

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**WORK-INTEGRATED LEARNING PROGRAMMES DIVISION**

**BITS-WIPRO Collaborative Program**

**M Tech Computing Systems and Infrastructure (WIMS)**

**Second Semester 2019-2020**

**COURSE HANDOUT**

**Part A**

|  |  |
| --- | --- |
| **Course Title** | Introduction to DevOps |
| **Course No(s)** | CSIW ZG515 |
| **Credit Units** |  |

**Course Description**

|  |
| --- |
| This course introduces the need for Devops, the evolution of Devops. It focuses on how Devops is influencing the software development lifecycle from the perspective of process, people and technology. It also focuses on version control, configuration management and automating them. This course also helps us to gain understanding between agile and Devops, how the cloud and DevOps work together to help businesses achieve their transformation. |

**Course Objectives**

|  |  |
| --- | --- |
| **No** | Objective |
| **CO1** | To learn the key ideas and techniques to bring development and operations together to produce higher-quality software and deliver it more quickly. |
| **CO2** | To learn the core principles, business and technical terms used in DevOps from perspective of business and IT teams |
| **CO3** | To gain knowledge of the Principles and practices of the DevOps Lifecycle including Continuous Integration, Continuous Inspection, Continuous delivery, Continuous deployment and Continuous monitoring. |
| **CO4** | To understand the usage of tools and technologies used for implementing DevOps. |

**Text Book(s)**

|  |  |
| --- | --- |
| No | Author(s), Title, Edition, Publishing House |
| T1 | DevOps: A Software Architect's Perspective (SEI Series in Software Engineering)  by Len Bass, Ingo Weber, Liming Zhu , Publisher: Addison Wesley (18 May 2015). |
| T2 | Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation by Jez Humble, David Farley.Publisher: Addison Wesley, 2011 |

**Reference Book(s) & other resources**

|  |  |
| --- | --- |
| No | Author(s), Title, Edition, Publishing House |
| R1 | Effective DevOps: Building A Culture of Collaboration, Affinity, and Tooling at Scale by Jennifer Davis ,Ryn Daniels.Publisher: O'Reilly Media, June 2016 |
| R2 | The DevOPS Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations byGene Kim, Patrick Debois, John Willis, Jez Humble, John Allspaw.Publisher: IT Revolution Press (October 6, 2016) |
| R3 | **Web Resources**:   1. <https://jenkins.io/> 2. <https://xebialabs.com/solutions/devops/> 3. <https://www.ibm.com/ibm/devops/us/en/casestudies/#all> 4. <https://git-scm.com/> 5. <https://hub.docker.com/> 6. <https://www.atlassian.com/git/tutorials/comparing-workflows> 7. <https://www.tutorialspoint.com/puppet/> 8. https://www.tutorialspoint.com/chef/ |

