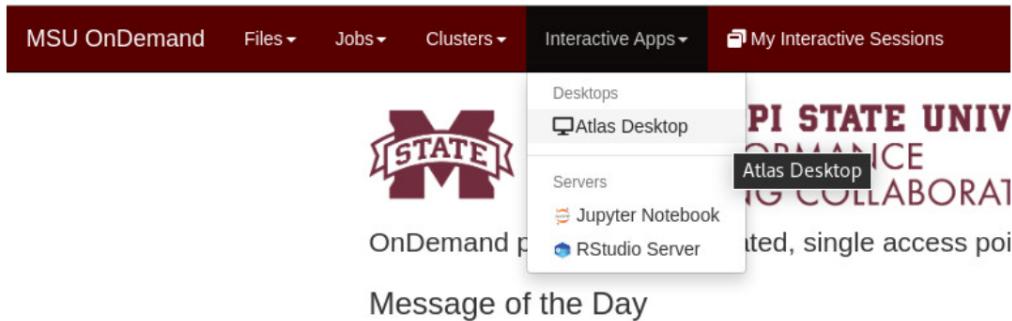


Protein Structure, prediction, search, and analysis with AI

(Day 3, November 21, 2024; 1:30 PM - 5:00 PM)

SET UP

Log into Atlas Open on Demand ([Dashboard - Mississippi State - Atlas Open OnDemand](#)).



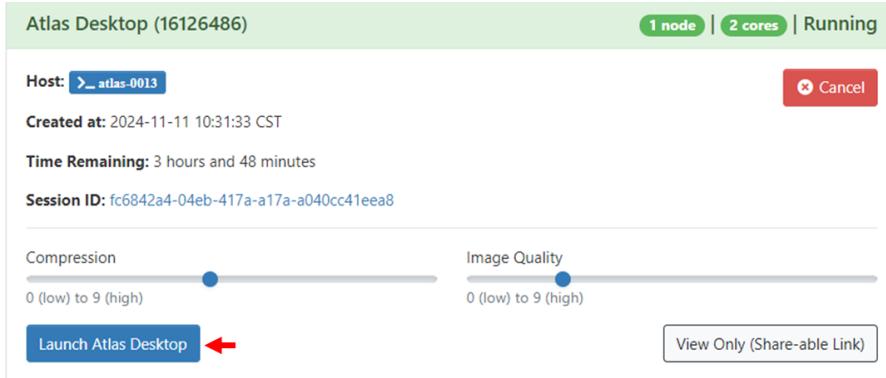
- ⇒ You need to specify what Slurm Account, Partition, and QOS to run the desktop under, how long you need the virtual desktop, how many nodes and processor be allocated for the session that you requested (--mem=32G -t 04:00:00 --account=scinet_workshop1).

The screenshot shows the "Atlas Desktop" configuration page. On the left is a sidebar with "Interactive Apps" selected, and "Atlas Desktop" is highlighted. The main area is titled "Atlas Desktop" and contains the following fields:

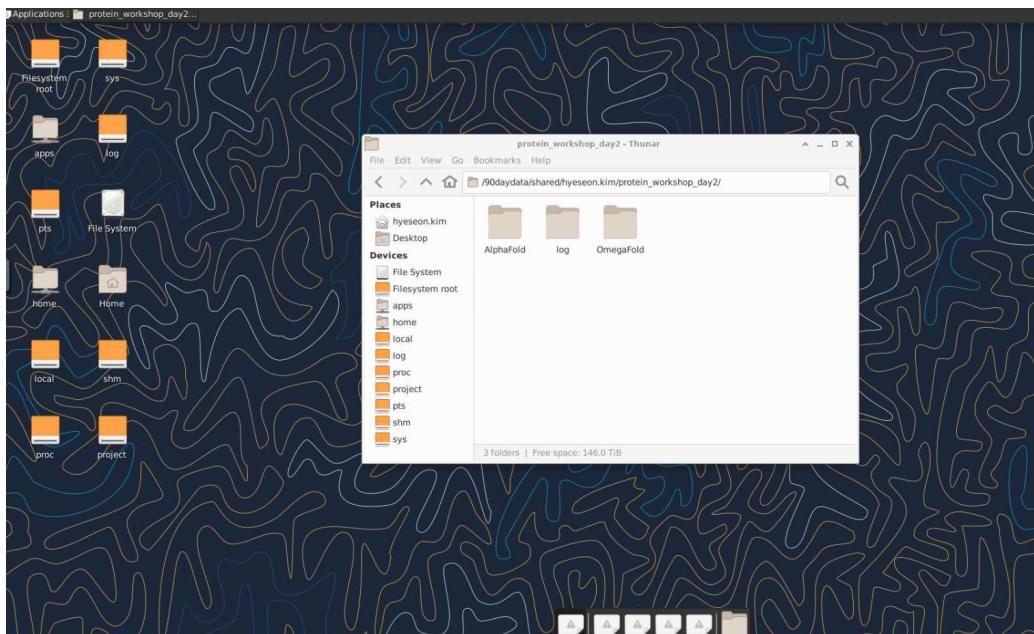
- Account (--account=value): scinet_workshop1
- Partition (--partition=value): atlas
- QOS (--qos=value): ood 8:00:00:00
- Number of hours (--time=value): 4
- Number of nodes (--nodes=value): 1
- Number of tasks (--ntasks=value): 2
- Additional Slurm Parameters: `--mem=32G -t 04:00:00 --account=scinet_workshop1`

A note at the bottom states: "Parameters such as --exclusive, --mem=value(M/G), or --reservation=value can be specified here". There is also a checkbox: "I would like to receive an email when my job is near the time limit". At the bottom is a large blue "Launch" button.

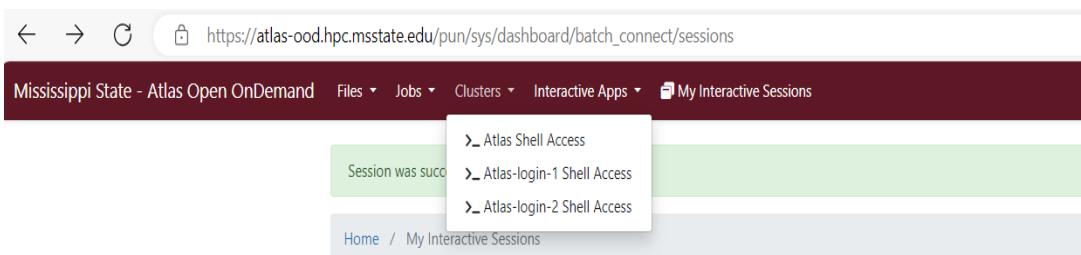
- ⇒ After setting up the necessary parameters for the batch job, click "Launch" to open the session into the scheduling queue.



- ⇒ Once the job has been given an allocation, you will be able to connect to the virtual desktop session by clicking the “Launch Atlas Desktop” button.



- ⇒ Users can access the terminal and navigate the folders in the virtual desktop.



- ⇒ To access the command shell via ood, go to the “Clusters” tab and select the shell access (<https://atlas-ood.hpc.msstate.edu/pun/sys/shell/ssh/Atlas-login.hpc.msstate.edu>).

```
Host: Atlas-login.hpc.msstate.edu

***** N O T I C E *****

This system is under the control of and/or the property of Mississippi State University (MSU). It is for authorized use only. By using this system, all users acknowledge notice of and agree to comply with all MSU and High Performance Computing Collaboratory (HPC2) policies governing use of information systems.

Any use of this system and all files on this system may be intercepted, monitored, recorded, copied, audited, inspected, and disclosed to authorized university and law enforcement personnel, as well as authorized individuals of other organizations. By using this system, the user consents to such interception, monitoring, recording, copying, auditing, inspection and disclosure at the discretion of authorized university personnel.

Unauthorized, improper or negligent use of this system may result in administrative disciplinary action, up to and including termination, civil charges, criminal penalties, and/or other sanctions as determined by applicable law, MSU policies, HPC2 policies, law enforcement or other authorized State and Federal agencies.

***** N O T I C E *****

Last login: Mon Nov 11 10:32:46 2024 from 130.18.14.123

NOTICE:

Atlas is a cluster system running Rocky 9.x configured as follows.

Configuration and documentation is located at

https://www.hpc.msstate.edu/computing/atlas/

[hyeseon.kim@atlas-login-1 ~]$
```

Build local files

```
mkdir -p /90daydata/shared/$USER/
cd /90daydata/shared/$USER/
mkdir -p protein_workshop
cd protein_workshop
mkdir log
cp -r /90daydata/shared/protein_structure_workshop/AlphaFold .
```

DO NOT EDIT ANY FILES IN /90daydata/shared/protein_structure_workshop/

SET UP

Log into Atlas using a command line.

```
ssh user.name@atlas-login.hpc.msstate.edu
```

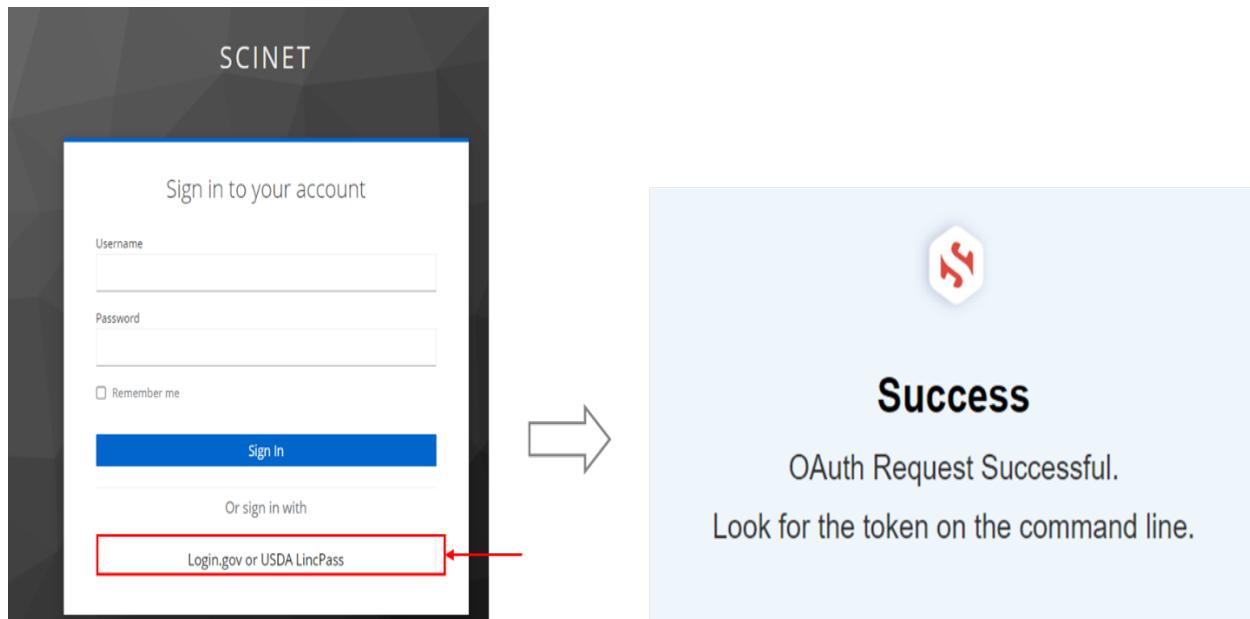
To log in to the cluster via SSH, you will first need to have SmallStepsCLI installed on a USDA controlled laptop. After installing it, you should be able to log in with LincPass or Login.gov.

```
Command Prompt - ssh hyes X + v
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HyeSeon.Kim>ssh hyeseon.kim@atlas-login.hpc.msstate.edu
✓Provisioner: keycloak (OIDC) [client: step-ca]
Your default web browser has been opened to visit:

https://verify.scinet.usda.gov/realms/scinet/protocol/openid-connect/auth?client_id=step-ca&code_challenge=IZQ05mGQ6eynZ
90u1dP21oDzYfdiF7nd2YvukRTIuWg&code_challenge_method=S256&nonce=71350382ede5b8b66267a2828c36e8c98594a70a85c87c3680db0c00
fe90436f&redirect_uri=http%3A%2F%2F127.0.0.1%3A10000&response_type=code&scope=openid+email&state=hOVNIq6SJKElnjbeTyCnxk
oCH9G2ftR
```

It will get to the below page. Once you login with LincPass, the message of “OAuth Request Successful” will be shown.



