Yet Another GIT Tutorial

by Giso H. Dal

Version 2.2
Outline

1. Introduction
2. Architecture
   - CVCS vs DVCS
   - SVN vs GIT
3. Commands
   - Local Repository
   - Remote Repository
4. Workflow
5. Tips and Tricks
Outline

1 Introduction

2 Architecture
   - CVCS vs DVCS
   - SVN vs GIT

3 Commands
   - Local Repository
   - Remote Repository

4 Workflow

5 Tips and Tricks
Why version control?

For the individual, have you ever:

- Made a change to your code, realised it was a mistake and wanted to revert back?
- Lost code or had a backup that was too old?
- Had to maintain multiple versions of a product?
- Wanted to see the difference between multiple versions of your code?
- Wanted to experiment with a new feature without interfering with working code?
- Wanted to synchronize your work across multiple devices?

For the group, have you ever:

- Wanted to work together on the same project on the same file, at the same time?
- Wanted to share your code, or let other people work on your code?
- Wanted to see how much work is being done, where, when and by whom?
Centralized Version Control
Centralized Version Control

Server

Bob
Phil
Amy
Centralized Version Control

Server

Bob

Phil

Amy
Centralized Version Control

Bob

Phil

Amy

Server

CVCS vs DVCS

SVN vs GIT
Centralized Version Control

Server

Bob

Phil

Amy
Centralized Version Control

Centralized Version Control is a method of managing software development where a single central server holds all the repository data. Changes are synchronized from clients to the server, ensuring all developers have access to the latest version. This typically involves a client-server architecture where clients (Bob, Phil, Amy) interact with a central server.

Clients
- Bob
- Phil
- Amy

Server

CVCS vs DVCS
SVN vs GIT

Commands
Local Repository
Remote Repository

Workflow

Tips and Tricks

Questions
Centralized Version Control

Bob

Phil

Amy

Client-Server
Distributed Version Control

Bob

Phil

Amy
Distributed Version Control

- Bob
- Phil
- Amy
Distributed Version Control
Distributed Version Control
Distributed Version Control

Server

Bob

Phil

Amy

CVCS vs DVCS
SVN vs GIT
Distributed Version Control

- Server
- Bob
- Phil
- Amy

Client-Server vs Peer-to-Peer
Distributed Version Control

Yet Another
GIT Tutorial
by Giso H. Dal

Introduction
Architecture
CVCS vs DVCS
SVN vs GIT
Commands
Local Repository
Remote Repository
Workflow
Tips and Tricks
Questions
Distributed Version Control
Distributed Version Control

CVCS vs DVCS
SVN vs GIT
Distributed Version Control

Client-Server vs Peer-to-Peer

Bob

Phil

Amy
## Subversion vs GIT

<table>
<thead>
<tr>
<th></th>
<th>Subversion</th>
<th>GIT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository Model</strong></td>
<td>Client-Server</td>
<td>Distributed</td>
</tr>
<tr>
<td>Development</td>
<td>Active</td>
<td>Active</td>
</tr>
<tr>
<td>Revision IDs</td>
<td>Numbers</td>
<td>SHA-1 Hashes</td>
</tr>
<tr>
<td>Branching/Merging</td>
<td>Hard; Directory</td>
<td>Easy; Snapshot</td>
</tr>
<tr>
<td>Partial Checkout</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supported Formats</td>
<td>subversion</td>
<td>git, cvs, subversion, hg, any that has a fast-exporter</td>
</tr>
<tr>
<td>Support Large Files</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Slow</td>
<td>Fast</td>
</tr>
<tr>
<td>Bisect</td>
<td>Third party</td>
<td>Yes</td>
</tr>
<tr>
<td>File Locking</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Work Offline</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create Repository</td>
<td>Only admin</td>
<td>Any user</td>
</tr>
<tr>
<td>Reliability</td>
<td>Reasonable</td>
<td>Very</td>
</tr>
<tr>
<td>Repository size</td>
<td>Reasonable</td>
<td>Small</td>
</tr>
<tr>
<td>Merge type</td>
<td>Implicit</td>
<td>Explicit</td>
</tr>
</tbody>
</table>
Subversion Workflow

Bob
- file1
- file2
- file3

Phil
- file1
- file2
- file3

Amy
- file1
- file2
- file3

Commit
Bob
file1
file2
file3

Commit
Phil
file1
file2
file3

Commit
Update
Merge
Amy
file1
file2
file3
Subversion Workflow

Bob
- file1
- file2
- file3

Phil
- file1
- file2
- file3

Amy
- file1
- file2
- file3

Commit
- Bob
- Phil

Update
- Amy

Merge
- Time

Commit
- Bob
- Phil

Commit
- Amy
Subversion Workflow

Commit

Bob

file1 1
file2 1
file3 1

Phil

file1 1
file2 1
file3 1

Amy

file1 1
file2 1
file3 1

Time

1
Subversion Workflow
Subversion Workflow

Bob

file1: 2
file2: 1
file3: 1

Phil

file1: 1
file2: 1
file3: 1

Amy

file1: 1
file2: 1
file3: 1

Commit

Time

1
2
3
4
5
6
Subversion Workflow

1. Bob
   - file1: 2
   - file2: 1
   - file3: 1

2. Phil
   - file1: 1
   - file2: 3
   - file3: 1

3. Amy
   - file1: 1
   - file2: 1
   - file3: 1

Time: 1 2 3 4 5 6
Subversion Workflow

Bob
- file1: 2
- file2: 1
- file3: 1

Phil
- file1: 1
- file2: 3
- file3: 1

Amy
- file1: 1
- file2: 1
- file3: 1

Commit

Time

1
2
3
Subversion Workflow

Bob

<table>
<thead>
<tr>
<th>File</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>file2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>file3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Phil

<table>
<thead>
<tr>
<th>File</th>
<th>1</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>file2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>file3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Amy

<table>
<thead>
<tr>
<th>File</th>
<th>1</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>file2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>file3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Time

1 2 3 4
Subversion Workflow

Bob

<table>
<thead>
<tr>
<th>File</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>2</td>
</tr>
<tr>
<td>file2</td>
<td>1</td>
</tr>
<tr>
<td>file3</td>
<td>1</td>
</tr>
</tbody>
</table>

Phil

<table>
<thead>
<tr>
<th>File</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>1</td>
</tr>
<tr>
<td>file2</td>
<td>3</td>
</tr>
<tr>
<td>file3</td>
<td>1</td>
</tr>
</tbody>
</table>

