**Story:**

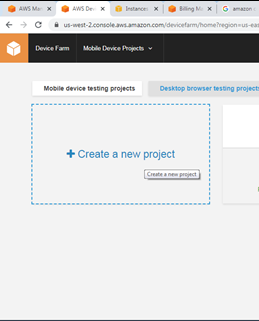
**Automated Mobile App Build & Test Using Amazon Device Farm**

**Steps:**

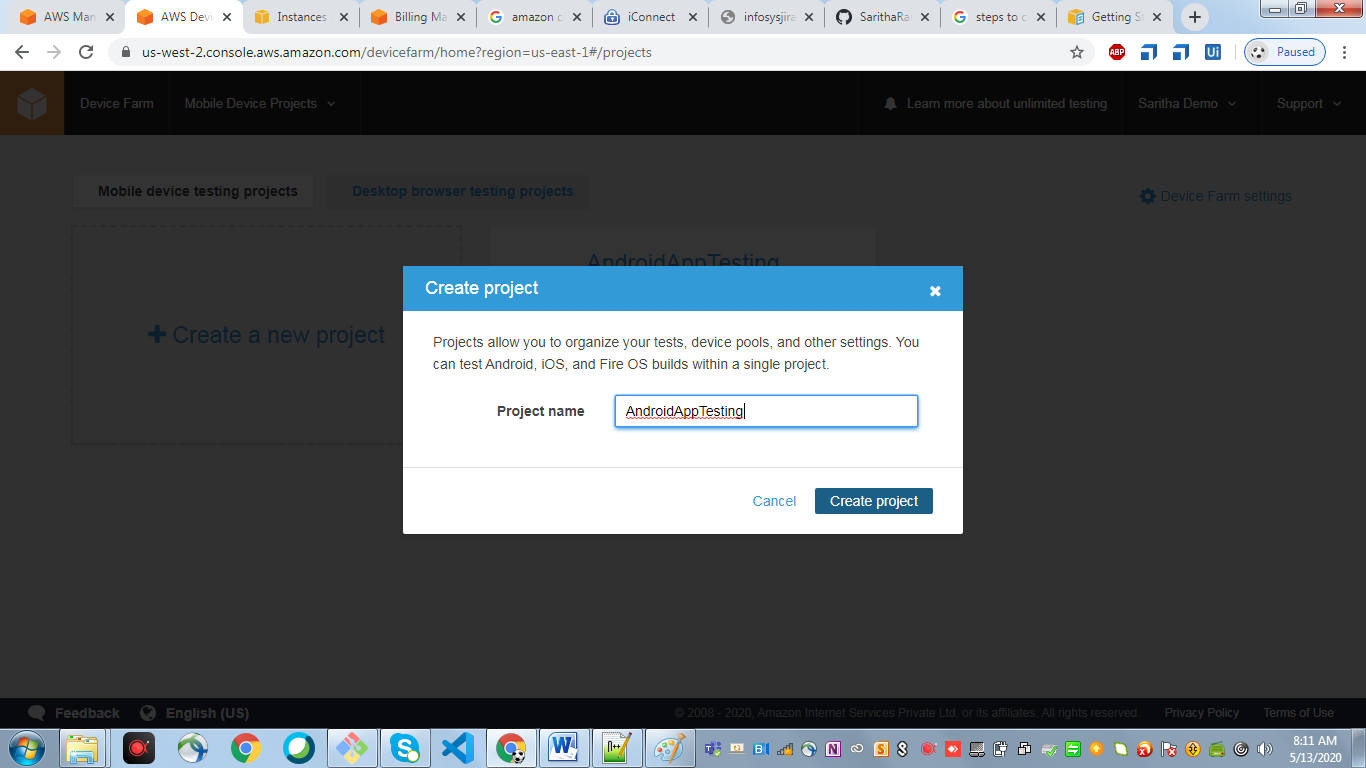
1. Get the Sample mobile app source code from GitHub

<https://github.com/SarithaRamineni/AndroidBuildAutomation>

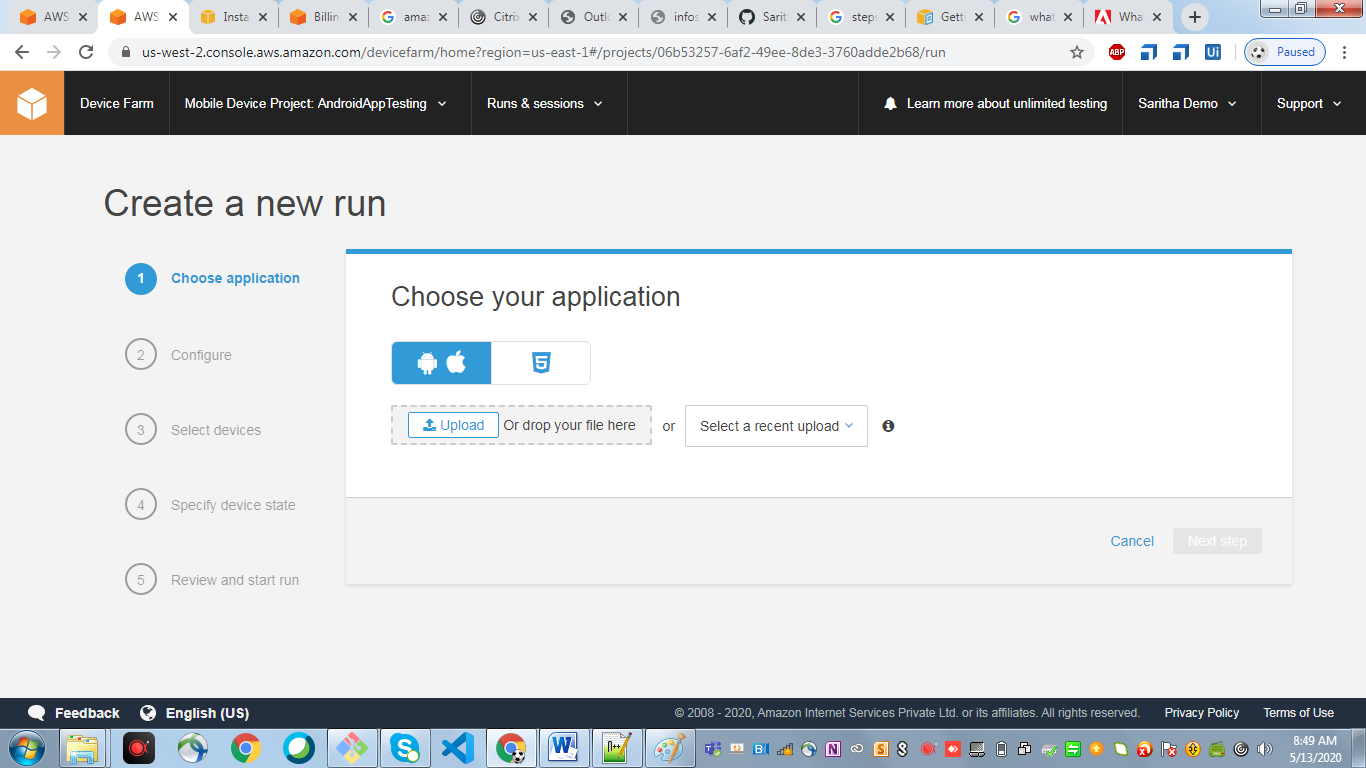
1. Go to amazon console and create a Device Farm Project
   * + Got to the DeviceFarm console and Click on the Create New Project



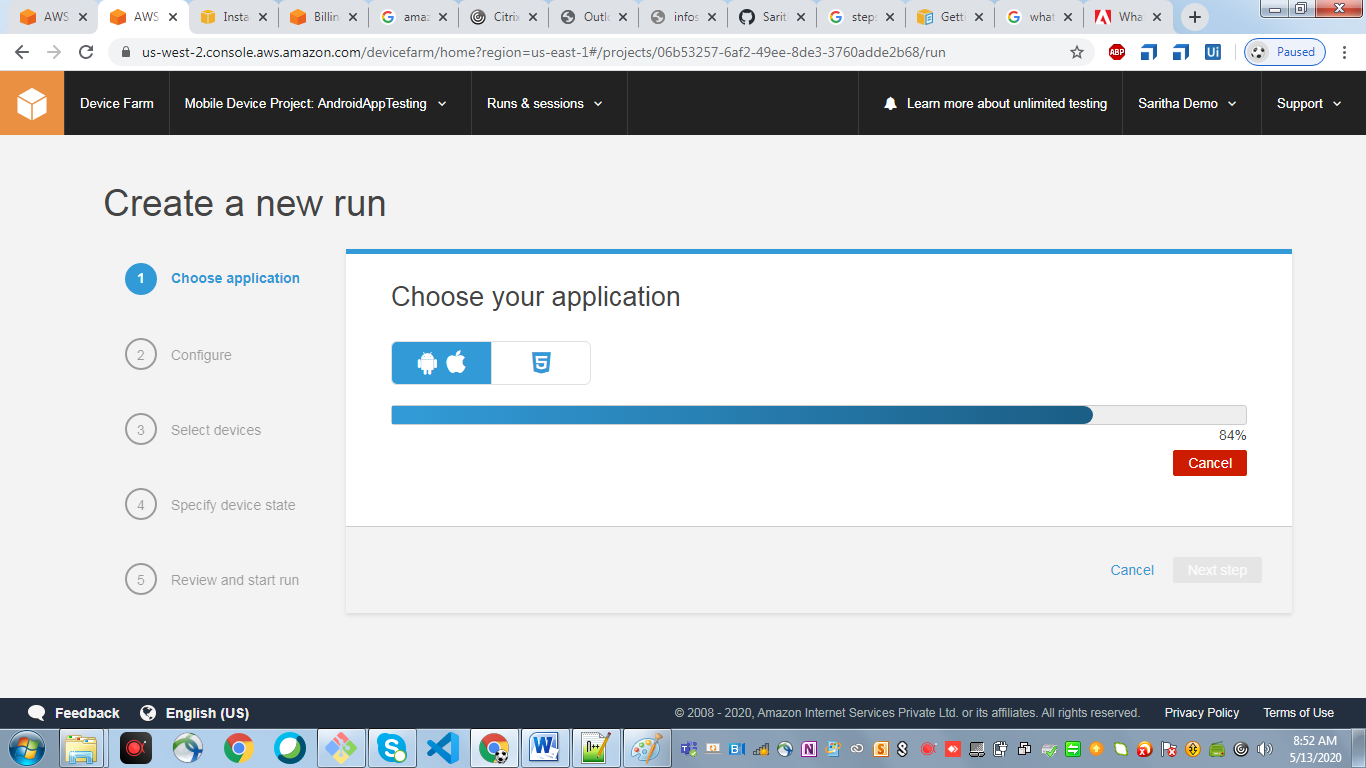
Give project Name and click on “Create Project”

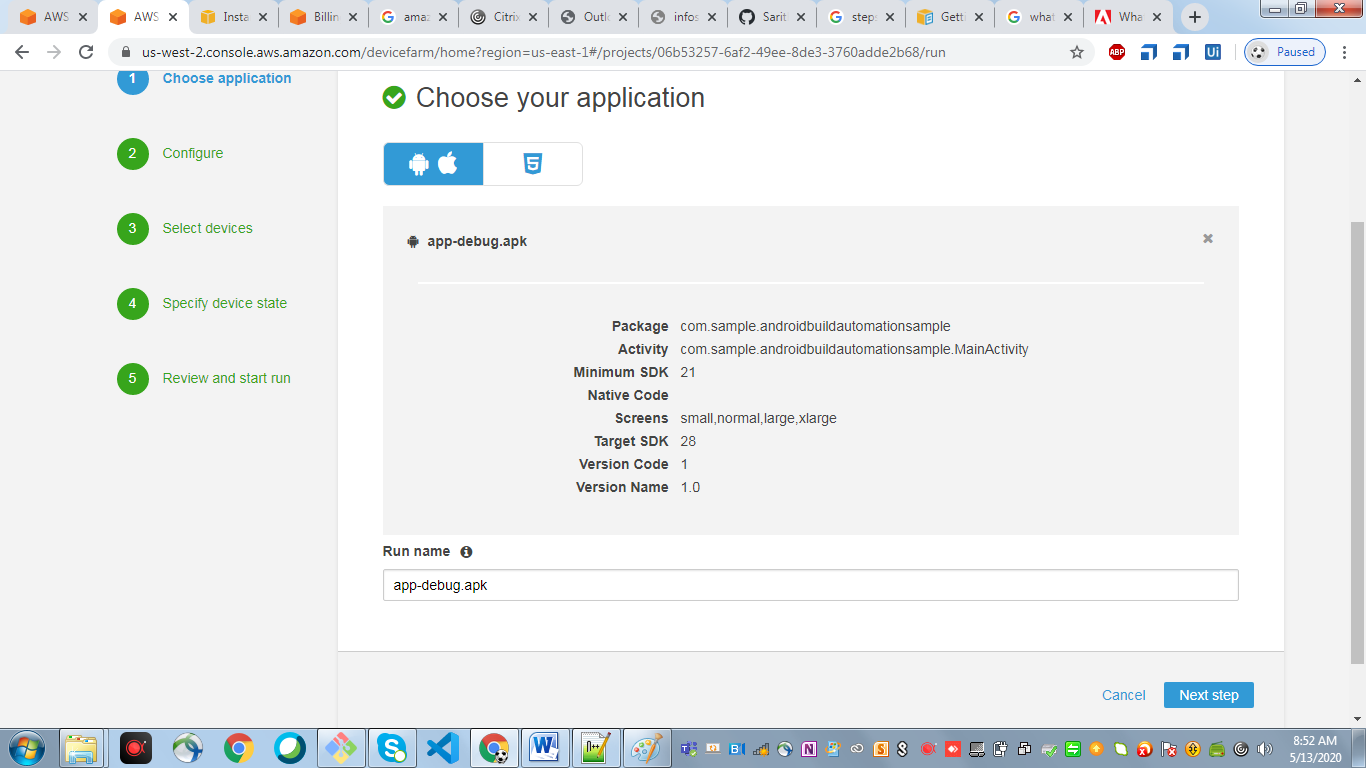


* + - Click on “Create a new run”
      * Choose Application (Android/IOS/Web App)