Request to get interactive session (2 cores, 32GB of memory, 4 hours, account information).

```
salloc -N 1 -n 2 --mem=32G -t 04:00:00 --account=scinet_workshop1
```

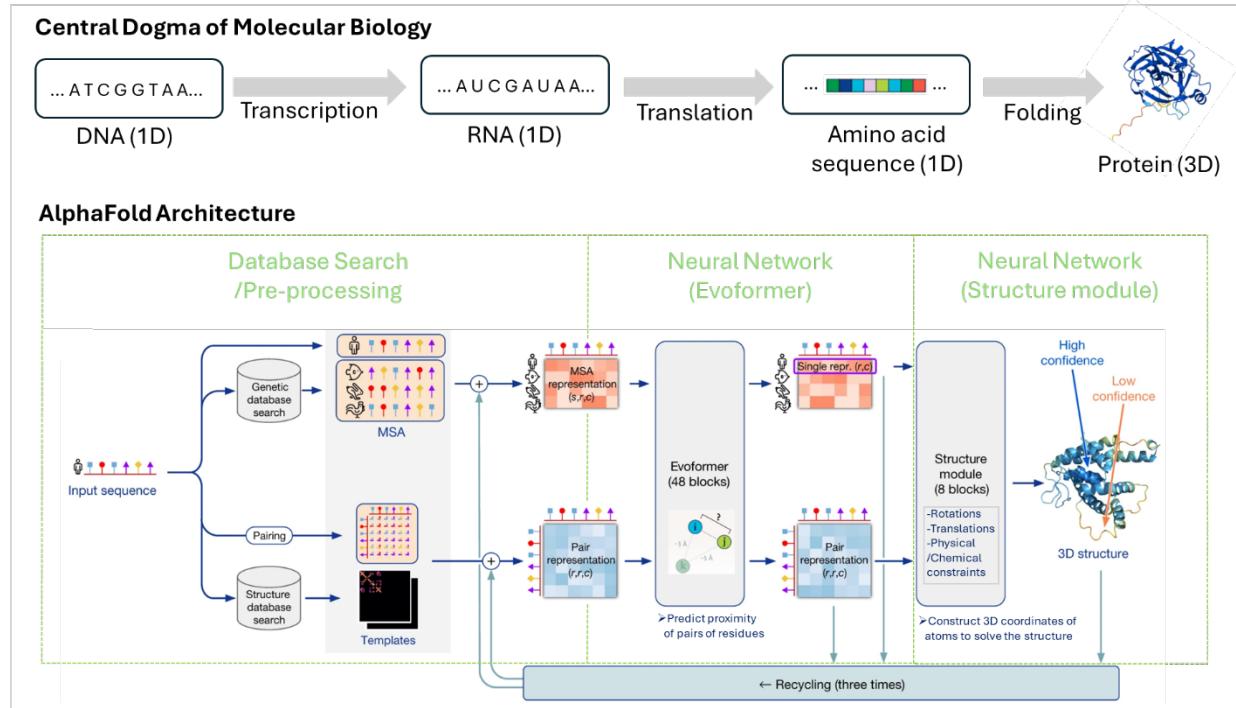
Build local files

```
mkdir -p /90daydata/shared/$USER/
cd /90daydata/shared/$USER/
mkdir -p protein_workshop
cd protein_workshop
mkdir log
cp -r /90daydata/shared/protein_structure_workshop/AlphaFold .
```

DO NOT EDIT ANY FILES IN /90daydata/shared/protein_structure_workshop/

AlphaFold 2 version

What is AlphaFold?



1. Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021). <https://doi.org/10.1038/s41586-021-03819-2>
2. Altman RB. A Holy Grail - The Prediction of Protein Structure. *N Engl J Med.* 2023 Oct 12;389(15):1431-1434. Epub 2023 Sep 21. PMID: 37732608. <https://www.nejm.org/doi/full/10.1056/NEJMcibr2307735>

AlphaFold takes as input the 1D sequence of a protein of unknown structure and a multiple sequence alignment (MSA) of many similar proteins found in different species and tissues. It creates a deep neural-network representation of the relationship between amino acids (e.g., the pair i and j) within the protein, as well as the relationship of these two positions across the evolutionary space represented in the MSA. These representations are linked to one another and “communicate” within the Evoformer, which uses knowledge of known pairs of 1D sequence and 3D structure to infer which amino acids are proximal. The Evoformer sends this information to the structure module, which models the position of the atoms within an amino acid in 3D and seeks configurations of the atoms that are compatible with the proximities provided by the Evoformer, as well as the physical and chemical constraints.

How to run AlphaFold 2 on a GPU node (Atlas HPC)?

I. FOLD MONOMER- SINGLE RUN: The fasta input file should have only one sequence. The code for monomer (--model_preset=monomer) should be included in the script. Note: Make sure sequences do not contain any special characters like “.”, “*”, “X”.

The folder of the “1_monomer_fasta” has a total of 10 monomer fasta input files.

Organism	Fasta input file (input sequences)	Length (amino acids)
Rice	Oryza_sativa_HMA_OsHIPP19	78
Maize Fungal pathogen	Ustilago_maydis_Cmu1_USTMA	97
Rice Fungal pathogen	Magnaporthe_oryzae_AvrPikD_C4B8B8	113
Wheat	Triticum_aestivum_A0A077RXP2	134
Maize Fungal pathogen	Fusarium_graminearum_TPP1_FGSG_11164	252
Maize Fungal pathogen	Fusarium_graminearum_NLS1_FGSG_04563	315
Maize Fungal pathogen	Fusarium_graminearum_TRI14_FGSG_03543	371
Maize Fungal pathogen	Fusarium_graminearum_I1RR40_FGSG_06549	474
Maize	Zea_mays_A0A1D6FS01	712
Rice	Oryza_sativa_Pik1	1142

Check local files that contain input files for monomer.

```
cd AlphaFold  
ls  
cd 1_monomer_fasta  
mkdir -p output
```

To run AlphaFold with a single run (one fasta file-1 protein sequence), create a batch script (.sub).

```
touch monomer_alphaFold_single_run_A100s.sub
```

Edit the below batch script for monomer AlphaFold2 with A100-MIG7 node (10 GB) for workshop.

```
#!/bin/bash  
#SBATCH --account=scinet_workshop1      #put HPC account name here, required on Atlas  
#SBATCH --job-name=alphaFold           #name of this job  
#SBATCH --partition=gpu-a100-mig7      #name of the partition (queue) you are submitting to  
#SBATCH --gres=gpu:a100_1g.10gb:1       #Specify your GPU partition to access reserved altas-a100-mig7  
#SBATCH -N1                           #number of nodes  
#SBATCH -n2                           #number of cores  
#SBATCH --ntasks=2  
#SBATCH --mem=32GB                     #Real memory (RAM) required (MB), 0 is the whole-node memory  
#SBATCH -t 04:00:00                     #time allocated for this job hours:mins:seconds  
#SBATCH --mail-user=YOUR.EMAIL@usda.gov    #CHANGE ME  
#SBATCH --mail-type=begin  
#SBATCH --mail-type=end  
#SBATCH --error=JobName.%J.err  
#SBATCH --output=JobName.%J.out  
  
date                                     #optional, prints out timestamp at the start of the job in stdout file  
  
module purge  
module load apptainer/1.3.3  
  
# Set environment variables  
export TF_FORCE_GPU_ALLOW_GROWTH=true  #Allows dynamic GPU memory allocation  
# Bind directories from the host to the container  
#export APPTAINER_BIND=/reference/data/alphaFold/2.3.0
```

```

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${PWD}"/Oryza_sativa_HMA_OsHIPP19.fasta
OUTPUT_DIR="${PWD}"/output
DATA_DIR=/alphafold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-
src=/ "$CONTAINER_PATH" \
    python /app/alphafold/run_alphafold.py \
        --fasta_paths="$FASTA_PATH" \
        --mgnify_database_path="$DATA_DIR"/mgnify/mgy_clusters_2022_05.fa \
        --uniref30_database_path="$DATA_DIR"/uniref30/UniRef30_2021_03 \
        --bfd_database_path="$DATA_DIR"/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
        --data_dir="$DATA_DIR" \
        --template_mmcif_dir="$DATA_DIR"/pdb_mmcif/mmcif_files \
        --obsolete_pdbs_path="$DATA_DIR"/pdb_mmcif/obsolete.dat \
        --output_dir="$OUTPUT_DIR" \
        --model_preset=monomer \
        --max_template_date=2030-01-01 \
        --uniref90_database_path="$DATA_DIR"/uniref90/uniref90.fasta \
        --pdb70_database_path="$DATA_DIR"/pdb70/pdb70 \
        --use_gpu_relax=True

date           #optional, prints out timestamp when the job ends
#End of file

```

Edit the below batch script for monomer AlphaFold2 with full A100 node.

```

#!/bin/bash

#SBATCH --account=scinet_workshop1      #put HPC account name here, required on Atlas
#SBATCH --job-name="alphafold"           #name of this job
#SBATCH -p gpu-a100                     #name of the partition (queue) you are submitting to
#SBATCH --gres=gpu:a100:1                #Specify your GPU partition to access reserved node
#SBATCH -N1                            #number of nodes
#SBATCH -n8                            #number of cores
#SBATCH --ntasks=16
#SBATCH --mem=256GB                      #Real memory (RAM) required (MB), 0 is the whole-node memory
#SBATCH -t 24:00:00                      #time allocated for this job hours:mins:seconds
#SBATCH --mail-user=YOUR EMAIL@usda.gov   #CHANGE ME
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --error=JobName.%J.err
#SBATCH --output=JobName.%J.out

date           #optional, prints out timestamp at the start of the job in stdout file

module purge
module load apptainer/1.3.3

# Set environment variables
export TF_FORCE_GPU_ALLOW_GROWTH=true  #Allows dynamic GPU memory allocation
# Bind directories from the host to the container
#export APPTAINER_BIND=/reference/data/alphafold/2.3.0

```

```

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${PWD}"/Oryza_sativa_HM_OsHIPP19.fasta
OUTPUT_DIR="${PWD}"/output
DATA_DIR=/alphafold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-
src=/ "$CONTAINER_PATH" \
    python /app/alphafold/run_alphafold.py \
    --fasta_paths="$FASTA_PATH" \
    --mgnify_database_path="$DATA_DIR/mgnify/mgy_clusters_2022_05.fa" \
    --uniref30_database_path="$DATA_DIR/uniref30/UniRef30_2021_03" \
    --bfd_database_path="$DATA_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt" \
    --data_dir="$DATA_DIR" \
    --template_mmcif_dir="$DATA_DIR/pdb_mmcif/mmcif_files" \
    --obsolete_pdbs_path="$DATA_DIR/pdb_mmcif/obsolete.dat" \
    --output_dir="$OUTPUT_DIR" \
    --model_preset=monomer \
    --max_template_date=2030-01-01 \
    --uniref90_database_path="$DATA_DIR/uniref90/uniref90.fasta" \
    --pdb70_database_path="$DATA_DIR/pdb70/pdb70" \
    --use_gpu_relax=True

date                      #optional, prints out timestamp when the job ends
#End of file

```

To open the batch script .sub file that you create, type the below command.

```
nano monomer_alphaFold_single_run_A100s.sub
```

The GNU nano text edition will show up,



Copy the monomer AlphaFold2 batch script in the nano shell. After copying the code in the batch file, and press Ctrl +X (^X) to close the GNU nano shell.

```

GNU nano 5.6.1                               monomer_alphafold_single_run_A100s.sub
# Set environment variables
export TF_FORCE_GPU_ALLOW_GROWTH=true #Allows dynamic GPU memory allocation
# Bind directories from the host to the container
#export APPTAINER_BIND=/reference/data/alphafold/2.3.0

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${PWD}"/Oryza_sativa_HMA_OsHIPP19.fasta
OUTPUT_DIR="${PWD}"/output
DATA_DIR=/alphafoold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-src=/ "$CONTAINER_PATH" \
python /app/alphafold/run_alfafold.py \
--fasta_paths="$FASTA_PATH" \
--mgnify_database_path="$DATA_DIR/mgnify/mgy_clusters_2022_05.fa" \
--uniref30_database_path="$DATA_DIR/uniref30/uniref30_2021_03" \
--bfd_database_path="$DATA_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt" \
--data_dir="$DATA_DIR" \
--template_mmcif_dir="$DATA_DIR/pdb_mmcif/mmcif_files" \
--obsolete_pdbs_path="$DATA_DIR/pdb_mmcif/obsolete.dat" \
--output_dir="$OUTPUT_DIR" \
--model_preset=monomer \
--max_template_date=2030-01-01 \
--uniref90_database_path="$DATA_DIR/uniref90/uniref90.fasta" \
--pdb70_database_path="$DATA_DIR/pdb70/pdb70" \
--use_gpu_relax=True

date                                         #optional, prints out timestamp when the job ends
#End of file

```

The status bar at the bottom of the terminal window shows the following keyboard shortcuts:

- Help**
- Write Out**
- Where Is**
- Cut**
- Paste**
- Execute**
- Location**
- Undo**
- Set Mark**
- Read File**
- Replace**
- Justify**
- Go To Line**
- Redo**
- Copy**

A red arrow points to the **Exit** key in the status bar.

