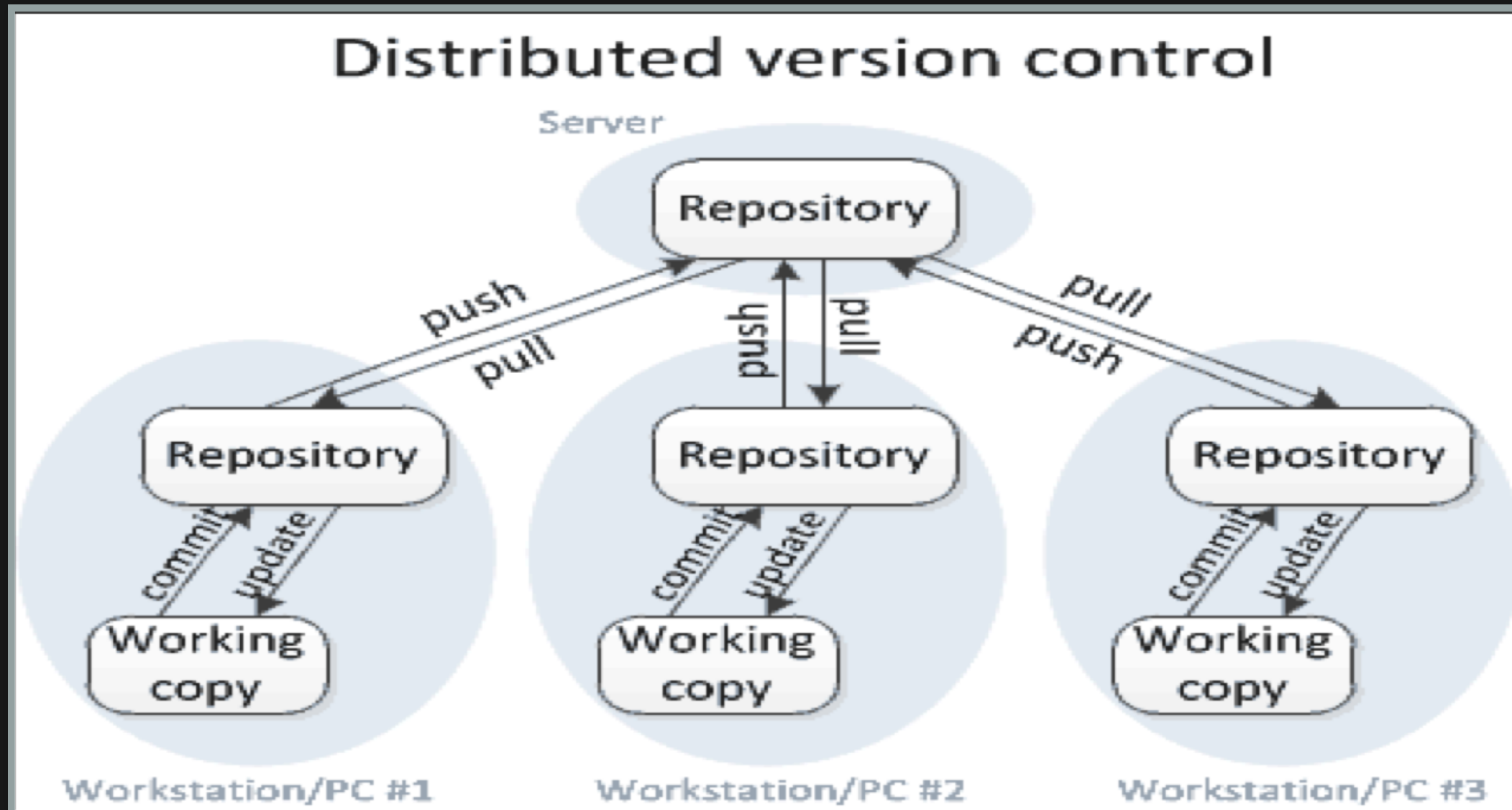
