**Here are the steps you’ll need to follow in order to setup and AWS EC2 RabbitMQ Broker**:

Register with Amazon Web Services (AWS)

Generate an AWS key pair

Create an AWS security group

Deploy RabbitMQ on an AWS cloud server

Log in to RabbitMQ

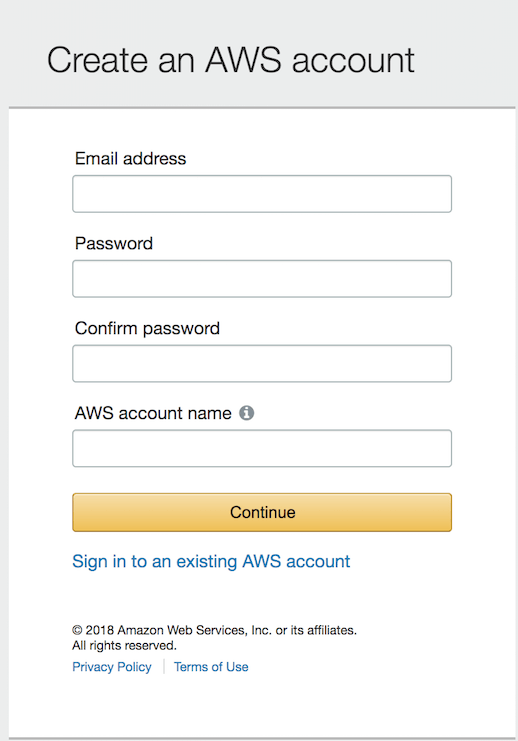
The next sections will walk you through these steps in detail.

**Step 1: Register with Amazon Web Services (AWS)**

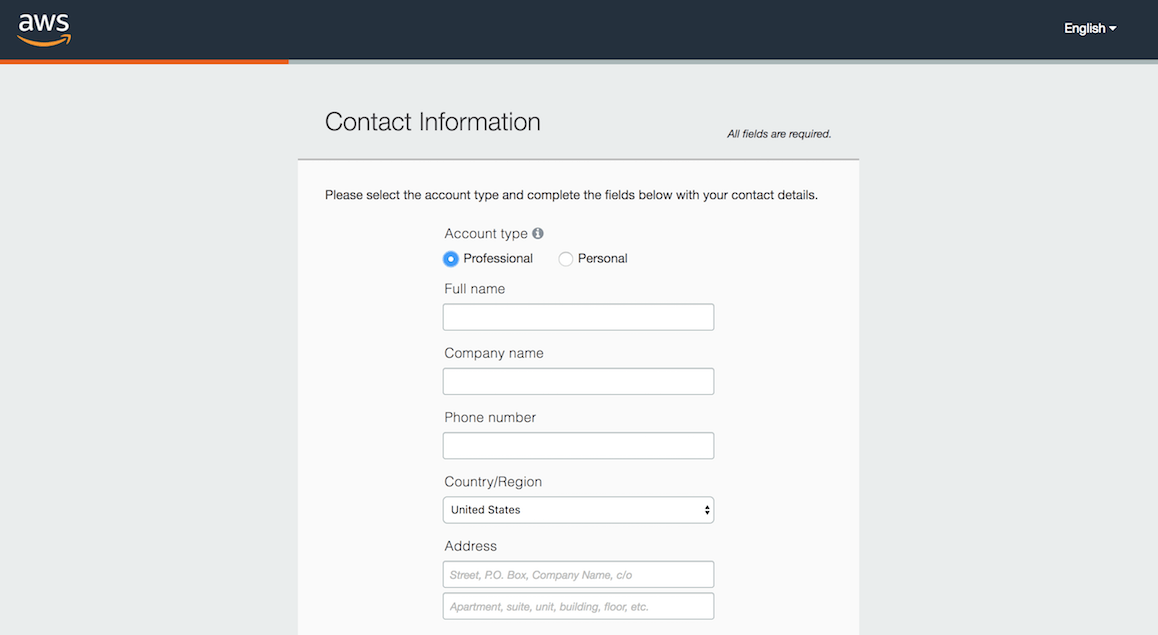
You will need an Amazon account to log in and sign up. If you already have an Amazon Web Services account obviously just skip this step.

To create it, follow these steps:

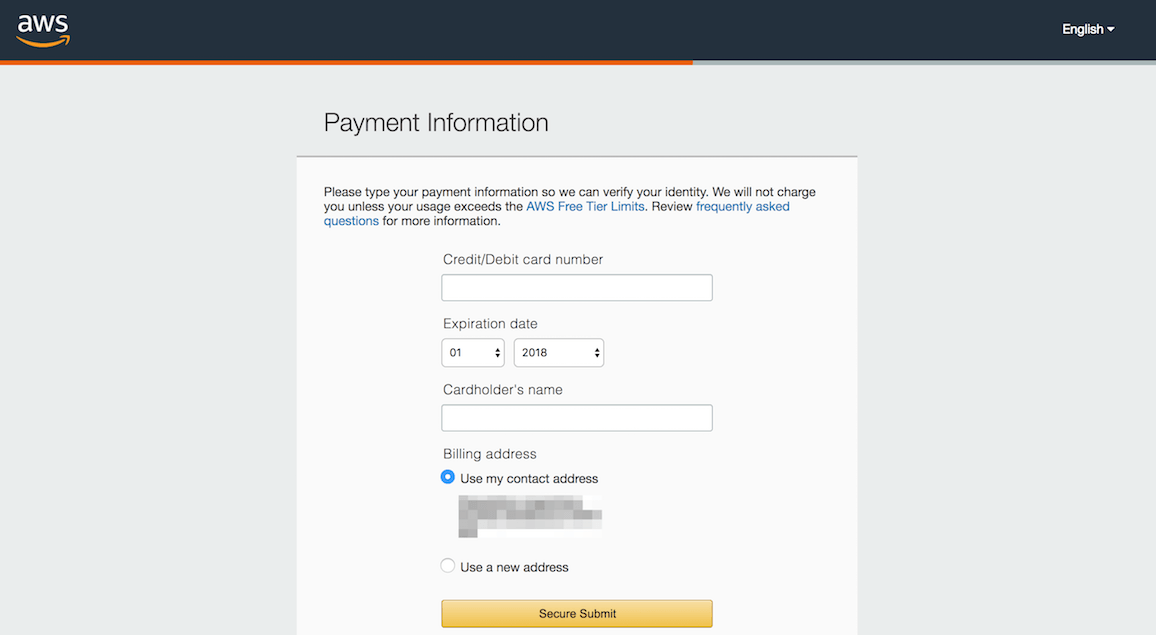
Browse to http://aws.amazon.com and click the “Create an AWS account” button at the top of the page. In the resulting page, enter an email address, a password, and an AWS account name. Then, click “Continue” to start the registration process.



Once you’ve signed into Amazon, sign up for AWS by selecting the account type and providing some basic contact information and your mobile phone number.



Once that’s done, proceed to the next stage by entering your credit card information. Click the “Secure Submit” button to continue with the account creation.



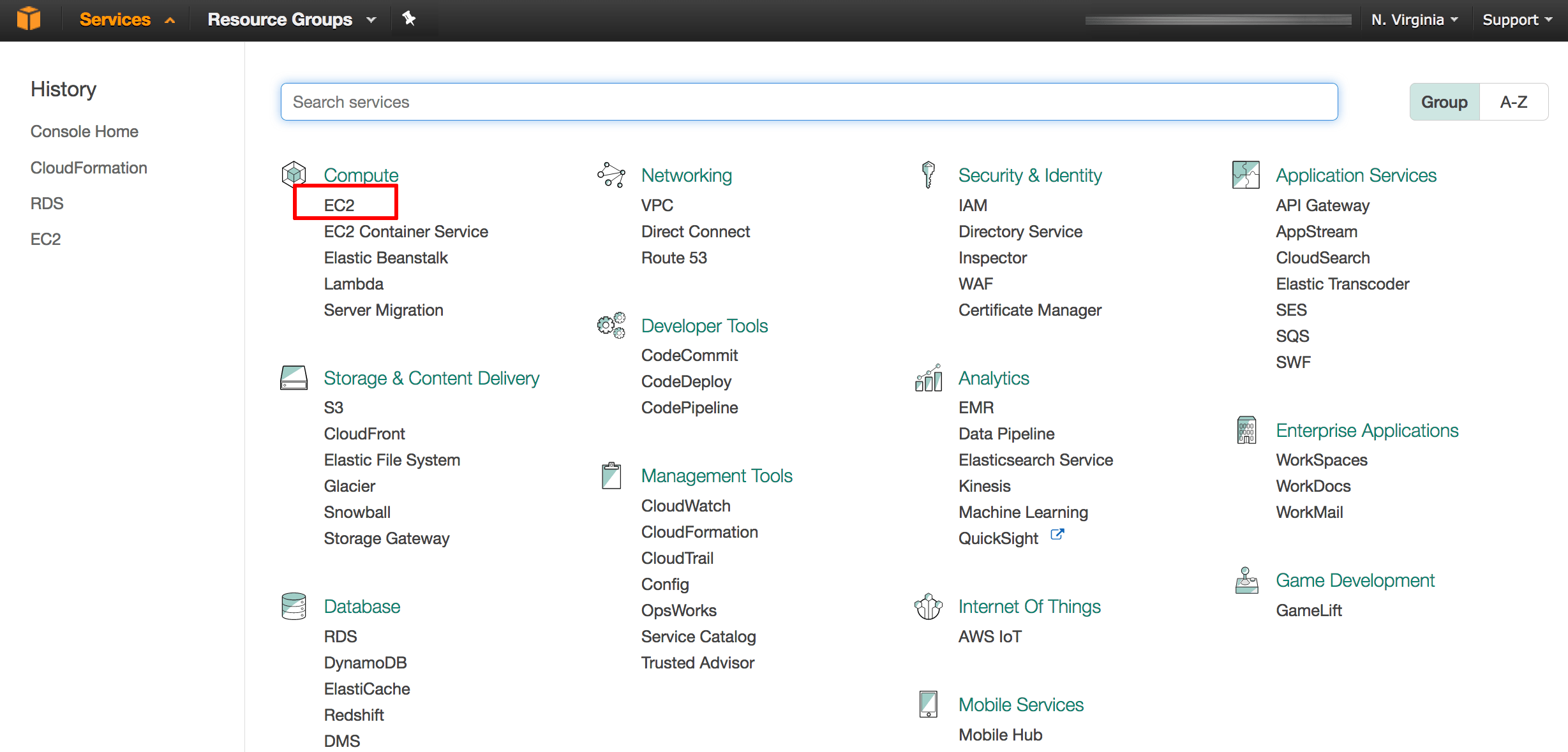
**NOTE:** When you first sign up for AWS, you get automatic access to the AWS Free Tier, which entitles you to 12 months of free usage up to certain limits. This includes 750 hours per month of free usage of Amazon EC2 micro servers, which are ideal for our Proof of Concept/Minimum Viable Product needs.

