📘 UNIT–5: DevOps and Other Agile Methodologies

Subject: Project Management — RGPV Exam Notes

---

1. DevOps Overview

Definition

DevOps is a software development approach that combines Development and Operations teams to deliver software faster, better, and more reliably.

Easy Introduction

Earlier, developers made software and operations team deployed it separately. This caused delay, errors, and blame-game.

DevOps solves this by making both teams work together.

DevOps = Development + Operations

Why This Topic is Important

DevOps is important because modern companies need:

- fast software delivery
- fewer errors
- automation
- continuous testing
- continuous deployment
- quick feedback

⭐ Most Important

🔥 Repeated in PYQs

🎯 Highly Expected

Detailed Explanation

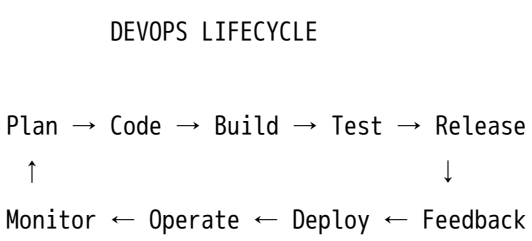
In DevOps, software development is not a slow one-time process. It is a continuous cycle.

Developers write code, testing is automated, code is integrated, application is deployed, and monitoring is done continuously.

DevOps Lifecycle

```
Plan
  ↓
Code
  ↓
Build
  ↓
Test
  ↓
Release
  ↓
Deploy
  ↓
Operate
  ↓
Monitor
  ↓
Feedback
```

Diagram / Flowchart



Components of DevOps

Component	Simple Meaning
Source Code Management	Code ko safely store karna
Continuous Integration	Code ko frequently merge karna
Automated Testing	Tests automatically run karna
Continuous Deployment	Software automatically deploy karna
Configuration Management	Servers/settings manage karna
Monitoring	Application performance check karna
Containerization	App ko container me run karna

Example

Suppose a team is building a shopping app.

In DevOps:

- Developer writes payment code
- Code goes to GitHub
- Automated build runs
- Automated tests run
- App deploys to server
- Monitoring checks errors

Advantages

- Faster delivery

- Better collaboration
- Less manual work
- Fewer deployment errors
- Better software quality
- Continuous feedback

Disadvantages

- Requires skilled team
- Initial setup is difficult
- Tools knowledge required
- Cultural change needed

Applications

- Software companies
- Cloud applications
- Banking apps
- E-commerce websites
- Mobile applications

Important Keywords for Exam

Development, Operations, Automation, CI/CD, Monitoring, Collaboration, Continuous Delivery

Conclusion

DevOps improves software delivery by combining development, testing, deployment, and monitoring into one continuous process.

---

2. Containerization Using Docker

Definition

Containerization is a technique of packaging an application with all its required files, libraries, and dependencies so that it can run in any environment.

Easy Introduction

Sometimes software works on one computer but not on another. This happens because of different settings, versions, or missing files.

Docker solves this problem by packing the application and its environment together.

Why This Topic is Important

Docker is important because it makes application deployment:

- fast
- reliable
- portable
- consistent

Detailed Explanation

Docker creates a container.

A container includes:

- application code
- libraries
- dependencies
- runtime environment
- configuration files

So the application runs same everywhere.

Docker Components

Component	Meaning
Docker Image	Blueprint/template of application
Docker Container	Running instance of image
Dockerfile	File containing build instructions
Docker Hub	Online repository for images
Docker Engine	Tool that runs containers

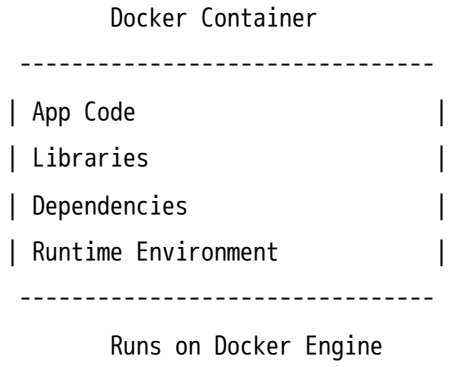
Working / Steps

1. Developer writes application code.
2. Dockerfile is created.
3. Docker image is built.
4. Docker container is started.
5. Application runs inside container.
6. Same container can run on any system.

