

# Maven Overview

Author: Nho Luong  
Skill: DevOps Engineer Lead

Microsoft

Nho Luong

has successfully passed all requirements for

Microsoft Certified: Azure Solutions Architect Expert

Credential ID: 3CF3D14D253FE551

Certification number: 46F8FB-7D3F5B

Earned on: June 14, 2024

Expires on: June 15, 2025

Online Verifiable

Microsoft Certified

EXPERT

aws

certified

nho luong

AWS Certified Solutions Architect - Professional

VALIDATION NUMBER: 119ee670510e4bb99fe2956152effb7e

VALIDATE AT: <https://aws.amazon.com/verification>

Issue Date: June 21, 2024

Expiration Date: June 21, 2027

Alpine

aws

HOME

PROFILE

EXAM REGISTRATION

EXAM HISTORY

CERTIFICATIONS

Certification status

Agreement

BENEFITS

DIGITAL BADGES

SUPPORT AND FAQS

Certification status

NHO LUONG

AWS04440150

Active

PROFESSIONAL

AWS Certified Solutions Architect - Professional

SAP

ACTIVE DATE

2024-06-21

EXPIRATION DATE

2027-06-21

VIEW MORE

Active

SPECIALTY

AWS Certified Security - Specialty

SCS

ACTIVE DATE

2024-06-19

EXPIRATION DATE

2027-06-19

VIEW MORE

Active

PROFESSIONAL

AWS Certified DevOps Engineer - Professional

DOP

ACTIVE DATE

2024-06-17

EXPIRATION DATE

2027-06-17

VIEW MORE

Active

ASSOCIATE

AWS Certified Solutions Architect - Associate

SAA

ACTIVE DATE

2024-06-15

EXPIRATION DATE

2027-06-21

VIEW MORE

Active

FOUNDATIONAL

AWS Certified Cloud Practitioner

CLF

ACTIVE DATE

2024-05-27

EXPIRATION DATE

2027-06-21

Activity

Training

Plans

Challenges

Credentials

Q&A

Achievements

Collections

Transcript

4 items

Sorted by expiration date

CERTIFICATION

Microsoft Certified: DevOps Engineer Expert

Expires on June 15, 2025 at 6:59 AM (UTC +07:00) • Earned on June 14, 2024

View certification details

Online Verifiable

Active

CERTIFICATION

Microsoft Certified: Azure Solutions Architect Expert

Expires on June 15, 2025 at 6:59 AM (UTC +07:00) • Earned on June 14, 2024

View certification details

Online Verifiable

Active

CERTIFICATION

Microsoft Certified: Azure Administrator Associate

Expires on June 15, 2025 at 6:59 AM (UTC +07:00) • Earned on June 14, 2024

View certification details

Online Verifiable

Active

CERTIFICATION

Microsoft Certified: Azure Fundamentals

Earned on May 24, 2024

View certification details

Online Verifiable

Active



# Build Automation Tool

---



# Why Automated Builds



Automated Build are best practice.



Manual procedures are mistake prone.



Automated builds are self documentary.



Automated Builds can be triggered by other tools(eg Jenkins,cron jobs).

# Code Build or Integration



Code Build is a process by which source code is converted into a standalone form that can be run on a particular type of platform.



One of the most important step of the software build is the compilation process, where the source code files are converted into executable files.



The process of building software is usually managed by build tool.



Build has been created when a certain point in development has been reached or the code is deemed ready for implementation., either for testing or release.

# Maven Introduction

Apache Maven is a software project management and comprehension tool.

Maven is a build automation tool used primarily for Java projects.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

Maven addresses two aspects of building software: how software is built, and its dependencies.

