Question Bank – Introduction to DevOps

UNIT - 3

1. **Explain Dockers. Explain why to use Dockers.**

- Docker is a containerization tool.
- It allows you to isolate an application from its host system so that the application becomes portable.
- And the code tested on a developer's workstation can be deployed to production with fewer concerns about execution runtime dependencies.
- A container is a system that embeds an application and its dependencies.
- Unlike a VM, a container contains only a light operating system with only the elements required for the OS, such as system libraries, binaries, and code uuuuuudependencies.
- The principal difference between VMs and containers is that each VM that is hosted on a hypervisor contains a complete OS.
- It is therefore completely independent of the guest OS that is on the hypervisor.
- Containers don't contain a complete OS – only a few binaries—but they are dependent on the guest OS, using its resources (CPU, RAM, and network).

Why use Dockers:

- **Easy to install and run software** without worrying about setup or dependencies.
- **Developers use Docker to eliminate machine problems**, i.e. "but the code is working on my laptop." when working on code together with co-workers.
- Operators use Docker to **run and manage apps in isolated containers** for better compute density.
- Enterprises use Docker to **securely build agile software delivery pipelines** to ship new application features faster and more securely.
- Since docker is not only used for the deployment, but it is also **a great platform for developmen**t, and helps in increasing customer's satisfaction.

2. **Advantages and disadvantages of Dockers.**

Advantages

- It **runs** the container in **seconds instead of minutes.**
- It uses **less memory.**
- It provides **lightweight** virtualization.
- It **does not require a full operating** system to run applications.
- It uses application dependencies to reduce the risk.
- Docker allows you to use a remote repository to share your container with others.
- It provides a continuous deployment and testing environment.

Disadvantages of Dockers:

- It **increases complexity** due to an additional layer.
- In Docker, it is **difficult to manage large amounts of containers**.
- Some features such as container self -registration, container self-inspects, copying files from host to the container, and more are missing in Docker.
- Docker is **not a good solution for applications that require a rich graphical interface**.
- Docker does not **provide cross-platform compatibility** means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

3**. List and explain the different features of Dockers.**

- Easily Scalable
- Easy and Faster Configuration
- Ability to Reduce the Size
- Increase productivity
- Reduced Infrastructure and Maintenance Costs
- Application Isolation
- Swarm
- Routing Mesh
- Services
- Security Management

OR

Containerization: Docker allows developers to package their applications and dependencies into self-contained units called containers, which can be easily deployed and run on any host machine that has the Docker runtime installed. This makes it easier to manage the dependencies and runtime environments for different applications, and ensures that applications will run consistently regardless of the host environment.

Isolation: Docker containers are isolated from each other and the host system, which means that each container has its own file system, network, and process space. This isolation ensures that the resources of one container are not impacted by the actions of other containers, and makes it easier to troubleshoot issues that may arise.

Portability: Docker images and containers are portable, which means they can be easily moved between different host machines and environments. This makes it easier to deploy applications in different environments, such as development, staging, and production.

Scalability: Docker makes it easy to scale applications by adding more containers to a cluster. This allows developers to quickly increase the capacity of an application to meet demand, and makes it easier to scale applications up or down as needed.

Ease of use: Docker provides a simple and easy-to-use command-line interface (CLI) for interacting with containers and images. This makes it easy for developers to get started with Docker, and allows them to automate many common tasks.

Community: Docker has a large and active community of users and developers, which

means there is a wealth of documentation, examples, and support available online. This makes it easier for developers to get help with any issues they may encounter while using Docker

4. Describe

i. Different components of Dockers

There are four components of docker:

- Docker client and server
  - This is a command-line-instructed solution by using the terminal to issue commands from the Docker client to the Docker daemon.
  - The communication between the Docker client and the Docker host is via a REST API.
  - The Docker daemon itself is actually a server that interacts with the 10 operating systems and performs services.
  - Docker daemon constantly listens across the REST API to see if it needs to perform any specific requests.
  - To trigger and start the whole process, use the Dockered command within the Docker daemon. And it will start all of the performances.
  - Then you have a Docker host, which lets you run the Docker daemon and registry.
- Docker image
  - A Docker image is a template that contains instructions for the Docker
  - container.
  - That template is written in a YAML, which stands for Yet Another Markup Language.
  - The Docker image is hosted as a file in the Docker registry.
  - The image has several key layers, and each layer depends on the layer below it.
  - Image layers are created by executing each command in the Dockerfile and are in the read-only format.
  - Start with base layer, which will typically have base image and base operating system.
  - And then have a layer of dependencies above that.
  - These then comprise the instructions in a read-only file that would become your Dockerfile.
  - There are four layers of instructions: From, Pull, Run
  - and CMD.
  - The From command creates a layer based on Ubuntu, and then add files from the Docker repository to the base command of that base layer.
  - Pull: Adds files from your Docker repository.
  - Run: Build your container.
  - CMD: Specifies which command to run within the container.
- Docker registry
  - The Docker registry is used to host various types of images and distribute the images from.
  - The repository itself is just a collection of Docker images, which are built on instructions written in YAML and are very easily stored and shared.
  - Give name tags to the Docker images so that it's easy to find and share

them within the Docker registry.
    - One way to start managing a registry is to use the publicly accessible Docker hub registry, which is available to anybody.
    - You can also create your own registry for your own use internally.
    - The registry that you create internally can have both public and private images that you create.
    - The commands you would use to connect the registry are Push and Pull.
    - Use the Push command to push a new container environment you've created from your local manager node to the Docker registry.
    - Use a Pull command to retrieve new clients (Docker image) created from the Docker registry.
    - A Pull command pulls and retrieves a Docker image from the Docker registry.
    - A Push command allows you to take a new command that you've created and push it to the registry, whether it's Docker hub or your own private registry.
- Docker container
    - The Docker container is an executable package of applications and its dependencies bundled together.
    - It gives all the instructions for the solution you're looking to run.
    - It is lightweight due to the built-in structural redundancy.
    - The container is also portable.
    - Another benefit is that it runs completely in isolation.

ii. Advanced components of Dockers

- **Docker Compose:**
    - It is designed for **running multiple containers as a single service**.
    - It does so by running **each container in isolation** but **allowing** the containers to **interact with one another.**
    - Write the compose environments using YAML.
    - Use Docker Compose if you are running an Apache server with a single database and you need to create additional containers to run additional services without having to start each one separately.
    - You would write a set of files using Docker compose to do that.
- **Docker Swarm:**
    - It is a service for containers that allows IT administrators and developers **to create and manage a cluster of swarm nodes** within the Docker platform.
    - **Each node of Docker swarm is a Docker daemon**, and all Docker daemons interact using the Docker API.
    - A swarm consists of two types of nodes: a manager node and a worker node.
    - A manager node maintains cluster management tasks.
    - Worker nodes receive and execute tasks from the manager node.

5. Explain the Docker Elements.

- **Dockerfiles** - template to create an image.
- **Volumes** - physical storage for containers
  Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

  - Volumes are easier to back up or migrate than bind mounts.
  - You can manage volumes using Docker CLI commands or the Docker API.
  - Volumes work on both Linux and Windows containers.
  - Volumes can be more safely shared among multiple containers.

- **Containers** - executable version of an image

6. **Describe the Dockerfile Instructions used in writing Docker files.**

https://www.learnitguide.net/2018/06/dockerfile-explained-with-examples.html

FROM
FROM instruction used to specify the valid docker image name.
BUILD
Build an image from a dockerfile
ADD and COPY
instruction is used to copy files, directories and remote URL files to the destination (docker container) within the filesystem of the Docker Images.
RUN
RUN instruction is used to execute any commands on top of the current image and this will create a new layer.
CMD
set a command to be executed when running a container
ENTRYPOINT
ENTRYPOINT instruction is used to configure and run a container as an executable
VOLUME
VOLUME instruction is used to create or mount a volume to the docker container from the docker host filesystem
ENV
set environment variables with key and value

7. List the steps involved in building and running a container on a local machine.

The execution of Docker is performed by these different operations:

1. Building a Docker image from a Dockerfile (docker build -t demobook:v1)

2. Instantiating a new container locally from this image (docker run -d --name demoapp -p 8080:80 demobook:v1)

3. Testing our locally containerized application

8. What are Kubernetes

Kubernetes is a popular open source platform for container orchestration that is, for the management of applications built out of multiple, largely self-contained runtimes called containers.

9. Explain

i. Key objects of Kubernetes.

