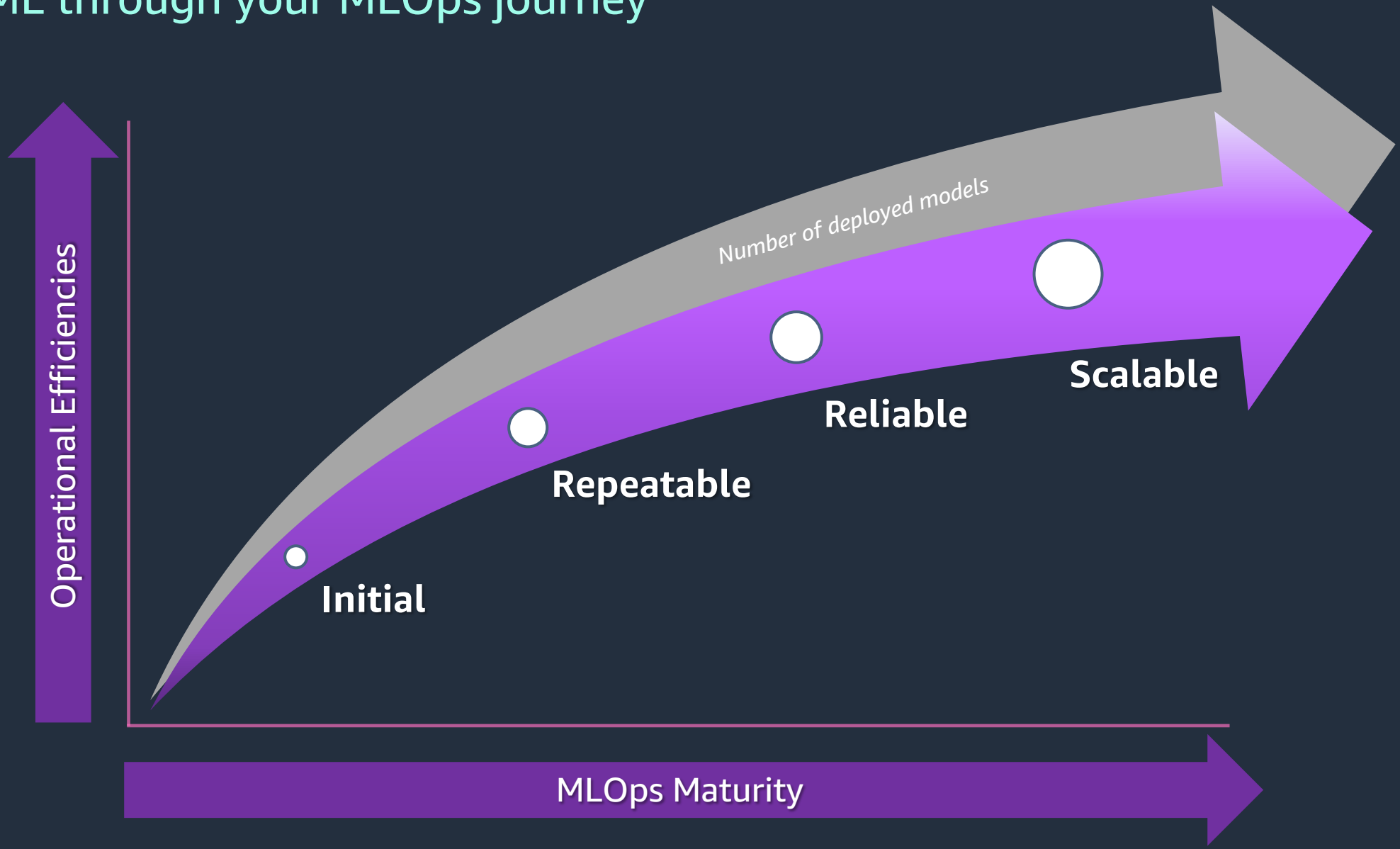
Agility
Faster iterations and increased repeatability

Quality
Increased quality, reliability, and transparency

Scale
Ability to efficiently build, deploy and manage models at scale

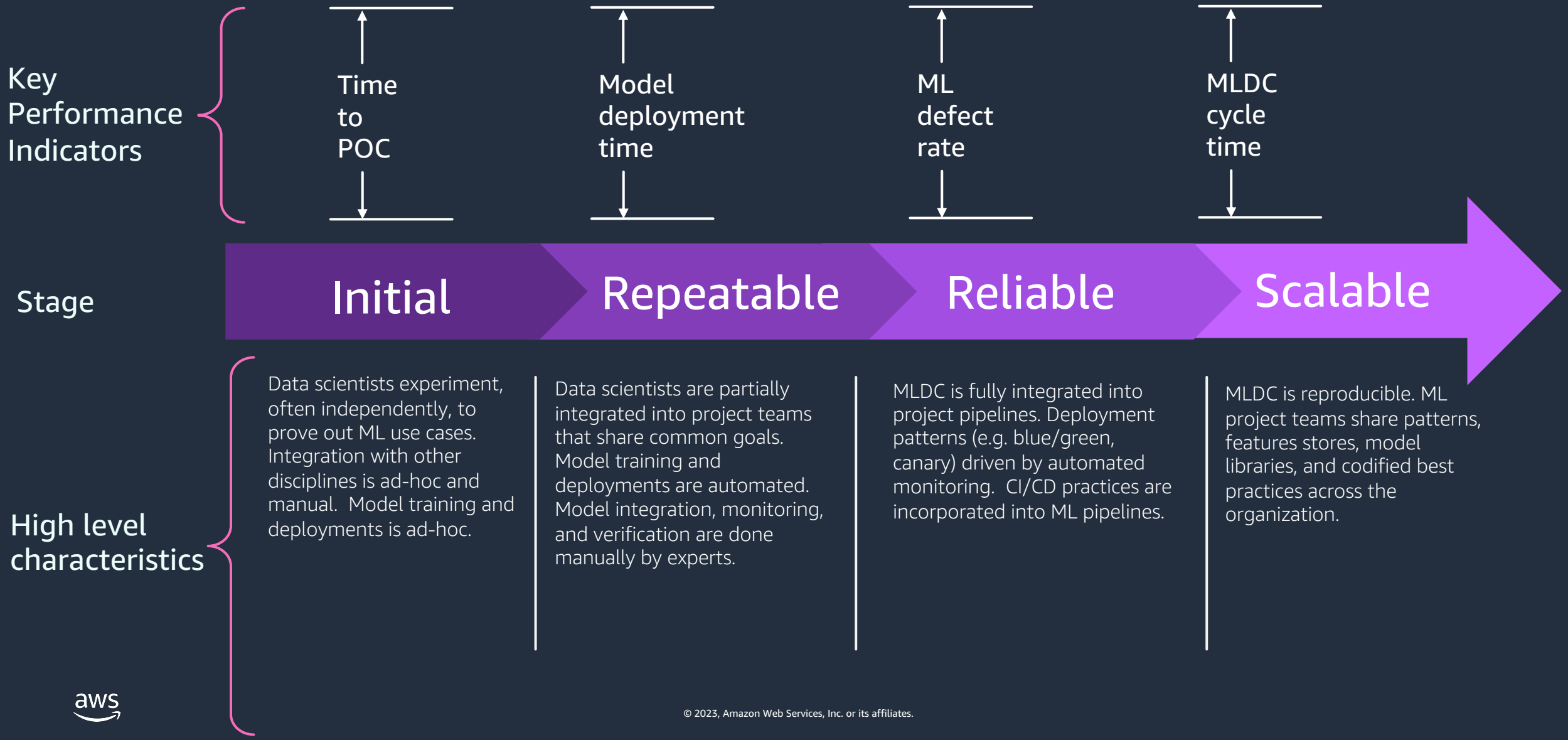
MLOps Maturity Framework

Scaling ML through your MLOps journey



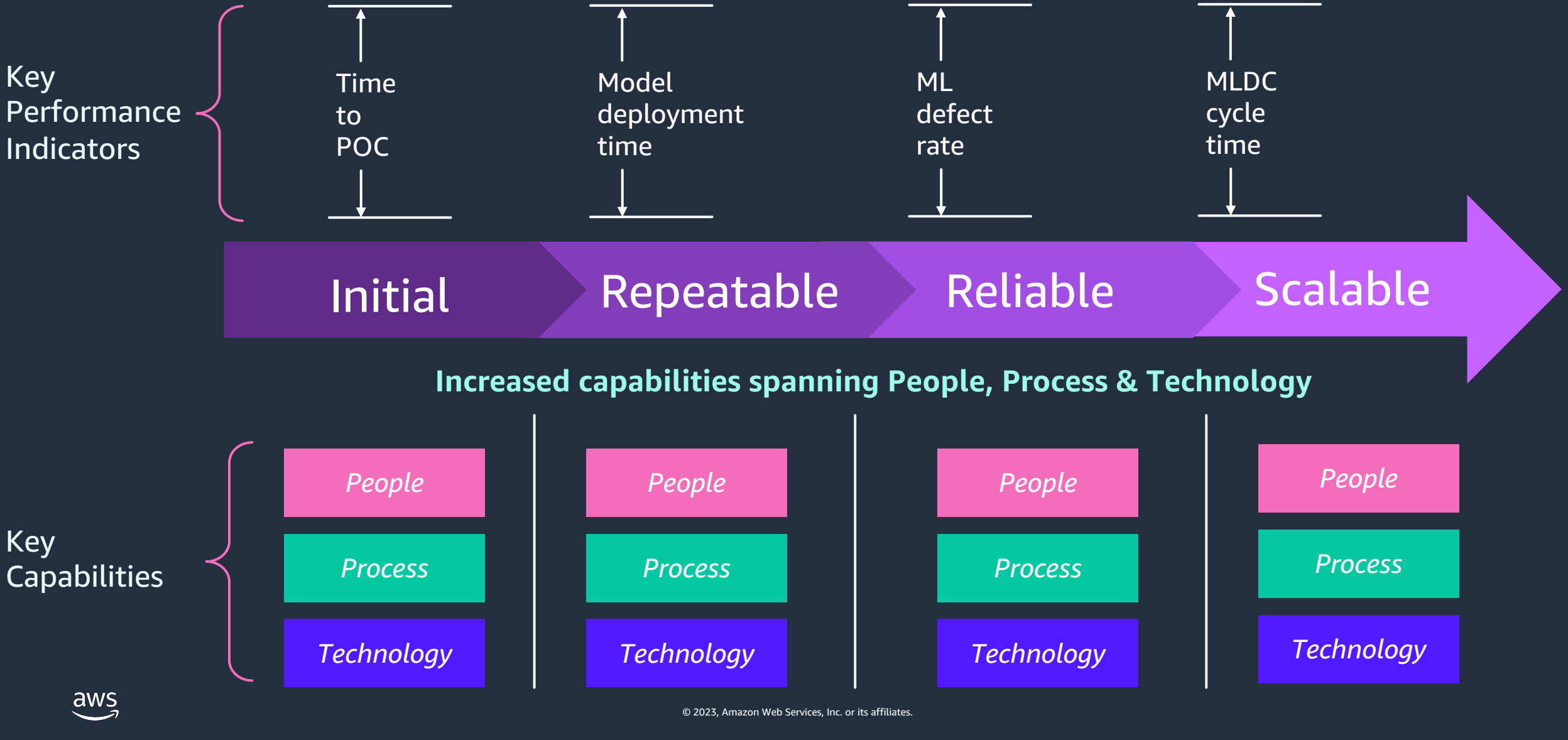
MLOps Maturity Framework

Scaling ML through your MLOps journey



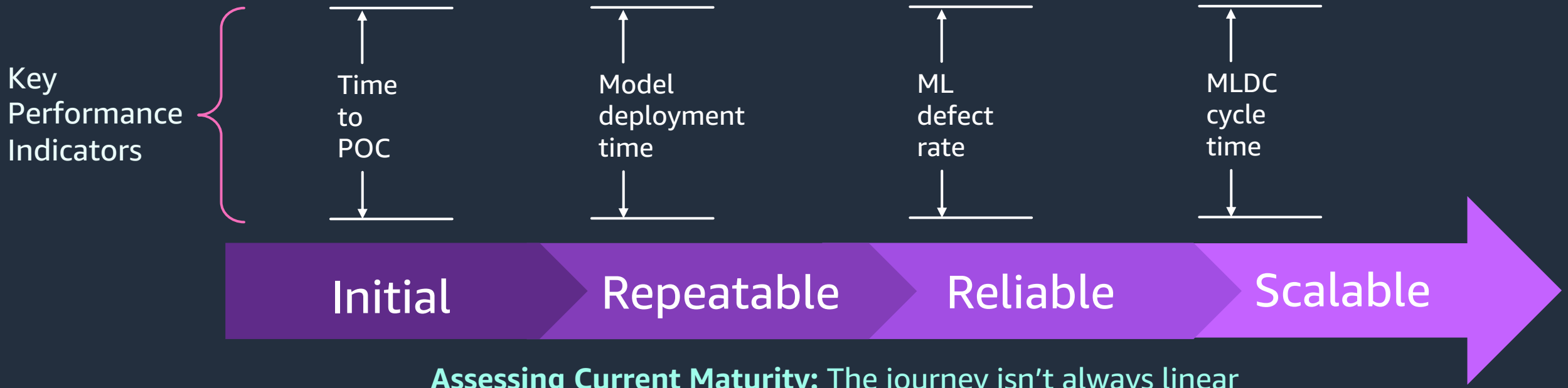
MLOps Maturity Framework

Assessing MLOps maturity

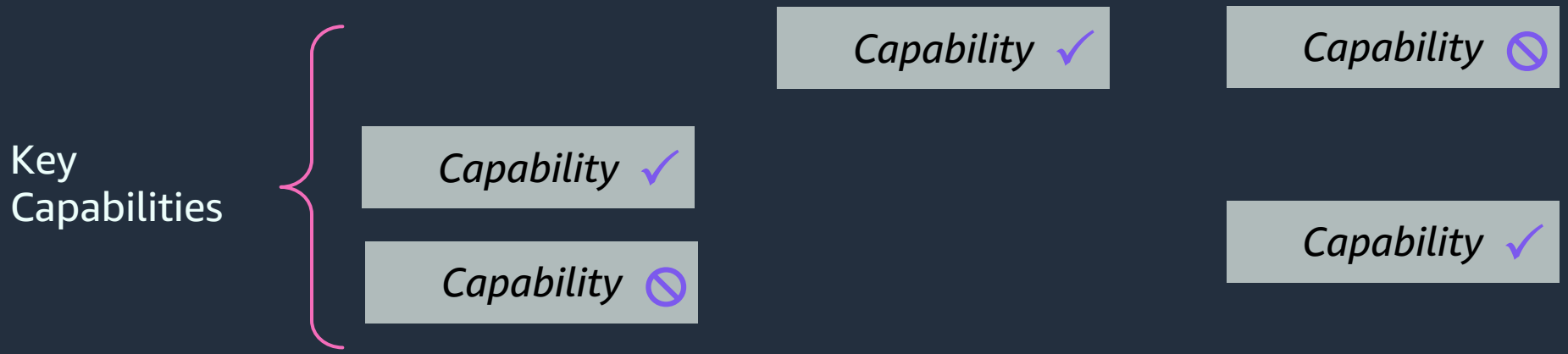


MLOps Maturity Framework

Assessing MLOps maturity

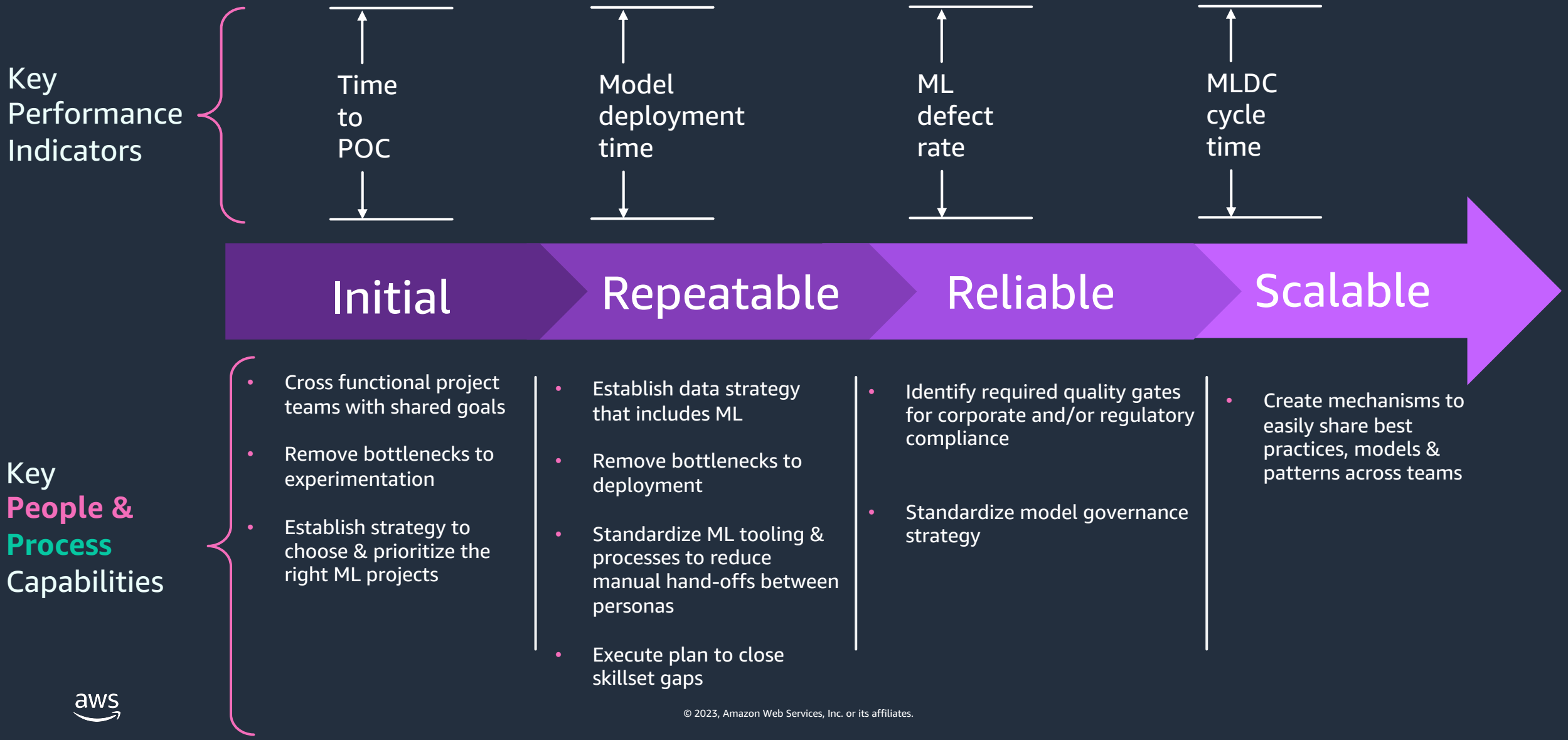


Assessing Current Maturity: The journey isn't always linear



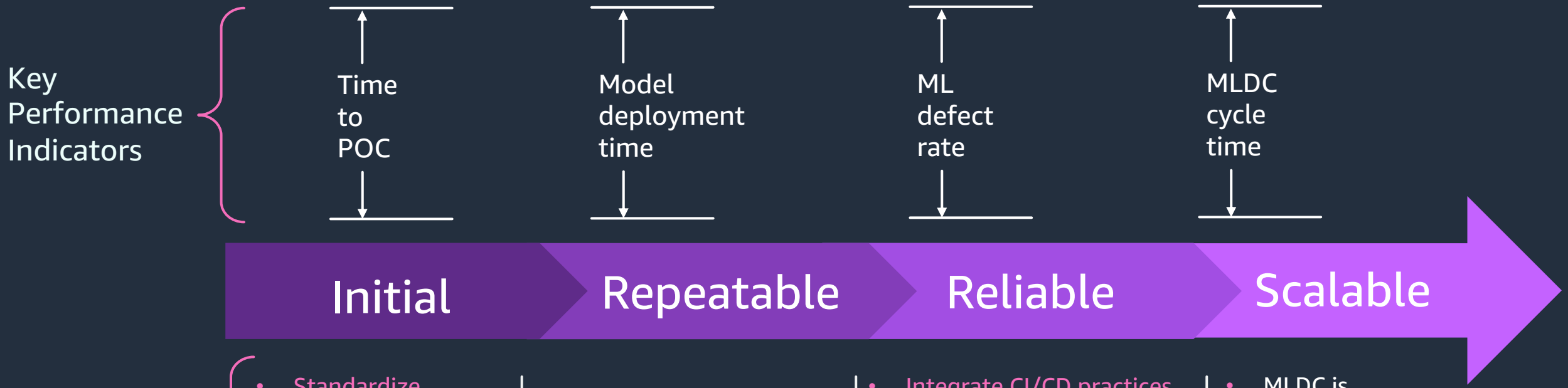
MLOps Maturity Framework

Scaling ML through your MLOps journey



MLOps Maturity Framework

Scaling ML through your MLOps journey



Key **Technology** Capabilities

Initial

- Standardize experimentation environments & processes
- Automate governed access to resources
- Track experiments
- Centralize management of feature data

Repeatable

- Automate model build, and deployment tasks
- Centralize management of models
- Standardize source code & artifact repositories ➔

Reliable

- Integrate CI/CD practices into ML pipelines for traceability, improved quality and lineage tracking
- Incorporate automated quality gates
- Automate continuous monitoring with feedback loops & re-training

Scalable

- MLDC is reproducible through standardized tooling & templates shared across teams

aws

Implementing MLOps Practices using Amazon SageMaker

Today's Focus

PREPARE DATA AND BUILD, TRAIN, DEPLOY AND MANAGE ML MODELS FOR ANY USE CASE

PREPARE

Geospatial
Visualize geospatial data

Ground Truth
Create high quality datasets for ML

Data Wrangler
Aggregate and prepare data for ML

Processing
Built-in Python, BYO R/Spark

Feature Store
Store, catalog, search, and reuse features

Clarify
Detect bias and understand model predictions

BUILD

Studio Notebooks & Notebook Instances
Fully managed Jupyter notebooks with elastic compute

Studio Lab
Free ML development environment

Built-in Algorithms
Integrated tabular, NLP, and vision algorithms

JumpStart
UI based discovery, training, and deployment of models, solutions, and examples

Autopilot
Automatically create ML models with full visibility

Bring Your Own
Bring your own container and algorithms

Local Mode
Test and prototype on your local machine

TRAIN & TUNE

Fully Managed Training
Broad hardware options, easy to setup and scale

Distributed Training Libraries
High performance training for large datasets and models

Training Compiler
Faster deep learning model training

Automatic Model Tuning
Hyperparameter optimization

Managed Spot Training
Reduce training cost by up to 90%

Debugger and Profiler
Debug and profile training runs

Experiments
Track, visualize, and share model artifacts across teams

Customization Support
Integrate with popular open source frameworks and libraries

DEPLOY & MANAGE

Fully Managed Deployment
Ultra low latency, high throughput inference

Real-Time Inference
For steady traffic patterns

Serverless Inference
For intermittent traffic patterns

Asynchronous Inference
For large payloads or long processing times

Batch Transform
For offline inference on batches of large datasets

Multi-Model Endpoints
Reduce cost by hosting multiple models per instance

Multi-Container Endpoints
Reduce cost by hosting multiple containers per instance

Shadow Testing
Validate model performance in production

Inference Recommender
Automatically select compute instance and configuration

Model Monitor
Maintain accuracy of deployed models

Kubernetes Operators & Components
Manage and monitor models on edge devices

Edge Manager
Manage and monitor models on edge devices

MLOps: Pipelines | Projects | Model Registry
Workflow automation, CI/CD for ML, central model catalog

Canvas
Generate accurate machine learning predictions—no code required

Studio | RStudio
Integrated development environment (IDE) for ML

Governance
Model Cards | Dashboard | Permissions

Implementing MLOps Practices using Amazon SageMaker

SageMaker Benefits

Develop Candidate Model(s)

Business Goals & ML Problem | Provision Environments | Perform Experiments

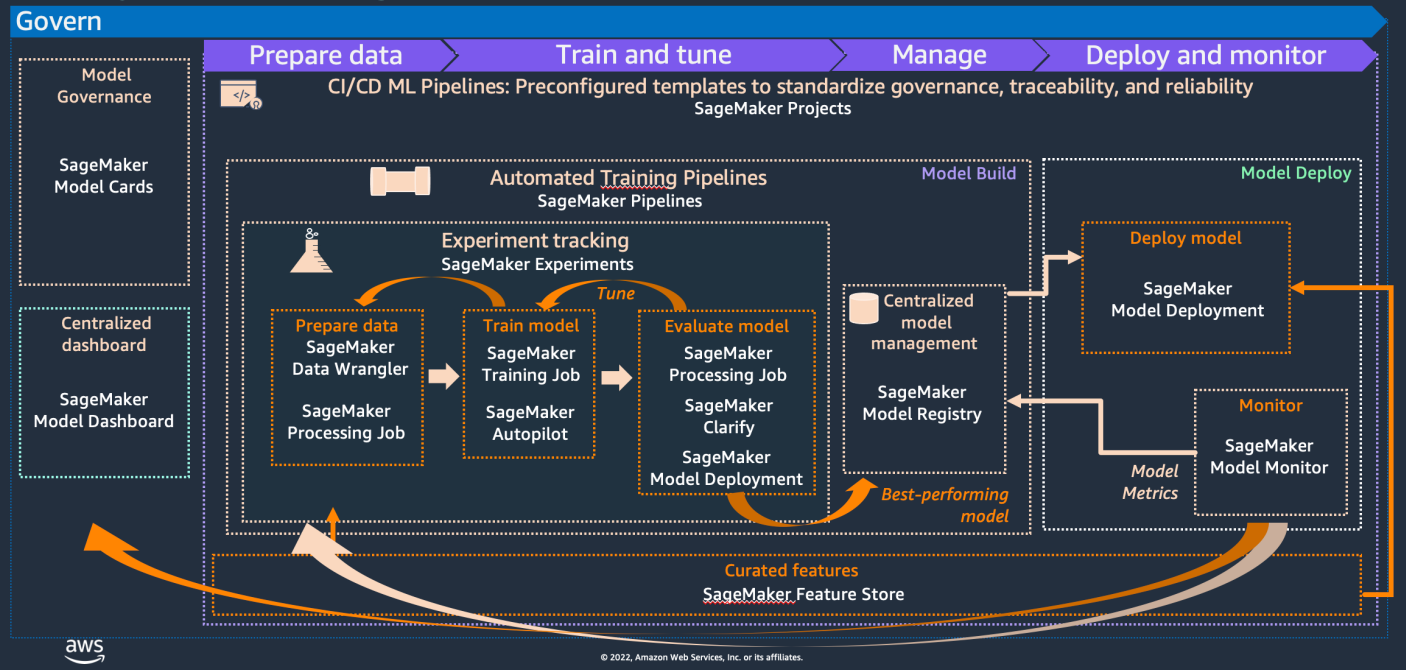
Deploy Candidate Model(s)

Feedback and Retraining

Automate Model Training | Package and Test Models | Deploy and Monitor

MLOps Maturity Framework in Practice

ML Project Phases – SageMaker View

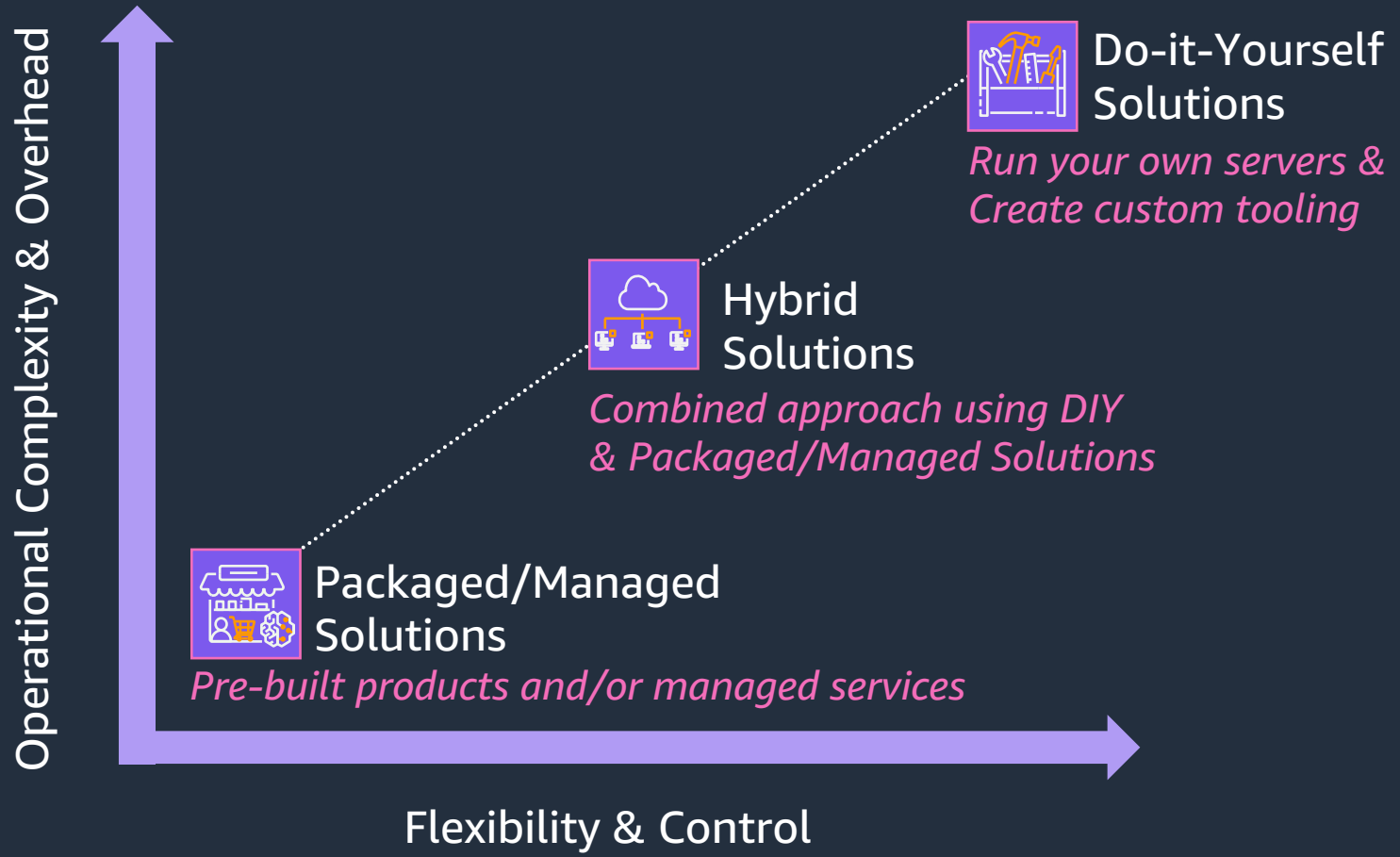


- End-to-End integrated features
- No servers to manage for your MLOps tooling
- Flexible options across ML use cases
- Purpose built for Machine Learning

aws

MLOps Technical Implementation Choices

Understand Your Tradeoffs



1 Start With Process

Key Lessons Learned

Key Lessons Learned

1. Start With Process
2. You Can’t Scale With People Issues

Key Lessons Learned

1. Start With Process
2. You Can’t Scale With People Issues
3. Technical Implementation(s) – One Size Does Not Fit All

Key Lessons Learned

1. Start With Process
2. You Can’t Scale With People Issues
3. Technical Implementation(s) – One Size Does Not Fit All
4. Standardize on Managed Services & Standard Components

Key Lessons Learned

1. Start With Process
2. You Can’t Scale With People Issues
3. Technical Implementation(s) – One Size Does Not Fit All
4. Standardize on Managed Services & Standard Components
5. Don’t Ignore Unique Aspects of the ML Development Lifecycle

Key Lessons Learned

1. Start With Process
2. You Can’t Scale With People Issues
3. Technical Implementation(s) – One Size Does Not Fit All
4. Standardize on Managed Services & Standard Components
5. Don’t Ignore Unique Aspects of the ML Development Lifecycle
6. Start Small & Iterate

Today's Agenda

Time	Type	Title
9:00 – 9:30	Presentation	MLOps Overview & Strategy
9:30 – 10:15	Presentation	Model Development & Experimentation
10:15 – 10:45	Lab	Utilizing SageMaker Experiments during model development
10:45 – 11:00	Break	
11:00 – 11:30	Presentation	Implementing repeatable mechanisms to automate and govern your model build and model deployment tasks
11:30 – 12:15	Lab	Automate model build process & manage model versions at scale
12:15 – 1:00	LUNCH	
1:00 – 1:30	Partner Session	
1:30 – 2:00	Service Team Panel	
2:00 – 2:45	Presentation	Implementing reliable mechanisms to deliver, monitor, and manage models
2:45 – 3:30	Lab	Reliably deliver and manage models at scale
3:30 – 3:45	Break	
3:45 – 4:00	Presentation	SageMaker Third-Party Integrations
4:00 – 4:45	Presentation	ML Platform Deep Dive
4:45 – 5:00	Closeout & Next Steps	

Initial Stage:

Model Development & Experimentation

MLOps Maturity Framework in Practice

Initial: Development & Experimentation with manual workflows

Key Performance Indicators

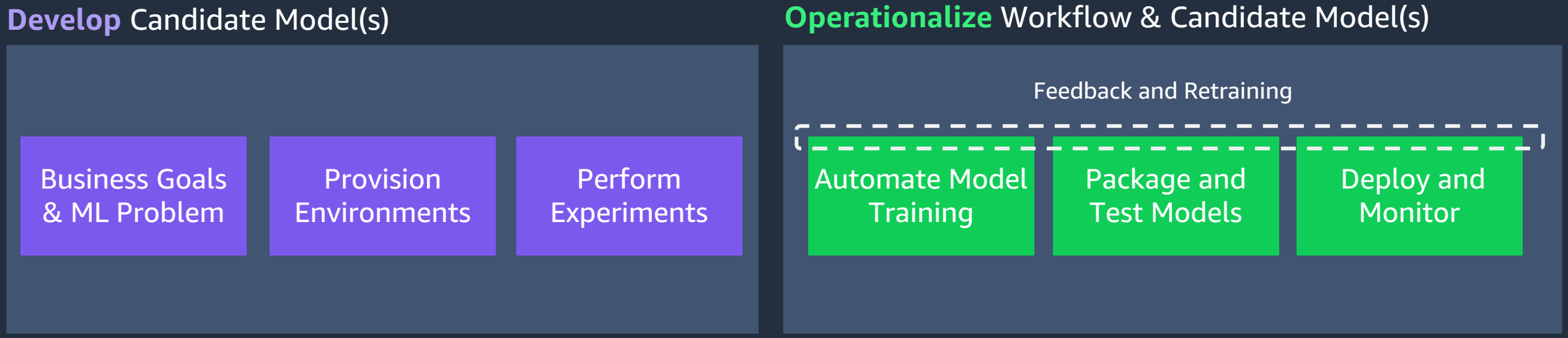
Initial

Key **Technology** Capabilities

- Standardize experimentation environments & processes
- Automate governed access to resources
- Track experiments
- Centralize management of feature data

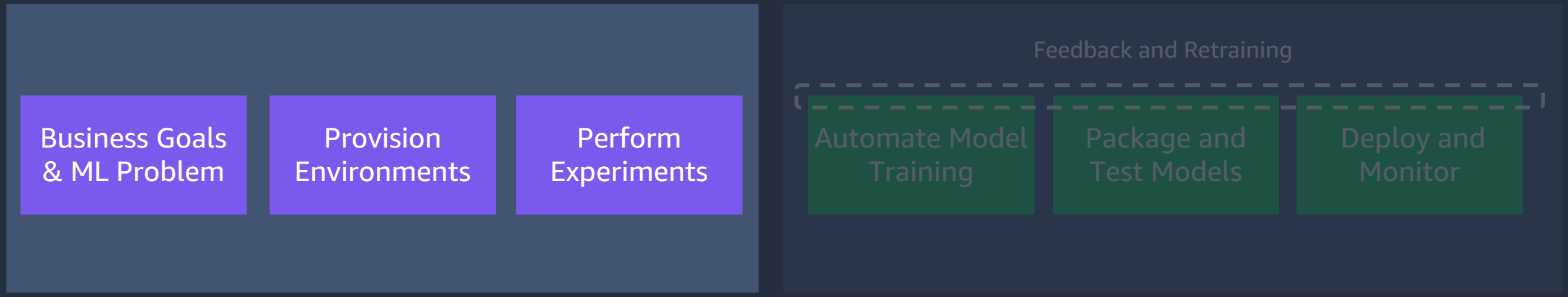
MLOps Maturity Framework in Practice

Initial: Development & Experimentation with manual workflows



MLOps Maturity Framework in Practice

Initial: Development & Experimentation with manual workflows



Reduce time to Proof-of-Concept

Challenge #1: Enabling innovation while balancing IT needs

Data Scientist wants to prove value of a new use case

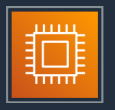
Perform iterative experimentation

- Understand data sources
- Try different data transformations
- Try different ML approaches
- Compare model performance

Development is mainly in a notebook or scripts

Access to model development resources

Data



Compute

Storage

Key Technical Challenge	• Enable innovation while balancing the needs of IT
Technical Solutions	• Provide access to governed data science environments with required resources using Infrastructure-as-Code (IaC), Configuration-as-Code (CaC), and Policy-as-Code (PaC)
On AWS	• Amazon SageMaker Role Manager • AWS Service Catalog

Enable Innovation while balancing the needs of IT

Initial: Standardize SageMaker Experimentation Environments

New!

Amazon SageMaker Role Manager

Key Benefits

Simplify permissions for ML activities

Guided workflows for role creation

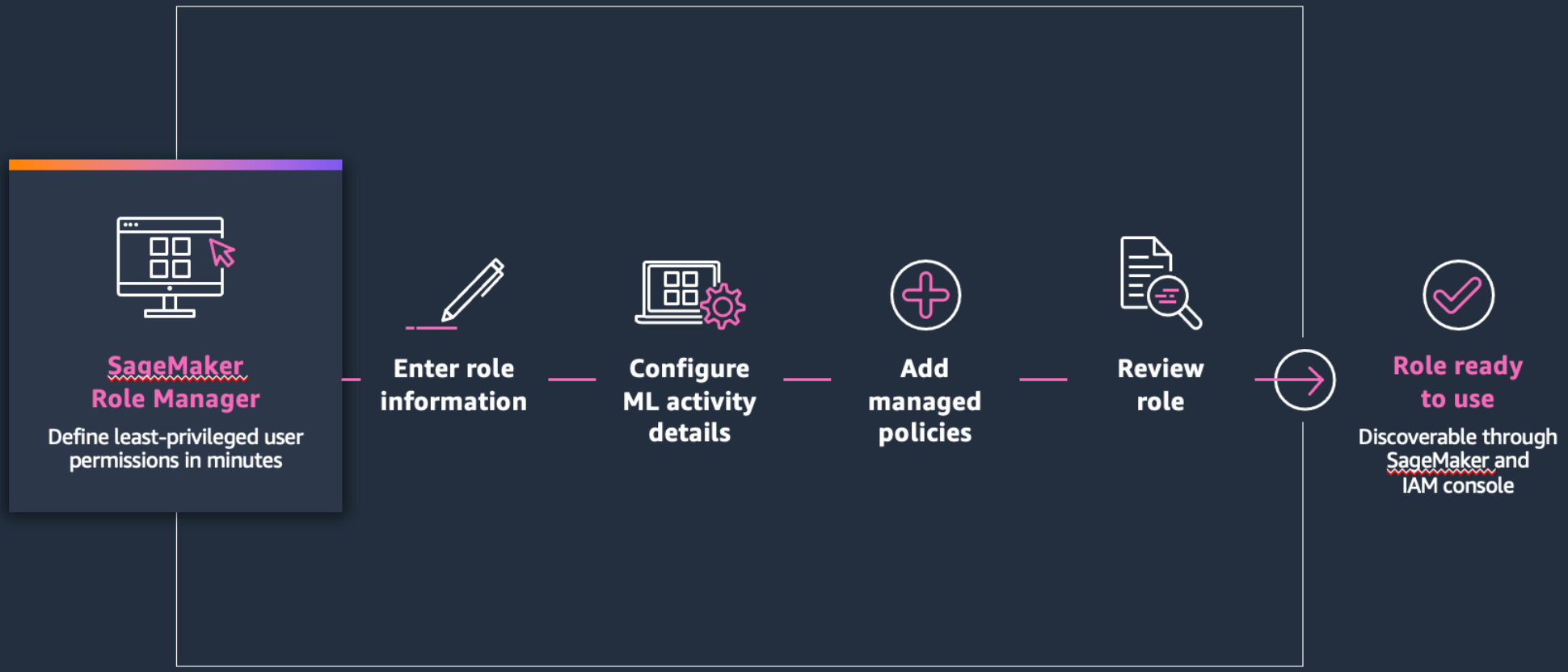
Accelerate user onboarding

Enable Innovation while balancing the needs of IT

Initial: Standardize SageMaker Experimentation Environments

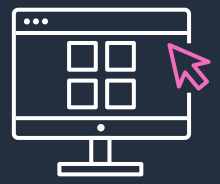
New!