**Content Structure**

|  |  |  |
| --- | --- | --- |
| **No** | **Title of the Module** | **References** |
| M0 | **Module 0: Foundational Terminology and Concepts**   * + - Software development lifecycle     - The Waterfall approach     - Agile Methodology     - Operational Methodologies: ITIL     - Development, Testing, Release, and Deployment Concepts     - Provisioning, Version Control     - Test Driven Development, Feature Driven Development     - Behavior-driven development   *(This module is to set the stage uniformly for all participants and will be covered based on the set of individuals enrolling for the course)* | T2-Chapter 1  R1-chapter 4 |
| M1 | **Module 1: Why and What is DevOps?**   * + - Problems of Delivering Software     - Principles of Software Delivery     - Need for DevOps     - Evolution of DevOps     - DevOps Practices     - The Continuous DevOps LifeCycle Process (Continuous Integration, Continuous Inspection, Continuous Deployment, Continuous Delivery, Continuous Monitoring)     - DevOps Culture     - Case Study- (IBM/Facebook/NetFlix) | T1- Chapter 1  T2- Chapter 1  R1- Chapter 2,3 |
| M2 | **Module 2: DevOps Dimensions**   * + - Three dimensions of DevOps – People, Process, Technology/Tools     - DevOps- Process       * DevOps and Agile       * Agile methodology for DevOps Effectiveness       * Flow Vs Non-Flow based Agile processes       * Choosing the appropriate team structure: Feature Vs Component teams       * Enterprise Agile frameworks and their relevance to DevOps       * Behavior driven development, Feature driven Development       * Cloud as a catalyst for DevOps     - DevOps – People       * Team structure in a DevOps       * Transformation to Enterprise DevOps culture       * Building competencies, Full Stack Developers       * Self-organized teams, Intrinsic Motivation     - Technology in DevOps(Infrastructure as code, Delivery Pipeline, Release Management)     - Tools/technology as enablers for DevOps | T1- Chapter1, 2,3  T2 – Chapter 6  R1- Chapter 11  R2 – Chapter 1, 3 |
| M3 | **Module 3: Source Code Management (Using GIT as an example tool)**   * + - * Version control system and its types       * Introduction to GIT       * GIT Basics commands (Creating Repositories, clone, push, commit, review)       * Git workflows- Feature workflow, Master workflow, Centralized workflow       * Feature branching       * Managing Conflicts       * Tagging and Merging       * Best Practices- clean code | T2-Chapter 2,14  R3- 4,6 |
| M4 | **Module 4: Continuous build and code quality**   * Manage Dependencies * Automate the process of assembling software components with build tools * Use of Build Tools- Maven, Gradle * Unit testing   + - * Enable Fast Reliable Automated Testing       * Setting up Automated Test Suite – Selenium       * Continuous code inspection - Code quality       * Code quality analysis tools- sonarqube | T1- Chapter 5  T2- Chapter 4, 6, 13  R2-Chapter 3 |
| M5 | **Module 5: Continuous Integration and Continuous Delivery**   * + - * Implementing Continuous Integration-Version control, automated build, Test       * Prerequisites for Continuous Integration       * Continuous Integration Practices       * Team responsibilities       * Using Continuous Integration Software (Jenkins as an example tool)       * Jenkins Architecture       * Integrating Source code management, build, testing tools etc., with Jenkins - plugins       * Artefacts management       * Setting up the Continuous Integration pipeline       * Continuous delivery to staging environment or the pre-production environment       * Self-healing systems | T2- Chapter 3, 15  R2- Chapter 3  R3-1 |
| M6 | **Module 6: Continuous Deployment**   * Deployment pipeline * Human-free deployments   + - Implementing and Automating the deployment process     - Deploying it to testing environments     - Releasing software into production     - Environment-based release patterns     - Rolling Back Deployments and Zero-Downtime Releases     - Blue/Green Deployment     - Rolling Upgrade     - The canary release pattern- Dark Launches | T1- Chapter 6, 12  T2- Chapter 10  R2- Chapter 3, 4 |
| M7 | **Module 7: Continuous Monitoring**   * + - Need for continuous monitoring     - Goals of monitoring     - Challenges of monitoring under continuous change     - Alert management     - Analytics     - Continuous customer feedback and optimization     - Use of ELK (Elasticsearch, Logstash, and Kibana) Stack | T1- Chapter 7  R1- Chapter 11 |
| M8 | **Module 8: Configuration Management**   * + - Infrastructure as code     - Managing Infrastructure and Environments(Production, pre-production, Test, Developer Environment)     - Environment provisioning     - Automating and Managing Server Provisioning     - Configuration management tools- Chef, Puppet     - Managing on-demand infrastructure, Auto scaling | T2- Chapter 2, 11  R1- Chapter 14 |
| M9 | **Module 9: Virtualization and Containerization**   * + - Virtualization     - Virtualization vs Containerization     - Containerization using Dockers     - Docker Images     - Micro-services and Containerization     - Current Trends- Kubernetes, DevOps on Cloud, Function-As-A-Service (AWS Lambda) | T1- chapter 13  R3- 5 |

