Iniciação Científica

Um Tutorial de introdução ao ambiente de Pesquisa

Creado por: Jhonatan A. A. Manco, Liviany P. Viana

jhonatanjaam@gmail.com,liviany.meteoro@gmail.com

Feliz Tutorial

Sumário

Sobre este documento

SEÇÃO 1

LINUX

O Linux é um sistema operacional UNIX. Unix significa que sistema operacional portável (executado independentemente da arquitetura), multitarefas (executa várias tarefas ao mesmo tempo) e multiusuário (permite mais de um usuário ao mesmo tempo).
Alguns sistemas do tipo UNIX são:

- Linux
- MacOs
- Solaris

Já o Linux é um sistema operacional UNIX de Código aberto e gratuito, disponibilizado sob a Licença Pública Geral (GPL) GNU. Qualquer usuário pode executar, estudar, modificar e redistribuir o código-fonte.

1.1 Instalação

As distribuições mais comuns encontradas no ambiente de pesquisa são:

- Ubuntu
- Mint
- Fedora
- Elementary OS

O link leva a pagina de instalação mais recente de cada versão, para propósitos deste tutorial se recomenda a instalação de Ubuntu, umas vez que a distribuição de Linux mais usada e difundida.
o Linux também pode ser usado nas novas versões de Windows(Windows 10 versão 2004 e superior) e é conhecido com o nome de WSL. No seguinte link estão as instruções para a instalação do WSL no ambiente Windows.

Link: WSL

Para propósitos deste tutorial, o ambiente WLS pode ser suficiente para realizar as tarefas. No entanto, para usuários que irão usar supercomputadores, instalar modelos meteorológicos, trabalhar com Fortran, entres outros, seu uso não é recomendado.
Uma vez instalado o sistema operacional Linux, são apresentados os comando mais usados dentro do terminal do sistema operacional Ubuntu. Para abrir o terminal, basta usar Ctrl+Alt+T, ou procurar nas aplicativos do sistema.
O terminal do Linux é um dos principais características do sistema operacional. Serve para receber os comandos do usuário a partir do teclado e repassa-os diretamente ao sistema operacional.
Os comandos mais usados dentro do terminal são :

- pwd: encontra o caminho completo do diretório atual.
- cd: Acessa uma determinada pasta (diretório)
- ou mudar de diretório atual, ex: cd .., cd /nomedapasta
- ls: Lista todos os arquivos do diretório
- mkdir: Cria um diretório
- rm: Remove um arquivo/diretório, seu uso não pode ser desfeito, apaga por completo
- df: Mostra a quantidade de espaço usada no disco rígido
- top: Mostra o uso da memória
- cat: Abre um arquivo
- mv: pode ser usado para mover ou renomear arquivos.
- clear: limpar o terminal se estiver cheio de muitos comandos usados anteriormente, blueCtlr+l
- : caminho do home ex: cd , vai para home
- vi: Abre o editor

Uma vez instalado o Linux e se familiarizado com seus comandos básicos dentro do terminal, na seguinte capítulo será instalado o Python usando o Anaconda.

SEÇÃO 2

Python Anaconda

Anaconda é a plataforma de distribuição de Python mais popular. Esta conta com o comando conda-install, que permite a instalação de miles de pacotes de código aberto tanto para Python, R e outras linguagens de programação.

2.1 Instalação

Para sua instalação basta entrar no seguinte link e procurar o pacote adequado para o sistema operacional correspondente:

Link: Anaconda

2.2 Criação do Ambiente de trabalho com Anaconda

Uma vez instalado, basta configurar o Anaconda com a versão do Python e os pacotes necessário para a tarefa desejada. Para tal objetivo é útil a criação de ambientes dentro do mesmo Anaconda, o que permite ter diferentes configurações de Python para propósitos diferentes (Python2.7, Python3.3, Python3.8). Neste tutorial será instalada, a modo de exemplo, a versão do Python 3.8, criando o ambiente de nome py1. Para isto abra o Terminal, $(Ctrl+t)$, e verifique que versão do Anaconda esta instalada e encontra-se atualizada:

```
Terminal
(base) computername:~$ conda -V
conda 4.11.0
(base) computername:~$  conda update conda
computerCollecting package metadata (current_repodata.json): done
```

```
Solving environment: done
...
```

