**TO DO**

1. **Atlassian Jira, Confluence**
2. **GIT/GITHUB (Actions in YAML as well)**
3. **Python – File, pointer, NumPy, Pandas, Request library**
4. Azure DevOps **(knowledge depth as much as possible)**
   1. What mean by variable groups?
   2. Task groups?
   3. Pipeline variables?
   4. Pull Request
   5. Gated Build
   6. Diff b/n classical editor and YAML in Azure Pipelines
   7. **Steps to configure pipeline**
5. **Linux –** (Shell scripting may be)- (To see permissions)
6. **Docker**
7. YAML and JSON
8. Azure Overview – Networking, Storage, Functions, Azure SQL, Cosmos dB
9. **Terraform – See the codes and structure of terraform**
10. **ARM Templates – (How to launch VM) – Structure of files**

* **Gone through Azure training.**
* **(Complete Revision needed)**
* **Docker troubleshooting**
* **Kubernetes commands**
* **Terraform Commands**
* Preparation Questions
* Introduction:
* About recent engagement:
* Questions from Resume:
* Python scripts
* SQL scripts (write codes for view, joins, etc.)
* Pyspark (RDD and Data frame)
* AWS components
* Databricks (general idea): web based platform to run Pyspark jobs, and provides the managed clusters for computing and HDFS and lots and lots more.
* Explain the process of CI and CD in your project, YAML files for GIT webhooks

**Introduction**

I am Shekhar Satyakar, I am working as an analyst in KPMG. I have been born and brought up in Ranchi Jharkhand. I did my graduation in the field of Electronics and Communication Engineering from BIT Mesra, Ranchi. I have been working in KPMG as an analyst from since July 2022.

**Recent Engagements**

* About MSIL project:
  + Recently I worked on a project where the clients needed the model to generate the keyword-based summary of their research papers. I

**Azure Virtual Machine**

**These are virtualized computing environments that run on the Ms Azure cloud Platform. These are very scalable, and flexible way to run a variety of applications and services on Windows or Linux.**

* Same as EC2 in AWS.
* Created Virtual machine in following steps
* Creating a resource group and selecting compute
* Given the following information
* Subscription, Resource group, Virtual machine, name, Region, Image, Size
* Mounted a persistent disk storage externally
* Onto networking segment
* Created virtual network (Vnet), and subnet inside Vnet
* Allowed public inbound ports – SSH
* After validation the VM was created

**Azure Storage**

**Blob storage, Queue, Table and Files**

**Types of Data**

* Structured
* Unstructured = **Blob storage**
* Semi-structured = **Table storage**

1. **Azure Blob Storage: Scalable cloud storage solutions provided by Azure. It is designed to store large amounts of unstructured such as texts or binary data (images, videos files and backups)**

* Scalable
* Stands for Binary large object
* Unstructured data
* Designed to store any kind of data inside the containers
* Three storage tiers
  + HOT: Best performance, fastest retrieval
  + Cool: Infrequently accessed data (lower availability, higher durability)
  + Archive: takes hours to retrieve

1. **Azure Queue Storage: These are the decoupling storage solution from Azure, it allows user to store and retrieve the messages in FIFO in asynchronous manner.**
2. **Azure Table Storage: These are the NoSQL key-value storage solution provided by Microsoft Azure. It allows users to store and retrieve large amount of semi-structured data in highly scalable manner.**

**Stores data in tables, consisting of rows and columns, each row is identified by a unique partition key and a row key.**

* Semi- Structured data
* NOSQL, No joins no schemas
* Designed for fast access
* It has many programming interfaces and SDK’s

1. **Azure File Storage: Azure file storage is a cloud-based file sharing service provided by Azure. It allows to create files and share the cloud access to various VMs via SMB (server message block) protocol.**

* It is different from Blob storage as it stores data in small chunks of file unlike blob in blob storage.
* Serverless SMB file shares
* Storage for **files** accessed via **shared protocols**
* Designed to extend **on-premises file shares** or implement **lift-and-shift** scenarios

**Storage Account**

* Group of services which includes
  + Blob
  + Queue
  + Table
  + File storage
* Used to store
  + Files
  + Messages and
  + Semi-structured data
* Highly scalable
* Highly durable
* Cheapest per GB storage

