Git Rebase Interactive Demo

Created by Guy Van den Nieuwenhof
https://github.com/guyvdn

Initialize Working folder

Initialize git

```
$ git init
```

Create an initial commit

```
$ echo initial commit > file
```

Stage and Commit the file

```
$ git add -A
$ git commit -m "initial commit"
```

Add ConnectionString

```
$ echo password > file
```

Stage and Commit the file

```
$ git add -A
$ git commit -m "connectionString"
```

Add a repository

```
$ echo repo > file1
```

Stage and Commit the file

```
$ git add -A
$ git commit -m "repo"
```

Fix the password

```
$ echo appSetting > file
```

Stage and Commit the file

```
$ git add -A
$ git commit -m "appSetting"
```

The password can still be accessed through the history

View content of the connectionstring

```
$ git show <ref connectionString>
```

We do not want this

Start Rebase Interactive

```
$ git rebase -i head~3
```

git-rebase-todo looks like

```
pick 66a12ba connectionString
pick 7a84f40 repo
pick 91c8e6c appSetting

# Rebase a68d1dd..91c8e6c onto a68d1dd (3 commands)
...
```

Change the todo file

We do not want to have the password in the history

```
pick 66a12ba connectionString
squash 91c8e6c appSetting
reword 7a84f40 repository
...
```

Save and close the file

Rebase Interactive

Choose a commit message for the squash

Change the name of repo commit to repository

When rebase is done

View the history

```
$ git history
```

View content of the connectionstring

```
$ git show <ref connectionString>
```

No trace of the password is visible except for the reflog which only exists local and will be garbage collected eventually.