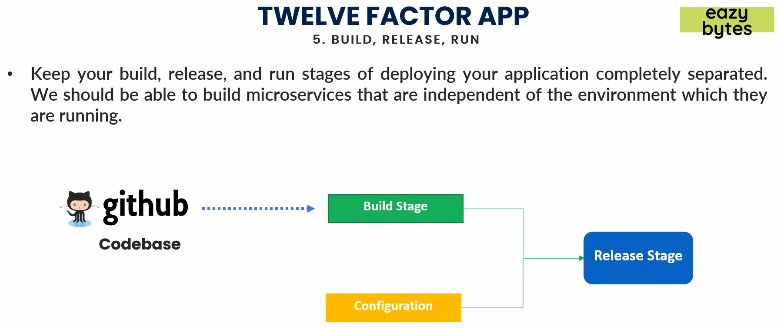
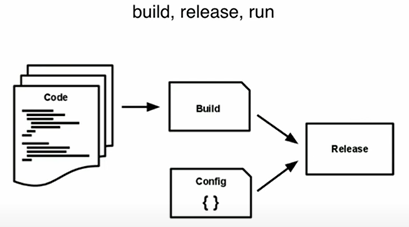
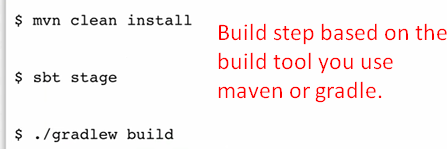
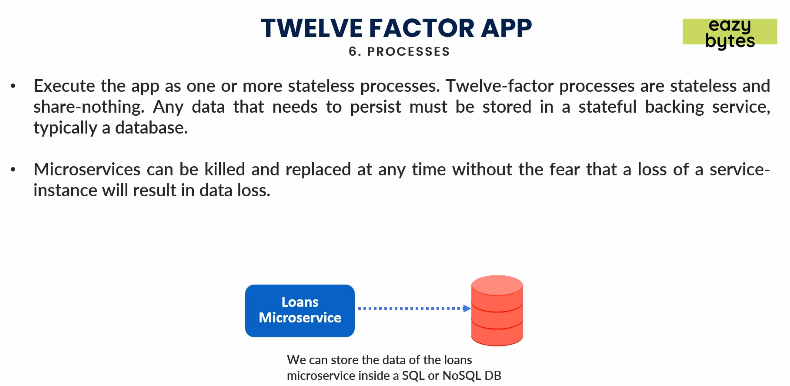
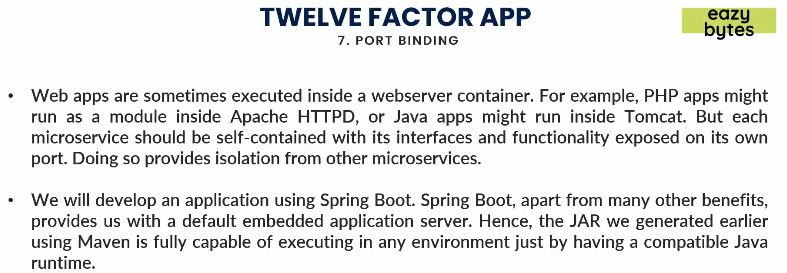
Factor 05 Build, Release, and Run:

1. This factor indicates that the build, release and run stages of deploying your app should be completely separated which means you should be able to build microservice that are independent of the environment in which you’re running.  
   It means if you’re building a docker image, that docker image could be used in any environment and to achieve that, your build stage should be isolated from the release stage.  
   Now, based on the environment in which we want to run our app, we combine the build artifacts with the specific configuration which is release stage and the final artifact contains everything source code and configuration to be run in that specific environment.
2. The following description is from the source <https://www.youtube.com/watch?v=94PxlbuizCU>
   1. This principle states that your deployment process should be executed in 3 discrete steps.   
      Ideally these are all automated into a single step but the covers there should be 3 discrete operations.
   2. Build: To compile your source code and generate build artifacts.
      1. Release: In which you combine build artifacts with your configuration for a particular environment because that configuration should be separated from your source code and you prepare a release image. That release image represents everything your app needs to run and finally you run the app from the release image.  
         
   3. To demonstrate this:  
        
      The release phase depends on the platform you’re deploying to.

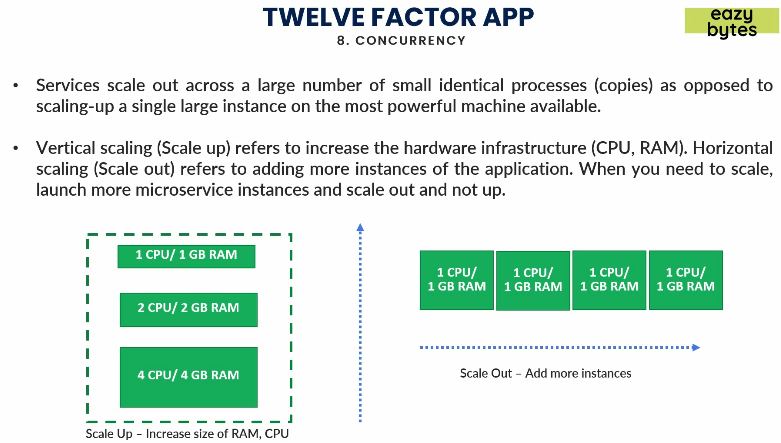
Factor 06: Processes

1. 
2. Microservice 1 calls microservice 2 and microservice 2 can call microservice 3.  
   So, make sure we are communicating in a stateless process.   
   Stateless we mean, we should not store anything inside a session like we did in Servlet and JSP.  
   The reason is that a microservice can have any number of instance can go down or can be killed anytime.  
   If there is a real need to store something across different request, then store in a DB so that other instances of the same microservice can use the data.

Factor 07: Port Binding

1. 
2. For same microservice, we can have n number of instances. But all these different microservices along with different instances must have their own ports defined.
3. This way, you are provides some isolation from other microservice inside your app.
4. Spring boot helps us to a great extent to maintain this principle like it gives you embedded server. Once executable jar is generated, using Docker command you can always control your ports that you need for your microservice.
5. So, this way we’re good if we’re using Docker, SpringBoot and Docker images for this port binding principle.

08: Concurrency

1. 
2. This principle indicates that if whenever you need to scale up or down your app, you should never do vertical scaling but you should always rely on the horizontal scaling.
3. Vertical scaling indicates the increasing of the resources in the infrastructure for an app.
4. Don’t scale up vertically. Because when you scale vertical, and later on you want to scale down, you need to shut down all the microservices whereas in horizontal scaling, you just kill some instances of a microservice without affecting other microservices.  
   This is what concurrency is teaching us.