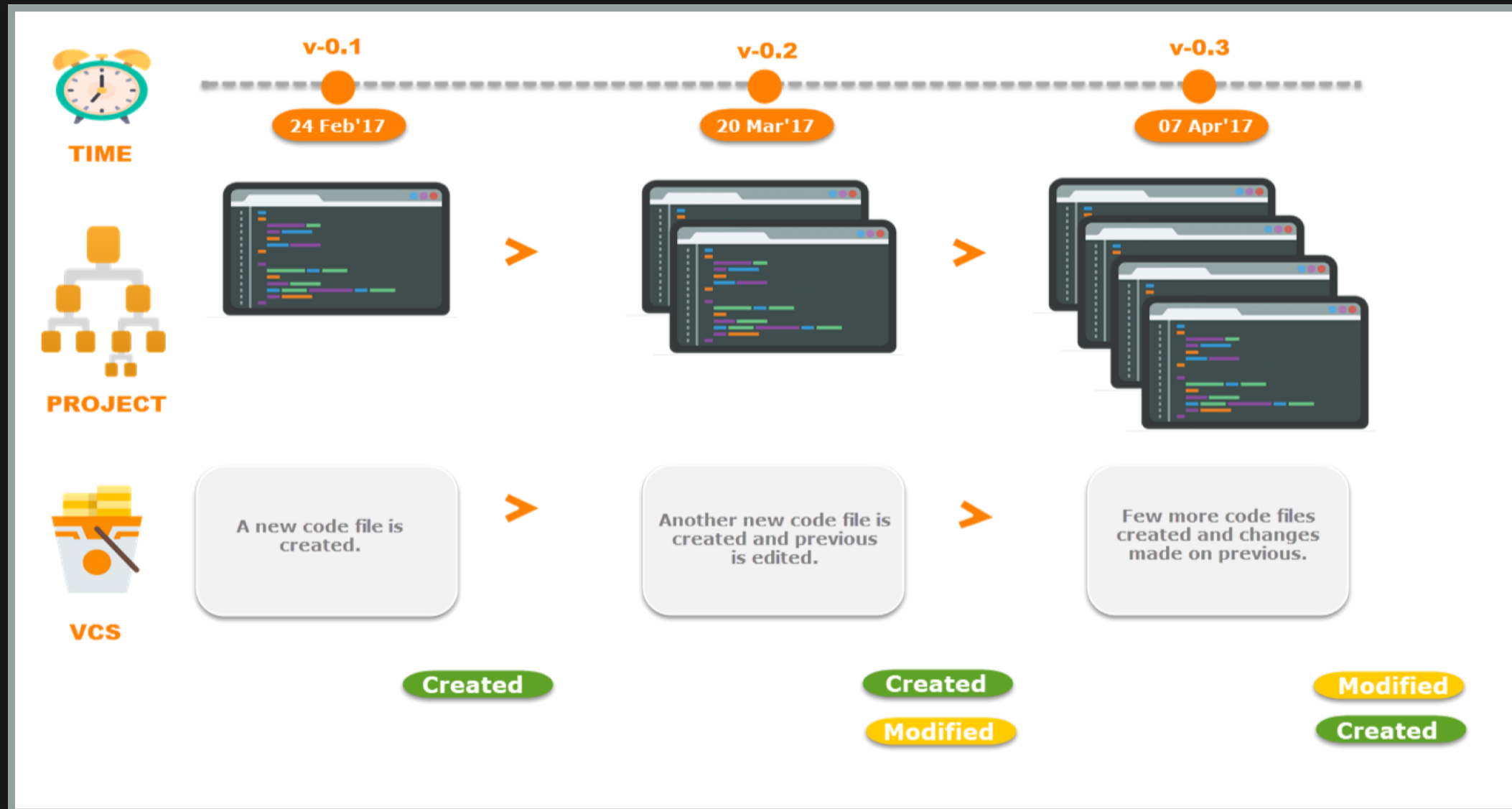




