DevOps Engineering Practices
lecture 1

Dr. Divya Srivastava
Assistant Professor
Bennett University, Greater Noida

January 31, 2022

Introduction to DevOps

1. It is just a culture to promote development and Operation process collaboratively
2. DevOps is the collaboration of process and software engineering over the lifecycle of a project.
3. The team works together to reach the final aim starting from design, build, test, deploy, maintenance and monitoring.
4. DevOps enables organizations to serve their customers strongly and better in the market.

Need of DevOps

1. Before DevOps, operation and development teams were working in an isolated environment.
2. Testing and Deployment activities mostly were performed in an isolated manner after design-build step, and they took more time than actual project completion time.
3. Team members usually spend a large amount of time in deploying, testing, designing, and building the projects
4. Human production errors were deployed during manual code conduction.
5. Operations and coding teams generally had different timelines and did not have proper synchronization that results in further delay.

Difference between DevOps and Traditional IT

Traditional IT	DevOps
Once the order for new servers is placed, the development team starts working on testing. The development team has to continue with heavy paperwork as required by enterprises to deploy the infrastructure.	Once the order for new servers is placed, the development team and operations team start the paperwork to set up new servers that result in better visibility of infrastructure equipment.
Projections about failover, data center locations, redundancy, and storage requirements are not clear as no inputs are available from the development team even if they have the depth knowledge of the application.	Projections about failover, data center locations, redundancy, and storage requirements are 100 percent clear because of accurate inputs given from the development team.
In old software development processes, the operations team has no idea of the progress of the development team. Operation team has to prepare a monitoring plan as per their own understanding.	In DevOps, the operations team have a complete idea of the progress of development. Operations team and development team work together to develop a monitoring plan that caters to the current business, and IT needs.
Before go-live, the load testing may crash the application, and the release may get delayed. It affects the overall cost of the project and project delivery deadline.	Before go-live, the load testing makes the application a little slow. The development team quickly fixes bottlenecks, and the application is released on time.

Figure 1: Difference between DevOps and Traditional IT

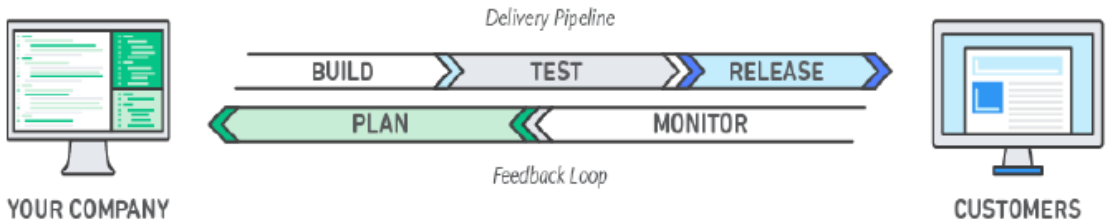

Figure 2: Software Delivery steps

Important Terminologies

1. **Agile**:Describe infrastructure, processes or tools that are adaptable and scalable
2. **Agile Software Development**: Methodology and philosophy, that focuses on user feedback, software quality, and the ability to respond quickly to changes and new product requirements
3. **Application Release Automation (ARA)**: A practice of deploying software releases to various environments and their configurations with as little human interaction as possible.

1. **Continuous delivery**: A software delivery process wherein updates are planned, implemented and released to end-users on a steady, constant basis
2. **Continuous integration**: A process that allows software changes to be tested and integrated into a code base on a continuous basis each time a change is made to code.
3. **Continuous Deployment**: A software development practice in which every code change goes through the entire pipeline and is put into production automatically, resulting in many production deployments without any human intervention at all.

1. **Immutable infrastructure**: An application service or hosting environment that, once set up, cannot be changed. It makes environments more robust and reliable because inadvertent changes are impossible to introduce.
2. **Microservices**: A type of application architecture in which applications are broken into multiple small pieces. For example, a microservices-based Web server might have its storage, front-end and security layers each operating as a separate service.
3. **Serverless computing**: A type of service that provides access to computing resources on demand, without requiring users to configure or manage an entire server environment.

Self Study Terminologies

1. **Configuration Management**
2. **Containers**
3. **Containerization**
4. **Continuous Testing**
5. **Integration Testing**
6. **Self-Service Deployment**
7. **Test Automation**
8. **Software Craftsmanship**
9. **Service Oriented Architecture**
10. **System Provisioning**