AI Document Assistant

Solution Overview

Motivation

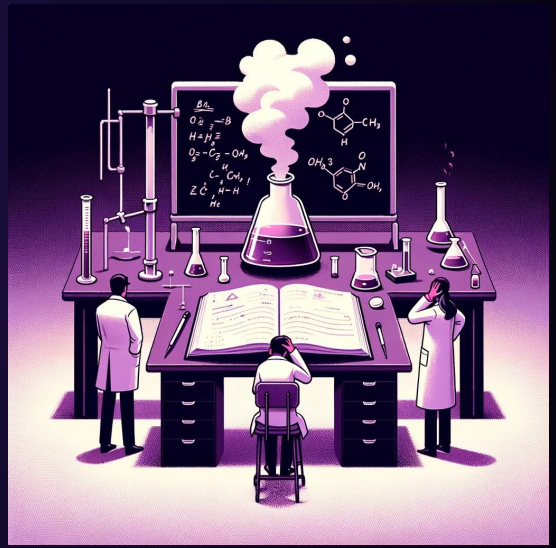
- **Objective**: Develop a reliable and efficient AI document assistant, ensuring security and rapid improvements.

Requirements

- **Systematic Model Development and Experiment Tracking**: Enable detailed tracking and versioning for auditability and reproducibility.
- **AWS Integration**: Ensure compatibility with AWS services like SageMaker and Glue.
- **Security and Compliance**: Maintain strict security, support self-hosting, and comply with standards.
- **Usability and Performance**: Provide a user-friendly, Python-compatible platform with clear results and cost-effective performance.

ML Experiment Tracking

ML teams struggle with **debugging** experiments, **sharing** results, and messy **model handover**.



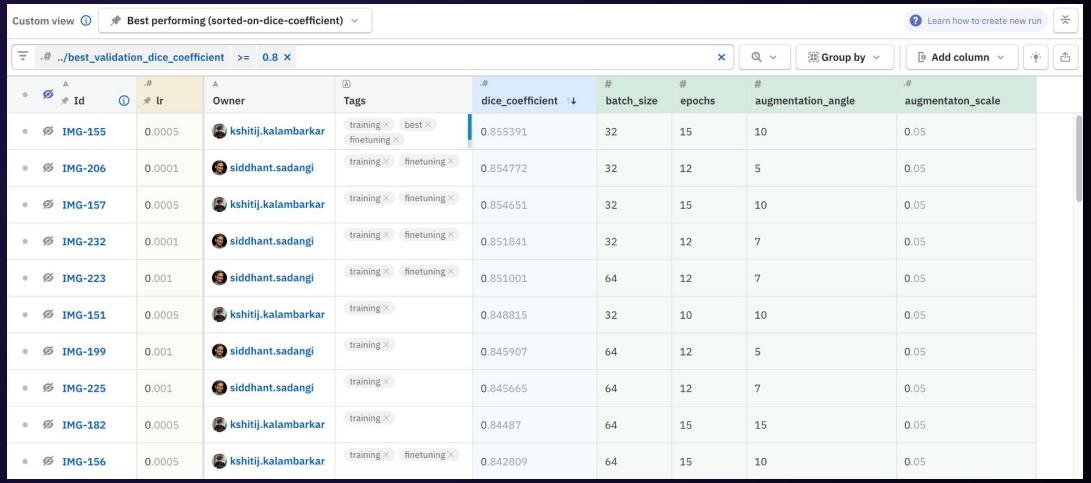
> *Large amount of ‘trial-and-error’ runs eventually becomes chaotic.*
>
> ***
>
> *…manually keeping track of all the different iterations leads to mistakes and wasted time to recreate past experiments.*
>
> ***
>
> *Proper experiment tracking was often deprioritized until reproducibility became a real issue, making it impossible to find the best model configurations or compare experiments.*

What is Neptune?

