What is the Health Data Analytics Platform (HDAP)?
HDAP is a software platform designed to support students and researchers in conducting health-oriented projects at Georgia Tech and our partner institutions.

What does HDAP do?
HDAP has three core functions:

1) Provide synthetic and de-identified sources of healthcare data
2) Provide tools to analyze these data
3) Provide a healthcare application development and hosting environment and app gallery

What data sources does HDAP provide?
Currently HDAP offers four datasets:

- *MIMIC*
  - A de-identified inpatient ICU dataset
  - 46,500 patients
- *EXACT*
  - A synthetic outpatient dataset
  - 10,500 patients
- *SyntheticMass*
  - A synthetic outpatient dataset
  - 20,000 patients
- *SynPUF*
  - A synthetic outpatient dataset
  - 2,300,000 patients

How do I access these data?
The raw datasets can be accessed in two ways:

- Via Postgres database connection (must be on the GT network)
  - *SynPUF*
    - jdbc:postgresql://data.hdap.gatech.edu:5434/synpuf_v5
    - Username: hdap_student
    - Password: hdapSynpuf4Students!
  - *MIMIC*
    - jdbc:postgresql://data.hdap.gatech.edu:5433/mimic_v5
    - Username: team0
    - Password: hdapM1m1c4Students!
  - *EXACT*
    - jdbc:postgresql://data.hdap.gatech.edu:5432/omop_v5
    - Username: hdap_student
    - Password: hdap3xact4Students!
  - SyntheticMass
    - Only FHIR access at this time

- Via FHIR
    - MIMIC (DSTU2)
        - Endpoint: http://ehr.hdap.gatech.edu:8080/gt-fhir-webapp/
    - EXACT (DSTU2)
        - Endpoint: http://ehr.hdap.gatech.edu:8080/gt-exact/
    - SyntheticMass (STU3)
        - Endpoint: https://apps.hdap.gatech.edu/hapi-fhir-jpaserver-example/
    - SynPUF
        - FHIR server pending

What tools does HDAP provide to explore and analyze data?

HDAP currently provides two ways to analyze healthcare data

- *ATLAS* - a software application with a GUI interface for exploring datasets, building patient lists based on criteria of interest, and analyzing those patients
- *Jupyter* - HDAP provides a Jupyter server through which you can create notebooks in Python or R that can access HDAP datasets. These notebooks can be shared with your colleagues and other researchers.

In addition, you can analyze data locally using the database connections . However, once results are generated, no HDAP data should be persisted on your machine.

Can I add new data sources to HDAP?

- For student projects, you can stand up a database within your Docker environment (see below) and add any data necessary for your project. Obviously, no identifiable health data can be used for your project!
- To add datasets accessible to all HDAP users, a request can be sent to hdap@gatech.edu

What role does HDAP play in projects for CS6440 or other Georgia Tech courses?

At minimum, HDAP should be used by CS6440 students for 1) access to FHIR servers and 2) application deployment. Optionally, students may utilize other HDAP data and analytic tools to support development of their projects.

How does the requirement to deploy on HDAP affect my application development?

Deploying your project on HDAP requires use of Docker and adherence to conventions as described below. During your development phase, you can use other hosting environments such as AWS or your local machine. However, you must test and ensure that your application works on HDAP at least two weeks prior to project delivery. It is strongly recommended that you deploy a ‘Hello World’ version of your application earlier in the semester to ensure that you understand the process and that your Docker environment is working correctly on HDAP. Your TAs may require this ‘Hello World’ deployment as a checkpoint in your project development.

Development Guide

Overall Steps

1. Setup Code Repository
2. Install Docker in your development environment
3. Build application following HDAP conventions
4. Define application environment with Dockerfile(s) and Docker Compose
5. Define build process with Jenkins
6. Deploy Application

Setup Code Repository

How do I set up my code repository?