Pods

- Pods are the smallest unit of deployment in Kubernetes. Contains a **group of containers**

Services

- **A group of pods**. Services provide a way to expose applications running in pods. Provides permanent IP address for pods.

Volumes

- Volumes are objects whose purpose is to **provide storage to pods**

Replication controller

- Used to ensure that a specified number of **replicas of a pod** are always running.

Namespaces

- The purpose of the Namespace object is to act as a separator of resources in the cluster.

Deployments

- A deployment is a declarative way to describe the desired state of a group of pods

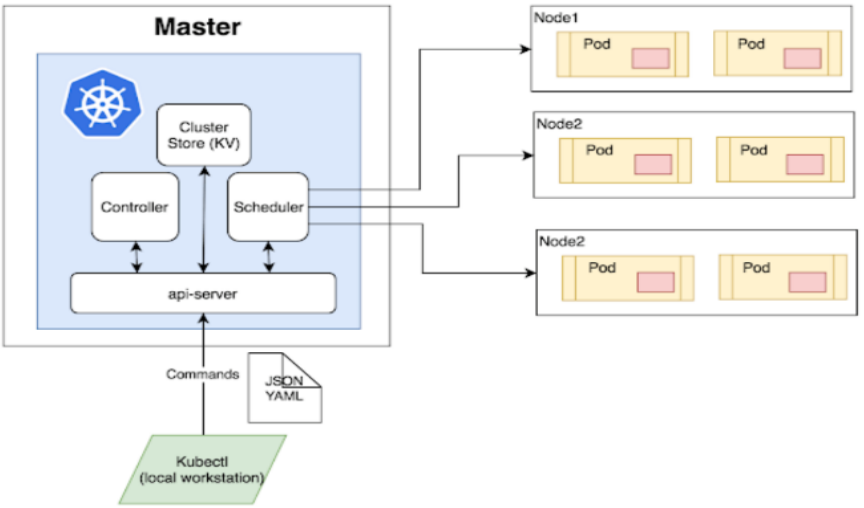
ii. Features of Kubernetes.

- Automated Scheduling
- Self-Healing Capabilities
- Automated rollouts & rollback
- Horizontal Scaling & Load Balancing
- Offers environment consistency for development, testing, and production
- Infrastructure is loosely coupled to each component and can act as a separate unit.
- Provides a higher density of resource utilization
- Offers enterprise-ready features
- Application-centric management
- Auto-scalable infrastructure
- You can create predictable infrastructure

(OR)

- Container orchestration
- Storage orchestration
- Self-healing ability
- Load Balancing
- Scalability
- Configuration management

iii. Kubernetes Architecture.



1. Master Node:

- The master node is the first and most vital component which is responsible for the management of Kubernetes cluster.
- It is the entry point for all kinds of administrative tasks.
- There might be more than one master node in the cluster to check for fault tolerance.

2. API Server:

- The API server acts as an entry point for all the REST commands used for controlling the cluster.

3. Scheduler:
   - The scheduler schedules the tasks to the slave node.
   - It stores the resource usage information for every slave node.
   - It is responsible for distributing the workload.
4. Worker/Slave nodes:
   - Worker nodes are another essential component which contains all the required services to manage the networking between the containers, communicate with the master node, which allows you to assign resources to the scheduled containers.
5. Kubelet:
   - This gets the configuration of a Pod from the API server and ensures that the described containers are up and running.

10. With the help of a Kubernetes deployment YAML specification file, deploy the instance of the Docker image with a web application.

