◆ 1. What is Docker?

☞ **Answer (short & clear for interview):**
Docker is a containerization platform that allows applications to run in isolated environments called **containers**. Unlike virtual machines, containers share the host OS but still keep applications independent.

Purpose:

- Portability (works the same everywhere)
- Consistency (same environment across Dev/QA/Prod)
- Fast (lighter than VMs)

◆ 2. Key Terms

- **Image** → Blueprint/template of an application (like a snapshot).
- **Container** → Running instance of an image.
- **Dockerfile** → Script with instructions to build an image.
- **Docker Hub** → Public registry for storing and sharing Docker images.

☞ **Analogy**:

- Image = Class in Java
- Container = Object (instance)

◆ 3. What is a Dockerfile?

☞ **Interview Answer:**
A Dockerfile is a text file with step-by-step instructions to build a Docker image.

Common Instructions:

```
# Base image (like OS + language)

FROM openjdk:17-jdk-slim
```

```
# Working directory inside container
WORKDIR /app

# Copy project files into container
COPY target/myproject.jar app.jar

# Command to run when container starts
ENTRYPOINT ["java","-jar","app.jar"]
```

◆ 4. Lifecycle of Docker

1. Write a Dockerfile
2. Build image → docker build -t myapp:1.0 .
3. Run container → docker run -d -p 8080:8080 myapp:1.0
4. Upload image → docker push username/myapp:1.0
5. Pull image → docker pull username/myapp:1.0

◆ 5. Docker Commands (Must Know for Interview)

Command	Purpose
docker --version	Check Docker version
docker images	List available images
docker ps	List running containers
docker ps -a	List all containers (including stopped)
docker build -t myapp:1.0 .	Build image from Dockerfile
docker run -d -p 8080:8080 myapp:1.0	Run container

Command	Purpose
docker exec -it <container_id> bash	Access running container
docker stop <container_id>	Stop container
docker rm <container_id>	Remove container
docker rmi <image_id>	Remove image
docker login	Authenticate with Docker Hub
docker push user/myapp:1.0	Push image to Docker Hub
docker pull user/myapp:1.0	Pull image from Docker Hub

◆ 6. Real-time Example (Selenium Test in Docker)

☞ Suppose you have a Selenium automation project with Maven (pom.xml). You package it into a JAR, then use Dockerfile:

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY target/selenium-tests.jar selenium-tests.jar

ENTRYPOINT ["java", "-jar", "selenium-tests.jar"]
```

Steps:

1. mvn clean package → creates selenium-tests.jar in target/
2. docker build -t selenium-tests:1.0 .
3. docker run selenium-tests:1.0

◆ 7. Upload to Docker Hub

1. docker login → enter Docker Hub credentials
2. Tag image:
3. docker tag selenium-tests:1.0 username/selenium-tests:1.0

4. Push image:
5. docker push username/selenium-tests:1.0

👉 Now anyone can pull it:

docker pull username/selenium-tests:1.0

◆ 8. Interview Q&A (Docker)

✅ Q1: Difference between Image and Container?

- Image → Template (blueprint)
- Container → Running instance of an image

✅ Q2: What is the difference between Docker and VM?

- VM → heavy, full OS, slow startup
- Docker → lightweight, shares host OS, faster

✅ Q3: What is a Dockerfile?

- A script with instructions (FROM, COPY, RUN, ENTRYPOINT) to build images

✅ Q4: How do you share Docker images?

- Push to **Docker Hub** or a private registry (like Nexus, Artifactory)

✅ Q5: Can Docker be used for Selenium Grid?

- Yes. Selenium Grid can run inside Docker containers using **Docker Compose**.

✅ Q6: How do you run multiple containers?

- Use **Docker Compose** (docker-compose.yml)

✅ Q7: Why use Docker in Automation Projects?

- Consistent environment across Dev/QA/Prod
- Easy CI/CD integration (Jenkins pipelines)
- No “works on my machine” problem

◆ 9. Docker in Automation (How You Answer in Interview)

☞ *In my Selenium + Cucumber project, we containerized the automation suite using Docker. We created a Dockerfile for the test JAR, built an image, and pushed it to Docker Hub. Our Jenkins pipeline pulls the latest image, runs containers, and executes tests in an isolated environment. This ensures consistent results across all environments.*

◆ 1. What is Jenkins?

☞ **Interview Answer:**
Jenkins is an **open-source CI/CD tool** (Continuous Integration & Continuous Delivery). It automates:

- Building the code
- Running tests (Selenium/Cucumber/TestNG)
- Generating reports
- Deploying applications

📌 **Purpose:**

- Removes manual execution
- Provides faster feedback
- Integrates with Git, Maven, Docker, AWS, Kubernetes, etc.

◆ 2. Why Jenkins in Selenium Project?

- Schedule and trigger test execution (daily, after every code push).
- Run automation on different environments.
- Generate and publish reports (TestNG, Cucumber, Allure).
- Integrate with Docker/Selenium Grid for parallel execution.
- Send email/Slack notifications to stakeholders.

☞ In interview: *We used Jenkins to schedule nightly regression runs, generate Allure reports, and notify the team with email/slack alerts.*

◆ **3. Jenkins Architecture (Simple)**

- **Jenkins Master (Controller)** → Manages jobs, schedules builds.
- **Jenkins Agent (Node/Slave)** → Executes builds on remote machines.
- **Plugins** → Extend Jenkins (Maven plugin, Git plugin, Docker plugin, Allure plugin, etc.).

◆ **4. How to Install Jenkins**

- Download .war from jenkins.io.
- Run with:
- java -jar jenkins.war
- Access: http://localhost:8080
- Install suggested plugins (Git, Maven, Pipeline, JUnit, Cucumber, etc.).

◆ **5. Jenkins Job Types**

1. **Freestyle Job** → Basic job (run shell commands, batch commands).
2. **Maven Job** → Directly runs Maven goals (mvn clean test).
3. **Pipeline Job** → Uses Groovy script (Jenkinsfile) → modern CI/CD.

◆ **6. Jenkins Pipeline (Declarative Example)**

☞ If interviewer asks you to **write a Jenkins pipeline for Selenium Maven project**:

```
pipeline {
    agent any
```

```
    tools {
        maven 'Maven-3.9.0'   // Preconfigured in Jenkins
        jdk 'JDK-17'
    }

    stages {
        stage('Checkout Code') {
            steps {
                git branch: 'main', url: 'https://github.com/user/selenium-
bdd.git'
            }
        }

        stage('Build & Run Tests') {
            steps {
                sh 'mvn clean test'
            }
        }

        stage('Generate Report') {
            steps {
                sh 'mvn allure:report'
            }
        }

        stage('Publish Report') {
```

```
        steps {
            allure includeProperties: false, jdk: '', results: [[path: 'target/allure-results']]
        }
    }
  }

  post {
    always {
      junit 'target/surefire-reports/*.xml'
    }
    success {
      mail to: 'team@company.com',
           subject: "Build Successful: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
           body: "The build passed. Check Jenkins for details."
    }
    failure {
      mail to: 'team@company.com',
           subject: "Build Failed: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
           body: "The build failed. Please check Jenkins logs."
    }
  }
}
```

☞ This pipeline does:

- Checkout from Git

- Run Maven tests
- Generate Allure report
- Publish results
- Send email notification

◆ 7. Important Jenkins Plugins for Automation

- **Git Plugin** → Integrate with GitHub/GitLab.
- **Maven Integration Plugin** → Run Maven builds.
- **JUnit Plugin** → Publish test reports.
- **Allure Plugin** → Publish Allure reports.
- **Email Extension Plugin** → Send email alerts.
- **Docker Plugin** → Run builds inside Docker containers.
- **Pipeline Plugin** → Define CI/CD pipelines.

◆ 8. Jenkins + Selenium Integration (Real-Time Flow)

1. Developer pushes code to GitHub.
2. Jenkins triggers build automatically (webhook).
3. Jenkins checks out latest code.
4. Jenkins runs mvn clean test (executes Selenium + Cucumber tests).
5. Reports (Cucumber/Allure/Extent) are generated in Jenkins workspace.
6. Jenkins publishes reports on dashboard.
7. Jenkins sends email/Slack notifications.

◆ 9. Jenkins + Docker (Interview Booster)

☞ Interviewers often ask: *Can you run Selenium tests in Docker using Jenkins?*

Answer: Yes. We used Jenkins pipeline with Docker plugin:

- Build Docker image of Selenium tests (via Dockerfile).
- Run containers with Selenium Grid (Hub + Nodes).
- Execute tests inside containers.
- Generate and publish reports.

◆ 10. Interview Q&A (Jenkins)

✓ Q1: What is Jenkins?
A: Jenkins is an open-source CI/CD tool that automates building, testing, and deployment.

✓ Q2: Difference between Jenkins Freestyle job and Pipeline job?

- Freestyle → Simple, GUI-based.
- Pipeline → Scripted (Groovy), more flexible, version-controlled.

✓ Q3: How do you integrate Jenkins with Selenium?

- Create Maven job → run mvn clean test
- Add reports plugin → publish test results
- Trigger job from GitHub commit/webhook

✓ Q4: How do you trigger Jenkins jobs automatically?

- Using SCM polling (Poll SCM)
- Webhooks from GitHub/GitLab
- Cron scheduling (H 2 * * * for 2 AM daily run)

✓ Q5: What is Jenkinsfile?
A: A text file containing pipeline definition written in Groovy, stored in project repo.

✓ Q6: What is Continuous Integration vs Continuous Delivery?

- CI → Automatically build/test code on commit.
- CD → Automatically deliver/deploy code to staging/prod.

✅ **Q7: How to handle test reports in Jenkins?**

- Use JUnit Plugin (for TestNG/JUnit)
- Use Cucumber Reports Plugin (for BDD)
- Use Allure Plugin (for advanced reports)

✅ **Q8: Can Jenkins run in Docker?**
Yes. Jenkins itself can be containerized (jenkins/jenkins:lts image).

◆ **11. How You Should Answer in Interview**

