

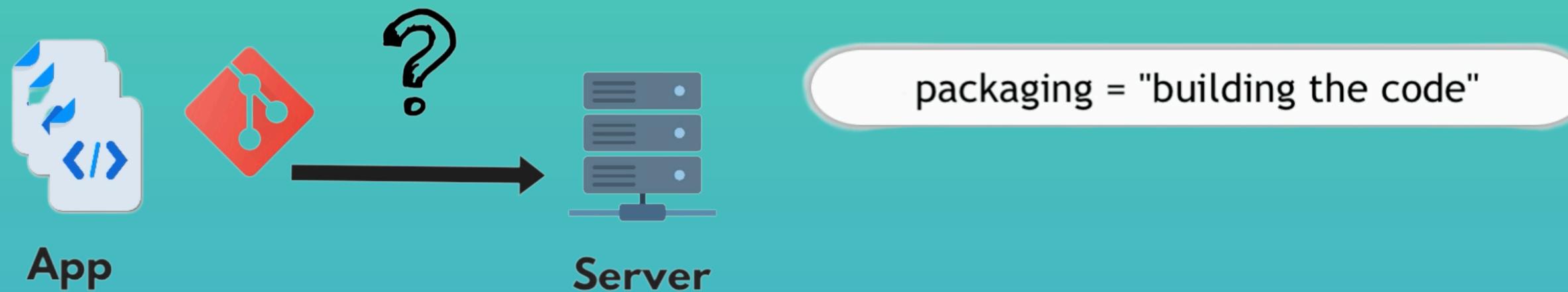


Build Tools & Package Manager

Key Takeaways

What are Build and Package Manager Tools? - 1

- Application needs to be deployed on a production server
- For that, we want to **package** application into a **single moveable file (artifact)**, also called "building the code"
- This is what a build tool or package manager tool does



What is an "artifact"?

- Includes application code and all its dependencies



artifact

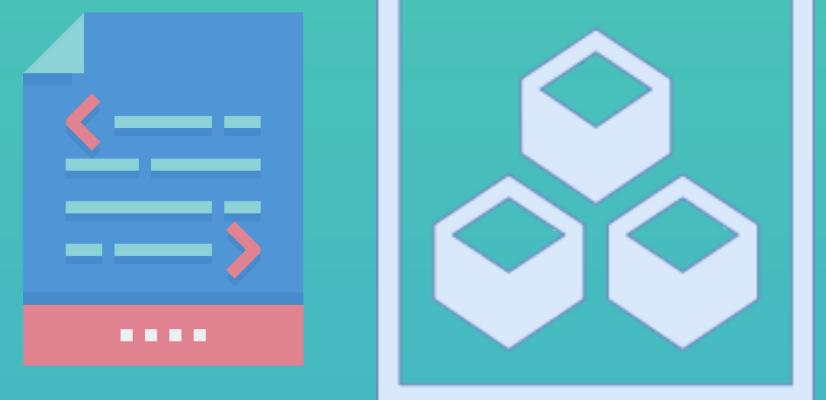
What are Build and Package Manager Tools? - 2

What does "building the code" mean?

- **Compiling** the code
- **Compressing** the code
- Package hundred of files **to 1 single file**

What is an "artifact repository"?

- Storage for artifacts
- To deploy artifacts multiple times, have backups etc.
- Examples: Nexus, Jfrog Artifactory



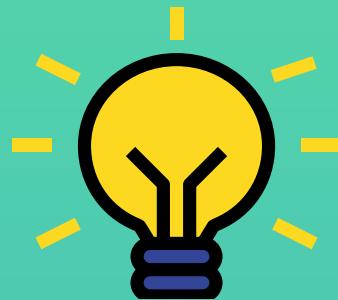
artifact repository

dev

test

production

Artifacts



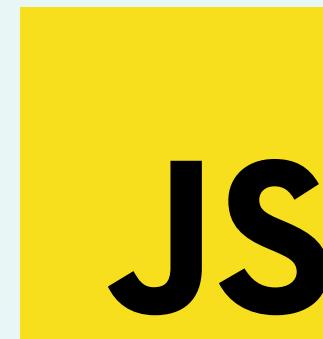
Artifact files look different for each programming language



- JAR = Java Archive
- Includes **whole code plus dependencies**, like Spring framework, datetime libraries, pdf processing libraries



JAR or WAR file



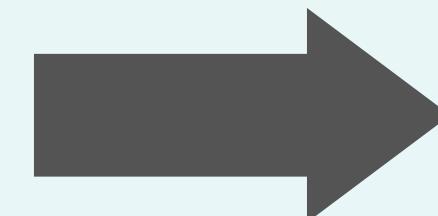
- JavaScript's artifact can be a zip or tar file



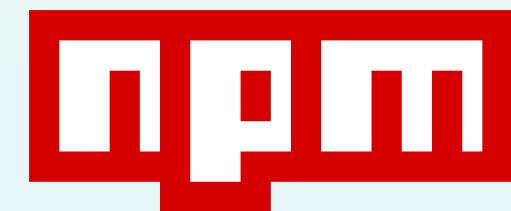
Zip or Tar file

Different Build Tools for different programming languages

Build tools we cover **in this module**:



- Java Build Tools: **Gradle and Maven**



- JavaScript Package Manager: **npm**

Installation and Setup of Projects - 1

- In this module you need to install following technologies and setup following projects:

Technologies

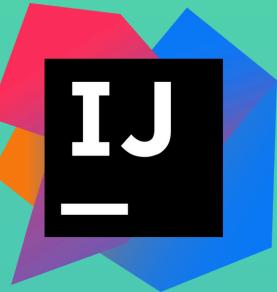


Projects

- **java gradle** application
- **java maven** application
- **nodejs** application

Installation and Setup of Projects - 2

- Download IntelliJ = **Code Editor**
- Install **OS Package Manager** (like Homebrew for Mac)
- Install Java, Maven, Node, npm



Clone Git Projects & Open in IntelliJ:

- Setup Java-**Maven** Project in IntelliJ
- Setup Java-**Gradle** Project in IntelliJ
- Setup Node.js-**npm** Project in IntelliJ