Neptune is the **most scalable** experiment tracker

that offers a **single place** to **track**, **compare**, and **collaborate on experiments**

- **50,000+** ML practitioners
- **40+** integrations
- **1M** DP/s ingestion rate



Id	lr	Owner	Tags	dice_coefficient	batch_size	epochs	augmentation_angle	augmentaton_scale
IMG-155	0.0005	kshitij.kalambarkar	training, best, finetuning	0.855391	32	15	10	0.05
IMG-206	0.0001	siddhant.sadangi	training, finetuning	0.854772	32	12	5	0.05
IMG-157	0.0005	kshitij.kalambarkar	training, finetuning	0.854651	32	15	10	0.05
IMG-232	0.0001	siddhant.sadangi	training, finetuning	0.851841	32	12	7	0.05
IMG-223	0.001	siddhant.sadangi	training, finetuning	0.851001	64	12	7	0.05
IMG-151	0.0005	kshitij.kalambarkar	training	0.848815	32	10	10	0.05
IMG-199	0.001	siddhant.sadangi	training	0.845907	64	12	5	0.05
IMG-225	0.001	siddhant.sadangi	training	0.845665	64	12	7	0.05
IMG-182	0.0005	kshitij.kalambarkar	training	0.84487	64	15	15	0.05
IMG-156	0.0005	kshitij.kalambarkar	training, finetuning	0.842809	64	15	10	0.05

Neptune Experiment Tracker

Integrations & Governance

- SaaS or self-hosted
- Extensive integrations for auto-logging & interoperability
- Top-tier security performance

Tracking

- Wide range of supported logged types
- Live monitoring at hyper-scale (async data ingestion, 1M DP/s rate, powered by Kafka)

Model Lifecycle

- Advanced visualization, comparison and search for debugging / finding best-performing model
- Model registry for reliable versioning and staging

Usability & Cost Efficiency

- High customization capabilities (metadata, dashboards, run views, and more), intuitive UI and developed collaboration features
- Transparent pricing without cost “traps”

Neptune Experiment Tracker

Neptune vs AWS SageMaker

Integrations & Governance

- SaaS or self-hosted
- Extensive integrations for auto-logging & interoperability
- Top-tier security performance


Tracking

- Wide range of supported logged types
- Live monitoring at hyper-scale (async data ingestion, 1M DP/s rate, powered by Kafka)


Model Lifecycle

- Advanced visualization, comparison and search for debugging / finding best-performing model
- Model registry for reliable versioning and staging


Usability & Cost Efficiency

- High customization capabilities (metadata, dashboards, run views, and more), intuitive UI and developed collaboration features
- Transparent pricing without cost “traps”


