Azure DevOps

1.Azure Devop

2.Sonar Cloud

1.Deployment Group: -

- It is a group target machine which are deploying my artifact
- Deployment groups represent the physical environments; for example, "**Dev**", "**Test**", or "**Production**" environment, like an **agent pool**.
- Deployment groups are only available with **Classic release pipelines** and are different from deployment **jobs**

I will create

1. Dev
2. QA
3. UAT or Production

System Prerequisites:

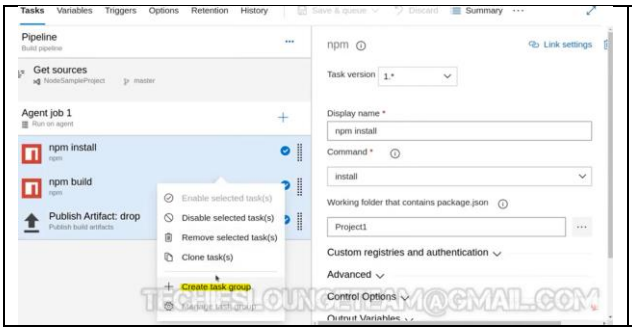
Give name --> select a type of targe to register

1. windows~ [login to VM]
2. Linux

- Registration Script [PowerShell]
- use a personal access token in the script for authentication
- Copy coli board

2.Task Group -

1. Re-User same **task group** in multiple pipelines
2. While creating classic pipelines tasks, we can merge multiples task to one task, this we can use multiple pipelines and if we made changes in task group, it would trigger all pipelines



[Task Groups in Azure Pipelines with Parameters - YouTube](https://www.youtube.com/watch?v=lmm5JGrzdxo)
[https://www.youtube.com/watch?v=lmm5JGrzdxo]

3.Library:

The library contains two types of assets:

1. **Variable Groups**
   We can pass the key and values and we can pass the values from the key vault using subscriptions[https://www.youtube.com/watch?v=1axCfFv7jmw]
   In yaml pipeline we are passing azure cli task to get the secret from the key vault
2. **Secure Files**
   Upload the secure file like ansible playbook and create a task like downloading the file and run the task [https://www.youtube.com/watch?v=ZFIzMzYv-Ho]

Agents:

1.Microsoft-hosted agents

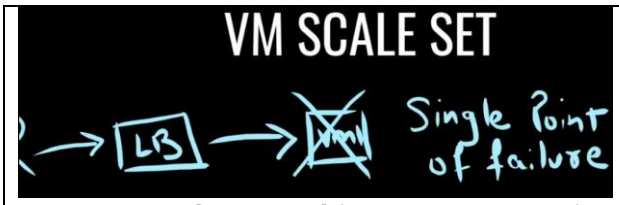
2.Self-Hosted Agents:

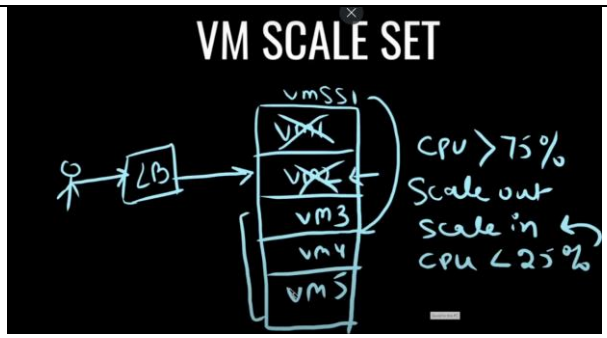
Way to set-up Self hosted agents:-

1. Azure VM [Login and install all commands]
2. Azure VMSS [Create VMSS and Run the bash script on create VM's]
3. Container e.t.c

- **Convenience**: You don't need to worry about **provisioning**, **managing**, or **maintaining** the agent infrastructure
- **Scaling**: Azure-hosted agents can automatically **scale up** or **down** based on the demand of your pipelines. This can be **cost-effective** because you only pay for the time the **agents are active** and **performing tasks**.

