Unit-V

Cloud Platforms for IOT

5.1) Software Defined Networking
5.2) Introduction to cloud storage models
5.3) Comm^n API
5.4) LAMP:-
- 5.4.1) AutoBahn for IOT
- 5.4.2) Xively cloud for IOT

5.5) Python Web Appl^n framework:-
- 5.5.1) Django Architecture & appln development with Django
- 5.5.2) Amazon Web service for IOT
- 5.5.3) SkyNet IOT Messageing platform
- 5.5.4) RESTful Web Service
- 5.5.5) GRPC
- 5.5.6) SOAP

Date

S.1) Introduct^n s/w Defined Networking.

- Software - Defined Networking (SDN) is a n/w architecture approach that enables the n/w to be intelligently & centrally (controlled) or 'programmed' using s/w appl^{ns}.
- This helps operators manage the entire n/w consistently & holistically, regardless of the underlying n/w technology.
- SDN lets user use s/w to provision new devices instead of using physical infrastructure, so IT administrators can direct n/w path & proactively arrange network services.
- Unlike traditional switches, SDN also has the ability to better communicate with devices using the n/w.

S.2) Introduct^n to Cloud storage model -

- Cloud storage model is model of cloud computing that stores data on the internet via cloud computing providers.
- Cloud storage is basically an online storage of data.
- Cloud storage can provide various benefits like greater accessibility & reliability, strong protect^n of data backup, rapid

deployment, and disaster recovery purposes.
- Moving to the cloud also decreases overall storage costs due to cutting costs incurred on purchase of storage devices & their maintenance.

- Let's discuss the 3 cloud storage models:-
1. Instance storage: Virtual disks in the cloud
2. Volume storage: SAN sans the physical
3. Object Storage: Web scale NAS

5.2.1> Instance storage: Virtual disks in the cloud:-
- In a traditional virtualized environment, the virtual disk storage model is the eminent one.
- It is crucial to note that instance storage is a storage model, not a storage protocol.
- This storage can be implemented in numerous ways.
For example, DAS is generally used to implement instance storage.
- It is often stated as ephemeral storage as the storage isn't highly reliable.

- Advantages & Disadvantages:-
1) Hard drive that instance storage run on are physically attached to the EC2 hosts which are running the store. Their endurance depends upon the lifetime of the instance attached to them.
2) Both instance & Elastic Block storage volume are stored in a series somewhere in the same AZ.
3) Due to the paucity & speed & persistance in instance storage, it is usually used to data that requires quick but temporary assessment, like swap or paging files.
4) However, it is also used to store data that requires regular replication to multiple location.
5) EC2's using instance storage does not ensure any data, which is the reason behind longer boot time when compared to instances backed by EBS.

- Volume storage:
- Volume storage is also known as block storage.
- As suggested by it's name, data is stored in structured blocks & volumes where files are split into equal-sized blocks.

- However, unlike objects, they don't possess any metadata.

- Advantages & Disadvantages:
1. Public cloud providers allow the creation of various file system on their block storage systems.
2. Additionally, an amazon EBS volume is accessed from an Amazon EC2 instance through an AWS shared or dedicated nw.
3. Another advantage of using volume/block storage is its backup mechanism.

- Object storage: Web-Scale NAS:
- Cloud-native appln's need space for storing data that is shared between different VMs.
- Amazon Simple Storage Service(s) caters to a single space across an entire region, probably, across the entire world.
- Object storage stores data as objects, unlike others which go for a file hierarchy system.
- Object storage also saves a substantial amount of unstructured data.

Page No.
Date

5.3) Communicatn API:-

- The applⁿ program interface, or API, is arguably what really ties together the connected "things" of the "internet of things".

- REST based commⁿ APIs:-
- Representational state transfer (REST) is a set of architectural principles by which you can design web services the web API, that focus on system resources and how resource states are addressed and transferred.

