**Java on AWS [[1]](#footnote-1)**

* Sign up for a an AWS account.
* Take care to stay within the Amazon free tier. Only run one EC2 virtual machine at one time.
* Use small virtual machine sizes – t1.micro or t2.micro, one at a time (maximum compute hours in the free tier is 750 per month)
* Consider using a pre-paid debit card to establish the AWS account rather than your credit card. American Express sells debit cards as small as $25.
* We will use AWS Elastic Beanstalk to deploy Java Web apps.[[2]](#footnote-2)
* Beanstalk will let you upload a Java WAR file, and will automatically create an EC2 virtual machine containing a Tomcat Java servlet engine, and deploy your app to it.
* We will provide you a Maven build script (pom.xml) which will build a Java WAR file.

**Maven Build [[3]](#footnote-3)**

* On your development machine (department Unix machine, or your own laptop), verify Maven is installed.

mvn -version

* If not installed, get it from maven.apache.org.
* Verify Java is installed, otherwise install it. (Run “javac -version” to verify.)
* Find proj0 bundle on the class Blackboard, download to your development machine and unpack.
* Navigate a command line window to the proj0/ folder and build the sample app:

mvn clean install

* This creates proj0/target/proj0-csi403.war. It contains compiled Java source files (those compiled from proj0/src/) as well as other WAR file configuration files. A WAR file uses the ZIP file format.

**Deploying the WAR to AWS using Elastic Beanstalk**

* From the AWS Services list, select Elastic Beanstalk.
* Click Create New Application, give it a name
* Select the new app, select Create New Environment
* Select a Web Server environment
* Choose Tomcat from the Platform list
* Click Upload your code, browse to the location of your WAR file
* Click Create Environment, wait a few minutes, see your assigned Elastic IP (EIP)
* Point your REST client (see below) at <http://yourEIP/yourServletName> and send a test message

**Redeploying an Updated WAR**

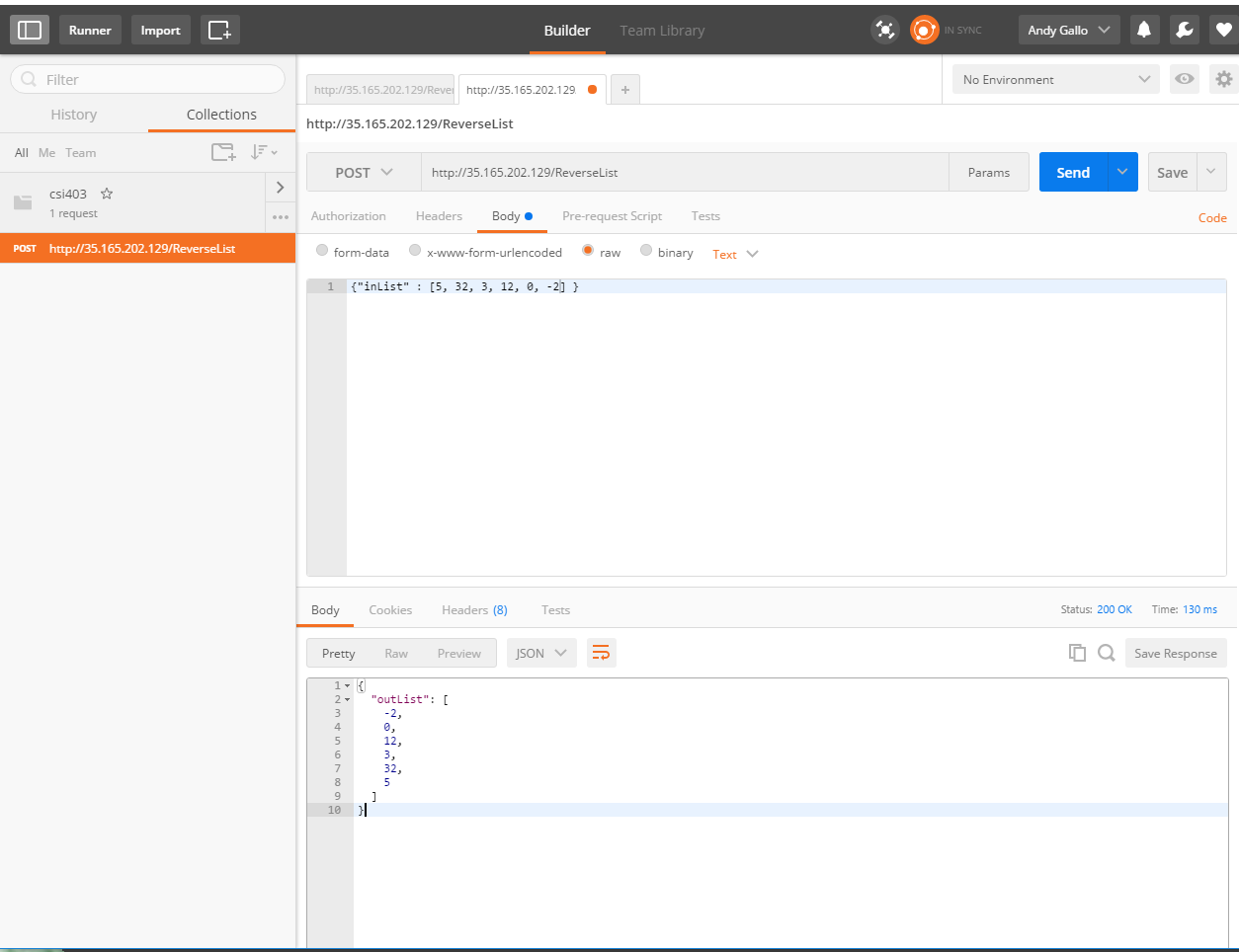
* From the AWS Services list, select Elastic Beanstalk
* Select the app
* Click Upload & Deploy, browse to the location of your WAR file
* Give the version a unique version label (e.g. myApp-1, myApp-2, etc.)
* Click Deploy, wait a few minutes (the EIP will remain the same)
* Point your REST client (see below) at <http://yourEIP/yourServletName> and send a test message

