Version Control with Git

git version control

Presented By :-Raman

Email:- ramansharma95@gmail.com

Phone:- +91-9739299502

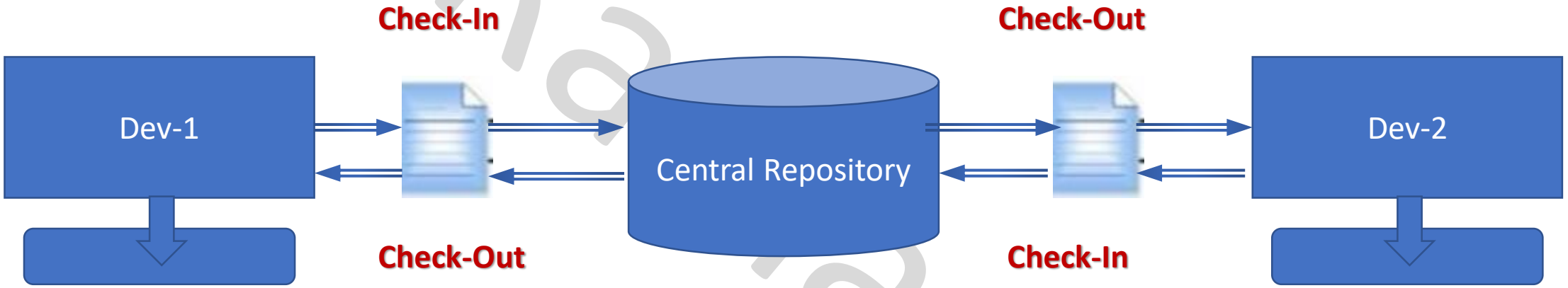
What is Version Control System

- Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Centralized Control System

- Centralized version control systems are based on the idea that there is a single “central” copy of your project somewhere.
- Committing” a change simply means recording the change in the central system.
- Other programmers can then see this change. They can also pull down the change, and the version control tool will automatically update the contents of any files that were changed.

