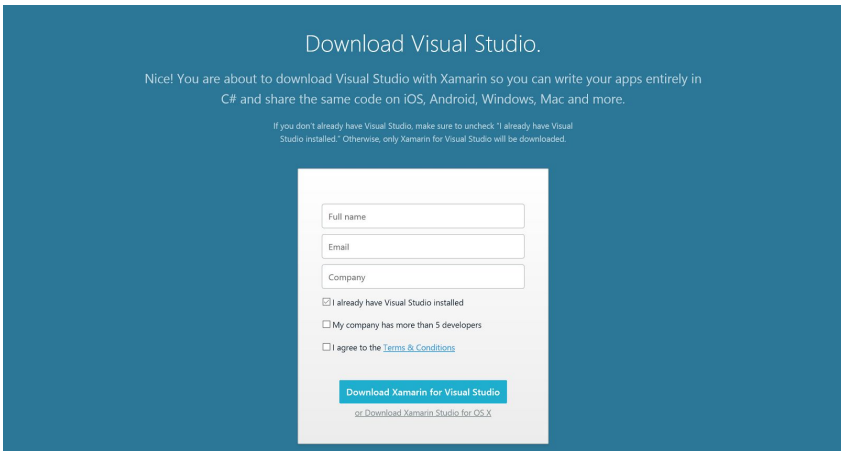
Windows Setup Checklist

You will need a PC running Windows 7 or newer (Windows 10 recommended).

Download the Xamarin Unified Installer

The first step is to download the Xamarin unified installer, which can be found at http://www.xamarin.com/Download:



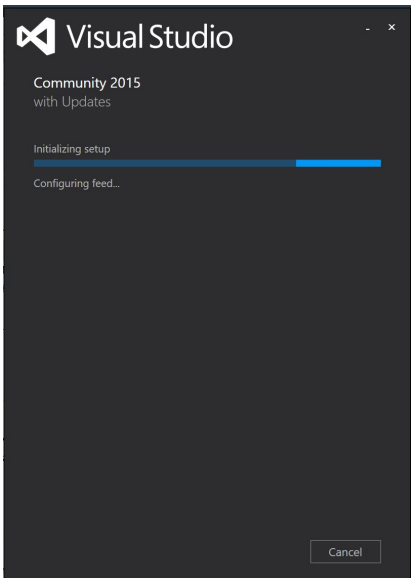
Depending on if you select the *I already have Visual Studio Installed* checkbox, you will be offered the option of downloading either Visual Studio Community Edition (if you don't have Visual Studio installed), or Xamarin for Visual Studio, via the Xamarin Unified Installer.

Regardless of which package you download, you can follow the instruction below.

Run the Xamarin Installer

Visual Studio Community installer

Open the installer to begin the installation process:



The installer will assess your system, and allow you select optional components.

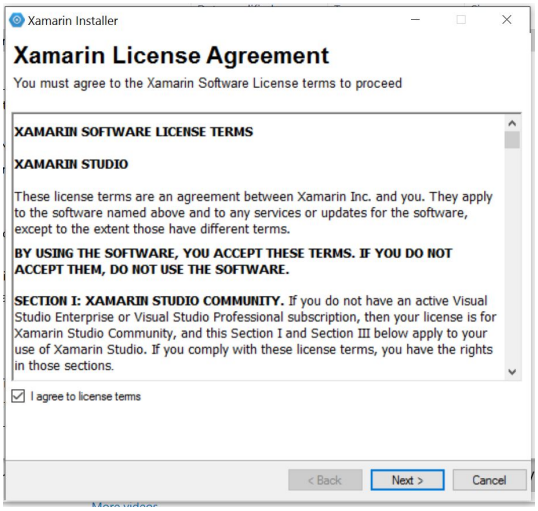
Make sure to select **Xamarin** from the menu when prompted.

