



# Essential Git For Developers

By:

Adam Culp

Twitter: @adamculp

<https://joind.in/14744>



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- About me

- PHP 5.3 Certified
- Work at Zend Technologies
- Organizer SoFloPHP (South Florida)
- Organized SunshinePHP (Miami)
- Long distance runner
- Judo Black Belt Instructor



Northeast  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- **Fan of iteration**

- Everything requires iteration to do well: (practice makes perfect)
  - Long distance running
  - Judo
  - Development
  - Avoid project managers
  - Version Control!



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

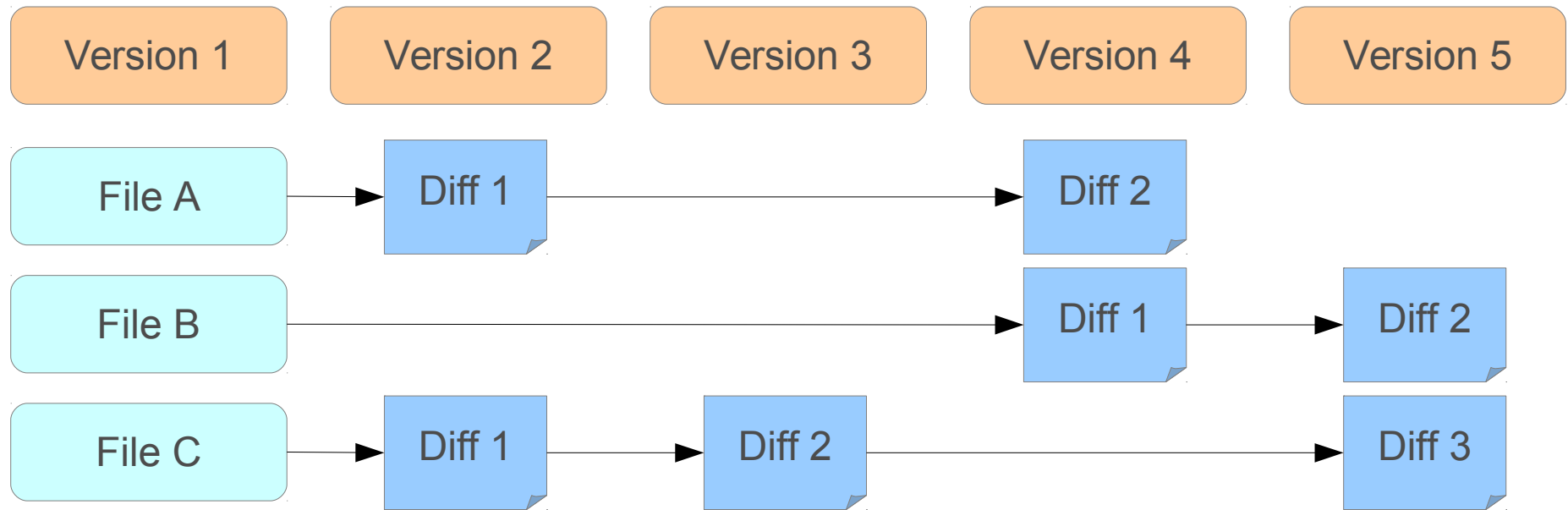
- Why use Git?
  - No centralization
    - No central server (unless desired)
  - Each clone = full repository
    - Git tracks state, history, and integrity
  - Branching and Merging work
  - Fast
    - Local vs Remote
    - Only one .git directory
  - Files to be committed are “staged” first
  - Free and Open Source
  - Flexible workflow



Northeast  
PHP CONFERENCE  
@NEPHP

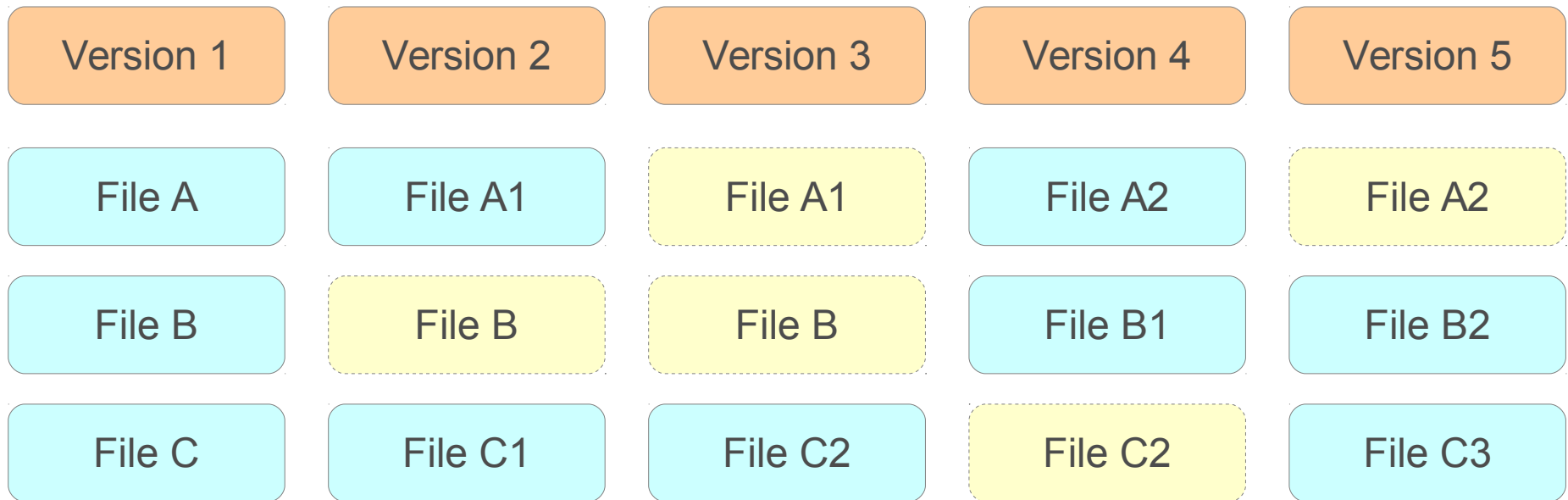
# Essential Git For Developers

- How Others Look At Data.
  - As files and the changes made to each file.



# Essential Git For Developers

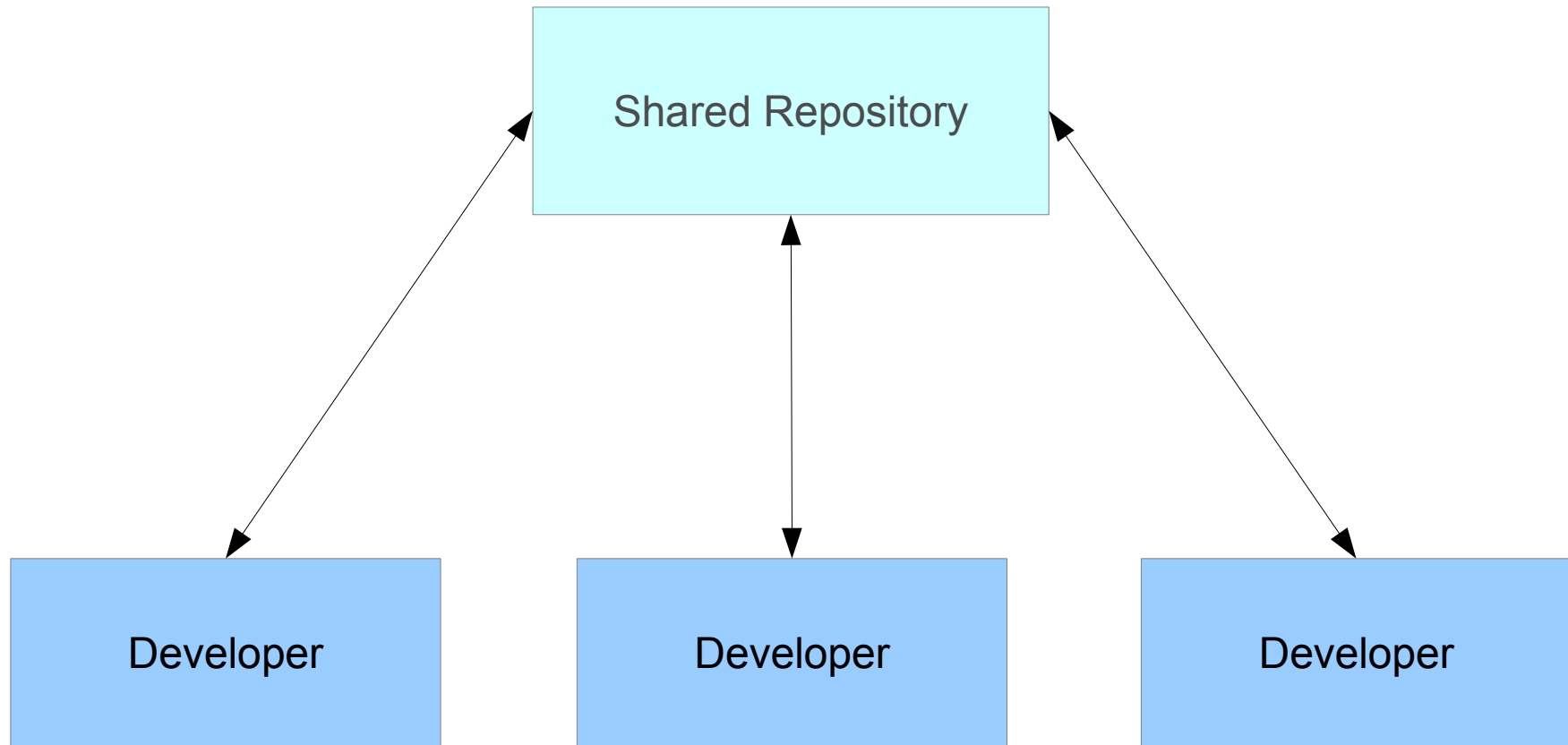
- **How Git Looks At Data.**
  - As whole files, not files + diffs.



