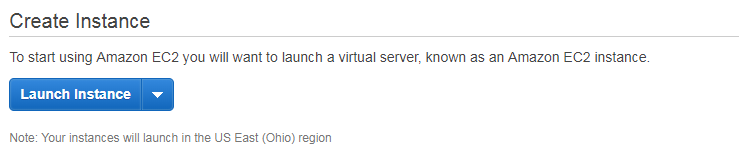
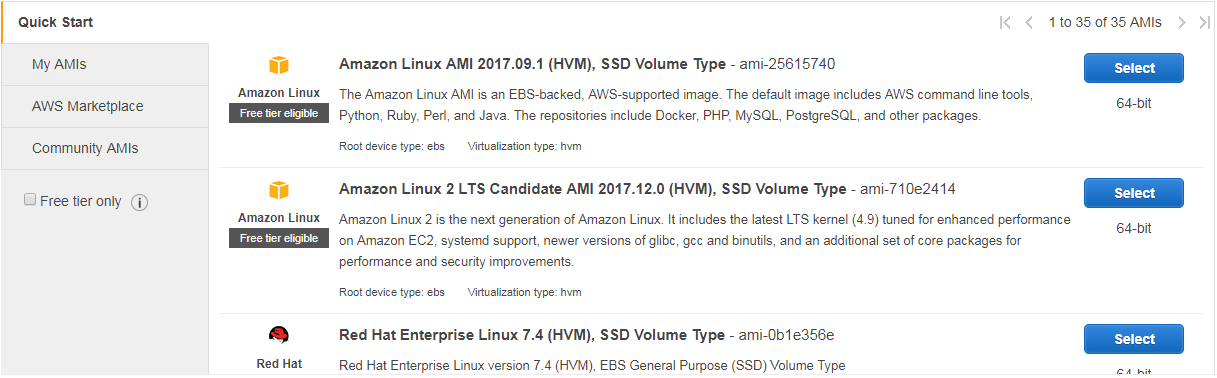
# Setting up Amazon Web Server for Raspberry PI

## Creating Ubuntu Virtual Machine

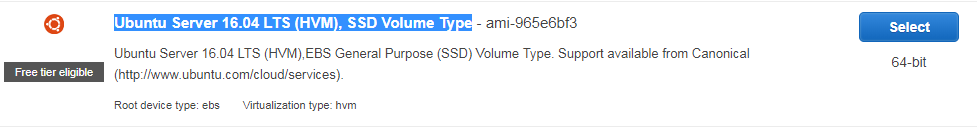
1. Go to “EC2 Dashboard”
2. Click on “Launch Instance” from EC2 Dashboard.



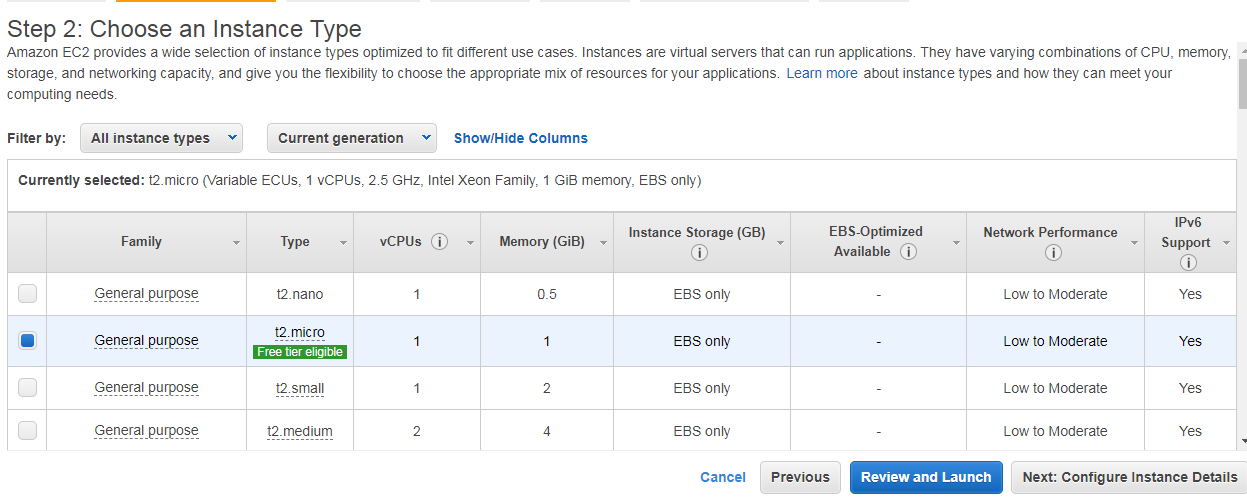
1. Select an Amazon Machine Image (AMI) from below options.



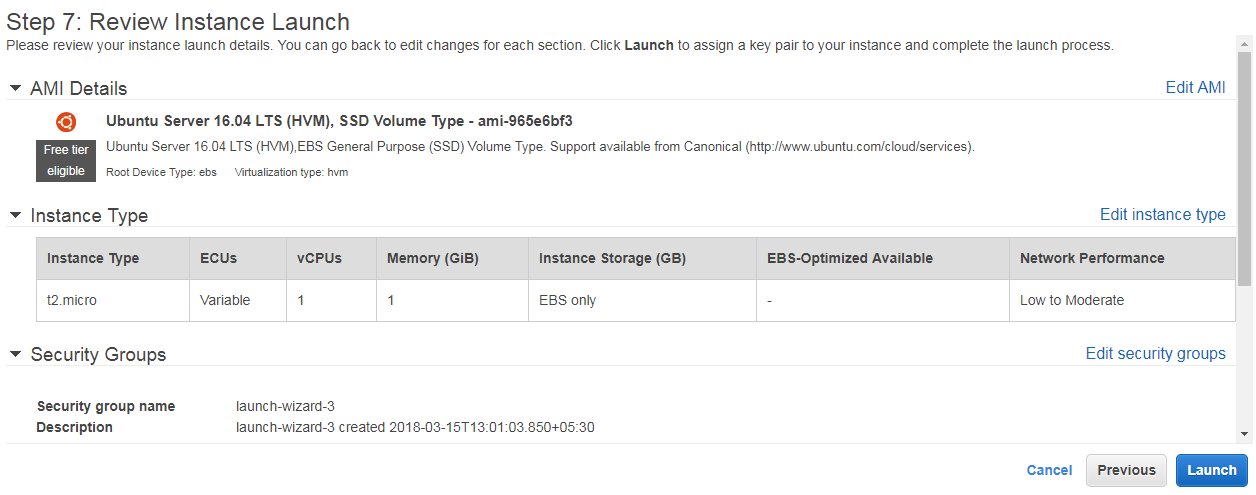
1. For Ubuntu, select “Ubuntu Server 16.04 LTS (HVM), SSD Volume Type” image.



