clearvision

Module 6

Collaboration Basics

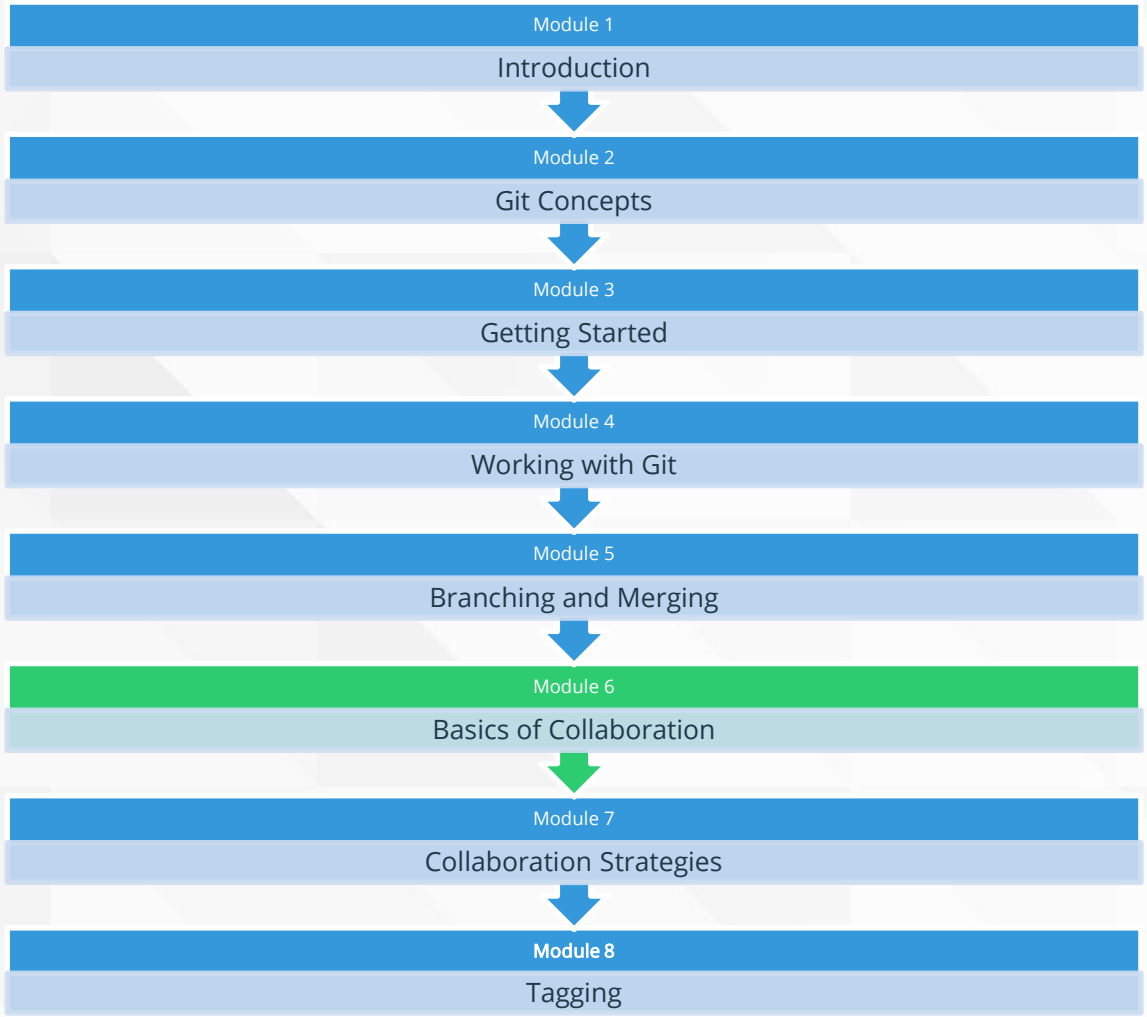
git

Module Contents

clearvision

Module Contents

- Sharing Changes
- Cloning Repositories
- Working with Remotes
- Remote Branches
- Bare Repositories
- Collaboration Strategies



