MicroServices Review

* **Architectural** software design
  + One large application is broken into several smaller independent services that communicate with each other
* Pros
  + Highly scalable
    - You scale per feature rather than duplicating the entire application
  + Greater Durability
    - One service can fail and the entire application does not crash
  + Develop and deploy each service individually
  + Separation of services allows each service to be written in any programming language using any technology
    - Very helpful with companies that have massive software ecosystems
  + Services are black box
    - They are already an abstracted piece of software that you can communicate with via http
* Cons
  + Microservices have extra configuration and upkeep
    - Think about configuring routing and registering services
  + Greatly increased network traffic
    - Applications require greater bandwidth
    - Introduced extra latency to an application
  + Debugging and fixing microservices can be difficult
    - The bug might lie 3 services away caused by another microservice
  + Services that “mostly” work can be compounded
    - A 99% B 99% C 99%
      * 97% chance a call that requires all three services works correctly

Technologies

* Eureka (Discovery Service / Registration Service)
  + Microservices will register with Eureka
    - @EnableEurekaServer (annotation that marks this application as the registry)
    - @EnableEurekaClient (annotation that tells this application to register with Eureka)
  + Serves as a registry / phone book where microservices can look to find other microservices
  + IP address of your services will change as you create and decommission them
* Zuul (Gateway Service / Routing service)
  + Allows you to crate a single unified domain for an API
  + Avoids having an excess of endpoints for your architecture
* Hystrix (Circuit breaker / Fallback assurance)
  + Provides a safety net and back up to your microservices
  + If a microservice fails hysteric can shunt incoming requests to an appropriate backup service
    - @HystrixComman(fallbackMethod = “”)
    - @CiruitBreaker
* FeignClient (Inter service communication client)
  + Allows us to communicate between our microservices
  + @EnableFeignClients
  + @FeignClient(“Something service”)
    - Whatever service we are trying to communicate with
    - Write the abstract methods in an interface where you five the uri mapping and method signature
* RestTemplate
  + Allows you to make HTTP requests
  + Useful for when you need to make HTTP requests to APIs not on your Eureka

YAML(YAML ain’t markup language)

* Data serialization language designed to be human readable
* Whitespace sensitive
  + Encourages two spaces over tabs
* A common format for configuration files
  + .xml
  + .json
  + .yaml
    - .yml

Docker

* Docker is a containerization software
* Container
  + Lightweight virtual environment to house your application
  + Fixes the “it works on my machine problem”
  + Running a docker container can be done on ANY linux machine that has docker
  + Greatly simplifies the deployment of applications
* Advantages over an entire VM aka an EC2
  + Containers are very fast to create
  + Less management that a VM
  + Can interface directly with the host machine’s OS so it is more efficient

Docker Terminology

* Image
  + Blueprint for a container
* Dockerfile
  + A step by step instruction on how to create an image
* Dockerhub
  + An online repository for Docker images
* Volume
  + The diskspace allocated to a container

Docker Commands

* Docker ps
  + Show all docker processes (running containers)
* Docker build
  + Use a Dockerfile to build an image
* Docker run
  + Use an image to create and run a container