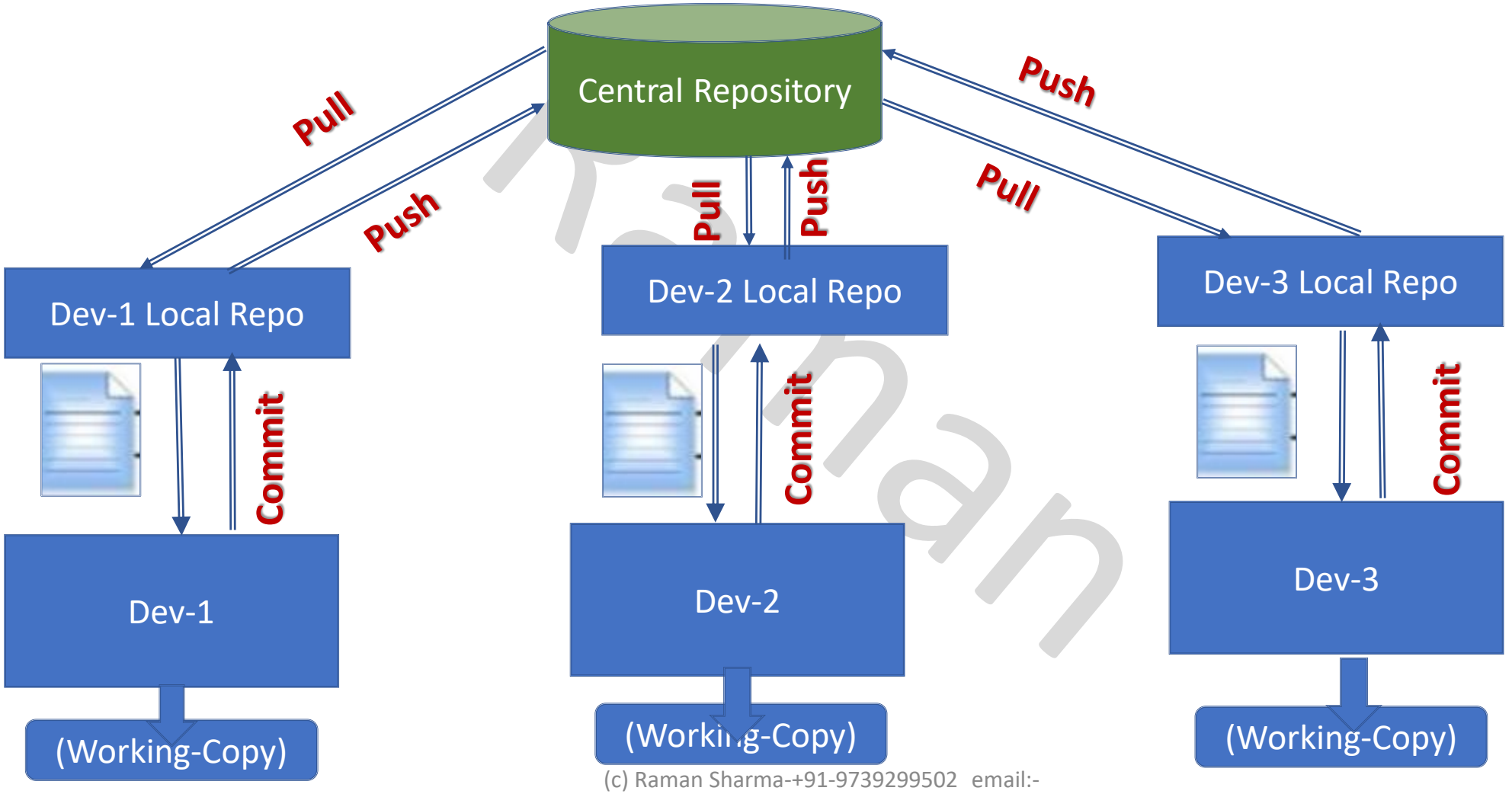
Client-Server Repository Model



Centralized Version Control Workflow

- Pull down any changes other people have made from the central server.
- Make your changes, and make sure they work properly.
- Commit your changes to the central server, so other programmers can see them.
- Some of the most common centralized version control systems you may have heard of or used are CVS, Subversion (or SVN) and Perforce.

Distributed Version Control Systems (DVCS)