Green means whole file, yellow means pointer to previous whole file.

# Essential Git For Developers

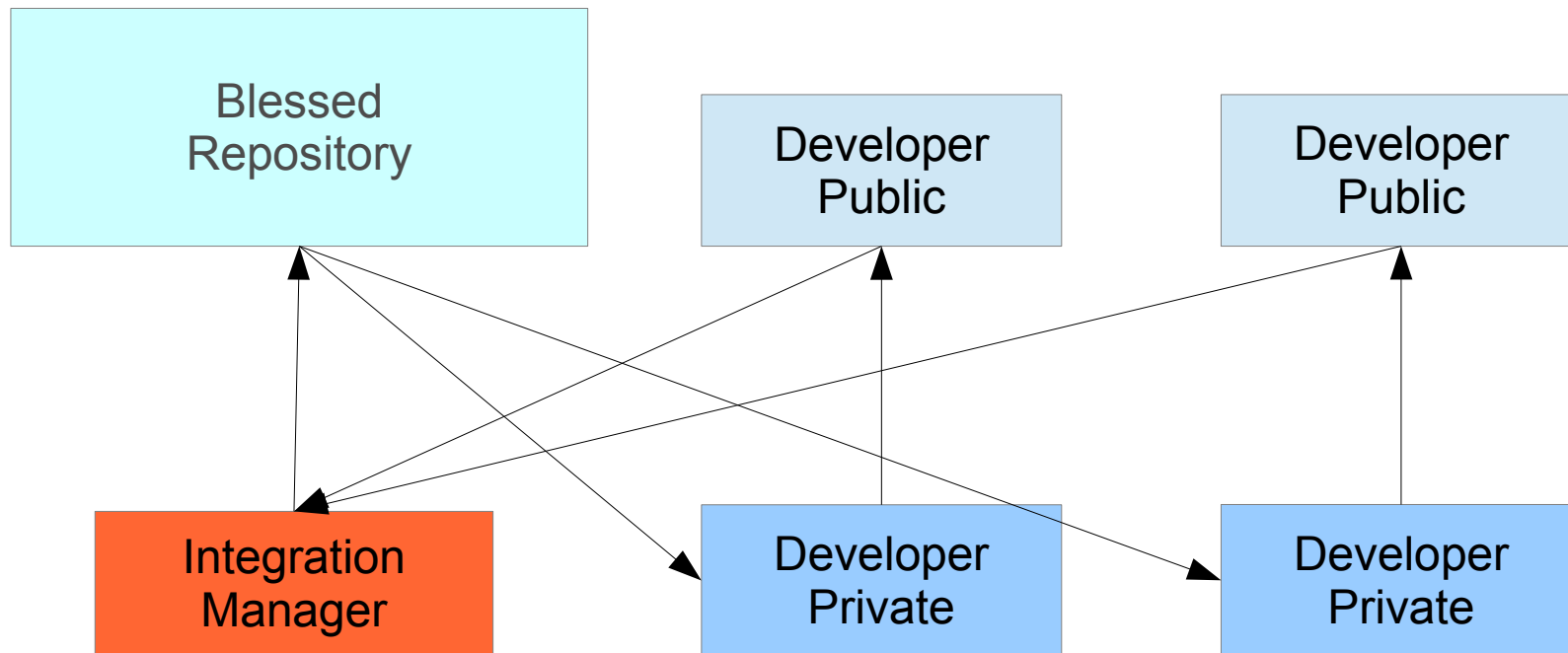
- Subversion-Style Workflow





# Essential Git For Developers

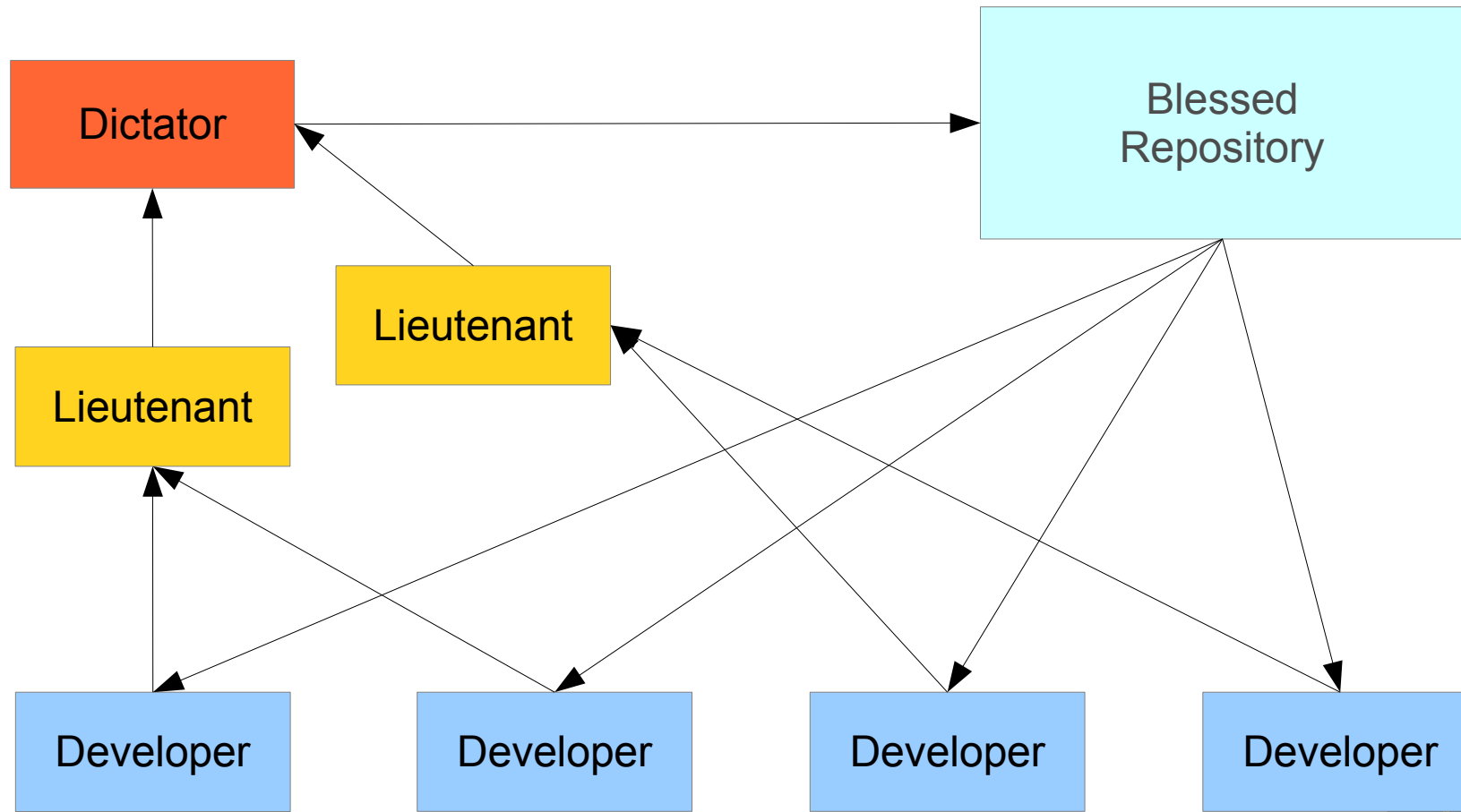
- Integration Manager Workflow





# Essential Git For Developers

- Dictator and Lieutenants Workflow



# Essential Git For Developers

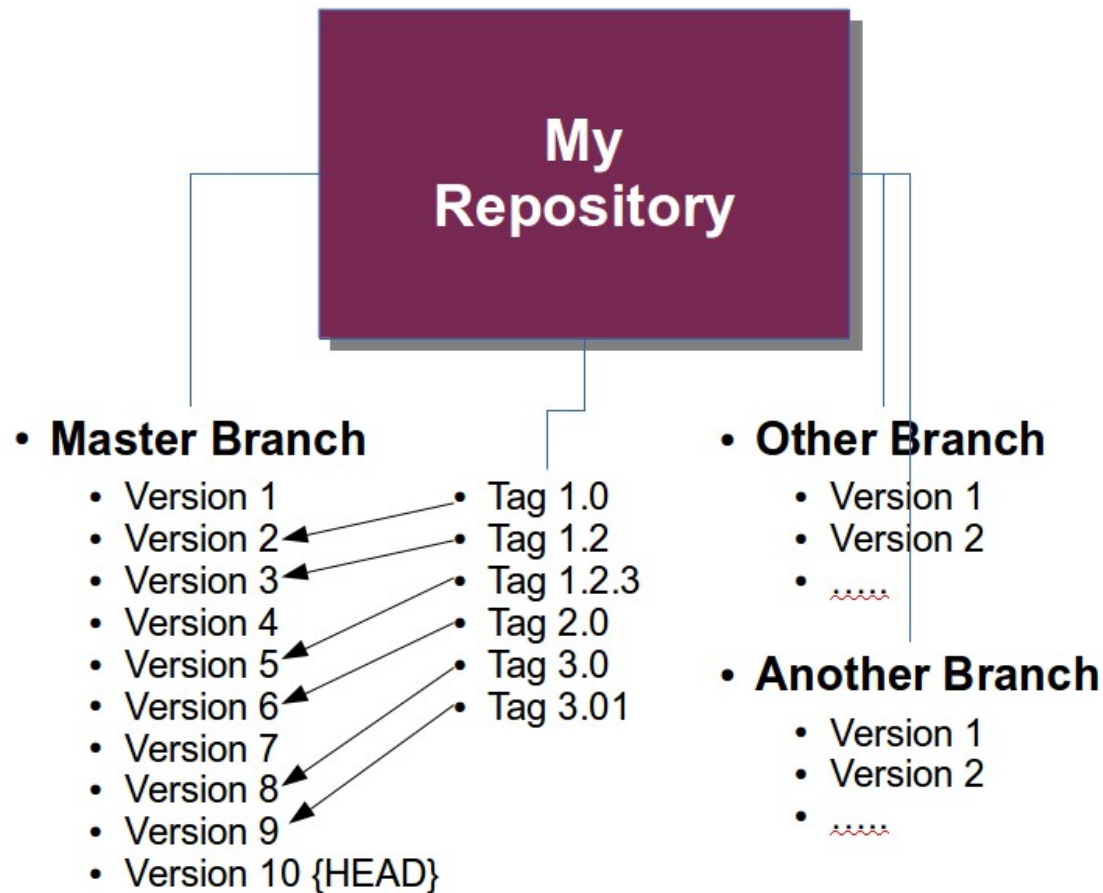
- **Single Developer**
  - One repository, everything in one basket.
    - Remember to backup