Amy

<table>
<thead>
<tr>
<th>File</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1</td>
<td>1</td>
</tr>
<tr>
<td>file2</td>
<td>1</td>
</tr>
<tr>
<td>file3</td>
<td>4</td>
</tr>
</tbody>
</table>

Commit

Bob: file1 2, file2 1, file3 1
Phil: file1 1, file2 3, file3 1

Update

Amy: file1 1, file2 1, file3 4
Subversion Workflow

Bob
- file1
  - 2
- file2
  - 1
- file3
  - 1

Phil
- file1
  - 1
- file2
  - 3
- file3
  - 1

Amy
- file1
  - 2
- file2
  - 3
- file3
  - 4

Time

Commit by Bob
Commit by Phil
Commit by Amy
Update
Merge
Yet Another GIT Tutorial
by Giso H. Dal

Introduction
Architecture
CVCS vs DVCS
SVN vs GIT
Commands
Local Repository
Remote Repository
Workflow
Tips and Tricks
Questions

GIT Workflow

Bob

<table>
<thead>
<tr>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Phil

<table>
<thead>
<tr>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Amy

<table>
<thead>
<tr>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
GIT Workflow

Bob
- file1: 1
- file2: 1
- file3: 1

Phil
- file1: 1
- file2: 1
- file3: 1

Amy
- file1: 1
- file2: 1
- file3: 1
GIT Workflow

Bob

<table>
<thead>
<tr>
<th></th>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Phil

<table>
<thead>
<tr>
<th></th>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Amy

<table>
<thead>
<tr>
<th></th>
<th>file1</th>
<th>file2</th>
<th>file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Update

Merge

Time

1

2

3

4
GIT Workflow

Bob
- file1: 2
- file2: 2
- file3: 2

Phil
- file1: 3
- file2: 3
- file3: 3

Amy
- file1: 5
- file2: 5
- file3: 5

Time
1 → 2
2 → 3
3 → 4
4 → 5

Merge

Files:
- file1
- file2
- file3
Yet Another GIT Tutorial
by Giso H. Dal

Introduction
Architecture
CVCS vs DVCS
SVN vs GIT

Commands
Local Repository
Remote Repository

Workflow
Tips and Tricks

Questions

GIT Workflow

Bob
- file1: 2
- file2: 2
- file3: 2

Phil
- file1: 3
- file2: 3
- file3: 3

Amy
- file1: 6
- file2: 6
- file3: 6

Time

1 -> 2
1 -> 3
1 -> 4
2 -> 5
3 -> 5
4 -> 5
5 -> 6
6

Commit
Bob
- file1: 2
- file2: 2
- file3: 2

Commit
Phil
- file1: 3
- file2: 3
- file3: 3

Commit
Amy
- file1: 6
- file2: 6
- file3: 6

Update
Merge
Outline

1. Introduction

2. Architecture
   - CVCS vs DVCS
   - SVN vs GIT

3. Commands
   - Local Repository
   - Remote Repository

4. Workflow

5. Tips and Tricks
The Commands: GIT Architecture

Local

- init
- add
- commit
- status
- log
- diff
- checkout
- branch
- merge
- revert
- reset

Remote

- remote
- branch -u
- clone
- push
- fetch
- pull

Workspace - Staging - Index+Repo - Repo
The Commands

- Local repository
  - Creating local repository
  - Using version control

- Remote repository
  - Creating and connecting to remote repository
  - Setup server without root privileges
Local Repository

1. Committing
2. Status
3. Branching
4. Merging
5. Revert, Reset and Delete
Comitting

1. Committing
2. Status
3. Branching
4. Merging
5. Revert, Reset and Delete
# Committing

<table>
<thead>
<tr>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workspace</strong></td>
<td><strong>Repo</strong></td>
</tr>
<tr>
<td><strong>Staging</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Index+Repo</strong></td>
<td></td>
</tr>
</tbody>
</table>

- **init**
- **add**
- **commit**
- **status**
- **log**
- **diff**
- **checkout**
- **branch**
- **merge**
- **revert**
- **reset**

- **remote**
- **branch -u**
- **clone**
- **push**
- **fetch**
- **pull**

---

**Commands**

<table>
<thead>
<tr>
<th>Local Repository</th>
<th>Remote Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong></td>
<td></td>
</tr>
<tr>
<td><strong>add</strong></td>
<td></td>
</tr>
<tr>
<td><strong>commit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>status</strong></td>
<td></td>
</tr>
<tr>
<td><strong>log</strong></td>
<td></td>
</tr>
<tr>
<td><strong>diff</strong></td>
<td></td>
</tr>
<tr>
<td><strong>checkout</strong></td>
<td></td>
</tr>
<tr>
<td><strong>branch</strong></td>
<td></td>
</tr>
<tr>
<td><strong>merge</strong></td>
<td></td>
</tr>
<tr>
<td><strong>revert</strong></td>
<td></td>
</tr>
<tr>
<td><strong>reset</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

**CVCS vs DVCS**

**SVN vs GIT**

---

**Questions**

**Tips and Tricks**

---

**Workflow**

---

**Introduction**

---

**Architecture**

---

**Yet Another GIT Tutorial**

by Giso H. Dal
## Committing

### Local Repository

### Staging Area

### Index

### Workspace

```
> echo 'hello 1'
```

```
> file1
```

```
> echo 'hello 2'
```

```
> file2
```

```
> git init
```

```
> git add .
```

```
> git commit -m 'initial'
```

```
> echo 'hello 3'
```

```
> file2
```

```
> git add .
```

```
> git commit -m 'updated file2'
```

Committing

> echo 'hello 1' > file1
Committing

> echo 'hello 1' > file1
> 

Local Repository

Staging Area

Index

Workspace

file1

hello 1
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
>
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
>

Local Repository

Staging Area

Index

Workspace

file1
hello 1

file2
hello 2
Committing

- `echo 'hello 1' > file1`
- `echo 'hello 2' > file2`
- `git init`
- `git add .`
- `git commit -m 'initial'`
- `echo 'hello 3' >> file2`
- `git add .`
- `git commit -m 'updated file2'`
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
>
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'

Local Repository

Staging Area

Index

Workspace

file1
hello 1

file2
hello 2
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'
>
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'
> echo 'hello 3' >> file2

Local Repository

Staging Area

Index

Workspace
Committing

> `echo 'hello 1' > file1`
> `echo 'hello 2' > file2`
> `git init`
> `git add .`
> `git commit -m 'initial'`
> `echo 'hello 3' >> file2`
>

![Diagram of Git workflow]

**Local Repository**
- Master

**Staging Area**
- file1
- file2

**Index**
- file1
- file2

**Workspace**
- file1:
  - hello 1
- file2:
  - hello 2
  - hello 3
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'
> echo 'hello 3' >> file2
> git add .