Amazon will now verify your identity, by making an automated call to your mobile phone number and prompting you to enter the PIN number displayed on the screen. Once your identity is verified, choose the “Basic” support plan (also free) and confirm your account. You will then be redirected to a welcome page, which includes a link to the AWS management console.

**Step 2: Generate an AWS Key Pair**

You will have generated an SSH key pair to access your EC2 instances. If you already have an SSH key pair for the AWS region you are operating in, skip this step. To generate an SSH key pair, which you will need to log in to your EC2 instances, follow the steps below:

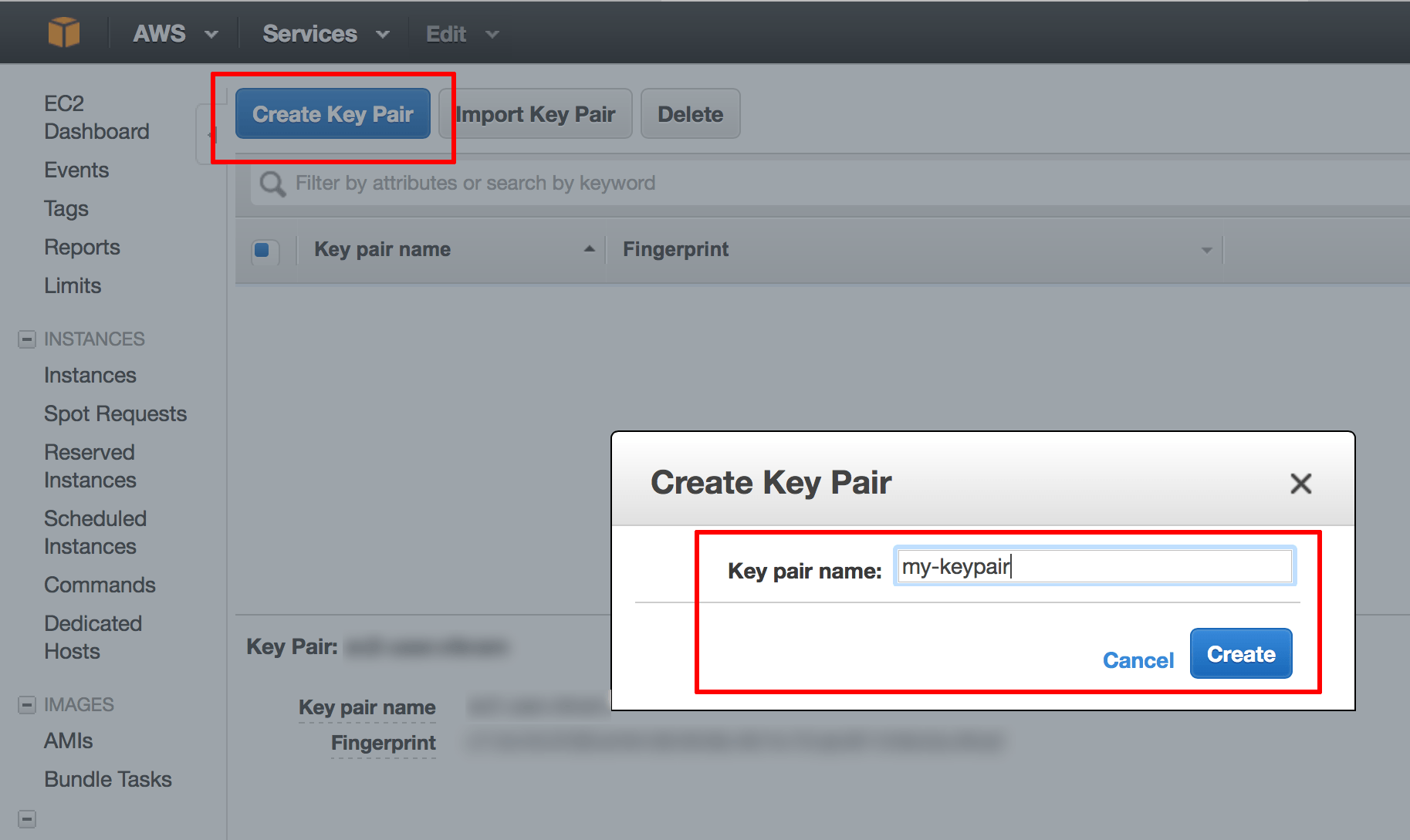
Log in to the AWS Console and from the Amazon Web Services menu, select the EC2 service.



Make sure you switch to the region where your instance will be launched using the region selector in the top right corner

From the Amazon EC2 dashboard, select the “Key Pairs” option in the “Network & Security” menu.

Click the “Create Key Pair” button. In the resulting dialog box, enter a name for the new key pair and click the “Create” button.



A new key pair, consisting of an SSH public and private key, will be generated. You will be prompted to download the private SSH key to your computer.

**NOTE:** You will only be able to download the private SSH key once. Store it safely as you will not be able to log in to your AWS servers without it.

**Step 3: Create an AWS Security Group**

You will need to create an AWS security group for your cloud server. By default, AWS cloud servers have their ports closed to secure them against external attacks. Since RabbitMQ is a Web application, it is necessary to open to the following incoming ports:

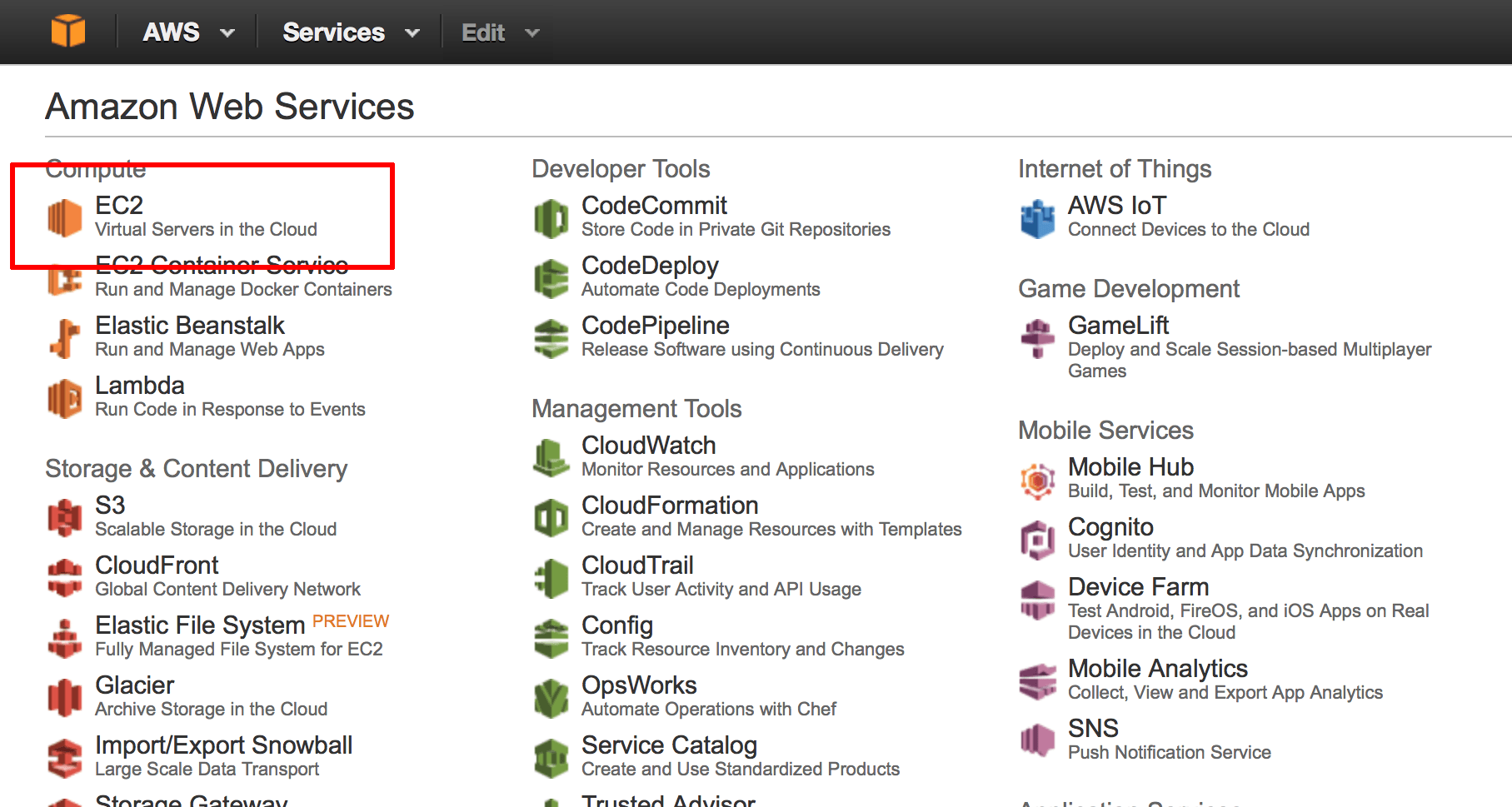
22: for ssh connections

5672: used by AMQP 0–9–1 and 1.0 clients without and with TLS

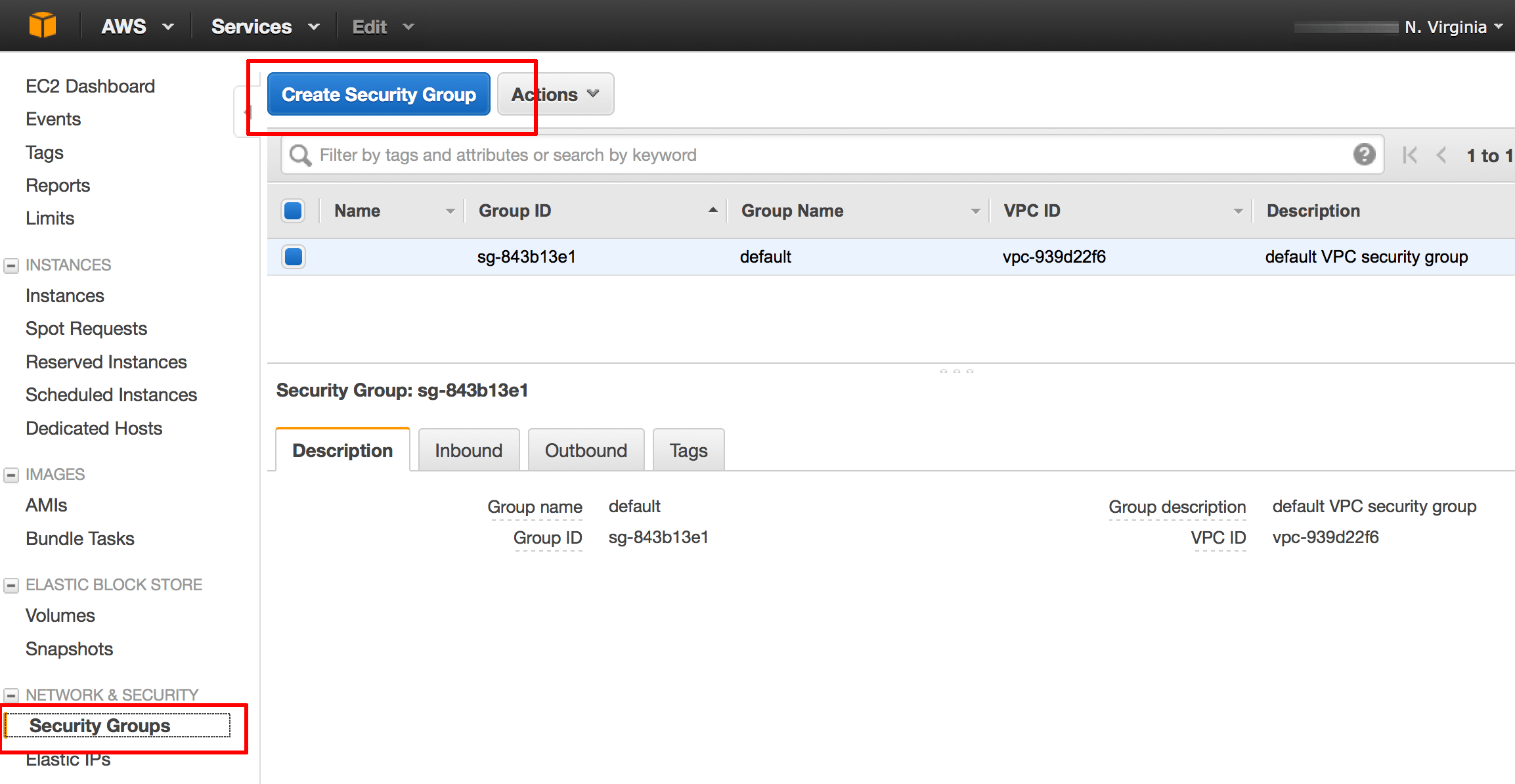
15672: HTTP API clients and rabbitmqadmin (Management Web GUI

25672: used by Erlang distribution for inter-node and CLI tools communication and is allocated from a dynamic range (limited to a single port by default, computed as AMQP port + 20000).

From the Amazon Web Services menu, select the EC2 service.

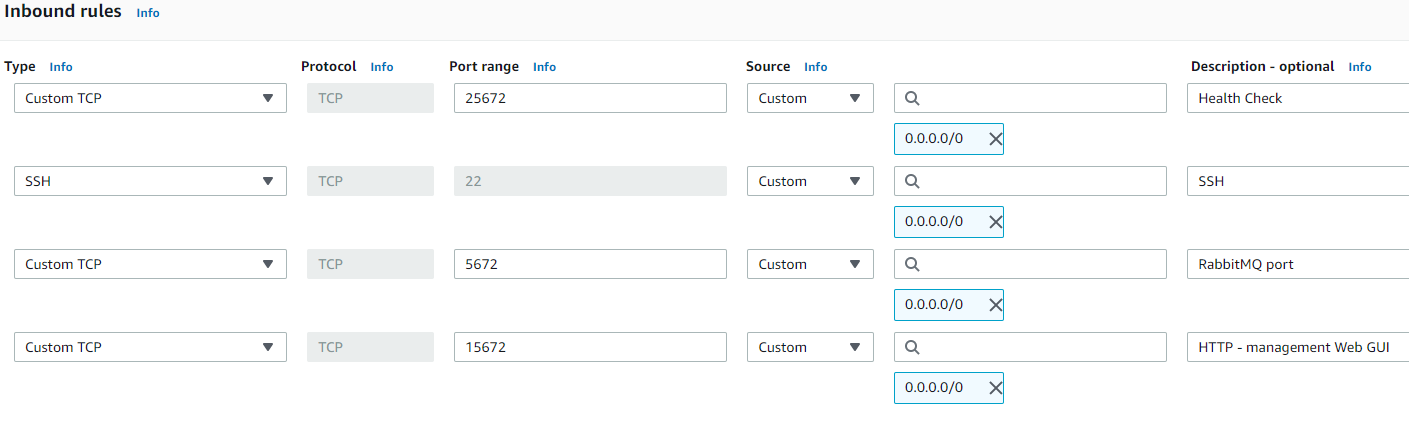


From the Amazon EC2 dashboard, select the “Security Groups” option in the “Network & Security” menu and click the “Create Security Group” button.



In the resulting dialog box, enter a name and description for the for the new security group.

Click the “Add Rule” button and add new rules 22 for SSH, 5672 for RabbitMQ, 15672 for HTTP - Management Web GUI, 25672 for Health Check using the following guidelines:



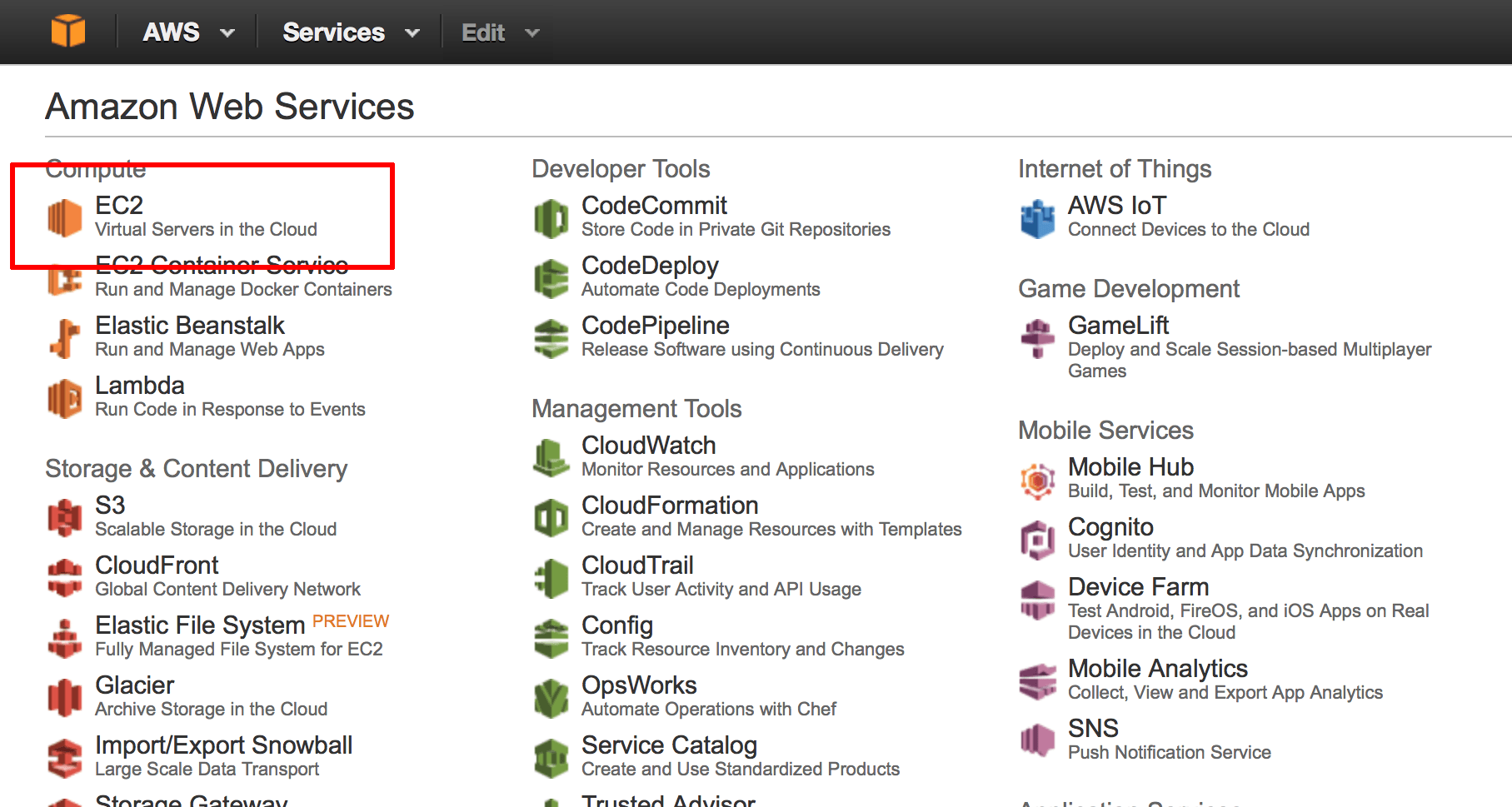
For Source: Use “Anywhere” to allow access from anywhere, or use “Custom IP” and specify an IP address range.