Developer  
Local  
Repository



# Essential Git For Developers

- Each 'git clone' == full repository



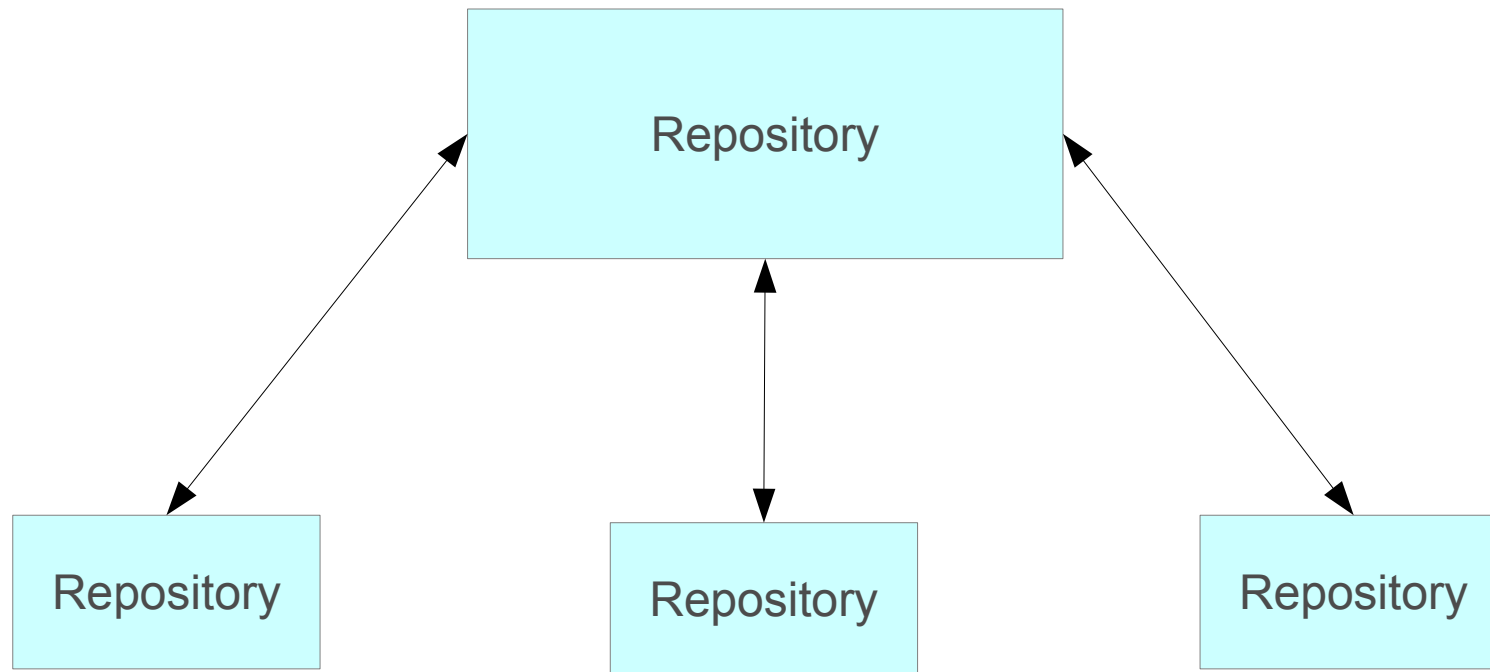
Origin = <https://path/to/repo>



**Northeast**  
PHP CONFERENCE  
@NEPHP

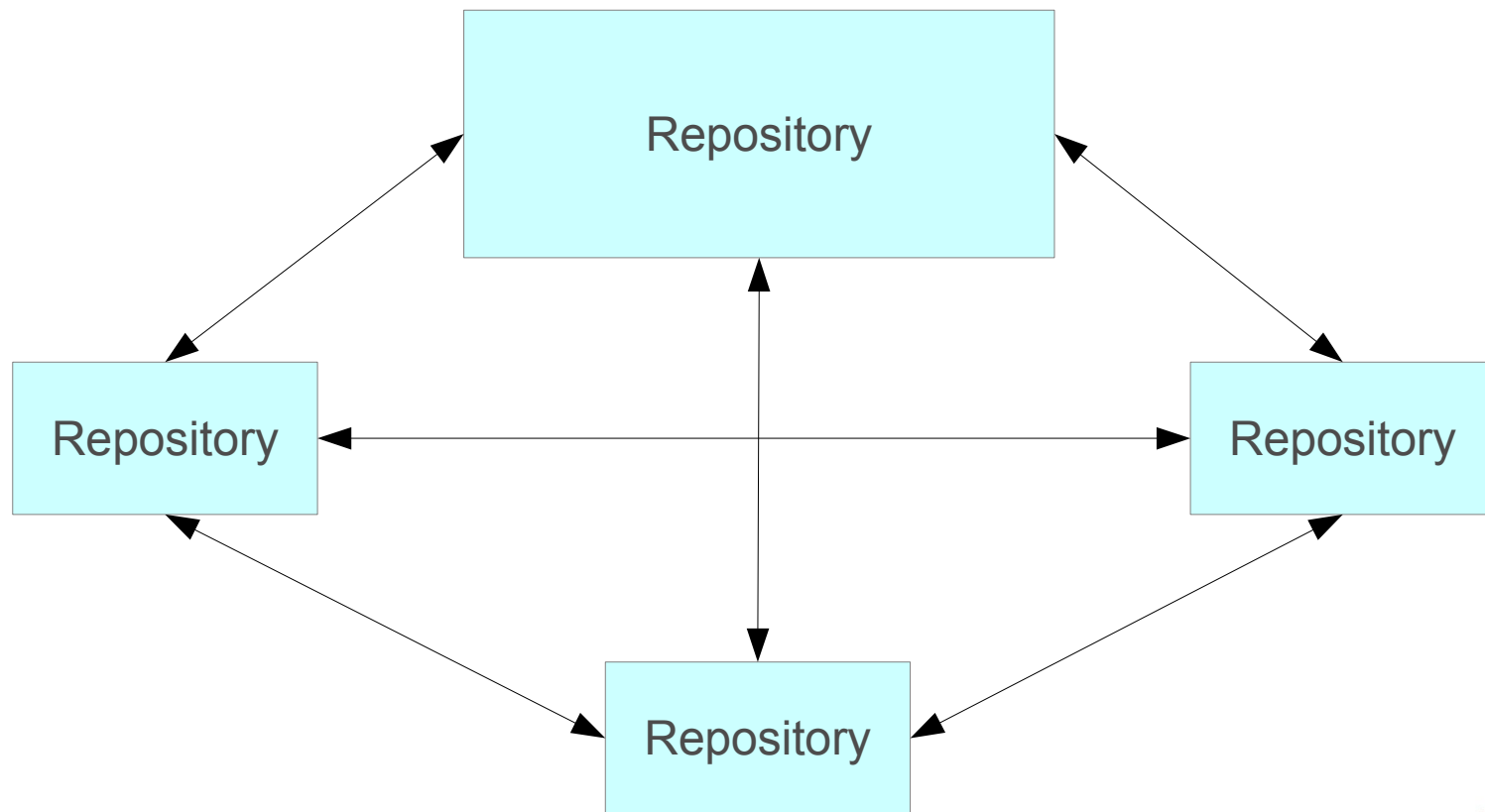
# Essential Git For Developers

- What is actually going on?
  - A bunch of repositories!



# Essential Git For Developers

- But it could be:
  - Repositories can connect in all directions.