Amazon SageMaker Role Manager



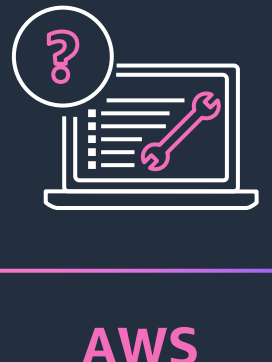
Enable Innovation while balancing the needs of IT

Initial: Standardize SageMaker Experimentation Environments



Amazon SageMaker Role Manager

Define least-privileged user permissions in minutes

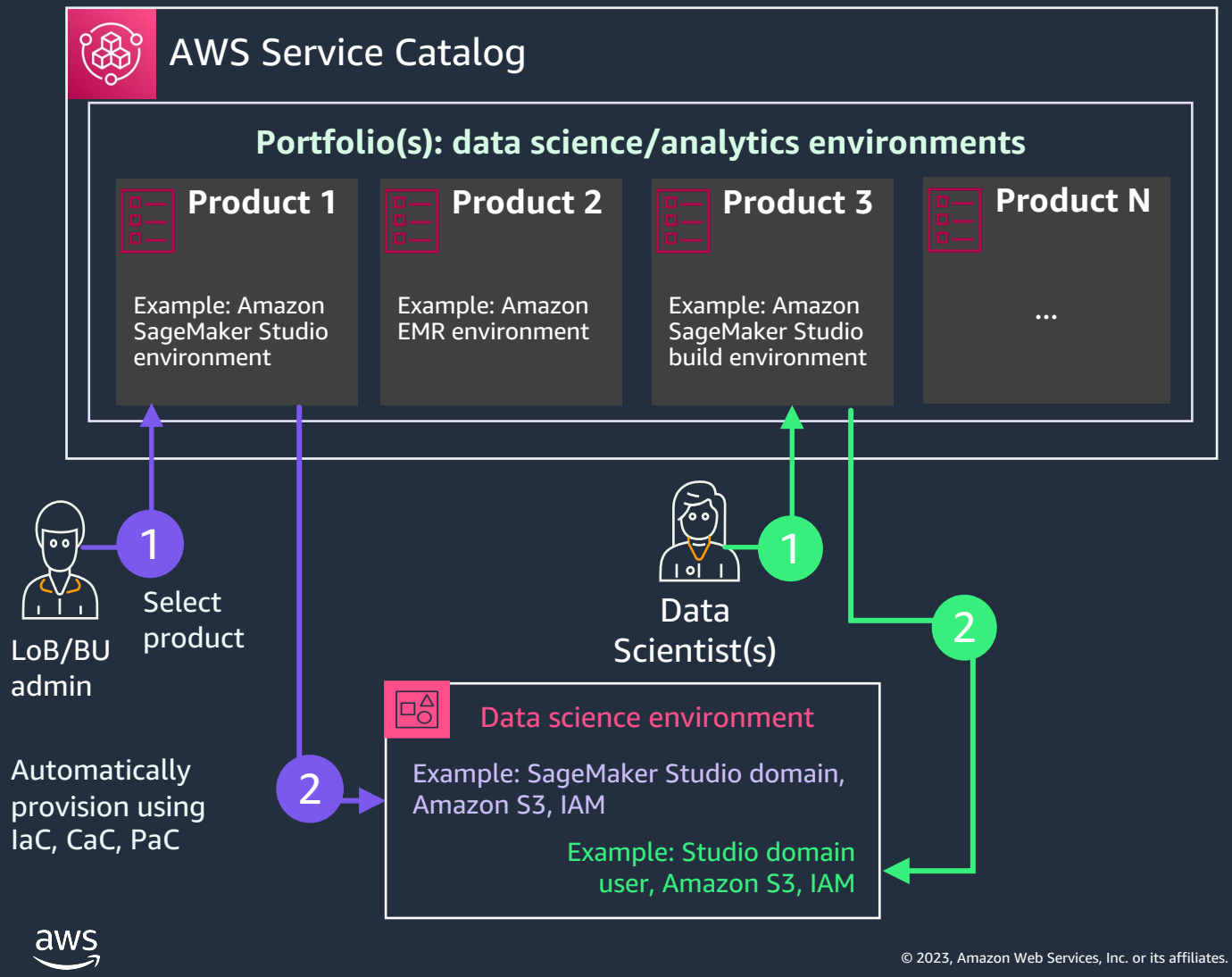


AWS Service Catalog

Standardize resource configuration

Enable Innovation while balancing the needs of IT

Initial: Standardize SageMaker Experimentation Environments



On AWS

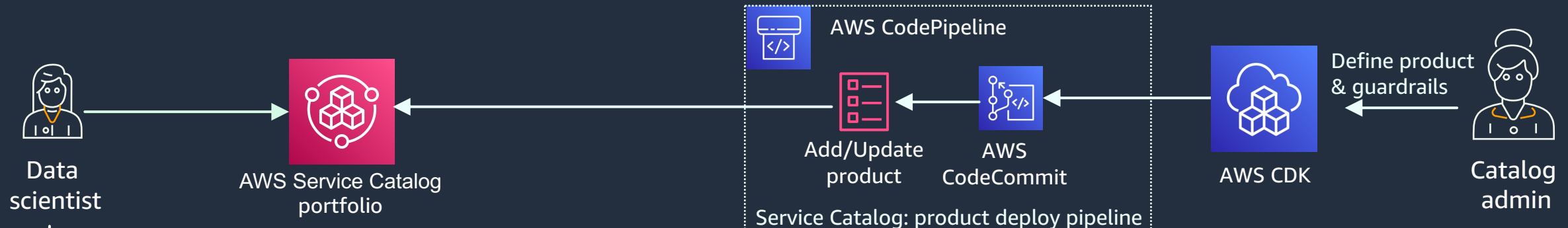
Use AWS Service Catalog to standardize the set up and provisioning of model build environments using IaC, CaC, PaC

Benefits

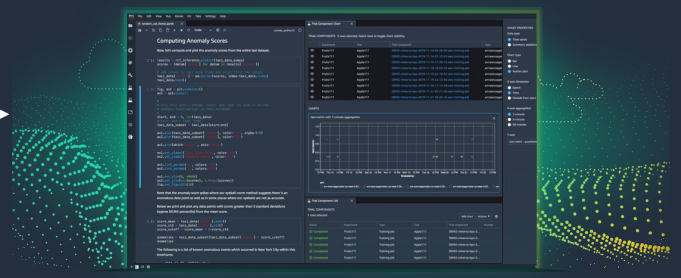
- Reduce bottlenecks to innovation
- Ensure repeatability and consistency
- Preconfigure security properties
- Implement guardrails via code

Enable Innovation while balancing the needs of IT

Initial: Standardize SageMaker Experimentation Environments



Consumption of product



Amazon SageMaker Studio

Underlying infrastructure is abstracted from data scientist

MLOps Maturity Framework in Practice

Initial Phase Goal: Standardize environments & experiment tracking

Govern

Prepare data | Train and tune | Manage | Deploy and monitor

Environment Setup

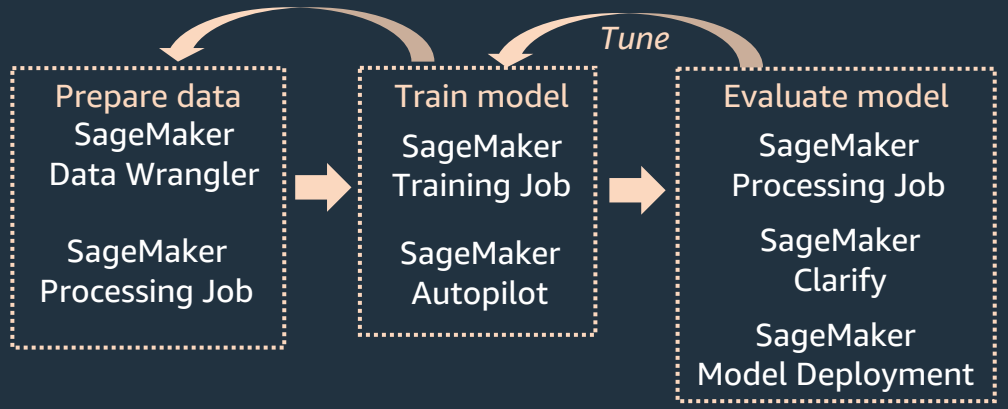
SageMaker Role Manager

AWS Service Catalog

Model Build

Model Development

SageMaker Studio



Reduce time to Proof-of-Concept

Challenge #2: Reduce feature engineering cycles & Increase consistency

Data Scientist wants to prove value of a new use case

Perform iterative experimentation

- Understand data sources
- Try different data transformations
- Try different ML approaches
- Compare model performance

Development is mainly in a notebook or scripts

Access to discover & use curated features for model building

Feature Data

Key Technical Challenge

- Redundant & inconsistent feature engineering

Technical Solutions

- Implement a feature store for discovery, reuse and consistency

On AWS

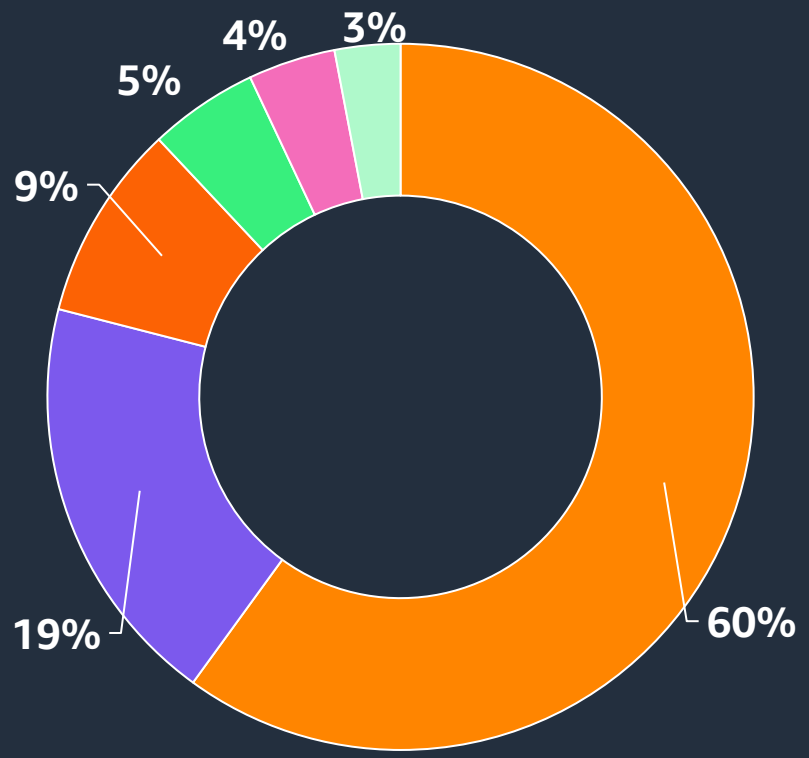
- Amazon SageMaker Feature Store

aws

Feature discovery, reuse, and consistency

Initial: Standardize Centralized feature stores

60% of time spent on data preparation



What data scientists spend the most time doing

- Cleaning and organizing data
- Collecting data sets
- Mining data for patterns
- Other
- Refining algorithms
- Building training sets

Source: Forbes survey of 80 data scientists, March 2016

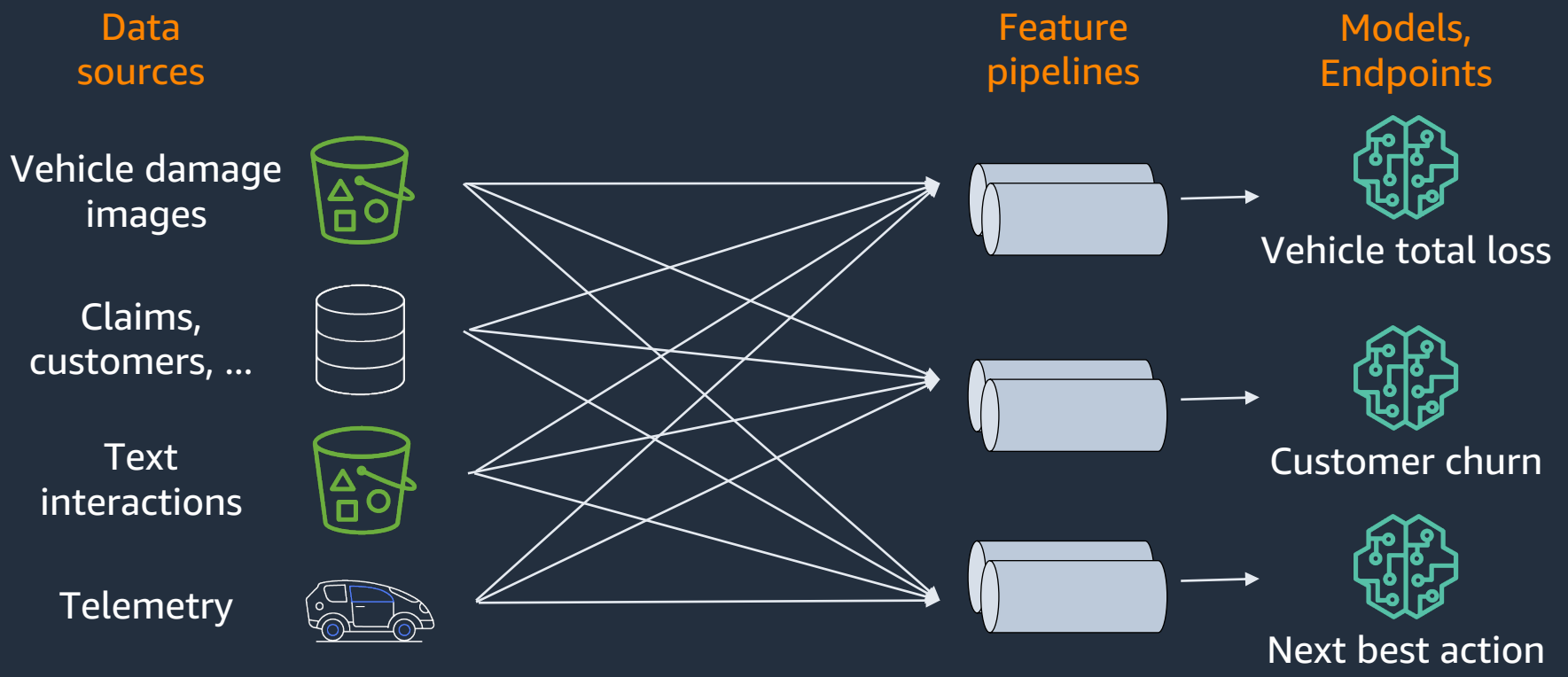
aws

Feature discovery, reuse, and consistency

Initial: Standardize Centralized feature stores

. . . and teams too often start from scratch

Standalone feature engineering for each new model



Challenges

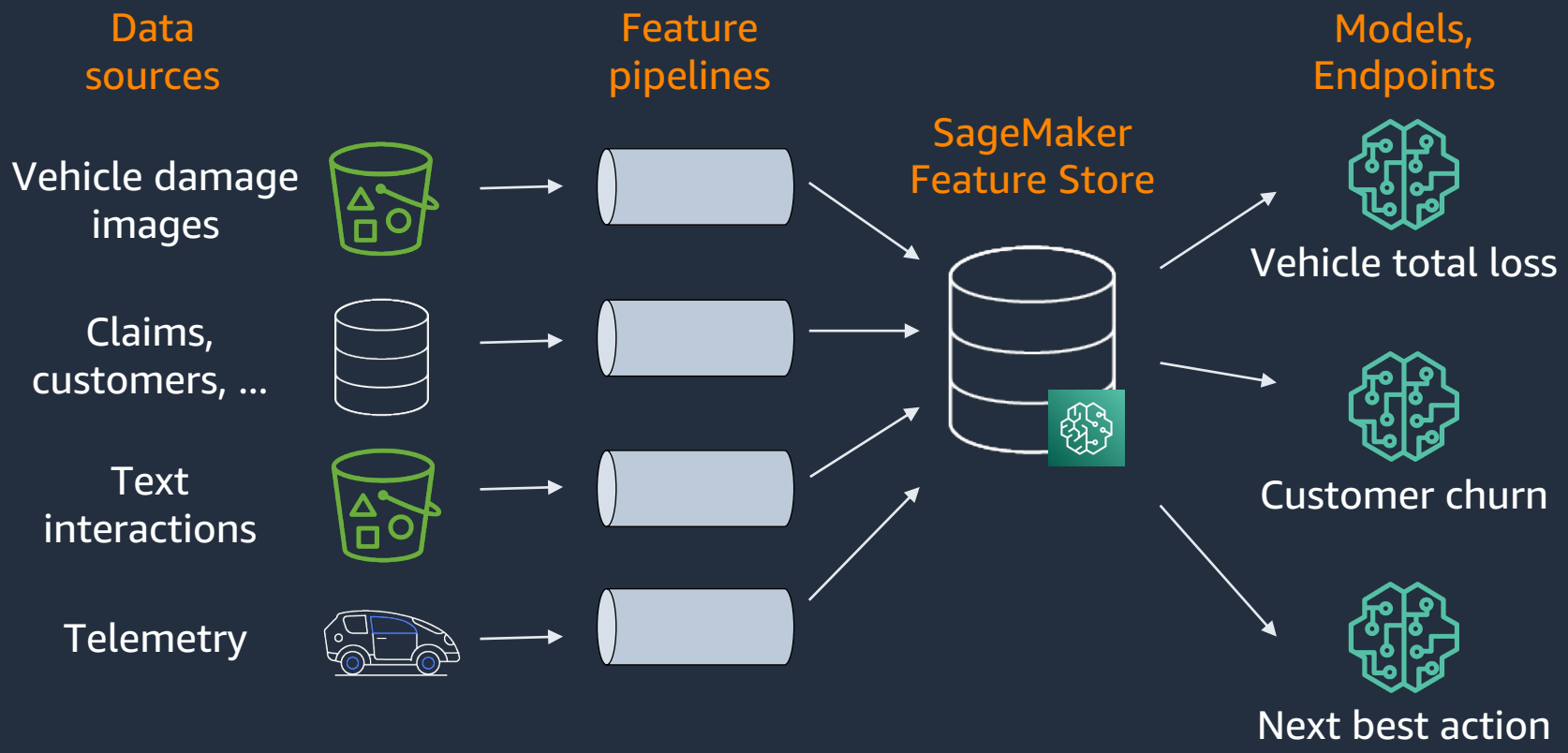
- Feature duplication
- Slow time to market
- Inaccurate predictions

Feature discovery, reuse, and consistency

Initial: Standardize Centralized feature stores

With SageMaker Feature Store...

Build features once, reuse them across teams and models



Benefits

- Feature reuse
- Reproducible features
- Accurate training datasets
- Low latency inference
- Consistent features for training and inference

SageMaker Feature Store

Accelerate machine learning with a feature store

Batch and streaming ingestion

High throughput writes for ingesting features

Online and offline features

Online features for real-time prediction, and offline features for historical data for model training and batch prediction

Feature metadata and data cataloging

Store metadata for features and leverage automatic data cataloging to easily query and extract feature data

Feature discovery and reuse

Search for features to reuse, before starting new development

Security and access control

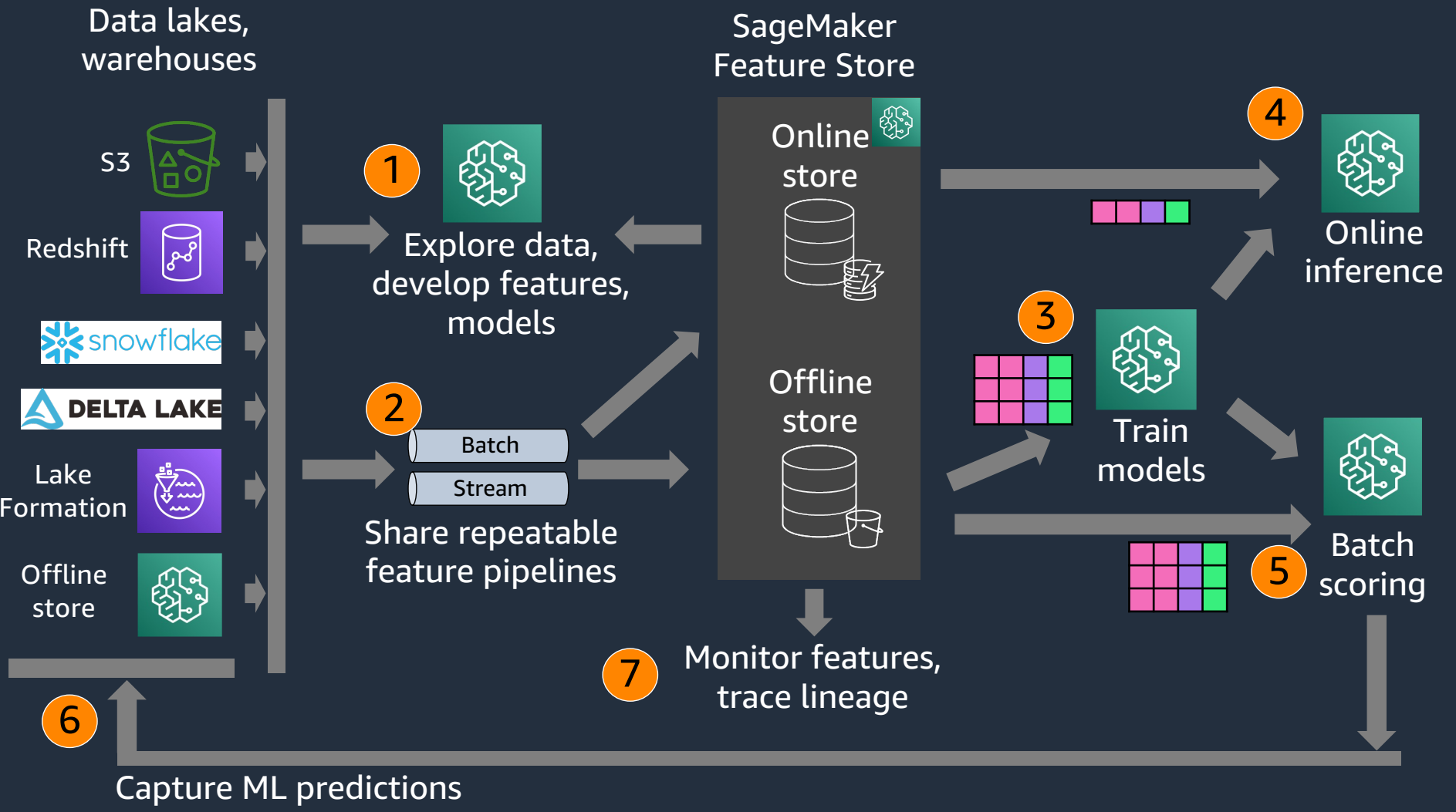
Access control for feature data and feature metadata, and support for encryption at rest, Amazon VPC, and AWS PrivateLink

Fully managed

Online features cached in low-latency store; maintain consistency between online and offline store to avoid train-infer skew

Amazon SageMaker Feature Store

Using Feature Store with data lakes, warehouses



Steps

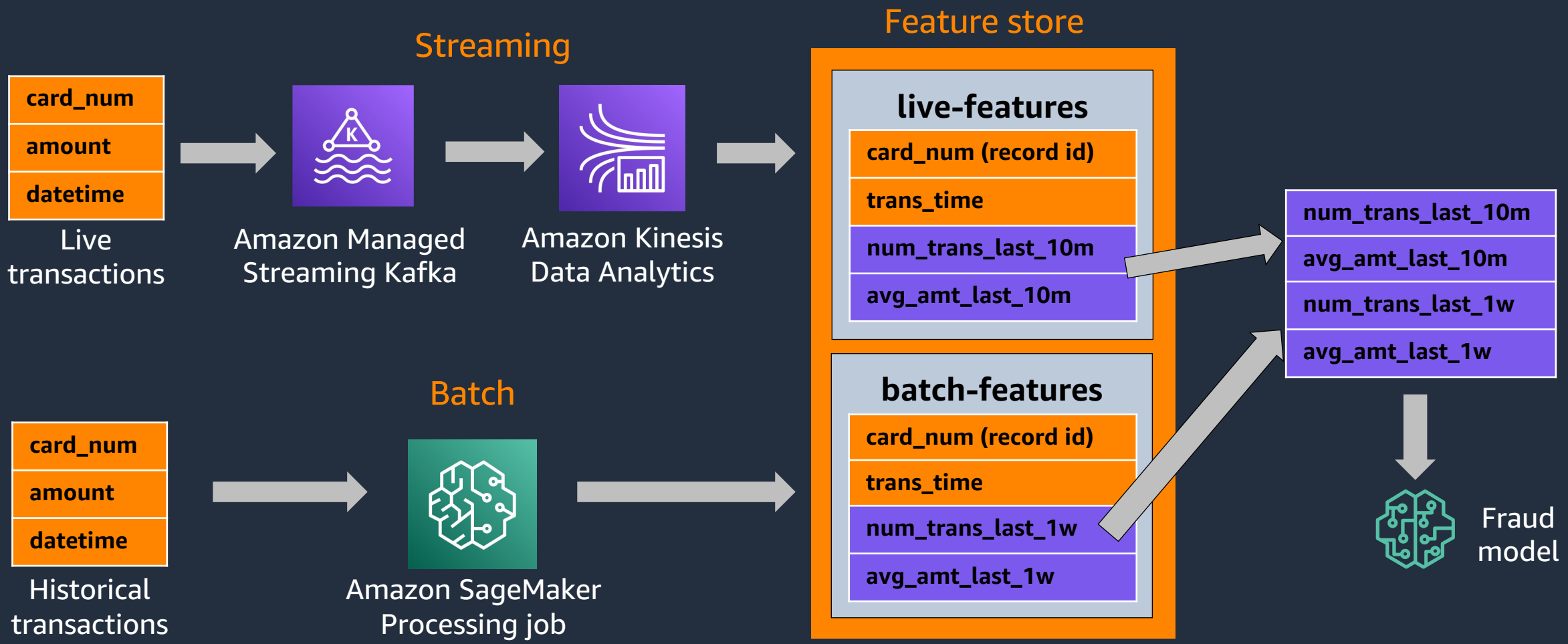
1. Use Studio notebooks or Data Wrangler to explore existing data, experiment with features and models
2. Create repeatable feature pipelines, share features
3. Train models from available features, with point-in-time feature accuracy
4. Make low latency online predictions without train/serve skew
5. Make batch predictions with latest feature values
6. Capture prediction results to drive downstream use cases
7. Monitor features, and trace lineage

aws

Amazon SageMaker Feature Store

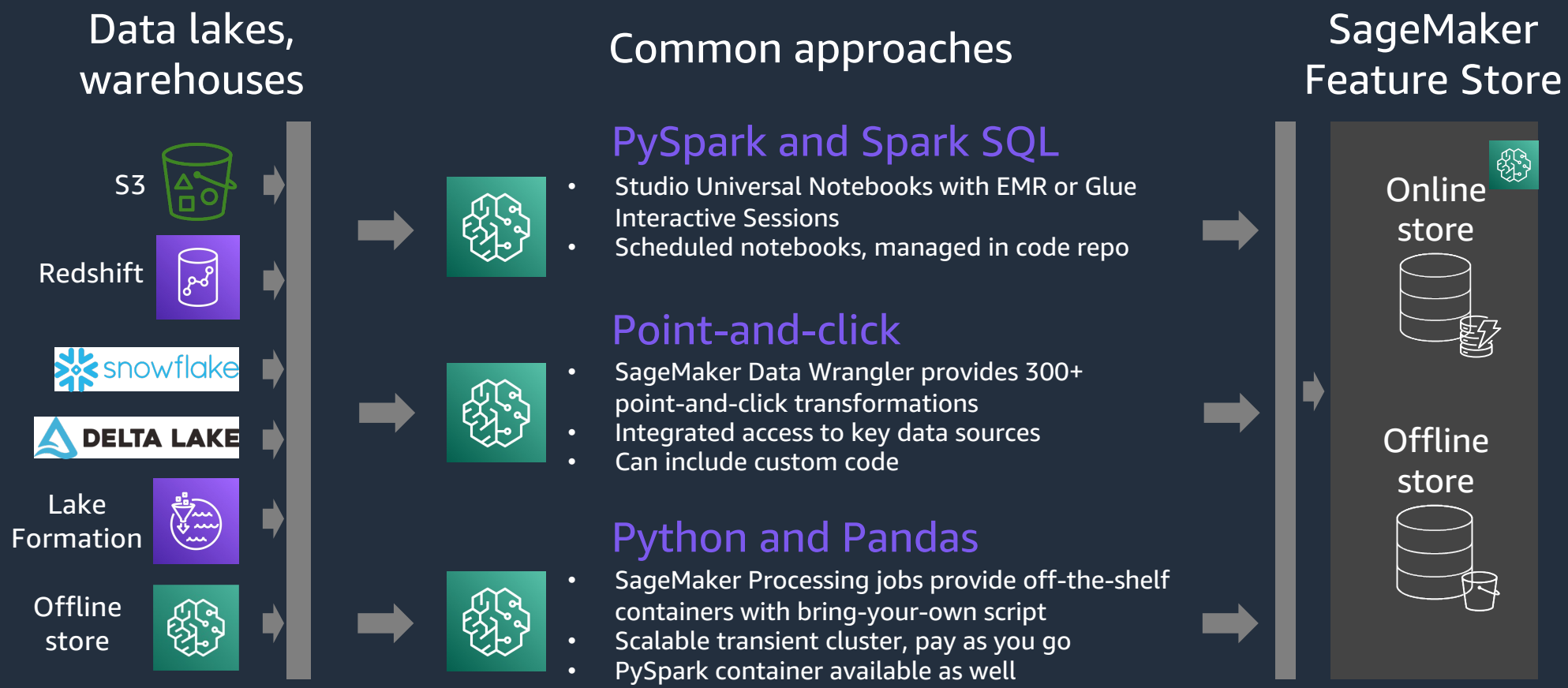
Features are materialized at different frequencies

Online models can pull latest features with low latency



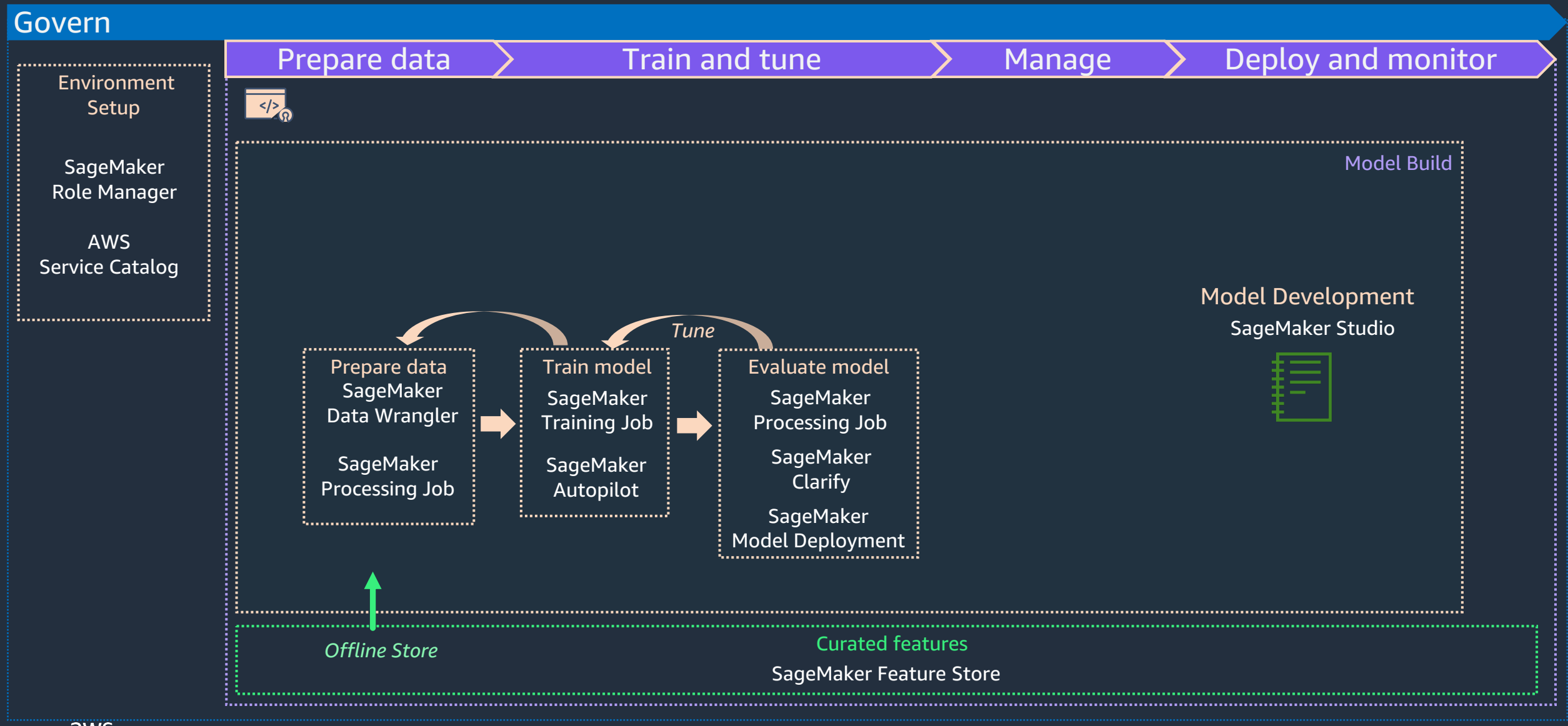
Amazon SageMaker Feature Store

Feature pipeline automation



MLOps Maturity Framework in Practice

Initial Phase Goal: Standardize environments & experiment tracking



Reduce time to Proof-of-Concept

Challenge #3: Efficient experiment evaluation and reliable experiment tracking

Data Scientist wants to prove value of a new use case

Perform iterative experimentation

- Understand data sources
- Try different data transformations
- Try different ML approaches
- Compare model performance

Each candidate model..

...can have many experiments

Experiment Tracking

For each experiment can you reliably determine...

- Model lineage
- Model performance

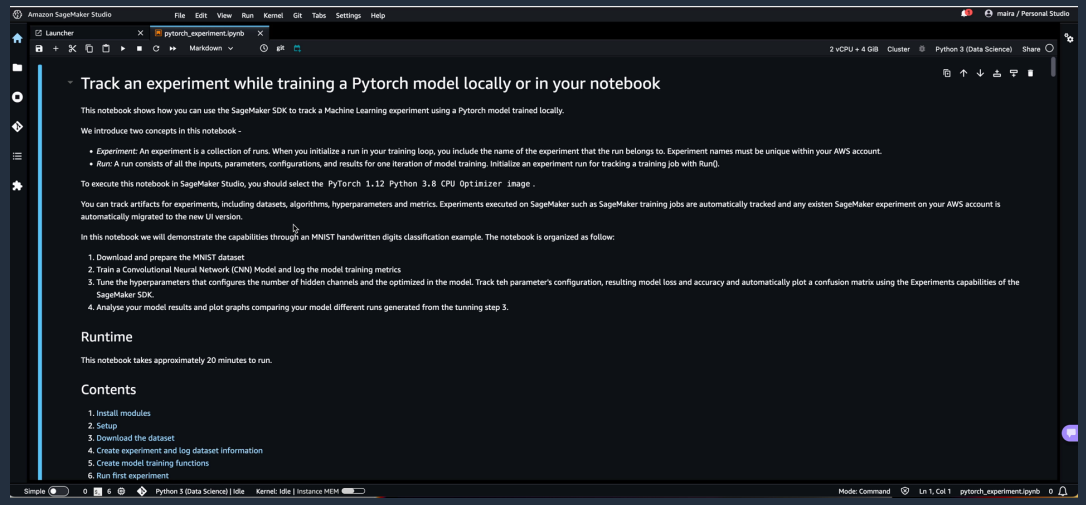
Key Technical Challenge	• Reliable experiment tracking for rapid iteration, management & deployment hand-off
Technical Solutions	• Manage experiments reliably and at scale with experiment tracking tooling
On AWS	• Amazon SageMaker Experiments

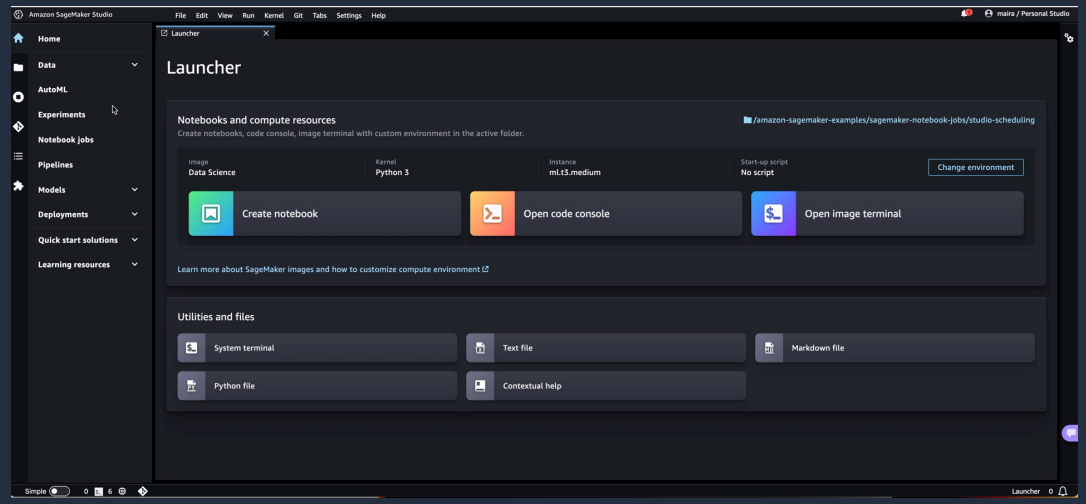
aws

Amazon SageMaker Experiments

Create, manage, analyze, compare and track experiments at scale

Recent update (Dec 2022)





Tracking at scale

Track parameters and metrics across experiments and users

Custom organization

Organize experiments by teams, goals, and hypotheses

Visualization

Easily visualize experiments and compare

Metrics and logging

Log custom metrics using the Python SDK and APIs

Fast iteration

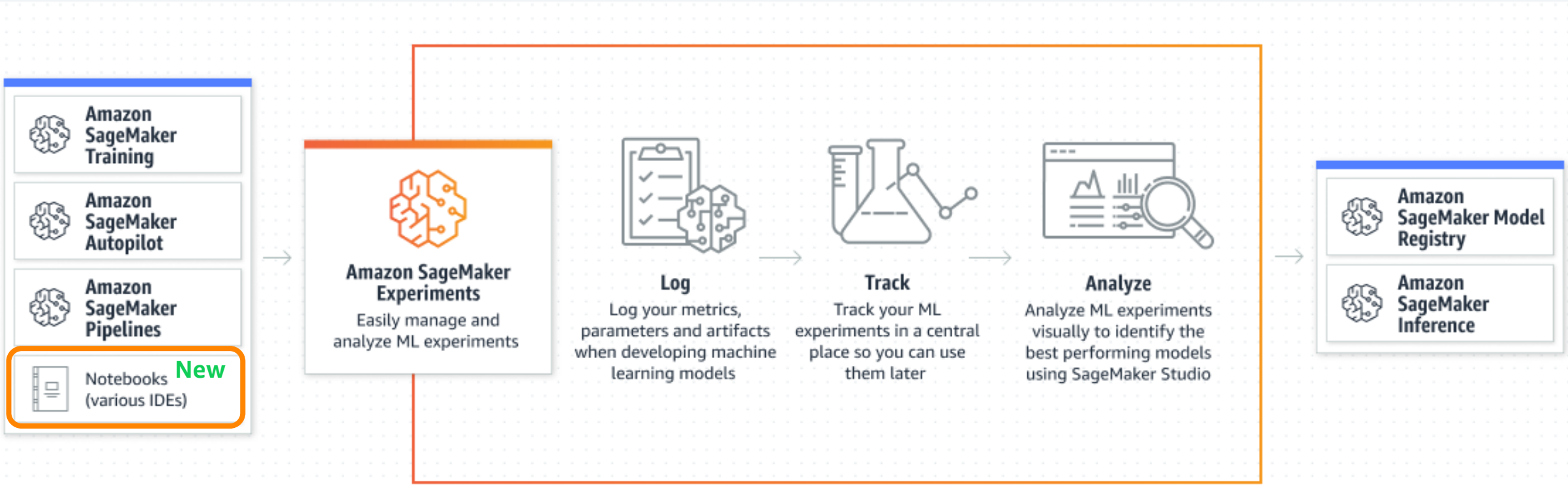
Quickly go back and forth, and maintain high-quality

55

Amazon SageMaker Experiments

Create, manage, analyze, compare and track experiments at scale

How it works



Amazon SageMaker Experiments

Create, manage, analyze, compare and track experiments at scale

Key Concepts

1. **Experiment:** An experiment is a collection of runs. When you initialize a run in your training loop, you include the name of the experiment that the run belongs to. Experiment names must be unique within your AWS account.

2. **Run:** A run consists of all the inputs, parameters, configurations, and results for one iteration of model training. Initialize an experiment run for tracking a training job with Run().

3. **Run Group (optional):** A run group is a unique identifier for groups of associated runs. For example, if the experiment is a pipeline experiment, then runs associated with a specific pipeline execution all belong to the same run group.