☞ *In my project, we used Jenkins pipelines to run Selenium + Cucumber automation. Jenkins pulled code from GitHub, executed tests using Maven, generated Allure reports, and published them on the Jenkins dashboard. We also integrated with Docker to run tests in Selenium Grid containers. The jobs were triggered automatically using GitHub webhooks and nightly CRON scheduling.*

◆ **1. What is CI/CD?**

☞ **Interview Answer:**

- **CI (Continuous Integration):**
  Developers frequently merge (integrate) their code into a shared repository. Every commit triggers automated builds and tests to ensure nothing breaks.
- **CD (Continuous Delivery / Deployment):**
  After CI is successful, the pipeline automatically prepares (delivery) or pushes (deployment) the application into staging or production.

📌 Purpose: **Automate everything → Build → Test → Deploy**.
It ensures:

- Faster feedback
- Fewer manual errors
- Higher software quality
- Quick releases

◆ 2. CI/CD Pipeline Stages (General)

Most interviewers ask: *Explain the flow of a CI/CD pipeline in real-time.*

☞ Typical pipeline flow:

1. **Source (SCM – GitHub/GitLab/Bitbucket)**
    - Developer pushes code.
    - Webhook triggers pipeline.
2. **Build**
    - Compile code.
    - Install dependencies (e.g., Maven mvn clean install).
3. **Test (Unit + Automation)**
    - Unit tests (JUnit/TestNG).
    - Selenium/Cucumber automation tests.
    - API tests (RestAssured).
4. **Report & Analysis**
    - Test reports (JUnit, TestNG, Cucumber, Allure).
    - Code quality analysis (SonarQube).
5. **Package**
    - Application packaged as .jar, .war, or Docker image.
6. **Deploy**
    - Push Docker image to registry (DockerHub, ECR).
    - Deploy to server (Tomcat, AWS EC2, Kubernetes, etc.).

7. **Monitor**
    - Logging & monitoring tools (ELK, Prometheus, Grafana).

◆ 3. CI/CD Tools Commonly Used

- **CI Tools:** Jenkins, GitHub Actions, GitLab CI, Bamboo, CircleCI.
- **Build Tools:** Maven, Gradle.
- **Containerization:** Docker.
- **Orchestration:** Kubernetes.
- **Quality Checks:** SonarQube.
- **Reports:** Allure, Extent, Cucumber, TestNG.

☞ Interview tip: *In Selenium project, we mainly use Jenkins, Maven, Docker, and GitHub for CI/CD pipeline.*

◆ 4. CI/CD Pipeline in Selenium Automation (Real-Time Example)

☞ How to **explain in interview**:

In my project, we implemented CI/CD for automation tests. Whenever code was pushed to GitHub:

1. Jenkins pulled the code.
2. Maven built and executed Selenium + Cucumber tests.
3. Reports (Allure, Cucumber HTML) were generated and published in Jenkins.
4. Docker was used to run tests in Selenium Grid containers (Chrome, Firefox).
5. Reports were shared automatically via email and Slack.
6. For delivery, Jenkins published artifacts to Nexus and triggered deployment scripts.*

◆ 5. Sample Jenkinsfile for CI/CD

Here's a **pipeline Groovy script** (Declarative) that covers CI/CD:

```
pipeline {
    agent any

    tools {
        maven 'Maven-3.9.0'
        jdk 'JDK-17'
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/user/selenium-
bdd.git'
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean install -DskipTests'
            }
        }

        stage('Test') {
            steps {
```

```
            sh 'mvn clean test'
        }
    }

    stage('Code Quality') {
        steps {
            sh 'mvn sonar:sonar'
        }
    }

    stage('Package & Dockerize') {
        steps {
            sh 'docker build -t selenium-bdd-app:latest .'
            sh 'docker tag selenium-bdd-app:latest myrepo/selenium-bdd-app:latest'
            sh 'docker push myrepo/selenium-bdd-app:latest'
        }
    }

    stage('Deploy') {
        steps {
            sh 'kubectl apply -f k8s/deployment.yaml'
        }
    }
}

