Remote Branches - - - - |

Remote references (जस्तै शाखाहरू र ट्यागहरू) तपाईंका remote repositories मा रहेका pointers हुन्।  
तपाईं explicitly **git ls-remote <remote>** वा **git** **remote show <remote>** कमाण्डको प्रयोग गरेर remote references को सूची देख्न सक्नुहुन्छ।

**Remote-Tracking Branches:**

**Remote-tracking branches** भनेको **remote** शाखाहरूको स्थिति प्रतिनिधित्व गर्ने references हुन्।  
यी स्थानीय references हुन्, जसलाई तपाईं आफैँ सार्न सक्नुहुन्न। Git ले नै यी references update गर्छ जब तपाईं network communication गर्नुहुन्छ।  
यी शाखाहरूले **remote repositories** को अन्तिम स्थितिलाई **"bookmark"** जस्तै सम्झाउने काम गर्छन्।

**Naming Convention:**

**Remote-tracking** शाखाहरूको नाम **<remote>/<branch>** को रूप हुन्छ।  
उदाहरणका लागि:

* **origin/master:** **Remote server** मा रहेको **master** शाखाको स्थिति।
* **origin/iss53: Remote server** मा रहेको **iss53** शाखाको स्थिति।

**Example:**

मानौं तपाईंको Git server **git.ourcompany.com** मा छ।  
जब तपाईं यस server बाट **clone** गर्नुहुन्छ:

1. Git ले यसलाई **origin** नाम दिन्छ।
2. सबै डाटा pull गरेर **remote** को master शाखाको **pointer (origin/master)** बनाउँछ।
3. तपाईंको आफ्नै master शाखा पनि बनाउँछ, जुन origin को **master** शाखाबाट सुरु हुन्छ।

यसरी तपाईंले **local branches** र **remote-tracking** **branches** बीच काम गर्न सुरु गर्न सक्नुहुन्छ।

**“Origin” is not Special**

Git मा “origin” को कुनै विशेष अर्थ छैन।  
जस्तै, शाखाको नाम “master” को पनि कुनै विशेष महत्व छैन।

**Master Branch:**

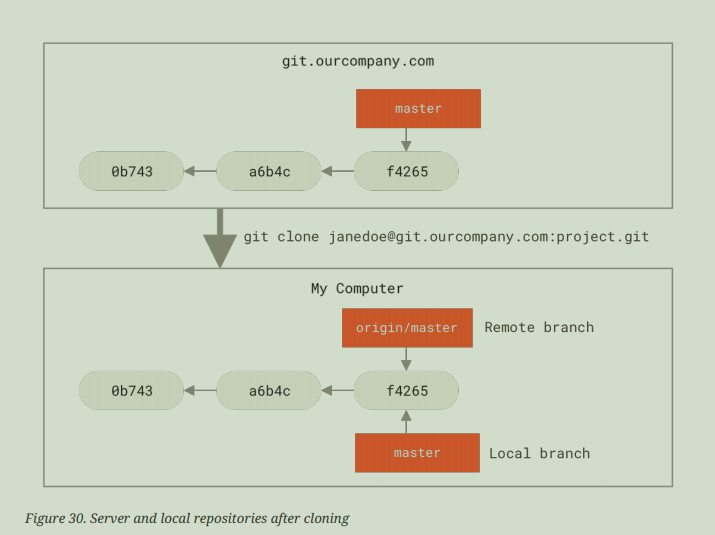
* git init चलाउँदा Git ले सुरुको शाखाको नाम master राख्छ।
* यस कारणले नै यो नाम सामान्य रूपमा प्रयोगमा आएको हो।

**Origin Remote:**

* जब तपाईं git clone चलाउनुहुन्छ, Git ले default remote को नाम origin राख्छ।
* तर, तपाईं यसलाई परिवर्तन गर्न सक्नुहुन्छ।
  + उदाहरणका लागि: git clone -o booyah चलाउँदा default remote को नाम booyah हुनेछ।
  + यसले booyah/master नामको default remote branch बनाउँछ।

**निष्कर्ष:**

Git मा master र origin केवल default नामहरू हुन्, तर तपाईं आफ्नो आवश्यकता अनुसार यी नामहरू परिवर्तन गर्न स्वतन्त्र हुनुहुन्छ।