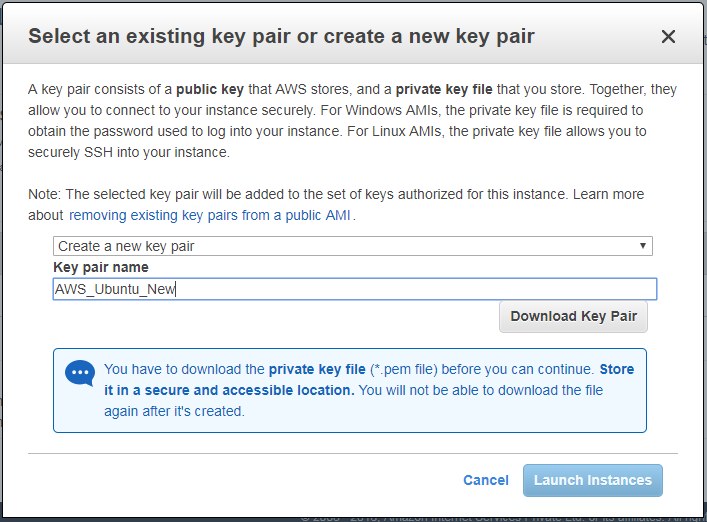
1. Choose instance type with configurations. (I will go with free version).



1. We can configure instance details with option “Next: Configure instance Details”.
2. After all configurations, click on “Review and Launch”, then it will show all instance configuration details as below.

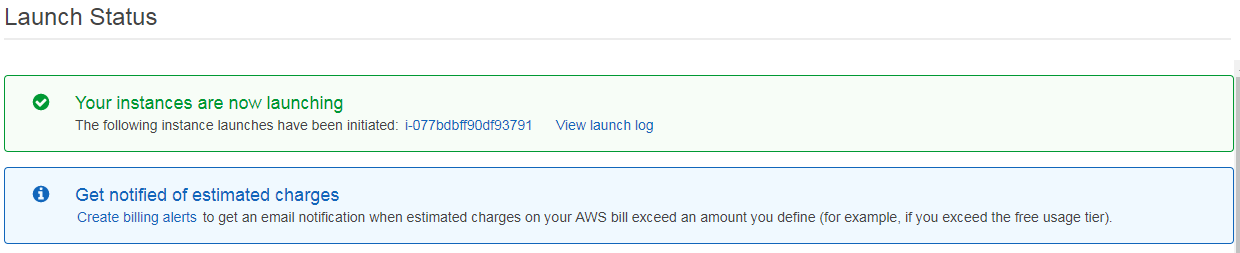


1. Click on “**Launch**” button to launch the instance, it will ask for creating a private create, define the key name, and click on “download key”.

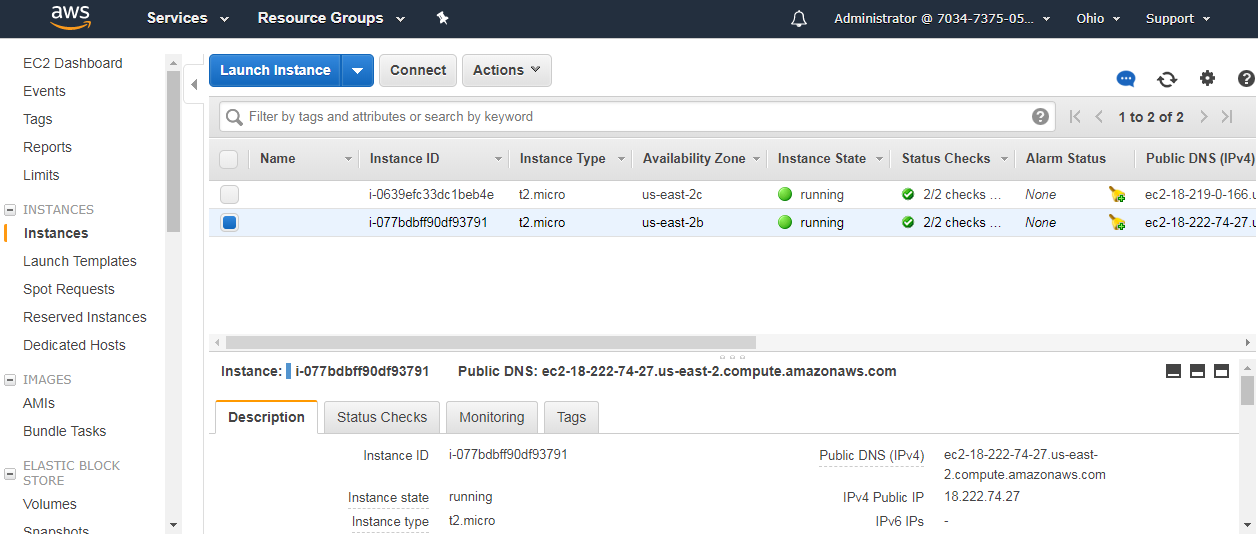


Note: This private key will be useful when you are trying to access server instance by SSH (putty software) or by SFTP (WinSCP software). So, we must download this key, and should be imported in putty and WinSCP. Here, the key name should be “AWS\_Ubuntu\_New.pem”.

1. Finally, we will get a message and the server instance is created successfully.



1. We can see all the server instance running in EC2 dashboard.



## Connecting to AWS Ubuntu Instance by using Putty (for SSH).

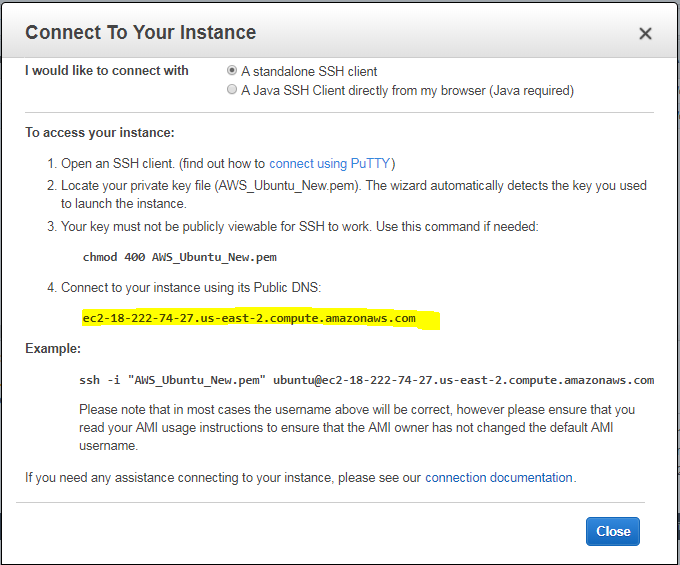
1. Download and install putty from <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>.
2. Start PuTTYgen (for example, from the Start menu, choose All Programs > PuTTY > PuTTYgen).
3. Under Type of key to generate, choose RSA.