Uma vez verificada a instalação o comando conda create -n será usado para criar o ambiente py1

```
(base) computername:~$  conda create -n nomeambiente(py1)
python=x.x(3.8) anaconda
```

onde nomeambiente é o nome que terá o ambiente criado, neste caso py1, e python=x.x define qual versão do Python será usada, podendo ser mudada, a modo de exemplo, para 2.7.
Para ativar o novo ambiente basta usar o comando conda activate com o nome do ambiente

```
(base) computername:~$  conda activate nomeambiente(py1)
(py1) computername:~$
```

Note que o nome do ambiente(py1) substitui o nome (base) no terminal.
Para desativar o ambiente basta usar o comando conda deactivate com o nome do ambiente .

```
(base) computername:~$  conda deactivate nomeambiente(py1)
(base) computername:~$
```

2.3 Instalação de Pacotes no Ambiente criado

Por defeito Anaconda e sus respetivos ambientes criados na seção anterior contam com os pacotes mais usados no Python, tais como: numpy,matplotlib, datetime ... entre outros.
Para instalar os pacotes necessários para as diferentes aplicações do Python para um determinado ambiente pode ser usado o comando:

```
(base)computername:~$conda install -n nomeambiente pacoteainstalar
```

O instalando o pacote dentro do ambiente desejado:

```
(py1) computername:~$   conda install -n  pacoteainstalar
```

será
Uma vez criado o ambiente de trabalho faze-se necessário instalar os pacotes necessário para um determinado objetivo. Neste caso são pacotes necessários para trabalhar com dados meteorológicos, o que será visto na seguente seção.

3

SEÇÃO

Pacotes do Python para aplicações Geofísica

3.1 Leitura de arquivos de dados

Existem uma grande variedade de arquivos que podem ser abertos com o Python instalando os pacotes necessários. O primeiro tipo de dado abordado serão os dados do tipo NETCDF(.nc, .cdf). Este é um dos formatos mais usados para guardar dados meteorológicos, tanto de Reanalises como de saídas de modelos. Para a leitura deste tipo de arquivos Python conta com uma biblioteca chamada netcdf-python. Para sua instalação use o comando conda install

```
(py1) computername:~$   conda install -c conda-forge netCDF4
```

3.2 Pacotes para plotar Mapas

Python oferece grande variedade de bibliotecas que facilitam a plotagem de mapas em duas dimensiones. Uma das mas usadas é BASEMAP. Este pacote é similar ao programas Grads e NCL, muito usados nas Ciências Atmosféricas. Ela depende de outra biblioteca que devem ser instalada antes, GEOS (Google Earth Overlay Server), uma vez que usa muitas das funções para criação de mapas desenvolvidas para esta. Sua instalação com o CONDA é simples:

```
(py1) computername:~$   conda install geos
```

Uma vez instalada o GEOS o Basemap pode ser instalado no CONDA assim:

```
(py1) computername:~$   conda install basemap
```

Se o comando anterior não funciona use este:

```
(py1) computername:~$  conda install -c conda-forge basemap
```

Finalmente instale estas ultima atualizações para os pacotes anteriores.

```
(py1) computername:~$ conda install basemap-data-hires
```

```
(py1) computername:~$ conda install -c conda-forge nc-time-axis
```

ou usando o pip

```
(py1) computername:~$     pip install nc-time-axis
```

3.3 Pacote Xarray

A biblioteca Xarray é um pacote que facilita a leitura de dados com dados de N-dimensão (multi-índices), tal como, coordenadas, mapeamento e rótulos, ou seja, ideal para trabalhar com dados de mais de 2 dimensões como os dados meteorológicos provenientes de reanálises (ERA5). Xarray é uma mescla das bibliotecas Pandas e NetCDF, facilitando a leitura e cálculo das variáveis. Para mais informações visite a pagina:

Link: Xarray.