Local Repository

master

1

Staging Area

Index

Workspace

file1

file2

hello 1

hello 2

hello 3
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'
> echo 'hello 3' >> file2
> git add .
>
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add .
> git commit -m 'initial'
> echo 'hello 3' >> file2
> git add .
> git commit -m 'updated file2'
Committing

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> git init
> git add.
> git commit -m 'initial'
> echo 'hello 3' >> file2
> git add.
> git commit -m 'updated file2'
>
## Status

1. Committing
2. Status
3. Branching
4. Merging
5. Revert, Reset and Delete
Status

Local
- Workspace
  - init
  - add
- Staging
  - commit
- Index+Repo
  - status
  - log
  - diff
  - checkout
  - branch
  - merge
  - revert
  - reset

Remote
- Repo
  - remote
  - branch -u
  - clone
  - push
  - fetch
  - pull
Status

Local Repository

Staging Area

Index

Workspace

Yet Another
GIT Tutorial
by Giso H. Dal

Introduction

Architecture
CVCS vs DVCS
SVN vs GIT

Commands

Local Repository
Remote Repository

Workflow

Tips and
Tricks

Questions

> 

echo 'hello 4'

> 

echo 'hello 5'

> 

git status
# On branch master
# Changes not staged.. :
# modified: file1
#

Untracked files:
file3

> 

git log
commit 2
Author: Its Me
Date: Feb 6 15:21:38 2014
updated file2...

> 

git diff file1
diff --git a/file1 b/file1
hello 1
+hello 5

master

Local Repository

1

2

master

Staging Area

Index

file1
file2

Workspace

file1
hello 1

file2
hello 2
hello 3
Status

> echo 'hello 4' > file3
> echo 'hello 4' > file3
>
---

Status

Local Repository

master

1 → 2

Staging Area

Index

1

file1

file2

Workspace

file1

hello 1

file2

hello 2
hello 3

file3

hello 4

Remote Repository

CVCS vs DVCS

SVN vs GIT

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions

Yet Another GIT Tutorial

by Giso H. Dal

Introduction

Architecture

CVCS vs DVCS

SVN vs GIT

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
Status

> echo 'hello 4' > file3
> echo 'hello 5' >> file1
>
Status

> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status
  # On branch master
  # Changes not staged.. :
  #   modified: file1
  #
  # Untracked files:
  #   file3

>
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status
    # On branch master
    # Changes not staged.. :
    # modified: file1
    #
    # Untracked files:
    # file3
> git log

commit 2
Author: Its Me
Date: Feb 6 15:21:38 2014
updated file2
...
Status

> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status
  # On branch master  
  # Changes not staged.. :
  # modified: file1

> git log
  commit 2
  Author: Its Me
  Date: Feb 6 15:21:38 2014
    updated file2
    ...

>
Yet Another GIT Tutorial

by Giso H. Dal

Introduction

Architecture
CVCS vs DVCS
SVN vs GIT

Commands
Local Repository
Remote Repository

Workflow

Tips and Tricks

Questions

Local Repository

Staging Area

Index

Workspace

Status

> echo 'hello 4' > file3
> echo 'hello 5' >> file1

> git status
  # On branch master
  # Changes not staged.. :
  # modified: file1
  #
  # Untracked files:
  # file3

> git log
  commit 2
  Author: Its Me
  Date: Feb 6 15:21:38 2014
    updated file2

  ...

> git diff file1

> echo 'hello 4' > file3
> echo 'hello 5' >> file1

> git status
  # On branch master
  # Changes not staged.. :
  # modified: file1
  #
  # Untracked files:
  # file3

> git log
  commit 2
  Author: Its Me
  Date: Feb 6 15:21:38 2014
    updated file2

  ...

> git diff file1
Status

> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status
  # On branch master
  # Changes not staged.. :
  # modified: file1
  #
  # Untracked files:
  # file3
> git log
  commit 2
  Author: Its Me
  Date: Feb 6 15:21:38 2014
    updated file2
  ...
> git diff file1
  diff --git a/file1 b/file1
  hello 1
  +hello 5
>
Branching
Branching

Local

Workspace

init

add

Remote

Repo

Index+Repo

commit

status

log

diff

checkout

branch

merge

revert

reset

Remote

remote

branch -u

clone

push

fetch

pull

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions

Yet Another GIT Tutorial

by Giso H. Dal

Introduction

Architecture

CVCS vs DVCS

SVN vs GIT

Branching

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions

Yet Another GIT Tutorial

by Giso H. Dal

Introduction

Architecture

CVCS vs DVCS

SVN vs GIT

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions

Yet Another GIT Tutorial

by Giso H. Dal

Introduction

Architecture

CVCS vs DVCS

SVN vs GIT

Commands

Local Repository

Remote Repository

Workflow

Tips and Tricks

Questions
Branching

```
> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4'
> file3
> echo 'hello 5'
> file1
> git status
# On branch foo
# Changes not staged.. :
# modified: file1
#
# Untracked files:
# file3
> git commit -am 'updated file1'
> echo 'hello 6'
> file2
> git commit -am 'updated file2'
```

Local Repository

```
master
```

Staging Area

Index

Workspace

```
file1
hello 1

file2
hello 2
hello 3
```

```
file1

file2
```
Branching

> git checkout 1
Branching

> git checkout 1
>
Branching

> git checkout 1
> git branch foo
Branching

> git checkout 1
> git branch foo

```
> echo 'hello 4'
> file3
> echo 'hello 5'
> file1
> git status
# On branch foo
# Changes not staged:
# modified: file1
#
# Untracked files:
# file3
```
Branching

> git checkout 1
> git branch foo
> git checkout foo

```bash
git checkout 1
```
Branching

> git checkout 1
> git branch foo
> git checkout foo
>
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3