यदि तपाईं आफ्नो स्थानीय master शाखामा केही काम गर्नुहुन्छ र यसैबीच कसैलाई अर्को व्यक्तिले git.ourcompany.com मा master शाखा अपडेट गर्छ भने, तपाईंको र तिनीहरूको इतिहास फरक दिशामा अगाडि बढ्छ।

**स्थिति:**

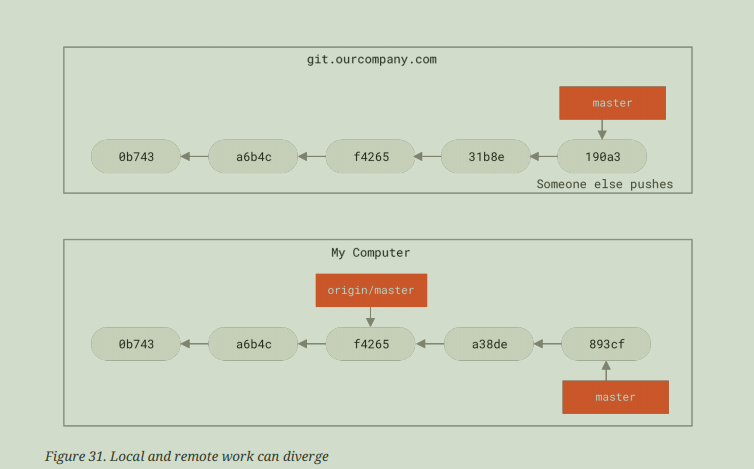
1. **तपाईंको स्थानीय परिवर्तनहरू:**
   * तपाईं आफ्नो master शाखामा काम गर्दै हुनुहुन्छ, नयाँ commits थप्दै।
2. **कोहीले origin मा परिवर्तनहरू पुश गर्छ:**
   * अर्को व्यक्तिले origin/master शाखा (जसको तपाईंको server मा master छ) मा commits थप्दछन्।
3. **Timestamps (समय) फरक:**
   * तपाईंको master शाखा र origin/master शाखाको commits अब फरक हुन्छन्।

**Remote Pointer:**

* जबसम्म तपाईं आफ्नो origin server सँग पुन: सम्पर्क गर्नुहुन्न (git fetch वा git pull को माध्यमबाट),
  + तपाईंको origin/master pointer (जो तपाईंको स्थानीय repository मा छ) **move हुँदैन**।
  + यो पुरानो अवस्थामै रहन्छ, जो अन्तिम पटक तपाईंले server सँग सम्पर्क गर्दा थियो।

**समाधान:**

* आफ्नो master र origin/master लाई sync गर्न, git pull वा git fetch चलाउनुहोस्।
  + **git pull:** commits ल्याउँछ र merge गर्छ।
  + **git fetch:** commits मात्र ल्याउँछ, merge गर्दैन।



तपाईंको कामलाई कुनै निश्चित **remote** सँग समकालिक (**synchronize**) गर्न, तपाईं **git fetch <remote>** कमाण्ड चलाउनुहुन्छ। उदाहरणको लागि, यहाँ git fetch origin प्रयोग गरिन्छ।

**कसरी काम गर्छ:**

1. **Remote Server Lookup:**
   * यो कमाण्डले origin remote server (हाम्रो केसमा git.ourcompany.com) को विवरण खोज्छ।
2. **Data Fetch:**
   * Server मा भएका नयाँ data (जसको तपाईंको स्थानीय repository मा अभाव छ) ल्याउँछ।
3. **Pointer Update:**
   * **origin/master pointer** लाई server मा भएको नयाँ र अधिक अद्यावधिक (updated) स्थिति अनुसार सार्दछ।

**ध्यान दिनु पर्ने कुरा:**  
git fetch केवल data ल्याउँछ र local शाखामा परिवर्तन गर्दैन। तपाईंले fetched data लाई merge गर्न चाहनुभयो भने, त्यसका लागि git merge अथवा git pull को आवश्यकता पर्छ।

A screenshot of a computer

Description automatically generated

धेरै **remote servers** भएको अवस्था देखाउनको लागि, र ती remote projects को शाखाहरू कस्तो देखिन्छ भन्ने बुझ्न, हामी एउटा उदाहरण लिन्छौं।

**परिदृश्य:**

तपाईंसँग अर्को **internal Git server** छ, जुन केवल एक **sprint टीम** ले विकासको लागि प्रयोग गर्छ। यो server को ठेगाना git.team1.ourcompany.com हो।

