Day 1

Fundamentals of AI Assisted DevOps

* Traditional AI

Traditional AI relies on structured data, pre-defined rules, and predictive models trained on historical data. It excels at classifications, forecasting, and onalmoly detection.

The primary use case : to predict events.

Eg: to predict climate change
"Basically those climate application are fed with historical data" provided as input.

These models are trained with historical data

$$\begin{tikzcd} \text{Data} & \\ \text{1970, 27 March} \arrow[r, bend left] & \text{AI app} \arrow[dd] \\ \text{1971, 28 march} & \\ \text{1973, 27 march} & \text{"Predict Temp"} \\ \text{- - - -} & \end{tikzcd}$$

Primary use case of traditional AI is to predict based on input "hist. data" that it is trained with.

Eg 2: Incident detection and Prediction
- use case: Predicting system failures before they occur.
- How it works:
  - use log based anomaly detection

pattern recognition (Eg: time series forecasting)

- If CPU usage suddenly spikes beyond a threshold, AI predicts potential issue.

- The system alerts devops teams to take prevention actions.

→ Limitations

- work only on pre-trained scenarios.
- cannot generate insights based on beyond structured input data.

---

* Gen AI "Generative AI"

as name suggests : helps to Generate Content.

$$\begin{tikzcd} \text{Generate} \arrow[r] \arrow[dr] \arrow[ddr] & \text{Text} \\ & \text{Images : Completely new..} \\ & \text{Videos} \end{tikzcd}$$

→ we give prompt to Gen AI.

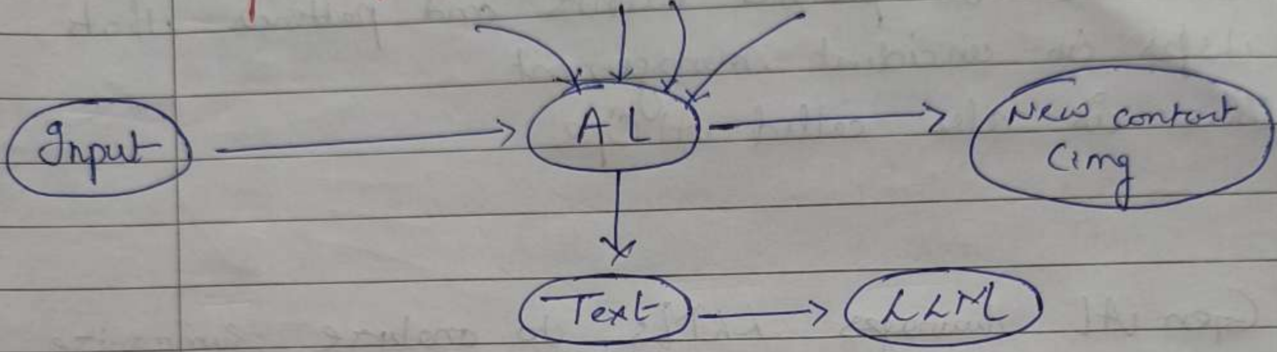
→ Gen AI is trained with huge amount of data from various sources.

→ Generate complete new image thats why called Gen AI.

→ Similarly it can generate new text also.

Text generating AI models are called LLM's

Large Language Models "LLM's" are part of Gen AI.



Difference between Gen AI and Traditional AI

Traditional AI	Gen AI
Primary use case is to "predict"	Primary use case is to "generate new text, imgs or videos"

Use case of traditional and Gen AI for DevOps Engineer.

① Traditional AI for DevOps
is to predict future events, identify patterns related to incident management.

Major challenge of S/w industry is to identify the events related to incident management or observability and report back to dev team.

As a human, we can see the only current metrics eg Cpu, RAM, log of System. But we can't predict the future state of System.

Primary use case of traditional AI for DevOps on SRE is to predict events and patterns thats helps in incident management
This is also called AIOps

Gen AI leverages LLMs to analyze, summarize, and even generate new content dynamically.

Example: AI Powered Incident Resolution & RCA
- use case: Automating root cause analysis (RCA) & remediation
- How it works:
  - Understanding logs and metrics: Gen AI processes unstructured log data, summarizes key issues, and suggests fixes.

  - Chat-based troubleshooting: Devops can ask Gen AI:
    "Why did my K8S Pod crash"? → AI analyzes logs and suggests probable causes like OOM (Out of Memory) errors.