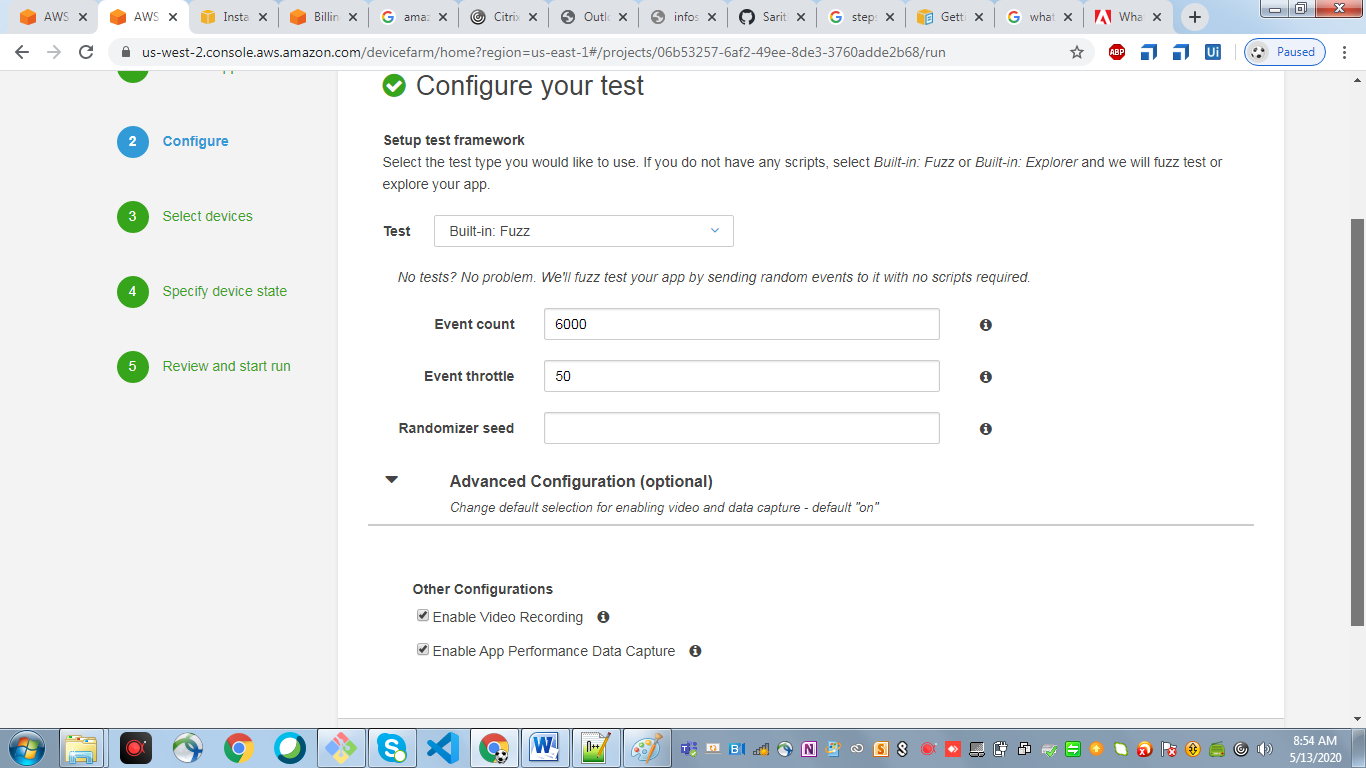


Click on upload and upload .apk file



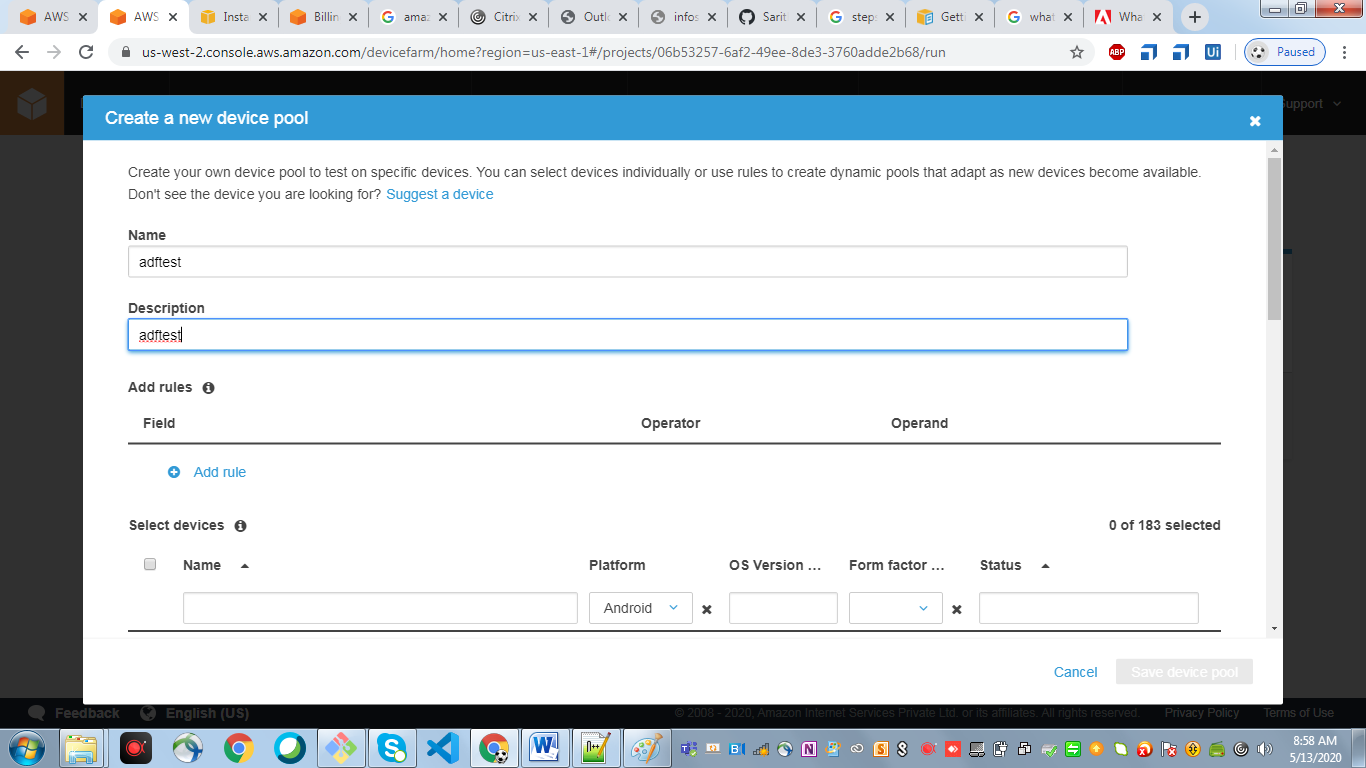


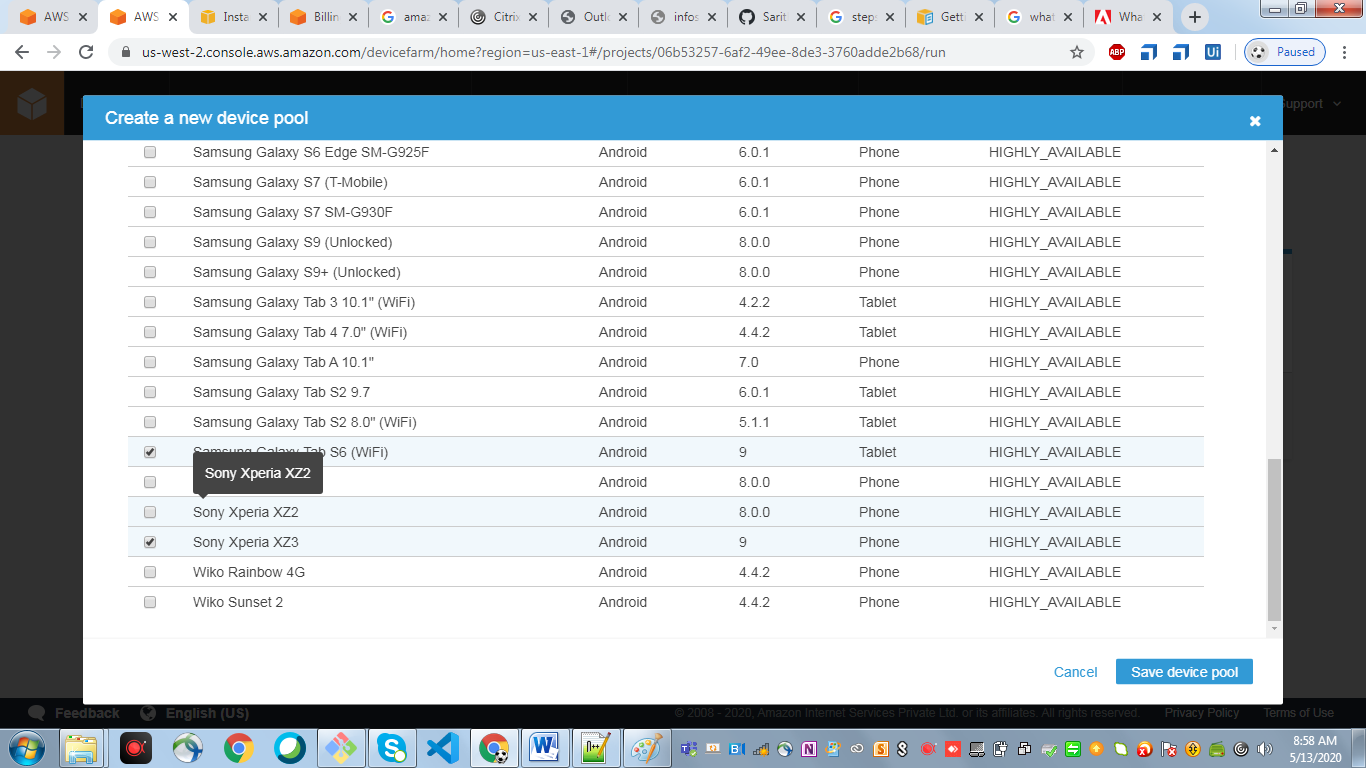
* + - Click on Next Step “Configure Your Test”

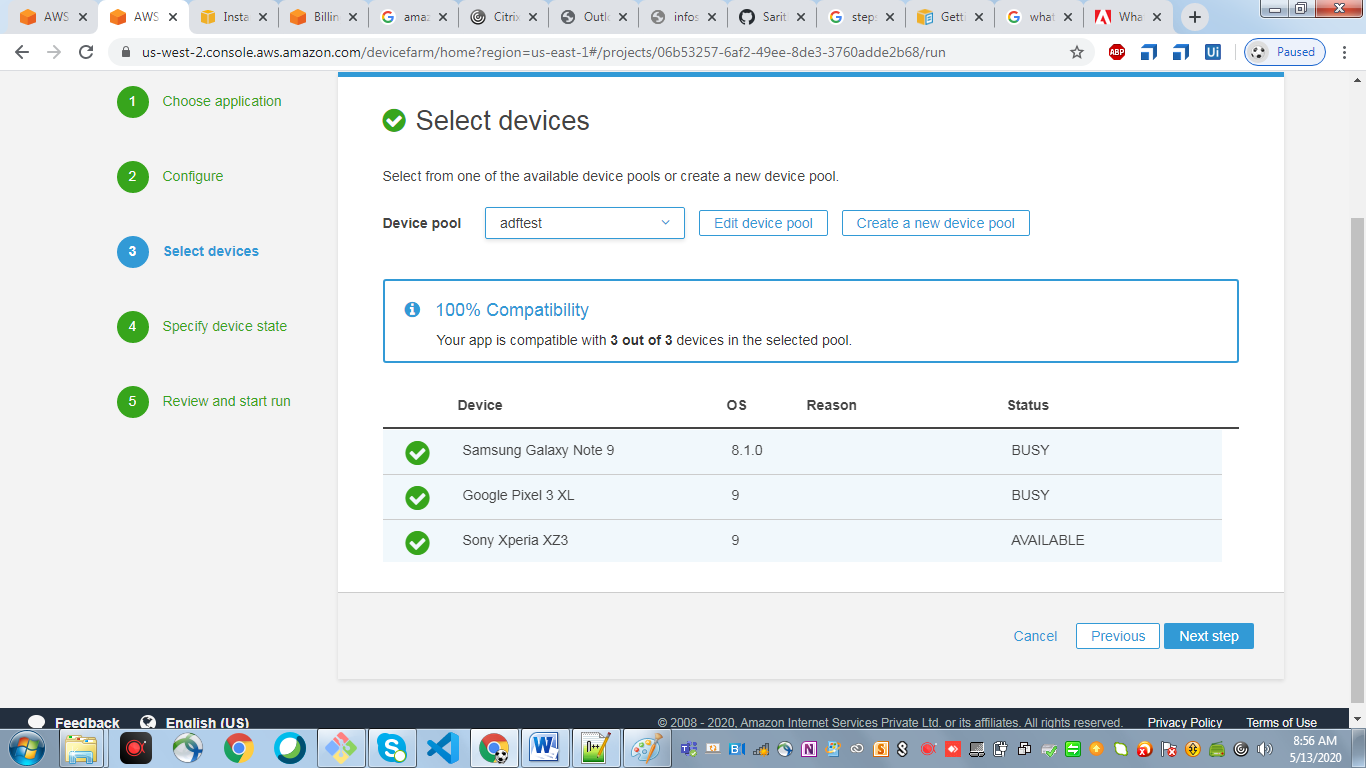


* + - Click on Next Step “Select Devices”