post {
```

```
        always {
            junit 'target/surefire-reports/*.xml'
        }
        success {
            echo "Build Successful!"
        }
        failure {
            echo "Build Failed! Please check logs."
        }
    }
}
```

👉 This does:

- CI → Build, Test, Reports, SonarQube analysis.
- CD → Dockerize + Deploy to Kubernetes.

◆ 6. CI vs CD (Interview Direct Answer)

- **CI** → Focus on **integration & testing** of code frequently.
- **CD** → Focus on **automated delivery/deployment** to environments (staging/prod).

◆ 7. Common Interview Questions on CI/CD

✓ Q1: What is CI/CD and why is it important?

👉 CI/CD automates build, test, and deploy → reduces manual errors, increases release speed.

✓ Q2: Which tools did you use for CI/CD?

👉 Jenkins for pipeline automation, Maven for build, Docker for containerization, Kubernetes for orchestration, SonarQube for code quality, Allure for reports.

✅ **Q3: How did you implement CI/CD for Selenium automation?**
👉 GitHub → Jenkins → Maven → Selenium/Cucumber tests → Allure reports → Docker Grid execution → Deploy artifacts.

✅ **Q4: What is the difference between Jenkins and CI/CD?**
👉 Jenkins is a **tool** used to implement CI/CD pipeline. CI/CD is the **process**.

✅ **Q5: How do you handle failures in CI/CD?**
👉 Jenkins pipeline post { failure { ... } } sends email/Slack alerts and logs.

✅ **Q6: Did you integrate CI/CD with Docker/Kubernetes?**
👉 Yes, we dockerized tests and ran them on Kubernetes cluster for scalability.

✅ **Q7: How do you ensure quality in CI/CD pipeline?**
👉 Automated tests + SonarQube + static code analysis + monitoring after deployment.

◆ **8. How You Should Answer in Interview**

👉 *In our project, CI/CD pipeline was implemented using Jenkins. CI ensured every commit triggered Maven build and Selenium tests with reports. CD automated deployment using Docker and Kubernetes. Reports were published in Jenkins and sent to stakeholders. This reduced manual work and ensured quick and reliable releases.*

◆ **1. What is SDLC?**

👉 **Interview Answer:**
SDLC (Software Development Life Cycle) is a **step-by-step process** used to design, develop, test, and deliver high-quality software. It defines **stages, roles, and deliverables** for building software in a structured way.

📌 Purpose:

- Ensure software is **delivered on time**, **within budget**, and **with high quality**.

- Reduce risks and rework.

◆ 2. SDLC Phases (Classic 7 Stages)

1. **Requirement Analysis**
    - Gather business requirements (BRD, SRS).
    - Stakeholders, BA, QA, and Dev team involved.
    - Interview Tip: *Testers understand requirements and prepare test scenarios here.*
2. **Planning**
    - Create project plan, cost estimation, resource allocation.
    - Decide technologies, tools (like Selenium, Jenkins, Docker).
3. **Design**
    - High-Level Design (HLD): Architecture.
    - Low-Level Design (LLD): Detailed modules, DB schema.
    - Testers prepare **test planning & test strategy**.
4. **Development (Coding)**
    - Developers write code using chosen language.
    - Testers may write **automation framework setup (Selenium + Cucumber)** parallelly.
5. **Testing**
    - QA validates the application using manual & automation testing.
    - Automation team runs **Selenium/Cucumber/TestNG/JUnit** scripts.
    - Reports generated (Allure, Extent, Cucumber).
6. **Deployment**
    - Application moved to production environment.

- CI/CD (Jenkins, Docker, Kubernetes) is used.

7. **Maintenance**
    - Post-release support.
    - Bug fixes, patch releases, enhancements.
    - Regression testing automated with Selenium.

◆ 3. Popular SDLC Models

In interviews, they may ask *which SDLC model your project followed*.

1. **Waterfall Model**
    - Sequential, each phase after the previous.
    - Rigid, less flexible.
    - Used in banking, govt projects.
2. **V-Model (Verification & Validation)**
    - Testing phase runs parallel to development phase.
    - Example: SRS → Acceptance Testing, LLD → Unit Testing.
3. **Agile Model (Scrum, Kanban)**
    - Iterative & incremental.
    - Small sprints (2–3 weeks).
    - Continuous feedback.
    - **Most common in real projects (Selenium + Cucumber + CI/CD fits here).**
4. **Spiral Model**
    - Risk-driven, iterative + waterfall + prototyping.
5. **Big Bang Model**
    - No process, just coding. Rarely used.

◆ 4. Where Testing & Automation Fit in SDLC

☞ **Manual QA:** Involved from requirement phase (review requirements, write test cases).
☞ **Automation QA (Selenium):**

- During development → prepare test framework.
- During testing → execute automation scripts.
- During CI/CD → integrate with Jenkins & Docker.

◆ 5. SDLC vs STLC (Interview Trap Question)

✓ **SDLC (Software Development Life Cycle):**
End-to-end software process → Requirement → Design → Development → Testing → Deployment → Maintenance.

