Java + Spring

Pre req's + steps

Chris Buckett
Sept 2022

valcon

Section 1

Download and install pre-requisites

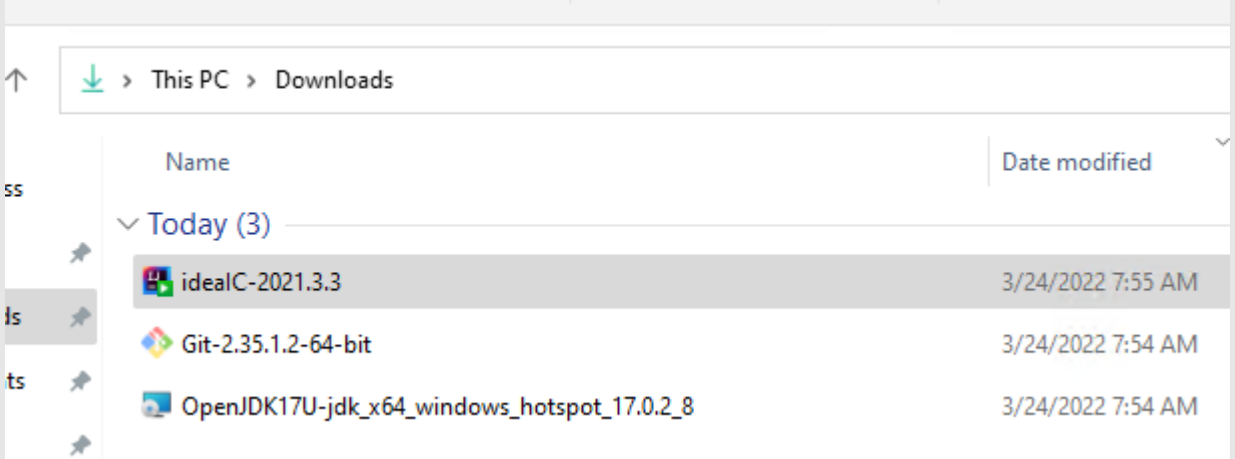
valcon

Pre-requisites

Java Development Kit (JDK): https://adoptium.net/

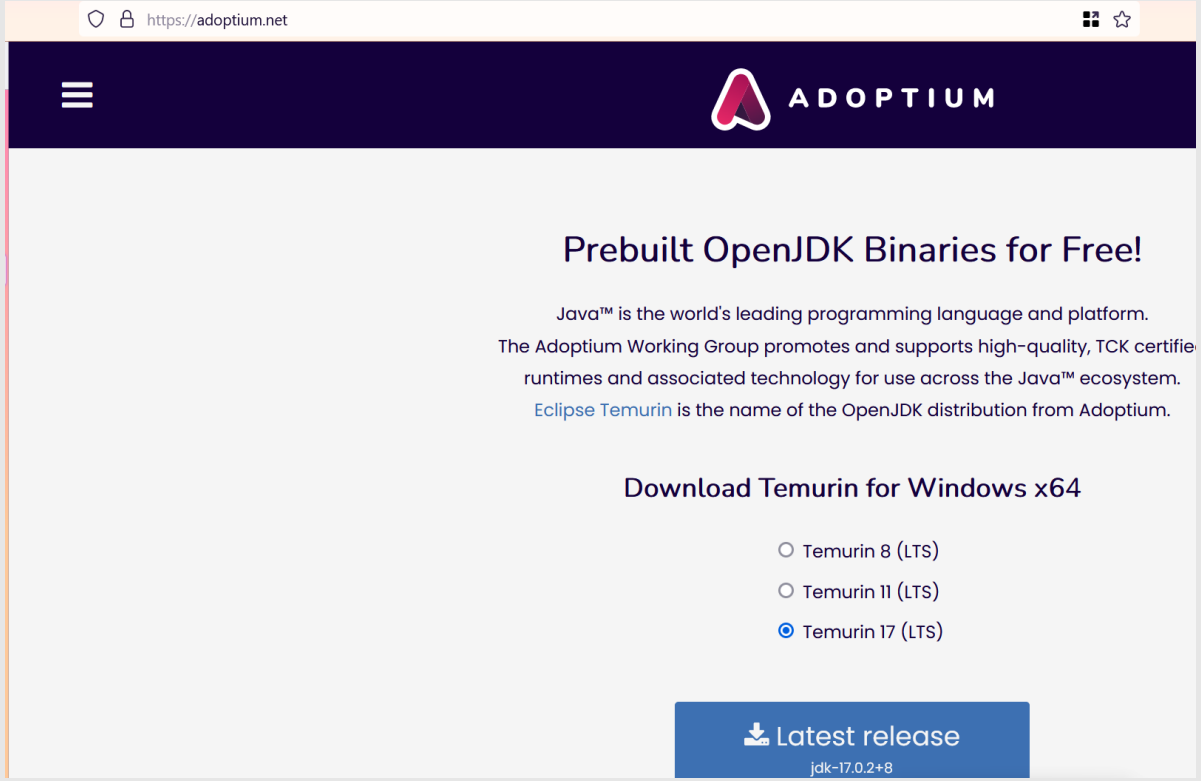
Git (64 bit): https://git-scm.com/download/win