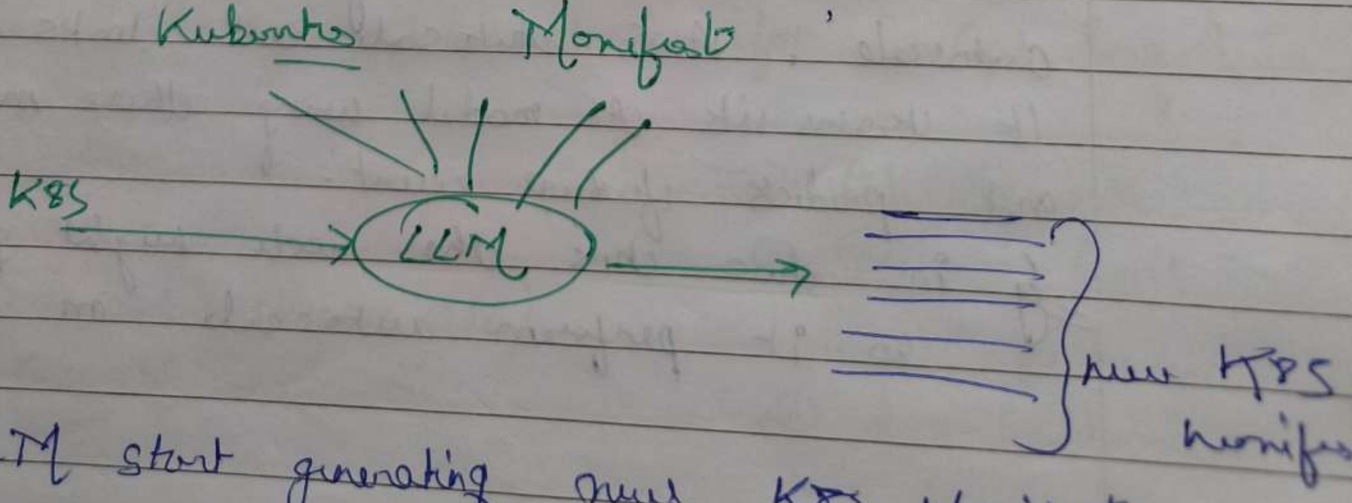
  - Auto-remediation: AI suggests and applies fixes (eg: increase ↑ memory limits in a YML file)

• Advantages :
  ◦ No need for extensive labeled training data.
  ◦ Can generate human-like explanations & solutions.
  - Adaptable to new/unseen failure patterns.

So in traditional AI in devops is to perform log analysis, Perform event management, can look at event and identify patterns and report any incident that can occur in future.

So in Gen AI for DevOps.
eg need to gen. K8S manifest . GPT4
So GPT 4 - AI Assitant in Chatgpt (LLM)
or Open AI models LAMA 3 or deepseek.

These are LLMs
LLMs are fed with huge amount of data.
eg So K8S is also provided as training material to these LLM

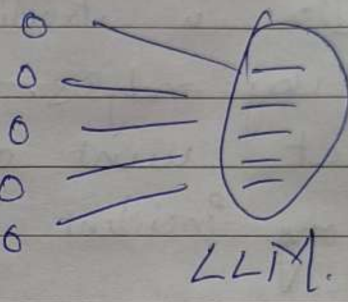


LLM start generating new K8S Manifest
Doesn't copy. generate acc. to user.

Understand

LLM ⟶ LLAMA 3 : trained with 407 B parameters
↓
with help of Super Computers.

huge amt of data can be processed with the ~~tt~~ Super Computers.



LLM are similar to human brains like neural networks. Once they are trained they don't go back to super computer.

Once LLM are trained with CPUs & GPUs, they respond back from memory.

Another ex. of trad. AI
is AWS Autoscale — they have prediction autoscale. AWS Autoscale cont. looks at metrics. It trains its AI model using these metrics and predicts future event.
Eg in two hrs the node might go down, so it perform autoscale on prediction.

LLMs part of Gen AI trained with billion of parameters with help of supercomputers.
Don't go back to source, they send data from memory.

Q What are the sources to train?
Eg Open AI initial trained with 80% of data from common crawl.
And secrets sources too.

# What is Prompt Engineering?
To comm^n with LLMs.

$$\begin{tikzcd} \text{Prompt} \arrow[r] & \text{LLM} \arrow[r, bend left] & \text{efficient output} \end{tikzcd}$$

efficient prompt

Medium to comm^n with LLM.