Xamarin Unified Installer

Open the installer to begin the installation process:

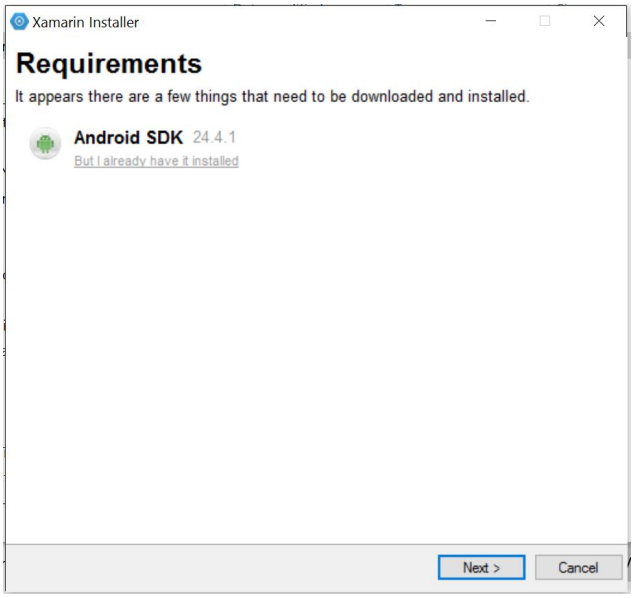
Step 1 – Xamarin License

The first step in the installation requires you to review and accept the license in order to proceed. To do this, click the checkbox and press **Next**:



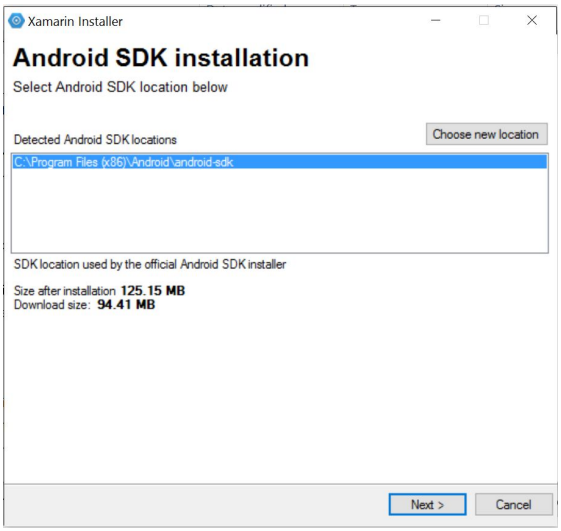
Step 2 – Identify the Required Components

Next, the installer will inspect the system to determine which, if any, required components are missing and need to be downloaded and installed. You can select which products you wish to download here:

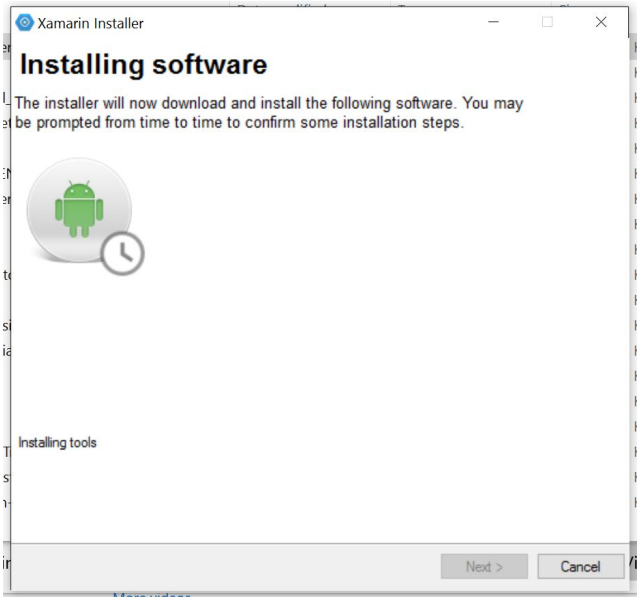


Step 3 – Install the Components

After identifying missing components, the Xamarin installer will download and execute the installers for the platform dependencies, as shown in the screenshots below.



The Xamarin unified installer will start the download and install process of the selected items:



When the installation is complete, close this window to exit the installer and begin working with Xamarin.