       RSA key in PuTTYgen
      

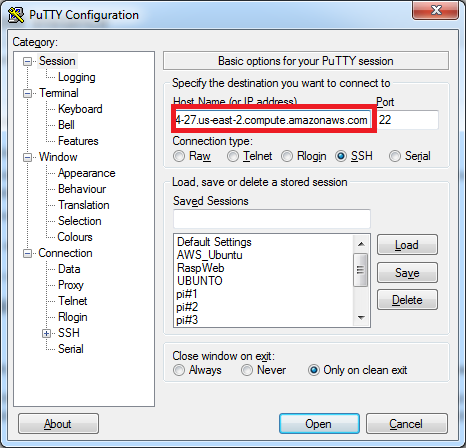
1. Choose Load. By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, select the option to display files of all types.


       Select all file types
      

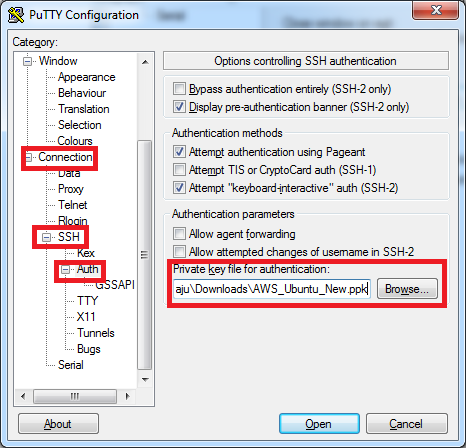
1. Select your .pem file for the key pair that you specified when you launched your instance, and then choose Open. Choose OK to dismiss the confirmation dialog box.
2. Choose Save private key to save the key in the format \*.ppk that PuTTY can use. PuTTYgen displays a warning about saving the key without a passphrase. Choose Yes.
3. Copy the Host name of instance from Amazon as shown below.



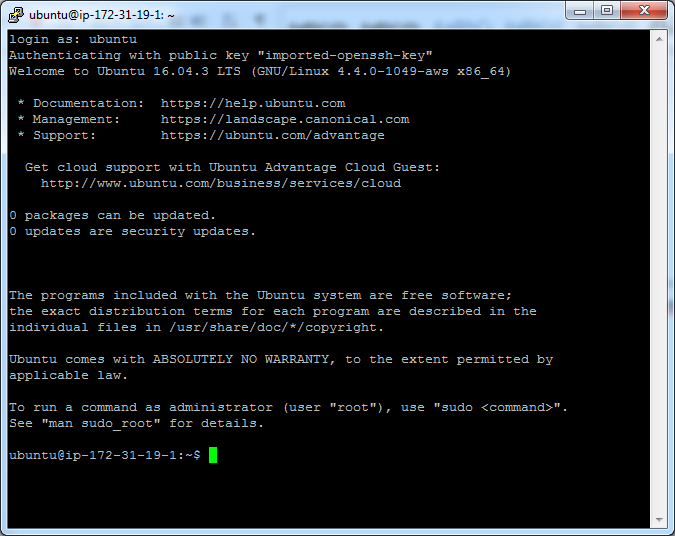
1. Open putty and past the host name.



1. Importing \*.ppk file which was generated at step f. In putty, select tab Connection 🡪 SSH 🡪 Auth. Then, browse the key file \*.ppk and click on “Open”.

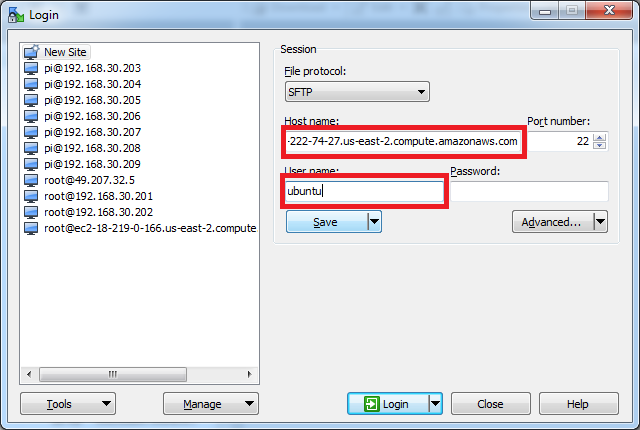


1. Now, the SSH command prompt will be open as shown below, now it asks for user name, the default user name for AWS Ubuntu Image is “ubuntu”. So enter user name “ubuntu” an press enter.

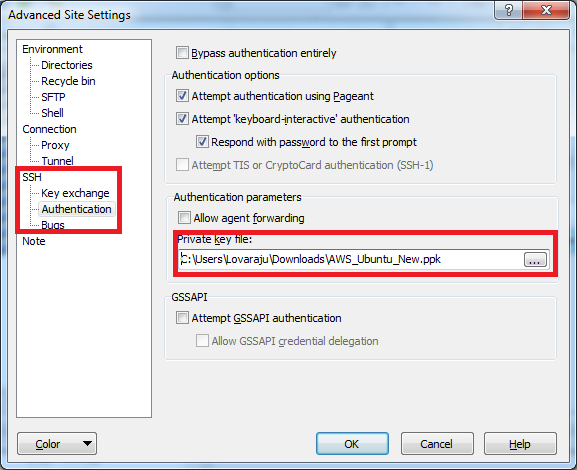


## Connecting to AWS Ubuntu Instance by using WinSCP (for SFTP).

1. Download and install WinSCP from <https://sourceforge.net/projects/winscp/>
2. Open WinSCP, enter AWS instance host name (copy from AWS), enter username “ubuntu”.

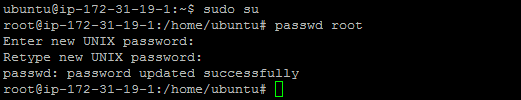


1. Import key \*.ppk by clicking on “Advanced” button, select tab SSH 🡪 Authentication. Then, browse the key file \*.ppk. Click on “OK” to connect to instance.



## Pre-installing packages on AWS Ubuntu Machine.

1. Connect to AWS Ubuntu machine by using putty.
2. Setting password to root.



1. Connect to root by using new password.



1. Update machine by using “sudo apt-get update”.



1. Upgrade machine by using “sudo apt-get upgrade”.



1. Install JRE (Java Runtime Environment) by using command “sudo apt-get install default-jre”. While installing, it prompts for confirmation, please enter “y”.



1. Install Python by using command “sudo apt-get install build-essential python-dev”. While installing, it prompts for confirmation, please enter “y”.