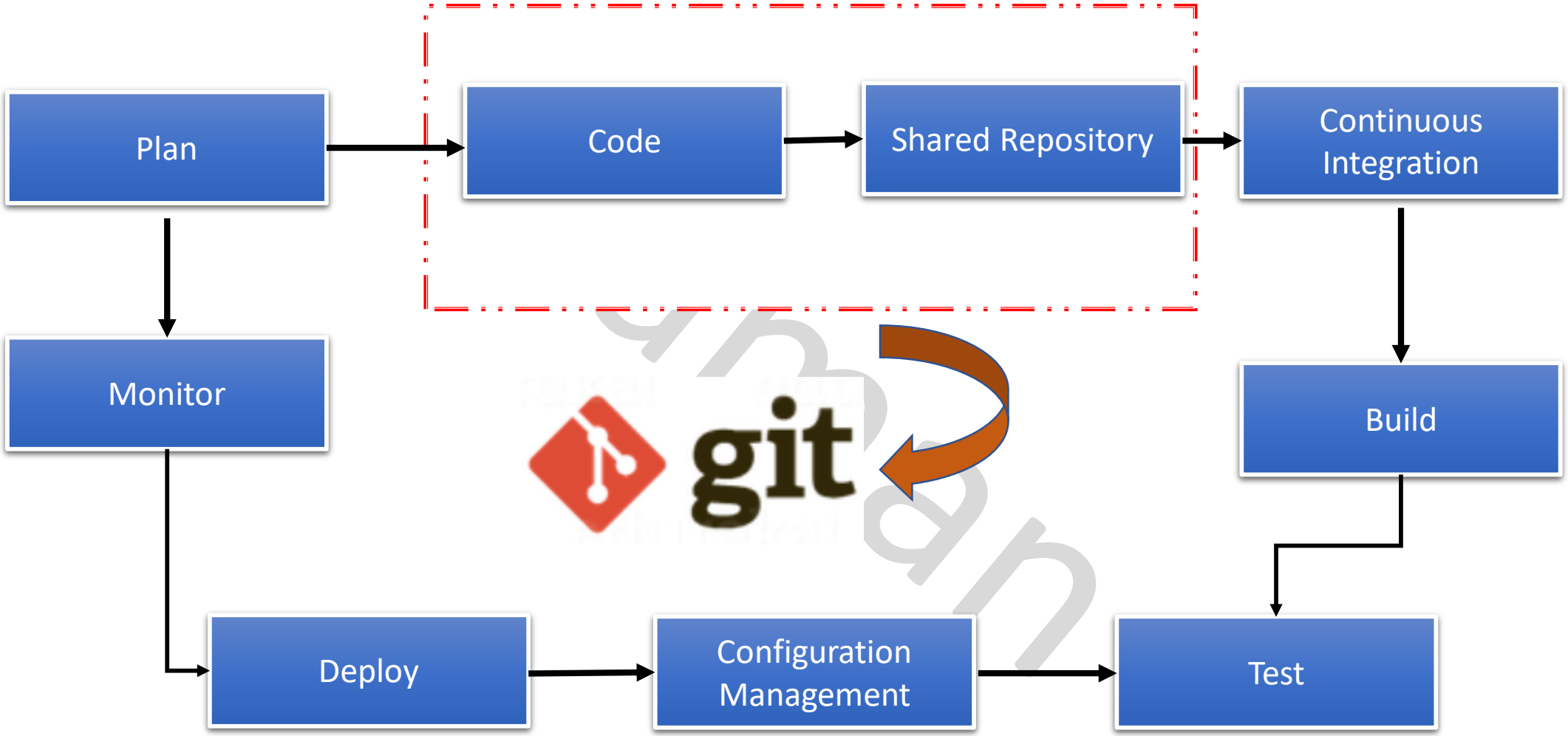
Advantages

- Performing actions other than pushing and pulling changesets is *extremely* fast because the tool only needs to access the hard drive, not a remote server.
- Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
- Everything but pushing and pulling can be done without an internet connection. So you can work on a plane, and you won't be forced to commit several bugfixes as one big changeset.
- Since each programmer has a full copy of the project repository, they can share changes with one or two other people at a time if they want to get some feedback before showing the changes to everyone.

What is Git

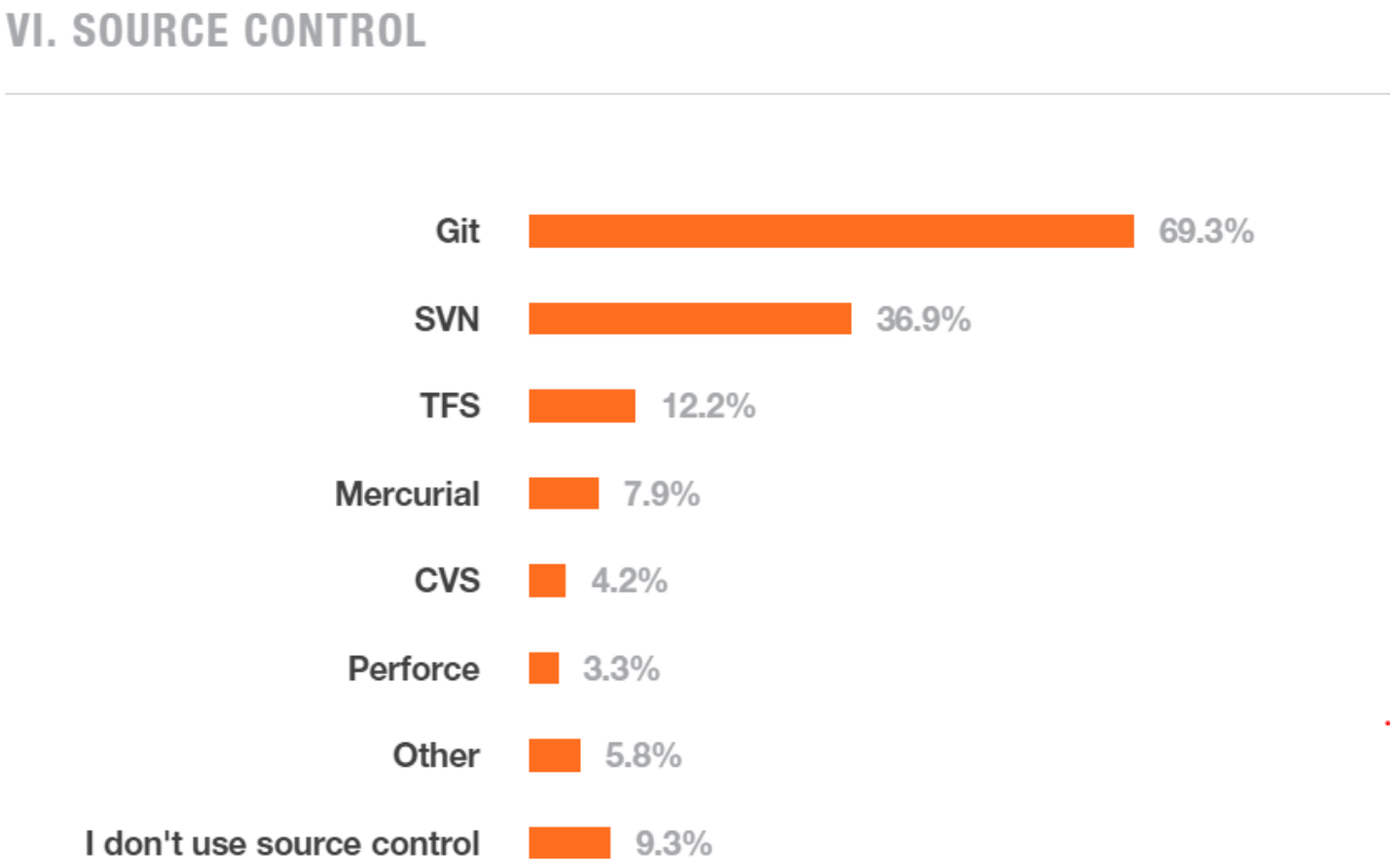
- Git is most commonly used version control system today and quickly becoming the standard for version control.
- Git is a free and open source version control system, originally created by Linus Torvalds in 2005.
- Git is distributed version Control System means your local copy of code is complete version control repository.
- You commit your work locally, and then sync your copy of the repository with the copy on the server.

