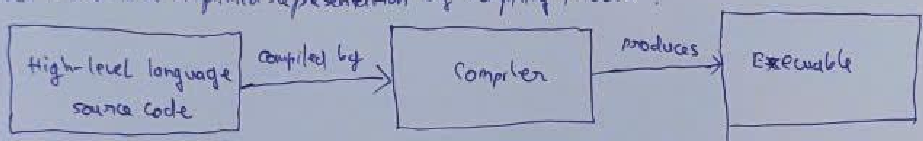
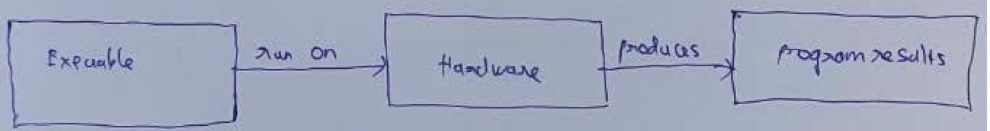
THEORY

1/ Explain the different between interpreters and compilers. Provide examples of programming languages that use interpreter and compiler. Discuss scenarios where one is preferred over the other

- Both interpreters and compilers must translate high-level language into machine language before they can be run

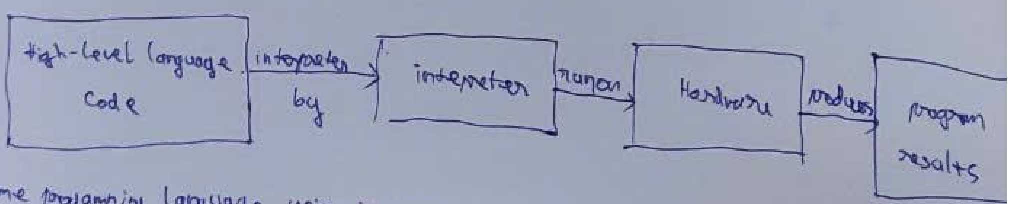
- About Compilers: is a program that read source code (typically written in a high-level language) and translate it into some other language (typically a low-level language (like assembly)
These machine language files are combined into an executable file (containing machine language instructions) like .exe, .dll that can be run or distributed to other. These executable files don't require the compiler to be installed.
Ex: here is a simplified representation of compiling process:





Some programming language using compiler: C/C++, Rust, GO ....

- About interpreters: is a program that directly execute instruction in the source code without requiring them to be compiled into an executable first.
Here is simplified representation of the interpretation process:



Some programming language using interpreter: python, Javascript, Ruby ...

Feature	Compiler	Interpreter
speed	Faster after Compilation	slower due to runtime over head
Error Detection	Detected before execution	Detected at runtime
security	Source Code is hidden	Script is exposed

scenario where a Compiler is preferred: High performance is Critical, Security

scenario where a Interpreter is preferred: Rapid Development and Testing, multiple platform

2. Different between Just-in-Time (JIT) compiler and traditional Compiler. provide examples of JIT implementation and their use case.

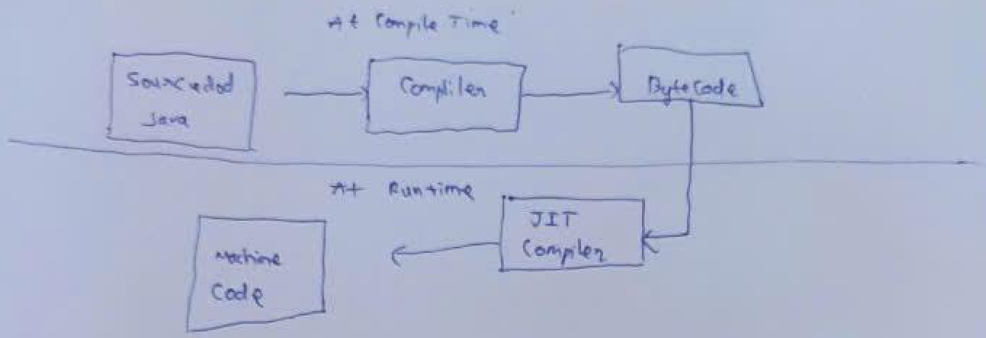
- ~~A JIT Compiler Combine both traditional compiler and interpreter~~

- Key different

Feature	traditional compiler	JIT Compiler
Compilation time	before program Execution (static)	During program Execution (dynamic)
- Dependency on runtime environment	require a runtime environment (eg JVM for Java, .NET CLR) to execute	dont requi a runtime enviroment

Example implement and their Use Cases

In Java programming Language have got a JVM (Java Virtual Machine)



3. Define CI/CD and its importance in DevOps. Explain the stages of a CI/CD. List tool commonly used in CI/CD and compare their features. Discuss common bottlenecks in CI/CD pipeline and strategies to resolve them.

