Analytics Vidhya | DATAHACK SUMMIT

Workshop

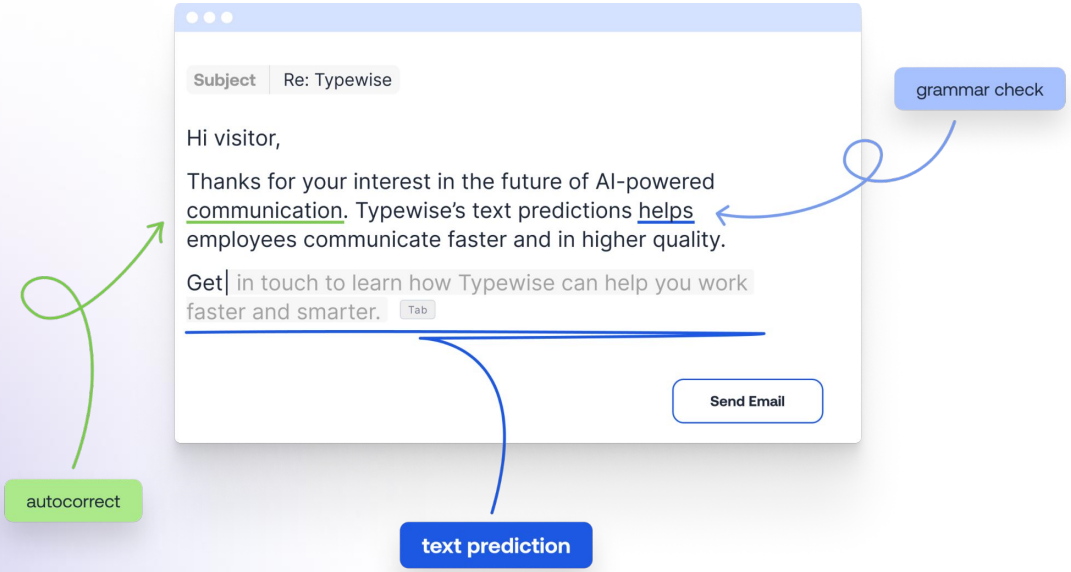
LLMOps – Productionalizing Real-world Applications with LLMs

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Typewise

3X your customer service & sales productivity with our AI Communication Assistant



- Researcher in reinforcement learning
- HSBC global DevOps
- HoE at AI startup funded by YC
- MLOps Engineer at Typewise

We are hiring for MLOps/DevOps !!

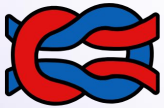
Course Overview Based on Feedback

Sagemaker Notebook

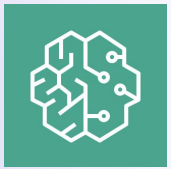
Kubernetes

LangChain
Prompt Management

Locust



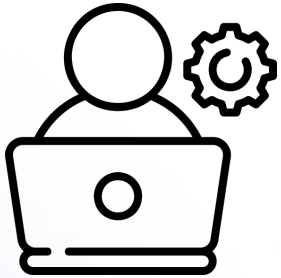
LangFuse
Prompt Management



Sagemaker Studio
Training & Pipelines

- All feedbacks taken
- Basic to advance
- LLMOps Overview >> Hands on session code and Slides available later

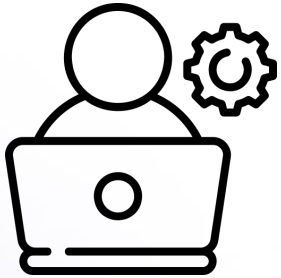
AI Research



As a GenAI Scientist / Specialist / Developer we want to:

- work with multiple foundation models
- do prompt engineering
- if needed, train on high-quality dataset
- if needed, access internal data with RAG

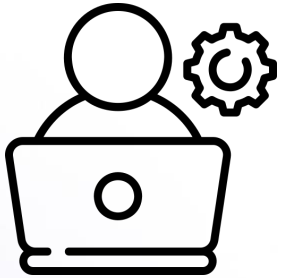
ML Engineer



As an AI/ML engineer, we want to:

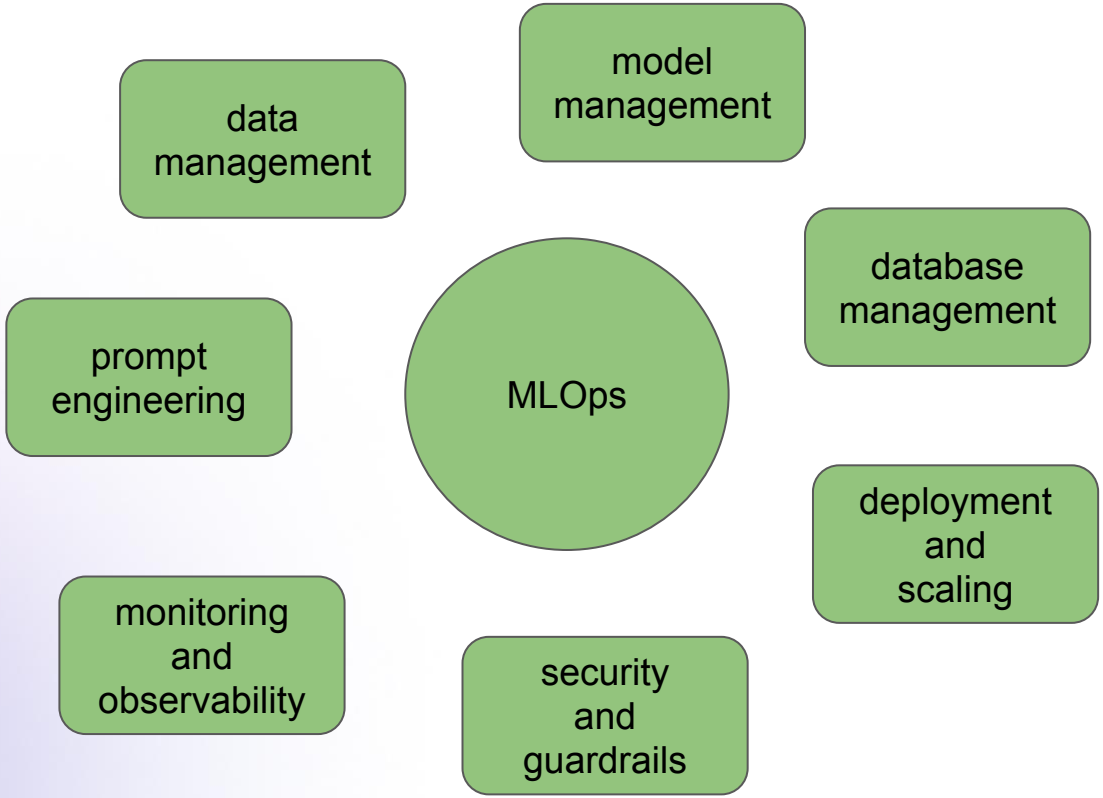
- work with multiple LLM APIs
- multi-GPU training
- manage vector databases
- deploy and scale ML models
- track and monitor all models

Why LLMOps ?



- Maintaining a single LLM project with multiple experiments can be challenging
- scaling to many projects can be a bottleneck
- A lot of the processes can be automated using LLMOps to avoid repetitive tasks and human errors
- Security and other best practices can be baked into the automated system

Major LLMOps components



LLMOps

Practices and processes involved in managing and operating large language models (LLMs)

Analytics Vidhya | DATAHACK SUMMIT

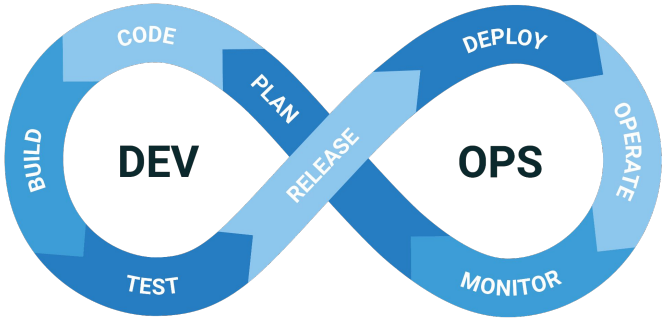
Workshop

CICD Pipelines

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Software Lifecycle



Continuous Integration - CI
DEV TEAM

- Plan the project scope
- VC for collaboration
- Central build and packaging
- Testing
- Release

Continuous Deployment - CD
OPS TEAM

- Deployment on environments
- Operate using tools
- Continuous monitoring

CI/CD in LLMs

Analytics Vidhya | DATAHACK SUMMIT

CI

Data Scientist / ML Engineer

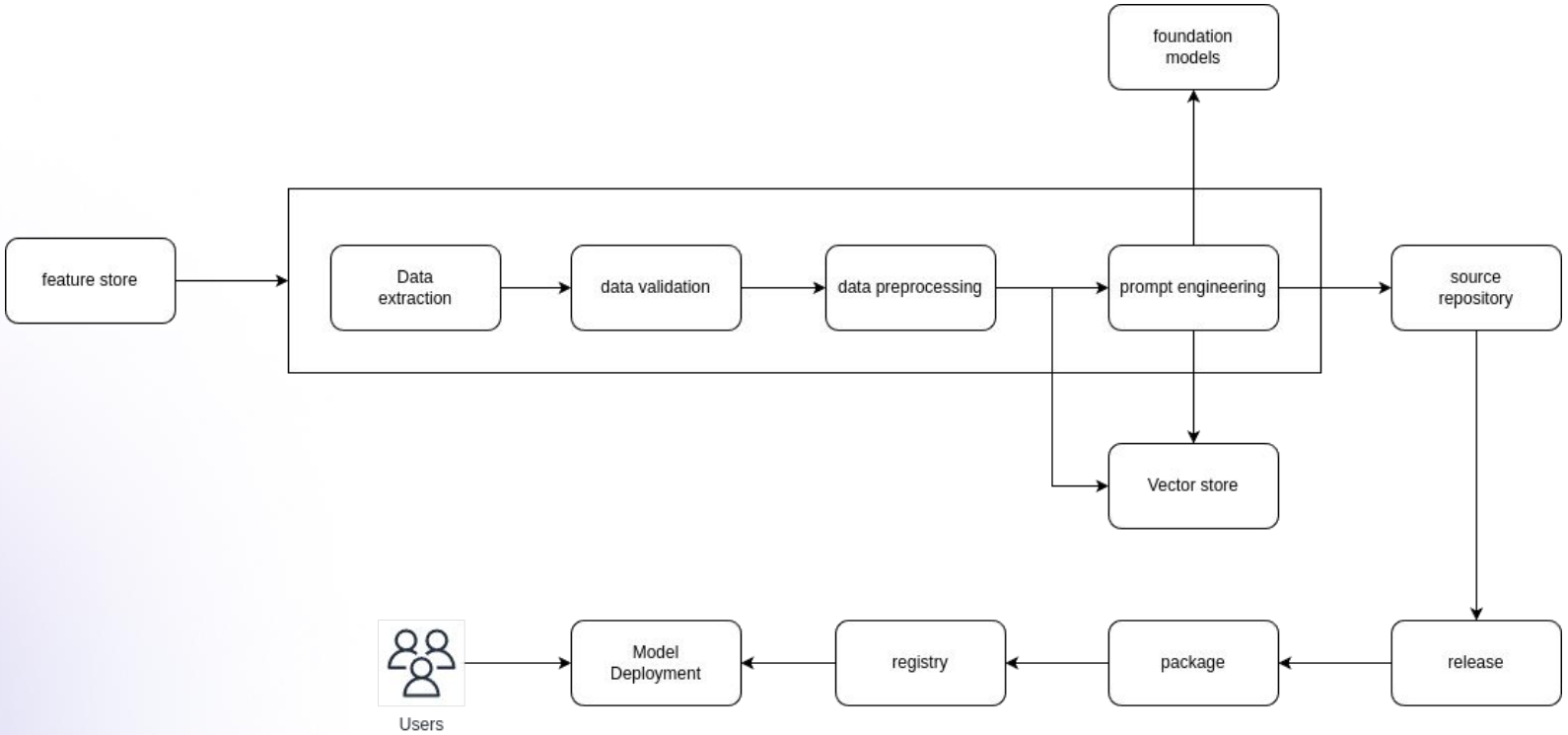
- Business use case problem framing
- Data store, Data Pipelines
- VC and easy collaboration
- EDA - exploratory data analysis
- Prompt engineering
- Training - distributed and scaled
- Reproducible experiments
- Continuous model feedback
- Re-training

CD

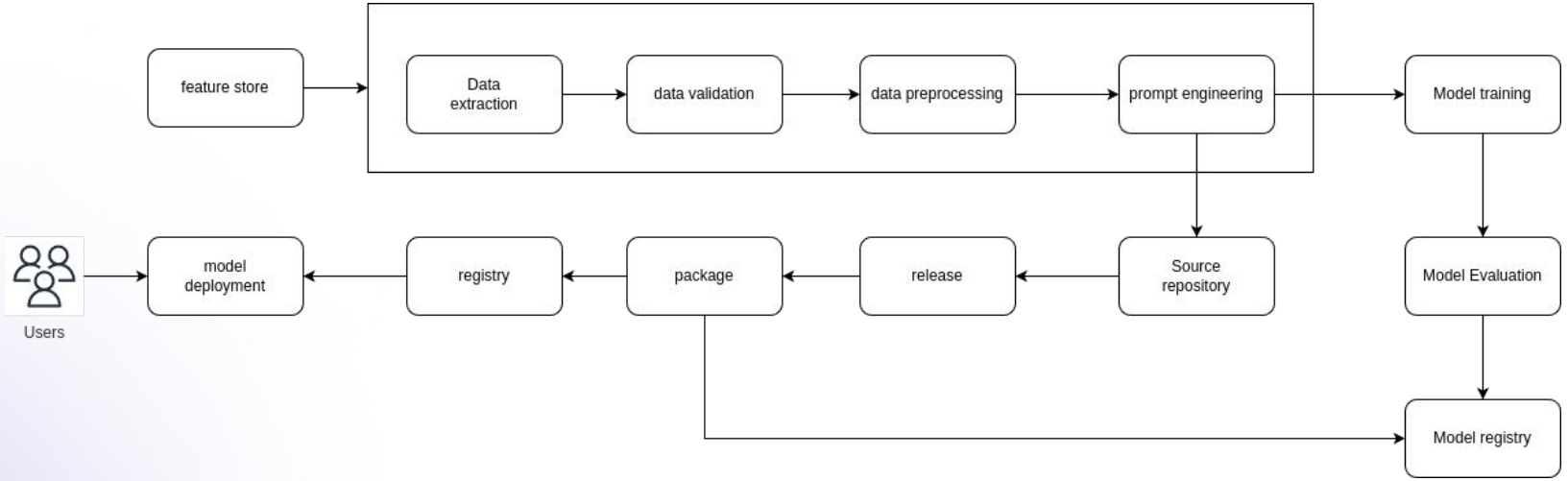
LLMOps Engineer

- Containerize models
- Security of servers and infrastructure
- Optimize startup and inference times
- Deployment
- Scaling models
- Monitor model performance

CI pipeline with Foundation models

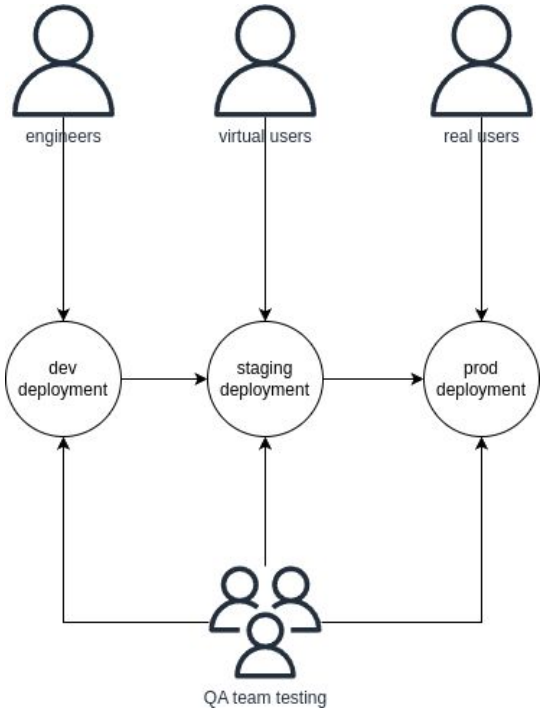


CI pipeline with model training



Advantages of CI/CD pipelines

- Automated infrastructure creation
- releasing to multiple environments
- Cost of each environment
- Multi-stage testing
- Security best practices
- Monitoring infrastructure changes



