Copyright Notice

Agile and DevOps Practices Introduction

Pre requisites for Participants

- Working on Java application development OR
- Working on Testing(QA) OR
- Participants from build/release management OR
- Project Manager/Architect OR
- Operation team

Skills and Objectives

- Training Objectives
- Work experience
- Current work
- Skill sets
- Agile knowledge
- Knowledge about tools
- Implementation platforms

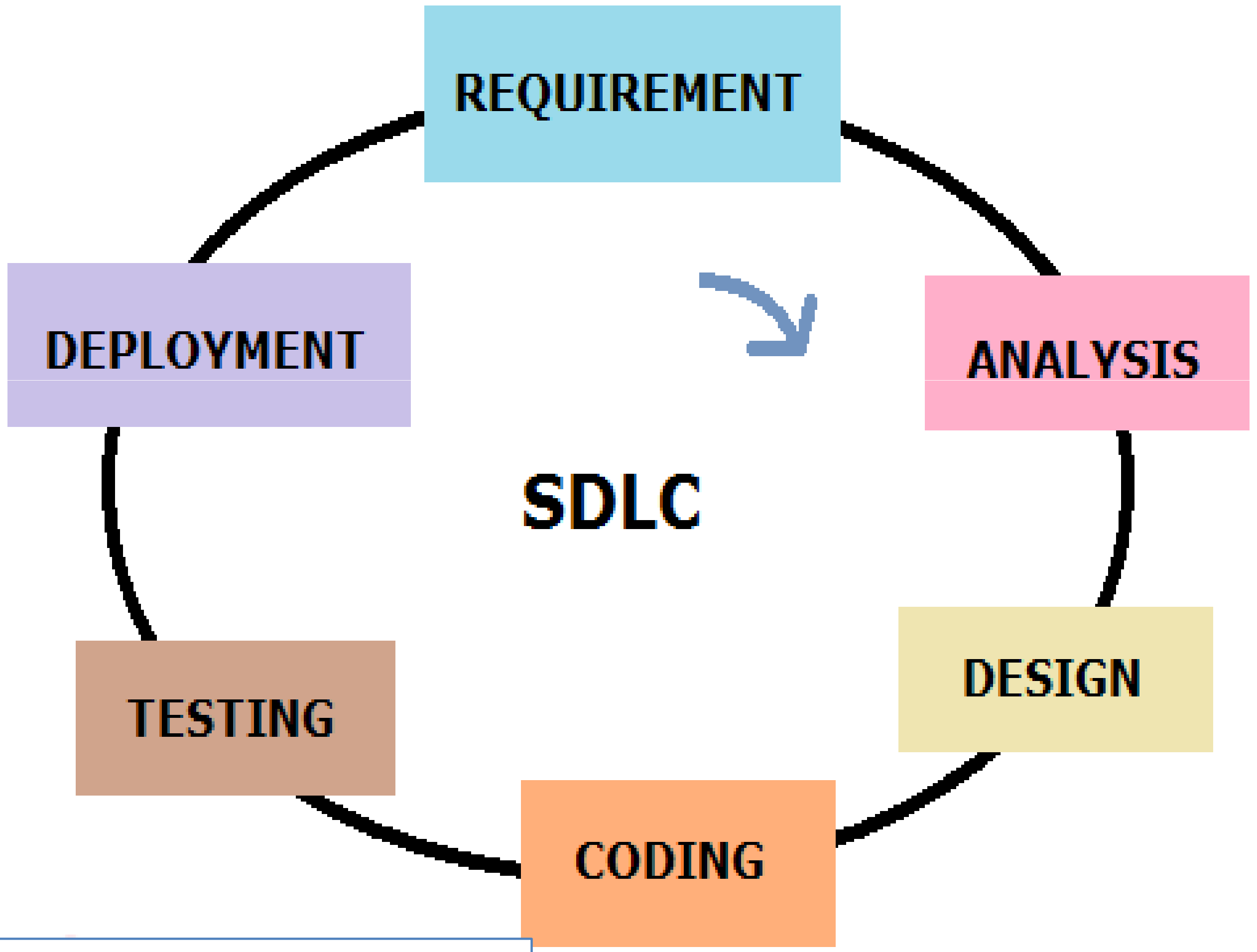
Focus

- This course is focused on Introduction to Agile Practices and Processes in DevOps and following the DevOps technical practices with tools.
- The objective is that the participants understand the Agile philosophy, DevOps culture, importance of collaborative work and become more groomed in their work to apply the Agile practices with tools in DevOps processes.