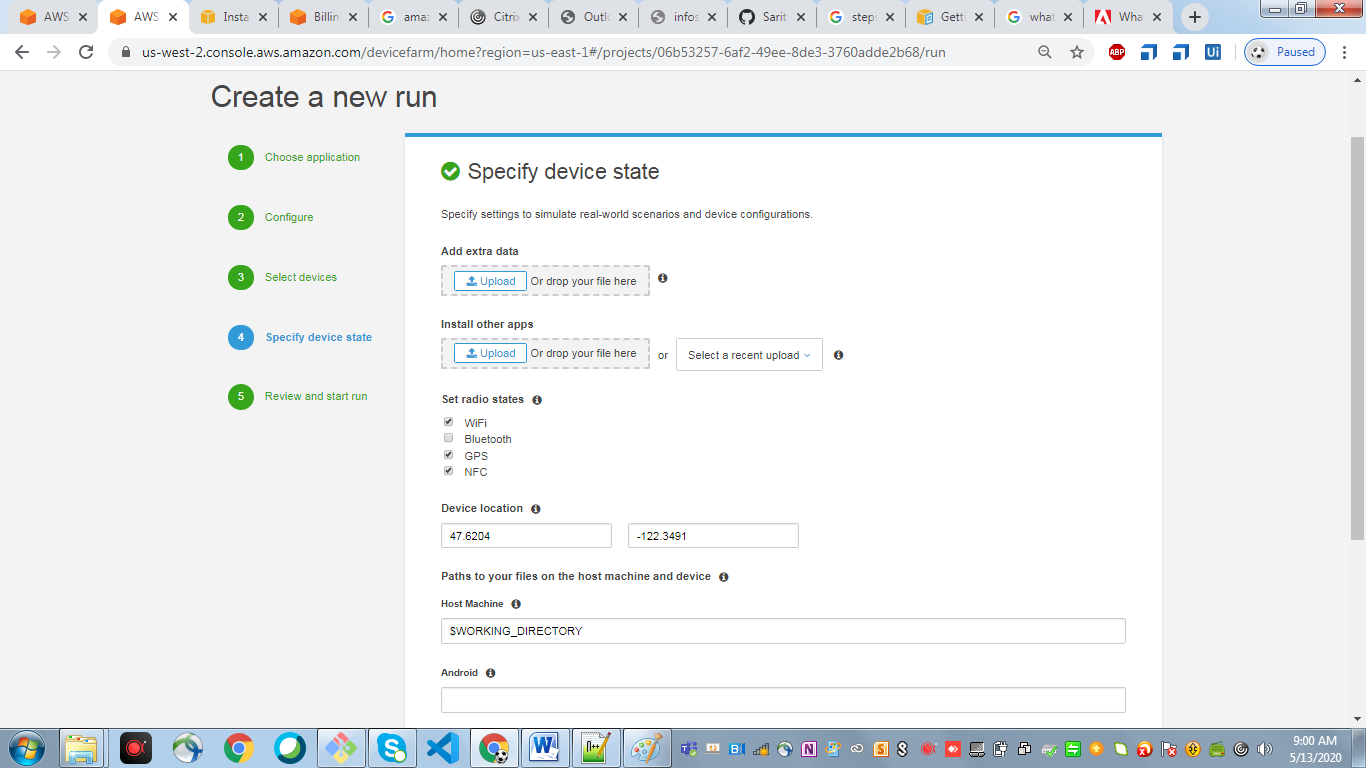
Create a device pool on which devices you want to perform your Android test