Git in DevOps



Standard Version Control System

https://insights.stackoverflow.com/survey/2015#tech-sourcecontrol



Thanks

Git installation

- Below command is used to update the yum repository so that git package should be available.

  sudo yum –y update

- Command to install git (it will ask to install the package then press Y)

  sudo yum install git

Thanks

Git Internals - Environment Variables

- Git always runs inside a bash shell, and uses a number of shell environment variables to determine how it behaves. Occasionally, it comes in handy to know what these are, and how they can be used to make Git behave the way you want it to.

Environment Variables - GIT_EXEC_PATH

- GIT_EXEC_PATH determines where Git looks for its sub-programs (like git-commit, git-diff, and others). You can check the current setting by running git --exec-path.

Environment Variables - HOME

- Home is not usually considered customizable (too many other things depends upon it) but it is where Git looks for the global configuration file.

Environment Variables – GIT_DIR

- GIT_DIR is the location of the .git folder. If this isn't specified, Git walks up the directory tree until it gets to ~ or /, looking for a .git directory at every step.

Git Commands

- git init

Git Commands-git init

- Name git-init - Create an empty Git repository or reinitialize an existing one.
- Synopsis

git init [-q | --quiet] [--bare] [--template=<template_directory>]

[--separate-git-dir <git dir>]

[--shared[=<permissions>]] [directory]

Git Commands-git init

- This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial HEAD file that references the HEAD of the master branch is also created.

    1. Create a directory called gitProject
    2. Change to gitProject directory
    3. Initialize git local repository under this folder.
    4. Check the files and folders (including hidden folders under gitProject), it should have .git folder

Git Commands-git init

```
[ec2-user@gitServer ~]$ mkdir gitProject
[ec2-user@gitServer ~]$ cd gitProject/
[ec2-user@gitServer gitProject]$ git init
Initialized empty Git repository in /home/ec2-user/gitProject/.git/
[ec2-user@gitServer gitProject]$ ls -la
total 0
drwxrwxr-x. 3 ec2-user ec2-user  18 Apr 23 15:48 .
drwx------. 5 ec2-user ec2-user 143 Apr 23 15:48 ..
drwxrwxr-x. 7 ec2-user ec2-user 119 Apr 23 15:48 .git
[ec2-user@gitServer gitProject]$ ls la .git
ls: cannot access la: No such file or directory
.git:
branches  config  description  HEAD  hooks  info  objects  refs
```