Apptainer

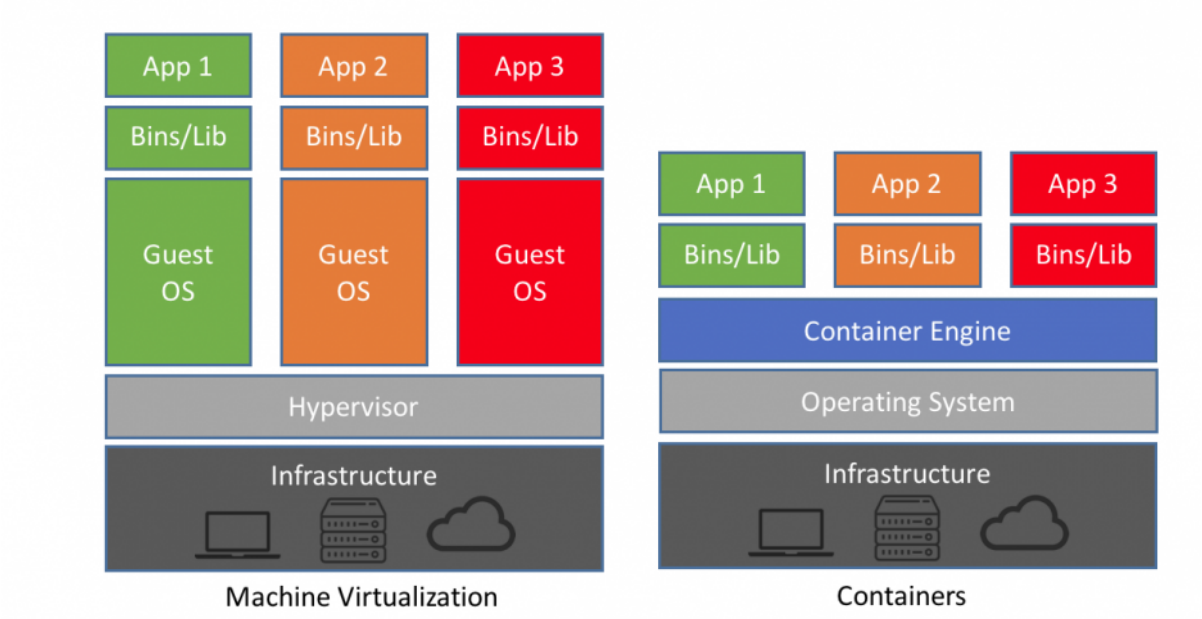
Sadržaj

- Uvod
  - Apptainer
- Apptainer i kontejneri
  - Izgradnja kontejnera - build
  - Nadogradnja kontejnera (lokalno) - shell --writable
  - Izgradnja receptom - .def
  - Korištenje javnih repozitorija
  - Izgradnja na Supeku
- Korištenje kontejnera
  - Podnošenje kontejnera na Supeku
  - MPI aplikacije u kontejneru
  - Grafički procesori i CUDA
  - NVIDIA NGC kontejneri

Uvod

Kontejner je standardna jedinica softvera koja pakira kod i sve njegove ovisnosti tako da se aplikacija može brzo i pouzdano pokretati u različitim okruženjima. Oni "sjede" na fizičkom serveru i njegovom operacijskom sustavu i s njim dijele kernel. Slični su virtualnim strojevima (eng. *virtual machines*) utoliko što su također virtualizirani sustavi.

Međutim, za razliku od virtualnih strojeva koji sadrže cijele operacijske sustave i sve ovisnosti koje im trebaju za izvršavanje zadanih operacija (te su zbog toga zahtjevne za resurse), kontejneri s hostom dijele kernel, te u sebi sadržavaju samo aplikacije, njihove ovisnosti i potrebne knjižnice (eng. *library*).



Slika 1: *Hipervizor sjedi na fizičkom stroju i omogućuje kreiranje virtualnih strojeva. Svaki virtualni stroj ima svoj operacijski sustav, i zauzima određene resurse potrebne za pokretanje odvojenih OS-a. S druge strane, kontejneri su povezani s operacijskim sustavom fizičkog računala dijeleći njegov kernel i odvojene su samo aplikacije i njihove ovisnosti. Zbog toga kontejneri troše značajno manje resursa od virtualnih strojeva.*

Apptainer

Apptainer je jedno od kontejnerskih rješenja. Glavna svojstva su mu:

- mobilnost
- reproducibilnost
- korisnička sloboda
- podrška za HPC

Mobilnost računanja omogućuje da se workflow definira u kontejneru za koji korisnik može biti siguran da će se moći izvršavati na različitim strojevima, neovisno o operacijskom sustavu (dok god je Linux) i infrastrukturi. Mobilnost je osigurana time što se Apptainer kontejneri spremaju u slike (eng. *image*) koje su jedna datoteka koja obuhvaća sve datoteke u kontejneru. Slike se mogu izgraditi kao *read-only* (zadano ponašanje) čime se osigurava dosljednost u ponovnom korištenju.

