

Git commands
`git pull`
`git push`
`git checkout -b {{name_of_new_branch}}`

Do "git branch" but ignore all cc_ and rcc_ branches
`dvc-cc git branch`

To take a look at the result branch you should use the gitlab or github webpage, but you can also use:
`git checkout rcc_0001_expname_A1`

Create for each remote result branch a local branch:
`dvc-cc git sync`

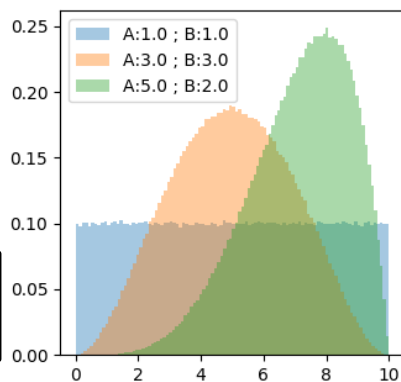
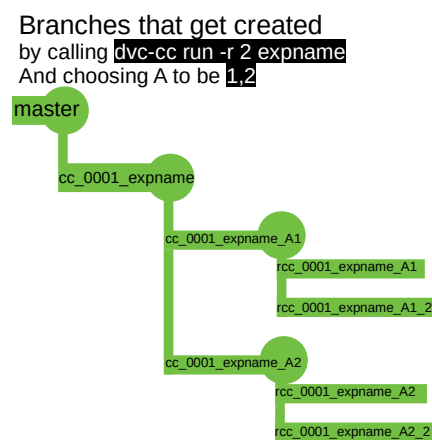
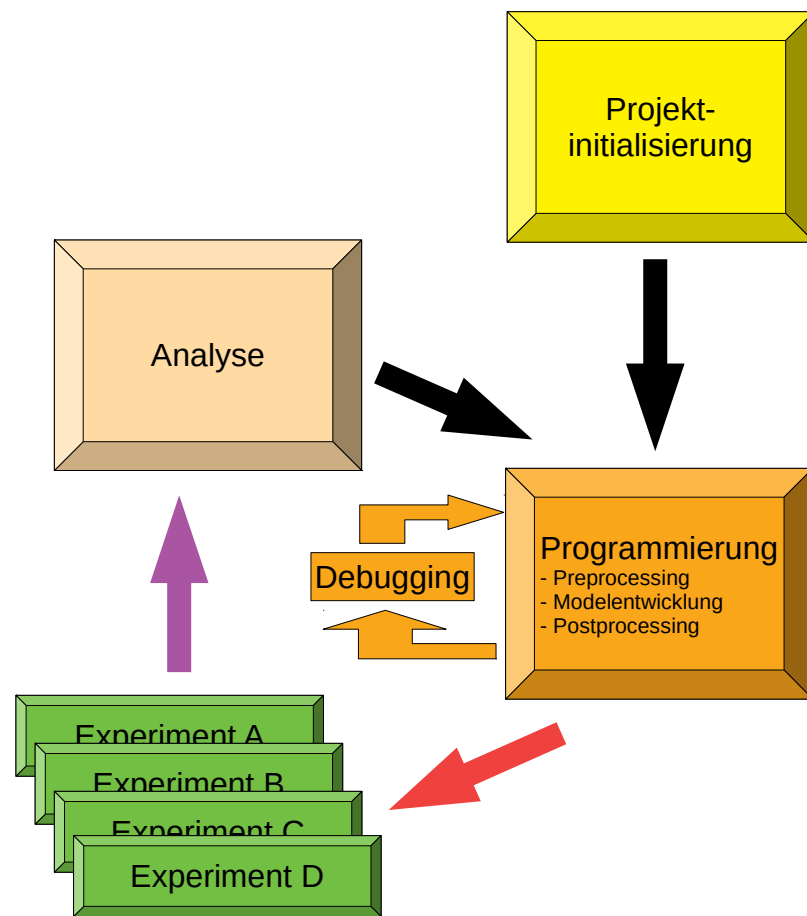
Show all nodes from the cluster
`dvc-cc status --node`

With `dvc-cc status` you can see your last experiments. The following parameters exist:
`-p 23` show all jobs from the 23. run
`-s` summaries the output
`-n 20` show the last 20 experiments.
`-f` show only failed
`-d` show details
`--detail-unchanged` show all details

Run experiments with `dvc-cc run {{experiment name}}`
If you have hyperparameters, you will be asked to set values for the hyperparameters. For this you have multiple options:
- use one value: i.e.: 412
- use multiple values: i.e.: 412, 512, 612
- use GridSearch with `--gs`: i.e.: min Value 5, Max Value 20, Num of draws 4, is the same as writing 5,10,15,20
- use RandomSearch-Global with `--rs`: If you use this for all parameters you will get asked ones, how many draws you want to do. This is the absolute value of experiments that will be created.
- use RandomSearch-Local with `--rsl`: you will get asked every time how many draws you want to do. It will create a GridSearch over all drawn parameters.

Show different beta curves
`dvc-cc hyperopt plot-beta`

Get output files
`dvc-cc output-to-tmp outputfile -p ID`



SSH-Connection
`mkdir data`
`sshfs {{username}}@remote-storage:/path-to-data-folder path-to-project/data`
SSH-Remove-SSHFS-Connection
`fusermount -u path-to-project/data`

Zwischenspeicherung des git-Passwortes für 30 Minuten
`git config --global credential.helper store`
`git config credential.helper cache 1800`

1. Create a Git-Repository
2. `dvc-cc init`
Set all parameters for the cluster or the Hardware that you need.
https://github.com/deep-projects/dvc-cc/blob/master/dvc-cc/tutorial/_settings.md
Hint: If a software is not installed in the docker container, you can write your software dependencies to the "requirements.txt".

Converting from local Jupyter Notebook to remote .py file:
- `dc`s for code that should run only on the server
- `#dch` at the start of a cell for running only local

```
In [ ]: 1 import argparse
        2
        3 parser = argparse.ArgumentParser()
        4 parser.add_argument('-A', '--valueA', type=int, default=None)
        5
        6 """dc
        7 args = parser.parse_args()
        8 """

In [ ]: 1 #dch
        2 args = parser.parse_args('-A 5'.split())
```

For reducing the typing you can use:
`dvc-cc hyperopt new-suggest`

Define a stage of the pipeline with `dvc-cc hyperopt new`:
This will create a ".dvc" file in the "dvc" folder or a ".hyperopt" file in the "dvc/.hyperopt" folder. To delete a stage you can safely remove this file.

name	saved in git	saved in dvc cache	save the checksum	description
-d	False	False	True	You use this to define dependencies (inputs) or everything from what this stage depends on.
-o	False	True	True	Large output files or folders
-O	True	False	True	Small output files or folders
-m	True	True	True	Metrics are output files but have a special feature that you can use with <code>dvc metrics show</code>
-M	True	False	True	Metrics see above. Find more information about metrics here .

i.e.: `dvc-cc hyperopt new -d code/train.py -o tensorboard -m summary.yml -f train.py "python train.py --valueA {{A}}"`

Overview over all parameters:
`dvc-cc hyperopt var all`

Fix a parameter:
`dvc-cc hyperopt var --set 100 A`