And then, press Y (Yes) to save your changes.

```

GNU nano 5.6.1                               monomer_alphafold_single_run_A100s.sub
# Set environment variables
export TF_FORCE_GPU_ALLOW_GROWTH=true #Allows dynamic GPU memory allocation
# Bind directories from the host to the container
#export APPTAINER_BIND=/reference/data/alphafold/2.3.0

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${PWD}"/Oryza_sativa_HMA_OsHIPP19.fasta
OUTPUT_DIR="${PWD}"/output
DATA_DIR=/alphafoold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-src=/ "$CONTAINER_PATH" \
python /app/alphafold/run_alfafold.py \
--fasta_paths="$FASTA_PATH" \
--mgnify_database_path="$DATA_DIR/mgnify/mgy_clusters_2022_05.fa" \
--uniref30_database_path="$DATA_DIR/uniref30/uniref30_2021_03" \
--bfd_database_path="$DATA_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt" \
--data_dir="$DATA_DIR" \
--template_mmcif_dir="$DATA_DIR/pdb_mmcif/mmcif_files" \
--obsolete_pdbs_path="$DATA_DIR/pdb_mmcif/obsolete.dat" \
--output_dir="$OUTPUT_DIR" \
--model_preset=monomer \
--max_template_date=2030-01-01 \
--uniref90_database_path="$DATA_DIR/uniref90/uniref90.fasta" \
--pdb70_database_path="$DATA_DIR/pdb70/pdb70" \
--use_gpu_relax=True

date                                         #optional, prints out timestamp when the job ends
#End of file

Save modified buffer?
Y Yes ←
N No          ^C Cancel

```

To submit a SLURM batch script file.

```
sbatch monomer_alphafold_single_run_A100s.sub
```

To check the status of a batch run and retrieve information about a submitted SLURM batch job.

```
squeue -u $USER
```

To cancel the job.

```
scancel ID
```

II. FOLD MONOMER- JOB ARRAY RUN: In the monomer folder, we have a 10 fasta files. With Job arrays, we should be able to run 10 monomer folding jobs in parallel at the same time. The configuration file (config.txt) that contain the JobID of the array and file name for the individual fasta files is required.

Check local files that contains input files and Config.txt file.

```
cd 1_monomer_fasta  
mkdir -p fasta_files  
cp -f *.fasta ./fasta_files  
mkdir -p output_array
```

Config. txt file look like this below as,

ArrayID	file
1	Oryza_sativa_HMA_OsHIPP19.fasta
2	Ustilago_maydis_Cmu1_USTMA.fasta
3	Magnaporthe_oryzae_AvrPikD_C4B8B8.fasta
4	Triticum_aestivum_A0A077RXP2.fasta
5	Fusarium_graminearum_TPP1_FGSG_11164.fasta
6	Fusarium_graminearum_NLS1_FGSG_04563.fasta
7	Fusarium_graminearum_TRI14_FGSG_03543.fasta
8	Fusarium_graminearum_I1RR40_FGSG_06549.fasta
9	Zea_mays_A0A1D6FS01.fasta
10	Oryza_sativa_Pik1.fasta

To run AlphaFold with SLURM job array, create batch script (.sub file).

```
touch monomer_alphafold_array_run_A100s.sub
```

Edit the below batch script for monomer AlphaFold2 with array run (email, input file, model flag).

```
#!/bin/bash  
#SBATCH --account=scinet_workshop1 #put HPC account name here, required on Atlas  
#SBATCH --job-name="alphafold"      #name of this job  
#SBATCH -p gpu-a100                #name of the partition (queue) you are submitting to
```

```

#SBATCH --gres=gpu:a100:1          #Specify your GPU partition to access reserved altas-0245
#SBATCH -N1                         #number of nodes
#SBATCH -n8                         #number of cores
#SBATCH --ntasks=16
#SBATCH --mem=256GB                 # Real memory (RAM) required (MB), 0 is the whole-node memory
#SBATCH -t 24:00:00                  #time allocated for this job hours:mins:seconds
#SBATCH --mail-user= YOUR.EMAIL@usda.gov    #CHANGE ME
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --error=JobName.%A_%a.err
#SBATCH --output=JobName.%A_%a.out
#SBATCH --array=1-10

date                                #optional, prints out timestamp at the start of the job in stdout file

module purge
module load apptainer/1.3.3

# Set config file that holds file names for the array
config=./Config.txt

# Set environment variables
export TF_FORCE_GPU_ALLOW_GROWTH=true  # Allows dynamic GPU memory allocation
# Bind directories from the host to the container
#export APPTAINER_BIND=/reference/data/alphafold/2.3.0

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${SLURM_SUBMIT_DIR}"/fasta_files
FASTA=$(awk -v ArrayTaskID="$SLURM_ARRAY_TASK_ID" '$1==ArrayTaskID {print $2}' "$config")
OUTPUT_DIR="${SLURM_SUBMIT_DIR}"/output_array
DATA_DIR=/alphafold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-
src=/ "$CONTAINER_PATH" \
    python /app/alphafold/run_alphafold.py \
    --fasta_paths="$FASTA_PATH"/"$FASTA" \
    --mgnify_database_path="$DATA_DIR"/mgnify/mgy_clusters_2022_05.fa \
    --uniref30_database_path="$DATA_DIR"/uniref30/UniRef30_2021_03 \
    --bfd_database_path="$DATA_DIR"/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
    --data_dir="$DATA_DIR" \
    --template_mmcif_dir="$DATA_DIR"/pdb_mmcif/mmcif_files \
    --obsolete_pdbs_path="$DATA_DIR"/pdb_mmcif/obsolete.dat \
    --output_dir="$OUTPUT_DIR" \
    --model_preset=monomer \
    --max_template_date=2030-01-01 \
    --uniref90_database_path="$DATA_DIR"/uniref90/uniref90.fasta \
    --pdb70_database_path="$DATA_DIR"/pdb70/pdb70 \
    --use_gpu_relax=True

date                                #optional, prints out timestamp when the job ends
#End of file

```

To open the batch script .sub file that you create, type the below command.

```
nano monomer_alphaFold_array_run_A100s.sub
```

⇒ Copy the above job array script in the nano shell. After copying the code in the batch file and press Ctrl +X (^X) to close the GNU nano shell. And then, press Y (Yes) to save your changes.

To submit a SLURM batch script file

```
sbatch monomer_alphaFold_array_run_A100s.sub
```

To check the status of a batch run and retrieve information about a submitted SLURM batch job

```
squeue -u $USER
```

III. FOLD MULTIMER- SINGLE RUN: The input fasta file should have multiple sequences, here is an example of two protein sequences from each of the plant and pathogen (e.g. Sequence 1 from fungal pathogens (*Magnaporthe oryzae*, *Fusarium graminearum*) and Sequence 2 from crop plants (rice, maize and/or wheat). The code for multimer (--model_preset =multimer) should be included in the batch script as a model flag.

Check local files that contain input files for multimer.

```
cd ..  
cd 2_multimer.fasta  
mkdir -p output_multimer
```

To run AlphaFold multimer (one fasta file-2 protein sequences), create a batch script (.sub file).

```
touch multimer_alphaFold_single_run_A100s.sub
```

Edit the below SLURM batch script for multimer AlphaFold2 (email, input file, model flag).

```
#!/bin/bash  
#SBATCH --account=scinet_workshop1 #put HPC account name here, required on Atlas  
#SBATCH --job-name="alphaFold" #name of this job  
#SBATCH -p gpu-a100 #name of the partition (queue) you are submitting to  
#SBATCH --gres=gpu:a100:1 #Specify your GPU partition to access reserved altas-0245  
#SBATCH -N1 #number of nodes  
#SBATCH -n8 #number of cores  
#SBATCH --ntasks=16  
#SBATCH --mem=256GB #Real memory (RAM) required (MB), 0 is the whole-node memory  
#SBATCH -t 24:00:00 #time allocated for this job hours:mins:seconds  
#SBATCH --mail-user= YOUR.EMAIL@usda.gov #CHANGE ME  
#SBATCH --mail-type=begin  
#SBATCH --mail-type=end  
#SBATCH --error=JobName.%J.err  
#SBATCH --output=JobName.%J.out  
  
date #optional, prints out timestamp at the start of the job in stdout file  
  
module purge  
module load apptainer/1.3.3
```

```

# Set environment variables
export TF_FORCE_GPU_ALLOW_GROWTH=true # Allows dynamic GPU memory allocation
# Bind directories from the host to the container
#export APPTAINER_BIND=/reference/data/alphafold/2.3.0

# Path to AlphaFold container
CONTAINER_PATH=/reference/containers/alphafold/2.3.2/alphafold-2.3.2.sif

# Paths to input and output directories (inside the container)
FASTA_PATH="${PWD}"/multimer_M_oryze_O_sativa.fasta
OUTPUT_DIR="${PWD}"/output_multimer
DATA_DIR=/alphafold-data

# Run AlphaFold
apptainer exec --nv -B /local/reference/data/alphafold/alphafold_2.3.0.squashfs:/alphafold-data:image-
src=/ "$CONTAINER_PATH" \
    python /app/alphafold/run_alphafold.py \
    --fasta_paths="$FASTA_PATH" \
    --mgnify_database_path="$DATA_DIR"/mgnify/mgy_clusters_2022_05.fa \
    --uniref30_database_path="$DATA_DIR"/uniref30/UniRef30_2021_03 \
    --bfd_database_path="$DATA_DIR"/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
    --data_dir="$DATA_DIR" \
    --template_mmcif_dir="$DATA_DIR"/pdb_mmcif/mmcif_files \
    --obsolete_pdbs_path="$DATA_DIR"/pdb_mmcif/obsolete.dat \
    --output_dir="$OUTPUT_DIR" \
    --model_preset=multimer \
    --max_template_date=2030-01-01 \
    --uniref90_database_path="$DATA_DIR"/uniref90/uniref90.fasta \
    --uniprot_database_path="$DATA_DIR"/uniprot/uniprot.fasta \
    --pdb_seqres_database_path="$DATA_DIR"/pdb_seqres/pdb_seqres.txt \
    --use_gpu_relax=True

date #optional, prints out timestamp when the job ends
#End of file

```

To open the batch script .sub file that you create, type the below command.

```
nano multimer_alphaFold_single_run_A100s.sub
```

⇒ Copy the above job array script in the nano shell. After copying the code in the batch file and press Ctrl +X (^X) to close the GNU nano shell. And then, press Y (Yes) to save your changes.

To submit a SLURM batch script file.

```
sbatch multimer_alphaFold_single_run_A100s.sub
```

To check the status of a batch run and retrieve information about a submitted SLURM batch job.

```
squeue -u $USER
```

How to visualize the alphaFold results sequences?

- Navigate to the output directory in Atlas using Globus.
- Transfer the PDB(s) or CIF(s) files of interest locally.
- Drag the files into Mol* (<https://molstar.org/viewer/>).

<https://molstar.org/>

The image shows two screenshots of the Mol* web-based toolkit. On the left, the homepage features the Mol* logo (Mol* in yellow and blue) and a brief description: "Mol* (/molstar/) is a modern web-based open-source toolkit for visualisation and analysis of large-scale molecular data". Below this is a screenshot of the Mol* Viewer interface, which displays a complex protein structure. A red arrow points to the "Open Mol* Viewer" button. To the right, a larger screenshot shows the Mol* Viewer interface with several protein structures visualized as green and orange ribbon models. A white arrow points from the text "Drag the files .pdb .cif" towards the viewer interface.

AlphaFold 3 version (<https://alphafoldserver.com/>)

AlphaFold3 code is officially released by Google DeepMind: <https://github.com/google-deepmind/alphafold3> and the SCINet team will work on installing it on both Ceres and Atlas.

The image shows the AlphaFold Server landing page. The background features a close-up of a protein structure with green and cyan ribbon models. Overlaid on the image is the text "AlphaFold Server" and "Powered by AlphaFold 3". At the bottom, it says "AlphaFold 3 model is a Google DeepMind and Isomorphous Labs collaboration". A red arrow points to the "Continue with Google" button, which is located at the bottom center of the page.

⇒ Click to “Continue with Google”: you need to have a Google account to use the server.

