**DevOps Methodology:** - DevOps is a software development methodology that stress communication, collaboration and interaction between software developers and IT professionals. It is a response to the interdependence of software developers and IT professionals.

1. Collaborative working of a Dev, Test, Infra, Ops together with a single unified goal.
2. Driven by automation.
3. Maturity is measured by speed, frequency, stability of release.

**Key reasons for DevOps Adoption:** -

1. Early time to market.
2. Improved quality reliability, stability while developing rapidly.
3. Regression efforts, Deployment efforts, Reduction in build time/incidents.

|  |  |
| --- | --- |
| **DevOps Features** | **DevOps Benifits** |
| Perform software development, testing and deployment in automated and synchronized way | Early time to market |
| Perform deployments with greater stability and quality | Stable application |
| Ensure effective collaboration between Dev and Ops team | Maintain proper SLA |

**Operations**

**Deployment**

**Release**

**Test**

**Build**

**Planning**

**Agile CI CD**

**DevOps**

**CI (Continues Integration):** - It is a software development method adopted from Extreme Programming. It helps in automation build process, frequent integrations, code quality, unit testing without manual interventions by using various open source/licensed tools.

**CD (Continues Deployments):** - CD allows constant flow of changes into production through an automated pipeline called CD Pipeline. This involves CV followed by CD. It is a series of practices to ensure that quality code can be deployed fast and safe to production. Since automation is used confidence is high in production. This is called CD.

**Alignment of Capabilities:** -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Business** | **Developers** | **Test** | **Infra** | **Ops** |
| Acceptance Testing | Version Control | Functional Testing | DB Deployments | Incident Management |
| Agile | Static/Dynamic Code checked | Defect Management | CD Automation |  |
|  | Code Quality | Performance Testing | Released Management |
| Code Coverage | Security Testing |  |
| Unit Testing |  |
| Build Automation |
| CI |

**CI Pipeline:** -

**Create Executable**

**Code Quality Analysis**

**Version Source Control**

**Code Coverage**

**Unit Testing**

V

**Deploy Executable**

1. Reduce risk of integration defects.
2. Help detect bugs and remove them faster.
3. Less integration due to automation.
4. Helps in frequent deployment.

**Open Tools:** -

Application Life Cycle Management -> JIRA

Version Source Control - > GIT

Static Code Analysis -> SonarQube/FindBugs

Dynamic Code Analysis -> JProbe

Code Coverage -> JaCoCo **CI Tools**

Unit Testing -> Junit

Build Automation -> Maven

Repository -> Nexus

CI -> Jenkins/Bamboo

Functional Testing -> Selenium/Cucumber

Defect Management -> Bugzilla

Performance Testing -> Apache JMeter **CD Tools**

Security Testing -> AppScan

CD Automation -> GO

Release Management -> XL Release

1. Open source tools are free, powerful.
2. Quick availability of these tools.
3. No license overhead.

**CI-CD Activity:** -

6

2

DEV 2

**CI**

**CD**

8

10

9

7

5

Binary Code

**Binary Code Repository**

3

4

1

**Source Code Repository**

**Source Code Commits**

DEV 1

DEV 3

DEV

1. Code Quality Analysis 5) Environment for Testing
2. Unit Testing 6) Testing
3. Code Coverage 7) Deployment to Pre-Preproduction
4. Build 8) Release Management

10) Deployment to Production

**CI Pipeline CD Pipeline**

**Version Source Control: -** A distributed version control system would be useful if teams are distributed. A proper branching needs to be done so that CI can be aligned to proper branch. GIT is a popular tool due to its distributed nature.

**Static and Dynamic Code Analysis:** - Some quality rules are defined by static code analyzer on code and test cases to ensure that code meets quality. These rules are configurable. We can add, edit, delete those rules. In Dynamic code analyzer it executes code and analyze for performance, memory utilization etc.

**Repository:** -

1. Right version of build is used and completely tested version goes for release.
2. It structures the deployment properly.

Release management tools help in planning, executing and tracking releases.

Finally, to provide required infra for automated pipeline and tools to run smoothly the infra layer is very important. They provide suitable environment for application to run, to maintain etc. **Ex:** - VMWare, AWS, Microsoft Azure etc.

**Alignment of Teams:** -

**DevOps**

**Shared Product Backlog**

**Ops Team**

**Dev Team**

1. Teams have single product backlog with both DEV and OPS.
2. Each team member is capable of selecting any item and work on it.
3. Cross skilling is possible.

**Alignment of Process:** -

**Scrumban (Infosys View)**

1. Single Product Owner with one Product Backlog.
2. Team works on prioritized user stories.
3. Cross skilling is possible.