# Project Object Model(POM)

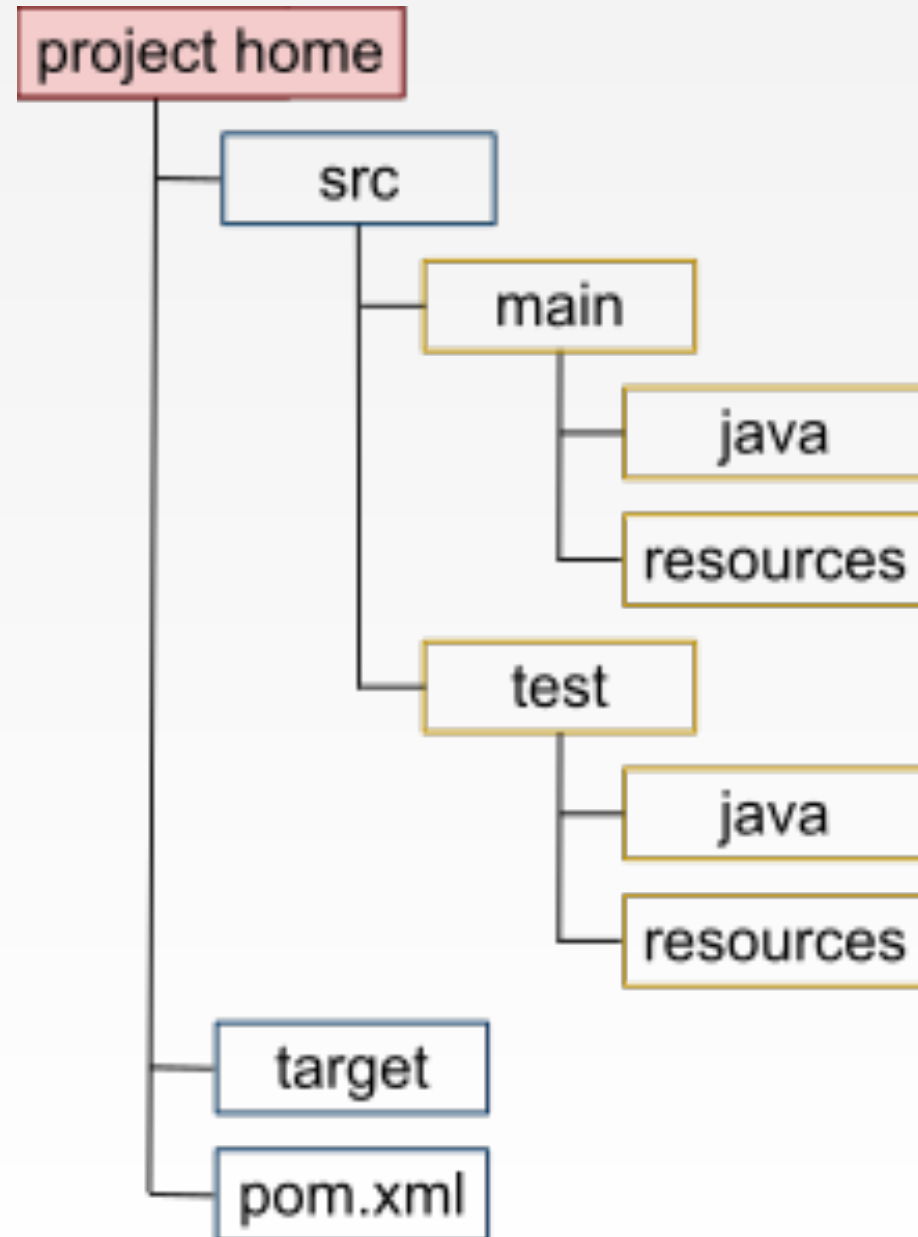
- Maven projects are configured

using a Project Object Model,  
which is stored in a pom.xml-file.

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>
  <!-- project coordinates, i.e. a group of values which uniquely identify this project -->
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>
  <!-- library dependencies -->
  <dependencies>
    <dependency>
      <!-- coordinates of the required library -->
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <!-- this dependency is only used for running and compiling tests -->
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

# A directory structure for a Java project auto-generated by Maven

---



# Directory structure of a Maven project

---

Directory name	Purpose
project home	Contains the pom.xml and all subdirectories.
src/main/java	Contains the deliverable Java sourcecode for the project.
src/main/resources	Contains the deliverable resources for the project, such as property files.
src/test/java	Contains the testing Java sourcecode (JUnit or TestNG test cases, for example) for the project.
src/test/resources	Contains resources necessary for testing.



