# Assignment: Module #1 - Core API Development and Deployment

## Objective

By the end of this assignment, you will be able to:

- Instantiate a Rails server application primarily for API use
- Configure the application for use with RDBMS and MongoDB
- Provision saging and production resources
- Deploy to separate staging and production instances
- Implement an end-to-end resource thread backed by RDBMS/ActiveRecord
- Implement an end-to-end resource thread backed by MongoDB/Mongoid
- Enable/require HTTPS for staging and production sites

## Requirements (180 mins)

I have created an honest estimated of 180 mins for this assignment based on your development environment being in place and giving you time to poke through some of the related documentation to the deployments.

1. (5 min) **Instantiate a Rails server application**. This application will primarily be used for RDBMS, MongoDB, and API implementation but may (depending on your decisions) also be responsible for deploying a future Angular web-client.
2. (15 min) **Configure the application for use with a relational and MongoDB database of your choice for development and test profiles**. It is not required that you use PostgresSQL for development and test profiles to pass this requirement. However, you are strongly encouraged to put that in place as the course moves forward with more complex database design.
    - Credentials may not be hard-coded in the source code (i.e., no credentials checked into Git) and must be provided through an environment variable
3. (30 min) **Provision (free) MongoDB databases for use in deployment**
    - Staging and production environments must have separate MongoDB instances * the production database URL must not be hard-coded into any source code and must be obtained through an environment variable that can be set in the Heroku deployment shell
4. (30 min) **Provision (free) Heroku and PostgreSQL resources for use in deployment**. PostgreSQL is commonly provisioned automatically for a Rails application based on the presence of the `pg` gem in the `Gemfile`.
    - Heroku production and staging URL names can be anything of your choice.
    - Logical Git remote names assigned to Heroku URL names should ideally be `staging` and `production`.
    - Staging and production environments must have separate PostgresSQL (occurs by default) instances.
    - The database URL must not be hard-coded into any source code and must be obtained through an environment variable that can be set in the Heroku deployment shell
5. (15 min) **Deploy a branch of your application to `production`** that displays information indicating the site is under construction. This site should have provisioned PostgreSQL and MongoDB resources but is not making use of them at this time. Your public repository must contain a tag for the version that is deployed to production for this assignment.
6. (15 min) **Deploy a branch of your application to `staging`** that addresses the functional requirements of this assignment. Your public repository must contain a tag for the version that is deployed to staging for this assignment.
7. (30 min) **Implement an end-to-end thread** from the API to the relational database for a resource called `Cities`.
    - The resource will have an `id:integer` (default) and `name:string` property
    - The resource name must be accessable via the `/api/cities` URI
    - The resource must be backed by a RDBMS using ActiveRecord
    - The resource will be deployed with a city with the name `Baltimore`

- A client performing a GET of of the `/api/cities` resource must at least get one city with name `Baltimore`

8. (30 min) Implement an end-to-end thread from the API to the MongoDB database for a resource called `States`.
    - The resource will have an `id:BSON::ObjectId` (default) and `name:string` property
    - The resource name must be accessible via the `/api/states` URI
    - The resource must be backed by MongoDB using Mongoid
    - The resource will be deployed with a state with the name `Maryland`
    - A client performing a GET of of the `/api/states` resource must at least get one state with name `Maryland`
9. (10 min) Configure your staging and production sites to require HTTPS for web communications

**Note**: (Optional) You may want to enable only the GET methods for the API interface to avoid anyone coming in and deleting or updating your city and state from what it is required to produce.

## Submission

Submit for grading

1. URLs to the `staging` and `production` deployments to be used via demonstration by the reviewer.
2. Link from a public Git repository for the two branches deployed. This will be used for code inspection by the reviewer.
3. Required git output to show

- Remotes and their URLs for source code (e.g., origin), staging, and production
- Remote branches for staging and production

## Version: 2016-12-19