ToolBox

Table of Contents

Linux

Keyboard Shortcuts

Table 1. Keyboard Shortcuts

Key Combination	Description
`CTRL+ALT+F1`	Open virtual terminal 1
`ALT+F7`	Return from virtual terminal to gui
`ALT+RIGHT CLICK+Drag`	Resize window in xfce

Git

Normal Flow

Update local from remote, push local commits

```
git checkout master     # switch to `master` branch and update working directory
git fetch --prune       # fetch from remote and remove local branches if removed from
upstream
git rebase              # replay local commits ontop of updated local branch
git push origin master  # push current HEAD to remote master branch
```

`git pull` is shorthand for `git fetch` followed by `git merge FETCH_HEAD`. Use `git pull --rebase` to do a rebase rather than a merge.

Update local from remote, merge local development branch

```
git checkout master     # Switches to `master` branch and updates the working
directory
git pull                # to update the state to the latest remote master state
git merge develop       # to bring changes to local master from your develop branch
git push origin master  # push current HEAD to remote master branch
```

Configuration

Hierarchy of configuration levels:

Modifier	Location	Description
`--local`	`.git/config`	local to a repo
`--global`	`~/.gitconfig`	global to a user
`--system`	`/etc/gitconfig`	system wide on OS

Configuration keys are "dot"-delimited strings formed in a hierarchical fashion.

Configuration (and aliases)

```
git config -l           # list configuration values (includes global)
git config --global -l  # list global configuration values (not local)

git config user.email   # list the configured value for key
git config user.email "1091512+gheinze@users.noreply.github.com"  # set email address

# set email address at user level (i.e. across repos)
git config --global user.email "1091512+gheinze@users.noreply.github.com"

git config --global core.editor "vim"
git config --global merge.tool kdiff3
git config --global color.ui auto
git config --global alias.ci commit
git config --global http.proxy
git config --global https.proxy
git config --global alias.amend 'commit --amend --no-edit'
#git config --global alias.hist "log --oneline --graph --decorate --all"
git config --global alias.hist 'log --graph --abbrev-commit
--pretty=format:'"'"'%Cgreen%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold
blue)<%an>%Creset'"'"''
```

You may wish to alias a remote. Or perhaps the name/location of the remote repo changed and you need to update the local configuration to reflect that.

Remote

```
git remote -v      # list the remote repositories

# add a remote endpoint for current repo
git remote add origin https://github.com/gheinze/debenture.git

# Change the url of the remote endpoint
git remote set-url origin https://github.com/accounted4/toolbox

git remote show origin
```

As `origin` is just an alias for a url, you could specify the url directly rather than using the alias. One can also add a `username:password` to the url. Or just `username` to be prompted for the password. Or use a credential manager for have the values looked up.

```
git fetch https://53gheinze@git.sonova.com/tloeg/sonovaDigitalCommerce master
```

Log

show recent commits

```
git log --oneline --graph --decorate --all --pretty=format:'%C(bold blue)<%an>%Creset'
git log --graph --abbrev-commit --pretty=format:'%Cgreen%h%Creset -%C(yellow)%d%Creset
%s %Cgreen(%cr) %C(bold blue)<%an>%Creset'
```

show files in a commit

```
git diff-tree --no-commit-id --name-only -r <sha>
git show --pretty="" --name-only  <sha>
```

Diffing

Diff back one commit

```
git log -p -1 [<source> | <commit>]
```

Diff staged file against HEAD

```
git diff --cached
```

Note: `--staged` is a synonym for `--cached`

Use tools such as `git difftool` or `git mergetool` to show a gui window with tool such as P4win set in global config. Or `gitk` to show commits.