VERSION CONTROL SYSTEM



VERSION CONTROL SYSTEM



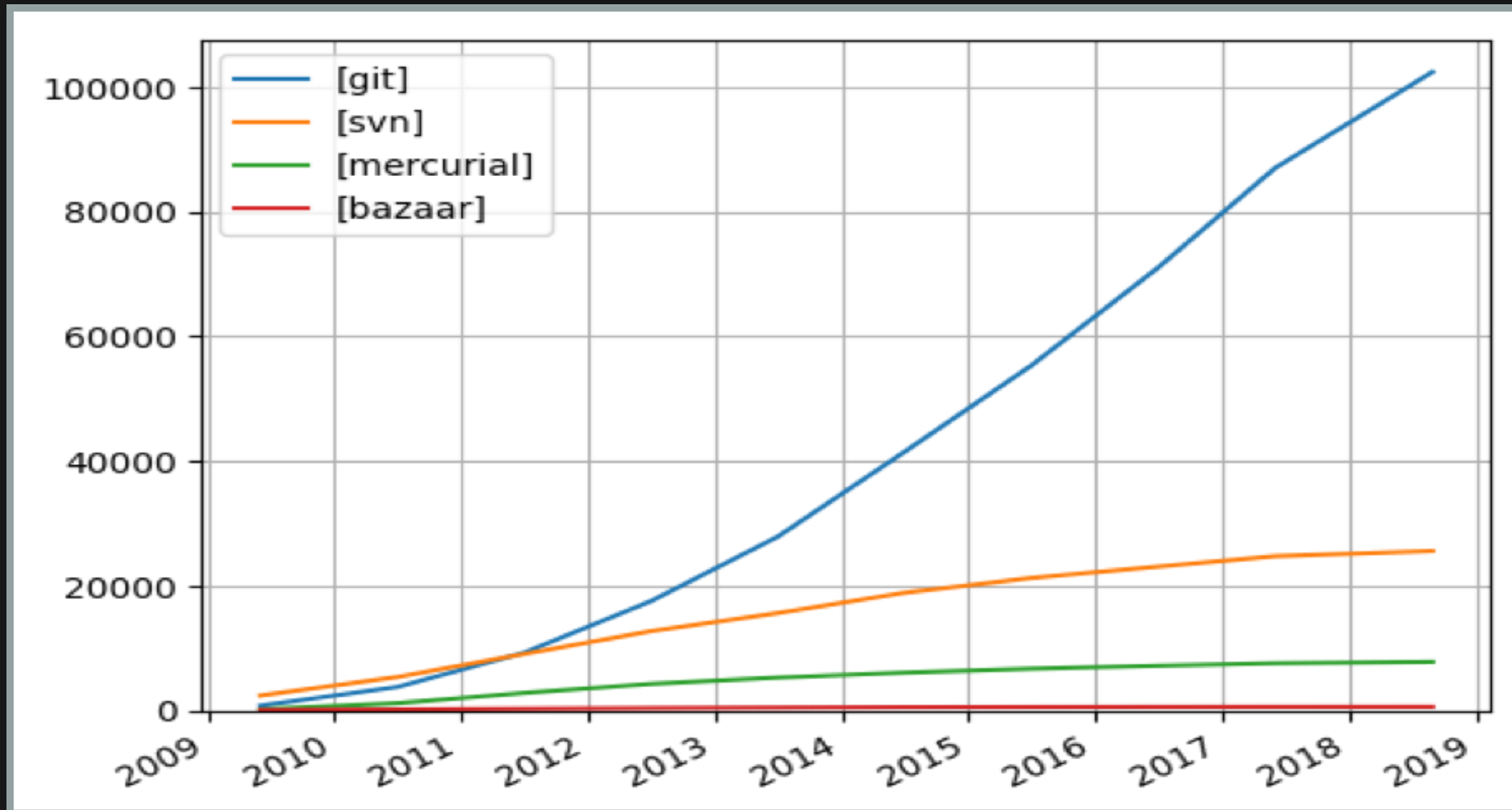
WHY VERSION CONTROL?

