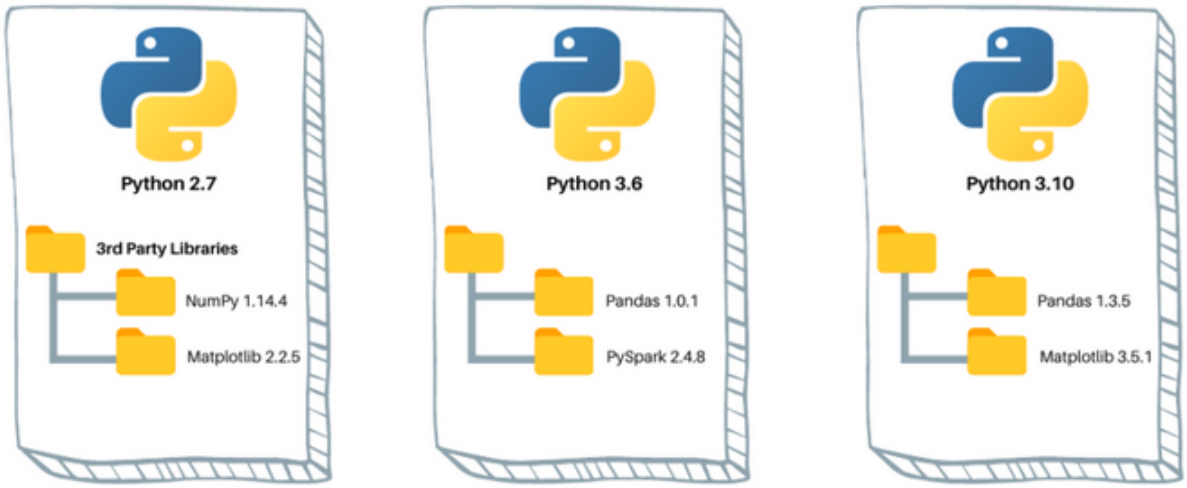
Python

- Conda
  - Conda virtualna okolina
    - Stvaranje virtualne okoline
    - Conda kanali
  - Upravljanje paketima
  - Conda virtualne okoline na klasteru
- Mamba
- Paralelno računanje - mpi4py
  - Instalacija mpi4py paketa unutar virtualne okoline
  - Primjer paralelnog računanja

Na klasteru Isabella dostupne su verzije python-a 2.7.5 i 3.6.8. Ukoliko je potrebna druga verzija pythona, preporuča se korištenje python virtualnih okolina.

Virtualna okolina je izolirana, funkcionalna verzija python-a koja održava svoje datoteke, direktorije i putanje sa specifičnim verzijama knjižnica.



Iako postoje mnogi alati za upravljanje virtualnim okolinama (conda, venv, virtualenv, pyenv, pyenv-virtualenv...) preporučamo korištenje conde/mambe za stvaranje i upravljanje virtualnim okolinama.

Na klasteru su dostupne miniforge3 i mambaforge conda distribucije:

Distribucija	Aktivacija	Package Manager	Verzija
Miniforge	source /apps/miniforge3/bin/activate	conda	4.13.0
		pip	22.1.2
Mambaforge	source /apps/mambaforge/mamba/bin/activate	mamba	0.27.0
		conda	22.9.0
		pip	22.1.2

Conda

Conda je alat otvorenog koda koji služi za upravljanje paketima kao i sustav za stvaranje i upravljanje virtualnim okolinama. Conda je stvorena za *Python* programe, ali može pakirati i distribuirati software za bilo koji jezik (*C libraries, R paketi, Java paketi...*).

Conda u funkciji *package manager-a* pomaže korisniku u traženju i instaliranju različitih paketa. Ukoliko postoji paket/program koji zahtjeva drugačiju verziju Python-a od trenutno instalirane verzije na operativnom sistemu, Conda omogućava kreiranje odvojene okoline sa drugačijom verzijom *Python-a*.

Conda virtualna okolina

Iako *Python* omogućava stvaranje i upravljanje virtualnim okolinama, Conda virtualna okolina ima neke prednosti. Glavni i preporučeni alat za upravljanje paketima u *Python* virtualnoj okolini je *pip*. Uporaba pip sustava je ograničena samo na *Python* pakete. Stvaranje virtualne okoline u Conda sustavu omogućava instalaciju svih paketa pa tako i korištenje *pip* alata kao alternativa Conda *package manager-u*.