Build an Artifact



- With a Build Tool you can build the artifact
- Specific to the programming language
- Build Tools have **command line tool** and **commands to execute the tasks**

"Build" Process in Maven or Gradle:

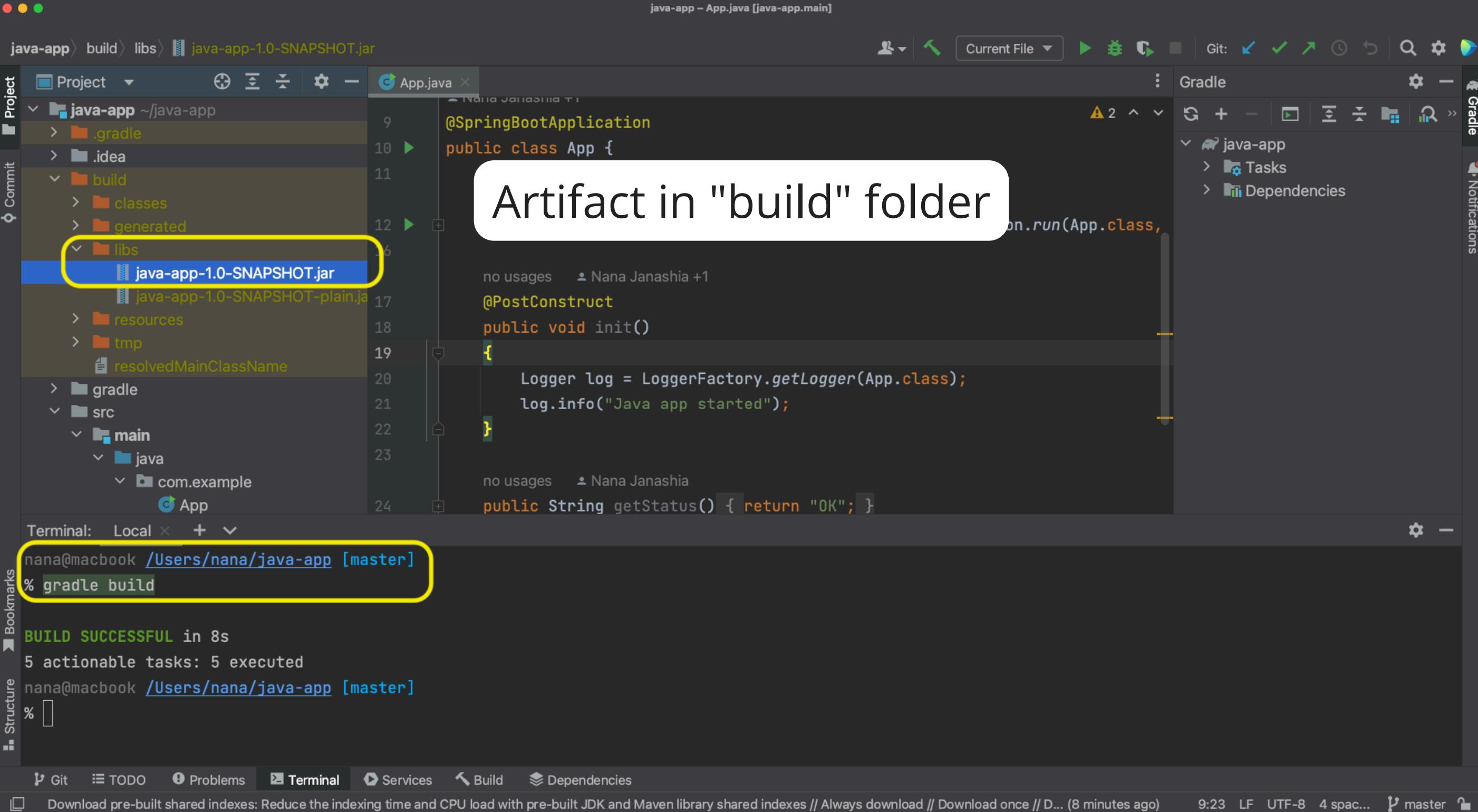
- Installs dependencies
- Compiles and compresses your code



- Configuration in Groovy
- Configuration in XML

- Example Build Tools for Java: Gradle and Maven
- Artifact: JAR or WAR file

Building artifact for java-gradle project



java-app - App.java [java-app.main]

Project Commit

Gradle Notifications

Artifacts in "build" folder

java-app/build/libs/java-app-1.0-SNAPSHOT.jar

App.java

```
@SpringBootApplication
public class App {
    public static void main(String[] args) {
        SpringApplication.run(App.class,
                args);
    }

    @PostConstruct
    public void init() {
        Logger log = LoggerFactory.getLogger(App.class);
        log.info("Java app started");
    }

    public String getStatus() { return "OK"; }
}
```

Terminal: Local

nana@macbook /Users/nana/java-app [master]

% gradle build

BUILD SUCCESSFUL in 8s
5 actionable tasks: 5 executed

nana@macbook /Users/nana/java-app [master]

%

Git TODO Problems Terminal Services Build Dependencies

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK and Maven library shared indexes // Always download // Download once // D... (8 minutes ago)

9:23 LF UTF-8 4 spac... master

Building artifact for java-maven project

The screenshot shows the IntelliJ IDEA interface with a Java-Maven project named "java-maven-app".

Left Panel (Project View): Shows the project structure with files like Application.java, pom.xml, Jenkinsfile, and script.groovy.