Set up

- Windows 7/10 or Linux 64 bit
- JDK 1.8
- Apache Tomcat
- GIT
- Jenkins CI server
- Eclipse
- Internet connection

SDLC



software development life cycle

SDLC Phases

- The **software development life cycle** (**SDLC**) is a specification of tasks to be performed at each step in the software development process.
- **It** is a structure followed by a development team within the software organization.
- It consists of a detailed plan describing how to develop, maintain and replace specific software.

Tasks in SDLC

- Requirement analysis
- Implementation
- Documentation
- Testing
- Deployment and testing
- Maintaining the software

Software development models

- Waterfall Model: This model involves finishing each phase completely before commencing the next one.
- V-Shaped Model: Similar to the waterfall model but with more importance placed on testing. Testing procedures are written even before the commencement of writing code. In this model, the system plan is generated before starting the development phase.

Incremental Model

- This life cycle model involves multiple development cycles.
- The cycles are divided up into smaller iterations.
- These iterations go through a set of phases including requirements, design, implementation and testing for every iteration.
- A working version of the software is produced during the first iteration, so working software is created early in the development process

Waterfall Model

This model is a sequential software development process model with defined phases as

1. Conception
2. Initiation
3. Analysis
4. Design
5. Construction
6. Testing
7. Production/Implémentation
8. Maintenance

Waterfall model pros and cons

- Good for systematic way of implementation where each thing is fixed upfront.
- The model enforces moving to the next phase only after completion of the previous phase.
- Difficult to manage the change in software requirement once the development has moved to next stages.
- The software delivery schedule is longer and mostly delayed further due to unaccounted delays in stages.
- Fault detection and rectification are long delayed cumbersome processes.

Agile Process of Development

- **Agile** means to be able to move quickly and easily.
- **Agile** software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of team members.

Agile Principles

- Agile refers to a set of values and principles set out in the **Agile Manifesto.**
- The Agile Manifesto basically underlines the following aspects:
- Interaction of individuals on tools and processes
- Collaboration with potential customers in the negotiation of project results
- Respond to change with a plan

Agile Methodology

- **Agile methodology** is a type of project management process, mainly used for software development.
- Where requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers.
- Agile methodology suggests smaller teams with incremental delivery of software.

Scrum Methodology

- Agile framework of specifications and followed to implement in the Agile software development process.
- Scrum comprises of five values: commitment, courage, focus, openness, and respect. Its goal is to develop, deliver, and sustain complex products through collaboration, accountability, and iterative progress.

Kanban

- A highly visual approach popularly used within Agile project management.
- It paints a picture of the workflow process, with an aim to identify any bottlenecks early on in the process, so that a higher quality product or service is delivered.
- Consists of practices such as Visualization, work definition,Flow,explicit policies, feedback loops and Collaborative evolution.

Agile Practices

- **Extreme Programming (XP)**: Suggests
- Small Release cycles
- Iterative development
- Small teams
- Simple Design
- Testing
- Refactoring
- Pair Programming

Lean in software development

- Lean software development is a translation of **lean** manufacturing principles and practices to the software development domain.
- Adapted from the Toyota Production System.
- **Lean** method concerns the optimization of a production process while agile is focused on the optimization of a development process

Lean Principles..

- Production should be based on demand and not on supply. It is simply about doing something when someone wants it and orders it, Instead of doing it first, hoping then that someone will need it.
- The production is more efficient if performed in small lots in order to exploit economies of scale.
- Taking the time to focus on quality also increases production and efficiency.
- Employers, not managers, are responsible for defining their method of working.
- Instead of than executing predefined tasks over and over again, workers must continually improve their way of working (the so-called “Kaizen”).

SAFe Agile

- The Scaled Agile Framework (SAFe) is a set of organization and workflow patterns intended to guide enterprises in scaling lean and agile practices.
- Along with large-scale Scrum (LeSS), disciplined agile delivery (DAD), and Nexus, SAFe is one of a growing number of frameworks that seek to address the problems encountered when scaling beyond a single team.