**यसलाई आफ्नो हालको project मा कसरी थप्ने:**

तपाईंले git remote add कमाण्ड प्रयोग गरेर यो नयाँ remote reference थप्न सक्नुहुन्छ।

**स्टेपहरू:**

1. **Remote Server थप्ने:**

git remote add teamone git.team1.ourcompany.com

यहाँ:

* + teamone: यो **shortname** हो, जुन git.team1.ourcompany.com को लागि प्रयोग हुनेछ।

1. **Shortname प्रयोगको लाभ:** अब, teamone नाम प्रयोग गरेर तपाईं सो server सँग काम गर्न सक्नुहुन्छ, जसले लामो URL हर समय टाइप गर्नुपर्ने झन्झटबाट बचाउँछ।
2. **Remote Branch देख्ने:** teamone remote मा भएका branches हेर्नको लागि:

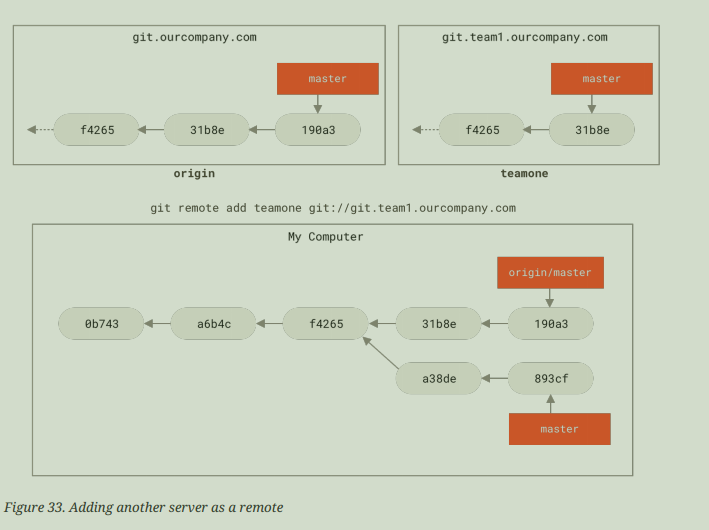
git fetch teamone

git branch -r

यसले ती शाखाहरूको सूची देखाउँछ, जस्तै teamone/master, teamone/feature-branch, आदि।

**निस्कर्ष:**

यो विधिले तपाईंलाई विभिन्न remote servers लाई सजिलै व्यवस्थित गर्न र ती projects बीच प्रभावकारी रूपमा समन्वय गर्न सहयोग पुर्‍याउँछ।



अब, तपाईंले git fetch teamone कमाण्ड चलाउन सक्नुहुन्छ ताकि **teamone** remote server मा भएका र तपाईंको local repository मा नभएका सबै डेटा ल्याउन सकियोस्।

**के हुन्छ git fetch teamone चलाउँदा?**

1. **डाटा जाँच:**  
   Git ले **teamone** server मा भएका commits र branches जाँच गर्छ।
   * यदि **teamone** server मा तपाईंको local repository मा नभएको नयाँ डेटा छ भने, त्यो डेटा fetch हुन्छ।
   * यदि त्यहाँ कुनै नयाँ डेटा छैन भने, कुनै डेटा fetch हुँदैन।
2. **Remote-Tracking Branch सेटअप:**  
   Git एउटा **remote-tracking branch** बनाउँछ, जसलाई teamone/master भनिन्छ।
   * यसले **teamone** server को master branch को हालको स्थिति (commit) मा pointer सेट गर्छ।
   * यद्यपि, तपाईंको local repository मा teamone/master को डेटा मात्रै आउँछ, तपाईंको local master branch मा स्वतः कुनै परिवर्तन हुँदैन।

**कमाण्ड:**

git fetch teamone

**परिणाम:**

* यदि **teamone** server को master branch मा नयाँ commits छैनन् भने:

From git.team1.ourcompany.com

\* [new branch] master -> teamone/master

यसले केवल **teamone/master** को स्थिति अपडेट गर्छ, तर कुनै नयाँ डेटा fetch हुँदैन।

* यदि नयाँ commits छन् भने:

From git.team1.ourcompany.com

\* [new branch] master -> teamone/master

\* [new branch] feature-x -> teamone/feature-x

यसले **teamone** server का सबै branches को स्थितिलाई तपाईंको local repository मा अपडेट गर्छ।