Local Repository

Staging Area

Index

Workspace

Index

file1

file2

Workspace

file1

hello 1

file2

hello 2

Staging Area

1

2

master

foo

Local Repository

1

2

master

foo

Index

file1

file2

Workspace

Index

file1

hello 1

file2

hello 2

Staging Area

1

2

master

foo

Local Repository

1

2

master

foo

Index

file1

file2

Workspace

Index

file1

hello 1

file2

hello 2
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
>
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
>
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

Local Repository

Staging Area

Index

Workspace

file1
hello 1
hello 5

file2
hello 2

file3
hello 4
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

# On branch foo
# Changes not staged.. :
# modified:  file1
#
# Untracked files:
#   file3

>
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

# On branch foo
# Changes not staged.. :
# modified:  file1
#
# Untracked files:
#   file3

> git commit -am 'updated file1'

Local Repository

Staging Area

Index

Workspace
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

```
# On branch foo
# Changes not staged.  :
# modified:  file1
#
# Untracked files:  
#  file3
```

> git commit -am 'updated file1'
>
```
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

# On branch foo
# Changes not staged.. :
#   modified: file1
#
# Untracked files:
#   file3

> git commit -am 'updated file1'
> echo 'hello 6' >> file2
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

  # On branch foo
  # Changes not staged. . . :
  # modified:  file1
  #
  # Untracked files:
  #  file3

> git commit -am 'updated file1'
> echo 'hello 6' >> file2
>
Branching

> `git checkout 1`
> `git branch foo`
> `git checkout foo`
> `echo 'hello 4' > file3`
> `echo 'hello 5' >> file1`
> `git status`

```
# On branch foo
# Changes not staged.. :
#   modified: file1
#   Untracked files:
#     file3
```

> `git commit -am 'updated file1'`
> `echo 'hello 6' >> file2`
> `git commit -am 'updated file2'`
Branching

> git checkout 1
> git branch foo
> git checkout foo
> echo 'hello 4' > file3
> echo 'hello 5' >> file1
> git status

# On branch foo
# Changes not staged.. :
# modified: file1
#
# Untracked files:
#   file3

> git commit -am 'updated file1'
> echo 'hello 6' >> file2
> git commit -am 'updated file2'
>
Merging

1. Committing
2. Status
3. Branching
4. Merging
5. Revert, Reset and Delete
Merging

Local

Workspace

init

add

Staging

commit

status

log

diff

Index+Repo

checkout

merge

revert

reset

Remote

Repo

remote

branch -u

close

push

fetch

pull

Local Repository

Remote Repository

Commands

CVCS vs DVCS

SVN vs GIT

Workflow

Tips and Tricks

Questions

Introduction

Architecture
Merging

```
git checkout master
```

```
git merge foo
```

Automatic merge failed;
fix conflicts and commit.

```
vim file2
```

```
git add .
git commit -m 'foo merged'
```

```
master
foo
```
Merging

> git checkout master
Merging

> git checkout master

>
> git checkout master
> git merge foo
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

>
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

> vim file2
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

> vim file2
>
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

> vim file2
> git add .

Local Repository

master

foo

Staging Area

Index

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3
hello 4

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3
hello 4
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

> vim file2
> git add .
>
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
Merge conflict in file2
Automatic merge failed;
fix conflicts and commit.

> vim file2
> git add .
> git commit -m 'foo merged'
Merging

> git checkout master
> git merge foo

Auto-merging file2
CONFLICT (content):
  Merge conflict in file2
Automatic merge failed;
  fix conflicts and commit.

> vim file2
> git add .
> git commit -m 'foo merged'
>
Revert, Reset and Delete

1. Committing
2. Status
3. Branching
4. Merging
5. Revert, Reset and Delete
Revert, Reset and Delete

Local

Workspace

init
add
commit
status
log
diff
checkout
branch
merge
revert
reset

Staging

Index+Repo

Remote

Repo

init
add
commit
status
log
diff
checkout
branch
merge
revert
reset
remote
branch -u
clone
push
fetch
pull
Revert, Reset and Delete

Local Repository

1 -> 2 -> 5

master

foo

Staging Area

Index

file1
file2
file3

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3
hello 4

Index

file1
file2
file3

Staging Area

file1
file2
file3

Local Repository

1 -> 2 -> 5

master

foo

Yet Another GIT Tutorial
by Giso H. Dal

Commands
Local Repository
Remote Repository

Workflow

Tips and Tricks

Questions

Introduction

Architecture
CVCS vs DVCS
SVN vs GIT
Revert, Reset and Delete

Yet Another GIT Tutorial
by Giso H. Dal

Introduction
Architecture
CVCS vs DVCS
SVN vs GIT
Commands
Local Repository
Remote Repository
Workflow
Tips and Tricks
Questions

> echo 'hello 7' >> file3

Local Repository
master

Staging Area

Index

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3

hello 4
> echo 'hello 7' >> file3
>
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset

Local Repository

- master
- foo

Staging Area

- file1
- file2
- file3

Index

- file1
- file2
- file3

Workspace

- file1:
  - hello 1
  - hello 5
- file2:
  - hello 2
  - hello 3
  - hello 6
- file3:
  - hello 4
  - hello 7
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'

Local Repository

Staging Area

Index

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3
hello 4
hello 7
Revert, Reset and Delete

Yet Another GIT Tutorial by Giso H. Dal

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
>

Local Repository

- master
- foo

Staging Area

- file1
- file2
- file3

Index

Workspace

- file1
  - hello 1
  - hello 5
- file2
  - hello 2
  - hello 3
  - hello 6
- file3
  - hello 4
  - hello 7
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
>

Yet Another
GIT Tutorial
by Giso H. Dal
Introduction
Architecture
CVCS vs DVCS
SVN vs GIT
Commands
Local Repository
Remote Repository
Workflow
Tips and Tricks
Questions

Local Repository

1
2
3
4
5
6

Staging Area

Index

file1
file2
file3

Workspace

file1
hello 1
hello 5

file2
hello 2
hello 3
hello 6

file3
hello 4

master
Revert, Reset and Delete

- `echo 'hello 7'` >> file3
- `git commit -am 'updated file3'`
- `git revert 6`
- `git branch -d foo`
Revert, Reset and Delete

> `echo 'hello 7' >> file3`
> `git commit -am 'updated file3'`
> `git revert 6`
> `git branch -d foo`
>
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset --hard 2
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset --hard 2
>
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset (--mixed) 2
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset (--mixed) 2
>
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset --soft 2
Revert, Reset and Delete

> echo 'hello 7' >> file3
> git commit -am 'updated file3'
> git revert 6
> git branch -d foo
> git reset --soft 2
>
Remote Repository

1. Creating and Connecting
2. Push and Pull
Creating and Connecting

1. Creating and Connecting
2. Push and Pull
Creating and Connecting

<table>
<thead>
<tr>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workspace</strong></td>
<td><strong>Repo</strong></td>
</tr>
<tr>
<td><strong>Staging</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Index + Repo</strong></td>
<td></td>
</tr>
<tr>
<td><code>init</code></td>
<td></td>
</tr>
<tr>
<td><code>add</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>commit</code></td>
</tr>
<tr>
<td></td>
<td><code>status</code></td>
</tr>
<tr>
<td></td>
<td><code>log</code></td>
</tr>
<tr>
<td></td>
<td><code>diff</code></td>
</tr>
<tr>
<td></td>
<td><code>checkout</code></td>
</tr>
<tr>
<td></td>
<td><code>branch</code></td>
</tr>
<tr>
<td></td>
<td><code>merge</code></td>
</tr>
<tr>
<td></td>
<td><code>revert</code></td>
</tr>
<tr>
<td></td>
<td><code>reset</code></td>
</tr>
<tr>
<td><code>remote</code></td>
<td></td>
</tr>
<tr>
<td><code>branch -u</code></td>
<td></td>
</tr>
<tr>
<td><code>clone</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>push</code></td>
</tr>
<tr>
<td></td>
<td><code>fetch</code></td>
</tr>
<tr>
<td></td>
<td><code>pull</code></td>
</tr>
</tbody>
</table>
Creating and Connecting

Remote does not exists:

local>

Local Repository

Staging Area

1 2

file1 file2

Remote Repository

master

Cloning into 'project'...
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
```

