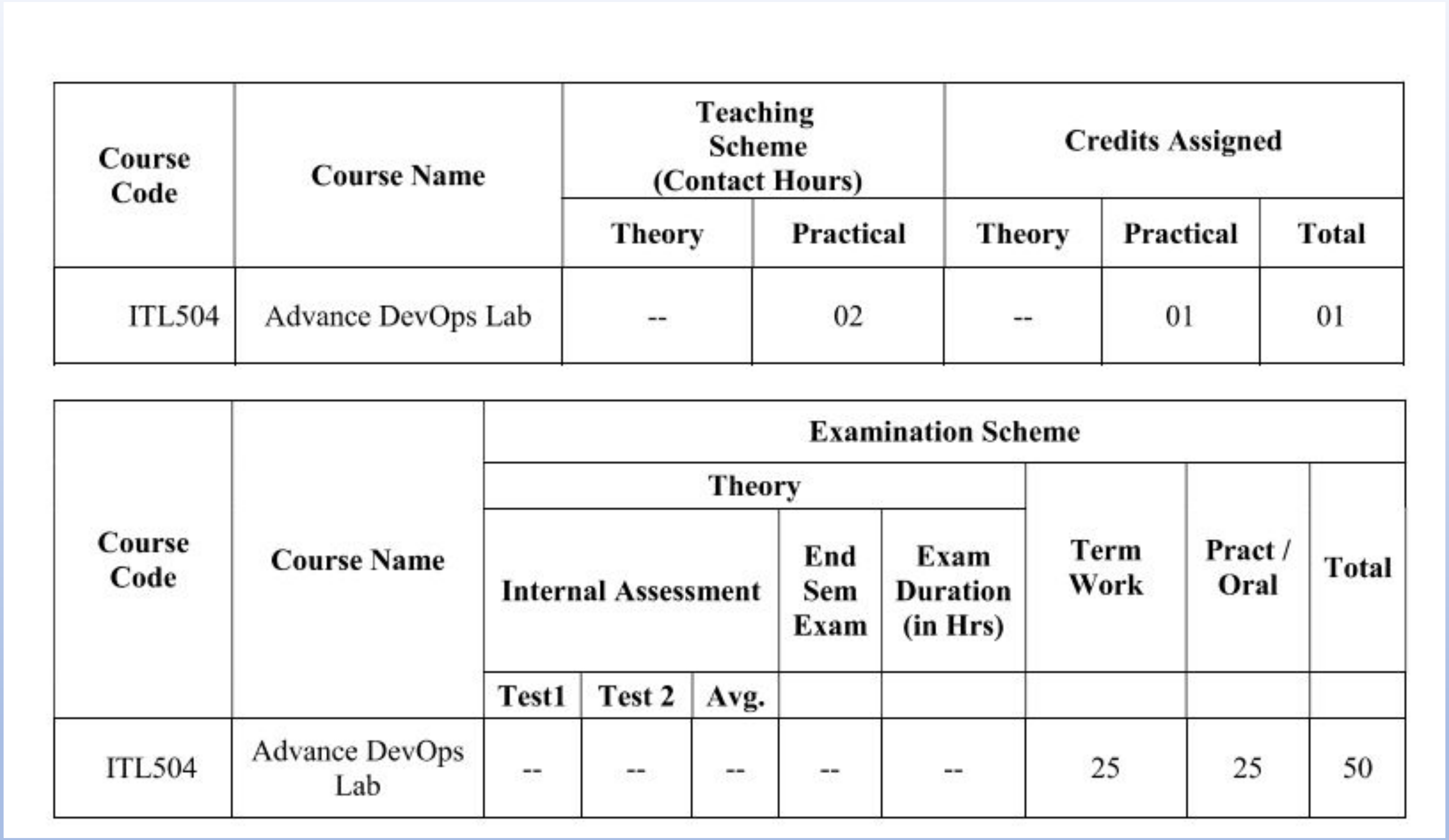
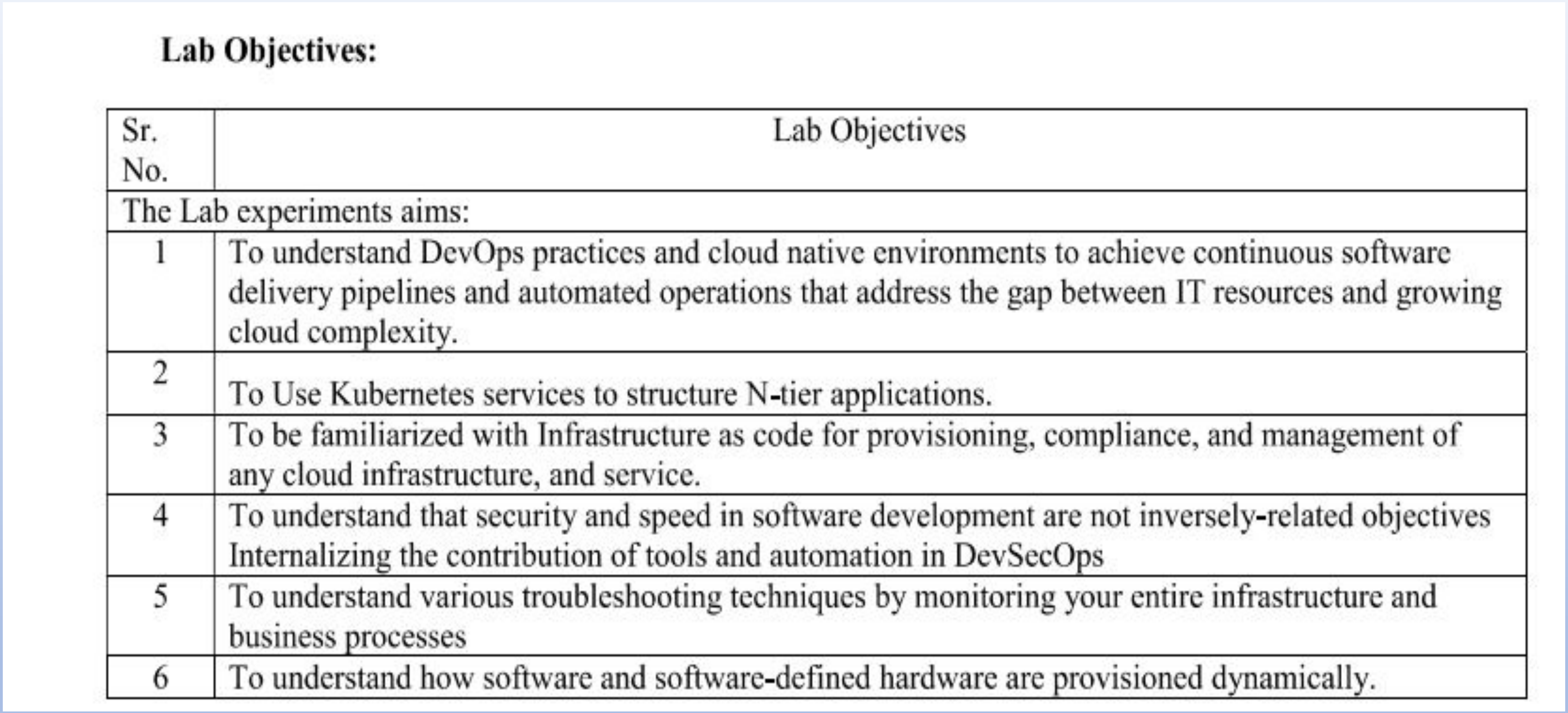