How VMSS will work is Load Balancer:

<table>
<tr>
<td> 

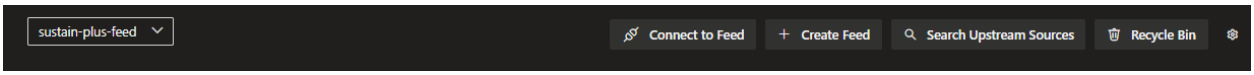
User --> LB --> [Only VM1] (VM not responding) --> user will not get the reponse is called single point of failure</td>
<td> 

Scale Out: CPU > 75% utilization, increase the VM's
Scale in: CPU < 25% utilization remove the VM's</td>
</tr>
</table>

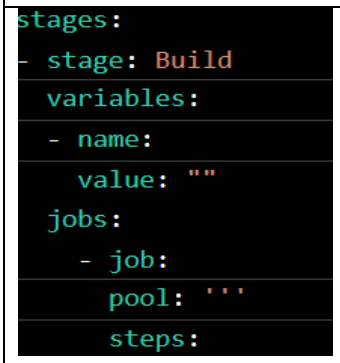
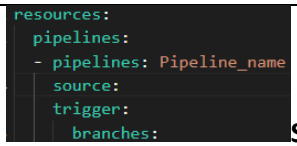
- *Devops --> Project Settings --> Agent Pool --> Add Pool [New and Existing] --> Pool Type [Self Hosted or VMSS] -->Azure Subscription --> Select VMSS --> Add the name --> Create*

Artifacts:

Artifacts store the below the projects



Java	*Artifacts can be POMs (project object models), JARs, or other files.*
.Net	*.ddl*
PHP	*.zip*

Yaml structure	Triggers [1 to 2 repo]	Dependency pipeline
		 **Schedules** 

```
stages:
- stage: Build
  variables:
  - name:
    value: ""
  jobs:
    - job:
      pool: ''
      steps:
```

```
resources:
  repositories:
  - repository: repo_name
    type: git
    name: project/repo_name
    ref: develop1
    trigger:
      branches:
        include:
        - 'develop1'
      paths:
        include:
        - path_name
```

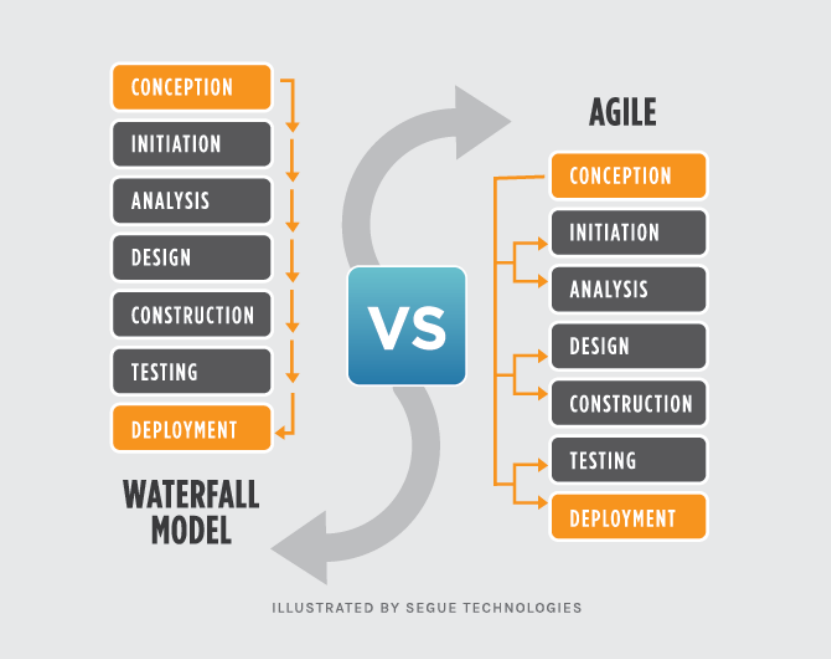
```
resources:
  pipelines:
  - pipelines: Pipeline_name
    source:
    trigger:
      branches:
```

```
schedules:
- cron: '0 8 * * *'
  displayName: Run 8AM
  branches:
    include:
    - develop
  always: true
```

Deployment Strategy:

1. **Blue Green**
2. **Rolling**
3. **A/B**
4. **Canary**

<u>Agile Vs Waterfall</u>



++

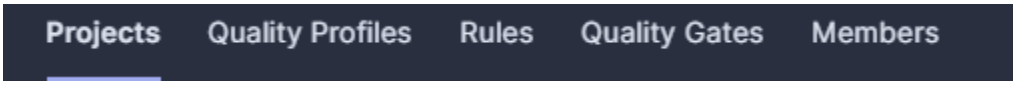
2.