Remote Repository

1. `master`
2. ` LOCAL REPOSITORY`

Staging Area

```
file1
file2
```

Local Repository

1. `LOCAL REPOSITORY`

```
Remote does not exists:
local> ssh user@lilo.science.ru.nl
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote>
```
Creating and Connecting

**Remote does not exists:**

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
```

```
local> git remote add origin
remote> git init --bare
remote> logout
local> git push origin master
local> git branch -u origin/master master
```

**Remote exists:**

```
local> git clone user@lilo.science.ru.nl:~/project.git
```

Cloning into 'project'...

```
local> (upstream)
master
```

Local Repository

```
1
master
```

Staging Area

```
file1
file2
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote>
```

```
local> git remote add origin \\
user@lilo.science.ru.nl:~/project.git
local>
local> git push origin master
local>
local> git branch -u origin/master master
```

Remote exists:

```
local> git clone \\
user@lilo.science.ru.nl:~/project.git
```

Cloning into 'project'...

```
local>
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
```

Remote Repository

Staging Area

Local Repository

1 2

- master

file1 file2
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote>
```

Remote Repository

```
1
master
```

Local Repository

```
2
```

Staging Area

```
file1
file2
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote>
```

Remote exists:

```
local> git clone user@lilo.science.ru.nl:~/project.git
Cloning into 'project'...
remote>
(upstream)
master

Local Repository

1
master

file1 file2

Staging Area

Remote Repository

origin
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
```

1. `master`
2. `origin/master`
Creating and Connecting

**Remote does not exists:**

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local>
```

```diff
1
2

Local Repository

```
master
```

Staging Area

```
file1
file2
```

Remote Repository

```
origin
```
Creating and Connecting

Remote does not exists:

local > ssh user@lilo.science.ru.nl
remote > mkdir ~/project.git
remote > cd ~/project.git
remote > git init --bare
remote > logout
local > git remote add origin \ user@lilo.science.ru.nl:~/project.git

local > git push origin master
local > git branch -u origin/master
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \user@lilo.science.ru.nl:~/project.git
local>
```

Remote Repository

Local Repository

```
1 2
master
```

Staging Area

```
file1  file2
```

Remote Repository

```
origin
```
Remote does not exists:

```bash
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \
    user@lilo.science.ru.nl:~/project.git
local> git push origin master
```

Cloning into 'project'...
```bash
local> git clone \
    user@lilo.science.ru.nl:~/project.git
```

```bash
Remote exists:
local> git clone \
    user@lilo.science.ru.nl:~/project.git
Cloning into 'project'...
```
Creating and Connecting

Remote does not exists:

```bash
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \
    user@lilo.science.ru.nl:~/project.git
local> git push origin master
local>
```
Creating and Connecting

Remote does not exists:

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \
    user@lilo.science.ru.nl:~/project.git
local> git push origin master
local> git branch -u \
    origin/master master
```
Creating and Connecting

Remote does not exists:

local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin
      user@lilo.science.ru.nl:~/project.git
local> git push origin master
local> git branch -u \
      origin/master master
local>

Remote exists:

local> git clone
      user@lilo.science.ru.nl:~/project.git
Cloning into 'project'...
Creating and Connecting

Remote does not exists:

```bash
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \
    user@lilo.science.ru.nl:~/project.git
local> git push origin master
local> git branch -u \n    origin/master master
local>
```

Remote exists:

```bash
local>
```
Creating and Connecting

Remote does not exists:

```bash
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \\
    user@lilo.science.ru.nl:~/.project.git
local> git push origin master
local> git branch -u \\
    origin/master master
local>
```

Remote exists:

```bash
local> git clone \\
    user@lilo.science.ru.nl:~/.project.git
```
Creating and Connecting

**Remote does not exist:**

```
local> ssh user@lilo.science.ru.nl
remote> mkdir ~/project.git
remote> cd ~/project.git
remote> git init --bare
remote> logout
local> git remote add origin \ user@lilo.science.ru.nl:~/project.git
local> git push origin master
local> git branch -u \ origin/master master
local>
```

**Remote exists:**

```
local> git clone \ user@lilo.science.ru.nl:~/project.git
```

Cloning into 'project'...

local>
Push and Pull

1. Creating and Connecting
2. Push and Pull
# Push and Pull

<table>
<thead>
<tr>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workspace</strong></td>
<td><strong>Repo</strong></td>
</tr>
<tr>
<td><strong>Staging</strong></td>
<td><strong>Index+Repo</strong></td>
</tr>
</tbody>
</table>

- `init`
- `add`
- `commit`
- `status`
- `log`
- `diff`
- `checkout`
- `branch`
- `merge`
- `revert`
- `reset`

