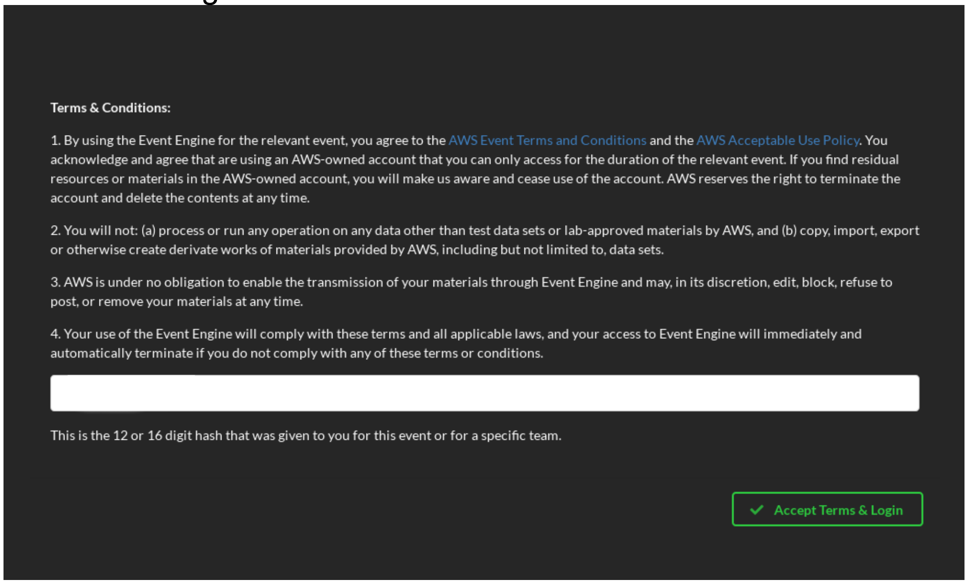
Easy-Build Singularity Containers

NIHR | Guy's and St Thomas' Biomedical Research Centre **Tutorial** aws

Welcome to the Easy-Build Singularity Containers Tutorial. The aim of the tutorial is to give an easy entry route for those who are not IT experts to build a Singularity Container, which then can be executed on any suitable platform, i.e. one which has Singularity installed. That is somehow similar to the Docker concept.

Accessing the Cloud instance

We are using an AWS Cloud9 instance. In the email I have sent out was an unique link to access this VM. Please paste this link in the webbrowser of your choice.
You should see something like this:



Terms & Conditions:

1. By using the Event Engine for the relevant event, you agree to the AWS Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.

2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.

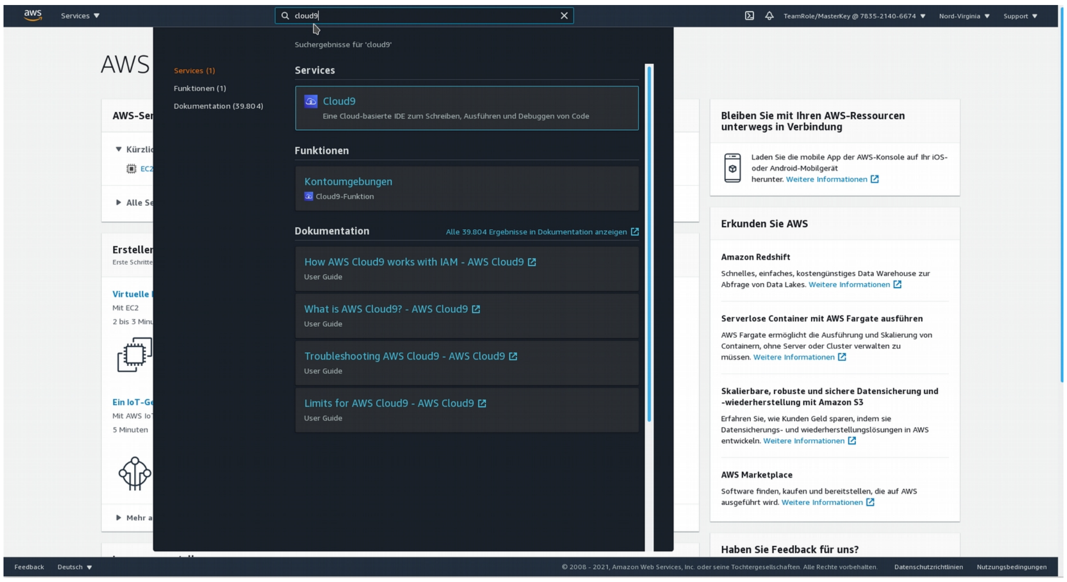
3. AWS is under no obligation to enable the transmission of your materials through Event Engine and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.