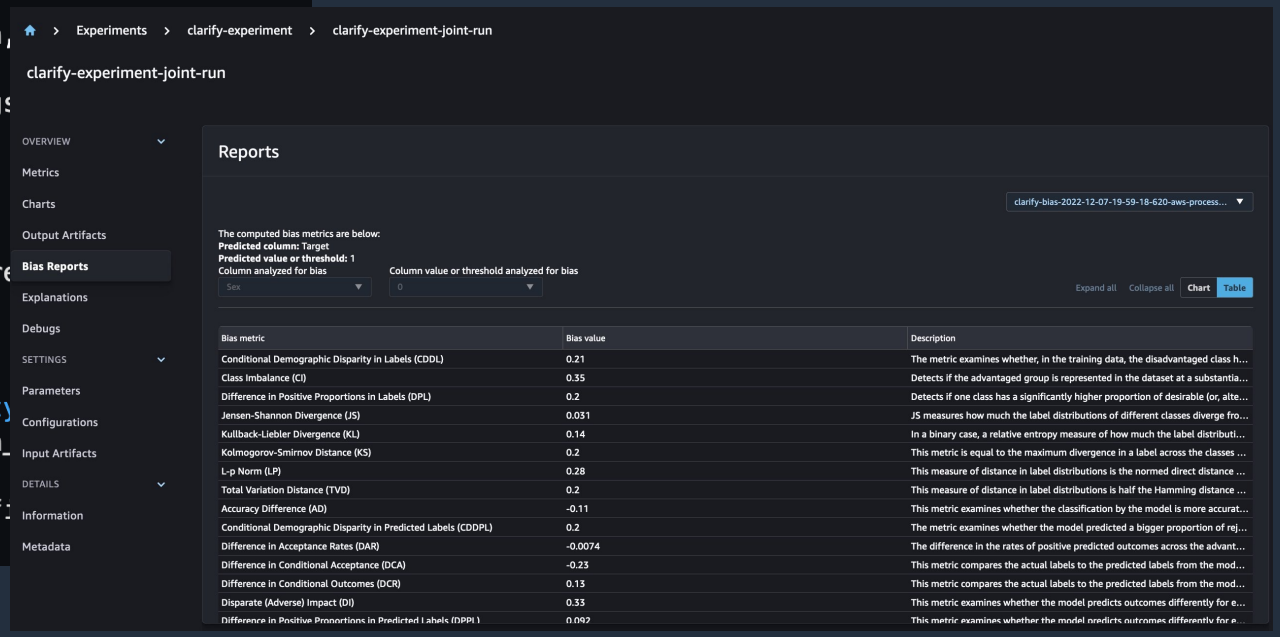
Amazon SageMaker Experiments

Create, manage, analyze, compare and track experiments at scale

Integration with SageMaker Clarify

```
with Run(
    experiment_name='clarify-experiment',
    run_name="joint-run",
    sagemaker_session=sagemaker_session,
) as run:
    xgb.fit({"train": train_input}, logs
    clarify_processor.run_bias(
        data_config=bias_data_config,
        bias_config=bias_config,
        model_config=model_config,
        model_predicted_label_config=pre
        pre_training_methods="all",
        post_training_methods="all",
    )
    clarify_processor.run_explainability
        data_config=explainability_data_
        model_config=model_config,
        explainability_config=shap_conf
    )
```

Bias Report



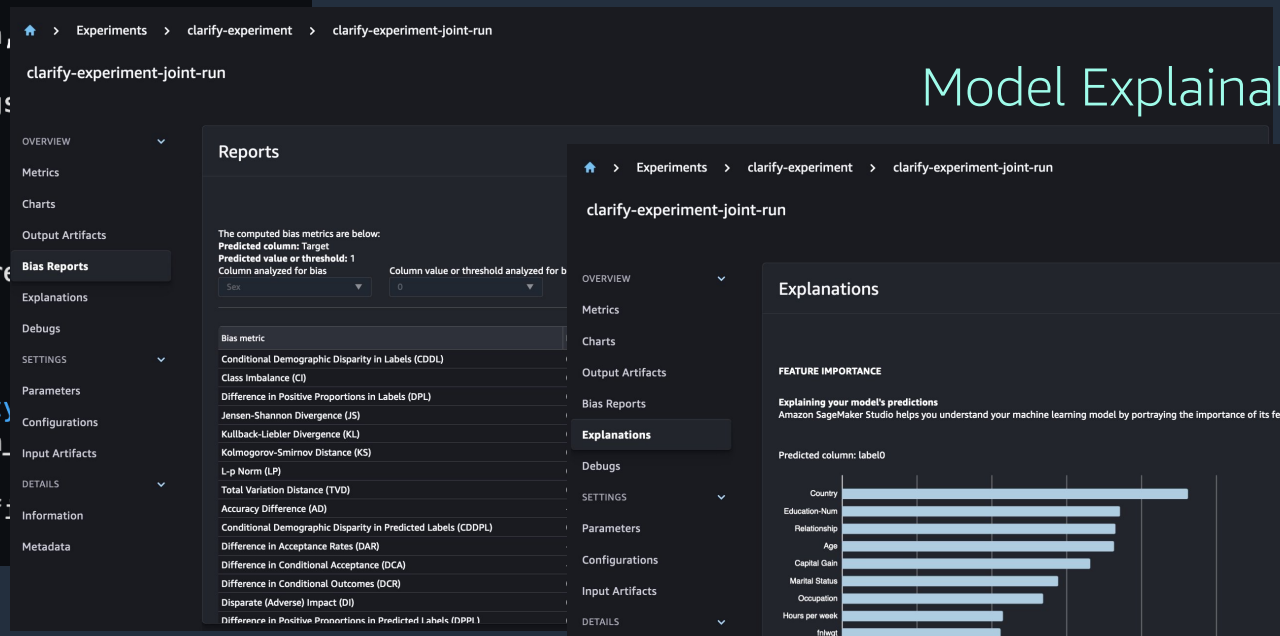
Amazon SageMaker Experiments

Create, manage, analyze, compare and track experiments at scale

Integration with SageMaker Clarify

```
with Run(
    experiment_name='clarify-experiment',
    run_name="joint-run",
    sagemaker_session=sagemaker_session,
) as run:
    xgb.fit({"train": train_input}, logs
    clarify_processor.run_bias(
        data_config=bias_data_config,
        bias_config=bias_config,
        model_config=model_config,
        model_predicted_label_config=pre
        pre_training_methods="all",
        post_training_methods="all",
    )
    clarify_processor.run_explainability
        data_config=explainability_data_
        model_config=model_config,
        explainability_config=shap_conf
    )
```

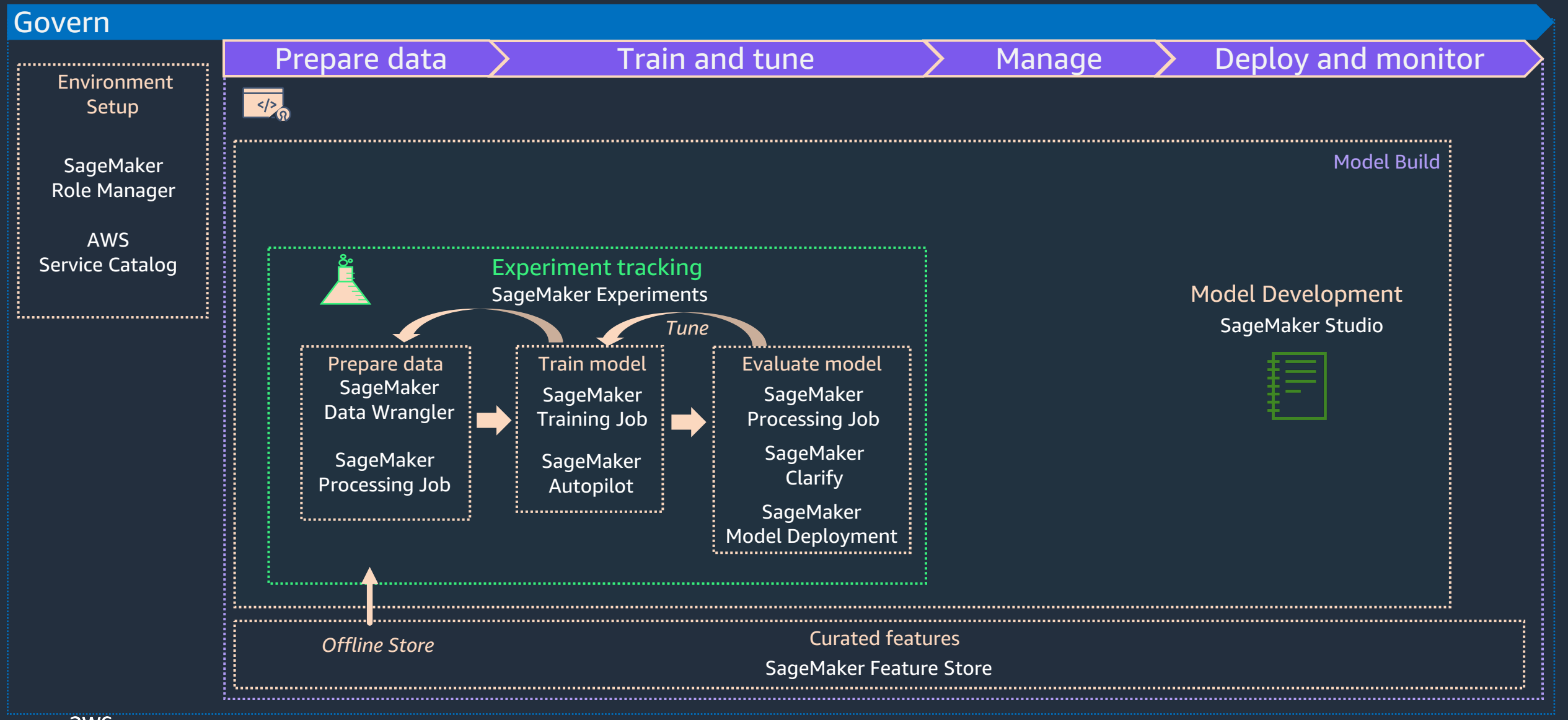
Bias Report



Model Explainability

MLOps Maturity Framework in Practice

Initial Phase Goal: Standardize environments & experiment tracking



Initial Stage

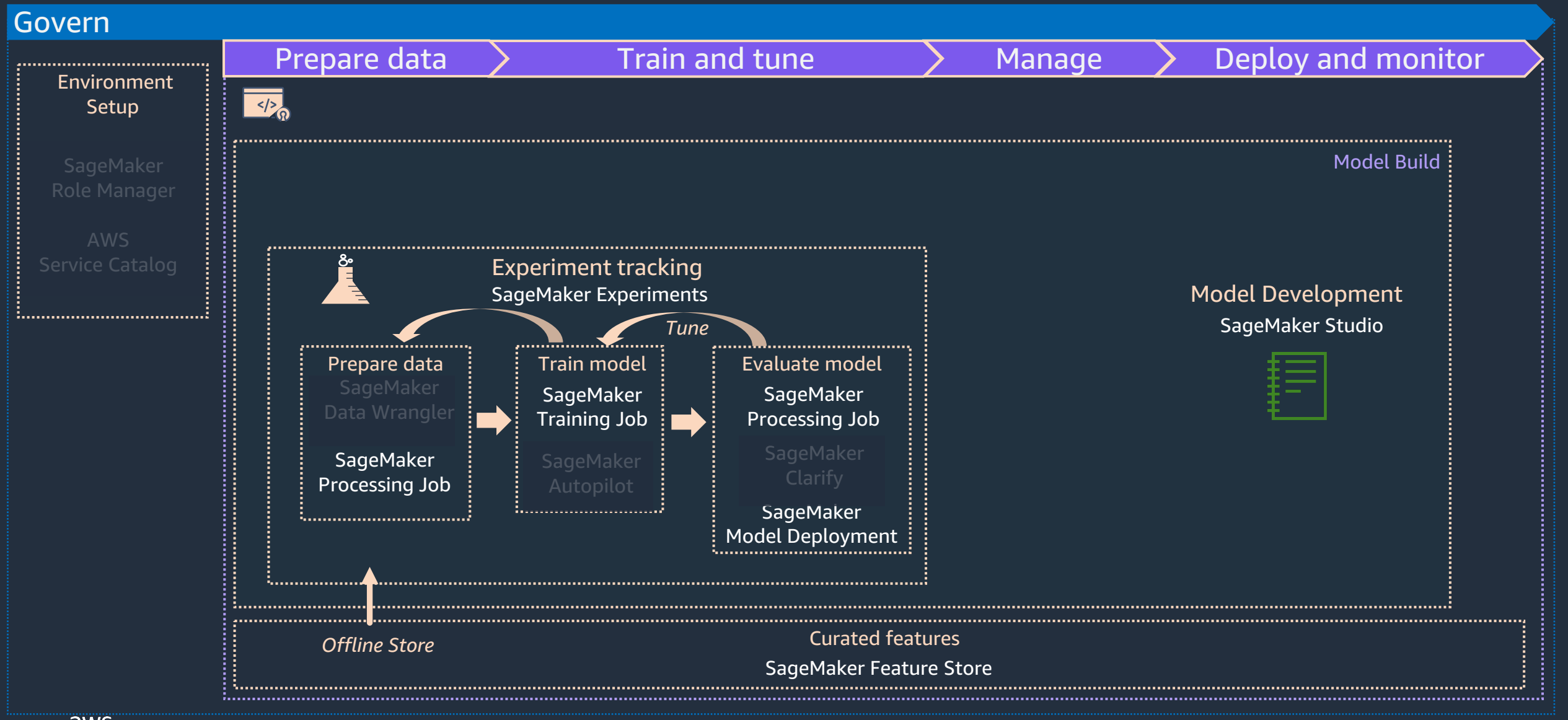
Key Technical Takeaways

1. Standardize data science environments for rapid experimentation with access to needed resources
2. Implement a feature store for feature discovery, reuse, and consistency
3. Implement a reliable mechanism/tooling for experiment tracking

Lab1: Initial

Utilizing SageMaker Experiments during model development

Lab 1: Amazon SageMaker - Features Used



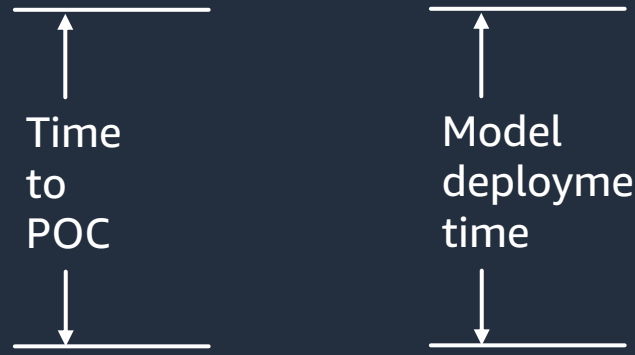
Repeatable Stage:

Implementing repeatable mechanisms to automate and govern your model build and model deployment tasks

MLOps Maturity Framework

Scaling ML through your MLOps journey

Key Performance Indicators



Initial → Repeatable

Key **Technology** Capabilities

- Standardize experimentation environments & processes
- Automate governed access to resources
- Track experiments
- Centralize management of feature data

- Automate model build, and deployment tasks
- Centralize management of models
- Standardize source code & artifact repositories ➔

aws

MLOps Maturity Framework in Practice

Repeatable: Creating a repeatable path to production

Develop Candidate Model(s)

- Business Goals & ML Problem
- Provision Environments
- Perform Experiments

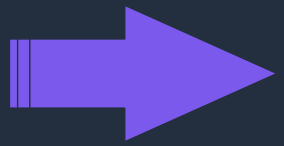
Operationalize Workflow & Candidate Model(s)

- Automate Model Training
- Package and Test Models
- Deploy

Moving from Initial to Repeatable Maturity Level

Repeatable: Creating a repeatable path to production

What if you have 50 models? 100 models? Thousands of models?



How do you document the model and provide model transparency across stakeholders?

How do you automate model build tasks for faster iteration, feature re-use, automated lineage tracking & model re-training?

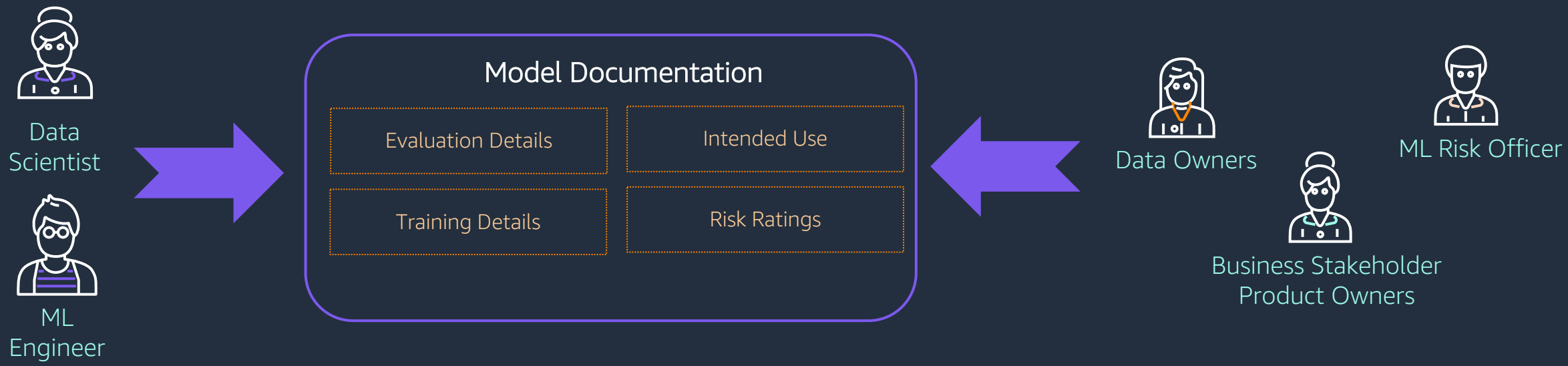
Establish minimum levels of quality?

Efficiently manage model versions?

Automate the path to production deployment?

Reduce time to deployment

Challenge #1: Reliable mechanism to document & share model informaton



Key Technical Challenge

- Lack of model transparency across stakeholders impacting the ability to effectively manage ML use cases and models at scale
- Time consuming process to accurately capture model information

Technical Solutions

- Centralize, standardize, and automate model documentation covering the complete ML use case lifecycle for scaling and governance

On AWS

- Amazon SageMaker Model Cards

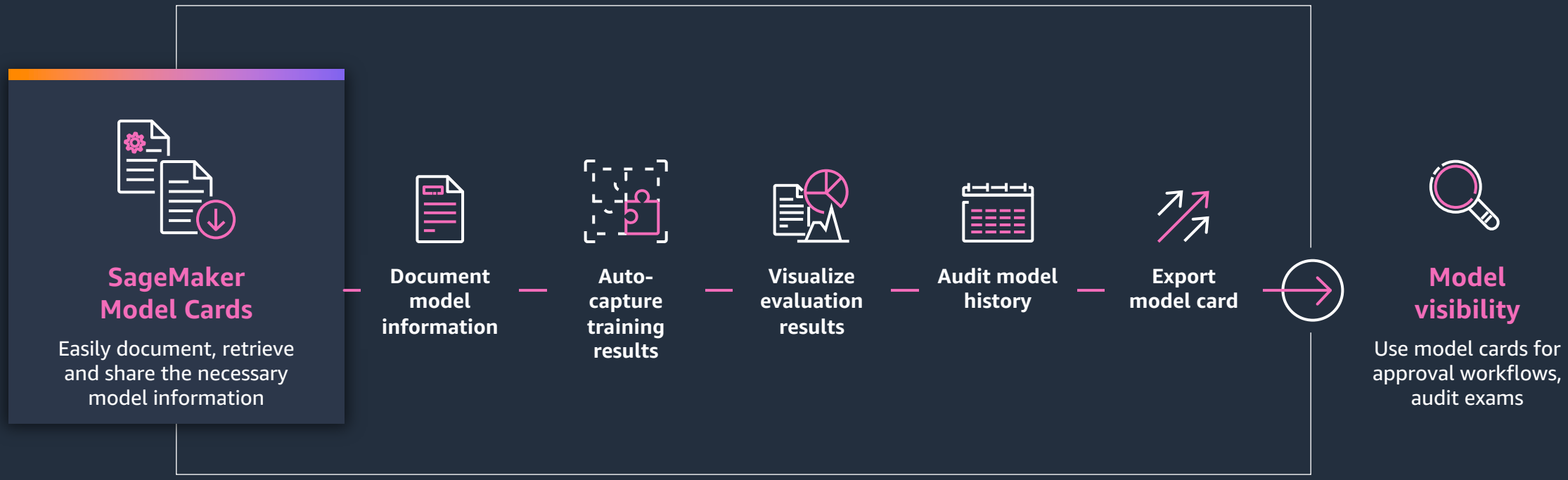
aws

Amazon SageMaker Model Cards

Create single source of truth for model information

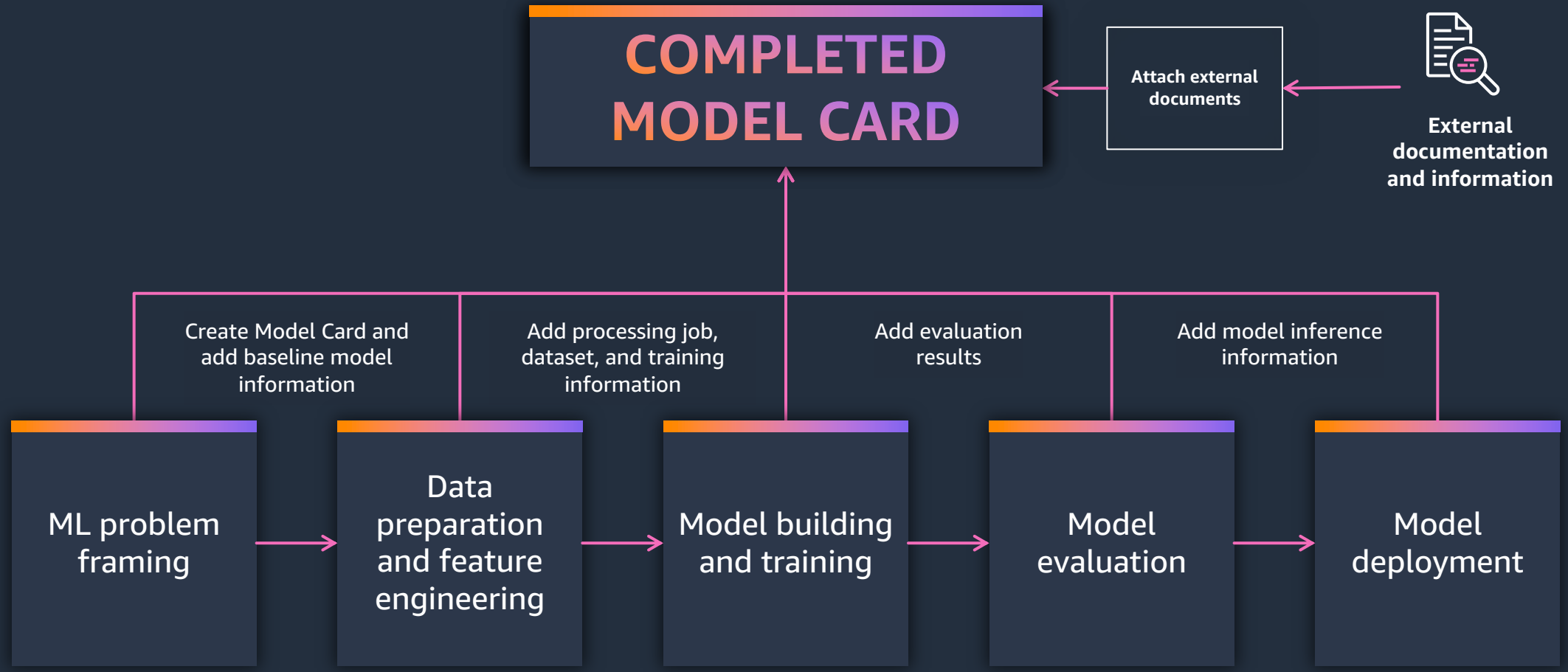
New!!

How it works...



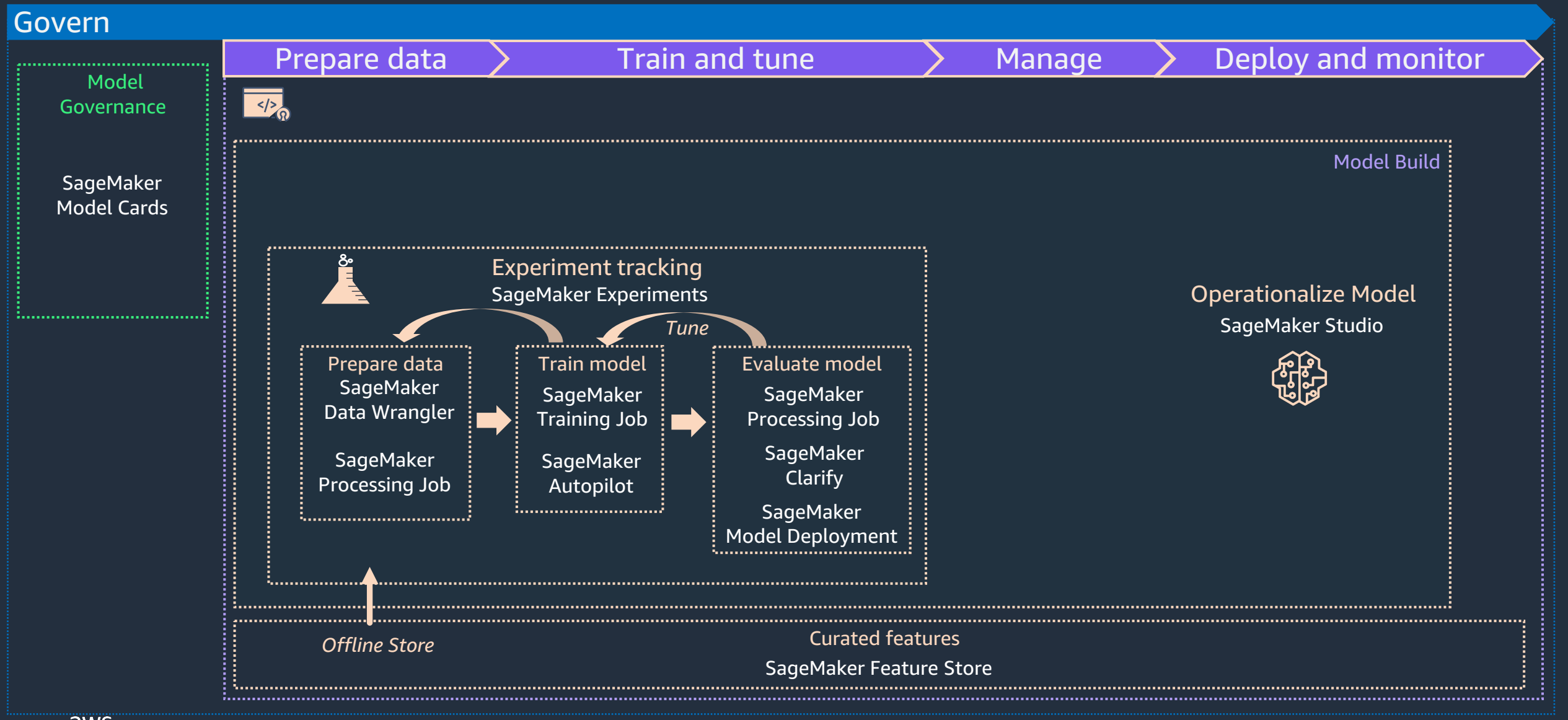
Amazon SageMaker Model Cards

Create single source of truth for model information



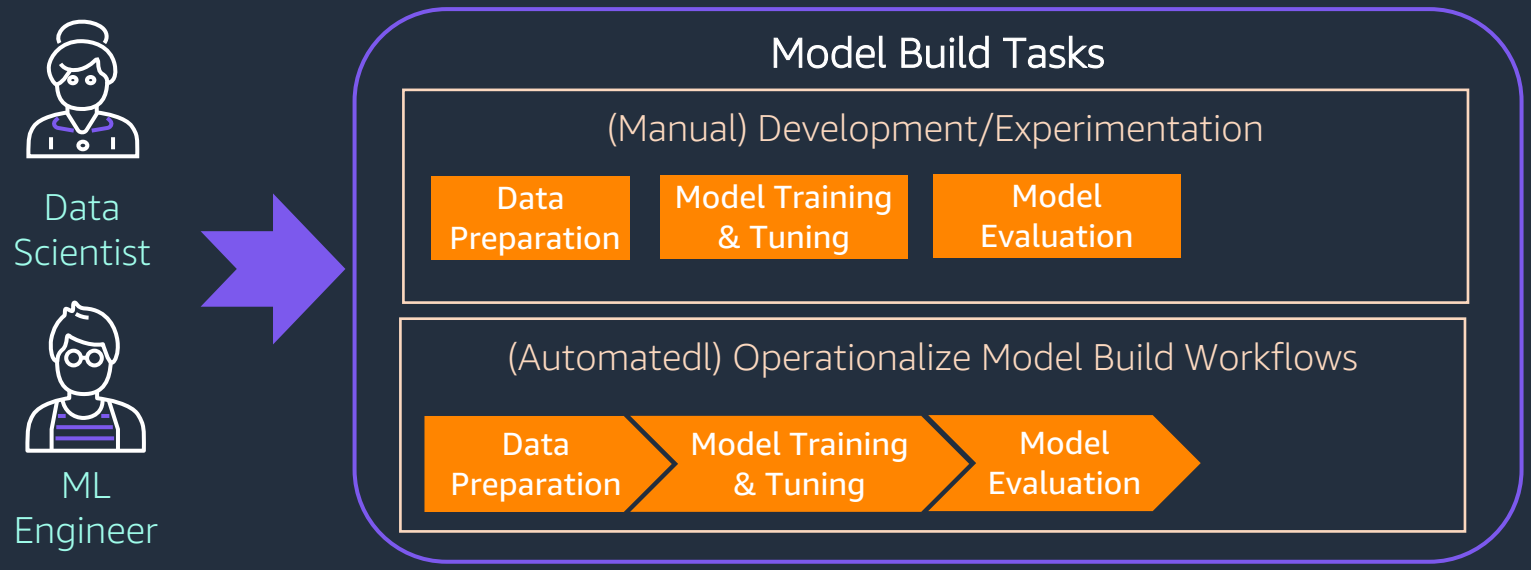
MLOps Maturity Framework in Practice

Repeatable Phase Goal: Create a repeatable path to production



Reduce time to deployment

Challenge #2: Rapidly experiment, deploy, and re-train



Key Technical Challenges

- Enable rapid experimentation across workflow steps and tasks for model tuning
- Operationalize model build workflows for model re-training

Technical Solutions

- Utilize workflow orchestration tooling to automate the steps in your model build workflows

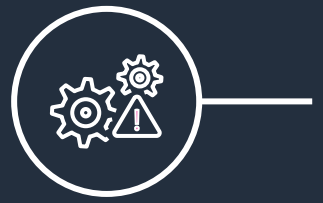
On AWS

- Amazon SageMaker Pipelines

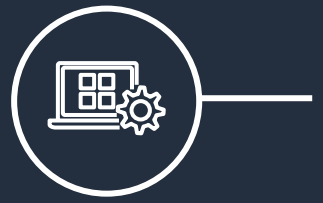
aws

Amazon SageMaker Pipelines

Create automated model build workflows using the Pipelines python SDK

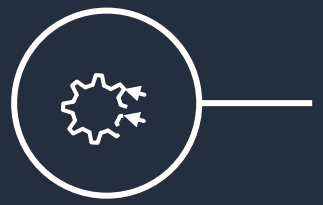


Experiment management
Automatically log pipeline executions to SageMaker Experiments to track model version metadata & performance

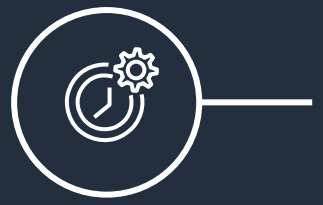


Debug and profile pipeline runs
Easy debugging of pipeline steps

Step Caching
Eliminate cycles recomputing previous pipeline steps



Integrated Pipeline Steps
Native integration with common SageMaker features, AWS Services, and the ability to integrate with non-native steps



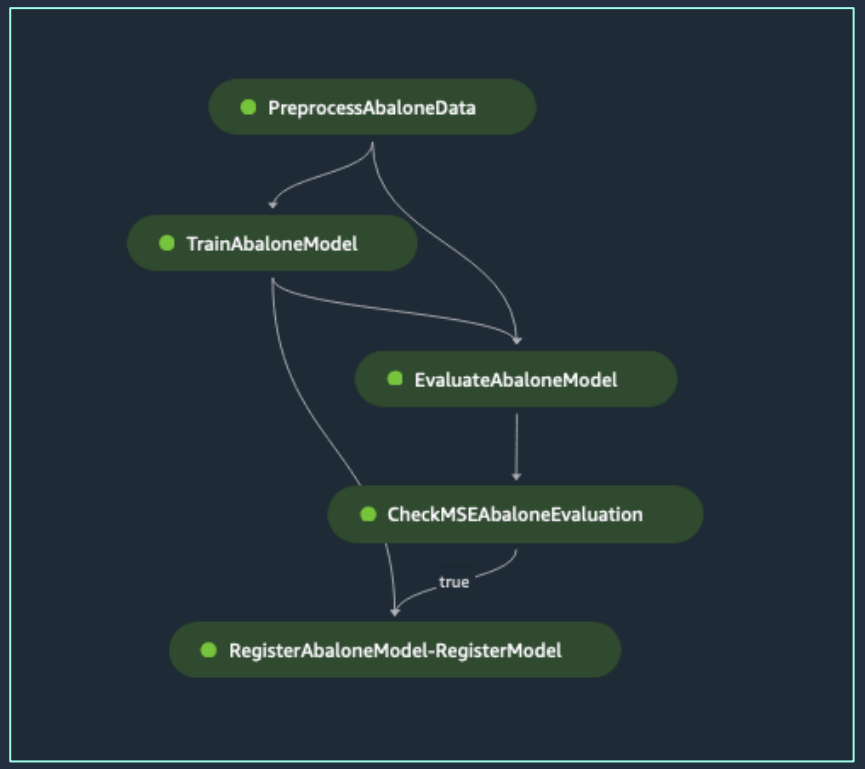
Ability to schedule & trigger pipeline runs
Integration with EventBridge to easily trigger and schedule pipelines

No servers to manage
Pipeline capabilities offered without any additional servers to manage reducing operational overhead

Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Programmatically create pipeline DAGs



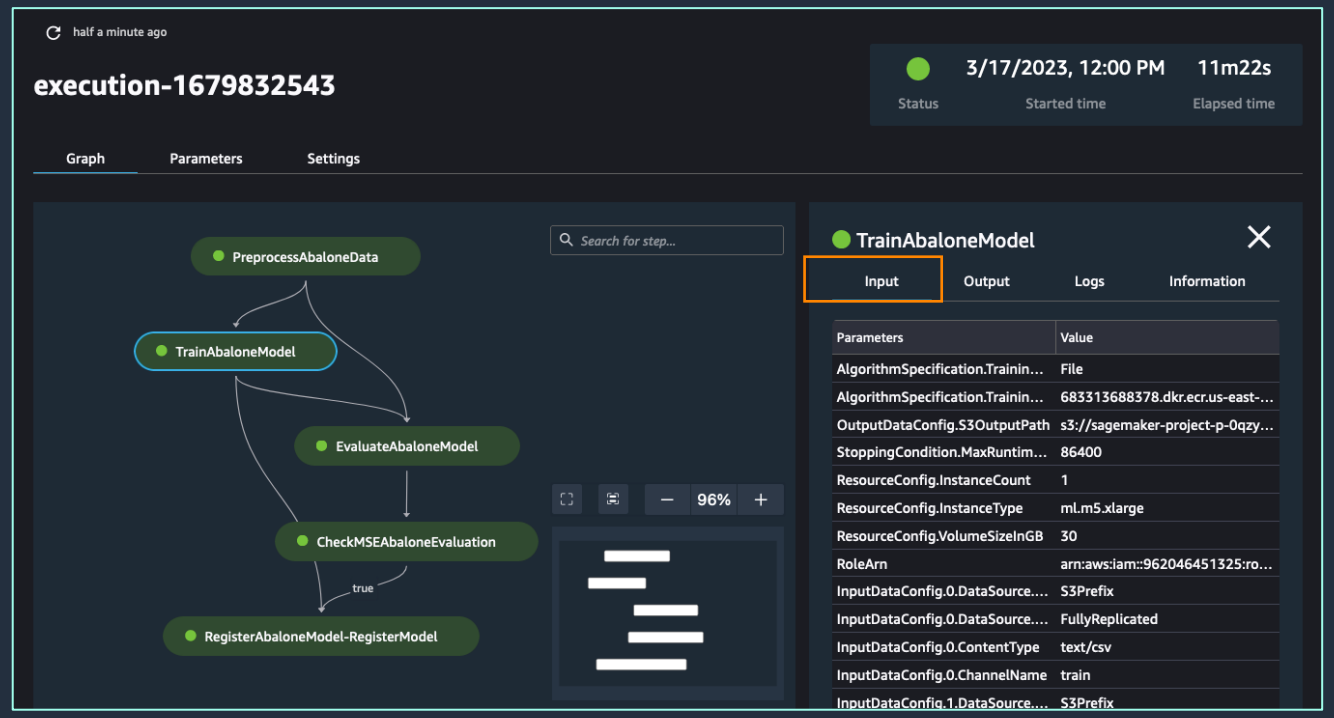
Supported Steps:

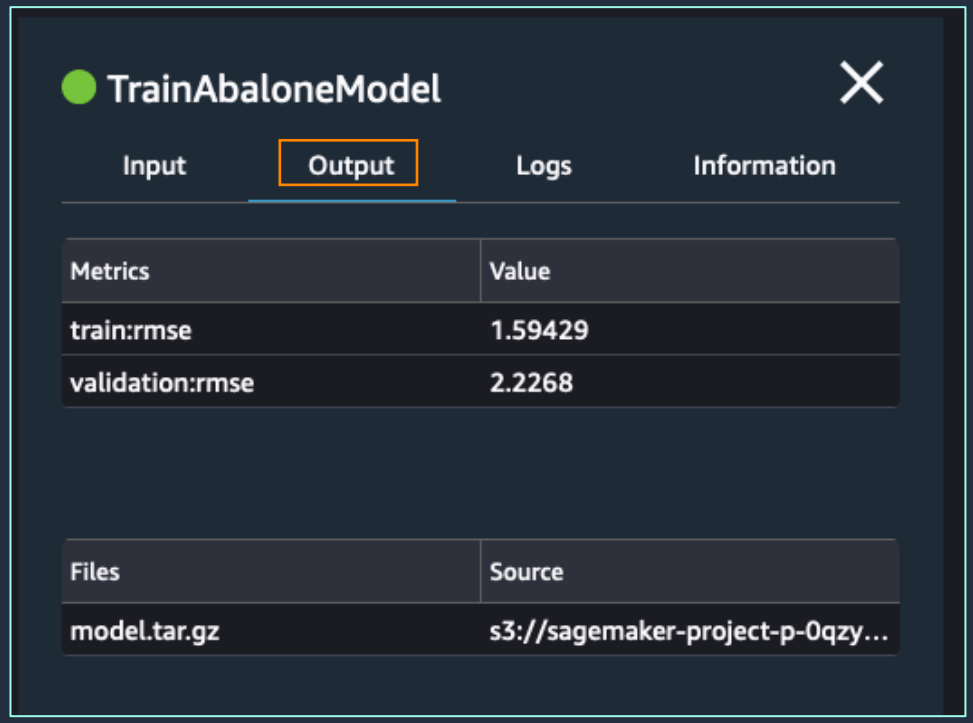
- **Processing:** Data Processing & Model Evaluation
- **Training:** Model Training using SageMaker Training Jobs
- **Tuning:** Hyperparameter Tuning Jobs
- **Condition:** Conditional step execution
- **Transform:** Batch predictions using SageMaker Batch Transform Jobs
- **Model:** Create and register model resource
- **Callback:** Incorporate additional tasks or AWS services into your workflow
- **Lambda:** Run a Lambda function
- **EMR:** Process tasks on a running EMR cluster
- **ClarifyCheck:** Run baseline drift checks against previous baselines.
- **QualityCheck:** Run baseline suggestions and drift checks against a previous baseline

Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Automatically collected input/output metadata for model lineage

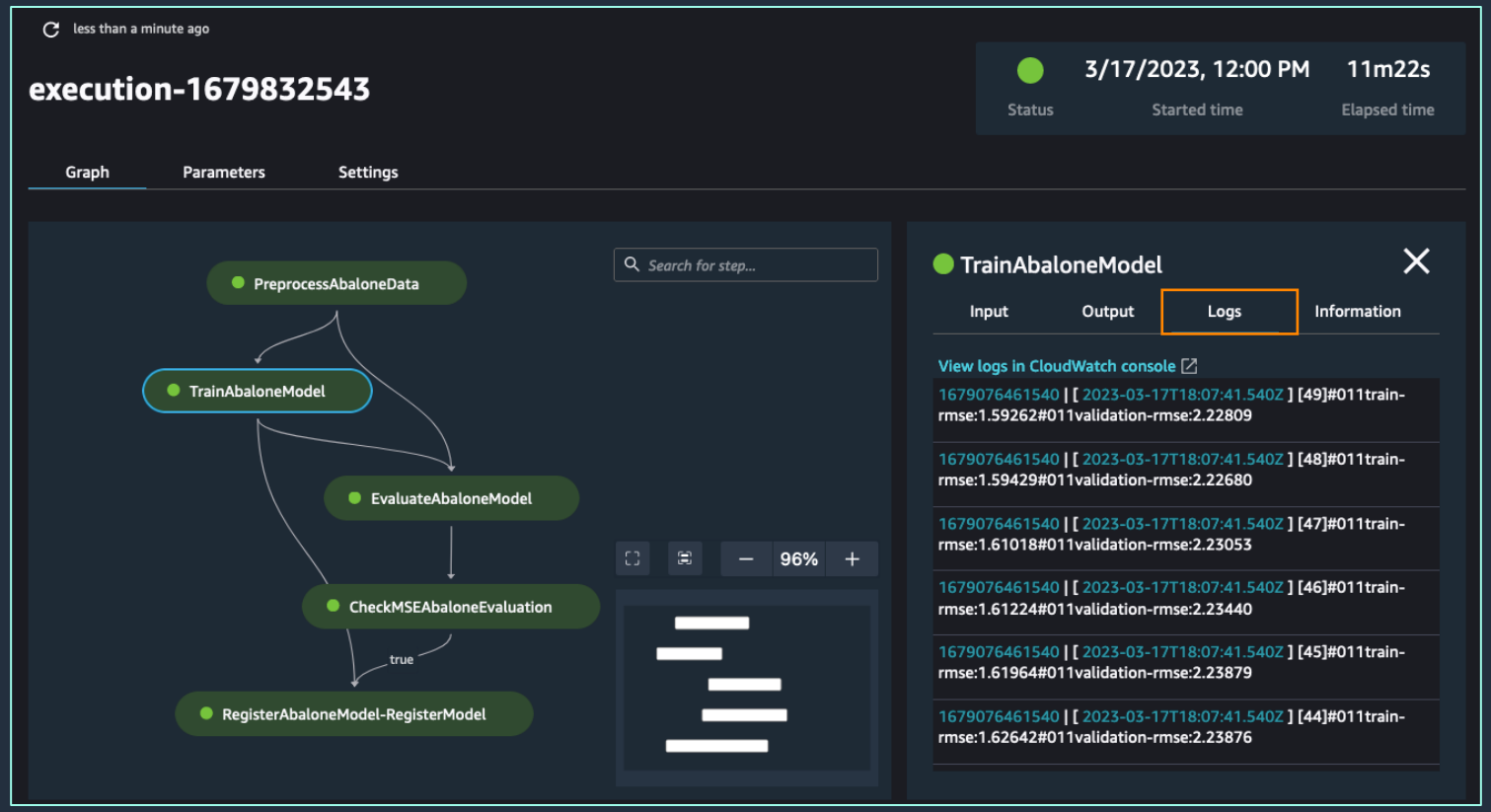




Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Easily Debug Steps

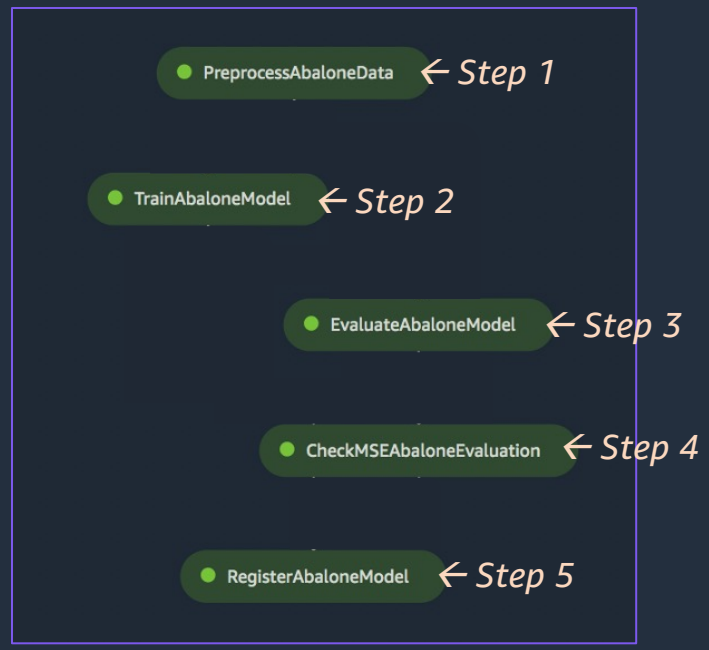


Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

How it works

1 Create Steps →



aws

Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

How it works...

1 Create Steps →

```
from sagemaker.inputs import TrainingInput
from sagemaker.workflow.steps import TrainingStep

step_train = TrainingStep(
    name="TrainAbaloneModel",
    estimator=xgb_train,
    inputs={
        "train": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "train"
            ].S3Output.S3Uri,
            content_type="text/csv"
        ),
        "validation": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "validation"
            ].S3Output.S3Uri,
            content_type="text/csv"
        )
    }
)
```

Example:
Training Step

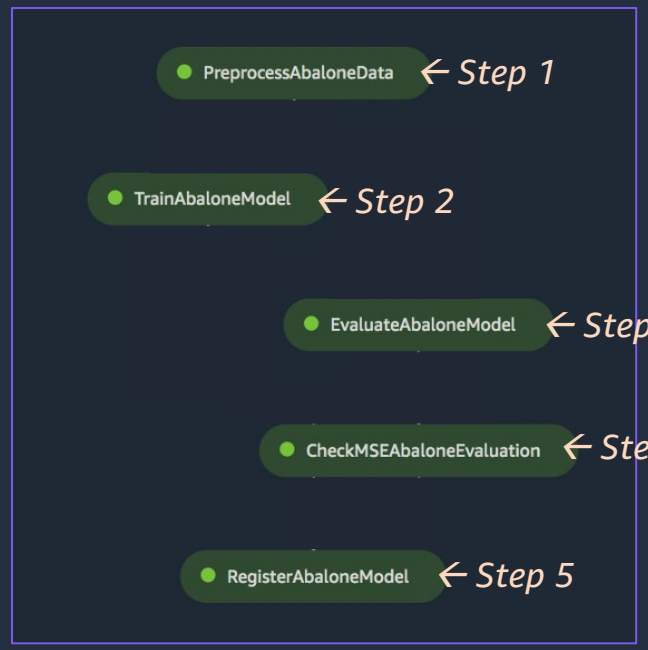
aws

Amazon SageMaker Pipelines

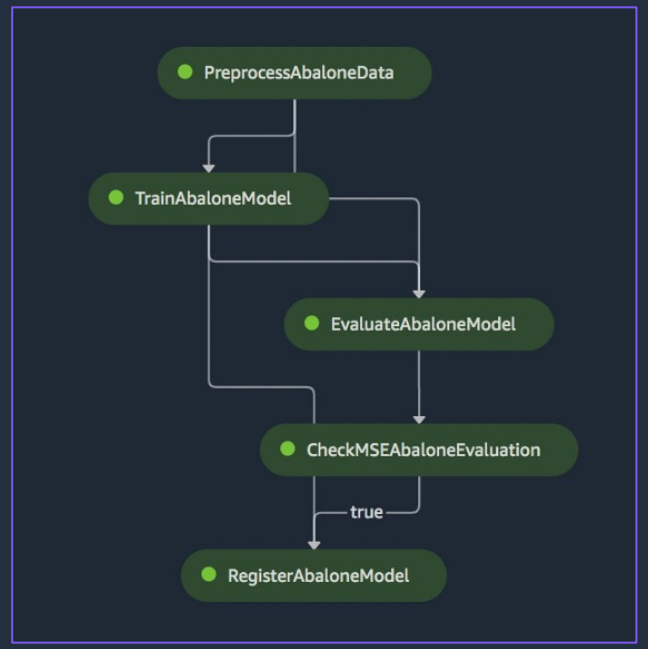
Orchestrate steps & automate tasks in your ML workflows

How it works...

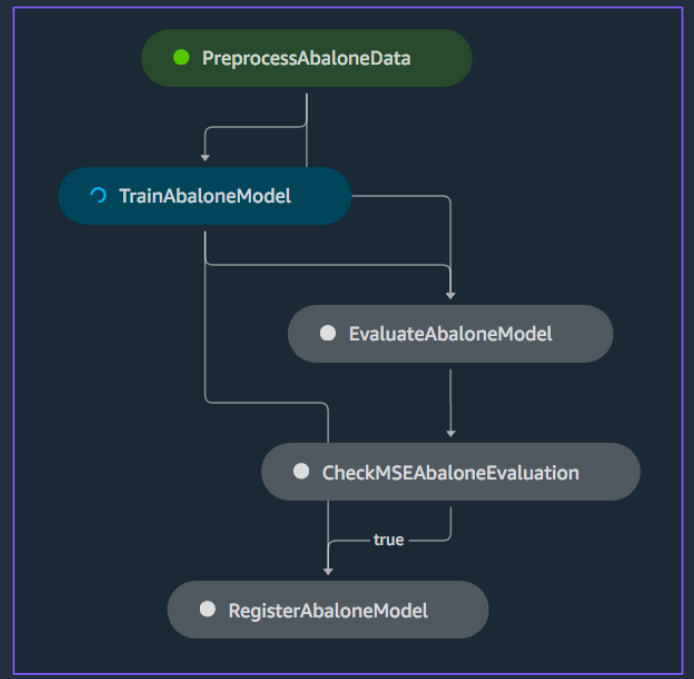
1 Create Steps →



2 Define Pipeline →



3 Start Pipeline →



aws

Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

How it works...