Middle Panel (Code Editor): Displays the pom.xml file content:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>3.0.5</version>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>

```

A yellow box highlights the "target" folder under the "java-maven-app" directory, which contains subfolders like classes, generated-sources, maven-archiver, and maven-status. A blue box highlights the generated artifact "java-maven-app-1.1.0-SNAPSHOT.jar".

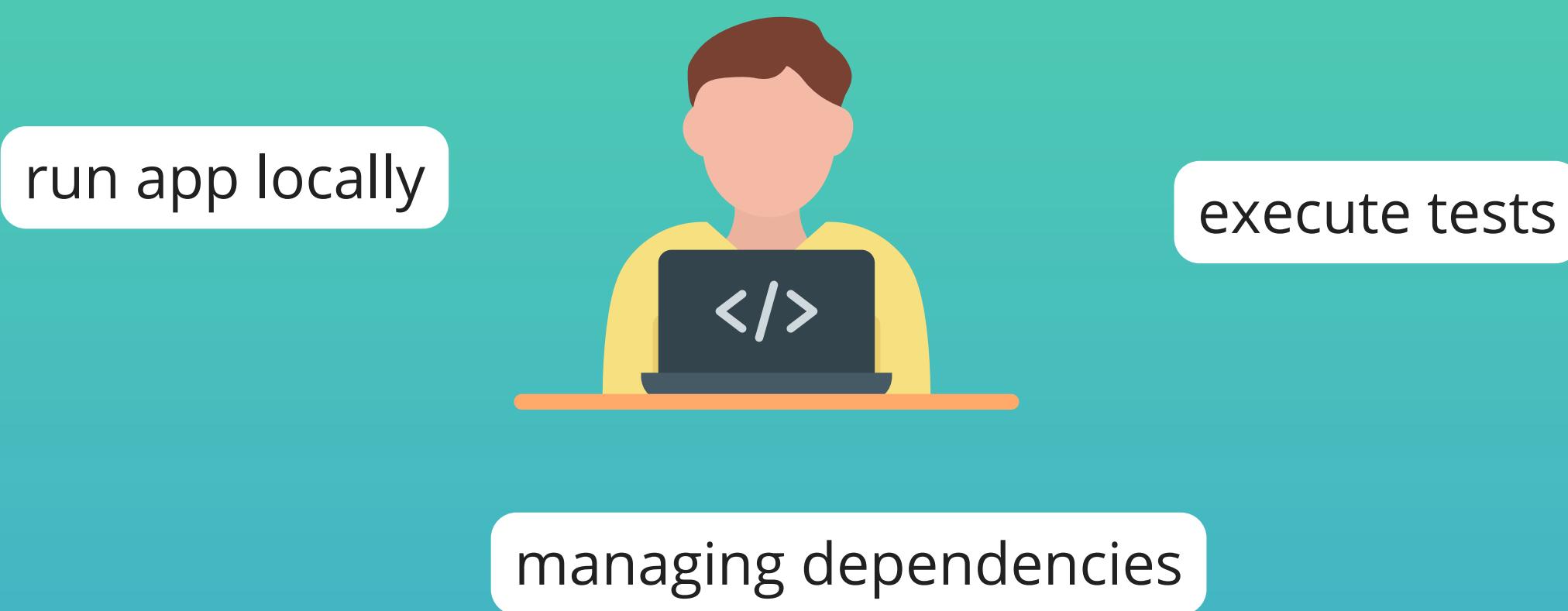
Bottom Panel (Terminal): Shows the command "mvn install" being run in the terminal, resulting in a successful build output:

```
nana@macbook /Users/nana/java-maven-app [master]
% mvn install
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.690 s
[INFO] Finished at: 2023-04-28T15:15:59+01:00
[INFO] ------------------------------------------------------------------------
nana@macbook /Users/nana/java-maven-app [master]
%
```

Callout Text: "Artifact in 'target' folder"

Build Tools for Software Development - 1

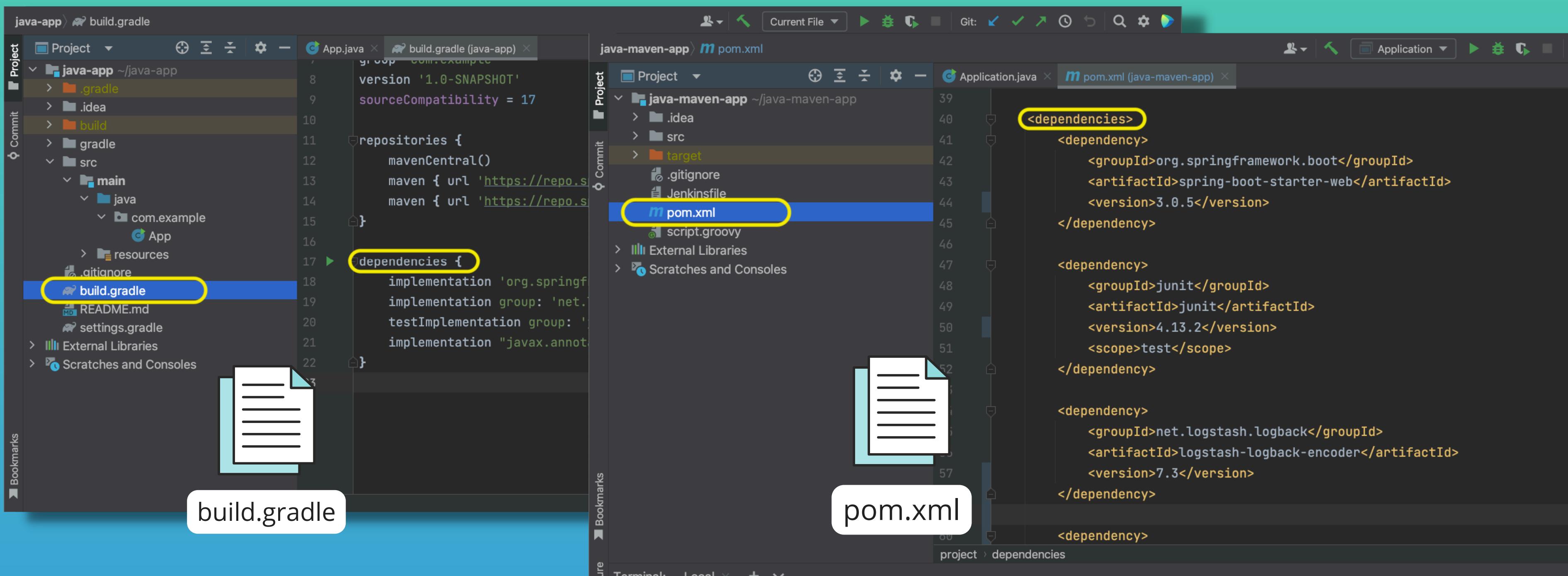
- Software developers need build tools **locally** when **developing the application**



- For example, a build tool is used to **manage the dependencies of a project**

Build Tools for Software Development - 2

- For that, build tools have a **dependencies file**:
- Whenever you need a new dependency, you can add it to the list



The image shows two side-by-side code editors comparing build files for Java projects.

Left Editor (Gradle Project):

- Project: `java-app`
- File: `build.gradle` (highlighted with a yellow box)
- Content:

```
version '1.0-SNAPSHOT'  
sourceCompatibility = 17  
  
repositories {  
    mavenCentral()  
    maven { url 'https://repo.spring.io/milestone' }  
    maven { url 'https://repo.spring.io/snapshot' }  
}  
  
dependencies {  
    implementation 'org.springframework:spring-web:5.3.10'  
    implementation group: 'net.logstash.logback', name: 'logstash-logback-encoder', version: '7.3'  
    testImplementation group: 'junit:junit', version: '4.13.2'  
    implementation "javax.annotation:javax.annotation-api:1.3.2"  
}
```

Right Editor (Maven Project):

- Project: `java-maven-app`
- File: `pom.xml` (highlighted with a yellow box)
- Content:

```
<dependencies>  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-web</artifactId>  
        <version>3.0.5</version>  
    </dependency>  
  
    <dependency>  
        <groupId>junit</groupId>  
        <artifactId>junit</artifactId>  
        <version>4.13.2</version>  
        <scope>test</scope>  
    </dependency>  
  
    <dependency>  
        <groupId>net.logstash.logback</groupId>  
        <artifactId>logstash-logback-encoder</artifactId>  
        <version>7.3</version>  
    </dependency>  
  
    <dependency>
```

Both editors include icons representing the build files: a document icon for `build.gradle` and a document icon for `pom.xml`.

Build Tools for Software Development - 3

- You can find all Java Libraries in Maven Central Repository:

The screenshot shows the Maven Repository website at mvnrepository.com/artifact/junit/junit/4.13.2. The page displays detailed information about the JUnit 4.13.2 artifact, including its license (EPL 1.0), categories (Testing Frameworks & Tools), and tags (testing, junit). It also shows its organization (JUnit), homepage (<http://junit.org>), and last update date (Feb 13, 2021). The page highlights that JUnit is the #1 ranked artifact in the Testing Frameworks & Tools category and is used by 122,433 artifacts. Below this, there is a code snippet of a Maven dependency declaration for JUnit. The left sidebar lists various popular categories such as Testing Frameworks & Tools, Android Packages, and JSON Libraries. The right sidebar lists indexed repositories like Central, Atlassian, and Sonatype.

JUnit » 4.13.2

JUnit is a unit testing framework to write and run repeatable automated tests on Java.

License	EPL 1.0
Categories	Testing Frameworks & Tools
Tags	testing junit
Organization	JUnit
HomePage	http://junit.org
Date	Feb 13, 2021
Files	jar (375 KB) View All
Repositories	Central HeavenArk GroovyLibs Minebench Lutece Paris Xceptance
Ranking	#1 in MvnRepository (See Top Artifacts) #1 in Testing Frameworks & Tools
Used By	122,433 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Gape Leiningen Buildr

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

Include comment with link to declaration

Compile Dependencies (1)

Indexed Artifacts (33.5M)

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- JVM Languages
- Core Utilities
- Mocking
- Language Runtime
- Web Assets
- Annotation Libraries
- Logging Bridges
- HTTP Clients
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities

Indexed Repositories (1914)

- Central
- Atlassian
- Sonatype
- Hortonworks
- Spring Plugins
- Spring Lib M
- JCenter
- JBossEA
- Atlassian Public
- KtorEAP

Popular Tags

- aar
- amazon
- android
- apache
- api
- application
- arm
- assets
- atlassian
- aws

Build Tools for Software Development - 4

Need a library for ElasticSearch database connection?

1. Find a dependency with name and version
2. You add it to dependencies file (e.g. pom.xml)
3. Dependency gets downloaded locally (e.g. local maven repo)



Run a Java Application

Locally, for example to test:

- Locate Jar file and execute:

`java -jar <name of jar file>`

On a **deployment server**:

- Copy the jar file to server, where application should run and execute "java -jar" command



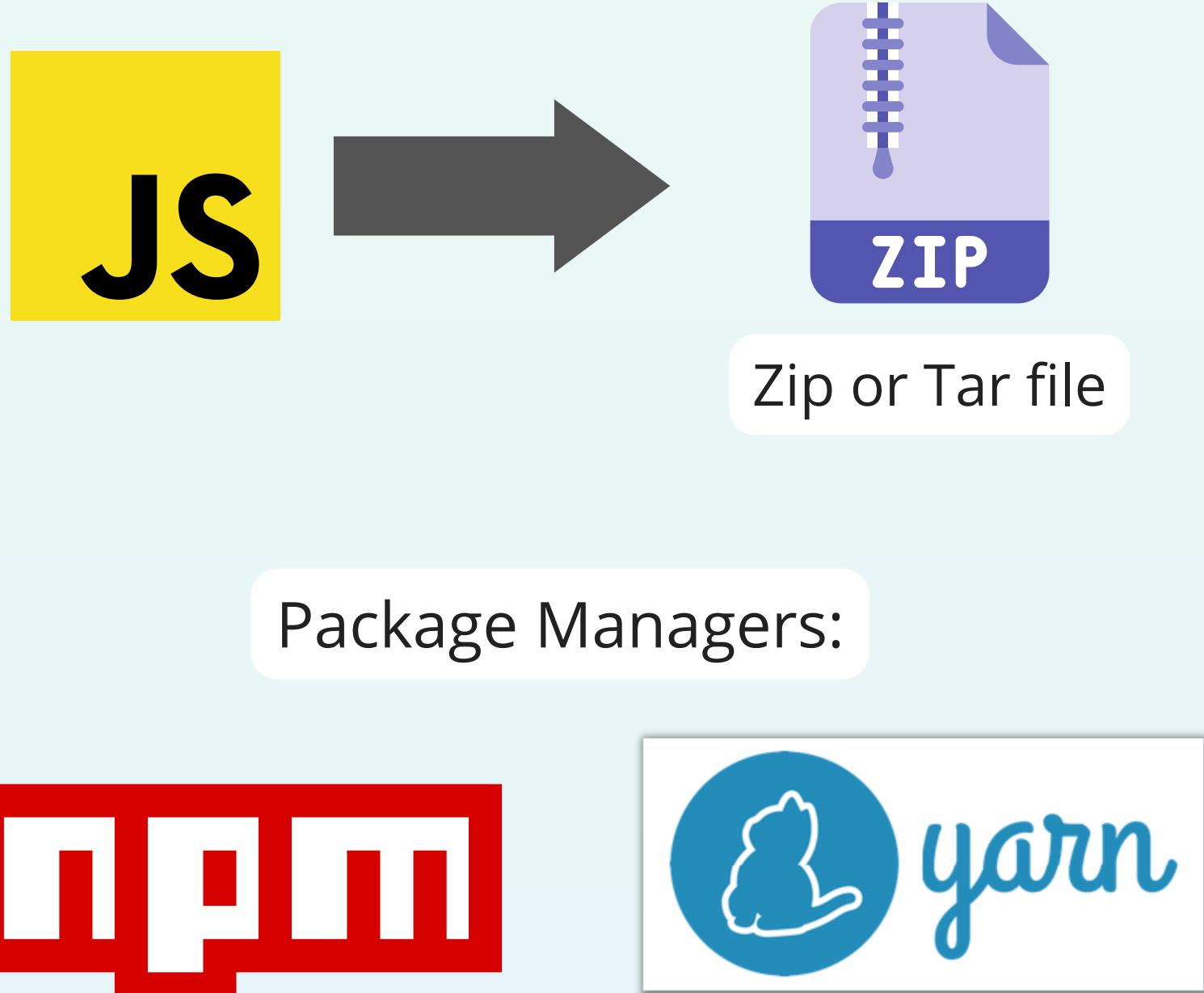
A screenshot of an IDE (IntelliJ IDEA) showing a Java Maven project named "java-maven-app". The project structure includes a "target" folder containing "classes", "generated-sources", "maven-archiver", and "maven-status" subfolders. Inside "target" is a "java-maven-app-1.1.0-SNAPSHOT.jar" file, which is highlighted with a yellow selection bar. Below the project tree is a terminal window showing the command: "% java -jar target/java-maven-app-1.1.0-SNAPSHOT.jar". The terminal output shows the command was run successfully.

