Create a new UVS repository

Goto repository folder and run uvs init

```
$ cd path_to_repo
$ uvs init
```

This will ask you to setup a password for this repository. The password should ideally be at least 8 characters long. Every UVS command to interact with this repository will ask this password. If it is lost, your repository history is lost for good. There is no way to recover any of it. Be sure to keep this password safe.

monitoring repository status commands

```
$ uvs status
```

This will tell you if working directory is clean (there no changes since last commit) or not.
It will also tell you what branch you are on currently.

```
$ uvs log-all
```

This will show a list of all commits in this repository. Including commit messages, author name, email ...

```
$ uvs log
```

This will show a list of commits on the current branch.

```
$ uvs show-refs
```

This will show all the references in this repository, this includes, head, all branches, and tags in the future (support for tags is not implemented yet in this version of UVS).
(this is equivalent to “$ git show-ref --head”, without --head git would only show branches)

```
$ uvs detail-commit -c _put_commit_id_here_
```

This will show all the details of the given commit id, like commit message, author name, email ...

create new commits

```
$ uvs commit -m  “put commit message here”
```

This create a new commit on the current branch, and advance the current branch to point to this commit. A commit is like taking a snapshot of the state of the repository at the time the commit was taken.

check out old commits / other branches.

```
$ uvs checkout -r  _put_commit_reference_here_
```

Given a commit reference, which is either a commit id or a branch name, this command will checkout that commit. This will restore the repository’s working directory to whatever contents it had when that particular commit was made. If a branch was checked out, it will also set current branch to that branch, so that future development is recorded on that branch. If a commit id was checked out, repository will go into detached HEAD state. This is similar to git’s detached head state. Under this state, new commits can be made like normal. At any point during this development a branch name can be given to this detached branch. Using the branch command. (see below)

To re-attach head to this or any other branch checkout that branch.

branch and merge commands (dvcs style)

$ uvs branch -n _put_branch_name_here_
Create a new branch reference and set it to point to whatever HEAD is pointing to right now. (detached or attached)

This command does not checkout the branch, just creates it. To set it to current branch check it out using the checkout command.

$ uvs merge -n _branch_name_to_merge_changes_from
Merge changes from given branch name into current branch. If there are conflicts, this will save the merge result under uvs_temp/merge_result
Resolve the conflicts however you want, Confirm this is the final version you want and go back to repository root and run the merge-finalize command to finalize this merge.

To abort the ongoing merge, just delete the uvs_temp/ folder.
uvs_temp/ folder contains 3 sub folders, common_ancestor branch1 branch2
This can be handy if you want to use kdiff3 to do the merging , just run

$ kdiff3 common_ancestor/ branch1/ branch2/ -o merge_result/
Now do the merge in kdiff3 and same as before run merge-finalize.

$ uvs merge-finalize -m “merge commit msg”
Finalize an ongoing merge with a given merge message.

---

dependencies
To run uvs alpha python2.7 is needed
Additionally python cryptography is also required, this can be installed with pip
$ sudo pip install cryptography

In order merging to work uvs needs the diff3 command to be installed on the system. This is part of gnu diffutils package, on ubuntu this can be installed by:
$ sudo apt-get update
$ sudo apt-get install diffutils

uvs alpha was developed on ubuntu, its not tested on windows or mac altho it may work there as well.