2 Define Pipeline →

```
from sagemaker.workflow.pipeline import Pipeline

pipeline_name = f"AbalonePipeline"
pipeline = Pipeline(
          name=pipeline_name,
          parameters=[
                    processing_instance_count,
                    model_approval_status,
                    ],
          steps=[step_process, step_train, step_eval, step_cond],
)

#Submit the pipeline definition
pipeline.upsert(role_arn=role)
```

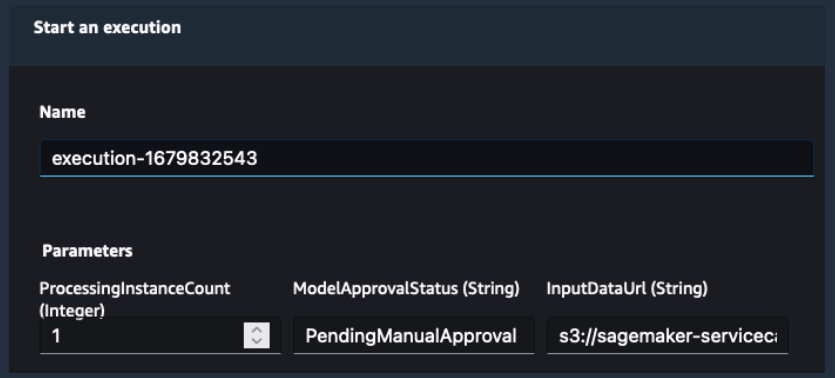
3 Start Pipeline →

programmatically

```
execution = pipeline.start()
```

~OR~

SageMaker Studio

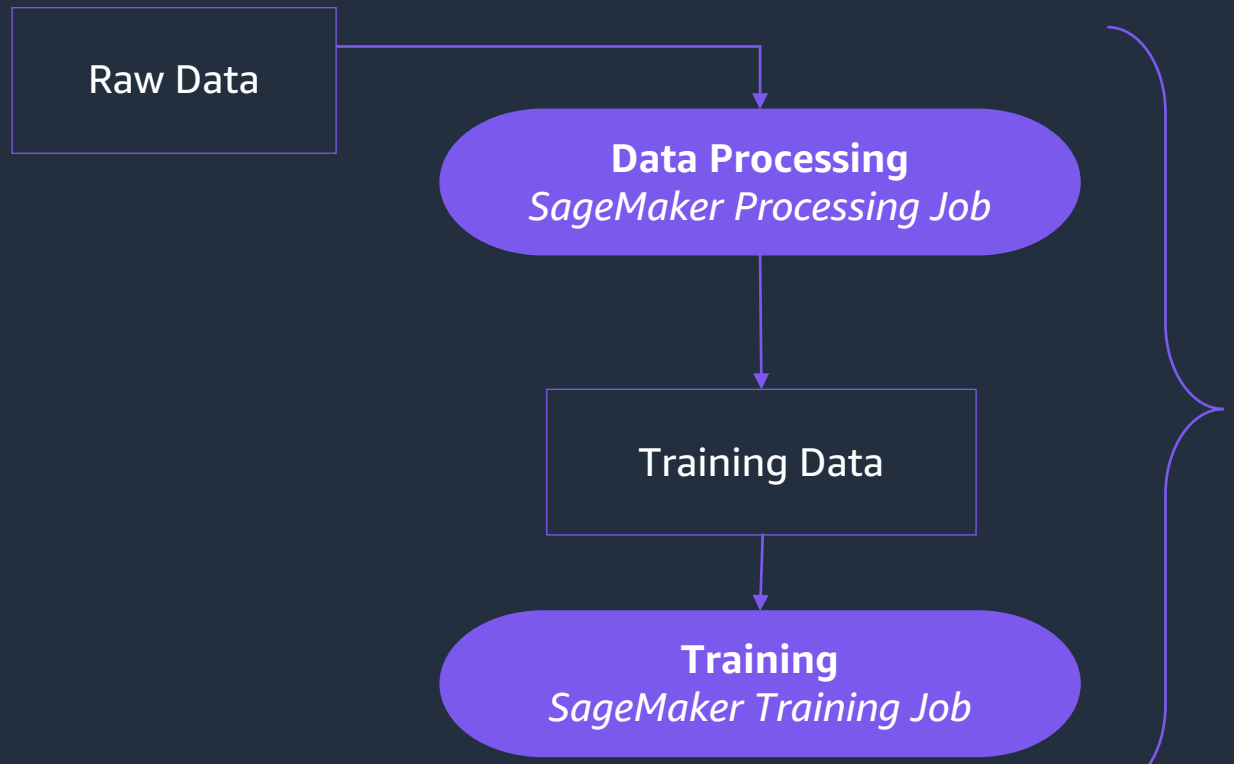


Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Feature Highlights: Step Caching

Cache successful results from previous step to avoid redundant processing ➔



What if you want to:
- tune hyperparameters?
- modify training code?

Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Feature Highlights: Runtime Parameters

Change parameters at runtime without changing your pipeline code →

1 Configure your parameter →

```
from sagemaker.workflow.parameters import (
    ParameterInteger,
    ParameterString,
    ParameterFloat
)

processing_instance_count = ParameterInteger(
    name="ProcessingInstanceCount",
    default_value=1
)
```

2 Pass in parameter on pipeline create →

```
pipeline = Pipeline(
  name=pipeline_name,
  parameters=[
    processing_instance_count
  ],
steps=[step_process]
)
```

3 Optionally, pass a non-default value for pipeline execution →

```
execution = pipeline.start(
    parameters=dict(
            ProcessingInstanceCount=4
    )
)
```

aws

Amazon SageMaker Pipelines

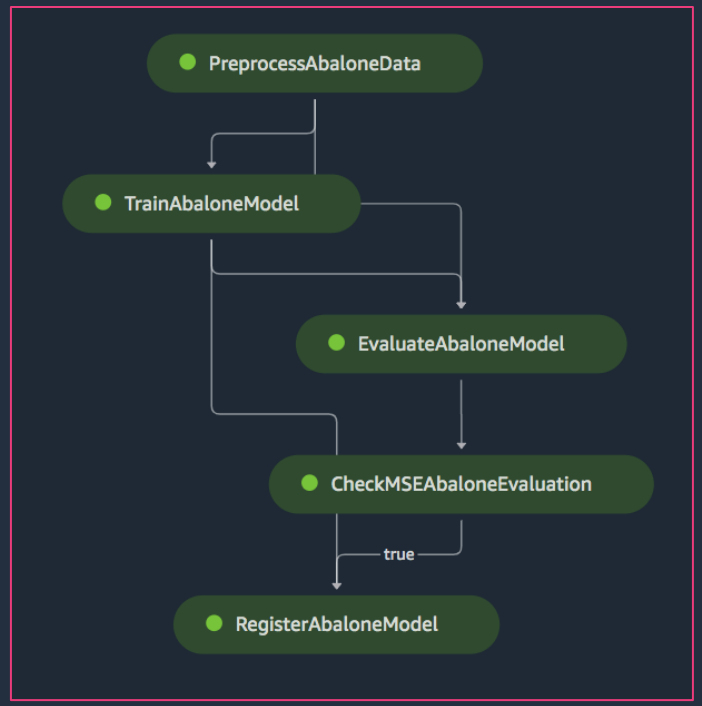
Orchestrate steps & automate tasks in your ML workflows

Feature Highlights: Amazon EventBridge Integration

Setup automatic pipeline triggers based on events or schedules ➔

Event

Trigger Pipeline



aws

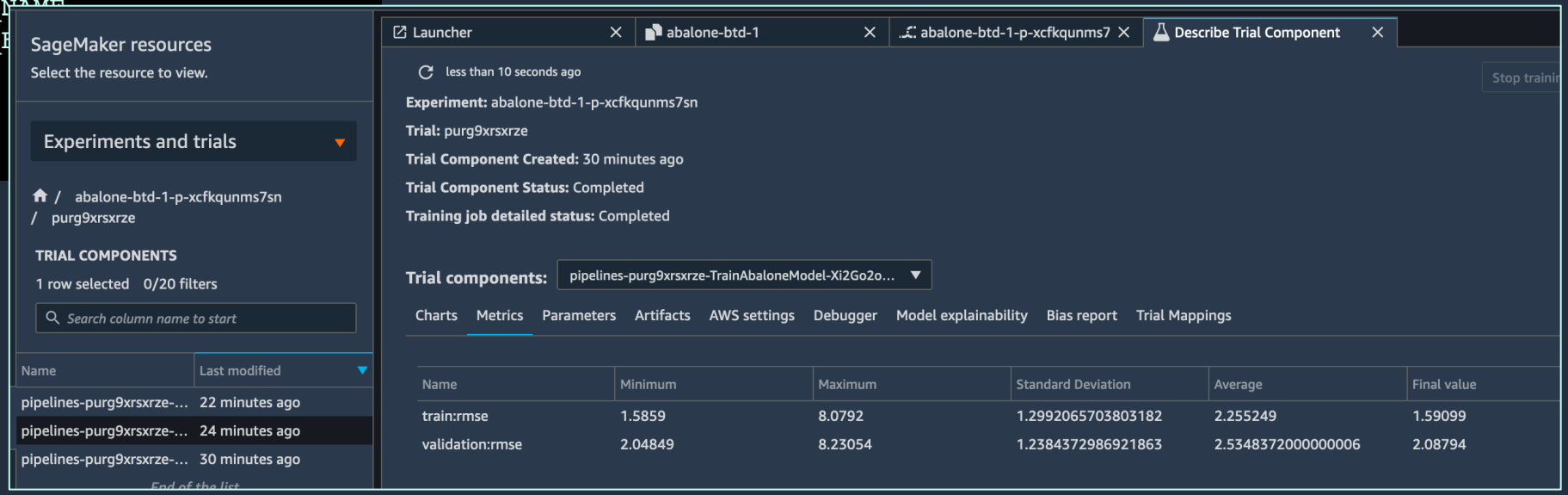
Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Feature Highlights: Experiments Integration

Automatically capture each pipeline run as an experiment ➔

```
Pipeline(
    name="MyPipeline",
    parameters=[...],
    pipeline_experiment_config=PipelineExperimentConfig(
      ExecutionVariables.PIPELINE_NAME
      ExecutionVariables.PIPELINE_E
    ),
    steps=[...]
)
```



Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

Deployment Support

For Real-Time/Near Real-Time Inference →

SageMaker Pipelines Option

Use Lambda
or
Callback Step

~OR~

Other Option

Deploy through CD tooling using IaC/CaC for rollback, consistency, and simplified resource management

For Batch Inference →

SageMaker Pipelines Option #1
Use Case: Training & Inference share the same lifecycle

Run batch inference as part of your training pipeline ~or~ as a separate pipeline

SageMaker Pipelines Option #2
Use Case: Training & Inference have a different lifecycle

Run training & batch inference in separate pipelines

aws

Amazon SageMaker Pipelines

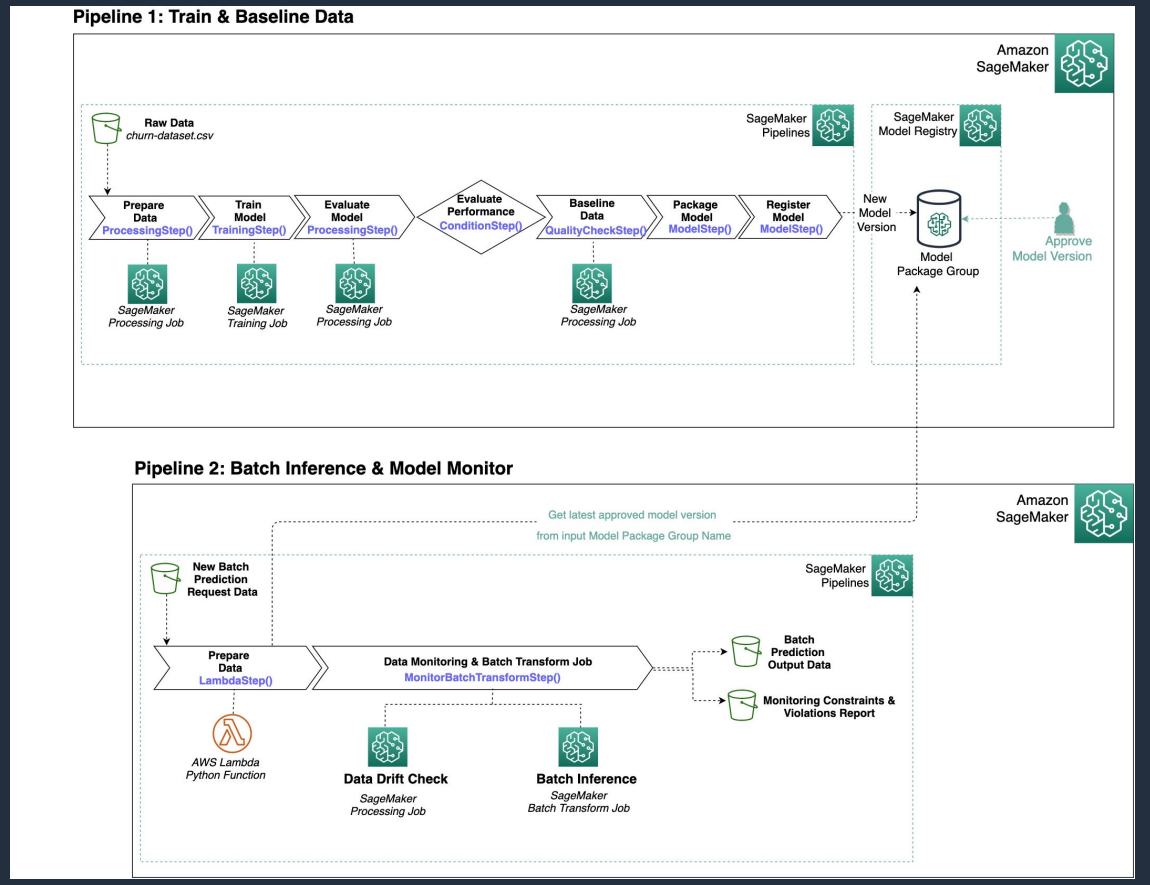
Orchestrate steps & automate tasks in your ML workflows

Using SageMaker Pipelines for Batch Inference

Option #2: Example

Use Case:

- Training & Inference have a different lifecycle
- Model is trained monthly
- Inference if performed daily



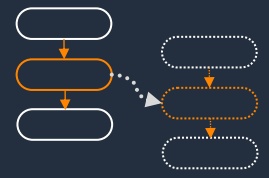
Amazon SageMaker Pipelines

Orchestrate steps & automate tasks in your ML workflows

What's new?

Local mode

Utilize local mode to quickly test your training, processing, and inference scripts



Cross account pipeline sharing

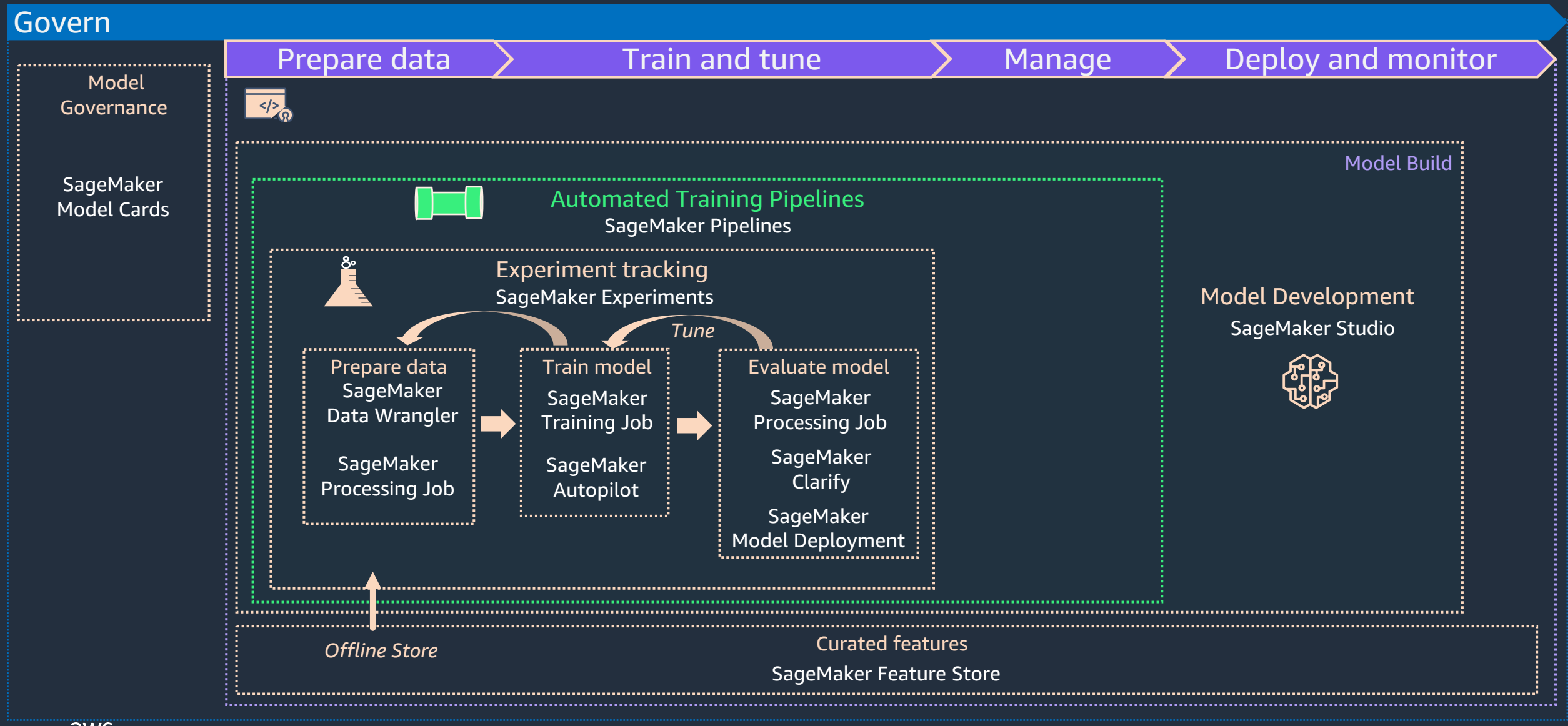
Share pipeline entities across AWS accounts and access shared pipelines through API calls

SageMaker Pipelines SDK & SageMaker Studio UI Enhancements

Updates providing new features, enhanced ease-of-use, and DAG visualization

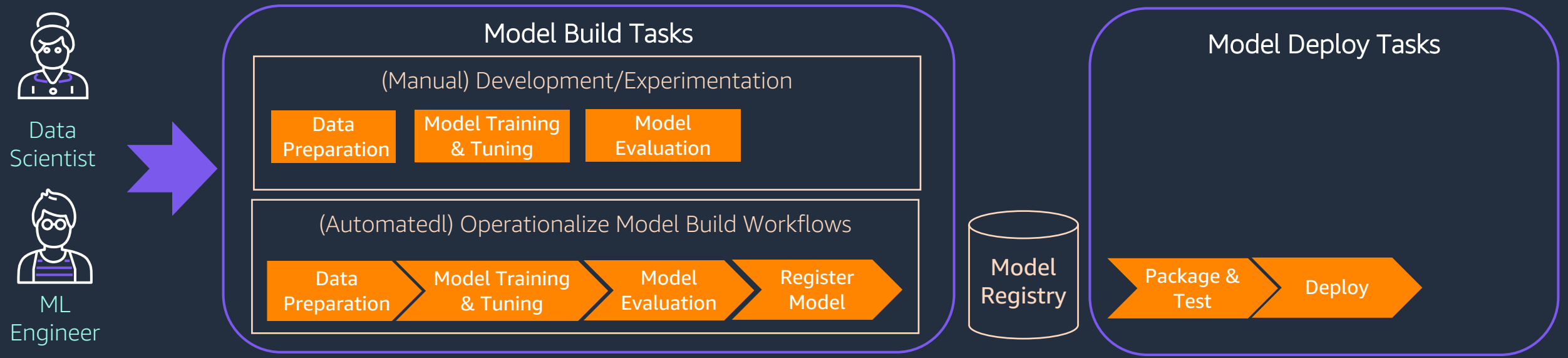
MLOps Maturity Framework in Practice

Repeatable Phase Goal: Create a repeatable path to production



Reduce time to deployment

Challenge #3: Ability to manage model versions at scale & streamline deployment



Key Technical Challenges

- Manage candidate model versions at scale
- Manage the hand-off between model build and model deploy tasks/personas

Technical Solutions

- Utilize a model registry to manage model versions (candidates for production deployment)
- Utilize a model registry to automate deployment workflows & approvals

On AWS

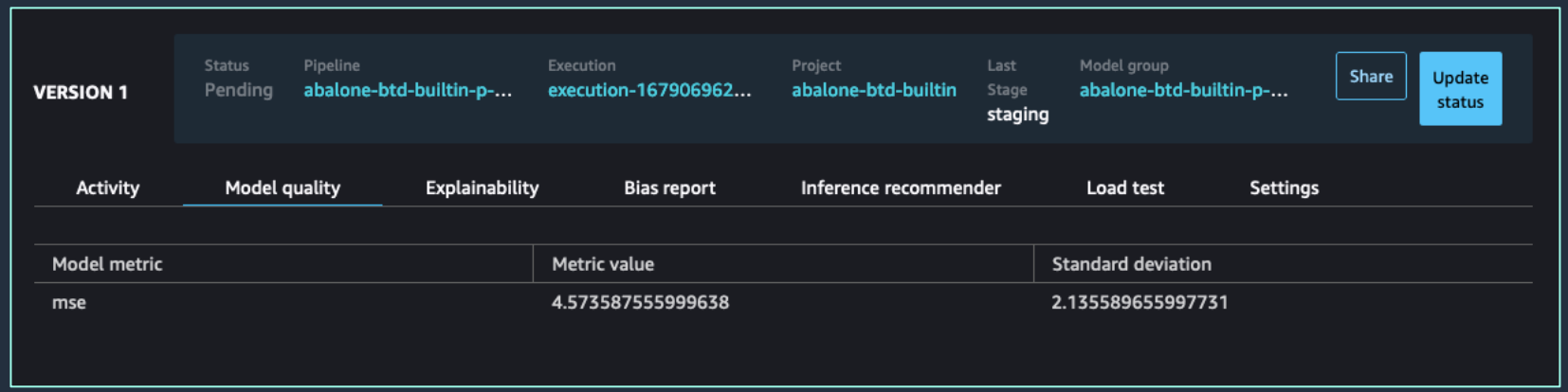
- Amazon SageMaker Model Registry

aws

Amazon SageMaker Model Registry

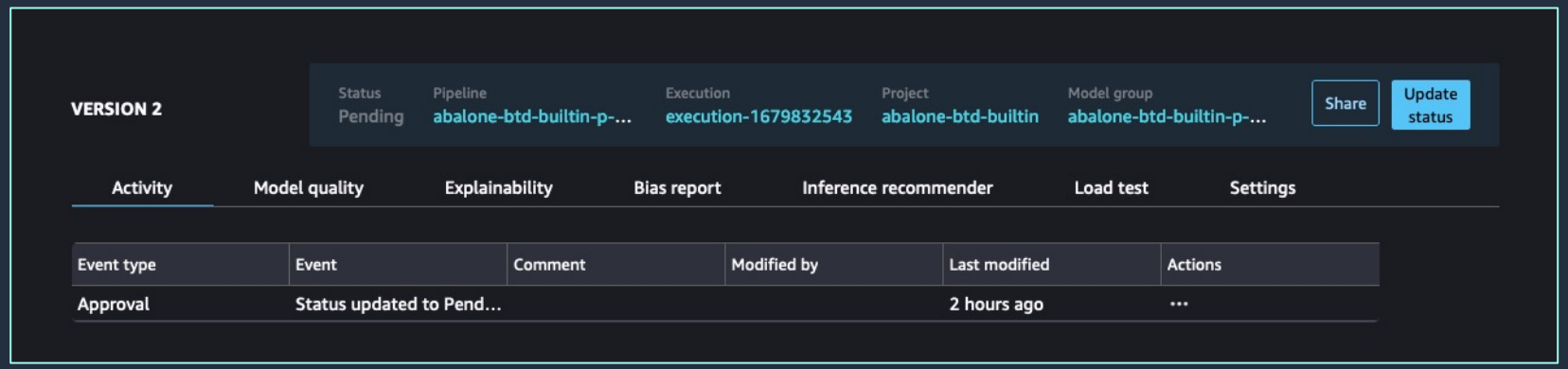
Centrally catalog & manage model versions

- Associate metadata with a model
- Track model performance metrics

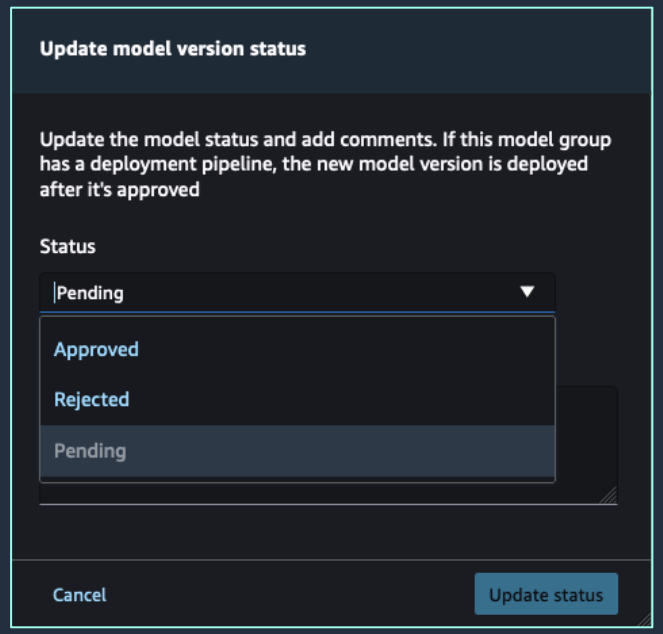


Amazon SageMaker Model Registry

Centrally catalog & manage model versions

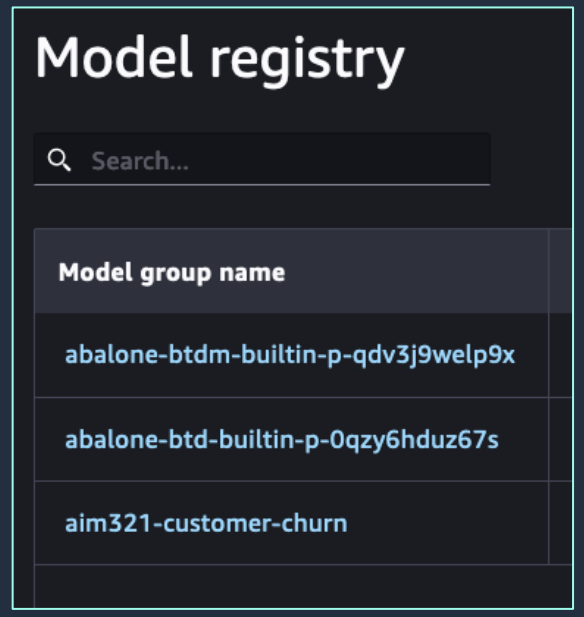


- Manage the approval status of a model
- Deploy models to production using Continuous Delivery or Continuous Deployment (with Projects)



Amazon SageMaker Model Registry

Centrally catalog & manage model versions



Key Concepts

Hierarchy →

Model Registry

Model Group	Model Group	Model Group
ML Problem: *Abalone Age*	***ML Problem:*** *Customer Churn*	***ML Problem:*** *(n)*
vn	vn	v1
v1	v2	
	v1	

aws

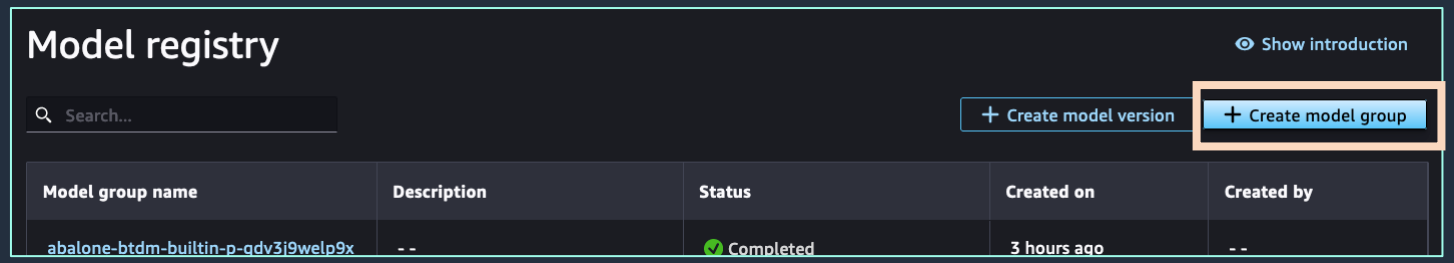
Amazon SageMaker Model Registry

Centrally catalog & manage model versions

How it works

1 Create Model Group →

SageMaker Studio



~OR~

programmatically

```
create_model_package_group_response = sm_client.create_model_package_group(**model_package_group_input_dict)
print('ModelPackageGroup Arn : {}'.format(create_model_package_group_response['ModelPackageGroupArn']))
```

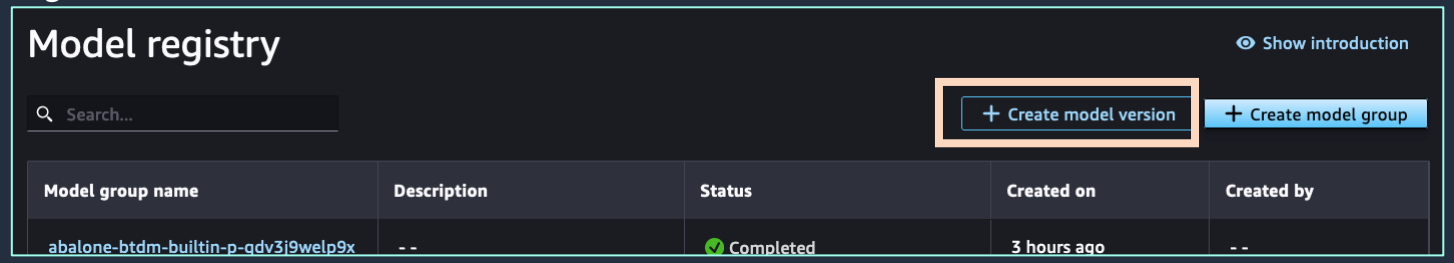
Amazon SageMaker Model Registry

Centrally catalog & manage model versions

How it works…

2 Register a Model Version →

SageMaker Studio



~OR~

programmatically

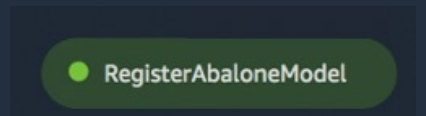
```
create_model_package_response = sm_client.create_model_package(**create_model_package_input_dict)
model_package_arn = create_mode_package_response["ModelPackageArn"]
print('ModelPackage Version ARN : {}'.format(model_package_arn))
```

~OR~

within a Pipeline, using RegisterModel Step:

```
step_register = RegisterModel(
    name="RegisterAbaloneModel",
    estimator=xgb_train,
    model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
    content_types=["text/csv"],
    response_types=["text/csv"],
    inference_instances=["ml.t2.medium", "ml.m5.large"],
    transform_instances=["ml.m5.large"],
    model_package_group_name=model_package_group_name,
    approval_status=model_approval_status,
    model_metrics=model_metrics,
)
```

=



Amazon SageMaker Model Registry

Centrally catalog & manage model versions

Example Workflow

1. Data Scientist registers a model SageMaker model registry.

   in PendingManualApproval state

vn
v2
v1

2. ML Eng/QA approves or rejects the model based on model metrics and other information

ML Engineer/QA

Approval

3. CI/CD pipeline is triggered to deploy the updated model to stagging account

4. Test model and run quality assurances in stagging account

Manual approval

(Optional) Automatic Approval

Manual Approval

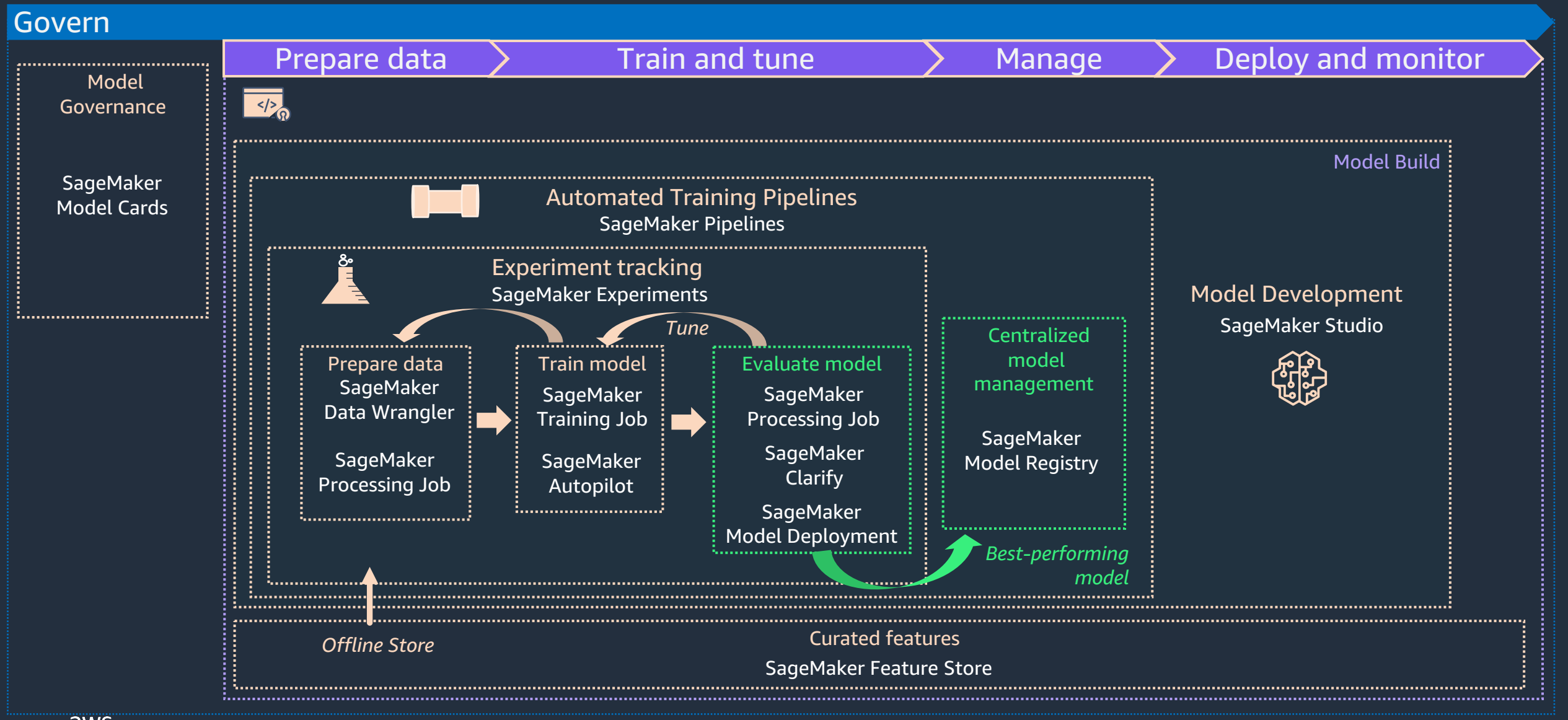
Model owner

5. CI/CD pipeline is triggered to deploy the updated model to production account

aws

MLOps Maturity Framework in Practice

Repeatable Phase Goal: Create a repeatable path to production



Repeatable Stage

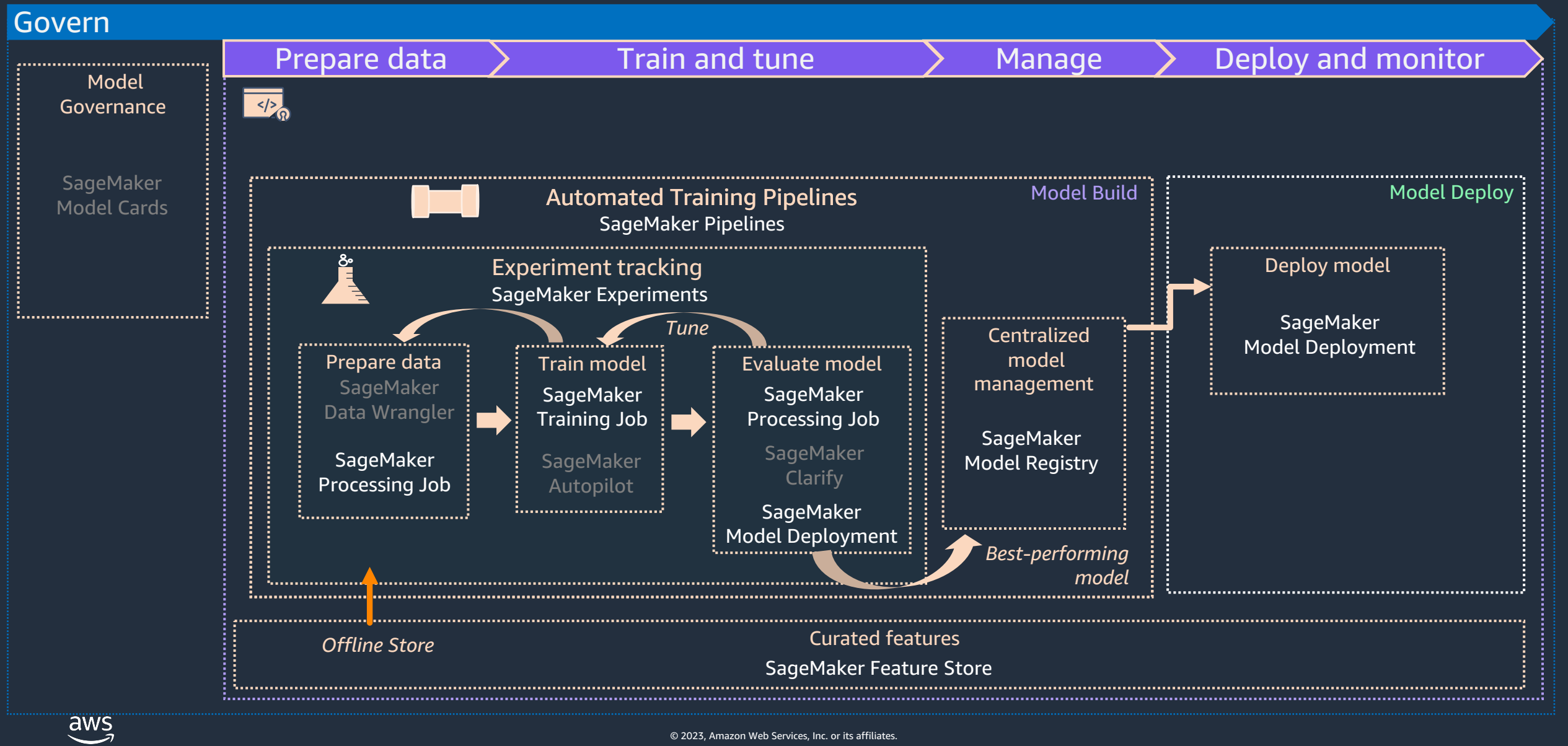
Key Technical Takeaways

1. Consider a repeatable mechanism to document model information that is transparent across stakeholders
2. Automate model build workflows for rapid development iterations, and model re-training
3. Implement a model registry to manage model versions at scale and bridge the gap between model build and model deploy activities

Lab2: Repeatable

Automate model build workflows & Manage model versions at scale

Lab 2: Amazon SageMaker - Features Used



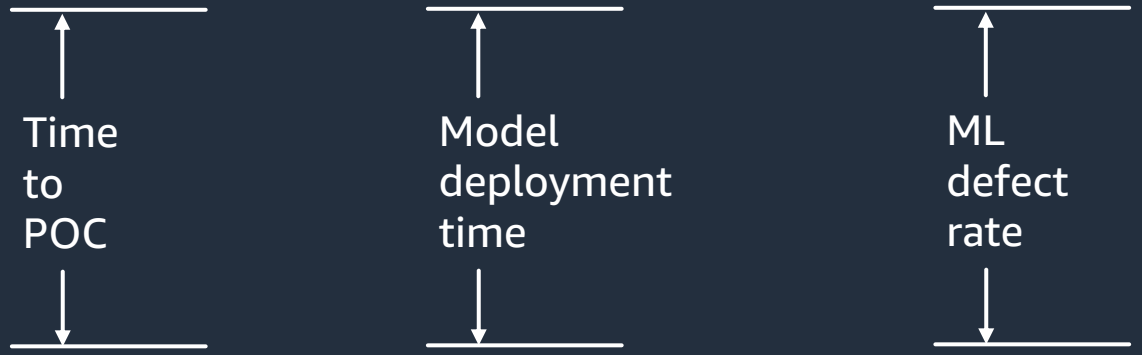
Reliable Stage:

Implementing reliable mechanisms to build, deliver, and manage models

MLOps Maturity Framework

Scaling ML through your MLOps journey

Key Performance Indicators

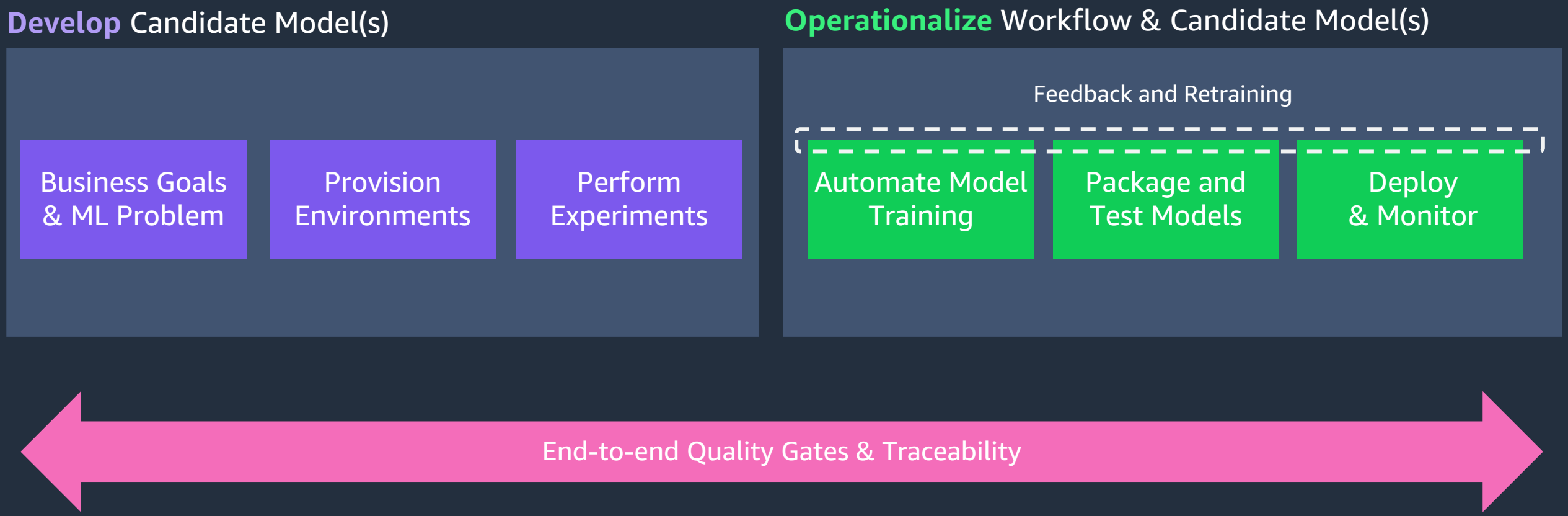


Initial	Repeatable	Reliable
Key **Technology** Capabilities		
• Standardize experimentation environments & processes	• Automate model build, and deployment tasks	• Integrate CI/CD practices into ML pipelines for traceability, improved quality and lineage tracking
• Automate governed access to resources	• Centralize management of models	• Incorporate automated quality gates
• Track experiments	• Standardize source code & artifact repositories ➔	• Automate continuous monitoring with feedback loops & re-training
• Centralize management of feature data		

aws

MLOps Maturity Framework in Practice

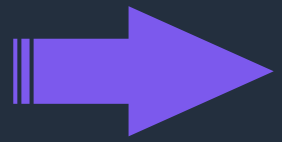
Reliable: Creating managed workflows optimized for quality & traceability

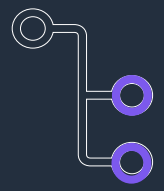


Moving from Repeatable to Reliable Maturity Level

Reliable: Creating managed workflows optimized for quality & traceability

How can I ensure quality of ML workflows and models running in production environments?