# Apache Maven Project

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example.training</groupId>
  <artifactId>maven-training</artifactId>
  <version>1.0</version>
</project>
```

- Maven uniquely identifies a project using:

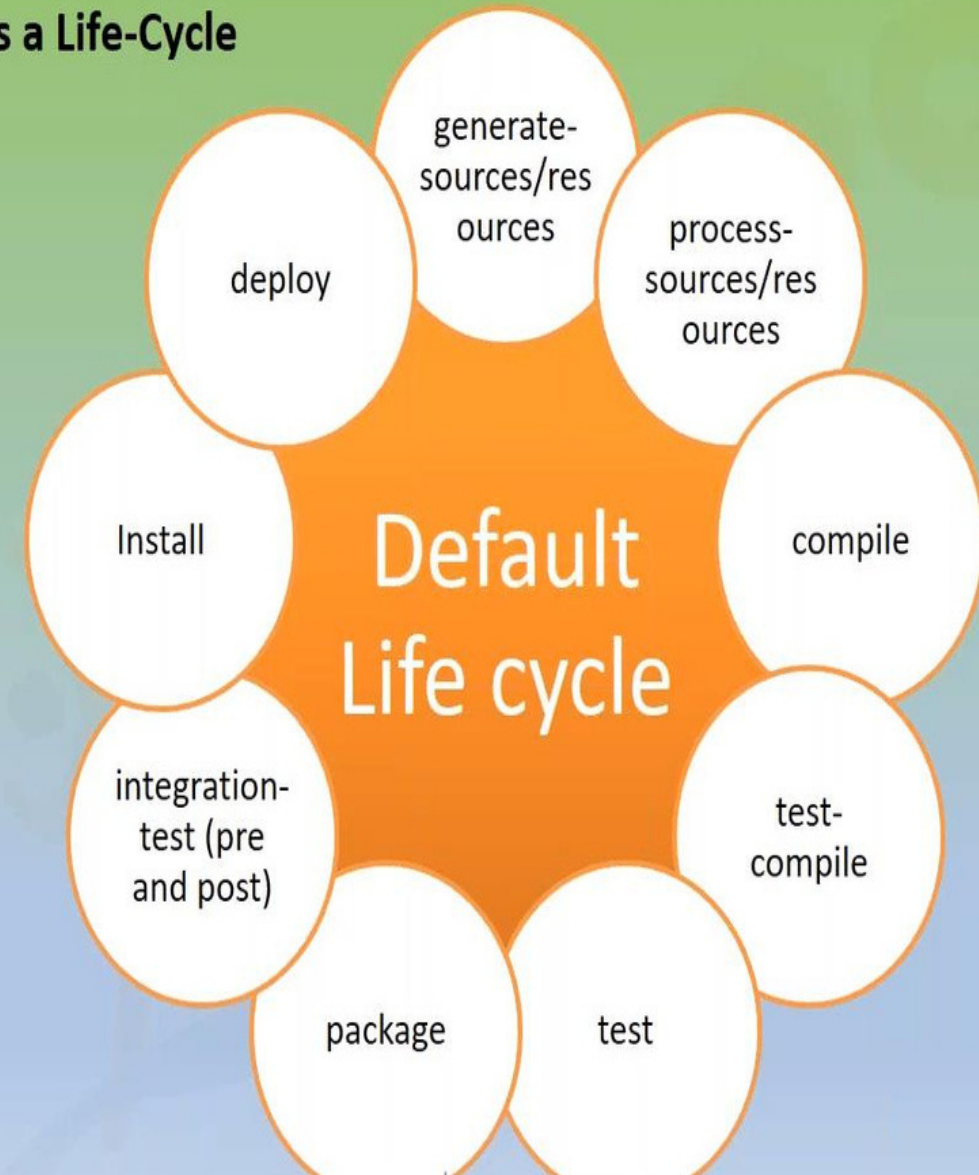
**groupId:** Arbitrary project grouping identifier

**artifactid:** Arbitrary name of the project.

**Version:** version of the project

# Maven Build Life Cycle

## Every Maven Build has a Life-Cycle

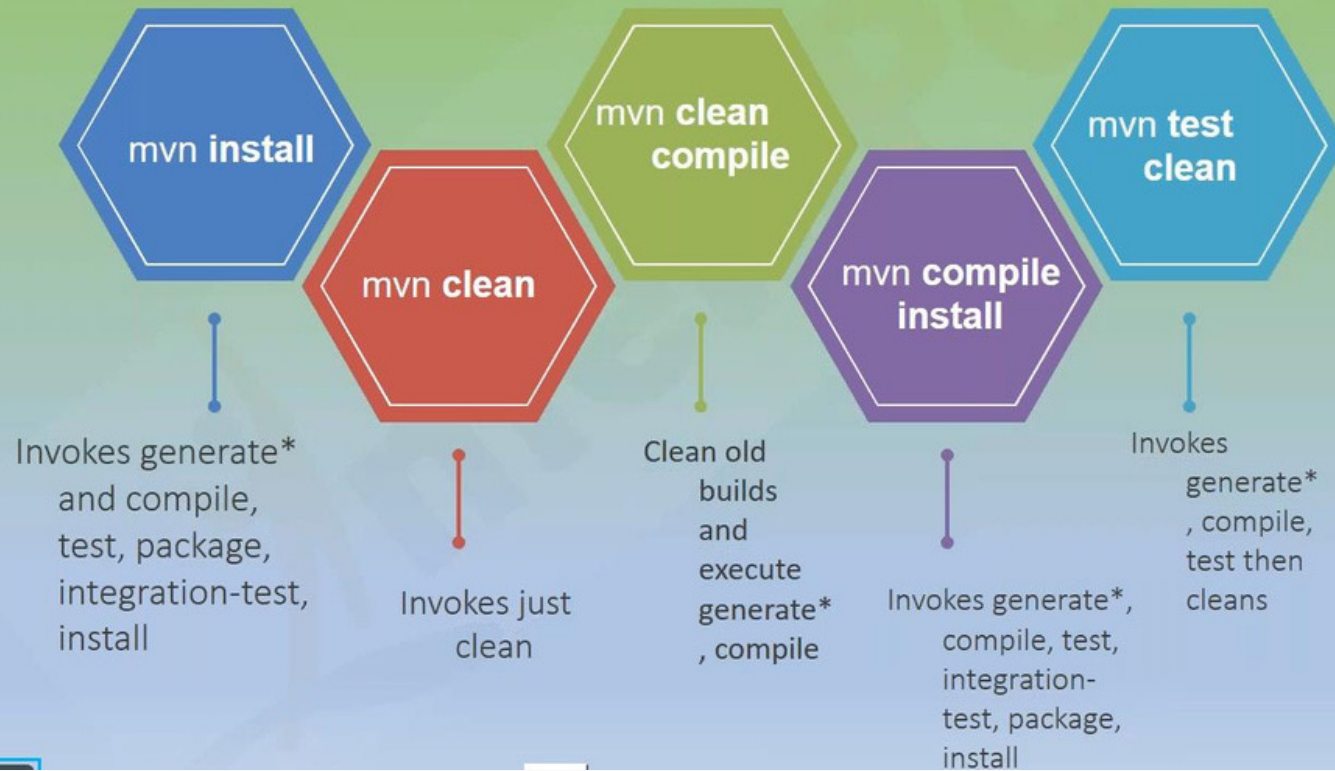


# Maven Build Lifecycle

---

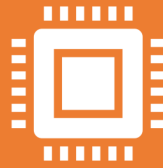
Phase	Handles	Description
prepare-resources	resource copying	Resource copying can be customized in this phase. Validates if the project is
validate	Validating the information	correct and if all necessary information is available.
compile	compilation	Source code compilation is done in this phase.
Test	Testing	Tests the compiled source code suitable for testing framework.
package	packaging	This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml. This phase installs the package in local/remote maven repository.
install	installation	Copies the final package to the remote repository.
Deploy	Deploying	

voke a Maven build you set a lifecycle “goal”



# Apache Maven Goals

# Apache Maven Dependency Management



Maven's dependency-handling mechanism is organized around a coordinate system identifying individual artifact such as software libraries and modules.



No need to store the libraries in SCM.



A central maven repository.

# Apache Maven Dependency Management

- Adding a Dependency.

- Dependency consist of

1. GAV

2. Scope: compile, test, provided  
(default=compile)

3. Type: jar, pom, war

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
```



**Thank You**