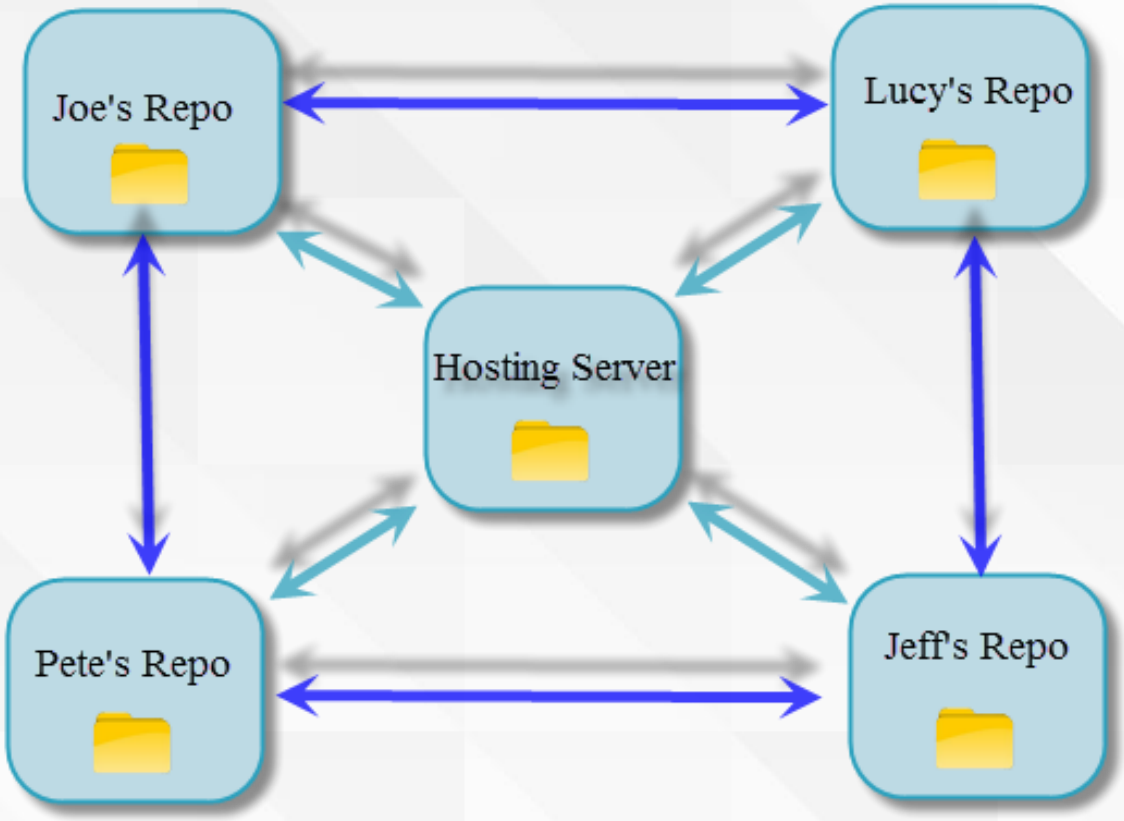
Collaboration

clearvision

- Although git is a distributed system, that does not mean distributed processes are essential.
- Git is flexible, it will fit into whatever process you wish to use
- Commonly, this will still involve a server which acts as the “master” copy or “blessed” codebase
- Users then may or may not use gits distributed nature to share changes with each other

Collaboration

clearvision



In this scenario, Lucy can configure git to talk to any other repository on the network.

- This is called a *remote*

Most commonly, commercial organisations will use a server solution to enforce process which will mean all communication goes through this location.