Reverting

Reverting

```
# Remove file from the index (i.e. the "about to be committed" area)
# without changing anything els (i.e. leave modified file modified in working dir)
git reset filename.txt

# Un-modify a file in the workspace
git checkout -- filename.txt

# Update both the working copy of `filename.txt` and its state in the index with that
from HEAD
git checkout HEAD -- filename.txt
```

An example of pretending the last commit did not happen...

```
# Moving HEAD before the last commit
git reset --soft HEAD^   # moves HEAD to previous commit, doesn't touch index or work
area
git push origin +master  # forces this push to master, orphaning the previous HEAD
```

A StackOverflow example:

```
git add -A; git commit -m "Start here."
git add -A; git commit -m "One"
git add -A; git commit -m "Two"
git add -A' git commit -m "Three"
git log --oneline --graph -4 --decorate

* da883dc (HEAD, master) Three
* 92d3eb7 Two
* c6e82d3 One
* e1e8042 Start here.

git reset --soft HEAD~3
git log --oneline --graph -1 --decorate

* e1e8042 Start here.

# Now all your changes are preserved and ready to be committed as one.
```

Rebasing

In the following example `feature1` is branched off of `master`. `master` continues on. `feature1` continues on. Now we want to replay the `feature1` commits on top of the new `master` HEAD. Then we move the `master` HEAD to be at the same location as `feature1` HEAD. Essentially we are moving all the commits from `feature1` onto the `master` branch. If you want `feature1` commits squashed into a single commit to be added to the `master` branch, then try merging instead of rebasing.

feature branch

```
    master
     |
A -> D
  \
    B -> C
         |
      feature1
```

rebase

```
    master   feature1
     |          |
A -> D -> B' -> C'
```

merge

```
          master, feature1
                |
A -> D -> B' -> C'
```

Rebasing

```
git commit                # on master [A]
git checkout -b feature1  # checkout feature1 (and create the branch if it doesn't
exist)
git commit                # on feature1 [B]
git commit                # on feature1 [C]
git checkout master
git commit                # on master [D]
git checkout feature1
git rebase master         # replay feature1 commits ontop of current master to create
revised feature1 [B', C']
git checkout master
git merge feature1        # since feature1 is linearly ahead of master now, move the
head of master to head of feature1
```

Stashing

Stashing

```
# Push local modifications to a new stash entry and roll back to HEAD (in working tree
and index)
git stash push -m "My interrupted work"

git stash list
git stash pop
```

Amending

RULE: **Don’t append public commits.**

```
# Change the message of the last commit:
git commit --amend -m "an updated commit message"

# Adding a missed file to the last commit: stage the missed file, then commit with:
# the "no-edit" will prevent prompting for a commit message and keep it the same as
the original commit.
git commit --amend --no-edit
```

Essentially, with ammend, you are avoiding moving HEAD with another commit.

Branching

```
# Create a new branch:
git checkout -b feature_branch_name

# Push your branch to the remote repository (`-u` for add upstream tracking
reference):
git push -u origin feature_branch_name

# Delete remote branch
git push origin --delete remote_branch_name
```

Tag

```
# Create a named tag
git tag gh_solrPoductSearchApiWithCompatibleProducts

# Push the tag to remote
git push --tags
```

Example: Merge feature into master

```
git checkout gh_removeLegacyUserManagement
git branch gh_removeLegacyUserManagement_bak  # create a backup reference
git log # determine number of commits from start of branch

# Put all the changes in the branch into one commit
# Go back to first commit of branch (in this case 2, or specify commit explicitly)
# Interactive editor will show "picks": change "pick" to "squash" for all subsequent
commits
git rebase -i HEAD~2

# Put the single commit from the branch into master
git checkout master
git fetch
git rebase
git cherry-pick e3b8fee61c08eab6a8996ece167e06b901d55d52
git diff HEAD~1
git push

# Delete local branch:

git branch -d gh_removeLegacyUserManagement_bak
error: The branch 'gh_removeLegacyUserManagement_bak' is not fully merged.
If you are sure you want to delete it, run 'git branch -D
gh_removeLegacyUserManagement_bak'.

$ git branch -D gh_removeLegacyUserManagement_bak
Deleted branch gh_removeLegacyUserManagement_bak (was 22da15c).

# Delete local and remote branch:
git push origin --delete gh_removeLegacyUserManagement
To https://git.sonova.com/tloeg/sonovaDigitalCommerce
 - [deleted]         gh_removeLegacyUserManagement

git branch -d gh_removeLegacyUserManagement
```

Notes

In GitHub, suffix a url with `#L18-L20` to highlight lines 18 - 20. Ex:

https://github.com/gheinze/asset-management/blob/master/a4-asset-manager/src/main/java/com/accounted4/assetmanager/Application.java#L9-L10

