mvn有哪些命令

**用户:**

mvn有哪些命令

**元宝:**

MVN（Maven）是一个强大的项目管理和构建工具，以下是常用的 Maven 命令及其用途：

### **1. 项目构建与编译**

* **mvn clean**清理项目，删除 target 目录（移除所有编译生成的文件）。
* **mvn compile**编译项目源代码，生成类文件到 target/classes。
* **mvn test**运行单元测试（位于 src/test/java），生成测试报告到 target/surefire-reports。
* **mvn package**打包项目（生成 JAR/WAR 文件到 target/），根据 pom.xml 中 <packaging> 配置。
* **mvn install**将生成的包安装到本地 Maven 仓库（默认在 ~/.m2/repository），供其他项目依赖。
* **mvn verify**执行所有检查（编译、测试、打包等），确保项目符合质量标准。

### **2. 依赖与仓库管理**

* **mvn dependency:tree**显示项目依赖树，查看传递性依赖冲突。
* **mvn dependency:resolve**下载并解析所有依赖。
* **mvn dependency:purge-local-repository**清除本地仓库中的依赖缓存（重新下载）。
* **mvn install:install-file**手动安装本地 JAR 到仓库（需指定 -Dfile=path/to.jar）。

### **3. 项目生成与更新**

* **mvn archetype:generate**使用 Maven 原型（模板）创建新项目。
* **mvn versions:update-parent**更新项目的父 POM 版本。
* **mvn versions:use-latest-versions**将依赖更新为最新版本。

### **4. 高级构建与多模块**

* **mvn clean install -DskipTests**跳过测试阶段快速构建。
* **mvn -pl module-name -am**仅构建指定模块及其依赖（多模块项目）。
  + 示例：mvn -pl core-module install
* **mvn deploy**将包部署到远程仓库（需配置 distributionManagement）。

### **5. 插件与自定义**

* **mvn help:effective-pom**查看合并所有父 POM 后的最终有效 POM。
* **mvn [phase] -Pprofile-id**激活指定 Profile（如 -Pprod）。
* **mvn site**生成项目文档站点（需配置 maven-site-plugin）。

### **常用组合命令**

* **清理并重新构建**
* **跳过测试快速打包**
* **多线程构建（加速）**

### **注意事项**

* 命令需在包含 pom.xml 的目录下执行。
* 部分命令需配合参数使用（如 -D 指定属性）。
* 可通过 mvn help:describe -Dcmd=<plugin-prefix> 查看插件详情。

如果需要更详细的帮助，可以运行 mvn --help 或查阅 [Maven 官方文档](https://maven.apache.org/)。

**用户:**

拉取依赖的时候提示 the trustAnchors parameter must be non-empty 怎么检查maven和jdk环境变量问题

**元宝:**

当 Maven 拉取依赖时出现 **the trustAnchors parameter must be non-empty** 错误，通常与 JDK 的 SSL 证书信任链配置或环境变量问题有关。以下是逐步检查和解决方法：

### **1. 检查 JDK 环境变量**

#### **(1) 确认 JDK 安装路径**

* 运行以下命令，检查 JAVA\_HOME 是否指向有效的 JDK 目录：

确保路径正确（如 /usr/lib/jvm/java-11-openjdk 或 C:\Program Files\Java\jdk-17）。

#### **(2) 检查 PATH 是否包含 JDK**

* 确保 PATH 中包含 JDK 的 bin 目录：

如果缺失，手动添加：

#### **(3) 验证 Java 版本**

* 运行以下命令，确认 Java 和 Maven 使用的 JDK 版本一致：

如果版本不一致（如 Maven 使用了其他 JDK），需调整 JAVA\_HOME。

### **2. 检查 Maven 配置**

#### **(1) 确认 Maven 的 settings.xml**

* 检查 Maven 的全局配置文件（通常位于 $M2\_HOME/conf/settings.xml 或用户目录的 ~/.m2/settings.xml）：
  + 确保 <mirrors> 或 <proxies> 配置正确，未指向无效的仓库或代理。
  + 临时注释掉所有镜像和代理配置，尝试直接连接中央仓库。

