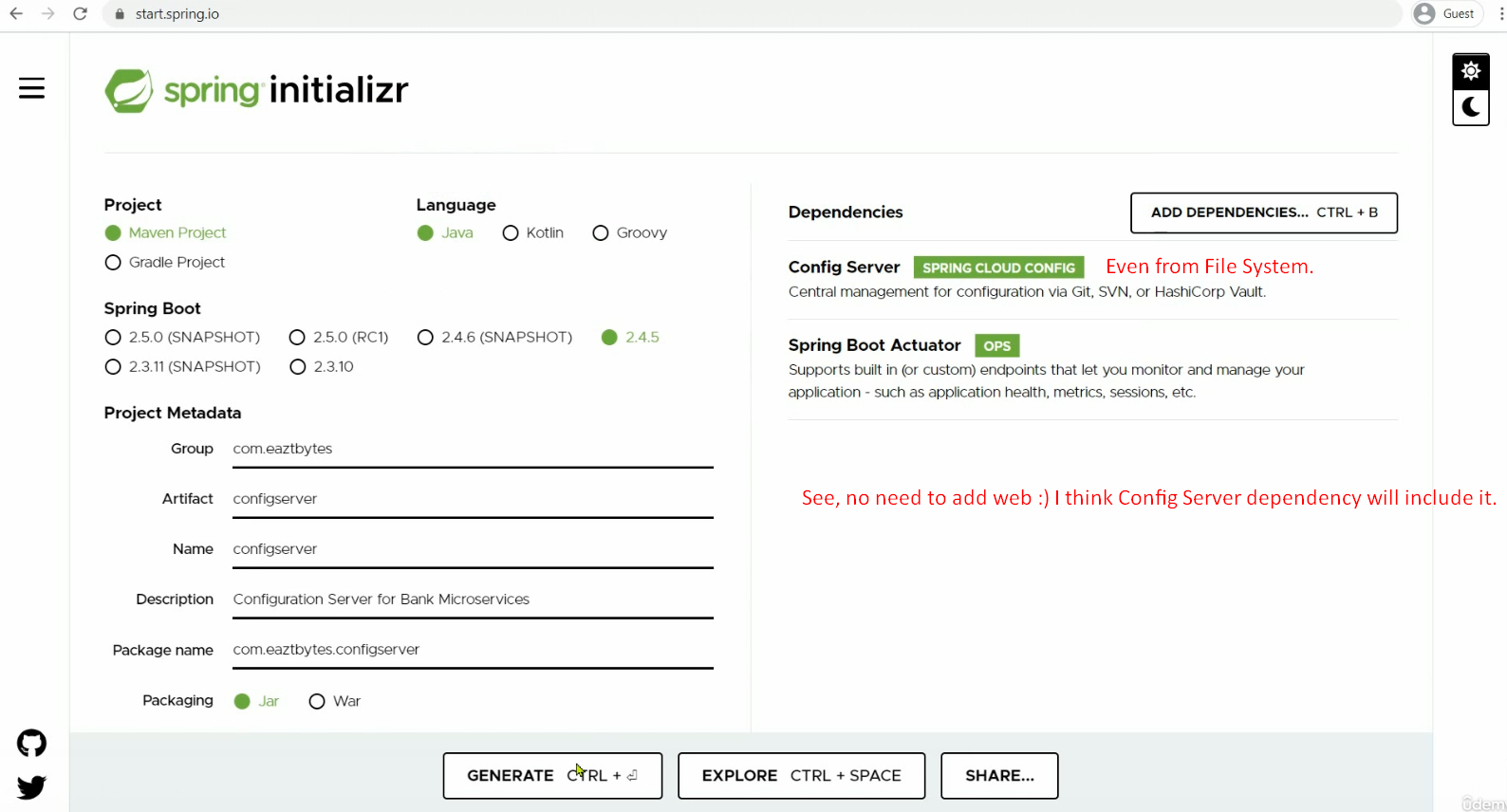
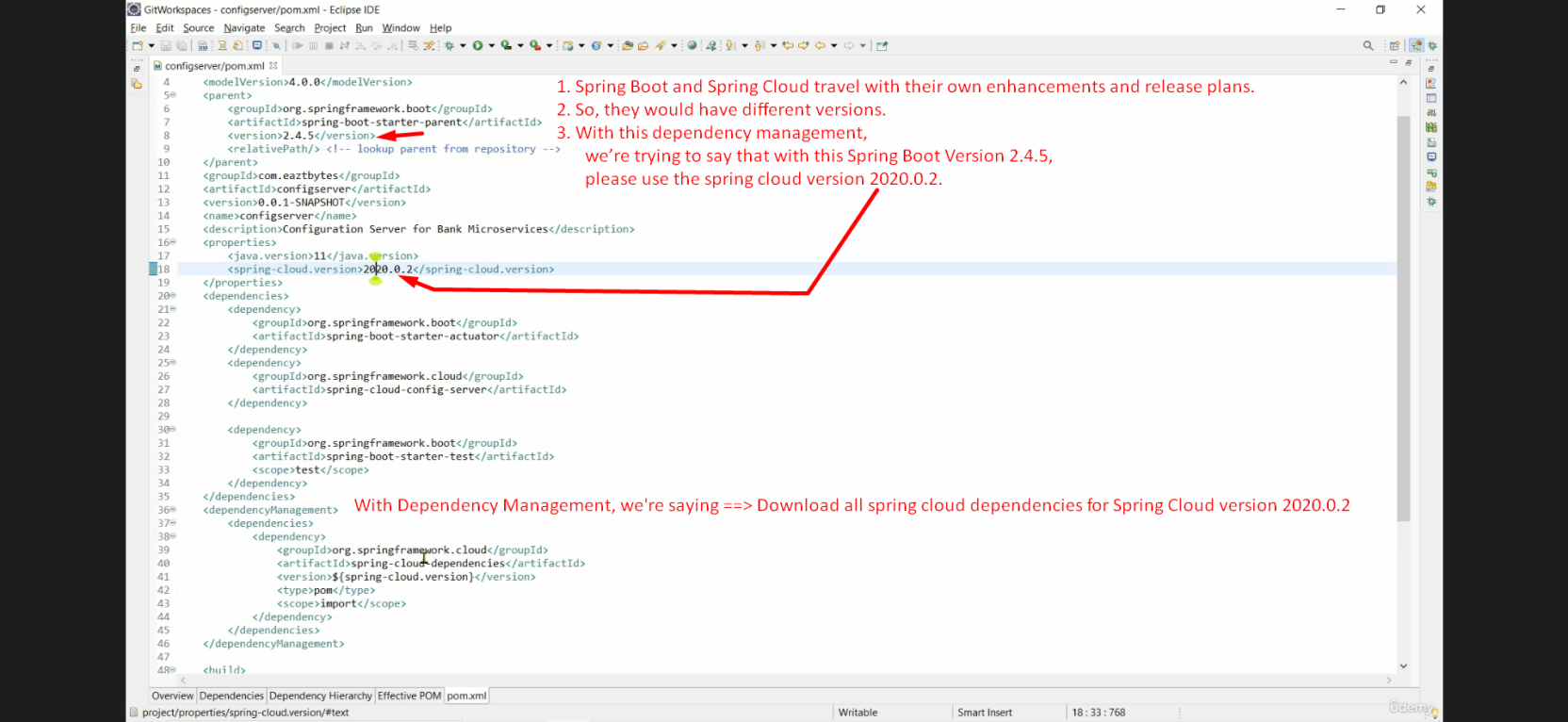
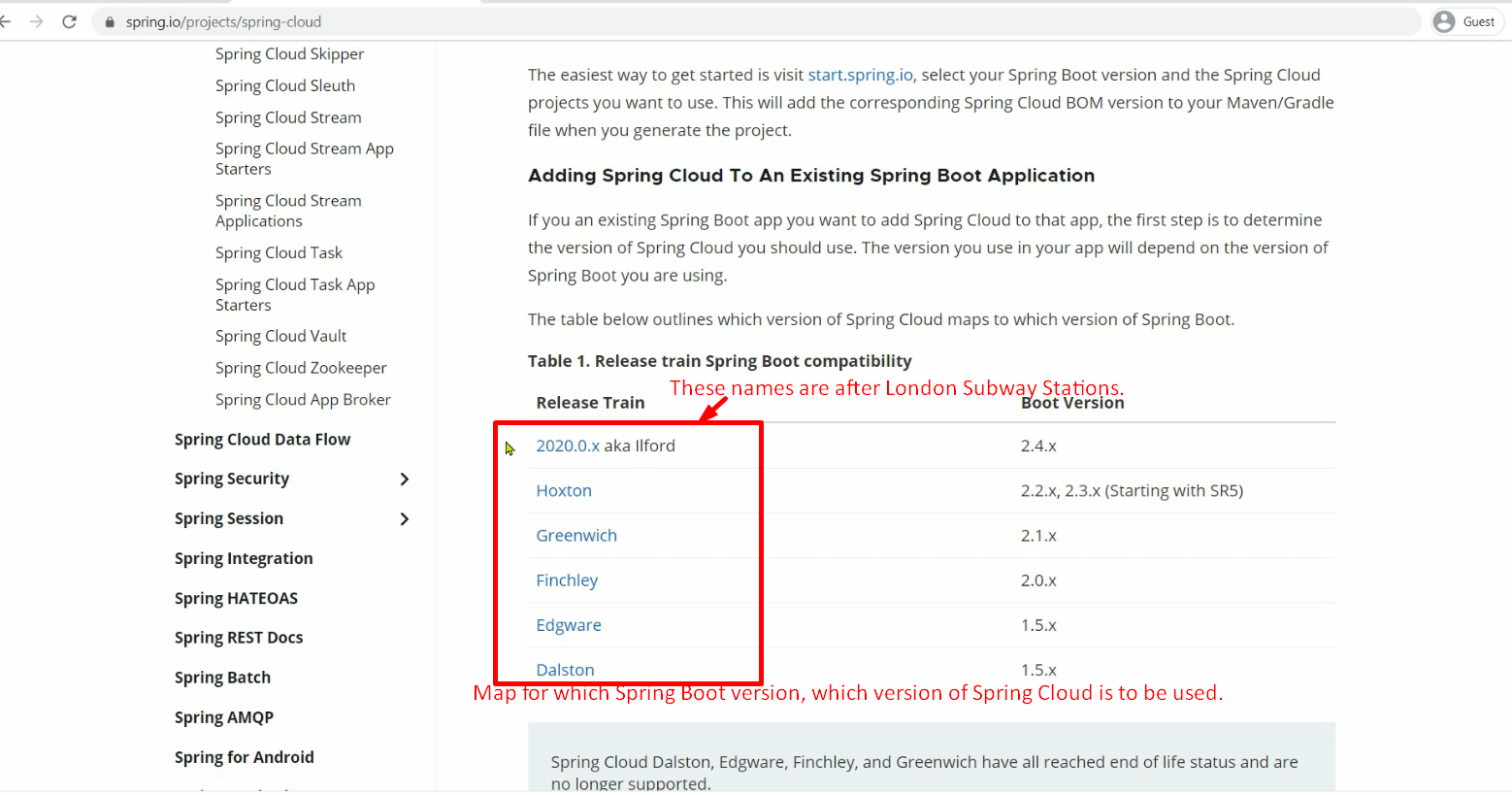
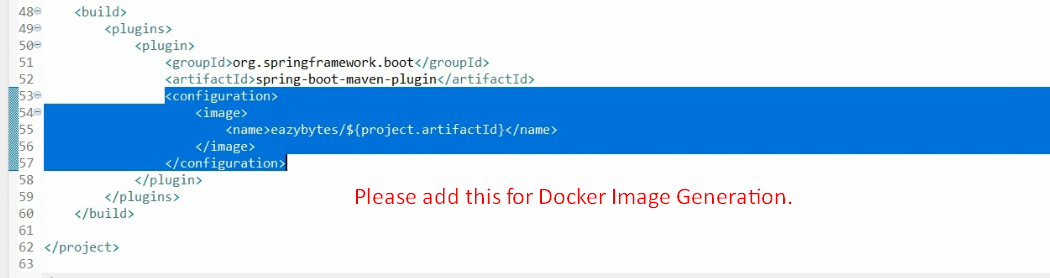
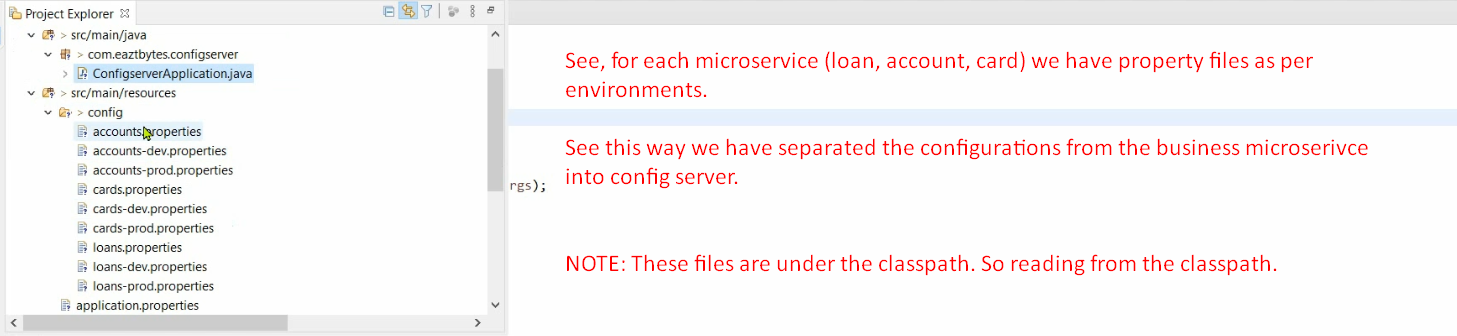
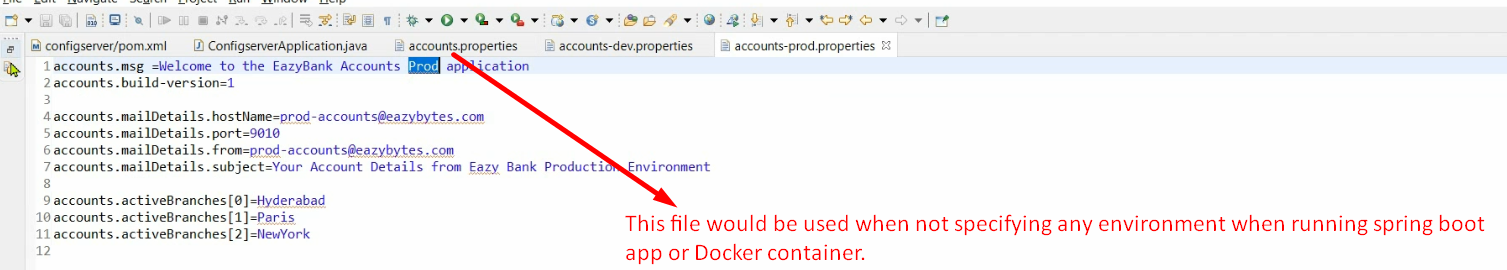
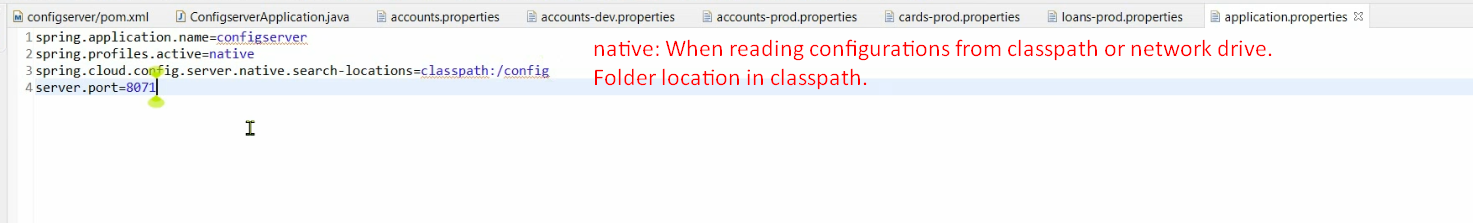
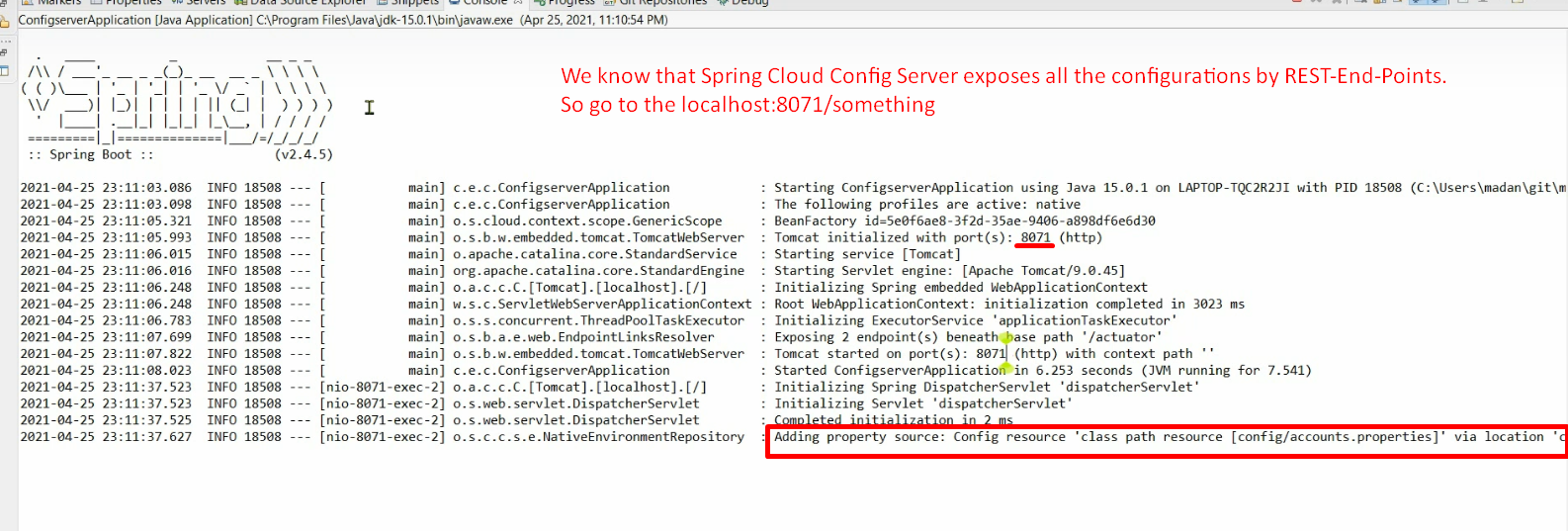
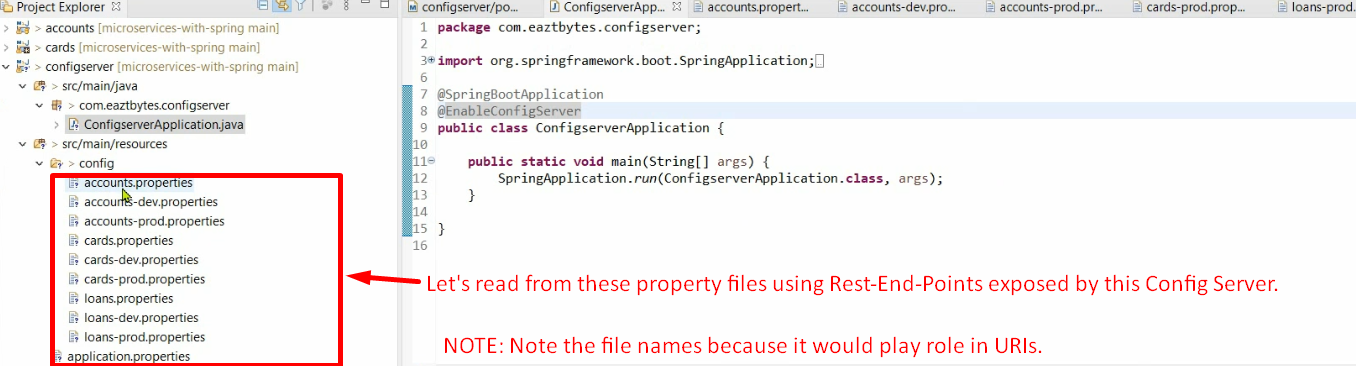
1. **Agenda:**