**Learning Outcomes:**

|  |  |
| --- | --- |
| **No** | **Learning Outcomes** |
| LO1 | Explain the need for DevOps and list down the primary benefits of DevOps from perspective of business and IT teams |
| L02 | List the ways in which DevOps uses new tools/technologies to deliver quality software more rapidly. |
| LO3 | Illustrate the practices of version control and configuration management. |
| LO4 | Summarize the essentials of continuous integration (CI) and outline the principles and practices of continuous delivery (CD) |
| LO5 | Implement an automated deployment pipeline and create a DevOps toolchain |

**Part B: Contact Session Plan**

|  |  |
| --- | --- |
| **Academic Term** | II Semester 2019 - 2020 |
| **Course Title** | Introduction to DevOps |
| **Course No** | CSIW ZG515 |
| **Lead Instructor** |  |

## Course Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **Type** | **Description** | **References** |
| **Module 1 Why and What is DevOps?** | | | |
| Pre-CH/CS | RL 1.1  RL1.2 | RL1.1 Foundational Terminology and Concepts   * RL1.1.1 Agile Methodology * RL1.1.2Operational Methodologies: ITIL   RL 1.2 Software Delivery   * RL2.1.1 Problems of Delivering Software * RL 2.1.2 Principles of Software Delivery   RL 1.3 About DevOps   * RL2.2.1 Need for DevOps * RL2.2.2 Evolution of DevOps * RL2.2.2 DevOps Practices |  |
| During CH/CS | CS 1 | * The Waterfall approach advantages and disadvantages * DevOps * Define the stages of a DevOps evolution * DevOps practices in organizations * The Continuous DevOps LifeCycle Process (Continuous Integration, Continuous Inspection, Continuous Deployment, Continuous Delivery, Continuous Monitoring) * Case Study- (IBM/Facebook/NetFlix) | T2-Chapter 1  R1-Chapter 4 |
| Post-CH/CS | HW/Lab |  |  |
| Lab Reference |  |  |  |
| **Module 2 DevOps Dimensions** | | | |
| Pre-CH/CS | RL 2.1  RL 2.2  RL 2.3 | RL2.1 Pillars of DevOps   * RL2.1.1 Three dimensions of DevOps – People, Process, Technology/Tools * RL2.1.2 DevOps Misconception * RL2.1.3 Agile Methodology - Scrum   RL2.2 DevOps- Process   * RL2.2.1 DevOps and Agile * RL2.2.2 Agile methodology for DevOps Effectiveness * Behavior Driven Development, Feature Driven Development and Test Driven Development   RL2.3 DevOps – People   * RL2.3.1 Team structure in a DevOps * RL2.3.2 Transformation to Enterprise DevOps culture * RL2.3.3DevOps Culture   RL2.4 DevOps-Tools   * RL2.4.1 Tools and Technology in DevOps * RL2.4.2 Cloud as a catalyst for DevOps |  |
| During CH/CS | CS2 | * DevOps- Process   Agile methodology for DevOps Effectiveness  Flow Vs Non-Flow based Agile processes  Choosing the appropriate team structure: Feature Vs Component teams  Enterprise Agile frameworks and their relevance to DevOps  Discuss (with examples and practical insights) Test Driven Development, Feature Driven Development, Behavior-driven development  Cloud as a catalyst for DevOps   * DevOps – People   Building competencies, Full Stack Developers  Self-organized teams, Intrinsic Motivation   * Technology in DevOps(Infrastructure as code, Delivery Pipeline, Release Management) * Tools/technology as enablers for DevOps * Discuss on Cloud as a catalyst for DevOps | T1 - Chapter 2, R1 - Chapter 4  Web technology for developers - https://developer.mozilla.org/en-US/docs/Web |
| Post-CH/CS | HW/Lab |  |  |
| Lab Reference |  |  |  |
| **Module 3 Source Code Management (Using GIT as an example tool)** | | | |