Analytics Vidhya | DATAHACK SUMMIT

Workshop

AWS Services

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Amazon Web Services

- Launched in 2006
- Started with EC2, S3 and SQS
- Over 200 services
- 33 Regions
- 105 Availability Zones

Create an AWS Account
aws.amazon.com

AWS Basics - Storage

EBS
Elastic Block Storage
Hard drives and SSDs

S3
Simple Storage Service
DropBox

ECR
Elastic Container Registry
Docker registry

AWS Basics - Compute

EC2
Elastic Compute Cloud
Datacenter servers

EKS
Elastic Kubernetes Service
Open source Kubernetes

AWS Basics - Developer tools

CodeCommit
similar to Github

CodeBuild
similar to Jenkins

CodePipeline
steps in series

AWS Basics - Orchestration

CloudFormation
IaC similar to Terraform

EventBridge
Cronjobs in Linux

AWS Basics - Artificial Intelligence

Sagemaker
ML Platform

Amazon Bedrock
Foundation models as API

Amazon Transcribe

Amazon Translate

Amazon Textract

Amazon Poly

AWS Basics - Identity and Access Management

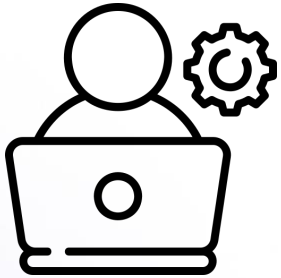
IAM
Identity and Access Management

Policy
Access and deny rules

User
User, groups, service accounts

Roles
Access to services

Setup up



- OpenAI token
- HuggingFace token
- HuggingFace permission
- Langfuse tokens
- AWS login
- AWS setup Studio

Analytics Vidhya | DATAHACK SUMMIT

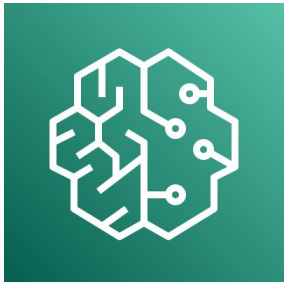
Workshop

Introduction to SageMaker

Speaker

Kartik Nighania
MLOps Engineer at Typewise

AWS SageMaker



Fully managed ML platform by AWS

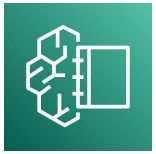
Complete ML CICD

- Default support for multiple libraries and frameworks such as HuggingFace, Pytorch, Scikit Learn
- Researchers can collaborate using various tools. For eg: VS Code IDE, Jupyter Notebook, Jupyterlab, RStudio
- Train models with automatic infrastructure provisioning
- Manage, deploy and maintain models

For Organizations

- No code solutions available for non-tech practitioners
- Team management and fine-grained access control
- Enforce infrastructure security best practices
- Easy to share findings with business stakeholders

Development Environments



Studio Lab
Free service by AWS

Canvas
No code service

RStudio
RStudio IDE in AWS

Notebook
Standalone Jupyter IDE

Studio
Complete ML Platform

Analytics Vidhya | DATAHACK SUMMIT

Workshop

Introduction to LangChain

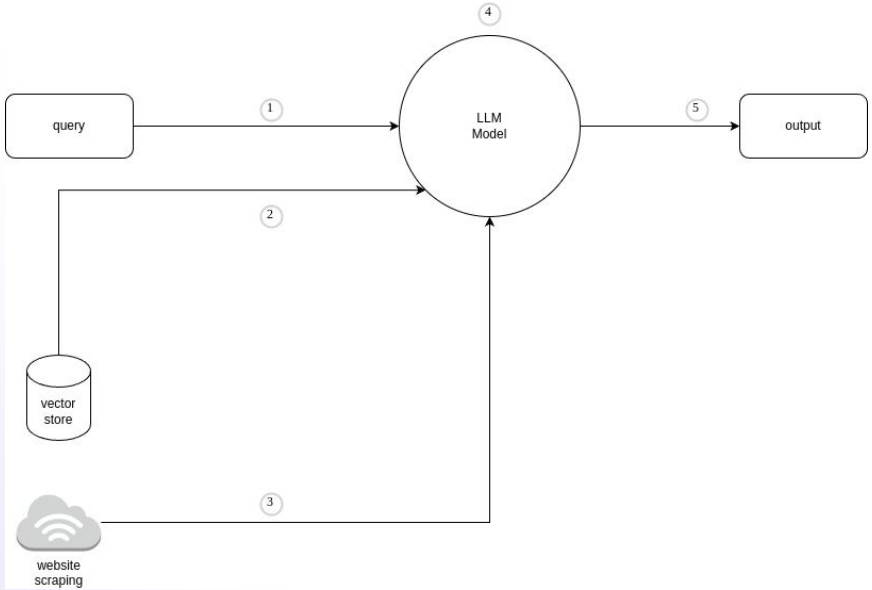
Speaker

Kartik Nighania
MLOps Engineer at Typewise

Introduction to LangChain

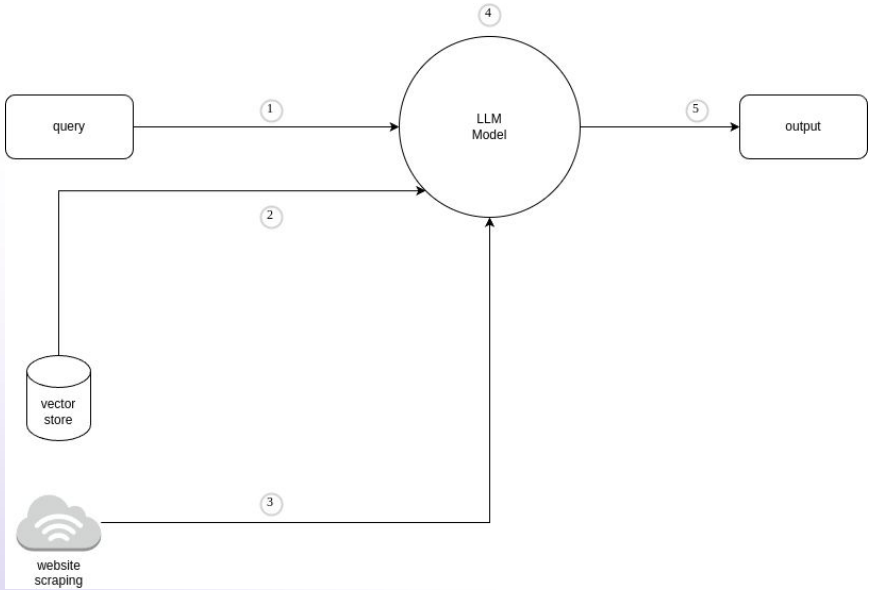
- LangChain is a framework for developing applications powered by large language models (LLMs)
- Available as a library in Python and Javascript
- Integrations with many third party APIs like Ollama, OpenAI
- Open source with 80k+ stars on Github

Why do we need LangChain ?



- Query changes with prompt engineering
- RAG: Vector store retrieval
- Tools: Website scraping
- Model with parameters
- Output structure parsing and delivery

LLM ecosystem



LangChain

- multiple prompt types
- multiple vector stores
- multiple tools
- multiple models
- Multiple output parsers
- agents

Analytics Vidhya | DATAHACK SUMMIT

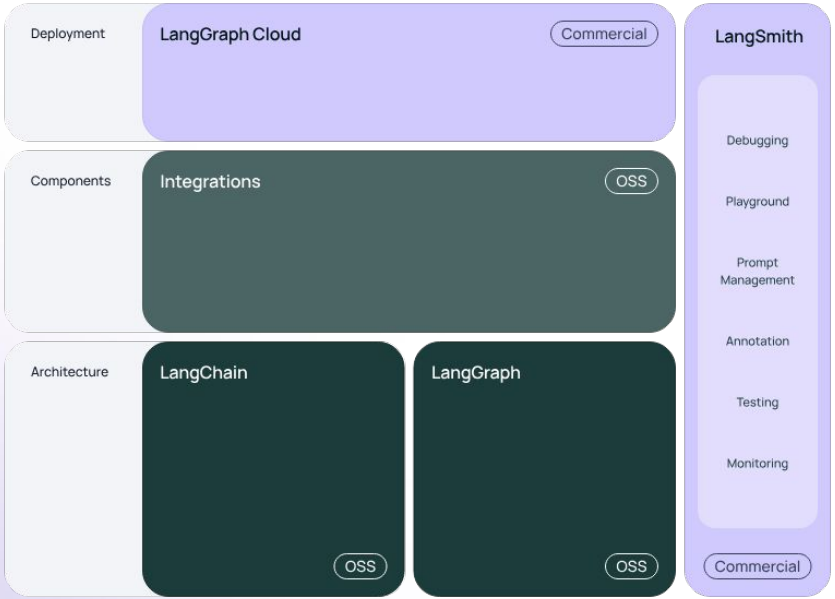
Workshop

LangChain for Continuous Integration

Speaker

Kartik Nighania
MLOps Engineer at Typewise

LangChain for Continous Integration (CI)



- LangChain: Chains, agents, and retrieval strategies
- open-source libraries:
  - core: Base abstractions and LCEL
  - community: e.g. @LangChain/openai
- LangGraph.js: multi-actor applications

- LangSmith: A developer platform that lets you debug, test, evaluate, and monitor LLM applications.

- LangGraph Cloud: Deploy LangGraph agents
- LangServe: Chains as REST APIs

- OSS vs Commercial

LangChain Components

LLMs
Raw foundation models

Chat Models
Fine-tuned for chat instructions

Agents
Self guiding LLMs

Document Loaders
Data ingestion

Chat Loaders
Data ingestion in chat format

Document Transformers
Convert raw data

Text Embedding Models
Embeddings for search

Memory
Chat messages

Graphs
Graph databases

Vector Stores
Database for search

Model Cache
Output caching

Stores
Storage solutions

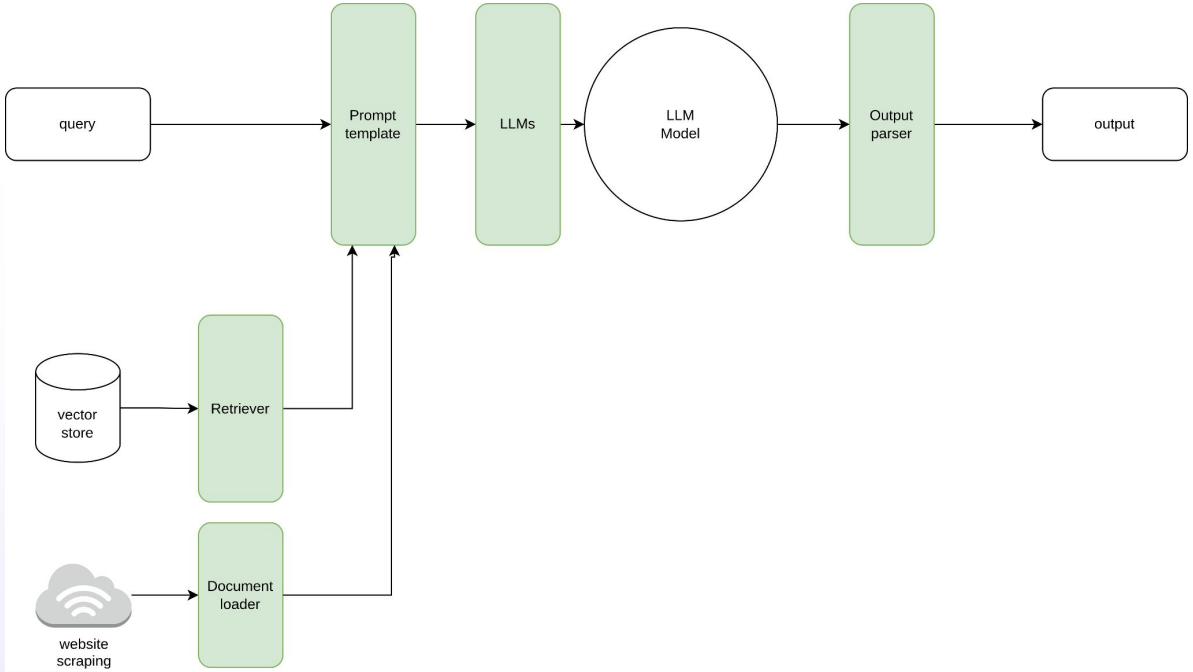
Retrievers
All data level transformations

Tools and Toolkit
Functionalities for LLMs and Agents

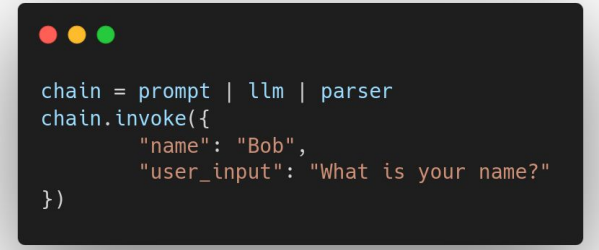
Callbacks
Deeper integration

Adapters
Copy interface

RAG pipeline with LangChain



Advantages of LCEL



LCEL stands for LangChain Expression Language
Orchestration that is declarative and composable in nature

- Streaming support
- Async and multi chain parallel execution support
- Retries and fallback support
- Intermediate steps and callback support
- Streaming for different output types with partial schema
- LangSmith for monitoring
- LangServe for deployment

Analytics Vidhya | DATAHACK SUMMIT

Workshop

Introduction to Langfuse

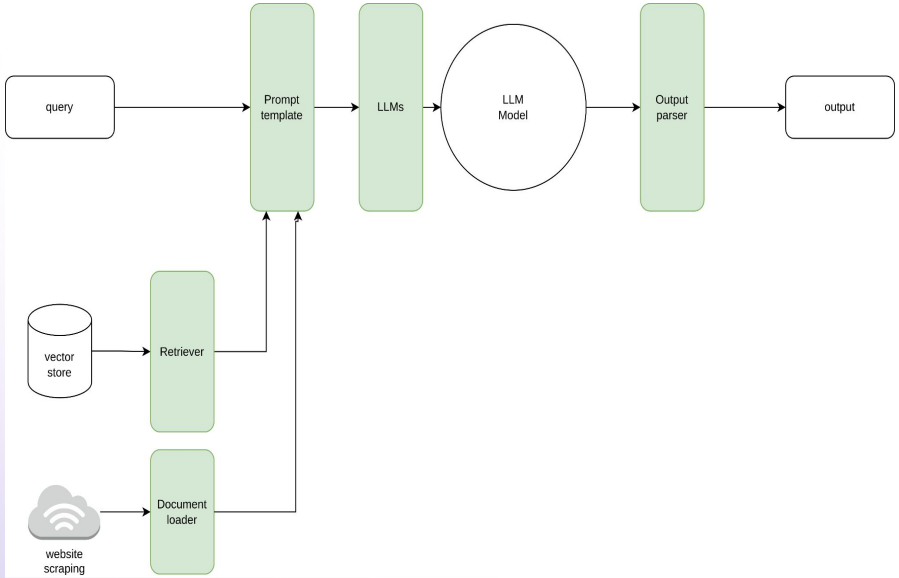
Speaker

Kartik Nighania
MLOps Engineer at Typewise

Introduction to Langfuse

- LLM Engineering Platform
- Open Source with 4.7k+ stars on Github
- Available as a paid cloud solution or self-hosted
- Can be used as an SDK in Python or Javascript
- Integrations with third party APIs like LlamaIndex and LangChain
- REST APIs also available

Why do we need Langfuse ?