- Web-Socket based commⁿ API:-
- Websocket APIs allow bi-directional full duplex commⁿ betⁿ clients and servers.
- Websocket commⁿ begins with a connⁿ setup request sent by the client to the server. The request is sent over HTTP & the server interprets it is an upgrade request.
- Websocket API reduce the n/w traffic and latency as there is no overhead for connⁿ setup & termination request for each message.
- Websocket suitable for IOT applⁿs that have low latency or high throughput requirements. So web socket is most suitable IOT commⁿ APIs for IOT system.

Date

5.4> WAMP: -

5.4.1> AutoBahn for IoT WAMP:

Autobahn for IOT: -

- Mainly used in cloud storage model for IoT & other messaging services.
- WAMP is a routed protocol, with all components connecting to a WAMP Router, where the WAMP Router performs message routing b/w the components.

- Some Important key Terminologies:
  - Transport
  - Session
  - Client
  - Router
  - Broker
  - Dealer
  - Appln Code

- WAMP for IoT: -
  - Web Appln Messaging Protocol is a sub-protocol of WebSocket which provides publish-subscribe and remote procedure call (RPC) messaging patterns.

5.4.2> Xively Cloud for IOT: -

- Xively is an IoT cloud platform that is "an enterprise platform for building, managing, and deriving business value from connected products".

- Xively is a system for deploying IOT appln's on the cloud. It is offered as PaaS.
- Subsequently, question is, what is Xively data stream?

S.5> Python WEB Appln Framework:-

S.5.1> Django Architecture & appln development with Django:-

- Django is a Python-based web framework that allows you to quickly creat efficient web appln's.
- It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database sqlites, etc.

- Why Django framework?
- Excellent documentation & high Scalability.
- Used by Top MNCs and companies such as Instagram, Disqus, Spotify, Youtube, Bitbucket, Dropbox, etc. and the list is never ending.
- Easiest framework to learn, rapid development & Batteries fully included.
- The last but not least reason to learn Django is Python, Python has huge library & features such as web scraping,

ML, IP, Scientific Computing, etc..

- Django architecture:-

- Django is based on MVT architecture.

MVT structure has the following three parts:-

1. Model:-
   Model is going to act as the interface of your data.
2. View:-
   The View is the user interface - what you see in your browser when you render a website.
3. Template:-
   A template consists of static parts of the desired HTML o/p as well as some special syntax describing how dynamic content will be inserted.

- Installation of Django:-

- Install python3 if not installed in your system, from here. Try to download the latest version of python. It's python 3.6.4 this time.
- install pip
- install virtual environment
- set virtual environment

- Creating a Project:-

Let's check how to create a basic project using Django after

You have installed it in your PC.
1. To initiate a project of Django on your PC, open Terminal & Enter the following command
django-admin startproject projectName.
2. A New folder with name projectName will be created. To enter in the project using terminal enter command.

cd projectName
Now run,
Python manage.py runserver
Now visit http://localhost:8000/,

- Creating an App:-
Django is famous for its unique and fully managed app structure. For every functionality, an app can be created like a completely independent module.

- To create a basic app in your Django project you need to go to directory containing manage.py & from there enter the command:

python manage.py startapp projectApp
Now you can see your directory structure as under:

- projectName
  - projectApp
    - migrations
    - __init__.py
    - admin.py
    - manage.py

- To consider the app in your proj. you need to specify your project name in INSTALLED_APPS list as follows in settings.py:

Appln def^n

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'projectAPP'
```

- So, we have finally created an app but to render the app using urls we need to include the app in our main project so that urls redirected to that app can be rendered. Let us explore it.

- Move to projectName -> projectName -> urls.py & add below code in the header from django.urls import include.
- Now in the list of URL pattern you need to specify app name for including your app urls. Here is the code for it -

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
```

Enter the app name in following syntax for this to work.
path('', include ('projectApp.urls')),

- Now you can use the default MVT model to create URLs, models, view, etc.