- Multiple Developers working on same code base
- Central Cloud Backup
- Incremental Progress, Know which version is live
- Branching -> multiple branches for different environments
- Change History? Can blame/praise the Commit/Developer

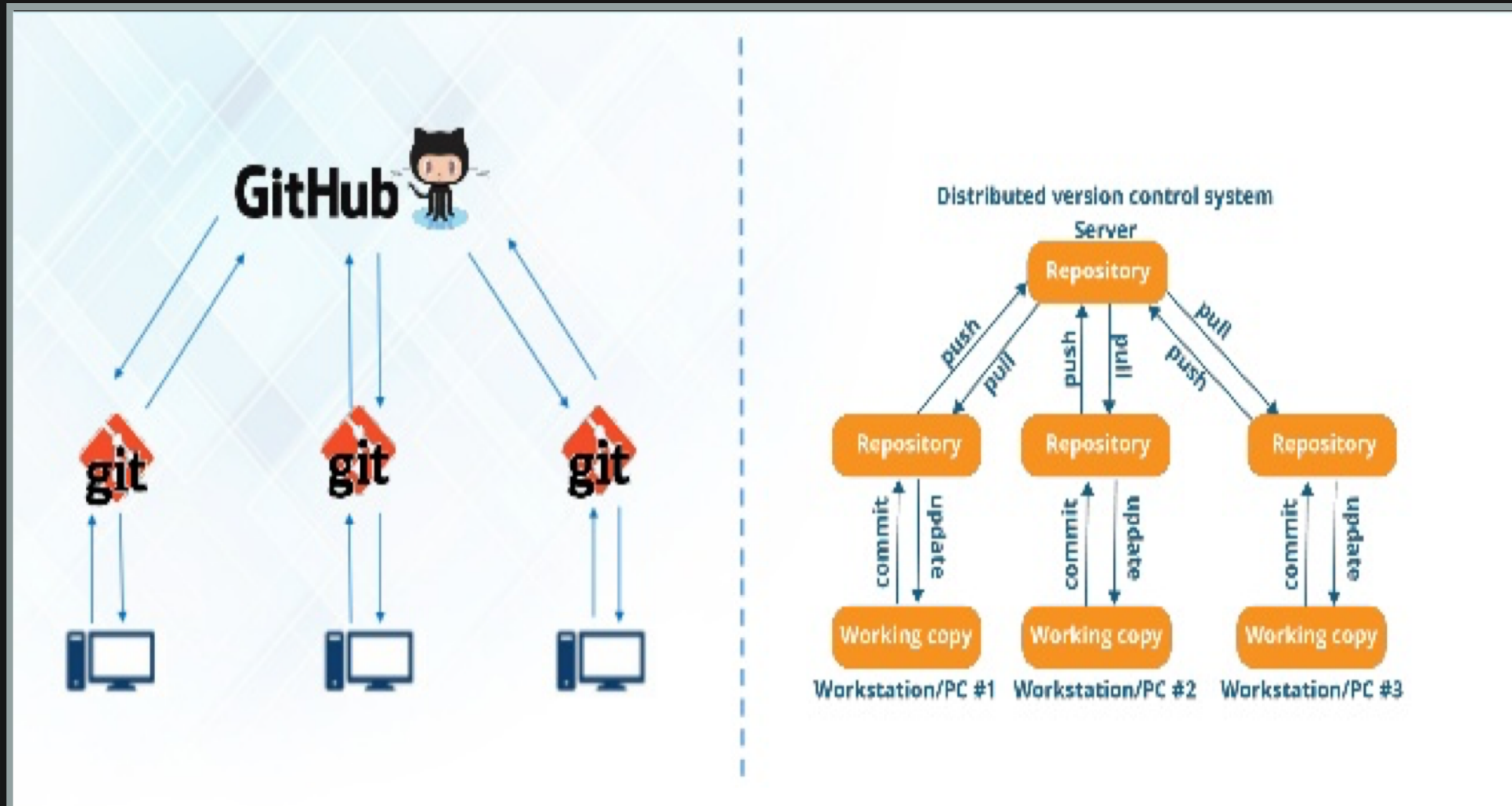
VERSION CONTROL TOOLS



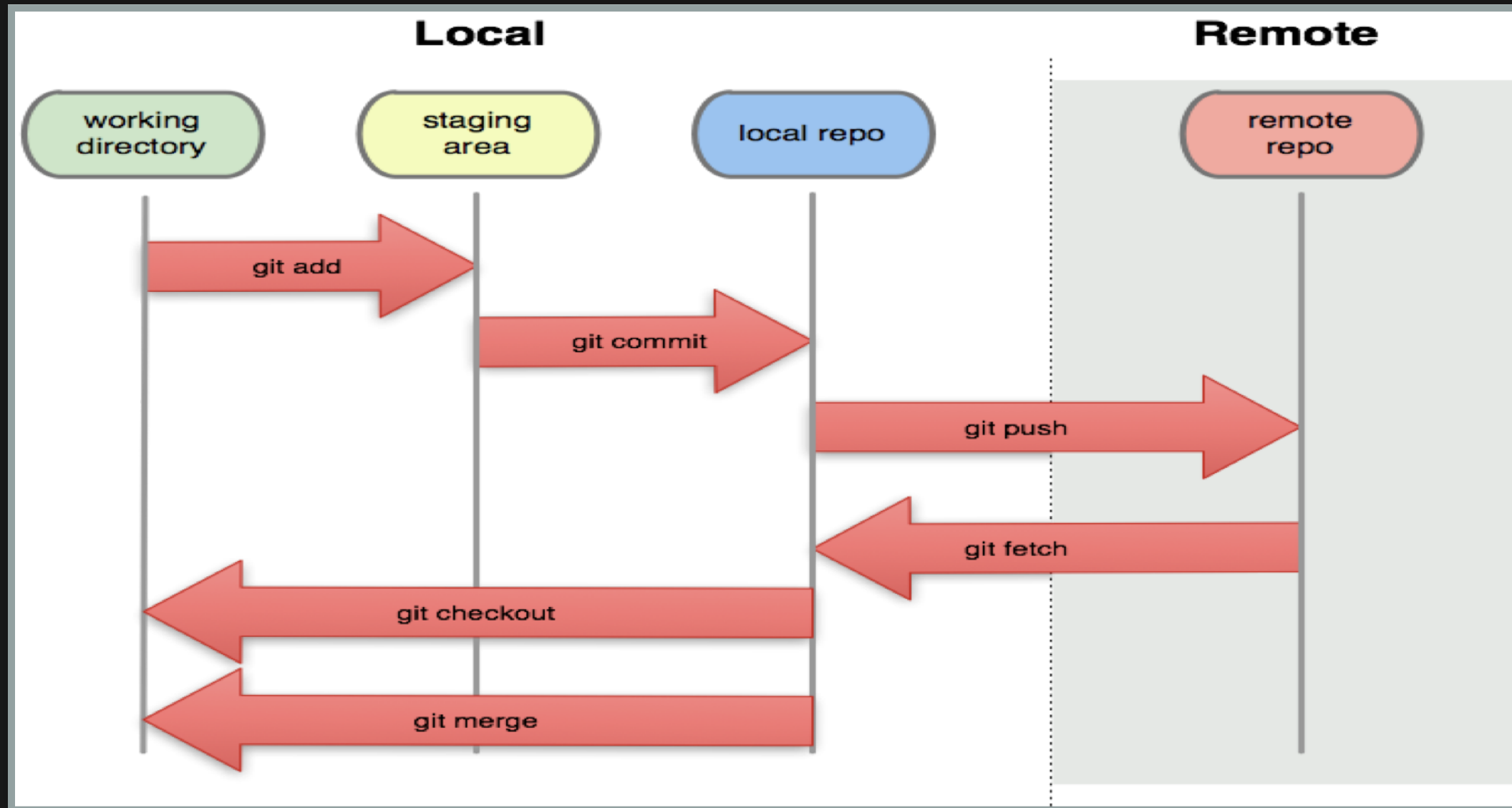
VERSION CONTROL TOOLS



GIT & GITHUB



GIT STAGES



INSTALL GIT

```
$ sudo apt-get install git-all
```

OR

```
Install Git: https://git-scm.com/downloads
```

GIT REPOSITORY

- A single project containing the code base of your application/service
- Can be
 - Private: Only certain people with access can see the repo
 - Public: Anyone can see the repo
- In your local git repository, a .git folder is created, it basically tracks all the changes/versions of your codebase.
- There is a local repository & a remote repository, you make changes to local repository and push them to remote.