✓ **STLC (Software Testing Life Cycle):**
Only testing process → Requirement Analysis → Test Planning → Test Case Design → Test Execution → Test Closure.

☞ Tip: If asked in interview, always say:
"SDLC covers the overall project, while STLC is a subset of SDLC focusing only on testing phases."

◆ 6. Common Interview Questions on SDLC

✓ **Q1: What is SDLC? Why is it important?**
☞ SDLC is a structured process to deliver high-quality software systematically, reducing risks and costs.

✓ **Q2: Which SDLC model did your project follow?**
☞ Most Agile projects → *We followed Agile Scrum. Each sprint (2 weeks) included requirement analysis, development, automation (Selenium + Cucumber), CI/CD pipeline in Jenkins, and testing.*

✓ **Q3: Difference between Waterfall & Agile?**

- Waterfall: Sequential, rigid, less feedback.

- Agile: Iterative, flexible, continuous feedback.

✅ **Q4: Where does automation fit in SDLC?**

👉 In **Testing phase** (execution & regression), and in **CI/CD pipeline** for continuous testing.

✅ **Q5: How did you contribute to SDLC in your project?**

👉 *I was involved in requirement analysis, test planning, building Selenium-Cucumber automation framework, integrating with Jenkins CI/CD, and generating test reports.*

◆ 7. How You Should Answer in Interview

👉 *In my project, we followed Agile SDLC. Each sprint was 2 weeks. Requirements were discussed in sprint planning, development was done in parallel with automation setup, and we used Selenium + Cucumber for automation. Jenkins CI/CD pipeline executed our automation scripts after every code commit. Reports (Allure, Extent) were published and shared with stakeholders. This ensured quick feedback and continuous delivery.*

◆ 1. What is STLC?

👉 **Interview Answer:**
STLC (Software Testing Life Cycle) is a **systematic process of testing activities** carried out to ensure software quality. It defines **phases in testing**, from requirement analysis to test closure.

📌 Purpose:

- Improve test coverage.
- Detect defects early.
- Deliver quality product.

👉 STLC is a **subset of SDLC** (it starts after requirements are defined).

◆ 2. Phases of STLC (with Real-Time Explanation)

There are **6 main phases** in STLC:

✅ 1. Requirement Analysis

- QA team studies **Business Requirement Document (BRD)**, **SRS**, or **User Stories**.
- Identify which features are testable, risks, priorities.
- Automation team checks feasibility (can we automate this in Selenium?).

Deliverables: Requirement Traceability Matrix (RTM).

✅ 2. Test Planning

- QA Lead/Manager creates **Test Plan** document.
- Define scope, test strategy, tools (Selenium, Cucumber, TestNG), resources, timeline.
- Entry & Exit criteria decided.

Deliverables: Test Plan, Effort Estimation.

✅ 3. Test Case Design / Development

- Testers write manual test cases.
- Automation team writes **step definitions, feature files, test scripts**.
- Prepare **test data** (Excel, JSON, DB, Mock data).

Deliverables: Test Cases, Automation Scripts.

✅ 4. Test Environment Setup

- Setup hardware, software, network, test environment.

- Install application build, browsers, Selenium Grid, Docker containers.
- Configure Jenkins CI/CD for automation.

Deliverables: Test Environment Readiness Report.

✅ 5. Test Execution

- Manual QA executes test cases.
- Automation QA runs Selenium-Cucumber/TestNG scripts.
- Defects reported in JIRA/Bugzilla.
- Regression testing automated using CI/CD.

Deliverables: Test Execution Report, Defect Logs.

✅ 6. Test Closure

- Check if all planned tests are executed.
- Defect leakage analysis (how many bugs missed).
- Prepare test summary report.
- Knowledge transfer & retrospective in Agile.

Deliverables: Test Summary Report (TSR), Lessons Learned Document.

◆ 3. STLC vs SDLC (Direct Interview Question)

✅ **SDLC:** Overall software development process (Requirement → Design → Development → Testing → Deployment → Maintenance).

✅ **STLC:** Only focuses on **Testing phases** (Requirement Analysis → Planning → Test Case Design → Execution → Closure).

☞ *So STLC is part of SDLC.*

◆ 4. Role of Automation (Selenium) in STLC

- **Requirement Analysis** → Check feasibility of automation.
- **Test Planning** → Decide framework (Selenium + Cucumber + TestNG).
- **Test Case Design** → Write feature files & step definitions.
- **Test Environment Setup** → Setup Jenkins CI/CD + Selenium Grid + Docker.
- **Test Execution** → Run automation scripts, generate reports (Allure, Extent).
- **Closure** → Share automation test summary.

◆ 5. Common Interview Questions on STLC

✓ Q1: What is STLC?
☞ STLC is a process of testing activities carried out in a systematic way to ensure software quality.

✓ Q2: Difference between SDLC and STLC?
☞ SDLC = Entire software process; STLC = Only testing life cycle.

✓ Q3: When does STLC start?