Enforce version control your code, data and models

Implement continuous deployment/delivery pipelines

Consider advanced deployment strategies to validate model performance and reduce risk

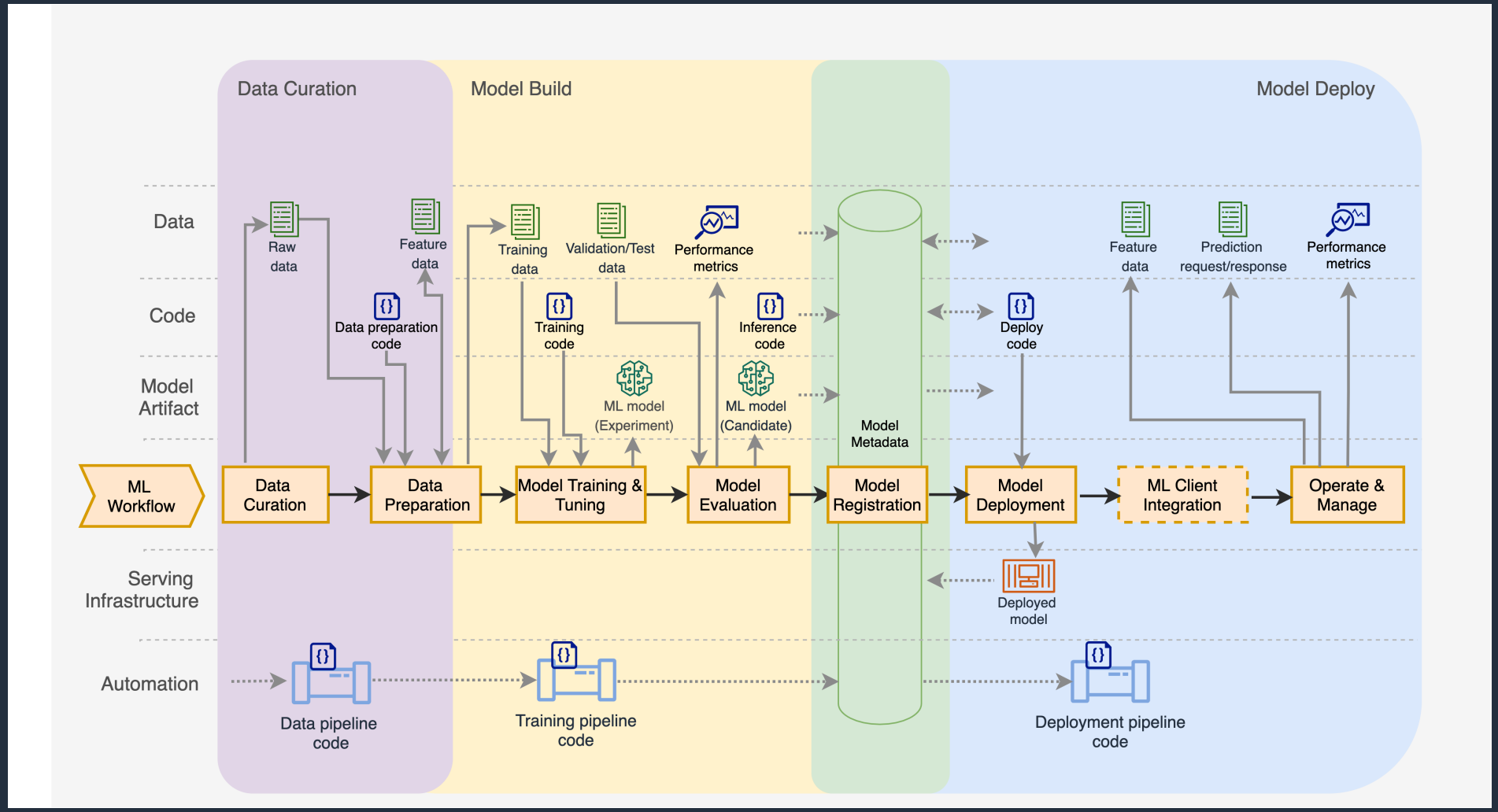
Create automated quality gates and automate business process spanning end-to-end workflows

Implement continuous monitoring and dashboards for transparency into model performance

Moving from Repeatable to Reliable Maturity Level

Reliable: Creating managed workflows optimized for quality & traceability

Without standardization, automation, and CI/CD, this is
A LOT to track and manage to be able to reliably reproduce and manage

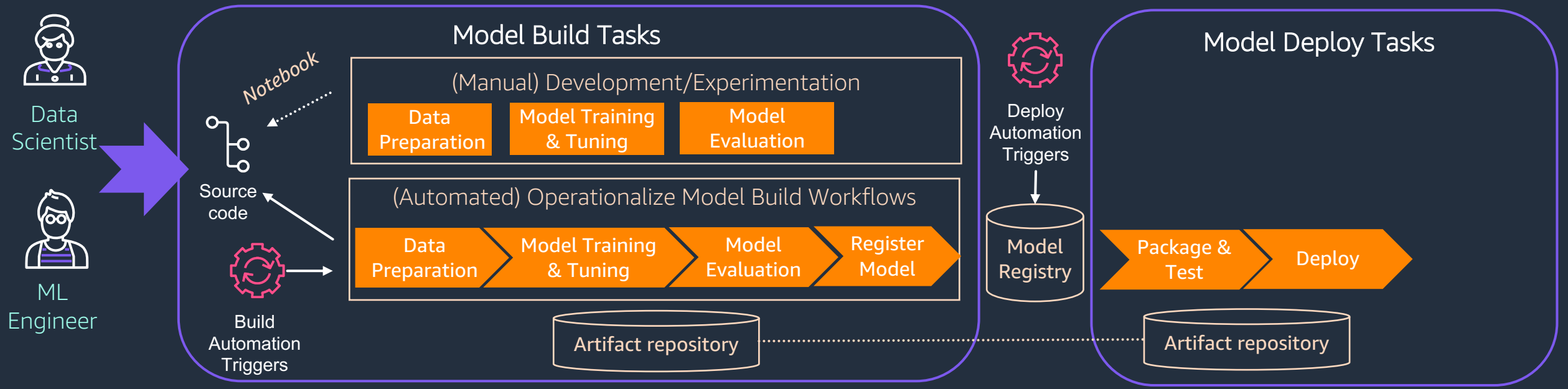


Reliable Stage:

Standardizing ML Projects

Creating a reliable path to production

Challenge #1: Standardize ML workflows and reduce manual hand-offs



Key Technical Challenges

- Identifying, implementing, and enforcing required quality gates & business processes across ML lifecycle
- Automate the hand-off between model build and model deployment tasks

Technical Solutions

- Implement CI/CD practices that consider implications unique to machine learning

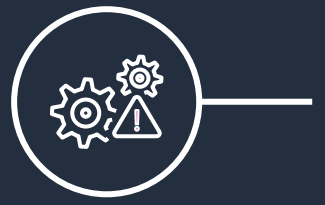
On AWS

- Amazon SageMaker Projects

aws

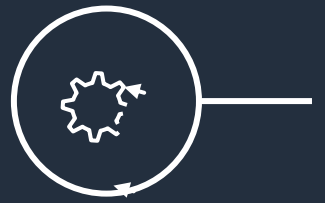
Amazon SageMaker Projects

Standardize machine learning workflows through MLOps project templates



Incorporate CI/CD practices
Automatically create standardized workflows that incorporate CI/CD practices through preconfigured templates

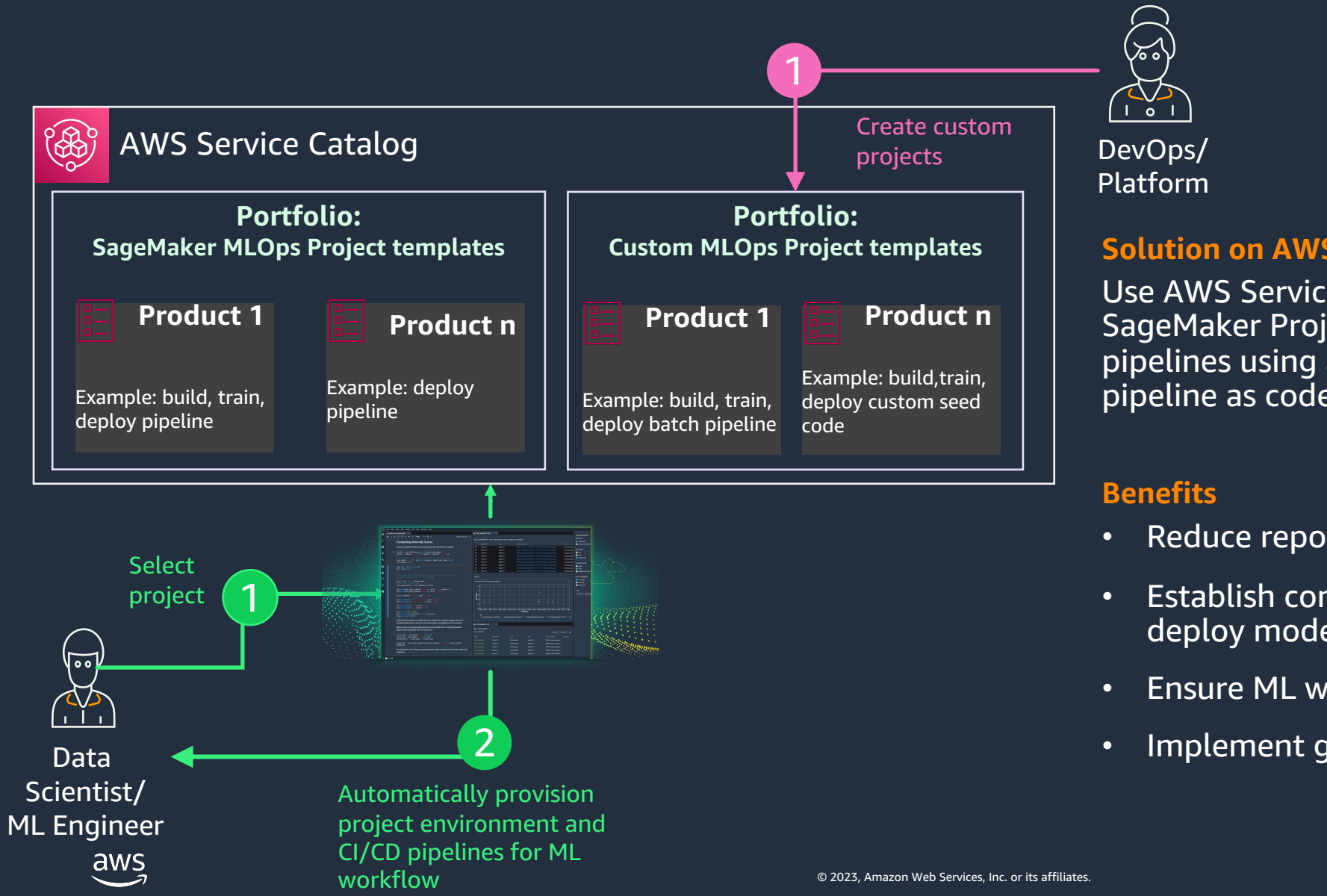
Standardize workflows across teams
Create custom workflows specific to your own processes and standards



Integrate with 3rd party tooling
Ability to integrate with common 3rd party tooling used for CI/CD

Amazon SageMaker Projects

Standardize ML workflows using MLOps templates



Solution on AWS

Use AWS Service Catalog, along with Amazon SageMaker Projects, to set up and provision ML pipelines using a combination of IaC, CaC, PaC, and pipeline as code

Benefits

- Reduce repository and CI/CD pipeline setup time
- Establish consistent mechanism to build and deploy models
- Ensure ML workflow traceability
- Implement guardrails and quality gates via code

Amazon SageMaker Projects

Standardize ML workflows using MLOps templates

Template Options

Built-In: SageMaker Managed

1. Build, Train, Deploy
   - Using AWS Developer Tools (AWS CodeCommit, AWS CodePipeline)
   - Using AWS CodePipeline & 3rd Party Git Repos
   - Using Jenkins & 3rd Party Git Repos
2. Build (Image Build), Train, Deploy
3. Build, Train
4. Deploy

Organization: Custom

- Create custom project to implement your own organizational standards for common workflow patterns

- Common reasons to create custom projects:
  - Workflows spanning multiple AWS accounts
  - Customize workflows for common patterns
  - Standardize source code repositories
  - Standardize workflows using custom seed code

TIP

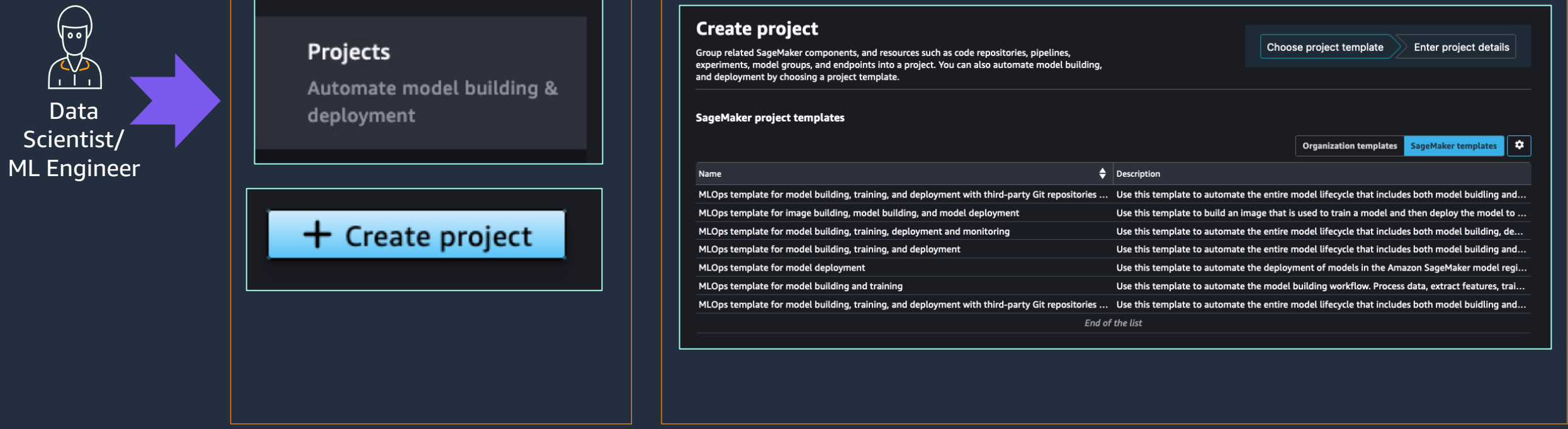
When creating custom Project templates, start with one of the built-in templates, or check out common custom project template examples here: https://github.com/aws-samples/sagemaker-custom-project-templates

aws

Amazon SageMaker Projects

Standardize ML workflows using MLOps templates

How it works

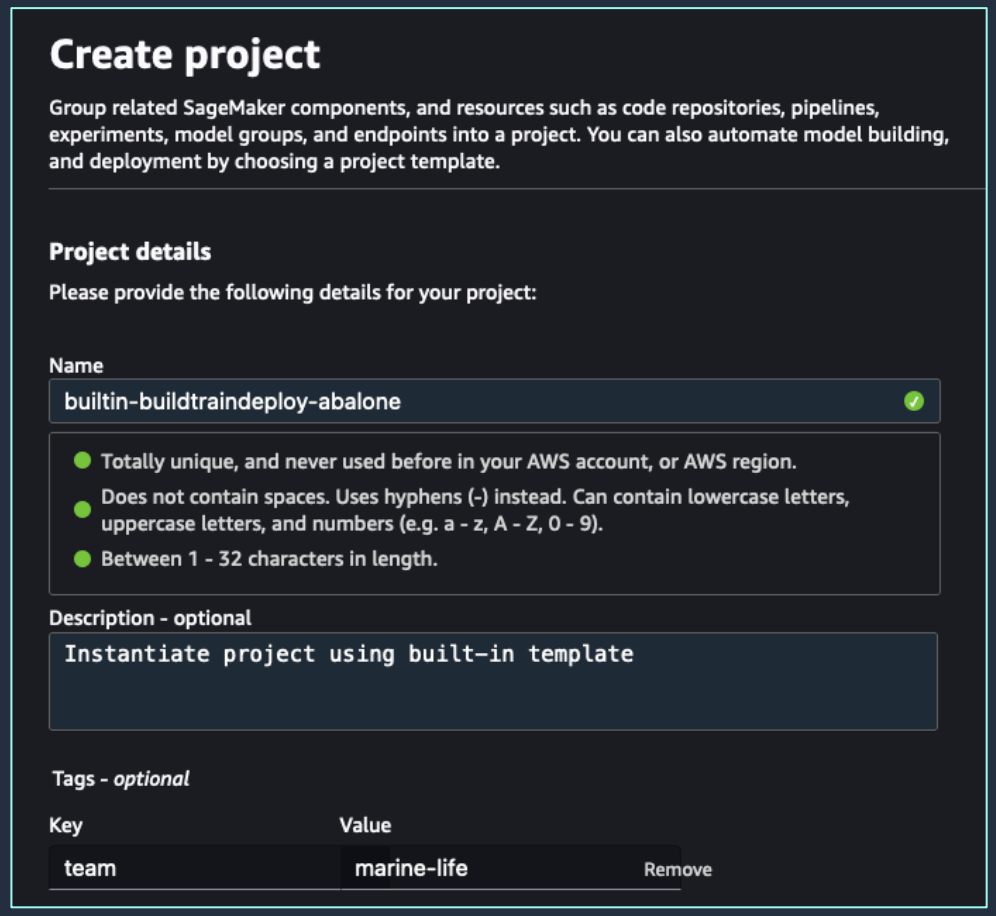


Amazon SageMaker Projects

Standardize ML workflows using MLOps templates

How it works…

3 *Enter Project Details*



- Ability to add any custom tags
- Tag propagation: Resources provisioned through built-in templates will have project-name & project-id Tags assigned. Example(s):
  - AWS CodePipeline → Pipelines
  - AWS CodeBuild → Build Projects
  - SageMaker Pipelines → Pipeline
    - → Training & Processing Jobs (via AWS CodeBuild)

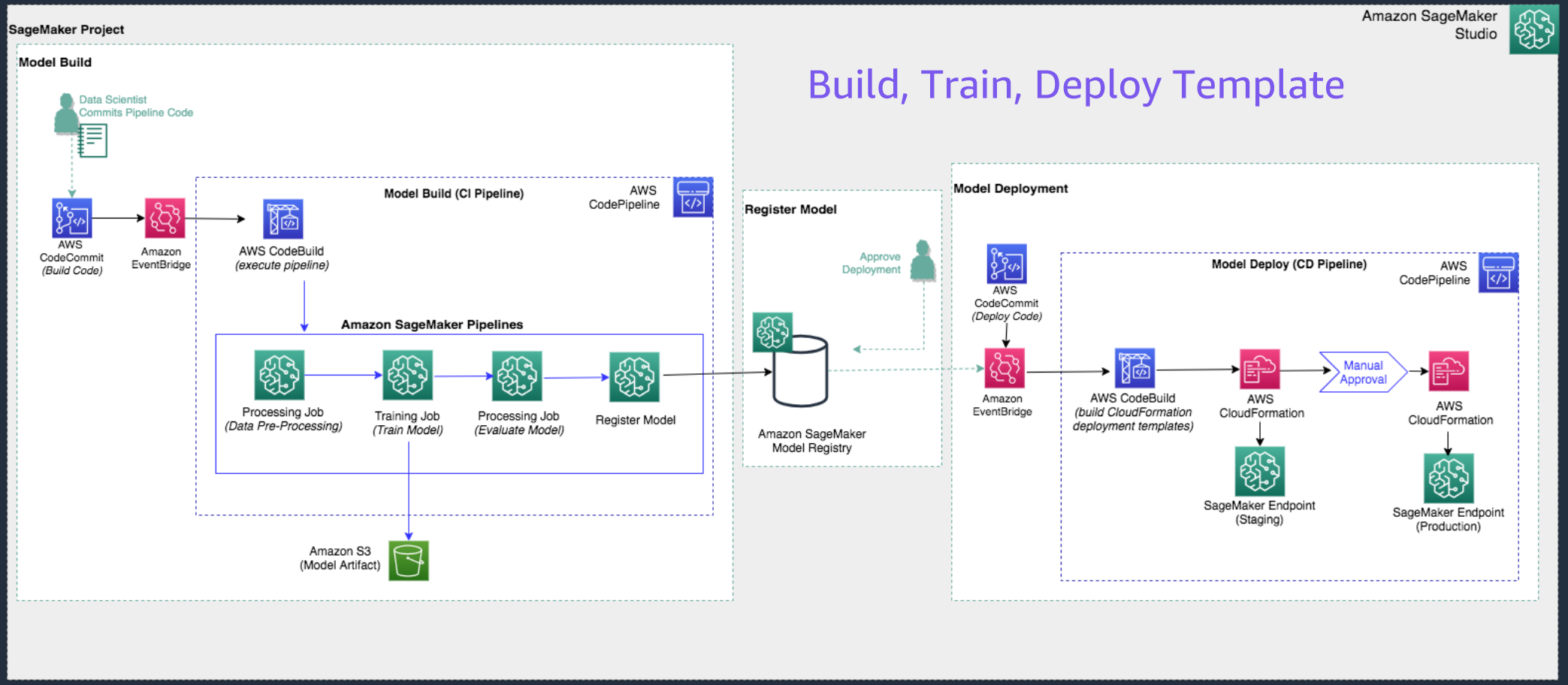
Everything needed to create a CI/CD Pipeline for Machine Learning gets automatically provisioned & configured for you……

Amazon SageMaker Projects

Standardize ML workflows using MLOps templates

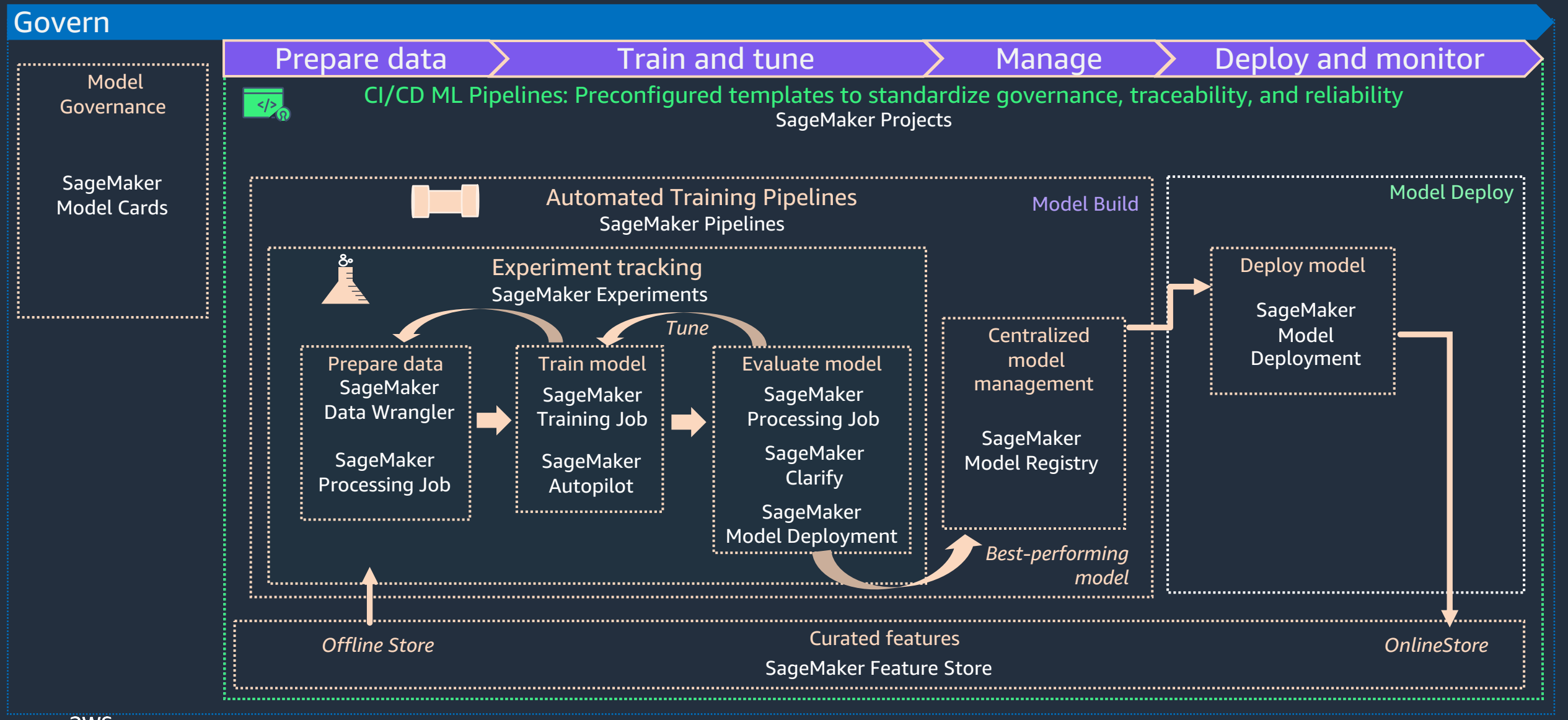
Automatic creation & configuration of CI/CD ML Pipeline

Reduce operational overhead in stitching together workflows



MLOps Maturity Framework in Practice

Reliable Phase Goal: Creating managed workflows optimized for quality & traceability

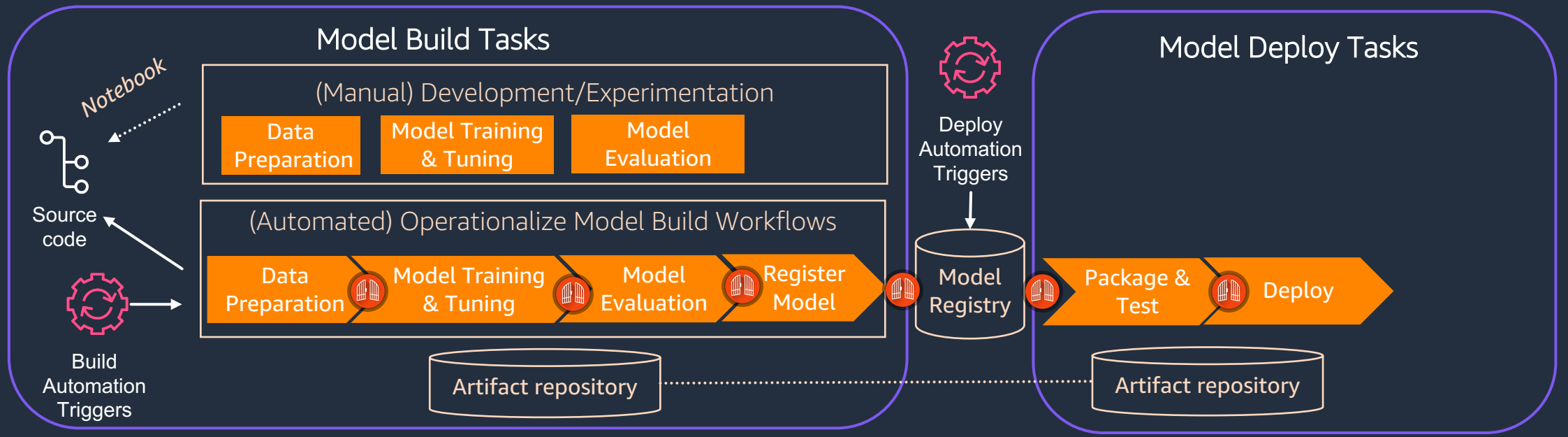


Reliable Stage:

MLOps Quality Gates

Implement automated quality gates across MLDC

Challenge #2: Identify and implement quality gates across the MLDC



Key Technical Challenges

- Identifying, implementing, and enforcing required quality gates & business processes across ML lifecycle

Technical Solutions

- Incorporate additional quality gates in your CI/CD pipeline

On AWS

- Amazon SageMaker Projects, Pipelines with built-in and custom steps
- Amazon SageMaker Clarify

aws

Considerations for quality gates

Reliable: Creating managed workflows optimized for quality

Data Quality Assurance

Before the model is trained, consider automated data validation for ➔

Data format - file format, structure, naming

Data Quality – missing values, feature distribution, feature completeness

Label annotation and data feasibility

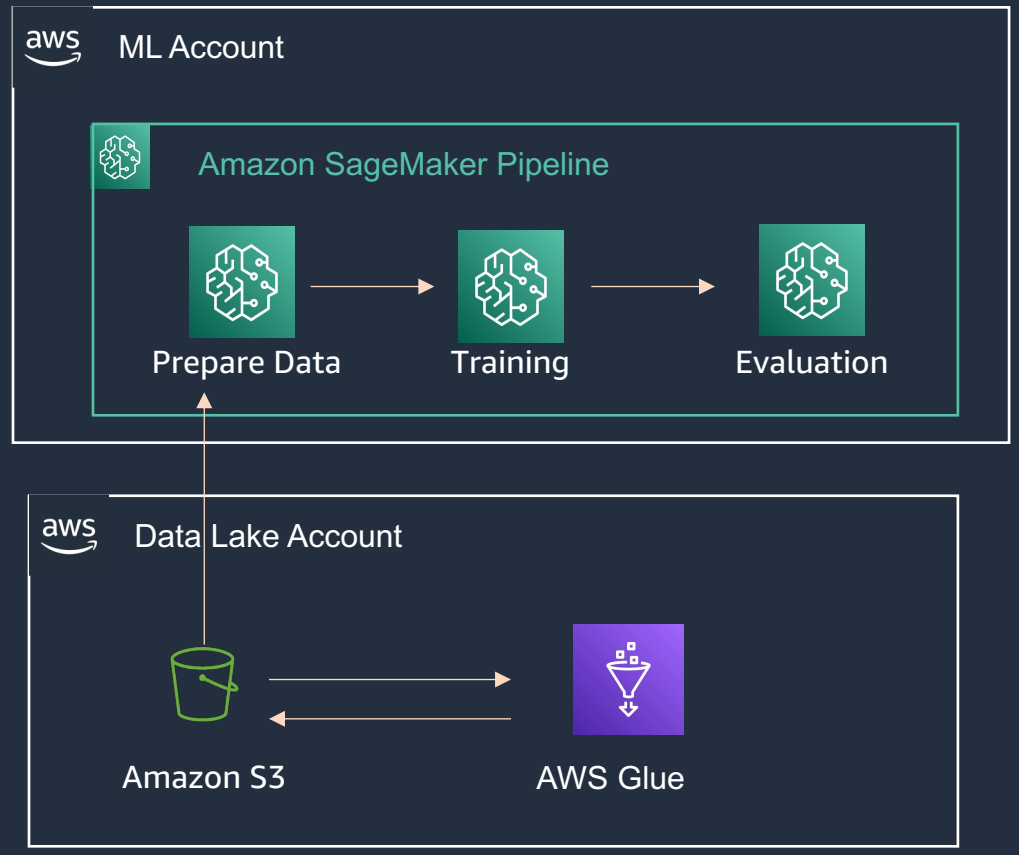
Considerations for quality gates

Reliable: Creating managed workflows optimized for quality

New!!

AWS Glue Data Quality continuous monitoring

DELIVER HIGH QUALITY DATA ACROSS YOUR DATA LAKES AND DATA PIPELINES



- Automatic data quality rule recommendations based on your data
- Keep data quality high with ongoing data analysis and quality checking
- Data quality for datasets in your data lake and data pipelines
- Cost-effective to scale with pay-as-you-go billing, with no lock-in

In - Preview

Considerations for quality gates

Reliable: Creating managed workflows optimized for quality and traceability

Package & Code Quality Assurance

Scan packages for common vulnerabilities & promote high coding standards to avoid bugs ➔

Vulnerability Scans

Scan containers and libraries for common vulnerabilities

Code style

indentation, formatting, documentation

Testing

unit test, integration test, end-to-end test

Performance

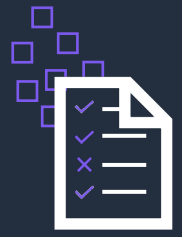
compilation of code, stress test, resource utilization optimization

Considerations for quality gates

Reliable: Creating managed workflows optimized for quality and traceability

Model & System Testing

Validate data, model and infrastructure for your ML workflow ➔



Data & Feature Tests

- Data validation
- Feature Importance
- Policy-compliant Data and feature Pipeline
- Unit tests for feature engineering process

Model Tests

- ML Metrics
- A/B experiments
- Performance Validation
- Fairness/Bias checks

ML System Tests

- Latency and throughput testing
- Compute resource utilization & right-sizing

Integration Tests

- Data, Modeling, Orchestration, 3rd Party tools integrations
- Ml consuming client integration tests

Amazon SageMaker Clarify

DETECT MODEL STATISTICAL BIAS AND UNDERSTANDING MODEL PREDICTIONS

Identify imbalances in data

Detect bias during data preparation

Check your trained model for statistical bias

Evaluate the degree to which various types of bias are present in your model

Explain overall model behavior

Understand the relative importance of each feature to your model's behavior

Explain individual predictions

Understand the relative importance of each feature for individual inferences

Detect drift in statistical bias and model behavior over time

Provide alerts and detect drift over time due to changing real-world conditions

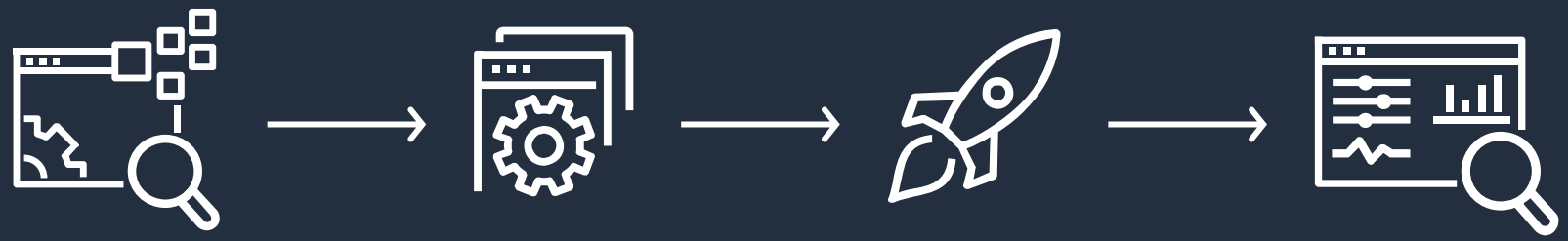
Generated automated reports

Produce reports on bias and explanations to support internal presentations

Amazon SageMaker Clarify

Incorporate model explainability and statistical bias detection

Model explainability & statistical bias detection



Build

Perform statistical bias analysis during exploratory data analysis

Training

Conduct statistical bias and explainability analysis after training

Deploy

Explain individual inferences for model in production

Monitor

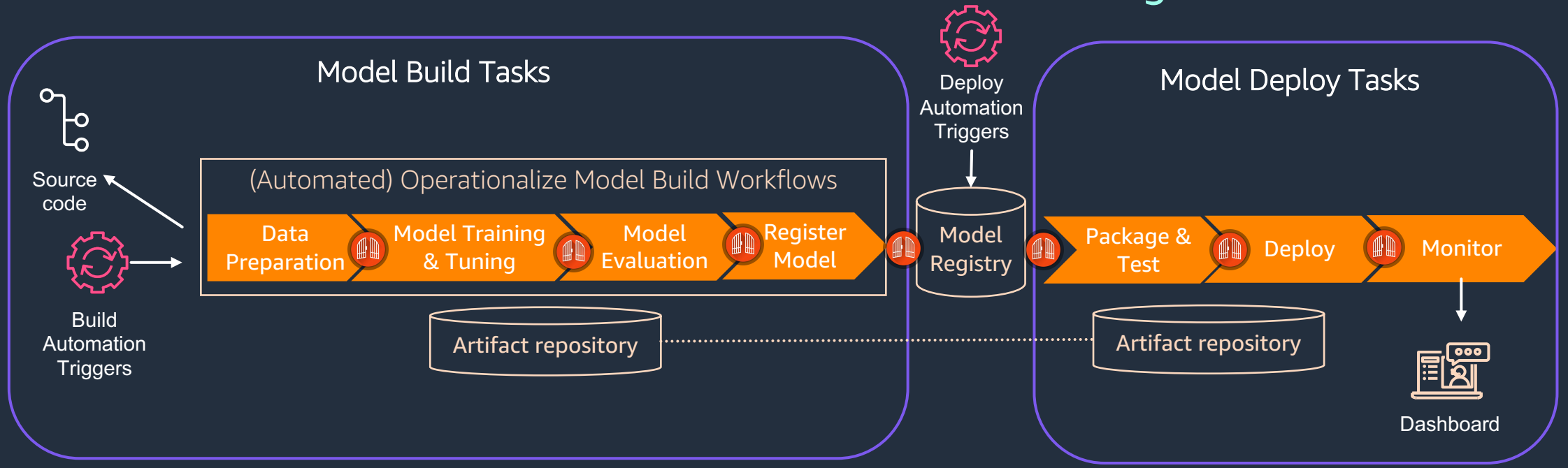
Validate statistical bias and relative feature importance over time

Reliable Stage:

Deployment Quality Gates

Minimize deployment risk & Continuous monitoring for ML

Challenge #3: Minimize risk with new model versions & monitoring ML workloads



Key Technical Challenges

- Monitoring and measuring model performance
- Deploying new model versions with reduced risk

Technical Solutions

- Incorporate advanced deployment strategies into CD pipelines
- Provide transparency into model performance across key stakeholders

On AWS

- Amazon SageMaker Model Monitor
- Amazon SageMaker Model Dashboard
- Amazon SageMaker Model Deployment

aws

Considerations for monitoring

Reliable: Creating managed workflows optimized for quality

Continuous Monitoring & Logging

Consider all monitoring dimensions ➔

Alerts & Dashboards

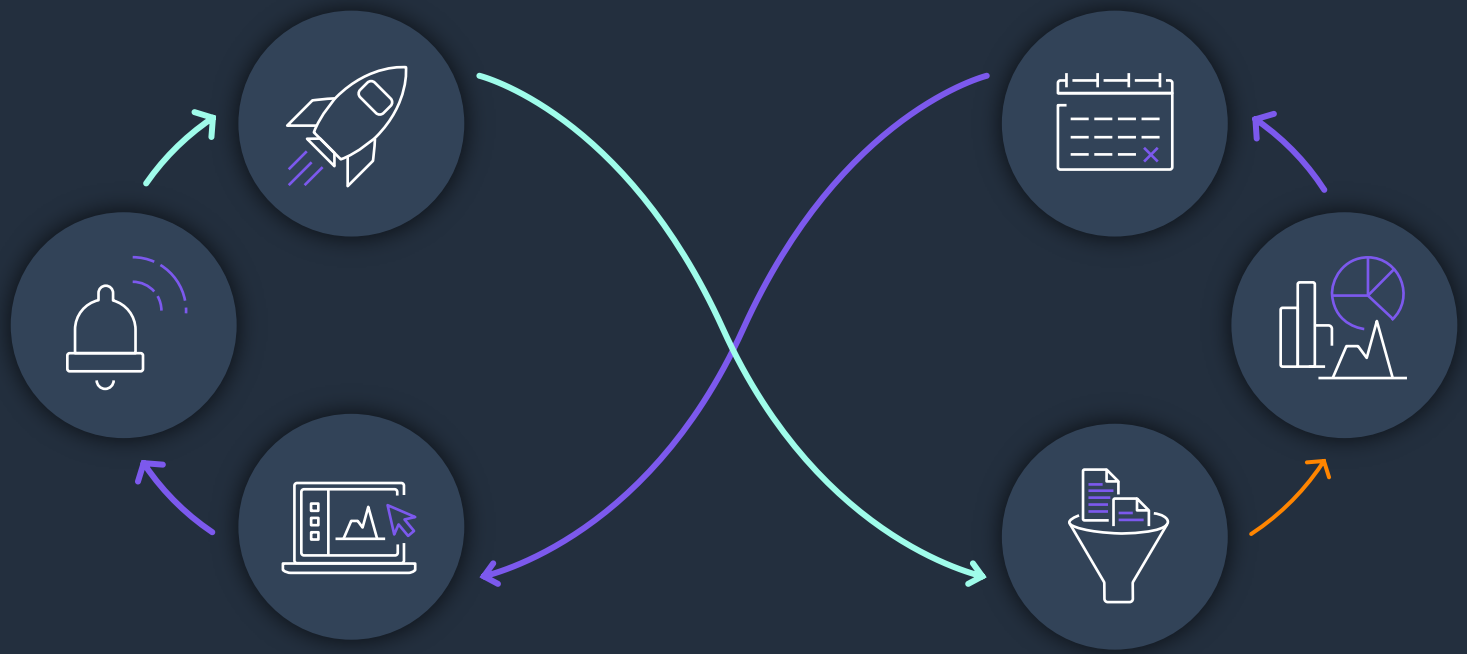
Stakeholder transparency

Amazon SageMaker Model Monitor

Automatically detect & mitigate drift

Easily incorporate model monitoring into your ML workflows

Concept & Data Drift



Deploy trained model
Enable data capture on SageMaker Endpoint

Schedule monitoring jobs
Detect and monitor data quality, model accuracy, bias, and explainability drift

CloudWatch alerts
Retrain models/ update data

Generate baseline
Statistics and constraints

View drift monitoring results
Reports in S3, visualize metrics in Studio, analyze in Notebooks