# Essential Git For Developers

- **Most common commands**

- `git config`
- `git init`
- `git clone`
- `git status`
- `git add`
- `git commit`
- `git log` or `show`
- `git branch`
- `git checkout`
- `git merge`
- `git pull` or `push`



# Essential Git For Developers

- **Help!**

- Adding '-h' to any command will return help on usage.

```
aculp@aculp-laptop:/sample_app$ git add -h
usage: git add [options] [--] <filepath>...

-n, --dry-run          dry run
-v, --verbose          be verbose

-i, --interactive      interactive picking
-p, --patch            select hunks interactively
-e, --edit             edit current diff and apply
-f, --force            allow adding otherwise ignored files
-u, --update           update tracked files
-N, --intent-to-add    record only the fact that the path will be added later
-A, --all              add changes from all tracked and untracked files
--refresh              don't add, only refresh the index
--ignore-errors        just skip files which cannot be added because of errors
--ignore-missing        check if - even missing - files are ignored in dry run

aculp@aculp-laptop:/sample_app$
```





# Essential Git For Developers

- **git config**
  - Easily set your information to accompany commits.
  - Generally a one time thing.

```
aculp@aculp-laptop:/sample_app$ git config --global user.name "Adam Culp"  
aculp@aculp-laptop:/sample_app$ git config --global user.email adamculp@uws.net  
aculp@aculp-laptop:/sample_app$ git config -l  
user.name=Adam Culp  
user.email=adamculp@uws.net  
core.editor=gedit  
core.autocrlf=input  
merge.tool=meld  
credential.helper=cache --timeout=3600  
core.repositoryformatversion=0  
core.filemode=true  
core.bare=false  
core.logallrefupdates=true  
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **git init**

- Instruct Git to track project by simply 'git init'.
- No more excuses! **Version all the things!**

```
aculp@aculp-laptop:/sample_app$ ls -la
total 8
drwxrwxr-x  2 aculp aculp 4096 Jun  4 14:39 .
drwxr-xr-x 26 root  root  4096 Jun  4 14:41 ..
aculp@aculp-laptop:/sample_app$ git init
Initialized empty Git repository in /sample_app/.git/
aculp@aculp-laptop:/sample_app$ ls -la
total 12
drwxrwxr-x  3 aculp aculp 4096 Jun  4 14:42 .
drwxr-xr-x 26 root  root  4096 Jun  4 14:41 ..
drwxrwxr-x  7 aculp aculp 4096 Jun  4 14:42 .git
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- **git clone {repo} {destination}**
  - Creates new repository based on another.
  - Cloned repository becomes “Origin”.
    - Internal pointer for where it came from.



# Essential Git For Developers

- **Example of 'git clone'**
  - Below we clone a repo from github.
  - We address the .git directory.
  - Specify where to put it.

```
aculp@aculp-laptop:/sample_app$ git clone https://github.com/adamculp/api-consumer.git api-consumer
Cloning into 'api-consumer'...
remote: Counting objects: 120, done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 120 (delta 47), reused 87 (delta 16)
Receiving objects: 100% (120/120), 16.27 KiB, done.
Resolving deltas: 100% (47/47), done.
aculp@aculp-laptop:/sample_app$
```



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- **git status**
  - Provides status of resources in the tracked project.

```
aculp@aculp-laptop:/sample_app$ git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- **git status**
  - Below 'git status' informs us of untracked files after created.

```
aculp@aculp-laptop:/sample_app$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       test_file_001.txt
#       test_file_002.txt
#       test_file_003.txt
nothing added to commit but untracked files present (use "git add" to track)
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **git add**
  - Stages files to be committed.

```
aculp@aculp-laptop:/sample_app$ git add test_file_001.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   test_file_001.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       test_file_002.txt
#       test_file_003.txt
aculp@aculp-laptop:/sample_app$
```





# Essential Git For Developers

- **git commit**
  - A 'git commit' includes all “staged” files.
  - Use '-m' to store a message with the commit.
    - Or git prompts user to add a message. (using default editor)

```
aculp@aculp-laptop:/sample_app$ git commit -m 'adding new file'
[master (root-commit) 64db56a] adding new file
 1 file changed, 1 insertion(+)
 create mode 100644 test_file_001.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       test_file_002.txt
#       test_file_003.txt
nothing added to commit but untracked files present (use "git add" to track)
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- **More on commits**
  - A commit should be:
    - Done OFTEN!
    - Commit messages
      - Always included
      - Short
      - Informative
    - Single commit per bug or ticket.



# Essential Git For Developers

- **git log**
  - Shows history of prior commits.
  - We've only done one, and here it is:

```
aculp@aculp-laptop:/sample_app$ git log
commit 64db56a8426155de30b31251a36d22040add431f
Author: Adam Culp <adamculp@uws.net>
Date:   Tue Jun 4 15:22:30 2013 -0400

    adding new file
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- `git show {commit hash}`
  - Hash optional, will show previous by default.
  - Shows commit + diff view of files.

```
aculp@aculp-laptop:/sample_app$ git show
commit 64db56a8426155de30b31251a36d22040add431f
Author: Adam Culp <adamculp@uws.net>
Date:   Tue Jun 4 15:22:30 2013 -0400

    adding new file

diff --git a/test_file_001.txt b/test_file_001.txt
new file mode 100644
index 0000000..484ba93
--- /dev/null
+++ b/test_file_001.txt
@@ -0,0 +1 @@
+This is a test.
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- What would a commit do?
  - We did a 'git add' for file #2, and modified file 1.

```
aculp@aculp-laptop:/sample_app$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   test_file_002.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:   test_file_001.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       test_file_003.txt
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- And now?
  - We did a 'git add' for modified file 1.

```
aculp@aculp-laptop:/sample_app$ git add test_file_001.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   test_file_001.txt
#       new file:   test_file_002.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       test_file_003.txt
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- And finally?
  - We did a 'git add' for new file 3.

```
aculp@aculp-laptop:/sample_app$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   test_file_001.txt
#       new file:   test_file_002.txt
#       new file:   test_file_003.txt
#
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **After the commit.**
  - All staged files were added.
  - A 'git status' reveals nothing new or different.

```
aculp@aculp-laptop:/sample_app$ git commit -m 'now all files are added'
[master cbc97ca] now all files are added
 3 files changed, 4 insertions(+)
 create mode 100644 test_file_002.txt
 create mode 100644 test_file_003.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch master
nothing to commit (working directory clean)
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- **Commits do not carry a version #**
  - Git doesn't use numbers like 1, 2, 3...
  - Instead uses hashes like 6e7e6999c879f460b5e1d7e29ffe9907062ec20a

```
aculp@aculp-laptop:/sample_app$ git log
commit cca6d96987bb3f573d947d5478af80d257219a0a
Author: Adam Culp <adamculp@uws.net>
Date:   Sun Jun 9 15:17:17 2013 -0400

    now all files are added

commit 9caa01bd6807a9d72051180e05e2e45f3cecf226
Author: Adam Culp <adamculp@uws.net>
Date:   Sun Jun 9 15:16:29 2013 -0400

    altered file #1 for commit

commit 3455b09c0f46bf9eadba529cff5980f7b19cdb36
Author: Adam Culp <adamculp@uws.net>
Date:   Sun Jun 9 15:15:10 2013 -0400

    added file #2

commit 6e7e6999c879f460b5e1d7e29ffe9907062ec20a
Author: Adam Culp <adamculp@uws.net>
Date:   Sun Jun 9 15:12:18 2013 -0400

    adding new file
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- Working in 'master' is bad.
  - Should not be working in the 'master' branch.
  - 'master' should be pristine version.
    - Most bug free.
    - Tested
    - Same as “Production”

```
aculp@aculp-laptop:/sample_app$ git commit -m 'now all files are added'
[master cbc97ca] now all files are added
 3 files changed, 4 insertions(+)
 create mode 100644 test_file_002.txt
 create mode 100644 test_file_003.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch master
nothing to commit (working directory clean)
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **git branch**
  - Shows a list of existing branches.
  - The \* indicates active branch.

```
aculp@aculp-laptop:/sample_app$ git branch
* master
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- `git branch {name} {branch}`
- Or `git checkout -b {name} {branch}`
  - Creates new branch.
  - Checkout -b checks out after creation.
  - Below we create a 'development' branch.
  - New branch has same state as active/specified branch.

```
aculp@aculp-laptop:/sample_app$ git branch development
aculp@aculp-laptop:/sample_app$ git branch
  development
* master
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **git checkout {name}**
  - Include “-b” flag to create new branch.
  - Switches to a specified branch.
  - Branches carry own state.
  - In file browser file contents different.

```
aculp@aculp-laptop:/sample_app$ git checkout development
Switched to branch 'development'
aculp@aculp-laptop:/sample_app$ git branch
* development
  master
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- What if?
  - A file has been edited, but not committed.
  - We are in 'development' branch.
  - What if we 'git checkout master'?

```
aculp@aculp-laptop:/sample_app$ vi test_file_001.txt
aculp@aculp-laptop:/sample_app$ git status
# On branch development
# 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:   test_file_001.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
aculp@aculp-laptop:/sample_app$
```





# Essential Git For Developers

- Change branch with uncommitted files
  - Merges uncommitted content on checkout.
    - Whether 'staged' or not.
  - Does NOT merge over newly created files. (changes only)
  - Conflicts get exciting. (Not covered in this talk.)

```
aculp@aculp-laptop:/sample_app$ git checkout master
M   test_file_001.txt
Switched to branch 'master'
aculp@aculp-laptop:/sample_app$ git status
# On branch 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:   test_file_001.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- **File not actually changed**

- On 'git checkout development' and commit:
  - File in development carries edit committed.
  - File in master is reset, even though merged previously.

master

```
This is a test.  
Add a new line.  
Insert another line to commit.  
~  
~
```

development

```
This is a test.  
Add a new line.  
Insert another line to commit.  
edit from development branch  
~  
~
```

# Essential Git For Developers

- But if commit done first
  - Commit only done on active branch.
  - Master branch is unchanged. ('git log' shown below)
  - Master files do not contain merged changes.

master