**निस्कर्ष:**

git fetch teamone ले **teamone** server बाट डेटा ल्याउन र **remote-tracking branches** को स्थितिलाई अपडेट गर्न प्रयोग गरिन्छ। यसले तपाईंलाई अन्य टीमले गरेको कामको बारेमा जानकारी राख्न मद्दत गर्छ, बिना तपाईंको local branches परिवर्तन गरेको।

A screenshot of a computer

Description automatically generated

Pushing (Translation) - - - -

जब तपाईं आफ्नो branch अरूसँग बाँड्न चाहनुहुन्छ, तपाईंले त्यसलाई कुनै remote server मा push गर्नुपर्छ जसमा तपाईंलाई write access छ।  
तपाईंका local branches स्वतः remote server सँग synchronize हुँदैनन्। तपाईंले चाहिने branches मात्र explicitly push गर्नुपर्छ।  
यसरी, तपाईं निजी branches (private branches) लाई आफ्नै कामका लागि राख्न सक्नुहुन्छ र केवल ती branches push गर्न सक्नुहुन्छ जसमा तपाईं सहकार्य गर्न चाहनुहुन्छ।

**उदाहरण:**

यदि तपाईंको branch को नाम serverfix छ र तपाईं यसमा अरूसँग काम गर्न चाहनुहुन्छ भने, यसलाई push गर्न निम्न कमाण्ड प्रयोग गर्न सकिन्छ:

git push <remote> <branch>

उदाहरण:

git push origin serverfix

**परिणाम:**

Counting objects: 24, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (15/15), done.

Writing objects: 100% (24/24), 1.91 KiB | 0 bytes/s, done.

Total 24 (delta 2), reused 0 (delta 0)

To https://github.com/schacon/simplegit

\* [new branch] serverfix -> serverfix

यो कमाण्डले:

* serverfix branch लाई origin remote मा push गर्छ।
* अब, यो branch सबैका लागि उपलब्ध हुन्छ जसले यस remote repository लाई access गर्छ।

यो एउटा छोटो तरीका हो। Git स्वचालित रूपमा serverfix branch को नामलाई refs/heads/serverfix:refs/heads/serverfix मा विस्तार गर्छ, जसको अर्थ हुन्छ, “मेरो serverfix local branch लाई लिनुहोस् र remote को serverfix branch लाई अपडेट गर्नुहोस्।” हामी refs/heads/ को बारेमा Git Internals मा विस्तारमा जानेछौं, तर सामान्यत: तपाईं यसलाई छुट्याउन सक्नुहुन्छ। तपाईं यसलाई git push origin serverfix:serverfix कमाण्डद्वारा पनि गर्न सक्नुहुन्छ, जसको मतलब "मेरो serverfix लाई लिनुहोस् र यसलाई remote को serverfix बनाउनुहोस्।"

यो तरीका तपाईंलाई स्थानीय branch लाई remote branch मा फरक नामले push गर्न अनुमति दिन्छ। यदि तपाईंलाई remote मा serverfix नामको branch चाहिँदैन भने, तपाईंले यो कमाण्ड चलाउन सक्नुहुन्छ:

git push origin serverfix:awesomebranch

यसले तपाईंको स्थानीय serverfix branch लाई remote परियोजनामा awesomebranch branch मा push गर्नेछ।

***पानीको पासवर्ड नहचाउनको लागि***

यदि तपाईं HTTPS URL प्रयोग गर्दै हुनुहुन्छ र Git सर्भरले तपाईंको प्रयोगकर्ता नाम र पासवर्ड प्रमाणीकरणको लागि सोध्नेछ। डिफल्टमा, यो तपाईंलाई टर्मिनलमा सोध्नेछ ताकि सर्भर थाहा पाउनेछ कि तपाईंलाई push गर्न अनुमति छ कि छैन।

यदि तपाईं हरेक पटक पासवर्ड टाइप नगर्न चाहनुहुन्छ भने, तपाईं एक "credential cache" सेटअप गर्न सक्नुहुन्छ। यसलाई सामान्यत: केही समयको लागि मेमोरीमा राख्नको लागि सजिलै सेटअप गर्न सकिन्छ। यसका लागि, git config --global credential.helper cache कमाण्ड चलाउनुहोस्।

अर्को पटक जब तपाईंको सहकर्मी सर्भरबाट fetch गर्छन्, तिनीहरूले सर्भरको serverfix संस्करणको रेफरेन्स प्राप्त गर्नेछन् जुन रिमोट ब्रान्च origin/serverfix अन्तर्गत हुनेछ:

$ git fetch origin

remote: Counting objects: 7, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 3 (delta 0)

Unpacking objects: 100% (3/3), done.