- Create project repository on GT GitHub: https://github.gatech.edu/
    - Use GT credentials for access
    - Store source code in repository
- Add TAs as collaborators to your repository.
    - Go to the “Settings -> Collaborators” menu option for your project
    - Add an entry for your TA
- Add a Webhook for the HDAP Jenkins server to be notified when updates are pushed to your repository
    - Go to the “Settings - > Hooks and Services” menu option for your project
    - Press the “Add webhook” button in the “Webhooks” section.
    - Add the following URL as the “Payload URL” https://apps2.hdap.gatech.edu/jenkins/github-webhook/
    - For “Content type” select “application/x-www-form-urlencoded”
    - Leave “Secret” empty
    - For “Which events would you like to trigger this webhook?” select “Just the push event.”
    - Press the “Add webhook” bottom.

Install Docker and Docker Compose

Why do I need to use Docker?

- Provides repeatable way to build/test application. If it runs in Docker on your personal machine then it will run in Docker on any machine.

Why do I need Docker Compose?

- Provides a standardized way to configure all containers used by an application and start/stop them with a single command.

How do I install Docker?

- Download and install Docker for your operating system. Follow the instructions here: https://docs.docker.com/install/

How do I install Docker Compose?

- Download and install docker-compose for your operating system. Follow the instructions here: https://docs.docker.com/compose/install/

Build Application

What languages/frameworks can I use for building my app?

- We've had success with the following
    - Languages: Java, Python, Javascript
    - Frameworks: Spring, Django, Angular, Node, Grails

What should I keep in mind while developing?

- Your GitHub repository must have the project source code. It MUST NOT contain only a docker-compose.yml file that pulls images from DockerHub.
- Your application should work from any deployment context. It might not be deployed at the root context of the application server. This means your application should work:
    - if deployed to the root context: https://exampleserver/
    - If deployed to another context https://exampleserver/application1/
- DO NOT hardcode URLs and port numbers into the application
    - Application port numbers will be changed at deployment time.
    - Use properties files for absolute URLS to external resources.
    - Use of a properties file simplifies deployment if URLs and port numbers need to be changed in source code prior to application deployment.

Using Docker

What do I need to know about the Dockerfile?

- The Dockerfile defines commands used to build a docker image. It defines the base image to start with, files from your source code to copy in to the image, commands to run to configure the image, and other application configuration information.
- Getting started guide https://docs.docker.com/get-started/
- Dockerfile instructions to know
    - **FROM** - sets a base image for the instructions in the file. For example an image based off the Ubuntu 16.04 Docker image would use the following "FROM ubuntu:16.04" https://docs.docker.com/engine/reference/builder/#from
    - **ENV** - sets environment variables to use in the image. For example "ENV JAVA_HOME="some_path"" would set the JAVA_HOME environment variable to use in the image. https://docs.docker.com/engine/reference/builder/#env

- **COPY** - copies files from the source code into the image. For example “COPY ./somefile /usr/local/app/somefile” would copy the file “somefile” in the source code to the /usr/local/app directory of the image. https://docs.docker.com/engine/reference/builder/#copy
- **RUN** - executes a command in an image. For example “RUN ls -la” would run the ls command in the image. https://docs.docker.com/engine/reference/builder/#run
- **CMD** - Provides the default command for the executing container. There can only be one CMD in the Dockerfile. For example “CMD ["postgres"] would launch Postgres. https://docs.docker.com/engine/reference/builder/#cmd
- For more details please see: https://docs.docker.com/engine/reference/builder/

What do I need to know about Docker Compose?

- Docker Compose is a tool for defining and running multi-container applications. It defines the information needed to configure the containers of your applications and launch them with a single command. For more information see https://docs.docker.com/compose/gettingstarted/ and https://docs.docker.com/get-started/
- For our deployment environment **Docker Compose V2** is the best to know https://docs.docker.com/compose/compose-file/compose-file-v2/
- It defines a service stack for your application.
- All services are placed on the same network.
- Applications defined in a docker compose service can access each other with URLs that use the service name.
    - For example if you have docker compose file with services “application-ui” and “application-db”. The application running in the “application-ui” service can access the database with the a JDBC URL of the following format “jdbc:postgresql://application-db/database”
- Recommended that docker-compose.yml files build images from source code to ease the deployment process. Allows for developers to make a change to source code, and quickly initialize a Docker service stack with the updated images.
- docker-compose instructions to know
    - **build** - defines how to build the image. https://docs.docker.com/compose/compose-file/#build
    - **restart** - sets the restart policy for the container. For example “restart: always” indicates the container should always be restarted. https://docs.docker.com/compose/compose-file/#restart
    - **ports** - maps host (the machine running Docker) ports to container ports. For example:

      ports:
        - “5000:8080”