Sonar Cloud

- Sonar's static application security testing (SAST) engine detects security **vulnerabilities**
- It can be eliminated before you **build and test** your application
- This tool supports **25+** main **programming languages and** can also be utilized with **multiple plugins.**

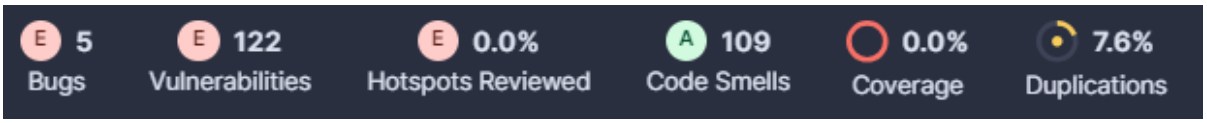
Software Quality	Sonar Cloud Reports
1. **Reliability [Bug]** 2. **Maintainability [Code smell]** 3. **Security [Security hotspot]**	1. **Vulnerabilities** 2. **Bugs** 3. **static code analysis** 4. **code smells** 5. **code duplications** 6. **code coverage** 7. **code complexity** 8. **unit testing**

- **SonarCloud.io --> select organization name and below the screen shots**



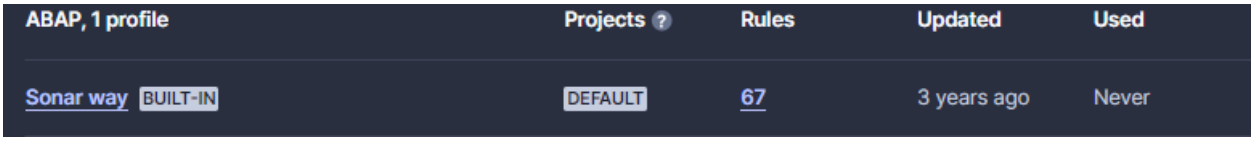
Projects: - select project --> Each Micro services have project -> Check the below the result's will get the which All PR details,

- [Yaml code Mention Project-Name and Project Key]
- [SonarPreparaion.yaml and SonarReport.yaml]

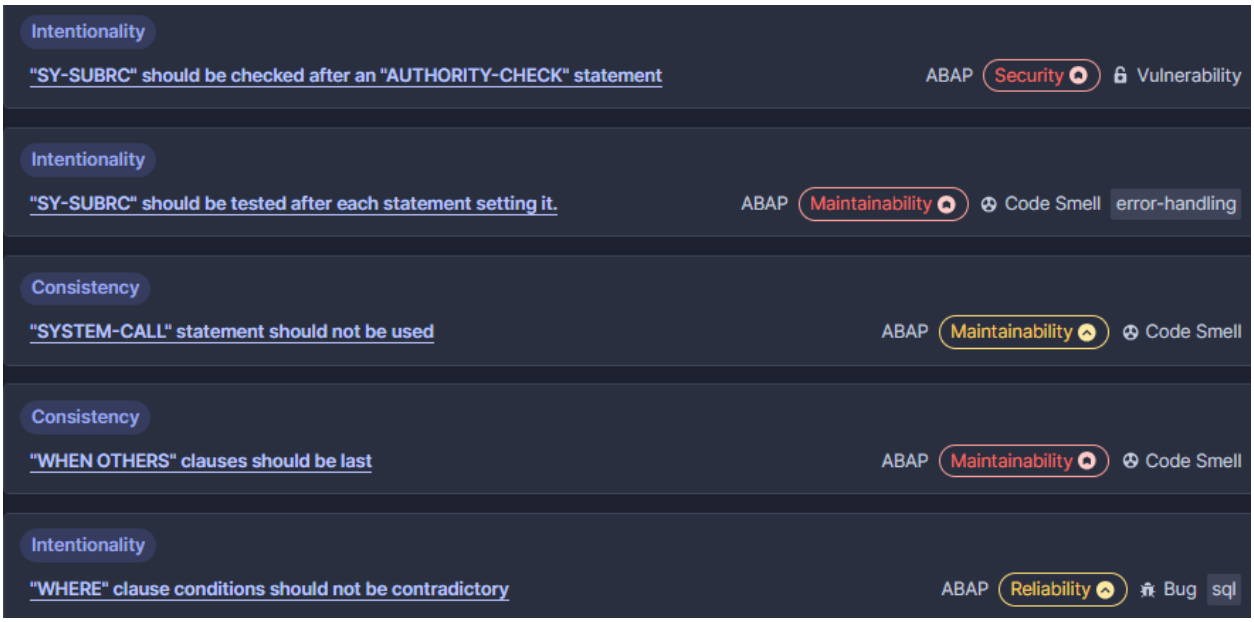


1.Explain SonarQube quality profile?

