- **In your own words, explain the difference between continuous integration, continuous delivery, and continuous deployment.**

  Continuous Integration: Developers merge their codes to a single repository. After each commit, an automated build and testing sequence would be triggered.

  Continuous Delivery: Changes in software can be made ready for production directly, by tracking the commits and testing the code in production-like (simulated) environments – Automating the entire software release process.

  Continuous Deployment: Every change in source code is tested and corrected by directly deploying in the production environment.

- **How does DevOps team model (e.g., site reliability engineer) differ than a a NoOps team model (e.g. Netflix team)? What differences in architecture allow for a NoOps model?**

  In a DevOps team model, Development and Operations work together throughout the software development life cycle to ensure quality at each phase. Operations collaborate with development in building, testing and maintain the software.

  Whereas in a NoOps model, Development and Operations never work together. Development relies on Continuous Integration.

- **Explain the principle of Every Feature is an Experiment**

  With Continuous Deployment, developers consider planned features as experiments, where some features could be taken offline after response from the users. Every feature is justified after having some real-time data regarding the feature.

- **Explain the principle of Be Fast to Deploy, Slow(er) to Release.**

  Deploy code into production, but the features aren't available to customers. Any new feature is deployed into production and evaluated, before being publicly announced. The customer would be running the code, but it isn't visible on the UI, kind of a *dark launch (shadow launch)* of the feature.