Korisnička sloboda se odnosi na mogućnost korisnika da unutar kontejnera instaliraju aplikacije s poželjnim verzijama i knjižnicama, neovisno o stanju na poslužitelju na kojem će taj kontejner kasnije pokretati. Dopremanjem kontejnera na bilo koji resurs, korisnici mogu koristiti aplikacije koje god žele.

Apptainer je kompatibilan s HPC-om. Podržava InfiniBand, ima podršku za MPI, radi bez problema s većinom raspoređivača poslova te ima podršku i za rad s GPU-ovima.

Korisnik unutar Apptainer kontejnera je isti onaj koji je i izvan kontejnera te ima jednake ovlasti unutar i izvan kontejnera. Ako se kontejner koristi na klasteru, korisnik ne može unutar kontejnera napredovati do root ovlasti te nema opasnosti da nehotično (ili namjerno) napravi štetu sustavu. Ako korisnik mora mijenjati kontejner, morat će to raditi na lokalnom računalu, na kojem ima root ovlasti, pa promijenjeni kontejner ponovno dopremiti na klaster.

Apptainer je vrlo jednostavan za korištenjem, i koristi se kao bilo koja druga aplikacija (ne pokreće se servis kao npr. u slučaju Dockera - zato dobro radi s raspoređivačima poslova!).

Ima svega nekoliko osnovnih naredbi:

- `build`: naredba za izgradnju kontejnera
- `exec`: naredba za izvršavanje naredbe u kontejneru
- `inspect`: naredba za gledanje labela, runscripti, test scripti te varijabli okoline
- `pull`: naredba za povlačenje slika sa Singularity ili Docker Huba
- `run`: naredba za pokretanje slike kao izvršne datoteke
- `shell`: naredba za pokretanje ljuske u kontejneru

Na Supeku je dostupna verzija v1.1.6 aplikacije Apptainer.

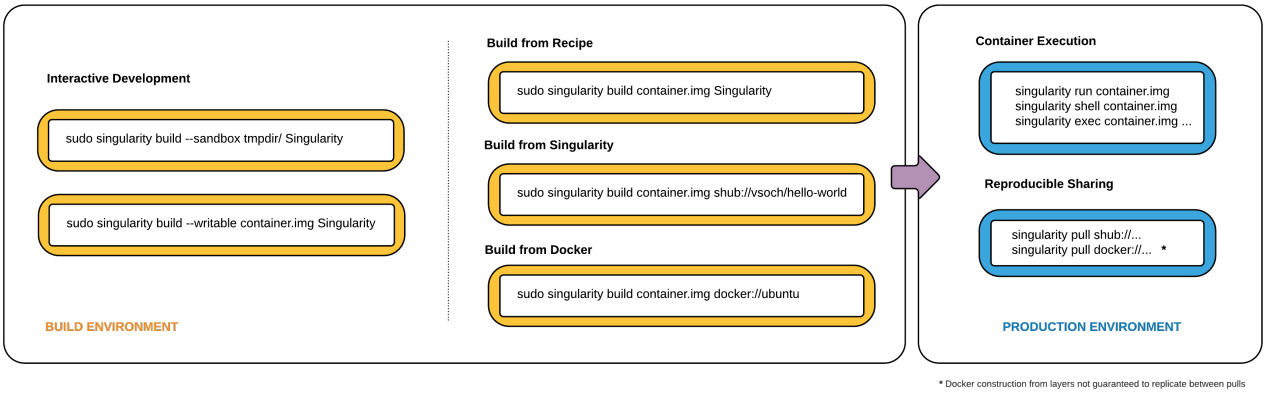
Apptainer i kontejneri

Tipični workflow za izgradnju kontejnera je sljedeći:

1. pronalaženje uputa za instalaciju tražene aplikacije
2. istraživanje odgovarajućeg operativnog sustava
3. kreiranje osnovnog kontejnera i njegova interaktivna nadogradnja
4. prebacivanje kontejnera u *image* ili izvršni oblik
5. dopremanje i izvršavanje na klasteru

Prva četiri koraka mogu se **u potpunosti** vršiti na korisničkom računalu, dok se posljednji vrši na Supeku gdje su dostupni značajniji resursi.

Napomena: Primjeri ispod se temelje na službenom predlošku, napravljenom u operativnom sustavu Ubuntu Focal Fossa (v20.04 LTS)



Slika 2: *Workflow za izgradnju i korištenje kontejnera. Slika je preuzeta*

sa stranice Singularitya; preteče Apptaineru koja funkcionira

na identičan način.