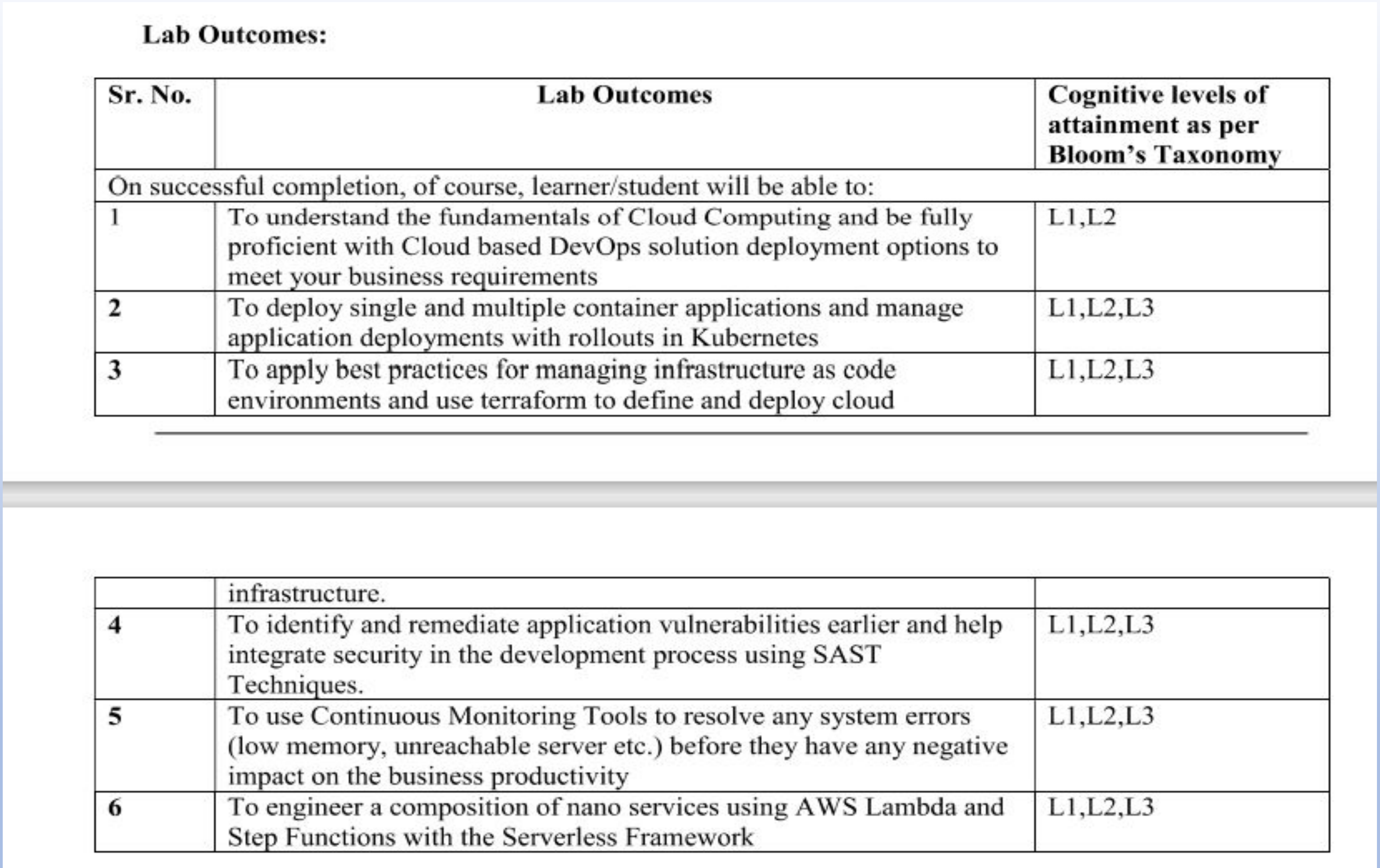
Course Code	Course Name	Teaching Scheme (Contact Hours)		Credits Assigned		
		Theory	Practical	Theory	Practical	Total
ITL504	Advance DevOps Lab	--	02	--	01	01

Course Code	Course Name	Examination Scheme							
		Theory					Term Work	Pract / Oral	Total
		Internal Assessment			End Sem Exam	Exam Duration (in Hrs)			
		Test1	Test 2	Avg.					
ITL504	Advance DevOps Lab	--	--	--	--	--	25	25	50



Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
1	To understand DevOps practices and cloud native environments to achieve continuous software delivery pipelines and automated operations that address the gap between IT resources and growing cloud complexity.
2	To Use Kubernetes services to structure N-tier applications.
3	To be familiarized with Infrastructure as code for provisioning, compliance, and management of any cloud infrastructure, and service.
4	To understand that security and speed in software development are not inversely-related objectives Internalizing the contribution of tools and automation in DevSecOps
5	To understand various troubleshooting techniques by monitoring your entire infrastructure and business processes
6	To understand how software and software-defined hardware are provisioned dynamically.



