DVC Cheatsheet

dvc.org/doc

Getting started

Install DVC

```
pip install dvc[<dependency>]
```
Optionally use a remote dependency:
`s3, azure, gdrive, gs, oss, ssh, all`

Initialize

```
dvc init
```
Use `-f` to overwrite an existing DVC cache

Troubleshooting

```
dvc doctor --v
```
We have a Troubleshooting section in the docs. You can also get help on Discord.

Remotes

Add a remote

```
dvc remote add <name> <url>
```

Modify a remote

```
dvc remote modify <name>
  <option> <value>
```

Push to and pull from remote

```
dvc push
dvc pull
```

Fetch from remote

```
dvc fetch
```
Downloads data from remote like `dvc pull` but doesn't place data in workspace.

iterative

Data versioning

Start tracking files

```
dvc add <file/directory>
git add . & git commit
```

Update tracked files

```
dvc add <file/directory>
dvc push  (if using remote)
git add . & git commit
```

Switch data version

```
git checkout <commit>
dvc checkout
```

Show status tracked files

```
dvc data status
```

Show differences commits

```
dvc diff
```

Remove unused files from cache

```
dvc gc
```

File structure

DVC moves files under its control to the `.dvc/cache`. It then creates `.dvc` files for each directory and file. The files in your workspace are replaced with reflinks to the cache.

Inside the cache, DVC uses its own structure based on file hashes. This lets it avoid file duplication.

External data

Both `import` and `get` download the data. `import` also tracks it with DVC.

Download from DVC project

```
dvc import <url> <path>
dvc get <url> <path>
```

Download from URL (e.g. S3)

```
dvc import-url <url> <out>
dvc get-url <url> <out>
```

Pipelines

Pipelines are defined in `dvc.yaml` and parameters in `params.yaml`

Create a new pipeline

```
dvc stage add <...>
```
Or edit dvc.yaml

Add a stage

```
dvc stage add
  -n <name> -d <dependency>
  -o <output> -p <parameter>
  <command to execute>
```
Or edit dvc.yaml

View pipeline DAG

```
dvc dag
```

Reproduce pipeline

```
dvc repro
```
Use `-f` to run the entire pipeline

Experiments

Run a new experiment

```
dvc exp run
  -S '<param>=<value>'
```
Use `--queue` to add to queue

Run experiment queue

```
dvc queue start
```

Show experiment table

```
dvc exp show
```

Apply experiment to workspace

```
dvc exp apply <exp>
```

Create branch from experiment

```
dvc exp branch <exp> <branch>
```

Push to and pull from remotes

```
dvc exp push <branch> <exp>
dvc exp pull
```

Remove experiment

```
dvc exp remove <exp>
```

Show and compare metrics or plots

```
dvc metrics show
dvc metrics diff <exp1> <exp2>
dvc plots show
dvc plots diff <exp1> <exp2>
```
Use `--open` to open the plots in browser

⭐ Also try the DVC extension for Visual Studio Code for easier experiment management!