From https://github.com/schacon/simplegit

 \* [new branch] serverfix -> origin/serverfix

यो महत्त्वपूर्ण कुरा हो कि जब तपाईं fetch गर्नुहुन्छ र नयाँ रिमोट-ट्र्याकिङ ब्रान्चहरू ल्याउनुहुन्छ, तपाईंले स्वचालित रूपमा तिनीहरूको स्थानीय, सम्पादन योग्य प्रति प्राप्त गर्न सक्नुहुन्न। यसको मतलब, यस अवस्थामा, तपाईंको पास नयाँ serverfix ब्रान्च छैन - तपाईंको पास केवल origin/serverfix प्वाइन्टर मात्र छ जसलाई तपाईंले परिवर्तन गर्न सक्नुहुन्न।

यस कामलाई तपाईंको वर्तमान कार्यशील ब्रान्चमा मर्ज गर्नको लागि, तपाईं git merge origin/serverfix चलाउन सक्नुहुन्छ। यदि तपाईं आफ्नै serverfix ब्रान्च चाहनुहुन्छ जसमा तपाईं काम गर्न सक्नुहुन्छ, तपाईं यसलाई रिमोट-ट्र्याकिङ ब्रान्चबाट आधारित बनाउन सक्नुहुन्छ:

$ git checkout -b serverfix origin/serverfix

Branch serverfix set up to track remote branch serverfix from origin.

Switched to a new branch 'serverfix'

यसले तपाईंलाई एक स्थानीय ब्रान्च दिन्छ जसमा तपाईं काम गर्न सक्नुहुन्छ जुन origin/serverfix बाट शुरू हुन्छ।

Tracking Branches - ट्र्याकिङ ब्रान्चेस

रिमोट-ट्र्याकिङ ब्रान्चबाट स्थानीय ब्रान्च चेकआउट गर्दा स्वचालित रूपमा के भनिन्छ "ट्र्याकिङ ब्रान्च" सिर्जना हुन्छ (र जुन ब्रान्चलाई यो ट्र्याक गर्छ, त्यसलाई "अपस्ट्रीम ब्रान्च" भनिन्छ)। ट्र्याकिङ ब्रान्चहरू स्थानीय ब्रान्चहरू हुन् जसको रिमोट ब्रान्चसँग सिधा सम्बन्ध छ। यदि तपाईं ट्र्याकिङ ब्रान्चमा हुनुहुन्छ र git pull टाइप गर्नुहुन्छ भने, Git स्वचालित रूपमा जान्छ कुन सर्भरबाट fetch गर्नुपर्छ र कुन ब्रान्चसँग मर्ज गर्नुपर्छ।

जब तपाईं एक रिपोजिटोरी क्लोन गर्नुहुन्छ, यसले सामान्यतया स्वचालित रूपमा master ब्रान्च सिर्जना गर्छ जुन origin/master लाई ट्र्याक गर्छ। यद्यपि, यदि तपाईं चाहनुहुन्छ भने, तपाईं अन्य ट्र्याकिङ ब्रान्चहरू पनि सेट अप गर्न सक्नुहुन्छ — जो अन्य रिमोटहरूमा रहेका ब्रान्चहरूलाई ट्र्याक गर्छन्, वा master ब्रान्चलाई ट्र्याक गर्दैनन्। साधारण केस भनेको तपाईंले देखेको उदाहरण हो**, git checkout -b <branch> <remote>/<branch>** चलाउनु। यो एक सामान्य अपरेशन हो र त्यसका लागि Git ले **--track** छोटो रूपमा समर्थन गर्दछ।

# git checkout --track origin/serverfix - - -

जब तपाईं git checkout --track origin/serverfix चलाउनुहुन्छ भने, Git serverfix नामको ब्रान्च सिर्जना गर्छ र यसलाई origin/serverfix ब्रान्चसँग ट्र्याक गर्न सेट गर्दछ। यसको परिणाम स्वरूप, तपाईं नयाँ ब्रान्च serverfix मा स्विच हुन्छन्।

वास्तवमा, यो कामको लागि अर्को छोटो तरीका पनि छ। यदि तपाईंले चेकआउट गर्न खोज्नुभएको ब्रान्चको नाम

(a) अस्तित्वमा छैन र

(b) केवल एक रिमोटमा मात्र मेल खान्छ भने, Git तपाईंको लागि ट्र्याकिङ ब्रान्च सिर्जना गर्छ:

$ git checkout serverfix

Branch serverfix set up to track remote branch serverfix from origin.