#### **(2) 清理 Maven 本地仓库**

* 删除本地仓库中可能损坏的依赖缓存：

然后重新拉取依赖：

### **3. 修复 SSL 证书信任链问题**

#### **(1) 检查 JDK 的证书库**

* 错误通常因 JDK 的 cacerts 文件（证书信任库）损坏或缺失导致。定位到 JDK 的证书库路径（如 $JAVA\_HOME/lib/security/cacerts），确认文件存在且非空。

#### **(2) 重新生成证书库**

* 如果 cacerts 损坏，尝试从其他正常机器复制该文件，或重新安装 JDK。

#### **(3) 临时绕过 SSL 验证（不推荐）**

* **仅限测试环境**，可通过 Maven 参数禁用 SSL 验证：

或在 settings.xml 中添加：

### **4. 其他可能原因**

#### **(1) 系统时间错误**

* 如果系统时间不正确，可能导致 SSL 证书验证失败。确保时间与当前时区匹配：

#### **(2) 防火墙/代理拦截**

* 检查网络是否阻止了对 Maven 仓库（如 repo.maven.apache.org）的访问。尝试直接访问仓库 URL：

### **总结步骤**

1. **确认 JAVA\_HOME 和 PATH 配置正确** → 修复环境变量。
2. **清理 Maven 本地仓库** → 删除 ~/.m2/repository。
3. **检查 JDK 的 cacerts 文件** → 修复或替换证书库。
4. **临时禁用 SSL 验证（测试用）** → 确认是否为证书问题。
5. **检查网络和系统时间** → 排除外部因素。

如果问题仍未解决，提供完整的错误日志（运行命令时加 -X 参数）可进一步分析：

**用户:**

echo %JAVA\_HOME% 执行这个命令返回的还是echo %JAVA\_HOME%

**元宝:**

如果执行 echo %JAVA\_HOME% 后返回的仍然是 echo %JAVA\_HOME%，而不是实际的 JDK 路径，说明 **JAVA\_HOME 环境变量未正确设置**，或者当前终端会话未加载该变量。以下是解决方法：

### **1. 检查 JAVA\_HOME 是否已定义**

#### **Windows 系统**

* **方法 1：通过命令查看**运行以下命令，检查是否返回有效的 JDK 路径：
  + 如果无输出，说明变量未定义。
  + 如果返回路径，但 echo %JAVA\_HOME% 仍无效，可能是当前终端未继承环境变量。
* **方法 2：通过系统属性查看**

1. 打开 **控制面板 > 系统和安全 > 系统 > 高级系统设置 > 环境变量**。
2. 在 **系统变量** 或 **用户变量** 中查找 JAVA\_HOME。
3. 如果不存在，需要手动添加（见下一步）。

#### **Linux/Mac 系统**

* 运行以下命令检查：

如果无输出，说明变量未定义。

### **2. 设置 JAVA\_HOME 环境变量**

#### **Windows**

1. **找到 JDK 安装路径**

* 默认路径类似：C:\Program Files\Java\jdk-21（具体版本号可能不同）。
* 如果不确定，在文件资源管理器中搜索 java.exe，其父目录即为 JDK 路径（如 ...\jdk-21\bin\java.exe）。

1. **手动添加变量**

* 打开 **环境变量** 窗口（方法见上文）。
* 在 **系统变量** 中点击 **新建**：
  + 变量名：JAVA\_HOME
  + 变量值：JDK 安装路径（如 C:\Program Files\Java\jdk-21）。
* 点击 **确定** 保存。

1. **更新 PATH**

* 在 **系统变量** 中找到 PATH，点击 **编辑**。
* 添加 %JAVA\_HOME%\bin，确保它位于其他 Java 路径之前。
* 保存所有更改。

#### **Linux/Mac**

* 编辑 ~/.bashrc 或 ~/.zshrc 文件，添加以下内容：
* 使配置生效：

### **3. 验证配置**

* **重新打开终端**（确保加载新环境变量）。
* 运行以下命令验证：

如果 java -version 和 mvn -v 显示的版本不一致，说明 Maven 未使用 JAVA\_HOME 指定的 JDK，需检查 Maven 配置或重启系统。