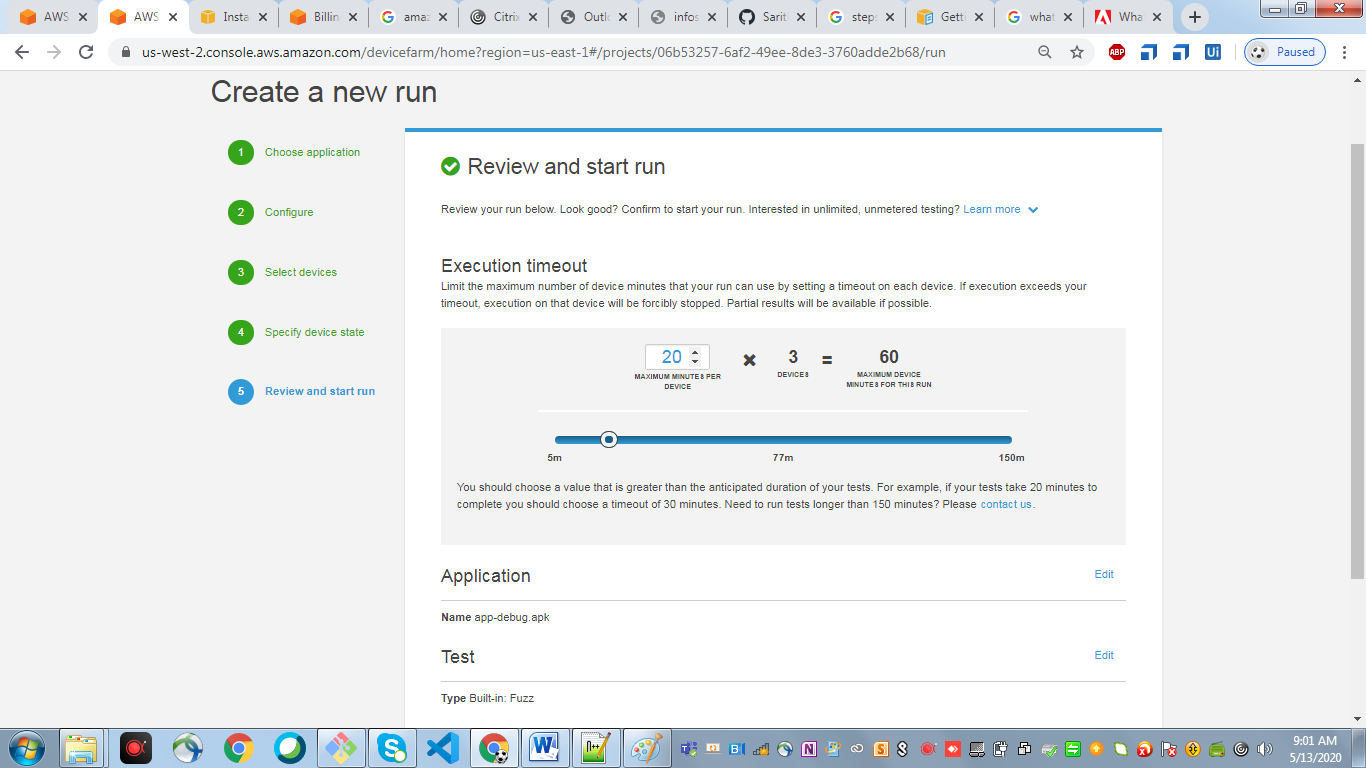


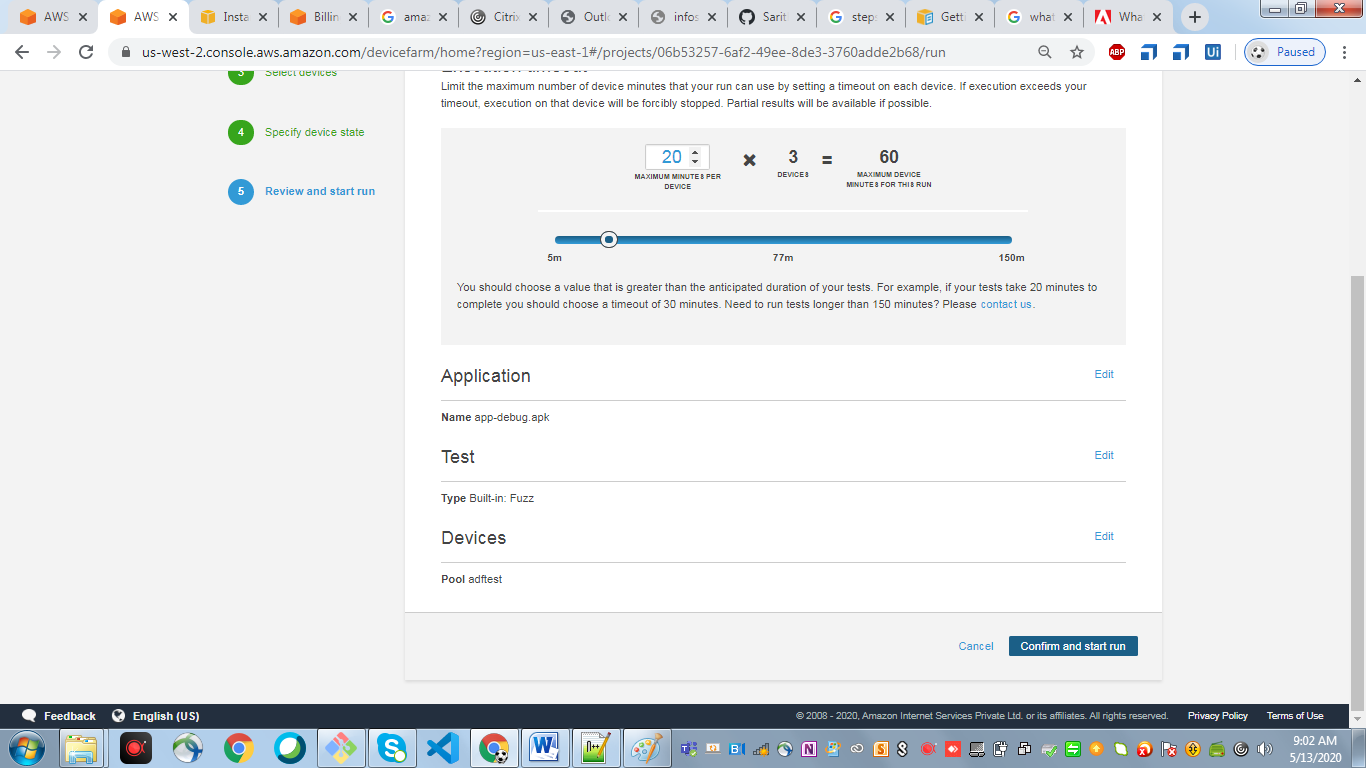


* + - Specify Device state

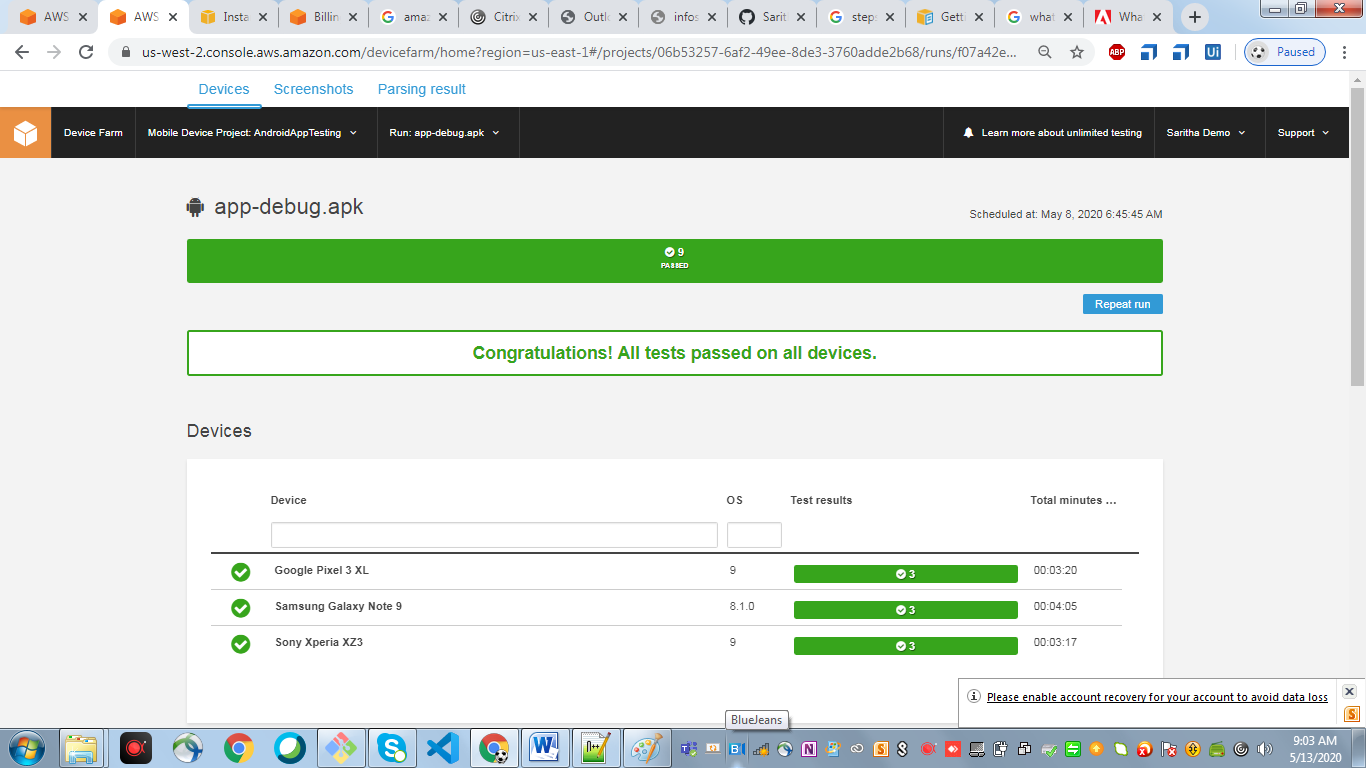


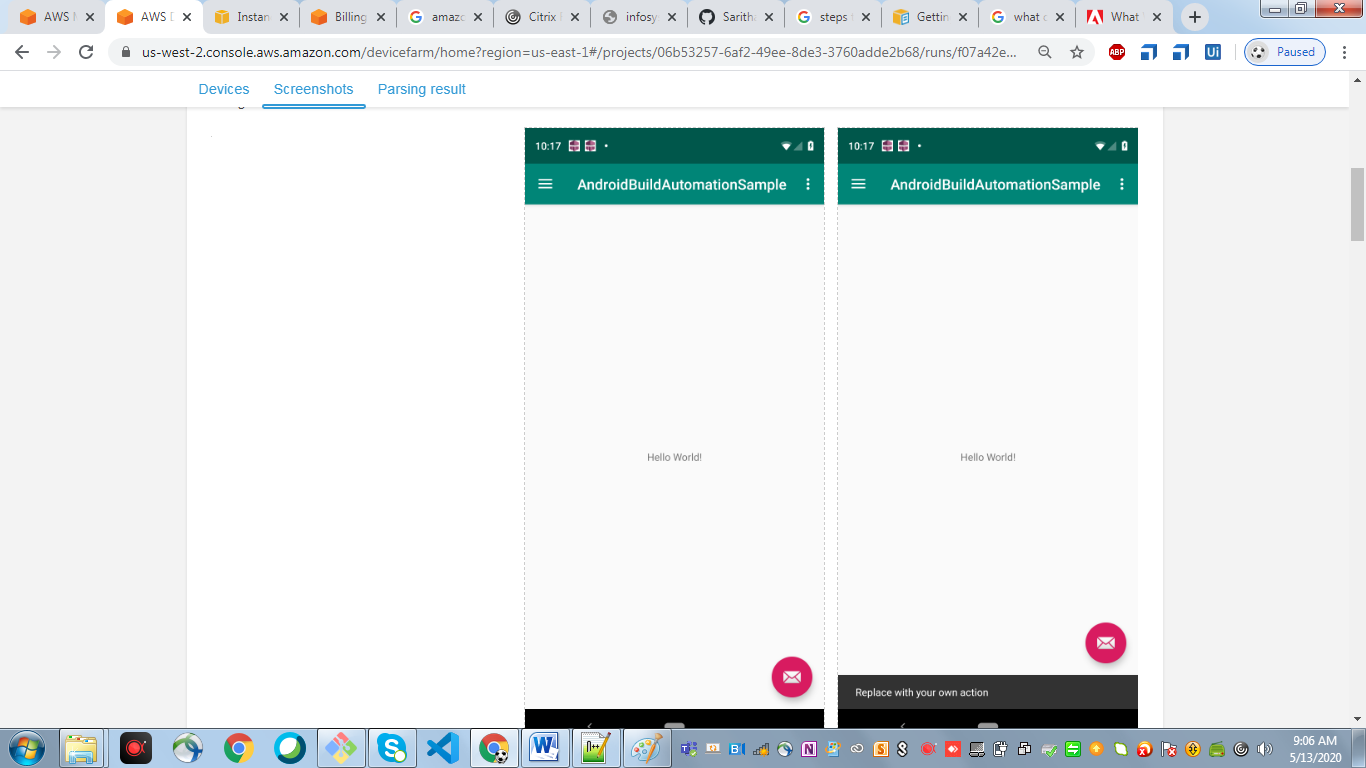
* + - Review and start run

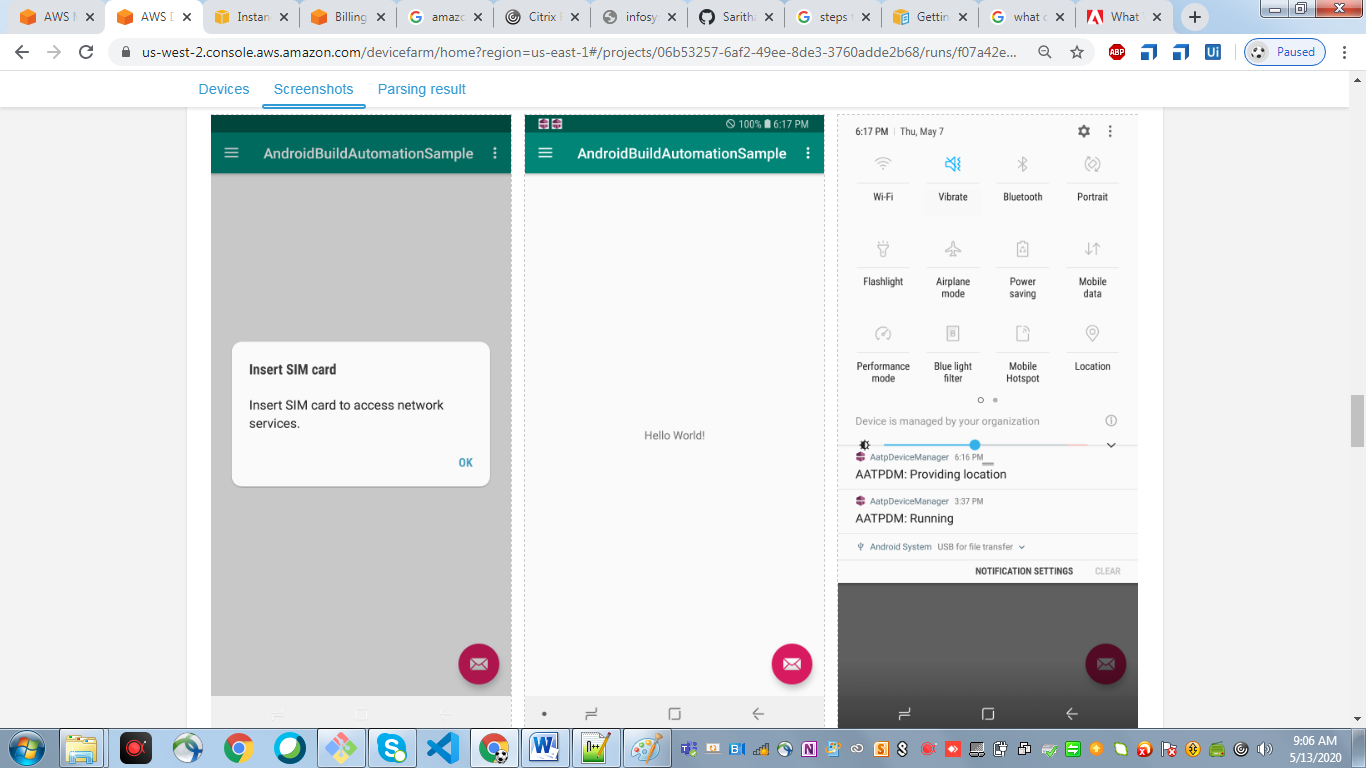


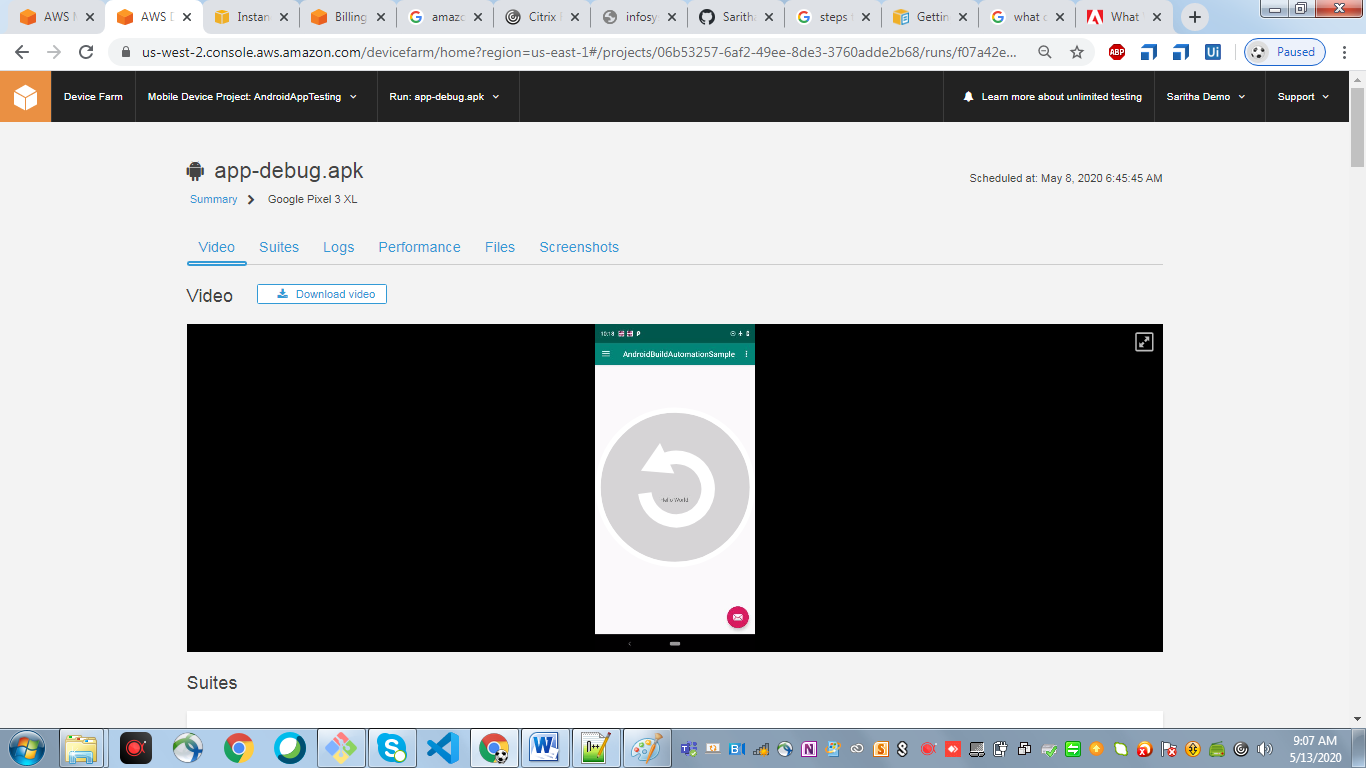


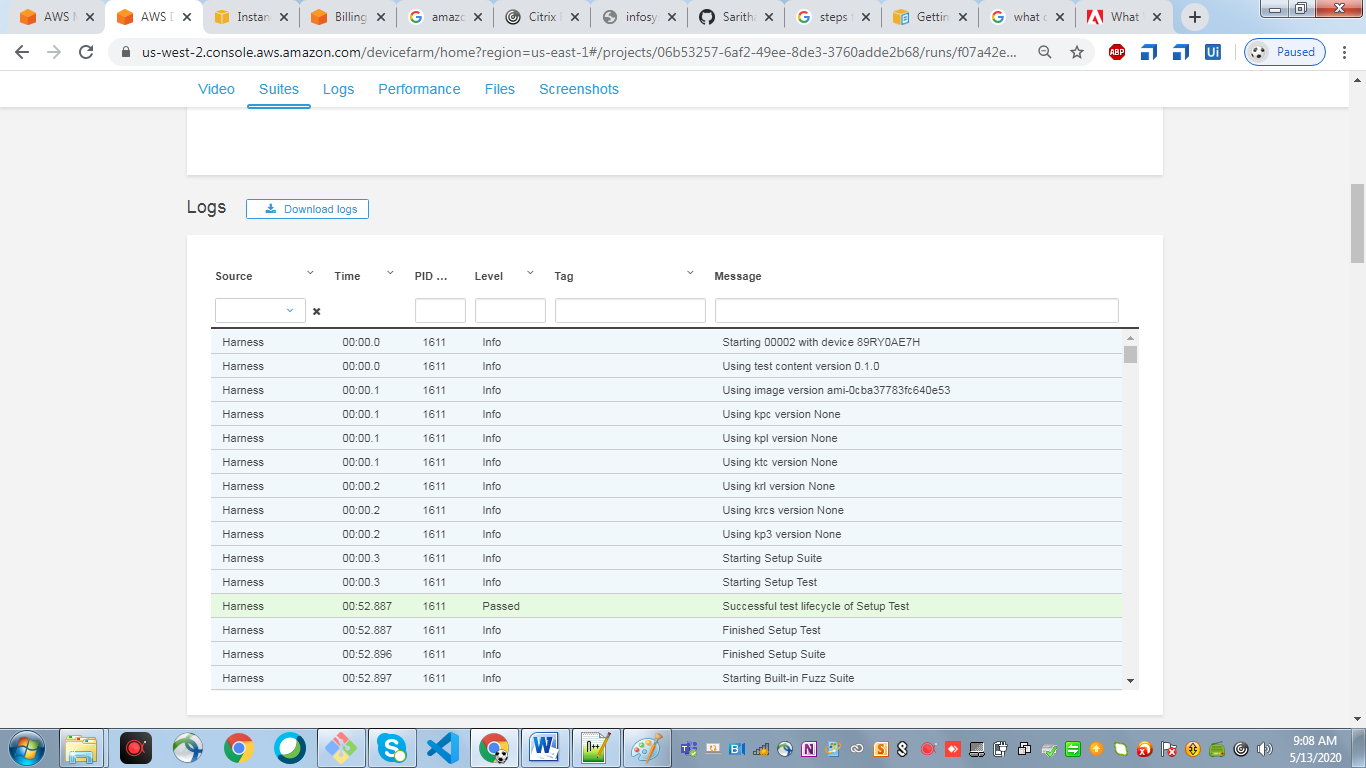
* + - Once the test is completed you can see the below screen