IntelliJ community edition https://www.jetbrains.com/idea/download/#section=windows



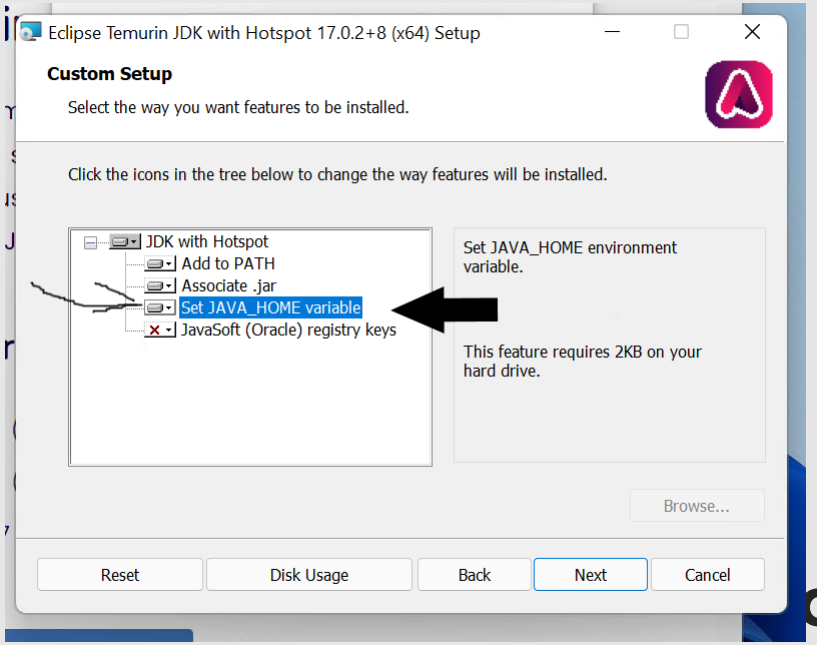
valcon

Download and install JDK17



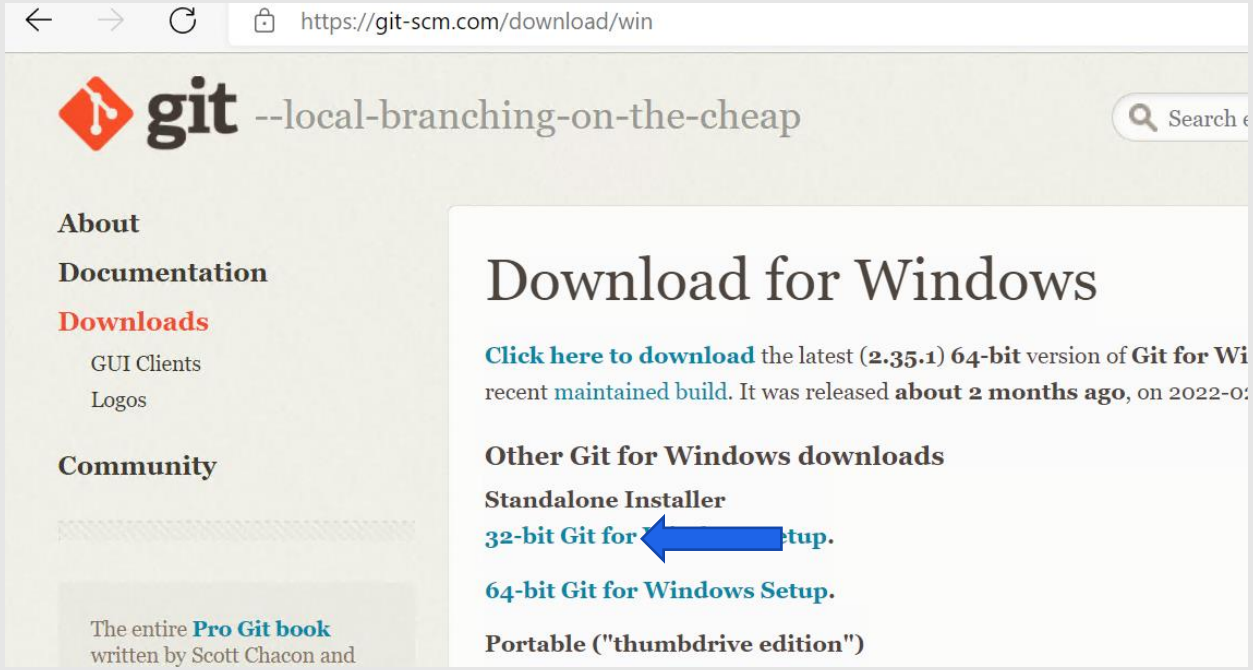
https://adoptium.net

- Download and install JDK 17
- Make sure installer sets JAVA_HOME



Download and install GIT command line tools

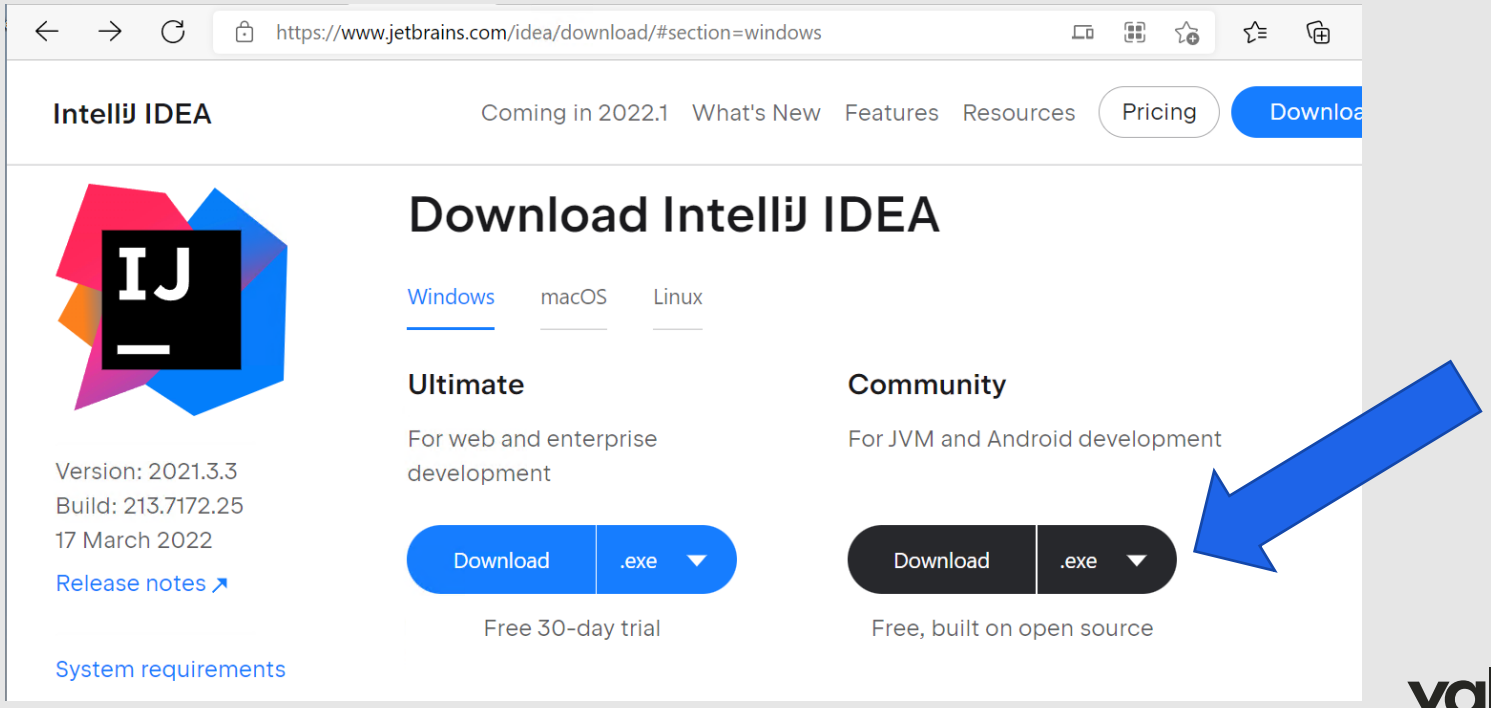
- https://github.com/git-for-windows/git/releases/download/v2.35.1.windows.2/Git-2.35.1.2-64-bit.exe
- Accept all defaults on install
  - (except the one about the default editor. You may want something other than VI – eg Notepad)



valcon

Download and install intellij community edition

- https://www.jetbrains.com/idea/download/#section=windows
- Accept defaults on install



valcon

Section 2

Checkout & run the starting point project

valcon

Fork the project

- When logged into your own github account, browse to: https://github.com/chrisbu/da2
- Click Fork, and make your own fork of the project, eg https://github.com/<yourusername>/da2

valcon

Check out starting point project

Open terminal (start > run > cmd [enter])
Checkout starting point project:

- `cd \`
- `mkdir dev`
- `cd dev`
- `git clone https://github.com/<your username>/da2`

Switch to step 1 tag

- `cd da2`
- `git checkout tags/Step1`
- `git branch Step1-YOUR_INITIALS`
- `git checkout Step1-YOUR_INITIALS`

Administrator: Command Prompt

```
C:\>cd\

C:\>mkdir dev

C:\>cd dev

C:\dev>git clone https://github.com/chrisbu/da2
Cloning into 'da2'...
remote: Enumerating objects: 60, done.
remote: Counting objects: 100% (60/60), done.
remote: Compressing objects: 100% (40/40), done.
Receiving objects: 100% (60/60), 64.58 KiB | 7.17 MiB/s, done.

Resolving deltas: 100% (5/5), done.

C:\dev>cd da2

C:\dev\da2>
```