Click the “Create” button to save your changes.

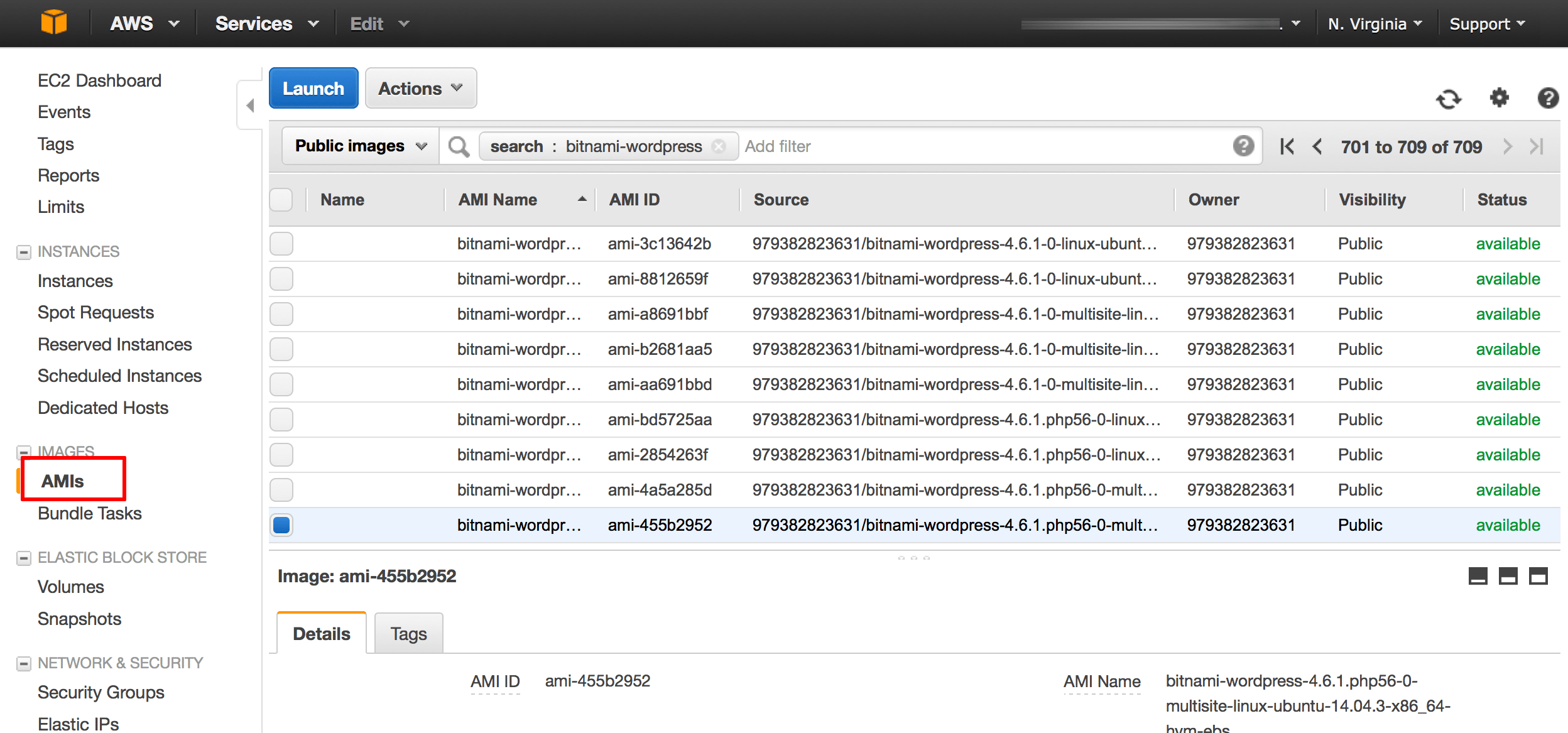
**Step 4: Deploy RabbitMQ on An AWS Cloud Server**

The next step is to launch a cloud server with the Bitnami RabbitMQ Amazon Machine Image (AMI) running on it. The AWS Console lets you do this in just a couple of clicks.

From the Amazon Web Services menu, select the EC2 service.



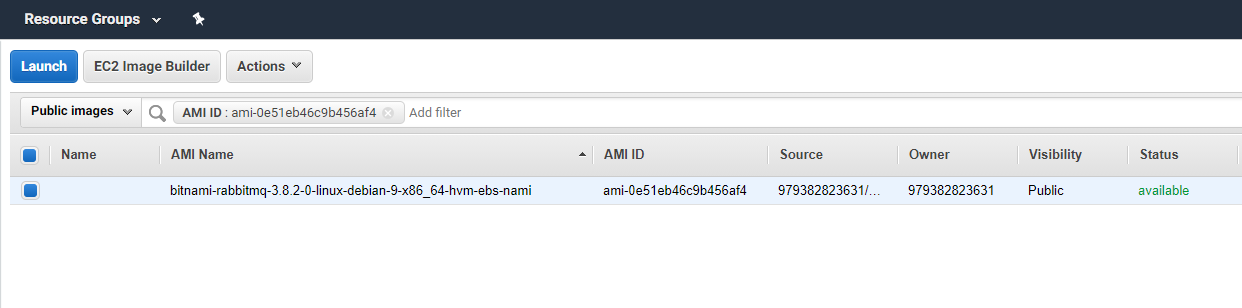
From the Amazon EC2 dashboard, select the “AMIs” option in the “Images” menu



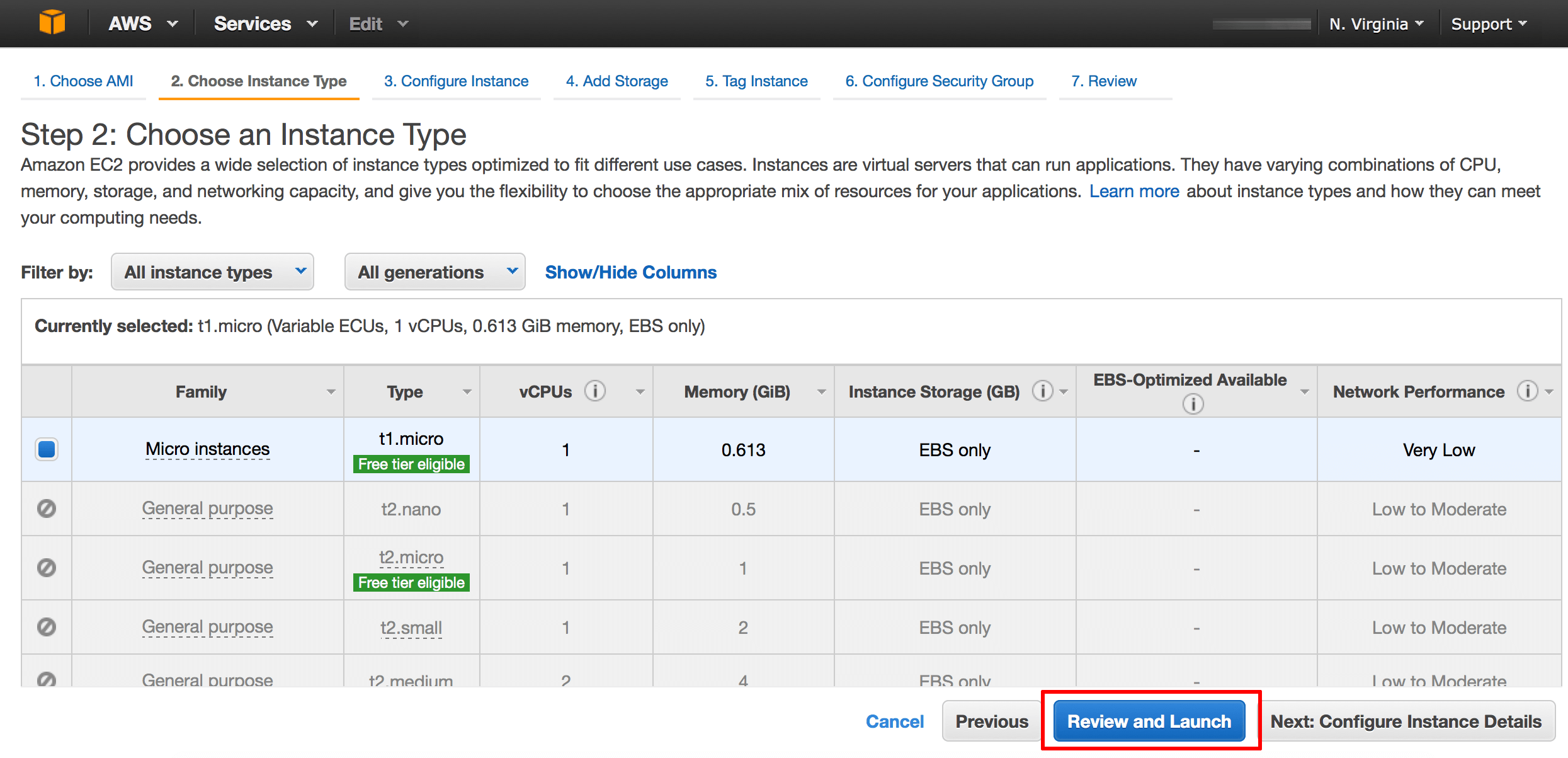
Search for the Bitnami RabbitMQ AMI by selecting “Public Images” and entering the search term “bitnami RabbitMQ” in the search bar at the top.