1. **Azure Disk Storage**

* Used for Virtual machines
* Persistent storage for virtual machines
* Comes with different sizes, types (SSD, HDD), performance tiers
* Disk can unmanaged or managed

**Azure Networking**

1. **Vnet**: **Emulation of physical networking infrastructure**. Intra Vnet communication can be done by Vnet peering, and VPNS.
2. **Subnets**: **Division of Vnet into smaller segments** to assign NSG separately as per the requirements.
3. **VPN Gateway**: used to establish **connection between Vnet and on-premises/Another Vnet.**
4. **NSG (Network Security Group): using this we set the inbound and outbound rules** to a subnet/NIC/Vnet/VM.
5. **CDN (Content Delivery Network): Contents are delivered across all azure regions** to **reduce the latencies**.
6. **NIC (Network Interface Card):** Kind of **virtual ethernet card by which one machine communicates with other machines.**
7. **Route Tables: It routes the connection** between **machines or Azure components to elements outside the Vnet** such as on-premises/ Internet/ Other Vnet, etc.
8. **Service Endpoint:** Allows to filter egress virtual network traffic to Azure storage accounts over service endpoints
9. **Bastion hosts: I**t is a **server** whose **purpose** is to **provide access to a private network from an external network** such as internet, accept server from SSL 443
10. **NAT Gateway (Network address translation): Translates the IP address of server or VM server/Internet to subnet/Azure components**

**Azure Encryption**

* Server Side and Client-side encryption.
* **CLIENT-SIDE ENCRYPTION**
* Keys can be managed or stored in on-premises or in other secure account.

**Azure Key Vault**

**Azure key vault is cloud-based service offered by Microsoft Azure that allows us to securely store and manage KEYS, SECRET and CERTIFICATES.**

* store **encryption keys**.
* Stores the application **secret** (Username, Password, Address, etc.) for web app.
* Stores **certificates** (encrypt the traffic in transit, or for authentication)

Azure key vault is a centralized cloud service for storing application secrets such as encryption keys, certificates, server-side tokens. Key vault helps in controlling application’s secret by keeping them in a single central location and providing secure access, permission control, and access logging.

**Secrets Management:** can securely store and control access to tokens, passwords, certificates, API keys and other secrets.

**Key Management:** Cloud based key management solution, which makes it easy to control the encryption keys to encrypt the data. Azure services integrate with key vault to further decrypt it.

**Certificate Management:** Manage, provision, and deploy public and private SSL/TLS certificates for use with internal connected resources.

**Soft delete:** Safe for accidental deletion, within retention period (default 90 days)

**Purge Protection:** enforce mandatory retention period for deleted vaults and vaults. ()

**Azure Web Apps**

**Cloud based solution provided by Microsoft Azure to build, deploy and manage the web applications in fully managed and scalable environment.**

Azure App service enables us to build and host **web apps**, **mobile backends,** and **restful APIs** in the programming language of our choice without managing the infrastructure.

Provides autoscaling and high availability and supports both Windows and Linux and enables automated deployments from GitHub, Azure Devops or other GIT repo.

* Optimized for devOps i.e., automated CICD pipelines
* Visual studio integration

**Azure Logic Apps**

**Cloud based solution from Microsoft Azure that allows us to build and automate workflows to integrate with various cloud and on-premises services. It provides a visual designer for creating workflows, which can be triggered by events or can be scheduled one.**

* Cloud based platform for creating and running automated workflows that integrate your apps, data, services, and systems.
* With this application platform it becomes easy to develop highly scalable integration solutions
* Azure functions vs Azure logic Apps,
* Logic apps have no code, we can focus on developing logic instead of code

**Azure Functions**

**It is a serverless computing service from Azure which allows to run small piece of code in response of certain triggers such as HTTP request, messages queues, or time-based schedules.**

* Can build connection between different Azure components as in
* COSMOS DB / Blob Storage/ Storage Queue -------------------🡪 Azure functions --------🡪 Blob storage/ Event hub, etc.

