Basics of Git and Version control

- Prepared by Team CALMS squad

Meaning of Version control in Software development

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.

For almost all software projects, the source code is like the crown jewels - a precious asset whose value must be protected. For most software teams, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort. Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.

Source: https://www.atlassian.com/git/tutorials/what-is-version-control

Why Git for you?

we'll discuss how Git benefits each aspect of Addweb **Developers** and **Designers** as well as **DevOps**, from your development team to your marketing team, and everything in between. By the end of this Presentation we all will understand why and what we need to learn about git.

Feature Branch Workflow

One of the biggest advantages of Git is its branching capabilities. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something—no matter how big or small—they create a new branch. This ensures that the master branch always contains production-quality code.

Example: Developers wants to create a New Feature that is required by client then they will create their own feature branch "**Feature/Name_of_feature**"

Ways of Creating Branches:

Normally new branches can be created from the console (**it would be recommended that be logged in on console of a github to get the clear idea of branching and committing**)

Go to branch > **Type the name of the branch** > It will ask you to create a branch with the name you searched > **New branch will be created**

If You want to create a new empty branch then it is called **Orphan branch**

Syntax in Terminal: Git checkout --orphan name_of_the_branch

Make the workflow of Git Clear in the mind:

git clone [url] : Downloads a project and its entire version history

Git status: To check the current status of your local code in your system

Git add: To add the changes you made in the branch of a repository you are working

git commit -m "[descriptive message]" : Records file snapshots permanently in version history

git push [alias] [branch] : Uploads all local branch commits to GitHub

git pull : Downloads bookmark history and incorporates changes

git fetch : Downloads all history from the repository bookmark

Use Cases and example:

First of all when we clone the repository check the .**gitignore** file. As you need to understand what business logic is because only code that makes any difference in your business logic must be added to the repository . Static content should be versioned at all.

When you are an owner of a branch and you want some else's code to be merged or added in.

Use: **git merge branch_name_that_you_wanttobeadded** (Combines the specified branch's history into the current branch)

Most of the time we are stuck at point where code gets conflicts :

Use: **git diff [first-branch]...[second-branch]** Shows content differences between two branches

Open Terminal.

Navigate into the local Git repository that has the merge conflict.

```
cd REPOSITORY-NAME
```

Generate a list of the files affected by the merge conflict. In this example, the file *styleguide.md* has a merge conflict.

```
$ git status
```

Open your favorite text editor and navigate to the file that has merge conflicts. To see the beginning of the merge conflict in your file, search the file for the conflict marker <<<<<<<. When you open the file in your text editor, you'll see the changes from the HEAD or base branch after the line <<<<<<< **HEAD. Next, you'll see =======,** which divides your changes from the changes in the other branch, followed by >>>>>>> **BRANCH-NAME**. In this example, one person wrote "open an issue" in the base or HEAD branch and another person wrote "ask your question in IRC" in the compare branch or **branch-a.**

Source: https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line

CONFIGURE TOOLING

Configure user information for all local repositories

git config --global user.name "[name]" : Sets the name you want attached to your commit transactions

git config --global user.email "[email address]" Sets the email you want attached to your commit transactions

Troubleshooting

git fetch [bookmark] Downloads all history from the repository bookmark

git diff Shows file differences not yet staged

git checkout [branch-name] Switches to the specified branch and updates the working directory

git branch -d [branch-name] Deletes the specified branch

git log Lists version history for the current branch

git show [commit] Outputs metadata and content changes of the specific commit

git reset [commit] Undoes all commits after [commit], preserving changes locally

Team CALMS squad

Contact Us

MAILING ADDRESS

705, Silicon Tower, Opp. Law Garden, Off C.G. Road,AhmedabadGujarat 380009, India

