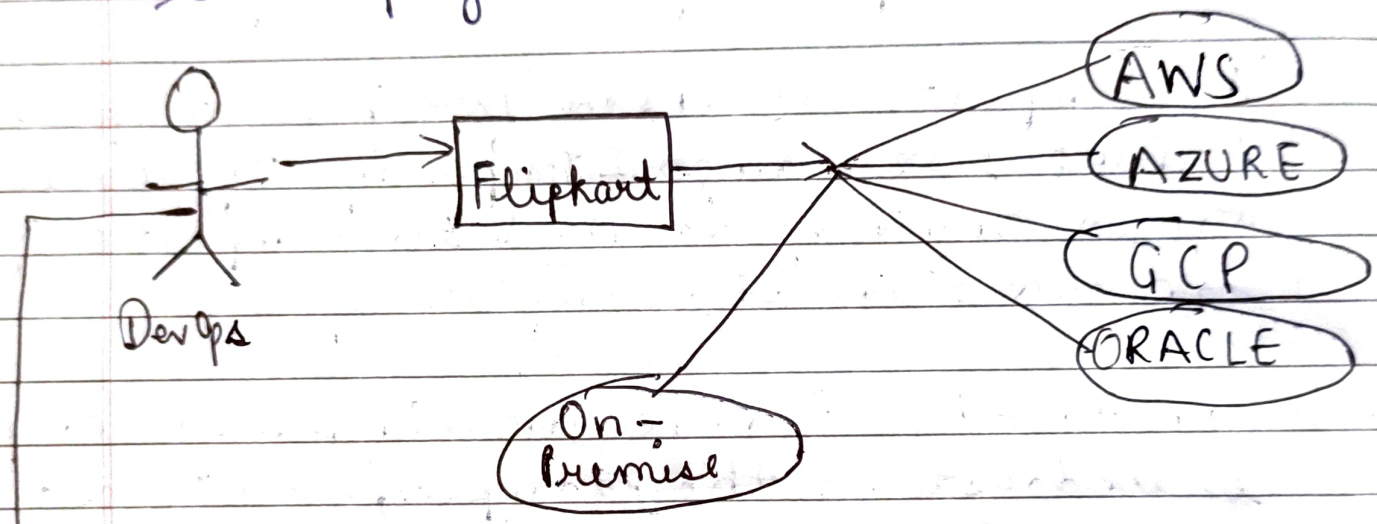
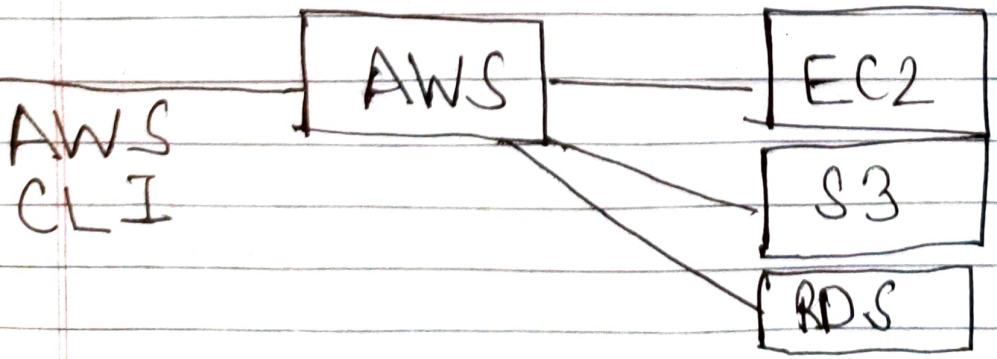
Day 16 : Infrastructure as Code - Terraform

Let's say, you're a DevOps Engineer, and you're working for an org - Flipkart.