4. Your use of the Event Engine will comply with these terms and all applicable laws, and your access to Event Engine will immediately and automatically terminate if you do not comply with any of these terms or conditions.

This is the 12 or 16 digit hash that was given to you for this event or for a specific team.

The white box at the bottom should contain your unique code. If it does not, it is also the last part of the URL.

In the search-bar at the top, search for “cloud9” as shown in the following picture.



Services ▼ cloud9

TeamRole/MasterKey @ 7835-2140-6674 ▼ Nord-Virginia ▼ Support ▼

Suchergebnisse für 'cloud9'

Services (1)
Funktionen (1)
Dokumentation (39.804)

Services

Cloud9
Eine Cloud-basierte IDE zum Schreiben, Ausführen und Debuggen von Code

Funktionen

Kontoumgebungen
Cloud9-Funktion

Dokumentation — Alle 39.804 Ergebnisse in Dokumentation anzeigen

How AWS Cloud9 works with IAM - AWS Cloud9
User Guide

What is AWS Cloud9? - AWS Cloud9
User Guide

Troubleshooting AWS Cloud9 - AWS Cloud9
User Guide

Limits for AWS Cloud9 - AWS Cloud9
User Guide

Bleiben Sie mit Ihren AWS-Ressourcen unterwegs in Verbindung

Laden Sie die mobile App der AWS-Konsole auf Ihr iOS- oder Android-Mobilgerät herunter. Weitere Informationen

Erkunden Sie AWS

Amazon Redshift
Schnelles, einfaches, kostengünstiges Data Warehouse zur Abfrage von Data Lakes. Weitere Informationen

Serverlose Container mit AWS Fargate ausführen
AWS Fargate ermöglicht die Ausführung und Skalierung von Containern, ohne Server oder Cluster verwalten zu müssen. Weitere Informationen

Skalierbare, robuste und sichere Datensicherung und -wiederherstellung mit Amazon S3
Erfahren Sie, wie Kunden Geld sparen, indem sie Datensicherungs- und wiederherstellungslösungen in AWS entwickeln. Weitere Informationen

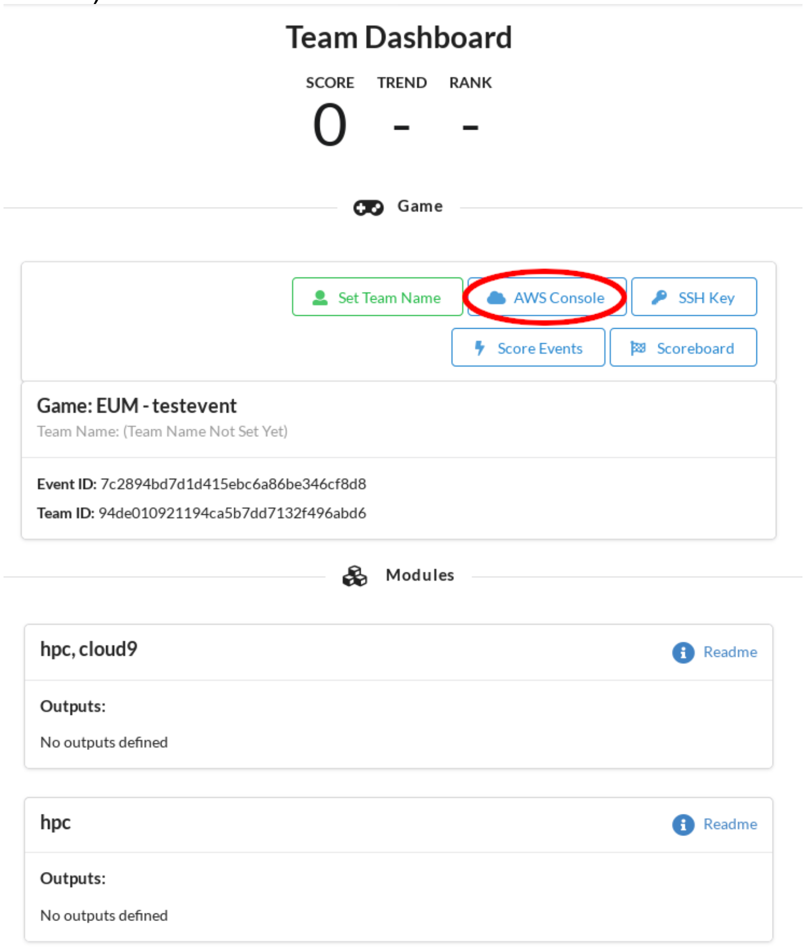
AWS Marketplace
Software finden, kaufen und bereitstellen, die auf AWS ausgeführt wird. Weitere Informationen