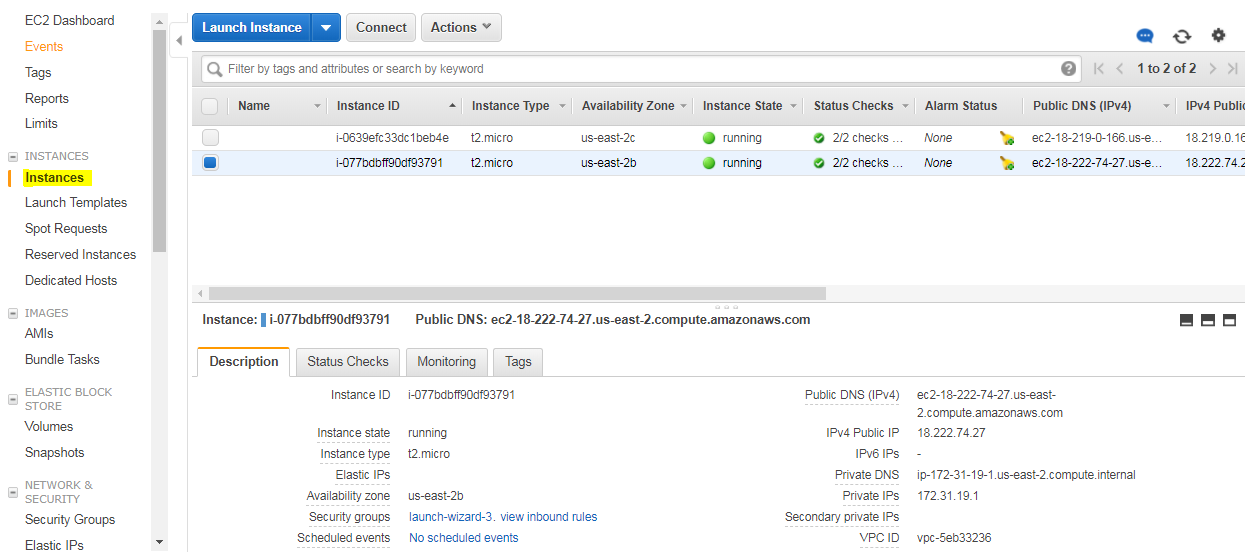
1. Installing Advanced Python Scheduler (APS) by using commands “apt install python-pip” and “pip install apscheduler”. While installing, it prompts for confirmation, please enter “y”.



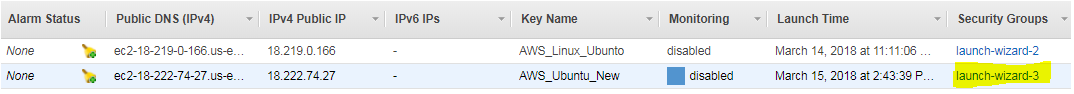


## Setting up Security Ports on AWS Ubuntu Machine.

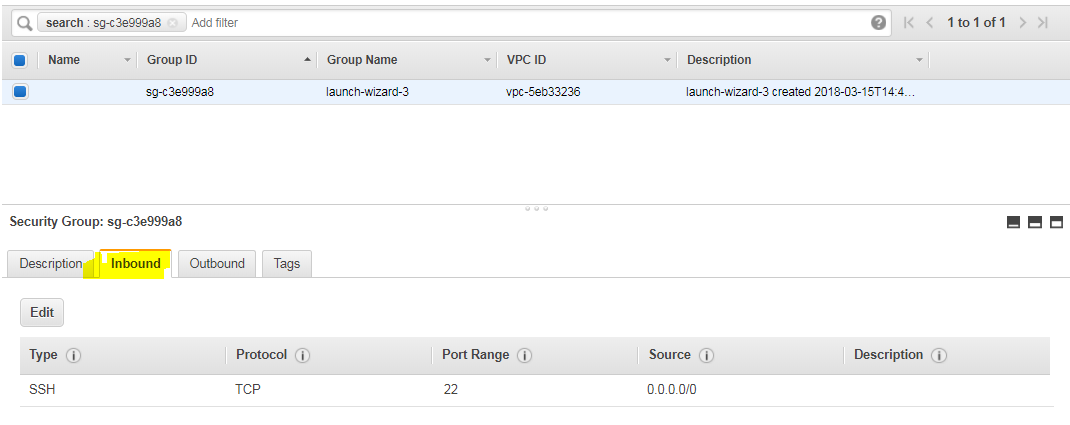
1. Navigate to the Instances tab on the **EC2 Dashboard** if you are not there already (click Instances in the left‑hand column.)



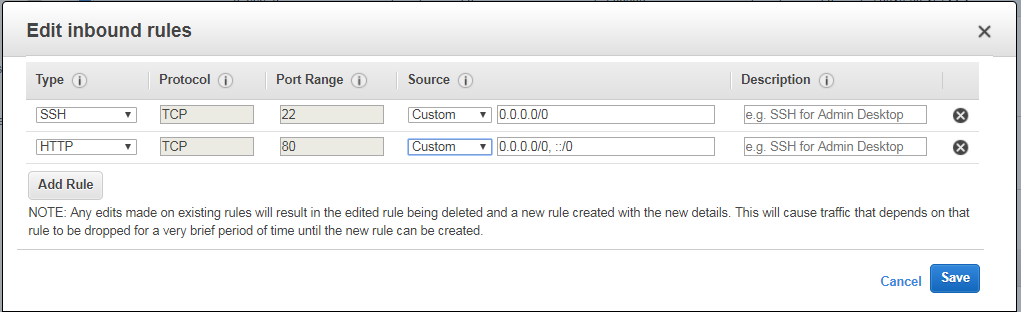
1. Select the instance, click on security groups link.



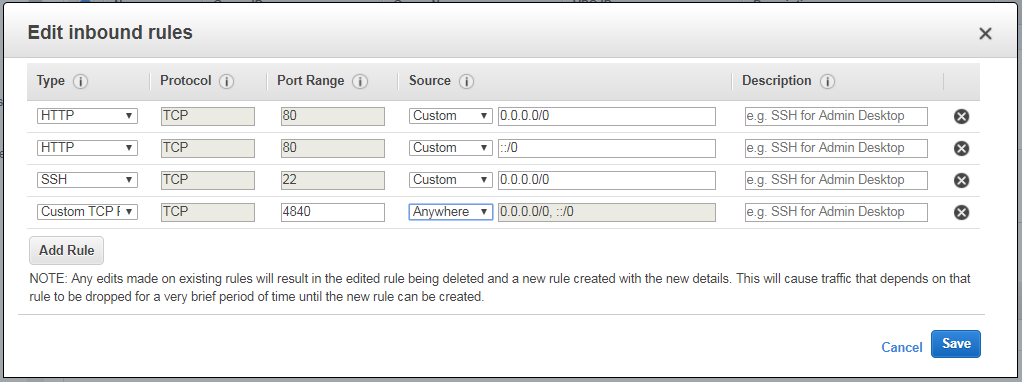
1. Click on “Inbound” tab.