```
Project: java-maven-app ~/java-maven-app
  .idea
  src
    target
      classes
      generated-sources
      maven-archiver
      maven-status
      java-maven-app-1.1.0-SNAPSHOT.jar
      java-maven-app-1.1.0-SNAPSHOT.jar.origina
    .gitignore
    Jenkinsfile
  pom.xml

Terminal: Local + 
nana@macbook /Users/nana/java-maven-app [master]
% java -jar target/java-maven-app-1.1.0-SNAPSHOT.jar
```

Build JavaScript applications - 1

- Artifact of a JavaScript application is e.g. a **zip or tar file**
- But no special artifact type defined
- In JS we have **package managers** and NOT build tools
- For JS we have **npm** and **yarn**
- **package.json** file for dependencies
- Package manager install dependencies, but not used for transpiling JS code



Build JavaScript applications - 2

- JavaScript libraries can be found in **npm repository**

This screenshot shows the npm search interface. The search bar at the top contains the query 'mongodb'. Below the search bar, it says '8223 packages found'. On the left, there is a sidebar titled 'Sort Packages' with options: 'Optimal' (selected), 'Popularity', 'Quality', and 'Maintenance'. The main area displays three package results:

- mongodb** (exact match)
The official MongoDB driver for Node.js.
Version: 5.3.0, Published: 10 days ago by w-a-james.
Tags: mongodb, driver, official.
- mongoose**
Mongoose MongoDB ODM.
Version: 7.1.0, Published: a day ago by vkarpov15.
Tags: mongodb, document, model, schema, database, odm, data, data.
- bson-objectid**
Construct ObjectIDs without the mongodb driver or bson module.
Tags: ObjectId, mongo, mongodb, bson, createFromHexString, hex.

This screenshot shows the npmjs.com package page for 'mongodb'. The URL in the browser is 'npmjs.com/package/mongodb'. The page title is 'mongodb' with a 'TS' badge. It shows the version 5.3.0 is public and was published 10 days ago. The 'Readme' tab is selected. The page content includes:

- MongoDB Node.js Driver**: A brief description stating it's the official MongoDB driver for Node.js.
- Upgrading to version 5?**: A link to the upgrade guide.
- Quick Links**: A table with links to Documentation, API Docs, npm package, and MongoDB.