```
aculp@aculp-laptop:/sample_app$ git log
commit cbc97ca75318a9091b954fd474eb3739a7760c23
Author: Adam Culp <adamculp@uws.net>
Date: Tue Jun 4 16:26:46 2013 -0400

    now all files are added

commit 64db56a8426155de30b31251a36d22040add431f
Author: Adam Culp <adamculp@uws.net>
Date: Tue Jun 4 15:22:30 2013 -0400

    adding new file
aculp@aculp-laptop:/sample_app$
```

development

```
aculp@aculp-laptop:/sample_app$ git log
commit 1b51ea07c3a009d9cc4e8d325ff82f5cd405e382
Author: Adam Culp <adamculp@uws.net>
Date: Tue Jun 4 17:20:36 2013 -0400

    altered file in development

commit cbc97ca75318a9091b954fd474eb3739a7760c23
Author: Adam Culp <adamculp@uws.net>
Date: Tue Jun 4 16:26:46 2013 -0400

    now all files are added

commit 64db56a8426155de30b31251a36d22040add431f
Author: Adam Culp <adamculp@uws.net>
Date: Tue Jun 4 15:22:30 2013 -0400

    adding new file
aculp@aculp-laptop:/sample_app$
```

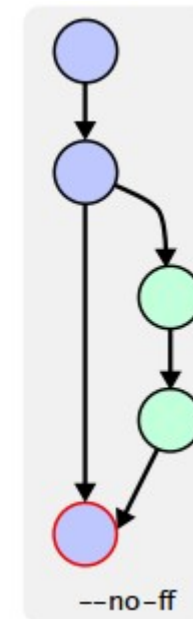
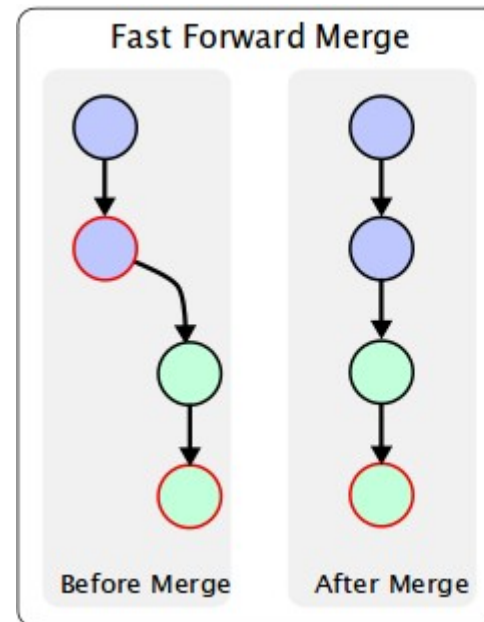
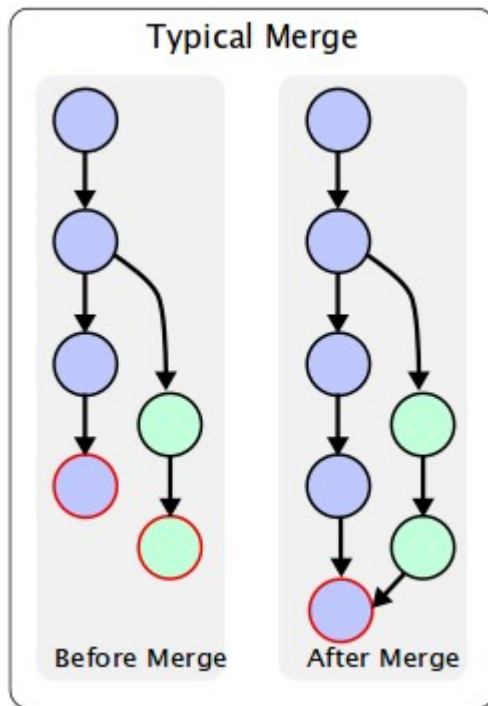
# Essential Git For Developers

- **git merge {branch}**
  - Git merges specified branch into active branch.
  - We merge change from development to master.
    - 'git checkout master'
    - 'git merge development'

```
aculp@aculp-laptop:/sample_app$ git merge development
Updating cca6d96..be0b6ae
Fast-forward
 test_file_001.txt |    2 ++
 1 file changed, 2 insertions(+)
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

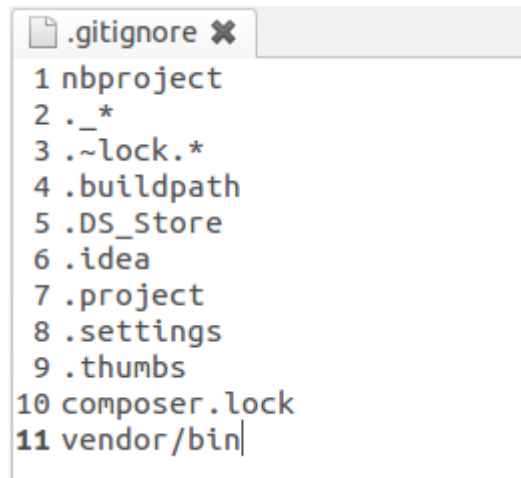
- What are “fast forward” commits?
  - Merges individual commits into flow as if a checkout never occurred.



# Essential Git For Developers

- **Ignoring**

- We can exclude:
  - Files
  - Folders
  - Config files with passwords ! ! !
- Simply add excluded content to the file '.gitignore'.

A screenshot of a code editor showing a file named '.gitignore'. The file contains a list of patterns to be ignored, numbered 1 through 11. The patterns are: 1 nbproject, 2 .\_\* , 3 ~/.lock.\* , 4 .buildpath, 5 .DS\_Store, 6 .idea, 7 .project, 8 .settings, 9 .thumbs, 10 composer.lock, and 11 vendor/bin. The cursor is at the end of the last line.

```
.gitignore ✖
1 nbproject
2 ._*
3 ~/.lock.*
4 .buildpath
5 .DS_Store
6 .idea
7 .project
8 .settings
9 .thumbs
10 composer.lock
11 vendor/bin|
```



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- Typical branches for teams

- Conventions:

- Testing, Staging and Master branches *off limits* but public.
    - Development public, public to all.
    - {user}-development branches local and private.

```
aculp@aculp-laptop:/sample_app$ git branch
* aculp-development
  development
  master
  staging
aculp@aculp-laptop:/sample_app$
```



# Essential Git For Developers

- Typical rules for branch usage
  - No code leaves {user}-development unless **finished** and **stable**.
    - Developers merge to development branch...period!
  - Do NOT merge conflicts into any public branch.

# RULES



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- **Commit procedure (origin pull/merge/push)**
  - Before merging changes to public development:
    - 'git checkout development'
    - 'git pull origin development'
      - Should be no conflicts.
    - 'git checkout {user}-development'
    - 'git merge development'
      - Fix conflicts
    - 'git checkout development'
    - 'git merge {user}-development'
    - 'git push origin development'



# Essential Git For Developers

- **Public and Private branches**

- Typically {user}-development branches remain private.
  - The team is not aware of commits done there.
  - Frequent commits encouraged.
- Development, Staging, and Master are public and entire team can view state/commits.
  - All developers can merge to development.
  - Only authorized people can merge to staging or master.

```
aculp@aculp-laptop:/sample_app$ git branch
* aculp-development
  development
  master
  staging
aculp@aculp-laptop:/sample_app$
```



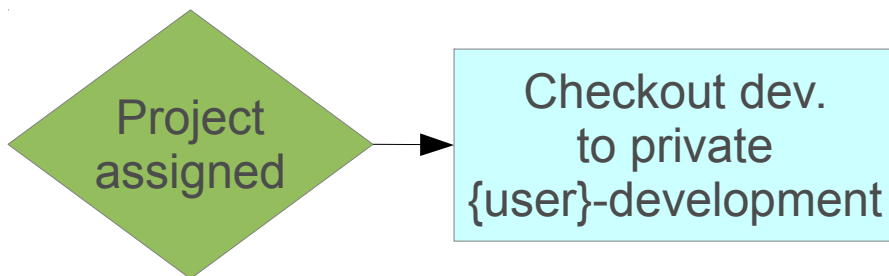
# Essential Git For Developers

- **Team Developer workflow**
  - Git is ideal for team development



# Essential Git For Developers

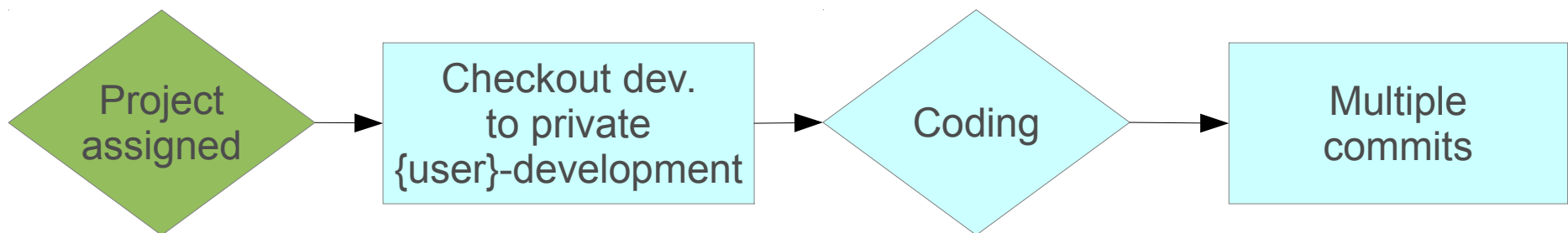
- **Team Developer workflow (private)**
  - Project/ticket assigned, create branch
    - 'git checkout development'
    - 'git branch {user}-development' or 'git checkout -b {user}-development'
  - Start coding.
  - Commit often.