Flowchart

```
Code
  ↓
Dockerfile
  ↓
Docker Image
  ↓
Docker Container
  ↓
Running Application
```

Diagram



Real-Life Analogy

Docker container is like a lunch box.

Lunch box contains complete food items. You can carry it anywhere and eat without depending on outside things.

Similarly, Docker container carries complete application environment.

Advantages

- Same app runs everywhere
- Fast deployment
- Less environment issues
- Lightweight than virtual machines
- Easy scaling

Disadvantages

- Security issues if poorly managed
- Requires Docker knowledge
- Persistent storage needs care

Applications

- Web app deployment

- Microservices
- Cloud computing
- DevOps pipelines
- Testing environments

Important Keywords

Container, Docker Image, Dockerfile, Docker Hub, Portability, Lightweight Deployment

Conclusion

Docker containerization makes software portable, consistent, and easy to deploy across different environments.

---

3. Managing Source Code and Automating Builds

Definition

Source Code Management is the process of storing, tracking, and managing changes in program code using tools like Git.

Automated Build means automatically converting source code into executable software using build tools.

Easy Introduction

In software projects, many developers work together. If everyone changes code separately, confusion happens.

Git helps manage code changes. Build automation helps convert code into working application automatically.

Why It Is Needed

- to manage code versions
- to avoid code loss
- to track developer changes
- to build software automatically
- to reduce manual errors

Detailed Explanation

Source Code Management

It helps in:

- version control
- collaboration
- rollback
- branch management
- code history tracking

Automated Build

It includes:

- compiling code
- downloading dependencies
- packaging software
- creating executable files

Tools

Work	Tools
Source Code Management	Git, GitHub, GitLab
Build Automation	Maven, Gradle, Jenkins, npm

Working / Steps

1. Developer writes code.
2. Code is committed to Git.
3. Code is pushed to repository.
4. Build tool automatically starts.
5. Application is compiled.
6. Build result is generated.

Flowchart

```
Write Code
    ↓
Commit Code
    ↓
Push to Git
    ↓
Automated Build
    ↓
Build Result
```

Example

A Java project uses GitHub and Maven.

When code is pushed to GitHub, Jenkins starts build and creates `.jar` file automatically.

Advantages

- Code tracking
- Team collaboration
- Less manual work
- Faster build process

- Easy rollback

Disadvantages

- Merge conflicts possible
- Requires tool knowledge
- Build setup may be complex

Applications

- Software teams
- DevOps pipelines
- Open-source projects
- Enterprise applications

Important Keywords

Git, Version Control, Repository, Commit, Build Automation, Continuous Integration

Conclusion

Source code management and automated builds help teams work together safely and deliver software faster.

---

4. Automated Testing and Test Driven Development

Definition

Automated Testing is the process of using tools/scripts to test software automatically.

Test Driven Development is a development approach in which tests are written before actual code.

Easy Introduction

Manual testing takes time and may miss errors. Automated testing runs tests quickly and repeatedly.

TDD means:

Write Test First → Write Code → Pass Test → Improve Code

Why It Is Needed

- to find bugs early
- to improve quality
- to reduce manual testing effort
- to support continuous integration
- to avoid repeated mistakes

TDD Working

```
Write Test
   ↓
Run Test
   ↓
Test Fails
   ↓
Write Code
   ↓
Test Passes
   ↓
Refactor Code
```

Diagram

```
        TDD Cycle

Red → Green → Refactor

Red = Test fails
Green = Test passes
Refactor = Improve code
```

Types of Automated Testing

Type	Meaning
Unit Testing	Small code part testing
Integration Testing	Multiple modules testing
Functional Testing	Feature testing
Regression Testing	Old features still working or not

Example

Before writing login code, developer writes test:

- correct password should login
- wrong password should show error

Then developer writes code to pass these tests.

