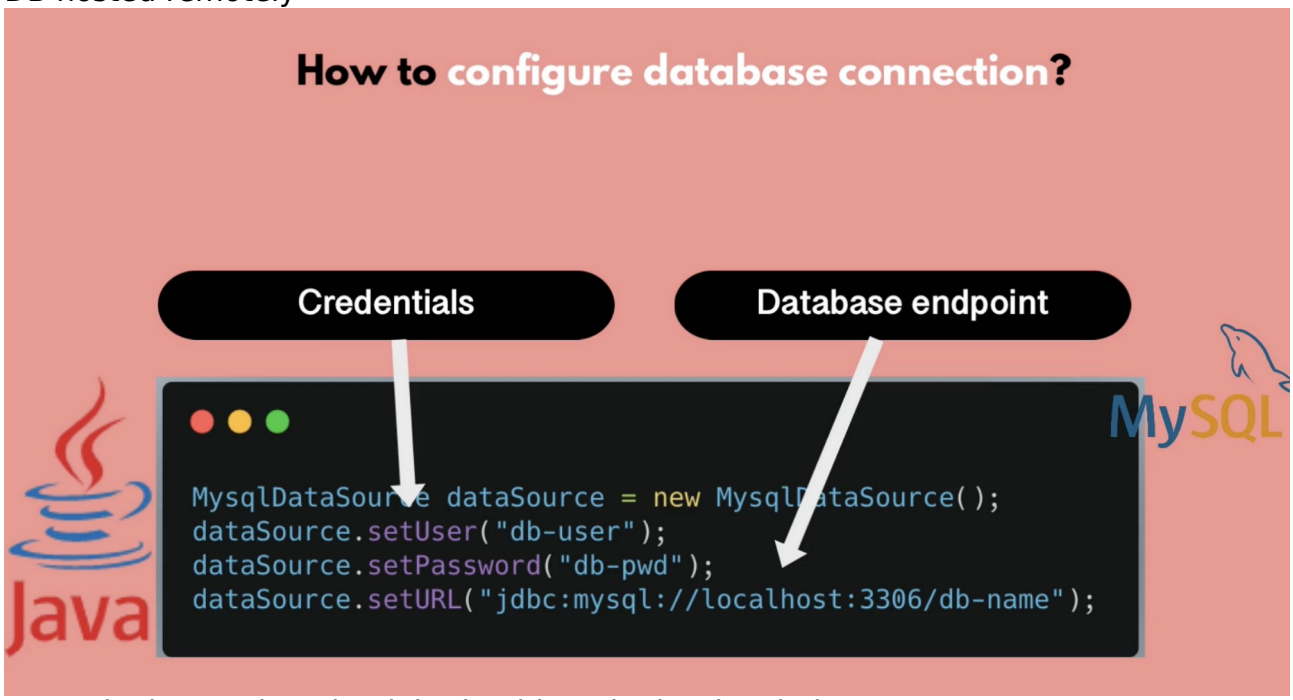
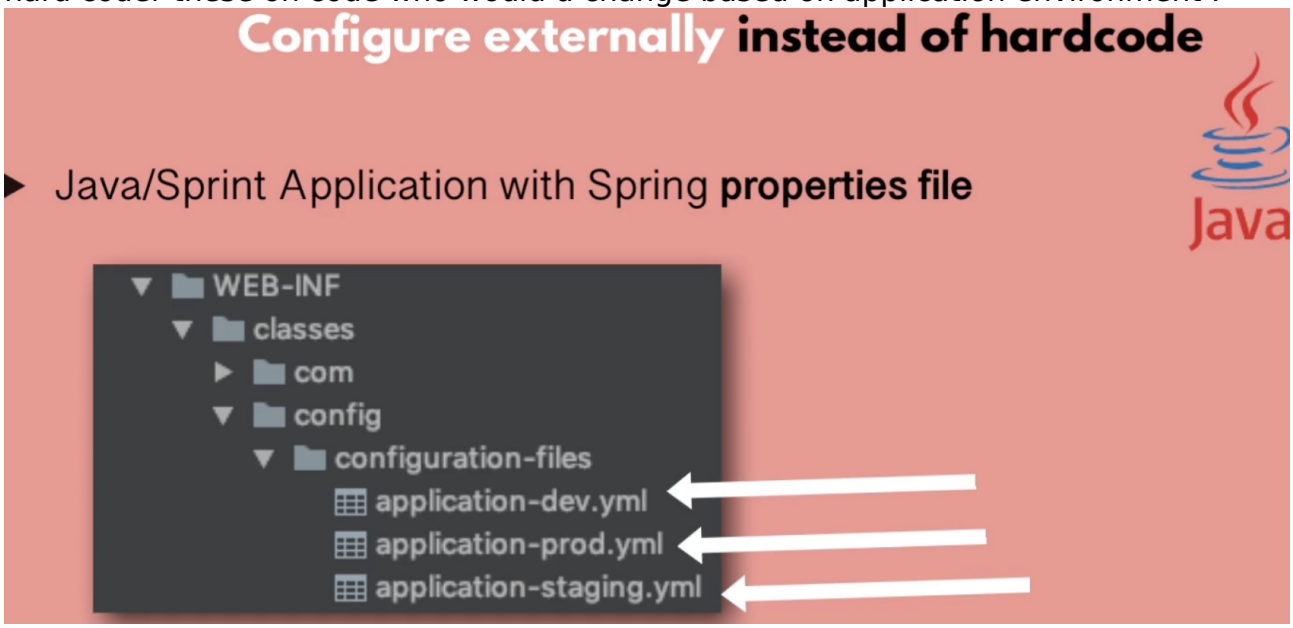
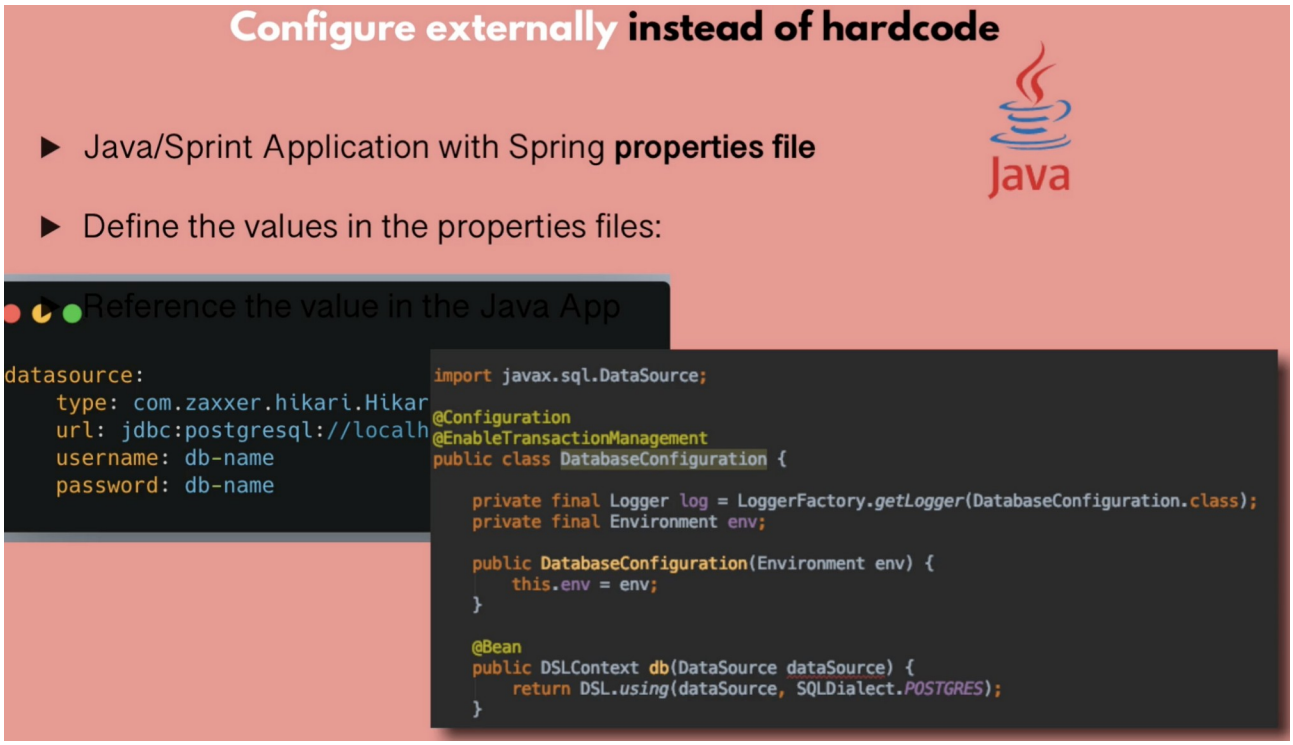
Developers need data base for local development :
Option 1 :
Each developer install mysql dB locally
Option 2 :
DB hosted remotely



DB endpoints and credentials should not be hard coded .
As should be defined at one place as environment variable.
Which we can configure from outside
Depending on which db u are running u can switch btw then eg: dev , prod , test db .
As your application in running on different servers.
Each has different endpoints and credentials (production will be more secure) so it u hard coder these on code who would u change based on application environment .



Before app is deployed , we need to install and configure db .
On the same or different DB server.



DATABASES IN PRODUCTION :
Replicate , regular backups ,performs under high load .
Developers them selfs do not usually manage db.

AS dev ops engineer :
How to config ,setup , manage DB , replicate , backup and restore .

DATABASES TYPES :

Key value DB : etcd from Kubernetes , redis , they are very fast (as in memory no hard drive (less space) , no joins , not primary DB for application .
Best for : cacheing , eg ;twitter , message queue,

Wide col dB : same key but value is divided into different col , schema less . Eg: cassandra , Hbase .
It is very scalable, queries similar to sql but much simpler no joints , limited in compare to relation DB .
Best for : time series unstructured data , record from iot devices. Sensors , historical records. Etc.

Document DB : mongoDB , dynamo db ,etc. this is also unstructured data there for u don't need schema before to add the data (schema less) , slower writes updated , no joins ,faster to read the data .
Best for : mobile apps , game app , most apps , content management .etc .

Relational db (most used): mysql , postgresql , : structured db , need a schema data type need to be defined first , need query language to read or write on db (SQL) , followed ACID : atomicity , consistency , isolation , durable. Ensured no half changes are made .
These are difficult to scale : to solve this issue cockroach DB comes .
Best for : banking ,

Graph db : YouTube , reddit .eg : neo4j , dgraph .best for : recommendation engine , patterns rec.
Search DB: search data from massive data entries . Similar to documented dB . Index is searched only . It create index for words .
Eg : elastic search , solar DB .

WE CAN USED THESE DB IN COMBINATION :
Eg: relation bd (most data) , search engine(to handle fast search) , key value-db (redis) (cache).

 # WHAT ARE BUILD AND PACKAGE MANAGER TOOLS :

App need to be deployed in production server .
Application code and all its dependencies need to be in servers so that users can use it .
U package your application in **single movable file** and then get that file in the server and run it there . That single movable file is called application Artifact . Packaging the application means “building the code”.
Building the code mean : compile , compress (hundred to one file), keep artifact on storage,
To deploy it multiple times and have backups etc .eg : dev , production , test servers .

Place where we store these artifact : jfrog artifactory , nexus .

What kind of file are these artifact ?

Artifact files looks different for different applications.
Eg : for java app : jar (java archive) or WAR file .
This jar file will contain whole code plus the dependencies like(spring framework etc.)

Install java and build tools :
1 . Java maven application :
IntelliJ will autodetect its a maven project and download all dependence
When u open a project on IntelliJ after cloning from GitHub.
Install maven . After that go to the maven project and run -=> m v n package command
This mv n package command will build your application .
2. Java gradle application :
IntelliJ will detect it is a gradle project and starts in starts installing gradle .
./gradle build to build the application .
3. React nodes:
For this we need to install nodes and n pm installed locally .
Cd api -> n pm start .

How to build the artifacts ?
These tools are specific to the programming language .
For Java there is maven and g radle .
How do maven and gradle tools build the artifact -> install dependency , compile and compress the code .

Different btw maven and cradle build tools ?
Maven uses xml language
Gradle uses groovy language .(more convinient for writing scripts etc.)
Both have command line tools and commands to execute the tasks .
.gradle build : nothing is installed in your machine globally just every thing is bundles together .

Build folder is generated after gradle build which will contain .jar file .

Gradle 6 does not support the java 16 version .
Take java version less then 16 .

MANAGING DEPENDENCIES :
You need to build tools locally when deloping the app.
In maven add dependencies in pom.xml file . Or build.gradle in gradle
First the dependencies will be downloaded before building the app.
.m2 / repository here all the dependencies are downloaded locally .

HOW DO YOU RUN THESE JAR FILE IN SERVER :
Java -jar build/libs/java-app-1.0-SNAPSHOT.jar (java gradle)
java-maven-app % java -jar target/java-maven-app-1.1.0-SNAPSHOT.jar (java maven)

So u can download artifact and start the application using these commands .

Building java application :
No special artifact type : z i p or tar. File .
Alternatives of maven and grad le for js is : n p m and yarn .
Package.json file used for managing dependencies.
N pm and yarn are package managers and not the build tools like maven and grad le .
It install dependencies but not used for transpiling js code .
package.json is same as pom.xml for maven and build.gradle for gradle where all the dependencies to be downloaded listed .

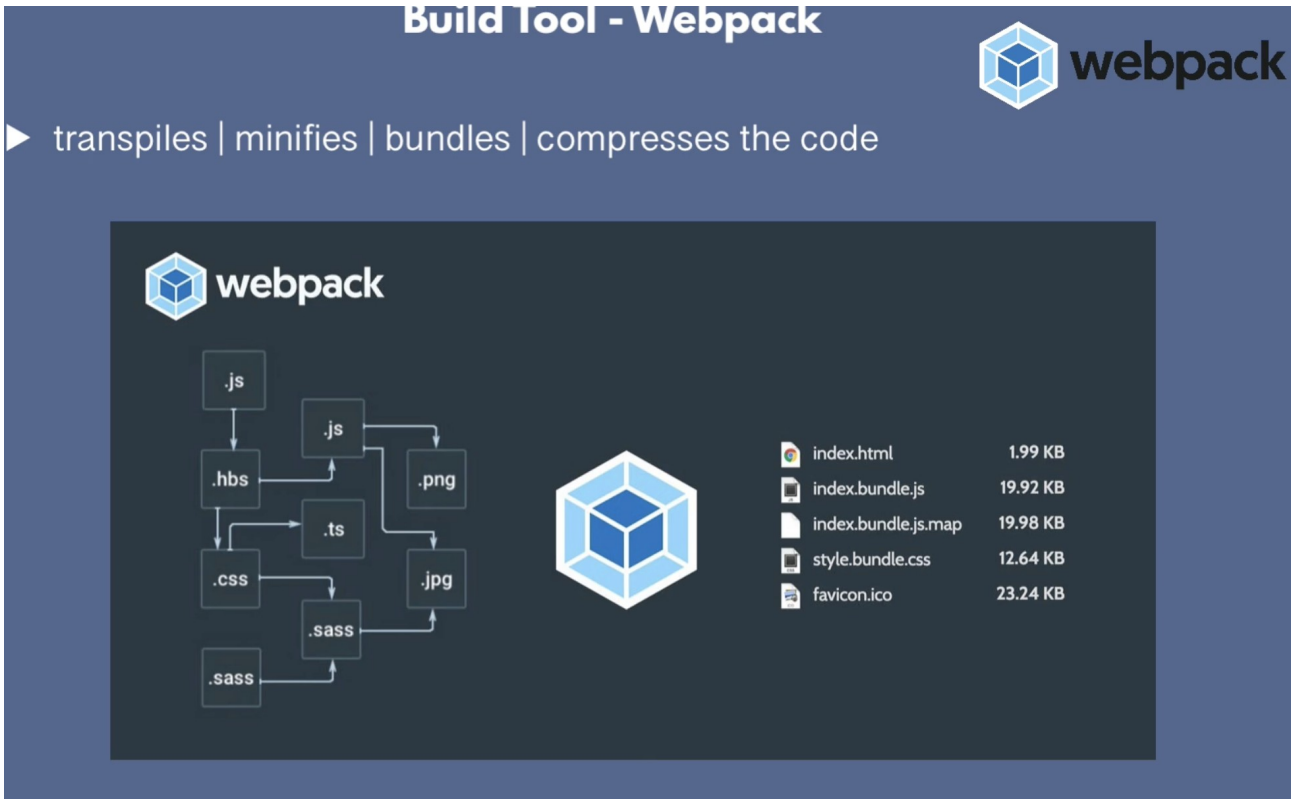
N pm repository has all the dependencies listed from where we can download them .
N pm start , n pm stop , np m test , n pm publish .

What does the zip or tar file include :?
Application code but not the dependencies .
There fore if u want to run app on server u must install dependencies on the server .
Then unpack zip or tar file and run the application .

N pm pack : to covert to tar file .

Packaging frontend code :
If frontend (react) and backend in (java , js, python) etc
Package frontend and backend code separately or can have common artifact for both .

Eg: js on both front end and backend , we can have separate package.json file for each js .
Or common package.json file where we manage all our dependencies.



Front end react code need to be transplied because browser will not understand it with that.
Because browser does not support latest js version or other code decorations like jsx .
So we need to transpiled to standard js syntax.
Also code need to be compressed because it should be too large for browser to download .
We have separate build tools for that : web pack

First go to that front end api folder where there are package.json file for front end then run command n pm install to install all dependencies . Then run —>

N pm run build : will call build : web pack : server.bundle.js will be created .
This is compress and transpires the code .

Eg: front end react js , backend java ,
Build frontend using web pack managing dependencies using n pm and yarn .
Package every thing in war file .

Common things in all the build tools :
Dependency file : package.json , pom.xml , build.gradle .
Repository for downloading the dependency .
Command line tools : for test , run , build or packaging the application .
Package managers : gradle , maven , yarn , pip , n pm .

When u build a artifact u need to publish that on the artifactory(eg: j frog) :
Build tools have command for that .

Then u can download (curl, wget) it any where.
Fetching it on the server .

BUILD TOOLS AND DOCKER :
NO need to build different artifact types (jar, war ,zip, tar).
We use only one artifact ie : docker image .
There fore no need for repository of different artifact types . We need just one for docker images.
No need for installing dependencies directly on the servers , u can install it directly on docker image.
Docker make it easy .
Docker image is also a artifact act as a alternative of all other artifact type.
No need to install java or n pm on the servers as do need to run commands like n pm run etc.
Execute everything on the image .
U still need to build the application into jar or web pack then convert into docker image .

BUILD TOOLS for DEVOPS :
Help developers on building the application .
Because u will know where and how it will run .
Developers install the dependencies locally they don’t build the application locally .
Build automation tools do this : build docker image -> push to repo -> run on servers .
Devops engineer : has to configure build automation tool / ci / cd pipeline . Which include installing dependencies , run test , build / bundle application code , push in repo and deploying on the server .
Build and package into docker image .
You need to execute test on build servers.