Decision Criteria

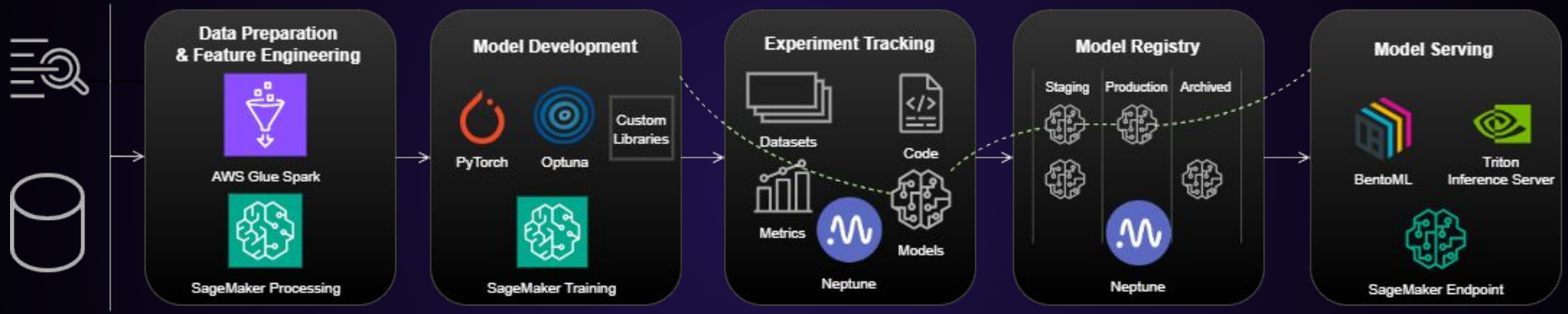
AWS Sagemaker vs Neptune comparison

For the decision criteria for choosing Neptune vs. SageMaker Studio, see the **Comparative Analysis** doc.

- Neptune (score: 19 out of 20)
  - Excels in scalable experiment tracking, security, and user-friendly features. While it requires some configuration for AWS integration, it maintains strong flexibility and performance.
- SageMaker Studio (score: 14 out of 20)
  - Excellent for AWS integration and usability within the AWS ecosystem. Has limited set of tracking features, lacks versioning & experiment comparison, non-transparent pricing mode, higher potential costs.

How does Neptune fit to your ML workflow?

```
pip install -U "neptune[aws]"
```

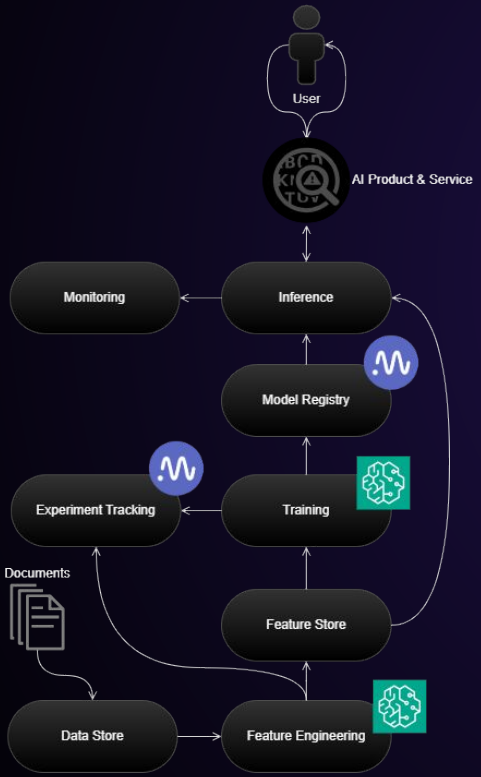


Kick-off your experiment tracking with **Neptune** in **AWS SageMaker** with this notebook.

Play with a [live example project] in the Neptune app.

Solution Architecture: High-Level View

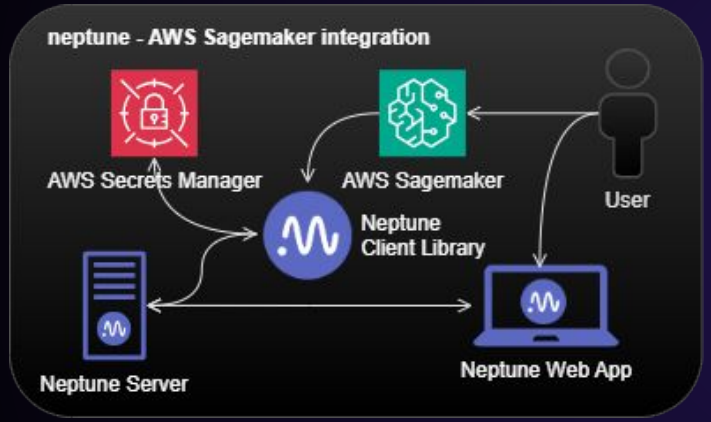
AI Document Assistant powered by Neptune and AWS SageMaker



How does Neptune fit to the solution architecture?