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: webapp
spec:
 selector:
    matchLabels:
      app: webapp
 replicas: 2
 template:
    metadata:
     labels:
       app: webapp
   spec:
      containers:
      - name: demobookk8s
        image: mikaelkrief/demobook:latest
        ports:
        - containerPort: 80
```

11. Explain the Azure Kubernetes Service.

AKS is an Azure service that allows us to create and manage a real Kubernetes cluster as a managed service. The advantage of this managed Kubernetes cluster is that we don't have to worry about its hardware installation, and that the management of the master part is done entirely by Azure when the nodes are installed on VMs. The use of this service is free; what is charged is the cost of the VMs on which the nodes are installed.

12. What are the advantages and disadvantages of Azure Kubernetes Service?

Advantages

- Ready to use
- Integrated monitoring services
- Cost effective
- Scalable
- Easy to understand and use
- Very easy to scale
- If we have an Azure subscription and we want to use Kubernetes, it's intuitive and quick to install.
- Using the kubectl tool does not require any changes compared to a local Kubernetes

Disadvantages

- Security is not very effective
- User defined routes can be hard to manage as your cluster gets bigger
- Since pods don't have IP addresses, there is additional latency when communicating among nodes
- You can only have one AKS cluster per subnet.

OR

- Limited to Azure: AKS is only available on the Azure platform, so if you are not using Azure, it may not be a suitable option.
- Potentially slower updates: Because AKS is a managed service, you may not have immediate access to the latest Kubernetes features and updates.

UNIT-4 QB

1. What is a Postman tool? Explain the uses of Postman.

- Postman is an API platform for developers to design, build, test and iterate their APIs.
- Postman is a free client tool in a graphical format that can be installed on any type of OS.
- Its role is to test APIs through requests.
- Allows to dynamize API tests through the use of variables and the implementation of environments.
- Postman is famous for its ease of use, but also for the advanced features that it offers.

Uses of postman:

1. Store information in an organized way
2. Effective integration with systems
3. Comfortable business logic implementation:
4. On-the-fly testing
5. Useful scripts
6. Tons of resources
7. Easy move to code repositories
8. Easy Creation of test suites
9. Capable of running tests in different environments
10. Data storage for Multiple tests.

    Or

    - Accessibility
    - Use Collections
    - Test development
    - Automation Testing:
    - Creating Environments
    - Debugging
    - Collaboration
    - Continuous integration

Uses :

Testing APIs: allows you to make HTTP requests to an API and view the response, which is useful for testing and debugging APIs. You can use Postman to test different endpoints, headers, and payloads to ensure that your API is functioning as expected.

Collaboration: Postman provides a variety of tools for collaborating with team

members, including the ability to share collections of API requests and responses. This can be useful for working on APIs as a team and sharing knowledge about how an API works.

Documentation: Postman can generate documentation for your API based on the requests and responses that you have captured. This can be useful for providing documentation to developers who are using your API.

Automation: Postman allows you to use scripts to automate testing and other tasks. This can be useful for reducing the time and effort required to test and develop APIs.

2. What is a request in Postman? Explain the main parameters of a request.

1. In Postman, the object that contains the **properties of the API** to be tested is called a request.
2. This request contains the configuration of the API itself, but it also contains the tests that are to be performed to check it's functioning properly.
3. The main parameters of a request are as follows:
    1. The URL of the API
    2. Its method: Get/POST/DELETE/PATCH
    3. Its authentication properties
    4. Its query string keys and its body request
    5. The tests that are to be performed before or after the execution of the API

3. What are the different variable scopes in Postman?

Global Variables: Enable you to access data between collections, requests, test scripts, and environments.

These are available throughout a workspace.

Have the broadest scope available in Postman and are well-suited for testing and prototyping.

Collection Variables: Available throughout the requests in a collection and are independent of environments.

Collection variables don't change based on the selected environment.

Collection variables are suitable if you're using a single environment, for example for auth or URL details.

Environment Variables: Enable you to scope your work to different environments, for example local development versus testing or production.

One environment can be active at a time. If you have a single environment, using collection variables can be more efficient, but environments enable you to specify role-based access levels.

Data Variables:

These come from external CSV and JSON files to define data sets you can use when running collections with Newman or the Collection Runner.

Data variables have current values, which don't persist beyond request or collection runs.

Local Variables:

These are temporary variables that are accessed in your request scripts.

These variable values are scoped to a single request or collection run, and are no longer available when the run is complete.

Suitable if you need a value to override all other variable scopes but don't want the value to persist once execution has ended.

4. Write the codes in Postman to perform the following tests in the Tests tab i. That the return code of the request is 200

```
pm.test("Status code is 200", function () {
 pm.response.to.have.status(200);
});
```

ii. That the response time of the request is less than 400 ms

```
pm.test("Response time is less than 400ms", function () {
 pm.expect(pm.response.responseTime).to.be.below(400);
});
```

iii. That the answer is not an empty JSON