Features are

1. Highly scalable
2. Event-driven
3. Integration with other Azure services
4. Easy to deploy
5. Multi language facility
6. Monitoring and logging

**Azure SQL:**

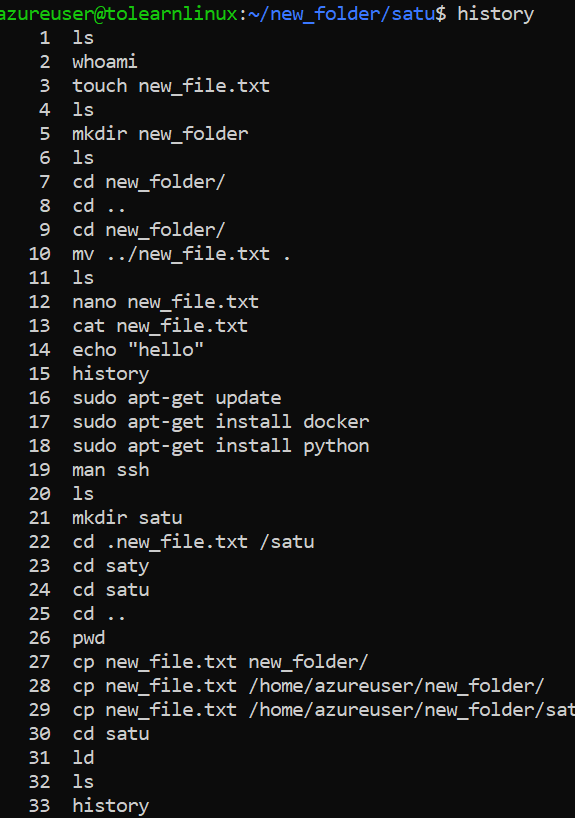
**It is a cloud-based relational database service provided by Azure cloud platform. It allows users to manage databases in the cloud using the SQL database engine**.

**Azure COSMOS DB**

**Azure Cosmos DB is highly scalable and globally distributed NoSQL database service that supports multiple data models and APIs**

* NoSQL database, can handle a variety of data types and structures, including document, key-value, and column-family data.
* Offers native supports for multiple APIs, including SQL, Mongo DB, Cassandra, Azure Table Storage, enabling developers to use their preferred programming language and data models.

**Linux Basics**

****

* 1. Copying files from local to server using scp
  2. >> sudo scp -i <pemfile> from <source> ubuntuazureuser@<Public IP>: home/root/dev/new folder
* To download from server to local machine

>> Sudo scp -I <pemfile> from ubuntuazureuser@<Public IP>: home/root/dev/new-folder/new-file.txt .

* **Permissions to the users**
* r – read, w- write, x- execute
* u- current users, g- user groups, o- others, a- all

>> ls -ltr

>> **d**rwxr-xr-x: d is directory, (owner{rwx}, group{rx}, all{x})

>> **-**rwxr--r--r: regular file: - is regular file, (owner{rwx}, group{r}, all{r})

* **Changing permissions**
* Chmod u+r <file\_name> // adding read permissions
* Chmod u-r <file\_name> // removing permissions from users
* Chmod ugo+w <file\_name> // giving write permissions to user, group and all
* Chmod u-rwx <file\_name> // removing all file names
* Chmod o+w <file\_name> // giving write permission to other users

**Numeric Method: chmod 754 <file\_name>**

* User+group+other
* Read write execute
* User = 7; 1 1 1
* Group = 5; 1 0 1
* Others = 4; 1 0 0
* Sudo: acronym of superuser

**HTTPS vs HTTP: HTTPS is secured HTTP, used in banking where the important credentials must be protected.**

**Docker**

1. What is Docker?

**Docker is an open-source containerization platform. It is used to automate the deployment of any application, using lightweight, portable containers.**

2. What and why containers?

**Containers are packages of applications with all necessary dependencies and configurations.**

* Can run same application with two different versions
* Has its own packed dependencies
* Own isolated environment

2. What is Docker Daemon?

It **listens for Docker API requests** and manages docker objects such as images, containers, networks, and volume.

3. **Docker file?**

It is instruction to build a docker image and further docker containers.