Trust self-signed certificate

```
GIT_SSL_NO_VERIFY=true git clone https://git.example.com/scm/repository.git
```

Meaning of: `--`

The special "option" `--` means "treat every argument after this point as a file name, no matter what it looks like." This is not Git-specific, it's a general Unix command line convention. Normally you use it to clarify that an argument is a file name rather than an option, e.g.

```
rm -f      # does nothing
rm -- -f   # deletes a file named "-f"
```

https://stackoverflow.com/questions/6561142/difference-between-git-checkout-filename-and-git-checkout-filename

`HEAD`

`HEAD` is the commit at the top of the branch.

`HEAD~1` is the commit 1 previous to `HEAD`

Useful links

- Interactive Git Visualization
- Git Workflows
- Visual Git Guide

Docker

Commands

Command	Description
`docker pull postgres:latest`	retrieve the image "postgres" tagged with "latest"
`docker container ls -a`	lists all containers (default ls just lists running containers)
`docker ps -a`	lists all running containers
`docker image ls`	list all images
`docker logs sql1`	checks the logs of the container "sql1"
`docker exec -it sql1 "bash"`	connect to container "sql1" interactively, starting bash
`docker stop sql1`	shutdown the container "sql1"
`docker rm sql1`	
`docker run --name ubuntu_bash --rm -i -t ubuntu bash`	create a new container in interactive mode, based on image "ubuntu", launching bash, and removing container on exit (-t is for "tty")

Notes on SQL Server Docker Container

Details: https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-linux-2017

Command line instructions are linux-based. For Windows, run PowerShell in Administrator mode and skip the "sudo".

- install docker v1.8+
- retrieve docker image for MS SQL Server

```
sudo docker pull microsoft/mssql-server-linux:2017-latest
```

- create docker container based on image

```
sudo docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=myPassword' -p 1433:1433 --name
sql1 -d microsoft/mssql-server-linux:2017-latest
```

Note:

- creates a docker container "sql1"
- defines the SQL SA password
- makes the internal container port 1433 available on port 1433 of the machine running docker port (-p <localPort>:<containerPort>), so if you already have SQL Server running locally on 1433, you may wish to change the localPort to 1434

Vopy a DB dump into the container volume to make it available for the DB restore function in the Management Studio application

docker cp jcProdDB_67_18June.bak sql1:/var/backups

Notes on a Postgres Docker Container

```
docker pull postgres:latest
docker run --name postgres_asset_manager -p 5432:5432 -e
POSTGRES_PASSWORD=mysecretpassword -d postgres
docker exec -it my-postgres bash
root@92fb691130e8:/# psql -U postgres
```

Java

Find Java Class in Jar file

Unix:

```
find ./ -name '*.jar' -exec grep -Hls MyClassName {} \;
```

Windows:

```
for /R %G in (*.jar) do @jar -tvf "%G" | find "ClassName" > NUL && echo %G
```

Add JNDI Datasource to Tomcat

Rather than configuring the DataSource properties within the application, it may be preferable to have the application request the DataSource from the container. Then administrators can modify configuration without poking into the application.

The following example is from the Tomcat JNDI Datasource example.

1. Copy the JDBC driver into `$CATALINA_HOME/lib`
2. Configure the database connection properties as a resource:

   $CATALINA_HOME/conf/server.xml

   ```
   <GlobalNamingResources>

       <Resource name="jdbc/accounted4" auth="Container"
                 type="javax.sql.DataSource"
                 driverClassName="org.postgresql.Driver"
                 url="jdbc:postgresql://127.0.0.1:5433/postgres"
                 username="tia"
                 password="tia"
                 maxActive="20"
                 maxIdle="10"
                 maxWait="-1"
                 removeAbandoned="true"
                 removeAbandonedTimeout="60"
                 logAbandoned="true"
                 />

   </GlobalNamingResources>
   ```

3. Expose JNDI resource in the Tomcat Application Context

$CATALINA_HOME/conf/context.xml

```
<Context>
    <ResourceLink
        global="jdbc/accounted4"
        name="jdbc/accounted4"
        type="org.postgresql.Driver"
    />
</Context>
```