| Pre-CH/CS | RL3.1  RL3.2 | RL 3.1 Introduction to Version Control   * RL3.1.1 Evolution of Version Control * RL3.1.2 Version control system and its types   RL 3.2 Introduction to GIT   * RL3.2.1 About GIT * RL3.2.1 GIT Basics commands   RL3.3 GIT workflows   * RL3.3.1 Feature workflow * RL3.3.2 Centralized workflow   RL3.4 Clean Code Management   * RL3.4.1 Best Practices of Clean Code |  |
| During CH/CS | CS 3 | * Centralized Version Control Systems * Distributed Version Control Systems * Overview of GIT * Git Feature branching * Managing Conflicts using GIT * Tagging and Merging operations in GIT * Benefits of Clean code | T1 - Chapter 3,  R1 - Chapter 5 |
| Post-CH/CS | HW/Lab | Create a project in GIT and perform basic operations |  |
| Lab Reference | Lab Capsule 3 | Module 3 Lab Sheet Source Code Management |  |
| **Module 4 Continuous build and code quality** | | | |
| Pre-CH/CS | RL4.1  RL4.2 | RL 4.1 Manage Dependencies   * RL 4.1.1 What is Dependency? * RL 4.1.2 Common Dependency Problems   RL 4.2 Build Management   * RL 4.2.1 Introduction to build * RL 4.2.2 Build Tools – Maven and Gradle   RL 4.3 DevOps approach for Testing   * RL 4.3.1 Traditional Vs. Unit Testing * RL 4.3.2 Automated Test Suite – Selenium   RL 4.4 Need for Code Inspection & Analysis   * RL 4.4.1 Continuous code inspection - Code quality * RL 4.4.2 Code quality analysis tools- sonarqube |  |
| During CH/CS | CS 4 | * Automate the process of assembling software components with build tools * Use of Build Tools- Maven, Gradle * Outline Unit testing in DevOps * Enable Fast Reliable Automated Testing * Setting up Automated Test Suite – Selenium * Effectiveness of Code quality in Continuous Code Inspection * Code quality analysis usingsonarqube | T1 - Chapter 4,  R1 - Chapter 6, 7, 8, 9, 10, 11 |
| Post-CH/CS | HW/Lab | Understand the workflow of Selenium and sonarqube |  |
| Lab Reference | Lab Capsule 4 | Module 4 Lab Sheet2 – Continuous build and code quality |  |
| **Module 5 Continuous Integration and Continuous Delivery** | | | |
| Pre-CH/CS | RL 5.1 | RL 5.1 Implementing Continuous Integration   * RL5.1.1 Continuous Integration * RL5.1.2 Using Continuous Integration Software   RL5.2 Continuous Integration System   * RL5.2.1 Introduction to Jenkins * RL5.2.2 Preparing your Jenkins environment * RL5.2.3 Integrating Source code management, build, testing tools etc., with Jenkins - plugins * RL5.2.4 Jenkins Pipeline   RL5.3 Artifacts management   * RL5.3.1 Importance of Artifact Management |  |
| During CH/CS | CS5 | * Overview of Continuous Integration-Version control, automated build, Test * Prerequisites for Continuous Integration * Continuous Integration Practices * Team responsibilities * Using Continuous Integration Software (Jenkins as an example tool) * Jenkins Architecture * Overview of Artifacts management * Setting up the Continuous Integration pipeline * Continuous delivery to staging environment or the pre-production environment * Self-healing systems | T1 - Chapter 5,  R1 - Chapter 12, 13 |
| Post-CH/CS | HW/Lab | Implementation of CI using Jenkins |  |
| Lab Reference | Lab Capsule 5 | Module 5 Lab Sheet- Continuous Integration |  |
| **MID SEMESTER EXAMINATION** | | | |
| **Module 6 Continuous Deployment** | | | |
| Pre-CH/CS | RL 6.1  RL 6.2  RL 6.3 | RL6.1 Continuous Deployment   * RL6.1.1 Introduction to Continuous Deployment * RL6.1.2 Importance of Automated Release Management * RL6.1.3 Deployment Pipeline * RL6.1.4 Pros and Cons of Continuous Deployment   RL6.2 Strategies of managing Deployment   * RL6.2.1Blue/Green Deployment * RL6.2.1 Rolling Upgrade |  |