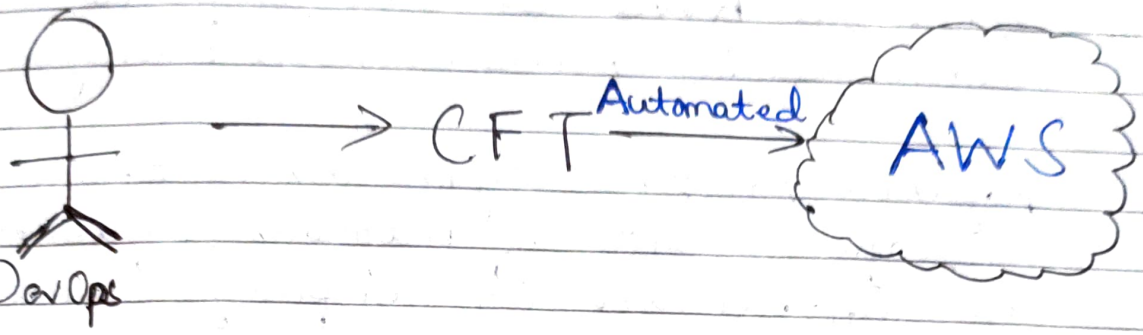
So, Flipkart has various options to deploy their applications. They can deploy their applications on On-Premise servers or can deploy them on Cloud.



Let's say, DevOps Engineer went with AWS Cloud for hosting their applications. So, he need to use various resources of AWS as per the use-case of the organization



Now, you have automated the process for creating and manging resources with the help of AWS CLI, rather than creating them through AWS Console



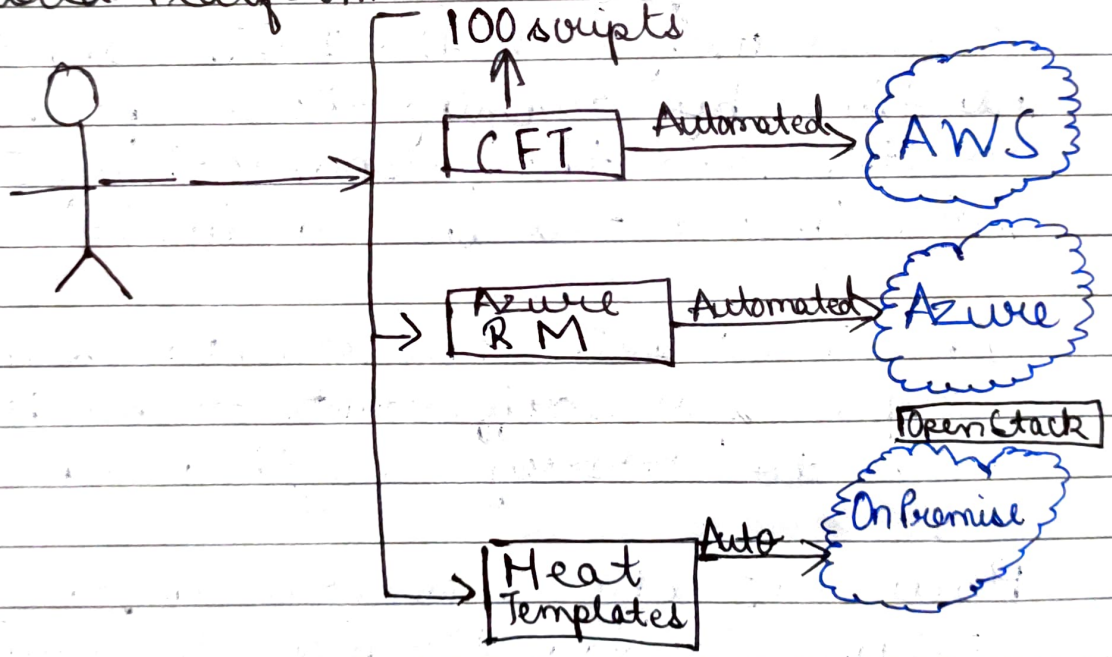
When a Developer comes to the DevOps now when he has automated the task of creating Resources using CFT, the DevOps can do it in no time by running the script he has already written and the resources (say EC2 instances) will be created in no time.