Lab Outcomes:

Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
On successful completion, of course, learner/student will be able to:		
1	To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements	L1,L2
2	To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes	L1,L2,L3
3	To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.	L1,L2,L3
4	To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.	L1,L2,L3
5	To use Continuous Monitoring Tools to resolve any system errors (low memory, unreachable server etc.) before they have any negative impact on the business productivity	L1,L2,L3
6	To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework	L1,L2,L3

Prerequisite: Operating System, Linux Administration, Java /Web Application Programming, Software Engineering, Cloud Computing and DevOps Ecosystem.

Hardware & Software Requirements:

Hardware Requirements	Software Requirements	Other Requirements
PC With following Configuration 1. Intel i3 core or above 2. 4 GB RAM or above 3. 500 GB HDD 4. Network interface card	1. Linux / Windows Operating system 2. VIRTUAL BOX/ VMWARE	1. Internet Connection for installing additional packages 2. GitHub account 3. AWS free tier account

V	DevSecOps: Continuous Monitoring	In this module, you will learn to detect, report, respond to the attacks and issues which occur within the infrastructure. • Introduction to Continuous Monitoring • Introduction to Nagios • Installing Nagios • Nagios Plugins (NRPE) and Objects Nagios Commands and Notification • Monitoring of different servers using Nagios	**04**	LO1, LO5
		Self-Learning Topics: Splunk, Snort, Tenable		
VI	NoOps: Serverless Computing	In this module, you will learn serverless computing platform like AWS Lambda, which allows you to build your code and deploy it without ever needing to configure or manage underlying servers. • AWS Lambda - Overview and Environment Setup • Building and Configuring the Lambda function (NODEJS/PYTHON/JAVA) • Creating & Deploying using AWS Console/CLI • Creating & Deploying using Serverless Framework **Self-Learning Topics:** AWS Lambda • Create a REST API with the Serverless Framework	**04**	LO1, LO6

List of Experiments:

Sr. No	Experiment Title
1	To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.
2	To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.
3	To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.
4	To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.
5	To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.
6	To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.
7	To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.
8	Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.
9	To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.
10	To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.
11	To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.
12	To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Term Work: Term Work shall consist of at least 12 to 15 practicals based on the above list. Also Term work Journal must include at least 2 assignments based on the self-learning topics mentioned in syllabus.

Term Work Marks: 25 Marks (Total marks) = 15 Marks (Experiment) + 5 Marks (Assignments) + 5 Marks (Attendance)

Practical & Oral Exam: An Practical & Oral exam will be held based on the above syllabus.