- This replicates the behaviour of a centralised system

git

Fetch, Push and Pull

clearvision

There are three key commands used to share data between repositories:

- Fetch
- Push
- Pull

Each of these commands can take additional arguments:

- The remote repository to contact
- The branch to use

git

Retrieving Remote Changes

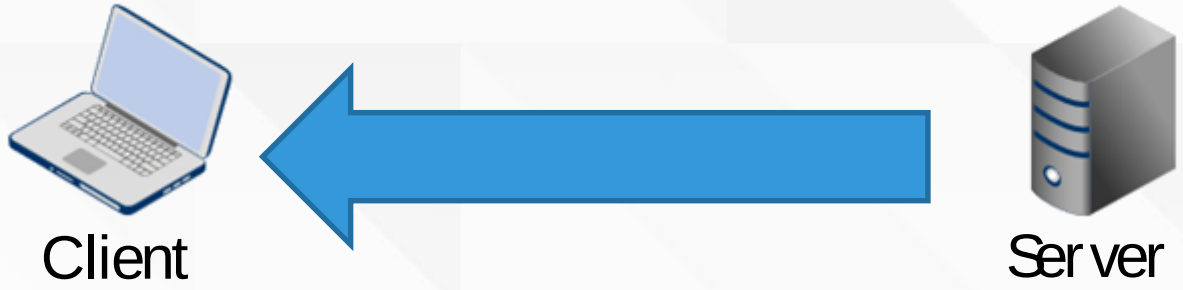
clearvision

The *git fetch* command retrieves all objects and their associated metadata that you do not yet have in your local repository.

The *git pull* command is essentially a wrapper.

This command runs a "git fetch" command, followed by a "git merge" command. That is to say, the pull command will automatically merge any fetched objects from your remote branches on to your local branches.

Without additional configuration, this requires that both branches have the same name



git

Sharing local changes

clearvision

The *git push* command is in effect the opposite of *git fetch, t*his command will send any changes that exist locally – but not remotely – to a remote repository.

Push will effectively do a merge in the remote repository, this will fail if:

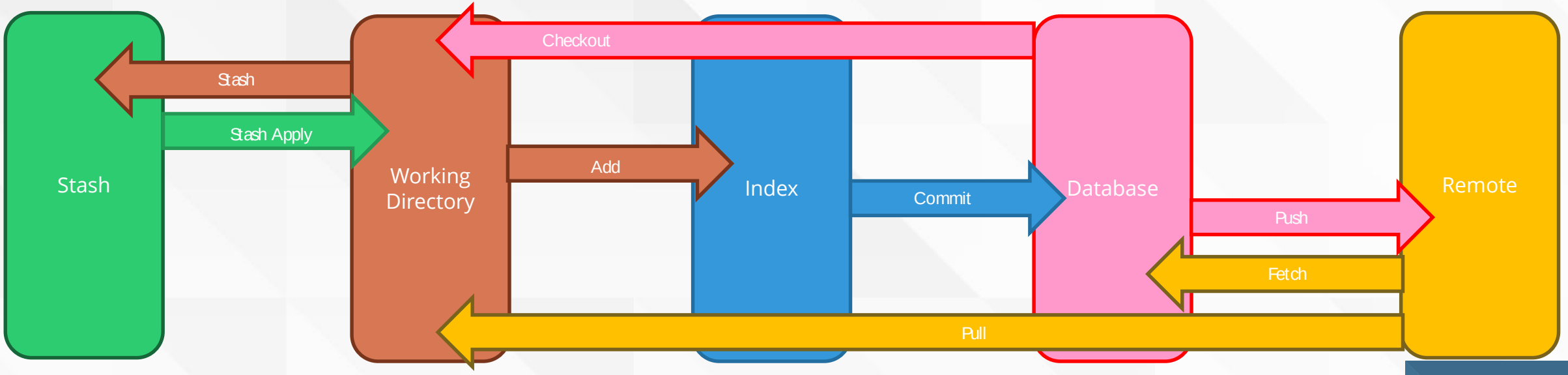
1. It is not a fast forward merge
2. Performing the merge would force the remote user to update their working directory (because your target branch is currently checked out)

git

The Complete Workflow

clearvision

The commands we use at which time depends on the areas of the git repository we want to work with, if we consider the entire process from a more holistic view, it looks something like this:



git

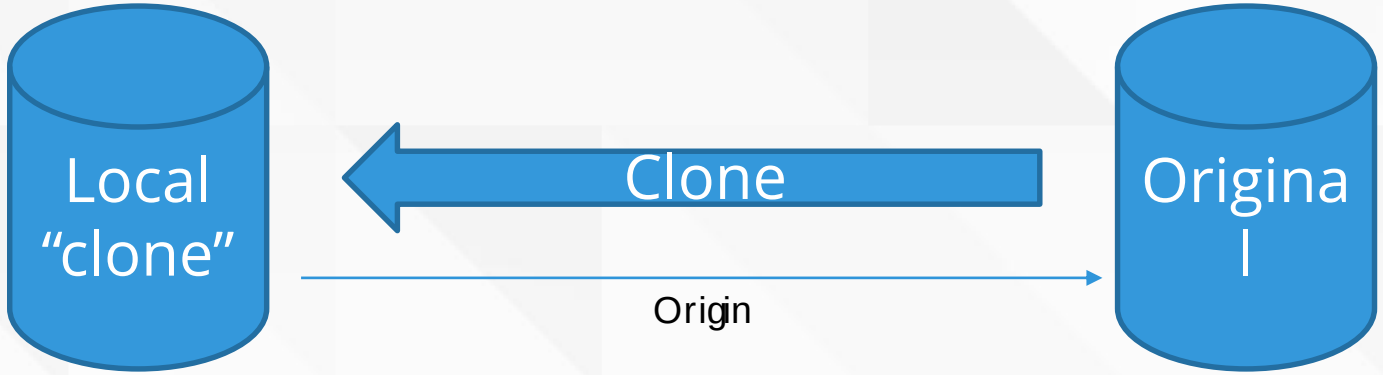
Cloning

clearvision

In most cases, defining where to pull from and push to is made easy, when you start working on a project, the first action you will take is to *clone* it.

A clone is a copy of a repository and contains all the objects from the original repository

By default, a new clone includes a reference back to its parent repository. This reference is in fact a remote called "origin".



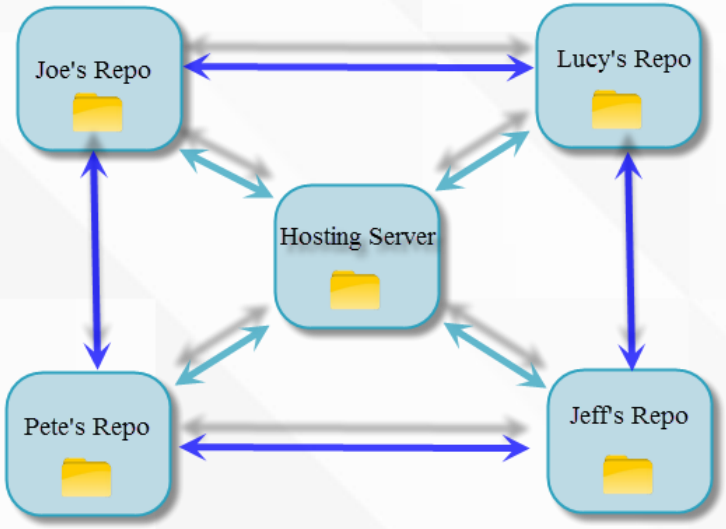
git

Cloning

clearvision

You are not limited to sharing changes with the *origin* unless your process demands it.

Use the command *git remote* to access other locations



add <name> <url>	Adds a new remote called <name> with the <url>
rename <name><newname>	Renames a remote called <name> to <newname>
set-url <name> <url>	Overwrites location of remote <name> with <url>

git

Remote Branches

clearvision

Remote branches are references that point to the commit that represents the last communication between your own repository and the remote repository.

When a repository is cloned, Git creates a remote branch in the clone associated with the branch that is currently checked out in the original repository. (usually master)

Remote branches are references that point to the commit that represents the last communication between your own repository and the remote repository.

