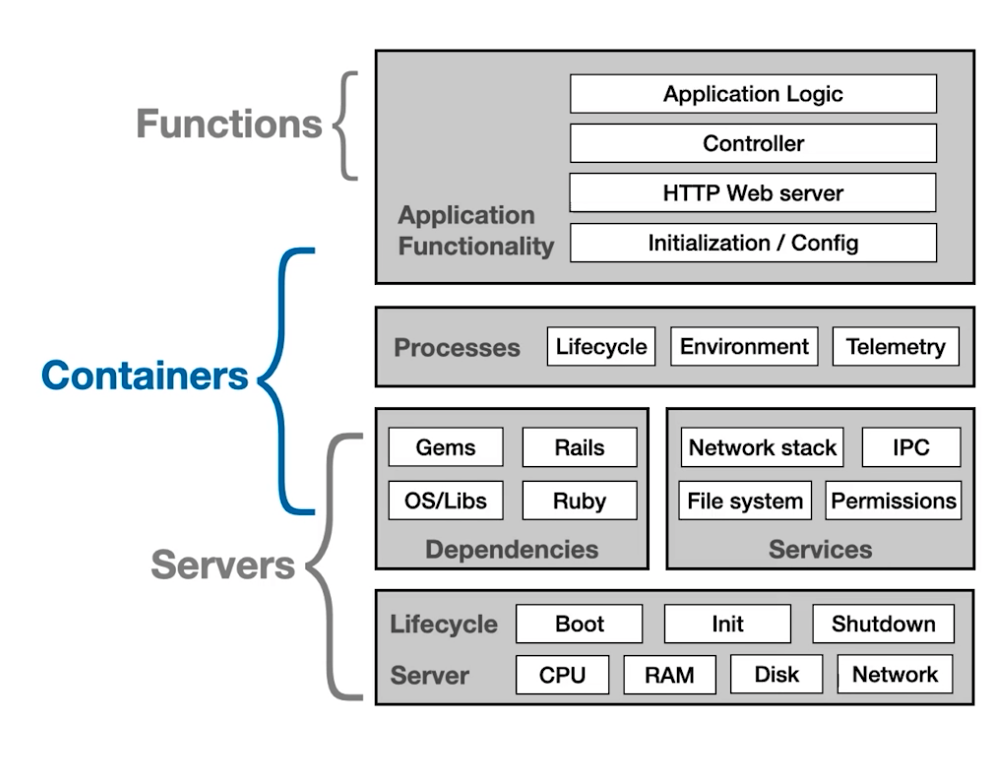
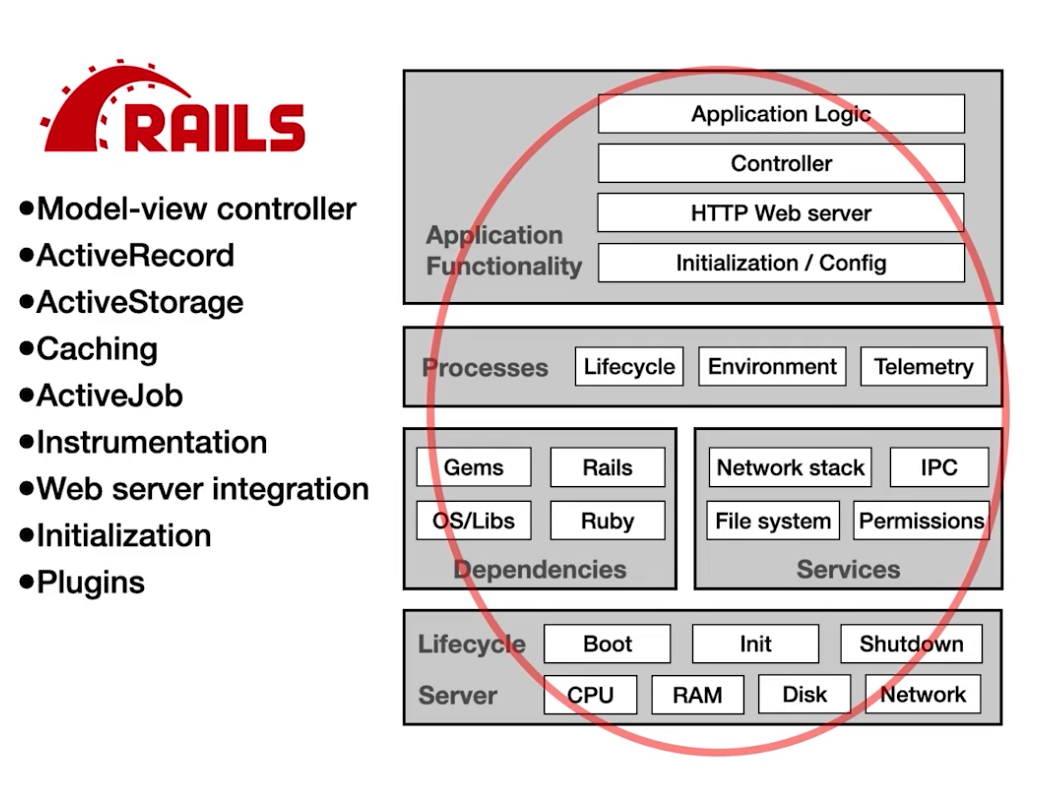
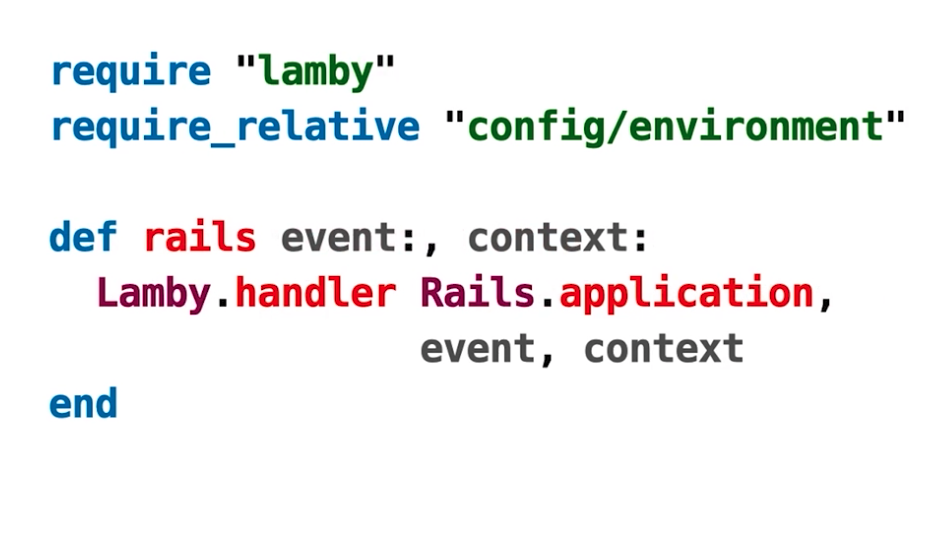
* Serverless is a bad term because it’s not precise. Servers are involved.
* What constitutes as serverless?
* When Amazon created EC2 services, they created abstractions that made devs feel like they were interacting with stand-alone machines
* FaaS - functions as a service
  + FunctionsFramework.http “hello” do |request| “Hello, world!\n” end
    - No routing
    - No controller
    - No disk (data in/ data out)
      * Might not be able to write to disk
    - No CPU
      * May be able to access CPU resources though
      * May alter CPU allocation as app is running
      * Upside is that you don’t pay for cost of CPU even when it’s not being used



* Containers don’t have a start-up/init and are not concerned with CPU/disk



* Rails is tightly coupled with the entire stack



* Lamby is a gem that lets you create FaaS

Tips for Rails apps in this day and age of server abstraction

* Do less during initialization
  + Less precomputation, less cache warmup
* Defer initialization
  + Lazily loading
  + Cold starts matter
* Be webserver agnostic
  + Whether it is puma or unicorn, it shouldn’t matter
* Don’t run background threads
  + If you need async, use TaskU
* Avoid the file system if possible
  + It might not be available
  + If you need to use temporary files, use ruby’s Tempfile for temp storage
* Log to stdout/sderr (not files)
* Use FileUtils instead of shelling
* Use Ruby’s standard interfaces instead of reaching out to OS tools
* Follow these if you write a gem