### **4. 常见问题**

#### **Q1: 设置后仍无效**

* **原因**：终端会话未重新加载环境变量。**解决**：关闭并重新打开终端，或重启系统。

#### **Q2: 路径包含空格或特殊字符**

* 如果 JDK 路径包含空格（如 C:\Program Files\Java\...），确保在 JAVA\_HOME 中直接使用路径，无需引号：

在 PATH 中引用时使用 %JAVA\_HOME% 即可自动处理空格。

#### **Q3: 多版本 JDK 冲突**

* 如果安装了多个 JDK，确保 JAVA\_HOME 指向目标版本，且 PATH 中其 bin 目录优先级最高。

### **总结**

* echo %JAVA\_HOME% 无输出 → 变量未定义。
* 通过系统设置或命令行添加 JAVA\_HOME 并更新 PATH。
* 重新启动终端后验证 java -version 和 mvn -v。

如果仍有问题，请提供以下信息进一步排查：

1. 操作系统版本。
2. JDK 安装路径。
3. 完整的环境变量 PATH 内容（运行 echo %PATH%）。

**用户:**

如果返回路径，但 echo %JAVA\_HOME% 仍无效，可能是当前终端未继承环境变量。 即使这种情况 怎么解决

**元宝:**

如果 set JAVA\_HOME 能返回正确的 JDK 路径，但 echo %JAVA\_HOME% 仍然无效（直接输出 %JAVA\_HOME% 而不是路径），说明 **环境变量未正确加载到当前终端会话**。以下是具体解决方案：

### **1. 原因分析**

* **环境变量未刷新**：新配置的 JAVA\_HOME 未加载到当前已打开的终端（Windows CMD/PowerShell 不会自动继承新设置的环境变量）。
* **变量名拼写错误**：可能大小写不一致（如 JAVA\_HOME vs java\_home）。
* **终端类型限制**：某些终端（如 VS Code 内置终端）可能需要手动刷新环境。

### **2. 解决方法**

#### **方法 1：关闭并重新打开终端**

* **最简单有效的方式**：直接关闭当前所有 CMD/PowerShell/终端窗口，重新打开一个新窗口，再执行 echo %JAVA\_HOME%。
* **原理**：新终端会重新加载系统环境变量。

#### **方法 2：手动刷新环境变量（不重启终端）**

* **Windows CMD**：运行以下命令强制刷新：

如果 refreshenv 不可用（需安装 Windows SDK 或手动定义函数），改用：

* **Windows PowerShell**：重新加载环境变量：
* **Linux/Mac**：执行 source ~/.bashrc 或 source ~/.zshrc（根据使用的 shell）。

#### **方法 3：检查变量作用域**

* 如果 JAVA\_HOME 是在 **用户变量** 中设置的，但当前终端以管理员身份运行，可能无法读取用户变量。**解决**：

1. 将 JAVA\_HOME 添加到 **系统变量**（控制面板中配置）。
2. 或者以普通用户身份运行终端。

#### **方法 4：验证变量名和值**

* 检查变量名是否严格匹配（包括大小写）：

如果显示 JAVA\_HOME=...，但 echo %JAVA\_HOME% 仍无效，可能是终端解析问题，尝试：

#### **方法 5：绕过终端直接验证**

* 通过 Java 命令验证环境变量是否实际生效：

如果 java -version 显示的版本与 %JAVA\_HOME% 不一致，说明 PATH 未正确配置。

### **3. 特殊情况处理**

#### **场景 1：VS Code 终端无效**

* VS Code 的终端默认不会实时继承系统环境变量。**解决**：

1. 关闭 VS Code，重新启动。
2. 或在 VS Code 终端中手动设置变量：

#### **场景 2：脚本或 IDE 中无效**

* 如果通过脚本（如 .bat 或 .sh）或 IDE（如 IntelliJ）调用时无效，需确保它们正确加载了环境变量：
  + **IntelliJ**：重启 IDE，或在 Run/Debug Configurations 中手动指定 JAVA\_HOME。
  + **脚本**：在脚本开头显式设置变量：

### **4. 终极排查步骤**

如果以上方法均无效，按顺序执行以下命令：

### **总结**

* **优先重启终端**：90% 的问题可通过关闭并重新打开终端解决。
* **手动刷新变量**：使用 refreshenv 或脚本强制加载。
* **检查作用域和拼写**：确保变量名一致且作用域正确（系统/用户）。
* **验证实际效果**：通过 java -version 确认 JDK 是否匹配。