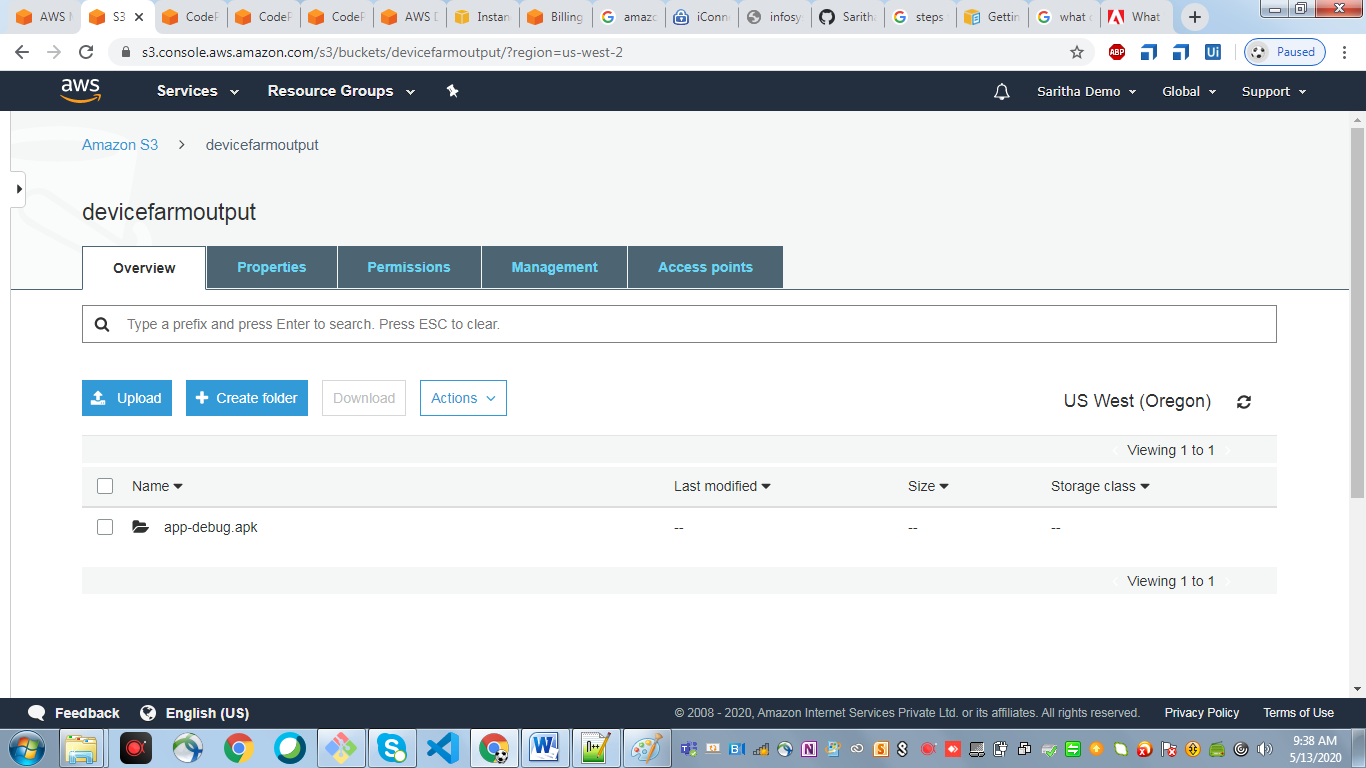




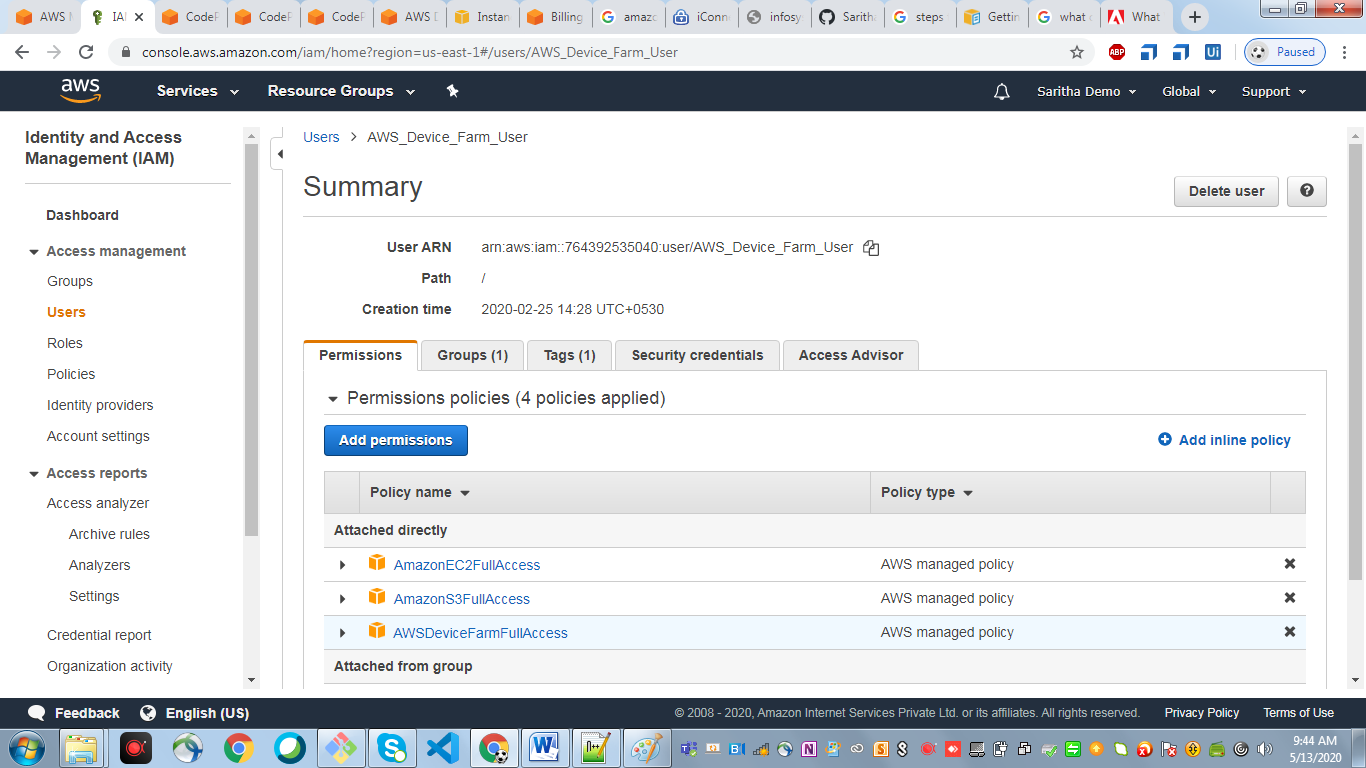


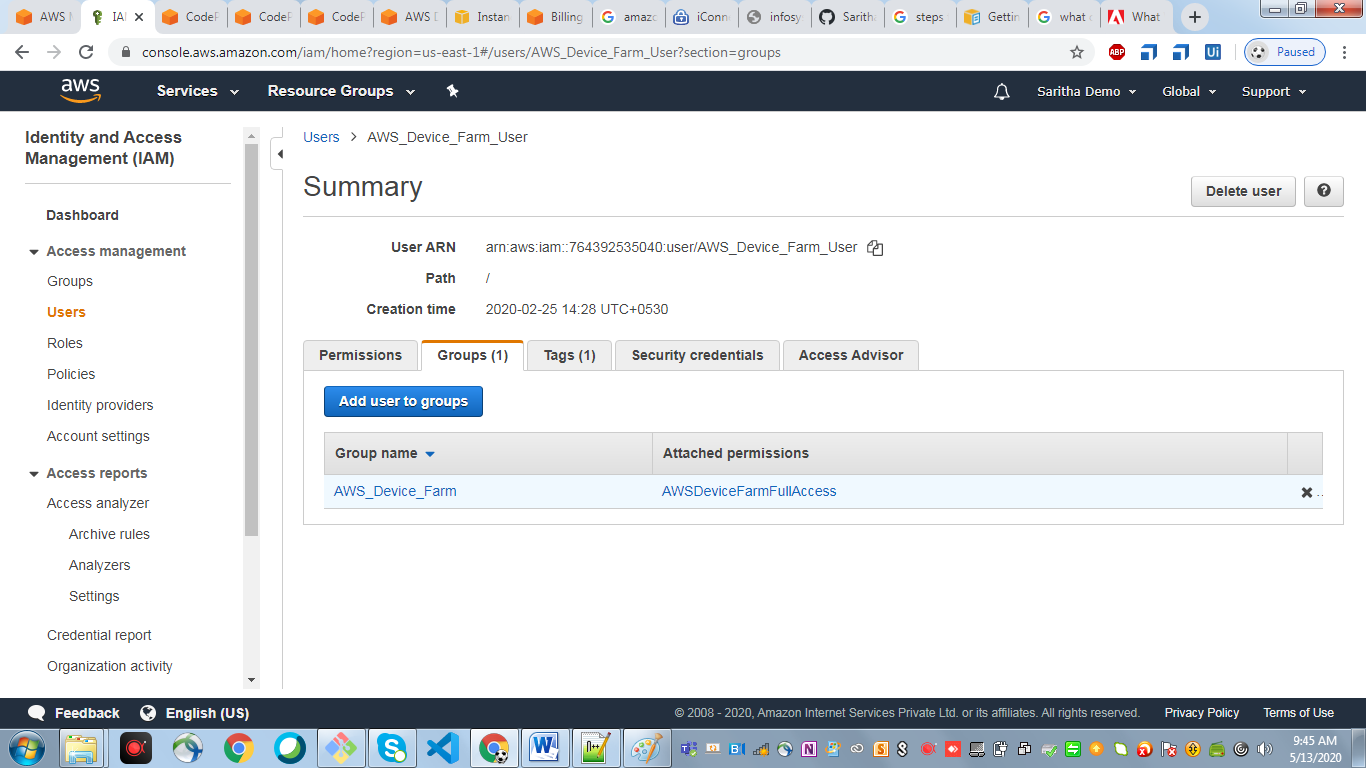
1. Go to amazon console and click on s3

Create a new bucket to deploy the artifacts



1. Create IAM user, roles and policies

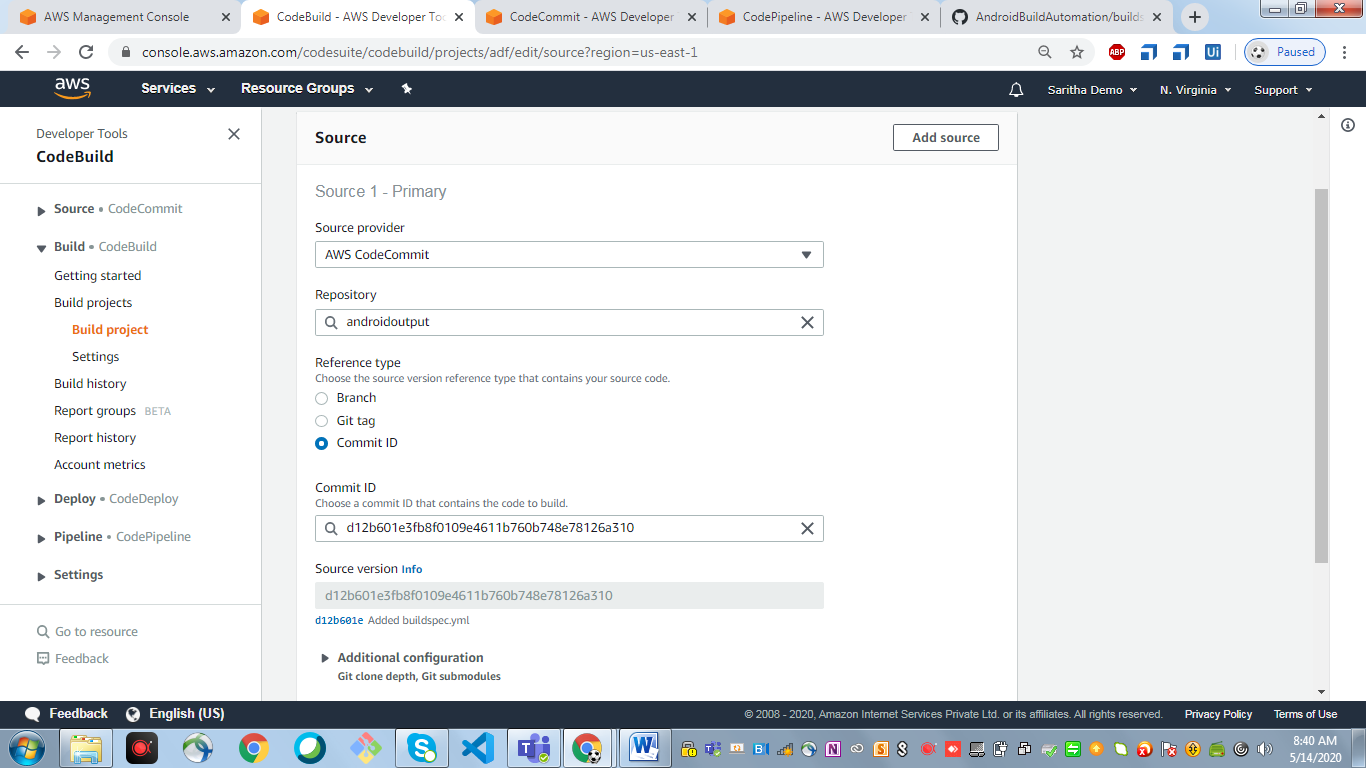




1. Go to the amazon console and click on the Code Build