Haben Sie Feedback für uns?

Feedback Deutsch ▼ © 2008 - 2021, Amazon Web Services, Inc. oder seine Tochtergesellschaften. Alle Rechte vorbehalten. Datenschutzrichtlinien Nutzungsbedingungen

Click at the suggested “Cloud9” right at the top at the “Service” tab.

Navigate further until you get to the “Team Dashboard”. We need to use the “AWS Console” tab (red circle):



Team Dashboard

SCORE TREND RANK
0 - -

Game

Set Team Name | AWS Console | SSH Key
Score Events | Scoreboard

Game: EUM - testevent
Team Name: (Team Name Not Set Yet)

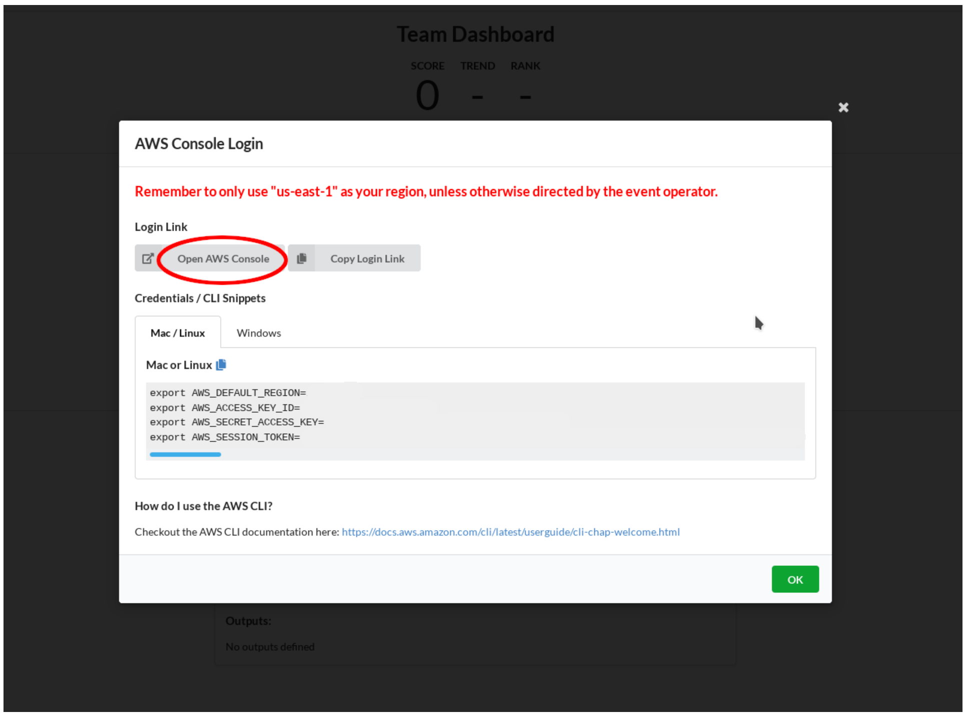
Event ID: 7c2894bd7d1d415ebc6a86be346cf8d8
Team ID: 94de010921194ca5b7dd7132f496abd6

Modules

hpc, cloud9 — Readme
Outputs:
No outputs defined

hpc — Readme
Outputs:
No outputs defined

This will open a box, again we want to click on the “Open AWS Console” tab.