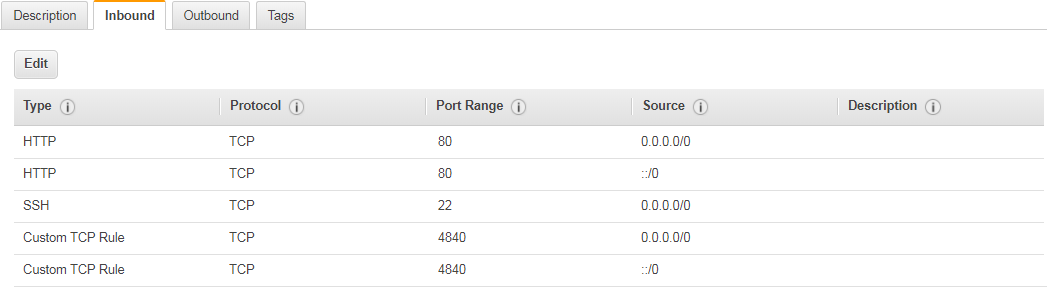
1. Edit port numbers by click on “Edit”.
2. Click on “Add Rule” button and “HTTP” port for allowing http requests. Save changes.



1. Add “Custom Rule” for allowing OPC Server requests with port no 4840. Save changes.



1. Please make sure the below ports are enabled or not.



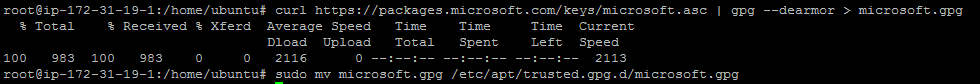
## Installing .NET CORE SDK 2.0 on AWS Ubuntu Machine.

Refer URL at <https://www.microsoft.com/net/core#linuxubuntu>

1. Register the trusted Microsoft signature key by entering the below commands:

curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg

sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg



sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/microsoft-ubuntu-xenial-prod xenial main" > /etc/apt/sources.list.d/dotnetdev.list'

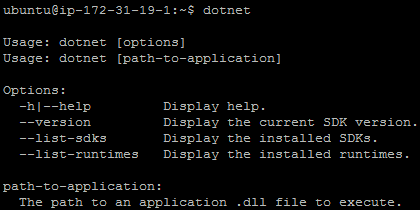


1. Install .NET Core SDK by command “sudo apt-get update” and “sudo apt-get install dotnet-sdk-2.0.0”. While installing, it prompts for confirmation, please enter “y”.



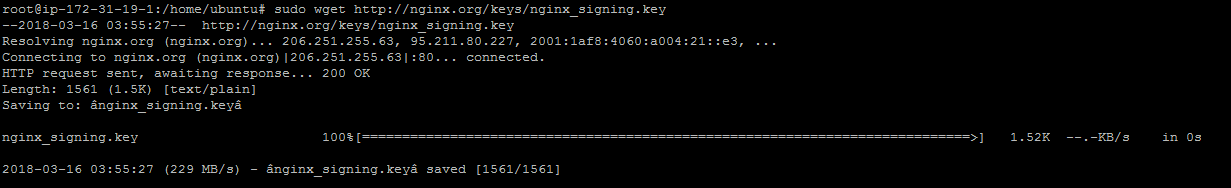


1. Verify donnet installation by executing “dotnet”.



## Installing Ngnix WebServer on AWS Ubuntu Machine.

1. Connect to AWS Ubuntu machine by using putty.
2. Change user to root by “sudo root” and enter password.
3. Download the NGINX signing key by “sudo wget <http://nginx.org/keys/nginx_signing.key>”.



1. Add the key by “sudo apt-key add nginx\_signing.key”.



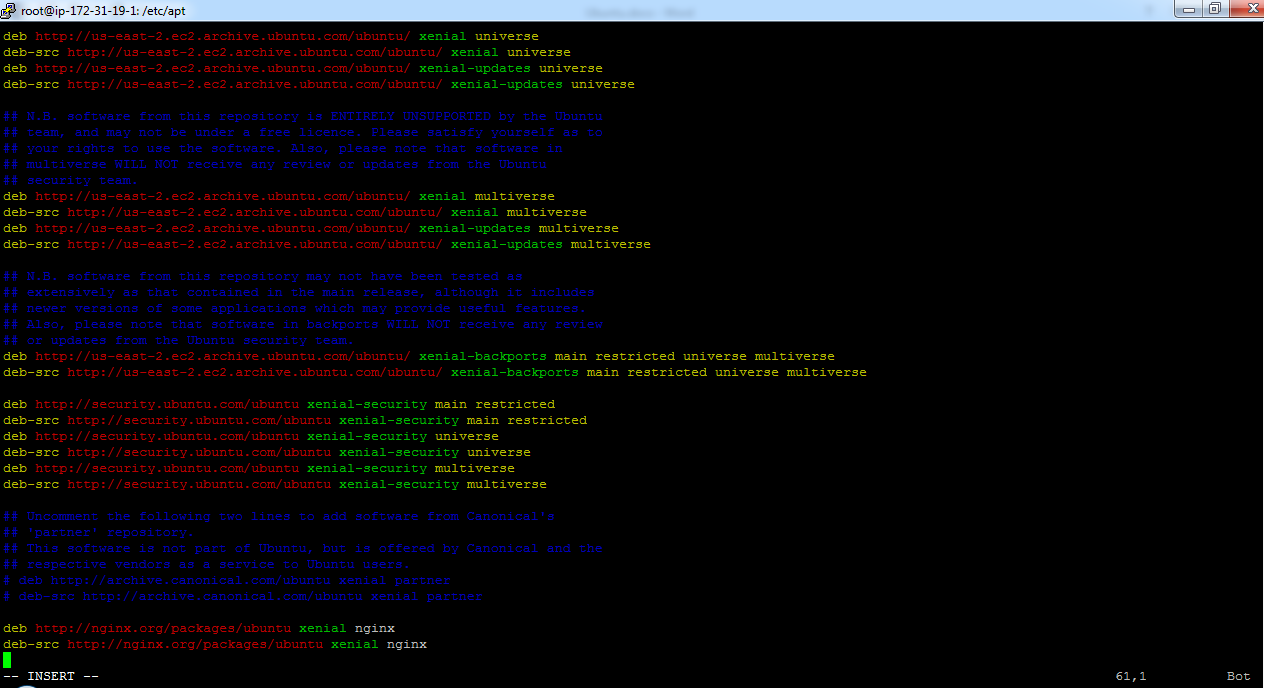
1. Change directory to /etc/apt by “cd /etc/apt”.



1. Edit the sources.list file by command “vi sources.list” and enter “i” to change to insert mode, appending this text at the end:

deb http://nginx.org/packages/ubuntu xenial nginx

deb-src http://nginx.org/packages/ubuntu xenial nginx



After appending lines, enter “:wq” to save the file.

1. Update the NGINX software by “sudo apt-get update”.



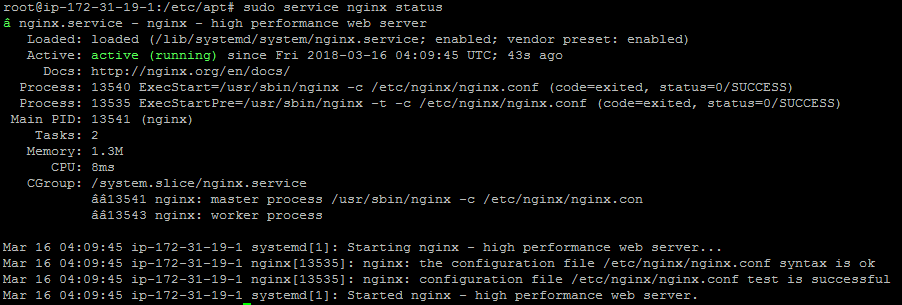
1. Install NGINX by “sudo apt-get install nginx”, type “Y” when prompted.