如果问题依旧，可能是系统环境损坏，建议重新安装 JDK 并彻底检查环境变量。

**用户:**

通过cmd打开输入是可以的 通过powershell输入命令不可以

**元宝:**

如果 echo %JAVA\_HOME% 在 **CMD** 中可以正确输出 JDK 路径，但在 **PowerShell** 中无效，这是因为 **CMD 和 PowerShell 使用不同的语法和环境变量访问方式**。以下是具体原因和解决方法：

## **1. 原因分析**

### **(1) 环境变量引用语法不同**

* **CMD** 使用 %变量名%（如 %JAVA\_HOME%）。
* **PowerShell** 使用 $env:变量名（如 $env:JAVA\_HOME）。

### **(2) 环境变量作用域问题**

* **PowerShell** 默认不会自动加载 **系统/用户环境变量**（特别是新打开的会话）。
* **CMD** 会自动继承系统环境变量，但 PowerShell 可能需要手动刷新。

## **2. 解决方法**

### **方法 1：在 PowerShell 中使用正确的语法**

PowerShell 访问环境变量的正确方式是：

而不是 %JAVA\_HOME%（CMD 语法）。

**示例：**

### **方法 2：手动刷新 PowerShell 环境变量**

如果 $env:JAVA\_HOME 仍然返回空，可能是环境变量未加载。可以手动刷新：

### **方法 3：检查 PowerShell 的启动配置**

如果每次打开 PowerShell 都遇到此问题，可以修改 $PROFILE 文件自动加载环境变量：

在 $PROFILE 文件中添加：

然后重新启动 PowerShell。

### **方法 4：检查是否 PATH 包含 Java**

如果 $env:JAVA\_HOME 仍然无效，但 java -version 能运行，说明 Java 已经安装在 PATH 中，但 JAVA\_HOME 变量未正确设置。可以手动设置：

### **方法 5：检查系统环境变量**

如果 PowerShell 仍然无法读取 JAVA\_HOME，可能是 **系统环境变量未正确设置**：

1. **打开“环境变量”设置**：

* Win + R → 输入 sysdm.cpl → 选择 **高级** → **环境变量**。

1. **检查 JAVA\_HOME 是否在“系统变量”中**：

* 如果没有，手动添加：
  + **变量名**: JAVA\_HOME
  + **变量值**: JDK 路径（如 C:\Program Files\Java\jdk-21）

1. **确保 PATH 包含 %JAVA\_HOME%\bin**：

* 在 PATH 变量中添加 %JAVA\_HOME%\bin（如果不存在）。

## **3. 验证是否修复**

在 **PowerShell** 中运行：

如果 java -version 和 mvn -v 显示的版本一致，说明修复成功。

## **4. 常见问题**

### **Q1: 为什么 CMD 可以，但 PowerShell 不行？**

* **CMD** 使用 %变量名%，而 **PowerShell** 使用 $env:变量名。
* **PowerShell** 不会自动加载所有系统环境变量，可能需要手动刷新。

### **Q2: 修改后仍然无效？**

* 检查是否 **以管理员身份运行 PowerShell**（某些系统变量需要管理员权限）。
* 确保 **JDK 安装路径正确**（如 C:\Program Files\Java\jdk-21）。
* 尝试 **重启电脑**，确保所有环境变量生效。

## **总结**

|  |  |
| --- | --- |
| 问题 | 解决方法 |
| **echo %JAVA\_HOME% 在 CMD 有效，但 PowerShell 无效** | 在 PowerShell 使用 $env:JAVA\_HOME |
| **$env:JAVA\_HOME 仍然为空** | 手动刷新：$env:JAVA\_HOME = [System.Environment]::GetEnvironmentVariable("JAVA\_HOME", "Machine") |
| **每次打开 PowerShell 都要重新设置** | 修改 $PROFILE 文件自动加载环境变量 |
| **java -version 能运行，但 $env:JAVA\_HOME 为空** | 手动设置 $env:JAVA\_HOME 或检查系统环境变量 |

如果仍然有问题，请提供：

1. $env:JAVA\_HOME 的输出。
2. java -version 的输出。
3. mvn -v 的输出。
4. 是否使用了 **管理员权限** 运行 PowerShell。

**用户:**

明明JAVA\_HOME配置的是1.8 为什么java -version命令返回的jdk11

**元宝:**

如果 **JAVA\_HOME 配置的是 JDK 1.8**，但运行 java -version 返回的是 **JDK 11**，说明 **系统实际使用的 Java 版本与 JAVA\_HOME 不一致**。以下是原因和解决方案：