Collect ground truth
Merge with predictions

aws

Amazon SageMaker Model Monitor

Automatically detect & mitigate drift

Supported Monitor Types

Detect divergence in data

Real time data capture from endpoints

Define Baseline

Pre-built container for analysis

Support custom analysis

Type, Num Present, Num Missing

Mean, Sum, Std Dev

KLL Sketch

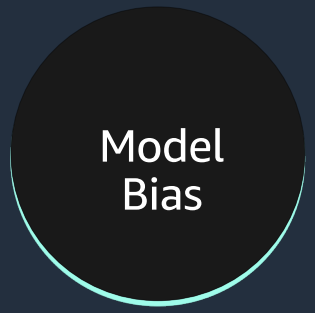
Detect quality degradation over time

Merge predictions with ground truth

Compare predictions to ground truth

Generate reports and violations

MAE, RMSE, F1



Track model balance

Overfitting

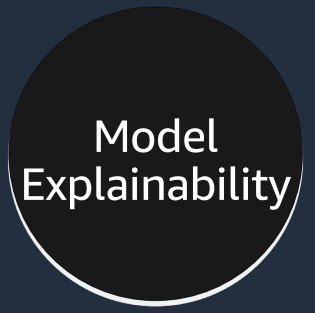
Underfitting

Class Imbalance

DPL

KL Divergence

LP-Norm



Feature Attribution

How much each feature contributed to predictions

Shapley Values

aws

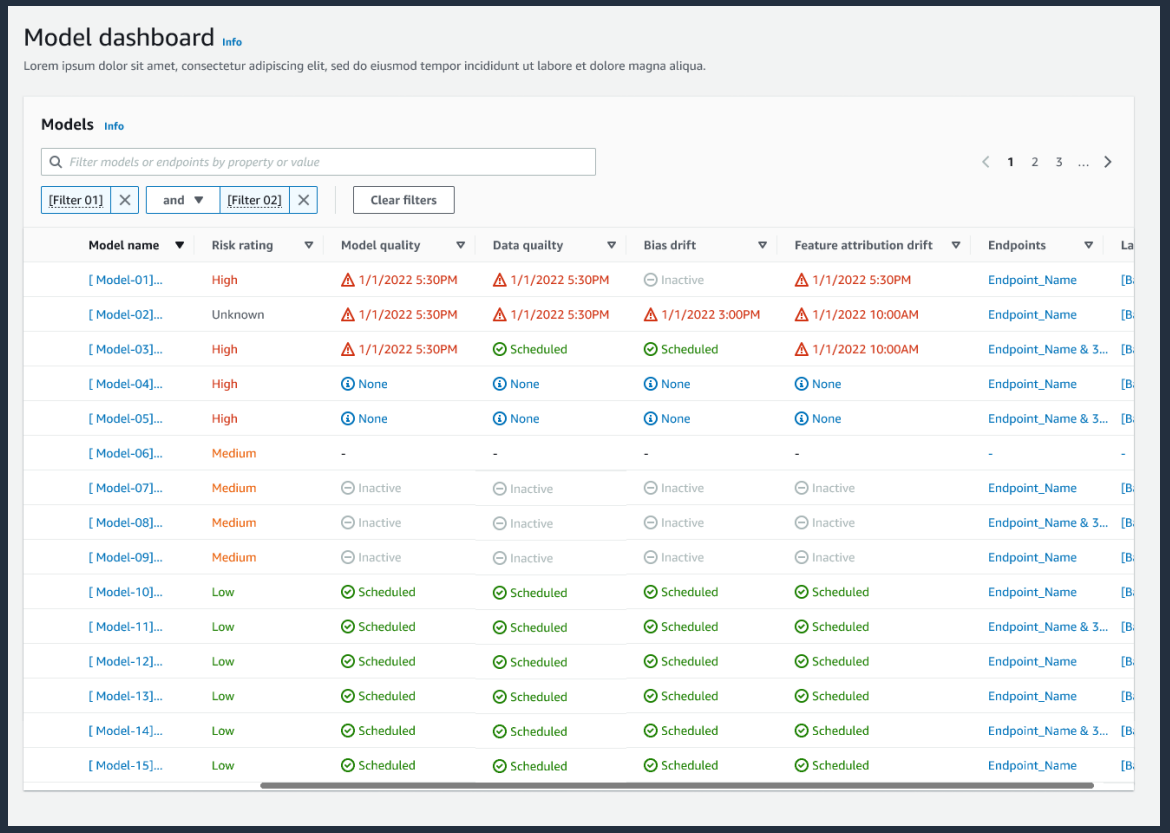
Amazon SageMaker Model Dashboard

Provide transparency into model performance

Recent release (re:invent 2022)

Unified monitoring across all your models in production

- Single pane of glass view for models, endpoints, batch transform jobs, and monitoring jobs
- Insights into model performance at endpoints and batch transform jobs
- View alerts for violations, missing, and inactive monitors
- Supports model quality, data quality, bias drift, and feature attribution drift

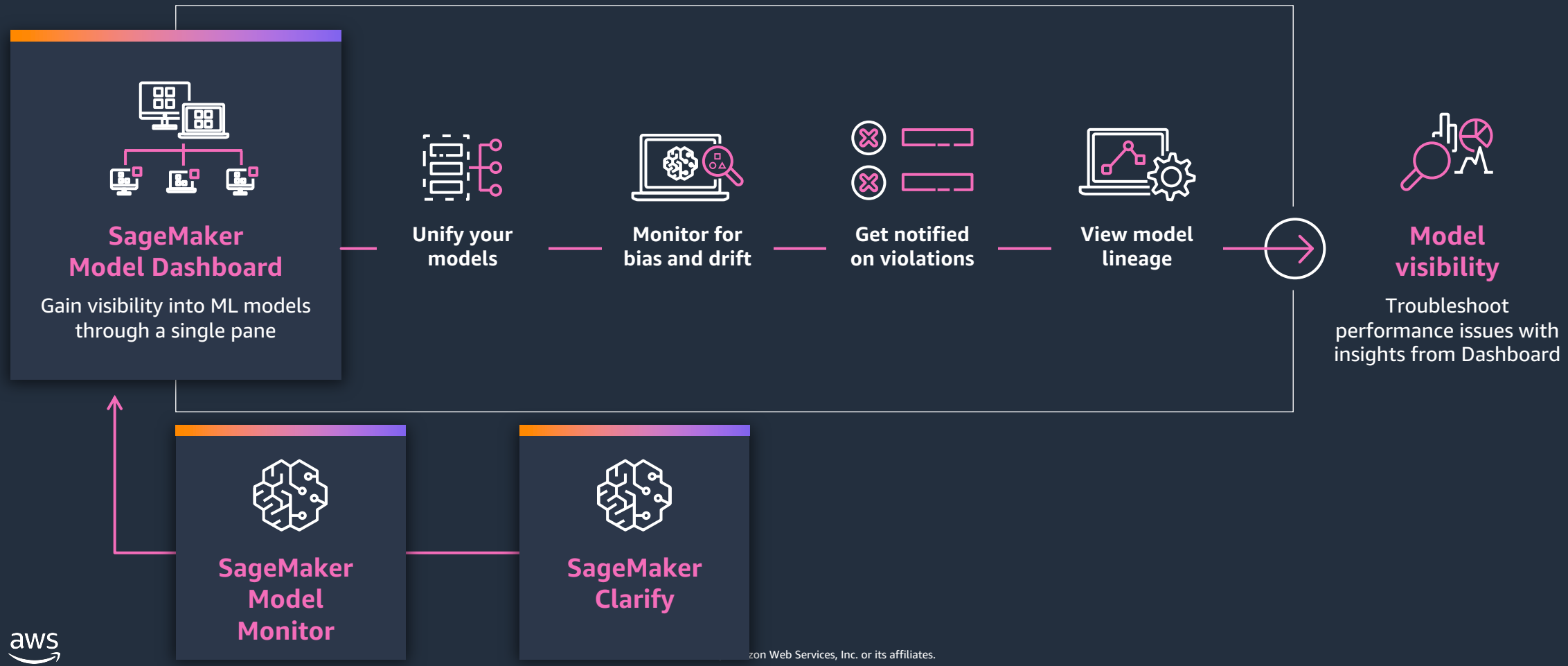


Amazon SageMaker Model Dashboard

Provide transparency into model performance

Recent release (re:invent 2022)

How it works



Reliable Stage:

MLOps Deployment Strategies

Considerations for model deployment strategies

Reliable: Creating managed workflows optimized for quality and traceability

Model Deployment Strategies

Validate new model versions with advanced deployment strategies to minimize deployment risk ➔

Shadow Test

Validate model versions in production without returning its outputs to users

A/B Tests

Validate model versions in production serving both outputs to users

Blue/Green Deployments

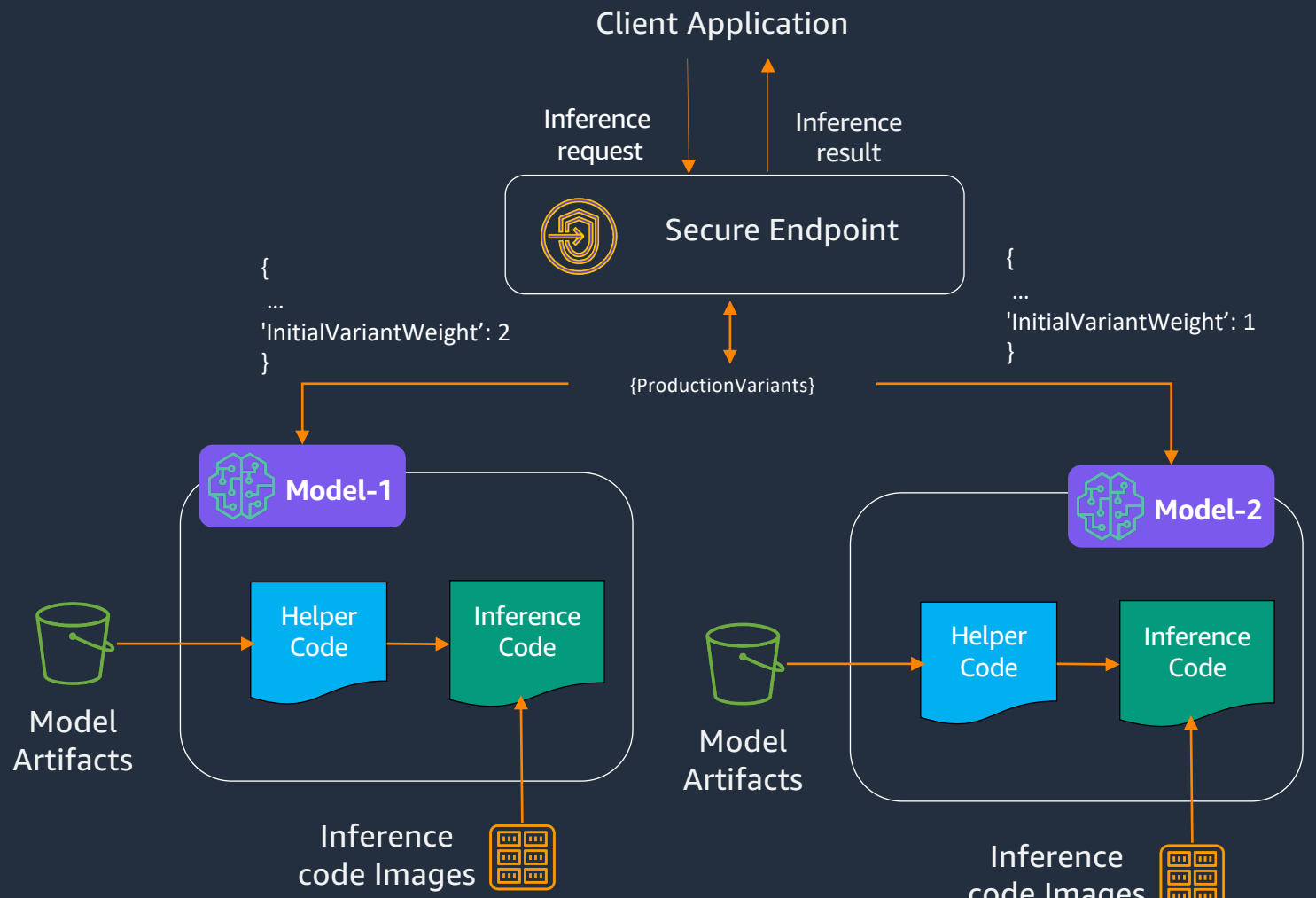
Securely update models in production with quick rollback functionality

Amazon SageMaker Model Deployment

Automate advanced model deployment strategies

A/B Testing
Compare model versions

- 1-10 Production Variants (Model Versions)
- All models must have the same I/O schema
- Endpoint Modification w/o service disruption
- Perform traffic distribution per variant based on weight or invoke a specific variant per request



Amazon SageMaker Model Deployment

Automate advanced model deployment strategies

Recent release (re:invent 2022)

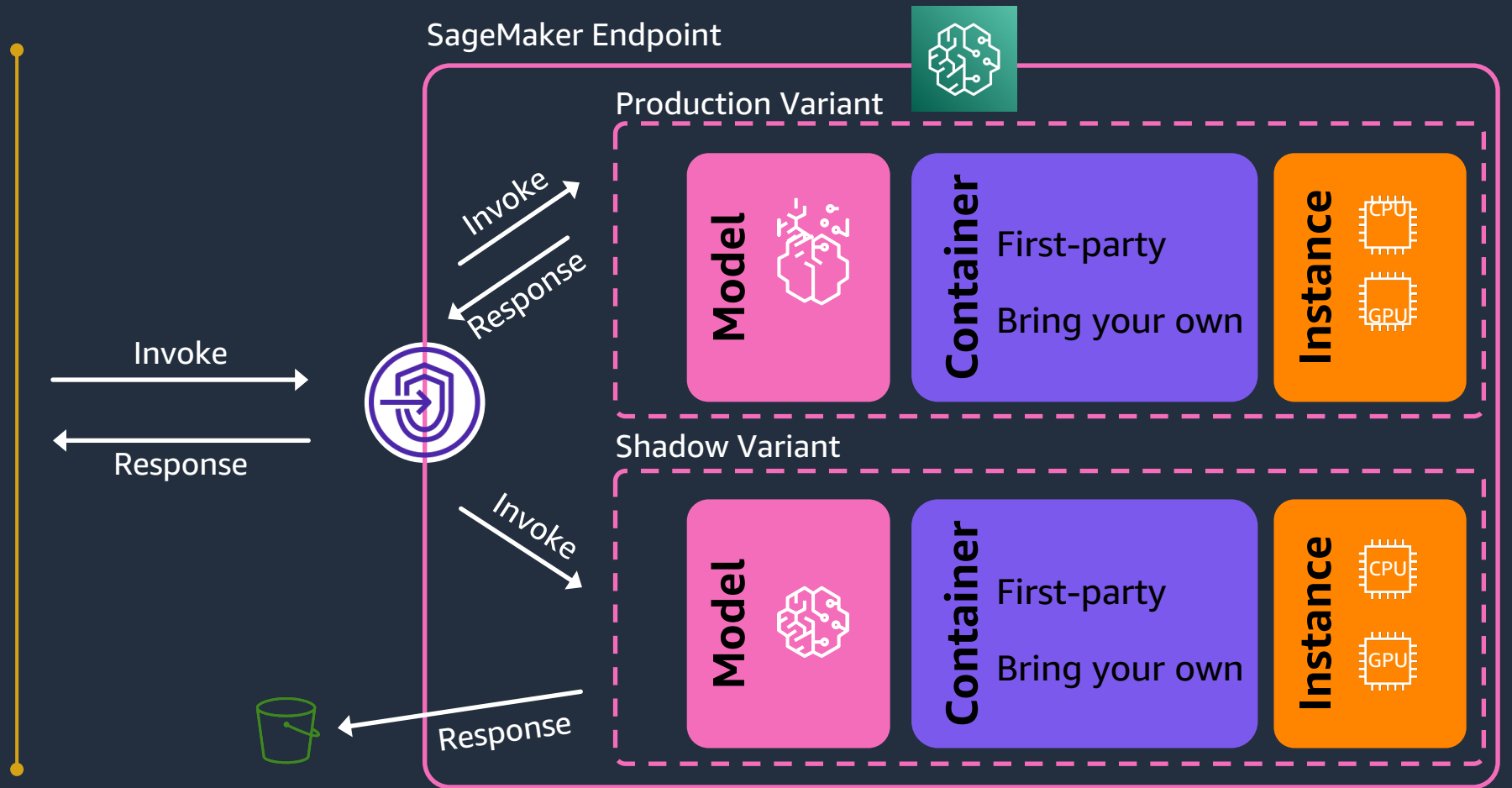
Shadow Testing

Validate new models in parallel to the model serving production traffic

SageMaker takes care of mirroring requests

Start small and dial up to control costs

Accessible through AWS console, CLI, APIs



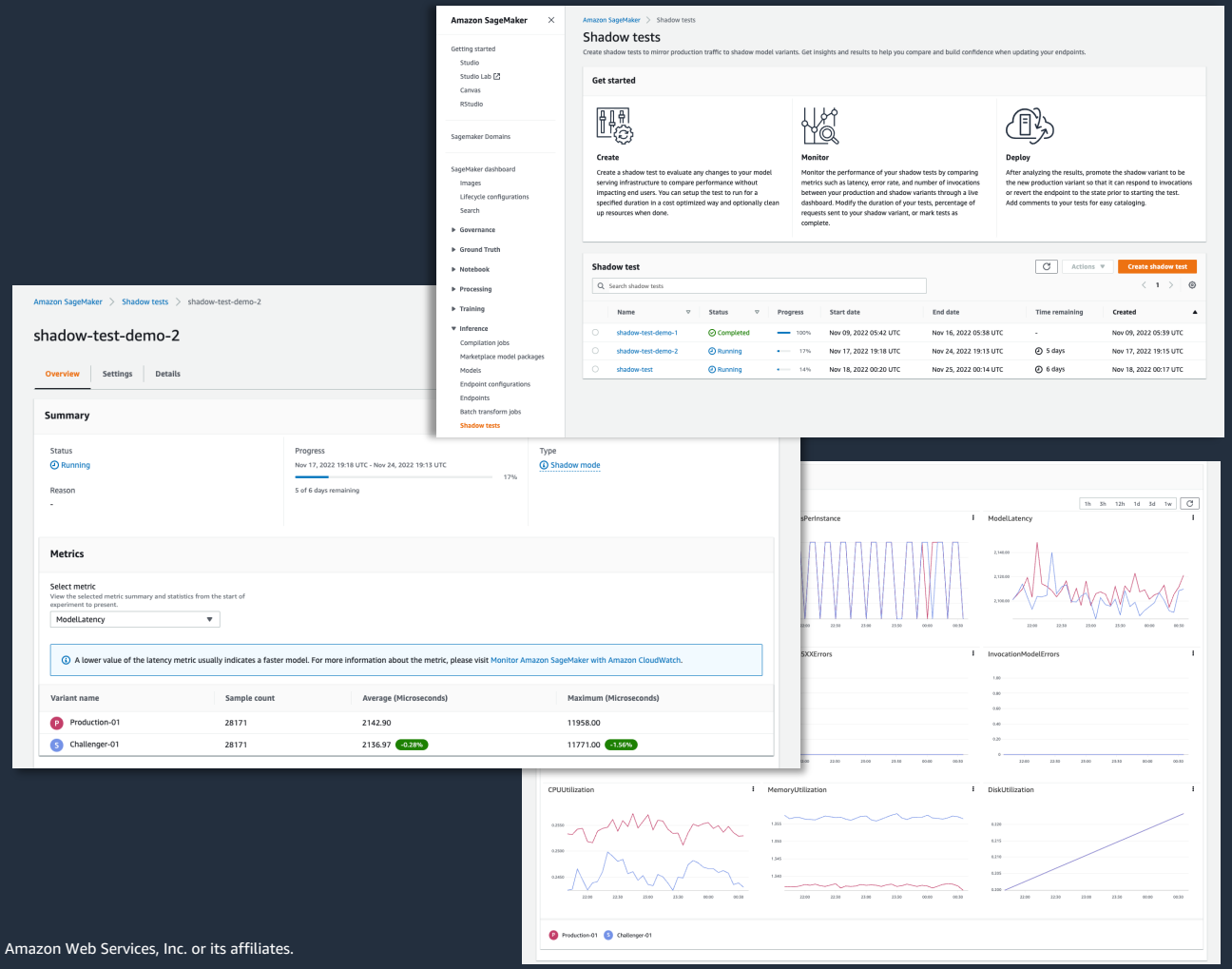
Amazon SageMaker Model Deployment

Automate advanced model deployment strategies

Shadow Testing

Tools to manage shadow tests

- Set up a test for a predefined duration
- Monitor operational performance through a live dashboard
- Deploy models with confidence or rollback



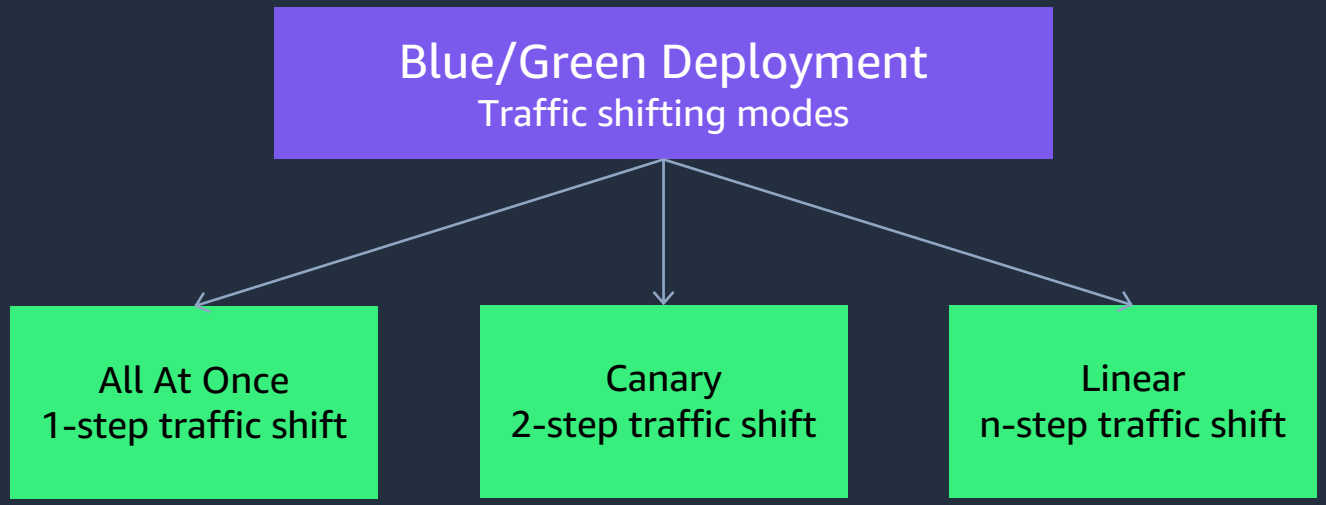
Amazon SageMaker Model Deployment

Automate advanced model deployment strategies

Deployment Guardrails

Safe deployments into production environments

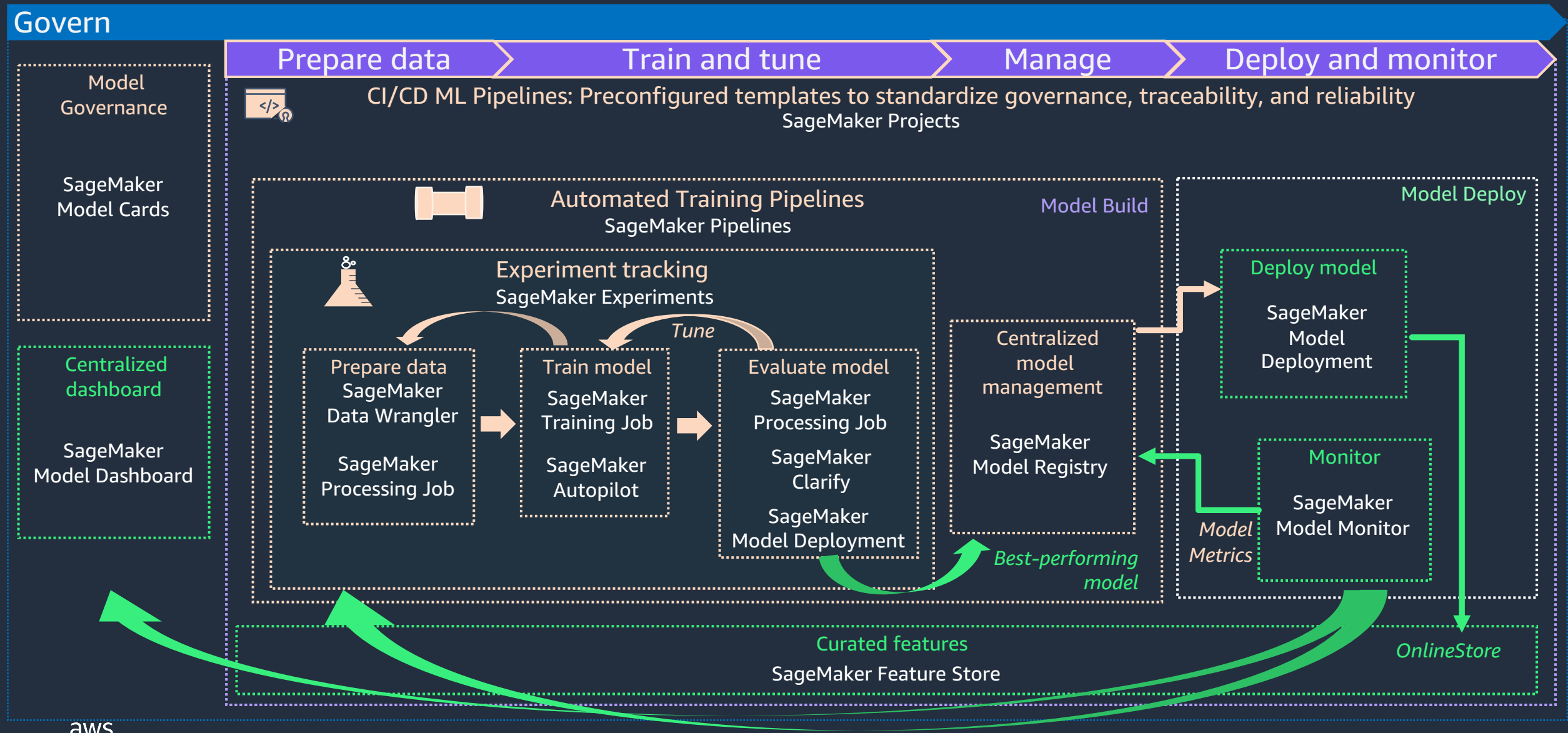
VALIDATE YOUR MODEL BEFORE PRODUCTION UPDATE



- Traffic shifting modes, such as canary and linear
- Deployment safety while updating production environments
- Built-in safeguards such as auto-rollbacks
- Fully managed deployment
- Visibility: Track the progress of your deployment

MLOps Maturity Framework in Practice

Reliable Phase Goal: Creating managed workflows optimized for quality & traceability



Reliable Stage

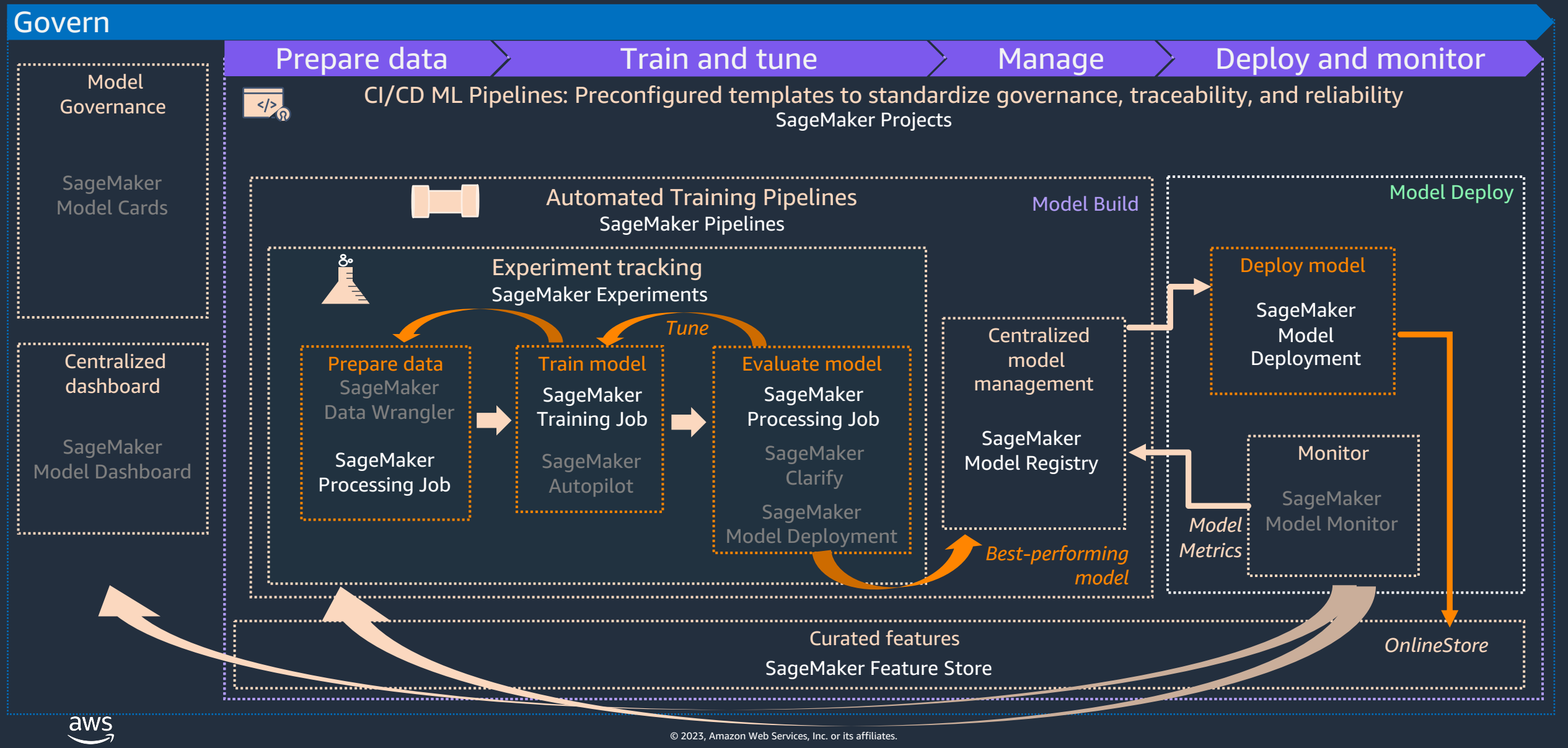
Key Technical Takeaways

1. Implement CI/CD practices to incorporate versioning of all key inputs and artifacts for model lineage and reproducibility and to streamline deployment
2. Identify & implement automated (or manual when necessary) quality gates that span the end-to-end workflow
3. Implement model monitoring to detect signals of data or concept drift
4. Consider advanced deployment strategies to minimize risk when deploying a new model version

Lab3: Reliable

Reliably build, deliver, and manage models at scale

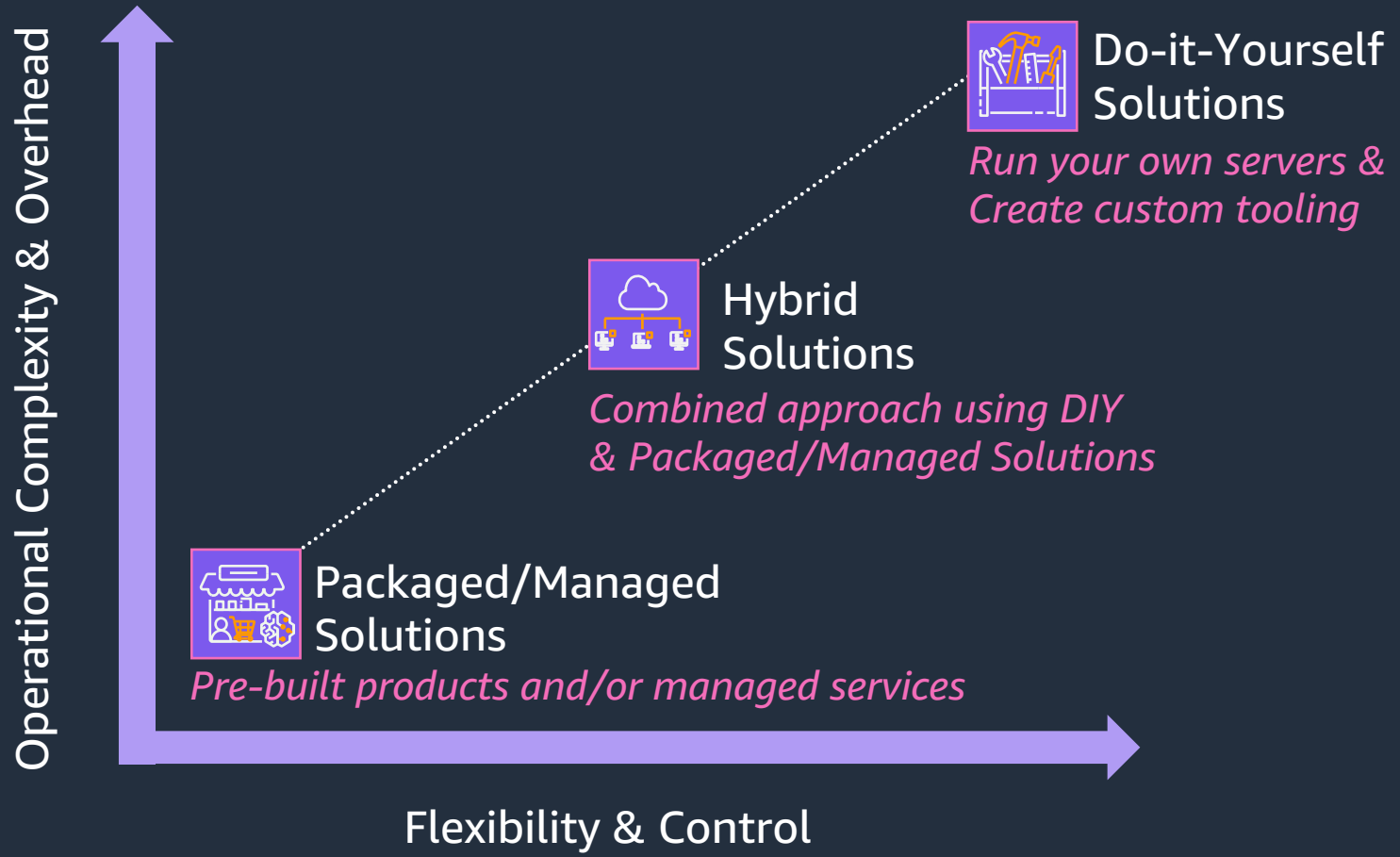
Lab 3: Amazon SageMaker - Features Used



AWS &
Third-party integrations

MLOps Technical Implementation Choices

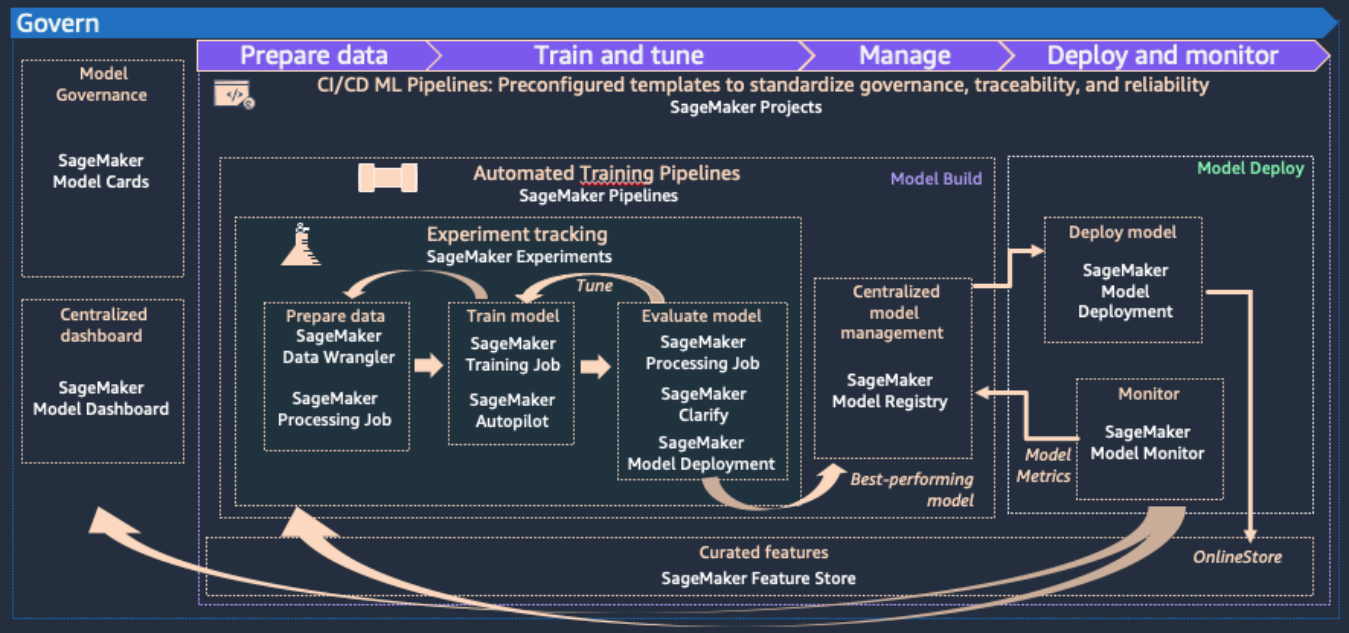
Understand Your Tradeoffs



Amazon SageMaker

Flexible across use cases & requirements

SageMaker offers the flexibility to utilize native features end-to-end….



…Or integrate with existing tooling as needed based on existing investments, skillsets, or organizational requirements

Tooling Considerations

Integrating SageMaker Jobs into MLOps Tooling

SageMaker native

- SageMaker Experiments
- SageMaker projects
- SageMaker Pipelines
- SageMaker Model Registry
- SageMaker Feature Store
- SageMaker Model Monitor

PLUS: SageMaker jobs to perform the tasks in your workflow (ex. SageMaker Training)

Integrate with SageMaker components or jobs

Do-it-yourself**

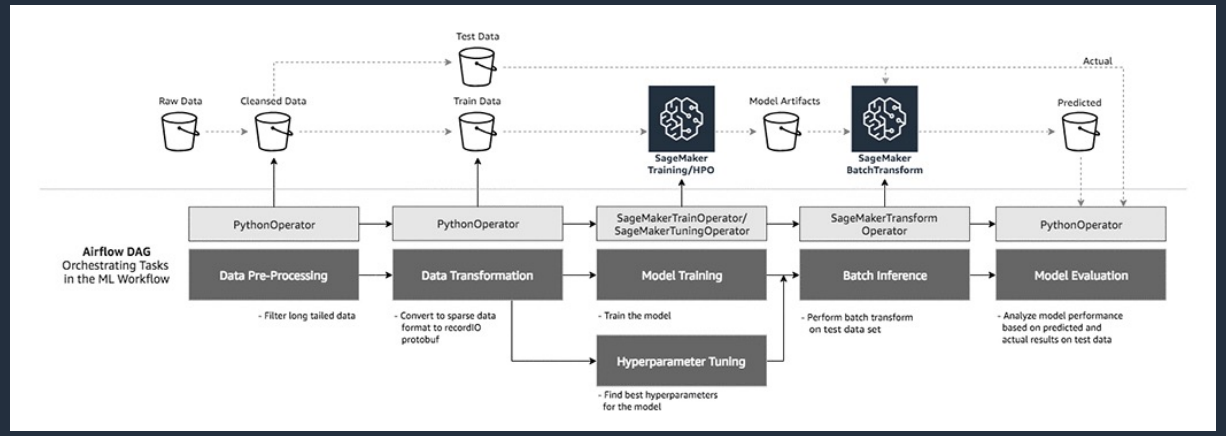
- Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
- AWS Step Functions
- DVC
- W&B
- Kubeflow
- mlflow
- fiddler
- Apache Airflow

***Note: This list is not inclusive of every do-it-yourself combination*

Tooling Considerations: Airflow

Integrating SageMaker Jobs into MLOps Tooling

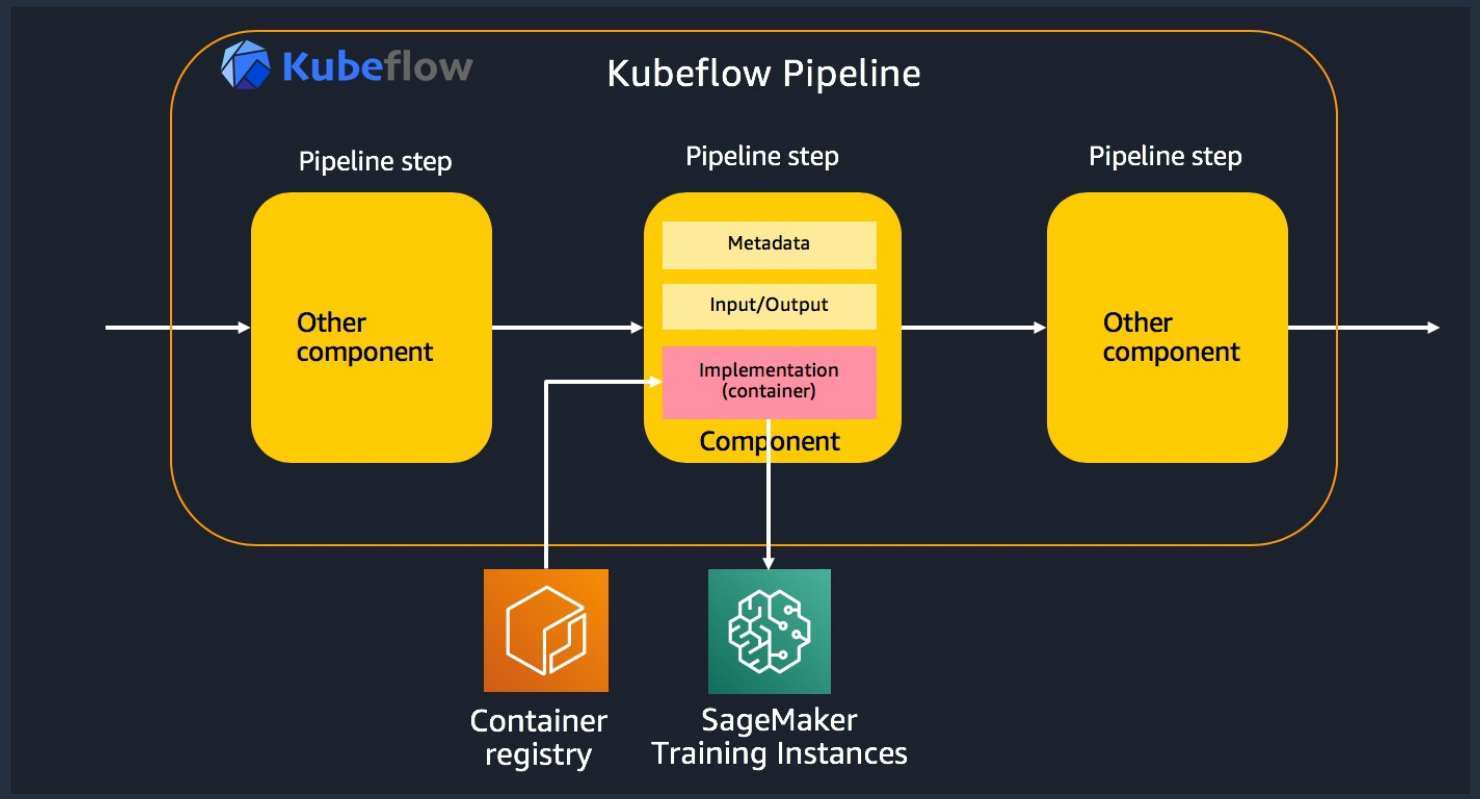
- **SageMaker Operators**
  - SageMakerTrainingOperator
  - SageMakerTuningOperator
  - SageMakerModelOperator
  - SageMakerTransformOperator
  - SageMakerEndpointConfigOperator
  - SageMakerEndpointOperator
  - SageMakerAutoMLOperator
  - SageMakerCreateExperimentOperator
  - SageMakerStartPipelineOperator
  - SageMakerRegisterModelVersionOperator



Tooling Considerations: Kubeflow

Integrating SageMaker Jobs into MLOps Tooling

SageMaker Components for Kubeflow Pipelines



Key Features:

- Run production model pipelines from open source in Kubeflow Pipelines
- Take advantage of faster and cheaper training with SageMaker using managed spot training, distributed training, and security compliance
- Run inference using SageMaker Batch or Hosting

Key Technical Takeaways

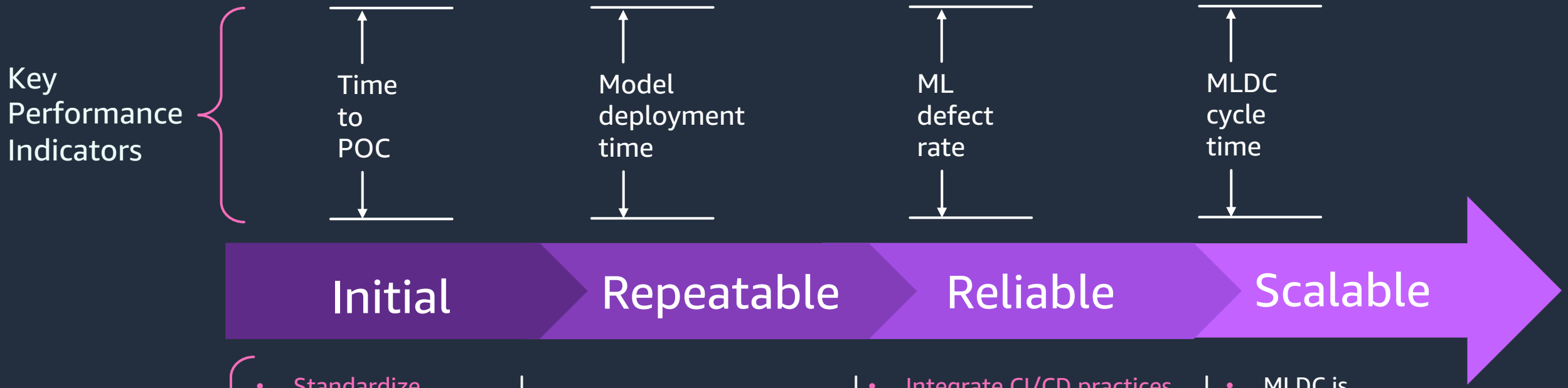
1. Consider the number of integrations you will need to manage
2. Standardize on core components such as Model Registry or Feature Store
3. Aim to utilize purpose built tooling
4. Utilize managed tooling to reduce operational overhead and server management

Scalable Stage:

Implementing standardized ML platforms & workflows across teams

MLOps Maturity Framework

Scaling ML through your MLOps journey



Key **Technology** Capabilities

Initial

- Standardize experimentation environments & processes
- Automate governed access to resources
- Track experiments
- Centralize management of feature data

Repeatable

- Automate model build, and deployment tasks
- Centralize management of models
- Standardize source code & artifact repositories ➔

Reliable

- Integrate CI/CD practices into ML pipelines for traceability, improved quality and lineage tracking
- Incorporate automated quality gates
- Automate continuous monitoring with feedback loops & re-training

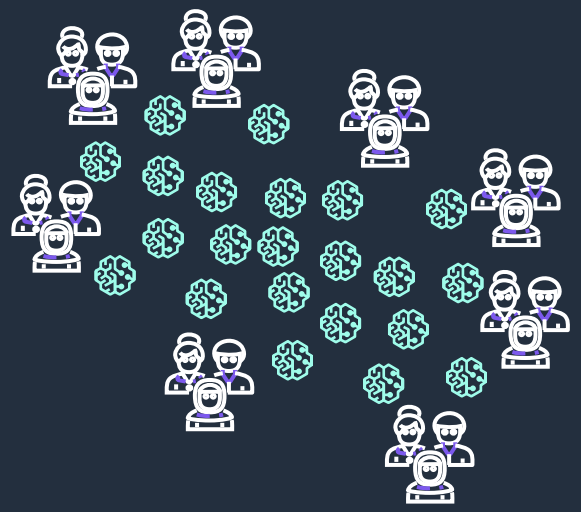
Scalable

- MLDC is reproducible through standardized tooling & templates shared across teams

aws

Moving from Repeatable to Scalable Maturity Level

Scalable: Shared best practices & provide a reproducible MLDC



What if you're supporting 100's/1000s of use cases? teams? tooling management & integrations?