Para sua instalação, use o código abaixo, cuja versão está em v0.11.3.

```
(py1) computername:~$   conda install -c anaconda xarray
```

3.4 Pacote MetPy

A biblioteca MetPy oferece ferramentas em Python para leitura, visualização e cálculos com dados meteorológicos. Podendo ser muito útil em varias áreas como a termodinâmica uma vez que conta com funções para calculo de variáveis como CIN, CAPE, LCL entre outros e gráficos de temperatura, pressão, radio sondagem, etc. Mais informações no seguinte link

Link: MetPy.

```
(py1) computername:~$   conda install -c conda-forge metpy
```

Outras funções pode ser instaladas usando o CONDA mas para os objetivos deste tutorial as bibliotecas necessárias para plotar dados meteorológicos já estão instaladas. Os próximos passos serão usar estar para plotar dados de Reanalises.

4

SEÇÃO

Dados de reanálises ERA-5

Os dados meteorológicos conhecidos por "reanálises", se referem aos dados que descrevem de forma mais precisa o tempo ou clima (lembrando que estes dois termos são diferentes!), ou seja, são dados gerados a partir das observações (uso de assimilação de dados) e previsão de tempo de curto prazo (modelos atmosféricos), utilizando modelos meteorológicos modernos. A partir disso, esses dados ajudam a entender o tempo/clima presente e passado de forma mais clara e robusta.

Um exemplo da banco de dados de reanálises, cita-se as do ERA5, que quer dizer, em inglês *ECMWF Reanalysis, Version 5* oriundo do *Centro Europeu de Previsões Meteorológicas de Médio Prazo* (ECMWF, em inglês). São dados mais robustos e completos com altíssima resolução espacial (31 km) e temporal (cada hora). Mais detalhes podem ser encontrados neste link *https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5*.

Para obter esses dados de reanálises, segue-se os passos-a-passos abaixo:

1. **Passo 1:** Cadastrar-se nesta página no canto superior direito (login/register);

   ```
   https://cds.climate.copernicus.eu/cdsapp#!/home
   ```

2. **Passo 2:** Após a realização do **passo 1**, entre com email e senha cadastrado, e busque os dados de seu interesse em *Datasets*. Por exemplo, os dados horário desde 1979 até os dias atuais;

   ```
   ERA5 hourly data on pressure levels from 1979 to present
   ```

3. **Passo 3:** Clicando nos dados acima, poderás verificar a paǵina com 4 (quatro) abas denominadas:

   ```
   Overwiew  Download data Quality assessment Documentation
   ```

4. **Passo 4:** Clique na aba *Download data* e aparecerá as opções que deves escolher: tipo de produto, variável, nível de pressão, ano, mês, dia, hora, área geográfica e formato (netcdf), como na .

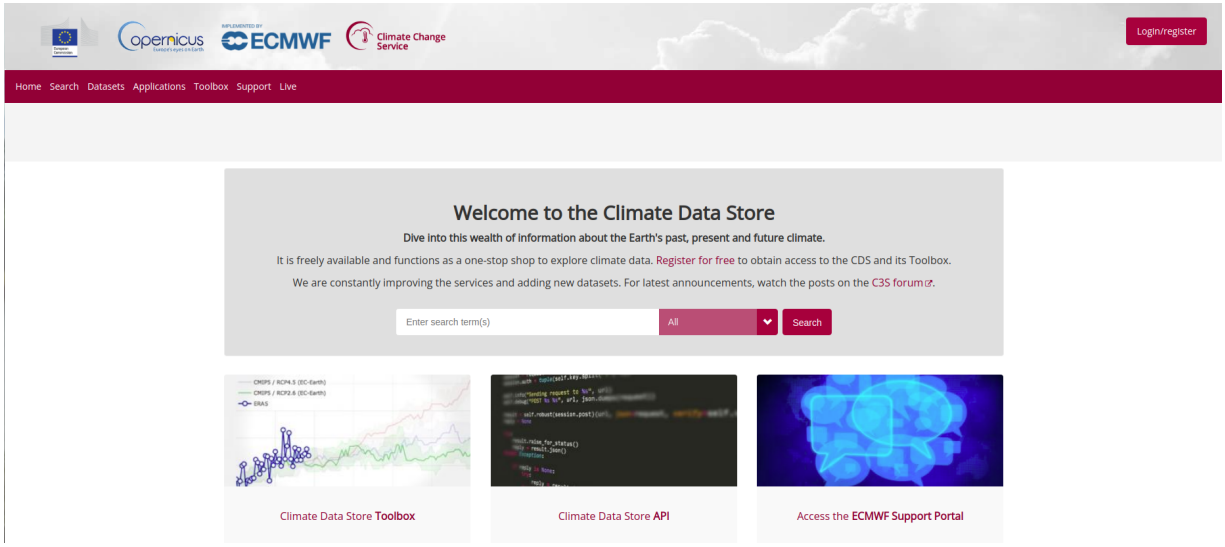

Figura 1: Caption

```
[Product type] [Variable] [Pressure level] [Year]
[Month] [Day] [Time] [Geographic area] [Format]
```

5. **Passo 5:** Após escolher as opções desejadas, clique em:

```
Submit form
```

SEÇÃO

Git

Uma ferramenta que permite ter controle de versão dos códigos desenvolvidos no Python e dos textos no LATEX é o Git. O controle de versão serve para alterar qualquer tipo de arquivo sempre mantendo um histórico das modificações realizadas, o que permite facilmente voltar a uma versão anterior dos códigos ou dos textos. O Git foi desenvolvido para proporcionar uma plataforma de trabalho para desenvolvimento de códigos por vários usuários, permitindo de um controle das alterações feitas por cada membro do grupo. Além disto o Git oferece uma ampla portabilidade uma que seu acesso é virtual (servidor em nuve do git) e também permite ser instalado em qualquer computador. Isto proporciona ao usuário a facilidade de alterar seus código com grande facilidade em diferentes maquinas de trabalho, além de permitir ter um histórico completo das modificações

Para trabalhar no Git basta criar uma conta numa numa das suas tantas plataformas de distribuição. Entre estas plataformas encontra-se o GitHub, BitBucked, entre outras. Estas plataforma irão fazer o controle de versão e guardar na sua nuvem os códigos inseridos. Neste tutorial recomendamos o uso do BitBucked uma vez que os códigos que serão desenvolvidos são para uso privado. No entanto se o objetivo desejado é compartilhar com uma determinada comunidade os desenvolvimentos recomenda-se o uso do GitHub. Para criar uma conta no Bitbucket entre no seguinte link e siga as instruções,

Link: BitBucket

Crie seu usuário usando um nome de usuário e senha, sem usar um login automático pela conta do Google, já que serão de muita importância para o gerenciamento do repositório.

Uma vez criada a conta, basta abrir um novo repositório, que será uma pasta virtual onde os códigos serão inseridos e controle de versão será feito pelo BitBucket. Para criar um novo repositório clique em qualquer dos círculos vermelhos (signo '+'), veja a figura:

Adicione as informações necessárias

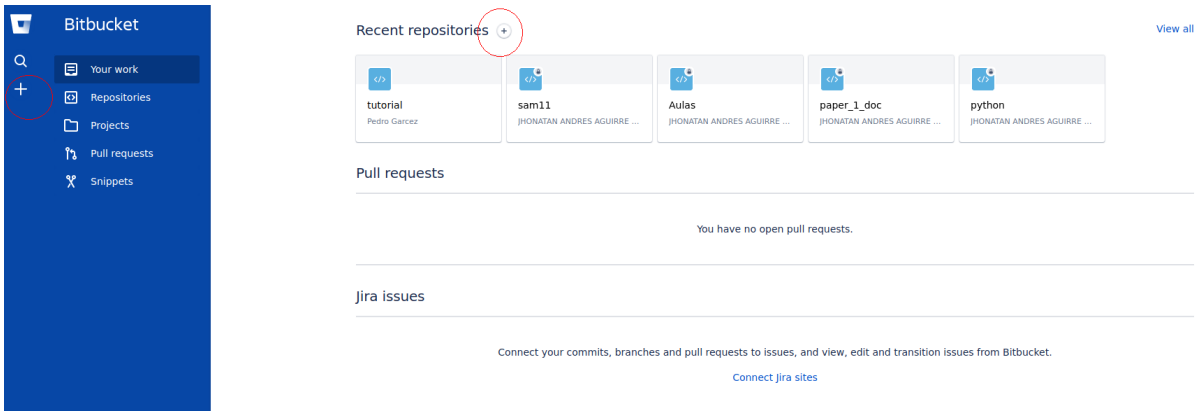

Figura 2: Criação de um repositório no Bitbucket

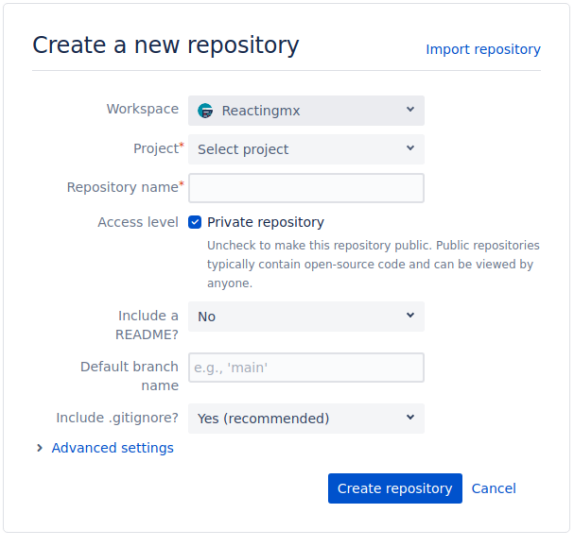

Figura 3: Arquivos criados por definição na criação do repositório "exemplo"no Bitbucket

Uma vez criado (recomenda-se o uso do REAME), Fig 4 o repositório está disponível para ser usado salvando os arquivos que precisem de um controle de versão, como relatórios em LATEX, códigos de Python, Fortran, entre outros. Na figura podem se ver dois arquivos que foram criados por definição, o REAME e o .gitignore. O REAME é um arquivo usado para descrever a finalidade do repositório, ajudar o usuário para uso dos códigos ou fazer comentários pertinentes. O .gitignore tem como finalidade excluir arquivos que não precisam do controle de versão seja porque são resultados dos códigos, são dados muito pesados que não podem ser compartilhados na rede (existe um limite máximo dependendo da versão do git), esse arquivos devem ir listados por filas exemplo: excluir o arquivo dados.nc.

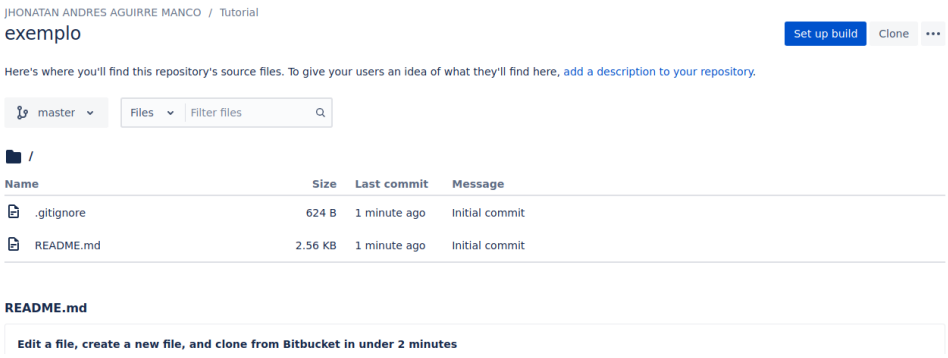

Figura 4: Criação de um repositório no Bitbucket

Além deste arquivos o Bitbucket mostra um pequeno tutorial de como criar um REAME e como inserir arquivos no repositório, o que será abordado seguidamente.

5.1 Clonando o Repositório Criado

No canto superior direito da Fig.4 encontra-se um botão para clonar o repositório, este botão proporciona um link direto para o repositório (https ou ssh). O link criado é único para o repositório e permite criar uma pasta contendo todos os arquivos do repositório em diferentes maquinas ou pastas. Para isto basta executar o link criado no terminal, na maquina e localização onde o repositório será criado:

```
(py1) computername:~local$
git clone https://usuario@bitbucket.org/usuario/exemplo.git
```

Uma vez clonado basta verificar na pasta o conteúdo da pasta, que vai ser igual ao criado no BitBucket, neste caso terá o REAME e .gitignore.