Advantages

- Bugs found early
- Better software quality
- Saves time

- Supports CI/CD
- Reliable testing

Disadvantages

- Initial time required
- Test writing skill needed
- Poor tests give wrong confidence

Applications

- Web applications
- Mobile apps
- Banking systems
- Large software projects

Important Keywords

Automated Testing, TDD, Unit Testing, Regression Testing, Red-Green-Refactor

Conclusion

Automated testing and TDD improve software quality by detecting errors early and making development more reliable.

---

5. Continuous Integration

Definition

Continuous Integration is a DevOps practice where developers frequently merge code into a shared repository and automated builds/tests are run.

Easy Introduction

In team projects, developers work on different features. If they merge code after many days, errors increase.

CI solves this by merging code frequently.

Why It Is Needed

- to detect bugs early
- to avoid integration problems
- to improve teamwork
- to ensure code works continuously

Detailed Explanation

In CI:

1. Developer writes code.
2. Code is pushed to Git.
3. Build starts automatically.
4. Tests run automatically.
5. Errors are reported quickly.

Flowchart

```
Developer Code
      ↓
Push to Repository
      ↓
Automatic Build
      ↓
Automatic Test
      ↓
Feedback to Developer
```

Example

If 5 developers are working on an app, each pushes code daily. CI system checks whether all code works together.

Advantages

- Early bug detection
- Better code quality
- Faster development
- Less integration risk

Disadvantages

- Requires automation setup
- Failed builds need quick fixing
- Test suite maintenance required

Applications

- Agile teams
- DevOps pipelines
- Software companies
- Cloud-based development

Important Keywords

Frequent Integration, Automated Build, Automated Test, Shared Repository, CI Pipeline

Conclusion

Continuous Integration helps teams merge and test code frequently, reducing bugs and integration problems.

---

6. Configuration Management

Definition

Configuration Management is the process of managing and maintaining software, servers, tools, and environment settings in a controlled way.

Easy Introduction

Software needs correct configuration to run:

- database setting
- server setting
- API keys
- environment variables
- software versions

Configuration management controls these settings.

Why It Is Needed

- to maintain consistency
- to avoid configuration errors
- to manage multiple servers
- to support automation

Detailed Explanation

Configuration management ensures that all environments like development, testing, and production are properly configured.

Tools

Tool	Use
Ansible	Server automation
Puppet	Configuration control
Chef	Infrastructure management
Terraform	Infrastructure creation

Working / Steps

1. Define required configuration.
2. Store configuration as code.
3. Apply configuration automatically.
4. Monitor changes.
5. Correct mismatch.

Flowchart

```
Define Config
      ↓
Store as Code
      ↓
Apply Automatically
      ↓
Monitor
      ↓
Maintain Consistency
```

Example

If a company has 50 servers, manually installing software on each server is difficult. Ansible can configure all servers automatically.

Advantages

- Consistent environment
- Less manual error
- Faster setup
- Easy server management

Disadvantages

- Tools are complex
- Wrong configuration can affect many systems
- Requires skilled team

Applications

- Cloud infrastructure
- DevOps automation
- Server management
- Enterprise applications

Important Keywords

Configuration, Infrastructure as Code, Automation, Environment Consistency, Ansible, Puppet, Chef

Conclusion

Configuration management ensures that software and infrastructure remain consistent, controlled, and reliable.

---

7. Continuous Deployment

Definition

Continuous Deployment is a DevOps practice in which every successfully tested code change is automatically deployed to production.

Easy Introduction

Earlier deployment was manual and slow. Continuous Deployment makes deployment automatic.

If code passes all tests, it is directly released to users.

