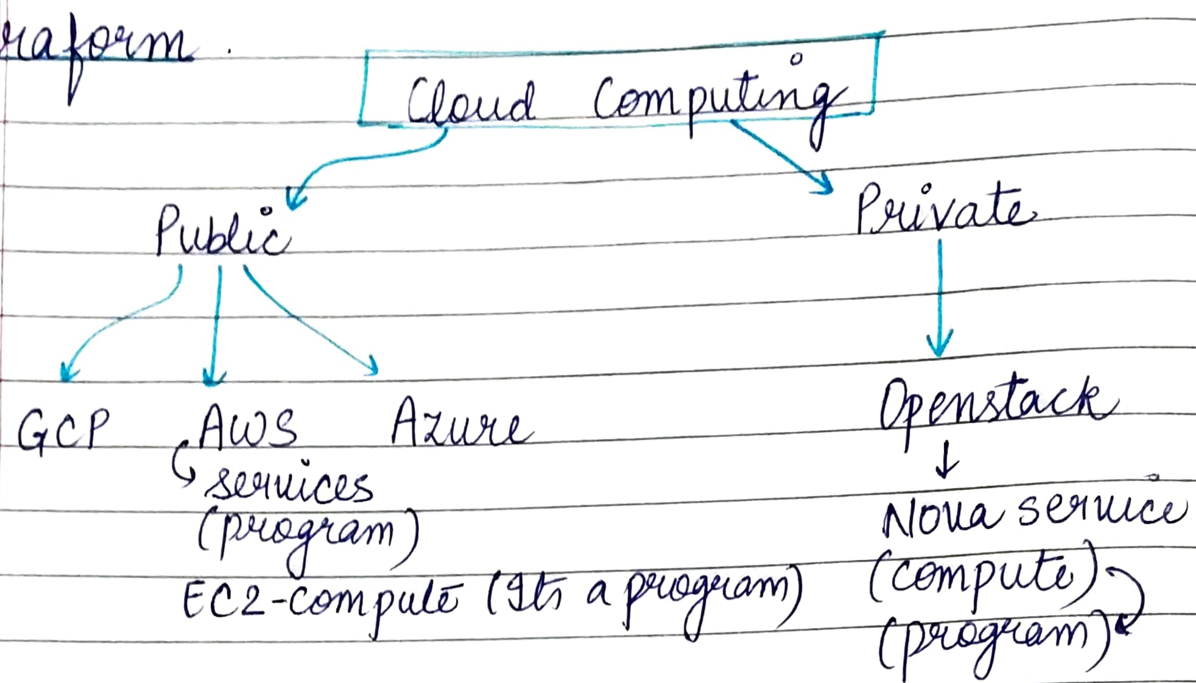
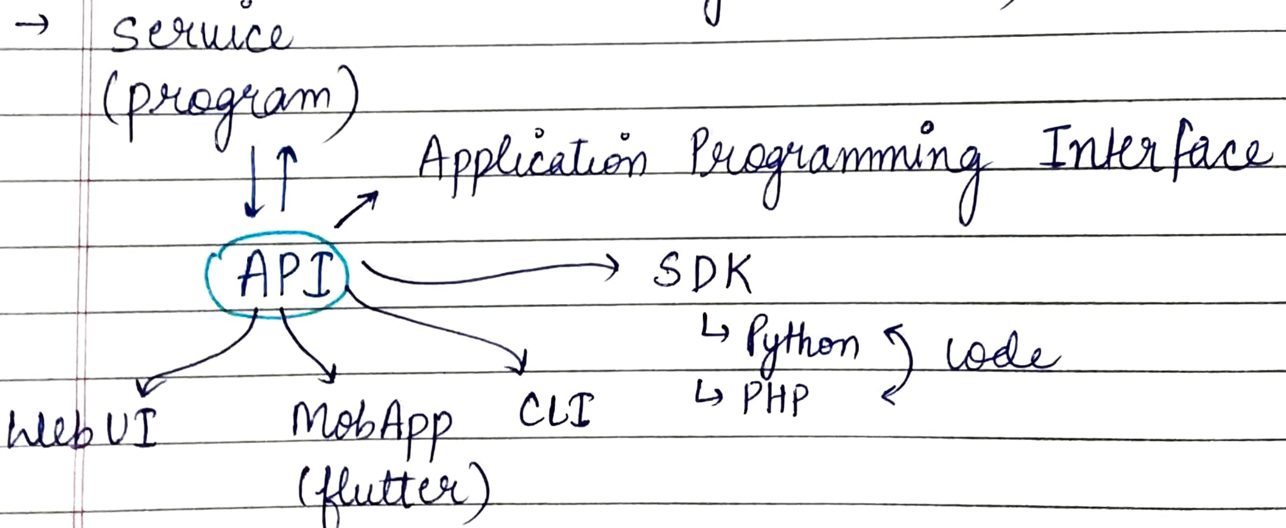
DAY-8