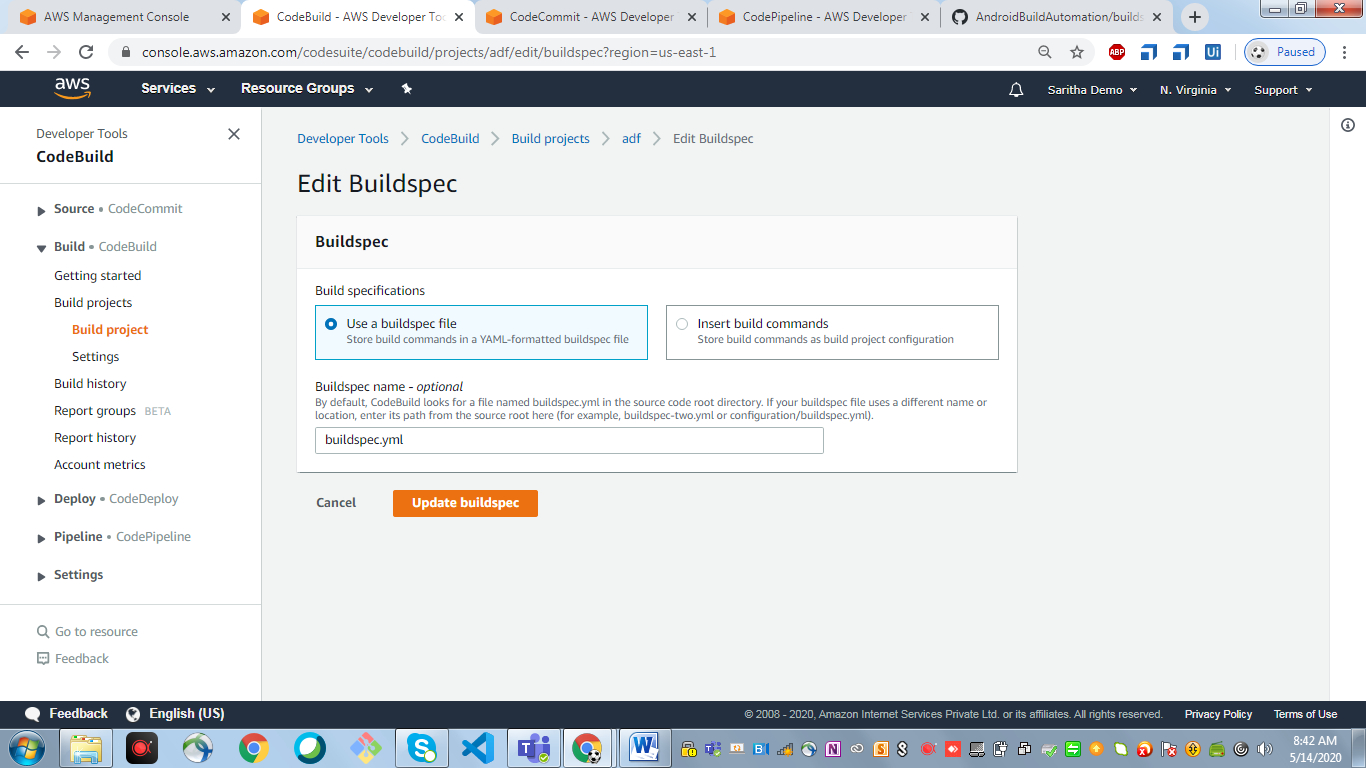
Create a build project and give the source provider as Github

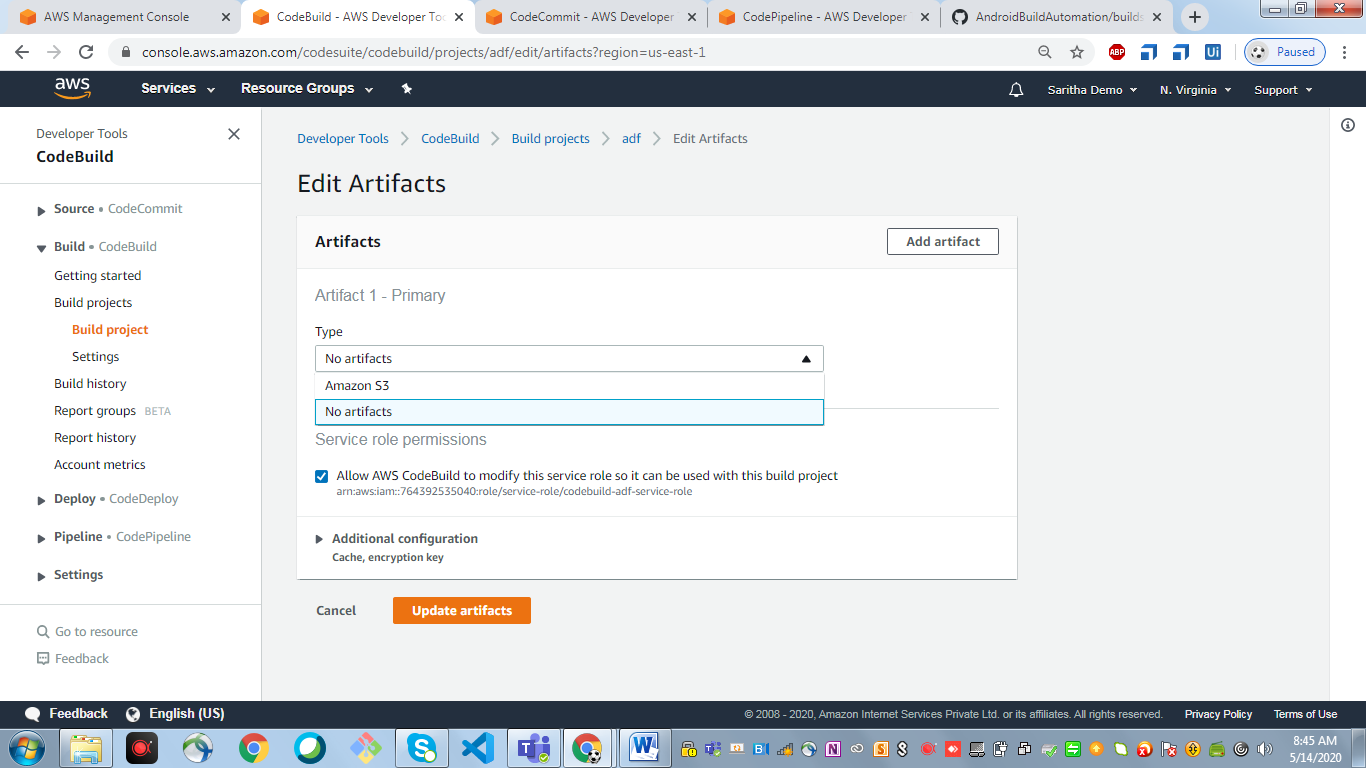


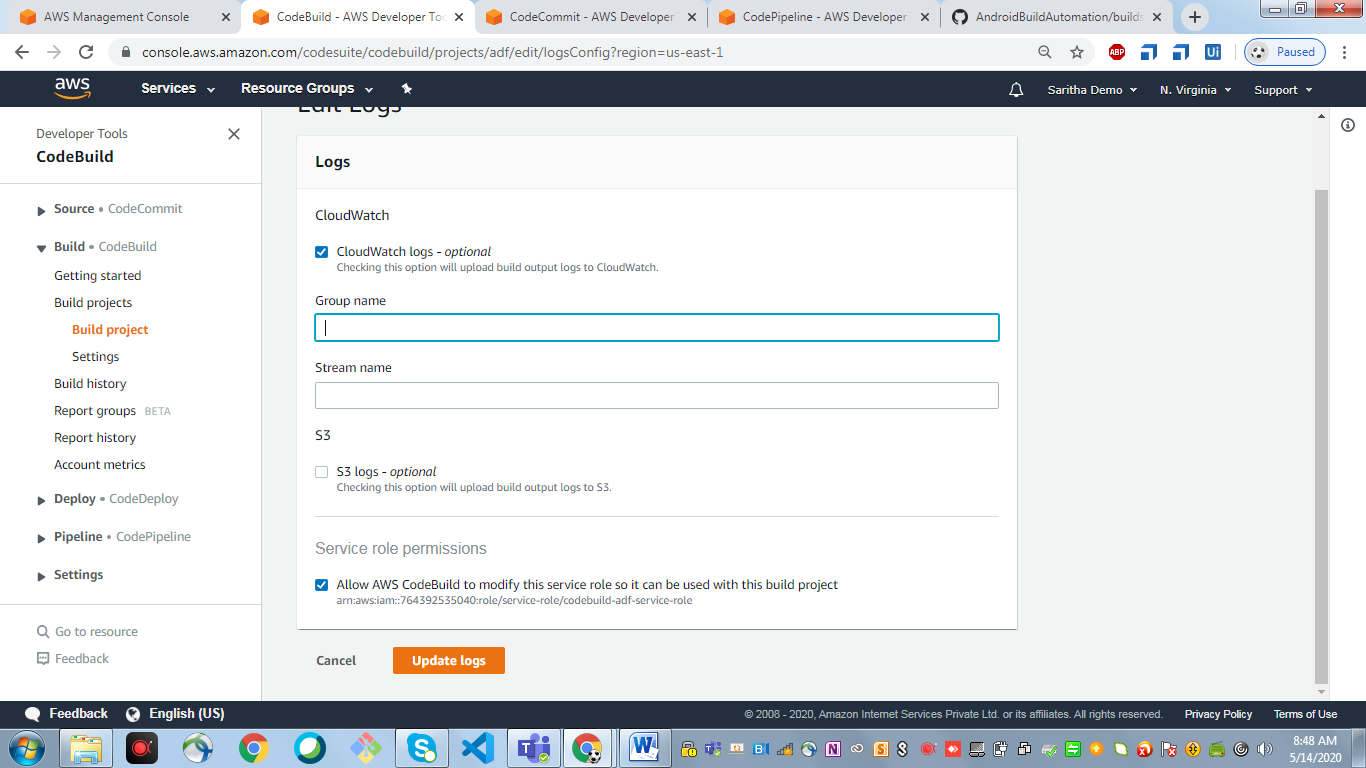


add buildspec.yml file to build the android app. Please find the file in below path

<https://github.com/SarithaRamineni/AndroidBuildAutomation/blob/master/buildspec.yml>