Izgradnja kontejnera - `build`

`build` je najbitnija komanda Apptainera kojom se kontejneri mogu izgraditi u obliku:

- *image* - slika ili oblik koji podržava samo čitanje, namijenjen finalnoj verziji kontejnera
- *sandbox* - direktorij ili oblik koji podržava i pisanje, namijenjen razvoju kontejnera

Primjerice, ako želimo direktno izgraditi kontejner s operativnim sustavom **ubuntu v20.04** (popularnim OS-om koji pruža veliki broj već prevedenih knjižnica kroz svoj upravitelj knjižnicama `apt`) osnovnu radnu verziju možemo izgraditi na sljedeći način:

```
# izgradnja kontejnera
[korisnik@kompjuter:~] $ apptainer build ubuntu_20.04.sif docker://ubuntu:20.04
...
INFO: Creating SIF file

# sadržaj trenutnog direktorija
[korisnik@kompjuter:~] $ ls -lrt
total 27100
-rwxr-xr-x 1 korisnik korisnik 27746304 svi  23 08:47 ubuntu_20.04.sif
```

Iako je ovaj *image* ograničene koristi (jer ubuntu sam po sebi obično ne sadrži aplikacije koje su nam potrebne), ovom komandom se mogu dohvatiti već pripremljeni recepti za koje postoje posvećeni repozitoriji, i koji mogu biti osnova za daljnju izgradnju kontejnera.

Ako želimo u gore naveden kontejner instalirati dodatne knjižnice, možemo to učiniti na dva načina:

1. interaktivnom izgradnjom korištenjem verzije *sandbox*
2. izgradnjom slike korištenjem definicijske datoteke *.def*

Nadogradnja kontejnera (lokalno) - `shell --writable`

Izrada kontejnera

Upute koje koriste `sudo` pri izgradnji nisu moguće na Supeku i Padobranu jer korisnici nemaju takve ovlasti.

Pri izgradnji ili nadogradnji na Supeku i Padobranu koristite opciju fakeroot (npr. za interaktivnu nadogradnju):

```
[korisnik@x3000c0s27b0n0 ~]$ apptainer shell --writable --fakeroot ...
```

U prvom slučaju, pri izgradnji kontejnera moramo navesti opciju `build --sandbox` koja će generirati kontejner u obliku direktorija `ubuntu_20.04`:

```
# izgradnja sandbox verzije
[korisnik@kompjuter:~] $ apptainer build --sandbox ubuntu_20.04 docker://ubuntu:20.04
...
INFO:    Creating sandbox directory...
INFO:    Build complete: ubuntu_20.04

# sadržaj trenutnog direktorija
[korisnik@kompjuter:~] $ ls -l
total 27104
drwxr-xr-x 18 korisnik korisnik     4096 svi  23 09:05 ubuntu_20.04
-rwxr-xr-x  1 korisnik korisnik 27746304 svi  23 08:47 ubuntu_20.04.sif
```

Novonastali *sandbox* kontejner možemo modificirati korištenjem opcije `shell --writable` kojom otvaramo ljusku unutar kontejnera sa svim korisničkim knjižnicama koje doprema.

Ispod se nalazi primjer osvježavanja instalacijskih repozitorija `apt` i instalacije jednostavne aplikacije cowsay:

```
# otvaranje sjednice u kontejneru
[korisnik@kompjuter] $ sudo apptainer shell ubuntu_20.04
INFO:    /etc/singularity/ exists; cleanup by system administrator is not complete (see https://apptainer.org
/docs/admin/latest/singularity_migration.html)
WARNING: Skipping mount /etc/localtime [binds]: /etc/localtime doesn't exist in container

# osvježavanje repozitorija
Apptainer> apt update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
...
All packages are up to date.

# instalacija cowsay
Apptainer> apt install cowsay -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libgdbm-compat4 libgdbm6 libperl5.30 libtext-charwidth-perl netbase perl perl-modules-5.30
...

# pokretanje aplikacije cowsay
Apptainer> /usr/games/cowsay Moo
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_ADDRESS = "hr_HR.UTF-8",
        LC_NAME = "hr_HR.UTF-8",
        LC_MONETARY = "hr_HR.UTF-8",
        LC_PAPER = "hr_HR.UTF-8",
        LC_IDENTIFICATION = "hr_HR.UTF-8",
        LC_TELEPHONE = "hr_HR.UTF-8",
        LC_MEASUREMENT = "hr_HR.UTF-8",
        LC_TIME = "hr_HR.UTF-8",
        LC_NUMERIC = "en_US.UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
 _____
< Moo >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Jednom kada smo zadovoljni sa *sandbox* verzijom koja je potrebna za izvršavanje naših aplikacija, možemo ju prevesti u *image* korištenjem naredbe `build`:

```
# prebacivanje sandbox verzije u image
[korisnik@kompjuter:~] $ sudo apptainer build ubuntu_20.04.sif ubuntu_20.04/
...
INFO:    Creating SIF file...
INFO:    Build complete: ubuntu_20.04.sif

# sadržaj trenutnog direktorija
[korisnik@kompjuter:~] $ ls -l
total 154496
drwxr-xr-x 18 korisnik korisnik      4096 svi  23 09:51 ubuntu_20.04
-rwxr-xr-x  1 korisnik korisnik 158195712 svi  23 09:58 ubuntu_20.04.sif
```

Novija image verzija sada će (osim osnovnog operativnog sustava ubuntu v20.04) sadržavati i aplikaciju cowsay.

Izgradnja receptom - `.def`

Drugi način (osim gore opisane interaktivne nadogradnje) je korištenjem recepta ili *definition* datoteka *.def* koje u sebi sadrže upute za izgradnju kontejnera.

Ovaj način je najpoželjniji jer osigurava dosljednu izgradnju *imagea* koja se može postepeno i dosljedno poboljšavati, no na uštrb brzine razvoja jer svakom se modifikacijom image mora ponovno izgraditi.

.def datoteke u sebi sadrže:

- zaglavlje ili *header* - definicija osnovnog operativnog sustava
- poglavlja ili *sections* - dijelovi koji definiraju razne dijelove izgradnje i korisnički okoliš koji je dostupan u kontejneru

U zaglavlju se definira bazni operacijski sustav koji će se koristiti u kontejneru. Osnovne ključne riječi su:

- `Bootstrap` - referencira tip baze koji će se koristiti
- `From` - ime kontejnera (ili referenca na *layere* u slučaju Dockera) koji će se koristiti

Nakon zaglavlja , izgradnja i namještanje okoline kontejnera zapisuje se u poglavljima. Neka od najvažnijih su:

- **%files** - dostavljanje datoteka u kontejner
- **%post** - izvršavanje komandi unutar kontejnera tijekom izgradnje
- **%environment** - definicija varijabli okoline unutar kontejnera tijekom izvršavanja (**nakon** njegove izgradnje)
- **%runscript** - zadane naredbe koje se izvršavaju korištenjem `apptainer run ...` ili direktnim izvršavanjem kontejnera

Primjer recepta sa službenih stranica napisan ispod u sebi će:

- izgraditi osnovni operativni sustav Ubuntu v20.04
- osvježiti apt repozitorije i instalirati aplikaciju `cowsay`
- uključiti izvršnu datotku `cowsay` iz `/usr/games` u korisničku okolinu
- definirati izvršnu naredbu `date | cowsay` kao zadanu
- definirati autora kontejnera

```
BootStrap: library
From: ubuntu:20.04

%post
    apt-get -y update
    apt-get -y install cowsay

%environment
        export LC_ALL=C
    export PATH=/usr/games:$PATH

%runscript
    date | cowsay

%labels
    Author GodloveD
```

Navedeni recept (npr. `ubuntu_20.04.def`) možemo izgraditi u *image* naredbom `build`:

```
# izgradnja imagea korištenjem naredbe build
[korisnik@kompjuter:~] $ apptainer build ubuntu_20.04.sif ubuntu_20.04.def
...
INFO:    Creating SIF file...
INFO:    Build complete: ubuntu_20.04.sif

# ispis detalja datoteke
[korisnik@kompjuter:~] $ ls -l ubuntu_20.04.sif
-rwxr-xr-x 1 marko marko 83701760 svi  25 10:41 ubuntu_20.04.sif
```

Korištenje javnih repozitorija

Proces izgradnje i nadogradnje kontejnera može se ubrzati korištenjem javnih repozitorija pri izradi recepta (u zagavlju) ili izgradnji *sandbox* kontejnera.

Neki od poznatijih repozitorija su:

- hub.docker.com - najpoznatija meka za kontejnere i obično prvi rezultat Google upita
- condaforge - kontejneri s već pripremljenim upraviteljima paketa conda
- Nvidia NGC - posvećeno svim aplikacijama koje koriste CUDU
- biocontainers.pro - posvećeno bioinformatici

Izgradnja na Supeku

U slučaju da niste u mogućnosti izgraditi kontejner na svojem osobnom računalu, pružamo mogućnost izgradnje kontejnera na pristupnom poslužitelju `log in-gpu.hpc.srce.hr`

Pri izgradnji, treba pripaziti na sljedeće:

- graditi u direktoriju `/scratch/apptainer` koji nije dio Lustre dijeljenog datotečnog sustava
- prebaciti kreirani *image* nazad u vaš korisnički direktorij, jer u protivnom neće biti vidljiv izvan pristupnog čvora `login-gpu.hpc.srce.hr`
- nakon izgradnje obrisati sve kreirane datoteke

Izrada kontejnera

Naredbe koje koriste `sudo` pri izgradnji nisu moguće na Supeku i Padobranu jer korisnici nemaju takve ovlasti.

Pri izgradnji ili nadogradnji na Supeku koristite opciju fakeroot (npr. za interaktivnu nadogradnju):

```
[korisnik@x3000c0s27b0n0 ~]$ apptainer shell --writable --fakeroot ...
```

APPTAINER_TMPDIR - Staza za privremene datoteke

Pri izgradnji ili nadogradnji, apptainer kreira privremene datoteke koje se spajaju u konačni kontejner i koristi stazu `/tmp` kao zadani direktorij.

`/tmp` direktorij na pristupnom poslužitelju je dijelom datotečnog prostora vezanog uz operativni sustav pristupnog poslužitelja i ima ograničen prostor od 178 gigabajta.

U trenutcima jače korisničke aktivnosti, a pogotovo višestruke izgradnje kontejnera koji mogu imati i do desetak gigabajta, zbog zadanog ponašanja s privremenim datotekama može doći do preopterećenja diskovnog prostora i smanjene funkcionalnosti operativnog sustava za sve korisnike.

U svrhu izbjegavanja ovog tipa preopterećenja, preporučeno je postaviti varijablu okoliša `APPTAINER_TMPDIR` prije izgradnje koja mijenja zadani direktorij i postaviti ga (primjerice) na direktorij unutar `/scratch` prostora (opisano u primjeru ispod)

Primjer *sandbox* izgradnje s dodatnim knjižnicama korištenjem Ubuntu v20.04 operativnog sustava:

```
# login na pristupni poslužitelj gpu
[korisnik@kompjuter ~]$ ssh korisnik@login-gpu.hpc.srce.hr
Last login: Wed May 24 09:23:06 2023 from x.x.x.x

# pomicanje u /scratch i kreiranje direktorija za izgradnju
[korisnik@x3000c0s27b0n0 ~]$ cd /scratch/apptainer
[korisnik@x3000c0s27b0n0 apptainer]$ mkdir ${USER}
[korisnik@x3000c0s27b0n0 apptainer]$ cd ${USER}

# postavljanje varijable okoliša za privremene datoteke
[korisnik@x3000c0s27b0n0 korisnik]$ export APPTAINER_TMPDIR=/scratch/apptainer/${USER}

# izgradnja sandbox verzije
[korisnik@x3000c0s27b0n0 korisnik]$ apptainer build --sandbox ubuntu_20.04_sandbox docker://ubuntu:20.04
[korisnik@x3000c0s27b0n0 korisnik]$ apptainer shell --writable --fakeroot ubuntu_20.04_sandbox
Apptainer> ...
Apptainer> ... dodatne komande za izgradnju kontejnera ...
Apptainer> ...
Apptainer> exit

# mijenjanje sanboxa u image, prebacivanje u korisniki i
[korisnik@x3000c0s27b0n0 korisnik]$ apptainer build ubuntu_20.04.sif ubuntu_20.04_sandbox
[korisnik@x3000c0s27b0n0 korisnik]$ mv ubuntu_20.04.sif ~
[korisnik@x3000c0s27b0n0 korisnik]$ cd ~

# brisanje direktorija za izgradnju
[korisnik@x3000c0s27b0n0 ~]$ rm -rf /scratch/apptainer/${USER}
```

Korištenje kontejnera

Kada jednom imamo gotov kontejner, postoji nekoliko naredbi za interagiranje s njim:

- `shell` - pokretanje korisničke ljuske
- `exec` - pokretanje naredbe
- `run` - pokretanje zadane naredbe

Naredbom `shell` pokreće se ljuska unutar kontejnera, te se interagira s kontejnerom kao da je pokrenuta ljuska na malom virtualnom stroju.

```
# otvaranje ljuske
[korisnik@kompjuter:~] $ singularity shell apptainer.sif
...

# izvršavanje komande unutar kontejnera
Apptainer> date | cowsay
 _______________________________
< Thu May 25 10:59:05 CEST 2023 >
 -------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

# izlaženje iz ljuske u kontejneru
Apptainer> exit

# izvršavanje iste komande izvan kontejnera
[korisnik@kompjuter:~] $ date | cowsay
bash: cowsay: command not found
```

Naredbom `exec` izvršava se naredba unutar kontejnera:

```
# izvršavanje naredbe cowsay unutar kontejnera
[korisnik@kompjuter:~] $ date | apptainer exec apptainer.sif cowsay
...
 < et, 25.05.2023. 11:05:46 CEST  >
 --
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Naredbom `run` pokreće se kontejner pozivanjem *runscripte* definirane u Apptainer receptu.

```
# izvršavanje komandom run
[korisnik@kompjuter:~] $ apptainer run apptainer.sif
...
 < et, 25.05.2023. 11:05:46 CEST  >
 --
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

# direktno izvršavanje
[korisnik@kompjuter:~] $ ./apptainer.sif
...
 < et, 25.05.2023. 11:06:05 CEST  >
 --
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Podnošenje kontejnera na Supeku

U slučaju da se koristi Apptainer kontejner na Supeku, podnošenje poslova je jednako kao i u slučaju aplikacija koje su instalirane direktno na klasteru.

Na primjer, ako želimo pokrenuti kontejner `apptainer.sif` nakon što ga dopremimo na Supek, minimalna skripta PBS `apptainer-test.sh` bit će:

```
#!/bin/bash
#PBS -N apptainer-test
apptainer run apptainer.sif
```

Koju možemo podnijeti na sljedeći način i provjeriti njen izlaz nakon malo vremena:

```
# podnošenje skripte apptainer-test.sh
[korisnik@x3000c0s25b0n0:~] $ qsub apptainer-test.sh

# nakon par sekundi provjerimo sadržaj direktorija
[korisnik@x3000c0s25b0n0:~] $ ls -l apptainer-test*
-rw------- 1 mkvakic hpc  75 May 25 11:31 apptainer-test.e10537
-rw------- 1 mkvakic hpc 229 May 25 11:31 apptainer-test.o10537
-rw-r--r-- 1 mkvakic hpc  63 May 25 11:31 apptainer-test.sh

# ispišemo sadržaj standardnog izlaza
[korisnik@x3000c0s25b0n0:~] $ cat apptainer-test.o10537
 _________________________________
< Thu May 25 09:31:17 Europe 2023 >
 ---------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

MPI aplikacije u kontejneru

Apptainer podržava paralelizaciju i širenje na više čvorova korištenjem MPI standarda na dva načina:

1. *hybrid model* - izvršavanje korištenjem MPI implementacije **u kontejneru**
2. *bind model* - izvršavanje korištenjem MPI implementacije **izvan kontejnera**

Na Supeku se preporučuje koristiti *bind model* zbog knjižnica koje osiguravaju integraciju sa sustavom za upravljanje PBS i mrežom Slingshot. Pri tom aplikacija koja koristi MPI mora biti ABI kompatibilna (kompajlirana sa MPICH verzija => v3.1) ili kompajlirana sa openMPI 5.x.x verzijom. Isto tako, potrebno je koristiti RHEL-8.6 kompatibilne linux distribucije.

Osnovni operativni sustav kontejnera

Zbog *bind model* spajanja, potrebno je koristiti operativne sustave koji su slični operativnom sustavu Supeka Red Hat Enterprise Linux 8.6.

Neki od mogućih su:

- Rocky Linux 8
- AlmaLinux 8
- CentOS Stream 8

Kako bi korisnicima olakšali izgradnju kontejnera i pokretanje aplikacija sa bind modelom, pripremili smo početne slike kontejnera povrh kojih je moguće instalirati i kompajlirati aplikacije. Pripremljene slike sadrže MPICH i openMPI kompatibilne verzije sa MPI implementacijama dostupnim na Supeku. Osim slika, pripremljeni su i odgovarajući moduli koji će spojiti potrebne knjižnice i direktorije na Supeku sa kontejnerom.

U tablici su prikazane putanje slika za izgranju kontejnera i odgovarajući moduli ovisno o MPI implementaciji:

MPI kontejner	OS kontejner	MPI model	MPI Supek	Polazni kontejner	Bind modul
mpich-4.1.1	Rocky-8	Bind	cray-mpich-8.1.26	/apps/utils/apptainer-images/rockylinux-8-mpich-4.1.1.sif	utils/apptainer-bind/cray-mpich
openMPI-5.0.1	Rocky-8	Bind	Openmpi- 5.0.1-gnu-8.5.0	/apps/utils/apptainer-images/rockylinux-8-openmpi-5.0.1.sif	utils/apptainer-bind/openmpi-5.0.1
openMPI-4.1.2	Ubuntu-22.04	Hybrid	OpenMPI 4.x.x	/apps/utils/apptainer-images/ubuntu-22.04-openmpi-4.1.2.sif	X

Primjer izgradnje kontejnera koristeći MPICH(v4.1.1) :

Definicijska datoteka (recept)

```
Bootstrap: localimage
From: /apps/utils/apptainer-images/rockylinux-8-mpich-4.1.1.sif

%files

  #kopiranje datoteka
  mpi.latency.c         /testmpi/

%post

  # prevedi aplikaciju u izvršnu datoteku
  mpicc     /testmpi/mpi.latency.c             -o /testmpi/mpi.latency
```

run_cray-mpich.sh

```
#PBS -q cpu
#PBS -l select=2:ncpus=1
#PBS -l place=scatter

cd $PBS_O_WORKDIR

module load utils/apptainer-bind/cray-mpich

mpiexec --no-transfer apptainer exec mpich.sif /testmpi/mpi.latency
```

mpi.latency.c

```
.mpitutorial.com
// This code is provided freely with the tutorials on mpitutorial.com. Feel
// free to modify it for your own use. Any distribution of the code must
// either provide a link to www.mpitutorial.com or keep this header intact.
//
// An intro MPI hello world program that uses MPI_Init, MPI_Comm_size,
// MPI_Comm_rank, MPI_Finalize, and MPI_Get_processor_name.
//
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
  // Initialize the MPI environment. The two arguments to MPI Init are not
  // currently used by MPI implementations, but are there in case future
  // implementations might need the arguments.
  MPI_Init(NULL, NULL);

  // Get the number of processes
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // Get the rank of the process
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  // Get the name of the processor
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int name_len;
  MPI_Get_processor_name(processor_name, &name_len);

  // Print off a hello world message
  printf("Hello world from processor %s, rank %d out of %d processors\n",
         processor_name, world_rank, world_size);

  // Finalize the MPI environment. No more MPI calls can be made after this
  MPI_Finalize();
}
```

Grafički procesori i CUDA

Apptainer pruža podršku za izvršavanje na grafičikim procesorima ako su aplikacije i kontejneri namijenjeni za njihovo korištenje.

Ova mogućnost jednostavno se ostvaruje korištenjem opcije `--nv` pri izvršavanju kontejnera komandama `exec` ili `run`.

Ispod se nalazi primjer izgradnje kontejnera PyTorch v1.13.1 za duboko učenje i njegovo testiranje na pristupnom poslužitelju s grafičkim čvorovima:

```
# spajanje na pristupni poslužitelj
[korisnik@kompjuter:~] $ ssh mkvakic@login-gpu.hpc.srce.hr

# izgradnja pytorch kontejnera
[korisnik@x3000c0s27b0n0:~] $ apptainer build pytorch_1.13.1.sif docker://pytorch/pytorch:1.13.1-cuda11.6-
cudnn8-runtime

# print raznih informacija korištenjem --nv
[korisnik@x3000c0s27b0n0:~] $ apptainer run --nv ./pytorch_1.13.1.sif python -c "import torch; print('Is cuda
available?:', torch.cuda.is_available())"
Is cuda available?: True
[korisnik@x3000c0s27b0n0:~] $ apptainer run --nv ./pytorch_1.13.1.sif python -c "import torch; print('Device
count:', torch.cuda.device_count())"
Device count: 1
[korisnik@x3000c0s27b0n0:~] $ apptainer run --nv ./pytorch_1.13.1.sif python -c "import torch; print('Current
device:', torch.cuda.current_device())"
Current device: 0
[korisnik@x3000c0s27b0n0:~] $ apptainer run --nv ./pytorch_1.13.1.sif python -c "import torch; print('Device
name:', torch.cuda.get_device_name(0))"
Device name: NVIDIA A100-PCIE-40GB
```

Nadogradnja sandbox kontejnera sa korištenjem --nv opcije moguća je nakon modifikacije sandbox direktorija **izvan** kontejnera.

Korištenje "sandbox" kontejnera sa –nv opcijom

```
# izgradnja test.sandbox sandbox slike kontejnera
[korisnik@x3000c0s27b0n0:~] $ apptainer build --sandbox --fix-perms test.sandbox test.def

# modifikacija slike kontejnera
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/usr/bin/nvidia-smi
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/usr/bin/nvidia-debugdump
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/usr/bin/nvidia-persistenced
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/usr/bin/nvidia-cuda-mps-control
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/usr/bin/nvidia-cuda-mps-server
[korisnik@x3000c0s27b0n0:~] $ mkdir -p test.sandbox/var/run/nvidia-persistenced/
[korisnik@x3000c0s27b0n0:~] $ touch test.sanbox/var/run/nvidia-persistenced/socket

# pokretanje "sandbox" kontainera sa --nv opcijom
[korisnik@x3000c0s27b0n0:~] $ apptainer shell --nv --writable --fakeroot test.sandbox
```

NVIDIA NGC kontejneri

```
NVIDIA driveri na pristupnom poslužitelju su nadograeni i usklaeni sa radnim poslužiteljima u sijenju
2024. godine!
Izgradnja valjanog kontejnera sa NGC repozitorija mogua je na pristupnom poslužitelju bez --bind opcije
ili dodatnih naredbi i konfiguracije.
```

Nvidia NGC je repozitorij namijenjen korištenju kontejnera optimiziranih za izvršavanje na NVIDIA grafičkim procesorima i sadrži sijaset recepata za popularne aplikacije (poput strojnog učenja) pogodnih za izgradnju i modifikaciju za specifične korisničke potrebe.

Pri nadogradnji (interaktivnoj ili putem definicijskih datoteka), jedna od skripti unutar kontejnera postavlja korisničku okolinu (`/etc/shinit_v2`) i osigurava tzv. "forward" kompatibilnost na način da učitava datoteku `/proc/driver/nvidia/version` s domaćina, stvarajući sloj datoteka (`/usr/local/cuda-11.8/compat/lib`) koje usklađuju verziju CUDE s driverima na pristupnom poslužitelju (525.60.13):

```
# izgradi proizvoljan kontejner ngc korištenjem repozitorija NGC ija verzija CUDE (npr. v11.8) ovisi o driveru
520.61.05
[korisnik@x3000c0s27b0n0 ~] apptainer build ngc-kontejner.sif ngc-kontejner.def
...

# ispiši sadržaj /usr/local/cuda-11.8/compat/lib unutar kontejnera
[korisnik@x3000c0s27b0n0 ~] apptainer exec ngc-kontejner.sif ls -l /usr/local/cuda-11.8/compat/lib.real/lib
13:4: not a valid test operator: (
13:4: not a valid test operator: 525.60.13
total 145702
lrwxrwxrwx 1 korisnik korisnik       12 Sep 29  2022 libcuda.so -> libcuda.so.1
lrwxrwxrwx 1 korisnik korisnik       20 Sep 29  2022 libcuda.so.1 -> libcuda.so.520.61.05
-rw-r--r-- 1 korisnik korisnik 26284256 Sep 29  2022 libcuda.so.520.61.05
lrwxrwxrwx 1 korisnik korisnik       28 Sep 29  2022 libcudadebugger.so.1 -> libcudadebugger.so.520.61.05
-rw-r--r-- 1 korisnik korisnik 10934360 Sep 29  2022 libcudadebugger.so.520.61.05
lrwxrwxrwx 1 korisnik korisnik       19 Sep 29  2022 libnvidia-nvvm.so -> libnvidia-nvvm.so.4
lrwxrwxrwx 1 korisnik korisnik       27 Sep 29  2022 libnvidia-nvvm.so.4 -> libnvidia-nvvm.so.520.61.05
-rw-r--r-- 1 korisnik korisnik 92017376 Sep 29  2022 libnvidia-nvvm.so.520.61.05
lrwxrwxrwx 1 korisnik korisnik       37 Sep 29  2022 libnvidia-ptxjitcompiler.so.1 -> libnvidia-ptxjitcompiler.
so.520.61.05
-rw-r--r-- 1 korisnik korisnik 19963864 Sep 29  2022 libnvidia-ptxjitcompiler.so.520.61.05
```

Pri izvršavanju aplikacija unutar ovog kontejnera, usklađenost kontejnera testira se izvršnom datotekom `/usr/local/bin/cudaCheck`:

```
# testiraj usklaenost na pristupnom poslužitelju GPU
[korisnik@x3000c0s27b0n0 ~] apptainer exec --nv ngc-kontejner.sif /usr/local/bin/cudaCheck
INFO:    underlay of /etc/localtime required more than 50 (95) bind mounts
INFO:    underlay of /usr/bin/nvidia-smi required more than 50 (474) bind mounts
13:4: not a valid test operator: (
13:4: not a valid test operator: 525.60.13
CUDA Driver OK

# spoji se interaktivnom sjednicom na jedan od radnih GPU vorova
[korisnik@x3000c0s27b0n0 ~] qsub -l ngpus=1 -I
qsub: waiting for job 107908.x3000c0s25b0n0.hsn.hpc.srce.hr to start
qsub: job 107908.x3000c0s25b0n0.hsn.hpc.srce.hr ready

# testiraj usklaenost na radnom voru
[korisnik@x8000c2s4b0n1 ~] apptainer exec --nv ngc-kontejner.sif /usr/local/bin/cudaCheck
INFO:    underlay of /etc/localtime required more than 50 (95) bind mounts
INFO:    underlay of /usr/bin/nvidia-smi required more than 50 (474) bind mounts
13:4: not a valid test operator: (
13:4: not a valid test operator: 525.60.13
CUDA Driver OK
```