Using AlphaFold Server:

Last Modified: May 8, 2024 | [Download PDF](#)

By using AlphaFold Server, you acknowledge you have read the [Google Privacy Policy](#) and this AlphaFold Server Privacy Notice:

- In accordance with the "Retaining Your Information" section of the [Google Privacy Policy](#), we may retain time-stamped records of the sequences you input into AlphaFold Server and the output it generates ("User Record") for an extended period of time to monitor compliance with, enforce and investigate potential violations of the Terms.
- Notwithstanding the "Exporting, removing & deleting your information" section of the [Google Privacy Policy](#), **deleting your Google account or amending or deleting your AlphaFold Server history will not delete your User Record**. We take steps to protect your privacy, including storing your User Record in a separate database from your [Google Account](#) data. If you would like to review, update or export your User Record, please notify us via the "submit feedback" feature in AlphaFold Server.

[Cancel](#) [Continue](#)

- How many jobs can I run on AlphaFold online server?** 20 jobs per day.
- What is the maximum job size allowed?** The total size of the job is limited by the number of 'tokens' in the structure - the limit is 5,000 tokens (e.g. Proteins: 1 token per standard amino acid residue) Note that each protein chain and nucleotide chain must contain at least 4 amino acids or nucleotides, respectively.

Remaining jobs: 20

AlphaFold Server allows you to model a structure consisting of many biological molecules [Learn more ^](#)

- Remaining jobs refresh each day
- Jobs can be up to 5,000 tokens - see more details on token calculation, accepted formats, seed selection and other features in our [FAQ](#)
- Use the entity bar to chemically modify proteins and nucleic acids
- Get in touch with the AlphaFold team if you have any questions

Explore these examples of structures to see it in action – try them out without using your quota until you begin editing!

Protein-RNA-Ion: PDB 8AW3 | Protein-Glycan-Ion: PDB 7BBV | Protein-DNA-Ion: PDB 7RCE

Ok, got it

Molecule type: Protein Copies: 1 Input >Paste sequence or fasta Input X

+ Add entity

Upload JSON Clear

Save job

Continue and preview job

Completed Saved draft In progress Examples Failed

No jobs found with filters specified

Items per page: 10 0 of 0 < > >>

I. FOLD MONOMER-To run monomer, copy a single protein and put it in input tab.

- Fusarium_graminearum_TPP1_FGSG_11164.fasta (252 amino acids)

```
> Fusarium_graminearum_TPP1_FGSG_11164 (Fungal effector protein)
MVKITSLVALAAPLVAAPNPQNNSPQIVGGTSASAGEFPFIVSITNNGGPWCGGTLLNANTVMTASHCVQGRSASAFAIRVGNSNSRTSGGVTSRV
SSIRMHPSFGSTLNNDVALKLSTSIPAGGSIAYGRLATSGSDPAAGSSLTVAGWGDTSEGGGVSPVNLLKVTVPVVS RATCRSQYGTSAITDNM
FCAGVTGGKDACQGDGGPIVDSSKTVVGIVSWGDGCARPNAAGVYARVGLRSWIDSNA
```

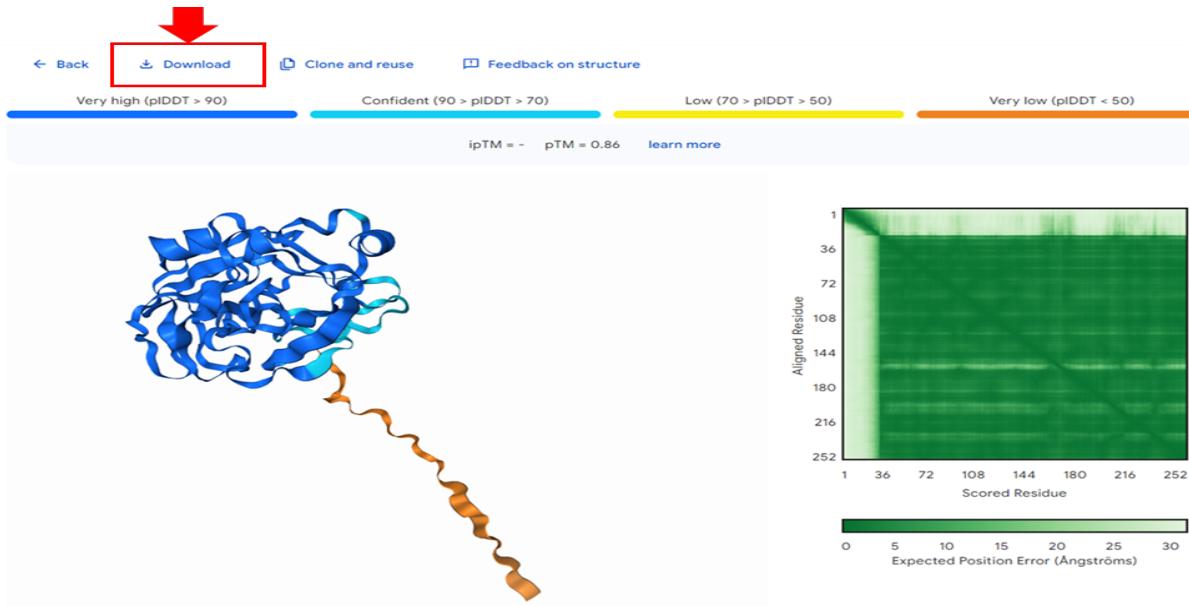
⇒ After that, press the button for “Continue and preview job” and then, also press the button for “Confirm and submit job”.

The screenshot shows the AlphaFold Server interface. On the left, a protein sequence is entered with 'Copies' set to 1. The sequence starts at residue 10 and ends at 252. On the right, a job summary is shown with 'Job name' set to '2024-11-02_15:09'. The 'Type' is 'Protein' with 'Copies' 1 and 'Sequence' 'MVKITSLVALAP(AA...PQNSPQIVGG...' (length 252). Below the sequence is a note: 'We've generated a job name for you, please edit'. A 'Seed' dropdown is set to 'Auto'. At the bottom, there are buttons for 'Save job', 'Continue and preview job' (highlighted with a red arrow), 'Go back and edit this job', and 'Confirm and submit job' (highlighted with a red arrow).

⇒ It takes a minute to complete the analysis

The screenshot shows the AlphaFold Server dashboard. It lists a single job entry: '2024-11-02_15:09' (highlighted with a red arrow) which was 'Modified' at '2024-11-02 15:10'. The status bar at the top shows 'Completed', 'Saved draft', 'In progress', 'Examples', and 'Failed'.

⇒ Once you click the log file then, you should be able to see the below results and you can also download the whole results (.json and .cif files) by clicking “Download” button.



Information

Type	Copies	Sequence
Protein	1	MVKITSLVALAP(AA...PQNSPQIVGG...
		(length 252)

Seed: 1951078022

II. FOLD MULTIMER-To run multimer, copy multiple protein sequences of interest and put them in input tab (for examples, sequence 1 from fungal pathogen, *Fusarium graminearum* and sequence 2 from maize).

- *Fusarium graminearum_I1RR40_FGSG_06549.* fasta (474 amino acids)
- *Zea_mays_A0A1D6FS01.* fasta (712 amino acids)

>***Fusarium graminearum_I1RR40_FGSG_06549 (Fungal effector protein)***

MKNSCSITLGSLLLLHAGAVLAGPVYGVDDILSPRHSKLRKRAECGPFIGSCNPGCCSESFGCTTGDFCGGSACQLEYSIDSCDTFFGPGSSTESI
SRPKIGSVPGSIKTCTPVGIALTFDDGPLTYTNLDLLDSKNVKATFFVAGNNRAKGHMDSSNPWPAMRRMHTAGHHIASHTWTHR
LNTVNSTIRTSEMIYNEMAFRNLFGWIPTYMRPPYLECNAGSGCLAEMSRLGYHVVVDQNVDTKDYEENDSPQLIQNSKNRYSAGVSTNSASNQYI
VLAHDVHDQTvhnlTSYMDTARSRGYRLTVGECLGDPRTNWRYTASRDRDVTSSTAATQTVPPPTKVSTTKATATGGLVISPNCRCGG
DTGYTCQGSAGFSCCSFYGYCGSSASYCGTGCDAFGETPPSGGGVHDTTNGVCGSEVNASCRNYSKTCQSQYGYCGSSATHCGTGCQKGFCT
CT

>***Zea_mays_A0A1D6FS01_Zm00001d010564 (Maize protein)***

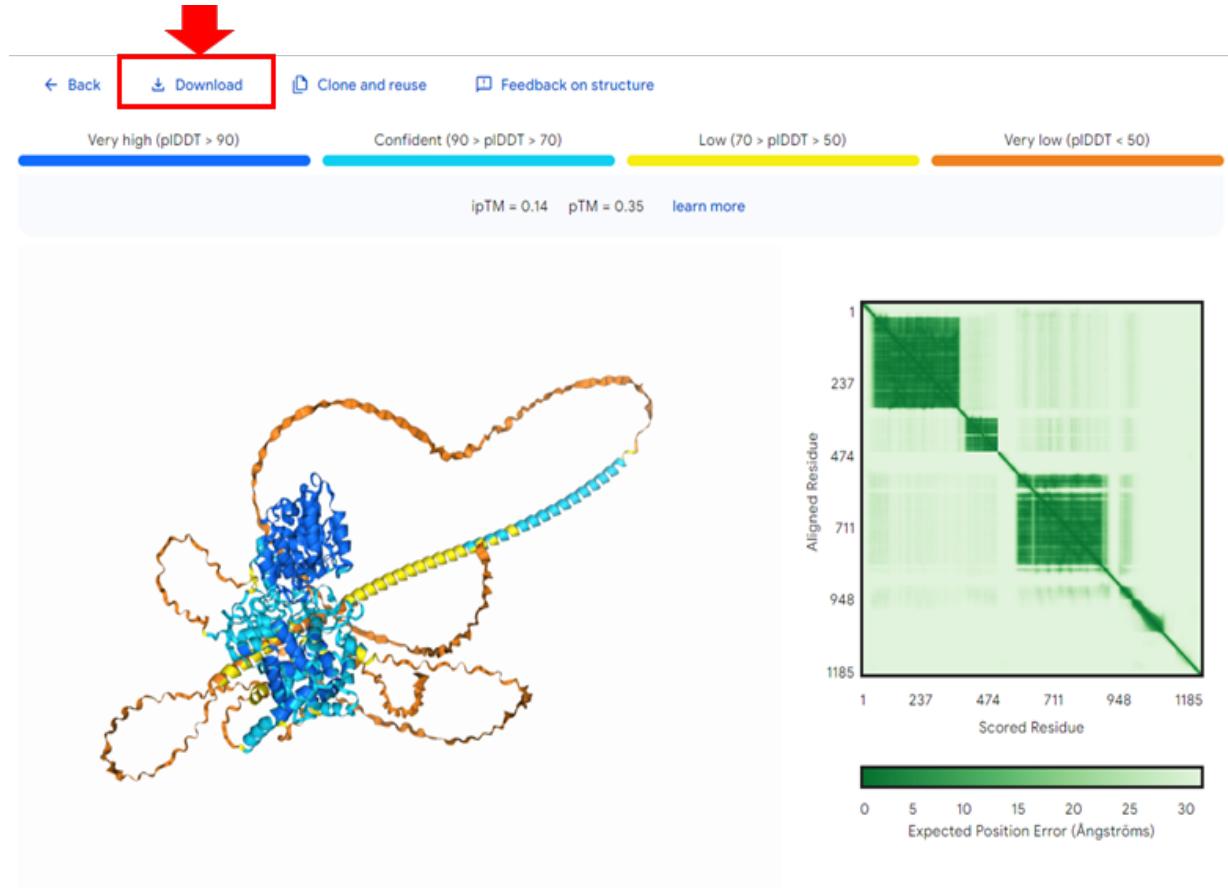
MMSLRIEKKKSSASKQQAKDVIHPVKVEEGKLSEDSDDEFYDVKVDPSEQVQPSDTGNADVGRSRQEEENYISKEEELCVHGLPMALRGE
WQAFVGTGARRVEGYYDNLAAEGELDKRSDSRTSEGVHEKWIGQIEKDLPRTPGHPALDEGRNALRRLIAYAKHNPSVGYCQAMNFA
GLLMLPMEENAFWTLVGMDDYFDGYFSEEMIESQVDQLVLEELVREKFPKLANHLDYGLQVAWVTGPWFLSIFTNVLPWESVLRVWDVL
LFDGNRVMFLRTALALLEFGPALVTTKADGDAVTLIQLSLAGSTFDSSQLVLTARMGYQSVNETILQELSNKHRPPVISAMEERAKGLGVWT
TNGLASKLYNFKRDPEPLVSLSDSTDQLSDVGDDTNQESDLMNMDEYGGIVNSEIDSPLDPKDQVAWLKELCRLIEERRSAVLRADELET
ALMEMVKQDNRRQLSAKVEQFEQEISELRQALSKQEQEAMQFQVLMRVEQELKIAEEARISAEQDAAAQRYAANVQEKYEEAMASLAQM
NRAVMAETMLEATLQYQSSQQKAMSPCPSPRPSMLDAPSQSSQEFQPRRKNLGPFLSWRDKNKEKPNNADDSTNTKSTNNDEM
ETSNTNDEKHRETLDLNSEQRAESPKADEVKMRATEPKDNDLPGVQLVTDDLNHHEQMKEIKL

The screenshot shows the Fold Multimer analysis interface. It displays two protein sequences in a grid format. The top sequence is *Fusarium graminearum_I1RR40_FGSG_06549* and the bottom sequence is *Zea_mays_A0A1D6FS01_Zm00001d010564*. Both sequences are 474 amino acids long. The interface includes dropdown menus for 'Molecule type' (set to Protein) and 'Copies' (set to 1). At the bottom, there is a 'Continue and preview job' button with a red arrow pointing to it, and a 'Save job' button.