Why It Is Needed

- faster release
- less manual deployment
- quick customer feedback
- reduced deployment delay

Continuous Deployment Flow

```
Code Commit
     ↓
Build
     ↓
Automated Testing
     ↓
Deployment
     ↓
User Feedback
```

Difference: CI vs Continuous Deployment

Basis	Continuous Integration	Continuous Deployment
Meaning	Code merge and test	Automatic release
Focus	Code quality	Fast delivery
Stage	Before deployment	After testing
Output	Build/test result	Live software

Example

A bug fix in shopping website is pushed to GitHub. Automated tests pass, and fix is deployed to live website automatically.

Advantages

- Very fast release
- Less manual work
- Quick feedback
- Faster bug fixing

Disadvantages

- Strong testing required
- Risk of faulty release
- Monitoring must be strong

Applications

- Web applications
- SaaS products
- E-commerce websites
- Cloud services

Important Keywords

Automatic Deployment, Production Release, CI/CD, Release Pipeline, Fast Delivery

Conclusion

Continuous Deployment helps companies release software automatically and quickly after successful testing.

---

8. Automated Monitoring

Definition

Automated Monitoring is the process of continuously tracking application, server, and system performance using monitoring tools.

Easy Introduction

After deployment, work is not finished. Application must be monitored continuously.

Monitoring checks:

- server up or down
- errors
- response time
- traffic
- CPU/memory usage

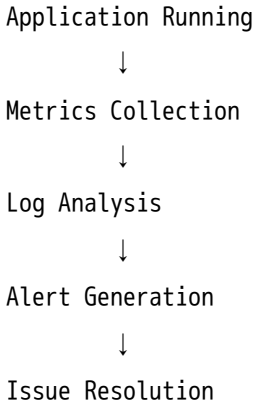
Why It Is Needed

- to detect problems quickly
- to improve performance
- to reduce downtime
- to support customer satisfaction

Working / Steps

1. Monitoring tool is installed.
2. Metrics are collected.
3. Logs are analyzed.
4. Alerts are generated.
5. Team fixes issues.

Flowchart



Tools

Tool	Use
Prometheus	Metrics monitoring
Grafana	Dashboard visualization
ELK Stack	Log analysis
Nagios	System monitoring

Example

If website server goes down, monitoring tool sends alert to DevOps team immediately.

Advantages

- Quick issue detection
- Reduced downtime
- Better performance
- Improved reliability

Disadvantages

- Setup required
- Too many alerts can confuse team
- Tools need maintenance

Applications

- Cloud apps
- Banking systems
- E-commerce platforms
- Large websites

Important Keywords

Metrics, Logs, Alerts, Dashboard, Uptime, Performance Monitoring

Conclusion

Automated monitoring ensures that software remains reliable, fast, and available for users.

---

9. Extreme Programming (XP)

Definition

Extreme Programming is an Agile methodology focused on high-quality software development through continuous testing, customer feedback, and frequent releases.

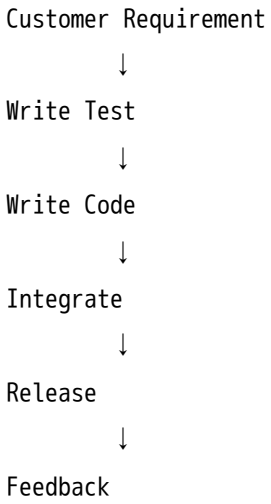
Easy Introduction

XP is used when requirements change frequently. It focuses heavily on coding quality and testing.

Important XP Practices

Practice	Meaning
Pair Programming	Two programmers work together
Test Driven Development	Tests written before code
Continuous Integration	Frequent code merging
Small Releases	Frequent software releases
Customer Involvement	Customer gives regular feedback

Flowchart



Example

In a banking app, developers continuously test every small change to avoid errors.