```
pm.test("Json response is not empty", function (){
 pm.expect(pm.response).to.be.json;
});
```

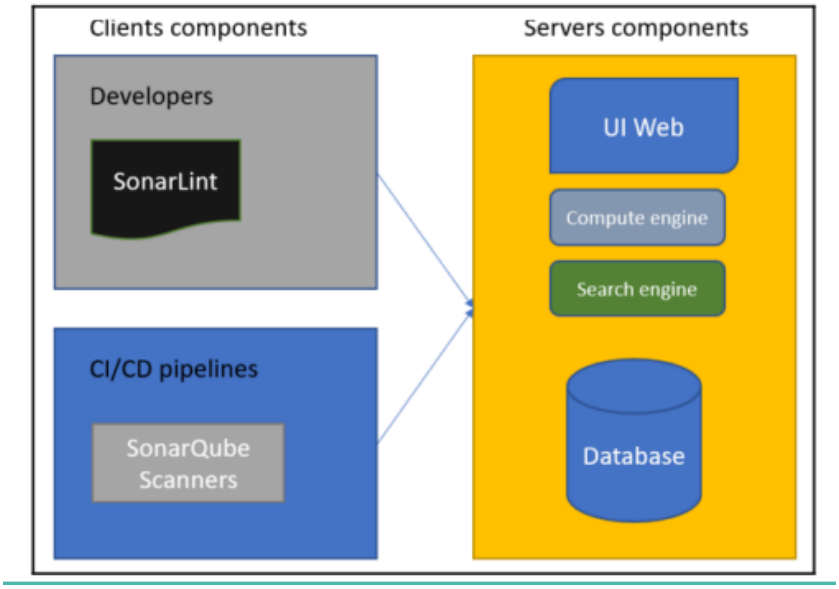
iv. That the return JSON response contains the userId property, which is equal to 1