Although RabbitMQ is tested with most major Linux distributions, Ubuntu support for Amazon EC2 seems to be strongest, so that's the distribution we will use. For my own testing I used bitnami-rabbitmq-3.8.2-0-linux-debian-9-x86\_64-hvm-ebs-nami.

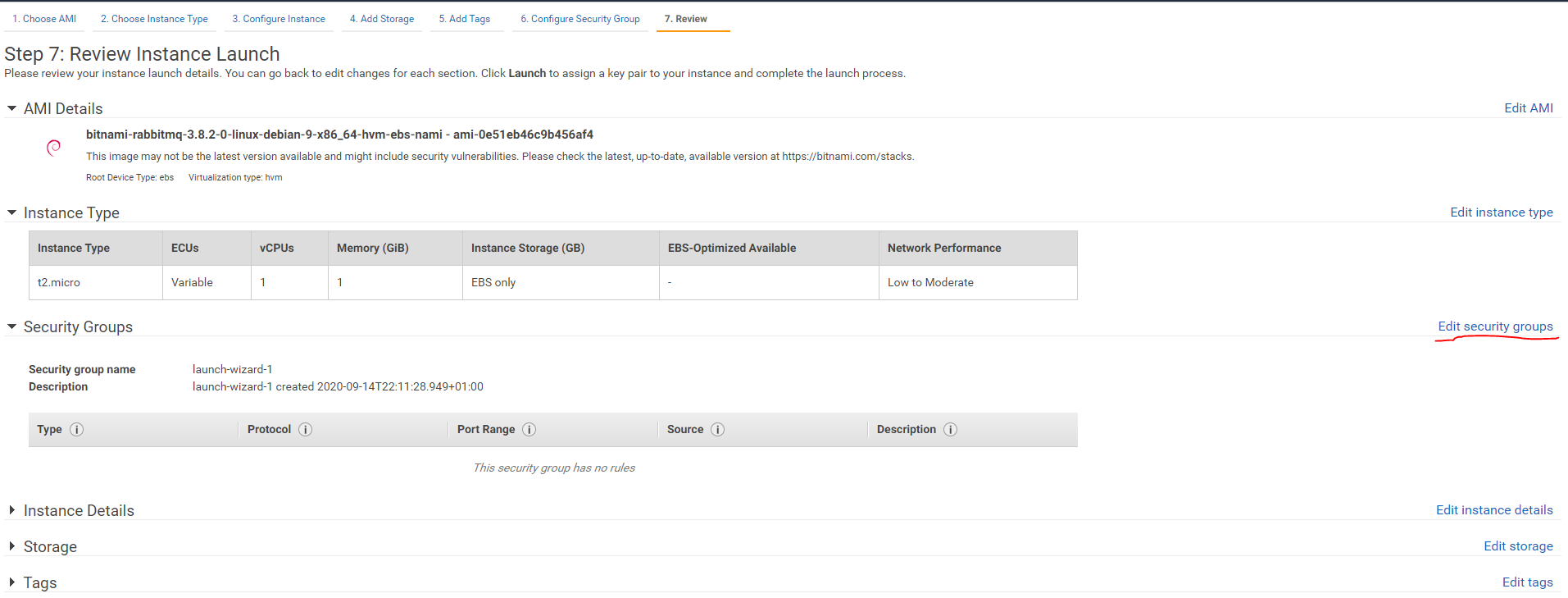
Select the image in the list of search results and click the “Launch” button.



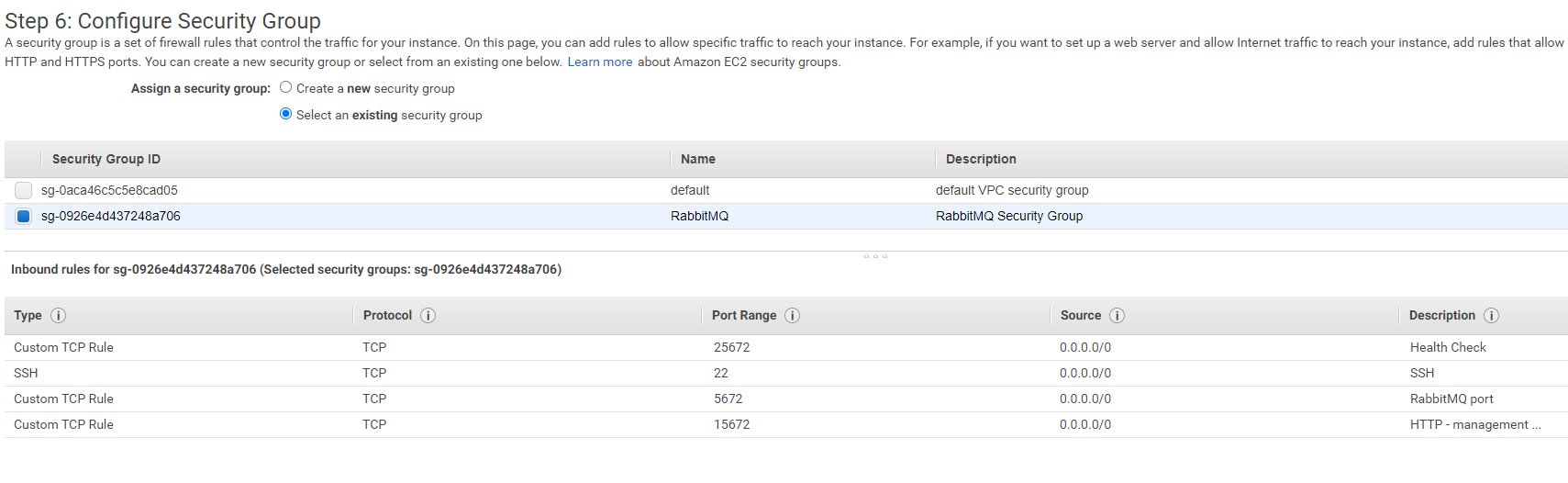
On the next screen, you will have to choose the instance type based on the load RabbitMQ is expected to handle. A “micro” server will work fine for a low-traffic. Then click Next: Configure details in the bottom.



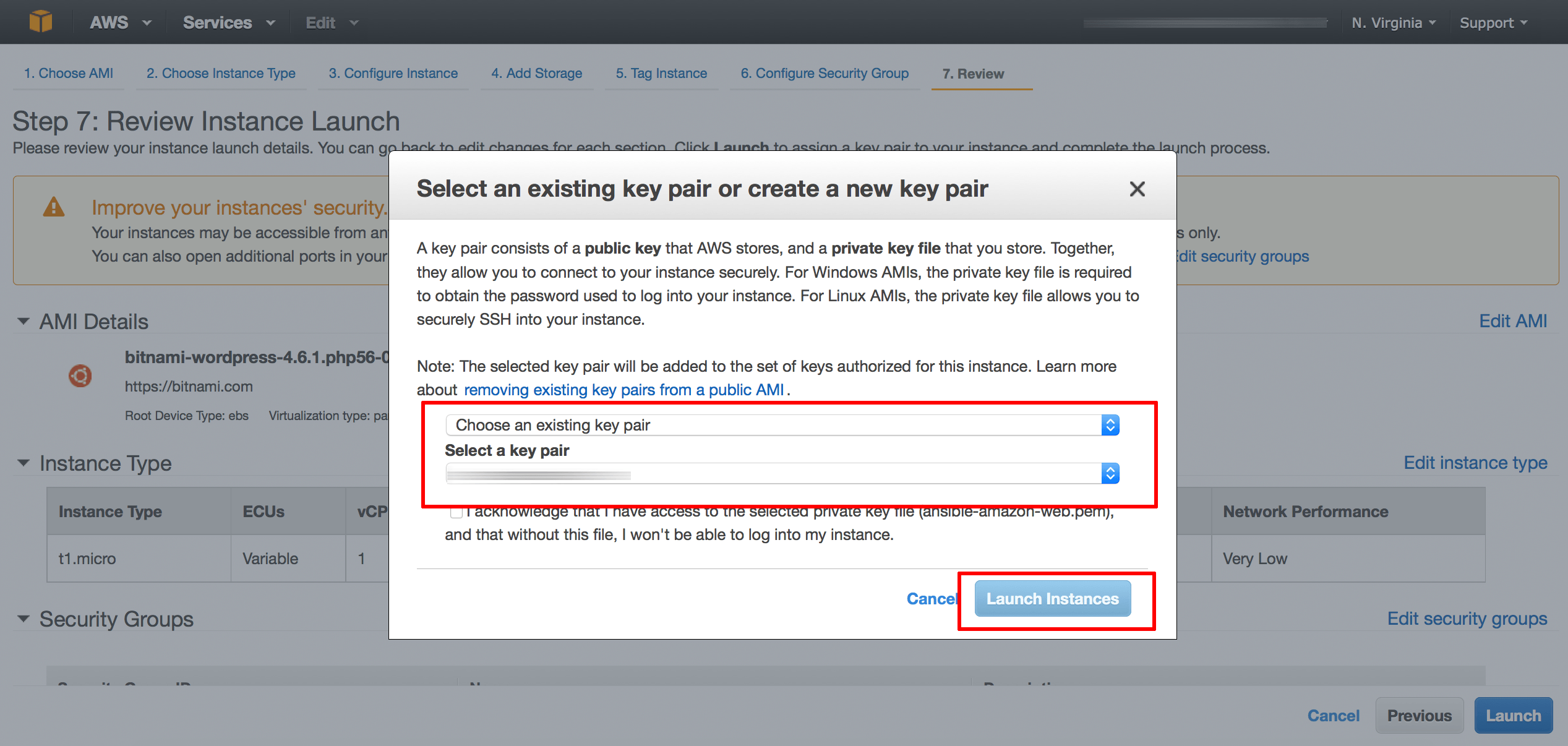
On the review page, click the “Edit security groups” link.



