Note, <https://virtualenv.pypa.io/en/legacy/userguide.html>, python’s activate simply sets the env path to come first.

# prompts are root land

> prompts are user land

see also: github.com/thomas-walker-lynch/project-share

## Project directory

### Root Land Work

<project\_env\_dir> is the directory the project environment goes into. For example ‘customer\_gateway\_2’, being the second version of the customer gateway project. This will also the user name made for the project.

<project\_name> is the name of the project. For example, customer\_gateway. This will also be the name of the repo.

These give all project members access to everything even the level above the repo (the project environment). These directions must be modified if only the administrator may modify a project’s enviromnent.

When a file is moved to a directory with acls, the acls will not be applied. In older version of linux the same was true when a file was copied to a directory with acls.

# umask 0000

# touch a

# touch b

# mkdir test\_directory

# setfacl -m o::-,d:o::- test\_directory

# ls -l

-rw-rw-rw-. 1 root root 0 Apr 24 16:39 a

-rw-rw-rw-. 1 root root 0 Apr 24 17:00 b

drwxrwx---+ 1 root root 6 Apr 24 16:39 test\_directory

# touch test\_directory/c

# cp a test\_directory/

# mv b test\_directory/

# ls -l test\_directory/

total 0

-rw-rw----. 1 root root 0 Apr 24 16:39 a

-rw-rw-rw-. 1 root root 0 Apr 24 16:39 b

-rw-rw----. 1 root root 0 Apr 24 16:39 c

There is a bug in linux where the umask for the Gnome launcher is not taken from the user’s environment, and the default has world read permissions. Hence when emacs is run from the launcher all files created with it have world read permissions independent of how the user has set their umask in startup files. See the file in this directory, ‘gnome-launcher-umask’. We need to set the systemd user umask so that users do not make world readable files then mv them to projects.

Create the project:

# adduser --disabled-login --gecos "" <project\_env\_dir>

# cd /home

# chmod +t,g+rwxs <project\_env\_dir>

+t only file owner may delete the file

g+s group will have group ownership of files created in directory with +s

# chmod -R o-rwx <project\_env\_dir>

remove world permissions from all files. Unfortunately world read permission is quite common due to gnome’s launcher having a built in umask.

# setfacl -R -m g:<project\_env\_dir>:rwX <project\_env\_dir>

# setfacl -R -m d:g:<project\_env\_dir>:rwX <project\_user\_name>

# setfacl -R -m m::rwX,d:m::rwX <project\_env\_dir>

all files in the directory tree are to be accessible to members of the group.

# setfacl -R -m o::-,d:o::- <project\_env\_dir>

no world permissions for files in the directory tree

-> replace project’s .bashrc make sure umask is 0007

-> look over other config files adduser put in

-> .profile is needed on our Deb 10, or su - <user> does not run .bashrc

# chmod go-rwx .bash\* .profile

Add a user to the project who will setup the bin scripts and populate the project. Note the section below “Admin adds users to project” for commands for adding a user to a group.

### User Land Work

# su - <project\_env\_dir>

> cd /home/<project\_env\_dir>

This part is a bit of a pain, because it has to be done from a project group participant who has access to the bin\_project\_template repo:

> git clone bin\_project\_template

Back to the user by the name of <project\_env\_dir>. Edit the start script and set the project name. Edit the start script to set the project\_env\_dir and the project variables.

> mv bin\_project\_template bin

> sudo chown <project\_env\_dir> bin

> chmod g-w bin

If this is a python project.

> virtualenv env

This worked last time, but not this time. I tried reinstalling it, etc. but no dice.

customer\_gateway\_2@reasoning-technology-server-1:~$ virtualenv env

Traceback (most recent call last):

File "/usr/local/bin/virtualenv", line 5, in <module>

from virtualenv.\_\_main\_\_ import run\_with\_catch

File "<frozen importlib.\_bootstrap>", line 983, in \_find\_and\_load

File "<frozen importlib.\_bootstrap>", line 967, in \_find\_and\_load\_unlocked

File "<frozen importlib.\_bootstrap>", line 677, in \_load\_unlocked

File "<frozen importlib.\_bootstrap\_external>", line 724, in exec\_module

File "<frozen importlib.\_bootstrap\_external>", line 859, in get\_code

File "<frozen importlib.\_bootstrap\_external>", line 916, in get\_data

PermissionError: [Errno 13] Permission denied: '/usr/local/lib/python3.7/dist-packages/virtualenv.py'

So instead:

# apt-get install python3-venv

# su - customer\_gateway\_2

> python3 -m venv env

> ls

bin env tmp

Add this to the group participants bashrc files:

alias pcd\_<project acronym>="/home/<project\_env\_dir>/bin/start”

If this is a django project, start by ‘activating’ the project. Using the alias will give an error because the project directory does not yet exist, but it will modify the PATH. Otherwise just run activate in the usual way. If this is not done, you will get a permissions error when trying to install uwsgi.

> pcd\_<project acronym>

> ls

bin env

> python -m pip install wheel

> python -m pip install uwsgi

> python -m pip install django

> python -m pip install dj\_database\_url # might need this

> django-admin.py startproject <project\_name>

> cd <project\_name>

> ls

manage.py etc.

Add data to the project:

(for customer\_gateway\_2 I copied the .git file from customer\_gateway, set the branch to master and did a checkout)

> git clone <project\_name>

To copy files use cat or tar, or fix the ownship and perms afterward if need be.

> tar cf - \* | ( cd /target; tar xfp -)

This is what the customer\_gateway looks like after being setup:

/home/customer\_gateway <-- project environment directory

├── bin <----------- RT project scripts

│ ├── pull

│ ├── push

│ └── start

├── customer\_gateway <--- project group participants share directory│ │ ├── PRIVATE.rst

│ ├── accounts

│ ├── customer\_gateway <--- settings directory made by django

...

│ ├── templates

│ └── www-auto

├── env <------ python environment

│ ├── bin

│ ├── lib

│ └── pyvenv.cfg

├── tmp <------ for temporary files

│ └── git\_pull\_timestamp

├── uwsgi.sh <- project environment files

├── uwsgi\_iphttp\_socket.ini |

└── uwsgi\_unixuwsgi.ini |

Note the linux acl system has a bug (personality feature?) where it treats the command

cp file\_src file\_dst

like the command

cp -p file\_src file\_dst

i.e. it preserves permissions of the source at the destination. This will cause users headaches when they have a non-permissive umask and they use cp to put files into a project directory. It also means we should add a maintenance task of checking permissions on files in projects. See:

<https://unix.stackexchange.com/questions/570795/how-to-make-a-directory-that-users-can-copy-files-to-and-share-them>

## Admin adds users to project

Each user’s umask should be set.

> umask 0007 # sets the umask

> umask # displays the umask

Add to .bashrc:

umask 0007

alias pcd\_<project acronym>="/home/<project\_env\_dir>/bin/start”

Add user to the project group:

# usermod -a -G <project\_env\_dir> <user>

Users will have to log back in for this to take effect.

Note also, remove user from a group:

# gpasswd -d <user> <project\_env\_dir>

,and force a user to logout:

# pkill -KILL -u <user>

## User enters a project

The user types the command:

> pcd\_<project acronym>

This starts a new shell with modified environment variables. The user leaves the project by exiting the shell. He or she will be back at the next prompt in the shell where p<project\_acronym> was typed.

For example,

> pcd\_cg # entering the customer\_gateway project

> pwd

/home/customer\_gateway/customer\_gateway

> ^d | exit

> history

….

pcq

>