Select platform and tooling choices that minimize operational overhead and total cost while maintaining IT controls

Consider standardized tooling

Consider readily available templates to streamline workflow

Governed ML Platform on AWS

Build & Govern ML workloads at scale

Agenda

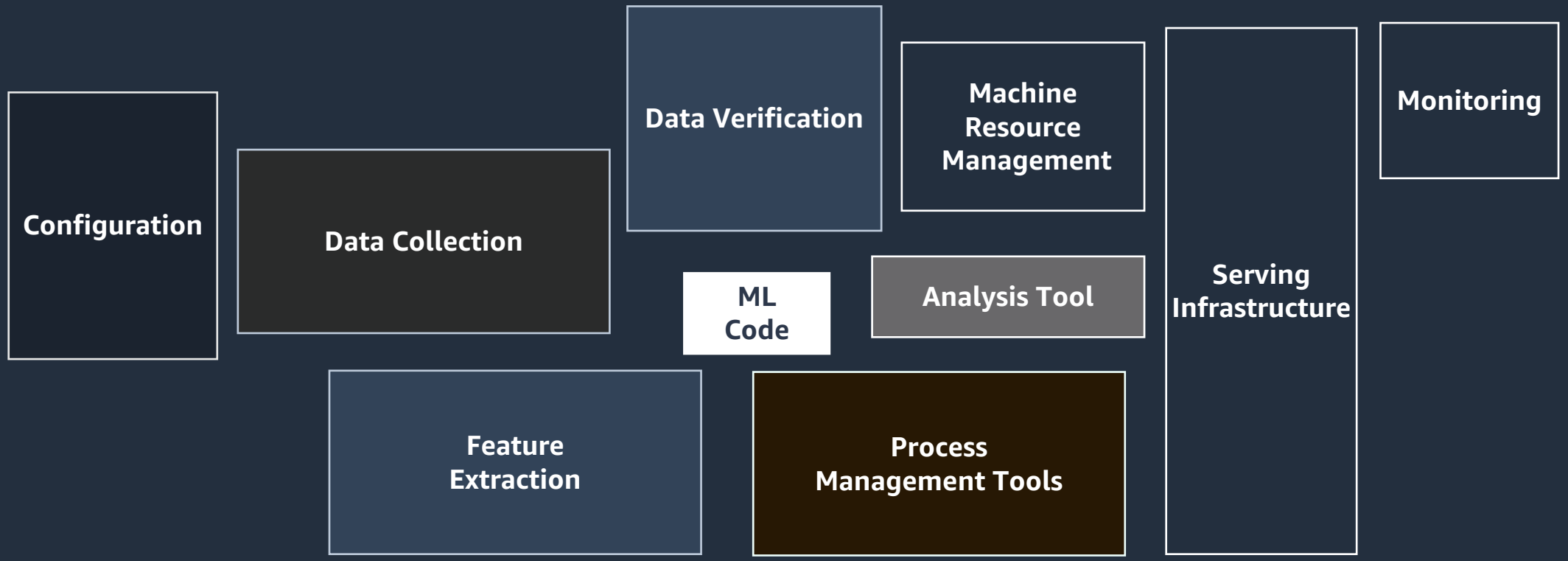
- Why build Governed ML Platform?
- Governed ML Platform Capabilities
- Governed ML Platform Reference Architecture
- Resources
- Q & A

ML and Path to Production

MACHINE LEARNING CODE AND DATA SCIENCE NOTEBOOKS…

ML and Path to Production

…IS ONE SMALL PART OF THE OVERALL DEPLOYMENT PICTURE



"Only a small fraction of real-world ML systems is composed of the ML code"

source: Hidden Technical Debt in Machine Learning Systems [D. Sculley, & al.] – 2015

https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

Challenges

Abstraction – Abstract infra setup and access controls

Scalability – Being able to scale resources for teams with isolation

Reliability – Perform intended function correctly and consistently at scale

Auditability – Logs, versions and dependencies of artifacts

Reproducibility – Need to recreate past experiments

Governability – Ability to centrally govern resources and manage risk

Governed ML Platform Capabilities

Build ML Foundations

- Operating Model
- Multi-Account Foundations
- Data Foundations
- Central Feature Store

Scale ML Operations

- Self-Service Onboarding
- Guided ML Workflow
- CI/CD Build & Deployment
- Central Model Registry

Observable ML

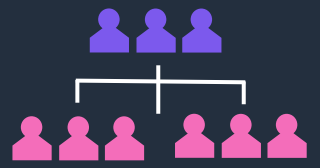
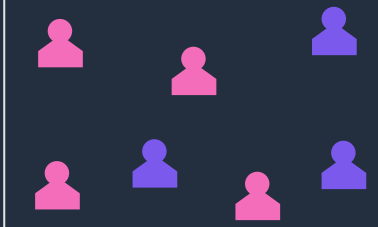
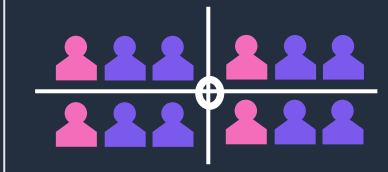
- Experiment Tracking & Lineage
- Centralizing Logging & Monitoring
- Model Governance
- Cost Control & Reporting

Secure ML

- Security Foundations
- Authentication & Authorization
- Data & Model Security
- Differential Privacy

1. Build Governed ML Platform Foundations

Operating Model

Team Structure	Definition
 Centralized	Data science activities are centralized within a single team or organization
 Decentralized	Data science activities are distributed across different business functions or divisions with no central governance
 Federated	Shared services and governance functions managed by centralized team and each business unit team manages their data science functions

Accounts are the building blocks

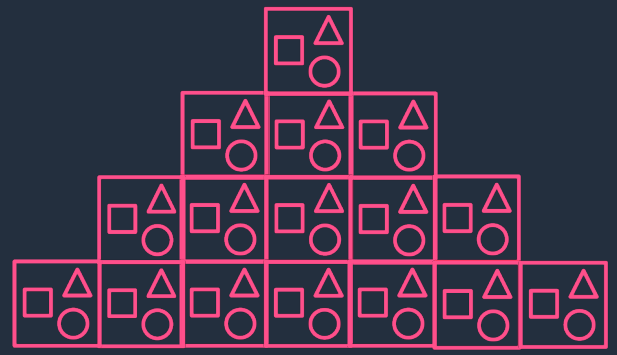
Think of AWS resources (instances, storage, etc.) as material and accounts as bricks

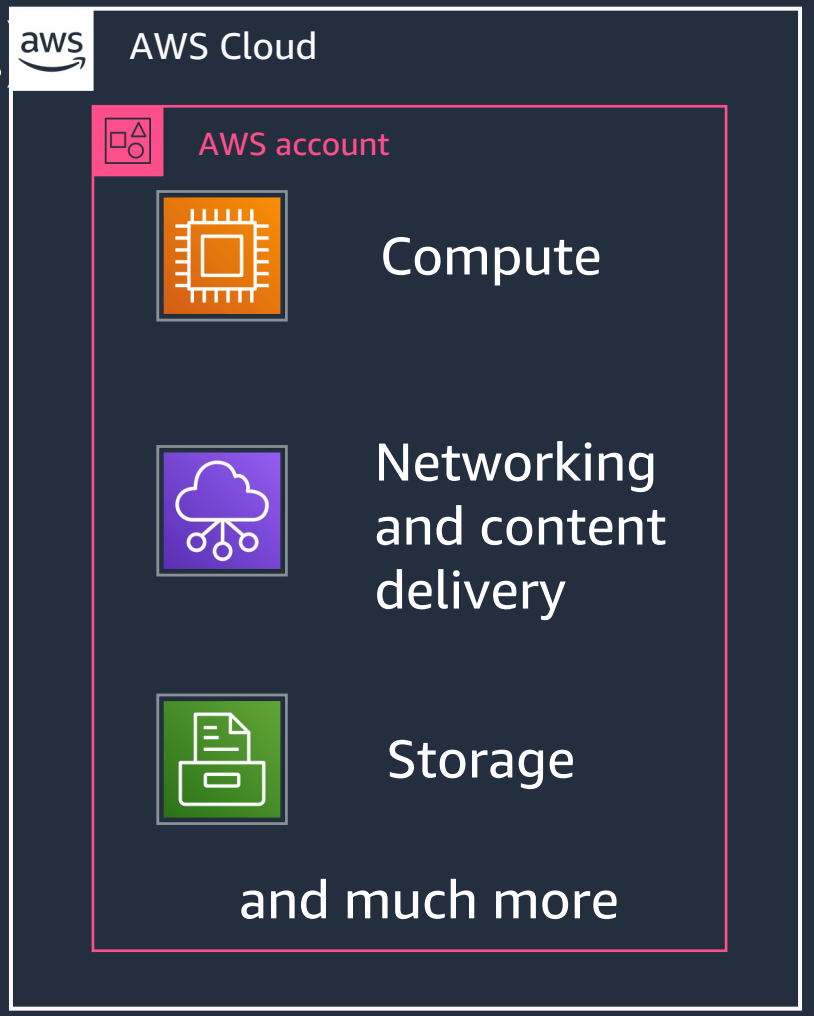
Scale across accounts

Account limits
Quotas

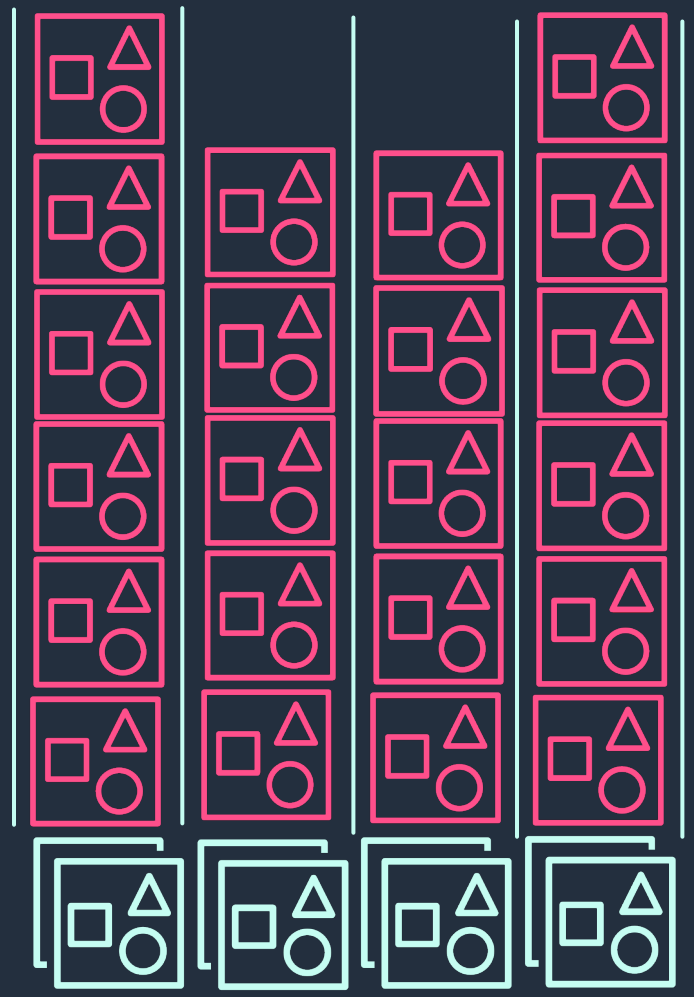
Security
Natural boundaries, isolation

Compliance/ business processes
Billing, custom requirements





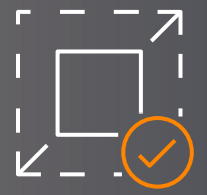
Framework for accounts: Organization Unit(OU)



To build out structure, you need supports, trusses, or frames in place to define the pattern for bricks

Organizational units (OUs) are the frames

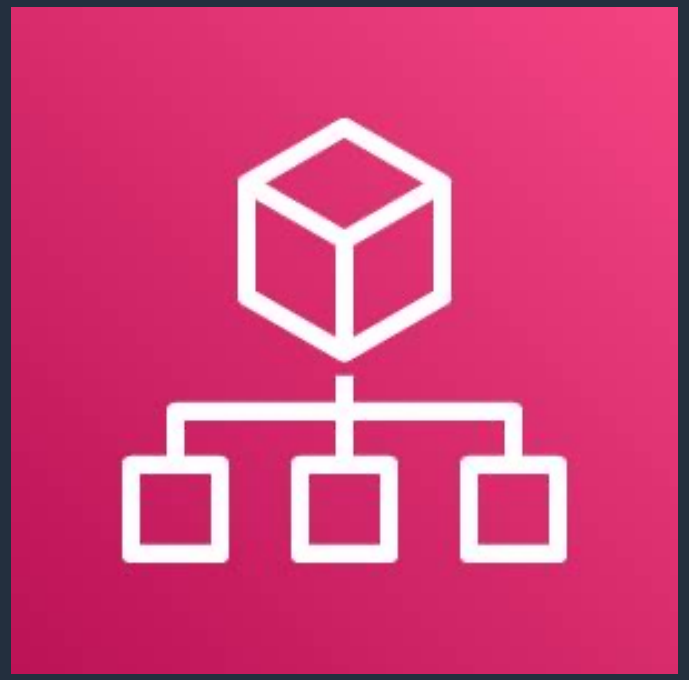
Design your OU structure around security and operational needs



Start small and broad, expand as needed

Guardrails/ policies are assigned to OUs

AWS Organizations



Provides you tools to centrally govern and manage your cloud environment

-Quickly scale by creating accounts and allocate resources

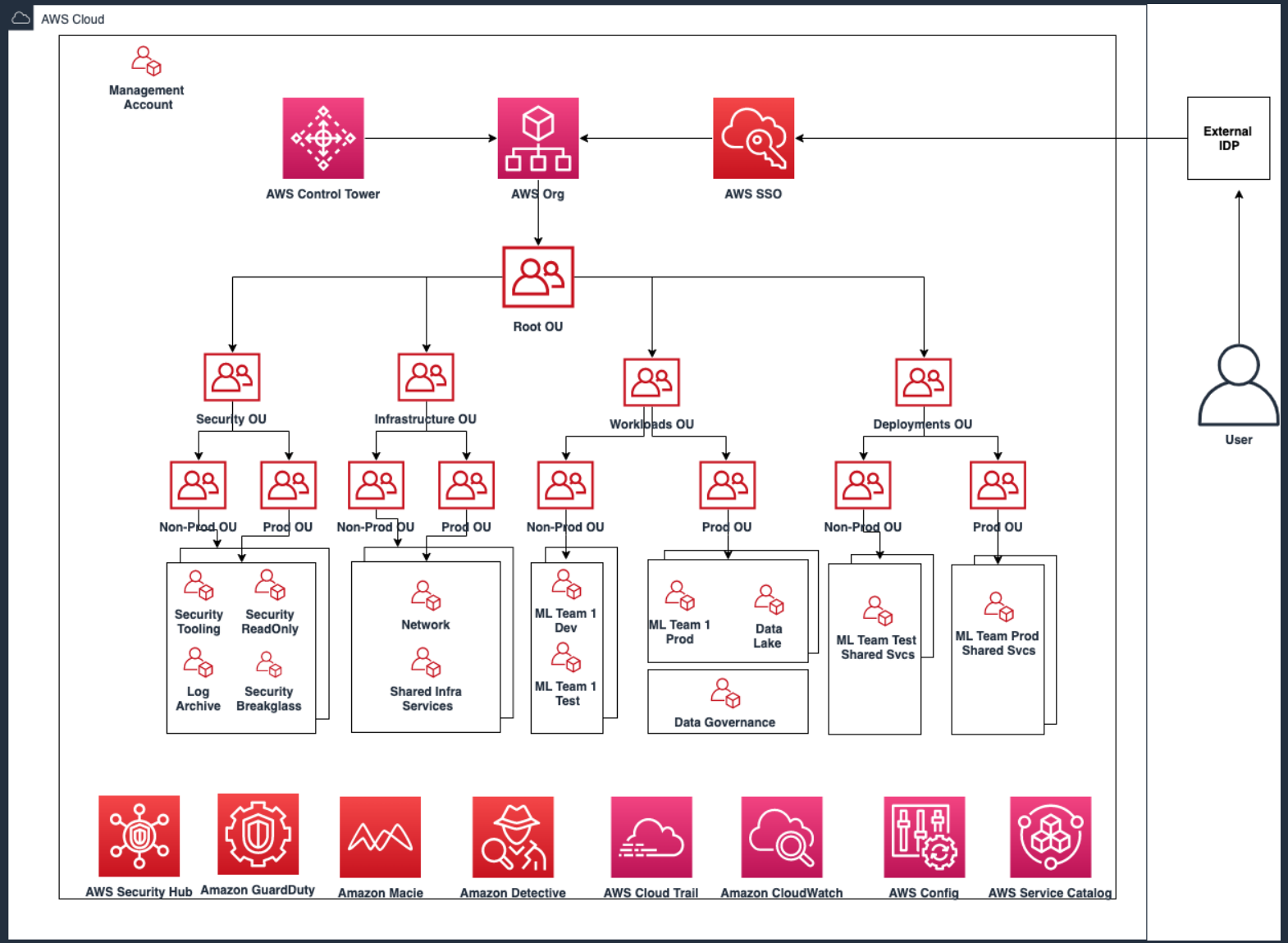
-Customize your environment by applying governance policies

-Secure and audit your environment

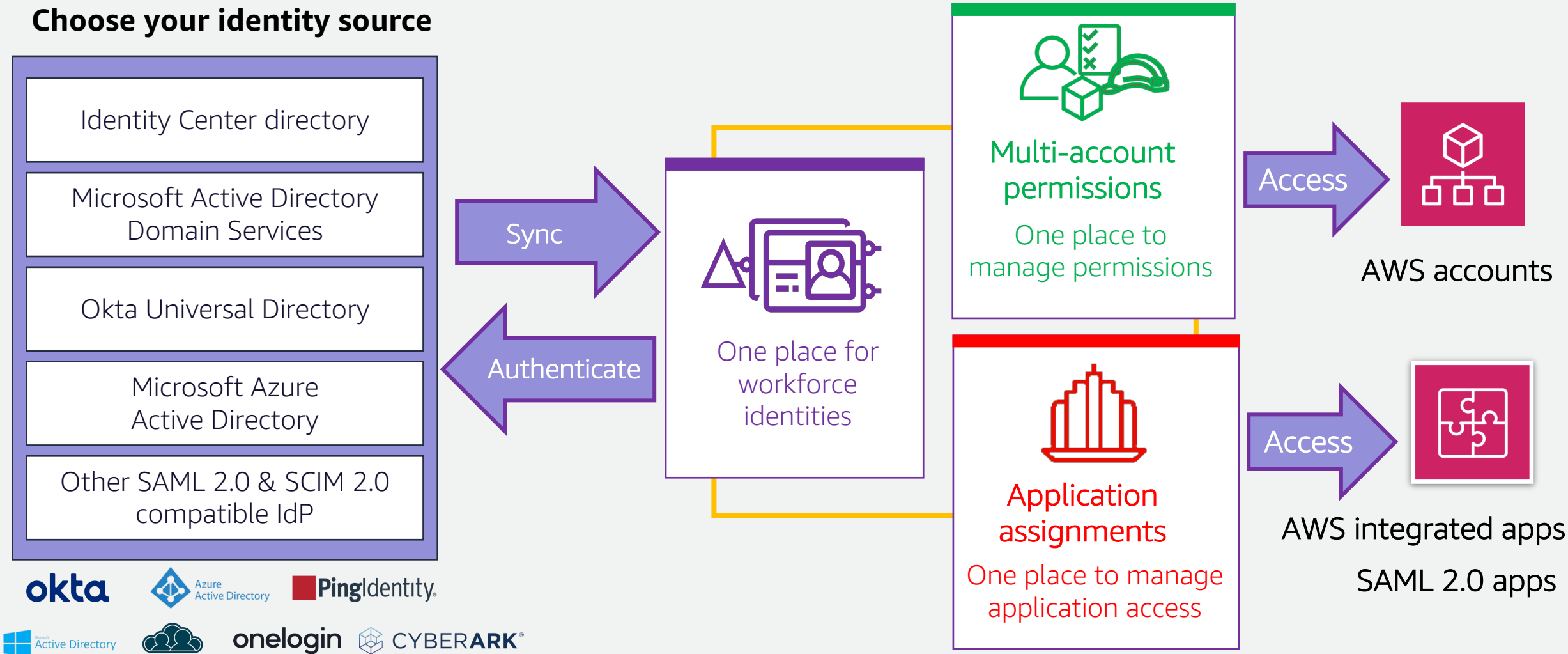
-Share resources and control access

-Manage costs and identify cost-saving measures

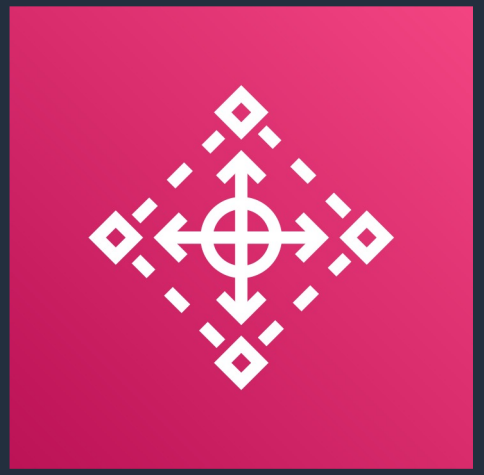
AWS Organization Structure



IAM Identity Center (formerly AWS SSO)



Overview of AWS Control Tower



AWS Control Tower

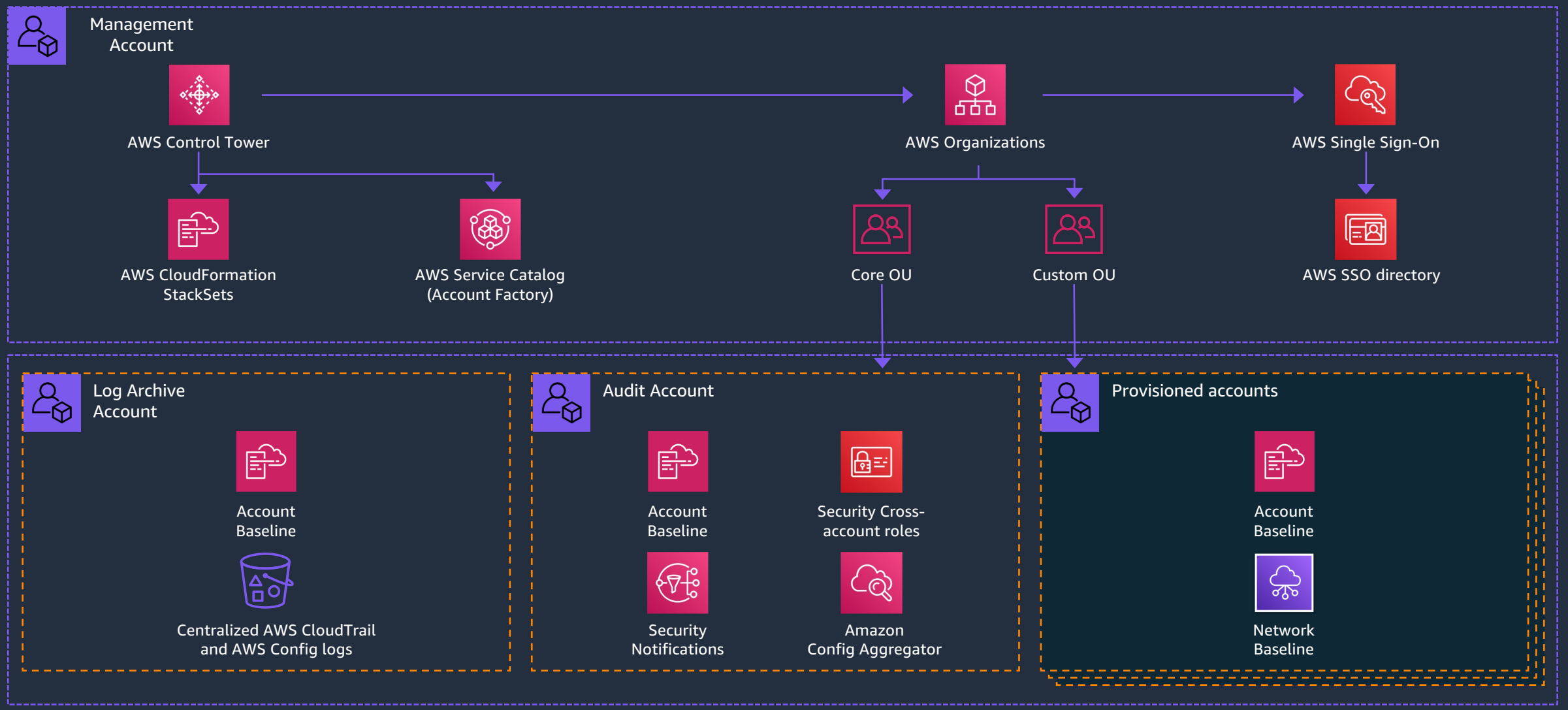
Easily set up and manage a secure multi-account environment

Service overview

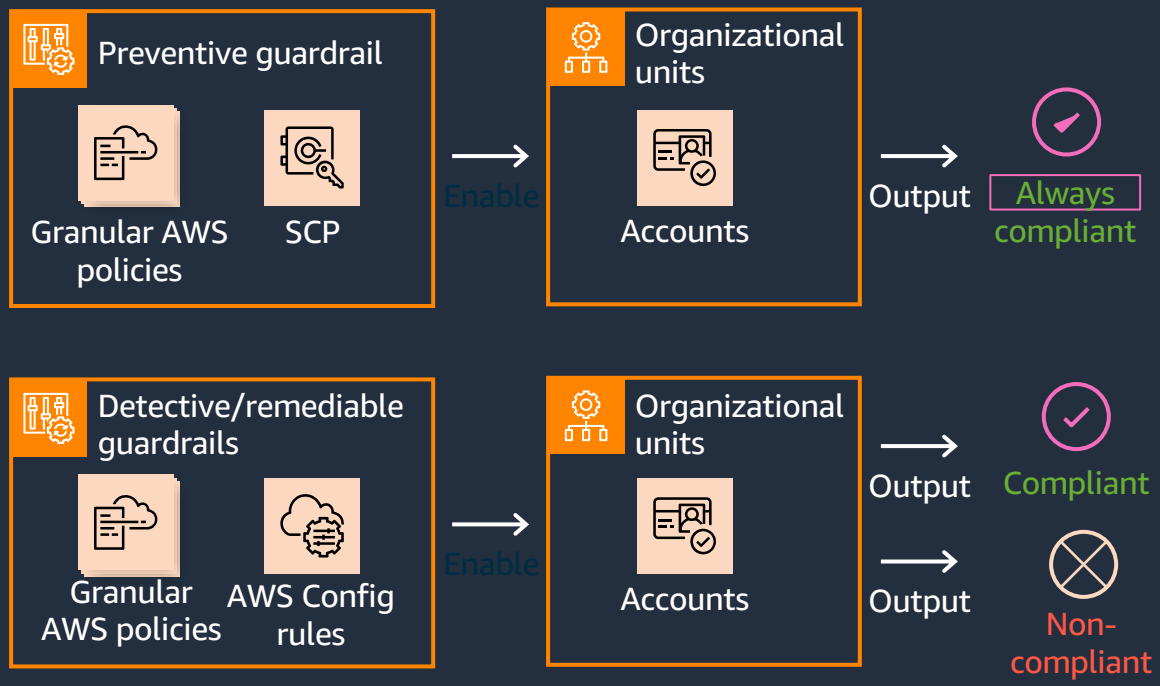
- Build an AWS management foundation based on best practices
  - ➢ Deploy Landing Zone using AWS Organizations, AWS CloudTrail, AWS IAM, etc.
- Install guardrails
  - ➢ Pre-packaged “guardrails” of security, operations, and compliance requests across the enterprise or only to specific accounts
- Free of charge
  (but incurs the cost of each AWS service required to configure the Landing Zone)

Reference)
https://aws.amazon.com/controltower/faqs/
AWS Control Tower FAQs

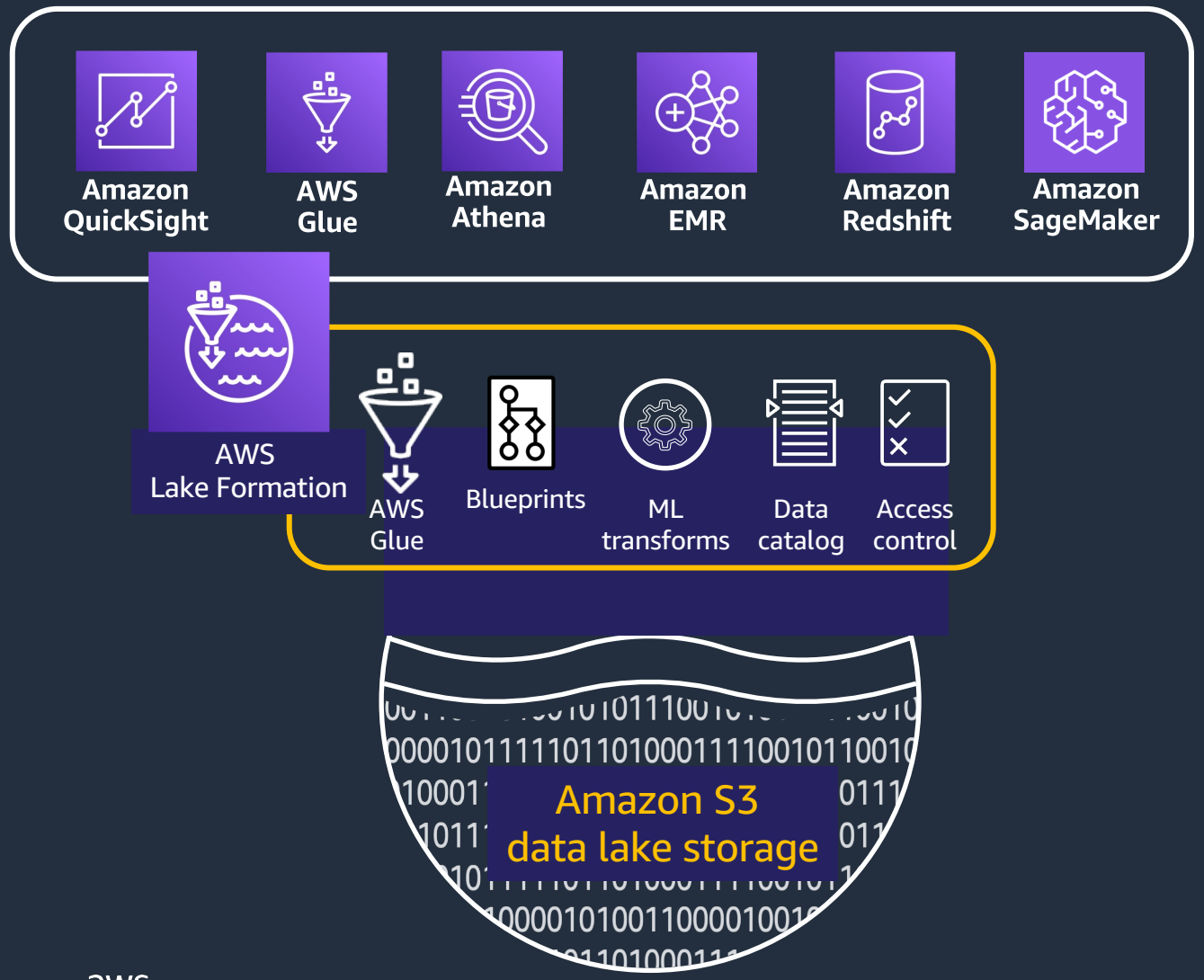
Landing Zone provisioned by AWS Control Tower



Establish guardrails



AWS Lake Formation: The foundation for data lakes



Single place to manage access

Open file formats

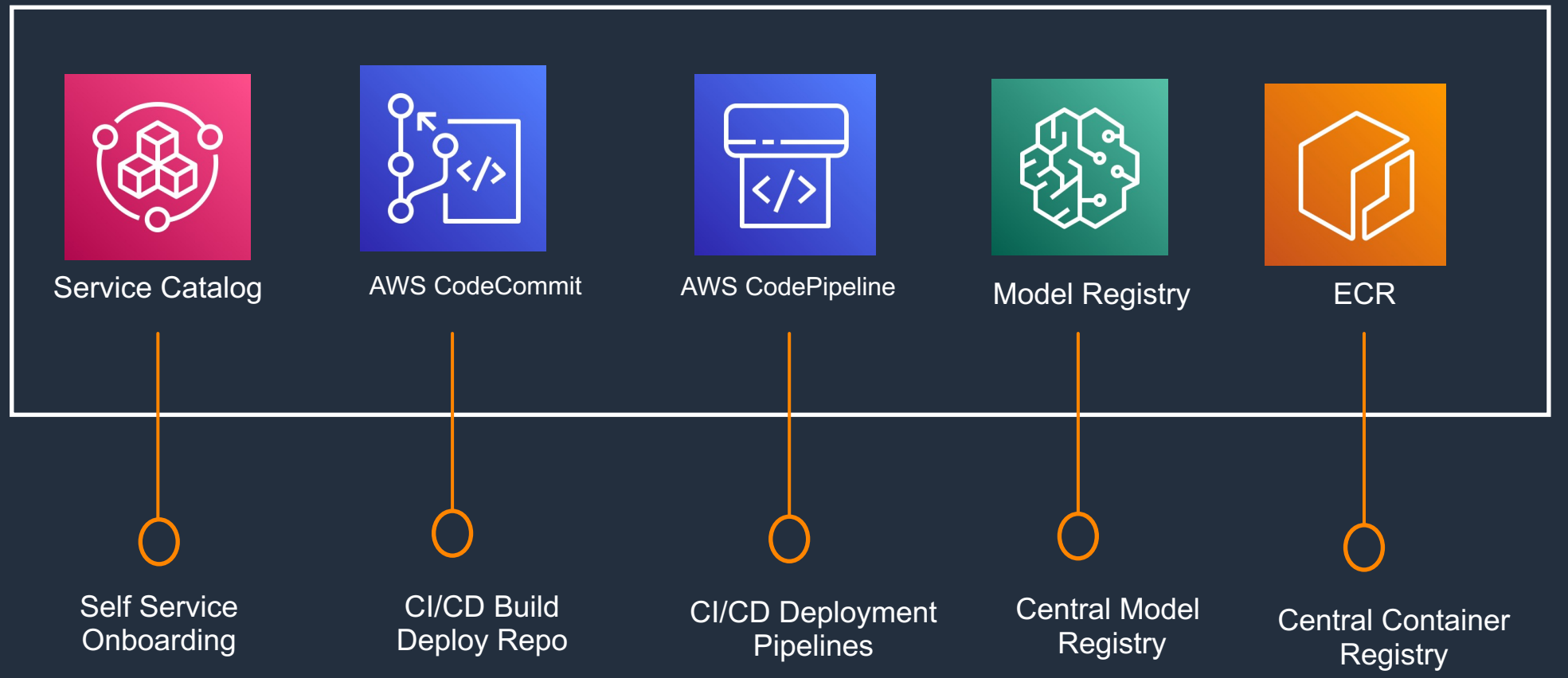
Efficient sharing

Ecosystem of integrated tools

Cost effective

2. Scale Governed ML Platform Operations

ML Platform Shared Services



Onboarding: ML Platform Key Personas and Roles

Advance Analytics Team
Data Lake

Data Engineer

Prepare & Ingest data building ETL pipelines for the ML use cases

Data Owners

Manage data sharing and provide access to other teams

Data Science Team
Experimentation & MLOps

Data Scientist

Create and Maintain the best ML models to solve business problems

ML Engineer

Collaborate with DS to productionize their code and models following development best practices

Platform Team
Secure Cloud/Data/ML Platform

Platform/MLOps Engineer

Standardize CI/CD, Container creation, model consumption, testing and deployment methodology based on business requirements

Security

Assess data, user, and service access creating policies and guardrails

Architects/ SysOps Engineer

Standardize account infrastructure e.g. VPC and endpoints, subnets, security groups, connectivity, user roles implementation

Business
Viz Dashboards, ML Adoption, & ROI

Business Stakeholder Product Owners

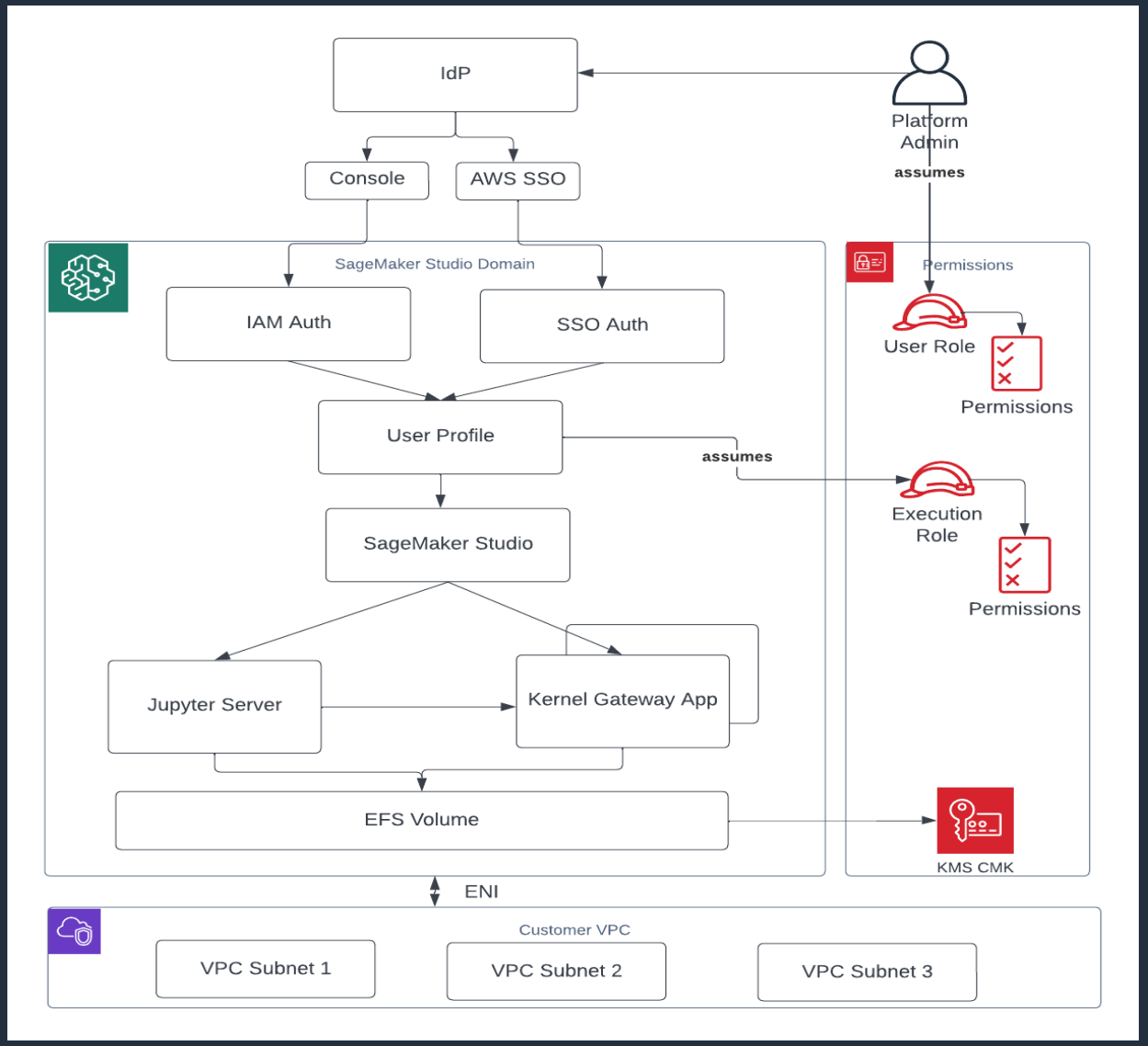
Define business problem, ML use case business KPIs, interact with other BUs, and make business decisions