☞ STLC starts once **requirements are gathered** (after SDLC's requirement phase).

✓ Q4: What are entry and exit criteria in STLC?

- Entry: Requirements are finalized, environment is ready.
- Exit: All tests executed, defects resolved, summary prepared.

✓ Q5: What are deliverables in STLC?
☞ RTM, Test Plan, Test Cases, Test Data, Test Execution Report, Defect Report, Test Summary Report.

✓ Q6: How do you integrate STLC with automation?
☞ In my project, STLC phases were automated using Selenium-Cucumber. Jenkins CI/CD ran automation nightly, and Allure reports were published as part of test closure.

◆ 6. How You Should Answer in Interview

☞ *In my project, we followed STLC to structure testing. We started with requirement analysis and created RTM. In test planning, we defined our Selenium-Cucumber BDD framework. Test cases were automated in parallel, environment was setup with Jenkins + Docker, and automation was executed on CI/CD. At closure, we generated Allure/Extent reports and shared a summary with stakeholders.*

◆ 1. What is Bug / Defect Life Cycle?

☞ **Interview Answer:**
A **Defect Life Cycle (or Bug Life Cycle)** is the **journey of a defect** from when it is discovered until it is closed. It defines the **different states** a defect goes through and the **actions of QA, Dev, and Manager** at each stage.

📌 Purpose:

- Ensure defects are tracked properly.
- Avoid duplicate / invalid bugs.
- Help prioritize and fix defects efficiently.

◆ 2. Defect Life Cycle States (Common Flow)

Here's the **standard flow** you can explain:

1. **New / Opened**
    - Tester finds a bug and logs it in a defect management tool (JIRA, Bugzilla, ALM, etc.).
    - Status = **New**.
2. **Assigned**
    - Lead/Manager assigns the bug to a developer.

- Status = **Assigned**.

3. **In Progress / Open**
    - Developer starts working on the bug.
    - Status = **In Progress**.
4. **Fixed / Resolved**
    - Developer fixes the bug and updates status to **Fixed/Resolved**.
5. **Retest**
    - QA retests the bug in the new build.
    - If fixed → Status = **Verified / Closed**.
    - If not fixed → Status = **Reopen**.
6. **Closed**
    - Tester confirms bug is fixed → **Closed**.
7. **Rejected / Not a Bug**
    - If developer thinks it's not a valid issue → **Rejected**.
8. **Deferred / Postponed**
    - Bug won't be fixed in this release (maybe low priority) → **Deferred**.
9. **Duplicate**
    - If bug is already logged → **Duplicate**.

◆ 3. Real-Time Example (Interview Style)

☞ *Suppose I find that the login button is not working in our application. I raise a defect in JIRA with severity = High and priority = P1. The bug goes to "New". The lead assigns it to a developer → status "Assigned". Developer fixes it → status "Fixed". I retest in next build, if it works fine, I mark as "Closed". If issue still exists, I reopen it.*

◆ 4. Tools Used for Defect Tracking

- **JIRA** → Most commonly used.
- Bugzilla, Mantis, HP ALM, Redmine, Azure DevOps.

◆ 5. Interview Tricky Points

✓ Q1: Who can close a bug?
☞ Only QA/Tester can close a bug after verification. Developer cannot close their own bug.

✓ Q2: What is difference between Severity & Priority?

- **Severity** → Impact of defect (Technical).
- **Priority** → Order in which defect should be fixed (Business).
  ☞ Example:
- High Severity, Low Priority → Spelling mistake in company name.
- Low Severity, High Priority → Wrong logo on home page.

✓ Q3: What are the possible statuses of a bug?
☞ New → Assigned → In Progress → Fixed → Retest → Closed → Reopen → Rejected → Deferred → Duplicate.

✓ Q4: What is Defect Leakage?
☞ When a bug is missed in testing and found in later stage (like UAT or Production).

✓ Q5: What is Defect Density?
☞ Number of defects per size of module/code.

◆ 6. Visual Flow (How to Explain in Interview)

☞ New → Assigned → In Progress → Fixed → Retest → Closed
(with alternate paths → Reopen / Rejected / Deferred / Duplicate).

◆ 7. How You Should Answer in Interview

◆ 1. What is Agile?

☞ **Interview Answer:**
Agile is a **software development methodology** based on **iterative and incremental** development. Work is divided into **small cycles (sprints)**, usually 2–4 weeks, and teams deliver working software frequently with continuous feedback.

📌 Core Ideas:

- Customer collaboration over contract negotiation.
- Responding to change over following a rigid plan.
- Working software delivered frequently.
- Teamwork and communication are key.

◆ 2. Agile Manifesto (4 Values)

1. **Individuals & interactions** over processes & tools.
2. **Working software** over documentation.
3. **Customer collaboration** over contract negotiation.
4. **Responding to change** over following a plan.

☞ *In interviews, mention these 4 points — shows you know the foundation.*

◆ 3. Agile Principles (12 in total, but important ones)

- Deliver working software frequently.
