**Microservices Notes**

**Monolith Architecture:**

The original architecture. All software components are executed in a single process. No distribution of any kind. Strong coupling between all classes.

**Service Oriented Architecture:**

Apps are services exposing functionality to the outside world. Services expose metadata to declare their functionality. Usually implemented using SOAP & WSDL. Usually implemented with ESB.

**Microservices Architecture:**

Microservices architectural style that structures an application as a collection of services that are. Highly maintainable and testable. Loosely coupled. Independently deployable. Organized around business capabilities. Owned by a small team.

**Deploying Microservices:**

Deployment of microservices is extremely important. Slow and complicated deployment will render the whole system ineffective and useless. Architect should be aware of deployment, not responsible.

**Pipeline:**

In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.

**Continuous Integration:**

Continuous integration is a software development process where developers integrate the new code they've written more frequently throughout the development cycle, adding it to the code base at least once a day.

**Continuous Deployment:**

 Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

**Testing Microservices:**

Testing is important in all systems and all architecture types, With Microservices it’s even more important. Testing Microservices poses additional challenges.

**Unit Tests:**

Tests individual code units, Method, interface, etc. In-process only. Usually automated and Developed by the developer.

**Integration Tests:**

Test the service’s functionality. Cover (almost) all code paths in the service. Some paths might include accessing external objects.

**End-to-End Tests:**

Test the whole flow(s) of the system, Touch all services. Tests from start to end.

**Service Mesh:**

Manages all service-to-service communication. Provides additional services.

Software Components that sit near the service and manage all service-to-service communication. Provides all communication services. The service interacts with the service mesh only.

**Service Mesh Services:**

* Protocol conversion
* Communication security
* Authentication
* Reliability (timeouts, retries, health checks, circuit breaking)
* Monitoring
* Service Discovery.
* Testing (A/B testing, traffic splitting)
* Load balancing.

**Logging:**

Logging should provide wholistic view on the system. Should allow tracing end-to-end flow. Should contain as much information as possible. Can be filtered using severity, module, time, etc