- `remote`
- `branch -u`
- `clone`
- `push`
- `fetch`
- `pull`

---

**CVCS vs DVCS**

**SVN vs GIT**

**Commands**

**Local Repository**

**Remote Repository**

**Workflow**

**Tips and Tricks**

**Questions**
Push and Pull

Local is ahead of Remote

> 

```
> git push
> Remote is ahead of Local
> git pull
> git pull = git fetch & git merge (upstream)
```
Local is ahead of Remote

> git push
Local is ahead of Remote

> git push
>

Remote is ahead of Local

$ git pull = git fetch & git merge (upstream)

Local Repository

1 2

master

Staging Area

file1 file2

Remote Repository

1 2

master

origin
Push and Pull

Local is ahead of Remote

> git push

Remote is ahead of Local

>
Push and Pull

Local is ahead of Remote

> git push

Remote is ahead of Local

> git pull

Git push

Remote is ahead of Local

> git pull

Git pull = git fetch & git merge (upstream)
Push and Pull

Local is ahead of Remote

> git push
>

Remote is ahead of Local

> git pull
>

git pull = git fetch & git merge
Remote Repository

Local

Workspace

init
add

Staging
commit
status
log
diff
checkout
branch
merge
revert
reset

Index+Repo

Remote

Repo

remote
branch -u
clone
push
fetch
pull
The Commands

Local

- init
- add
- commit
- status
- log
- diff
- checkout
- branch
- merge
- revert
- reset

Remote

- remote
- branch -u
- clone
- push
- fetch
- pull
The Commands: Committing

Local
- Workspace
- Staging
- Index+Repo

Remote
- Repo

- init*
- add*
- commit*
- status
- log
- diff
- checkout
- branch
- merge
- revert
- reset
- remote
- branch -u
- clone
- push
- fetch
- pull
The Commands: **Status**

<table>
<thead>
<tr>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workspace</strong></td>
<td><strong>Repo</strong></td>
</tr>
<tr>
<td><strong>Staging</strong></td>
<td><strong>Index+Repo</strong></td>
</tr>
<tr>
<td>init</td>
<td>remote</td>
</tr>
<tr>
<td>add</td>
<td>branch -u</td>
</tr>
<tr>
<td>commit</td>
<td>clone</td>
</tr>
<tr>
<td>status*</td>
<td>push</td>
</tr>
<tr>
<td>log*</td>
<td>fetch</td>
</tr>
<tr>
<td>diff*</td>
<td>pull</td>
</tr>
<tr>
<td>checkout</td>
<td>revert</td>
</tr>
<tr>
<td>branch</td>
<td>reset</td>
</tr>
<tr>
<td>merge</td>
<td></td>
</tr>
</tbody>
</table>
The Commands: Branching

**Local**
- **Workspace**
  - `init`
  - `add`
  - `commit`
  - `status`
  - `log`
  - `diff`
  - `checkout*`
  - `branch*`
  - `merge*`
  - `revert`
  - `reset`

**Staging**
- `commit`
- `status`
- `log`
- `diff`
- `checkout*`
- `branch*`
- `merge*`
- `revert`
- `reset`

**Index+Repo**
- `commit`
- `status`
- `log`
- `diff`
- `checkout*`
- `branch*`
- `merge*`
- `revert`
- `reset`

**Remote**
- `remote`
- `branch -u`
- `clone`
- `push`
- `fetch`
- `pull`
The Commands: Alter Commit History

Local

<table>
<thead>
<tr>
<th>Workspace</th>
<th>Staging</th>
<th>Index+Repo</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>add</td>
<td>commit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>log</td>
</tr>
<tr>
<td></td>
<td></td>
<td>diff</td>
</tr>
<tr>
<td></td>
<td></td>
<td>checkout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>branch*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>merge</td>
</tr>
<tr>
<td></td>
<td></td>
<td>revert*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset*</td>
</tr>
</tbody>
</table>

Remote

<table>
<thead>
<tr>
<th>Repo</th>
</tr>
</thead>
<tbody>
<tr>
<td>remote</td>
</tr>
<tr>
<td>branch -u</td>
</tr>
<tr>
<td>clone</td>
</tr>
<tr>
<td>push</td>
</tr>
<tr>
<td>fetch</td>
</tr>
<tr>
<td>pull</td>
</tr>
</tbody>
</table>
The Commands: Connect to Remote

Local

Workspace

Staging

Index+Repo

Remote

Repo

init

add

commit

status

log

diff

checkout

branch

merge

revert

reset

remote*

branch -u*

clone*

push

fetch

pull
The Commands: Local-Remote Data Exchange

Local

- **Workspace**
  - init
  - add
  - commit
  - status
  - log
  - diff
  - checkout
  - branch
  - merge
  - revert
  - reset

- **Staging**

- **Index+Repo**

Remote

- **Repo**
  - remote
  - branch -u
  - clone
  - push*
  - fetch*
  - pull*
Outline

1. Introduction
2. Architecture
   - CVCS vs DVCS
   - SVN vs GIT
3. Commands
   - Local Repository
   - Remote Repository
4. Workflow
5. Tips and Tricks
We have seen WHAT commands to use, a workflow determines HOW to use them.
Workflow: The Wrong Way!

1. Initial commit
2. First release
3. New function with bug
4. Bug fixed
5. Minor Improvement
6. Second release
### Workflow: Adding Structure

<table>
<thead>
<tr>
<th>master</th>
<th>develop</th>
<th>feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial commit</td>
<td>Working version</td>
<td>First release</td>
</tr>
<tr>
<td>New function with bug</td>
<td>Bug fixed</td>
<td>Update working version</td>
</tr>
<tr>
<td>Minor improvement</td>
<td>Second release</td>
<td></td>
</tr>
</tbody>
</table>
## Workflow: Adding Structure

<table>
<thead>
<tr>
<th>master</th>
<th>develop</th>
<th>feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Initial commit" /></td>
<td><img src="image" alt="1" /></td>
<td></td>
</tr>
</tbody>
</table>
Workflow: Adding Structure

- Initial commit
- Working version

Diagram:
- master
- develop
- feature

1. Initial commit
2. Working version
Workflow: Adding Structure

1. Initial commit
2. Working version
3. First release
Workflow: Adding Structure

- Initial commit
- Working version
- First release
- New function with bug
Workflow: Adding Structure