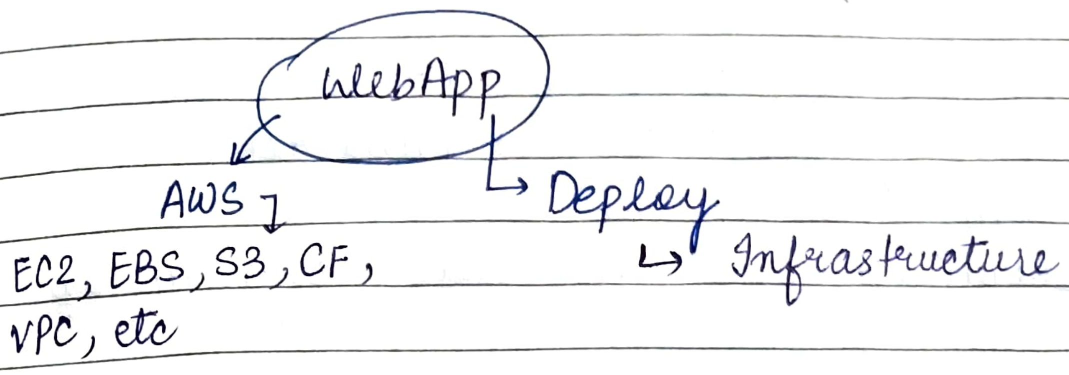
* Terraform.



→ Service is a kind of program.
Eg: EC2 (compute service, AWS)
EBS (block storage service)



▷ API (Application Programming Interface)
→ To work with any cloud service provider
Eg: AWS, GCP, etc
→ Interact with AWS and services
→ To connect to AWS or any other cloud service provider.



→ To launch one project, many service Integration is needed to establish Infrastructure for it

→ Sometimes, we may want different services from different Cloud Service Provider (CSP's) or some part from public cloud and some part of private cloud, to deploy your project.

Eg:- We are taking some services from AWS and some from GCP or Azure or openstack.

Hybrid Cloud :- Use of more than one cloud to deploy project Infrastructure

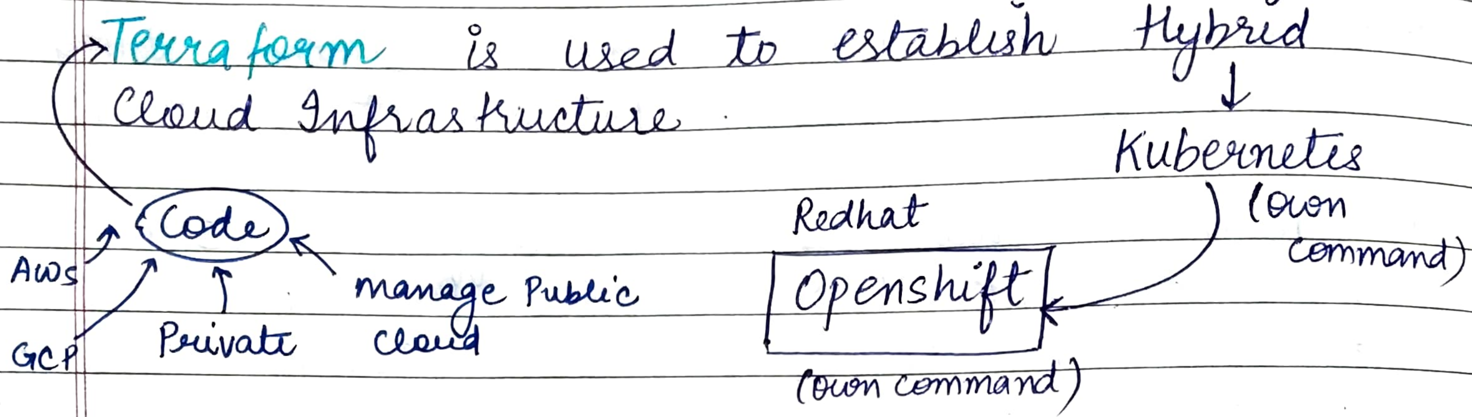
▷ Challenge's :-

→ Each Public Cloud Service provider or private CSP have different WebUI and CLI.

↳ Therefore, for a company person they have to learn every cloud and cloud program to use Hybrid Cloud Infra.
[This is a problem] ↓
Solution ↓
Terraform

(CLI)
A type of tool/program to get command over every cloud service provider (CSP's) program (use of one tool),
Standardized tool to code/program to every CSP is Terraform.