# Essential Git For Developers

- **Team Developer workflow (private)**

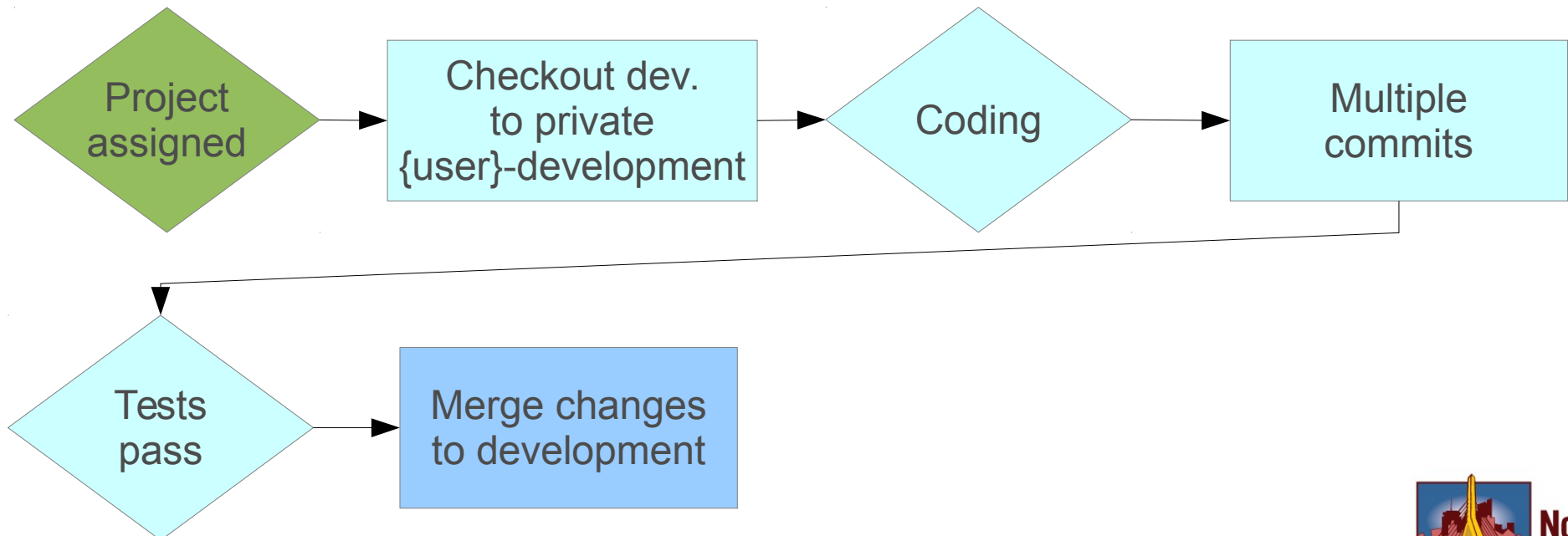
- Regularly commit code.
  - 'git add {filename}' X each file
  - 'git commit -m {commit message}'
- Regularly pull from origin.
  - 'git checkout development' **followed by** 'git pull origin development'
  - 'git checkout {user}-development' **followed by** 'git merge development'



# Essential Git For Developers

- **Team Developer workflow (private)**

- Development completed, ready for QA testing.
  - 'git checkout development'
  - 'git pull origin development' should be no conflicts.
  - 'git merge {user}-development' should be no conflicts.

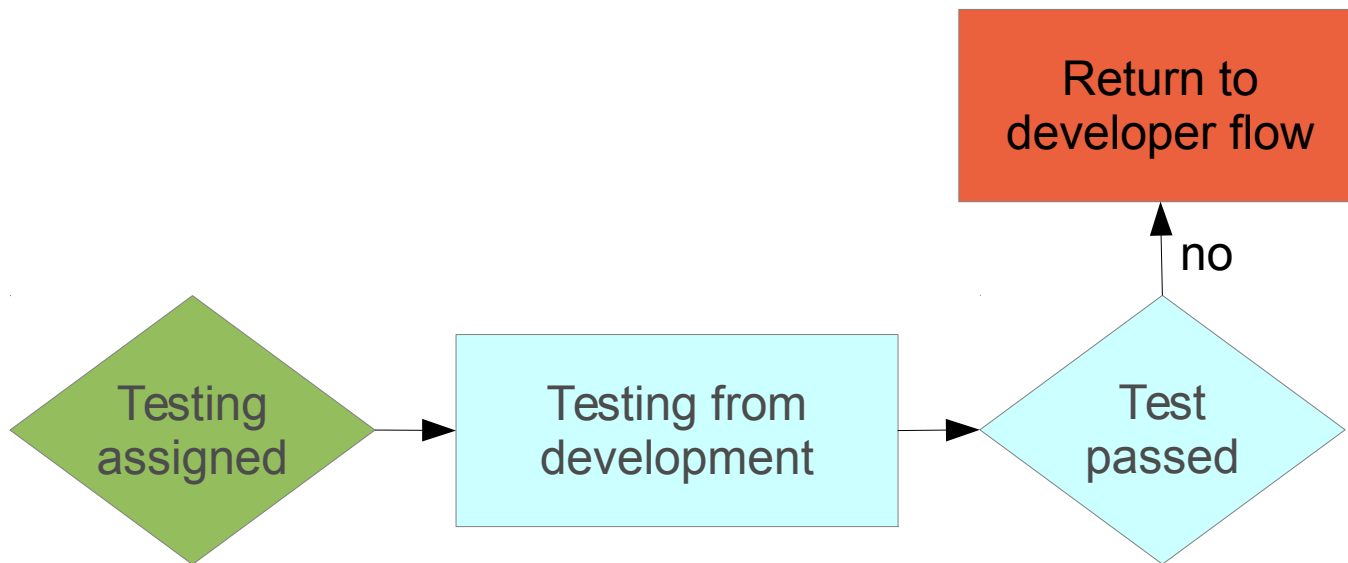




# Essential Git For Developers

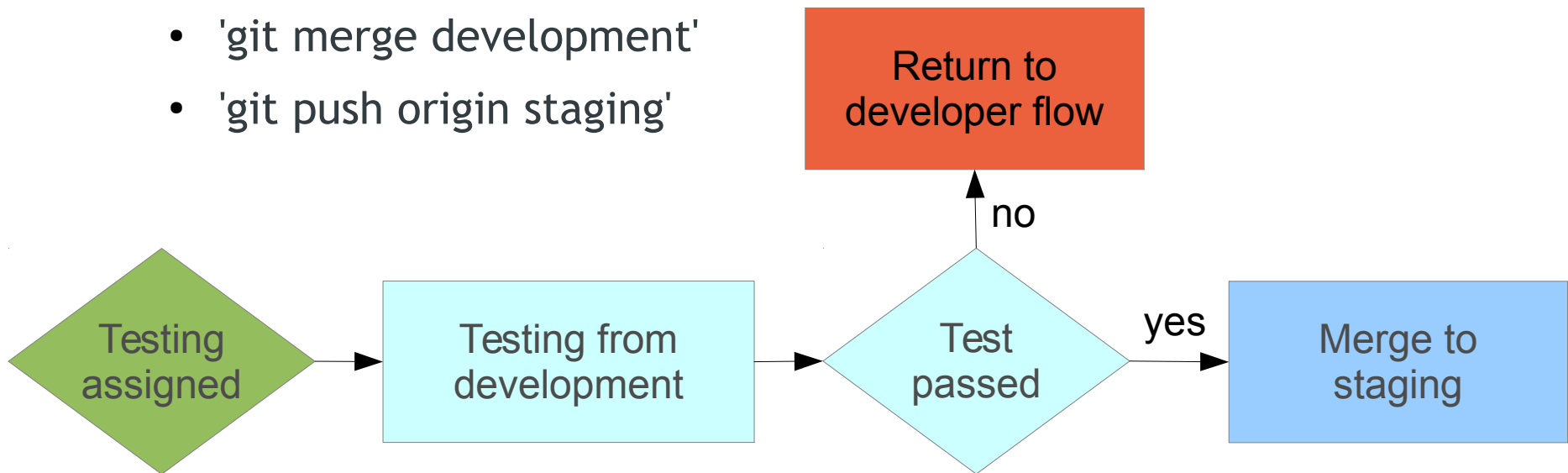
- **Team QA workflow (public)**

- Testing done in development branch.
- Failed → developer picks up on {user}-development.
- Bug fixed → re-push to development.



# Essential Git For Developers

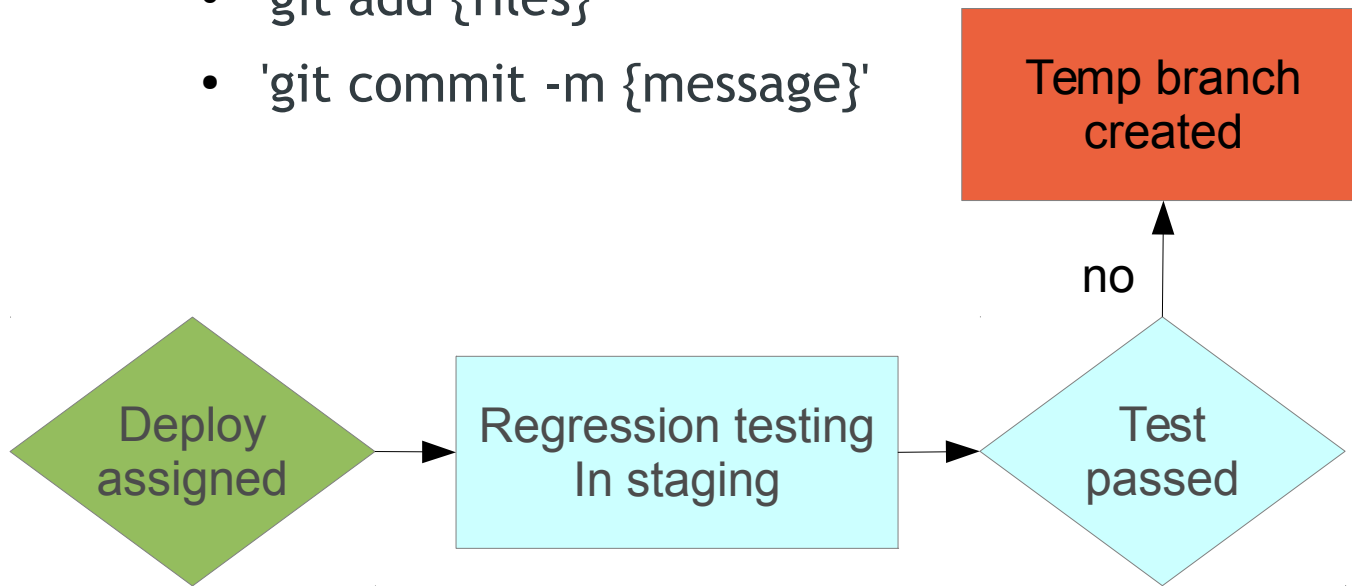
- **Team QA workflow (public)**
  - Testing done in development branch.
  - Success → merge to staging
    - 'git checkout staging'
    - 'git pull origin staging'
    - 'git merge development'
    - 'git push origin staging'