- For a single LLM application in production we need to
  - Debug issues in production
  - Analyze and iterate on multiple solutions
  - Monitor the performance in production
  - Continuously run tests and create new releases
- Multiple teams needs to research and collaborate on multiple projects
- Multiple LLM applications needs to be released and maintained at the same time.

Langfuse ecosystem

Develop

- Track LLM calls and other app logic
- Supports API, SDK and 3rd party libraries
- Langfuse UI: inspect and and debug calls
- Prompt Engineering with playground
- Prompt Management

Monitor

- Analytics Dashboard: cost, latency, quality
- Evaluation: custom LLM based evals
- Real user feedback with scores and comments

Test

- Datasets: benchmark performance
- Track version and releases

Analytics Vidhya | DATAHACK SUMMIT

Workshop

Langfuse for CI/CD

Speaker

Kartik Nighania
MLOps Engineer at Typewise

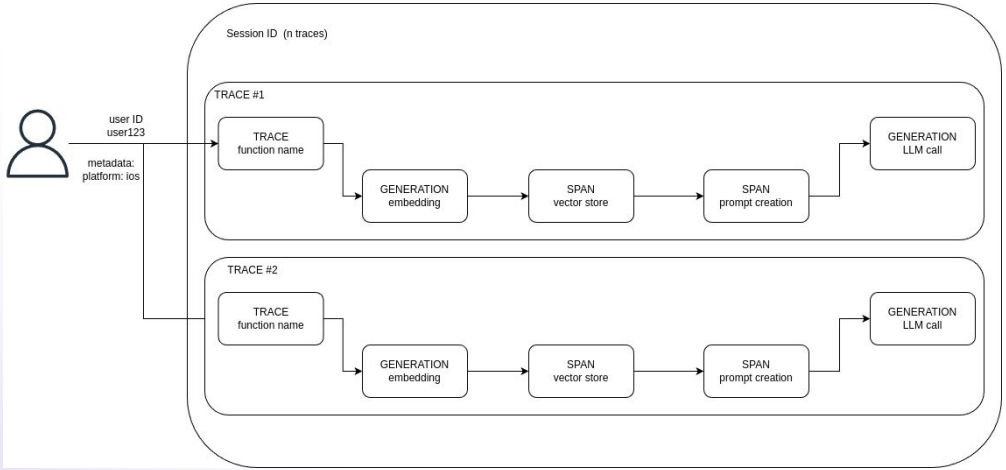
Langfuse for CI/CD

- Langfuse components
- Langfuse integration with Langchain
- Versioning and releases with Langfuse

Team and data management

- Projects to group experiments
- Users with Single Sign-On
- RBAC support
  - Owner: all ownership
  - Admin: all owner feature except project delete and transfer
  - Member
  - Viewer
- Cloud based SaaS solution provided
- On-prem deployment also available
- ISO and SOC2 certifications

Tracing in Langfuse



- **Trace:** A trace typically represents a single request or operation.
- **Session:** grouping of multiple traces
- **Span:** unit of work in a trace
- **Generation:** log generation of AI models. Additional information of the prompt, model and completion

Langfuse python Decorator

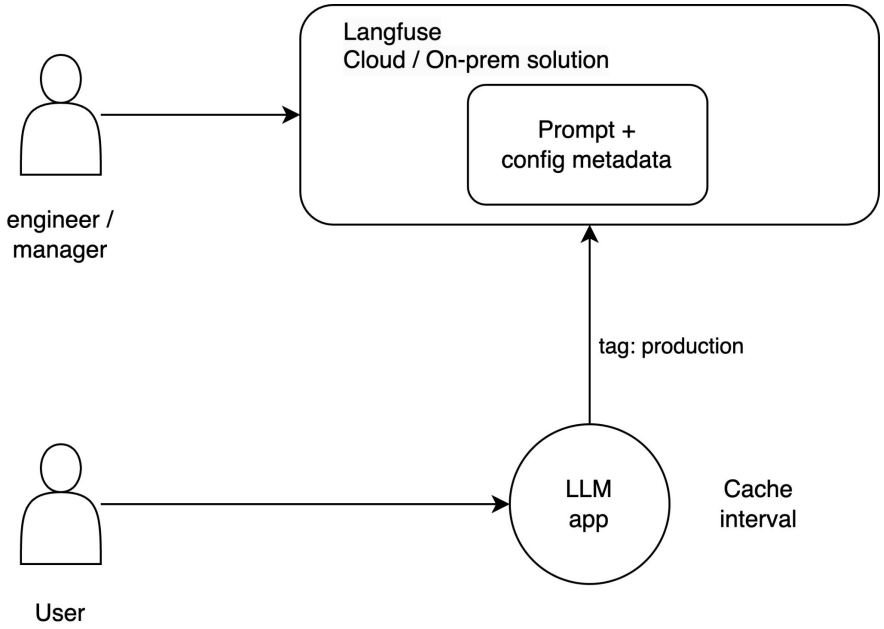
The @observe() decorator can be applied to functions we want to trace.

By default it captures:

- timings/durations
- function name
- args and kwargs as input dict
- returned values as output
- Additional tags, metadata and IDs can be added

- Support for callback in LangChain

Langfuse for Prompt management



- Decoupling: deploy new prompts without re-deploying your application.
- Non-technical users can create and update prompts via Langfuse Console.
- Quickly rollback to a previous version of a prompt.

Platform benefits:

- Track performance of prompt versions in Langfuse Tracing.

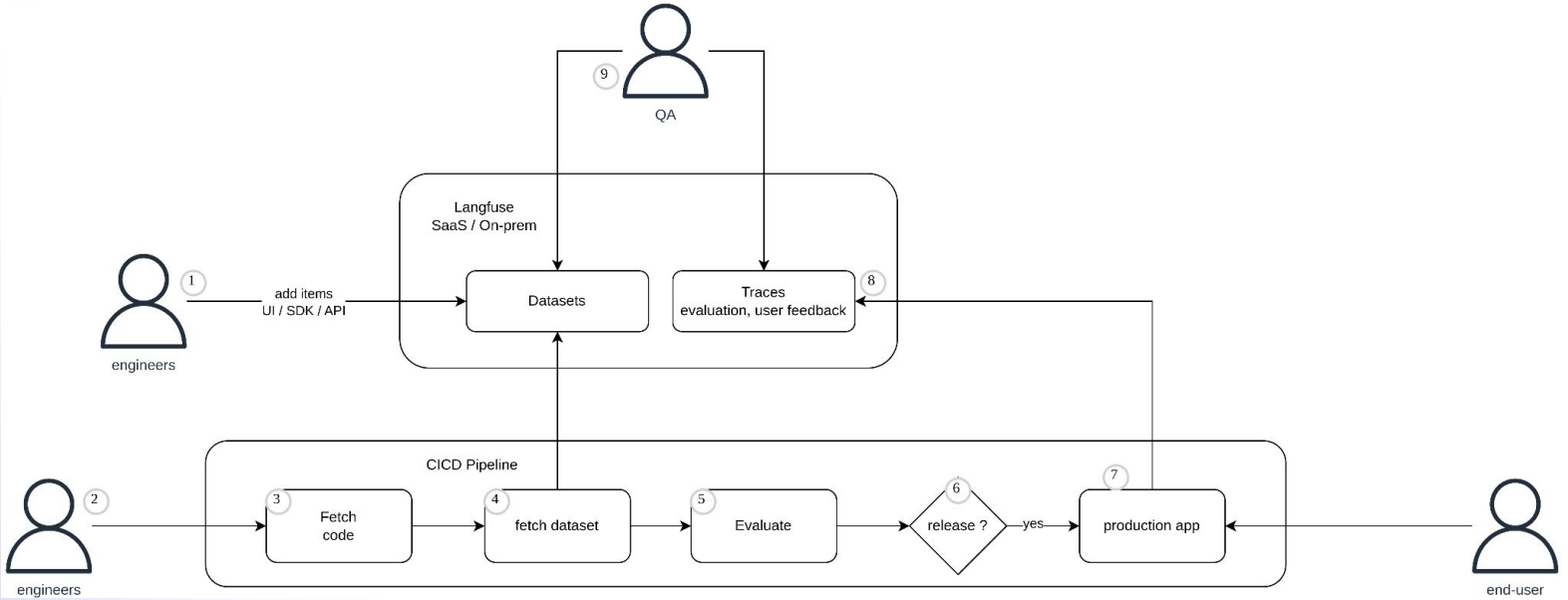
Langfuse Scores and evaluations

- Scores serve as an object to store evaluation metrics in Langfuse
- Score consists of NUMERIC, CATEGORICAL or BOOLEAN dataType, name and value/stringValue and additional metadata such comment, source and various IDs

- Scores can be of various types:
  - Self annotated in UI: manual scores given by the team
  - User Feedback: Can be integrated in the UI and directly shown to the user
  - Model-based Evaluation:
    - we can use LLMs to score trace calls based on varied criterias like toxicity, hallucination etc
    - We can also create our own custom model evaluation.
    - These can be run automatically after every trace based on certain filter criteria
    - We can also fetch the runs and upload the computed scores to the UI

Langfuse Datasets

- Datasets in Langfuse are a collection of inputs (and expected outputs) of an LLM application.
- They are used to benchmark new releases before deployment to production.



Analytics Vidhya | DATAHACK SUMMIT

Workshop

SageMaker Deep Dive

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Domains

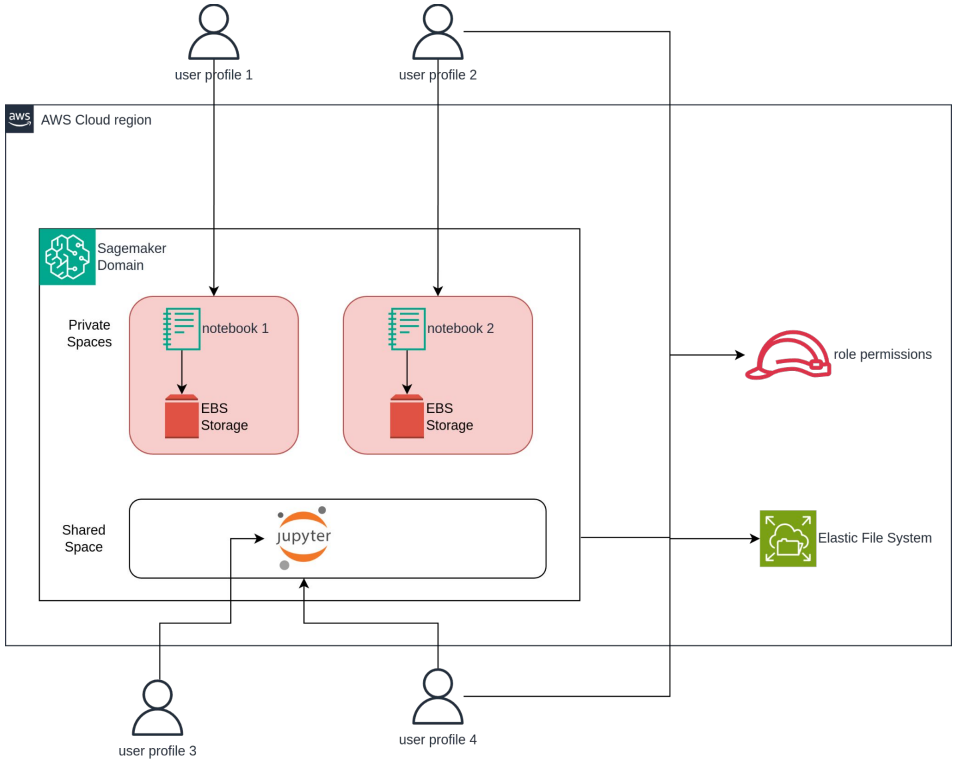
Domain consists of

- A list of authorized users
- An EFS volume
- Private and public shared spaces
- All shared AWS SageMaker services can be seen by all uses within a given domain

Domain can be created by:

- **Quick setup**
  - Domain created accessing the AWS resources using public internet
- **Standard setup**
  - AWS resources are accessed internally without connecting to the internet
  - More fine grained control over security and networking

Domains


user profile 1
user profile 2
AWS Cloud region
Sagemaker Domain
Private Spaces
notebook 1
EBS Storage
notebook 2
EBS Storage
role permissions
Shared Space
jupyter
Elastic File System
user profile 3
user profile 4

SageMaker Virtual Machines

https://aws.amazon.com/sagemaker/pricing/

On-Demand Pricing

Studio Classic | JupyterLab | Code Editor | RStudio | Notebook Instances | Processing | TensorBoard | Data Wrangler | Feature Store | Training | MLflow | Real-Time Inference | Asynchronous Inference | Batch Transform | Serverless Inference | JumpStart | Profiler | HyperPod | Inference optimization

Amazon SageMaker Studio Classic

Studio Classic offers one-step Jupyter notebooks in our legacy IDE experience. The underlying compute resources are fully elastic and the notebooks can be easily shared with others, allowing seamless collaboration. You are charged for the instance type you choose, based on the duration of use.

Region: Asia Pacific (Mumbai)

Standard Instances	vCPU	Memory	Price per Hour
ml.t3.medium	2	4 GiB	$0.054
ml.t3.large	2	8 GiB	$0.108
ml.t3.xlarge	4	16 GiB	$0.215
ml.t3.2xlarge	8	32 GiB	$0.43
ml.m5.large	2	8 GiB	$0.121

- EC2 servers are prefixed with **ml** for SageMaker
- Based on Amazon Linux 2 OS

Instance classes:

- Standard Instances: t, m series
- Compute optimized: c series
- Memory optimized: r series
- GPU optimized: g, p series
- DL Inference optimized: inf series

Training with SageMaker

Built-in Algorithms

Default support for common algorithms from various providers

Default support for frameworks:

- PyTorch
- Tensorflow
- Hugging Face
- Chainer
- Scikit Learn
- MXNet
- Spark ML

Script Mode

- Container provided by AWS
- User provides script file to run with the container
- Support for training and deployment

Custom Container

- Create your own deep learning container
- Support for training and deployment with other SageMaker resources

Analytics Vidhya | DATAHACK SUMMIT

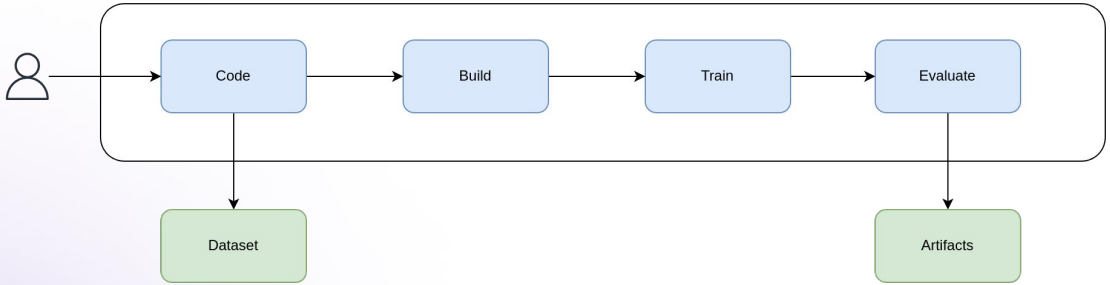
Workshop

SageMaker for model training

Speaker

Kartik Nighania
MLOps Engineer at Typewise

ML Workflow



We need to organize, track, compare and evaluate ML experiments

What needs to be tracked ?