- Welcome changing requirements at any stage.
- Business people and developers work together daily.
- Continuous attention to quality and testing.
- Regular reflection and adaptation (retrospectives).

◆ 4. Agile Frameworks (Models under Agile)

☞ Agile is the **umbrella** → there are frameworks under it:

1. **Scrum** (most popular)
    - Sprint-based (2–4 weeks).
    - Roles: Product Owner, Scrum Master, Development Team.
    - Events: Sprint Planning, Daily Scrum (Standup), Sprint Review, Retrospective.
2. **Kanban**
    - Visual workflow (Board with To-Do, In-Progress, Done).
    - Continuous delivery, no fixed sprint.
3. **XP (Extreme Programming)**
    - Focuses on coding best practices (pair programming, TDD, continuous integration).

☞ In automation interviews, they mostly ask about **Scrum**.

◆ 5. Agile Ceremonies (Scrum Events)

☞ These are **mandatory interview points**:

1. **Sprint Planning** – Plan work for the sprint.
2. **Daily Standup** – 15 min meeting to discuss progress/blockers.
3. **Sprint Review** – Demo of completed work to stakeholders.
4. **Sprint Retrospective** – Team reflects and improves process.

◆ 6. Agile Artifacts

- **Product Backlog** → List of all features (maintained by Product Owner).
- **Sprint Backlog** → Features selected for current sprint.
- **Burndown Chart** → Tracks remaining work in sprint.

◆ 7. Agile in Selenium / Automation Projects

☞ Where **we testers** come into picture:

- Automation team starts preparing **framework** during initial sprints.
- Testers write automation scripts parallel to development.
- Regression suite automated using Selenium + Cucumber.
- Jenkins CI/CD pipeline triggers tests after every code push.
- Reports (Allure/Extent/Cucumber) shared at end of sprint.

◆ 8. Example (How to Explain in Interview)

☞ *In my project, we followed Agile Scrum. Each sprint was 2 weeks. During sprint planning, we received user stories. I prepared test scenarios, automated high-priority regression cases using Selenium-Cucumber, and executed them via Jenkins CI/CD. We participated in daily standups, sprint reviews, and retrospectives. At sprint closure, we shared Allure/Extent reports with the team.*

◆ 1. What is Scrum?

☞ **Interview Answer:**
Scrum is an **Agile framework** used to develop, deliver, and sustain complex products.
It is based on **short iterations called sprints (2–4 weeks)** where a small cross-functional team delivers **working software frequently** with continuous feedback.

📌 Scrum = Roles + Events + Artifacts.

◆ 2. Scrum Roles (Who does what)

Scrum defines **three roles**:

1. **Product Owner (PO)**
    - Owns the **Product Backlog**.
    - Defines priorities & business value.
    - Acts as a bridge between stakeholders and team.
2. **Scrum Master**
    - Facilitator / Servant Leader.
    - Removes impediments, ensures Scrum rules are followed.
    - Not a manager, but a coach.
3. **Development Team**
    - Cross-functional team (Developers + Testers + Designers + Analysts).
    - They deliver the **increment** (working software).

◆ 3. Scrum Events (Ceremonies)

☞ 5 important events:

1. **Sprint**
    - Time-boxed cycle (2–4 weeks).
    - Goal: Deliver a potentially shippable product increment.
2. **Sprint Planning**
    - Decide what stories to work on.
    - PO explains user stories → Team selects what they can complete → Creates Sprint Backlog.
3. **Daily Scrum (Standup)**
    - 15-minute meeting daily.
    - Each member answers:
        - What did I do yesterday?
        - What will I do today?

        - Any blockers?

4. **Sprint Review**
    - End of sprint. Team demonstrates completed features to stakeholders.
5. **Sprint Retrospective**
    - Team reflects → what went well, what didn't, how to improve next sprint.

◆ 4. Scrum Artifacts

☞ 3 key artifacts:

1. **Product Backlog**
    - List of all features / requirements (maintained by PO).
2. **Sprint Backlog**
    - Features selected for current sprint.
3. **Increment**
    - The working software delivered at the end of sprint.

◆ 5. Scrum Flow (Step by Step)

1. PO prepares **Product Backlog** (all requirements).
2. In **Sprint Planning**, team selects items → creates **Sprint Backlog**.
3. Team works in **Sprint (2–4 weeks)** → Developers + Testers together.
4. Daily **Scrum Standups** to sync progress.
5. At sprint end → **Sprint Review** (demo).
6. Then → **Retrospective** (improve process).
7. Deliver **Increment** to customer.

◆ 6. Scrum in Testing / Automation (Your Role as QA)

☞ In interviews, always connect **Scrum → QA role**:

- Participate in **story grooming** → understand acceptance criteria.
- Write **test scenarios & cases** early.
- Automate regression cases during sprint (Selenium + Cucumber).
- Continuous integration (Jenkins CI/CD).
- Defect reporting in JIRA, verify fixes in same sprint.