      Will map host port 5000 to container port 8080
      https://docs.docker.com/compose/compose-file/#ports
    - **depends_on** - indicates a dependency between services. For example

depends_on:
  - app_db

Indicates the service depends on the app_db service
https://docs.docker.com/compose/compose-file/#depends_on

- **volumes** - maps volumes to the container. For example:

  volumes:
    - /some/host/dir:/some/container/dir

  Maps the directory "/some/host/dir" to the "/some/container/dir" in the container for the service.
  https://docs.docker.com/compose/compose-file/#volumes

Where can I get more Docker help?

- For more information see the getting started guide https://docs.docker.com/get-started/

Are there any project examples?

- Yes, the following projects provide examples:
  - https://github.gatech.edu/es130/ADEWS
  - https://github.gatech.edu/es130/hdap_angular_sample
  - https://github.gatech.edu/es130/hdap_angularjs_sample

Deploy with Jenkins

Jenkins is used as the Continuous Integration (CI) tool for applications deployed to HDAP. The HDAP Jenkins server uses the Jenkins Pipeline capability for building projects. Specifically, it uses project defined Jenkinsfiles for detailed instructions for building and deploying applications.

Why should I use a Jenkinsfile?

- A Jenkinsfile provides a standardized way to tell Jenkins how to test, build, and deploy your application.
- Will allow HDAP staff and TA to easily set up Jenkins jobs to build and deploy your applications.
- Jenkins will follow the steps defined in your Jenkinsfile to build and deploy your application, and provide feedback when your application no long builds or deploys.

How do I use a Jenkinsfile?

- Create a file named "Jenkinsfile" in the root directory of your project. This file will contain instructions, defined in a Groovy DSL or script, that tell Jenkins how to test, build, and deploy your application.
- You can define all the stages for the CI pipeline for your application. Each stage can use different Docker images to perform the necessary steps.
- Jenkinsfile syntax can be found here: https://jenkins.io/doc/book/pipeline/syntax/
- Jenkinsfile directives to know

- **stages** - contains multiple stage directives outlining the CI process.
- **stage** - contains the details for a specific stage of the CI process.
- **steps** - contains the series of one or more steps to perform during a stage.
- **sh** - defines a shell script to run within a step.
- **script** - defines a groovy script to run as part of the CI process.
- **docker** - contains docker configuration information for a specific stage. This directive is specifically useful to indicate a specific image to use to run a stage of the CI process. For example:

```
stage('Test'){
        agent{
            docker{ image 'node:9' }
        }
        steps{
            sh '"echo something"'
        }
}
```

Will create a stage that uses the Docker Node version 9 image to create a container and run the script “echo something”

- More details can be found here https://jenkins.io/doc/book/pipeline/jenkinsfile/

How do I check my Jenkins build status?

- Navigate to the HDAP Jenkins site https://apps2.hdap.gatech.edu/jenkins/
- Select the Jenkins Job for your project
- Look at the “Build History” column on the left of the screen. If the build was successful it will have a green or blue dot next to it. If the build failed it will have a red dot next to it.

How do I check output for a failed Jenkins build?

- Navigate to the HDAP Jenkins site https://apps2.hdap.gatech.edu/jenkins/
- Select the Jenkins Job for your project
- Select the failed build in the “Build History” section.
- Select “Console Output” in the menu on the left side of the screen.
- The logs for the failed build will be displayed.

Is there a Jenkinsfile sample/template?

- You can find sample Jenkinsfiles in the following projects
    - https://github.gatech.edu/es130/hdap_angular_sample/blob/master/Jenkinsfile
    - https://github.gatech.edu/es130/ADEWS/blob/master/Jenkinsfile

Application Deployment

How do I get my application deployed?

- Prior to deployment, run the HDAP App Check app. This app will check to ensure that the development conventions have been followed.
- After successfully passing the HDAP App Check, talk with your TA to get Jenkins Job created and the application containers deployed. The TA will provide you with a URL on the Jenkins server where you can view project build status.
- Your application will be deployed to <project_sub_domain>.apps.hdap.gatech.edu