Select Administrator: Command Prompt

```
Resolving deltas: 100% (5/5), done.

C:\dev>cd da2

C:\dev\da2>git checkout tags/Step1

C:\dev\da2>git branch Step1-cjb

C:\dev\da2>git checkout Step1-cjb
Switched to branch 'Step1-cjb'

C:\dev\da2>
```

valcon

Check the project builds and runs

1. From da2 folder, run:
`mvnw clean package`

```
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:12 min
[INFO] Finished at: 2022-03-23T12:44:04Z
[INFO] ------------------------------------------------------------------------

C:\dev\da2>
```

Expected output at the end of the mvnw command

The mvnw command uses **maven** to “clean” the codebase (ie, remove previous builds), and “package” the code, which compiles the code, runs tests and outputs a java jar file (in the target/ folder).
Maven is a package manager and build tool. You use it to declare which dependencies your code needs (Java equivalent of Javascript’s npm or Python’s pip).
When you run this command you’ll see maven downloading (and caching locally in ~/.m2/) the project’s dependencies. We’ll explore this more.

2. Run the project: `mvnw spring-boot:run`
This starts up the application (web server), you can browse to http://localhost:8080 (and see an error page at the moment).

3. CTRL+C a couple of times to stop the application.

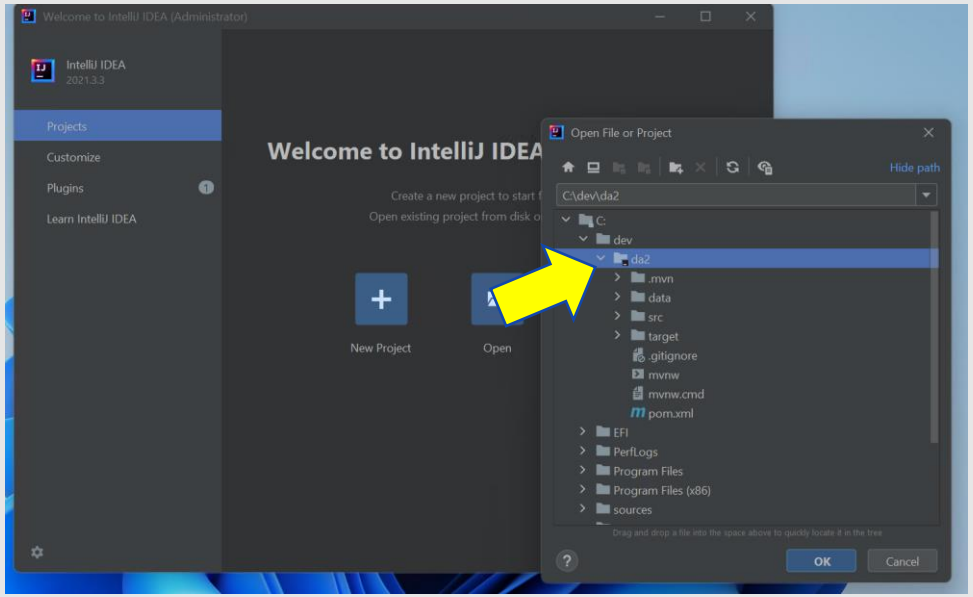
valcon

Section 3

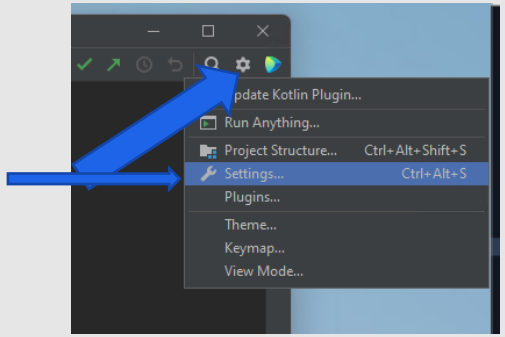
Open and run the project with IntelliJ

valcon

Open the project with Intellij

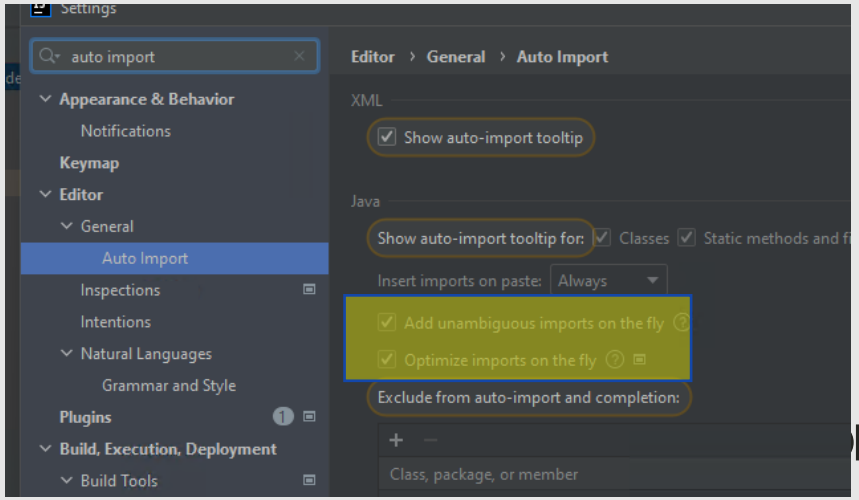


- Click Open and select the project folder.
- If prompted to Trust the project, do so.
- It'll take a while to open, downloading pre-shared indexes etc… (status bar at the bottom)



Change a couple of settings:

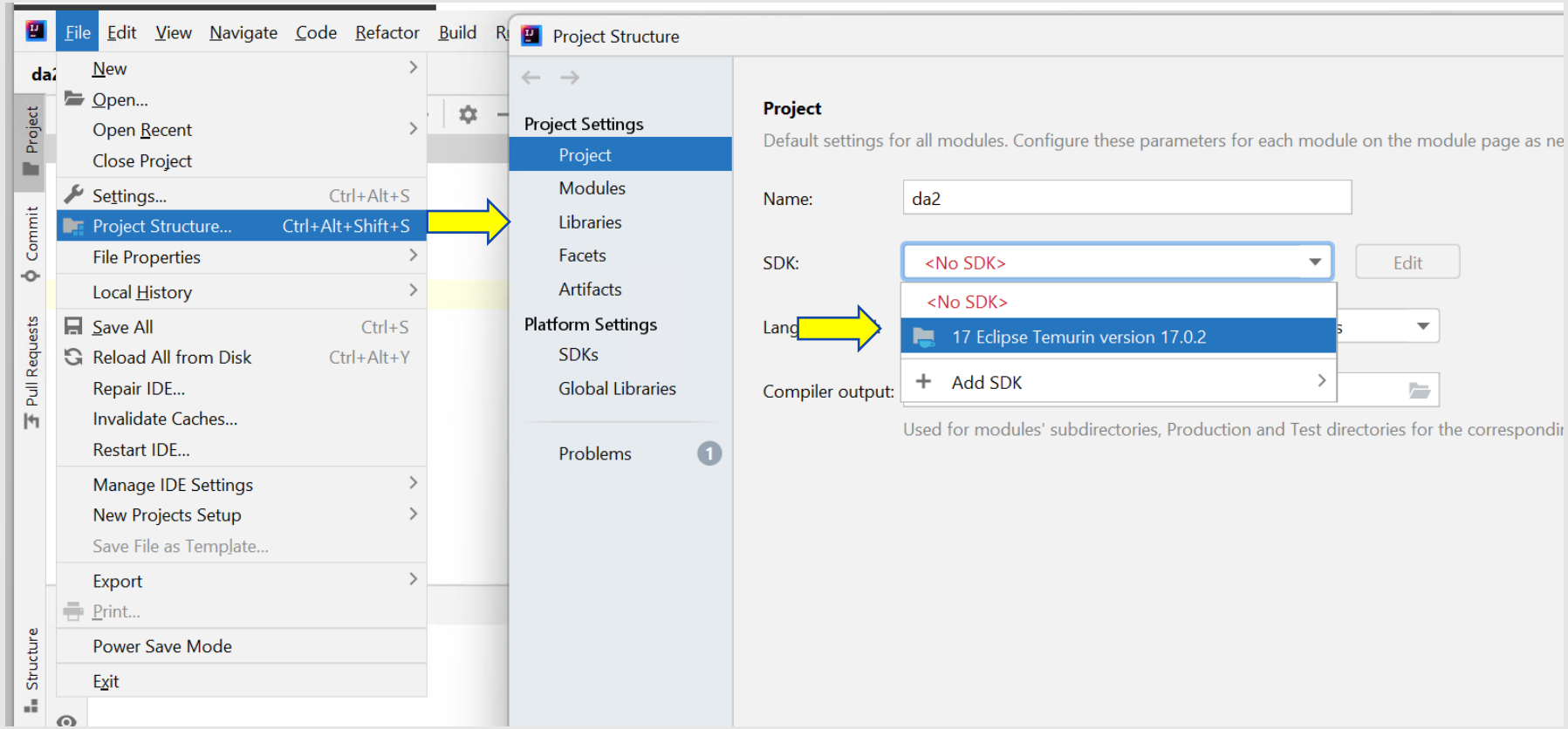
- Go into settings (cog -> Settings)



- **Optional:** Appearance -> Theme. Feel free to pick your favourite theme (I prefer intellij light)
- **Mandatory**: Type "Auto Import" in the search box
- **Tick** "Add unambiguous imports on the fly" and "Optimize imports on the fly"

Set the JDK on the project structure

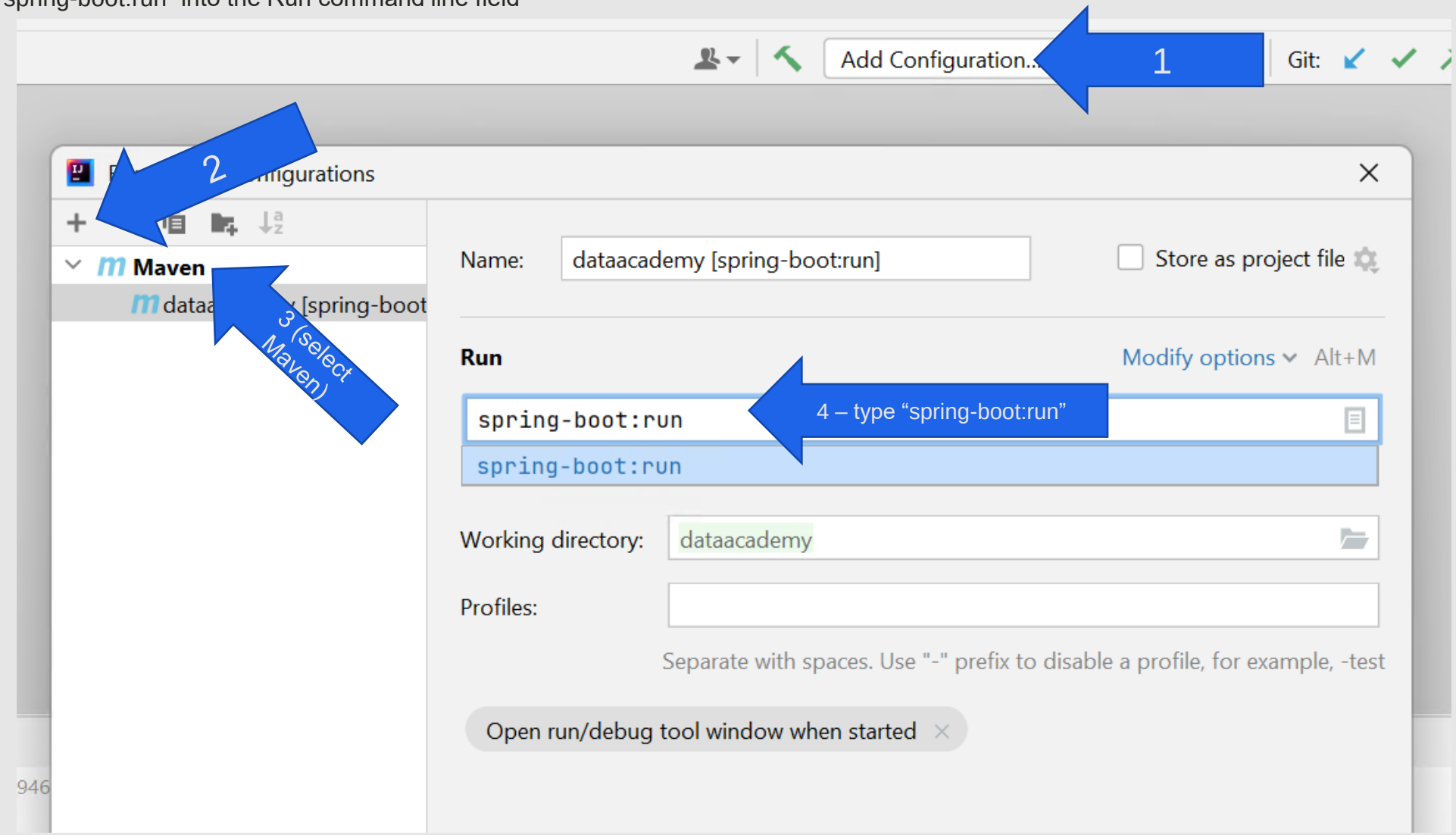
Select the JDK 17 we previously installed
as the project JDK



valcon

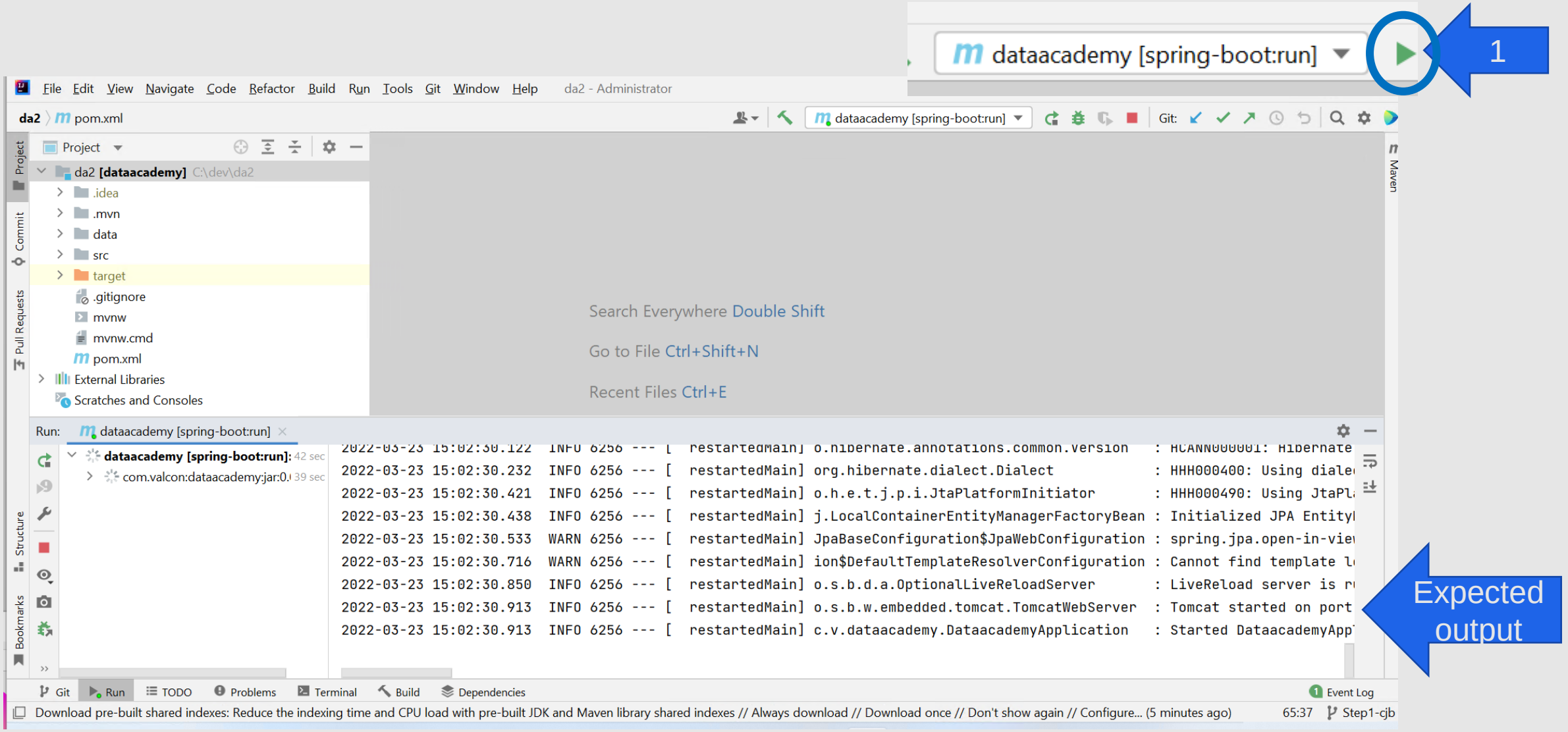
Create a run configuration

Click “Add Configuration”
Click + and select “Maven”
Type “spring-boot:run” into the Run command line field



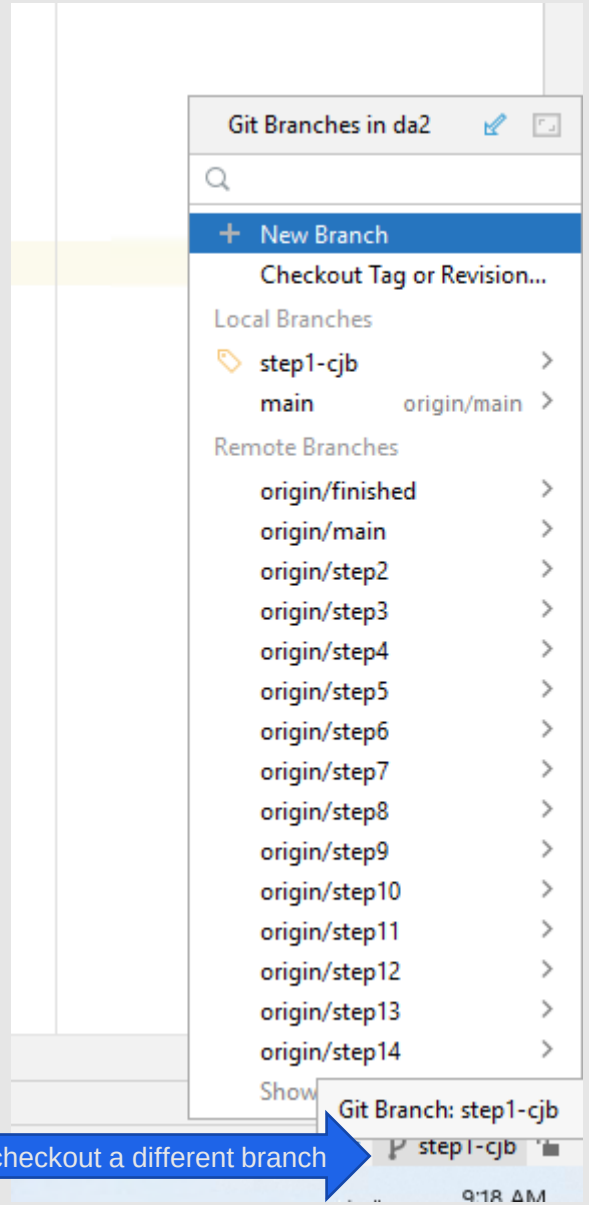
valcon

Run the application in Intellij

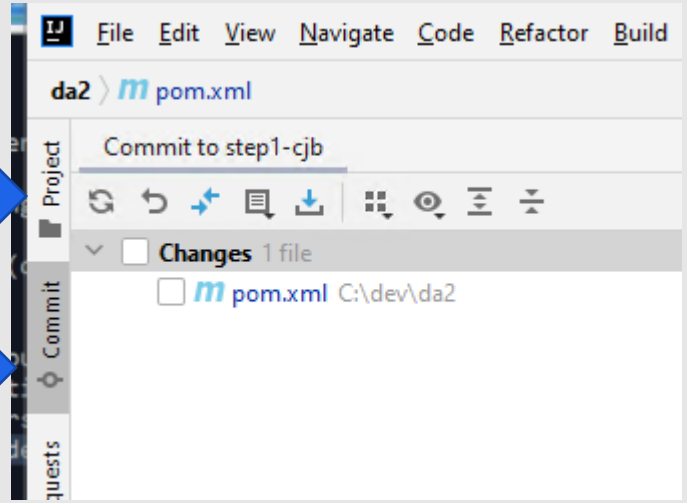


valcon

Git in IntelliJ + other hints

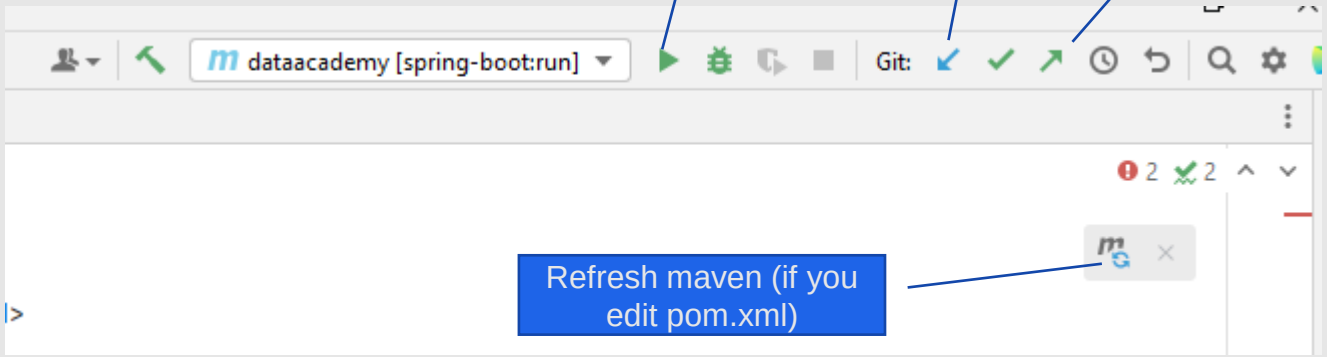


Click to checkout a different branch



Click to see your project files

Click to see modified files for commit



Run / Stop and restart

Git Pull

Git Push

Refresh maven (if you edit pom.xml)

Press Shift twice to open a “do anything / go anywhere” menu

valcon

Section 4

Lessons and exercises

valcon

Steps to talk through + exercises

Each step has a corresponding branch in the repo.
Github links show diff between steps

Step1 – exploring the project
- Look at the maven pom
- Version numbering
- Dependencies
- Add springdoc-openapi-ui (dependency)
- Swagger url: http://localhost:8080/swagger-ui.html

Step2: simple GET request
https://github.com/chrisbu/da2/compare/Step1...step2
- Add a data object (Order)
- Add a controller with GET and return an instance of that object.

Step3: Add a database (H2)
https://github.com/chrisbu/da2/compare/step2...step3
- Change order to an Entity
- Add an Order Service
- Add an Order DAO
- Wire up the Order to the Controller
- Add a database - application.properties
- Populate with data - http://localhost:8080/h2-console

Step4: Add more functionality
https://github.com/chrisbu/da2/compare/step3...step4
- Add getOrderById

Step5: Add parent/child relationship
https://github.com/chrisbu/da2/compare/step4...step5
- Add order items - parent/child relationship

Step6: Add service with external callout
https://github.com/chrisbu/da2/compare/step5...step6
- Add shipping details external callout

Step 7: Add unit tests
https://github.com/chrisbu/da2/compare/step6...step7
- Add unit tests, mockito for both
- Add in memory database for h2 via property files

Step 8: Method injection
https://github.com/chrisbu/da2/compare/step7...step8
- Add alternative method injection for shipping service

Step9: Running from the command line (skip checkout)
https://github.com/chrisbu/da2/compare/step8...step9
- Command line mvnw clean package
- java -jar target/...jar

Step10: Sending data to the application
https://github.com/chrisbu/da2/compare/step9...step10
- Save a new record

Step 11: Adding authentication
https://github.com/chrisbu/da2/compare/step10...step11
- In memory spring security

Step 12: Add better authentication
https://github.com/chrisbu/da2/compare/step11...step12
- Database backed spring security

Step13: Read security details in business logic
https://github.com/chrisbu/da2/compare/step12...step13
- Get current logged in user

Step 14: Exercises
https://github.com/chrisbu/da2/compare/step13...step14
- Exercise 1: fix the unit test with Mockito and mock security service
- Exercise 2: User logged in user as customer name. Only allow customer to retrieve or save their own orders.
- Exercise 3: fix the unit to test exercise 2; add another unit test to confirm a user can't retrieve another users data