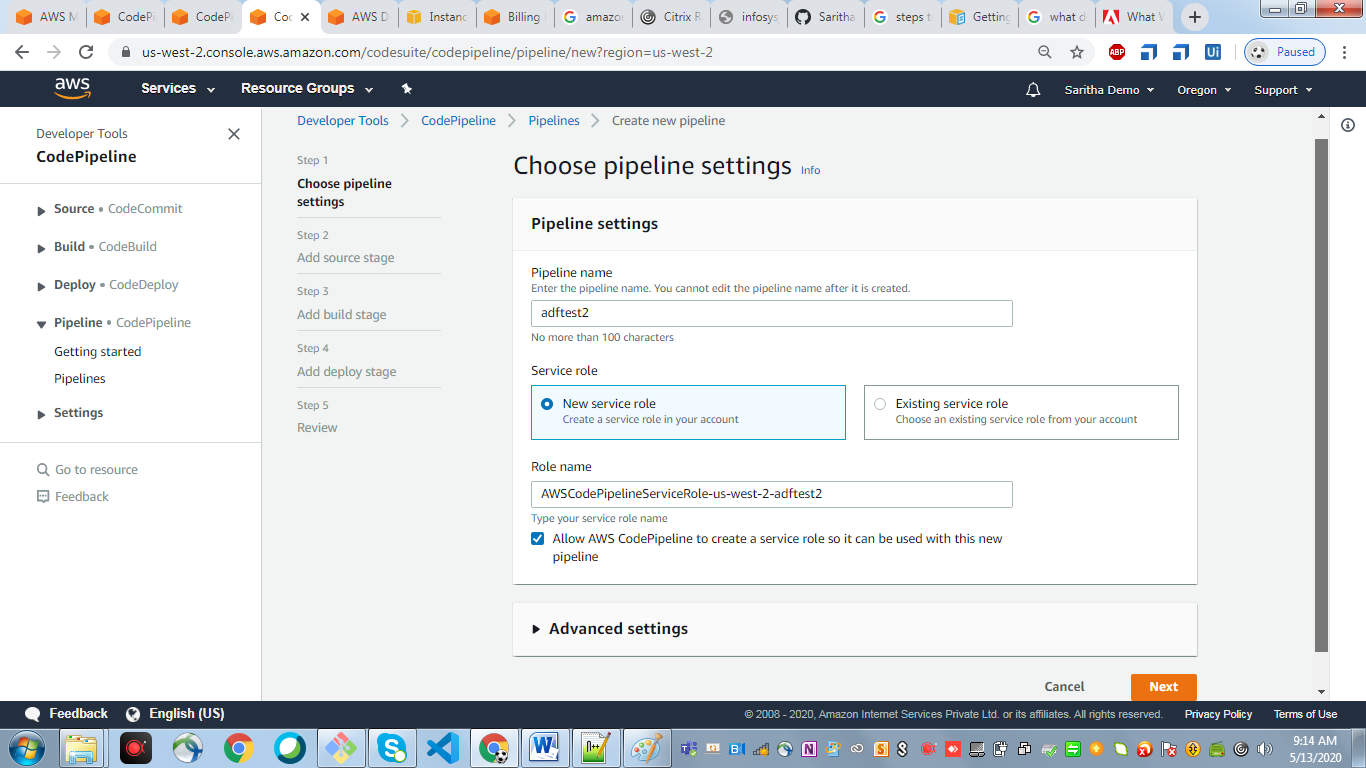


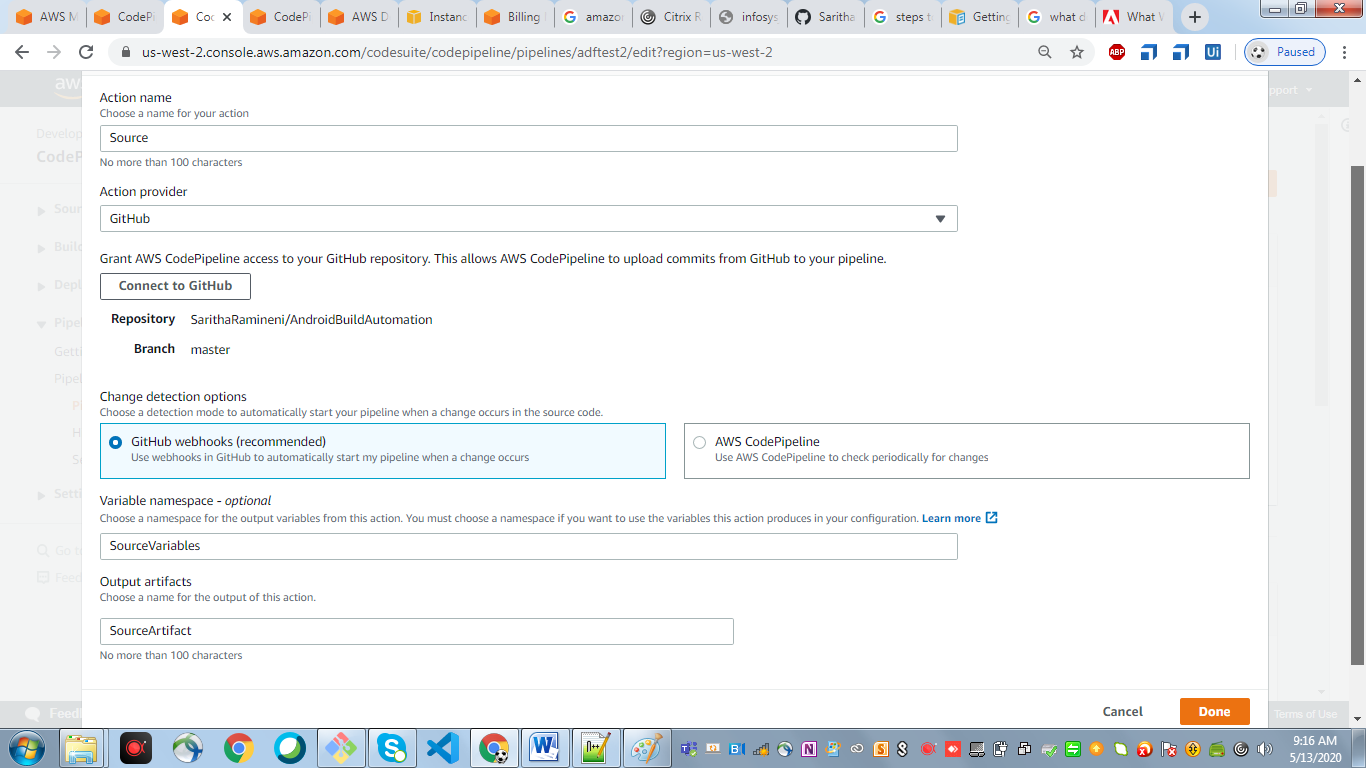




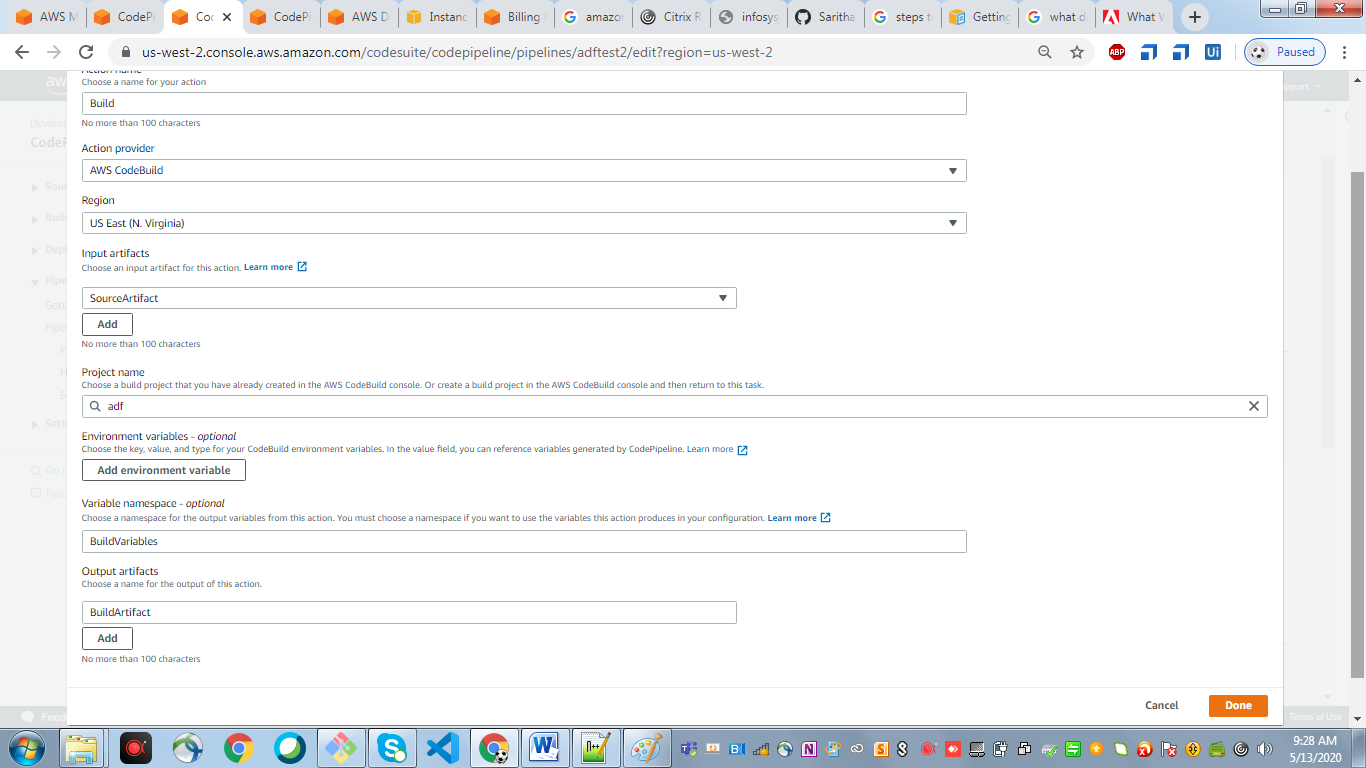
1. Go to the amazon console and click on the Code pipeline