LAB - GIT REPOSITORY

1. Creating Github Account
2. Create & Cloning a Repository
3. Add SSH key

CREATING GITHUB ACCOUNT

Go to github.com and Signup, your username will be your github address e.g.

```
github.com/kaootali
```

ADD SSH KEY

Configure your git with your config so that your commits can have this configuration.

```
git config --global user.name "Your Name"  
git config --global user.email "you@example.com"
```

For pushing changes to Github/cloning a private repo, it needs to authenticate you, so for every push, you will have to enter your username & password which can cause problems. So to solve that we can add an SSH key of your PC, so that Github knows whenever you are trying to clone/push changes, that this PC is secured and has SSH(secure shell).

Generate SSH key locally

```
ssh-keygen -t rsa -b 4096 -C "you@example.com"  
eval $(ssh-agent -s)  
ssh-add ~/.ssh/id_rsa  
cat ~/.ssh/id_rsa.pub
```

Copy the SSH key and then we need to go to the GitHub account and click on

```
Profile -> Settings -> SSH & GPG Keys -> New SSH Key -> Paste.
```

This will add an SSH key to your account, to test if it is working enter

```
ssh -T git@github.com
```

CREATE & CLONING A REPOSITORY

Click the + at the top right, New Repository, Enter Repository Name, Description, Chose Public and initialize with a README. Your Repository will be created with following link

```
github.com/<username>/<repo-name>
```

Once the repo is created, you can clone it, by clicking the Clone or Download, and copying the link. Go to bash

Using https

```
git clone https://github.com/<user-name>/<repo-name>.git
```

Using ssh

```
git clone git@github.com:<user-name>/<repo-name>.git
```

GIT COMMANDS

- Tells the changes made in the local repo only:
 - `git status`
- Shows file differences not yet staged:
 - `git diff`
- Add the file to staging:
 - `git add`
- Remove the file from staging :
 - `git reset`
- Commit to local Repository:
 - `git commit`
- Push the local repository changes to remote repo:
 - `git push`

LAB - GIT COMMANDS

1. Make changes to your project
2. Checking File Differences
3. Commit changes to your local repository
4. Resetting Differences

MAKE CHANGES TO YOUR PROJECT

You should have cloned the repo created in previous Lab.

```
cd <repo-name>  
ls
```

You will be seeing the README.md file. So We will add a new file in the repo.

```
touch a.txt
```

Edit this file either in vim or any text editor. Now see the status of your repo. It should be like

```
git status  
On branch master  
Your branch is up to date with 'origin/master'.  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
    a.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

meaning a file is added that is currently untracked i.e. it is not present in git history

CHECKING FILE DIFFERENCE

Add this file to staging area

```
git add a.txt
```

Now check git status, it should return

```
git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   a.txt
```

It means file is in staging, but it should be committed to local repo. Now edit the README.md file, add a bit description like "This is a test repo" or anything else.

Now check status

```
git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   a.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md
```

It shows that new file a.txt is yet to be committed, and README.md has been modified but it needs to be staged for commit first.

Now see the differences that have been made

```
git diff
diff --git a/README.md b/README.md
index 5297f8f..2b26b81 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
-# test-git
\ No newline at end of file
+# test-git
+This is a test repo
```

It shows that `-# test-git` was removed and `+# test-git`, `+This is a test repo` were added.

Add this file to staging. And then check status

```
git add README.md
git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
    new file:   a.txt
```

Check differences in staging area

```
git diff --staged
diff --git a/README.md b/README.md
index 5297f8f..2b26b81 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
-# test-git
-\ No newline at end of file
+# test-git
+This is a test repo
diff --git a/a.txt b/a.txt
new file mode 100644
index 0000000..557db03
--- /dev/null
+++ b/a.txt
@@ -0,0 +1 @@
+Hello World
```

It shows both the differences of README.md as well as a.txt.

COMMIT CHANGES TO LOCAL REPOSITORY

Now we will commit the changes in the local repository.