Advance DevOps

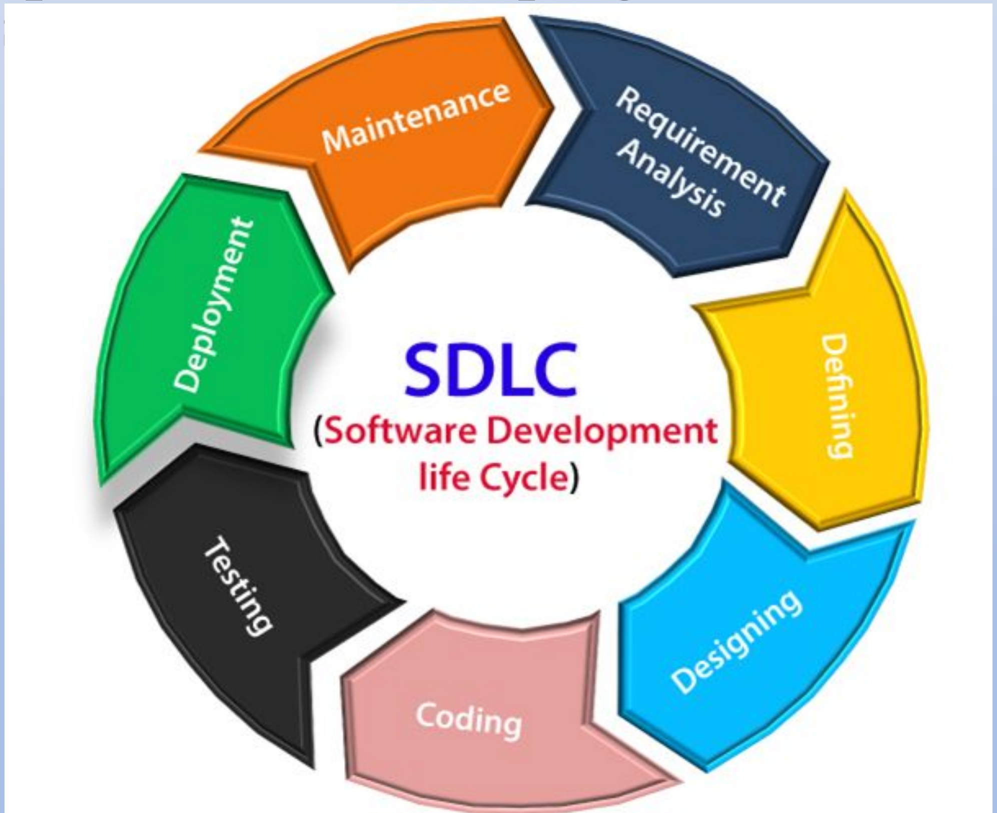
By- Ms. Snehal Mhatre

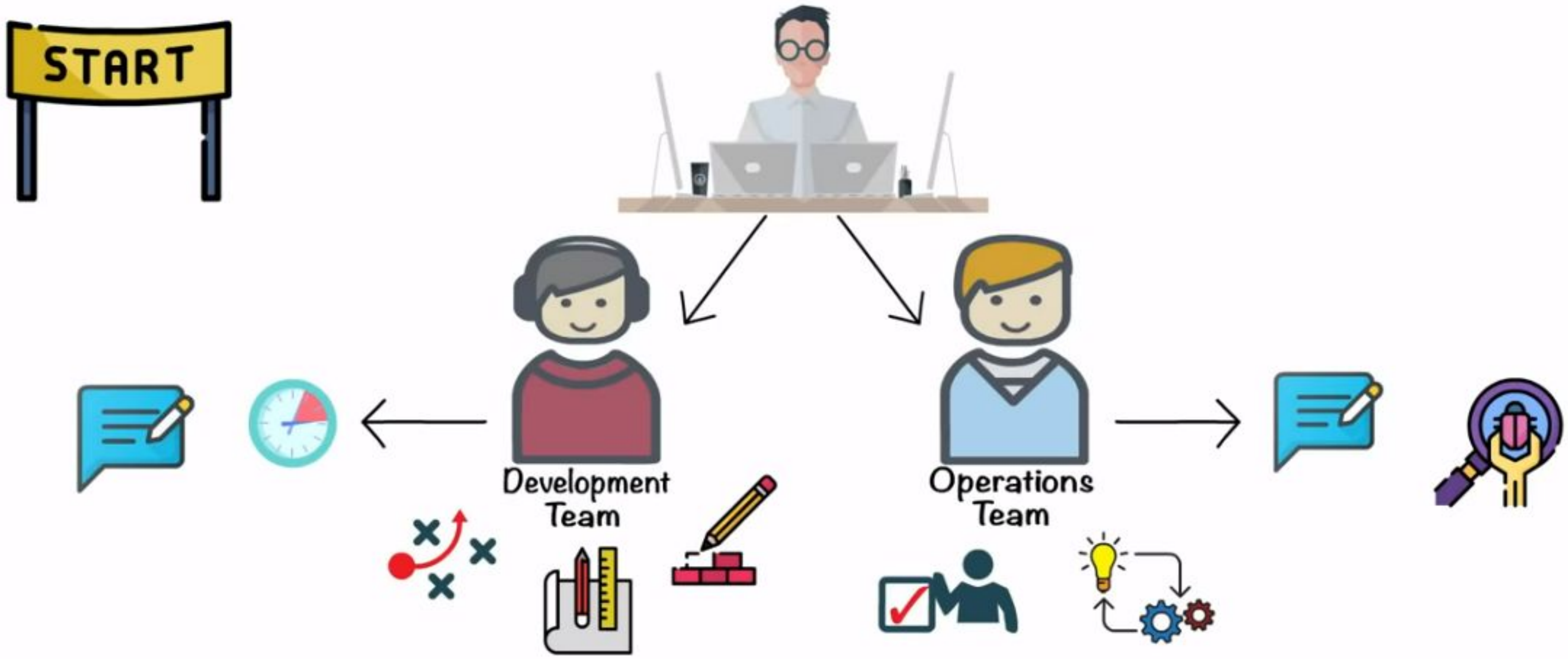
Software Development Life Cycle (SDLC)