Advantages

- High software quality
- Fast feedback
- Less bugs
- Strong teamwork

Disadvantages

- Requires skilled team
- Customer availability required
- Pair programming may feel costly

Applications

- Software projects
- High-change projects
- Quality-sensitive systems

Important Keywords

Pair Programming, TDD, Continuous Integration, Small Releases, Customer Feedback

Conclusion

XP is an Agile methodology that focuses on quality, testing, and customer involvement.

---

10. Feature Driven Development (FDD)

Definition

Feature Driven Development is an Agile methodology where software is developed feature by feature.

Easy Introduction

FDD focuses on building small useful features one by one.

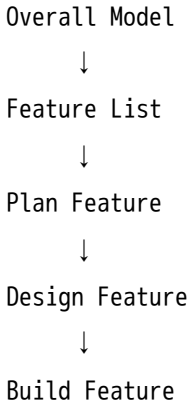
Example:

- login feature
- payment feature
- search feature
- notification feature

Steps of FDD

1. Develop overall model.
2. Build feature list.
3. Plan by feature.
4. Design by feature.
5. Build by feature.

Flowchart



Advantages

- Easy progress tracking
- Feature-wise delivery
- Suitable for large projects
- Customer sees clear output

Disadvantages

- Less suitable for very small projects
- Requires clear feature list
- Documentation needed

Applications

- Business applications
- Large software systems
- Enterprise projects

Important Keywords

Feature List, Feature-wise Development, Design by Feature, Build by Feature

Conclusion

FDD is useful when project can be divided into clear and small features.

---

11. DSDM

Definition

DSDM stands for Dynamic Systems Development Method. It is an Agile methodology focused on fixed time, fixed cost, and frequent delivery.

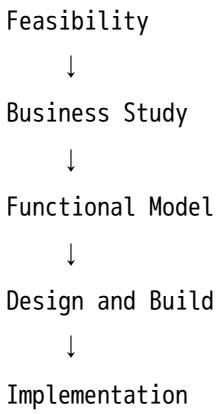
Easy Introduction

DSDM says project should deliver business value quickly while controlling time and cost.

Main Principles

- active user involvement
- frequent delivery
- business needs focus
- quality control
- iterative development

Flowchart



Advantages

- Strong business focus
- Good time control
- User involvement
- Quality delivery

Disadvantages

- Requires trained team
- Not suitable for small simple projects
- User involvement is compulsory

Applications

- Business systems
- Enterprise projects
- Time-sensitive projects

Important Keywords

Dynamic System, Business Focus, Time-boxing, Iterative Development, User Involvement

Conclusion

DSDM is an Agile method suitable for business projects where time and cost control are important.

---

12. Crystal Methodology

Definition

Crystal is a family of Agile methodologies that focuses on people, communication, teamwork, and project size.

Easy Introduction

Crystal believes that people and communication are more important than heavy processes.

Features

- people-focused

- flexible process
- communication-based
- suitable for different project sizes

Types

Type	Use
Crystal Clear	Small teams
Crystal Yellow	Medium teams
Crystal Orange	Large teams

Flowchart

```
Team Size
   ↓
Project Criticality
   ↓
Choose Crystal Method
   ↓
Work with Communication
```

Advantages

- Flexible
- People-focused
- Simple process
- Good communication

Disadvantages

- Less documentation
- Depends heavily on team skill

- Not suitable for highly regulated projects

Applications

- Small software teams
- Flexible projects
- Communication-heavy projects

Important Keywords

People, Communication, Teamwork, Lightweight Method, Flexibility

Conclusion

Crystal methodology focuses on team communication and flexibility rather than strict processes.

🔥 MOST IMPORTANT TOPICS

1. DevOps Overview ⭐⭐⭐⭐⭐
2. Docker Containerization ⭐⭐⭐⭐⭐
3. Continuous Integration ⭐⭐⭐⭐⭐
4. Continuous Deployment ⭐⭐⭐⭐
5. Automated Testing & TDD ⭐⭐⭐⭐
6. XP, FDD, DSDM, Crystal ⭐⭐⭐⭐
7. Configuration Management ⭐⭐⭐
8. Automated Monitoring ⭐⭐⭐

⭐ MOST IMPORTANT 7-MARK QUESTIONS

1. Explain DevOps and its components.
2. Explain Docker containerization.
3. Explain Continuous Integration.
4. Explain Continuous Deployment.
5. Explain Automated Testing and TDD.
6. Explain Configuration Management.
7. Explain Automated Monitoring.
8. Explain Extreme Programming.
9. Explain FDD, DSDM and Crystal.