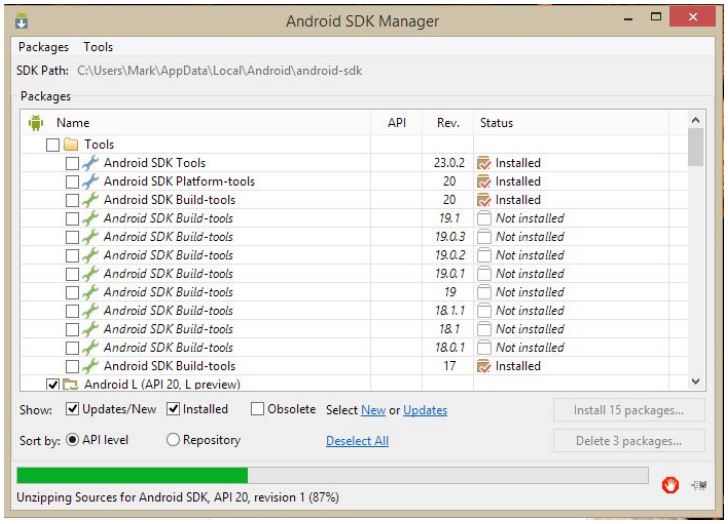
Setup Android Emulators

To test Android applications, you can either deploy to a physical device (preferred), or use an Android emulator. If you plan to use an emulator, we recommend you use **Visual Studio Android Emulators** as it delivers higher performance over the Android SDK emulator.

1. Download the Visual Studio Android Emulators install from https://www.visualstudio.com/en-us/features/msft-android-emulator-vs.aspx, for Windows you can download the standalone package, and run on Hyper-V.
2. Once installed, run the application and download a pre-configured image, there are several to choose from.
3. Launch the image from the Xamarin Android Player app by selecting it in the list and clicking the Play button.
4. It should then show up in the emulators list in Xamarin or Visual Studio.

Verify your Setup

1. Launch Xamarin Studio (OS X or Windows) or Visual Studio (Windows).
2. The IDE should prompt you to either start a trial with Xamarin, or enter to your registered Xamarin account. Go ahead and do one of these two steps so you be able to build full applications, including the T-shirt test app.
3. Verify that you have all the Android SDK versions you will need. We recommend installing 4.0 through 6.0. You can get to the Android SDK options using the **Tools > Android > Start Android SDK Manager** in Visual Studio.



4. Download the Tasky application from, https://github.com/xamarin/mobile-samples/tree/master/Tasky unzip it onto your desktop (or some other easily accessible location) and open the solution.
    a. **Note**: Be careful about path lengths on Windows – mobile projects tend to create deep subdirectories and you can quickly exceed the path length on Windows machines. It is best to place the solutions into shorter paths such as your desktop or right in the root of the drive.
5. Open the solution (.sln) using your IDE of choice and following the instructions below for iOS, Android or both.

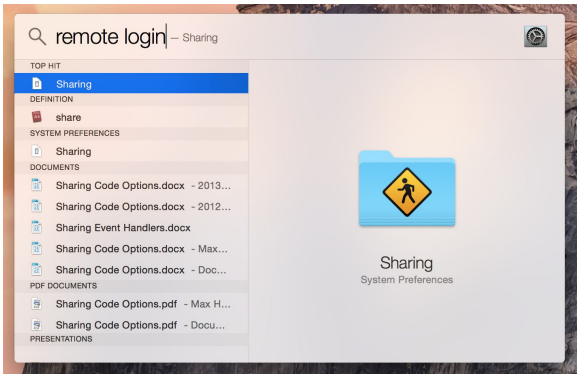
Pairing Windows with a Mac host for iOS development

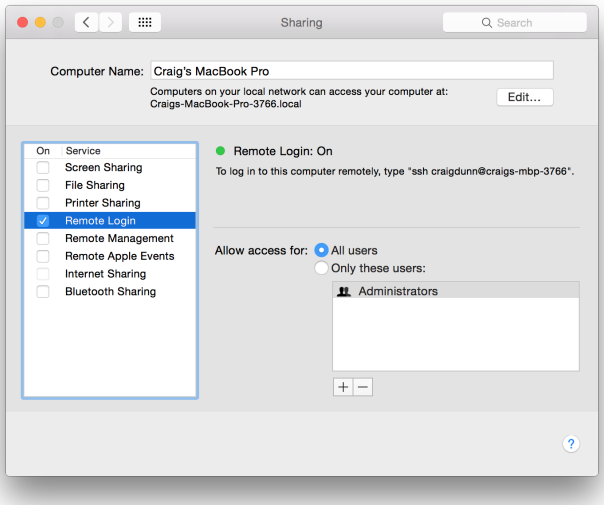
If you plan to do any development for iOS, you will need Visual Studio 2012 or newer **and** a Mac host with the Xamarin tools installed. You will do your development on the Windows computer with Visual Studio, and the Mac is used to build and package up the iOS application using Apple's tools.

The computers will need to be on the same network and be able to ping each other. This section details how to connect the two computers together through the Xamarin Build Host and assumes you have already setup both the Mac and Windows computers properly.