Initial commit
Working version
First release
New function with bug
Bug fixed

master  develop  feature
Workflow: Adding Structure

Initial commit
Working version
First release
New function with bug
Bug fixed
Update working version
Workflow: Adding Structure

1. Initial commit
2. Working version
3. First release
4. New function with bug
5. Bug fixed
6. Update working version
7. Minor improvement
Workflow: Adding Structure

- Initial commit
- Working version
- First release
- New function with bug
- Bug fixed
- Update working version
- Minor improvement
- Second release
Outline

1 Introduction

2 Architecture
   - CVCS vs DVCS
   - SVN vs GIT

3 Commands
   - Local Repository
   - Remote Repository

4 Workflow

5 Tips and Tricks
Tips and Tricks

- **Best Practice Tips**
- **Ignore (Temporary) Files or Directories**
- **Create Custom Commands**
Tips and Tricks

Best practice

1. Use a `development` branch next to a `release` branch.
2. Pull before Push.
3. Append `.git` to directory of remote repository, e.g., `project.git`.
4. GIT server without root privileges: use `Gitolite` to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

Cheat Sheet: http://ndpsoftware.com/git-cheatsheet.html

All configurations and customizations are located in `~/.gitconfig`
Tips and Tricks

**Best practice**

1. Use a **development** branch next to a **release** branch.
2. Pull before Push.
3. Append `.git` to directory of remote repository, e.g., `project.git`.
4. GIT server without root privileges: use **Gitolite** to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

- Cheat Sheet: http://ndpsoftware.com/git-cheatsheet.html
- All configurations and customizations are located in `~/.gitconfig`
Best practice

1. Use a development branch next to a release branch.
2. Pull before Push.
3. Append .git to directory of remote repository, e.g., project.git.
4. GIT server without root privileges: use Gitolite to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

Cheat Sheet: http://ndpsoftware.com/git-cheatsheet.html

All configurations and customizations are located in ~/.gitconfig
Tips and Tricks

- **Best practice**
  1. Use a `development` branch next to a `release` branch.
  2. Pull before Push.
  3. Append `.git` to directory of remote repository, e.g., `project.git`.
  4. GIT server without root privileges: use Gitolite to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)


- All configurations and customizations are located in `~/.gitconfig`
Tips and Tricks

■ **Best practice**

1. Use a **development** branch next to a **release** branch.
2. Pull before Push.
3. Append `.git` to directory of remote repository, e.g., `project.git`.
4. GIT server without root privileges: use **Gitolite** to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

■ **Cheat Sheet**: [http://ndpsoftware.com/git-cheatsheet.html](http://ndpsoftware.com/git-cheatsheet.html)

■ All configurations and customizations are located in `~/.gitconfig`
Tips and Tricks

■ Best practice

1. Use a development branch next to a release branch.
2. Pull before Push.
3. Append .git to directory of remote repository, e.g., project.git.
4. GIT server without root privileges: use Gitolite to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

■ Cheat Sheet: http://ndpsoftware.com/git-cheatsheet.html

■ All configurations and customizations are located in ~/.gitconfig
Tips and Tricks

- **Best practice**
  1. Use a *development* branch next to a *release* branch.
  2. Pull before Push.
  3. Append `.git` to directory of remote repository, e.g., `project.git`.
  4. GIT server without root privileges: use **Gitolite** to allow other people access to repositories on your account. (or the hard way: restricted SSH keys in combination with an access script.)

- **Cheat Sheet**: [http://ndpsoftware.com/git-cheatsheet.html](http://ndpsoftware.com/git-cheatsheet.html)

- All configurations and customizations are located in `~/.gitconfig`
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

>
**Ignoring (Temporary) Files or Directories**

All files in `.gitignore` will be ignored:

> `echo 'hello 1' > file1`
**Ignoring (Temporary) Files or Directories**

All files in `.gitignore` will be ignored:

```bash
> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
```

```bash
$ git init
$ git add .
$ git commit -m 'initial'
```

*Ignore with regex:*  
Ignore vim swap files, e.g., `file1.swp`, `file1.swo`, `file1...`  
Ignore editor backup files  
Ignore OS generated files  
`*.sw*`  
`*~`
Ignoring (Temporary) Files or Directories

All files in `.gitignore` will be ignored:

```plaintext
> echo 'hello 1' > file1
> echo 'hello 2' > file2
```
All files in `.gitignore` will be ignored:

```bash
> echo 'hello 1' > file1
> echo 'hello 2' > file2
>```

All files in `.gitignore` will be ignored:

```bash
> echo 'hello 1' > file1
> echo 'hello 2' > file2
>```
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
Ignoring (Temporary) Files or Directories

All files in `.gitignore` will be ignored:

```
> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> 
```
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
>

Local Repository

Staging Area

Index

Workspace

file1
hello 1

file2
hello 2

.gitignore
file2
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
Ignoring (Temporary) Files or Directories

**All files in *.gitignore will be ignored:**

```bash
> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
>```

---

**Local Repository**

**Staging Area**

- `file1`
- `.giti*`

**Index**

**Workspace**

- `file1`
  - `hello 1`
- `file2`
  - `hello 2`
- `.gitignore`
  - `file2`
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

- `echo 'hello 1' > file1`
- `echo 'hello 2' > file2`
- `echo 'file2' > .gitignore`
- `git init`
- `git add .`
- `git commit -m 'initial'`
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'
>
Ignoring (Temporary) Files or Directories

All files in `.gitignore` will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
- Ignore editor backup files
- Ignore OS generated files
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'
>

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
- Ignore editor backup files
- Ignore OS generated files
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
  * sw?
- Ignore editor backup files
- Ignore OS generated files

Local Repository

master

1

Staging Area

Index

Workspace

- file1
  hello 1

- file2
  hello 2

- .gitignore
  file2
  * sw?
Ignoring (Temporary) Files or Directories

All files in `.gitignore` will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
  `*.swp`?
- Ignore editor backup files
- Ignore OS generated files
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

> echo 'hello 1' > file1
> echo 'hello 2' > file2
> echo 'file2' > .gitignore
> git init
> git add .
> git commit -m 'initial'
>

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
  * .sw?
- Ignore editor backup files
  * ~
- Ignore OS generated files
Ignoring (Temporary) Files or Directories

All files in .gitignore will be ignored:

- echo 'hello 1' > file1
- echo 'hello 2' > file2
- echo 'file2' > .gitignore
- git init
- git add .
- git commit -m 'initial'

Ignore with regex:

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
  * .sw?
- Ignore editor backup files
  * ~
- Ignore OS generated files

Local Repository

Staging Area

Index

Workspace

file1
- hello 1

file2
- hello 2

.gitignore
- file2
  * .sw?
  * ~
Ignoring (Temporary) Files or Directories

**All files in .gitignore will be ignored:**

- `echo 'hello 1' > file1`
- `echo 'hello 2' > file2`
- `echo 'file2' > .gitignore`
- `git init`
- `git add .`
- `git commit -m 'initial'`

**Ignore with regex:**

- Ignore vim swap files, e.g., file1.swp, file1.swo, file1...
  
  `*.sw?`

- Ignore editor backup files
  `*~`

- Ignore OS generated files
  `.DS_Store`
  `Thumbs.db`

---

**Local Repository**

- Master
  
  1

**Index**

- file1
  
  .gitignore

**Workspace**

- file1
  - hello 1

- file2
  - hello 2

- .gitignore
  - file2
    - `*.sw?`
    - `*~`
    - `.DS_Store`
    - Thumbs.db
Create Custom Commands

- **Aliases to commands** in ~/.gitconfig, i.e.