Remember to only use "us-east-1" as your region, unless otherwise directed by the event operator.

```
export AWS_DEFAULT_REGION=
export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_SESSION_TOKEN=
```

Checkout the AWS CLI documentation here: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

This should open a terminal inside the browser. You might want to maximise it (top right corner) and you might want to increase the font (top right corner, cogwheel symbol).

There are a few things we need to do before we can make use of the VM. The space there is quite tight, that is as some Docker images are pre-installed. We want to remove them like this:

```
$ df -h
$ docker rmi $(docker image ls -q |xargs)
$ df -h
```

You should see that some space has been freed up now.

Next, we might need to install Singularity. As we are using CentOS, the command is this:

```
$ sudo yum install singularity
```

The next step is to clone the GitHub account with all the scripts to build the containers:

```
$ git clone https://github.com/sassy-crick/Singularity-Easybuild.git
```

This should clone the GitHub account into your current working directory.
As you might have noticed, inside the “Singularity-Easybuild” folder are two subfolders:

– definitions
– scripts

For now, we only need the script folder as it contains the bash-scripts we need.
For a production environment, either you are copying these scripts, or the ones you want to use, in your ~/bin folder, if you are on Linux, or you memorize the path to it. Up to you.

Now we got the bash script, but how are we using them? The other bit of information we need is the **name** of the EasyBuild Configuration file, often referred to as EC. We can get that from their GitHub repository here:

https://github.com/easybuilders/easybuild-easyconfigs/tree/master/easybuild/easyconfigs

Look out for files with the suffix **.eb**, as these are the EC files we want.

In our tutorial we are only using two as they are quick to build. The aim is to understand how to use the provided tools, not to spent hours building software.
We are going to use

– bzip2-1.0.6.eb
– Cmake-3.12.1.eb

Both are quick to build, the former is faster so I would recommend to do this. Also, we will open up that container in the next part, so any time saved now is time we got later.

Creating the Singularity Definition File

First we build the Singularity Definition file. You will need to substitute the PATH/TO with your path to the bash scripts (see above).

```
$ PATH/TO/container-build-centos7-envmodules.sh bzip2-1.0.6.eb
```

You will see the following output:

```
The Singularity definition file ~/.singularity/sing-eb.conf does not exist.
Please use this file to set the author and email address which is used in the Singularity
definition file.
The use of this is optional.
The syntax is:
author="YOUR NAME"
email="EMAIL ADDRESS"
Do we need a second Easybuild recipe (y/N)?: n
```

It is not mandatory to provide your name and email address here. This is, however, probably a good feature if you want to publish the Singularity Definition File later with your publication, or put it into a repository. We do not have a second EC, which could be a home-made file or some file from the development branch of EasyBuild. Here we really need the file and not only the file name.

Once executed, we should see our expected file:

```
$ ls
Singularity.bzip2-1.0.6-envmod-centos7
```

Building the Singularity Container

Now that we have the definition file, we can build the container like this:

```
$ sudo singularity build bzip2-1.0.6-envmod-centos7.sif Singularity.bzip2-1.0.6-envmod-centos7
```

The syntax is as follow:

- sudo is needed to build the container
- singularity is the command we execute
- build is what we want to do (building a container)
- bzip2-1.0.6-envmod-centos7.sif is the name of the image, use something sensible
- Singularity.bzip2-1.0.6-envmod-centos7 is the name of the Definition file

This will take some time, depending on the programs you want to install, how many cores you have and your network speed.