Stvaranje virtualne okoline

Prije kreiranja virtualne okoline, potrebno je učitati **conda-miniforge3** naredbom:

```
$ source /apps/miniforge3/bin/activate
```

Virtualna okolina stvara se naredbom **conda create**. Prilikom pozivanja naredbe potrebno je dodijeliti ime novog virtualne okoline zastavicom **"-n"** ili **"-p"** (ins talacija u određeni direktorij) te verziju *Python-a* koja će se instalirati **"python=x.y"**.

```
$ conda create -n <env_ime> python=<npr. verzija 3.9>
```

Instalacija u zadani direktorij

```
$ conda create -p </home/korisnik/env_ime> python=<npr. verzija 3.9>
```

Ukoliko korisnik želi prilikom stvaranja nove virtualne okoline instalirati i određene pakete, moguće ih je navesti prilikom izvođenja naredbe create. Instalacija paketa moguća je i nakon stvaranja virtualne okoline.

```
$ conda create -n <env_ime> python=3.8 <paket_1> <paket_2> <paket_3> ...
```

Pregled kreiranih virtualnih okolina moguće je vidjeti naredbom:

```
$ conda env list
```

Aktivacija virtualne okoline izvodi se naredbama:

```
$ conda activate <env_ime>
```

Command prompt pokazati će naziv aktivnog virtualne okoline

```
(<env_ime>) [<user>@teran ~]$
```

Deaktivacija virtualne okoline:

```
$ conda deactivate
```

Uklanjanje okoline:

```
$ conda env remove -n <env_ime>
```

Conda kanali

Conda paketi dohvaćaju se sa različitih kanala - URL-a/adresa direktorija. Učitavanjem miniforge3 okoline, učitan je i kanal za pretraživanje paketa **conda-forge.** Conda-forge/miniforge je kreacija GitHub zajednice. Svi paketi instalirani preko conda forge kanala su *Open Source* tipa. Korisnik je u mogućnosti dodati još kanala za pretraživanje paketa kao npr. bioconda - kanal specijaliziran za *Open Source* software iz područje bioinformatike.

Glavni/default kanal Anaconda Inc.-a ne nalazi se na početnom popisu zbog aktualnih uvjeta korištenja tj. zbog mogućnosti kršenja istih. Anaconda je 2020 godine promjenila uvjete korištenja. U trenutku pisanja ovog teksta, ukoliko se Anaconda distribucija koristi u komercijalne svrhe, licenca za korištenje se plaća.

Naredbe za pregled, dodavanje i brisanje kanala:

Popis učitanih kanala

```
$ conda config --show channels
```

Popis učitanih kanala sa prioritetima pretrage

```
$ conda config --get channels
```

Dodavanje kanala - sa najvećim prioritetom

```
$ conda config --add channels <ime_kanala>
```

Dodavanje kanala sa najnižim prioritetom ili promjena prioriteta postojećeg kanala na najniži

```
$ conda config --append channels <ime_kanala>
```

Uklanjanje kanala

```
$ conda config --remove channels <ime_kanala>
```

Upravljanje paketima

Preporučeni način instalacije paketa/aplikacija u virtualnoj okolini je korištenjem conda upraviteljem tj. **conda install** naredbom.

```
$ conda install <paket_ime_1> <paket_ime_2>
```

Moguće je navesti više od jednog paketa koje će Conda instalirati, a naredbu je moguće detaljno modificirati nekom od dostupnih zastavica.

Ime virtualne okoline u kojoj će se instalirati paket definira se zastavicom **-n** ili **--name.** Ako nije navedeno ime okolinea, paket će se instalirati u trenutno aktivnoj virtualnoj okolini.

```
$ conda install <paket_ime_1> <paket_ime_2> -n <env_ime>
```

Conda install naredbom pokušati će se instalirati posljednja verzija navedenog paketa, a da bi se paket uspješno instalirao, moguće je da će se instalirati i dodatni paketi ili ažurirati već instalirani paketi. Ako se ažuriranje ostalih paketa želi spriječiti, potrebno je dodati zastavicu :

> *--freeze-installed*

Ako se želi instalirati određena ili starija verzija aplikacije/paketa potrebno je navesti verziju:

```
$ conda install <paket_ime>=<verzija> --> npr. matplotlib=1.4.3
```

Dostupne verzije i pakete moguće je pretraživati sa **conda search** naredbom. Search naredbom pretražiti će se učitani kanali i izlistati dostupne verzije paketa.

```
$ conda search <paket_ime>
```

Dodatne naredbe za pregled, instalaciju, uklanjanje i ažuriranje paketa:

Naredba pokazuje sve instalirane pakete i njihove detalje u aktivnom virtualnom okruženju

```
$ conda list
```

Pretraga određenog paketa

```
$ conda list | grep -i <ime_paketa>
```

Brisanje/deinstalacija paketa.

```
$ conda uninstall <ime_paketa>
```

Ukoliko nije moguće instalirati paket/aplikaciju sa "conda install" naredbom moguće je služiti se "pip install" naredbom.

```
$ pip install <paket_ime>
```

Conda update naredba ažurira pakete na zadnju kompatibilni verziju.

Ažuriranje navedenog paketa.

```
$ conda update <paket_ime>
```

Ažuriranje cijelog aktivnog virtualnog okruženja (bez ažuriranja Python-a!).

```
$ conda update --all
```

Ažuriranje Python-a

```
$ conda update python
```

Conda virtualne okoline na klasteru

Na klasteru postoje unaprijed pripremljene conda virtualne okoline. Pripremljene conda virtualne okoline učitavaju se sa naredbom **module load <virtualna-okolina>.** Ukoliko unaprijed pripremljena virtualna okolina ne sadrži sve potrebne pakete, moguće ju je lokalno nadograditi sa naredbama:

```
$module load <virtualna okolina>
$conda create --prefix $LOCALPKGS python=<Verzija pythona virtualne okoline>
$conda install --prefix $LOCALPKGS <potrebni-python-pkg>
ili
$pip install --prefix $LOCALPKGS <potrebni-python-pkg>
```

Mamba

Mamba je reimplementacija conde u C++-u i kompatibilna je sa conda-om. Mamba je puno brža u slučaju rješavanja *dependecy-a/ovisnosti* paketa koje doprema. Virtualne okoline stvorene mambom mogu se aktivirati i nadograđivati conda-om i obrnuto. Korištenje je jednako kao i za condu sa razlikom u conda/mamba riječi prilikom pisanja naredbe.

Primjer stvaranja okoline sa mambom

```
$ mamba create -n <env_ime> python=<npr. verzija 3.9>
```

Paralelno računanje - mpi4py

Mpi4py paket omogućava paralelno izvođenje programa na više od jednog čvora. U vrijeme pisanja ovog teksta, mpi4py paket za **python verziju 3.10** nije dostupan na conda-forge kanalu, ali ga je moguće instalirati pomoću pip paket upravitelja.

Instalacija mpi4py paketa unutar virtualne okoline

Conda Install

```
$ conda install mpi4py
```

U slučaju instalacije pip upraviteljem preporučeno je da se učita/definira neki od postojećih MPI modula dostupnih na *cluster-u* (npr. mpi/openmpi41-x86_64).

Pip install

```
$ module load mpi/openmpi41-x86_64
$ export MPICC=$(which mpicc)
$ pip install mpi4py
```

Primjer paralelnog računanja

Primjer se sastoji od dvije skripte. Skriptom *testrun.sge* definirati će se virtualna okolina, potrebni moduli (ukoliko su potrebni), broj jezgri za paralelizaciju kao i skripta koja će se paralelizirati (*test.py*). *S*a naredbom mpirun pokrenuti će se isti broj instanci skripte *test.py* kao i broj zadanih jezgri za proračun (4). Za ispravno pokretanje ove skripte **potrebna je instalacija numpy paketa i mpi4py** u virtualnoj okolini.

Kada se program pokreće sa MPI-om, svi procesi su grupirani u komunikator (MPI.COMM_WORLD). Varijable komunikatora ne mjenjaju se nakon što je komunikator kreiran. Veličina komunikatora tj. zbroj svih procesa nosi naziv **size**. Svaki od procesa unutar komunikatora ima jedinstveni broj za identifikaciju - **rank**.

Kroz primjer paralelnog računanja stvoriti će se niz od 100 000 000 nasumičnih brojeva. Stvoreni niz podijeliti će se na 4 parcijalna niza koji će zatim biti podijeljeni svakom od stvorenih procesa. Svaki od procesa osrednjiti će parcijalni niz i vratiti vrijednost glavnom procesu za finalno osrednjavanje. Rezultat finalnog osrednjavanja trebao bi biti blizak broju 10.

Kreiranje okruženja i instalacija potrebnih paketa

```
$ source /apps/miniforge3/bin/activate
$ conda create -n test_env python=3.9
$ conda activate test_env
$ conda install mpi4py numpy
$ conda deactivate
```

testrun.sge

```
#! /bin/bash
#$ -cwd
#$ -N testmpijob
#$ -pe *mpi 4

source /apps/miniforge3/bin/activate

module load mpi/openmpi41-x86_64

conda activate test_env
mpirun -np $NSLOTS python ./test.py
conda deactivate
#module purge
```

test.py

```
from mpi4py import MPI
import numpy as np
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# generira niz velikog proja podataka na RANK_0
numData = 100000000
data = None
if rank == 0:
    data = np.random.normal(loc=10, scale=5, size=numData)

# inicijalizira prazan niz za prihvatiti razdjeljene podatke
partial = np.empty(int(numData/size), dtype='d')

# šalje podatke ostalim "radnicima"
comm.Scatter(data, partial, root=0)

# inicijalizira prazan niz za prihvatiti osrednjene parcijalne nizove
reduced = None
if rank == 0:
    reduced = np.empty(size, dtype='d')

av_loc=np.average(partial)
# osrednjuje parcijalne nizove i zatim ih skuplja u RANK_0
comm.Gather(av_loc, reduced, root=0)

if rank == 0:
    print('Full Average:',np.average(reduced))
```

Podnošenje posla

```
$ qsub testrun.sge
```