Create new pipeline and give the below details

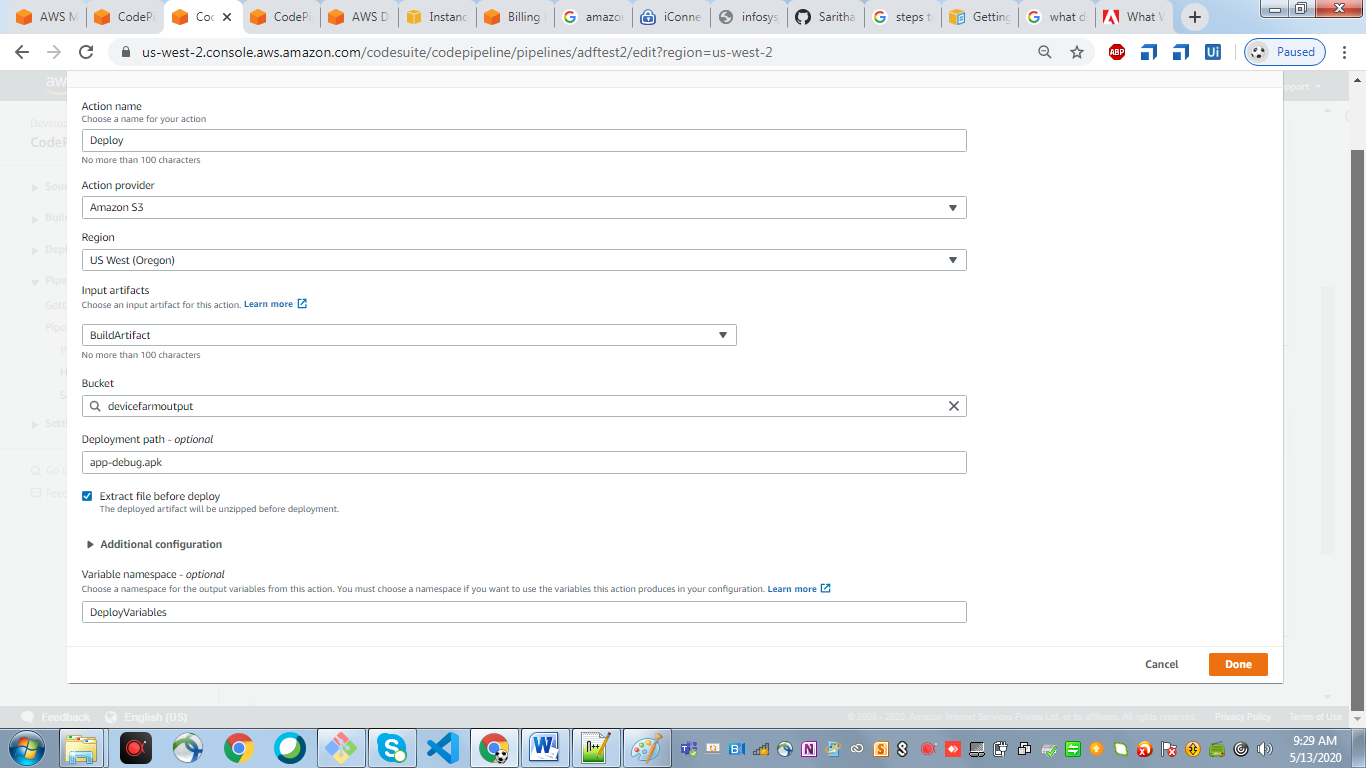


Add Source Stage

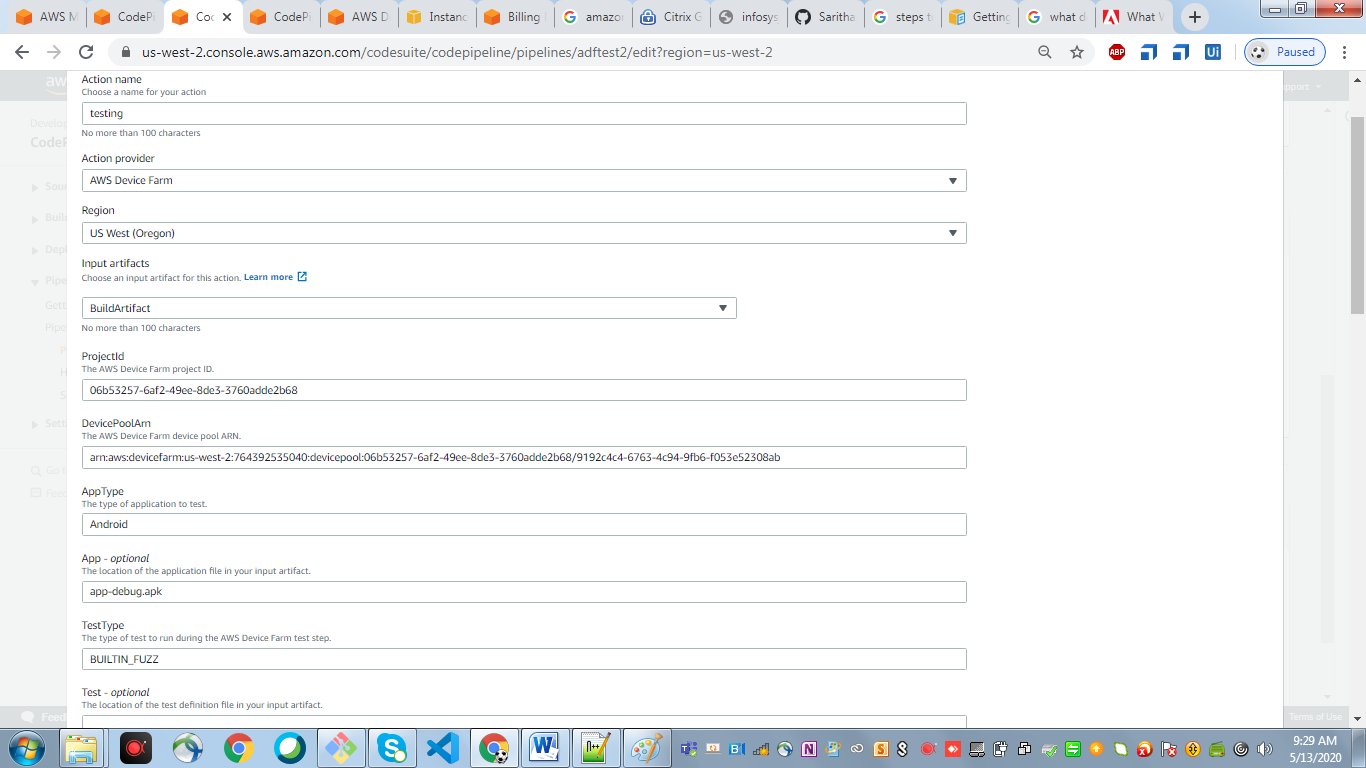
Add Build Stage



Add Deploy Stage

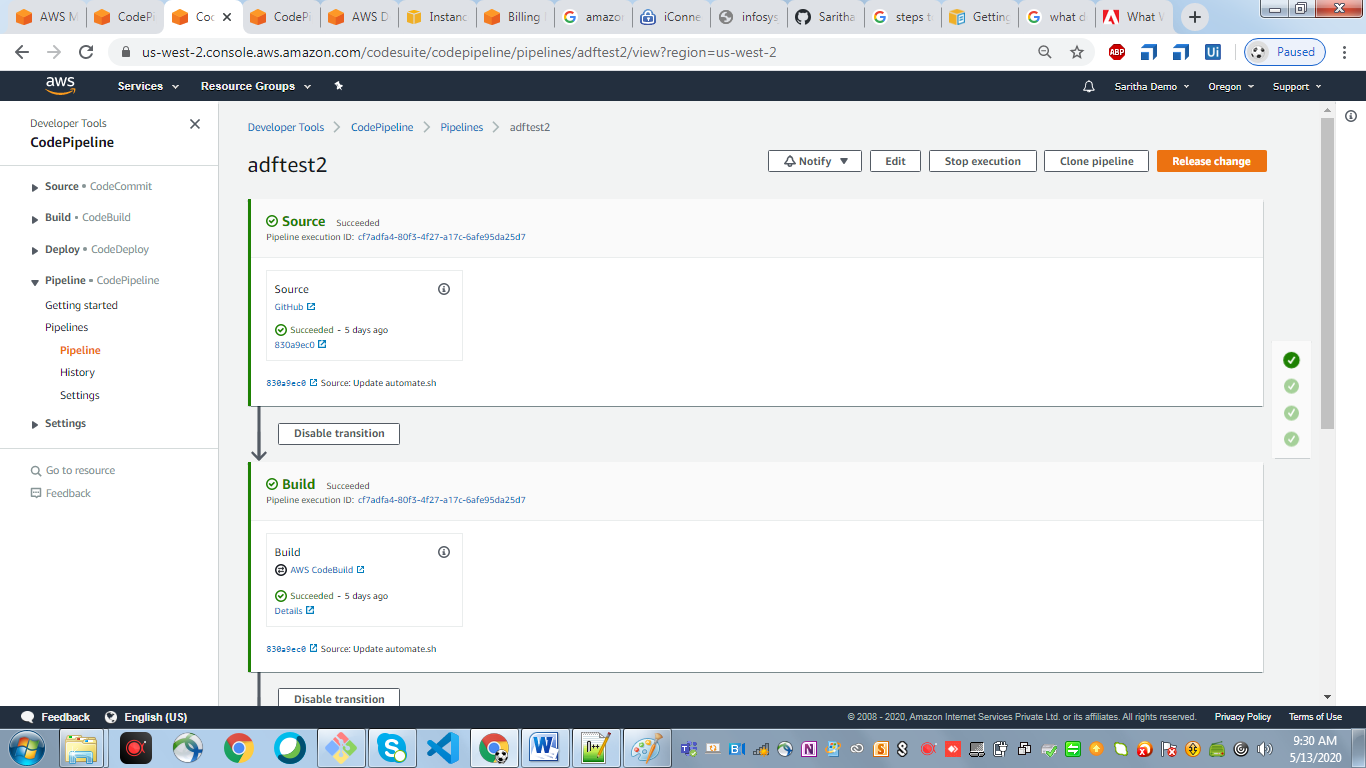


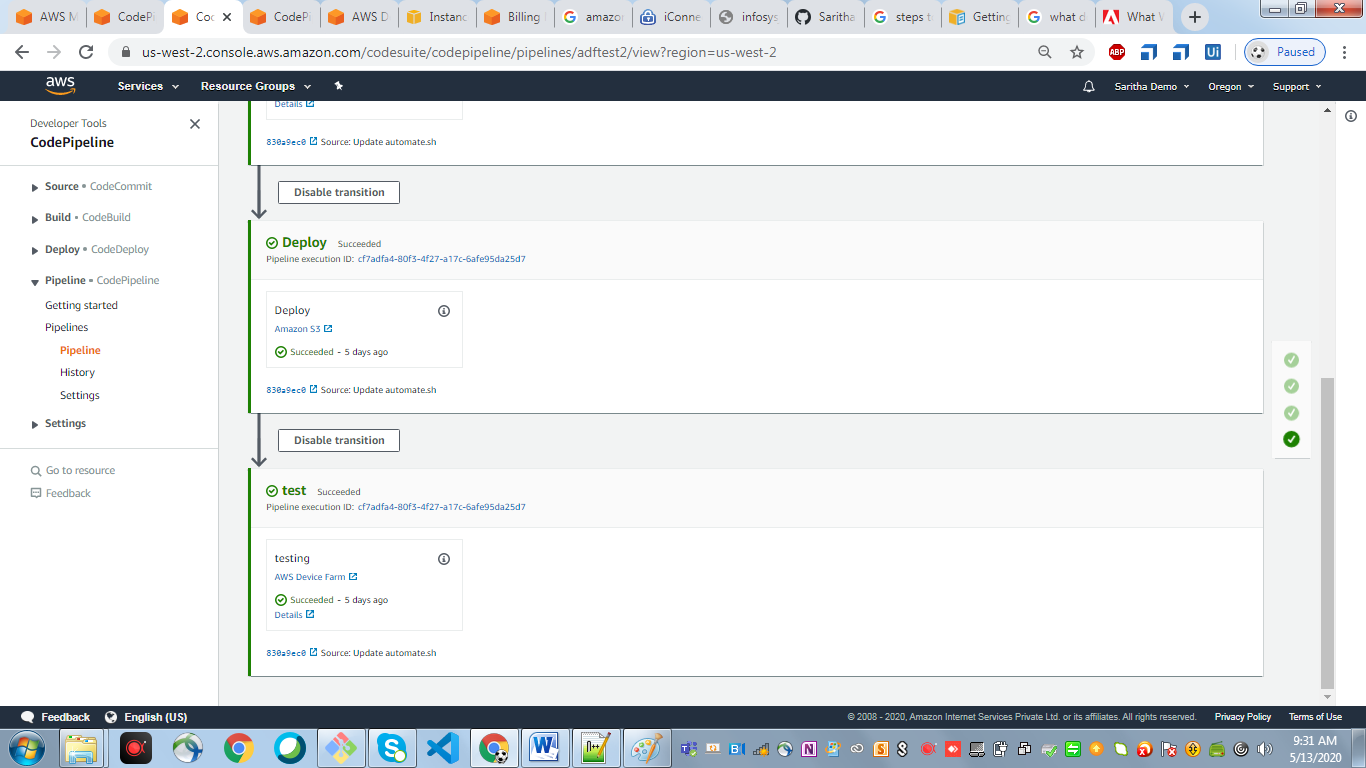
Add Test Stage



Once the pipeline is created then make some changes in any one of the code files and see whether the pipeline is triggered

Once the pipeline is completed you can see “pipeline succeeded” as shown below





**Optional:** For manual check to build the android app

To check whether the given android app development code is working fine or not.

1. Launch AWS Ec2 (Ubuntu) instance and install java and Gradle

**Steps to launch Ec2 instance**

* Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
* Choose **Launch Instance**.
* In **Step 1: Choose an Amazon Machine Image (AMI)**, find an Amazon Linux AMI at the top of the list and choose **Select**.
* In **Step 2: Choose an Instance Type**, choose **Next: Configure Instance Details**.
* In **Step 3: Configure Instance Details**, provide the following information:
  + For **Network**, choose the entry for the same VPC that you noted when you created your EFS file system in [Step 1: Create Your Amazon EFS File System](https://docs.aws.amazon.com/efs/latest/ug/gs-step-two-create-efs-resources.html).
  + For **Subnet**, choose a default subnet in any Availability Zone.
  + For **File systems**, make sure that the EFS file system that you created in [Step 1: Create Your Amazon EFS File System](https://docs.aws.amazon.com/efs/latest/ug/gs-step-two-create-efs-resources.html) is selected. The path shown next to the file system ID is the mount point that the EC2 instance will use, which you can change. Choose **Add to user data** to mount the file system when the EC2 is launched.
  + Under **Advanced Details**, confirm that the user data is present in **User data**.
* Choose **Next: Add Storage**.
* Choose **Next: Add Tags**.
* Name your instance and choose **Next: Configure Security Group**.
* In **Step 6: Configure Security Group**, set **Assign a security group** to **Select an existing security group**. Choose the default security group to make sure that it can access your EFS file system.
* You can't access your EC2 instance by Secure Shell (SSH) using this security group. SSH access isn't required for this exercise. To add access by SSH later, you can edit the default security and add a rule to allow SSH. Or you can create a new security group that allows SSH. You can use the following settings to add SSH access:
  + **Type:** SSH
  + **Protocol:** TCP
  + **Port Range:** 22
  + **Source:** Anywhere 0.0.0.0/0
* Choose **Review and Launch**.
* Choose **Launch**.
* Select the check box for the key pair that you created, and then choose **Launch Instances**.

1. To Build the Android app we need Gradle

Grdale Installation Steps

**Installing Gradle:**

**Prerequisites:**

Gradle runs on all major operating systems and requires only a Java Development Kit version 8 or higher to run.

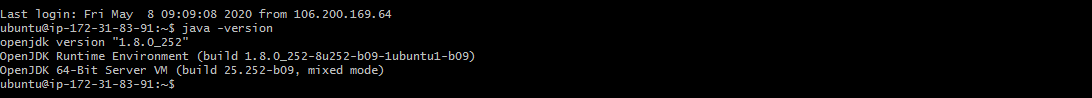
1. Install OpenJDK

$ sudo apt update

$ sudo apt install openjdk-8-jdk

– To verify that Java was successfully installed, run the following command:

$ java -version



2.Open terminal and execute the below command to install Gradle

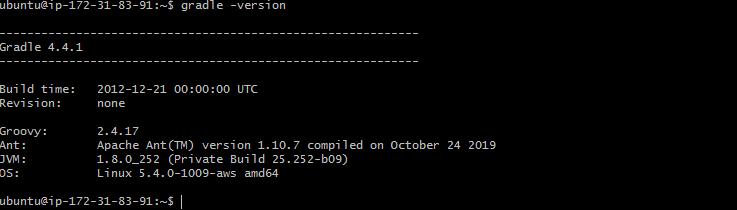
$ sudo apt-get install gradle

$ sudo wget https://services.gradle.org/distributions/gradle-5.2.1-bin.zip -P /tmp

$ sudo unzip -d /opt/gradle /tmp/gradle-\*.zip

– To verify that Gradle was successfully installed, run the following command:

$ gradle -version



3. Setup environment variables

$ sudo vi /etc/profile.d/gradle.sh

GRADLE\_HOME=/opt/gradle/gradle-5.2.1

PATH=${GRADLE\_HOME}/bin:${PATH}

export GRADLE\_HOME

export PATH

$ sudo chmod +x /etc/profile.d/gradle.sh

**Run the below commands to get the Android Emulator running on EC2**

$ sudo apt update

$ sudo apt install openjdk-8-jre unzip

$ wget https://dl.google.com/android/repository/sdk-tools-linux-4333796.zip

$ unzip sdk-tools-linux-4333796.zip -d android-sdk

$ sudo mv android-sdk /opt/

$ export ANDROID\_SDK\_ROOT=/opt/android-sdk

$ echo "export ANDROID\_SDK\_ROOT=/opt/android-sdk" >> ~/.bashrc

$ echo "export PATH=$PATH:$ANDROID\_SDK\_ROOT/tools" >> ~/.bashrc

$ source ~/.bashrc

$ cd /opt/android-sdk/tools/bin

$ ./sdkmanager --update

$ ./sdkmanager --licenses

$ ./sdkmanager "system-images;android-25;google\_apis;armeabi-v7a" "emulator" "platform-tools"

$ touch ~ubuntu/.android/repositories.cfg

$ mkdir /opt/android-sdk/platforms

$ /opt/android-sdk/tools/bin/avdmanager -v create avd -f -n MyAVD -k "system-images;android-27;google\_apis;armeabi-v7a" -p "/opt/android-sdk/avd"

1. Copy the source code from github and go to Project root directory and run the below commands to build the Android app

$ git clone https://github.com/SarithaRamineni/AndroidBuildAutomation.git

$ cd $PROJECT\_DIR (ex: cd AndroidBuildAutomation)

$ chmod 777 gradlew

- "chmod +x ./gradlew"

- "echo \"Build process start for Android application\""

- "echo \"Cleaning..\""

- "./gradlew clean"

- "echo \"cleanBuildCache..\""

- "./gradlew cleanBuildCache"

- "echo \"Build .. \""

- "./gradlew build"

- "echo \"assembleDebug..\""

- "./gradlew assembleDebug"

All the above build steps are now available in buildspec.yml in below path.

<https://github.com/SarithaRamineni/AndroidBuildAutomation/blob/master/buildspec.yml>

$ ./gradlew clean

|  |
| --- |
| ubuntu@ip-172-31-83-91:~/AndroidBuildAutomation$ ./gradlew clean  Starting a Gradle Daemon (subsequent builds will be faster)    BUILD SUCCESSFUL in 14s  2 actionable tasks: 2 executed |

$ ./gradlew cleanBuildCache

ubuntu@ip-172-31-83-91:~/AndroidBuildAutomation$ ./gradlew cleanBuildCache

BUILD SUCCESSFUL in 1s

1 actionable task: 1 executed

$ ./gradlew build

ubuntu@ip-172-31-83-91:~/AndroidBuildAutomation$ ./gradlew build

> Task :app:lint

Ran lint on variant debug: 4 issues found

Ran lint on variant release: 4 issues found

Wrote HTML report to file:///home/ubuntu/AndroidBuildAutomation/app/build/reports/lint-results.html

Wrote XML report to file:///home/ubuntu/AndroidBuildAutomation/app/build/reports/lint-results.xml

Deprecated Gradle features were used in this build, making it incompatible with Gradle 6.0.

Use '--warning-mode all' to show the individual deprecation warnings.

See https://docs.gradle.org/5.1.1/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 48s

$ ./gradlew assembleDebug

ubuntu@ip-172-31-83-91:~/AndroidBuildAutomation$ ./gradlew assembleDebug

Deprecated Gradle features were used in this build, making it incompatible with Gradle 6.0.

Use '--warning-mode all' to show the individual deprecation warnings.

See https://docs.gradle.org/5.1.1/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 3s

Now check whether the .**apk** file is created in the below path

/home/ubuntu/AndroidBuildAutomation/app/build/outputs/apk/debug

ubuntu@ip-172-31-83-91:~/AndroidBuildAutomation/app/build/outputs/apk/debug$ ls -l

total 2300

-rw-rw-r-- 1 ubuntu ubuntu 2347501 May 13 02:28 app-debug.apk

-rw-rw-r-- 1 ubuntu ubuntu 226 May 13 02:28 output.json

**Important Notes:**

1. Amazon DeviceFarm will work only in us-west-2 region.
2. To build the android app using gradle we must need to mention the gradle build steps in the buildspec.yml file.

Ex: <https://github.com/SarithaRamineni/AndroidBuildAutomation/blob/master/buildspec.yml>

1. Please follow the above optional steps to check whether the given mobile app code is working fine or not.