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

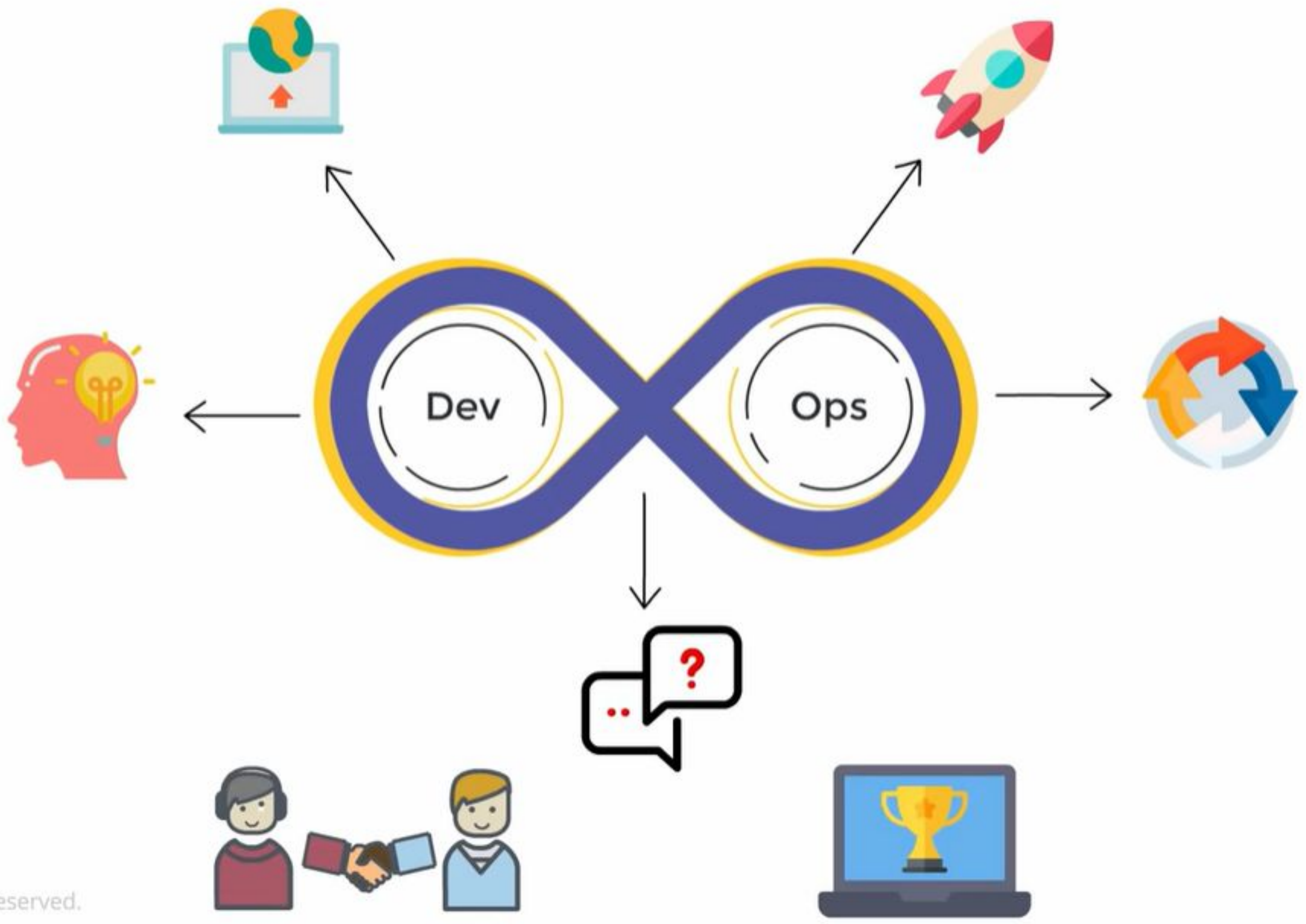
SDLC Cycle

SDLC Cycle represents the process of developing software. SDLC framework includes the following step





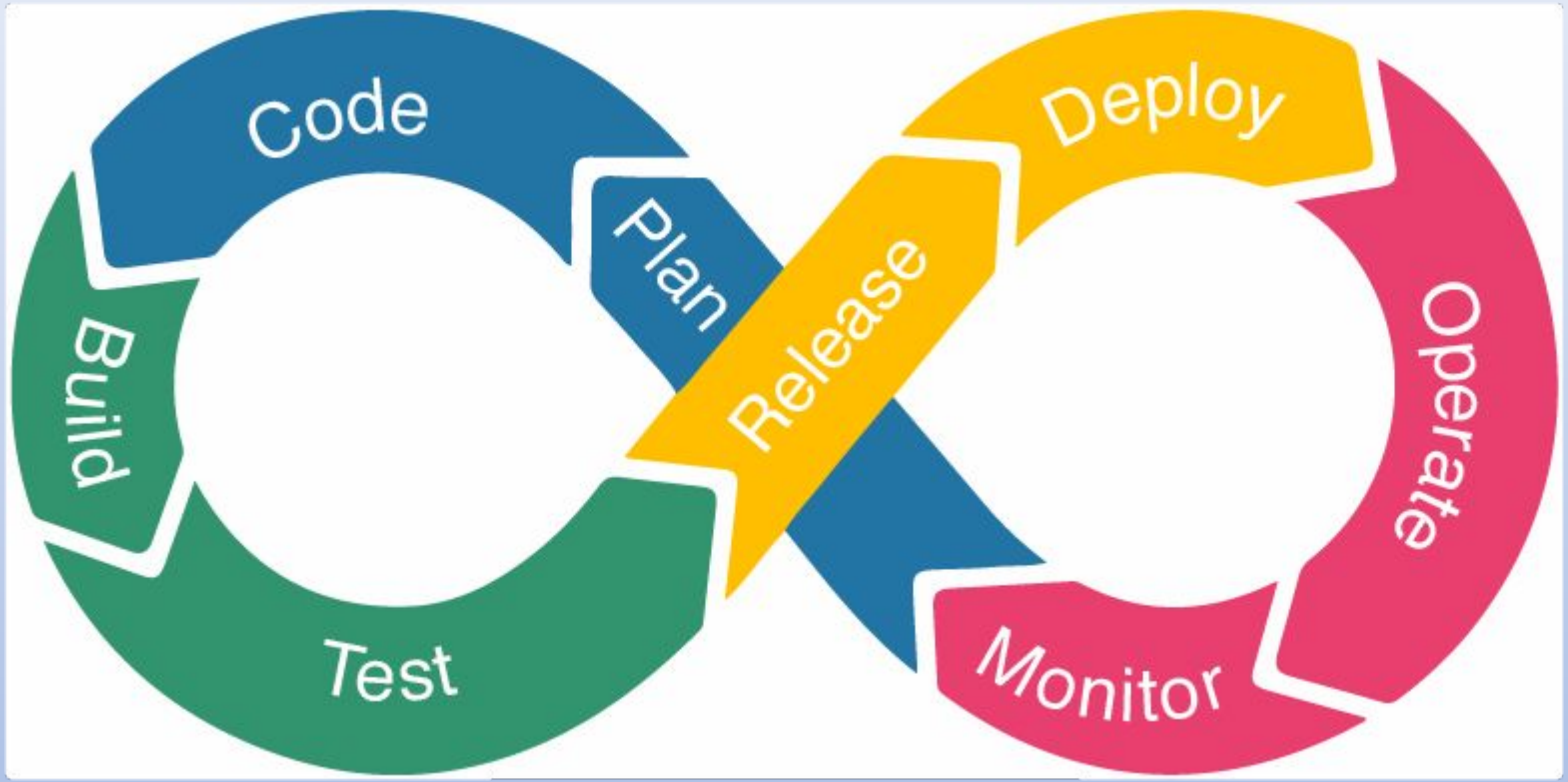
This undoubtedly extended timelines and delayed the entire software development cycle