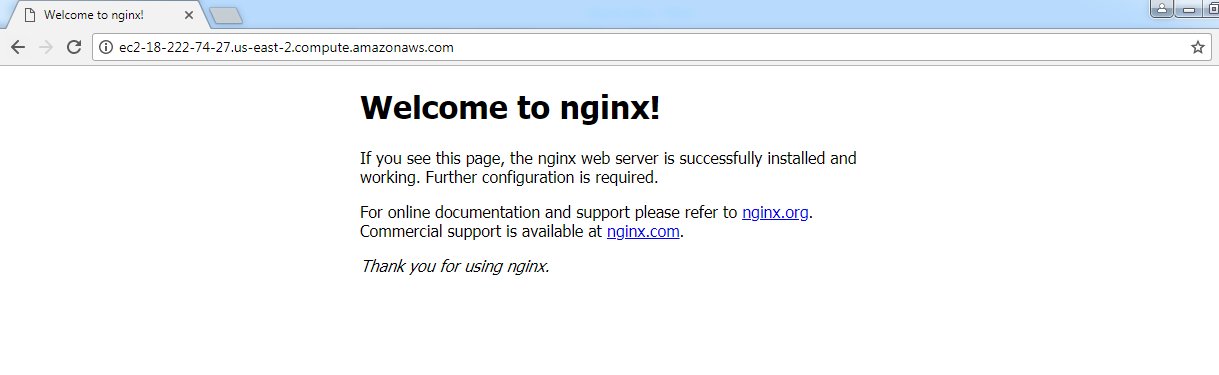
1. Start NGINX by “sudo service nginx start”.



1. Check running status of NGINX by “sudo service nginx status”.



1. Verify the nginx from browser, go to Amazon EC2 Dashboard, copy the Public DNS address and paste it browser.



## Setting Proxy Server on AWS Ubuntu Machine.

1. Connect to AWS Ubuntu machine by using putty.
2. Change user to root by “sudo root” and enter password.
3. Change to directory by “cd /etc/nginx/conf.d”.
4. Backup “default.conf” file by command “cp default.conf default.conf.bak”.
5. Edit “default.conf” file by “vi default.conf” and press “i” to change to insert mode.
6. Append below lines to “default.conf” file and save file by enter “:wq”.

server {

listen 80;

location / {

proxy\_pass http://localhost:5000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection keep-alive;

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

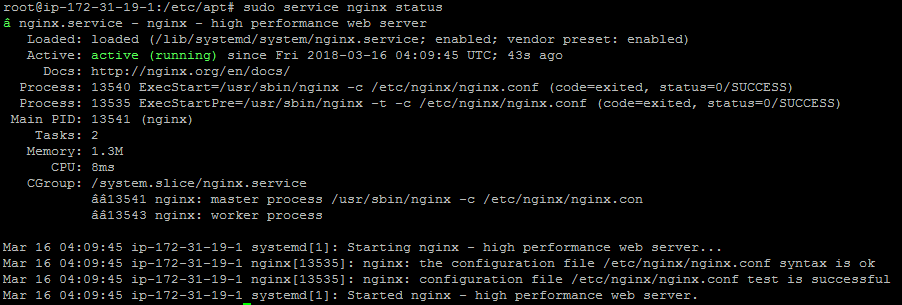
}

}

1. Restart nginx server by “sudo service nginx restart”.

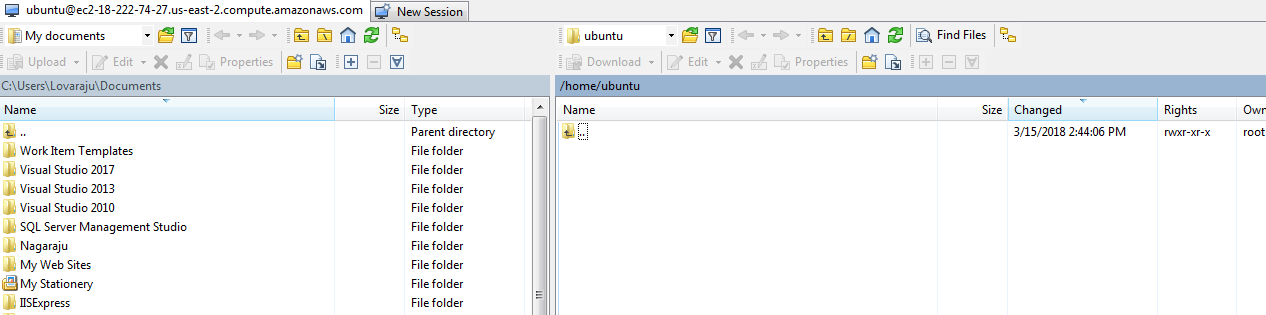


1. Check running status of NGINX by “sudo service nginx status”.

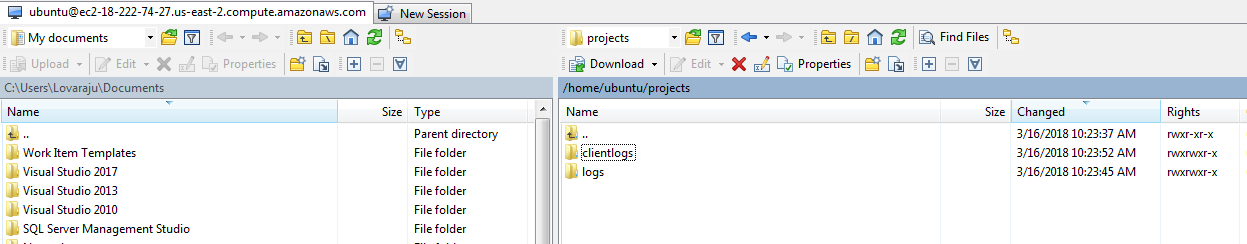


## Running OPC Server on AWS Ubuntu Machine.

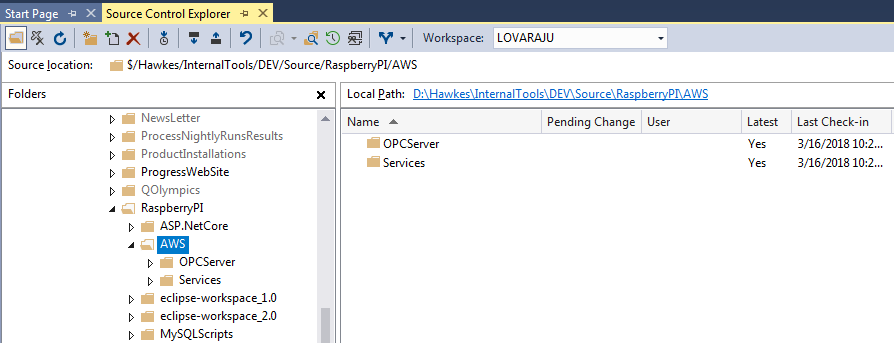
1. Connect to AWS Ubuntu machine by using WinSCP for file transferring.