4. **Components of Dockerfiles**

**FROM**: specifies the base image

**WORKDIR**: sets the working directory

**COPY**: copies the files from host machines to the containers

**RUN**: runs the command within the containers to install package

**EXPOSE**: specifies which port must be exposed by the containers

**CMD**: specifies the commands to run once container starts

Step 1: create a file named Dockerfile

Step 2:

# Getting base image from ubuntu

**>> FROM ubuntu**

**>> RUN apt-get update** # runs while creation of dockerfile

**>> CMD [“echo”, “Hello world from ubuntu docker image”]** #runs after dockerfile is created

Step 3: dockerise the folder to build docker image

>> docker built . # . if current location else pass the location

>> docker built . -t my\_docker\_image

Step 4: Run image using image id

>> docker ps

>> docker run <image\_id>

Docker files other examples

**FROM** node:13-alpine

**RUN** mkdir -p /home/app

**COPY**. /home/app #runs on the host

**CMD** [“node.js”, “server.js”]

2. Name and explain the various Docker components.

The three main Docker components are:

Docker **Client**. Performs Docker build pull and run operations to open communication with the Docker Host. The Docker command then employs Docker API to call any queries to run.

Docker **Host (Machine responsible for running one or more containers)**. Contains Docker daemon, containers, and associated images. The Docker daemon establishes a connection with the Registry. The stored images are the type of metadata dedicated to containerized applications.

**Registry**. This is where Docker images are stored. There are two of them, a public registry and a private one. Docker Hub and Docker Cloud are two public registries available for use by anyone.

7. Explain virtualization.

Virtualization is the means of employing software (such as Hypervisor) to create a virtual version of a resource such as a server, data storage, or application. Virtualization lets you divide a system into a series of separate sections, each one acting as a distinct individual system. The virtual environment is called a virtual machine.

**Docker Images**

1. Build an Image from a Dockerfile: **docker build -t <image\_name>**
2. Build an Image from a Dockerfile without the cache: **docker build -t <image\_name> . –no-cache**
3. List local images: **docker images**
4. Delete an Image: **docker rmi <image\_name>**
5. Remove all unused images: **docker image prune**

**Docker Hubs**

1. **Login into Docker: docker login -u <username>**
2. **Publish an image to Docker Hub: docker push <username>/<image\_name>**
3. **Search Hub for an image: docker search <image\_name>**
4. **Pull an image from a Docker Hub: docker pull <image\_name>**

**General Commands**

1. **Start the docker daemon: docker -d**
2. **Get help with Docker. Can also use –help on all subcommands: docker --help**
3. **Display system-wide information: docker info**

**Containers**

1. **Create and run a container from an image, with a custom name:**
   1. **docker run --name <container\_name> <image\_name>**
2. **Run a container with and publish a container’s port(s) to the host.**
   1. **docker run -p <host\_port>:<container\_port> <image\_name>**
3. **Run a container in the background: docker run -d <image\_name>**
4. **Start or stop an existing container: docker start|stop <container\_name> (or <container-id>)**
5. **Remove a stopped container: docker rm <container\_name>**
6. **Open a shell inside a running container: docker exec -it <container\_name> sh**
7. **Fetch and follow the logs of a container: docker logs -f <container\_name>**
8. **To inspect a running container: docker inspect <container\_name> (or <container\_id>)**
9. **To list currently running containers: docker ps**
10. **List all docker containers (running and stopped): docker ps --all**
11. **View resource usage stats: docker container stats**

**Azure DevOps**

Azure devOps is a set of tools and services provided by Microsoft that provides the platform for managing the entire devOps lifecycle like

i.e., from planning to building and monitoring CICD pipeline.

PLAN 🡪 CODE 🡪 TEST🡪 PACKAGE🡪DEPLOY

Boards REPO …... PIPELINE ……

…………...CI………………………………………🡪 …CD….

**Components**

1. **Azure Boards -** Project management tool for **Agile** or **Scrum** processes
2. **Azure Pipelines -** Written in YAML

**Has steps**

* Test
* Package
* Build docker image
* Push to repo
* We test and build the files using .NET commands and
* Build image with docker commands
* Task is a pre-created script offered as a convenience
* Can perform multiple jobs within a pipeline (e.g., running code on multiple environments as in Windows, Linux., etc.)
* Has different agents to run package on different environment.