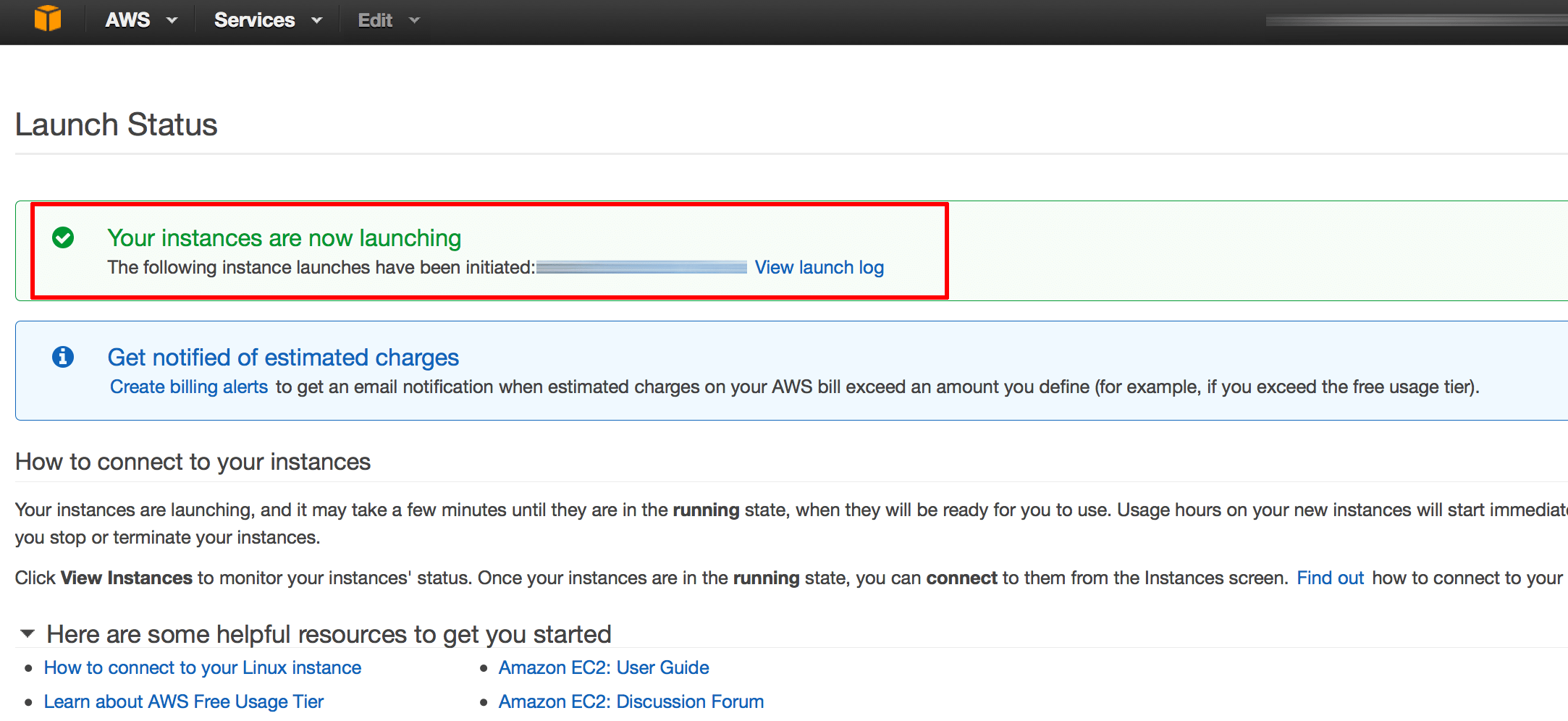
On the “Configure Security Group” page, choose the option to “Select an existing security group”. Find the security group you created earlier and select it. Click the “Review and Launch” button to proceed.



Verify that the correct key pair (created earlier) will be used for the server.



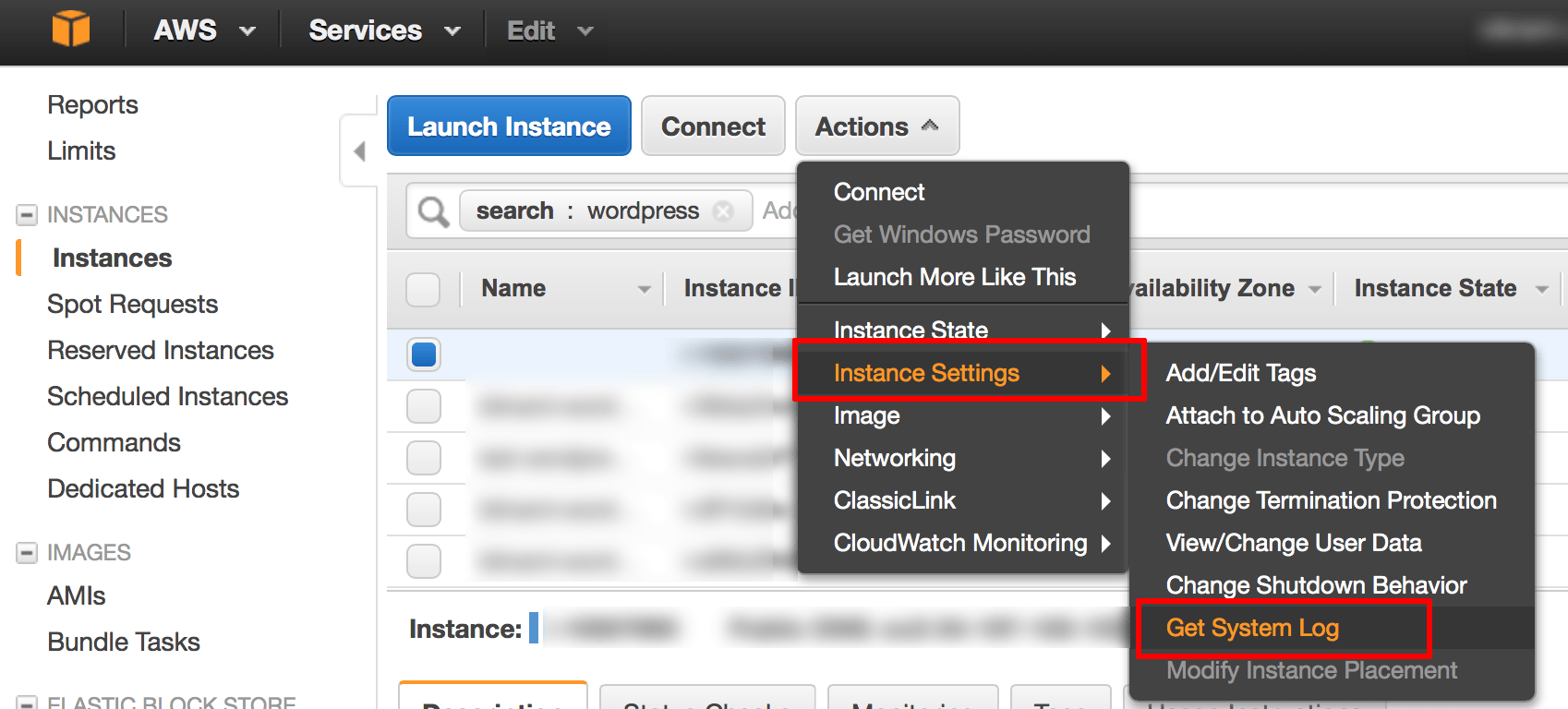
Confirm your selection by hitting the “Launch Instances” button and the AWS Console will now begin spinning up the new server.



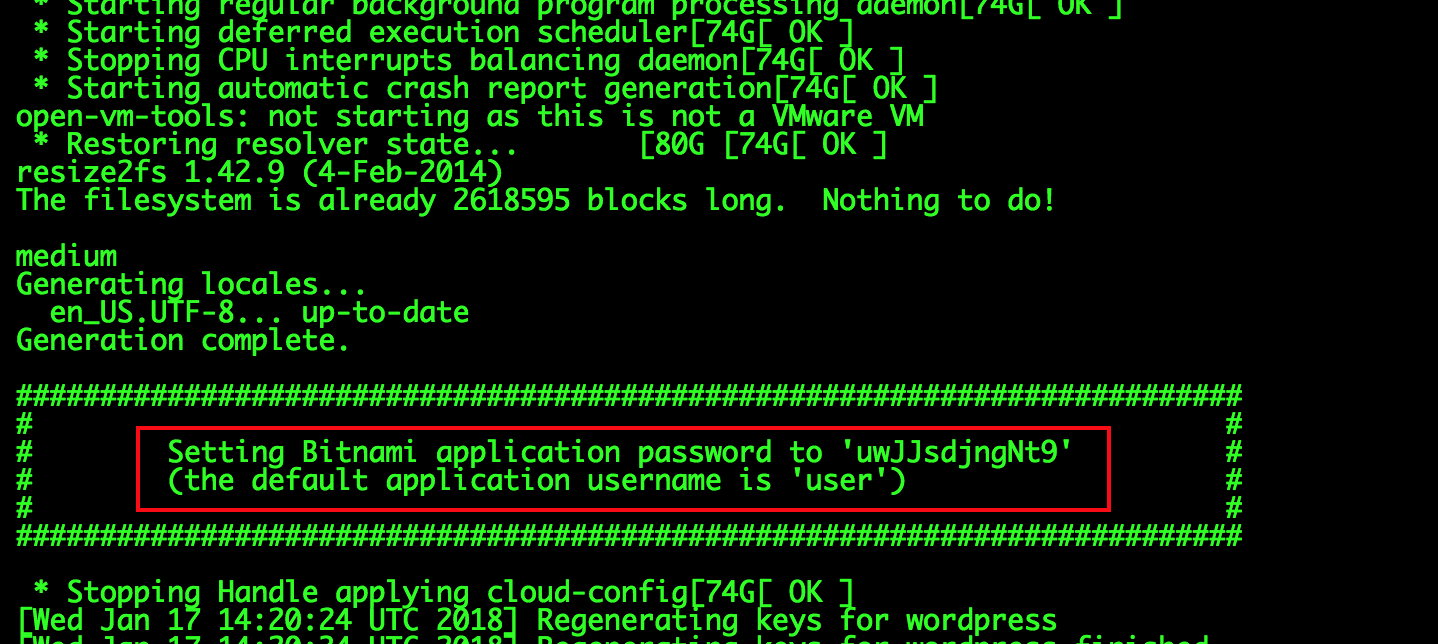
The process usually takes a few minutes, and you can use the EC2 Dashboard to check the status of the server. Once the server has launched, you will be able to obtain its public IP address and public DNS name from the EC2 Dashboard.

To get the administrator username and password, follow these steps:

* In the left navigation bar, select the “Instances -> Instances” menu item.
* Select your instance.
* From the “Actions” drop-down menu, select the “Get System Log” menu item.



Review the system log until you find a message with the application username and password.



**NOTE: This password is only shown the first time you start the image. Please save it in a safe place.**

However, you can also obtain the username and password at any time by checking the bitnami\_credentials file as detailed here: <https://docs.bitnami.com/aws/faq/get-started/find-credentials/#option-2-find-credentials-by-connecting-to-your-application-through-ssh>

**Log in to RabbitMQ Console**

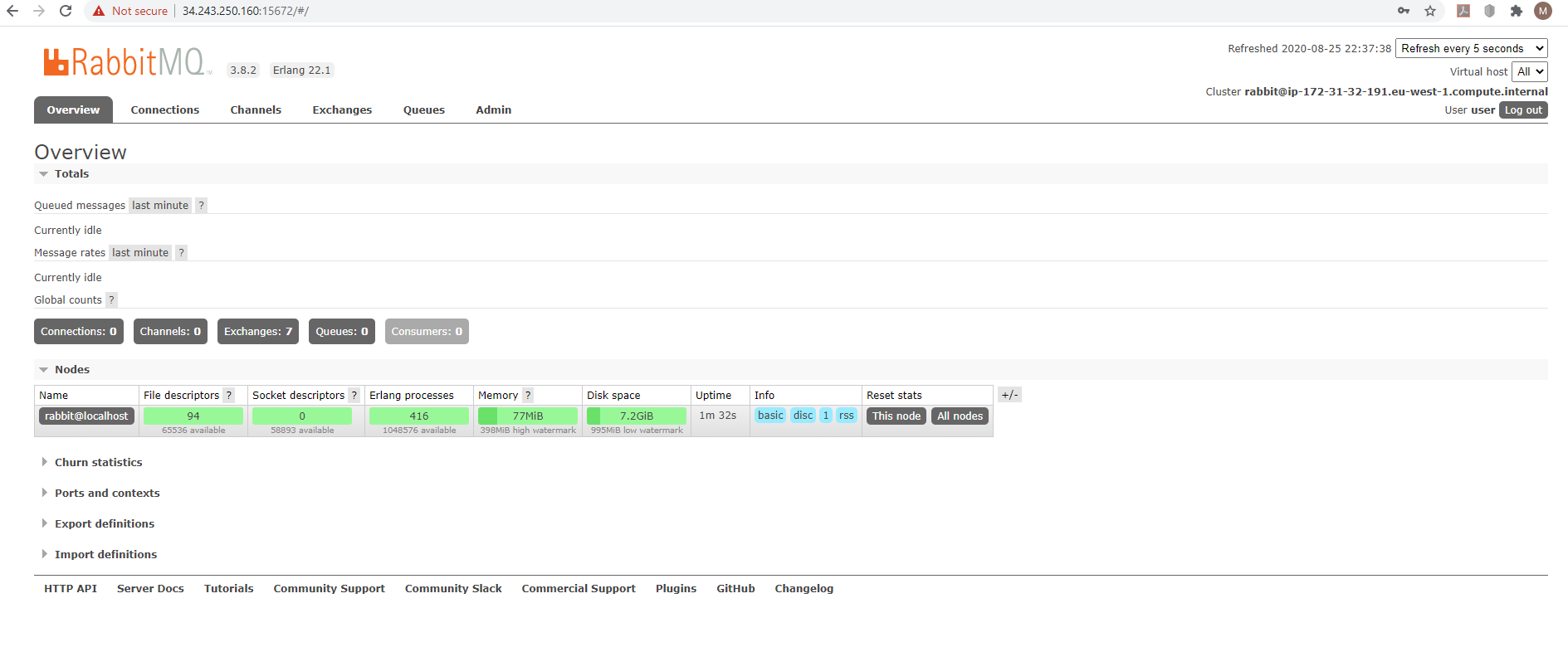
At this point, you should be able to browse to the RabbitMQ Management Console, by entering the cloud server IP address or DNS name directly into your browser’s address bar. You should now see your login page as shown below. You can use