Mac Setup

To set up the Mac host, you must enable communication between the Xamarin extension for Visual Studio and your Mac. To do this, you will need to allow **Remote Login** on your Mac by following the steps below:

1. Open *Spotlight*(⌘-Space) and search for *Remote Login* and then select the *Sharing* result. This will open *System Preferences* at the *Sharing* panel:
2. 

3. Tick the *Remote Login* option in the *Service* list on the left in order to allow Xamarin for Visual Studio to connect to the Mac:
4. 

5. Make sure that *Remote Login* is set to allow access for *All* users, or that your Mac username or group is included in the list of allowed users in the list on the right.

In addition to this, if you have the OS X firewall set to block signed applications by default, you may need to allow mono-sgento receive incoming connections. An alert dialog will appear to prompt you if this is the case.

Providing you have a current, open session on your Mac, it should now be discoverable by Visual Studio if it's on the same network.

Visual Studio will start and stop the agent on your Mac, so there is nothing else that you, as a user, need to run.

NOTE: Be aware that Visual Studio will not check that the Xamarin.iOS SDK and Xcode exist and have compatible versions. That will be checked by the build agent, resulting in build errors; and by the designer agent, resulting in designer errors. Always make sure that you are on the same distribution channel on both Visual Studio and Xamarin Studio.

Windows Setup

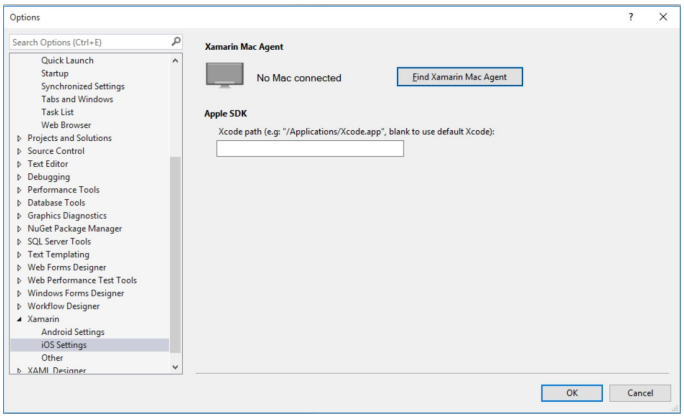
Beyond installing Xamarin tools on your Windows machine, there is no other setup that you will need to do.

Connecting

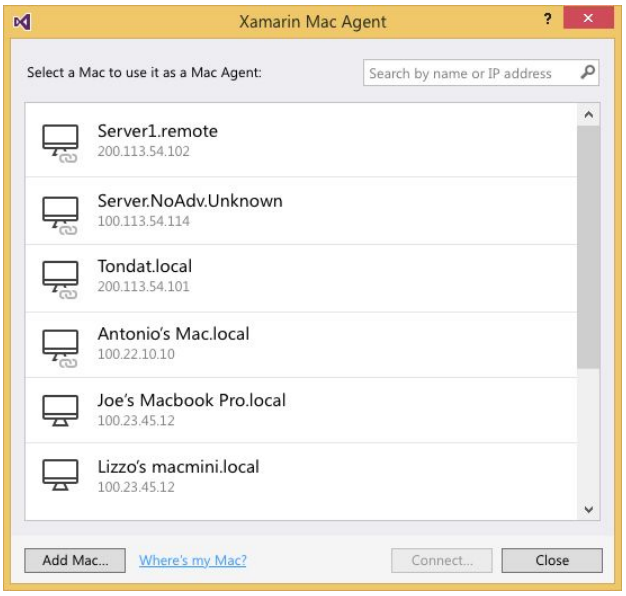
There are two ways to connect to your Mac build host:

On the iOS toolbar:

Or by browsing to *Tools* > *Options* in Visual Studio, selecting *Xamarin* > *iOS Settings* and clicking the *Find Xamarin Mac Agent* button:

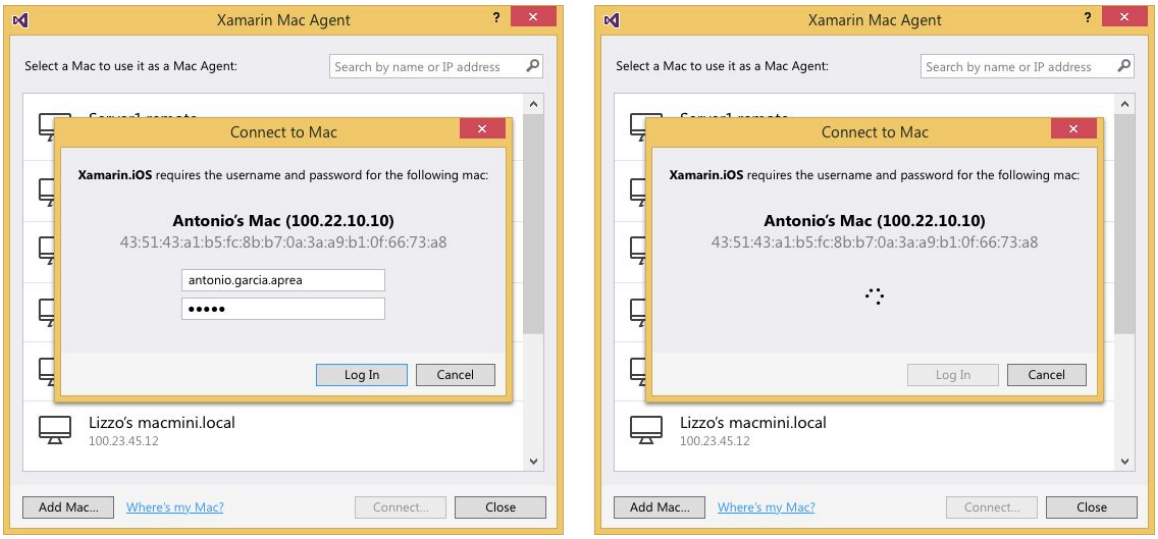


Navigating either way will lead you to the **Mac Agent** dialog, illustrated below:



This will display all the machines that have either been previously connected and are stored as known machines, or machines that are available for *Remote Login*.

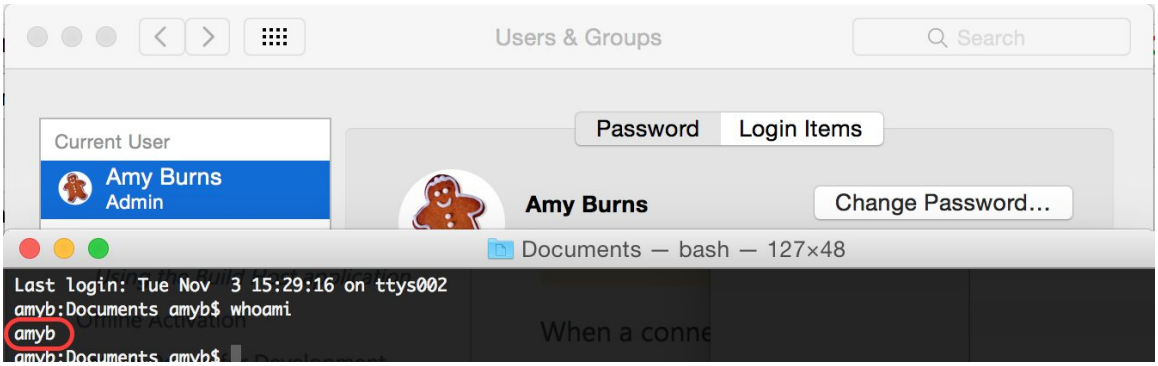
Select a Mac by double-clicking on it to connect to it. The first time that you connect to a Mac, you will be prompted to enter your Mac user credentials (which must be an Administrator account) to allow the remote connection:



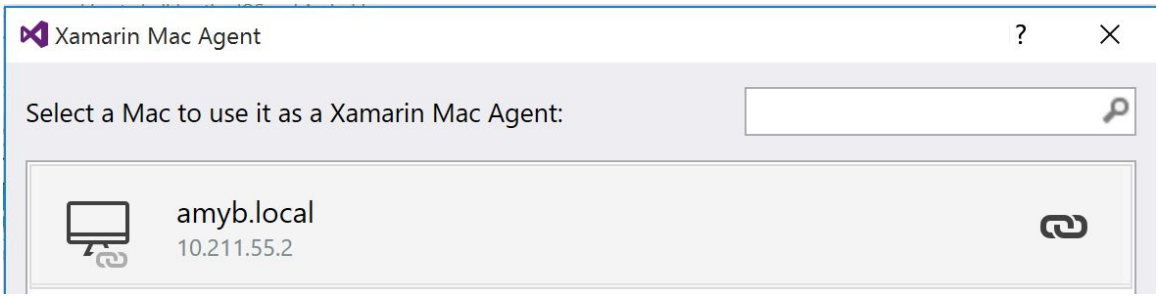
The agent will use these credentials to create a new SSH connection to the Mac. If it succeeds, an SSH key will be created, and will be registered in the `authorized_keys` file on that Mac. On subsequent connections the agent will use the username and key file to connect to the most recently connected known build host.