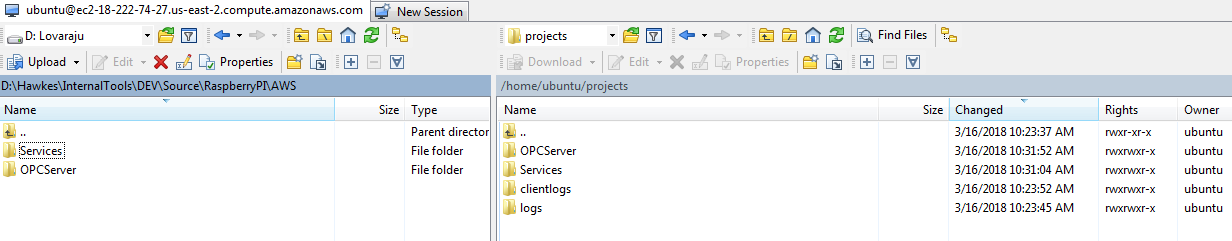


1. Create a new folder with name “projects” at /home/Ubuntu.
2. Open projects folder, create folders “logs”, “clientlogs”.



1. Copy “OPCServer” and “Services” folders from TFS location: $/Hawkes/InternalTools/DEV/Source/RaspberryPI/AWS to AWS Ubuntu machine location: /home/Ubuntu/projects.





1. Connect to AWS Ubuntu machine by using putty.
2. Change user to root by “sudo root” and enter password.
3. Change to directory by “cd /home/ubuntu/projects/Services”.



1. Copy all the services to system folder by “sudo cp \*.service /lib/systemd/system”.



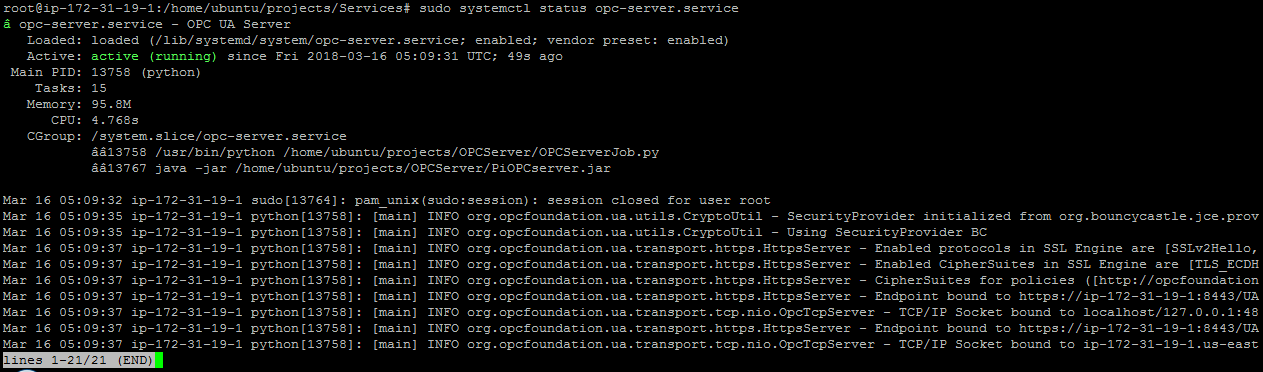
1. Reload all services by “sudo systemctl daemon-reload”.



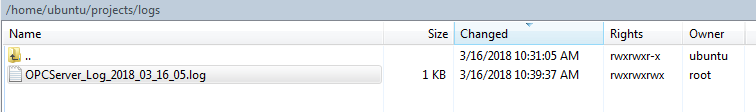
1. Start OPCServer by “sudo systemctl start opc-server.service”, to enable this service in machine reboot by “sudo systemctl enable opc-server.service”.



1. Check OPCServer running status by “sudo systemctl status opc-server.service”.

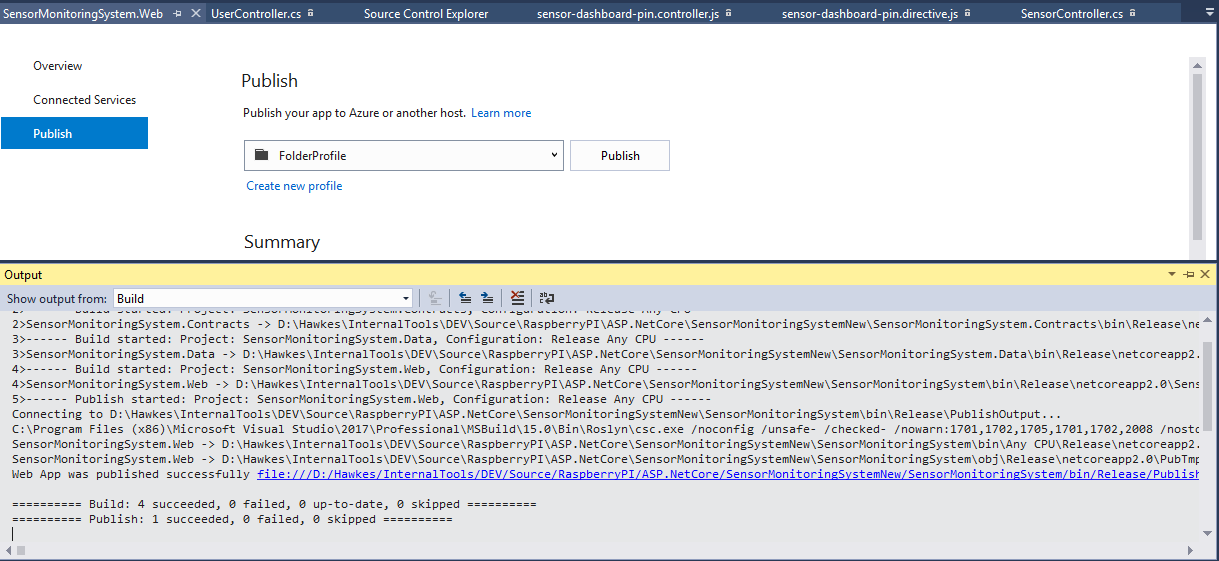


1. Also, you can verify OPC Server running status by logs. Connect to WinSCP, navigate to folder “/home/Ubuntu/projects/logs”, and check whether the logs are created or not. If logs are created successfully, OPC server working fine.