- CI/CD: Definition and Importance in Devops

CI/CD (Continuous Integration/Continuous Deployment) is a methodology in Devops that automates the software development lifecycle to ensure software are built, tested and deployed efficiently and reliably.

Ex: CI: A project have a main source code. When developers develop new feature and integrate into the main source code, CI triggering automated build and test to detect issue (unit test, error syntax)

CD: Have a lot environment to develop software (develop, stagging, product). CD by automating the deployment via environments finally to realase to product environment after passing automated tests.

- Importance in Devops:

Faster Development Cycles: Automate repetitive task and speed up develop

improved quality: Catched bug early with automated testing

increased reliability: Reduces human error with consistent, automated process

Stages of a CI/CD pipline (8 Stages)

1 Plan

- Identify requirement, design solution, and ensure Collaboration across team from the St
- Ensure the proposed changes integrate smoothly with the existing system

2. Code

- write code according to the plan
- use version control (eg Git) to maintain Consistency betwen developers

3. Build

- Merge code into a main repo and run automated tests (eg unit test, integrat test,

4. Test

- Deploy the build to test environment, performing automated and manual test

5. ~~Relea~~ Release

- prepare tested build for release

6. Deploy

- Deploy the application to the production environment

7. Operate

- Monitor and maintain the application and infrastructure to ensure Smooth ope
- Implement auto-scaling, user behavior tracking, and Feedback collections mechani

8. Monitor

- Set up monitoring tools to performance bottleneck and application issues
-

- Common bottlenecks in CI/CD pipeline and strategies to resolve them.

1. Slow build time

Cause: Uptimized Code, large repo, Inefficient build process, Cache dependences

Resolve: parallelize build, modularize code base

2. Flaky or slow test

Cause: Unstable test suites or test with high execution time

Resolve: Use test prioritization, run test Critical First

4. Define Cloud Computing and its key characteristics (e.g scalability, elasticity) Compare Iaas, Paas, Saas with example. Discuss the benefits of using cloud computing in Devops

- Cloud Computing: "The Cloud" refer to server that are accessed over the internet and software and database that run on those servers. Cloud servers are allocated in data center all over the world. By using cloud computing users and companies do not have to manage physical server themselves or run software application on their own machines.

- Characteristic of cloud computing:
  + Scalability: The ability to handle increasing workload by adding resources as needed
  + Elasticity: The capability to automatically increase or descrease dynamically based on real-time demand
  + On-Demand Self-Service: user can provision computing resourse as need without requireig any human administrators
  + Security: Cloud provider invest heavily in security measure to protect their user's data and ensure the privacy of sensitive information

- Compare Iaas, Paas, Saas

	Infrostructure as a Service (Iaas)	Platform as a Service (Paas)	Software as a Service (Saas)
Definition	provide virtualized ~~the~~ computing resource over the internet	provide platform to develop run, and manage application without handle infrastructure ~~(server, storage)~~	Instead of installing app on their device Saas app are host on Cloud Server, user can access over Internet
Control level	OS, middleware runtime	Application, data	Software interface
use case	Deploying and managing virtual machine, storage, network	Application development and testing envinonment	Access productivity too Like email
Example	AWS EC2, Microsoft Azure, virtual maching	Heroku, Google app engine	Gmail, salesforce Microsoft 365

the part of observability

key different

IaaS: provides raw computing resource for maximum flexibility (e.g. setting up virtual machine, custom software

PaaS: simplified development by abstracting Infrastructure management (e.g. deploy web app without managing server)

SaaS: offer ready-to-use app for user (eg email, collab tools)

- Benefit of cloud computing in Devops
+ Faster Development and deployment
+ Cost Efficiency : Pay-as-you-go (scaling resource as needed)
+ High Availability and Disaster Recovery :
+ Automation: Cloud provider support infrastructure as code (IaC), enabling automated scaling, provisioning, and configuration of environments.

5. Compare Docker Containers and virtual machines in term of architecture performance and use case. list advantages of using Docker over VM in DevOp work flow

	Docker Container	Virtual Machine (VM)
Architecture	Containers share the host OS kernel	VM run full OS, each with own kernel
Size	Small (dependencie packaged) eg 100GB can run 1000 containers	large eg 100 GB can run 5 VMs
Resource Usage	Share resources efficiently	Consume more resource
Performance	nearly same performance with OS due shared OS	Slightly slower due hardware virtualization
Use Case	Ideal for microservices, CI/CD Pipeline, Container orchestration	Suitable for running multiple OS instances on legacy systems.