Note: You must use the *username* and not the *full name* when entering your credentials.

You can find this out by using the `whoami` command in Terminal. For example, from the screenshot below, the account name will be amyb and not Amy Burns:



When a connection has been successfully made, it will display in the Host Selection dialog with a **connected** icon next to it, as illustrated below:



There can only be one connected Mac at any one time.

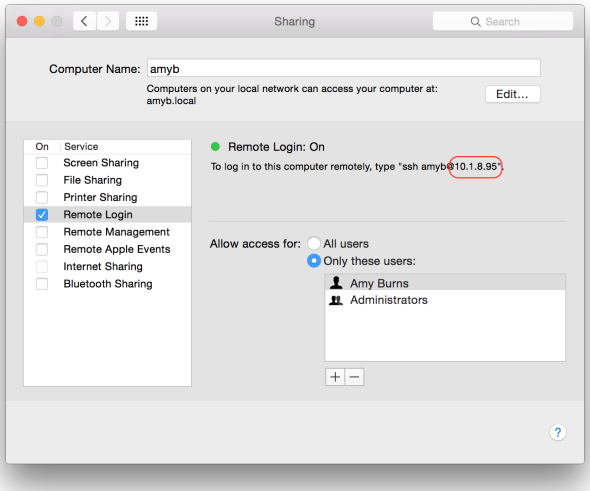
Each machine in the list, whether connected or otherwise, will display a context menu on right-click, allowing you to*Connect*, *Disconnect*, or *Forget the Mac* as needed:

If you choose to *Forget this Mac*, you will need to re-enter your credentials to connect to it again.

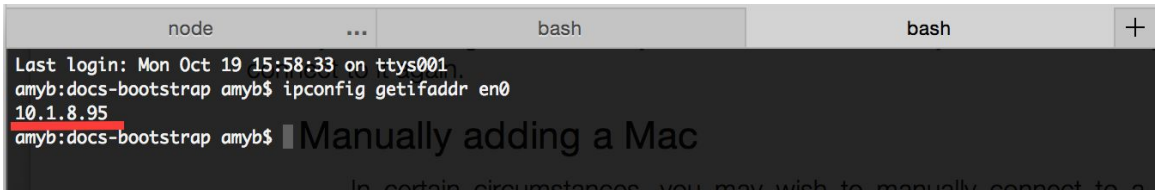
Manually adding a Mac

In certain circumstances, you may wish to manually add a Mac if you cannot see its mDNS name listed in the Host Selection dialog. To do this, follow the steps below:

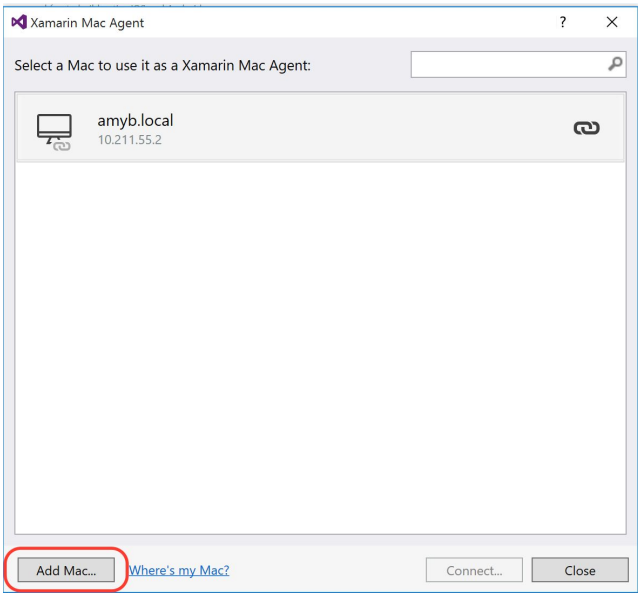
1. Locate your Mac's IP address by either browsing to the *System Preferences* > *Sharing* > *Remote Login* on your Mac:

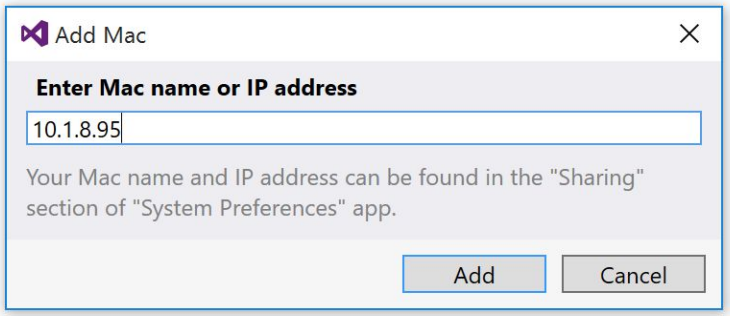


2.
3. Or, if you prefer to use the command line you can find out your IP address by entering ipconfig getifaddr en0 into Terminal:

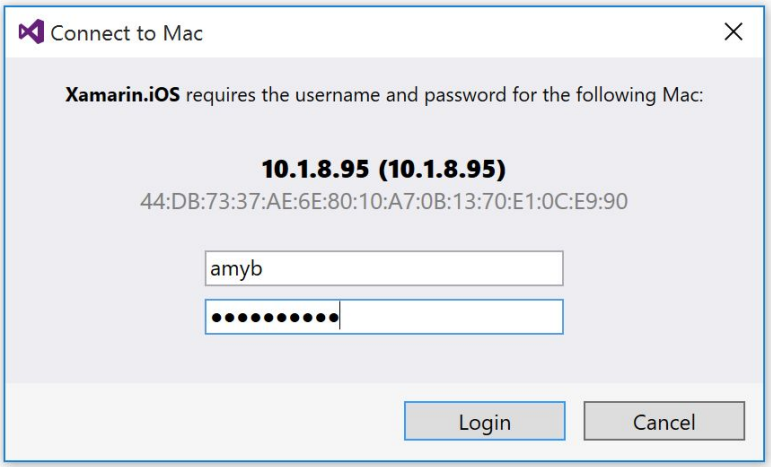
4. 

5. Return to Visual Studio and in the Host Selection dialog, select **Add Mac...**:

6. 

7. Enter the IP address of you Mac into the Add Mac dialog and click **Add**:
8. 

9. Finally, enter the username (not full name) of your Mac admin account and the corresponding password:

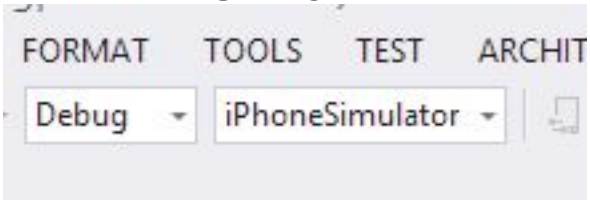


10.

Once you click **Login**, Visual Studio will log into the Mac machine using SSH and will add this Mac as a known machine.

Testing iOS on Windows with Visual Studio

1. Make sure your setup is connected to the Mac build host (see above instructions to verify this).
2. Make sure the **TaskyiOS** project is the active project in Visual Studio (you can right-click and choose *Set As Startup Project* to change it if necessary).
3. Select the **Debug** configuration and **iOS Simulator** from the toolbar.



4. Select the iPhone 5 iOS 8.0 simulator from the iOS toolbar. If you don't see the toolbar, right click in an open space in the toolbar area and make sure **iOS** is checked for display.

5. Build and run the application by clicking the **Start** button in the toolbar – this should launch the iPhone simulator on the Mac.

Testing Xamarin.Android on Windows

1. Make sure the **TaskyDroid** project is the active project in Xamarin or Visual Studio (you can right-click and choose *Set As Startup Project* to change it if necessary).
2. Select the **Debug** configuration and **AnyCPU** from the toolbar.
3. Select an available Andoid emulator in the Android toolbar. If you don't see the toolbar choices, right click in an open space in the toolbar area and make sure **Android** is checked for display.

4. If you do not see any emulators, then you will need to configure at least one You can open the Android SDK tool with the **Tools > Android > Open Android Emulator Manager** menu option as described above.
5. Build and run the application by clicking the **Start** button in the toolbar – this should launch the Android emulator (if it's not running), install the application and then run it.