```
git commit -m "Add a.txt and update README"
git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
    (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

It shows that your local branch is ahead of origin/master by 1 commit that we just did. And the working tree is clean i.e. no other file has been changed or is in staging area

RESETTING DIFFERENCES

Now we have two changes in a commit. We want to reset these differences. There are 3 possible options.

METHOD 1: MIXED

(Default) Remove the commit from local repo, and also remove changes from staging area

```
git reset --mixed HEAD^
git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  a.txt
```

Notice changes have been removed from local repo as well as staging area. So we will need to add and commit files again.

METHOD 2: SOFT

Remove the commit from local repo, but keep the files in staging area

```
git add .
git commit -m "Add a.txt and update README"
git reset --soft HEAD^
git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
    new file:   a.txt
```

Notice that files are still in staging area but have been removed from local repo, now you will just need to commit them again.

METHOD 3: HARD

Remove the commit from local repo, and staging area and even change the local files.

```
git commit -m "Add a.txt and update README"
git reset --hard HEAD^
git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Notice the changes have been removed from local repo, as well as the working tree i.e. the files are now in initial state that was before your working.

PUSH AND PULL

- Push
 - Transfer commits from your local repository to a remote repository
 - Share modifications with remote team members
 - Commands
 - `git push origin`
- Pull
 - Download content from a remote repository and update the local repository
 - Get modification from remote team members
 - Commands
 - `git pull origin`

LAB - PUSH AND PULL

1. Push to Remote Repository
2. Pull from Remote Repository

PUSH TO REMOTE REPOSITORY

You will create a new file in the master branch and then push the changes to the remote repository

Create a new file:

```
$ vi push-example.txt
```

Commit new file

```
$ git add push-example.txt  
$ git commit -m "new file created"
```

Your current changes are in the local repository to verify that go to Github and verify that push-example.txt does not exist. Push changes from local repository using the command:

```
$ git push origin master
```

PULL FROM REMOTE REPOSITORY

You will create a file on Github and then pull those changes in your local repository

Go to your Github repository that you created.

Click **Create New File**

Create a File:

The screenshot displays a GitHub repository interface. At the top, a summary bar shows '1 commit', '1 branch', '0 releases', and '1 contributor'. Below this, a navigation bar contains buttons for 'Branch: master', 'New pull request', 'Create new file' (highlighted with a black box), 'Upload files', 'Find File', and a green 'Clone or download' button. The main content area shows a commit history for 'umermunir' with the message 'Create Readme.md'. The latest commit is identified by hash '6173ec9' and is dated '4 minutes ago'. Below the commit list, a file named 'Readme.md' is shown with a 'Create Readme.md' button and a timestamp of '4 minutes ago'. At the bottom, another 'Readme.md' file entry is visible with an edit icon.

Commit new file:

DiceLabs / pull-example.txt


Cancel

<> Edit new file

Preview

Spaces2No wrap

1 This is pull example file



Commit new file

Pull Example

Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file

Cancel

Next you will pull the latest changes from remote to local repository:

```
# verify you are on master branch  
$ git checkout master  
$ git pull origin master
```

List contents of your local repository, you will see the newly created file.

GITHUB PAGES

- Static site hosting service
- Host your
 - Project website
 - Organization website
 - Personal website
- Does not support server side code such as PHP, Python or Ruby
- GitHub Pages are publicly available on the internet, even if their repositories are private.

ASSIGNMENT

HOST YOUR RESUME ON GITHUB PAGES

INSTRUCTIONS

- Create your Resume Repository on GitHub
- Host it through GitHub Pages