- Dataset
- Data preprocessing code
- Versioned libraries and framework
- Hyperparameters
- Experiment logs
- Training metrics
- Evaluation metrics
- Model weights
- Compare different runs together

Development Environment

Runtime environments

- Code Editor: VS Code
- JupyterLab
- Canvas
- RStudio
- Studio classic
- MLflow

Integration

- REST APIs
- CLI
- SDK

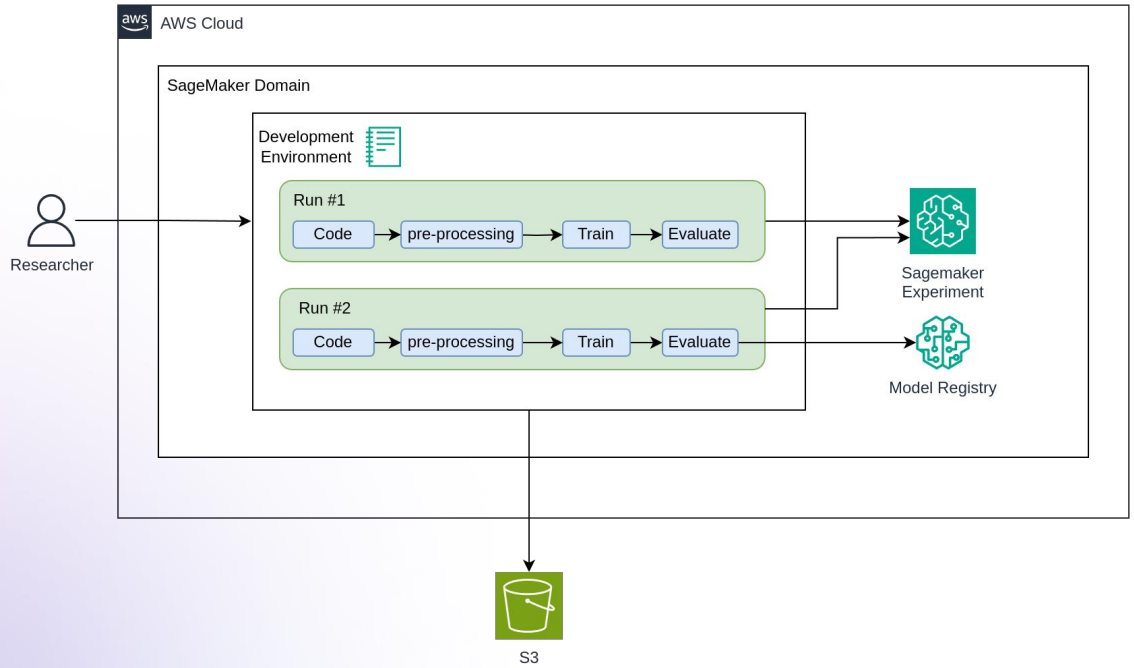
Python SDK

- SageMaker pypi
- Boto3 pypi

Other services

- Jumpstart
  - Quickly test FM
  - Test end-to-end solutions from AWS Marketplace
- Bedrock
  - Pay as you go FM in AWS regions

SageMaker Experiments



- Experiment consists of multiple runs in it
- Run consists of multiple metrics logged and stored in SageMaker experiment
- Deep integration with SDK
- Organized in Studio dashboard UI

As of Nov 23

- MLFlow is the default one
- We also have Tensorboard integration

Analytics Vidhya | DATAHACK SUMMIT

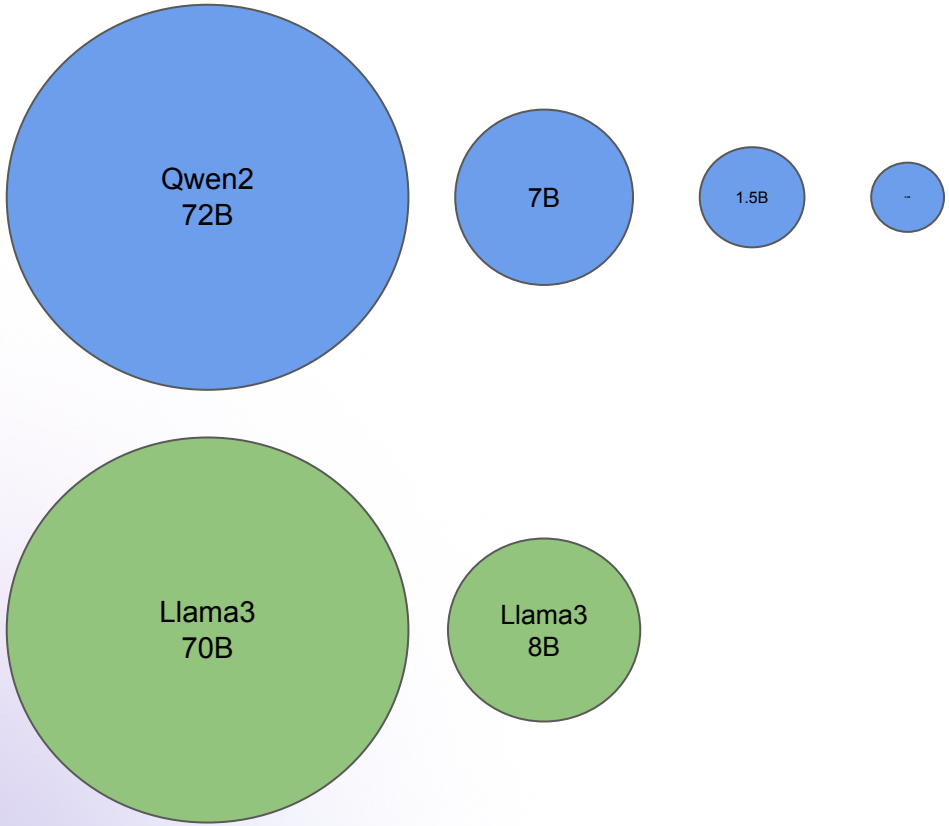
Workshop

Project Overview

Speaker

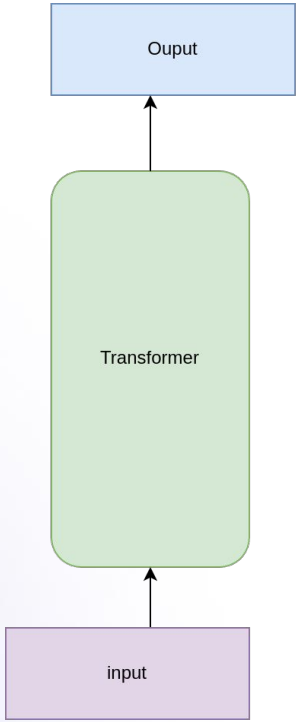
Kartik Nighania
MLOps Engineer at Typewise

State of LLMs



- The rise of open source models like Llama, Qwen, MistralAI performing close to proprietary models like OpenAI and Anthropic in multiple LLM Benchmarks
- During inference 1 Billion params @ 32-bit FP
  4 Bytes x 10e9 = 4 GB of VRAM
- Llama3 70B inference cost: 70 x 4 = 280 GB of VRAM
- AWS p4d.24xlarge
  - 96 vCPUs
  - 1237 GB
  - 8 x A100 GPUs = 320 GB
  - On-demand = 32.77 USD/hr
  - Per month cost = $23594.4 = 19.7 lacs

Full fine-tuning



Training GPU usage:

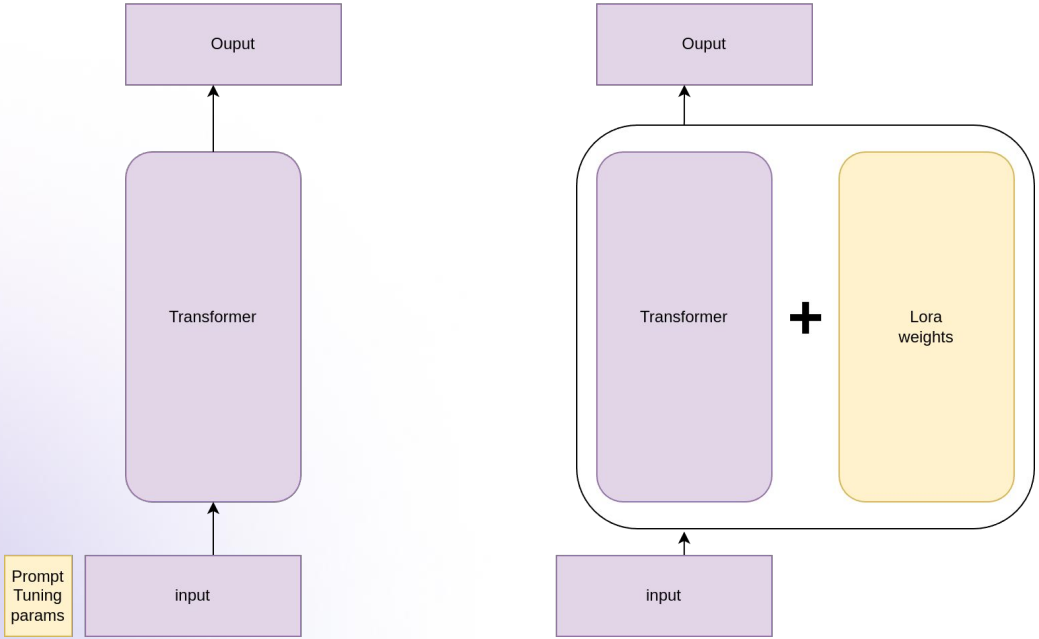
- For 1 parameter in transformer weights
  - 2 x Optimizer states
  - 1 x Gradient
  - 2 x Activations and temp memory

4 GB weights = 4 x 6 = 24 GB @ 32FP

Issues with full fine-tuning:

- large memory size
- Maintaining and deploying multiple models
- Catastrophic forgetting on fine-tuning leading to poor results on general tasks

Parameter efficient fine-tuning



Fine-tuning

- Fine-tuning techniques like SFT, RLHF, PPO
- We will fix to SFT Supervised Fine Tuning

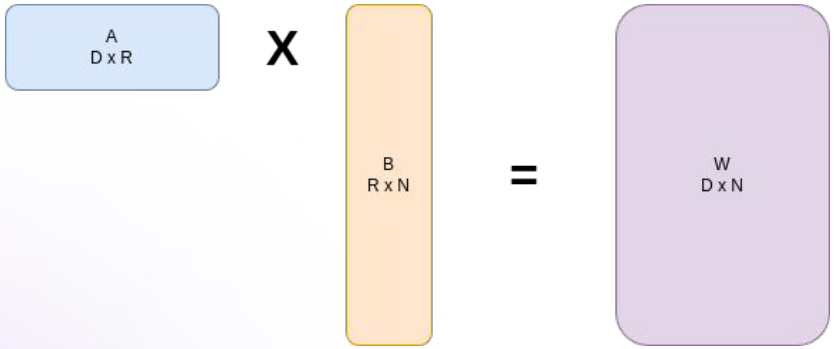
In SFT

- All or some weights kept frozen
- Either small number of layers are trained
- Or new layers are trained and added later

Popular techniques include

- Prompt Tuning
- LoRA

LoRA: Low-Rank Adaptation



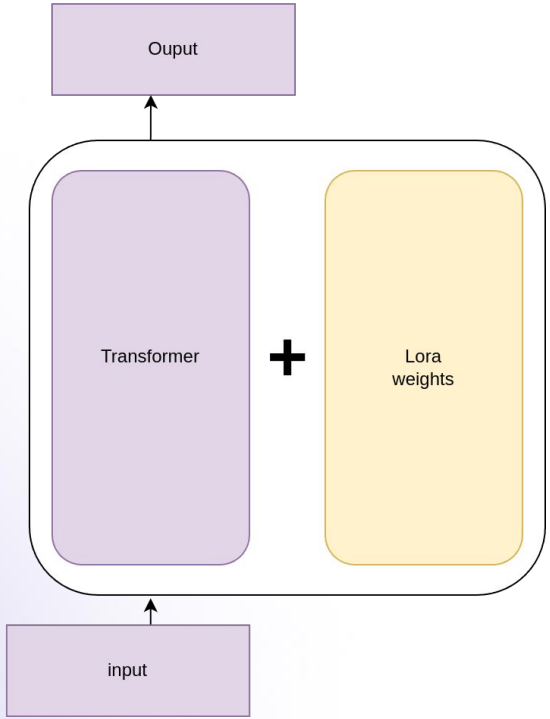
LoRA training

- Reparameterize by adding low rank decomposition matrices to different layers of the transformer
- Subset of params to fine tune - mostly the later layers as they contain most of the combined knowledge
- By research at Microsoft a value of rank between 4-10 gives good performance

Advantages of LoRA

- Single GPU training for 7-10 billion parameter models
- Scores are comparable to full-fine tuned models
- Many LoRA adapters can be trained for several downstream task with the same base model
- No added latency due to fused weights at runtime

Project Overview



- Prepare dataset for fine-tuning
- Foundation Model as base model
- Fine tuning using Hugging Face libraries
  - Quantization to reduce model size
  - **QLoRA** based supervised fine-tuning (SFT)
- Evaluate the LLMs output

Analytics Vidhya | DATAHACK SUMMIT

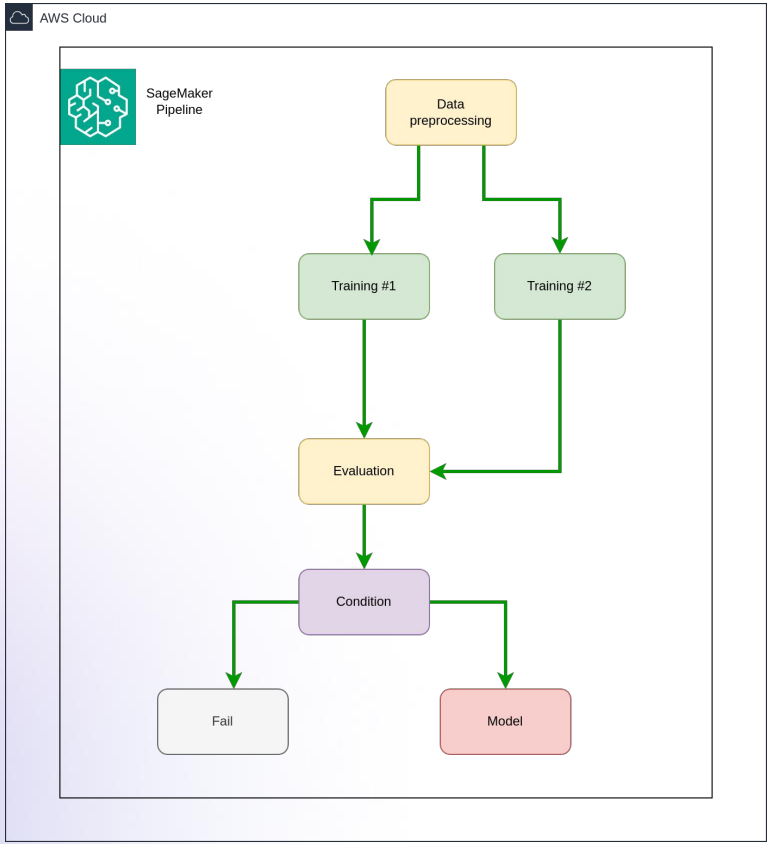
Workshop

SageMaker Pipelines

Speaker

Kartik Nighania
MLOps Engineer at Typewise

SageMaker Pipeline



- A series of steps with a specific purpose
- A directed acyclic graph (DAG) is automatically created
- Automatically creates and manages EC2 infrastructure
- Available as a Python SDK
- Fully integrated with SageMaker Studio UI

Why do we need a pipeline ?

Automatic step retry
Step caching

Global parameters

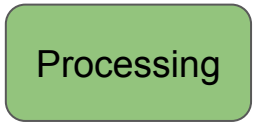
Data Lineage Tracking
Integration with S3 for IO

containerization

Infrastructure provisioning
Support for spot-instances

SageMaker Experiments integration
Automatically track hyperparameters
Model artifacts
Evaluation metadata
Other metrics

SageMaker Pipeline Steps



Processing Step

- Used to create a processing job for data processing

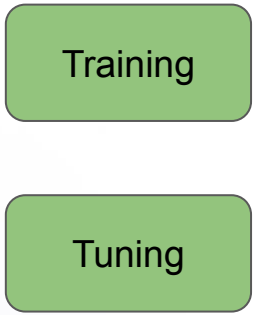
Use case

- For data pre and post processing
- For model evaluation after training
- Can also output files as evaluation reports

How it works

- SageMaker starts a processing job with an EC2 machine
- Takes the data from S3 and copies it over EC2
- Runs the container with the given script
- Copies the output artifacts back to S3
- De-provisions the EC2 machine

Model Training steps



- **Training Step:** Used to create a training job for training a model
- **Tuning Step:** Used to create a hyperparameter tuning job for a given model

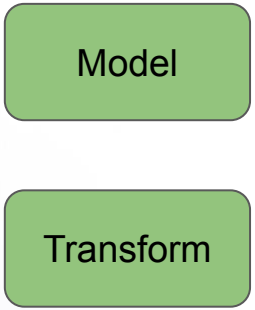
Use case

- Train model with pre-built containers or use a custom container
- Hyperparameter tuning can run models in parallel to speed-up training time
- Based on the tuning objective top 50 performing versions are retained

How it works

- SageMaker starts a job with EC2 machine(s)
- Takes the training and validation data from S3 and copies it over EC2
- Runs the container(s) with the given script
- Tracks container logs
- Tracks metrics using the container logs regex pattern
- After training de-provisions the EC2 machine(s)

Model Inference Steps



Model Step: Creates a new model for the model registry.
The Model Registry is a versioned system of managing models and has all the information needed to successfully retrieve/deploy the versioned model

Transform Step: Used to batch run inference of a model on a given dataset

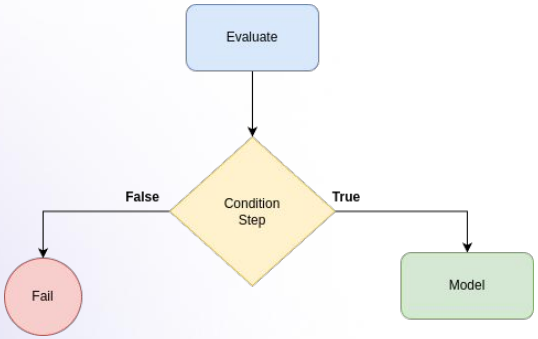
How it works

- SageMaker starts a transform job with EC2 machine(s)
- Takes the data from S3 and optionally splits into chunks
- Runs inference on the dataset
- Optionally aggregates the data and store the output data to S3
- Afterwards de-provisions the EC2 machine(s)

Logical Steps

Condition

Fail



Condition Step: compare step properties of a previous step with the defined condition(s) to take further steps based on evaluation results

Fail step: stop execution when a desired condition or state is not achieved. This also marks that pipeline's execution as failed.

Use case:

- Fail the pipeline if a given target evaluation metric is not achieved
- Early stop a pipelines execution to save on AWS resources
- Do not register a given model if the performance is not acceptable

Event based systems

Callback

Lambda

EventBridge

Callback step: Send a message to AWS SQS queue and halt the pipeline till further response is received from the queue consumer

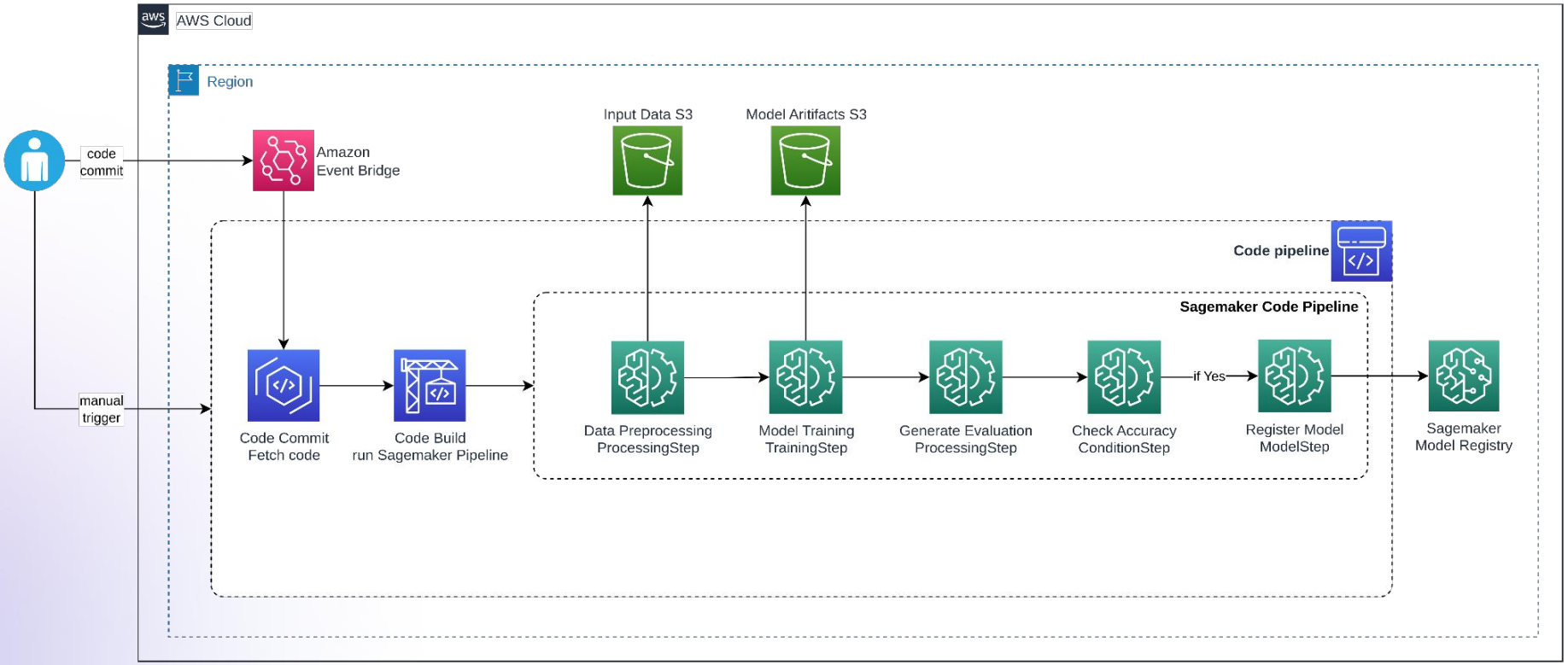
Lambda step: Create a Lambda function or run an existing lambda function. The pipeline waits for the Lambda function (max 10 mins) or the execution fails.

AWS EventBridge: Service that can monitors status change events across many features in SageMaker. These triggers in the state changes of SageMaker pipeline, jobs, model etc can be used to further notify and automate systems.

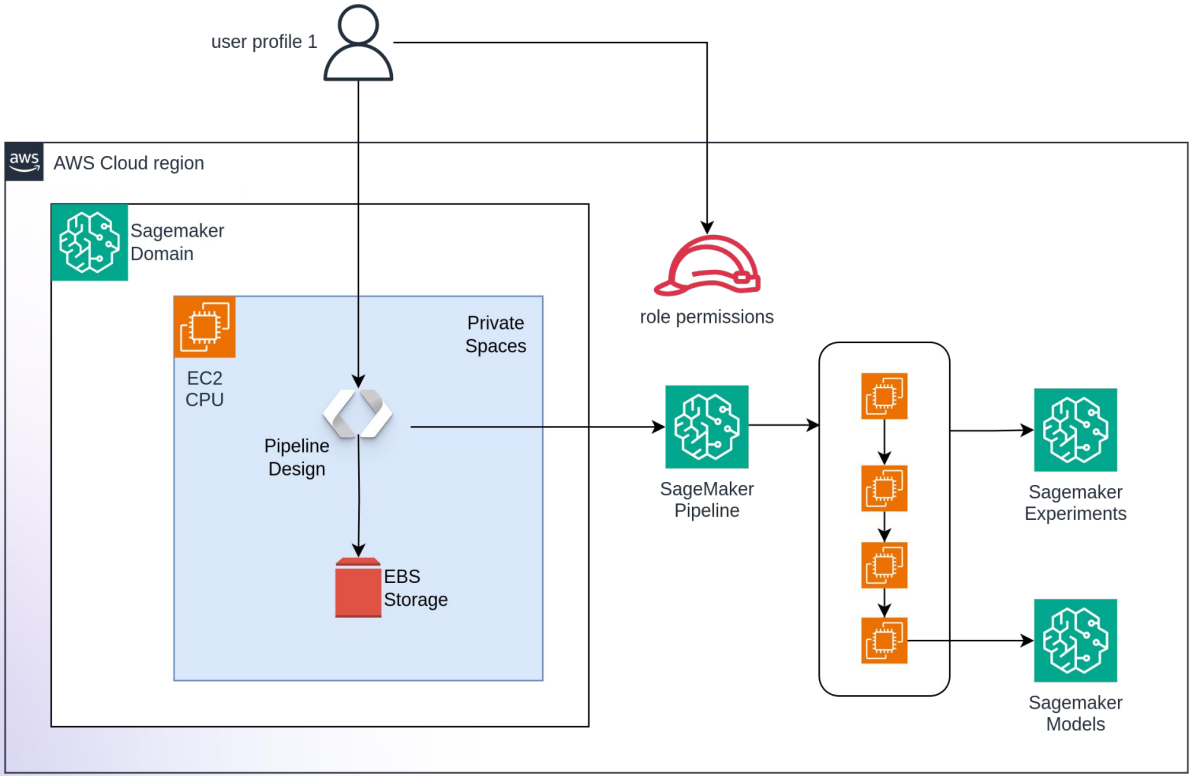
Use cases:

- Notify team on pipeline, job or training status
- Deploy a model when its state changes to approved
- Call other microservices and notify transform job completion

Pipeline Example



SageMaker Infrastructure



- Standardized library requirements using containers
- Modularized Data fetching and pre-processing
- Run different steps on different machines for cost saving
- Infra-scaling for pre-processing and hyperparameter tuning
- Automatic tracking in SageMaker Experiments
- Easy pipeline rerun using global parameters

Analytics Vidhya | DATAHACK SUMMIT

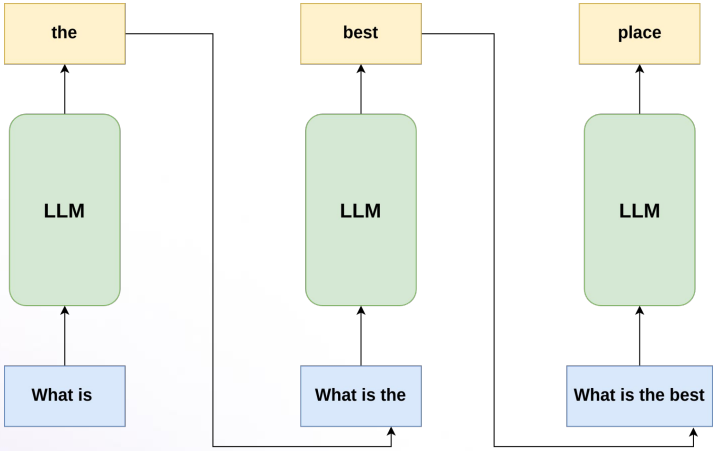
Workshop

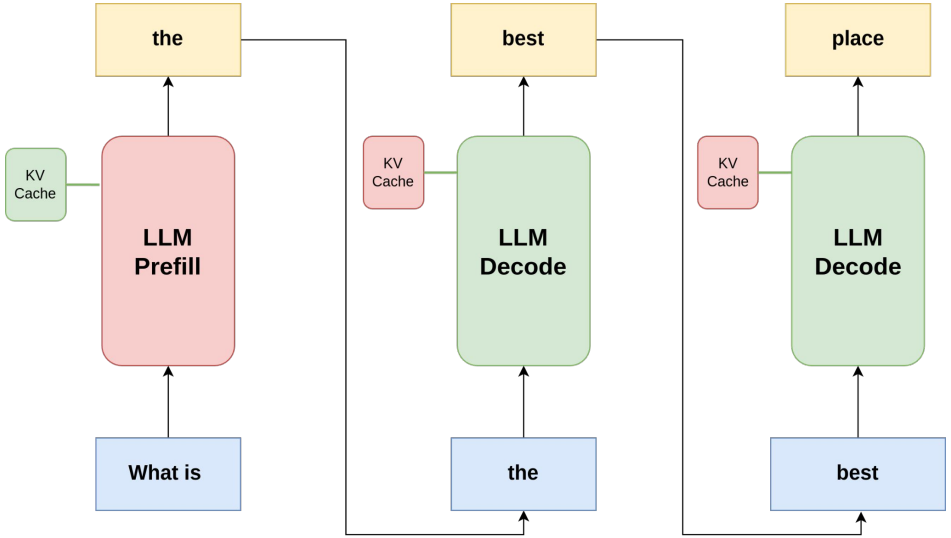
Continuous Deployment

Speaker

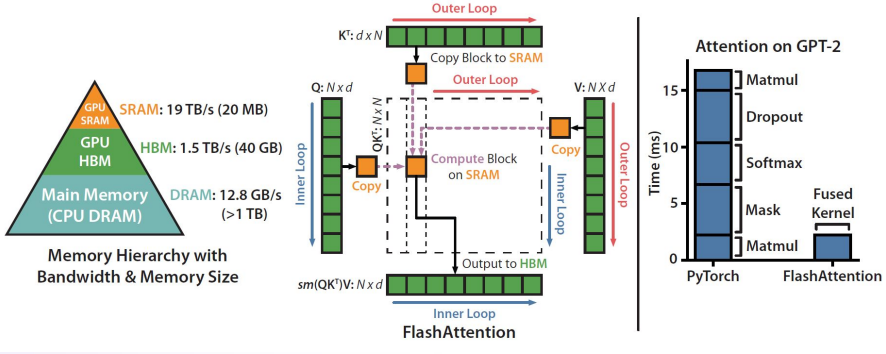
Kartik Nighania
MLOps Engineer at Typewise

KV caching


the
best
place
LLM
LLM
LLM
What is
What is the
What is the best


the
best
place
KV Cache
KV Cache
KV Cache
LLM Prefill
LLM Decode
LLM Decode
What is
the
best