4. Obtain the JNDI reference from the Spring context

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:jee="http://www.springframework.org/schema/jee"

       xsi:schemaLocation=
            "http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.0.xsd
            ">

    <!-- Use JNDI to find the database connection -->
    <jee:jndi-lookup id="dataSource"
        jndi-name="/jdbc/accounted4"
        resource-ref="true"
    />


</beans>
```

Finance

Negative Amortization

Copied from: http://www.vertex42.com/ExcelArticles/negative-amortization.html

Definition of Negative Amortization

Negative Amortization is the increase in Principal through the addition of unpaid interest.

Most definitions describe this as occurring when a payment is insufficient to cover the interest due, resulting in the interest being added to the loan balance.

The result of negative amortization is that you end up paying interest on your unpaid interest.

Example Amortization Table

The amortization table below illustrates how missing a payment results in interest added to the Principal (see the numbers highlighted red). In this example, the second monthly payment of a 30 year loan is skipped.

Initial Principal	100,000
Monthly Rate	0.50%
Term (in years)	30
Monthly Payment	599.55

No.	Payment	Interest Due	Balance
			100,000.00
1	599.55	500.00	99,900.45
2	0.00	499.50	100,399.95
3	599.55	502.00	100,302.40
4	599.55	501.51	100,204.36
5	599.55	501.02	100,105.83
6	599.55	500.53	100,006.81
7	599.55	500.03	99,907.30
Total:	3,597.30	**3,504.60**	

Negative Amortization Example

The interest due is calculated as the Monthly Rate * Previous Balance, so on Payment No. 3, you end paying even more interest than you did on your first payment. In the table, I summed the total payment and interest over the first 7 payments and highlighted the balance at the end of the 7th month in blue. In this example, it cost about $3000 to make up for the one missed payment (to bring the balance back to the amount after the first payment). This doesn't take into late fees or penalties. Simple Interest Amortization

When handling missed payments, the alternative to negative amortization is "simple interest amortization", where the unpaid interest is NOT added to the Principal balance, but instead is accrued in a separate account to be paid off first before the Principal. The table below shows how simple interest amortization compares to the negative amortization example when the second payment is missed.

No.	Payment	Interest Accrued	Interest Accrual Balance	Principal Balance	Amount Owed
				100,000.00	100,000.00
1	599.55	500.00	0.00	99,900.45	99,900.45
2	0.00	499.50	499.50	99,900.45	100,399.95
3	599.55	499.50	399.45	99,900.45	100,299.90
4	599.55	499.50	299.41	99,900.45	100,199.86
5	599.55	499.50	199.36	99,900.45	100,099.81
6	599.55	499.50	99.31	99,900.45	99,999.76
7	599.55	499.50	0.00	99,899.71	99,899.71
Total:	3,597.30	**3,497.01**			

Simple Interest Amortization with Missed Payment

The thing to notice is that when the second payment is missed, the Principal balance stays the same, and therefore the interest due on payment No 3 is still only 499.50. The total amount owed still goes

up, because you are going to have to pay off the interest that has accrued, but at least you aren't paying interest on interest.

Negative Amortization & Compound Interest

If "simple interest" is defined as paying interest on the principal only, then negative amortization is basically "compound interest" because it results in interest being paid on interest. I bring this up because if you are using my amortization schedule spreadsheet or other loan calculators and mortgage calculators that let you choose a compound period that is different than the payment period, you need to realize an important fact:

Negative Amortization can arise when calculating the effective rate per payment period if the compound period is shorter than the payment period.

When the compound period is different than the payment period, an amortization calculator often uses an effective interest rate, derived from the compound interest formula. For a description of this formula, see "Calculating the Rate Per Period". Such is the case with Canadian mortgages where the compound period is semi-annual and payments are usually made monthly. That's fine. Where you'd run into negative amortization is if you selected a Weekly Compound Period with a Monthly Payment Period (the weekly compound period being shorter than the monthly payment period).

When does this occur? I don't know. I've had many requests to add a daily compound period to my loan amortization schedule, so I complied. But, perhaps what people really are wanting is an amortization table that accrues interest daily as in a so-called "simple interest mortgage"?