Git configurations

- As you read briefly in Getting Started, you can specify Git configuration settings with the git config command. One of the first things you did was set up your name and email address:
- There are 3 levels of git config; project, global and system checking
- project: Project configs are only available for the current project and stored in .git/config in the project's directory.
- global: Global configs are available for all projects for the current user and stored in ~/.gitconfig.
- system: System configs are available for all the users/projects and stored in /etc/ gitconfig.

- Create a global config with name
- $ git config user.name “akhil"
- Create a global config with email
- $ git config --global user.email “akhil10.gandla@gmail.com“
- Create a system config checking
- $ git config –list
- For ssl checking
- $ git config –global http.sslverify “false”

- **Git Config Username Command**
- To set your Git username, run the git config –global user.name command. You should specify both your first and last name but your username can be anything you want to attach to your commits.
- Your Git username does not need to be the same as your version control username, such as the one you use on GitHub.
- You’ll need to set up your identity when you first install Git. This is mandatory because every commit contains your name and your email address. You cannot change the authorship information associated with a commit once it has been created.
- There are pieces of information you need to specify: your name and email.
- Let’s configure our username values using the git config command:
- git config --**global** user.name “akhil"
- This will set our user name to akhil. All of our future commits will refer to this information. We’ve used the –global option to apply this default git config to all repositories owned by our user.

- **Git Config Email Command**
- To configure your Git email address, run the git config –global user.email command. This git config email command accepts one argument: your email address.
- git config --**global** user.email “akhil10.gandla@gmail.com"
- We can see our configuration values have been set by checking our global configuration file (git config --list):

Git annotates

- **Annotates each line in the given file with information from the revision which last modified the line**. Optionally, start annotating from the given revision. When specified one or more times, -L restricts annotation to the requested lines.
- **Use Annotate in Git**
- **annotate** command is used in git to track each line of the file based on the commit information. This command annotates from the given revision of the file. Another command exists in git, like this command, called **blame** command, but it generates output in a different format and has no backward compatibility feature like **annotate** command. The purpose of using the different options of **annotate** command and the uses of this command in git have been explained in this tutorial.

- **Different Options of annotate Command**
- **annotate** command has many options to retrieve different types of information of the files used in the GitHub repository. The purposes of some mostly used options of annotate command have been described below:

Option	Purpose
-b	It is used to display blank SHA-1 for boundary commits.
–show-stats	It is used to include additional statistics at the end of the output.
-L <n,m>	It is used to annotate the line range from n to m.
-L :<funcname>	It is used to annotate the line based on the function name.
-l	It is used to display long revision that is off by default.
-t	It displays the raw timestamp that is off by default.
-n, –show-number	It is used to display the original line number.
-e, –show-email	It is used to display the author's email in place of the name.
–date <format>	It is used to specify the date format.
–first-parent	It can be used to determine when a line is mentioned to a particular integration branch rather than to the history.

–incremental	It is used to display the result incrementally.
–encoding=<encoding>	It is used to specify the encoding used for the author names and commit summaries.
-M[<num>]	It is used to detect moved or copied lines within a file.
-C[<num>]	It is used to detect the lines moved or copied from other files that were modified in the same commit.
–ignore-rev <rev>	It is used to ignore revision.
–ignore-revs-file <file>	It is used to ignore revisions listed in the file.
-h	It is used to show help messages.