- Neptune complements AWS Sagemaker - a trusted choice for ML workflows.
- Neptune brings
  - **centralized place** for your **models** and **model life cycle** artifacts,
  - **live tracking** of **training** and **feature engineering** pipelines,
  - and **ease of integration** into existing workflow

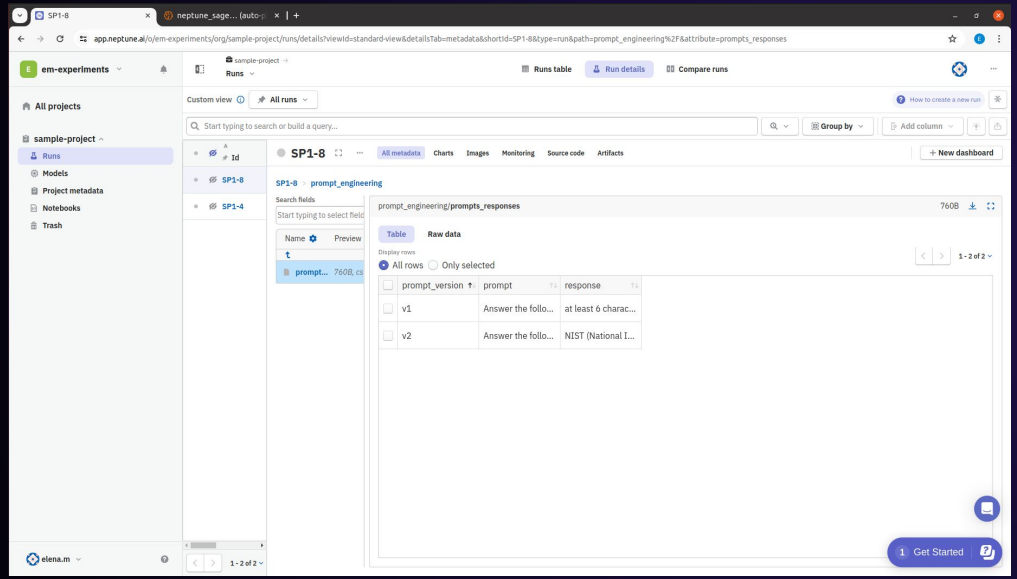
Solution Architecture: Integration



- **Neptune Client Library**: Connects AWS SageMaker with Neptune for easy communication.
- **Neptune Web App**: Offers an easy-to-use interface for real-time experiment monitoring and management.
- **Neptune Server**: Manages model artifacts and experiment data centrally.
- **AWS SageMaker**: Uses Neptune to track experiments and version models during pipeline execution.
- **AWS Secrets Manager**: Safeguards credentials for secure integration.

Demo

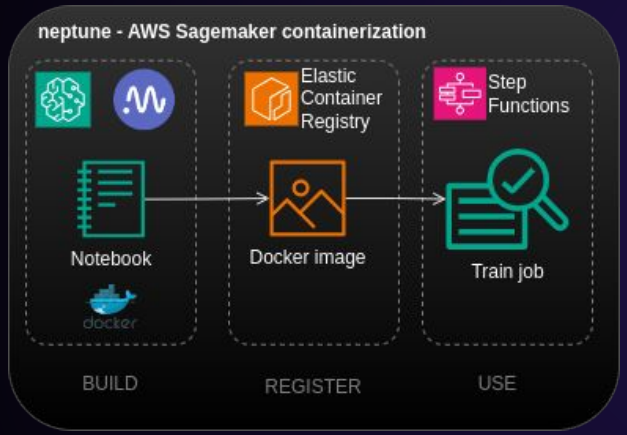
To kick-off your experiment tracking with Neptune in AWS SageMaker, see the notebook.