Likewise, DevOps has created several other scripts for different tasks. He doesn't have to worry and the work is going smooth

However, the Organization due to some reason decided to migrate from AWS. to may be Microsoft Azure.
So, now the scripts that the DevOps has written will be of no use because CFT that DevOps Engineer has written is absolutely specific to AWS Cloud Platform. Therefore, it will not work with Azure.

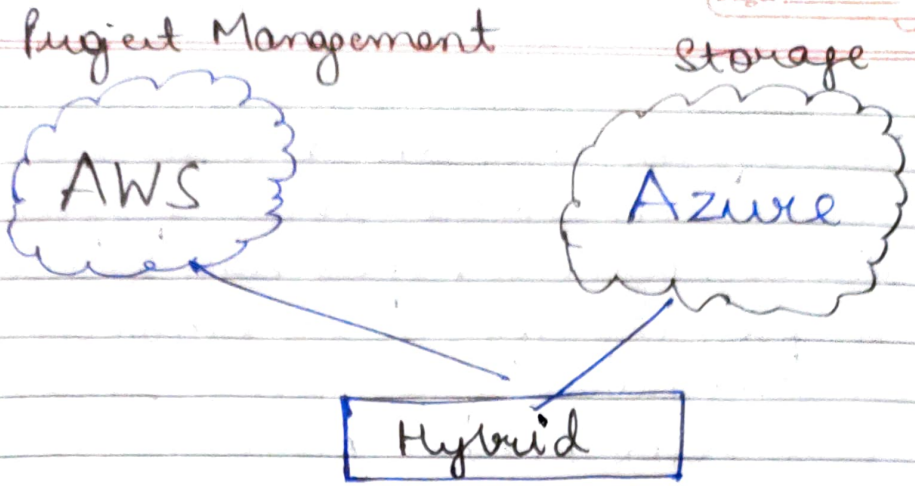
Now, the DevOps again has to make efforts to rewrite scripts that are compatible with Azure.
Azure Resource Manager is analogous to CFT in AWS. So, the scripts has to be converted to Azure Resource Manager

Again, in the Organization there is a requirement of migration from Azure to On-Premise. So, company wants to purchase their own servers, and using OpenStack, they will build their own Cloud Platform.



Heat Templates are used to automate processes on Openstack.

In real world scenario, the Organizations are opting for Hybrid Cloud Models. that further increased the complexity



Hybrid Cloud is provision where you can create apart of Cloud Infrastructure on AWS, and another part on Azure.

This is done because one Cloud Provider (let's say Azure) is giving you good support for Storage, so you've decided to go ahead with Azure for Storage. Likewise, for Project Management related services you want to use AWS.

So, the DevOps working for an organization that uses Hybrid Cloud Model, he has to learn multiple things like CFT for AWS, and Azure Resource Manager for Azure.

X X

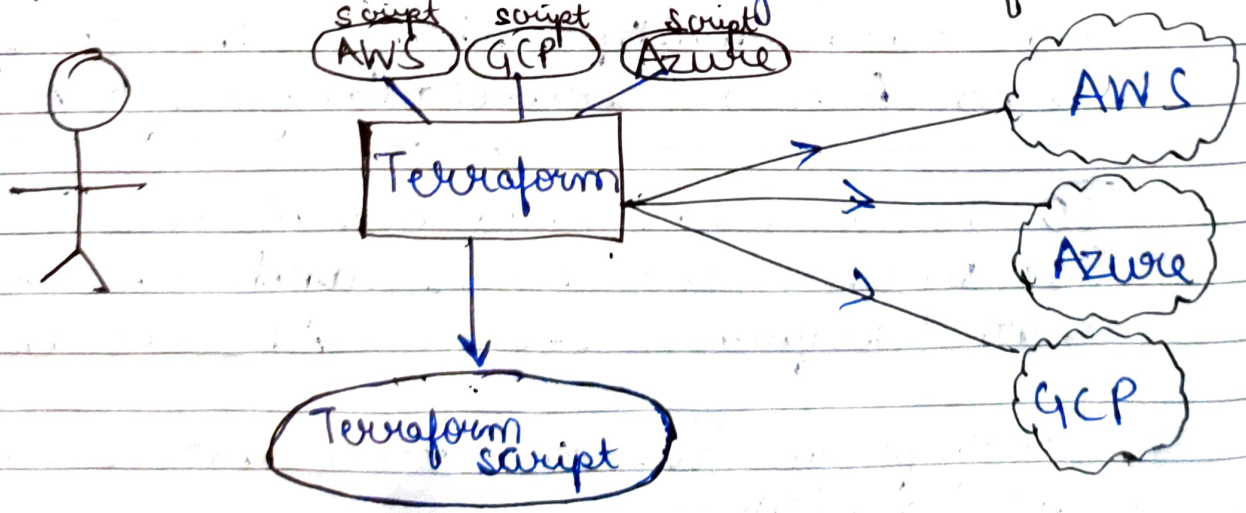
To overcome the above problem, Terraform has been introduced, and the concept that Terraform uses is API as Code