SAFe advantage

- SAFe is developed by and for practitioners, by leveraging three primary bodies of knowledge: agile software development, lean product development, and systems thinking.
- SAFe promotes alignment, collaboration, and delivery across large numbers of agile teams.

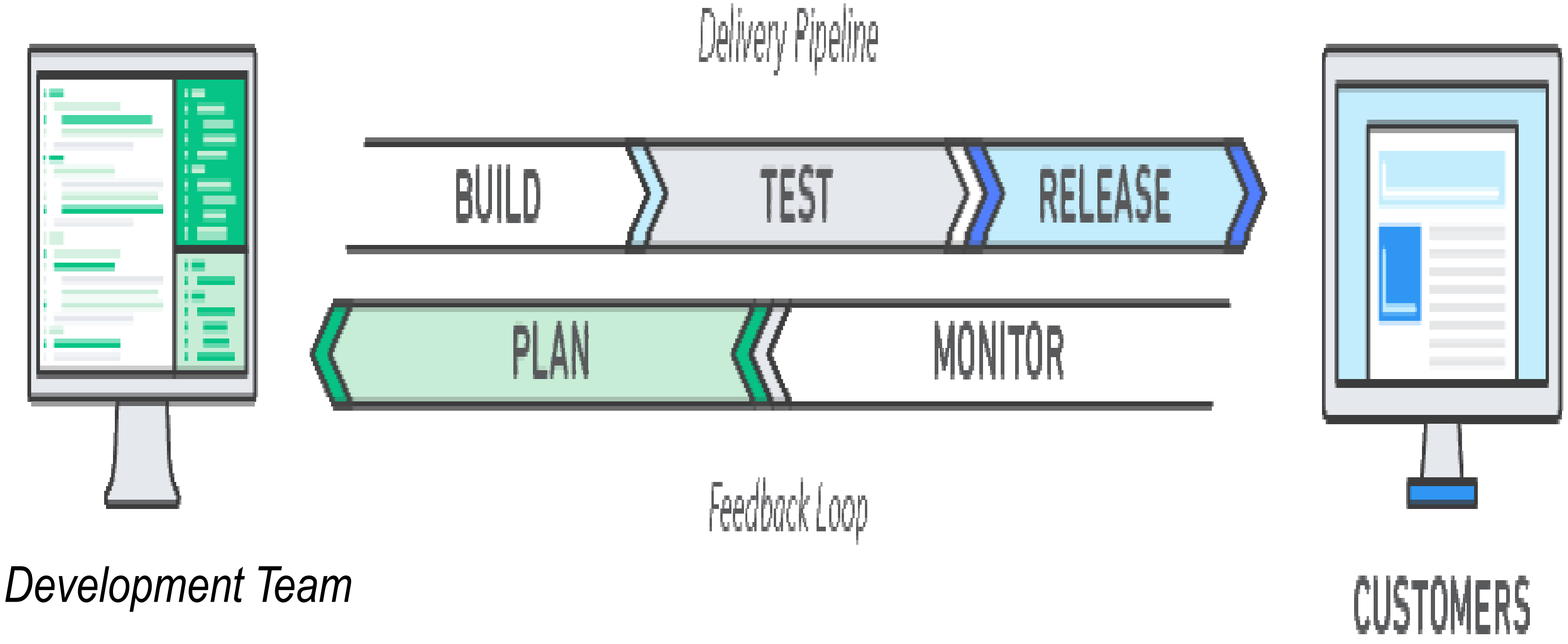
Lean-Agile

- The Lean-Agile Mindset is the combination of beliefs, assumptions, and actions of SAFe leaders and practitioners who embrace the concepts of the Agile Manifesto and Lean thinking.
- It’s the personal, intellectual, and leadership foundation for adopting and applying SAFe principles and practices.

Best Practices or SDLC

- **Customer collaboration**
- **Smaller teams**
- **User Stories**
- **Test-driven development (TDD)**
- **Automated tests**
- **Pair programming**
- **Continuous Integration**
- **Acceptance testing with test automation**
- **Burndown charts**

Application Delivery and Feedback



Why DevOps ?

- Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking.
- Software no longer merely supports a business; rather it becomes an **integral component of every part of a business**.
- Companies interact with their customers through software delivered as *online services or applications* and on all sorts of devices.
- Businesses use software applications to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations.
- DevOps needed to make the software development and delivery and management as collaborative effort and make it as quick , easy and responsive work.

What is DevOps

- DevOps means ***collaboration*** with **Development and Operations.**
- It is the collaboration between developers and other parties involved in building, deploying, operating, and maintaining software systems
- It involves cultural philosophies, practices, and tools that increases an organization's *ability* to deliver applications and services at high velocity
- DevOps enables the ***evolving and improving*** products at a faster pace.
- This ***speed*** in implementation enables organizations to better serve their customers and compete more effectively in the market.

About DevOps

- It is about defining a flow from development through full scale operation of a system.
- It is about the feedback to earlier stages of a development workflow.
- It focuses on automating processes required to release and change software.
- DevOps focus is on whole software development and delivery process and post delivery operations as well.

How DevOps works?

- With DevOps model, development and operations teams are no longer “siloed.”(isolated).
- The teams are formed as smaller single units to make collaboration quick and easy.
- These teams work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

DevOps Goals

- DevOps is about systems thinking, process definition, team coordination, and feedback loops
- DevOps is about continuous improvement for
  - To improve quality and reduce the number of incidents in production
  - Improves the overall efficiency in the process as the team learns best practices.

DevOps Goals..

- Reduce testing time
- Reduce the amount of changes required due to incomplete or conflicting requirements
- Design the simplest solution required to meet the requirements
- Leverage standardized integrated software tools
- Create robust production systems which lower maintenance costs
- Produce software to the highest quality
- Improve the performance of individuals and teams

Benefits of DevOps

- **The collaboration increases the speed of delivery and corrections**
- People innovate fasters and in more better manners for customers.
- **Adaption** to changing markets become more better and become more efficient at driving business results.
- The DevOps model **enables** the developers and operations teams to achieve these results.

Improved Collaboration

- Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability.
- Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows.
- This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

Process Automation

- Manual work involves patience and tend to increase errors and reduce efficiencies.
- The repetitive tasks can be automated with tools to make the process less error prone and increase the efficiency.

Rapid Delivery

- Increase the **frequency** and pace of releases so you can innovate and improve your product faster.
- The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage.
- The Continuous integration and continuous delivery practices automate the software release process, from build to deployments.

Reliability with Automation

- Ensures the ***quality*** of application updates and infrastructure changes so application can be delivered with more reliability.
- The practices like continuous integration and continuous delivery make the testing of each change as functional and safe.
- Monitoring and logging practices helps to stay informed of performance in real-time.

Scale it Up with automation

- Operate and manage the infrastructure and development processes at scale. e.g. for more no of users and more volumes of data .
- Automation and consistency helps to manage complex or changing systems efficiently and with reduced risk.
- For example, infrastructure as code helps to manage the development, testing, and production environments in a repeatable and more efficient manner.

Security

- Move quickly while retaining control and preserving compliances.
- Adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques.
- For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