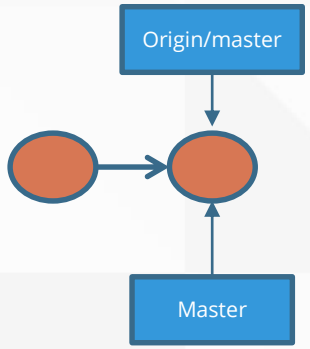
Do not be fooled by the name! remote branches exist locally within your own repository.

git

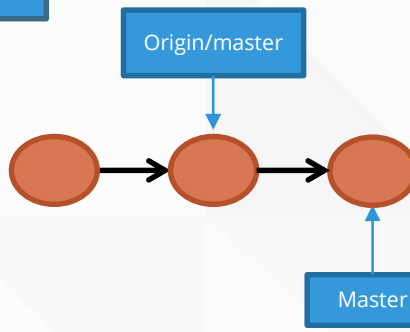
Remote Branches

clearvision

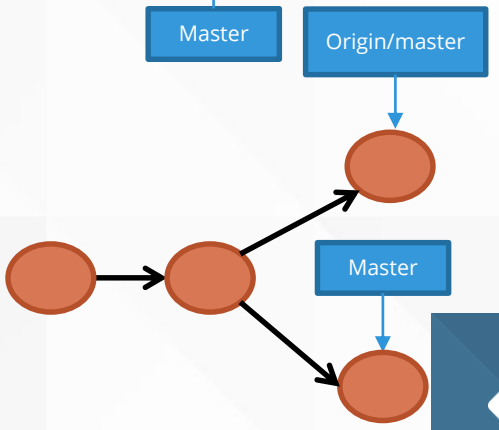
- Both references may start out in the same place



- When you merge in a local feature branch, your master branch is updated
  - The remote branch is unaffected



- But when you fetch other people changes, the two can diverge
  - At which point you may need to merge them together (which *git pull* does for you)



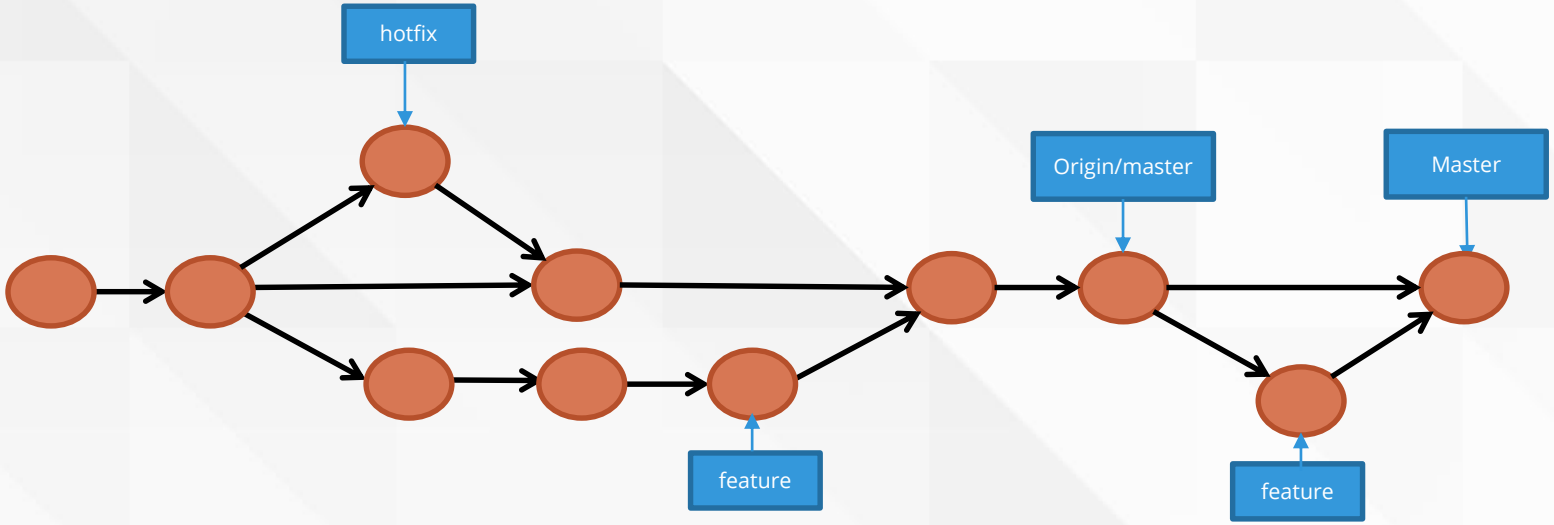
git

Remote Branches

clearvision

Remember that all branches are nothing more than a reference which track changes

Remote branches remain stationary unless changes are made in the remote repository