## **1. 原因分析**

### **(1) PATH 环境变量优先级问题**

* PATH 变量中可能包含多个 Java 路径，且 **JDK 11 的路径在 PATH 中比 JDK 1.8 更靠前**，导致系统优先使用 JDK 11。
* **示例**：

这里 jdk-11\bin 在 jdk1.8.0\_301\bin 之前，所以 java -version 返回 JDK 11。

### **(2) 未正确设置 JAVA\_HOME**

* JAVA\_HOME 可能被其他程序（如 IDE、Maven、系统服务）覆盖。
* **检查方式**：

如果返回的不是 JDK 1.8 路径，说明 JAVA\_HOME 被修改。

### **(3) 多个 JDK 安装导致冲突**

* 如果系统安装了多个 JDK（如 JDK 8 和 JDK 11），并且 PATH 中包含了多个 Java 路径，可能导致版本混乱。

## **2. 解决方案**

### **方法 1：调整 PATH 顺序**

1. **查看当前 PATH**：
2. **确保 %JAVA\_HOME%\bin 在 PATH 的最前面**：

* 打开 **环境变量设置**（Win + R → sysdm.cpl → **高级** → **环境变量**）。
* 编辑 PATH，将 %JAVA\_HOME%\bin 移到最顶部：

1. **重新打开 CMD/PowerShell**，检查：

### **方法 2：手动指定 Java 路径**

如果不想修改 PATH，可以 **直接使用完整路径** 运行 Java：

这样会强制使用 JAVA\_HOME 指定的 JDK。

### **方法 3：检查 JAVA\_HOME 是否被覆盖**

某些程序（如 IDE、Maven）可能会覆盖 JAVA\_HOME：

* **在 CMD 中运行**：

返回的路径如果不是 %JAVA\_HOME%\bin\java.exe，说明 PATH 中有其他 Java。

### **方法 4：卸载冲突的 JDK**

如果不需要 JDK 11，可以 **卸载它**：

1. **控制面板 → 卸载程序**，找到 JDK 11 并卸载。
2. **删除残留的 Java 路径**：

* 检查 PATH 是否仍然包含 JDK 11 的路径，手动删除。

### **方法 5：使用 update-alternatives（Linux/Mac）**

如果是 **Linux/Mac**，可以使用 update-alternatives 切换默认 Java：

然后选择 JDK 1.8。

## **3. 验证修复**

1. **检查 JAVA\_HOME**：

确保返回的是 JDK 1.8 路径（如 C:\Program Files\Java\jdk1.8.0\_301）。

1. **检查 PATH**：

确保 %JAVA\_HOME%\bin 在最前面。

1. **检查 java -version**：

应该返回类似：

## **4. 常见问题**

### **Q1: 为什么 JAVA\_HOME 是对的，但 java -version 还是 JDK 11？**

* **原因**：PATH 中有其他 Java 路径优先级更高。
* **解决**：调整 PATH，确保 %JAVA\_HOME%\bin 在最前面。

### **Q2: 修改 PATH 后仍然无效？**

* **可能原因**：终端会话未刷新环境变量。
* **解决**：关闭并重新打开 CMD/PowerShell。

### **Q3: 如何彻底删除 JDK 11？**

1. **卸载 JDK 11**（控制面板 → 卸载程序）。
2. **删除残留的 Java 文件夹**（如 C:\Program Files\Java\jdk-11）。
3. **清理 PATH**，移除 JDK 11 的路径。

## **总结**

|  |  |
| --- | --- |
| 问题 | 解决方案 |
| **JAVA\_HOME 是 JDK 1.8，但 java -version 返回 JDK 11** | 调整 PATH，确保 %JAVA\_HOME%\bin 在最前面 |
| **PATH 中有多个 Java 路径** | 删除或调整顺序，让 JDK 1.8 优先 |
| **JDK 11 残留导致冲突** | 卸载 JDK 11 并清理 PATH |
| **终端未刷新环境变量** | 关闭并重新打开 CMD/PowerShell |

如果仍然有问题，请提供：

1. echo %JAVA\_HOME% 的输出。
2. echo %PATH% 的输出。
3. where java 的输出。