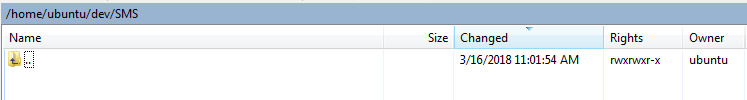


## Running ASP.NET Core 2.0 Website on AWS Ubuntu Machine.

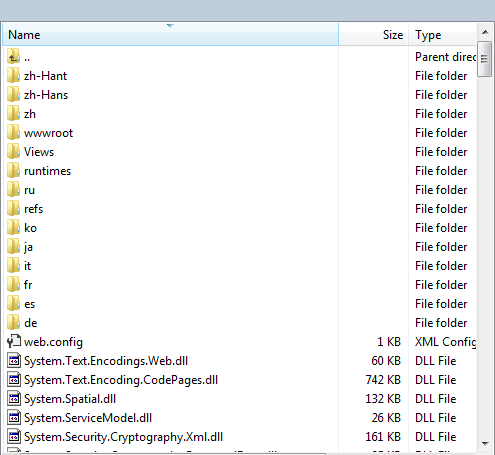
1. Connect to TFS location: $/Hawkes/InternalTools/DEV/Source/RaspberryPI/ASP.NetCore/SensorMonitoringSystemNew, open solution “SensorMonitoringSystem.sln” with Visual Studio 2017.
2. Clean and build entire solution.
3. Publish solution to local folder by using FolderProfile.



1. Connect to AWS Ubuntu machine with WinSCP, navigate folder “/home/Ubuntu”. Create folder “dev”, navigate to it. Create folder “SMS”, navigate to it.



1. Navigate to website publish folder from right panel of WinSCP.



1. Copy all files from publish folder to “/home/Ubuntu/dev/SMS”.
2. Connect to AWS Ubuntu machine by using putty.
3. Change user to root by “sudo root” and enter password.
4. Change to directory by “cd /home/ubuntu/projects/Services”.



1. Copy all the services to system folder by “sudo cp \*.service /lib/systemd/system”.



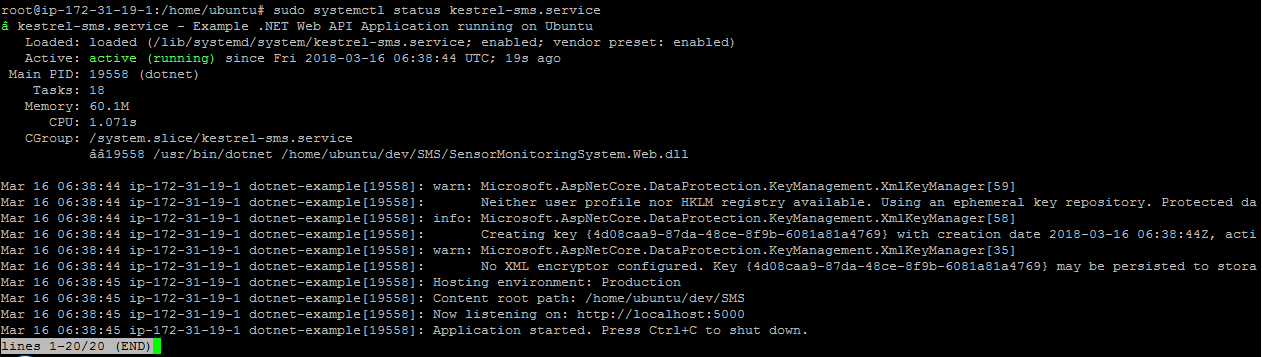
1. Reload all services by “sudo systemctl daemon-reload”.



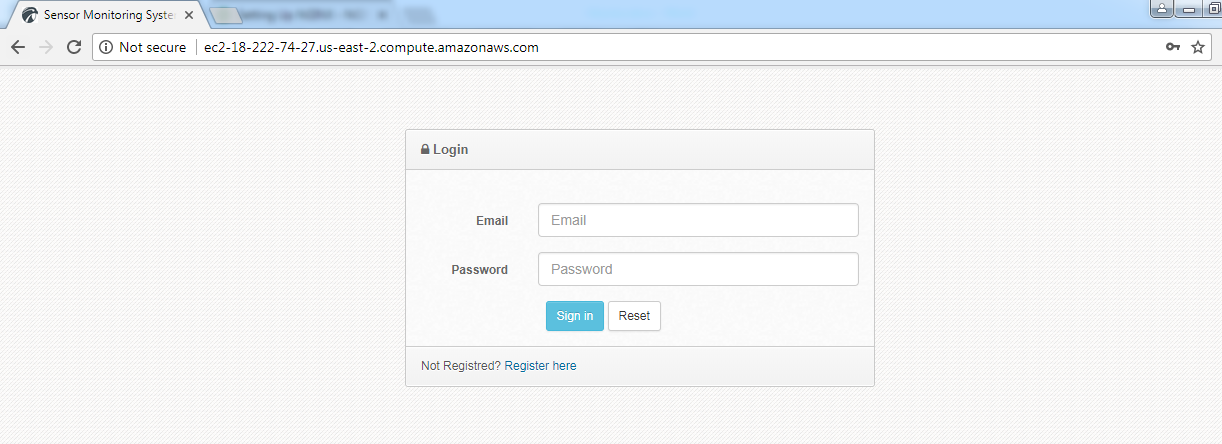
1. Start website by “sudo systemctl start kestrel-sms.service”, to enable this service in machine reboot by “sudo systemctl enable kestrel-sms.service”.



1. Check website running status by “sudo systemctl status kestrel-sms.service”.



1. Open website from browser, go to Amazon EC2 Dashboard, copy the Public DNS address and paste it browser.



## Running Email service on AWS Ubuntu Machine.

1. Copy folder “SendErrorReports” from TFS: $/Hawkes/InternalTools/DEV/Source/RaspberryPI/ASP.NetCore/SendErrorReports to AWS Ubuntu machine location “/home/Ubuntu/dev”.
2. Connect to AWS Ubuntu machine by using putty.
3. Change directory by “cd dev/ SendErrorReports”.
4. Clean and build by commands “dotnet clean”, “dotnet restore”, “dotnet build” and “dotnet publish”.









1. Change user to root by “sudo root” and enter password.
2. Change to directory by “cd /home/ubuntu/projects/Services”.



1. Copy all the services to system folder by “sudo cp \*.service /lib/systemd/system”.



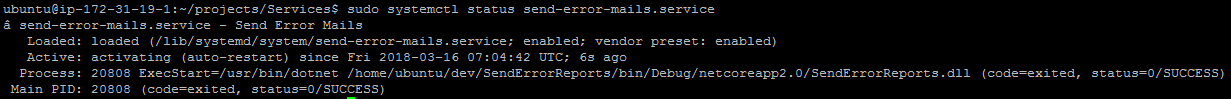
1. Reload all services by “sudo systemctl daemon-reload”.



1. Start e-mail service by “sudo systemctl start send-error-mails.service”, to enable this service in machine reboot by “sudo systemctl enable send-error-mails.service”.



1. Check e-mail service running status by “sudo systemctl status send-error-mails.service”.



## References:

<https://www.nginx.com/blog/setting-up-nginx/>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>