KV Cache Optimization



KV cache Optimization

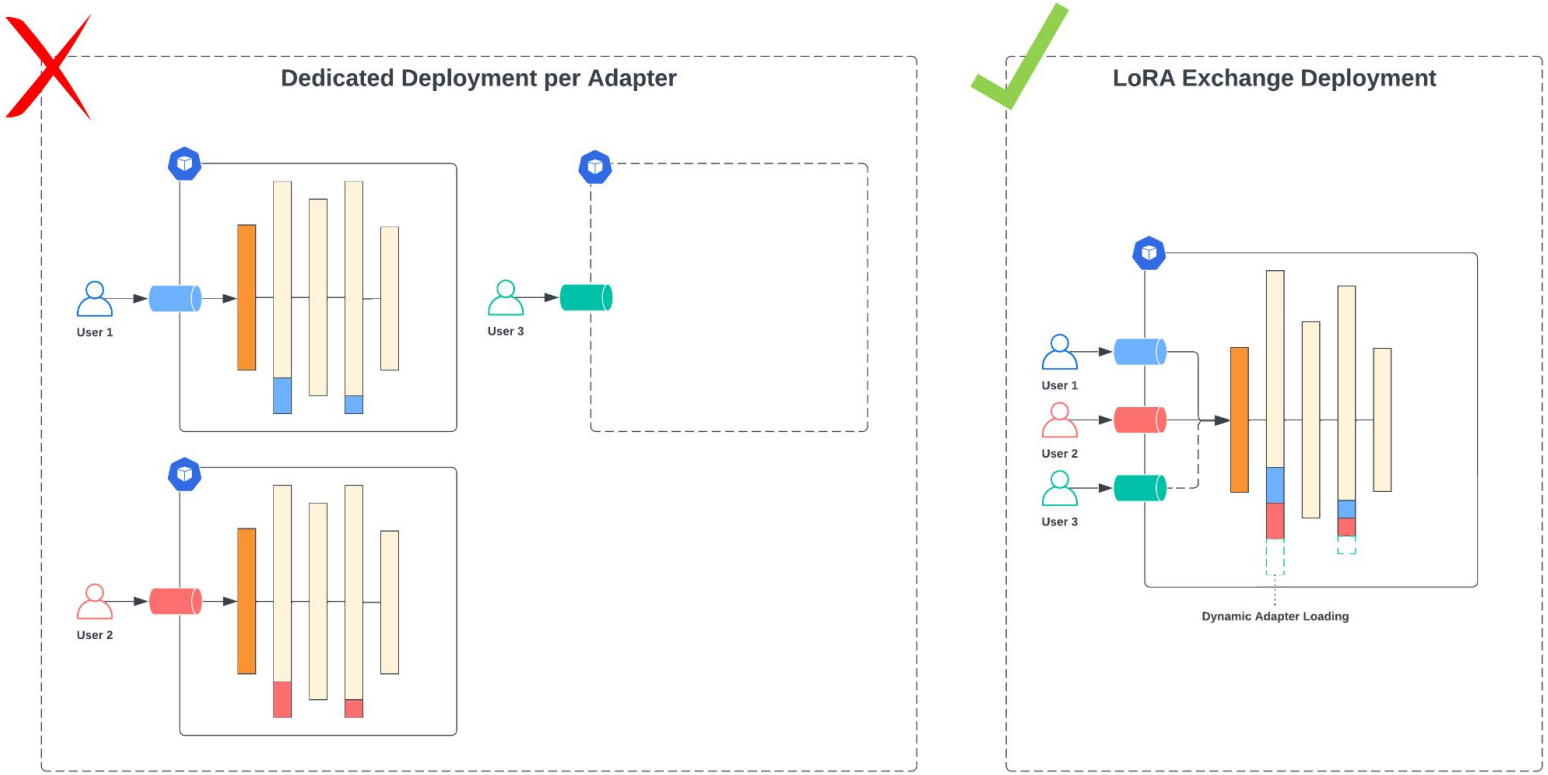
Flash attention

- effectively use of GPU memory layers
- fusing multiple kernels
- Used both at training and inference

PagedAttention of KV cache

- Saving KV cache in noncontiguous space in memory
- Saves wasted slots due to fixed storage reservation on context length

LoRAX: LoRA Exchange



Inference Optimization

Analytics Vidhya | DATAHACK SUMMIT

Static Batching

- Multiple request processed by the model at every inference step
- All request prefill and decode together
- Latency vs throughput tradeoff

Continuous Batching

- Token level batching. Prefill and Decode phases while processing request at a given time
- Easy swap-in new request tokens
- Swap-out token and return request output

Dynamic Adapter

- Weights to be loaded from storage just-in-time as new request received

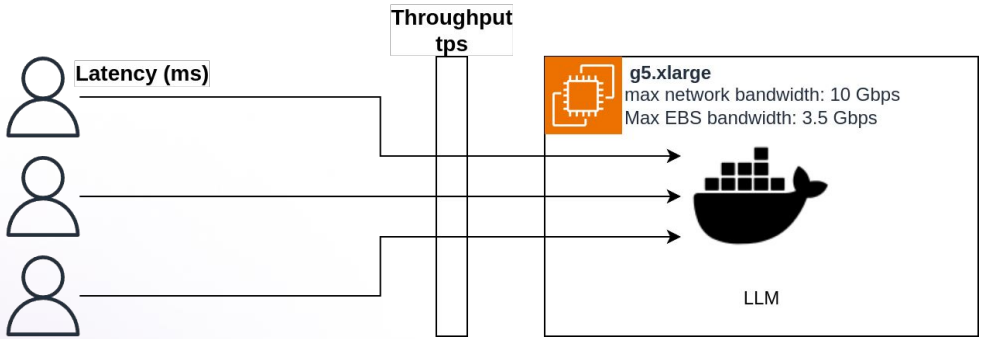
Tiered Weight Caching

- At runtime, offloading of adapter weights to CPU and disk to avoid out-of-memory errors

Continuous Multi-Adapter Batching

- To support batching of different request we will also need different adapters to do inference together
- Mask created and applied at runtime to hide adapter outputs
- Ensures each batch request gets the right adapter

Understanding LLM inference



- Models take tokens as input
- Latency vs throughput vs bandwidth
- Throughput depends on the number of concurrent users and is measured in tokens per second (tps)
- Higher the concurrent users higher the latency.

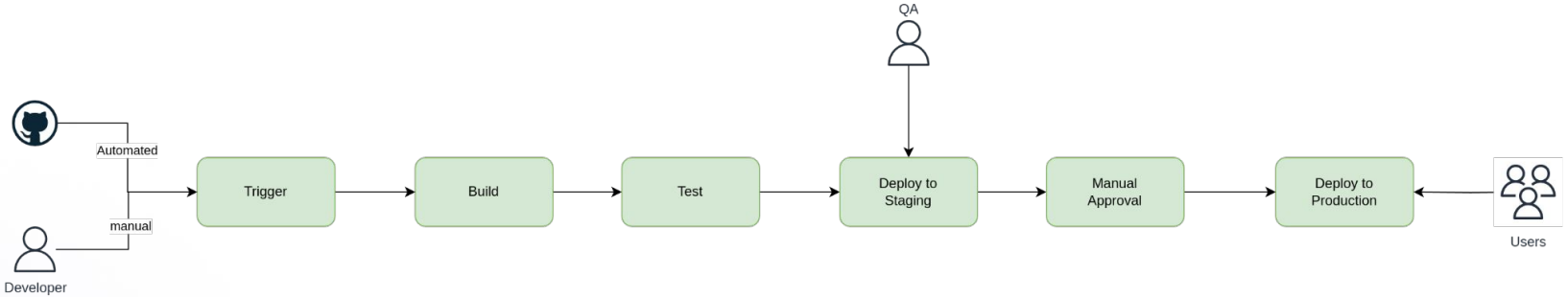
LLM inference (TTFT vs TPS):

- Prefill phase
  - compute heavy
  - The wait time we see in UI (lower TTFT)
- Decode phase
  - memory transfer overhead
  - Higher batch size is better (high TPS)

LLM inference memory

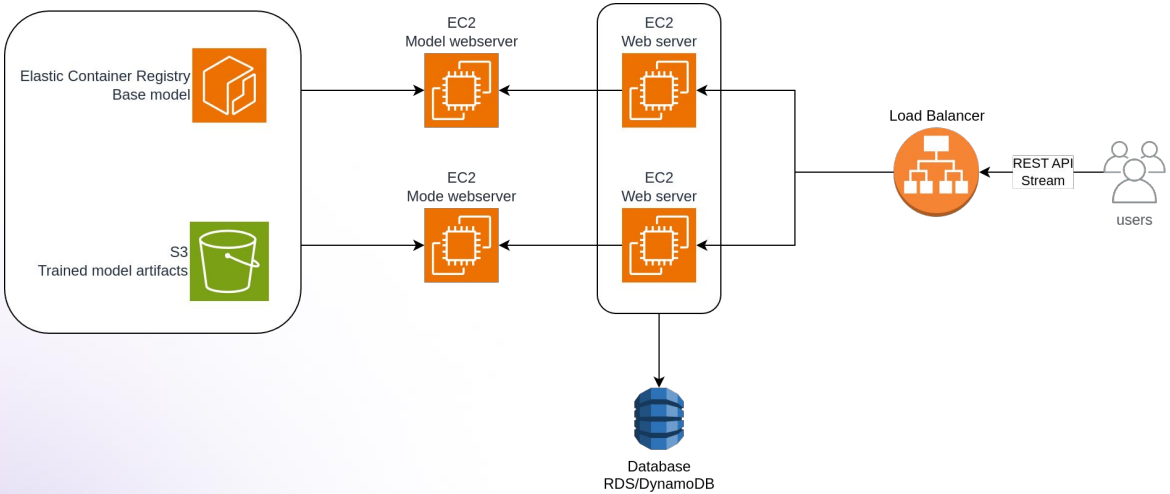
- Model weights:
  - parameter weights
  - Can be quantized
- KV cache size:
  - Depends on batch size
  - Few GBs in size

Continuous Deployment



- Code changes to an application are released automatically into the production environment
- Generally there are 3 stages
  - Development: between multiple teams
  - Staging/UAT: simulate an internal prod environment
  - Production: open to actual users
- Application follows versioning system like semantic versioning.
- Semantic versioning consists of major, minor and patch version. For eg: 3.2.1
- Major version can lead to breaking changes whereas, updates are put in minor changes. Security and bug fixes are usually added in patch version.

LLM as a service



- General use case: REST APIs with support for streaming
- Models are mostly on a GPU backed VM
- Load balancer evenly distributes load to all web servers and automatically adds them as they scale
- Advantages of having a middleware service:
  - LLM frameworks don’t have web-app security built-in
  - Separately scale model vs main application
  - Control client facing API structure
  - Integrate with tools like databases and vector stores
  - Log metrics and monitor with tools like Langfuse

Deployment Types: APIs

Ollama

- Local testing of open models
- Quick setup
- Interact with CLI and REST APIs

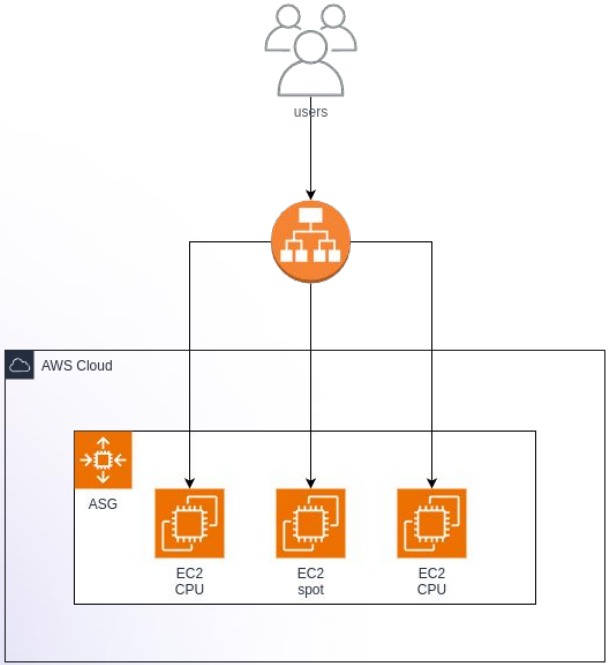
Third Party Providers

- Pay per use model
- Data boundary not defined
- Possibility to fine-tune models

AWS Bedrock

- Pay per use foundation models
- Provisioned throughput options for high usage
- Possibility to train foundation models
- Region specific service for data boundary

Bare Metal servers



Server based deployments

- Use custom VM images having webserver installed
- startup scripts to install packages and run the application
- Automatic scaling based on metrics like CPU and memory
- Spot instances
  - typically 70-90% cheaper
  - Stateless application and Client side retry needed
- Mixed Instance type also possible to configure consisting of on-demand and spot instances
- Paradigm exists in GCP and Azure as well
- Easy to setup and migrate to other clouds
- CPU vs GPU vs TPU vs LPU

Container Deployment Solutions

Elastic Beanstalk

- Platform as a Service
- Easy learning curve
- automatically handles:
  - capacity provisioning
  - load balancing
  - Scaling
  - application monitoring

Elastic Kubernetes Service

- Infrastructure as a Service
- Open sourced version
- Steep learning curve
- More control over infrastructure
- Supports many third plugins

Elastic Container Service

- Infrastructure as a Service
- Proprietary technology
- Steep learning curve
- More control over infrastructure

SageMaker Endpoints

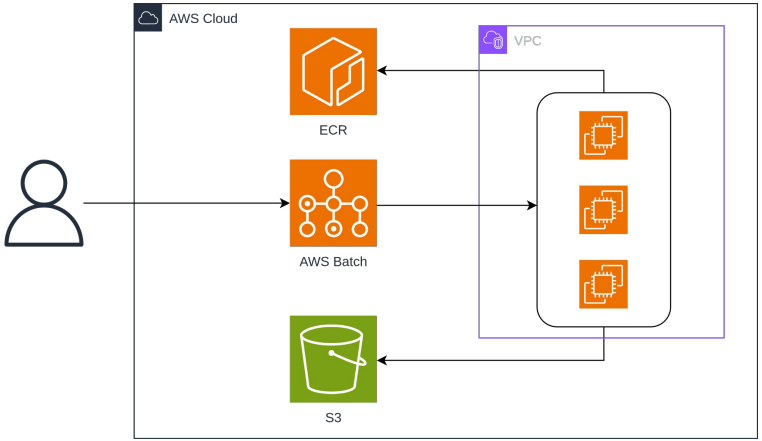
- Platform as a Service
- Easy learning curve
- Default support for multiple LLM models

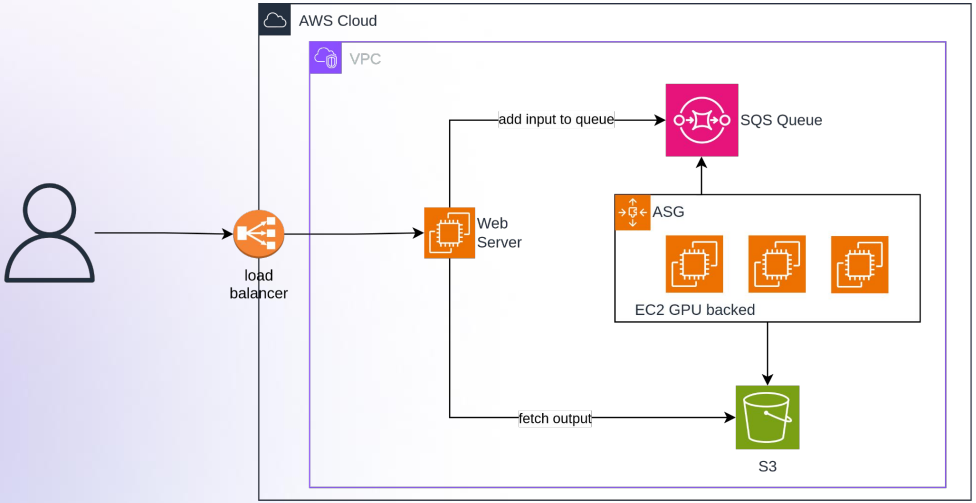
Async Processing

Analytics Vidhya | DATAHACK SUMMIT

AWS Batch

- Launch containers using EC2, Fargate or EKS
- Scaling handled by AWS batch
- Resources removed after task completion
- Support for trigger events





Asynchronous APIs

- Send request to queue for processing
- Use a queue to decouple requests
- Separately scale model EC2 instances
- Client side code retires to check for output

Analytics Vidhya | DATAHACK SUMMIT

Workshop

SageMaker for Continuous Deployment

Speaker

Kartik Nighania
MLOps Engineer at Typewise

SageMaker Endpoints

After we create our model we can deploy it to SageMaker using **SageMaker Endpoints**

An endpoint consists of the:

SageMaker Model
All details about a models configuration. Such as:

- Model container. Supports custom containers
- Model artifacts from S3
- IAM role permission
- Network settings

SageMaker Endpoint Configurations
Defines how a model needs to be deployed and scaled.
Types of configurations include:

- Provisioned
- Provisioned - Async Invocation
- Serverless

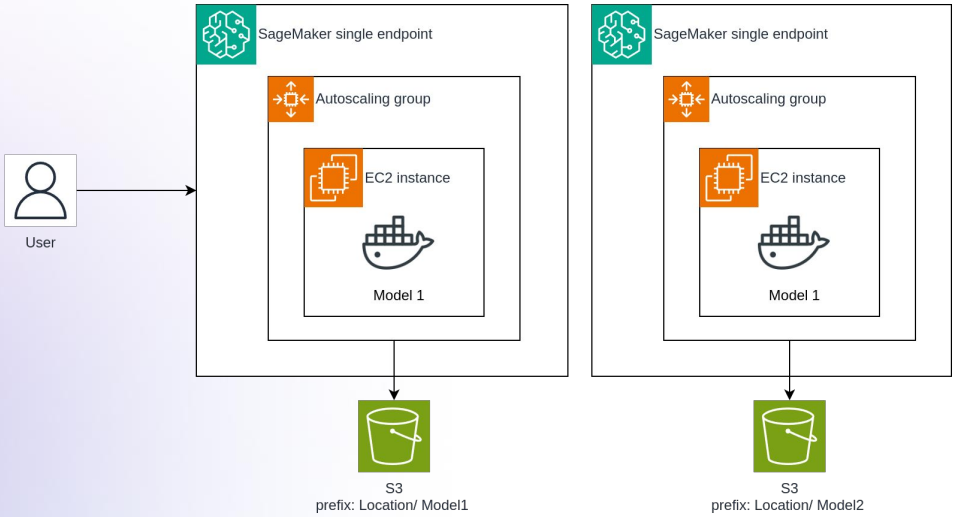
- **Model card:** share model details with teams and stakeholders
- **Data capture:** switch to enable saving request input/output in S3 for further evaluation

Real-time Inference

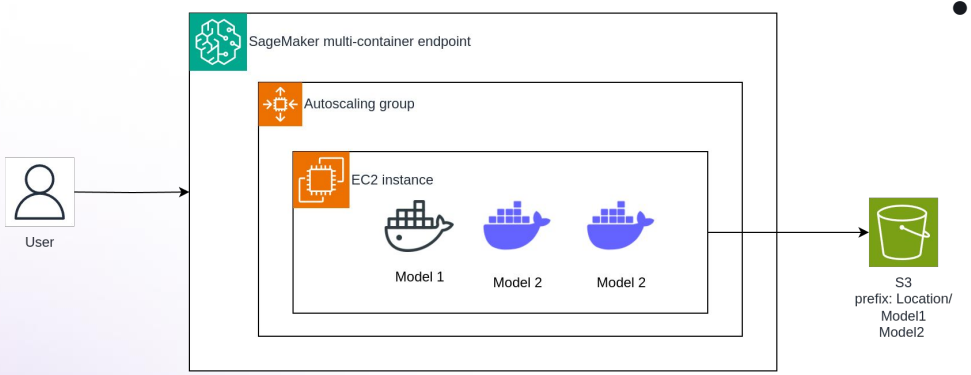
- Deploy models as web servers and keep them persistent
- In build support for autoscaling
- Useful for high concurrency traffic in real-time
- Max 6 MB request payload and 60 MB response time
- Support for both CPU and GPU instances

Hosting Types

- **Single model**: Deploy and scale a single model type on an instance

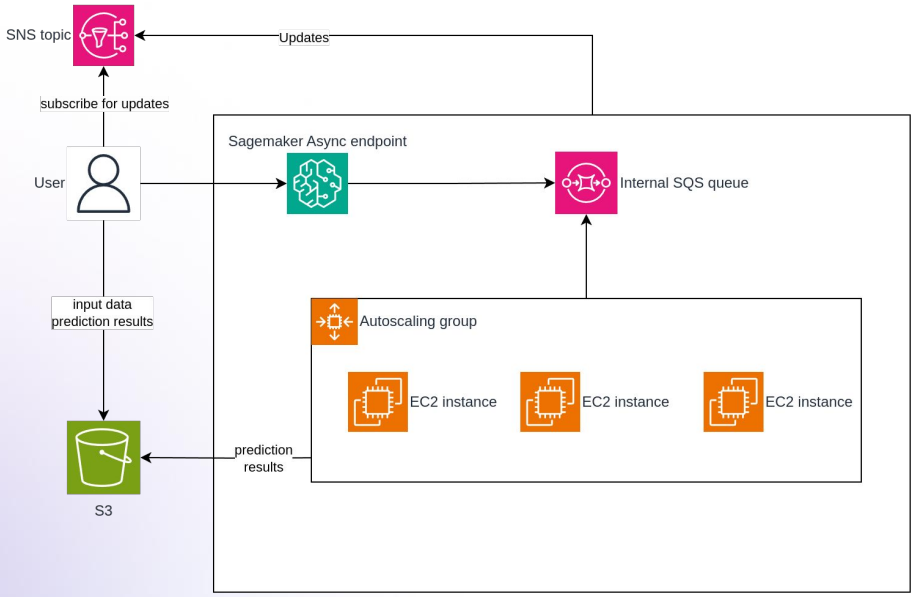


Real-time Inference



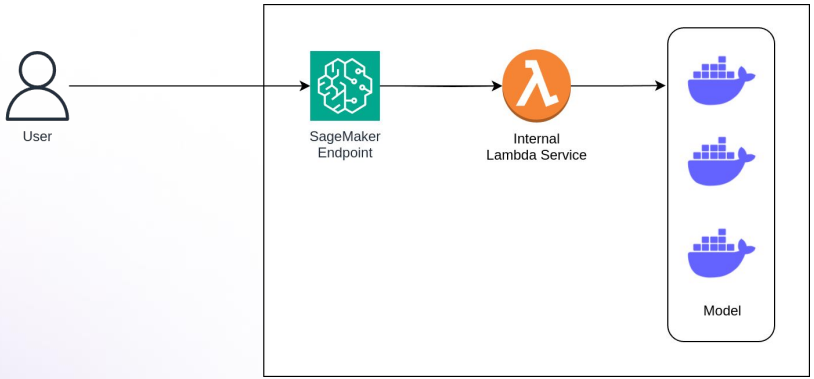
- **Multi-model** in single endpoint
  - Share instance resources between models
  - Host similar models that share the exact same container image
  - Different model artifacts are loaded from same prefix folder
  - The invocation specifies the target model to be invoked
  - SageMaker auto scales different model types based on invocation request patterns of these model
  - Efficient GPU and CPU sharing saves cost
- **Multi-model** with multi container in single endpoint
  - Host upto 15 different types of containers on a single endpoint

Asynchronous Inference



- Provision servers but call them in asynchronously
- Requests with large payload sizes up to 1GB
- long processing time upto 1 hour
- near real-time latency
- Decoupled infrastructure using queues and notification topics
- Support for both CPU and GPU instances

Serverless Inference



- Ideal for spiky traffic pattern with high concurrency limits
- Cold start time associated
- Fixed RAM values ranging from 1-6 GB maximum
- CPU scaled based on RAM
- No GPU support
- Container size of 10 GB with 5 GB additional storage
- Payload size of 4 MB with 60 seconds runtime

- Better to use custom AWS Lambda with upto 10 GB RAM with 29 second runtime

Batch Transform jobs

- To get predictions for an entire dataset
- No persistent endpoint. Servers launched based on set concurrency
- Distributed dataset feature also available
- Loads and saves output in S3
- Servers are destroyed upon completion

SageMaker Projects

- CloudFormation Template
- Code Repository
- SageMaker Pipelines
- SageMaker Experiments
- Model Groups
- Endpoints

Create end-to-end LLMOps / MLOps systems with 1 click deployment. This ensures that we can:

- fully automated using CloudFormation therefore not prone to human errors
- enforce security best practices
- setup templates based on organization needs
- Use AWS provided pre-built templates and modify based on needs
- Share across teams in an organization

Analytics Vidhya | DATAHACK SUMMIT

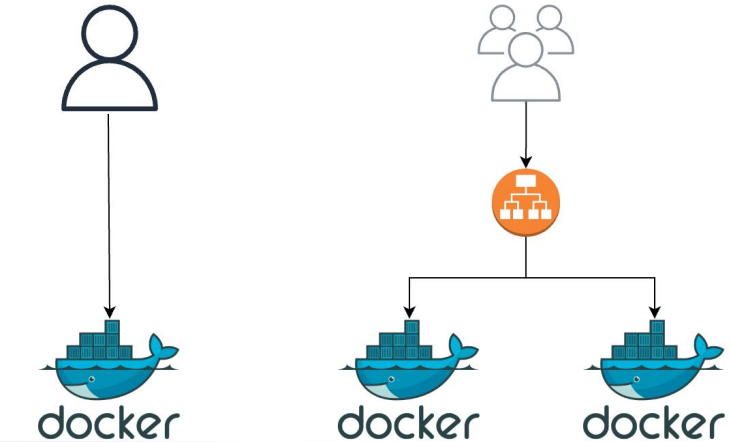
Workshop

Introduction to kubernetes

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Managing containers



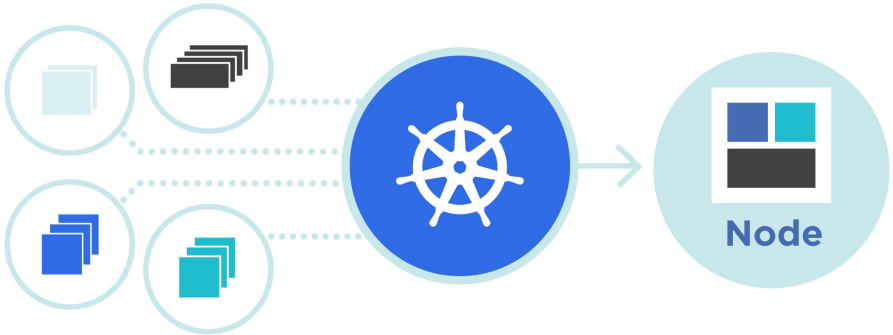
As the services grows in-most cases shift the application to multiple microservices. This leads to:

- Manually managing servers
- Health check of EC2 machines
- Back up the underlying volume

Application at scale has multiple challenges:

- Reproducible and smooth deployment
- Secrets and environment variable management
- Careful resource allocation
- Logging, monitoring and auto-heal
- Microservice communication
- Auto-scaling and auto load-balancing
- Fine-grained application access to engineers

Kubernetes

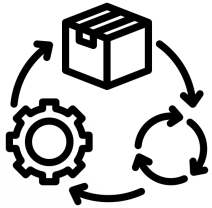


Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

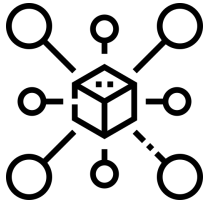
It groups containers that make up an application into logical units for easy management and discovery.

Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

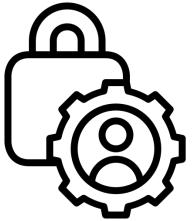
Kubernetes features



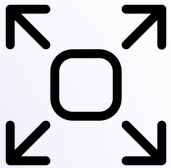
Fully manage application lifecycle



Easy microservice communication



Role based access control



Auto scaling of containers and nodes

Third party plugins

Cron Jobs

Analytics Vidhya | DATAHACK SUMMIT

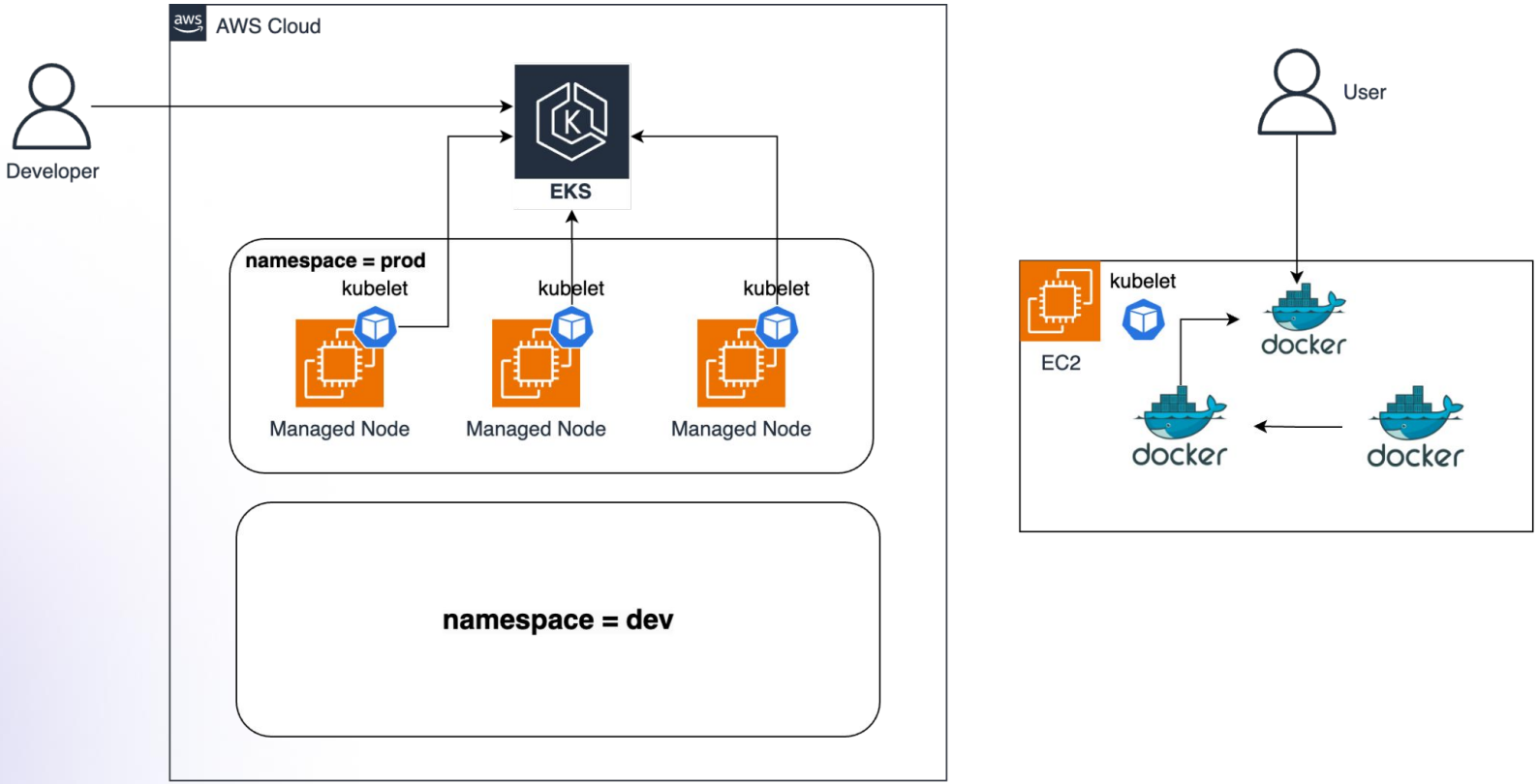
Workshop

Kubernetes Components

Speaker

Kartik Nighania
MLOps Engineer at Typewise

Managing VMs



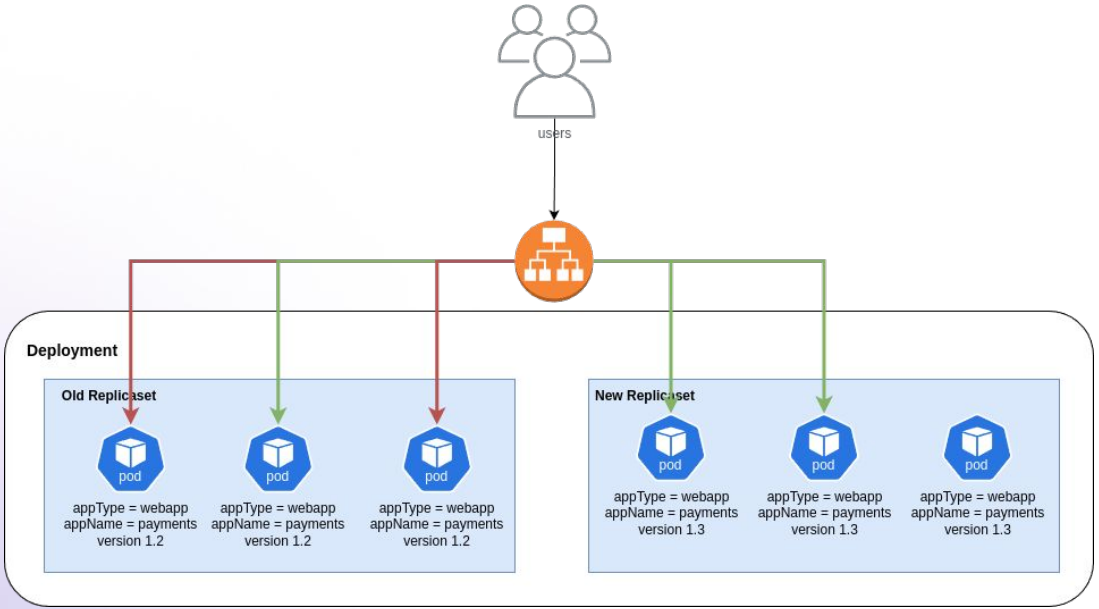
Kubernetes

appType = webapp

appName = payments

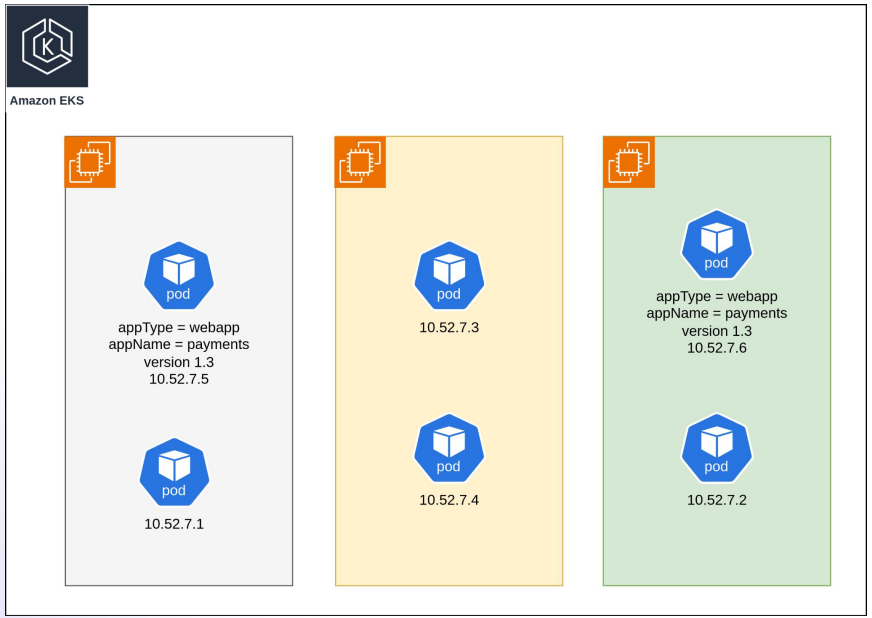
- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes
- A Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers
- Kubernetes supports many container runtimes such as docker, containerd etc
- Pods can have labels that uniquely defines pod of the same type in a sea of pods
- This can be used to filter pods and manage them
- A pod will be assigned to a node automatically by K8s based on certain conditions

Deployment



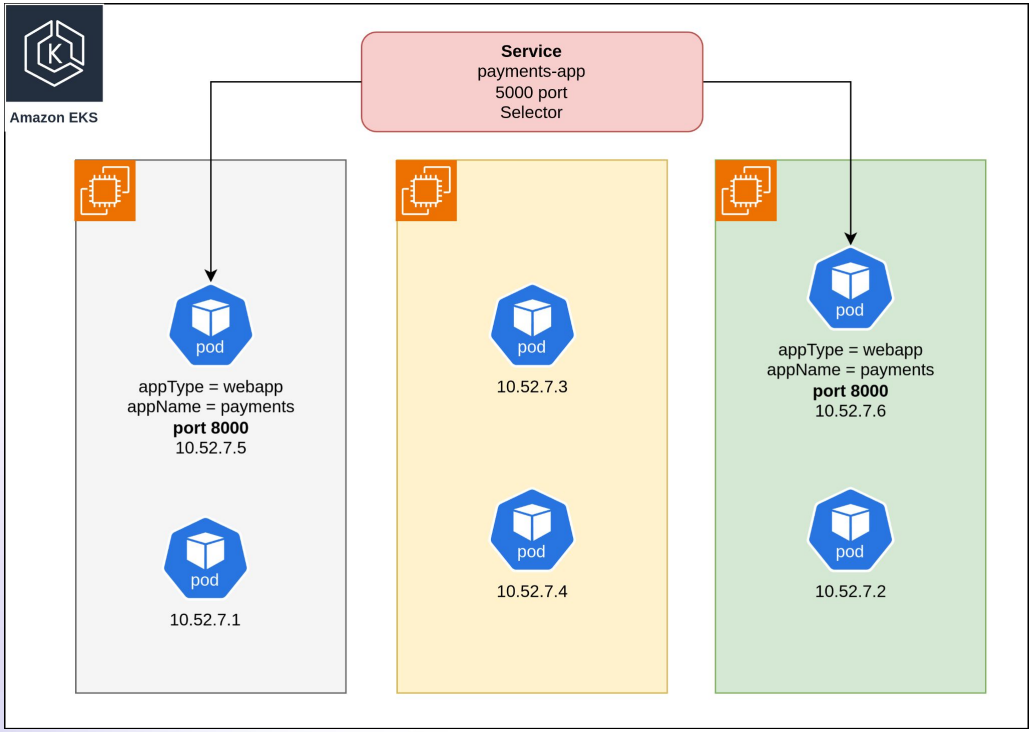
- A collection of pods is called a replicaset. This is a logical collection done using `labels` given to a pod and `matchLabels` given to a replicaset
- Health of all the pods in a given replicaset is automatically managed to the desired count
- We can scale a replicaset by a single command
- Deployment manages the lifecycle of an app by controlling its replicasets
- Updating an app version is done by creating new replicasets and deprovisioning old one. Types include `rolling` to avoid downtime and `recreate`

Networking in Kubernetes



- Each pod gets its own IP address by the internal CNI (container networking interface)
- Given our payment app how will the user connect to this app ?
- What happens if the node dies leading to pod created on another node. How will the communication happen ?

Service Type



- A Kubernetes Service is an abstraction which defines a logical set of Pods running somewhere in your cluster, that all provide the same functionality
- When created, each Service is assigned a unique IP address (also called clusterIP) which doesn't change and is added to the internal DNS
- Service uses selectors to find its pod of the same type
- Communication to the Service will be automatically load-balanced out to some pod that is a member of the Service
- In this case a microservice can connect to the payments app using `http://payments-app:5000` where 5000 is the service port which will hit target port 8000 of the pods

Analytics Vidhya | DATAHACK SUMMIT

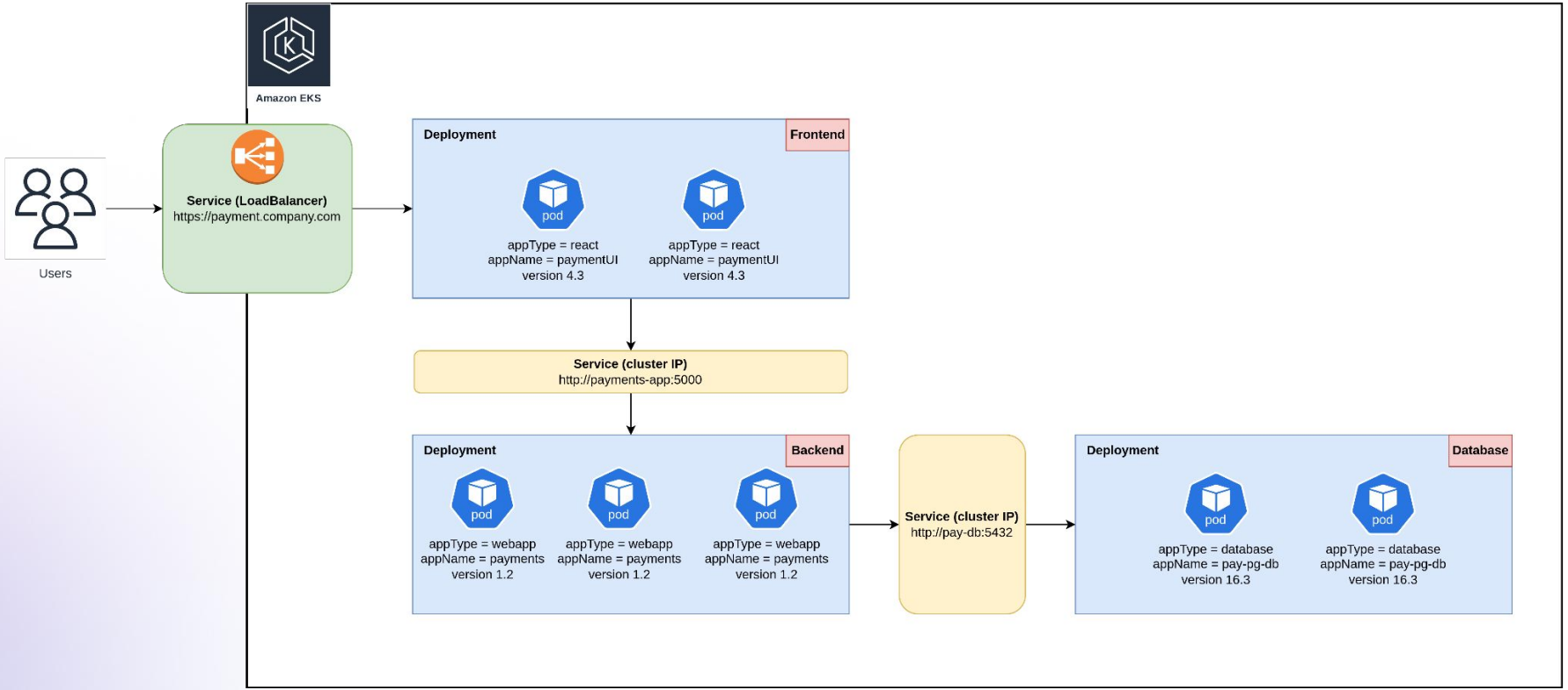
Workshop

Project Introduction

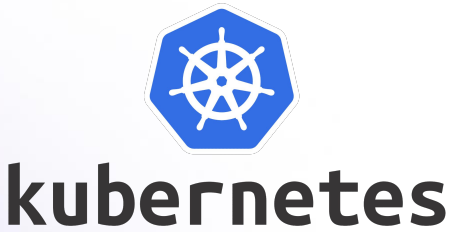
Speaker

Kartik Nighania
MLOps Engineer at Typewise

Microservices



Project Overview



Advantages of vLLM:

- State-of-the-art serving throughput
- Efficient management of attention key and value memory with Paged Attention
- Continuous batching of incoming requests
- Quantization: GPTQ, AWQ
- Optimized CUDA kernels

Create a Kubernetes cluster in EKS (Elastic Kubernetes Service)
Create a deployment in K8s with the vLLM docker container

Analytics Vidhya | DATAHACK SUMMIT

Workshop

Autoscaling in Kubernetes

Speaker

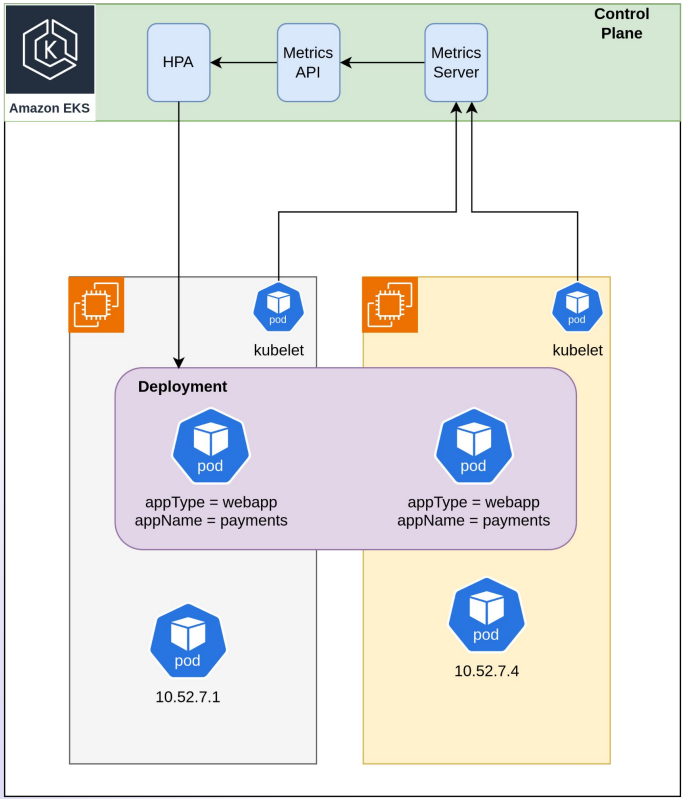
Kartik Nighania
MLOps Engineer at Typewise

Autoscaling

Challenges in scaling:

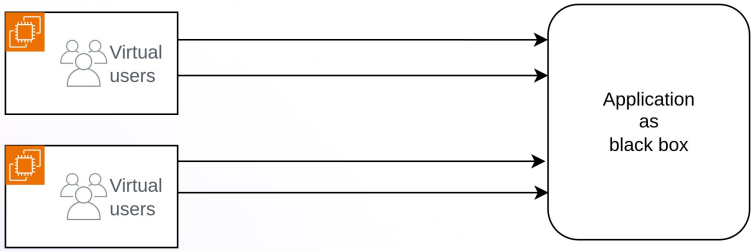
- Automatically scale containers based on certain metric like CPU, memory, RPS (request per second)
- New containers to be automatically created and assigned to nodes
- New nodes to be automatically created with all the necessary OS and packages
- Manage nodes and containers health
- Remove nodes when not needed to save cost
- Graceful termination of containers to avoid bad user experience

HPA (Horizontal Pod Autoscaler)



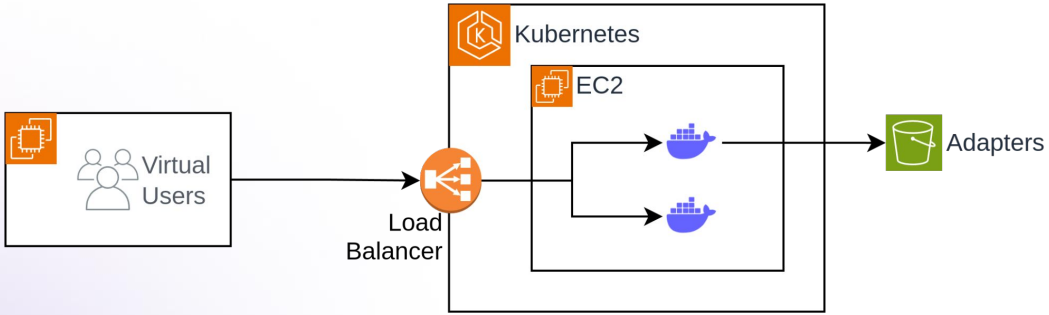
- Horizontal Pod Autoscaler automatically updates a workload resource such as a Deployment, with the aim of automatically scaling the workload to match demand
- In vertical scaling would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload
- There is also a cluster autoscaler that scales nodes based on the usage metrics

Load Testing



- Black box tests focus on testing the system's behavior and features and not how it works internally
- Load testing is a subset of performance testing that generally looks for how a system responds to normal and peak usage
- We are looking for slow response times, errors, crashes, and other issues to determine how many users and transactions the system can accommodate before performance suffers
- We simulate real-user behaviour using code also known as virtual users
- These requests are then bombarded to the application and critical metrics are measured

Project Overview



- Use Locust for load testing
- Use multi-LoRA serving to have multiple adapters in the same container

- Create virtual users and test bench
- Gather key metrics using load testing

Analytics Vidhya | DATAHACK SUMMIT

Workshop

Module Recap

Speaker

Kartik Nighania
MLOps Engineer at Typewise

- Module 1: Overview of LLMOps
- Module 2: AWS SageMaker for Data Scientists
- Module 3: LangChain for Continuous Integration
- Module 4: Langfuse for CICD
- Module 5: SageMaker Pipelines
- Module 6: Continuous Deployment
- Module 7: Kubernetes for Continuous Deployment
- Module 8: Real world testing

Thank You