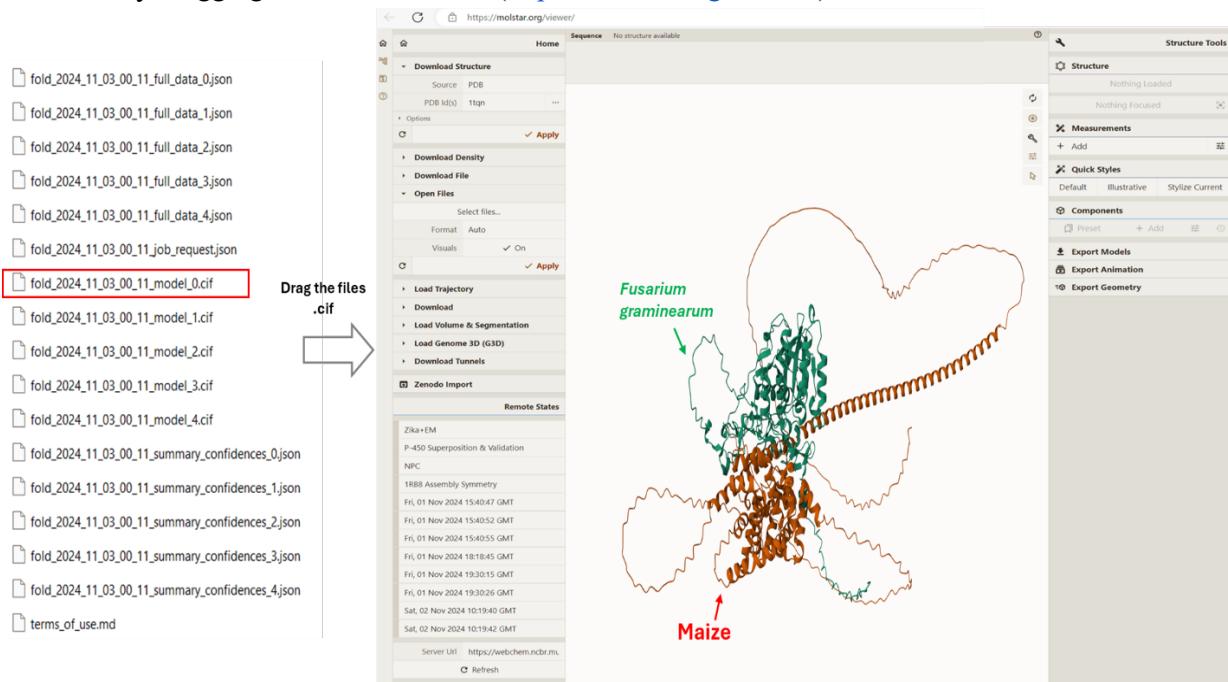
⇒ It takes 7 minutes to complete the analysis

<input type="checkbox"/>	Name	Modified
<input checked="" type="checkbox"/>	2024-11-03_00:11	2024-11-03 00:18

⇒ Download the files by clicking the “Download button”



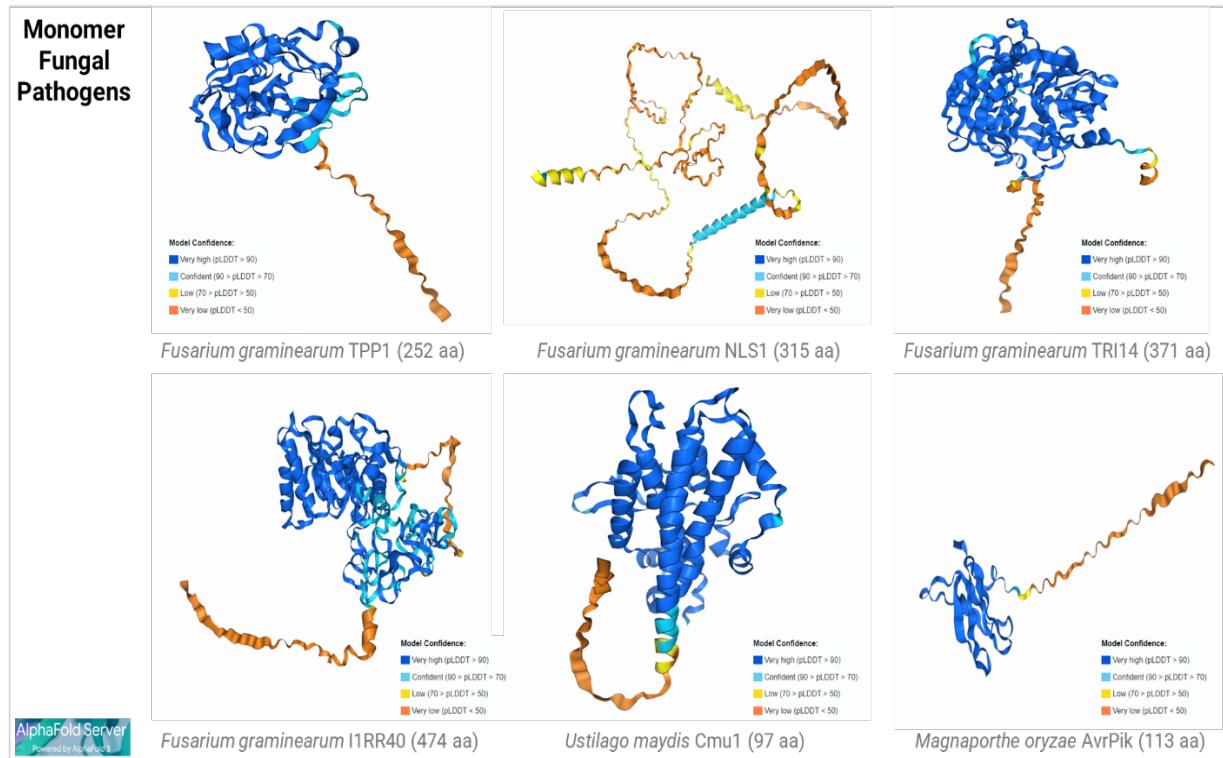
The downloaded results (.cif file) from Alphafold 3 server also can be viewed from Molstar software by dragging the files into Mol* (<https://molstar.org/viewer/>).



Note: The deepmind/Alphafold 2 is also available <https://build.nvidia.com/deepmind/alphafold2>

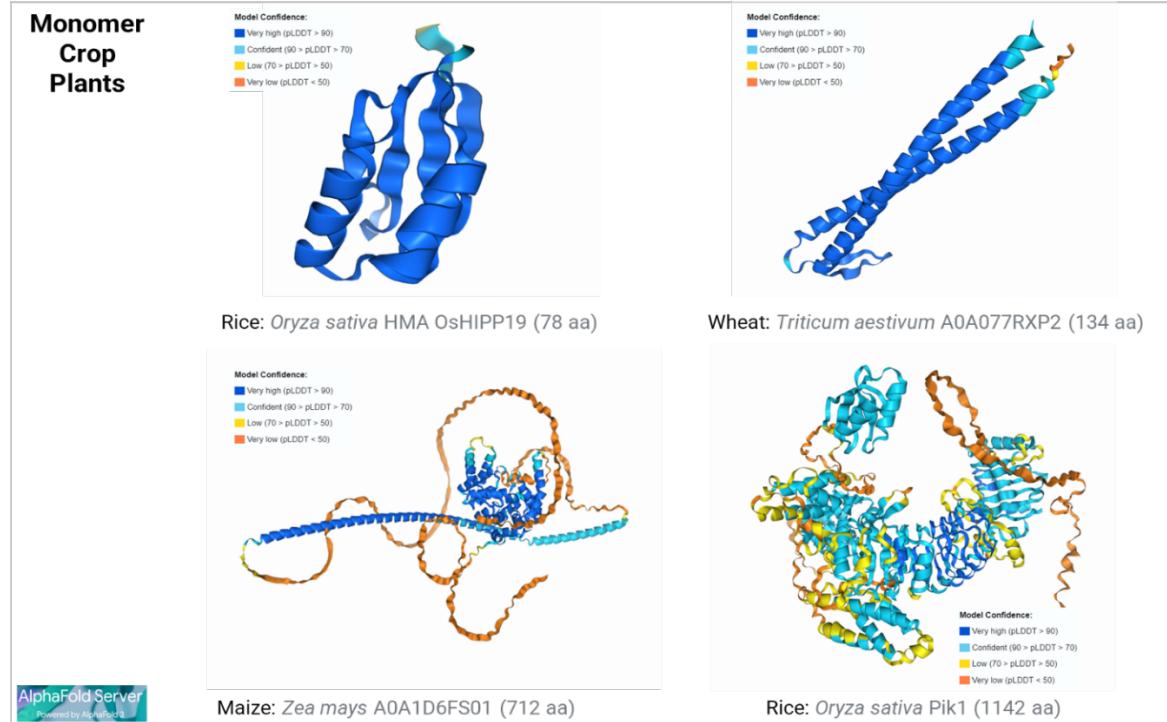
The NVIDIA AlphaFold2 is a graphical user interface that offer a simple and easy to deploy route for self-hosted AI application. Predict protein structure from amino acid sequences but also predict multiple sequence alignment for a given sequences against a series of protein sequence databases and provides accurate model behind a consistent API.

AlphaFold Results



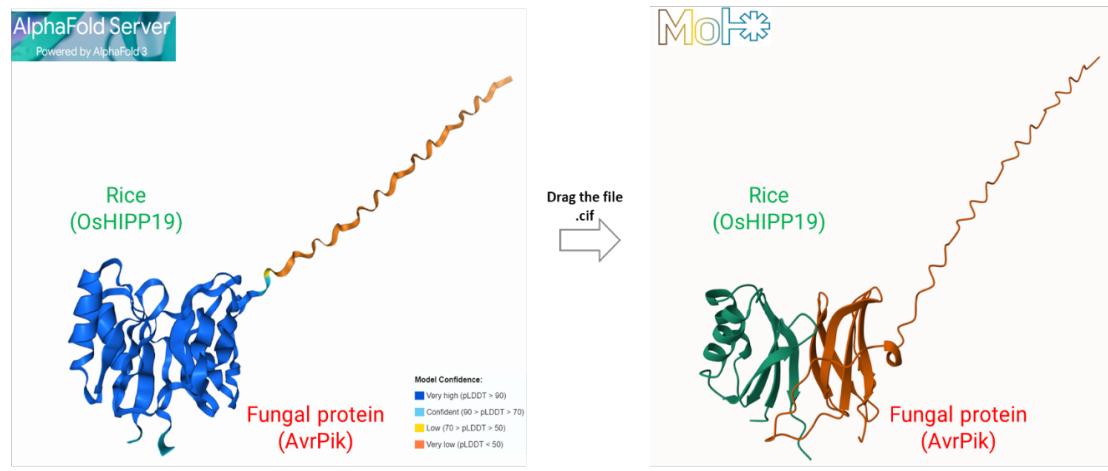
3 publications by utilizing the AlphaFold/FoldSeek programs available on SCINet

- Fusarium_graminearum_TPP1_FGSG_11164 (252 aa) <https://doi.org/10.1101/2024.08.30.610543>
- Fusarium_graminearum_NLS1_FGSG_04563 (315 aa) <https://doi.org/10.1094/MPMI-12-22-0254-R>
- Fusarium_graminearum_TRI14_FGSG_03543(317aa) <https://doi.org/10.3390/applicmicrobiol4020058>