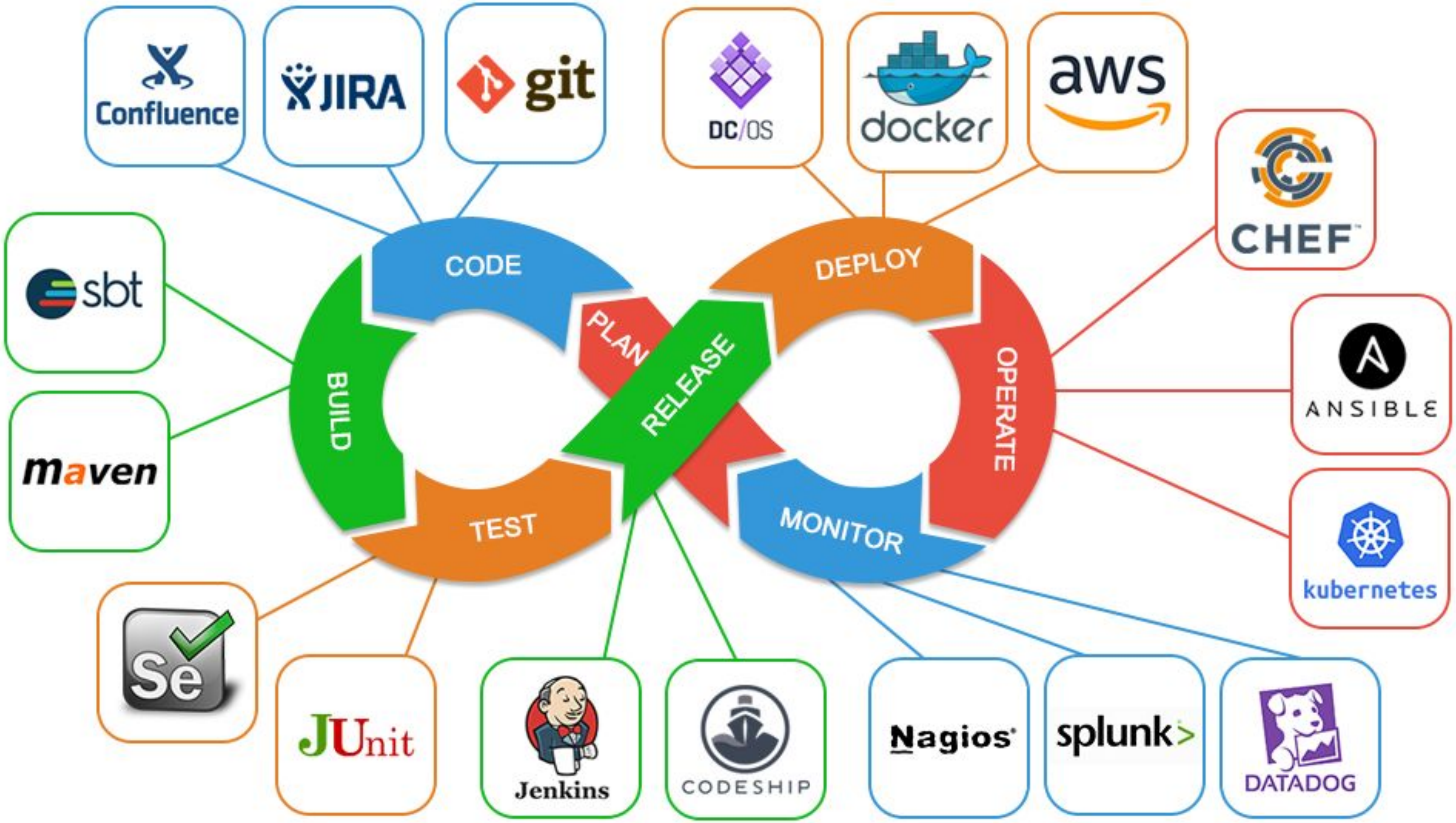

Dev
Ops

DevOps Lifecycle:

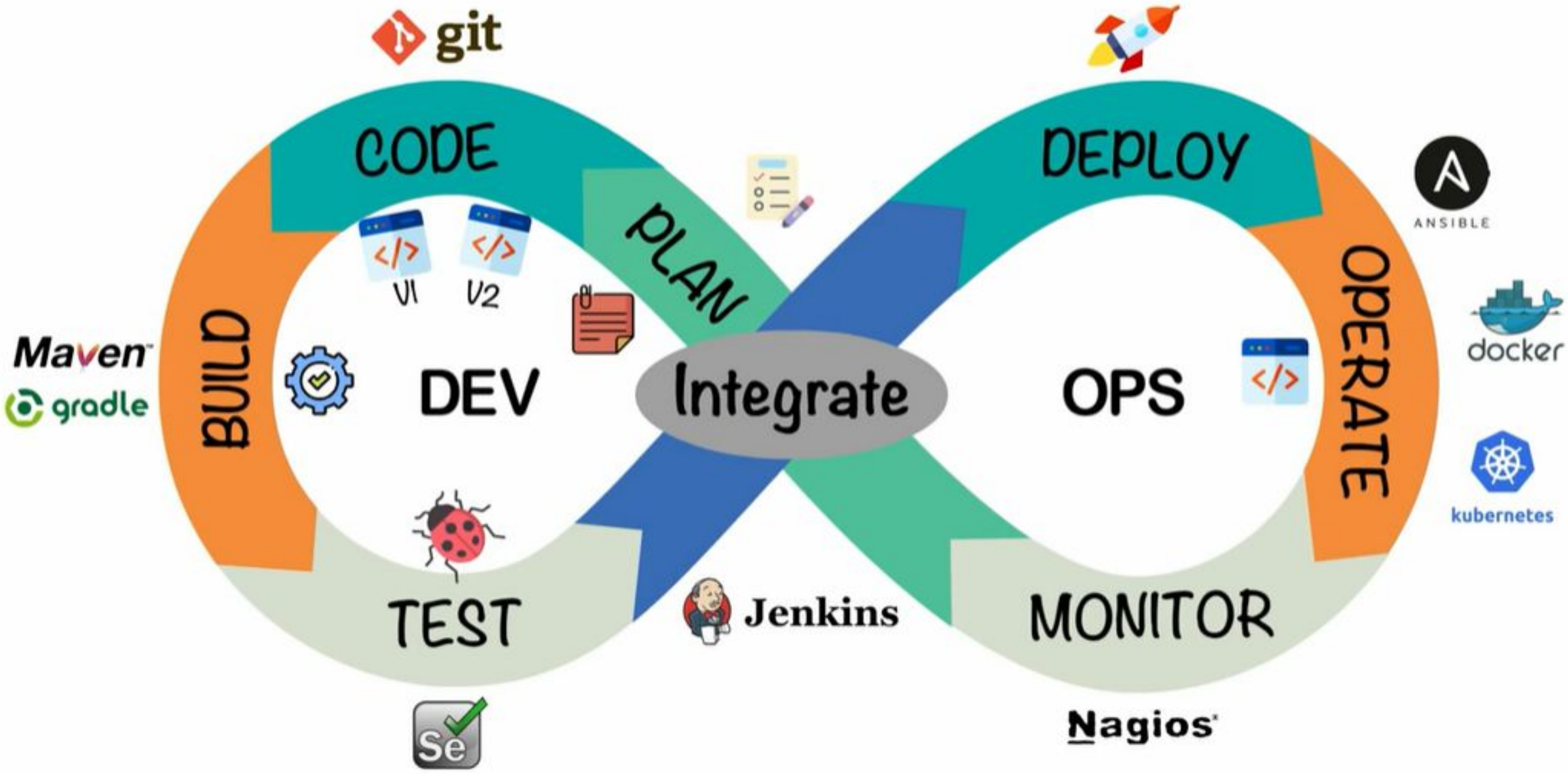
DevOps is a culture that maintains collaboration between Development and Operation. So it’s a combination of two words “Development” and “Operation”. DevOps aims to increase the organization’s speed to deliver an application from the development phase to production.