Using Neptune - What you can log

Solution Architecture: Containerization

[How to use Neptune in SageMaker training job]

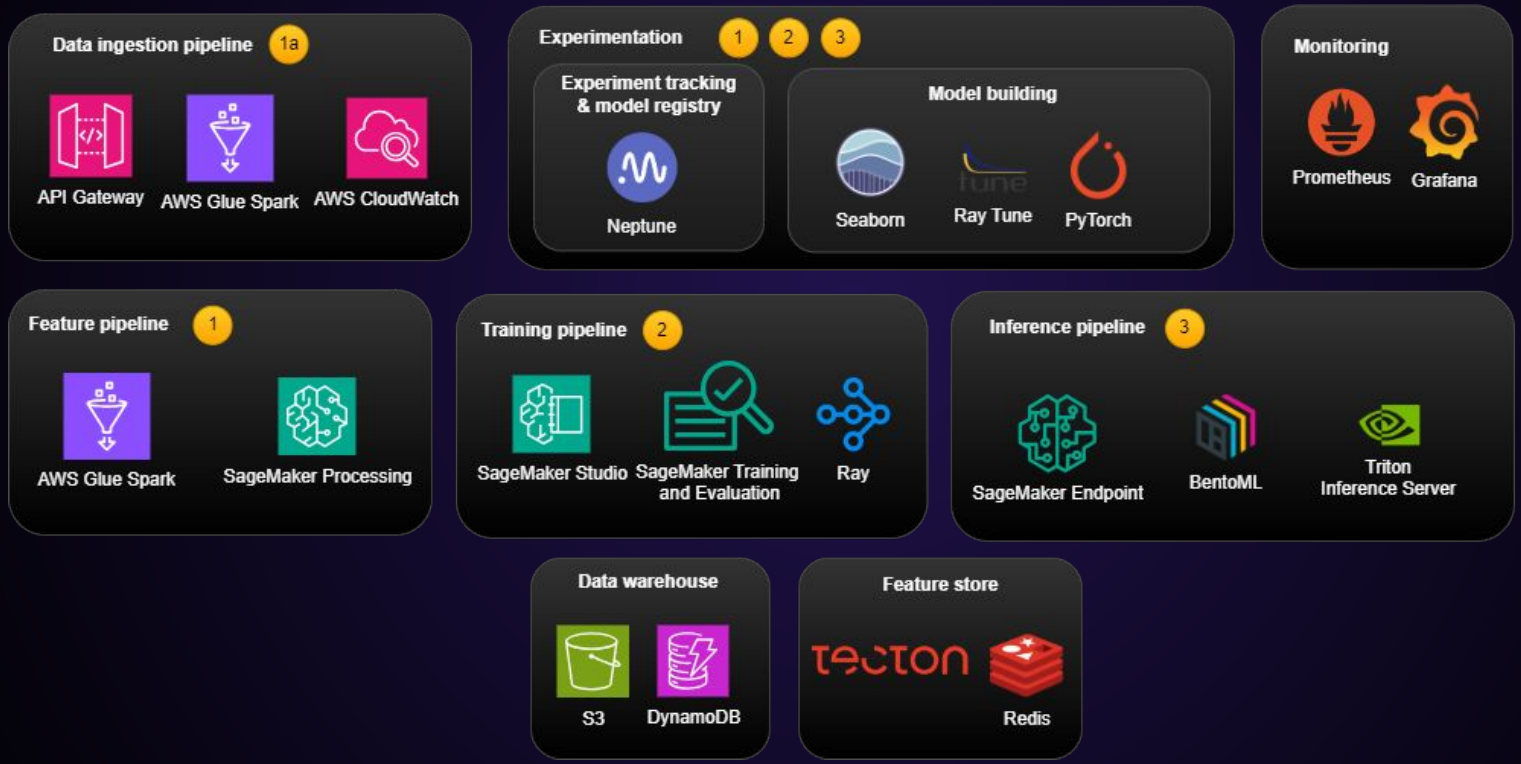


- Custom Docker images with Neptune pre-installed can be built from Jupyter Notebooks.
- Push Docker images to Amazon Elastic Container Registry (ECR) for scalable deployment.
- Run SageMaker training jobs as usual with Neptune logging integrated inside Docker containers.