⭐ MOST IMPORTANT 14-MARK QUESTIONS

1. Explain DevOps lifecycle with components and diagram.
2. Explain Docker containerization and its role in DevOps.
3. Explain CI/CD pipeline with automated testing and deployment.
4. Explain automated testing, TDD and continuous integration.
5. Explain XP, FDD, DSDM and Crystal Agile methodologies.

🎯 PYQ-BASED EXPECTED QUESTIONS

Very High Probability

- DevOps overview and components
- Docker containerization
- Continuous Integration
- CI/CD pipeline
- XP methodology

High Probability

- Automated Testing and TDD
- Continuous Deployment
- Configuration Management
- FDD
- DSDM

Medium Probability

- Automated Monitoring
- Crystal methodology
- Source code management
- Build automation

⚡ ONE-NIGHT REVISION NOTES

Topic	One-Line Revision
DevOps	Dev + Ops collaboration
Docker	App with dependencies in container
Git	Source code management
Automated Build	Code to executable automatically
TDD	Test first, code later
CI	Frequent code merge + test
CD	Automatic deployment
Monitoring	Track performance/errors
XP	Testing + pair programming
FDD	Feature-wise development
DSDM	Time-boxed business Agile
Crystal	People and communication focused

📌 SMART STUDY PLAN

2-Hour Revision Strategy

Time	Topic
25 min	DevOps lifecycle
20 min	Docker
20 min	CI/CD
15 min	Automated Testing + TDD
15 min	Configuration + Monitoring
25 min	XP, FDD, DSDM, Crystal

5-Hour Preparation Strategy

Time	Topic
1 hour	DevOps overview + components
1 hour	Docker + Source Code + Build
1 hour	Testing + TDD + CI
1 hour	Deployment + Monitoring
1 hour	XP + FDD + DSDM + Crystal

One-Night Priority Order

1. DevOps Overview
2. Docker Containerization
3. Continuous Integration
4. Continuous Deployment
5. Automated Testing + TDD
6. XP
7. FDD
8. DSDM
9. Crystal
10. Configuration Management
11. Monitoring

🧠 MEMORY TRICKS

DevOps Lifecycle

PCBT RDOM

- P = Plan
- C = Code
- B = Build
- T = Test
- R = Release
- D = Deploy
- O = Operate
- M = Monitor

TDD

RGR

- R = Red
- G = Green
- R = Refactor

FDD

MFPDB

- M = Model
- F = Feature list
- P = Plan
- D = Design
- B = Build

Scrum/Agile Methods

XDFC

- X = XP
- D = DSDM
- F = FDD
- C = Crystal

🚀 TOPPER ANSWER WRITING TIPS

For 7 marks:

```
Definition
↓
Easy Explanation
↓
Diagram
↓
Advantages
↓
Applications
↓
Conclusion
```

For 14 marks:

```
Introduction
↓
Definition
↓
Detailed Explanation
↓
Lifecycle / Flowchart
↓
```

```
Tools / Components
↓
Example
↓
Advantages
↓
Disadvantages
↓
Applications
↓
Conclusion
```

Keywords to Underline

DevOps, CI/CD, Docker, Container, Automated Testing, TDD, Configuration Management, Continuous Deployment, Monitoring, XP, FDD, DSDM, Crystal

Final Tip

Agar time bahut kam hai, to sabse pehle:

DevOps lifecycle + Docker + CI/CD + XP/FDD/DSDM/Crystal

prepare karo. Ye Unit–5 ke sabse scoring topics hain.