As everybody knows after development, the application needs to go under some phases like build, test, deploy, operate and monitor. Each phase has its standards.




Confluence
JIRA
git
DC/OS
docker
aws
CHEF
sbt
maven
ANSIBLE
kubernetes
CODE
PLAN
DEPLOY
BUILD
RELEASE
OPERATE
TEST
MONITOR
Se
JUnit
Jenkins
CODESHIP
Nagios
splunk>
DATADOG

The DevOps culture is implemented in several phases with the help of several tools



- **Plan:** In this stage, teams identify the business requirement and collect end-user feedback. They create a project roadmap to maximize the business value and deliver the desired product during this stage.
- **Code:** The **code development** takes place at this stage. The development teams use some tools and plugins like *Git* to streamline the development process, which helps them avoid security flaws and lousy coding practices.
- **Build:** In this stage, once developers finish their task, they commit the code to the shared code repository using build tools like Maven and Gradle.
- **Test:** Once the build is ready, it is deployed to the test environment first to perform several types of testing like user acceptance test, security test, integration testing, performance testing, etc., using tools like JUnit, Selenium, etc., to ensure software quality.

- **Release:** The build is ready to deploy on the production environment at this phase. Once the build passes all tests, the operations team schedules the releases or deploys multiple releases to production, depending on the organizational needs.
- **Deploy:** In this stage, Infrastructure-as-Code helps build the production environment and then releases the build with the help of different tools.
- **Operate:** The release is live now to use by customers. The operations team at this stage takes care of server configuring and provisioning using tools like Chef.
- **Monitor:** In this stage, the DevOps pipeline is monitored based on data collected from customer behavior, application performance, etc. Monitoring the entire environment helps teams find the bottlenecks impacting the development and operations teams' productivity.

There are many tech giants and organizations that have opted for the DevOps approach

amazon

NETFLIX

Walmart

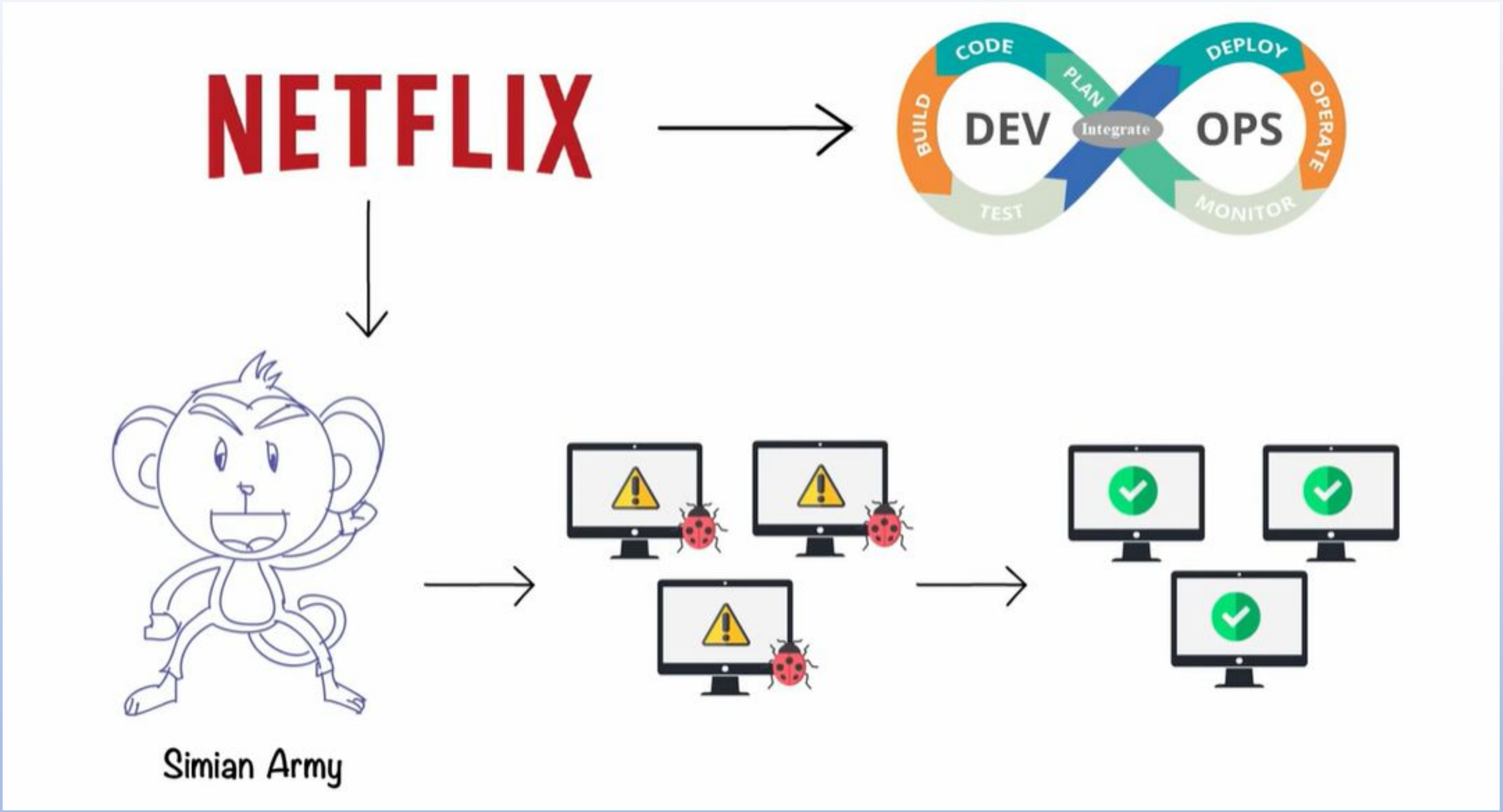
Netflix introduced its online streaming service in 2007