```
pm.test("Json response userId eq 1", function (){
 var jsonRes = pm.response.json();
 pm.expect(jsonRes.userId).to.eq(1);
});
```

5. Explain the tool used to automate tests that are written in Postman.

1. The tool used to automate tests that are written in Postman is called Newman. Newman is a command-line tool that allows you to run Postman collections from the command line. It is useful for automating tests and integrating Postman into your continuous integration and delivery (CI/CD) pipeline.
2. To use Newman, you first need to export your Postman collection and environment as a JSON file. You can then use the newman command to run the collection, passing in the path to the JSON file as an argument. You can also specify various options and arguments to customize the behavior of Newman, such as setting environment variables and ignoring certain test failures.
3. Newman has a number of features that make it useful for automating tests, including the ability to run tests in parallel, generate reports, and integrate with other tools such as Jenkins and Azure DevOps.
4. Overall, **Newman is a useful tool for automating tests that are written in Postman and integrating them into your CI/CD pipeline.** It provides a simple and efficient way to run your Postman collections and ensure that your APIs are functioning as expected.

6. Describe the architecture of SonarQube.



- SonarQube is a client-server tool, and its architecture is

composed of artifacts on the server side and also on the client side

- The components that make up SonarQube on the server side are as follows:
    1. A SQL Server, MySQL, Oracle, or PostgreSQL database that contains all the analysis data.
    2. A web application that displays dashboards.
    3. The compute engine, which is in charge of retrieving the analysis and processes.
    4. It puts them in the database.
    5. A search engine built with Elasticsearch.

- The client-side components are as follows:
    1. The scanner, which scans the source code of the applications and sends the data to the compute engine.
    2. The scanner is usually installed on the build agents that are used to execute CI/CD pipelines.
    3. SonarLint is a tool that's installed on developers' workstations for real time analysis.

7. Describe the real-time analysis with SonarLint.

1. SonarLint is a static analysis tool that is used to identify code quality issues in real-time as you are writing code. It integrates with popular code editors and IDEs, such as Visual Studio and Eclipse, and provides real-time feedback on code issues such as bugs, vulnerabilities, and style issues.
2. One of the main benefits of using SonarLint is that it allows you to identify and fix code issues as you are writing code, rather than waiting until later in the development process. This can save time and effort by reducing the need for manual code review and testing.
3. SonarLint works by analyzing your code and comparing it to a set of predefined rules that are designed to identify code quality issues. When it finds an issue, it will highlight the affected code and provide a description of the issue and a suggested fix. You can then choose to apply the fix or ignore the issue, depending on your needs.
4. Overall, SonarLint is a useful tool for improving code quality and reducing the time

and effort required to identify and fix code issues. It is particularly useful for real-time analysis, as it allows you to identify and fix code issues as you are writing code.

8. Explain

i. Penetration Testing.

- Penetration testing (or pen testing) is a security exercise where a cybersecurity expert attempts to find and exploit vulnerabilities in a computer system.
- The purpose of this simulated attack is to identify any weak spots in a system's defenses which attackers could take advantage of.
- Pen testing involves the attempted breaching of any number of application systems, (e.g., application protocol interfaces (APIs), frontend/backend servers) to uncover vulnerabilities, such as unsanitized inputs that are susceptible to code injection attacks
- Stages of penetration testing are :
    - Planning and reconnaissance
    - Scanning
        - Static analysis
        - Dynamic Analysis
    - Gaining Access
    - Maintaining Access
    - Analysis and WAF configuration

- Methods
    - External
    - Internals
    - Blind
    - Double-Blind
    - Targeted

ii. Performance Testing.

- Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload.
- The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application.
- It is a subset of performancece engineering and is also known as "Perf Testing".
- The focus of Performance Testing is checking a software program's
    - 1. Speed : Determines whether the application responds quickly
    - 2. Scalability : Determines the maximum user load the software application can handle.
    - 3. Stability : Determines if the application is stable under varying loads.

- Types
    - Load Testing

- Stress Testing
- Endurance Testing
- Volume Testing
- Scalability Testing

9. Describe the different metrics used to determine the performance of an application.

The performance of an application is determined by metrics such as the following:

1. Its response time

2. The use of resources (CPU, RAM, and network)

3. The error rate

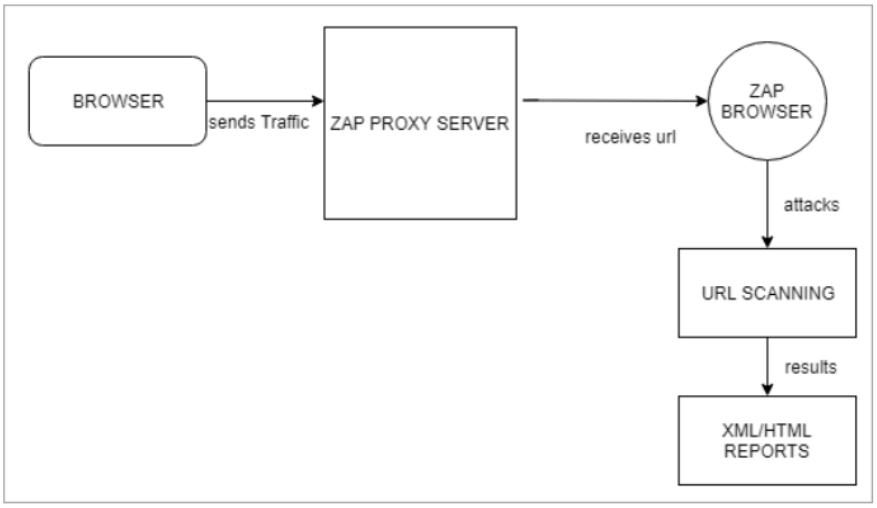
4. The number of requests per second

10. What is the use of the ZAP tool? List and explain the functionalities supported by ZAP.

Use of ZAP

- Zap provides cross-platform i.e. it works across all OS (Linux, Mac, Windows)
- Zap is reusable
- Can generate reports
- Ideal for beginners
- Free tool

11. with a neat diagram.



ZAP is an open-source free tool and is used to perform penetration tests. The

main goal of Zap is to allow easy penetration testing to find the vulnerabilities in web applications.

ZAP creates a proxy server and makes the website traffic pass through the server. The use of auto scanners in ZAP helps to intercept the vulnerabilities on the website.

12. Describe the ZAP Terminologies.

1. Active Scan: Active scanning seeks out potential vulnerabilities using known attacks. It's worth noting that Active Scan can only find certain vulnerabilities.

2. Passive Scan: ZAP by default scans all HTTP requests and responses sent and received from the application

3. Spider: Spider is a crawler, a tool that allows you to discover and map all the links available in the application

4. Fuzzer: This is a technique that involves sending a lot of incorrect or unexpected data to the tested application

5. Alerts: Website vulnerabilities are flagged as high, medium and low alerts

6. Session: Session simply means to navigate through the website to identify

the area of attack.

7. Context: It means a web application or a set of URLs together.

8. API: ZAP provides an API that allows other programs to interact with it.