5.12 Amazon Web Services for IoT:-

- AWS IoT is an Amazon Web services platform that collects & analyzes data from "internet-connected devices & sensors & connects that to AWS cloud apps.
- An AWS user accesses AWS IoT with the AWS mgmt. console, s/w development kits (SDKs) or the AWS Command Line Interface.
- The IoT service includes a Rule Engine feature that enables an AWS customer to continously ingest, filter, process & route data that is streamed from connected devices.
- The feature also allows the user to configure how data interacts with other big data and automation services, such as AWS Lambda, Amazon Kinesis, Amazon S3, Amazon DynamoDB & Amazon Elasticsearch Service.
- Device Shadow is an optional rule that enables an appl

to query data from devices & send commands through RESTful APIs.
- The optional Device Registry feature lets a developer register & track devices that are connected to the services, including metadata for each device such as model number & associated certificates.
- Each Thing associated with a Thing Type can have up to 50 attributes & 3 searchable attributes.
- A developer can also use open-source AWS IoT device SDKs to optimize memory, power & n/w bandwidth consumption for devices.
- AWS IoT supports HTTP, MQTT & WebSockets, comm protocols b/w connected devices & cloud apps through the device gateway, which provides secure 2-way comm while limiting latency.
- AWS requires devices, apps & users to adhere to strong authentication policies via X.509 Certificates, AWS Identity & Access mgmt credentials or third-party authentication via Amazon Cognito.
- AWS IoT offers a free tiered service. Beyond that tier, a customer is charged according to the number of published msgs.

S.S.3> Sky NET :-

- Open source instant messaging platform.
- SkyNet API supports both HTTP REST & real-time WebSockets.
- SkyNet allows you to register devices on the n/w.
- A device can be anything including sensors, smart home devices, cloud resources, drones.
- Each device has an UUID & secret token.

S.S.4> RESTFUL WEB Services :-

- RESTful web services :- are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.
- In the REST architectural style, data & functionality are considered resources & are accessed using Uniform Resource Identifiers (URIs), typically links on the Web.
- The resources are acted upon by using a set of simple, well-defined operations.

- The REST architectural style constrains an architecture to a client/server architecture & is designed to use a stateless comm^n protocol, typically HTTP.
- The following principles encourage RESTful appl^ns to be simple, lightweight, and fast:
  - Resource identification through URI:
    A RESTful web service exposes a set of resources that provide a global addressing space for resource & service discovery.
  - Uniform interface:
    Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE.
  - Self-descriptive message:
    Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and other.
  - Stateful interactions through hyperlinks:-

Every interaction with a resource is stateless; i.e. request msg. are self-contained, stateful interactions are based on the concept of explicit state transfer.

S.No 6> SOAP:-

- Well start with SOAP, the oldest web-focused API protocol that remains in widespread use. Introduced in the late 1990s,
- API protocols - but know that if you want to dive deep into the history of SOAP & the concepts that make it possible, you have to start with RPC
- SOAP stands for simple object access protocol.
  Some developers might take issue with the notion that SOAP is "simple", because using a SOAP protocol requires a fair bit of work.
- You have to create XML documents to make calls, and the XML formatting that SOAP requires isn't exactly intuitive.
- On the other hand, SOAP relies on standard protocols, especially HTTP and SMTP. That means that you can use SOAP in virtually every type of environment, because these protocols are available

on all operating system.

5.5.6) GRPC :-

- GRPC is an open source API that also falls within the category of RPC. Unlike SOAP, however, GRPC is much newer, having been released publicly by Google in 2015.
- Like REST & SOAP, GRPC uses HTTP as it's transport layer. Unlike these other API protocols, however, GRPC allows developers to define any kind of function calls that they want, rather than having to choose from predefined option (like GET & PUT in the case of REST).
- Another imp advantage of GRPC, at least for many use cases, is that when you make a call to a remote system using GRPC, that call appears to both the sender and the receiver as if it were a local call, rather than a remote one executed over the network. This simulation avoids much of the coding complexity that you'd otherwise have to contend with in order for an appln to handle a

remote call.

- The ability of GRPC to simplify otherwise complex remote calls has helped make it popular in the context of building API, for microservices or Docker-based appln, which entail massive number of remote calls.