Quality Profiles: Many of quality profiles added and If we click on rules



Rule: It will show what are the rules we implemented



2.These are the sample Rules

Prathusha?

3.Explain the difference between SonarLint and SonarCloud?

SonarCloud:

- the is main server that performs complete analysis
- The analysis is to provide your code base a 360 Â° view of the quality.

SonarLint:

- this is available only in the **IDE** (Visual Studio and Eclipse)
- when you enter your code, fix quality issues, Like a spell checker
- SonarLint is an **agent** which connects with SonarCloud and executes the analysis remotely.

4.How can you create reports in SonarQube?

For Angular Code, I am running the CI Seduced pipeline, once in a day, it will generate all report and sent to the respective team

5.What is the difference between Sonar Runner and Sonar Scanner?

“Runner” is the old name for "Scanner".

6.How is the architecture of the SonarQube?

1. Source Code
2. Sonar Scanner
3. Sonar Analyzer
4. SonarQube Database or send the report

++

Docker

Docker File:

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

1.various Docker components

1. Docker Client
2. Docker Host
3. Registry

2.What’s the difference between virtualization and containerization?

- Virtualization - abstract version of a **physical machine**
- containerization - abstract version of an **application**.

3.What do you know about the Docker system prune?

It’s a command used to remove all stopped containers

$ docker system prune

4.How do Docker daemon and the Docker client communicate with each other?

Docker client:

- The Docker client is the command-line interface (CLI) or graphical user
- The client sends instructions to the **daemon**, which then performs the requested actions (e.g., pulling an image, creating a container)

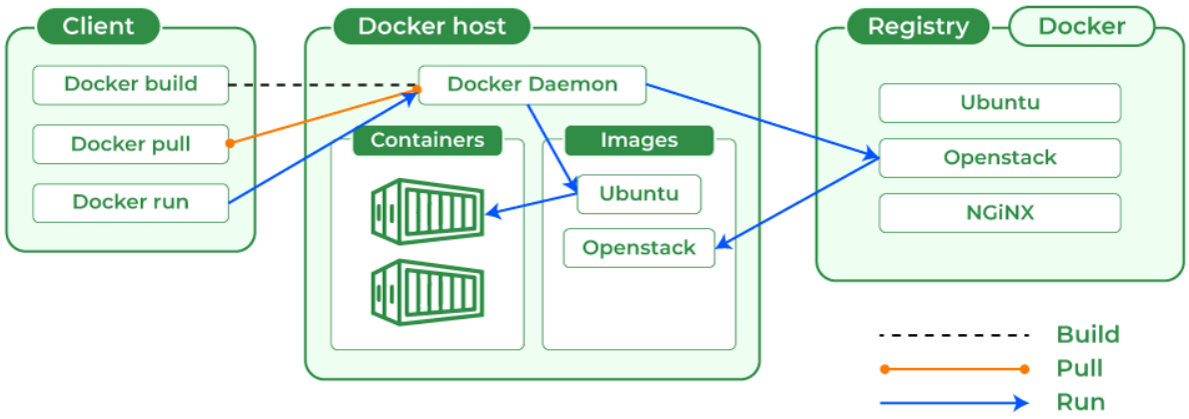
Docker daemon [Docker server]

- The Docker daemon exposes a REST API (called the Docker Engine API)
- The daemon manages containers, images, networks, and volumes. It ensures that containers are isolated and secure.

Docker Compose.

define your application's **services, networks, and volumes in a docker-compose.yml file**. This file specifies how **one or more containers** that make up your application are configured.