- Advantag using Docker Containner over VM in DevOp work flow

Fast deployment and scalability:

+ Enabling rapid testing and deployment in CI/CD pipline
+ Horizontal scaling is easy with container orchestration tool like kubernets

Seamless integration with DevOp tool

+ Docker integrates with CI/CD tools (e.g Jenkins, Github Action)
Containe orchestration (like kubernet) to automate work flow.

Resource Efficient

+ Docker Container light weight

Enhanced Portability

+ Containers encapsulate application and dependencies, making them platform-agnost and easy to move across diff environment.

6. Define IaC and It benefit: Explain the Role of Terraform in automating Infrastructure. Provide an example use case of Terraform in a DevOp project

- IaC (Infrastructure as Code): is the practice of managing and provisioning Computing Infrastructure (e.g server, network, storage) using code rather than manual process
IaC defining infrastructure configuration is a declarative or imperative language applying them consistently across environments

- Benefit of IaC:

+ Automation and Speed: Eliminates manual Configuration, enabling Faster provisioni deployment
+ Consistenty
+ Cost-effecient: Reduces human error, minimize dowsize, optimize resource usage
+ Improved Collab: Allow team work on infrastructure like Software Code, better Collab between developer and operation team

mainx lifcycle

Role of terraform in Automating Infrastructure

- It use a declarative approach to define resource and support multiple cloud service providers, such as AWS, Azure, Google Cloud

+ Multi-cloud support: managing resource across diff cloud platform with single configuration language

+ State management: Mantain a state file to track resources

+ Dependence Management: Automatically determine resource dependencies and apply change in the correct order

Example use case Terraform in Devop:

scenario: Deploy a scalable web app on AWS

We use Terraform to determine resource for webapp to automate deployment with high availability and scalability on AWS

use terraform to define resource like:

+ Amazon instan EC2 for web app

+ An Elastic load Balance (ELB) to distribut traffic.

+ An Auto Scaling Group (ASG) to handle varying traffic load.

+ Security Group for access control

+ S3 bucket for static file storage

7. Define observability and its importance in a production environment

Explain the three pillars of observability: logs, mestrics and traces

list tool commonly used

How to intergrate into Devop lifecycle

- Observability: is the process of making a system's internal state more transparent system are made observable by the data they produce, which in turn help you to determine If your infrastructure or application is heathy and functioning normally.

Logs: are time-stamped records of event that happen over time, such as error log files. Logs help you understand the behavior of infrastructure and application as well as of user and business.

logs often contain the root cause of a failure or issue.

You can be use logs to answer these question:

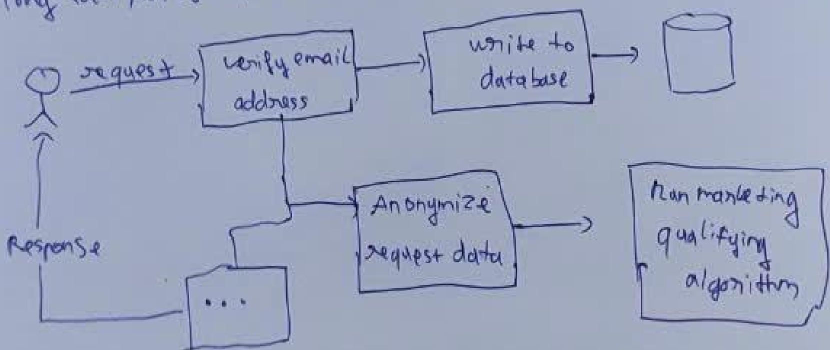
+ How many request are processed per second?
+ what percentage of requests are failing?
+ How many distinct users are visiting the site per day?

Metric: is a ~~mu~~ numeric measurement that is timestamped to indicate when it was collected. If you track an application or system

~~Metrics~~
Metric maybe measure:

+ How many system resource are currently being used, such as memory or storage
+ How many user are accessing the application right now?

Traces: provide end-to-end insight because It track a system request as it travel through multiple location/component of a system that is distributed or based on microservices. Traces are made up of spans, Spans record how long each part of request takes, such as this simple request shown.



The following Diagram of System request

Traces make it easier to understand root cause and locate which part of the system are slow or having other problems

How to Integrat of observability int the DevOps lifcycle

+ Development phase: integrate observability tools to debug and optimizing code during development

+ Testing phase: Moniton during automated test to detect issue early

+ Deployment phase: validate by analyzing system logs, metrics, traces to ensure Stability

+ Production phase: Moniton live system for anomalies, latency, or Failures and respond proactively.