1. **Azure Repos –** Alternative to different VCS as GITHUB, GITLAB, etc. Also has GIT workflow.
2. **Azure tests plans**
3. **Azure Artifacts:** Can store and share different packages as in npm, nudget, maven.
   1. Now a days we don’t produce such artifacts anymore we do use different containers

PLAN 🡪 CODE 🡪 TEST🡪 PACKAGE🡪DEPLOY (DEV)🡪DEPLOY (TEST)🡪 DEPLOY (PROD)

This Dev test and prod also available in Azure Pipeline

**Terms to know in Azure DevOps**

1. **Variable Groups**: Group of variables (storing any classified information or encryption keys) which could be applied to one or more Azure pipelines.
2. **Task Groups**: Groups the task so that we can use the same set of tasks in multiple pipelines. These are parameterized so adaptive to the different conditions of pipelines.
3. **Pipeline variables:** Defines a value which can be used within a pipeline,Variables gives a convenient way to get key bits of data into various parts of the pipelines. Can be created at root, stage, and job level.

Eg

Variables:

Myvariables: “Hello, World!”

We can call the referenced variables as

Steps:

* + 1. Scripts: echo ${{ Variables.Myvariables }}

1. **Pull requests (Azure Repos):** Used to merge a branch to some other branch showing the changes one by one which makes it easy for the developers to collaborate on code changes before merging them into the main branch.
   1. With pull requests, a copy of the code is made in a separate branch, which can be reviewed and tested by other developers
2. **Gated Build:** It is a build process that includes additional validation steps before changes are merged into the main branch.
   1. Instead of merging with the main branch, the changes are submitted to a pre-merge build that runs automated test and other validations steps to ensure that the changes are valid.
3. **Difference between classic editor and YAML:**

* Classic editor provides the graphical user interface, good for people who are less acquainted with codes, but are less flexible also as per the use cases.
* In contrast, YAML provides a way to define pipelines using text based YAML files that can be stored in a repository.
* YAML Can have versions unlike classic editor.
* In a nutshell YAML is helpful in creating complex scenarios.

1. **What are the steps to configure the Azure pipelines? (See the steps also)**
   1. **Creating a new pipeline:** click on Pipeline menu and selecting create pipeline
   2. **Choosing a pipeline template**
   3. **Configure the pipeline settings:** such as source code repo, built and deployment triggers and build deployment agents.
   4. **Defining the build process:** Need to define the steps to build the application
   5. **Defining the deployment process:** Defines the steps required to deploy the process
   6. **Testing and debugging the pipeline:** TO test and configure the pipeline, You can use Azure Debugging tools to identify any issues and can fix them
   7. **Monitoring the pipelines:** Finally need to monitor your pipelines to ensure that it continues to work properly.
2. **Azure pipelines Agents:** Software which runs inside a machine (having CPU, rams etc.) and execute the pipeline task.

**Concepts of Devops:**

Helps in working of Dev team and Operational team in coordination.

**P**lanning: Azure Boards

**E**xecution: Repos

**T**esting: Test\_Plans

**U**AT

**L**ive

**A**ssistance

Pipelines: has series of tasks as in install npm, update to particular version,. Etc

**Artifacts: are the output of a pipelines**

**Templates for Azure Pipelines**

* Can distribute entire YAML files into pieces of YAML files, as in
* test.yml, build.yml, deploy.yml ,.etc
* the regular template of Azure Pipeline
  + **trigger # trigger from Github/ scheduled one**
  + **pool # the vmImage**
  + **variables #the variables for a pipelines**
  + **steps: # series of tasks that has to be performed**
    - **tasks #**

**FAQs**

* What are Azure pipelines?

Ans: Azure pipelines are the series of tasks as in packaging, building, publishing and so on and so forth with the help of Agents, Agents are underlying softwares that help in execution of pipelines and automations.

Azure pipelines provided by Microsoft is

Pipeline as a code which are written in YAML.

* + 1. Trigger: # github push or scheduled one
    2. Pull: #vmImage on which pipeline has to be run
    3. Variables: #variables or variables groups
    4. Steps:
       1. Task1
       2. Task2

**ARM Templates**

* **ARM templates are way to define the infrastructure and configuration of Azure in a declarative way**.
* ARM templates are written in **JSON** and can be used in create, deploy, and manage Azure Resources including Virtual Machines, storage accounts.
* Components of ARM templates are
  + ***Resources***: Contains the resources that we want to create and update.
  + ***Parameters***: To pass values into ARM templates when it is deployed.
  + ***Variables***: Store and reuse the key bits in immutable string form within the ARM templates.
  + ***Outputs:*** Allow to expose values from ARM template to other parts of the infrastructure or to users to use.
* Can be deployed **using Azure portal, Azure CLI, Azure PowerShell**, or through a CICD pipeline.