- Install**: A command to install the package via npm: 'npm i mongodb'.
- Repository**: A link to the GitHub repository: 'github.com/mongodb/node-mongodb-n...'.
- Homepage**: A link to the MongoDB homepage: 'github.com/mongodb/node-mongodb-...'.
- Weekly Downloads**: A chart showing 4,391,228 weekly downloads.
- Version**: The current version is 5.3.0.
- License**: The license is Apache-2.0.

The footer features the 'TECHWORLD WITH NANA' logo.

Site	Link
Documentation	www.mongodb.com/docs/drivers/node
API Docs	mongodb.github.io/node-mongodb-native
npm package	www.npmjs.com/package/mongodb
MongoDB	www.mongodb.com

Build JavaScript applications - 3

Command Line Tool - npm



- **npm install:** installs the dependencies
- **npm start:** start the application
- **npm stop:** stop the application
- **npm test:** run the tests
- **npm publish:** publish the artifact

What does the zip/tar file include?

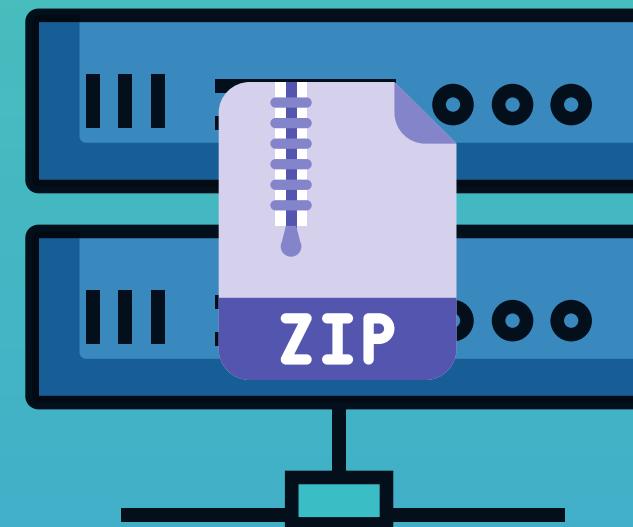
- application code, but **NOT the dependencies**

Build JavaScript applications - 4

Run JavaScript application on server:

1. Copy Zip/Tar file to server
2. Unpack zip/tar
- 3. Install dependencies**
4. Run the application

- Zip/Tar file includes the application code, but **NOT the dependencies**



Flexible JavaScript - 1



JavaScript world is **more flexible** and less
standardized compared to Java

If application consists of different programming languages:

Frontend: React.js (JS Library)

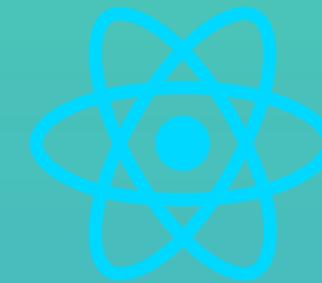
Backend: Java, Node.js, Python etc.

- You can package frontend and backend code separately
- Or in a common artifact file

Flexible JavaScript - 2

If application consists of JavaScript in Frontend and Backend:

Frontend: React.js (JS Library) **Backend:** Node.js (JS Library)



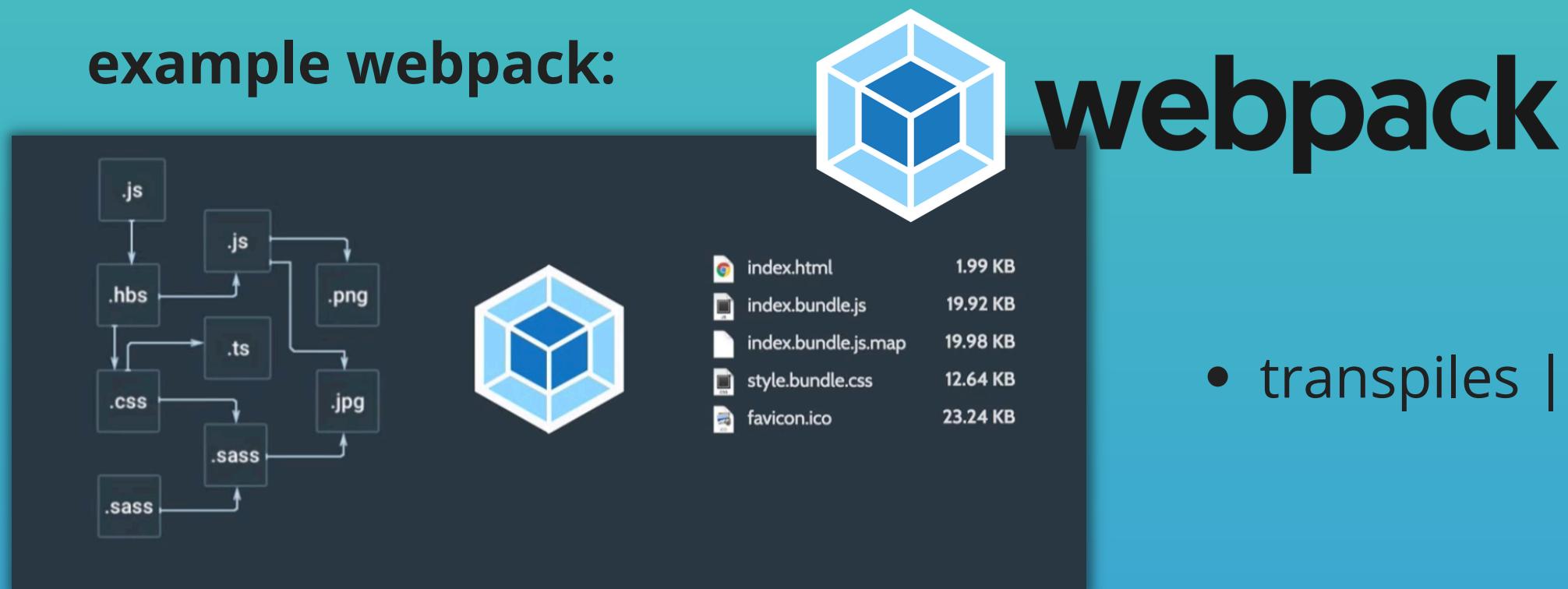
- You can have a **separate package.json file** for frontend and backend
- Or have a **common package.json file**

Package Frontend Code - 1

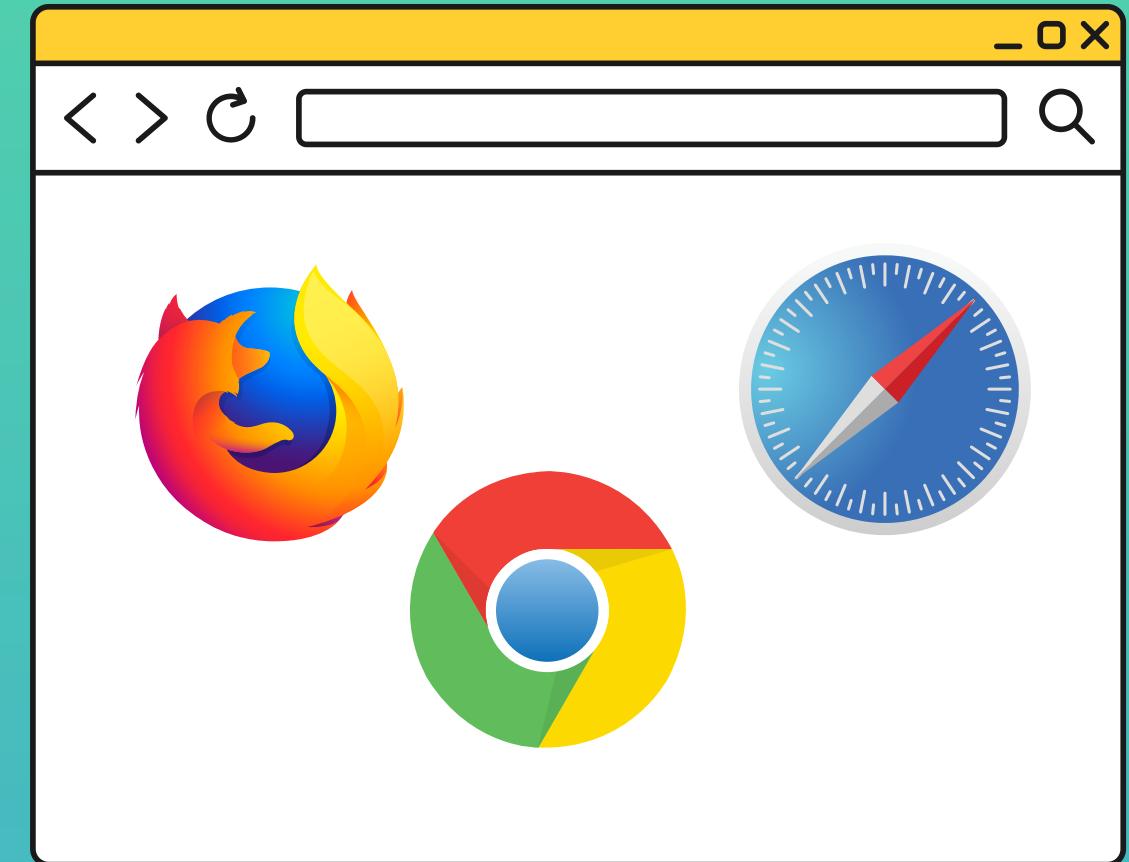


Frontend Code runs in the browser.
Browser doesn't support latest JS versions or other fancy code decorations, like JSX in React.js

- That's why frontend/React Code needs to be **transpiled** into browser compatible code
- Code needs to be **compressed/minified**
