The objective of this project was to create a self-contained set of scripts to simulate a Rayleigh-Taylor Instability using a parallel incompressible Navier-Stokes solver with AMR on AWS using an open-source solver pulled from the Lawrence Berkeley National Lab (this solver forms a component of my PhD research code). To accomplish this, a cluster on the AWS in a master-slave format was created using a combination of python boto3 and bash scripting where necessary. For this assignment, t2 Amazon Linux instances were used. The following steps were adopted to accomplish this task:

1) A security group called 'SG_RESCALE' is created. To allow instances to communicate freely between themselves, all ports from 0-65535 are made open to all other instances within this security group.
2) A key-pair called 'ec2-keypair' is also created which will allow SSH access into the Linux instance.
3) A new EC2 instance of type t2.micro (using create_instances in boto3) is created. This will be the default instance or master node. It is associated with the key-pair and security group created previously. Its relevant attributes (private/public IP addresses, zone and instance ID) are hashed and written to file for future reference.
4) A bash script is then generated which takes as input the destination public IP address. It facilitates the transfer of AWS config files to this instance, updates the requisite libraries and then installs OPENMPI, gcc compilers and git onto this default Linux instance. The path to the MPI libraries is appended to the bash profile such that they are loaded whenever the current shell or future shells are started. 'ssh-keygen' is called to generate a RSA key which is stored in ~/.ssh/authorized_keys.
5) To forgo calling this setup repeatedly on each slave node, the image is saved and will simply be duplicated for slave instances.
6) Using this image, new "n_slave instances" slave instances are spun up within the same security group using the create_instances command in boto3 and the private IP addresses for each node are saved and a hostfile is simultaneously generated for use with mpirun. The private node IP addresses are appended to /etc/hosts/ on the master node. Since these instances are replicas, they have the same ~/.ssh/authorized_keys file allowing the master node to access them freely. In principle, this approach is scalable to any number of instances (limited by AWS).
7) A volume is created (using create_volume in boto3) of size 'volume_size' and attached to the master node (default instance) using 'attach_volume'. Once the volume is ready, it is converted to an ext4 filesystem and then mounted to /home/ec2-user/data. To mount it onto each slave node, each slave node is accessed, and this volume is mounted using the master node's private IP address.
8) The AMR framework and required linear libraries housed in AMReX, and the Navier- Stokes solver IAMR is cloned, and the code is compiled on the shared volume created previously. The simulation is initiated using mpirun and by specifying the hostfile which contains the localhost and slave-node aliases. Once the simulation is completed (after 50 time-steps), the results (plt* files readable in Visit https://wci.llnl.gov/simulation/computer-codes/visit/) and the output of the simulation (aws-rayleigh-taylor.out) is available in the aws-results folder created.
9) The instances are looped over and terminated, the attached volumes, security group and keypair are then deleted.

Note: aws-.sh scripts are created using python files to accomplish some of the tasks; alternatively one could have as easily stored the public IP address and read it through a bash script to SSH onto the AWS instance OR setup password-less between my local machine and the AWS instance.

A sample output from Visit depicting the unstable Rayleigh Taylor configuration of the denser fluid (red) above the lighter fluid in blue and the subsequent mixing. The simulation output is recorded after 50 time-steps with 2 levels of adaptive refinement (refinement criteria based on vorticity magnitude). Re-gridding is performed every 2 iterations. The black boxes are used to visualise the AMR block-structured. Near the interface, the grid is twice as fine as the coarse outer mesh.