Multimer 1 : Fungal effector AVR-Pik binds the HMA domain of the rice protein OsHIPP19

- Fungal protein: *Magnaporthe oryzae* AvrPik (113 aa)
- Rice protein: *Oryza sativa* HMA domain OsHIPP19 (78 aa)

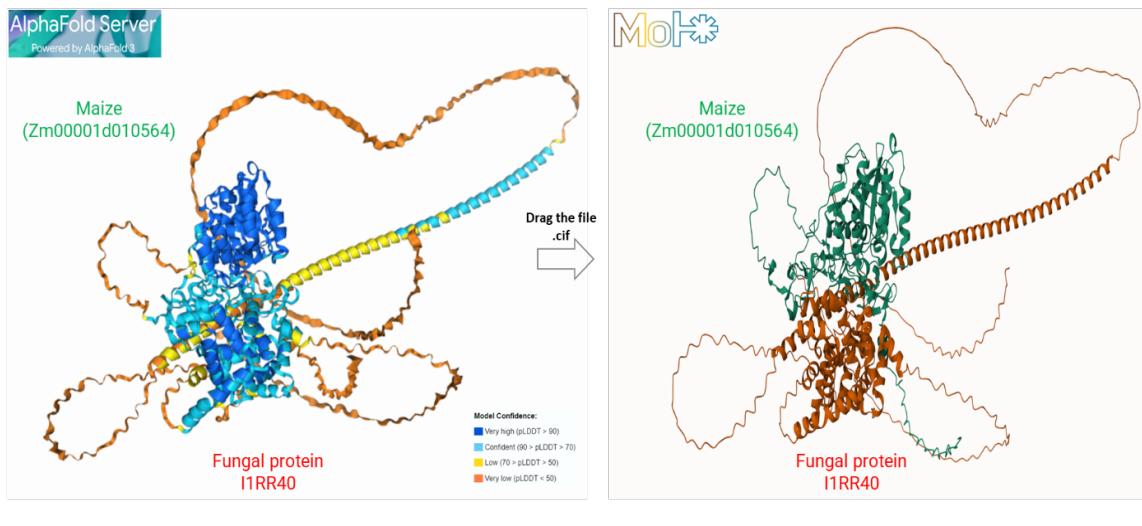


1 publication by utilizing the ESMFold/RF diffusion/FoldSeek programs available on SCINet

- Protein-protein interactions between rice plant-derived resistance (R) gene and pathogen-derived avirulence (Avr) effector of *M. oryzae* <https://doi.org/10.1101/2024.09.17.613523>

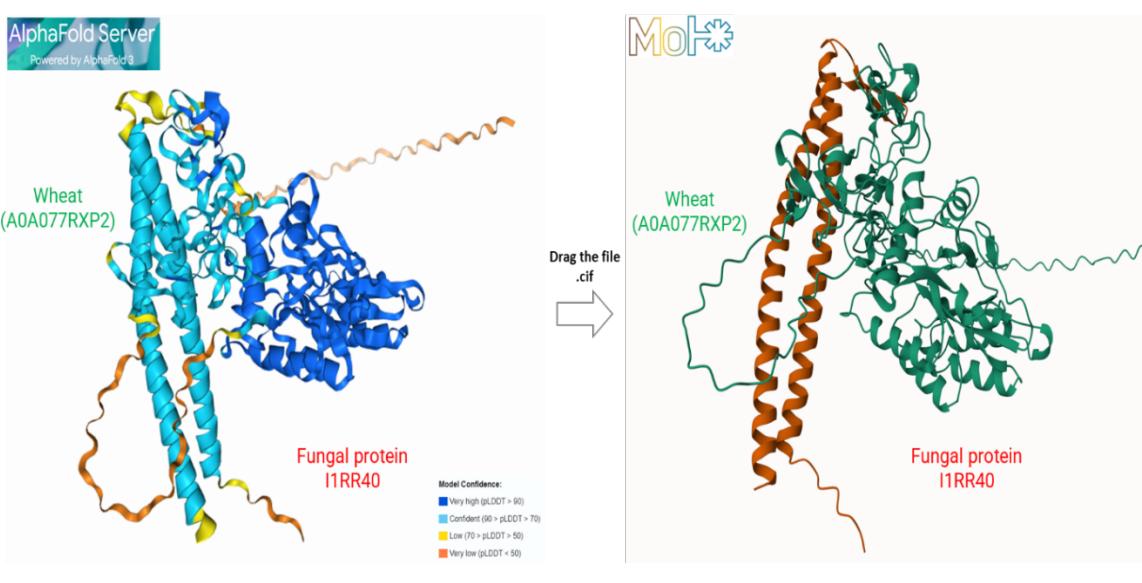
Multimer 2 : Fungal effector I1RR40 binds the Ypt/Rab-GAP domain of the maize protein Zm00001d010564

- Fungal protein: *Fusarium graminearum* I1RR40 FGSG_06549 (474 aa)
- Maize protein: *Zea mays* A0A1D6FS01 (Ypt/Rab-GAP domain of Zm00001d010564) (712 aa)



Multimer 3 : Fungal effector I1RR40 binds the prefoldin subunit 6 of the wheat protein A0A077RXP2

- Fungal protein: *Fusarium graminearum* I1RR40 FGSG_06549 (474 aa)
- Wheat protein: *Triticum aestivum* A0A077RXP2 (prefoldin subunit 6 of A0A077RXP2) (134 aa)



2 publications by utilizing the AlphaFold/ESMFold/ESM-Variant programs available on SCINet

- Maize PanEffect <https://doi.org/10.1093/bioinformatics/btae073> and databases <https://www.maizegdb.org/effect/maize/>
- Fusarium Protein ToolKit (Fusarium PanEffect) <https://doi.org/10.1186/s12866-024-03480-5> and databases <https://fusarium.maizegdb.org/>

SET UP

Estimated time: (< 5 minutes)

Build local files

```
cp -r /90daydata/shared/protein_structure_workshop/ESMFOLD .
cd ESMFOLD
```

Task #1: Use ESMFold to predict protein structures.



Selfed ear segregating for multiple aleurone and endosperm genes. From the mutants of maize collection at MaizeGDB, originally collected by Dr. Gerald Neuffer.

We would like to predict the protein structures for five protein-coding genes related to kernel phenotypes: *sul1* (sugary/normal), *y1* (white/yellow endosperm), *sh1* (shrunken/normal kernel), and *wx1* (waxy/normal), and *gl1* (glossy1).

Run ESMFold on a set of five maize proteins (5-10 minutes):

Parameters:

- Input FASTA file: ./fasta/maize5.fasta
- Output directory: ./pdb/
- Chunk size: 32

```
sbatch bulk_esmfold.sh ./fasta/maize5.fasta ./pdb/ 32
```

```

#!/bin/bash -l

#SBATCH -A scinet_workshop1
#SBATCH --partition=gpu-a100
#SBATCH --job-name=esmfold
#SBATCH --output=./log/ESMFold.%J.out
#SBATCH --error=./log/ESMFold.%J.err
#SBATCH -t 04:00:00
#SBATCH --mem=32GB
#SBATCH --ntasks=2
#SBATCH --gres=gpu:1

# Load necessary modules for the job
module load miniconda3
module load cuda
module load python/3.12.5

date

# Activate the ESMFold conda environment
source activate /90daydata/shared/protein_structure_conda/esmfold_env
export TRANSFORMERS_CACHE=/90daydata/shared/protein_structure_conda/.cache/

# Define input arguments
PDB_FILE=$1
OUT_DIR=$2
CHUNK_SIZE=$3

# Run the Python script for ESMFold monomer with specified arguments
echo "Running bulk_esmfold_transformers.py ${PDB_FILE} ${OUT_DIR} ${CHUNK_SIZE}"
python3 ./python/bulk_esmfold_transformers.py ${PDB_FILE} ${OUT_DIR} ${CHUNK_SIZE}

# Deactivate the Conda environment after the script completes
conda deactivate

date

```

Check the status of slurm job:

```
squeue -u $USER
```

View log files:

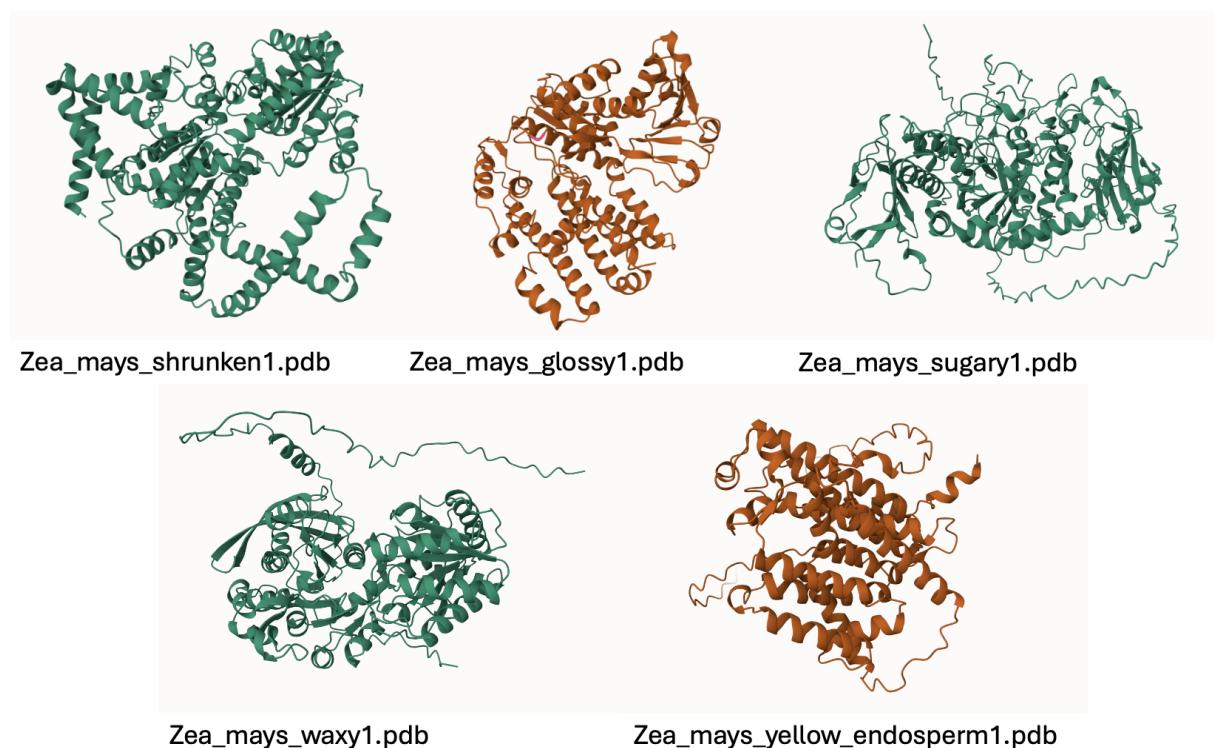
```
cd log  
ls -ltrh  
tail ESMFold.<JOBID>.out  
tail ESMFold.<JOBID>.err  
cd ..
```

Calculate the global pLDDT scores for each PDB file (< 1 minute):

```
sbatch scores.sh ./pdb/ ./output/maize5_scores.tsv  
head ./output/maize5_scores.tsv  
sort -k3,3 -r ./output/maize5_scores.tsv
```

Visualize the sequences:

- Navigate to the pdb directory in Atlas using Globus
- Transfer the PDB(s) files of interest locally
- Drag the files into Mol* (<https://molstar.org/viewer/>)



The predicted 3D structures of the five maize proteins by ESMFold. Visualization using the Mol* webserver.

Task #2: Run ESMFold-multimer on a set of five Fusarium/maize multimer protein sequences (~5 - 10 minutes).



Maize ears after *F. graminearum* inoculation. (A) Two ears of the resistant line. (B) Two ears of the susceptible line. The infected area is indicated by a red ellipse. Image from Figure 1 of “Transcriptomic responses in resistant and susceptible maize infected with *Fusarium graminearum*.” (Yuan et al. 2020).

Fusarium species, like other plant pathogenic fungi, secrete small proteins called effectors that allow them to bypass plant defenses and cause disease. In this task, we aim to predict the protein structures for five *Fusarium graminearum* effector proteins in complex with five corresponding proteins from the maize B73 genome.