| During CH/CS | CS6 | * Human-free deployments * Implementing and Automating the deployment process * Deploying it to testing environments * Releasing software into production * Environment-based release patterns * Rolling Back Deployments and Zero-Downtime Releases * Case study on Blue/Green Deployment * Illustrate Rolling Upgrade with real time examples * The canary release pattern- Dark Launches | T1 - Chapter 7,  R1 - Chapter 14 |
| Post-CH/CS | HW/Lab | Implement CI/CD Pipeline using Jenkins |  |
| Lab Reference | Lab Capsule 6 | Module 6 Lab Sheet- Continuous Delivery and Continuous Deployment |  |
| **Module 7 Continuous Monitoring** | | | |
| Pre-CH/CS | RL 7.1  RL 7.2 | RL7.1 Introduction to Continuous Monitoring   * RL7.1.1 Monitoring :: Let the system work for you * RL7.1.2 Importance of Monitoring in DevOps * RL7.1.3 Monitoring Tools in DevOps   RL7.2 Introduction to ELK   * RL7.2.1 Use of ELK |  |
| During CH/CS | CS7 | * Need for continuous monitoring * Goals of monitoring * Challenges of monitoring under continuous change * Alert management * Analytics * Continuous customer feedback and optimization * Overview of ELK (Elasticsearch, Logstash, and Kibana) Stack | R1 - Chapter 21 |
| Post-CH/CS | HW/Lab |  |  |
| Lab Reference |  |  |  |
| **Module 8 Configuration Management** | | | |
| Pre-CH/CS | RL 8.1  RL 8.2 | RL8.1 Infrastructure as a code   * RL8.1.1 Introduction to Infrastructure as a code * RL8.1.2 Automation in Infrastructure Management   RL8.2 Configuration Management   * RL8.2.1 Importance of Configuration Management * RL8.2.2 On-demand Infrastructure Management * RL8.2.3 CM Tools- Puppet & Chef * RL8.2.4 CM Tools- Ansible (Agentless) |  |
| During CH/CS | CS8 | * Managing Infrastructure and Environments(Production, pre-production, Test, Developer Environment) * Environment provisioning * Automating and Managing Server Provisioning * Enterprise solutions Chef, Puppet and Ansible * Managing on-demand infrastructure, Auto scaling | T1 - Chapter 9, R1 - Chapter 23  PHP  http://php.net/manual/en/getting-started.php |
| Post-CH/CS | HW/Lab | \*\*\*\* |  |
| Lab Reference |  |  |  |
| **Module 9Virtualization and Containerization** | | | |
| Pre-CH/CS | RL 9.1 | RL9.1Virtualization and Containerization   * RL9.1.1 Introduction to Virtualization * RL9.1.2Introduction to Containerization * RL9.1.3 Containerization using Dockers   RL9.2 Micro-services and Function as a Service   * RL9.2.1 Overview of Micro-services * RL9.2.2 Introduction to AWS Lambda * RL9.2.3 Current Trends |  |
| During CH/CS | CS9 | * Virtualization vs Containerization * Dockers and Docker Images * Micro-services and Containerization * Current Trends- Kubernetes, DevOps on Cloud, Function-As-A-Service (AWS Lambda) | T1 - Chapter 9, R1 - Chapter 23 |
| Post-CH/CS | HW/Lab |  |  |
| Lab Reference |  |  |  |
| **CS10 : Review Session** | | | |

***Select Topics and Case Studies from business for experiential learning***

|  |  |  |
| --- | --- | --- |
| **Topic No.** | **Select Topics in Syllabus for experiential learning** | **Access URL** |
|  | Version Control using GIT  Creating repositories in GIT,  Exercises to demonstrate the use of GIT operations andcommands(Push, pull, clone etc.,)  Creating branches and merging branches using GIT | R3 |
|  | Installation of Jenkins and Configuration of Jenkins to work with different version control, build and testing tools | R3 |
|  | Create jobs and projects in Jenkins | R3 |
|  | Demonstration of continuous integration with Jenkins through source code polling and build triggers | R3 |
|  | Demonstrate continuous inspection with Jenkins using sonarqube to ensure code quality | R3 |
|  | Demonstration of continuous deployment/delivery to staging/production environment with Jenkins. | R3 |