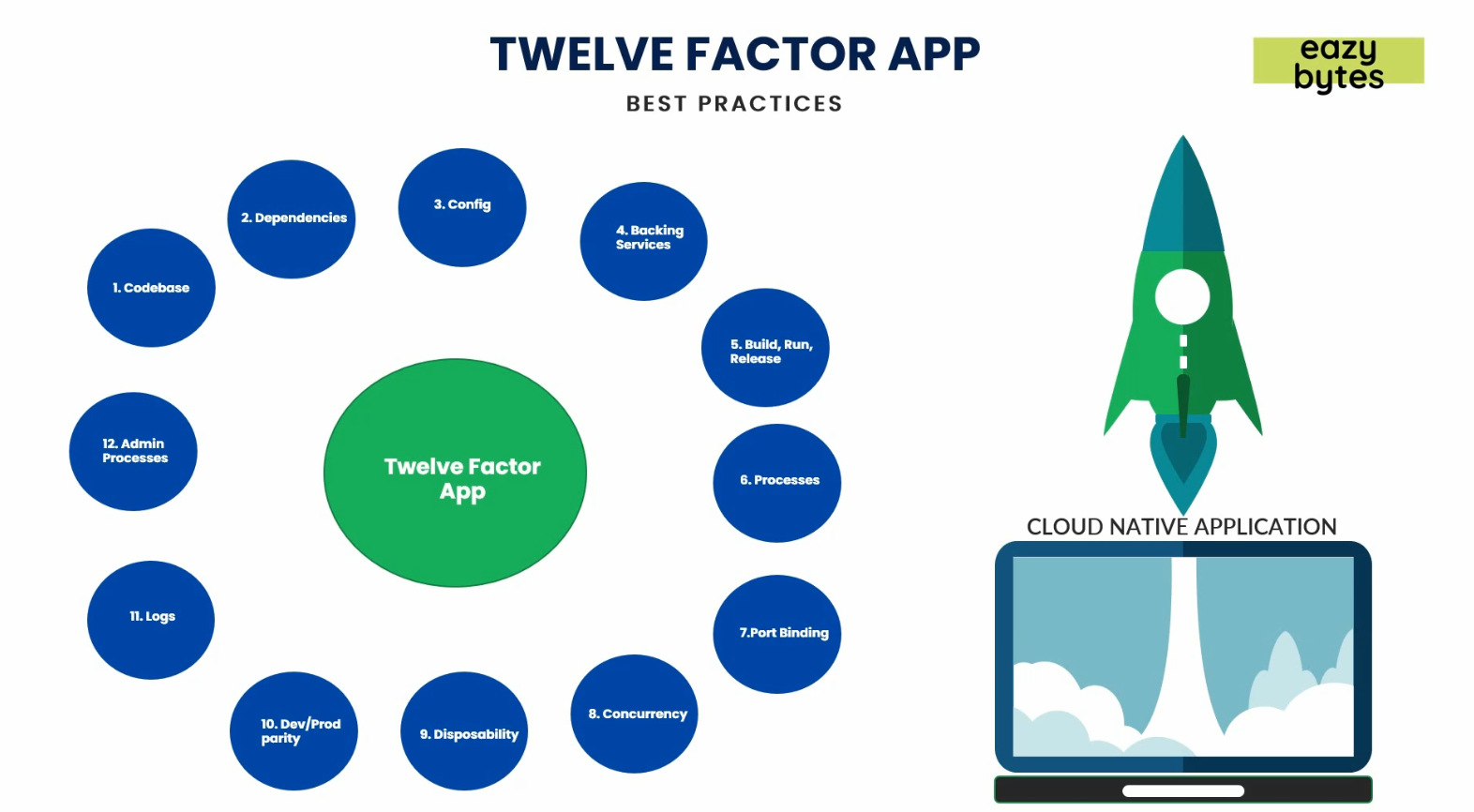
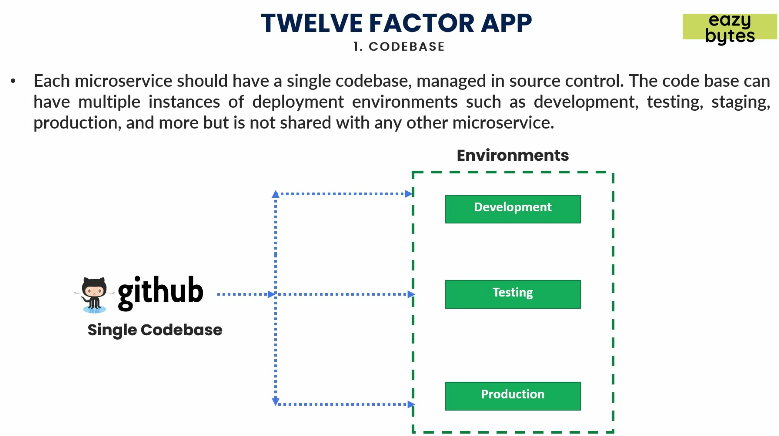
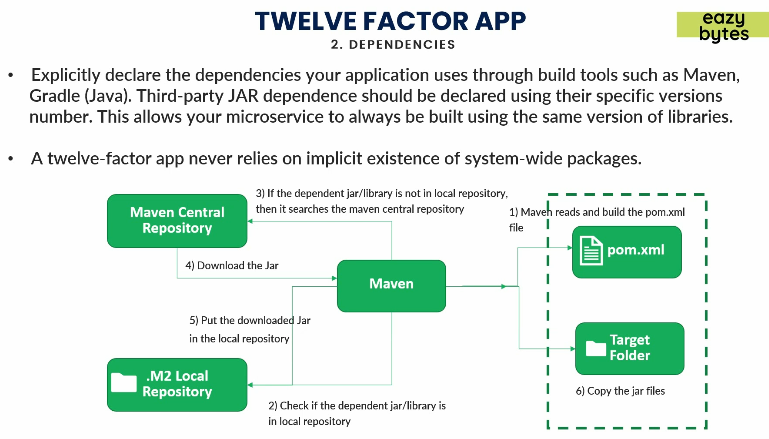
1. Let’s discuss 12 principles or factors to build/develop a successful Cloud Native App.
2. 
3. 
4. NOTE: Remember these 12
   1. As they are needed in interviews
   2. And when developing successful microservice.
5. Let’s discuss them one by one.
6. 12 factors