Business Stakeholder ML Consumers

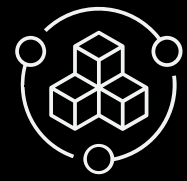
Consumers of ML results from other BUs, driving business decision making

aws

Onboarding: SageMaker Domain Management

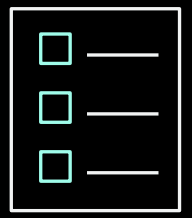


AWS Service Catalog



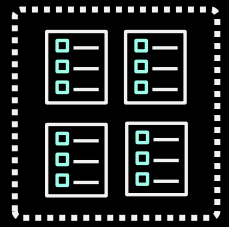
Product

JSON, YML or Terraform



IT service or resource

Portfolio



Where admins save collections of products

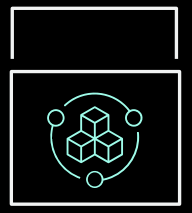
Constraint

Security, governance and deployment controls

Administrator tasks

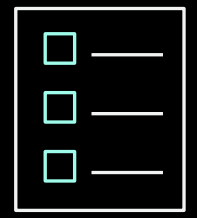
User tasks

Provisioned products



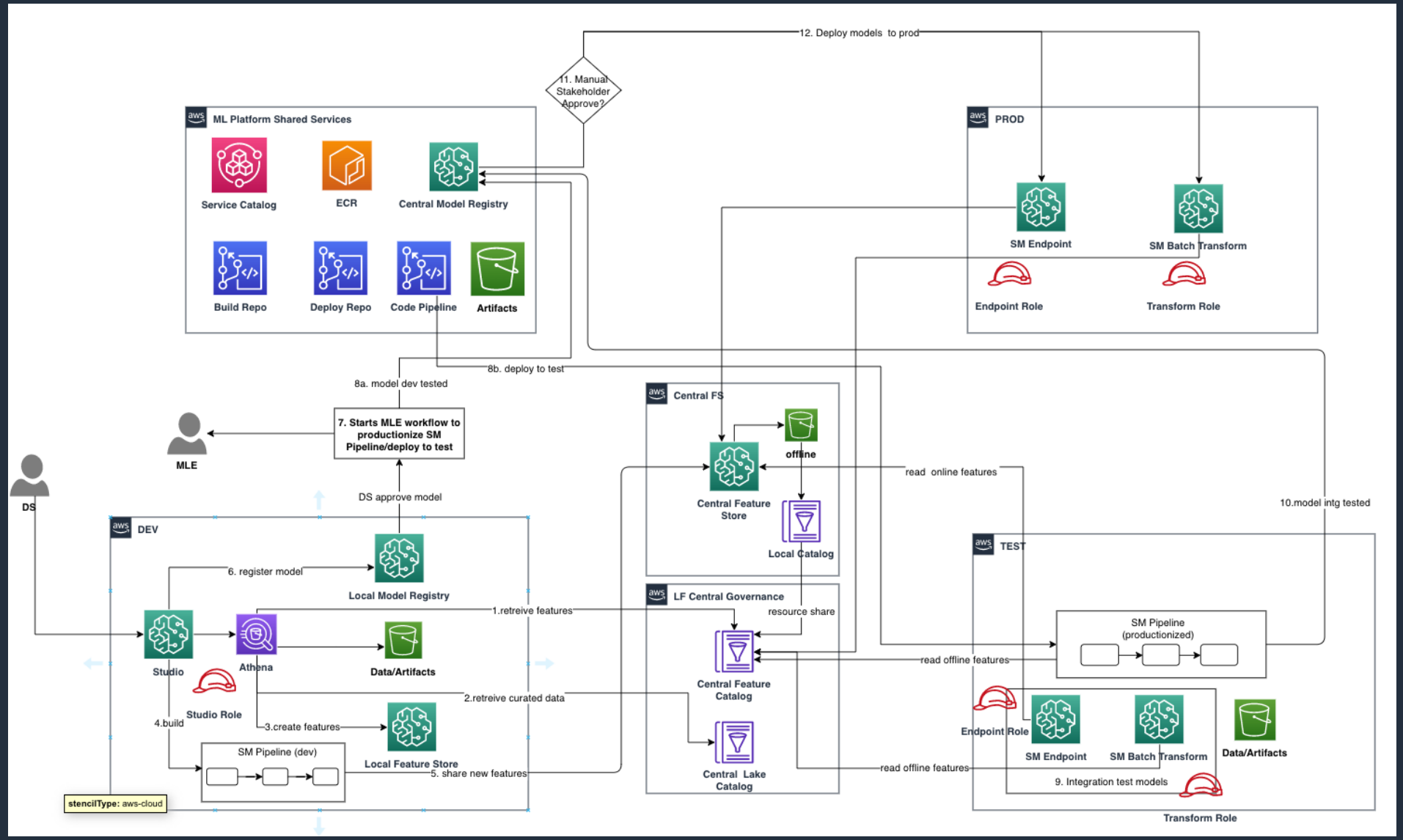
A SC deployed and configured product

Products list



Where end users see what products they have access to

Central Model Registry and Feature Store Workflow



3. Observable ML

Experiment Tracking with SageMaker Experiments

Runs 7

Search

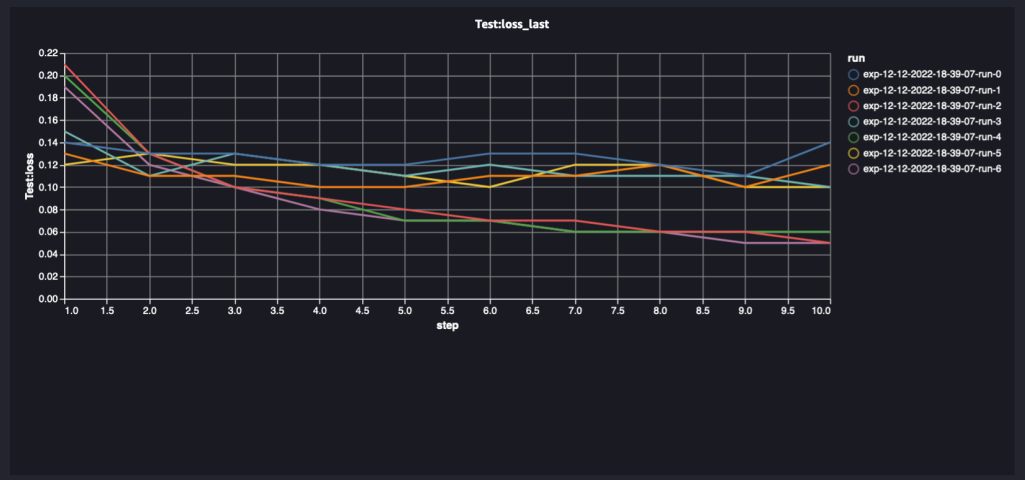
Name	Id	Run Group	Type	Test:loss	Train:accuracy	Test:accuracy	Train:loss
exp-12-12-2022-18-39-07-run-6	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.04963957405090332	98.46	98.47	0.051136534420897564
exp-12-12-2022-18-39-07-run-5	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.10452271881103516	97.17	97.05	0.09755590740218759
exp-12-12-2022-18-39-07-run-4	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.058010439300537106	98.12666666666667	98.21	0.059526144051055115
exp-12-12-2022-18-39-07-run-3	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.09766765747070312	96.95666666666666	97.26	0.10599174875694638
exp-12-12-2022-18-39-07-run-2	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.053717627716064455	98.12666666666667	98.32	0.06060071001363297
exp-12-12-2022-18-39-07-run-1	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.11910663757324219	96.83333333333333	96.78	0.1146532233208418
exp-12-12-2022-18-39-07-run-0	Default-Run-Group-exp-12-1...	Default-Run-Group-exp-12-1...	- -	0.14192571716308594	95.465	95.64	0.1484463992357254

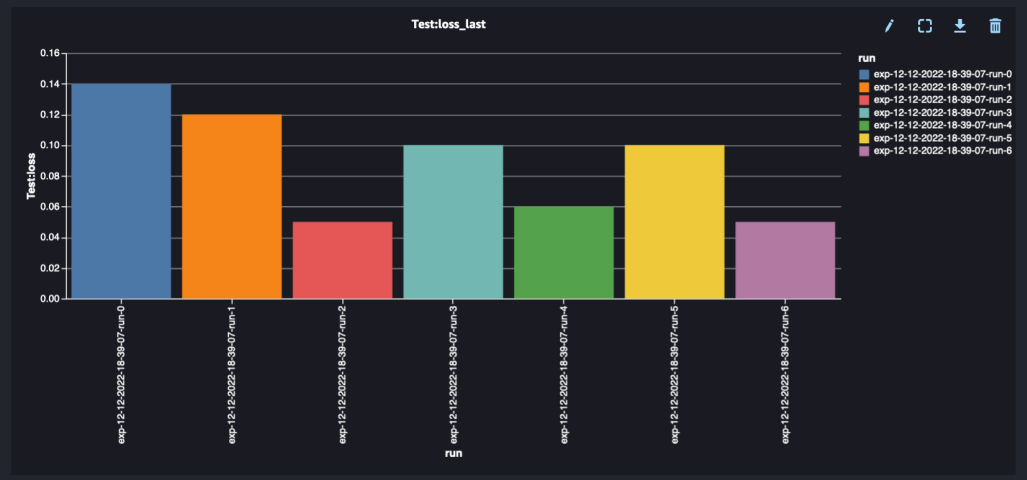
Refresh

Runs per page 10 Go to page 1 Page 1 of 1

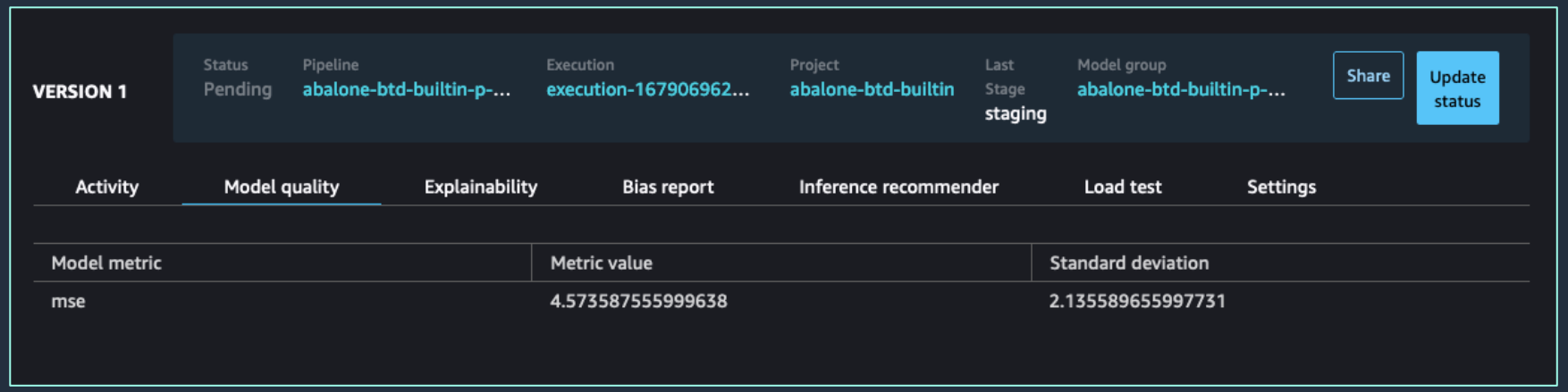
CHARTS 2

Compact View Add Chart

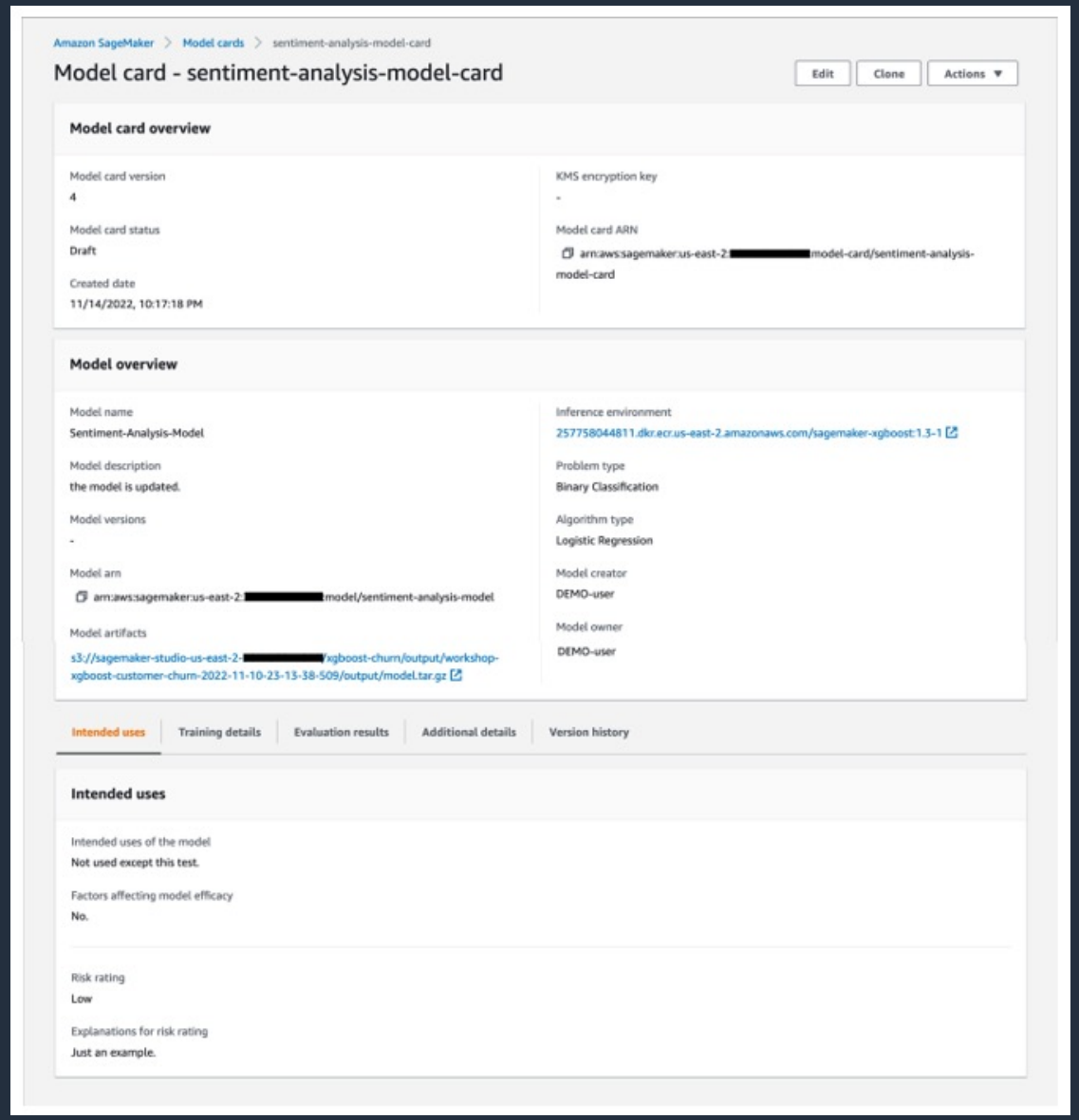




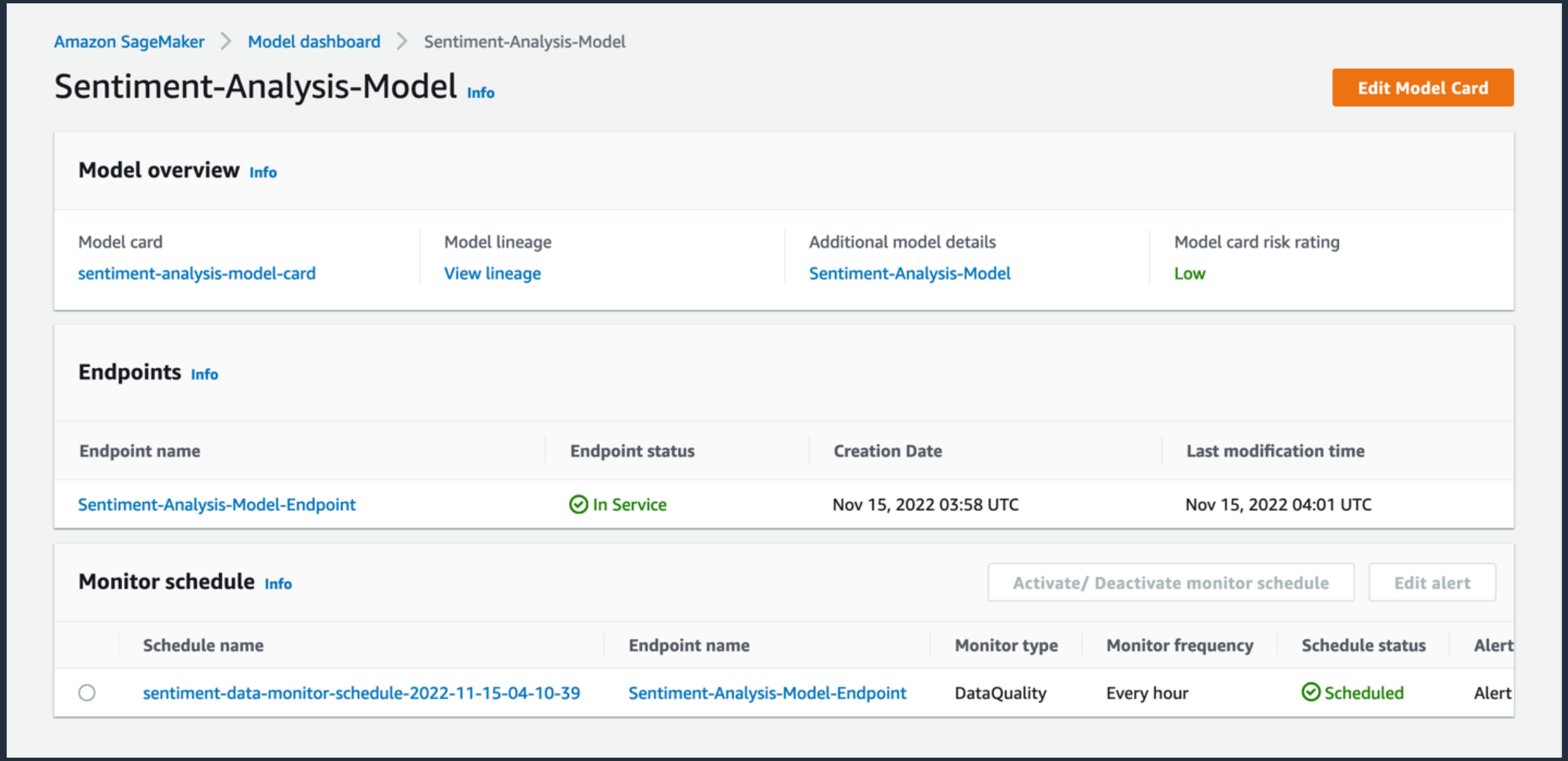
Model inventory using SageMaker Model Registry



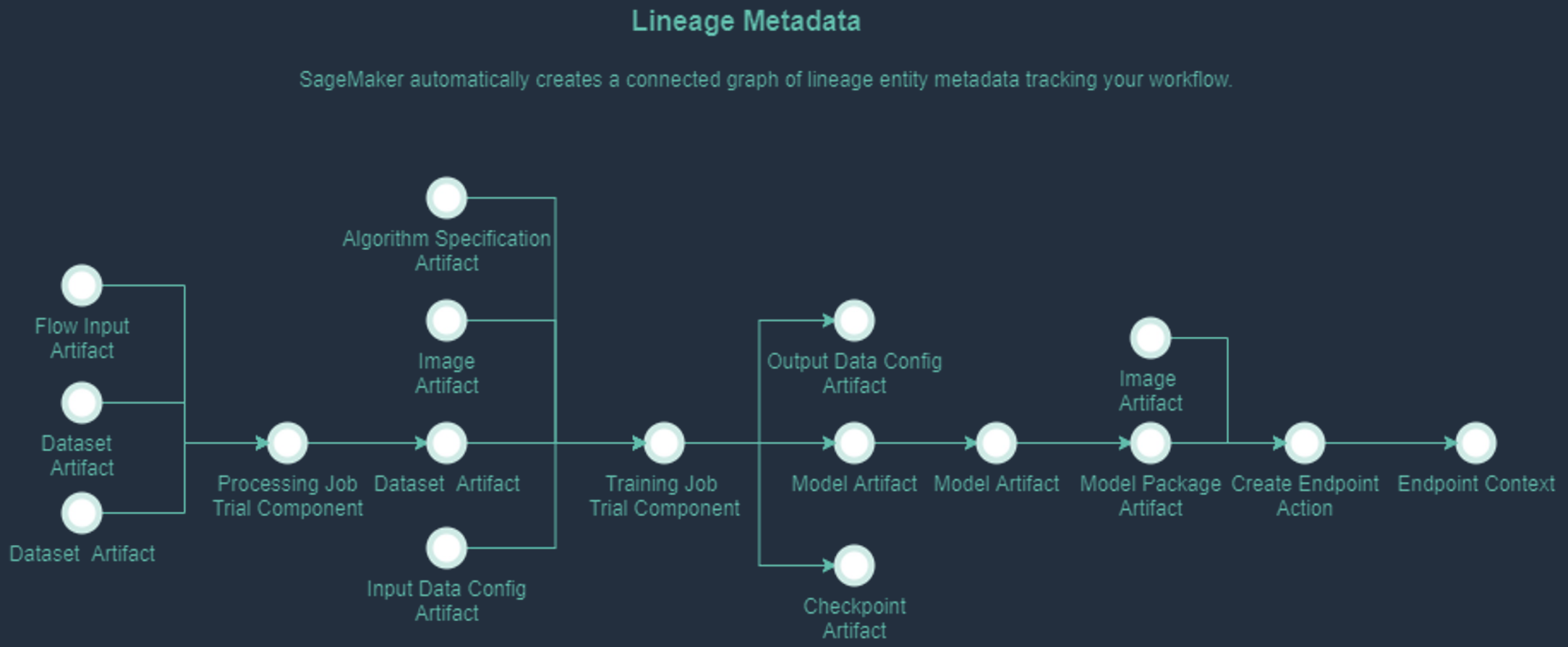
SageMaker Model Cards



SageMaker Model Dashboard

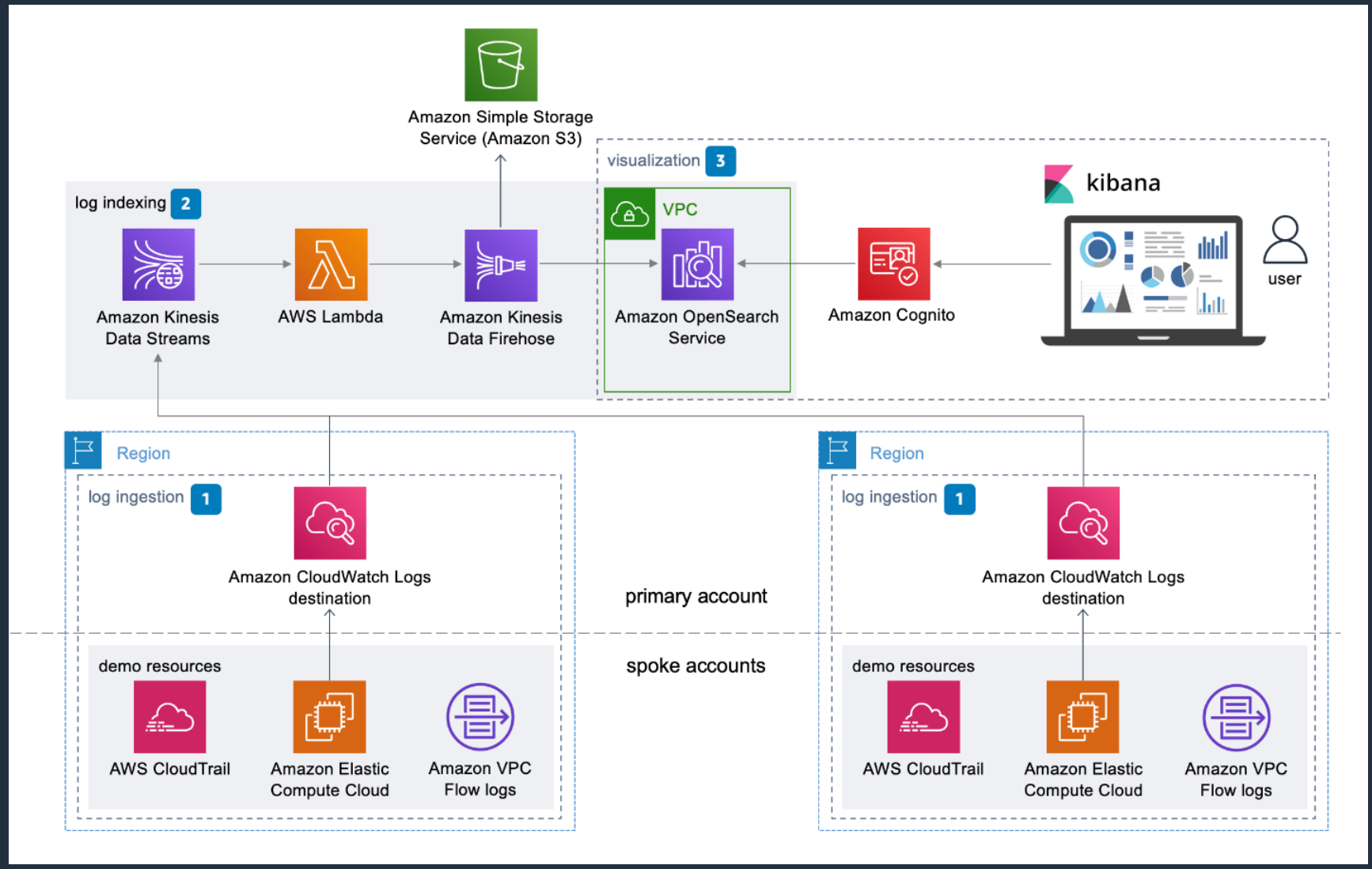


SageMaker ML Lineage Tracking



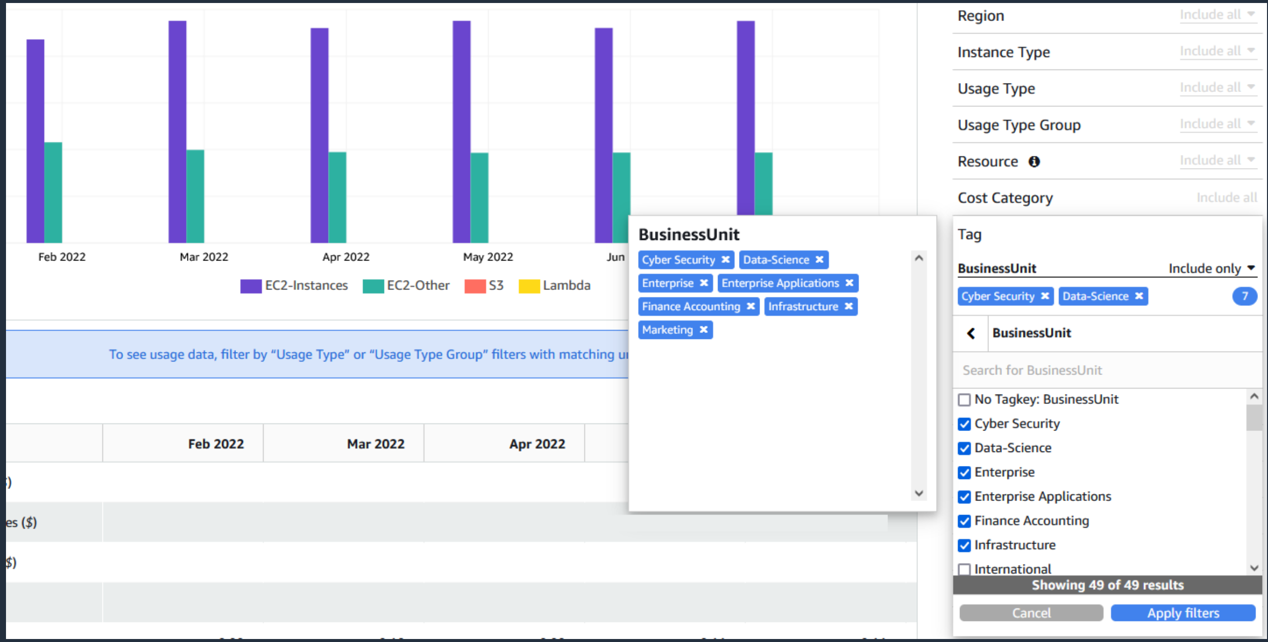
Use the SageMaker Python SDK to query lineage and manually track entities

Centralized Logging and Monitoring



Cost Control and Reporting in SageMaker

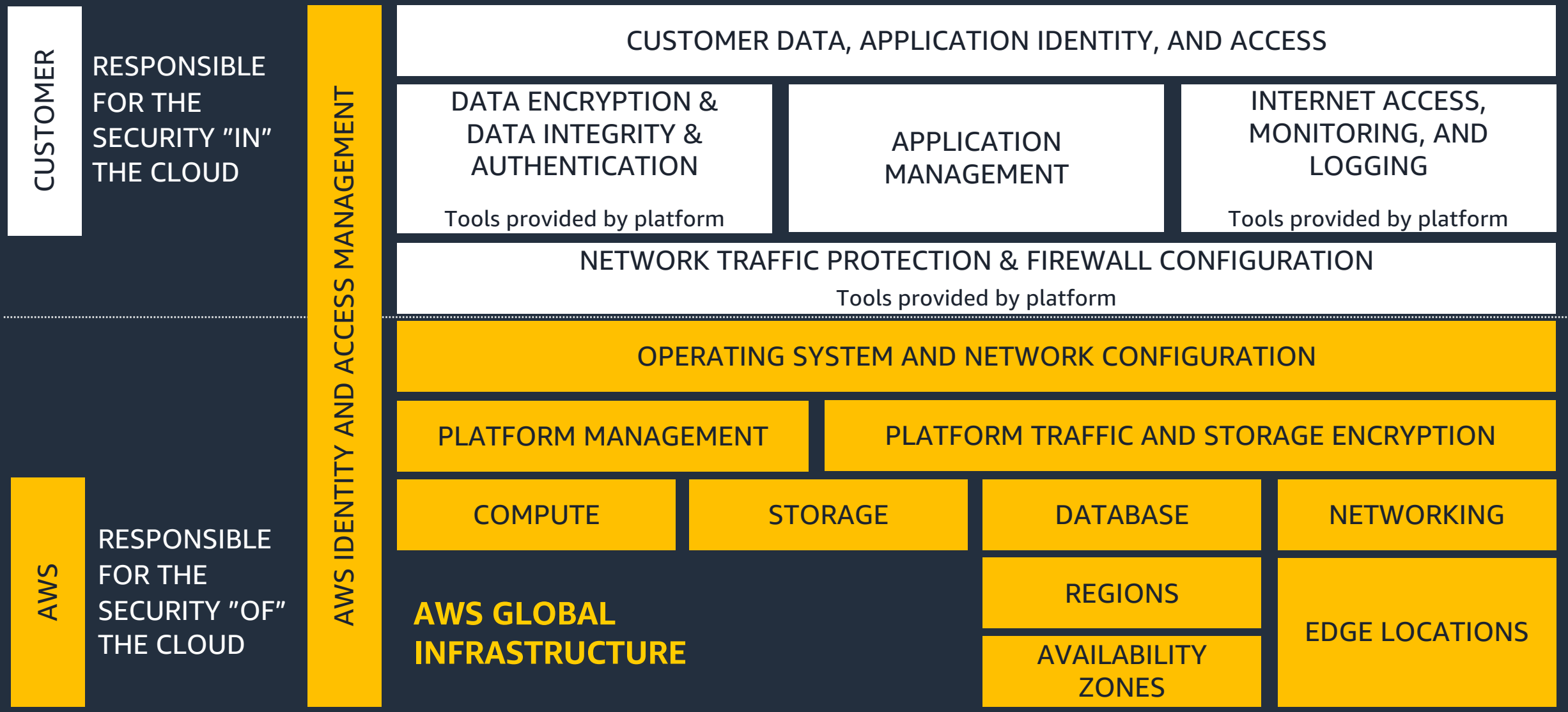
- Cost allocation for ML environments and workloads using resource tagging in SageMaker
- Enforce tagging using IAM policies
    - Use *Tag policies* to standardize tagging at the organization level
- View ML spend through **AWS Cost Explorer** and **AWS Cost and Usage Report (CURs)**
- Use **AWS Budgets** to provide early warning when ML costs exceed (or forecasted to exceed) a threshold



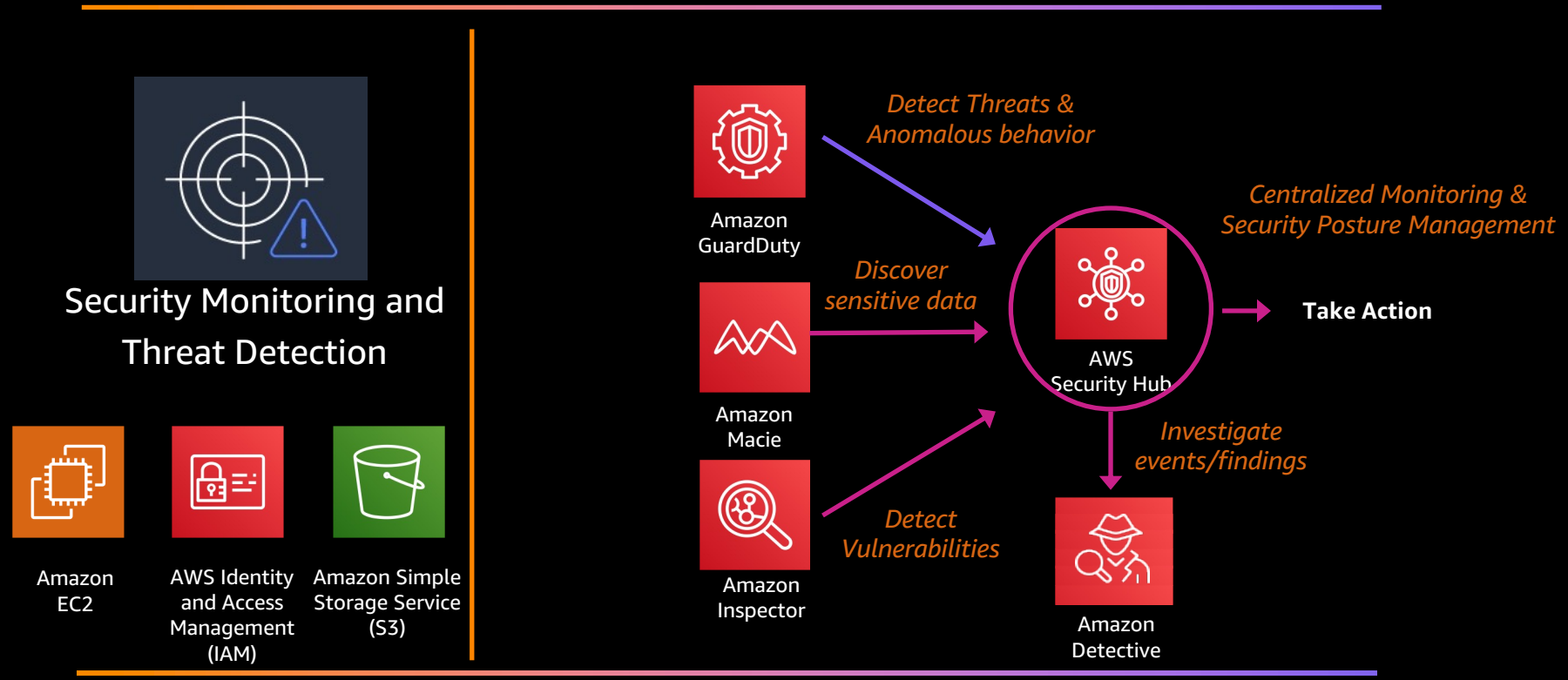
AG	AH	AI	AJ	AK	AL	AM	AN	AO
product/regi	product/servicename	product/usagetype	product/vcpu	pricing/publi	pricing/term	pricing/unit	resourceTags/user:cost-center	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	
us-west-2	Amazon SageMaker	USW2-Host:ml.m5.xlarge	4	0.23	OnDemand	Hrs	TF2WorkflowEndpoints	

4. Secure ML

SHARED RESPONSIBILITY MODEL: SAGEMAKER STUDIO



AWS Security Hub: Centralize Security Monitoring



Amazon SageMaker security features help you go from idea to production faster

Infrastructure and network isolation

Control data traffic across SageMaker components over a private network, and ensure appropriate ingress/egress with single-tenancy

Authentication and authorization

Define, enforce, and audit who can be authenticated and authorized to use Amazon SageMaker resources

Data protection

Ensure automatic data encryption at rest and in transit with flexibility to bring your own keys

Monitoring and auditability

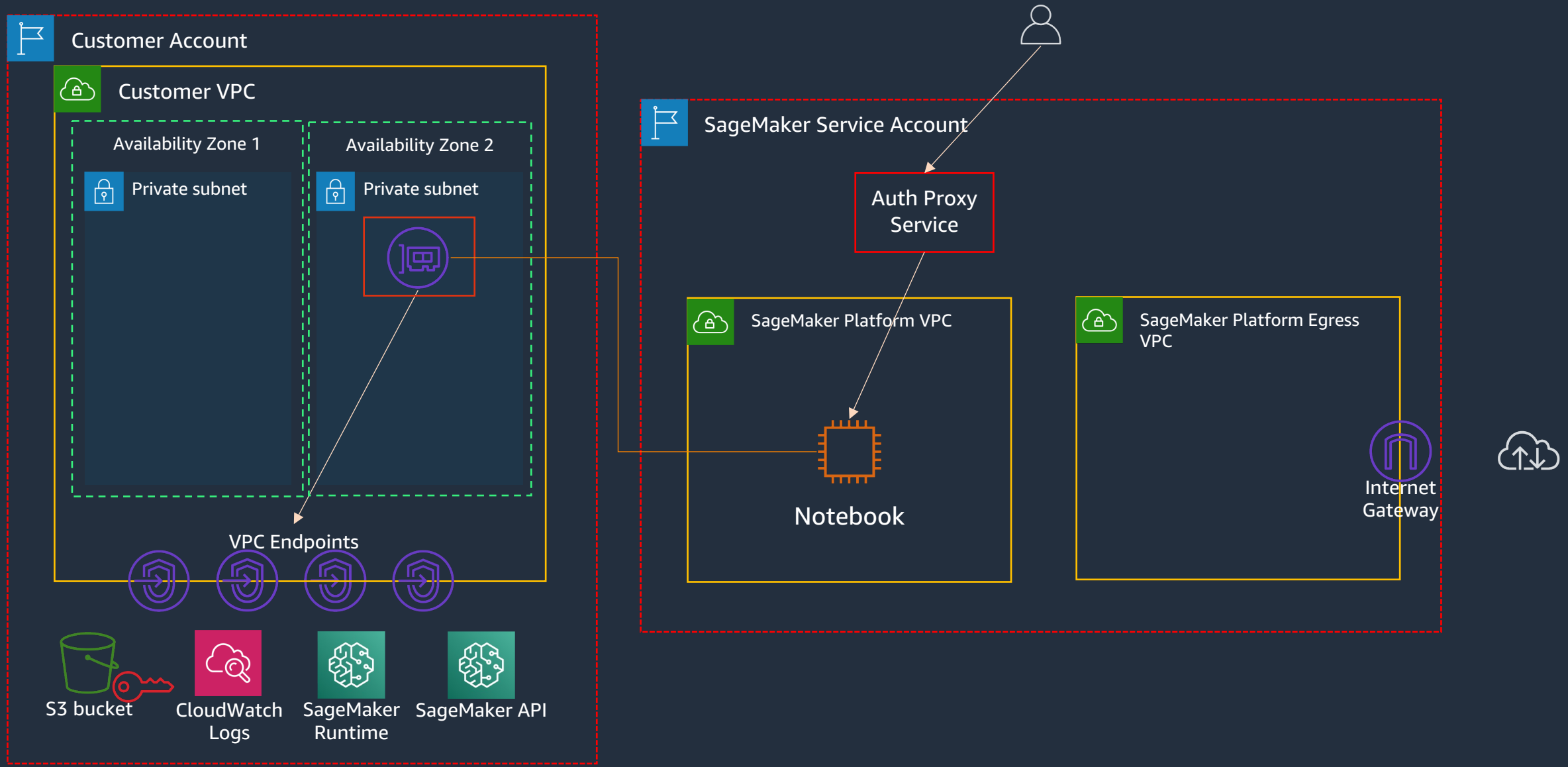
Track, trace, and audit all API calls, events, data access, or interactions down to the user and IP level to ensure quick remediation

Compliance certifications

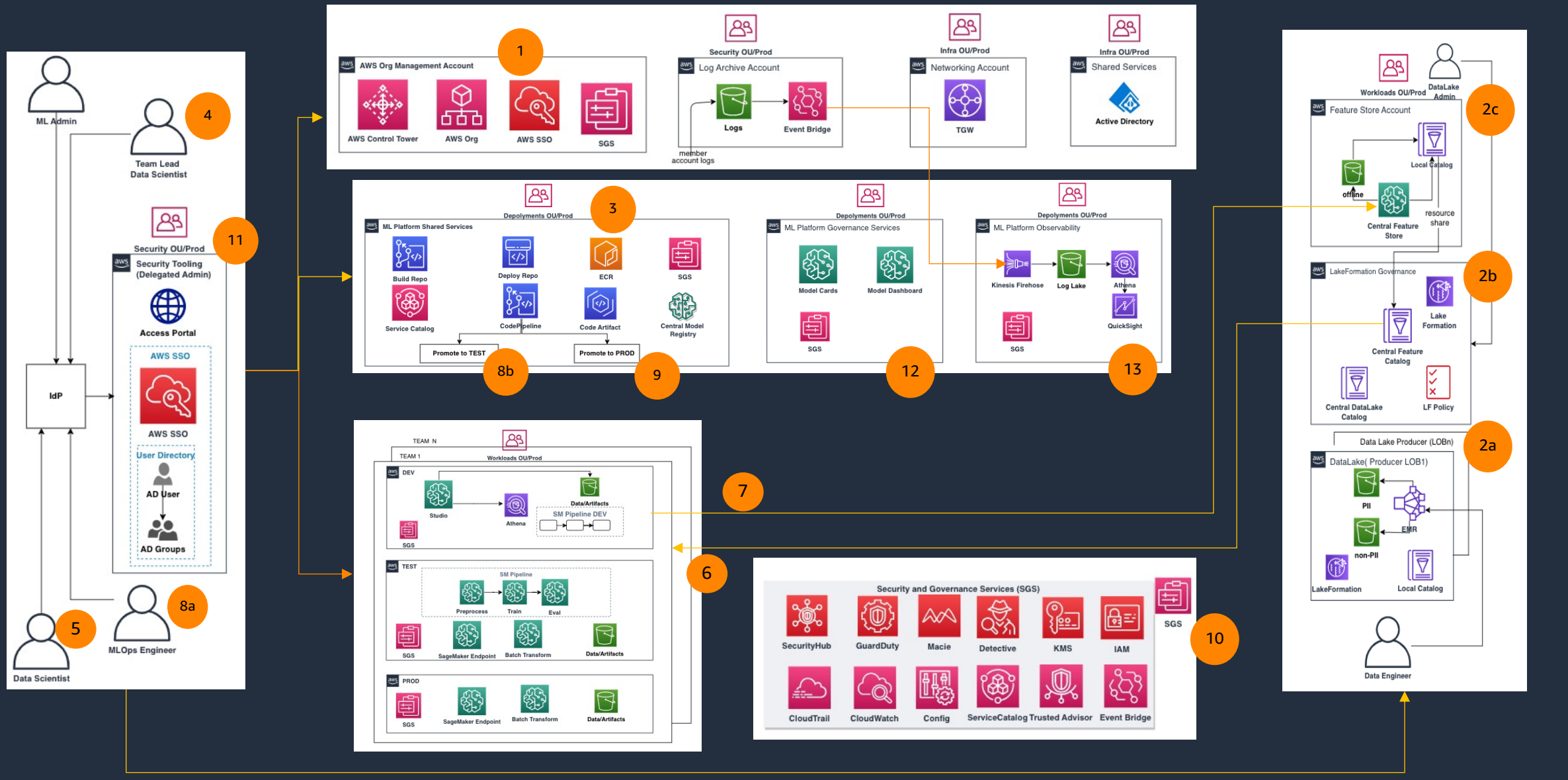
Inherit the most comprehensive compliance controls, and easily abide by your industry's legislation

aws

SageMaker Studio - VPC connectivity (recommended)



Governed ML Platform Reference Architecture



Key Takeaways

01 **Build Strong ML Platform Foundations**

Establish right operating model with multi-account and data strategies

02 **Scale ML Platform Operations**

Centralize Shared Services and Scale operations with platform level services e.g. on-boarding, CI/CD

03 **Create an Observable ML Platform**

Centralize logging, monitoring, experiment tracking and leverage ML governance services

04 **Secure your ML Platform**

Centralize Identity, Security, Network services, and leverage preventative & detective controls

Resources

- *SageMaker Studio Best Practices WhitePaper*
- *Building Secure ML Platform Whitepaper*
- *Onboarding SageMaker Studio with AWS SSO and Okta*
- *ML Platform Setup Blogs*
- *Macie Sensitive Data Detection*
- *SageMaker workshops for Studio*
    - *SagMaker End-to-End Workshop*
    - *SagMaker Immersion Day*
    - *Secure Data science with SageMaker Studio*