# AI landscape (AI tools for DevOps eng.)

(1) → AI Chatbox
① Claude
② llama (locally on m/c)
③ deepseek

(2) → AI Agents
① Github Copilot workspace
② ~~new~~ bold.new

③ → AI Assistants
① Github Copilot Workspace] - free
② pieces for developers.
③ Cursor. ai (paid) "less free token"

④ → Programming lang for scripting
① Python with ~~flask~~ API, Django

python with fast- API

Before moving to practical
Q LLMs?
An LLM is an advanced AI system trained on vast amount of text data to understand, generate, and process human language.
These models use deep learning techniques, particularly transformers (like GPT, BERT or llaMA), to recognize patterns, predict words ~~or~~ and generate human like responses.

⇒ Day 2 Prompt Engineering.

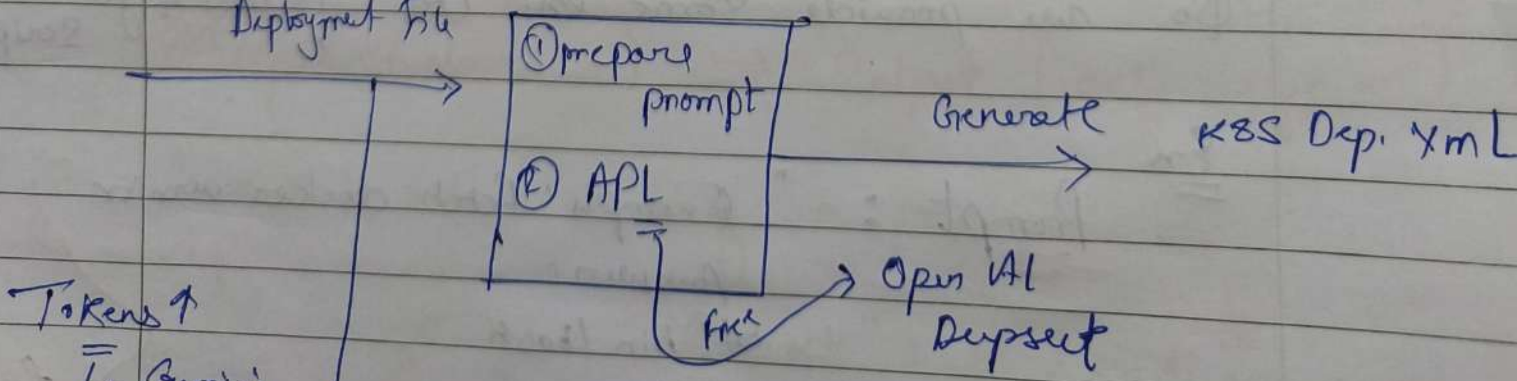
① prepare prompt
② API call to LLM
 └ Open AI
 └ Deepseek

→ Do write specific prompt which gives you specific result. Avoid unnecessary prompt to avoid charges.

→ More no. of token ↑ More API Requests.
 ↓
 are related to words.

Eg in Gemini for every 100 words - 50-60 token.

diff AI model chare different pricing.
(use prompts wisely). → for Cost Opt $.



Prompt Concepts

① Zero Shot Prompting (Direct Prompting)
we give prompt or gen. without a example

ex
we have generated a K8S manifest, or models are already trained with. So we don't require a example.

useful when we are working with popular and familiar concepts.

② Few shot Prompting (provide some example)
we give some example, and then prompt.

Eg in our org, we have standard to follow while writing shell script, The LLMS don't know our standard of org.

So we provide some ex. to follow & gen. a script.

Ex
Prompt : " Example: Fetch docker version
Answer:
#! bin/bash
####
#Author: Amit
#version: V1
"
- - - .. //prompt

— format of shell

few Shot Prompting: most recommended approach.

(3) Multi Shot Prompting:
similar to few shot prompting, but it
gives more examples with prompt.

(4) COT "Chain Of Thoughts"
is a prompting, enhance the performance of
large long models. (LLMs)

COT encourages LLMs to use reasoning
capabilities of LLMs.

Derive best output, with COT.

—— × ——

Note Its Tip try to write more elaborated input
f output provided by LLM is conciseD.

* Input (clear ✓) → * Output (only reqr ✓)

* * Example: when writing a prompt.

(1) Always give "Context": Eg I am devops eng, working on this project
(2) "Instruction"
(3) "Example" (few shot prompt) → better perf.
(4) "Output format" Eg in json, md format