**FAQS**

1. What are ARM templates and how do they help in automation of Azure resources?

**🡪 These are Json files that define Azure resources and their configuration settings. They help automate the deployment of Azure resources by providing a declarative way to manage infra-structure as a code.**

1. Structure of ARM templates? And key components

🡪 Structure of ARM templates are as

* 1. **$schema and content Version:** definingschema and the version
  2. **Parameters:** elements defining input parameters of the template
  3. **Variables:** elements defining reusable expressions and values
  4. **Resources:** elements defining Azure resources that must be used
  5. **Outputs:** elements defining outputs values of the template

Infrastructure as a code for Azure.

* The regular creation of any cloud infrastructure is very lengthy if the architecture is similar and creating same thing twice can be inconvenient to use.
* Another option is using PowerShell using Service Management API, where is each apart is treated separately I n the Azure portal, if any one of them fails all the serialized codes will fail.
* ARM templates by Azure are the best use case for above case scenario.
* ARM (Azure Resource Manager) can deploy, update, or delete all the resources at a time in a single coordinated operation.
* It’s a json file that defines one or more resources to deploy to a resources group.
* Supports Parallelizing in the deployment
* We can also serialize deployment as per dependencies

Codes and structure

* + “$schema”

**Terraform: Infrastructure Provisioning tool**

Automates and manage infrastructure

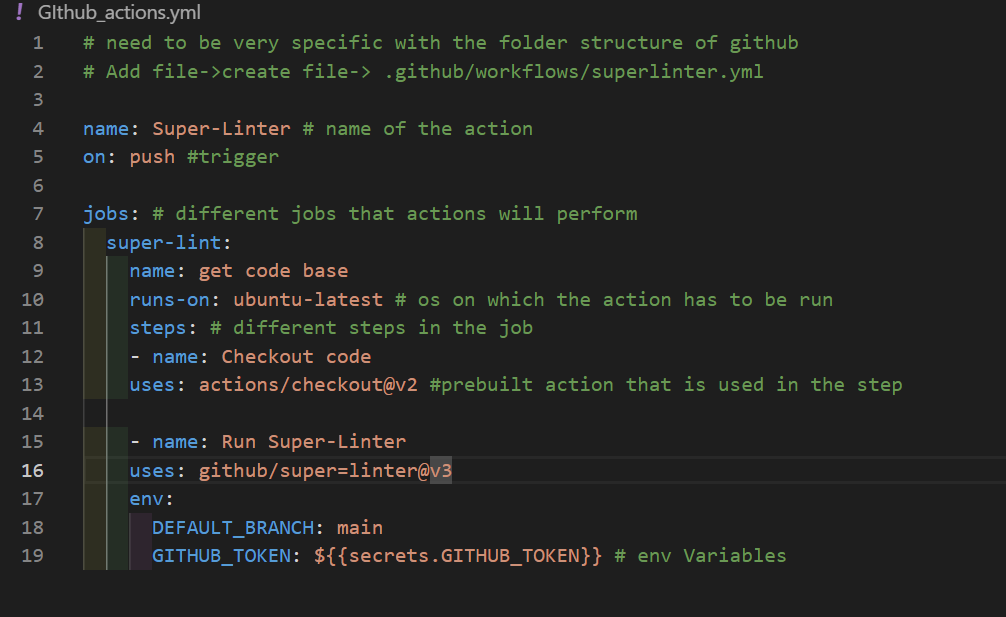
* **Terraform is an infrastructure provisioning tool, whereas Ansible is mainly a configuration tool where we configure an infrastructure as in installing updates, deploy applications, etc**.
* Compares the current state vs desired state (provisioned through config files)
* Json code
  + Provider “azure” {
  + Version = “~> 2.0”
  + Region = “us-east-1”
  + }
  + resource “azure\_Vnet” example” {
  + cidr\_block = “10.0.0.0/16”}

**Imperative Vs Declarative**

* Imperative:
* update the application
* remove 2 servers
* add permissions, etc.
* Declarative (Declarative)
* Just compares the current state of Cloud to the desired state

**GitHub Actions Workflow**

**These are YAML file which is, yet another CICD platform that allows to automate the build, test, deployment and creating workflows.**

****