Remote संग काम गर्ने --------

कुनै पनि Git प्रोजेक्टमा सहकार्य गर्न सक्षम हुनको लागि, तपाईंलाई तपाईंका Remote Repositories व्यवस्थापन गर्ने तरिका जान्नुपर्छ। Remote Repositories भनेको तपाईंको प्रोजेक्टको त्यस्तो संस्करण हो, जुन इन्टरनेट वा कुनै नेटवर्कमा कहीं होस्ट गरिएको हुन्छ।

तपाईंसँग धेरै Remote Repositories हुन सक्छन्, जसमा प्रायः कुनै एउटा केवल **read-only** हुन्छ भने अर्को **read/write** को लागि तयार हुन्छ। अरूसँग सहकार्य गर्न, ती Remote Repositories व्यवस्थापन गर्नुपर्छ, र आवश्यकता पर्दा आफ्नो काम साझा गर्नको लागि तिनमा Data Push तथा Pull गर्नुपर्छ।

Remote Repositories व्यवस्थापनमा समावेश हुन्छ:

1. Remote Repository थप्ने तरिका।
2. अब मान्य नभएका Remote हटाउने।
3. विभिन्न Remote Branch हरूको व्यवस्थापन गर्ने।
4. तिनीहरूलाई Tracked वा Non-Tracked भनेर परिभाषित गर्ने।

यस खण्डमा, हामी तीमध्ये केही Remote Management का सीपहरू सिक्नेछौं।

Remote Repository तपाईंको लोकल मेसिनमै हुन सक्छ।
यो सम्भव छ कि तपाईं यस्तो "Remote" Repository मा काम गरिरहनु भएको हो, जुन वास्तवमा तपाईंको आफ्नै होस्टमा छ। "Remote" भन्ने शब्दले सधैं Repository नेटवर्क वा इन्टरनेटमा कतै टाढा छ भन्ने संकेत गर्दैन, तर यसले मात्र भन्न खोजेको हो कि यो **अरू कुनै स्थानमा छ।** यस्तो Remote Repository सँग काम गर्दा पनि, अन्य Remote Repository जस्तै, सबै **Standard Pushing, Pulling,** र **Fetching Operations** गर्नु पर्छ।

Showing Your Remotes (रिमोट सर्भरहरू देखाउनुहोस्) ------

रिमोट सर्भरहरू कन्फिगर गरेको देखाउनको लागि, git remote कमाण्ड चलाउन सक्छौं। यसले तिमीले निर्दिष्ट गरेको हरेक रिमोट ह्यान्डलको शॉर्टनेम सूचीबद्ध गर्दछ। यदि तिमीले रिपोजिटरी क्लोन गरेको छौ भने, कम्तीमा origin देखिनेछ, जुन Git ले क्लोन गरेको सर्भरको डिफल्ट नाम हो।

```
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
```

```
$ git remote
origin
```

तिमी -v विकल्प पनि प्रयोग गर्न सक्छौं, जसले Git ले स्टोर गरेको URL देखाउँछ जसलाई रिमोटमा पढ्न र लेख्न प्रयोग गरिन्छ:

```
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
```

यदि तिमीले एक भन्दा धेरै रिमोटहरू कन्फिगर गरेको छौ भने, `git remote -v` कमाण्डले ती सबैको सूची देखाउँछ। उदाहरणका लागि, धेरै सहकर्मीहरूसँग काम गर्नको लागि कन्फिगर गरिएको एक रिपोजिटरी यस्तो देखिन सक्छ:

```
$ cd grit
$ git remote -v
bakkdoor https://github.com/bakkdoor/grit (fetch)
bakkdoor https://github.com/bakkdoor/grit (push)
cho45 https://github.com/cho45/grit (fetch)
cho45 https://github.com/cho45/grit (push)
defunkt https://github.com/defunkt/grit (fetch)
defunkt https://github.com/defunkt/grit (push)
koke git://github.com/koke/grit.git (fetch)
koke git://github.com/koke/grit.git (push)
origin git@github.com:mojombo/grit.git (fetch)
origin git@github.com:mojombo/grit.git (push)
```

यसको मतलब, तिमी यी प्रयोगकर्ताहरूको कुनै पनि योगदान सजिलै *pull* गर्न सक्छौं। साथै, तिमीलाई ती रिमोटहरूमध्ये कुनैमा *push* गर्ने अनुमति पनि हुन सक्छ, तर यो जानकारी यहाँबाट स्पष्ट हुँदैन

Adding Remote Repositories (नयाँ रिमोट रिपोजिटरी थप्ने) - -

हामीले पहिल्यै git clone कमाण्डले कसरी स्वतः origin रिमोट थप्छ भनेर उल्लेख गरिसकेको छौं। नयाँ रिमोट रिपोजिटरी स्पष्ट रूपमा थप्नका लागि, तलको प्रक्रिया अनुसरण गर:

नयाँ रिमोट थप्ने कमाण्ड

```
git remote add <shortname> <url>
```

तिमीले `shortname` को रूपमा एउटा छोटो नाम तोक्न सक्छौं, जसलाई सजिलै रिमोटको रूपमा प्रयोग गर्न सकिन्छ।

उदाहरण

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb https://github.com/paulboone/ticgit (fetch)
pb https://github.com/paulboone/ticgit (push)
```

नयाँ रिमोटबाट डेटा फेच गर्ने

अब तिमी pb छोटो नाम प्रयोग गरेर सम्पूर्ण URL लेख्न नपरी रिमोट फेच गर्न सक्छौं। उदाहरणका लागि, यदि तिमीले Paul को रिमोटबाट सबै जानकारी फेच गर्न चाहन्छौं भने:

```
$ git fetch pb
remote: Counting objects: 43, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 43 (delta 10), reused 31 (delta 5)
Unpacking objects: 100% (43/43), done.
From https://github.com/paulboone/ticgit
 * [new branch]      master     -> pb/master
 * [new branch]      ticgit     -> pb/ticgit
```

नयाँ रिमोटका ब्रान्चहरू

अब Paul को master ब्रान्च स्थानीय रूपमा pb/master को रूपमा उपलब्ध छ। तिमी यसलाई आफ्नो कुनै ब्रान्चमा मर्ज गर्न सक्छौं, अथवा यसलाई चेकआउट गरेर निरीक्षण गर्न सक्छौं।

Fetching and Pulling from Your Remotes (रिमोटबाट फेच र पुल गर्ने)

डेटा फेच गर्ने

रिमोट प्रोजेक्टबाट डेटा प्राप्त गर्न, तिमीले git fetch कमाण्ड चलाउन सक्छौ:

```
$ git fetch <remote>
```

यो कमाण्डले उक्त रिमोट प्रोजेक्टमा गएर त्यो रिमोटमा भएका डेटा ल्याउँछ जुन तिम्रो स्थानीय रिपोजिटरीमा छैन। यसपछि, तिमीले उक्त रिमोटका सबै ब्रान्चहरूको रेफरेन्सहरू हेर्न, मर्ज गर्न, अथवा निरीक्षण गर्न सक्छौ।

उदाहरण

यदि तिमीले रिपोजिटरी क्लोन गरेका छौ भने, origin नाम अन्तर्गत उक्त रिमोट रिपोजिटरी स्वतः थपिन्छ। त्यसैले:

```
$ git fetch origin
```

यो कमाण्डले तिमीले क्लोन गरेको (वा पछिल्लो पटक फेच गरेको) बेला यता उक्त सर्वरमा भएका सबै नयाँ काम फेच गर्छ।

ध्यान दिनुहोस्:

git fetch कमाण्डले केवल डेटा डाउनलोड गर्छ; यसले तिम्रो कुनै काममा स्वतः मर्ज गर्दैन वा हाल काम गरिरहेको कोडलाई परिमार्जन गर्दैन। तिमीले म्यानुअली यसलाई आफ्नो काममा मर्ज गर्नुपर्ने हुन्छ।

डेटा फेच र मर्ज (पुल) गर्ने

यदि तिम्रो हालको ब्रान्च रिमोट ब्रान्चलाई ट्र्याक गर्न सेट गरिएको छ (यो जानकारी अर्को सेक्सनमा हेर्न सकिन्छ), भने तिमी git pull कमाण्ड प्रयोग गर्न सक्छौ।

यो कमाण्डले स्वचालित रूपमा फेच गर्छ र त्यसपछि उक्त रिमोट ब्रान्चलाई तिम्रो हालको ब्रान्चमा मर्ज गर्छ।

उदाहरण