- Demo working automation reports (Extent/Allure) in sprint review.

1 Waterfall Model (Linear Sequential Model)

The Waterfall model is a **traditional SDLC model** where the software development process flows **sequentially** from one phase to another, like a waterfall. Each phase must be **completed before the next begins**, and there is no overlap.

Steps / Phases:

1. **Requirement Analysis:**
   All requirements are collected from stakeholders and documented in a **Software Requirement Specification (SRS)**. Testers and developers review the document to ensure clarity and completeness.
2. **System / High-Level Design:**
   Architects and designers prepare the **system architecture**, database design, and module design. The overall structure of the software is planned.
3. **Detailed / Low-Level Design:**
   Each module's detailed design is prepared, defining logic, data flow, and interface details.
4. **Implementation / Coding:**
   Developers write code according to the design documents. Each module is implemented sequentially.

5. **Testing / Verification:**
   Once development is complete, QA tests the complete application using **manual and automation testing**. Defects are reported, fixed, and retested.
6. **Deployment / Installation:**
   The tested application is deployed in the production environment for use by end-users.
7. **Maintenance:**
   Post-deployment, maintenance activities such as bug fixes, updates, and enhancements are carried out.

Key Idea: Linear, sequential process — works best for **small projects with stable requirements**.

---

2 V-Model (Verification & Validation Model)

The V-Model is an **extension of Waterfall** where testing activities are planned **parallel to development phases**. The left side of the "V" represents development, and the right side represents corresponding testing.

Steps / Phases:

1. **Requirement Analysis → Acceptance Testing:**
   Functional and non-functional requirements are gathered and documented. Corresponding **acceptance tests** are defined to validate requirements.
2. **System Design → System Testing:**
   Overall system architecture and design are created. **System tests** are planned to validate the design at the system level.
3. **High-Level / Architecture Design → Integration Testing:**
   Modules and components are designed. **Integration tests** are prepared to test interactions between modules.
4. **Low-Level / Module Design → Unit Testing:**
   Detailed module design is done. **Unit tests** are created to verify individual modules.

5. **Implementation / Coding → Execution of Unit Testing:**
   Developers write code for each module. Unit tests are executed to ensure each module works as expected.
6. **Deployment / Acceptance Testing Execution:**
   After successful system and integration testing, acceptance testing is performed, and the system is deployed.

Key Idea: Every development phase has a corresponding **testing phase**, allowing early detection of defects.

---

3 Spiral Model (Iterative & Risk-Driven Model)

The Spiral model combines **iterative development** with **risk management**, suitable for **large and complex projects**. Development happens in cycles or **spirals**, with each spiral delivering an increment.

Steps / Phases in Each Spiral:

1. **Requirement Gathering and Planning:**
   Requirements are collected for this iteration. Planning is done regarding resources, timeline, and deliverables.
2. **Risk Analysis:**
   Potential risks are identified, and mitigation strategies are planned. This ensures high-risk areas are addressed early.
3. **Design & Prototyping:**
   A prototype or preliminary design is created for the iteration. Stakeholders review and provide feedback.
4. **Implementation / Coding:**
   Development team codes the features planned for this spiral.
5. **Testing / Verification:**
   The increment is tested for defects. Regression testing is performed if necessary.
6. **Evaluation / Planning Next Spiral:**
   After testing, the software is evaluated, feedback is gathered, and planning for the next spiral begins. The cycle repeats until the final product is ready.

Key Idea: Iterative development with **continuous risk assessment**, allowing flexibility for changing requirements.

Summary

- **Waterfall:** Linear, sequential — best for small projects. Steps: Requirement → Design → Implementation → Testing → Deployment → Maintenance.
- **V-Model:** Linear + parallel testing — each dev phase has a corresponding test phase. Steps: Requirement → Acceptance Test, System Design → System Test, High-Level Design → Integration Test, Low-Level Design → Unit Test → Coding → Execution → Deployment.
- **Spiral:** Iterative + risk-driven — software is developed in repeated spirals with continuous evaluation. Steps per spiral: Requirement → Risk Analysis → Design/Prototype → Coding → Testing → Evaluation/Next Spiral.