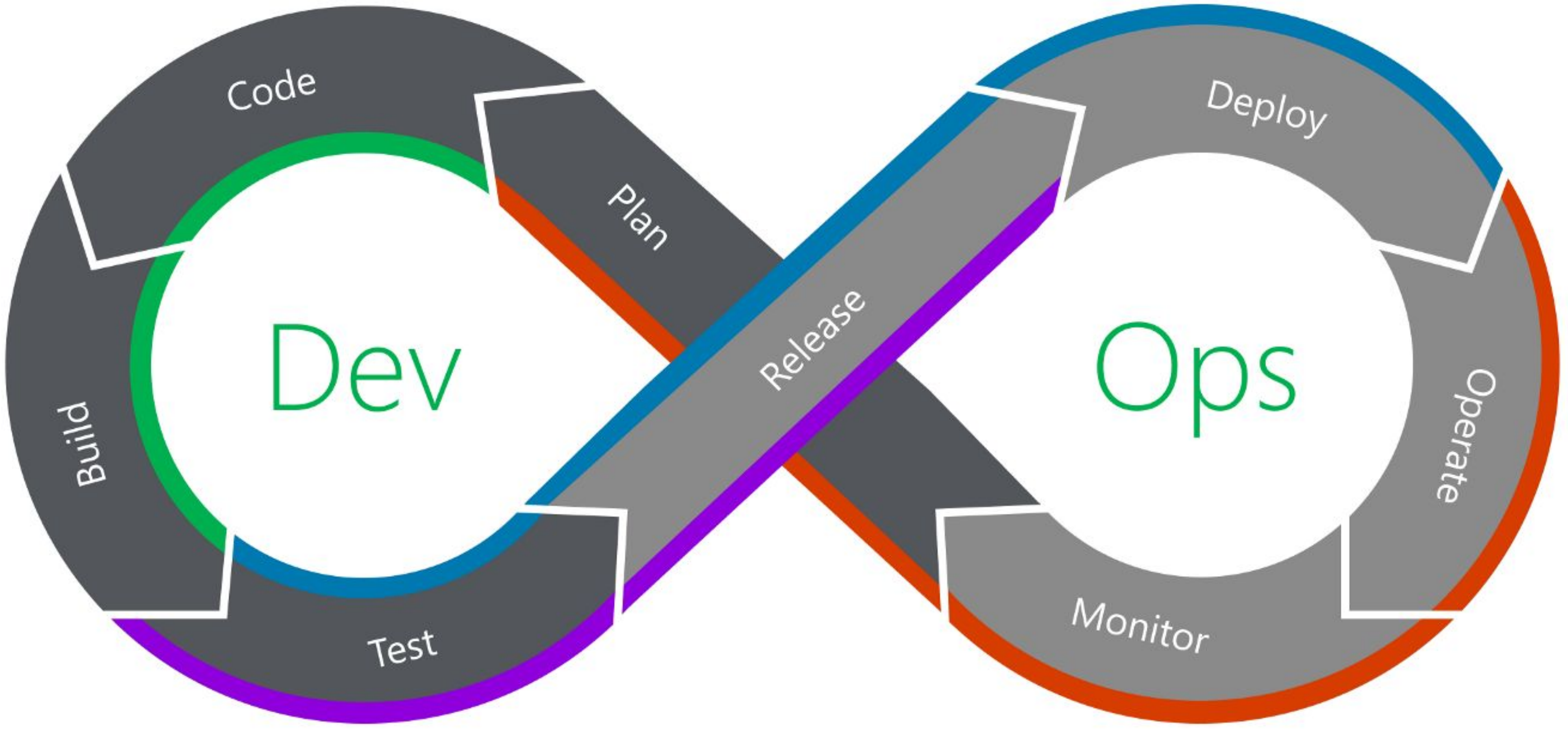
In 2014

www.netflix.com
NETFLIX
1 Hour

$\longrightarrow$ $200,000

However, now Netflix can cope with such issues


NETFLIX
CODE
PLAN
DEPLOY
BUILD
DEV
Integrate
OPS
OPERATE
TEST
MONITOR
Simian Army



🟩 Continuous Integration (CI) 🟦 Continuous Deployment (CD)

🟪 Continuous Delivery (CD) 🟥 Continuous Feedback (CF)