- There are separate tools for that: **Build Tools/Bundler**, for example **webpack**:



- transpiles | minifies | bundles | compresses the code



Package Frontend Code - 2

-  **Bundle** frontend code **with webpack**
-  **Manage dependencies** with **npm or yarn**
-  **Package** everything into a **WAR file**

Build Tools for other programming languages

- **Java:** maven | gradle
- **JavaScript:** npm | yarn | webpack
- **Python:** pip
- **C/C++:** conan
- **C#:** NuGet
- **Golang:** dep
- **Ruby:** RubyGems



But, concepts very similar to other
programming languages

Pattern in all these tools:

1. dependency file

package.json

pom.xml

build.gradle

2. repository for dependencies

3. command line tool, to:

test

start the app

build app

publish app

4. package managers

gradle

npm

maven

yarn

pip

dep

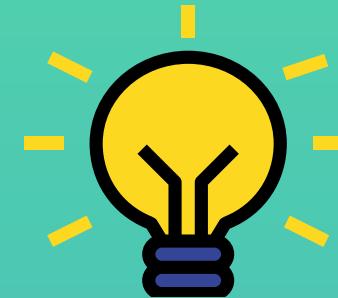
Publish an artifact

1. Build jar or zip package

2. **Push to artifact repository** to save those packages for later use or for downloading on a remote server

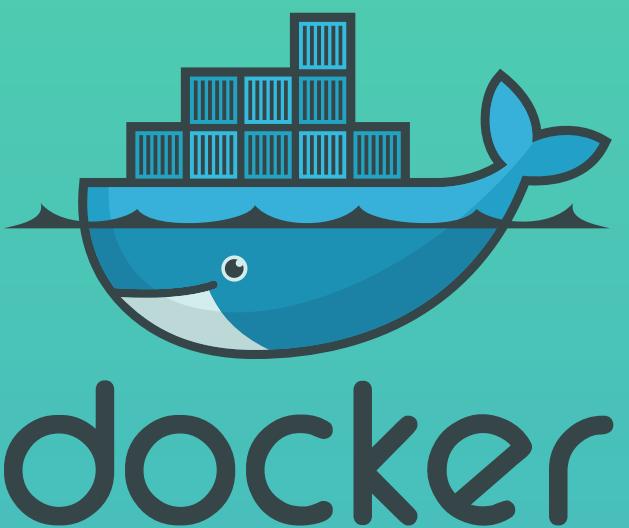


3. Then you can download (curl, wget) it anywhere



Change in how we use artifacts

- We don't keep jar or zip files, because we have Docker
- We don't build them locally, because we have Jenkins and other Build Automation Tools



Jenkins

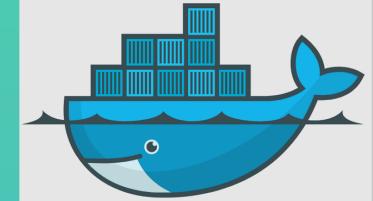
Build Tools and Docker - 1



No need to build and move different artifact types (e.g. Jar, War, Zip)

Just 1 artifact type: Docker Image

Image



- We build those Docker images from the applications



No need for a repository for each file type

Just 1 Docker Image



No need to install dependencies on the server!

Execute install command inside Docker Image

Build Tools and Docker - 2



Docker makes it easier

Docker Image is an artifact



Docker Image is an alternative for all other artifact types

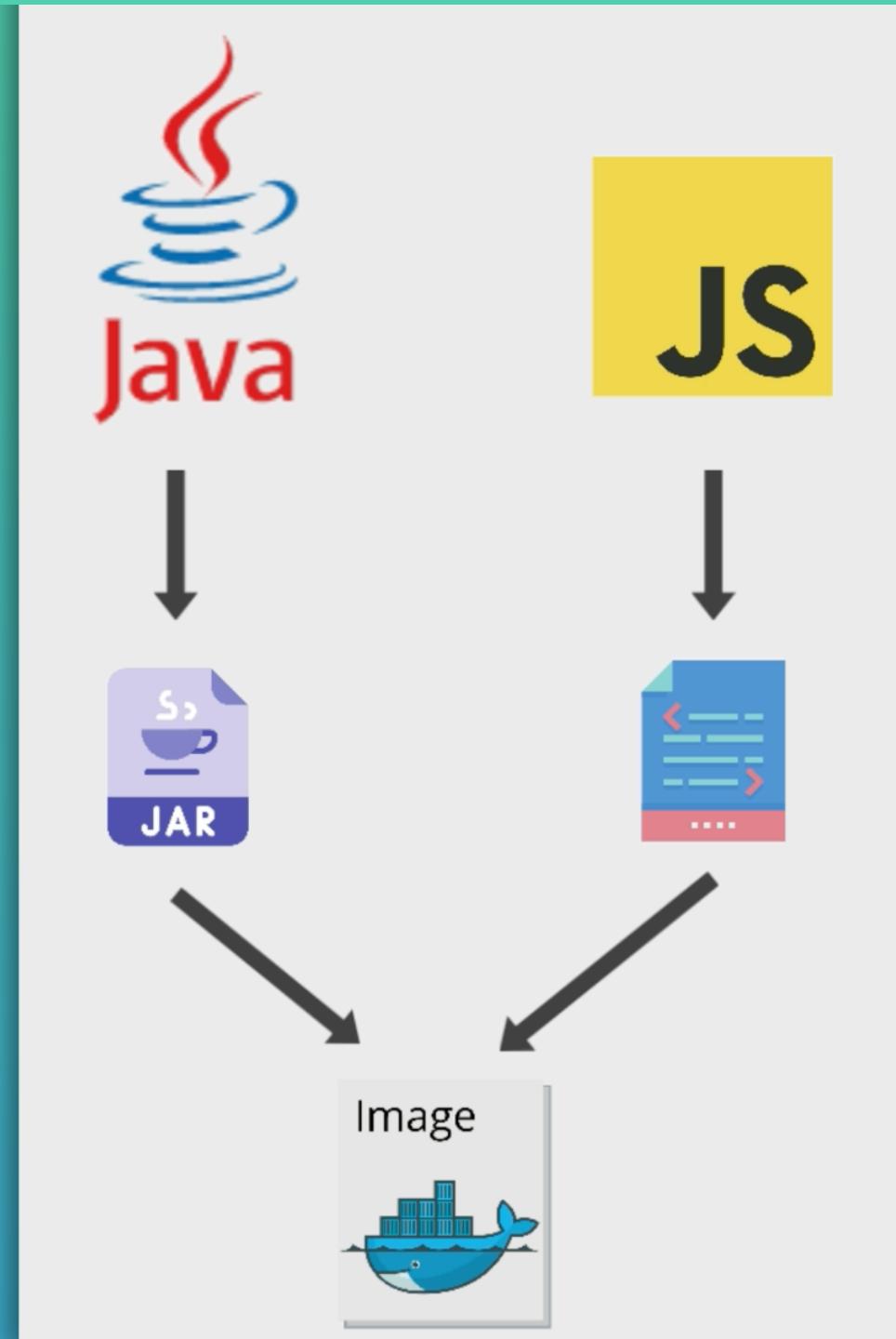


You don't need to install npm or java on the server

Execute everything in the Image



Build Tools and Docker - 3

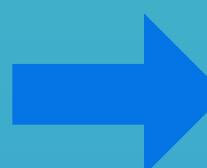


A screenshot of a code editor showing a Dockerfile for a Java application named "java-app". The Dockerfile contains the following code:

```
FROM amazoncorretto:17-alpine-jdk
EXPOSE 8080
COPY ./build/libs/java-app-1.0-SNAPSHOT.jar /usr/app
WORKDIR /usr/app
ENTRYPOINT ["java", "-jar", "java-app-1.0-SNAPSHOT.jar"]
```

The project structure on the left includes .gradle, .idea, build (containing classes, generated, libs, resources, tmp, resolvedMainClassName), src, .gitignore, build.gradle, Dockerfile (which is selected), README.md, and settings.gradle.

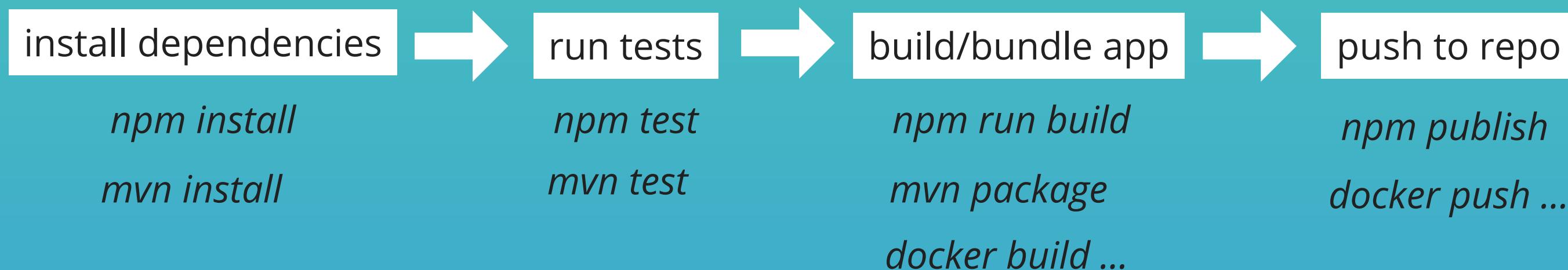
- You still need to build the application!



More about Docker in a later Module!

Why should you know these Build Tools as a DevOps engineer?

- **Help developers** building the application, because you know where and how it will run on deployment servers
- You need to **configure the build automation tool** or CI/CD Pipeline, like execute tests on the build servers, build and package into Docker Image, run the application on server



So, you don't execute anything locally