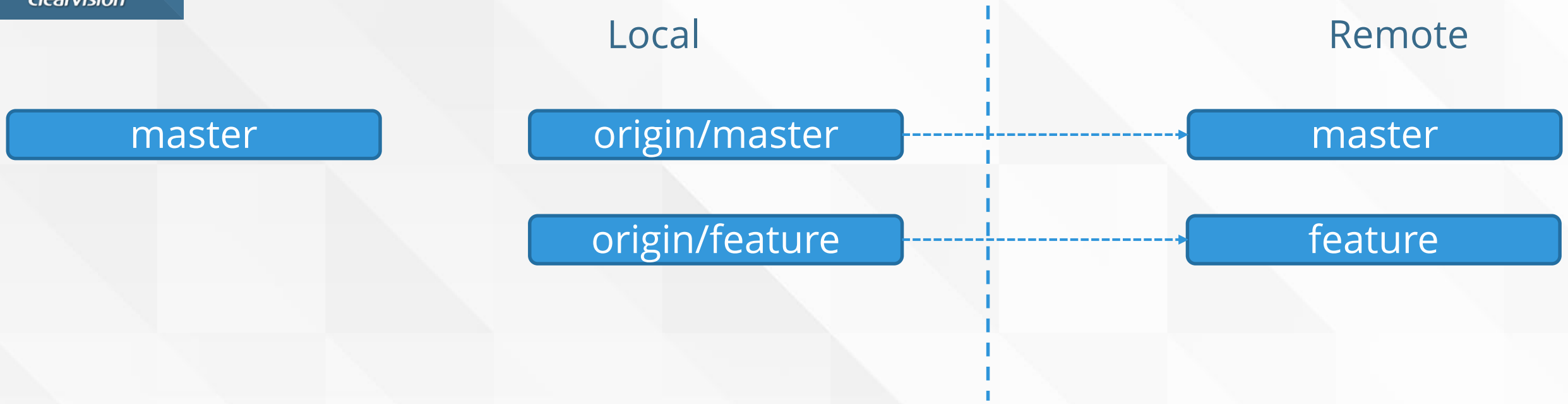
NOTE - Because Git uses remote branches internally to keep your repository in-sync, you cannot manually merge or commit changes onto a remote branch as it will become out-of-sync with the branch in the original repository it is meant to be tracking.