**admin:** admin as username

**password:** from the launch configuration log above.



You should now be able to see your RabbitMQ Console

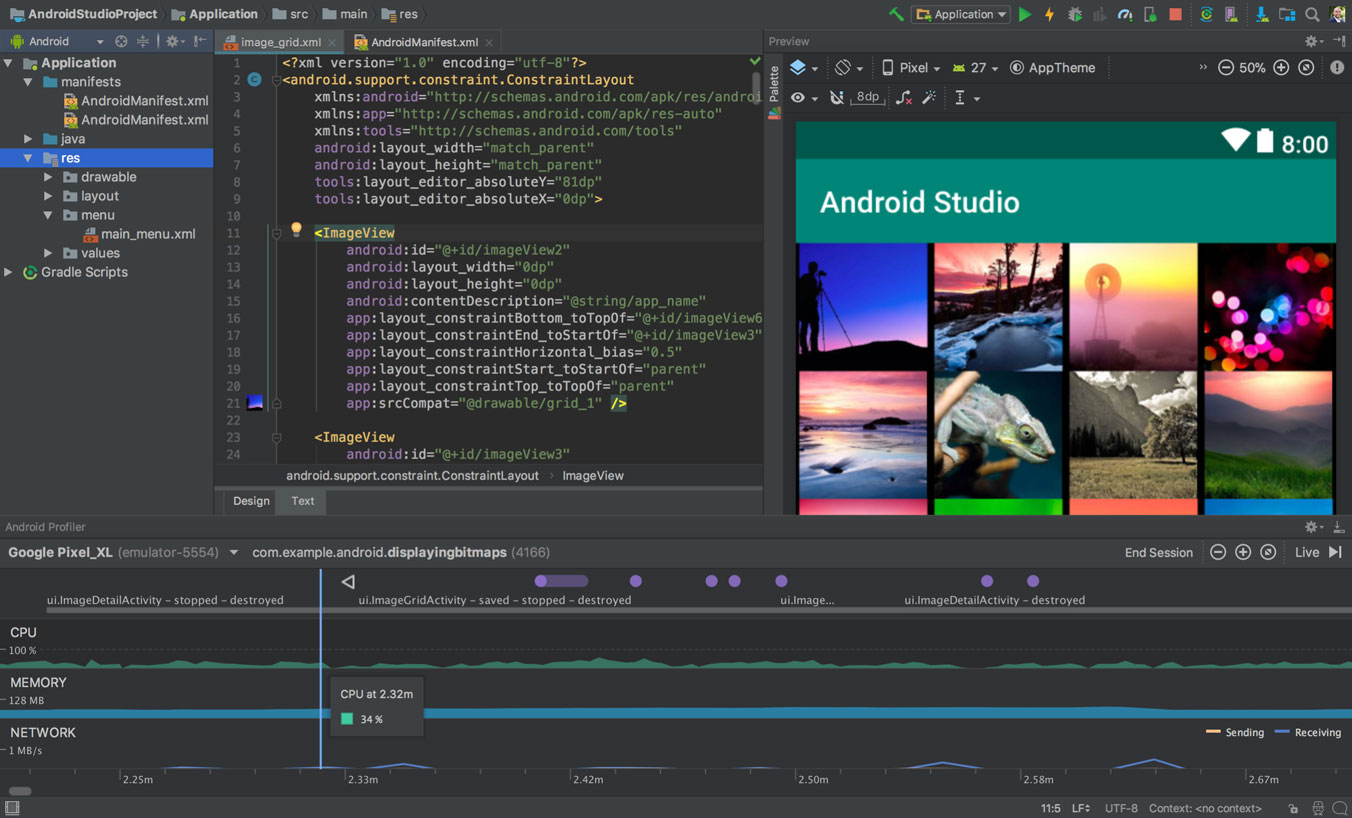


SETUP LISTNERS

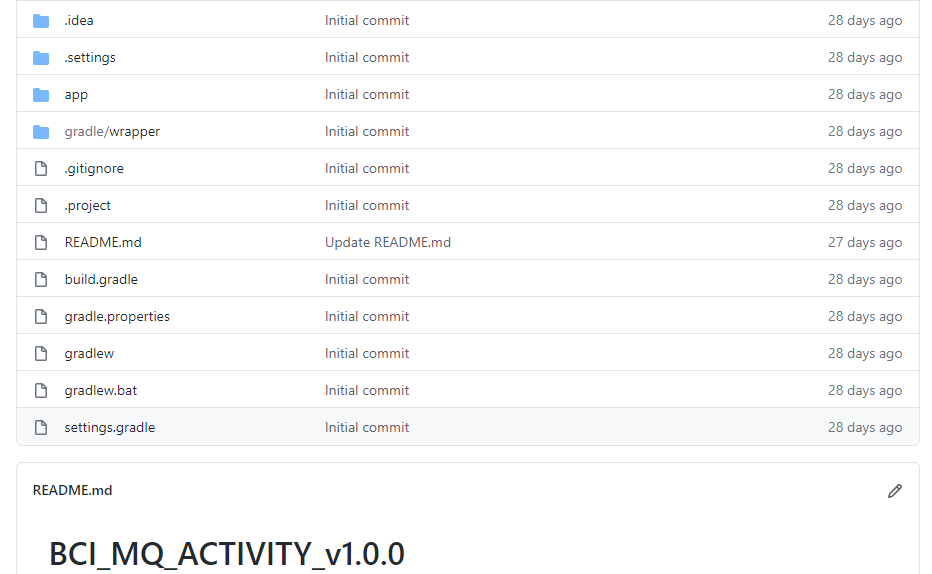
**Install Android Studio**

Download Android Studio for here: <https://developer.android.com/studio> - I have used version 4.0.1 for development of this project but latest should work fine.

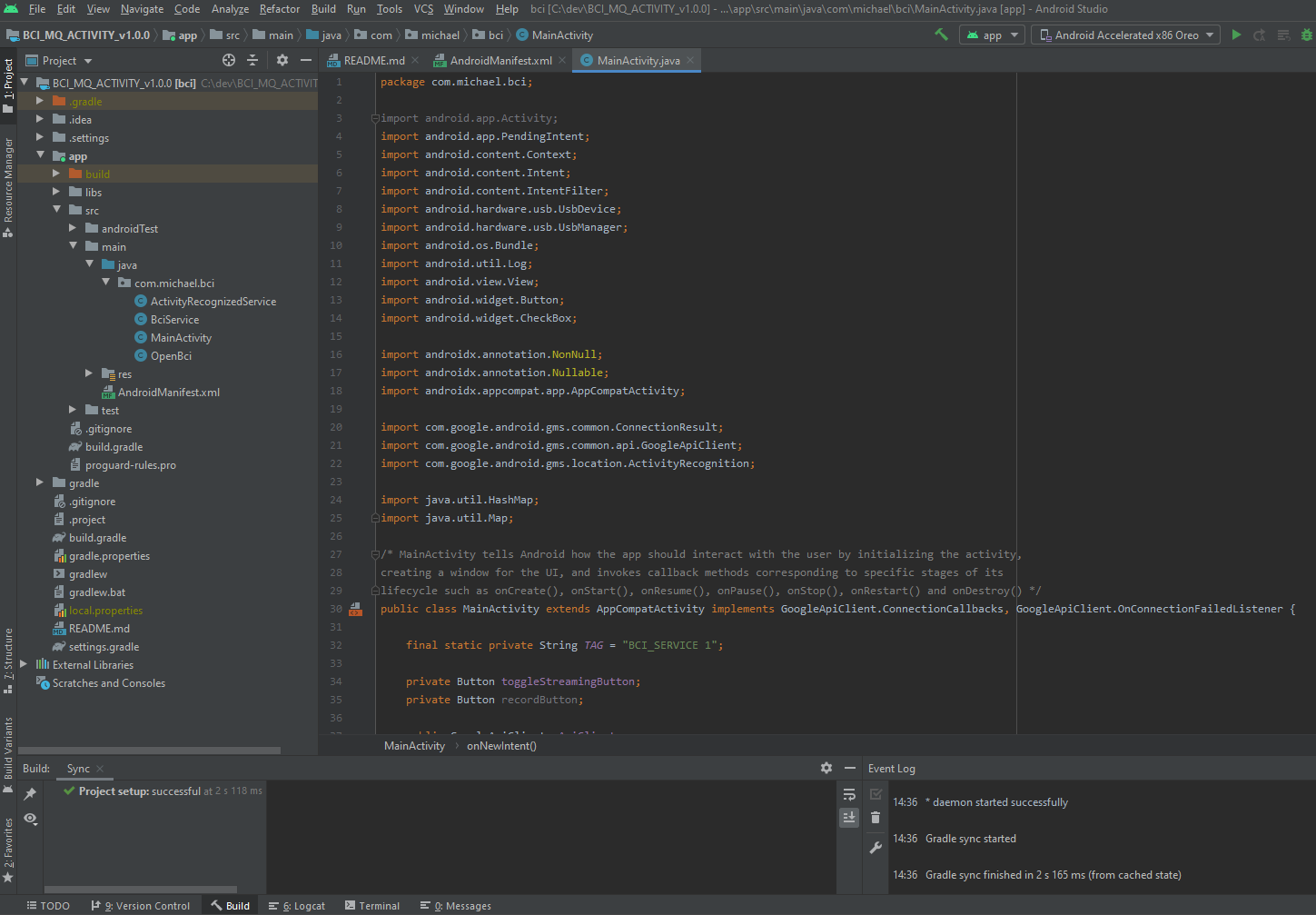
Once downloaded open the IDE



Download Source Code from GitHub here: <https://github.com/michaelmcmahon/BCI_MQ_ACTIVITY_v1.0.0>



Open the project in Android Studio IDE



**Setup Android Bridge Debug (ABD)**

The inbuilt Android emulator is unfortunatley not very helpful for debugging the BCI application as we need to test using a real-time EEG signal. We need to be able plug in the OpenBCI dongle via OTG into an Android phone to test and debug over an Android Debug Bridge (ADB) connection.

Android Debug Bridge (adb) is a command-line tool that lets you communicate with a device for installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device. It is a client-server program that includes three components: A CLI client on your development machine, a daemon (adbd) which runs commands on a device and a server which manages communication between the client and the daemon.

abd is included in the Android SDK Platform-Tools package - download this package with the SDK Manager, which installs it at android\_sdk/platform-tools/ - <https://developer.android.com/studio/intro/update#sdk-manager>

Before you can start debugging on your device you need to open the Settings app on your phone, select Developer options, and then enable USB debugging.

**NOTE:** On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to *Settings > About phone* and tap Build number seven times. Return to the previous screen to find Developer options at the bottom.

**NOTE:** When you connect a device running Android 4.2.2 or higher, the system shows a dialog asking whether to accept an RSA key that allows debugging through this computer. This security mechanism protects user devices because it ensures that USB debugging and other adb commands cannot be executed unless you're able to unlock the device and acknowledge the dialog.

Connect your device to the development machine using USB

Open Command Prompt and navigate to: *CD C:\Users\YOUR USERNAME\AppData\Local\Android\Sdk\platform-tools*

Type 'adb devices' and this will show a list of devices attached to the development machine

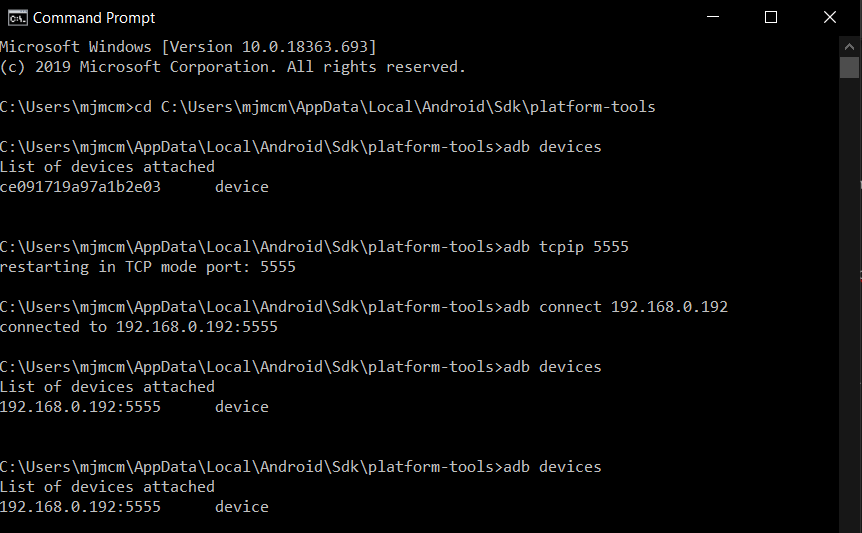
Set the target device to listen for a TCP/IP connection on port number 5555 using command 'adb tcpip 5555' and wait for this step to complete. (Make sure the port number selected is not already in use)

Disconnect the USB cable from the target device.

Find the IP address of the Android device. For example, on a Nexus device, you can find the IP address at Settings > About tablet (or About phone) > Status > IP address.

Connect to the device by its IP address 'adb connect *IP address* '

Confirm that your host computer is connected to the target device by typing 'adb devices'



WORK STILL IN PROGRESS