Terraform is used to establish Hybrid Cloud Infrastructure.



Terraform
↓
Its a software/tool

Hashicorp Configuration Language (HCL) → to write Code.

Kubernetes, AWS, Oracle Cloud, GCP, Azure, etc
→ providers in Terraform.

Example :-
When we create an application in enterprise we create it for client, worldwide users, before deployment we test it in our personal environment and do hit and trial
For worldwide deployment, we have to do every Infra deployment again → Solution ↓
Infrastructure as a Code

Date ______
Page ______

✱ Dont use AWS CLI or CSP WebUI to launch your Infrastructure (not recommended) / manual deployment (not recommended)

→ We should write our own code in Python (PHP) or any other language in any Infrastructure
↓
Infrastructure as a Code

We have different SDK for python for Infrastructure deployment or Infrastructure as a Code.

▷ Terraform → Infrastructure as a code.
Terraform
↳ We do everything writing a code
↓
Provide standardization (use of one language to work on every Infrastructure)

Terraform
code ← run $\Longrightarrow$ EC2 (to launch)

→ They are intelligent and store/copy the Instance ID.

→ They record instance details, they keep on tracking.

In one click, whole Infra destroys ← power of Terraform

record → In → Track
↳ 'State file'
destroy

→ For managment of cloud (all cloud in the world)

▷ Terraform → download from site.
(tool)
→ extract it somewhere.
→ Windows → environment variable →(to set path)

$$\begin{tikzcd}
\text{environment variable} \arrow[d] \\
\text{Command prompt}
\end{tikzcd}$$

$$\begin{tikzcd}
 & \boxed{\text{folder}} \arrow[dl] & \text{(workspace)} \\
\boxed{\text{file}} \arrow[d] & & \\
\boxed{\text{code}} & &
\end{tikzcd}$$

→ We have to tell terraform which Cloud we have to manage
↓
Provider in terraform

→ cd Desktop
→ mkdir tera
→ cd tera
→ dir
→ mkdir mytest
→ cd mytest.
→ notepad ec2.tf (.tf extension necessary)

ec2.tf Notepad
→ Provider aws.
access key 354c9fe426
secret key efcab343620zxyaf

region = ap-south-1

] logic for writing code.

$$\begin{tikzcd}\text{Terraform} \arrow[r] \arrow[d] & \text{Providers} \arrow[r] & \text{Example code} \\ \text{They Know our AWS API.} & & \end{tikzcd}$$

$$\begin{tikzcd}\text{Code} \arrow[r, "\text{copy}"] & \boxed{\text{SCM}} & \text{github} \arrow[l] \arrow[d] \\ & & \text{your access key and password can be visible} \arrow[ll, bend left, "\text{solution}"'] \end{tikzcd}$$

aws configure
give access key, secret key
and tell terraform about it.

OR

use AWS IAM and create multiple users
with different powers.

» aws configure --profile myprofile

↑ contains
access, secret key
and region details.

terraform code.

```
provider "aws" {
    region  = "         "
    profile = "         "
}
```

} HCL language.

→ terraform init (initialising code / folder)

• plugins made terraform intelligent
(program created by any CSP expert)

↓

Made terraform intelligent.

* [terraform init] command
for first time we create code we
write this command
↓
command will search for file with
.tf extension
↓
search for provider in .tf file
↓
download plugins for the provider they
found in the .tf file.

Using terraform I would like to go to AWS
and interact with it's service
launch instance — EC2.

→ terraform apply (command)

- HCL → declarative language

- Languages

Imperative (not intelligent)
↓
Python
↓
Keep on launching
OS not checking
of its presence.

declarative (Intelligent) (we put our desires)
↓
HCL
first checks if the OS is there
or not of the name and
then launches.

desire → Terraform → declarative

▷ Terraform Site → Providers → AWS → EC2 → Resources.
↓
Copy Syntax

○ variables == arguments
Eg :- Key-name, ami, etc.

(space) "myinstance" (Infrastructure as a code)
↑
Keyname
↑
In future

```
(Keyword) resource "aws_instance" {
    ami = "ami-07a839b2a12"
    instance-type = "t2.micro"
    key-name = "myKey.pem"
    security-groups. = ["launch-wizard-2"]
    tags = {
        Name = "Linux OS 1"
    }
}
```

treats as list therefore [" "]

→ terraform apply ← [command]

○ whatever terraform create, update, modify they know about it
↓
stores the record
↓
"State file"
↓
▷ (terraform.tfstate)

○ terraform destroy [command]
↑
destroys the infrastructure in one go.