Docker Architecture



FROM	specify Docker Image Name
MAINTAINE	person who creates the Docker Image
CMD	Execute Commands when your Docker Image is deployed.
RUN	executes the command when you are building Image
LABEL	specify metadata information of Docker Image.
EXPOSE	specify Network port for Docker container
ENV	Environment Variables with key and value.
ARG	set Environment variables with key and value during the image build
ADD	Copies a file and directory from your host to Docker image
COPY	Copies a file or directory from your host to Docker image
ENTRYPOINT	specifies a commands that will executes when the Docker container starts
VOLUME	create or mount volume to docker container.
USER	user name
WORKDIR	set the working directory.
STOPSIGNAl	system call signal that will be sent to the container to exit
SHELL	used to set the default shell.
HEALTHCHECK	Check container health by running a command inside the container

Build & Upload to ACR: -

docker load --input $(System.DefaultWorkingDirectory)/drop/ImageName.tar

docker build -t image_name:latest .

docker login Con_registry --username registry_name --password $(Password)

docker tag ImageName:latest Container_registry.azurecr.io/ImageName:latest

docker push Container_registry.azurecr.io/ImageName:latest

Commands

docker run --name Cont_name -p -it -d 7000:8080 tomme

docker rename old_name new_name

docker exec -it container_name /bin/bash

docker ps

docker ps -all or -a

docker start or stop or kill prune or restart or pause or unpause or inspect

docker stop -f $(docker ps -a)

docker start -f $(docker ps -a)

docker rm -f container_id

docker rmi -f image_name

docker image rmi $(docker images -q)

Note : -t is create contaienr and enter in to the contianer

docker volume *ls*

docker volume create <volume_name>

docker volume inspect<volume_name>

docker volume rm <volume_name>

docker volume prune

++

Bash/linux

Bash Interview Questions

3	**touch**	Create Empty files
4	**cat > filename**	create new file and writes the content and ctrl+D to save the content into file.
5	**cat file1 file2 > file3**	Creates new file and content of both files will be copied to new file
6	**cat**	Display content of file
7	**ls**	List the content of Directory
8	**ls *.txt**	list all the files with given extension
9	**pwd**	current working directory
10	**cp**	Copy a file or directory
11	**mv**	moves a file or directory
12	**head**	Display first 10 Lines of a file
13	**tail**	Display last 10 lines of a file
14	**tac**	Display file content Lines in reverse
15	**more**	similar to cat and here we can display large content by using ENTER , SPACEBAR
15	**more**	similar to cat and here we can display large content by using ENTER , SPACEBAR
16	**id**	Display id of user / group
17	**clear**	clear the screen
18	**vi**	text editor to write programs of text
19	**grep**	Filter to search given pattern in the file content
20	**diff**	Compares the content of two different files
21	**ping**	Check the connectivity status of server
22	**history**	Review all the commands which you have entered
23	**hostname**	Display hostname
24	**hostname -i**	Display host ip
25	**chmod**	Change the user/group permissions to access file
26	**nl**	Display the Line Numbers
27	**wc**	Given number of Lines , Words and Characters available in the file content
28	**uniq**	Remove duplicates of file content / it can remove only continous duplicates
29	**rmdir**	Removes the specified directory (directory should be empty)
30	**rm**	Remove file

Cd dir1 Cd ~ (till symbol) cd .. Cd ../.. cd ../../..	Mv to Dir Home Dir dir back double back double double
Cat f1 f2 f3 > f4 cat f1 cat > f1 cat >> f2	All files merge into the one file
Ls –R or tree ls –a or ls –ltr	List of files in sub dir Hidden file
cp file1 dir_path cp file_path tar_dir_path cp -R	Copy f1 to dir path Copy source f1 to tar f2
Rm f1 or rm f1 f2 rm –rf rmdir dir1	Remove file Force fully remove file remove folder

Git

Bash Interview Questions

Kubernetes

Bash Interview Questions

Terrafeom

Bash Interview Questions

- Terraform is one of the **most** popular IAC tools used by every cloud engineer.
- open-source IAC tool created by HashiCorp.
- It is used to create, update, delete and version your infrastructure on multiple cloud platforms.

  Terrform support’s
- AWS
- GCP
- Azure
- IBM Cloud

Terraform backend is the platform where the Terraform State **Snapshots** are stored
Private Module Registry Terraform Cloud feature allows us to share Terraform modules across our organization

What are the most useful Terraform commands?

Some of the most useful Terraform commands are:

- terraform **init** - initializes the current directory
- terraform **refresh** - refreshes the state file
- terraform **output** - views Terraform outputs
- terraform **apply** - applies the Terraform code and builds stuff
- terraform **destroy** - destroys what has been built by Terraform
- terraform **graph** - creates a DOT-formatted graph
- terraform **plan** - a dry run to see what Terraform will do