Terraform is a tool that is developed by Hashicorp

Terraform introduction in the market gave an idea that if you are a DevOps Engineer, you can create Terraform Scripts. Terraform will handle the automation of resources whether resources are on AWS, Azure, GCP etc.

All DevOps Engineer has to do is let the Terraform know that what is the Cloud Provider we are using, and Terraform will accept scripts for a specified Cloud Provider, and it will automate.

So, just by minimal changes in **Modules** or Cloud Provider Configuration, you can migrate from one Provider to another smoothly. You don't have to learn different tools, and by just going through Documentation for Terraform, you can learn to automate any Cloud Infrastructure



API as Code

API as Code

API as Code helps in automating any Cloud Infrastructure (AWS, Azure, OpenStack) using their APIs

Terraform will talk to the API of AWS or Azure or GCP internally

Once you have written Terraform Template or Terraform script, you don't have to perform anything. Terraform will convert the Terraform Module or script to the specified Provider's readable API or API request that the Provider understands.

After that, Terraform let the Provider execute the task, and provide back the Response.

API

API stands for Application Programming Interface that allows communication to an application and get the response.

Whenever we want to perform a certain task programmatically, we make use of APIs

For example, if we want to search for a piece of information on Google, so instead of going through the GUI process (by opening the browser using mouse, and typing the search text on URL bar using keyboard,

we will run a script. What that script will do is, it will talk to Google, executes the action, and sends back the response

To perform this kind of process, what Developers has done, they have come up with the concept of API.
What any organization does, it exposes its APIs, and it says that instead of performing an action manually, you can use CURL or HTTP GET Request on Terminal. Using them, talk to the APIs, get whatever information you want, and I will send you the Result.

The Programmatic Interface is known as API.

Terraform uses the same concept as API
All the Cloud Providers including AWS, Azure or GCP has their own APIs

Developers has the option to directly talk to the APIs, but to do that they must know some programming language. Moreover, in case of complex tasks, the GET Requests or POST Requests are bit difficult to be included in Shell Scripts or Python Scripts.

To solve this problem, Terraform came into picture
Terraform actively looks at the APIs given

IaC → Automate Infrastructure

API as Code → Terraform

by various Cloud Providers, and Terraform has their own modules. So, it says instead of Developers directly talking to the APIs, they can make use of Terraform script.

In Terraform scripts, you'll have to write bunch of lines depending upon the task you're going to perform, and you can take help from the extensive documentation, that Terraform maintains.

Let's say, you want to create EC2 instances, so you can go the Documentation and search for EC2 instance, and you'll find a module corresponding to EC2. So, as said before, using this EC2 module, you'll write bunch of lines, like we does in English (High-Level Language)

Once user submits the request in Terraform file, Terraform will receive the input, and it will convert the input (request) into an API call This is happening in the background.
Terraform will send the API call, executes the action, get the response from the Cloud Provider, and again converts it to be send back to the Developer (or person who ran the script)

This is how Terraform utilizes the concept of API as Code.