```
$ git pull
```

यो कमाण्डले तिमीले क्लोन गरेको सर्वरबाट डेटा फेच गर्छ र स्वचालित रूपमा तिम्रो काममा मर्ज गर्न प्रयास गर्छ।

जब तिमीले git clone कमाण्ड चलाउँछौ, यो स्वचालित रूपमा तिम्रो स्थानीय master ब्रान्चलाई उक्त सर्वरको रिमोट master (वा डिफल्ट ब्रान्च) ट्र्याक गर्न सेट गर्छ।

मुख्य बुँदा

- **git fetch**: डेटा ल्याउँछ तर मर्ज गर्दैन।
- **git pull**: डेटा ल्याउँछ र त्यसलाई मर्ज गर्ने प्रयास गर्छ।
  तिमीले कुन काम गर्ने निर्भर गर्दछ कि तिमी मर्ज स्वतः चाहन्छौ वा म्यानुअल रूपमा काम गर्न चाहन्छौ।

Configuring git pull Behavior (Git Pull को बिहेवियर कन्फिगर गर्नु) ---

Git Version 2.27 पछि थपिएको चेतावनी

Git 2.27 संस्करणदेखि, यदि **pull.rebase** भेरिएबल सेट गरिएको छैन भने, git pull कमाण्डले चेतावनी देखाउनेछ। Git तिमीलाई यो भेरिएबल सेट गर्न अनुरोध गर्दै रहन्छ। तिमीले यो भेरिएबल सेट गरेर git pull को व्यवहारलाई आफ्नो आवश्यकता अनुसार परिभाषित गर्न सक्छौ।

Default Behavior (डिफल्ट बिहेवियर): Merge Commit

यदि तिमी git pull गर्दा Git लाई पहिले जस्तै (fast-forward मर्ज गर्न सकिन्छ भने त्यसलाई गर्छ, अन्यथा मर्ज कमिट बनाउँछ) काम गर्न दिन चाहन्छौ भने, निम्न कन्फिगरेसन प्रयोग गर:

```
$ git config --global pull.rebase "false"
```

यो **डिफल्ट बिहेवियर** हो।

Rebase Behavior (Rebase गर्न चाहिने अवस्थामा):

यदि तिमी git pull गर्दा **rebase** चाहन्छौ (तिम्रो कामलाई रिमोटको अपडेटहरूमा rebased गर्न), भने यसलाई निम्न प्रकारले सेट गर:

```
$ git config --global pull.rebase "true"
```

यसले git pull गर्दा हरेक पटक मर्जको सट्टा **rebase** प्रक्रिया अपनाउनेछ।

मुख्य निर्णय

- **Merge Commit चाहिन्छ?** pull.rebase लाई false मा सेट गर।
- **Rebase चाहिन्छ?** pull.rebase लाई true मा सेट गर।

कन्फिगरेशन स्थायी रूपमा सेट भएपछि, Git तिमीलाई यो विषयमा फेरि चेतावनी दिने छैन।

Pushing to Your Remotes (तिम्रो Remotes मा Push गर्ने)--

Git Push कमाण्डको प्रयोग

जब तिमी आफ्नो प्रोजेक्टलाई यस्तो अवस्थामा पुऱ्याउँछौ जहाँ तिमीले अरूसँग सेयर गर्न चाहन्छौ, तिमीले यसलाई **upstream** मा पठाउनुपर्छ। यसका लागि प्रयोग हुने कमाण्ड यस्तो छ:

```
$ git push <remote> <branch>
```

Example: Pushing the master branch to origin

यदि तिमी आफ्नो master शाखालाई origin सर्वरमा पठाउन चाहन्छौ, भने निम्न कमाण्ड चलाऊ:

```
$ git push origin master
```

महत्त्वपूर्ण कुरा:

1. **Write Access आवश्यक छ:**

   यो कमाण्ड त्यतिबेला मात्र काम गर्दछ जब तिमीले यस्तो सर्वरबाट क्लोन गरेको छौ जहाँ तिमीलाई **write access** छ।

2. **Conflict Management (सामूहिक काम गर्दा):**
   - यदि तिमी र अर्को व्यक्तिले एउटै समयमा प्रोजेक्ट क्लोन गरेर काम गरेका छौ भने:
     - अर्को व्यक्तिले पहिले push गरेमा, तिम्रो push असफल हुनेछ।
     - यस अवस्थामा, तिमीले पहिले **fetch** गर्नुपर्छ र उनीहरूको कामलाई आफ्नो काममा **merge** गर्नुपर्छ।

ध्यान दिनुपर्ने कुरा:

Git Branching मा यस विषयमा थप जानकारी पाइन्छ, विशेषगरी जब तिमीले धेरै रिमोट ब्रान्चहरूमा काम गरिरहेका छौ।

Inspecting a Remote (Remote को जानकारी हेर्नु) - -

Git Remote Show Command

कुनै विशेष रिमोटको बारेमा विस्तृत जानकारी हेर्न, तिमीले निम्न कमाण्ड प्रयोग गर्न सक्छौ:

```
$ git remote show <remote>
```

Example: Inspecting origin

यदि तिमी origin रिमोटको बारेमा जानकारी लिन चाहन्छौ भने, यो कमाण्ड चलाऊ:

```
$ git remote show origin
```

Output Example:

```
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master tracked
    dev-branch tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
```