Parameters:

- Input FASTA file: ./fasta/fusarium_maize.fasta
- Output directory: ./pdb/
- Chunk size: 32

```
sbatch bulk_multimer.sh ./fasta/fusarium_maize.fasta ./pdb/ 32
```

```

#!/bin/bash -l

#SBATCH -A scinet_workshop1          # Account name for the job
#SBATCH --partition=gpu-a100          # Partition to submit the job
#SBATCH --job-name=esmfold            # Name of the job
#SBATCH --output=./log/ESMFold.%J.out # Standard output
#SBATCH --error=./log/ESMFold.%J.err # Standard error
#SBATCH -t 04:00:00                  # Maximum runtime of 4 hours
#SBATCH --mem=32GB                   # Allocate 32GB of memory
#SBATCH --ntasks=2                   # Number of tasks
#SBATCH --gres=gpu:1                 # Request 1 GPU resource

# Load necessary modules for the job
module load miniconda3             # Load Miniconda module
module load cuda                    # Load CUDA for GPU support
module load python/3.12.5           # Load Python version 3.12.5

date                                # Print the current date and time

# Activate the ESMFold conda environment
source activate /90daydata/shared/protein_structure_conda/esmfold_env
export TRANSFORMERS_CACHE=/90daydata/shared/protein_structure_conda/.cache/

# Define input arguments
PDB_FILE=$1                          # PDB file to process
OUT_DIR=$2                            # Output directory for results
CHUNK_SIZE=$3                         # Chunk size (32, 64, 128, etc.)

# Run the Python script for ESMFold multimer with specified arguments
echo "multimer_esmfold_transformers.py ${PDB_FILE} ${OUT_DIR}"
${CHUNK_SIZE}
python ./python/multimer_esmfold_transformers.py ${PDB_FILE} ${OUT_DIR}
${CHUNK_SIZE}

# Deactivate the Conda environment after the script completes
conda deactivate

date                                # Print the date

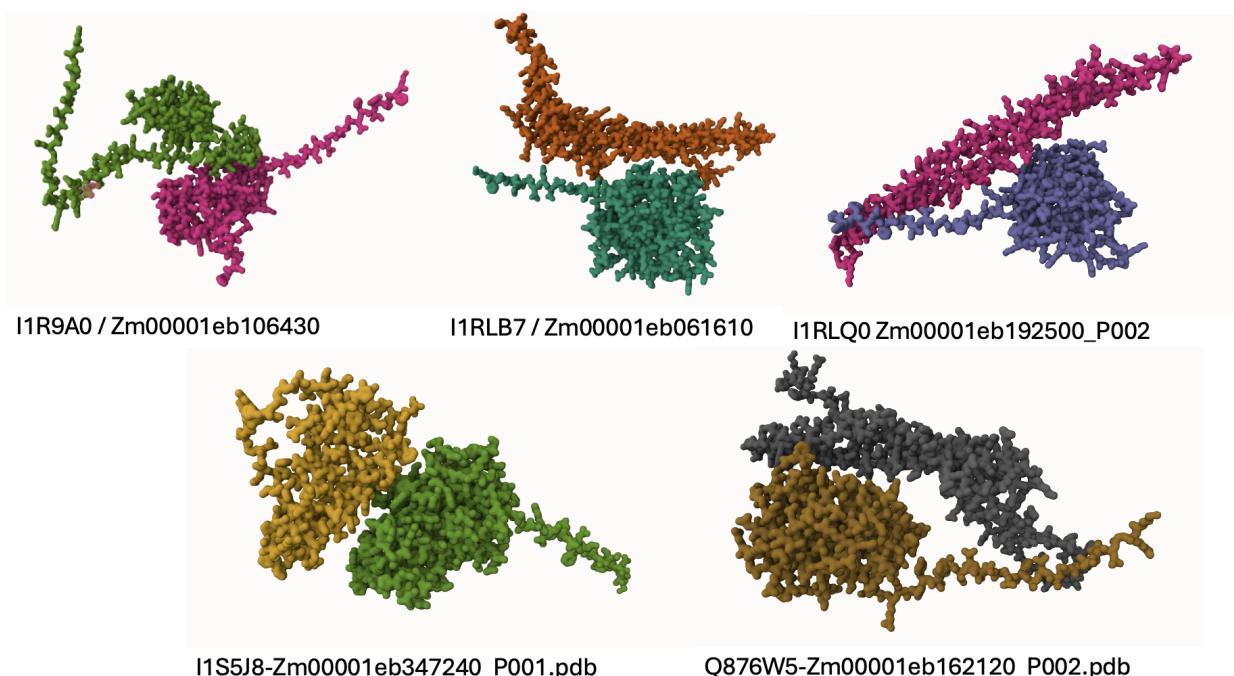
```

Recalculate the global pLDDT scores for each PDB file (< 1 minute):

```
sbatch scores.sh ./pdb/ ./output/maize5_scores.tsv  
tail ./output/maize5_scores.tsv
```

Visualize the sequences:

- Navigate to the pdb directory in Atlas using Globus
- Transfer the PDB(s) files of interest locally
- Drag the files into Mol* (<https://molstar.org/viewer/>)



The predicted 3D structures of the five Fusarium/maize complexes predicted by ESMFold. Visualization using the Mol* webserver.

Notes:

- Make sure sequences do not contain any special characters like “.”, “*”, “X”.
- Run the ESMFold on a GPU node.

Task #3: Run ESMFold on a set of five maize multimer protein sequences on the ESMFold webserver (~5 - 10 minutes).

- Navigate to <https://esmatlas.com/resources?action=fold>
- Try out the example sequences.
- Upload your own sequences (400 amino acid limit)

Fold Sequence [Learn more ↗](#)

Fold Sequence ▼

>PETase

MGSSHHHHHHSSGLVPRGSHMRGPNTAASLEASAGPFTVRSFTVSRPSGYGAGTVYPTNAGGTVGAIIVPGYTARQS
SIKWWGPRLASHGFWITIDTNSTLDQPSRSSQQMAALRQVASLNGTSSPIYGKVDARMGVMGWSMGGGLISAA
NNPSLKAAPQAPWDSNSTFSSVTPTLIFACENDSIAPVNSSLAPIYDMSMRNAKQFLEINGGSHSCANSQNSQALIGK

Try an example:

[Plastic degradation protein - PETase](#) [Antifreeze protein - 1EZG](#) [AI-generated protein - 8CYK](#)

[7-bladed propeller fold - Neuraminidase](#)

>PETase

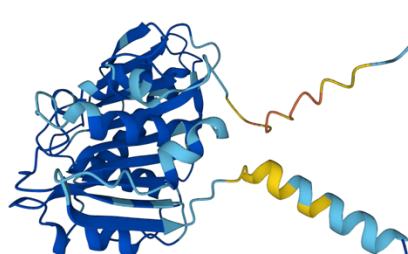
Sequence of Chain 1: A

11	21	41	51	61	71	81	101	111
NGSSHHHHHHSSGLVPRGSHMRGPNTAASLEASAGPFTVRSFTVSRPSGYGAGTVYPTNAGGTVGAIIVPGYTARQS	SIKWWGPRLASHGFWITIDTNSTLDQPSRSSQQMAALRQVASLNGTSSPIYGKVDARMGVMGWSMGGGLISAA	NNPSLKAAPQAPWDSNSTFSSVTPTLIFACENDSIAPVNSSLAPIYDMSMRNAKQFLEINGGSHSCANSQNSQALIGK						
121	131	141	151	161	171	181	191	201
RSSQQMAALRQVASLNGTSSPIYGKVDARMGVMGWSMGGGLISAA NNPSLKAAPQAPWDSNSTFSSVTPTLIFACENDSIAPVNSSLAPIYDMSMRNAKQFLEINGGSHSCANSQNSQALIGK								
211	221	231	241	251	261	271	281	291
NGGHSCANSQNSQALIGKKGVAWMKRFMDNDTRYSTFACEPNSTRVSDFR7ANCSLEDPAANKARKEAELAAATAEQ								

Download

[PDB file](#) [Sequence](#)

The predicted structure is colored by local prediction confidence (pLDDT) per amino acid location. Blue indicates confident predictions (pLDDT > 0.9), while red indicates low confidence (pLDDT < 0.5).

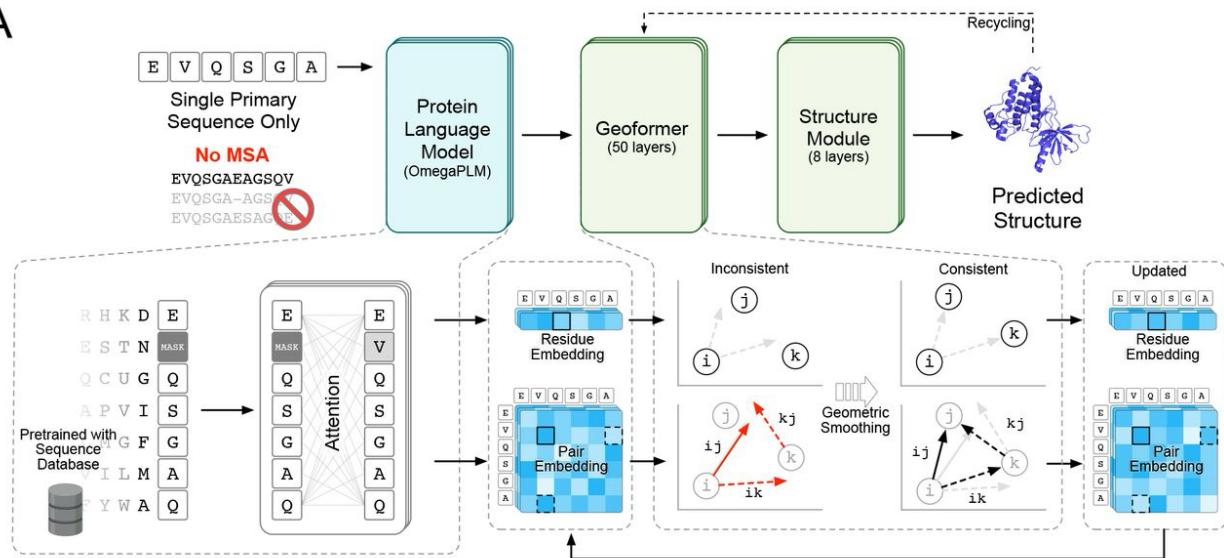


Predict protein structure with [ESMFold](#)

OmegaFold

WHAT IT IS: An AI program that predicts protein structures using *de novo* methods, which distinguishes it from other protein structure prediction programs (e.g., AlphaFold and RoseTTAFold).

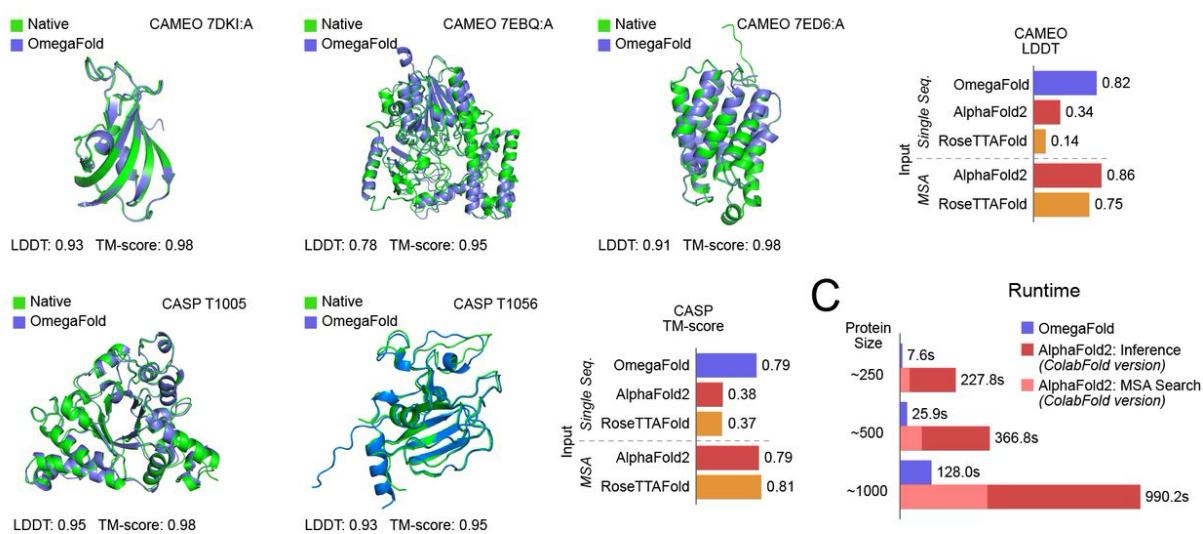
A



Source: Wu et al. 2022

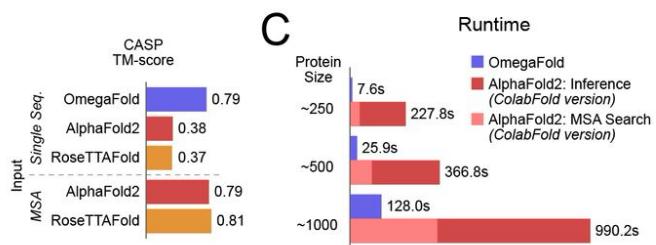
OmegaFold achieved superior or comparable prediction accuracy for newly reported anti-bodies and orphan proteins at faster runtimes when compared to RoseTTAFold and AlphaFold, respectively. **These results make OmegaFold an attractive option for protein structure prediction when multiple sequence alignments (MSAs) are not available or fail to provide clear evolutionary signal.**

B



Source: Wu et al. 2022

C



NOTE: THIS IS FOR REFERENCE ONLY. WE WILL NOT RUN THIS CODE. To use OmegaFold on the command line, if needed, request a compute node using the “salloc” command and then load the module.