**Modifying the Sample WAR**

* The proj0 sample app defines a servlet called Reverse List. Suppose we wanted to make a copy of proj0, call it proj1, and add a new Servlet to it.
* Make a copy of proj0/ to proj1/ on the filesystem.
* Add new Java source files to proj1/src/main/java/csi403. Use the Java Servlet interface as shown in the Reverse List example.
* Modify proj1/src/main/webapp/WEB-INF/web.xml and add a new block of XML for your Servlet, following the example shown for the Reverse List service.
* Run the maven build (mvn clean install). (You can optionally alter the pom.xml to change the name of the WAR file produced.)

**Invoking Your Service**

* Your Servlet should accept JSON as an HTTP POST and return JSON. See the sample code in Reverse List.
* To test your service you can use any mechanism for sending the JSON via POST.
* Suggest making a file containing some input JSON, e.g. using the samples provided with the project descriptions.
* Install a Chrome browser tool like Postman[[4]](#footnote-4), point it at your deployed app URL. Copy and paste the JSON into Postman and HTTP POST it to your service.
* Alternatively, use a command line tool like curl[[5]](#footnote-5) to POST your JSON file to your service.



1. AWS is not required for this class. You are welcome to use any host you prefer. All the major cloud providers offer free usage tiers – Microsoft Azure, Google Cloud, etc. [↑](#footnote-ref-1)
2. This is by no means the only way to deploy Java to AWS. A new mechanism called AWS Lambda functions is particularly interesting for its use of the serverless microservice model. We will not cover this in class, but you are welcome to use it. [↑](#footnote-ref-2)
3. It is certainly possible to import our sample pom.xml into Eclipse or another Java IDE and build from there. We will not cover the use of such tooling in this class, but their use is encouraged. You are also welcome to not use our Maven script and instead to make your own using the build tooling of your choice. [↑](#footnote-ref-3)
4. https://www.getpostman.com/ [↑](#footnote-ref-4)
5. https://curl.haxx.se/dlwiz/?type=bin [↑](#footnote-ref-5)