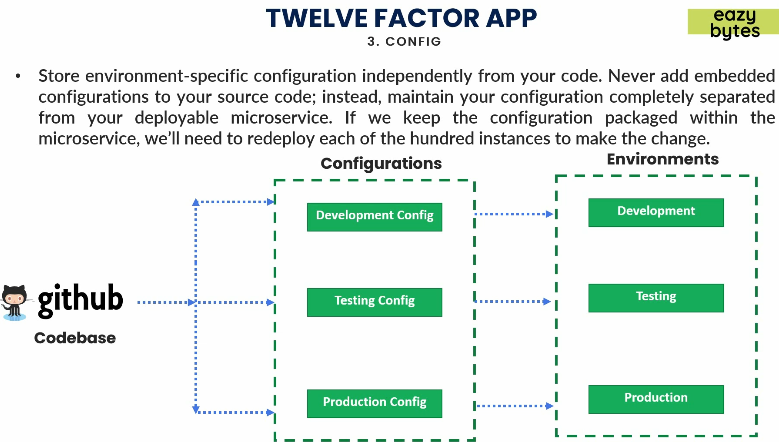
01 Code Base:

1. 
2. Each microservice must have its own codebase/repository.
3. Means each must have its own github repository.
4. This way you can achieve **parallel development feature** or agility feature using microservice.
5. Using same codebase, you can maintain different environments like you can deploy the same code base after building a docker image into development, testing, production environments.  
   Regardless of how many environments you have, you must have a separate code base/repository for each microservice.

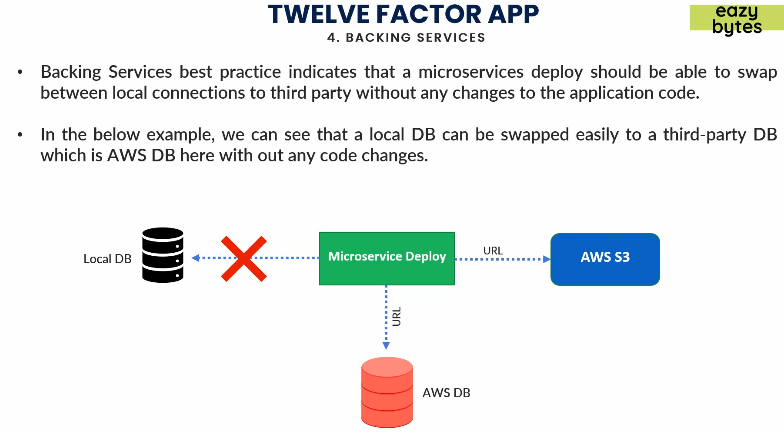
Factor 02: Dependencies

1. 
2. Explicitly specify all the required dependencies in build configuration (Maven, Gradle).
3. Don’t add any dependencies manually in classpath.
4. The reason is that once you create an image from the microservice and the same image should be shared with other team members and should be used in any environment.   
   If you don’t do so, then you are losing the feature of cloud native which is building & delivering your services faster into production.

Factor 03: Configuration

1. 
2. You should not have environment specific configuration inside your code.  
   You should maintain them independently.
3. Like you may have different configuration for DB, SMPT based on environments.
4. Instead, you should maintain such environment specific configuration outside of your source code.  
   In this way, when deploying docker image in any environment, it will work properly without making any change to the container.
5. **NOTE**: Previously, we gave configuration in application.properties file.  
   But we will see how to externalize the configuration somewhere like somewhere in github repo and based on the environment, container can refer the corresponding configuration.
6. **Two advantages**:
   1. No need to change the content of container when deploying in different environment.
   2. No need to restart container as property file resides outside of the container.

04: Backing Services

1. Very similar to 3rd principle (Configuration).
2. Think of a scenario where your microservice is using a DB.
3. In local environment, you use local DB but in Cloud environment, you use cloud provided DB.
4. This factor says that you should not make changes inside your docker image to change your backing service (DB)
5. This can be achieved by maintaining your configuration in an external manner.