We can then check to see if the module was loaded successfully.

```
salloc -A YOUR_PROJECT  
module load omegafold  
omegafold --help
```

The syntax for running OmegaFold is as follows:

```
omegafold INPUT_FILE.fasta OUTPUT_DIRECTORY
```

INPUT DATA:

For our analyses today, we will use any one or more of the 10 monomer fasta input files.

Organism	Fasta input file (input sequences)	Length (amino acids)
Rice	Oryza_sativa_HMA_OsHIPP19	78
Maize Fungal pathogen	Ustilago_maydis_Cmu1_USTMA	97
Rice Fungal pathogen	Magnaporthe_oryzae_AvrPikD_C4B8B8	113
Wheat	Triticum_aestivum_A0A077RXP2	134
Maize Fungal pathogen	Fusarium_graminearum_TPP1_FGSG_11164	252
Maize Fungal pathogen	Fusarium_graminearum_NLS1_FGSG_04563	315
Maize Fungal pathogen	Fusarium_graminearum_TRI14_FGSG_03543	371
Maize Fungal pathogen	Fusarium_graminearum_I1RR40_FGSG_06549	474
Maize	Zea_mays_A0A1D6FS01	712
Rice	Oryza_sativa_Pik1	1142

Build local files

We should have already built our own \$USER and working directory, so let's go into the “protein_workshop” sub-folder and copy the OmegaFold folder and its contents into your own working directory.

```
cd /90daydata/shared/$USER/  
cd protein_workshop  
cp -r /90daydata/shared/protein_structure_workshop/OmegaFold .
```

```
cd OmegaFold
```

DO NOT EDIT ANY FILES IN /90daydata/shared/protein_structure_workshop/

RUN OMEGAFOLD – SINGLE JOB:

To run OmegaFold, create SLURM batch script (.sub file)

```
nano omegafold_single_job.sub
```

Edit the below SLURM batch script (email, input file, and maybe output directory)

```
#!/bin/bash
#SBATCH --account=scinet_workshop1 #put HPC account name here, required on Atlas
#SBATCH --job-name="omegafold"      #name of this job
#SBATCH -p gpu-a100                #name of the partition (queue) you are submitting to
#SBATCH --gres=gpu:a100:1           #Specify your GPU partition to access reserved altas-0245
#SBATCH -N1                         #number of nodes
#SBATCH -n8                         #number of cores
#SBATCH --ntasks=16
#SBATCH --mem=256GB                 #Real memory (RAM) required (MB), 0 is the whole-node memory
#SBATCH -t 24:00:00                  #time allocated for this job hours:mins:seconds
#SBATCH --mail-user=YOUR_EMAIL@usda.gov    #CHANGE ME
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --error=OmegaFold.%J.err
#SBATCH --output=OmegaFold.%J.out

date                                #optional, prints out timestamp at the start of the job in stdout file
module purge

#load the omegafold program
module load omegafold/1.1.0

#Define bash variable containing the path to input Fasta - must not contain * as a translation stop
INPUT_FILE=PATH/TO/FILE.fasta

#Define bash variable containing the path to output directory
OUT_DIR=PATH/TO/OUTPUT_DIRECTORY

#Run the program
omegafold "$INPUT_FILE" "$OUT_DIR"
```

To execute the job, run the script with the following code:

```
sbatch omegafold_single_job.sub
```

If needed, the job can be cancelled using the following code, where <JOB NUMBER> is replaced with your job's ID:

```
scancel <JOB NUMBER>
```

PARALLELIZE OMEGAFOLD – JOB ARRAY:

To run OmegaFold in parallel using a SLURM job array, create batch script (.sub file)

```
nano omegafold_array.sub
```

Edit the below SLURM batch script (email, input file, and maybe output directory)

```
#!/bin/bash
#SBATCH --account=scinet_workshop1 #put HPC account name here, required on Atlas
#SBATCH --job-name="omegafold"      #name of this job
#SBATCH -p gpu-a100                #name of the partition (queue) you are submitting to
#SBATCH --gres=gpu:a100:1           #Specify your GPU partition to access reserved altas-0245
#SBATCH -N1                        #number of nodes
#SBATCH -n8                        #number of cores
#SBATCH --ntasks=16
#SBATCH --mem=256GB                 #Real memory (RAM) required (MB), 0 is the whole-node memory
#SBATCH -t 24:00:00                  #time allocated for this job hours:mins:seconds
#SBATCH --mail-user=YOUR.EMAIL@usda.gov    #CHANGE ME
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --error=OmegaFold.%A_%a.err
#SBATCH --output=OmegaFold.%A_%a.out
#SBATCH --array=1-10                 #The number of jobs run at once can be set using "%" and your number

date                                #optional, prints out timestamp at the start of the job in stdout file

module purge

#load the omegafold program
module load omegafold/1.1.0

#Define the path to the config file
config=./Config.txt

#Define the input
FASTA=$(awk -v ArrayTaskID="$SLURM_ARRAY_TASK_ID" '$1==ArrayTaskID {print $2}' "$config")

#Define the output directory
OUTPUT_DIR=$PATH/TO/OUTPUT_DIRECTORY

omegafold $PATH/TO/"$FASTA" "$OUTPUT_DIR"
```

We can control the number of jobs that will run at once by modifying the array line

```
#SBATCH --array=1-10%2          #this will run two jobs at once until all jobs are completed
#SBATCH --array=1,3,5            #this will only run jobs for array IDs 1, 3, and 5
```

The configuration file should contain this text:

ArrayID	file
1	Oryza_sativa_HMA_OsHIPP19.fasta
2	Ustilago_maydis_Cmu1_UTM1.fasta
3	Magnaporthe_oryzae_AvrPikD_C4B8B8.fasta
4	Triticum_aestivum_A0A077RXP2.fasta
5	Fusarium_graminearum_TPP1_FGSG_11164.fasta
6	Fusarium_graminearum_NLS1_FGSG_04563.fasta
7	Fusarium_graminearum_TRI14_FGSG_03543.fasta
8	Fusarium_graminearum_I1RR40_FGSG_06549.fasta
9	Zea_mays_A0A1D6FS01.fasta
10	Oryza_sativa_Pik1.fasta

NOTE: WE WILL NOT RUN THIS CODE

To execute the job, run the script with the following code:

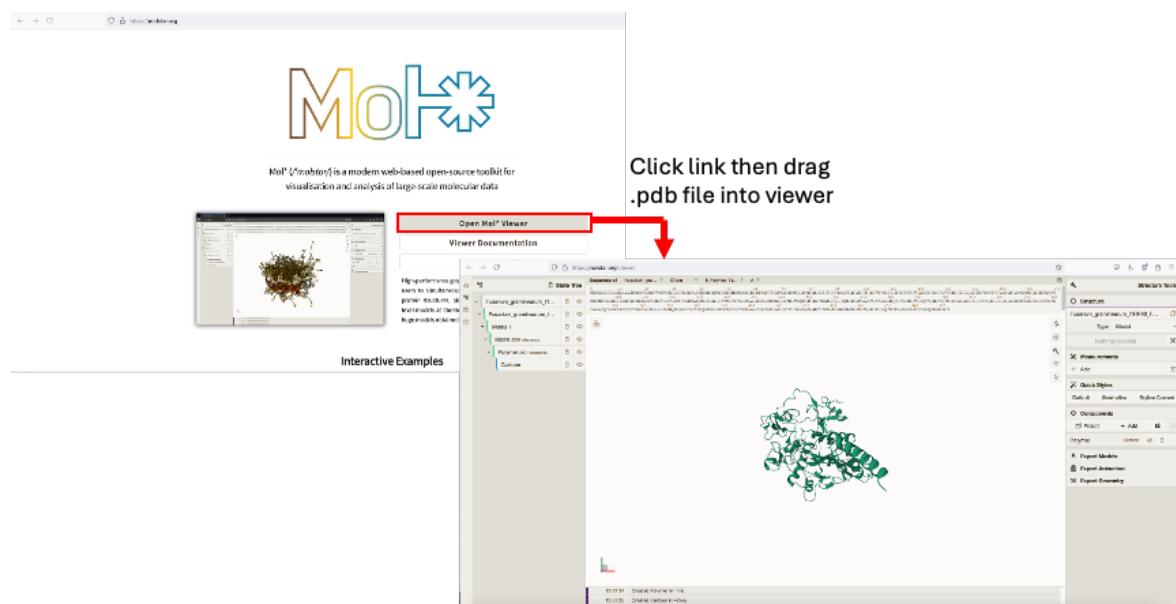
```
sbatch omegafold_array.sub
```

If needed, the job can be cancelled using the following code, where <JOB NUMBER> is replaced with your job's ID:

```
scancel <JOB NUMBER>
```

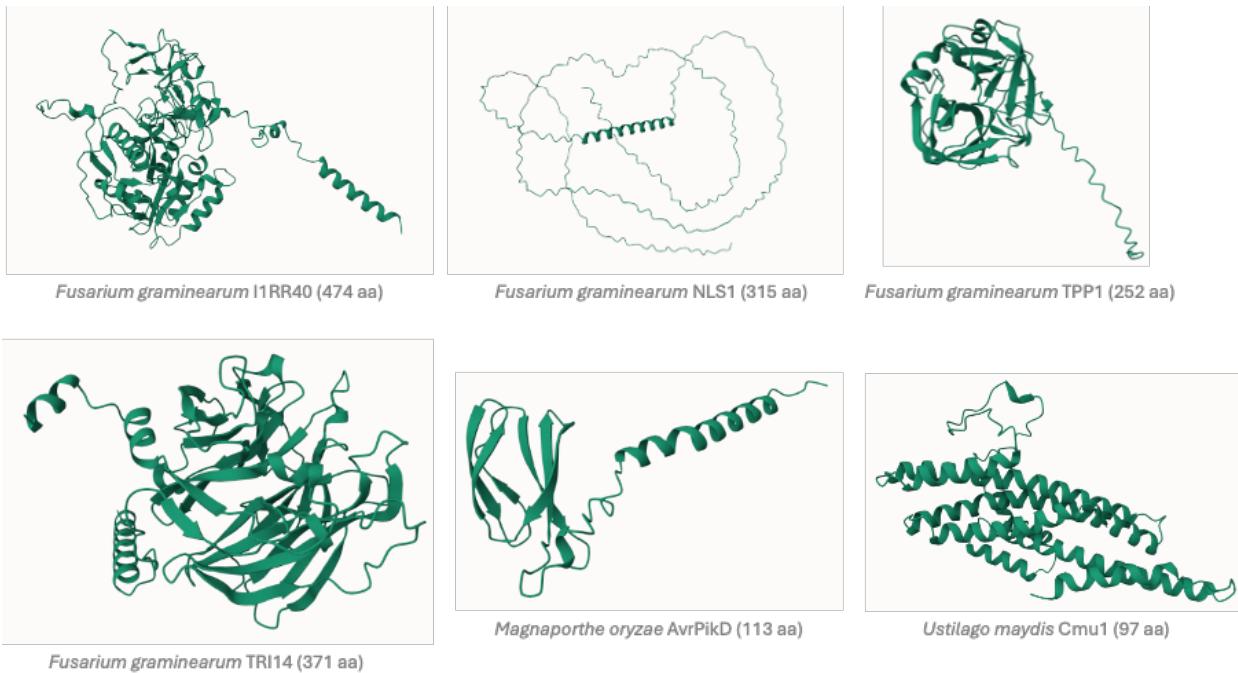
VISUALIZATION (Mol*):

We will use the Mol* online platform to visualize our results: <https://molstar.org/>



RESULTS:

PATHOGENS:



PLANTS:

