YEAR 02

B.Tech Computer Science and Engineering in DevOps

Build and Release Management

MODULE 1

Introduction to Build and Release Management

Contents

MODULE 1

Introduction to Build and Release Management

You will be discussing about build/release management, best practices of build and release, build abstraction and dependency management.

Module Objectives

At the end of this module, you will be able to:

- Understand what is a build

- Explain build and release management

- Define different aspects in build and release management

- Discuss the best practices for build and release management

- Describe build abstraction

- Explain declarative dependency management

Module Topics

Let us take a quick look at the topics that we will cover in this mod

1. What is a build

2. Introduction–Build and release management

3. Build Management

4. Release management

5. Best practices for build and release management

6. Build abstraction

7. Dependency management
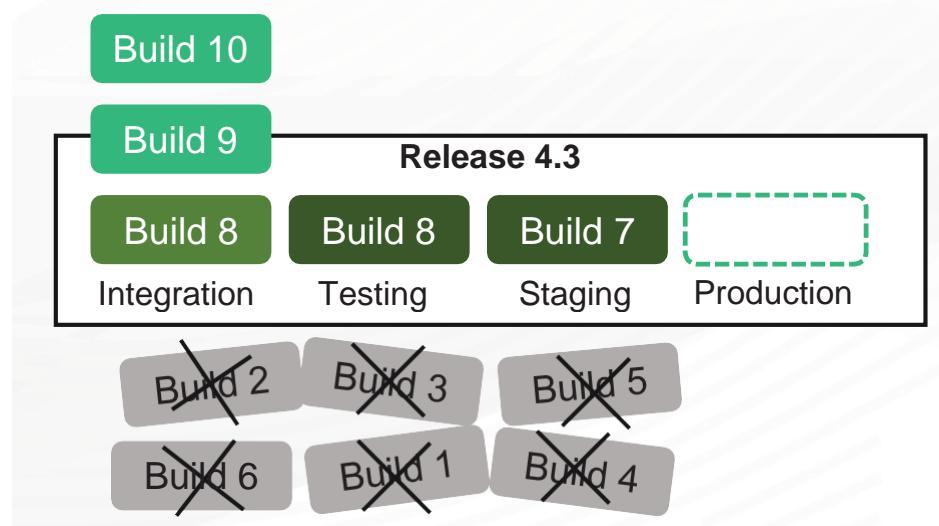
1.1 What is a Build?

- A Build is a version of an application or program.

- Every build is identified by a build number.

- Compilation of build gives tangible results.

- It refers to a process of converting source code into application.

The following figure represents how multiple builds are integrated, staged and deployed to production.

Facilitator Notes:
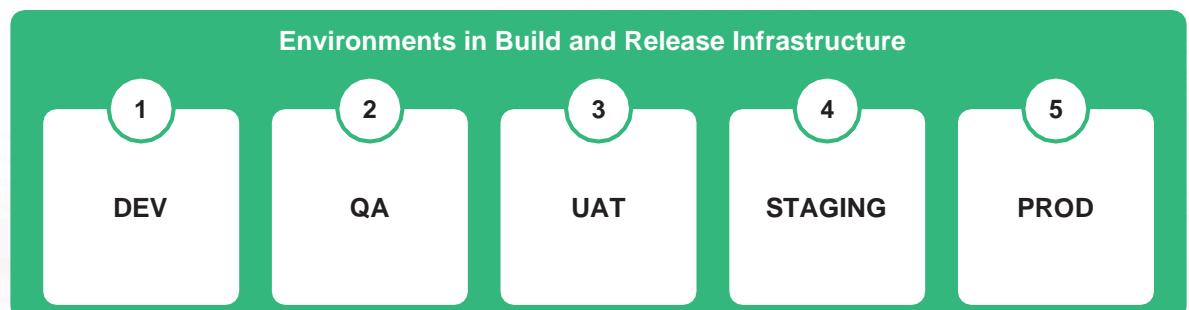
Give an introduction about the term 'Build' to the participants.

Building an application or software involves various stages. Each build has different build numbers and it is always built from a source code repository like git. Building an application or a software requires build tools like ant, maven, gradle, etc. Build tools compile the source code files into reusable executable files or packages.

1.2 Introduction—Build and Release Management

Build and release management is the process of managing, planning, scheduling and controlling a software build throughout its life cycle.

These are the 5 types of environments in build and release infrastructure:

Build and Release management will control the life cycle of a software product, the process of planning, managing, scheduling and controlling the build in different stages and environments like development, testing, staging and production stages.

Generally, most organizations will have 5 types of environments for their build and release infrastructure.

- **DEV** – Development team maintains this environment to write their code. Only the development team has access to this environment. QA or other teams don't have access to this environment. Dev team uses this environment, mostly to write unit test cases.

- **QA** – QA environment is owned by the testing team and the actual testing takes place in this environment. DEV team does not have access to the QA environment. After coding completion, the code is moved from DEV to QA environment for test execution.

- **UAT** – User Acceptance Test environment is used by business users to conduct testing after system testing has been completed. In this phase, the product is tested from the business point of view. UAT environment access is restricted only to the business users and on some occasions, temporary access will be provided to the QA team if in case of business users need assistance.

- **STAGING** – The STAGING environment exactly resembles the production environment. The application installed in the staging environment should match closely with the production environment.

- **PROD** – The PROD environment is the production environment which is accessed by real users and none of the DEV and QA teams have access to this environment.

Build and release management has few aspects. Let's look into detailed aspects of build management and release management in the next slides.

What did You Grasp?

Topic Analysis

1. Which team has access to the production build environment?
   A) **Development team**
   B) **Testing team**
   C) **Business users**
   D) **None of the above**

2. Which team has access to the UAT build environment?
   A) **Development team**
   B) **Testing team**
   C) **Business users**
   D) **None of the above**

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.
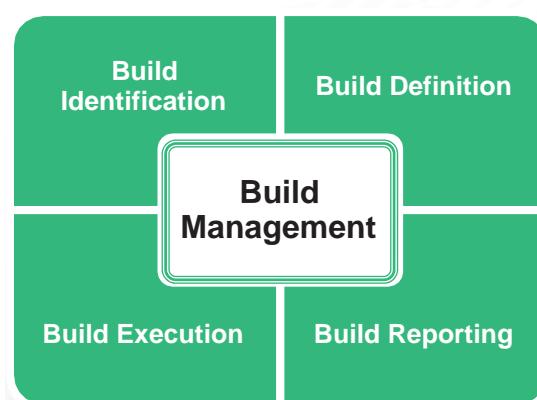
Answer:

1. d. None of the above

2. c. Business users

2.1 Build Management

Software build is a process of converting source code into executable applications or programs. Build management plays very important role in releasing an application.

Build management process can be broken into several distinct aspects:
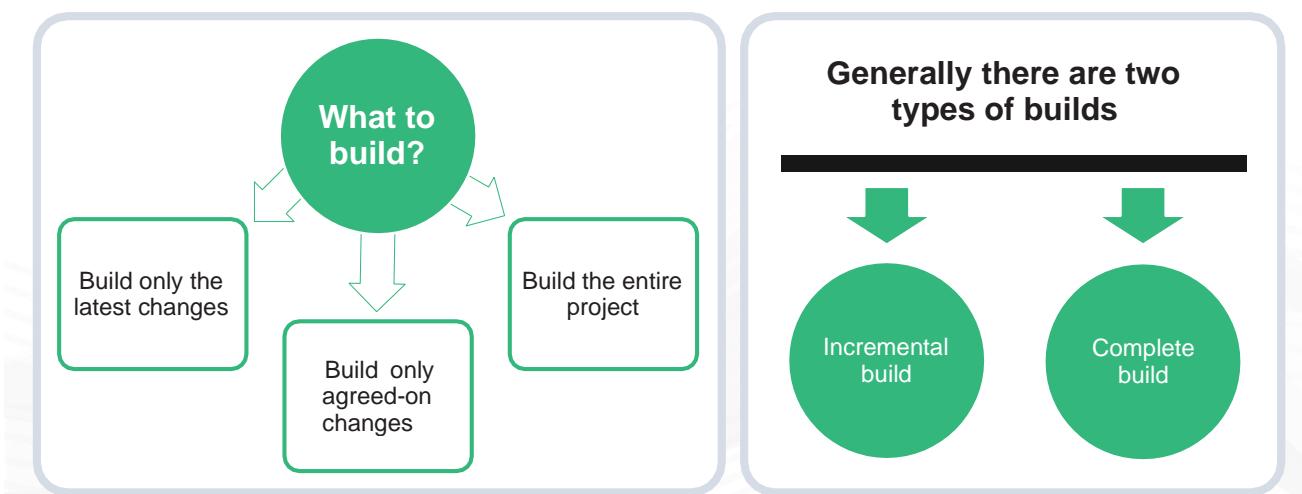
Build Identification	Build Definition
Build Execution	Build Reporting

Build Management

Build and release management is to make sure every build is consistent. You should always know why you are building it, what you are building and always should be able to run the build again in the future. If this is right then it is much more productive and you can concentrate on other things.

The **build process** is described below:

- Fetching source code from a central repository

- Compile the source code and check/download the dependencies

- Run automated unit/smoke tests to verify the integrity of the build

- Once build is successful, store the artifacts and send notifications of the build

2.1.1 Build Identification
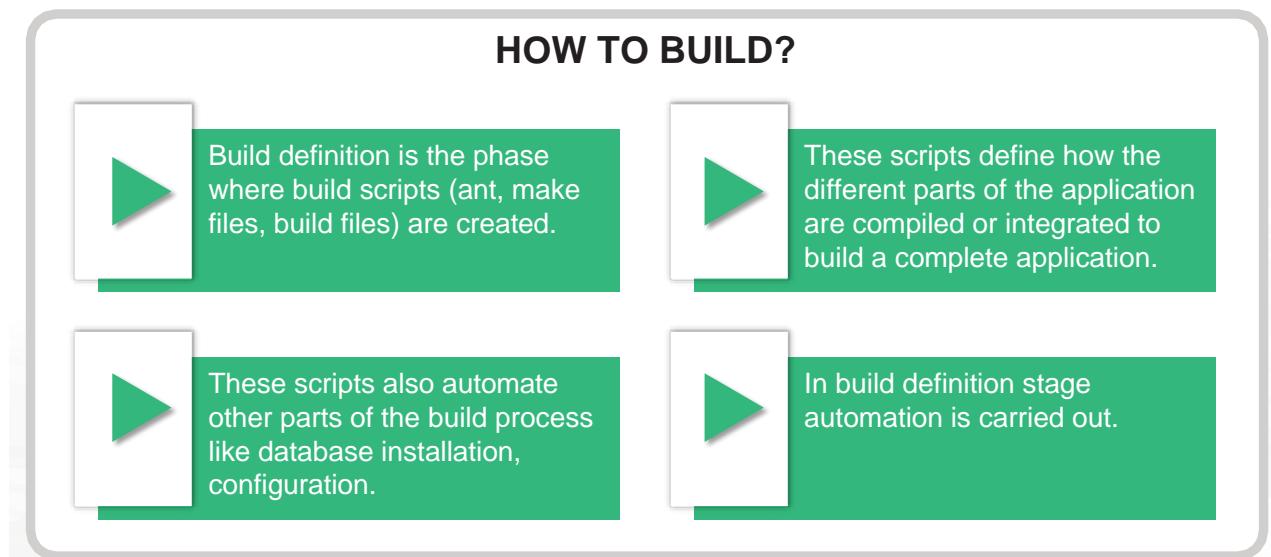
Build identification is used to identify what changes to build – the latest changes, agreed-on changes and complete changes. Initially build identification is an informal process where private builds or integration build is created. During the later stages, identification is more formal where release builds are carried out.

Generally, there are two types of build types:

**Complete Build:** In this type, the build is performed from scratch. The dependencies are figured out, compiles the entire source code and bundles the output into build artifacts.

Incremental Build: In this type, the build tool uses the 'last build state' and build only new changes since last build. Incremental builds are much faster since it builds only incremental changes.

2.1.2 Build Definition

HOW TO BUILD?

▶ Build definition is the phase where build scripts (ant, make files, build files) are created.

▶ These scripts define how the different parts of the application are compiled or integrated to build a complete application.

▶ These scripts also automate other parts of the build process like database installation, configuration.

▶ In build definition stage automation is carried out.

Facilitator Notes:

Explain the participants about the build definition process.

Build definition phase tells how to build. In this phase build files, scripts, make files are created which define the compilation to produce a complete system or application. In build definition stage most of the build automation process is carried out. The scripts created in this stage will automate other processes such as database installation and configuration.

Creating build scripts are the first step to automate the build execution. These scripts can be a shell script, python script or XML files, etc. The goal of creating the scripts is to get faster, safer and better builds.

2.1.3 Build Execution

Execution of Build

During build execution, scripts defined in build definition phase are used

Build execution can be triggered in multiple ways – direct invocation or by a schedule

Build execution speed can be increased by build distribution

Build can be executed in a number of ways

Facilitator Notes:

Explain the participants how a build is executed.

Build execution phase depends on the scripts, build files, etc., defined in build definition phase. Build execution speed can be increased in two ways – distribute build across multiple machines or build parts of the system that is changed since the last build.

A Build can be executed in a number of ways – Manual invocation, automatic schedule and source code build trigger.

- **Manual Invocation:** It is the common build trigger. If the build should be triggered to compile the code changes, the user can trigger the build using a build tool.

- **Automatic Schedules:** Build execution is triggered at specific intervals, regardless of the source code changes. The best example is Nightly/Weekly Builds.

- **Source code Trigger:** Build execution is triggered if a developer commits the code into the source code repository.

2.1.4 Build Reporting

Build execution results are captured in this phase. It includes:

Results of build – Success/Failure

Compilation reports

Unit test reports

Release reports

Automatic notification of build success/failure is sent via email can be published to a website
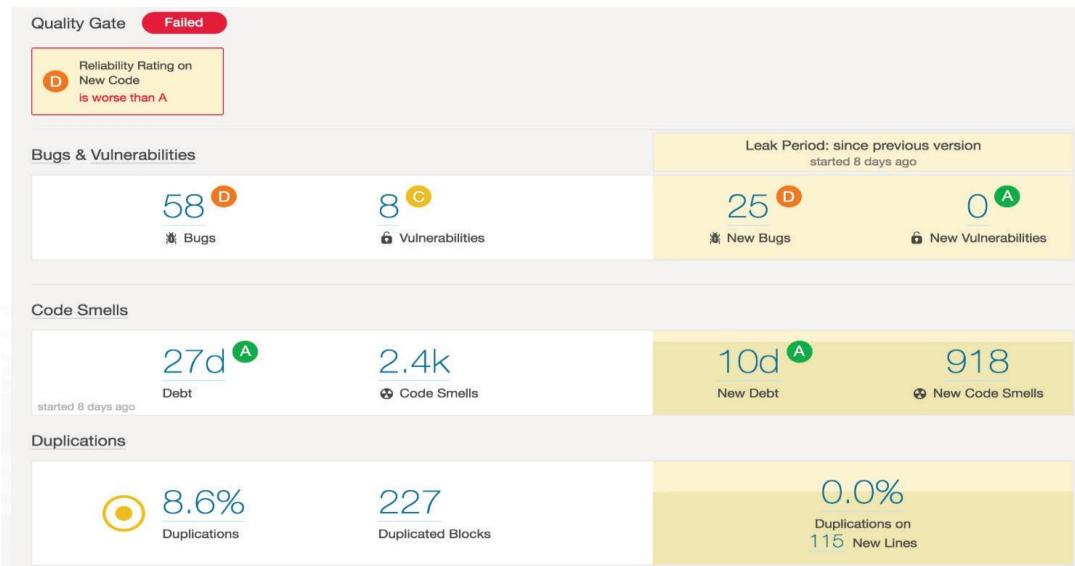
Facilitator Notes:

Discuss about build reporting in detail.

In build reporting phase, the build execution results (success/failure) are recorded. Execution results include:

- **Compilation reports:** The compile reports are generated during the build. Compile Reports will provide an overview to developers into what happens in their software during the compile: how much output is generated from their code, which Java packages and classes are creating large JavaScript output, and how the code is split.

- **Unit test reports:** The Maven Surefire Plugin can generate unit test results automatically for your project. To enable the unit test report in the project documentation site, add the maven-surefire-report-plugin to the reporting section of your project's pom.xml.

- **Release reports:** The releases report shows all the releases that each project has including cross-project releases. In the report view, you'll see the release date, progress, and all the associated issues.

- **Build metrics:** Build metrics for a maven build include metrics like total time taken for the build, status of build SUCCESS or FAILURE, test results etc.

- **Sonarqube reports** (Code coverage, checkstyle and find bugs reports)

2.1.4.1 Build Reporting – Sample Sonarqube Report

> **Facilitator Notes:**
>
> Discuss about sample sonarqube report

The above screenshot gives an overview of number of bugs and vulnerabilities in the project, duplicate lines in the code and code highlights the best practices that is to be used while writing the code. This report is generated using Sonarqube and it can collect the unit test reports with the help of other plugins like Jacacco.

2.1.4.2 Build Reporting – Build status

The above screenshot gives an overview of the build status. This report is captured using Jenkins.

<div style="background-color: #3cb878; border-radius: 12px; padding: 12px;">

How to generate a build report

The build tool is used to execute the java source code independently. It can compile the entire project, compile the code, package the code to a jar etc.

Maven is a widely used build tool and provides a common platform to perform these activities which makes programmer's life easier.

</div>

Build Life Cycle:

Basic maven phases are listed below.

clean: deletes all artifacts and targets which are created already.

compile: used to compile the source code of the project.

test: test the compiled code and these tests do not require to be packaged or deployed.

package: package is used to convert your project into a jar or war etc.

install: install the package into the local repository for use of another project

The script below shows a sample project-team site report generated in maven.

```
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>2.1.2</version>
        <reportSets>
          <reportSet>
            <reports>
              <report>project-team</report>
            </reports>
          </reportSet>
        </reportSets>
      </plugin>
    </plugins>
  </reporting>
  ...
</project>
```

Facilitator Notes:

Discuss about build reporting in detail.

In maven, build reporting plugins will be executed during the site phase. "**mvn site**" is used to build the reports. This phase picks the reports in **<reporting>** section defined in the reporting plugin.
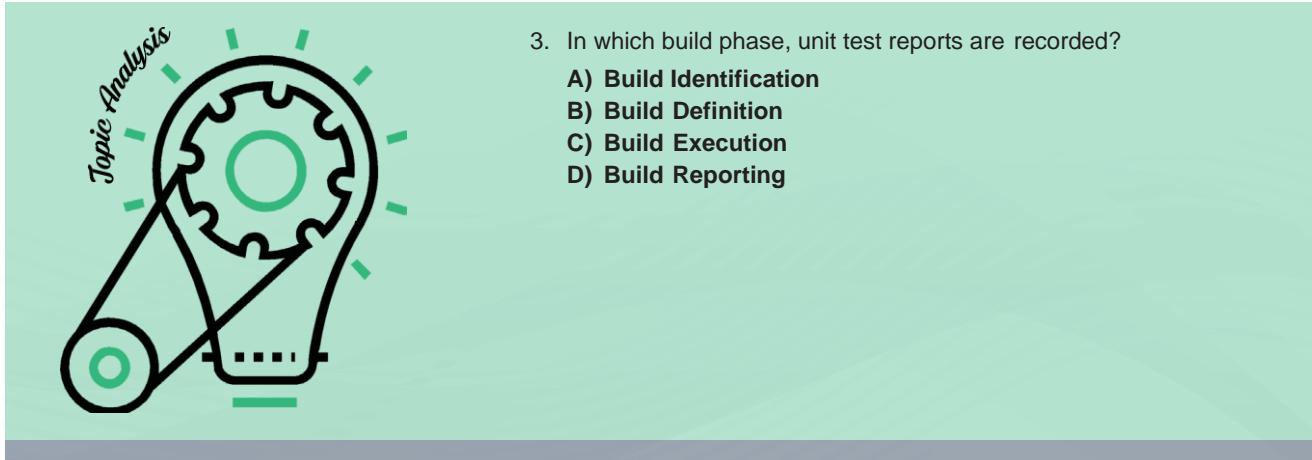
In the above diagram, **project-team** site reports will be generated if we run "**mvn site**".

What did You Grasp?

Topic Analysis

1. In which build phase, scripts are defined?
   A) **Build Identification**
   B) **Build Definition**
   C) **Build Execution**
   D) **Build Reporting**

2. In which build phase, only latest or agreed-on code changes are built?
   A) **Build Identification**
   B) **Build Definition**
   C) **Build Execution**
   D) **Build Reporting**

3. In which build phase, unit test reports are recorded?
   A) Build Identification
   B) Build Definition
   C) Build Execution
   D) Build Reporting

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.

Answer:

1. b. Build Definition

2. c. Build Execution

3. d. Build Reporting

3.1 Release Management

A release is made with an intention of being made available to end user. Release management comprises of following things:

Release management is concentrated on creating a software release in a disciplined way. Benefits of release management are as follows:
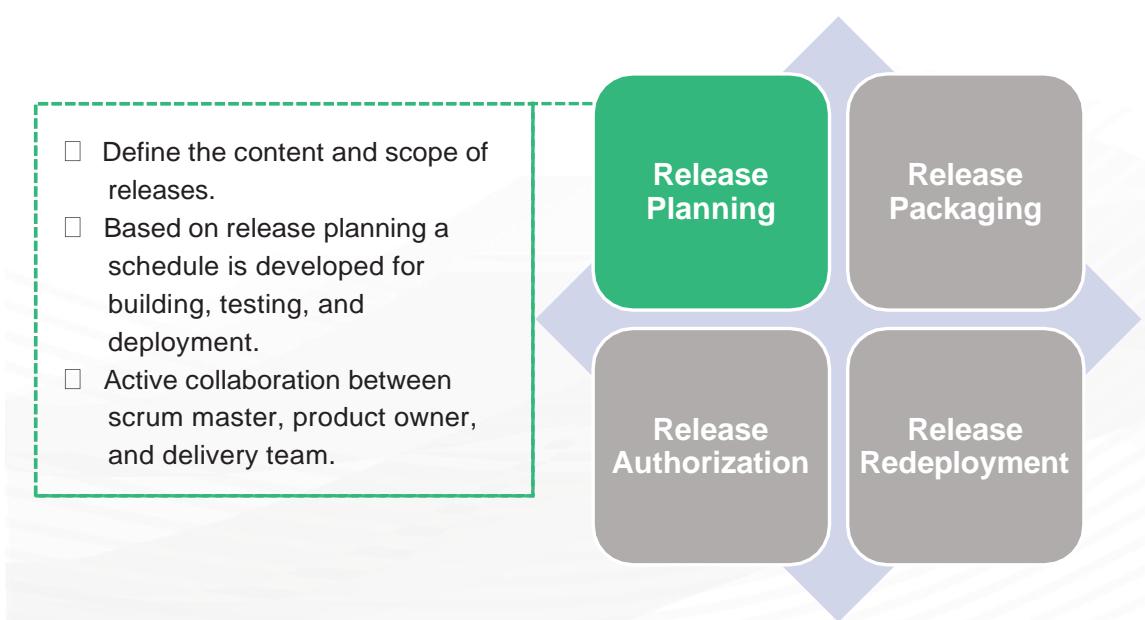
- **Efficiency** – Reduce overall time to find, fix, and deliver reliable, quality software.

- **Productivity** – Teams can focus on the more important things than trying to figure out what steps or version have caused the issue.

- **Reducing risk** –The ability to identify the when, where, what, and how for a release.

- **Collaboration between teams** – Cross team collaborations to discuss critical issues by using transparent and definite processes.

- **Configuration management** – Know more about environment setting, dependencies required in the production, test, staging and dev environments.

3.1.1 Release Planning

Release planning will set the expectation of the features to be implemented and completed. It involves following aspects:
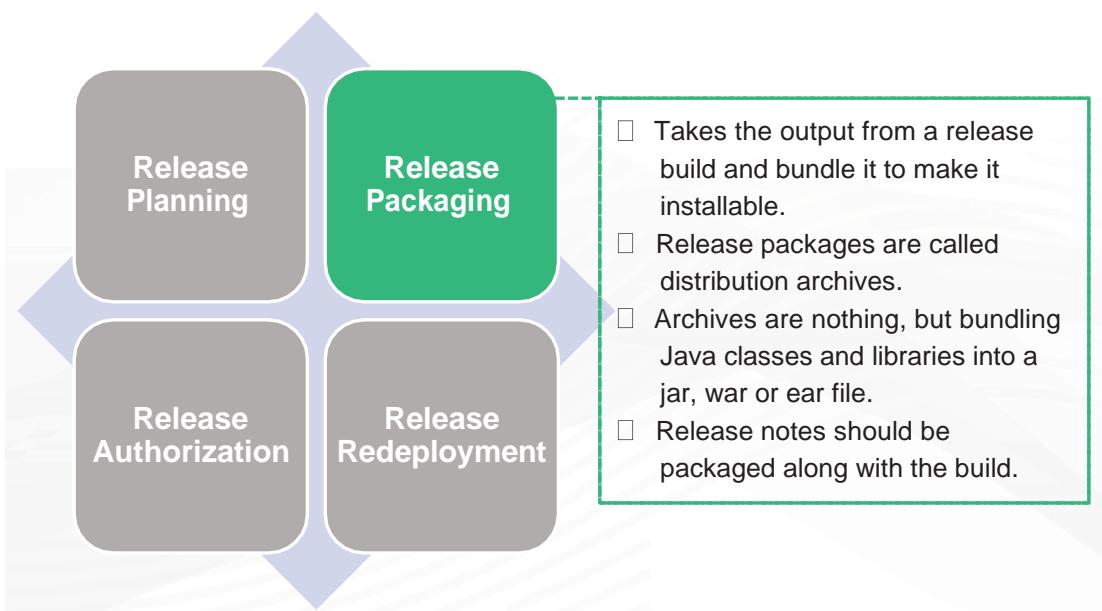
3.1.2 Release Packaging

Following are the key details of release packaging:

The output of a release build is bundled into an installable file and are called distribution archives. Archives are nothing but bundling set of source code, and required libraries into a jar, ear or war and make it available to be installed on a server.
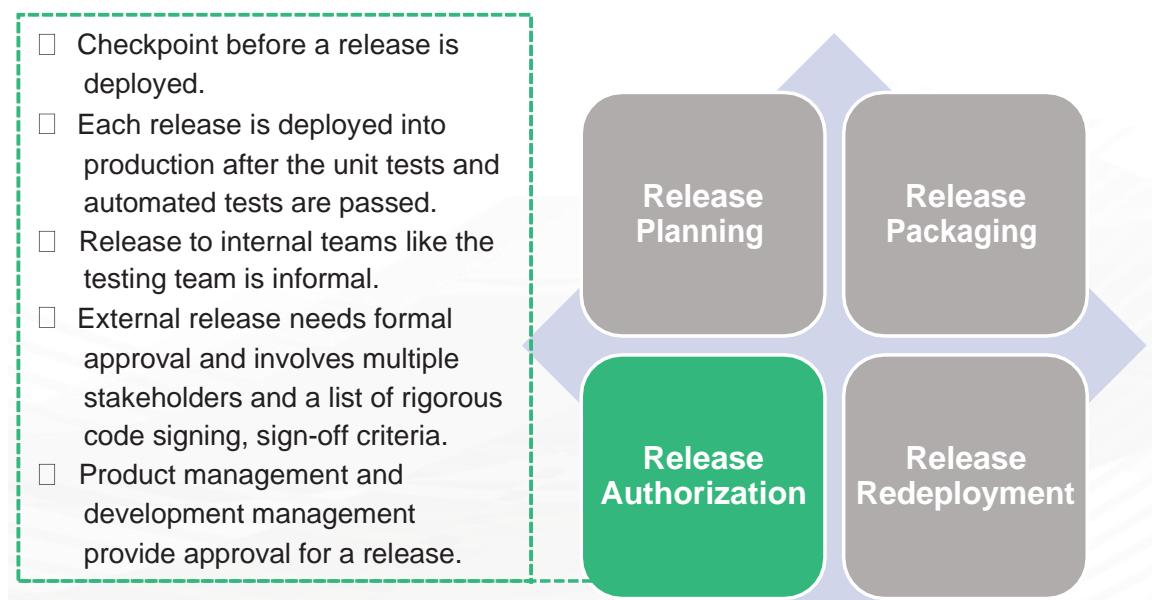
The key activities of release packaging are as follows:

- Integrate and assemble the components in a controlled manner

- Create the required documentation for release, including test plans, procedures, installations, and scripts

- Monitor the release build and how to detect and react to the problems from the release build

- Relevant parties should be notified about the availability of the release package

3.1.3 Release Authorization

Following are the key details of release authorization:

☐ Checkpoint before a release is deployed.
☐ Each release is deployed into production after the unit tests and automated tests are passed.
☐ Release to internal teams like the testing team is informal.
☐ External release needs formal approval and involves multiple stakeholders and a list of rigorous code signing, sign-off criteria.
☐ Product management and development management provide approval for a release.

Release Planning	Release Packaging
Release Authorization	Release Redeployment

Facilitator Notes:

Discuss about release authorization.

Before the build is released into production, approval is required from multiple stakeholders. 99% of the automated/unit tests should pass and the build should be properly code signed, bug-free, documented properly, adherence of coding standards, and all features are tested properly.

The Project manager is responsible to select the list of features to be released. Once the Sanity test is passed, the testing team will perform Functional Testing, Security Testing, Non-Functional Testing, System Testing, Load Testing, Performance Testing, User Acceptance Testing etc. If all the tests are passed release will be authorized.

3.1.4 Release Deployment

Following are the key details of release deployment:

Release Planning	**Release Packaging**
Release Authorization	**Release Redeployment**

☐ Release build is deployed after the approval.
☐ Release package is deployed into the live environment.
☐ Downtime should be minimized for a new release.
☐ Release deployments are automated.
☐ Train end users on usage of the product.

Facilitator Notes:

Explain the concept of release deployment.

Release deployment is deploying or updating the application from development to other environments like dev, test, stating, and finally to the production environment.

The main components of deployment are as follows:

● **Packaging** – of the components to be deployed

● **Dependencies** – Managing application and infrastructure dependencies for the deployment

● **Promotion** – Moving tested packages from one environment to another environment (e.g., Moving a package from development to testing, testing to staging, and staging to production)

● **Deployment** – using the release build package contents and install/configure the application

- **Compliance** – documenting audit results, and to validate the configuration of the application that is deployed

3.2 Best Practices for Build and Release Management

<table>
<tr>
<td>**Best Practices**</td>
<td>The best practices for build and release management are listed below:
↬ **Automate the build and release process through the CI/CD**
↬ **Clear requirements**
↬ Regression and integration testing before the release
↬ Make things **immutable**
↬ Run unit tests and smoke tests
↬ Sent build status notifications</td>
</tr>
</table>
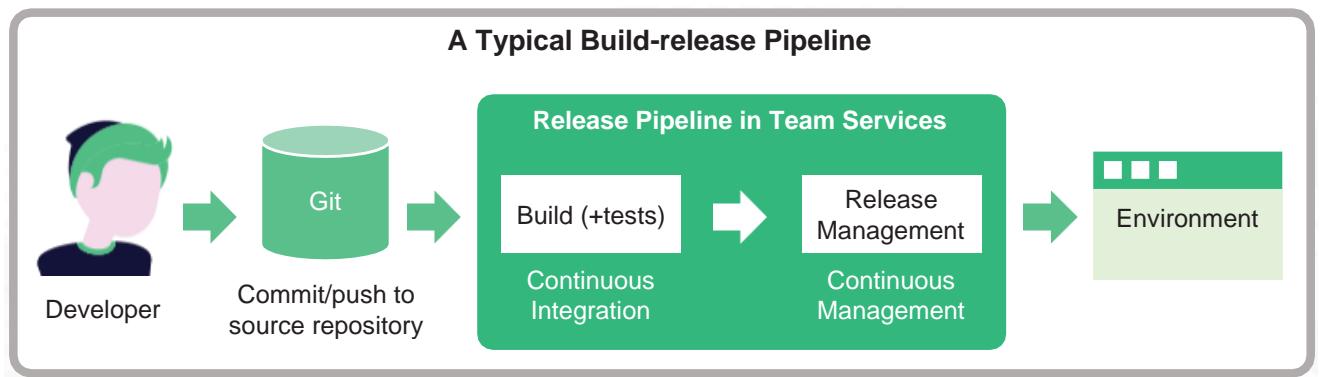
Facilitator Notes:

Discuss about best practices for build and release management.

- **Automate the build**, both on the development and testing side through the CI/CD. Automation reduces the cost and reduces the chance of making manual mistakes. This ensures reproducibility and repeatability of the entire build process.

- **Collect clear requirements**, because most of the defects found in the application can be tracked back to requirement collection.

- Run complete regression and integration testing before the release to make sure new changes will not break the existing system.

- Minimize the impact of a release by **eliminating or minimizing downtime** and t**esting for regressions** before release.

- Make things **immutable**: Deploy a new image that contains the configurations instead of using the existing machine. This avoids bugs from due to an unexpected series of actions.

- Create an extensive test suite (unit tests and smoke tests) for every build to find the bugs/ issues early.

- Send build notifications for every build whether the status success or failure. If the build fails, developers will understand the exact reason for failure in the notification. Generally, build notification would be sent through email or slack notification.

- Store the build logs

- Roll back strategy

- Tagging the release

A Typical Build-release Pipeline

Release Pipeline in Team Services

Git
Developer
Commit/push to source repository

Build (+tests)
Continuous Integration

Release Management
Continuous Management

Environment

Facilitator Notes:

Discuss about best practices for build and release management.

- Store the build logs, so that developers can always go back the refer the logs of any particular build to view the build compilation reports.

- In some scenarios build deployment will break the application. A careful approach should be followed to rollback the production deployment because sometimes the rollback might cause more issues.

- It is important to tag every release. Tag creates a snapshot of the source code at that point in time. In future point of time, it can be used to rollback or build the source code that is tagged.

What did You Grasp?

1. Which of the following phase delivers deployment archives?
   A) Release planning
   B) Release packaging
   C) Release authorization
   D) Release deployment

2. Which of the following phase involves code signing and sign-off criteria for a release ?
   A) Release planning
   B) Release packaging
   C) Release authorization
   D) Release deployment

3. In which phase content and scope of a release is set?
   A) Release planning
   B) Release packaging
   C) Release authorization
   D) Release deployment

4. In which phase approval is required to release a build?
   A) Release planning
   B) Release packaging
   C) Release authorization
   D) Release deployment

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.

Answer:

1. b. Release packaging

2. b. Release authorization

3. a. Release planning

4. d. Release deployment

Group Discussion

Facilitator Notes:

Divide the students into groups and discuss about build/release management aspects and best practices of build/release management in detail.

We've so far discussed about different aspects of build and release management and best practices of build and release management. Form different groups and each group should talk about one concept in detail along with analogies or examples to show your understanding.

4.1 Declarative Dependency Management

Following are the key details of declarative dependency management:

Dependency management enables a large build to be broken into smaller component builds, and each builds publishing their output into a common repository for use by other dependent modules.

Dependency management can be used to control third-party libraries that are used in the project.

The master project contains a build file that helps in building the child modules in the correct order by using the declared dependencies.

Maven is one of the build tools that converts source code into executable applications. The main challenges with using a third party library is to manage its dependencies. Maven provides an option for declarative dependency management. Every module can declare the dependency version, that is required to build the application. Maven will automatically download these dependencies into the classpath.

In a nutshell, we learnt:

1. Build is a version of the application or program identified by build number
2. Build and release management is the process of managing, planning, scheduling and controlling a software build throughout its life cycle
3. Build identification is used to identify what changes to build
4. Build definition is the phase where build scripts are created
5. Build execution phase depends on the scripts defined in the build definition
6. Build reporting phase the build execution results are collected
7. Release management comprises of planning, packaging, authorization and deployment
8. Build abstraction provides the ability to bundle all the classes into a single jar
9. Declarative dependency management is used to centralize the dependency information