They do not update in real time – use *git fetch*

git

Remote vs Local

clearvision

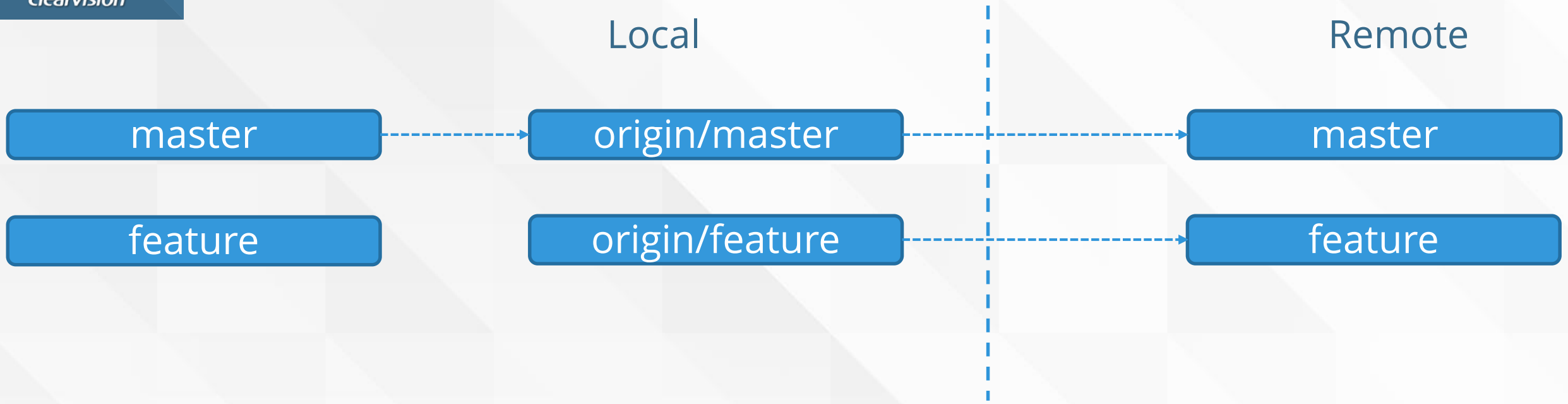


Note that we can have a remote branch, without having a local version – this can be merged into our own branches at any time.

git

Working On Remote Branches

clearvision



You can work on the contents of a remote branch by creating a local branch

- To simplify, this should have the same name.

- *Git checkout –b feature origin/feature*

git

Tracking Branches

clearvision

Git's default branch mapping for pull actions uses branches with the same name

i.e origin/master -> master

If you want to define a relationship in the other direction:

- *git branch -u origin/master*
  - *master -> origin/master*

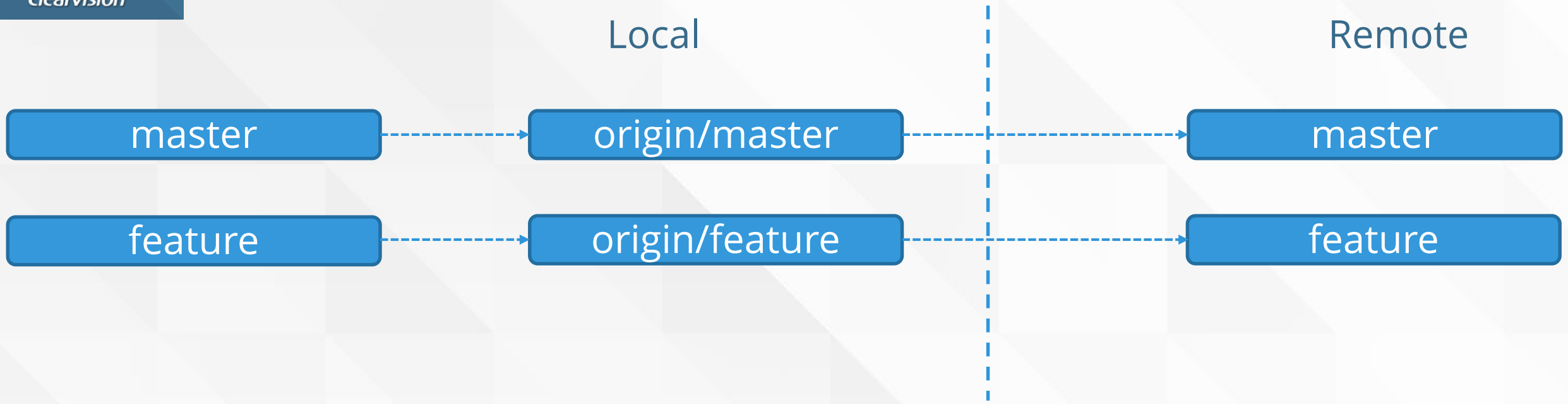
Without this relationship you will need to define where to push to

- *git push origin feature*

Once defined, the local branch is said to be "tracking" the remote branch.