Switched to a new branch 'serverfix'

यस अवस्थामा, Git स्वचालित रूपमा origin/serverfix को ट्र्याकिङ ब्रान्च सिर्जना गर्छ र तपाईं serverfix नामको नयाँ ब्रान्चमा स्विच हुनुहुन्छ।

यदि तपाईंले रिमोट ब्रान्चको नामसँग भिन्न नामको स्थानीय ब्रान्च सेट गर्न चाहनुहुन्छ भने, तपाईंले सजिलैसँग पहिलो संस्करणलाई विभिन्न स्थानीय नामसँग प्रयोग गर्न सक्नुहुन्छ:

$ git checkout -b sf origin/serverfix

Branch sf set up to track remote branch serverfix from origin.

Switched to a new branch 'sf'

अब, तपाईंको स्थानीय ब्रान्च sf स्वचालित रूपमा origin/serverfix बाट पुल हुनेछ।

यदि तपाईंको पास पहिले नै एक स्थानीय ब्रान्च छ र तपाईं त्यसलाई रिमोट ब्रान्चसँग सेट गर्न चाहनुहुन्छ जुन तपाईंले हालै डाउनलोड गर्नुभएको हो, वा तपाईंले ट्र्याक गर्दै हुनुहुने अपस्ट्रीम ब्रान्चलाई परिवर्तन गर्न चाहनुहुन्छ भने, तपाईं git branch को -u वा --set-upstream-to विकल्पको प्रयोग गर्न सक्नुहुन्छ जुन यसलाई स्पष्ट रूपमा कुनै पनि समय सेट गर्नका लागि हो।

$ git branch -u origin/serverfix

Branch serverfix set up to track remote branch serverfix from origin.

**Upstream शॉर्टह्याण्ड**

जब तपाईंको पास ट्र्याकिङ ब्रान्च सेटअप हुन्छ, तपाईं यसको अपस्ट्रीम ब्रान्चलाई @{upstream} वा @{u} शॉर्टह्याण्डसँग सन्दर्भ गर्न सक्नुहुन्छ। त्यसो भए, यदि तपाईं master ब्रान्चमा हुनुहुन्छ र यसले origin/master ट्र्याक गर्दैछ भने, तपाईं यसरी भन्न सक्नुहुन्छ:

git merge @{u}

यसले git merge origin/master को सट्टा काम गर्नेछ।

यदि तपाईं देख्न चाहनुहुन्छ कि तपाईंले कुन ट्र्याकिङ ब्रान्चहरू सेटअप गर्नुभएको छ भने, तपाईं git branch सँग -vv विकल्प प्रयोग गर्न सक्नुहुन्छ। यसले तपाईंका स्थानीय ब्रान्चहरूलाई थप जानकारीसहित सूचीबद्ध गर्नेछ, जसमा प्रत्येक ब्रान्च के ट्र्याक गर्दैछ र तपाईंको स्थानीय ब्रान्च अगाडि, पछाडि वा दुवै छ भनेको जानकारी पनि सामेल हुनेछ।

$ git branch -vv

  iss53 7e424c3 [origin/iss53: ahead 2] Add forgotten brackets

  master 1ae2a45 [origin/master] Deploy index fix

\* serverfix f8674d9 [teamone/server-fix-good: ahead 3, behind 1] This should do it

  testing 5ea463a Try something new

यहाँ हामी देख्न सक्छौं कि हाम्रो iss53 ब्रान्च origin/iss53 ट्र्याक गर्दैछ र "ahead" छ दुईको मतलब हामीसँग दुई कमिटहरू छन् जुन सर्भरमा पठाइएको छैन। हामीले देख्न सक्छौं कि हाम्रो master ब्रान्च origin/master ट्र्याक गर्दैछ र यो अपडेटेड छ। त्यसपछि हामी देख्न सक्छौं कि हाम्रो serverfix ब्रान्च teamone/server-fix-good ब्रान्च ट्र्याक गर्दैछ र यो "ahead" छ तीनको र "behind" छ एकको, यसको मतलब सर्भरमा एउटा कमिट छ जसलाई हामीले मर्ज गर्न सकेका छैनौं र तीन कमिटहरू स्थानीय रूपमा छन् जुन हामीले पठाएका छैनौं। अन्तमा हामी देख्न सक्छौं कि हाम्रो testing ब्रान्च कुनै पनि रिमोट ब्रान्च ट्र्याक गर्दै छैन।