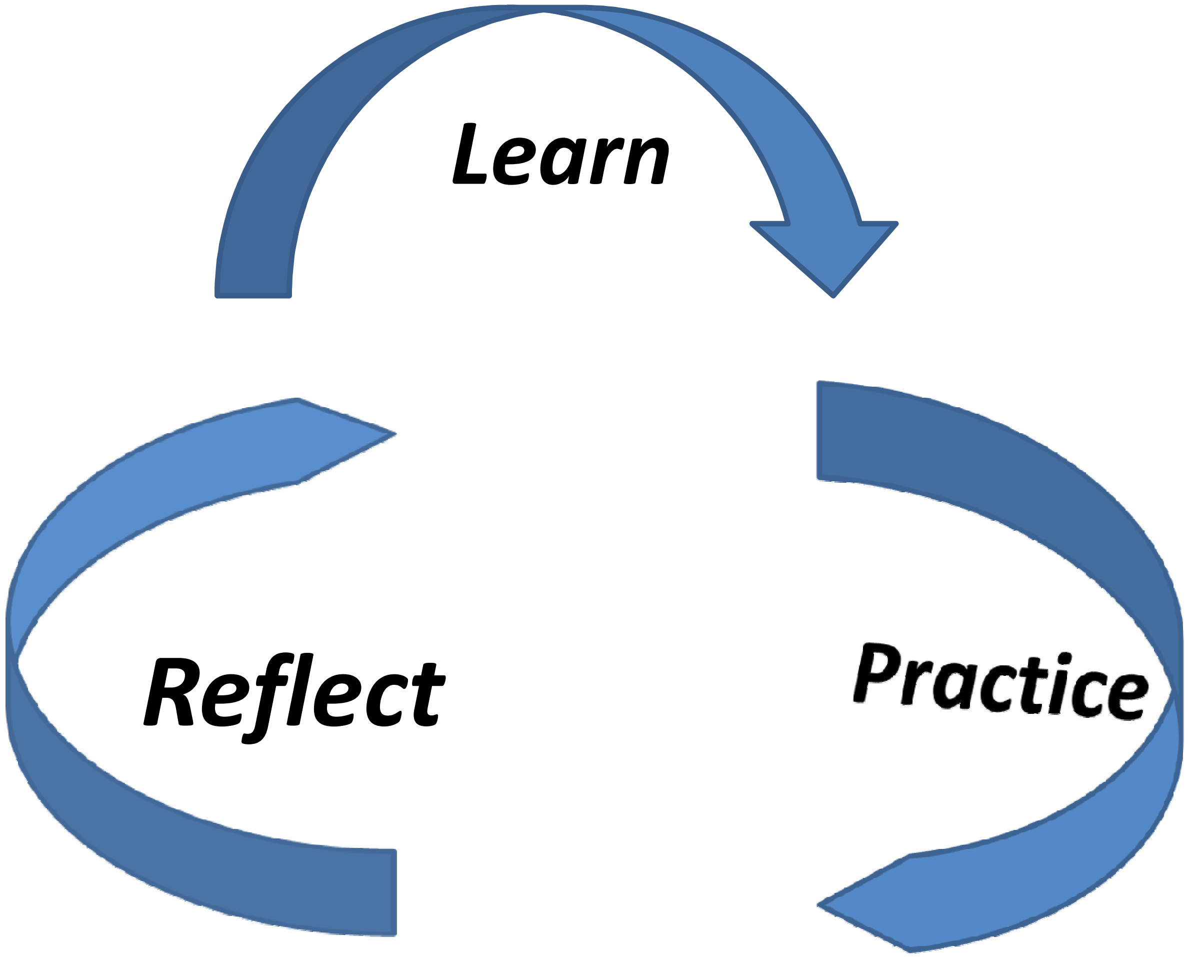
DevOps Principles

- Strive for clarity –challenge uncertainties in requirements
- Challenge the validity of assumptions
- Seek simplicity and reliability; do not strive to produce the optimal design
- Seek the least risky design that meets requirements
- Avoid Gold Plating –do not add extra code or features beyond the bare minimum
- Fix the root cause of each issue instead of patching to fix the symptoms

DevOps culture

- DevOps culture needs to be built and maintained.
- Adapt to change by continually changing.
- The organization's culture needs to evolve with it.
- People are resistant to change, should be able to adapt to change.
- **People need to work together and not in isolation**

Continuous Learning and Improvement



Technical Challenges

- Technology gets constantly evolving,
- Working environment is evolving
- External influences affect the tools selections : economic, legal, and social, licensing.
- To have a standardized set of tools across the organization
- Different users or teams using different tools leads to major issues, should avoid it.

Organization Practices

- Ensure Licensing and compliance
- Adapt to versioning incompatibilities
- Ensure smooth support for the tools and people.
- Provide smooth support and maintenance to customers.
- Ensure and maintain the DevOps culture throughout the organization.

DevOps Tool Chain

- Code development tools
- Source code repositories
- Database systems
- Artifact building tools
- Build and release management
- Testing tools
- Integration tools
- Security and identity management
- Configuration management tools
- Packaging and deployment tools
- Bug tracking and issue management tools
- Monitoring tools
- Documentation and communication tools
- Project tracking tools