  ```
  [alias]
  st = status
  co = commit
  ```

- **Aliases to functions** (you can find these in my config)

  ```
  git ls
  List files under version control
  git tar
  Create tar file from files under version control
  git commit-now (alias: 'cn')
  Commit with timestamp as message
  git commit-now-push (alias: 'cnp')
  Push after a commit-now
  git tree
  Show commit history as colored tree structure
  ```
Create Custom Commands

- Aliases to commands in ~/.gitconfig, i.e.

  [alias]
  
  st = status
  
  co = commit

- Aliases to functions

  (you can find these in my config)

  - git ls
    
    List files under version control
  
  - git tar
    
    Create tar file from files under version control
  
  - git commit-now (alias: 'cn')
    
    Commit with timestamp as message
  
  - git commit-now-push (alias: 'cnp')
    
    Push after a commit-now
  
  - git tree
    
    Show commit history as colored tree structure
Create Custom Commands

- **Aliases to commands in */.gitconfig*, i.e.

  
  [alias]
  
  `st = status`
  `co = commit`

- **Aliases to functions** *(you can find these in my config)*

  - `git ls`
    *List files under version control*
  - `git tar`
    *Create tar file from files under version control*
  - `git commit-now` (alias: ‘cn’)
    *Commit with timestamp as message*
  - `git commit-now-push` (alias: ‘cnp’)
    *Push after a commit-now*
  - `git tree`
    *Show commit history as colored tree structure*
Create Custom Commands

- Aliases to commands in ~/.gitconfig, i.e.
  [alias]
  st = status
  co = commit

- Aliases to functions (you can find these in my config)
  - git ls
    List files under version control
  - git tar
    Create tar file from files under version control
  - git commit-now (alias: 'cn')
    Commit with timestamp as message
  - git commit-now-push (alias: 'cnp')
    Push after a commit-now
  - git tree
    Show commit history as colored tree structure
Create Custom Commands

'git tree' instead of 'git log'
Create Custom Commands
'git tree' instead of 'git log'

```
silveru:../git/usecase
● git log
```

```
silveru 0 | [1 bash] [2 bash] [3 bash] [4 bash] 18-03-2014 16:46
```
Create Custom Commands

'git tree' instead of 'git log'

```
commit 03d95fb493df334334d152572f1e06008c83531d (HEAD, master)
Author: Committer Name <committer.name@cs.ru.nl>
Date:   Tue Mar 18 16:45:28 2014 +0100

    updated file3

commit ef1e3a3fd5514c9c2381422c52a7af9df3d46db7
Merge: 9f7b78c a840934
Author: Committer Name <committer.name@cs.ru.nl>
Date:   Tue Mar 18 16:45:11 2014 +0100

    foo merged

commit a840934359b4e9d507bac5bcf4810fb77a002dbc (foo)
Author: Committer Name <committer.name@cs.ru.nl>
Date:   Tue Mar 18 16:44:39 2014 +0100

    updated file2

commit acbfb299b0407fd9b05d2af611e1492cf85356fb

silveru 0 | [1 git] [2 bash] [3 bash] [4 bash] 18-03-2014 16:46
```
Create Custom Commands

'git tree' instead of 'git log'

silveru:~/git/usecase

silveru 0 | [1 git] [2 bash] [3 bash] [4 bash]  18-03-2014 16:46
Create Custom Commands

'git tree' instead of 'git log'

silveru:../git/usecase
  ● git tree

silveru 0 | [1 git] [2 bash] [3 bash] [4 bash] 18-03-2014 16:46
Create Custom Commands

'git tree' instead of 'git log'

```bash
silveru:../git/usecase
  ● git tree

* 03d95fb  (HEAD, master)
 |   [66 seconds ago]  Committer Name: updated file3
* ef1e3a3
 | \ [83 seconds ago]  Committer Name: foo merged
 | \ a840934  (foo)
 | | [2 minutes ago]  Committer Name: updated file2
 | \ * acbfb29
 | | [3 minutes ago]  Committer Name: updated file1
 | | 9f7b78c [4 minutes ago]  Committer Name: updated file2
 | | 6b1b69b [5 minutes ago]  Committer Name: initial

silveru:../git/usecase
  ●

silveru 0 | [1 git] [2 bash] [3 bash] [4 bash] 18-03-2014 16:46
```
Create Custom Commands

'git vimdiff' instead of 'git diff'
Create Custom Commands

`git vimdiff` instead of `git diff`
Create Custom Commands

'git vimdiff' instead of 'git diff'

```bash
silveru:./git/usecase
  ● git diff 6b1b69b 03d95fb file1
diff --git a/file1 b/file1
index e48df24..088e365 100644
--- a/file1
+++ b/file1
@@ -1 +1,2 @@
  hello 1
+hello 5

silveru:./git/usecase
  ●
```

silveru 0 | [1 git] [2 bash] [3 bash] [4 bash] 18-03-2014 16:47
Create Custom Commands

'git vimdiff' instead of 'git diff'

silveru:../git/usecase
Create Custom Commands

'git vimdiff' instead of 'git diff'
Create Custom Commands

'git vimdiff' instead of 'git diff'
Questions?

Email me at
gdal at cs.ru.nl
for any further questions.