4/5/2022

1. What is Taylorism?
    a. Adoption of command and control management
    b. Organizations divided into functional silos
    c. Decision-making is separated from work
    d. Ex. Project management -> architects-> Developers-> Testers -> Operations -> Security

2. Software engineering is not civil engineering
    a. Treat software development like product development

3. Required DevOps behavior
    a. Since enterprises see change as complex and time - consuming. DevOps breaks big projects into a series of small, manageable changes.
4. What is infrastructure as code
    a. It describes infrastructure in an executable textual format
    b. Ephemeral infrastructure can be used and then discarded. Servers are built on demand, via automation
    c. Rather than patching a running container, immutable delivery is making changes to the container image, then redeploying a new container
5. What is continuous integration and continuous delivery (CI/CD)
    a. CI and CD are two things
        i. CI - Continuously building,testing and merging to master
        ii. CD - Continuously deploying to a production- like environment
        iii. Benefit is faster reaction time, moving faster, and reducing the risk in integrating code and ensures that code can be rapidly and safely deployed to production
    b. CI/CD pipeline
        i. Unit testing
        ii. Code quality checks
        iii. Integration testing
        iv. Security testing
        v. Vulnerability scanning
        vi. Package signing
    c. How DevOps manages risk
        i. Deployment is king

Summary and Highlights

- Taylorism was designed for factory work and software development is bespoke, that is, more like craftwork, and that working in silos leads to mistakes and bottlenecks.
- Team ownership and stable teams make software development more like product development rather than project management.
- Developers want innovation, while Operations want stability.
- Required DevOps behaviors include shared ownership, collaboration, embracing change, and data-driven responses.
- Infrastructure as Code describes infrastructure in a textual executable format.
- Ephemeral infrastructure can be used and then discarded because servers are built on demand, via automation, using Infrastructure as Code techniques.
- Continuous Integration is building, testing, and integrating every developer change into the master branch after tests have passed.
- The benefits of Continuous Integration include faster reaction time, moving faster, and reducing the risk in integrating code.
- Continuous Delivery ensures that code can be rapidly and safely deployed to production by delivering every change to a production-like environment.
- The five principles of Continuous Delivery have to do with quality, working in small batches, automation, continuous improvement, and shared responsibility.