   1. How to create a service which will act as Config Server which will read all the configurations from a centralized repo containing all the configurations.
2. Config Server is a spring boot app. actually it is also a microservice.
3. 
4. Why this dependency? 
5. 
6. 
7. **@EnableConfigServer**:
   1. This annotation would convert your microservice into **Spring Cloud Config Server** which then can read configurations from centralized repository and can expose all your properties through Rest-End-Points.
8. Now the next step is configuration inside Config server to tell it from where it has to read the configurations.
9. Multiple places from where configurations can be read
   1. File System (local or shared network)
   2. Classpath.
   3. Github.
   4. S3 Bucket from Cloud.
10. 
11. 
12. How to tell our Config server about where to read configurations from?
    1. Go to the application.properties file.
    2. Configure the followings:
       1. spring.profiles.active=native  
          When reading from classpath or from a network drive.  
          When reading from git, give “git” instead of “native”.
       2. 
13.   
    Spring Cloud Config will expose all the properties files through RESTful Endpoints.
14. 
15.   
    NOTE: [http://localhost:8071/accounts/<env](http://localhost:8071/accounts/%3cenv)> 🡸 If <env> doesn’t match with any property file then default file would be returned.  
    NOTE: <http://localhost:8071/accounts-dev/default> is also working which is returning only accouts-dev without account.properties (this is default property file).
16. 