The build steps are like this:

- creat a chroot environment and install a bootstrap image there
- update this, so we got the latest packages
- install any package mention in the Definition file
- prepare the image so EasyBuild can be installed. This includes pip and the “easybuild” user installation amongst other things
- modify the bash environment of the “easybuild” user

- create the script to install the software as user “easybuild”
- install the requested software with EasyBuild as user “easybuild”
- - download the packages
- - build the packages
- remove anything we don't really need to make sure the container does not get too big
- do the post-installation for the container. Here not only the module which we need to execute the container will be loaded, also some meta-data like Author etc. will be added
- pack the container and create the image

Once this is done, you can use the container like this (silly example, for demonstration only):

```
$ singularity run bzip2-1.0.6-envmod-centos7.sif bzip2 --help
```

Instead of --help you also can use --version, it is up to you.

Opening container and add software to it

Obviously, building a large container with only bzip in it is a bit of a waste. So, we are going to open up the container and install Cmake in it. Again, that is only to demonstrate how you can do it.

First we unpack the container:

```
$ sudo singularity build --sandbox Cmake-3.12.1-envmod-centos7 bzip2-1.0.6-envmod-centos7.sif
```

The “--sandbox” means we are building a “chroot” like file system. Note: As the container is compressed, the file system might be larger!

Once this is done, we go inside the file system:

```
$  sudo singularity shell -w CMake-3.12.1-envmod-centos7
```

The “-w” means writeable, which is also the reason why we need to use “sudo” again.

Inside the container we do:

```
Singularity> su -l easybuild
```

(Note the changed prompt!)

As we are using Environment Modules, there is a bit more to do. We push that into an alias, as shown here:

```
# alias
alias eb='eb --robot --modules-tool=EnvironmentModulesC --module-syntax=Tcl --
download-timeout=1000'
```

So, every time you are executing the command “eb”, in reality you do the whole string printed above! The important bit here is the “--robot” option, which is one of the really good strenght of EasyBuild: it is resolving all dependencies and build not only the requested, but also any dependent software for your software!

We do that now:

```
# eb Cmake-3.3.1.eb
```

Now you can watch the robot working for you whilst you can do something differently. This is great as we are all busy.

When it is all done, we can check what has been build:

```
# module av

------------------------------ /app/modules/all--------------------------------------------------------------
CMake/3.3.1 bzip2/1.0.6 ncurses/5.9
```

As you can see, we got the requested “CMake” installed, the dependency “ncurses” and we still have “bzip” from the first install.

Now we become root inside the container again and do some required changes:

```
# exit
```

First we need to install a text editor of our choice. I am using “vim”, but you can do a different one as well.
Note: you also could do the changes outside the container as root.

```
> yum install vim
> vim /environment
```

change:

```
# load module(s) corresponding to installed software
module load bzip2/1.0.6
```

to

```
# load module(s) corresponding to installed software
module load CMake/3.3.1
```

Save and close your text editor.

For larger installation, it might be a good idea to remove any unwanted files. We need to do that manually, before it was the script doing that:

```
> cd /scratch/
> du -sh .
9.1M    .
!!!! CAREFUL HERE !!!!
> rm -rf /scratch/*
```

Now we exit completely and build the final container:

```
> exit
```

```
$ sudo singularity build CMake-3.3.1-envmod-centos7.sif Cmake-3.12.1-envmod-centos7
```

Note: depending on what you have done, you might run out of disk space. Thus, it might be a good idea to remove any ".sif" containers first before you build it.

Exercise

I have provided a Singularity container containing GCC-9.3.0 in a Debian-9 environment utilizing the Environment Modules. You can get it like this (remember the tight space we have):

```
$ aws s3 cp s3://ee-assets-prod-us-east-1/modules/45c5cd1e2b8749228899ab149ccddada/v1/GCC-9.3.0-envmod-debian10.sif .
```

(Apologies for the small font. Please note the dot at the end of the line!)

You can take that container with you and play around with it. Here all you need is a Singularity installation. This can be on any Linux, Mac and should work for Windows as well.

I hope you found that brief introduction useful. Suggestions are welcome, any beer please sent to Christian Kniep and his team from Amazon, who very kindly provided the VMs for this session.

Disclaimer: Although everything was tested, I am not liable in any shape or form for any problems or errors. :-)