```
    master pushes to master (up to date)
```

Output मा के जानकारी पाइन्छ:

1. **Fetch URL:**
   यो URL हो जसबाट Git ले रिमोट डाटा **pull** (डाउनलोड) गर्छ।
2. **Push URL:**
   यो URL हो जसमा Git ले डाटा **push** (अपलोड) गर्छ।
3. **HEAD Branch:**
   यो **default branch** हो, जुन Git ले प्राथमिक रूपमा ट्र्याक गर्छ।
4. **Remote Branches:**
   रिमोटमा भएका ब्रान्चहरूको सूची देखिन्छ (जस्तै master, dev-branch)।
5. **Local Branch Configuration for Pull:**
   यहाँबाट बुझिन्छ कि कुन स्थानीय ब्रान्च **git pull** गर्दा रिमोटको कुन ब्रान्चसँग मर्ज हुन्छ।
6. **Local Ref for Push:**
   यो बताउँछ कि कुन स्थानीय ब्रान्च रिमोटको कुन ब्रान्चमा **push** हुन्छ।

महत्त्वपूर्ण कुरा:

यो कमाण्ड प्रयोग गरेर, तिमीले रिमोटको ट्र्याकिङ सेटअप र ब्रान्चहरूको अवस्था बुझ्न सक्छौ। यसले तिम्रो कामलाई धेरै व्यवस्थित र स्पष्ट बनाउँछ।

Git Remote को जानकारी हेर्ने --

Git मा remote repository को विस्तृत जानकारी पाउन git remote show <remote> कमाण्ड प्रयोग गरिन्छ। यसले remote URL, tracking branches, merge/push सम्बन्धित विवरण देखाउँछ।

कमाण्ड:

```
$ git remote show origin
```

नतिजा:

```
* remote origin
  URL: https://github.com/my-org/complex-project
  Fetch URL: https://github.com/my-org/complex-project
  Push URL: https://github.com/my-org/complex-project
  HEAD branch: master
  Remote branches:
    master tracked
```

```
    dev-branch tracked
    markdown-strip tracked
    issue-43 new (next fetch will store in remotes/origin)
    issue-45 new (next fetch will store in remotes/origin)
    refs/remotes/origin/issue-11 stale (use 'git remote prune' to
remove)
  Local branches configured for 'git pull':
    dev-branch merges with remote dev-branch
    master merges with remote master
  Local refs configured for 'git push':
    dev-branch pushes to dev-branch (up to date)
    markdown-strip pushes to markdown-strip (up to date)
    master pushes to master (up to date)
```

यस नतिजाले दिने जानकारी

1. **Remote Repository को URL**:
   - **Fetch URL**: https://github.com/my-org/complex-project
     यो URL fetch गर्न प्रयोग गरिन्छ।
   - **Push URL**: परिवर्तनहरू server मा पठाउन यो URL प्रयोग गरिन्छ।

2. **HEAD Branch**:
   - master ब्रान्च default HEAD branch हो।

3. **Remote Branches**:
   - **Tracked Branches**:
     Local repository मा track भइरहेका remote branches (जस्तै master, dev-branch, markdown-strip)।
   - **New Branches**:
     issue-43 र issue-45 नयाँ branches हुन्। git fetch गर्दा यी branches remotes/origin/ मा स्टोर हुन्छन्।
   - **Stale Branches**:
     refs/remotes/origin/issue-11 stale branch हो। यसको अर्थ, यो branch server बाट हटाइएको छ तर local मा अझै छ। यसलाई हटाउन:
     ```
     $ git remote prune origin
     ```

4. **Local Branches Configured for Pull**:
   - **dev-branch**: यो branch remote dev-branch सँग merge हुन्छ जब तपाईं git pull चलाउनुहुन्छ।

- **master**: यो branch remote master सँग merge हुन्छ।

5. **Local Branches Configured for Push**:
   - Local branches (dev-branch, markdown-strip, master) remote branches मा Push गर्न तयार छन्।

मुख्य कुरा:

यो कमाण्डले remote repository मा भएका branches, नयाँ branches, stale branches, र merge/push को सेटअपबारे जानकारी दिन्छ। Git repository को remote संग समन्वय गर्न यो धेरै उपयोगी छ।

Remote हटाउने (Removing a Remote) ---

यदि कुनै remote server अब प्रयोगमा छैन भने, तपाईंले git remote remove वा git remote rm प्रयोग गरी त्यसलाई हटाउन सक्नुहुन्छ।

कमाण्ड:

```
$ git remote remove paul
```

नतिजा:

```
$ git remote
origin
```

यो कमाण्डले paul remote हटाउँछ।

हटाएपछि, त्यस remote सँग सम्बन्धित सबै remote-tracking branches र configuration settings पनि मेटिन्छ।

मुख्य कुरा:

1. **Remote को नाम परिवर्तन:**

   ```
   git remote rename <पुरानो नाम> <नयाँ नाम>
   ```

2. **Remote हटाउने:**

   ```
   git remote remove <नाम>
   ```

   या

   ```
   git remote rm <नाम>
   ```

यो प्रक्रियाले Git repository मा remote server व्यवस्थापन गर्न सजिलो बनाउँछ।