# Essential Git For Developers

- **Team Deployment Mgr. workflow (public)**

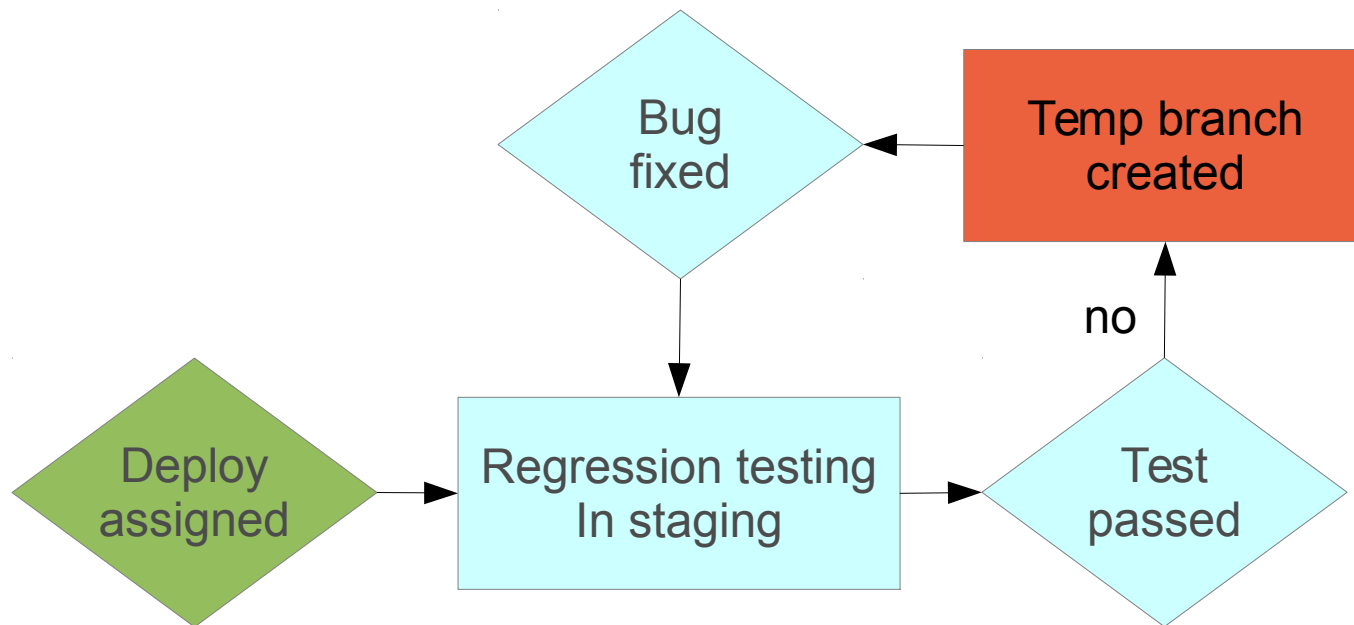
- Regression testing done in staging branch.
- Testing failed:
  - 'git branch -b {tempname}-staging'
- Code to fix bug
  - 'git add {files}'
  - 'git commit -m {message}'



# Essential Git For Developers

- **Team Deployment Mgr. workflow (public)**

- Send fix back for regression testing done in staging branch.
  - 'git merge staging' just to check for conflicts.
  - 'git checkout staging'
  - 'git merge {tempname}-staging'

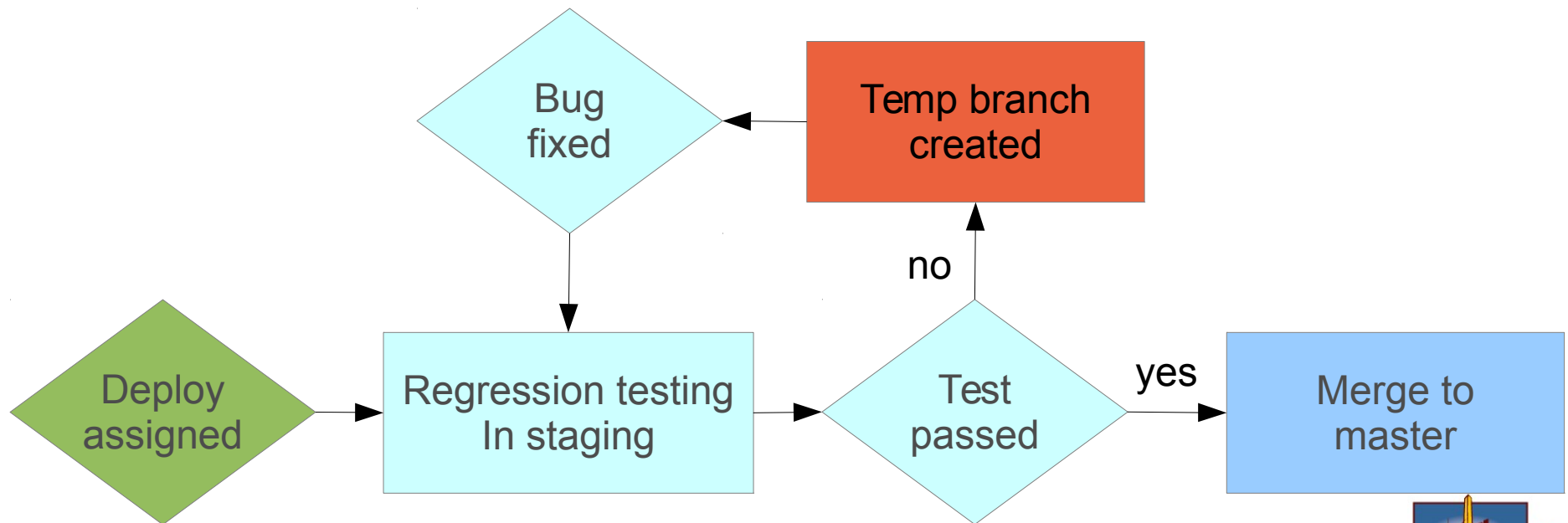


# Essential Git For Developers

- **Team Deployment Mgr. workflow (public)**

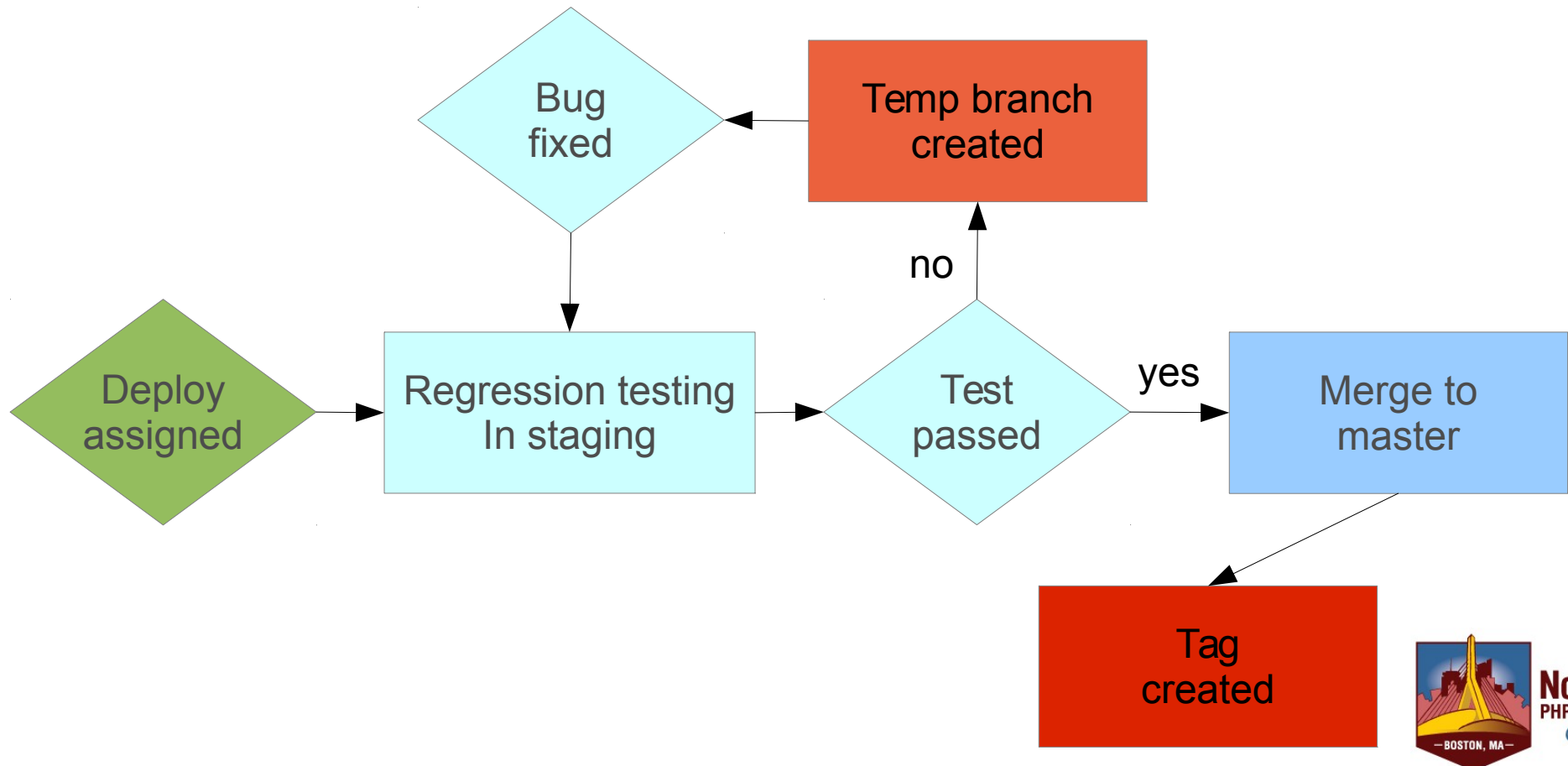
- If regression tests pass:

- 'git merge master' in case of conflicts
    - 'git checkout master' then 'git pull origin master'
    - 'git merge staging'



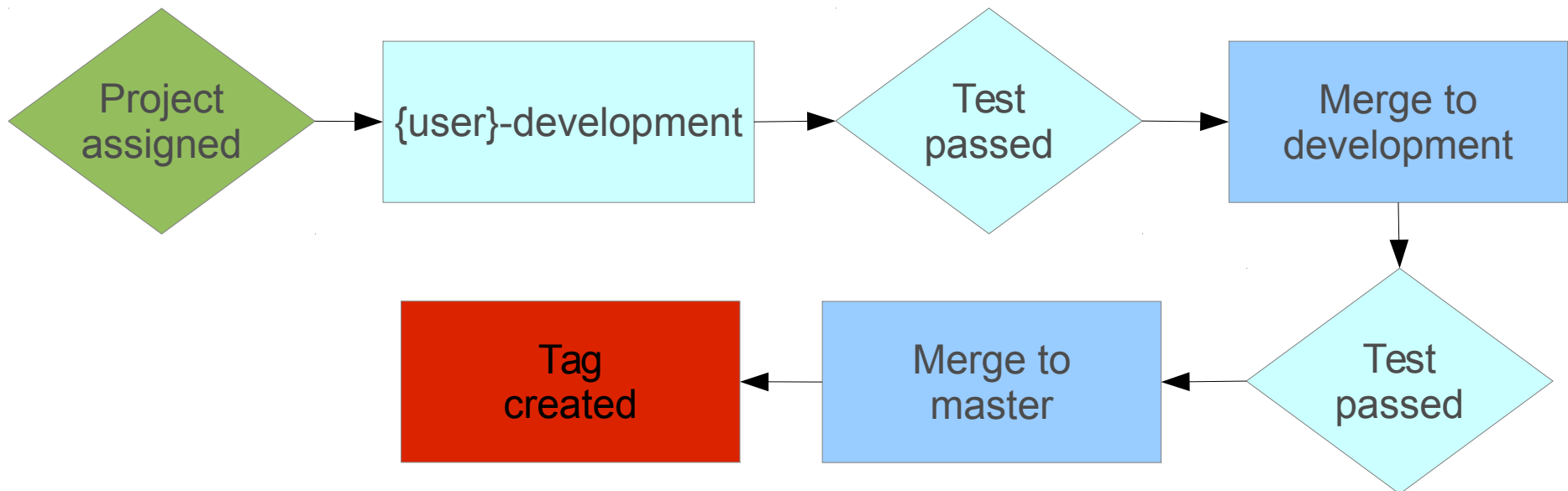
# Essential Git For Developers

- **Team Deployment Mgr. workflow (public)**
  - All is good, create annotated tag.
    - `'git tag -a v1.0 -m '{message}'` (Note: 'git tag' lists all tags.)



# Essential Git For Developers

- **Single Developer workflow (small project)**
  - Pretty similar, but no staging.
  - Note: Still use {user}-development per task/project/ticket.

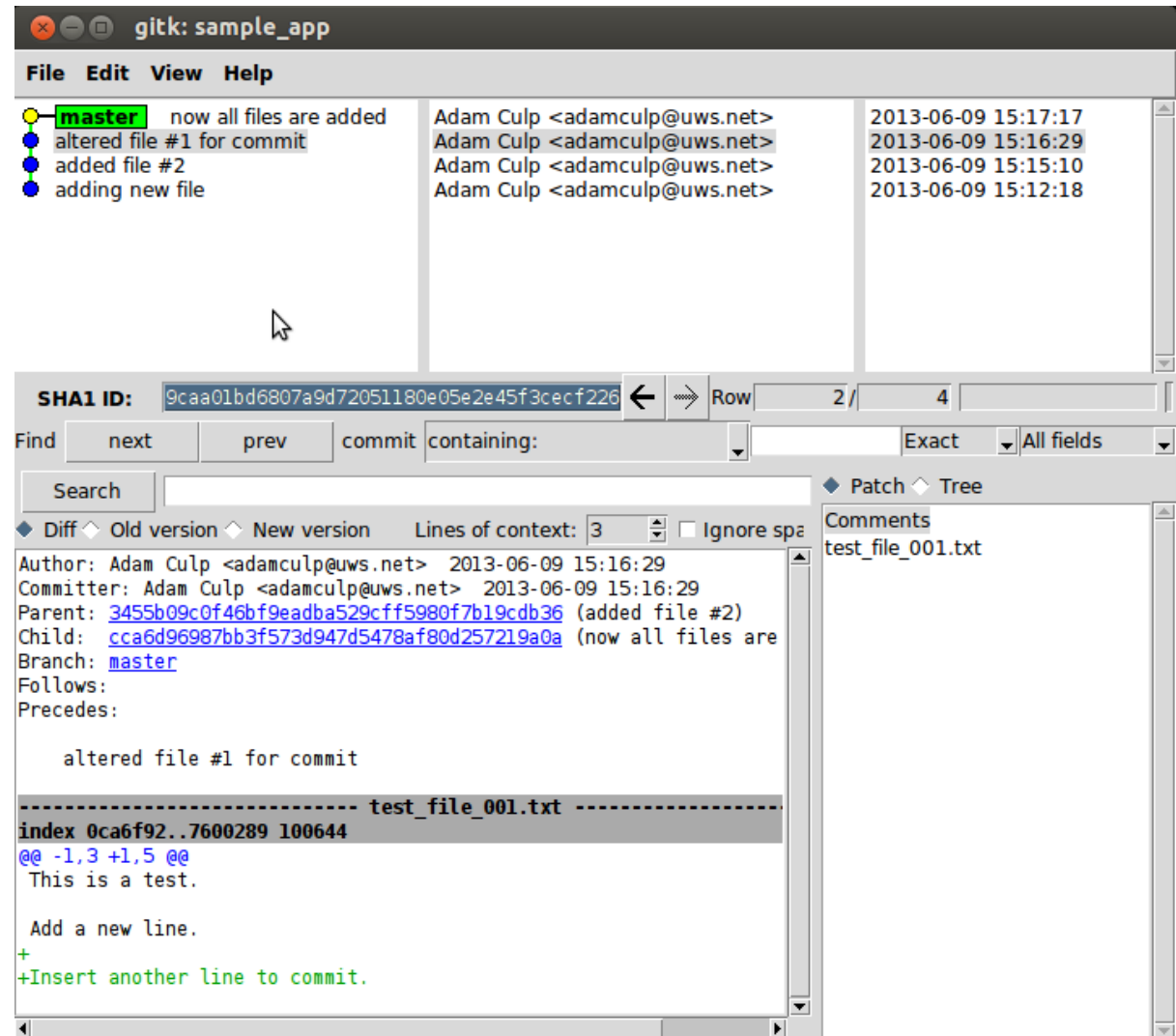




# Essential Git For Developers

- Tools

- gitk
- gitx
- git-cola
- SmartGit
- GitEye
- TortoiseGit
- IDE
  - Eclipse
    - Zend Studio
    - Aptana
  - PHPStorm
  - etc.



# Essential Git For Developers

- **github.com**
  - Great place to share code.
  - Promotes collaboration
  - API enhances connectivity and use
  - Awesome plugins
  - Easy to use



# Essential Git For Developers

- **github - how to clone locally**
  - Standard 'git clone' command
  - Now have a clone local to work on.
  - Follow workflow as shown earlier.

```
aculp@aculp-laptop:/sample_app$ git clone https://github.com/adamculp/api-consumer.git api-consumer
Cloning into 'api-consumer'...
remote: Counting objects: 120, done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 120 (delta 47), reused 87 (delta 16)
Receiving objects: 100% (120/120), 16.27 KiB, done.
Resolving deltas: 100% (47/47), done.
aculp@aculp-laptop:/sample_app$
```

# Essential Git For Developers

- **Conclusion**

- Always use source control!!!
- Git is an easy solution, just 'git init'.
- Plan a workflow and stick with it...ALWAYS!
- 3rd party repositories = backed up
- Git easy to connect to from anywhere.
- Love iteration!



**Northeast**  
PHP CONFERENCE  
@NEPHP

# Essential Git For Developers

- **Resources**

- <http://nvie.com/posts/a-successful-git-branching-model/>
- <http://github.com>
- <http://training.github.com/>
- <https://bitbucket.org/>
- <http://git-scm.com>



**Northeast**  
PHP CONFERENCE  
@NEPHP

- Thank you!

Essential Git For Developers  
Adam Culp

<http://www.geekyboy.com>

<http://RunGeekRadio.com>

Twitter @adamculp

<https://joind.in/14744>

Questions?



**Northeast**  
PHP CONFERENCE  
@NEPHP