Working On Remote Branches

clearvision

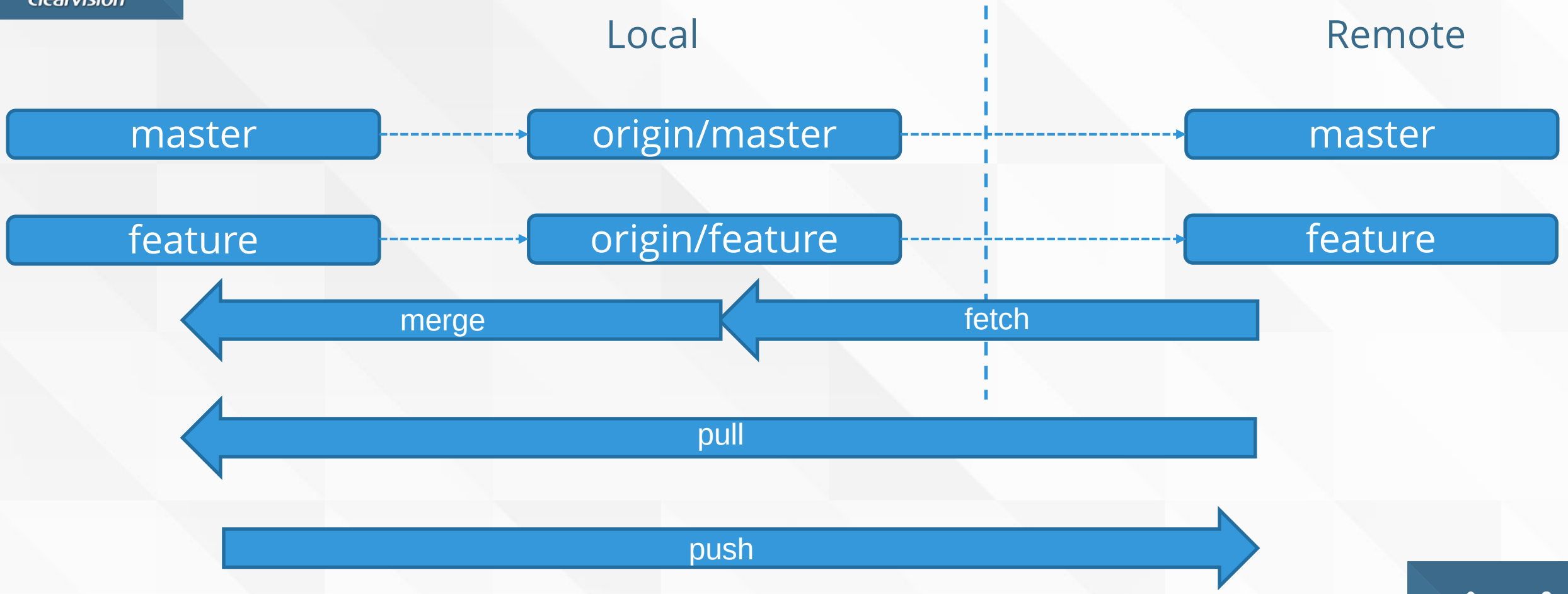


This tracking relationship defines the default settings for push actions

git

Working On Remote Branches

clearvision



git

Fetching , Pulling and Pushing - Recap

clearvision

- Fetch <remote> <branch>
  - Updates remote branches with changes from the remote repository
  - Use this when you want to test or investigate changes before merging into your own code
  - Defaults to fetching all changes from all remotes
  - <remote> specifies to only update from the specified remote
  - <branch> specifies a single branch from <remote>. If <remote> is not provided, defaults to origin

- pull <remote> <branch>
  - Updates remote branches and merge into local branches
  - Defaults to origin when no options are provided and assumes branches of a matching name
  - Can be set up to default to branch specified by configuration (created automatically by checkout –b)
  - <remote> pulls all changes from <remote> to all branches with a matching name
  - <branch> specifies which branch to pull from – names do not need to match

Fetching , Pulling and Pushing - Recap

clearvision

- push <remote> <branch>
  - Updates remote repository with your own changes
  - Defaults to use upstream relationship – if no upstream branch is set the command will fail
  - <remote> specifies which remote to push to
  - <branch> specifies which branch to update in <remote>. If <remote> is not provided, command will fail
  - Push will fail if:
    - <remote> contains changes not present in local repo
    - <branch> is currently checked out by the owner of <remote>
- fetch, push and pull all support the --all option
  - Specifies to update to or from all remotes using matching branch names
- If in doubt, use <command> <origin> <branch> to avoid any mistakes
  - E.g. *git pull origin feature*

git

clearvision

Lab Exercise

git

clearvision

End of Module

git