महत्वपूर्ण कुरा यो हो कि यी संख्या केवल तपाईंले प्रत्येक सर्भरबाट अन्तिम पटक फेच गर्नुभएको समयदेखि छन्। यस कमाण्डले सर्भरहरूमा पुग्दैन, यो तपाईंलाई ती सर्भरहरूबाट स्थानीय रूपमा क्यास गरिएको जानकारी बताउँदैछ। यदि तपाईंलाई वास्तविक रूपमा अपडेटेड "ahead" र "behind" संख्या चाहिन्छ भने, तपाईंलाई सबै रिमोटहरूबाट फेच गर्नुपर्नेछ र त्यसपछि यो चलाउनुपर्नेछ। तपाईं यसलाई यसरी गर्न सक्नुहुन्छ:

$ git fetch --all; git branch -vv

पुलिंग – Pulling --

जबकि git fetch कमाण्डले तपाईंलाई अझै नभएका सबै परिवर्तनहरू सर्भरबाट ल्याउनेछ, यो तपाईंको कार्य क्षेत्रलाई बिल्कुलै परिवर्तन गर्दैन। यो तपाईंलाई डेटा ल्याएर तपाईंलाई आफैं मर्ज गर्न दिन्छ। तर, git pull नामक एक कमाण्ड छ जुन प्रायः एक git fetch र त्यसपछि git merge को संयोजन हुन्छ। यदि तपाईंले अघिल्लो खण्डमा देखाइएका जस्तो ट्र्याकिङ ब्रान्च सेटअप गर्नुभएको छ भने, चाहे त्यसलाई स्पष्ट रूपमा सेट गरेर होस् वा क्लोन वा चेकआउट कमाण्डहरूले तपाईंको लागि बनाएको होस्, git pull ले तपाईंको वर्तमान ब्रान्च ट्र्याक गर्दैछ भन्ने सर्भर र ब्रान्च हेरेर त्यस सर्भरबाट डेटा ल्याउनेछ र त्यसपछि उक्त रिमोट ब्रान्चलाई मर्ज गर्ने प्रयास गर्नेछ।

सामान्यतया, git pull को जादु प्रायः भ्रमित पार्न सक्छ, त्यसैले git fetch र git merge कमाण्डहरूलाई स्पष्ट रूपमा प्रयोग गर्नु राम्रो हुन्छ।

# रिमोट ब्रान्च मेट्नु - - -

मानौं तपाईं र तपाईंका सहकार्यकर्ताहरूले एउटा रिमोट ब्रान्चको साथ काम सकिसक्नुभएको छ — मानौं तपाईंले एउटा फीचर सम्पन्न गरिसक्नुभएको छ र यसलाई तपाईंको रिमोटको master ब्रान्च (वा जुनसुकै ब्रान्च तपाईंको स्थिर कोडलाइनमा छ) मा मर्ज गर्नुभएको छ। तपाईं रिमोट ब्रान्च मेट्न चाहनुहुन्छ भने git push को --delete विकल्प प्रयोग गर्न सक्नुहुन्छ। यदि तपाईंले सर्भरबाट serverfix ब्रान्च मेट्न चाहनुहुन्छ भने, तलको कमाण्ड चलाउनुहोस्:

$ git push origin --delete serverfix

To https://github.com/schacon/simplegit

 - [deleted] serverfix

यसले मूल रूपमा सर्भरबाट केवल प्वाइन्टरलाई हटाउँछ। Git सर्भरले सामान्यतया डेटा त्यहाँ थोरै समयको लागि राख्छ जबसम्म गार्बेज कलेक्शन चल्दैन, त्यसैले यदि यो गल्तीले मेटियो भने, यो प्रायः सजिलैसँग पुनःप्राप्त गर्न सकिन्छ।

------------------------ END ------------------