5.2 Adicionando Arquivos no GIT

Para adicionar arquivos no repositório basta entrar na pasta clonada e como primeiro passo verificar que a versão clonada contenha as últimas atualizações feitas.

Este passo deve ser executado sempre que inicie-se um novo trabalho no repositório para evitar conflitos de versão entre as diferentes pastas clonadas em diferentes computadores. Não trabalhe em diferentes computadores ao mesmo tempo, pode levar a erros de versão no Git.

Para atualizar o GIT:

```
(py1) computername:~local$
git pull --all
```

Uma vez atualizado o Git, no primeiro ingresso, serão configurados o e-mail e o usuário para cada maquina na qual o repositório foi clonado. Isto permite que seja identificada em que maquina e quem realizou as alterações. Para isto basta:

```
(py1) computername:~local$
git config --global user.email "you@example.com"
(py1) computername:~local$  git config --global user.name "Your Name"
```

Uma vez configurado e atualizado o repositório, este esta pronto para guardar os códigos e as alterações que serão feitas. Para adicionar um arquivo no repositório use:

```
(py1) computername:~local$ g
it add filetoadd
```

Para verificar o processo o Git conta com o comando:

```
(py1) computername:~local$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   filetoadd
```

Onde é mostrado, no terminal, a etapa do processo de adição do arquivo, neste caso, 'Changes to be commited' O seguinte passo é consignar (commit) o arquivo com um comentário pertinente que permita identificar as alterações feitas. Para isto use o comando:

```
(py1) computername:~local$ git commit -m 'comentario'
```

Como foi feita uma adição de arquivo um comentário pertinente seria 'adicionando arquivo filetoadd'. Quando é modificado um arquivo este comentário permite identificar as altercações do código sem ter que entrar para ver o que foi feito.
Uma vez consignado o estado do repositório pode ser consultado:

```
(py1) computername:~local$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

O qual esta indicando que nosso repositório está a frente do nosso repositório virtual (o clonado, origin/master), uma vez que este conta com um arquivo novo. Assim, esta mudança dever ser reportada (push) ao repositório principal (origin/master), para o que é necessária usar o seguinte comando:

```
(py1) computername:~local$ git push origin
```

Se é consultado o estado do repositório, obtém-se :

```
(py1) computername:~local$
no changes added to commit (use "git add" and/or "git commit -a")
```

que especifica que repositório está atualizado e que novas alterações podem ser feita e consignadas no repositório.
Se é consultado o repositório virtual diretamente no BitBucked a alteração feita ficou disponível, neste caso o novo arquivo adicionado deve ser encontrado.

Cabe destacar, que uma vez o repositório virtual seja atualizado, todas as demais versões do repositório devem ser atualizadas para conter as novas mudanças. Se isto não for feito, erros de verão serão criados e terão que ser solucionados usando o comando git stash, git checkout, git rm (consulte o git help: Bitbucket help)

5.3 Outros comandos uteis no Git

Para não estar repetindo a senha do repositório cada vez que seja atualizado o repositório basta pedir para o Git guardar a senha usando os seguintes comandos:

```
(py1) computername:~local$
git config --global credential.helper store
git pull
```

6

SEÇÃO

Gráficos no Python, Mapas

Uma vez instalado o Python e todas as bibliotecas recomendadas, nesta seção serão mostrando como usar elas para plotar Mapas a partir de dados meteorológicos.
Como ponto de partida foi usado o tutorial netcdf_python [**xx**], onde os comandos básicos para fazer um mapa no Python são apresentados usando os dados de Reanalises do NCEP/NCAR [**ReanalysisProject**].
Para começar abra um editor de texto, o da sua preferencia, Ubuntu conta com o gedit. Uma vez aberto o editor inicie um novo arquivo o qual tera um nome do código e a extensão do Python (ex.main.step1.py) e cole as seguintes linhas que são comando que o Python ira executar.

- Para importar as bibliotecas necessárias que serão usadas no programa