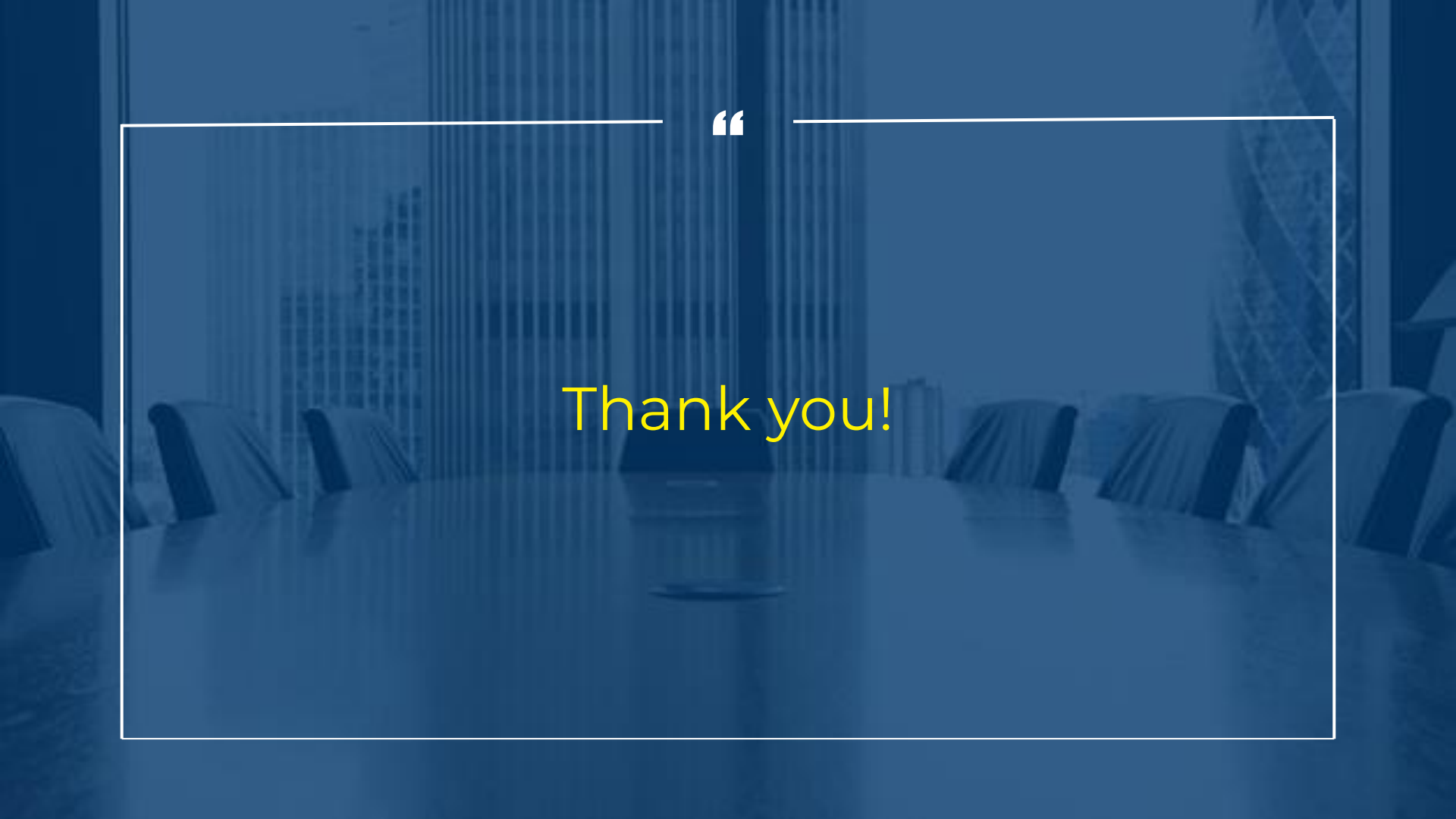
EMAIL ADDRESS

coffee@addwebsolution.com

PHONE NUMBER

India: +91 079 4005 8816
UK: +44-141-628-2177
USA: +1 305 432 2289

AddWebSolution


“
Thank you!