**EXP.1**

In today’s world software delivery is the heart of today’s top technology trends that say it can be anything .. ex: Big data, IoT, Cloud, Social Business, Touch commerce. So in order to get a product from “Ideation” to “Realization”, we need applications to be delivered quickly and frequently to production.

So to give a better insight into the above terms, we have looked back to our traditional approach towards software delivery. Years back DevOps didn’t exist, Application development was following the Traditional model. For example, the Dev team on one hand will be writing code, fixing bugs and raising build/deploy requests to the Ops team. The Ops team is responsible for Collaborate, Build, Deploy the source code from Dev to Production and Setup monitoring, maintain the infrastructure and operate.

So the challenge here is to get the source code to production. It was taking several hours and even months to get the product released to the customer in traditional release management.

To address this problem, and to increase the collaboration between developers and operations DevOps came into picture. Then the philosophy has changed from **“Dev and Ops” to “DevOps”**

DevOps uses tools such as Git, Vagrant, Docker, Ansible, Jenkins

1. Speed (Frequent and Faster product delivery in a short span of time.)
2. Rapid Delivery
3. Reliability
4. Greater collaboration between teams (Dev/Ops/Business)
5. Scale
6. Continuous Integration/Testing/Delivery/Deployment
7. Automation.
8. Security

**EXP.3**

First we will talk about IDE.

IDE is generally used by Software developers who execute many different types of tasks during the software development life cycle of an application like:

* compile and build the source code
* correct syntax errors
* debug runtime errors
* maintain version control changes to the source code
* execute source code tests etc

IDEs often has various abilities like:

* language-specific editors
* code completion capabilities
* syntax highlighting
* programming language documentation
* code debugging
* source control management tools, such as Git, SVN, or Mercurial

In our experiment we are going to use Git. Why it is used, how to install it, how to use it for various purposes like version control and all.

Some of the basic operations in Git are:

1. Initialize

2. Add

3. Commit

4. Pull

5. Push

Some advanced Git operations are:

1. Branching

2. Merging

Git can be installed on ubuntu as well

**Sudo apt-get install git**

**Git version**

**Mkdir git-dvcs**

**Cd git-dvcs**

**Git config --global**

As there are no users defined, let us define it using following two commands

**git-dvcs$ git config --global user.name "vishal"**

**/git-dvcs$ git config --global user.email "vsbadgujar@apsit.edu.in"**

Now, check the list of users

**git config --global –list**

Let us create a repository for version control named “git-demo-project”

**vishal@vishal:~/git-dvcs$ mkdir git-demo-project**

**vishal@vishal:~/git-dvcs$ cd git-demo-project/**

Now, initialize the repository using following command

**vishal@vishal:~/git-dvcs$ git init**

**Git add .**  (. means current directory)

To check the status of repository, use

**vishal@vishal:~/git-dvcs/git-demo-project$ git status**

A Version Control System (VCS) enables you to efficiently manage and collaborate on code changes with others. Version control systems provide many benefits, including:

* The ability to review and restore old versions of files.
* The ability to compare two versions of the same file to identify changes.
* A record or log of who made changes at a particular time.
* Mechanisms for multiple users to collaboratively modify files, resolve conflicting changes, and merge the changes together.

There are several open source version control systems available including:

* CVS
* SVN
* Git
* Mercurial

**EXP.2**