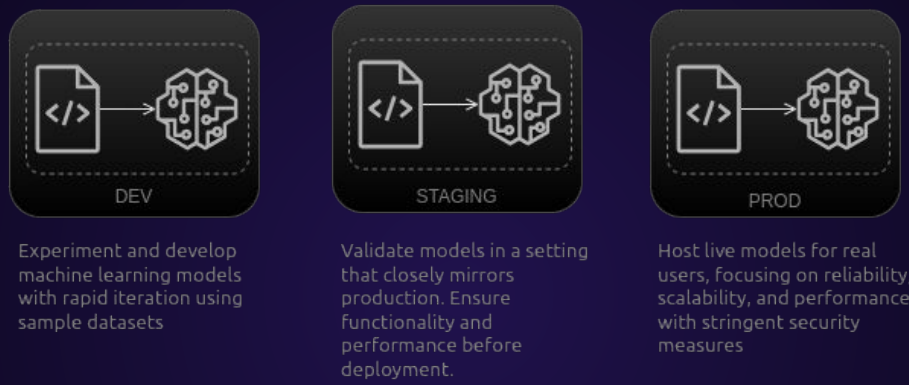
Supporting Materials

- Solution Architecture: Sample Tool Stack
- Multi-Environment Implementation
- Recommended LLMOps Workflow

Solution Architecture: Sample Tool Stack

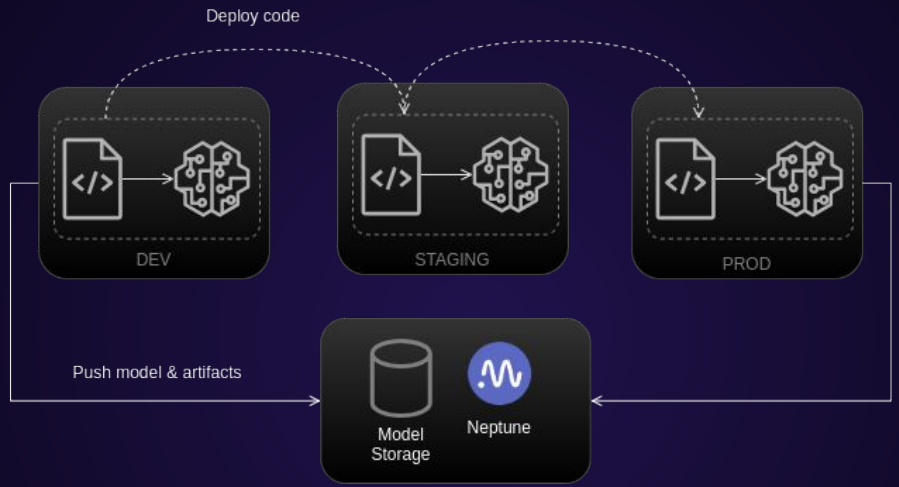


Multi-Environment Implementation



- Account Separation: Use separate AWS accounts for Development, Staging, and Production to ensure isolation and security.
- CI/CD Pipelines: Implement AWS CodePipeline to automate build, test, and deployment processes across all environments.
- Security: Implement IAM roles with least privilege, encrypted data storage, and logging across all environments.
- Infrastructure as Code (IaC): Use Terraform or AWS CloudFormation to define and provision infrastructure consistently across environments.

Recommended LLMOps Workflow



- Code Promotion Over Model Promotion: Promoting code rather than models ensures reproducibility and consistency across environments.
- Central Model Registry: Use a centralized model registry, such as Neptune, to track, version, and manage model artifacts. This helps in maintaining a single source of truth for models, making it easier to track their lifecycle.
- CI/CD Workflow Integration: Integrate CI/CD pipelines with Neptune tracker for stage transitions, request, review, and approve changes.