**Trial setup WebRTC with Wowza Streaming Engine in AWS EC2 Instance t2 micro and M3 Large.**

**AWS Instance t2micro and M3 Large on OS Ubuntu**

**Media Service Requirements**

1. Get a Wowza Streaming Engine To use Wowza Streaming Engine™ software, you need a valid license. You can: Register for a free standard trial or developer trial to receive a trial license key by email.

Purchase a monthly, annual or perpetual license.

Chose the wowza software on base your Operating System. Like

**Windows OS installation:**- To install Double-click the WowzaStreamingEngine-4.7.6-windows-installer.exe installer file and follow the onscreen instructions.

**Mac System:-** To install Open WowzaStreamingEngine-4.7.6-osx-installer.dmg.Double-click the installer package icon and follow the onscreen instructions.

**Linux System :-** To install sudo chmod +x WowzaStreamingEngine-4.7.6-linux-x64-installer.run, sudo ./WowzaStreamingEngine-4.7.6-linux-x64-installer.run, Follow the onscreen instructions.

Link for Wowza Application :- <https://www.wowza.com/downloads/WowzaStreamingEngine-4-7-6/WowzaStreamingEngine-4.7.6-linux-x64-installer.run>

**wowza steam engine default location :- /usr/local/WowzaStreamingEngine-4.7.5/**

Note :- Every changes in Wowza steam engine required to restart the wowza steam engine.

**Command to start or stop the Wowza steam engine**:-

sudo service WowzaStreamingEngine start

sudo service WowzaStreamingEngine stop

To run Wowza™ Transcoder on Windows Server 2008 or 2012 the following components are required: .NET Framework 3.5.1 Desktop Experience

2. Download the Wowza Streaming Engine InstalleWowza Streaming Engine's WebRTC implementation supports the following codecs:

Video: VP8, VP9, H.264

Audio: Opus, Vorbis, PCMU, PCMA

1. https://www.wowza.com/downloads/user/5b630b395380e/WowzaStreamingEngine-Update-4.7.6.zip

WebRTC Preview AddOn package contents

The WebRTC AddOn package includes the following folders and files:

html/publish: HTML example of WebRTC publishing

html/play: HTML example of playing WebRTC content

html/chat: HTML example of WebRTC chat

Requirements :-

1. Wowza setup we had required Wowza key and key is available for 30 day, for the key we had create account in wowza portal, The information we had to share in the wowza steam engine”First Name, Last name, email id, company name, and Password.
2. After the confirmation of Wowza email, we can login in wowza portal. And download the Wowza steam enginee as OS”Linux, Windows, Mac”.

1. **Install the WebRTC Preview AddOn**

**Default Direcotry of Wowza:-** /usr/local/WowzaStreamingEngine-4.7.5/

Enable the preview.

To do this, edit [install-dir]/conf/Server.xml and add the following property to the <Properties> container at the bottom of Server.xml:

<Property>

<Name>webrtc76de5tceo3l18xdh9e7ga</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

Edit [install-dir]/conf/VHost.xml and make the following changes:

Configure HTTP Provider for SDP exchange: Add the following XML snippet as the second-to-last entry in the list of HTTPProviders for HostPort entry 443 (SSL port). It's important that it be the second-to-last entry.

<HTTPProvider>

<BaseClass>com.wowza.wms.webrtc.http.HTTPWebRTCExchangeSessionInfo</BaseClass>

<RequestFilters>\*webrtc-session.json</RequestFilters>

<AuthenticationMethod>none</AuthenticationMethod>

</HTTPProvider>

1. Configure WebRTC streaming: Add the following properties to the <Properties> container at the end of the file (see Description, below, for necessary modifications to property values).

<Property>

<Name>webrtcKeyStorePath</Name>

<Value>${com.wowza.wms.context.VHostConfigHome}/conf/[streamlock-filename].jks</Value>

<Type>String</Type>

</Property>

<Property>

<Name>webrtcKeyStorePassword</Name>

<Value>[streamlock-password]</Value>

<Type>String</Type>

</Property>

Description

webrtcKeyStorePath:

Path to StreamLock certificate that will be used for WebRTC DTLS handshake. Replace [streamlock-filename] with your StreamLock certificate filename.

webrtcKeyStorePassword: Password for StreamLock certificate that will be used for WebRTC DTLS handshake. Replace [streamlock-password] with your StreamLock certificate password.

Create an application for live streaming named webrtc:

Create a WebRTC applications folder ([install-dir]/applications/webrtc) and a WebRTC configuration folder ([install-dir]/conf/webrtc)

Copy the [install-dir]/conf/live/Application.xml file to the new [install-dir ]/conf/webrtc folder.

Edit the [install-dir]/conf/webrtc/Application.xml file and change the <Application>/<Name> to webrtc:

<Application>

<Name>webrtc</Name>

Configure the webrtc live application. Edit [install-dir]/conf/webrtc/Application.xml and add the following properties to the <Properties> container at the end of the file (see Description, below, for necessary modifications to property values).

<Property>

<Name>webrtcEnablePublish</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<Name>webrtcEnablePlay</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<Name>webrtcEnableQuery</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<!-- comma separated list of IP addresses and the transport information. For multiple IP's use a pipe character to separate the lists -->

<Name>webrtcIceCandidateIpAddresses</Name>

<Value>[wowza-streaming-engine-external-ip-address],udp</Value>

<Type>String</Type>

</Property>

<Property>

<Name>webrtcUDPBindAddress</Name>

<Value>0.0.0.0</Value>

<Type>String</Type>

</Property>

<Property>

<Name>webrtcPreferredCodecsAudio</Name>

<Value>opus,vorbis,pcmu,pcma</Value>

<Type>String</Type>

</Property>

<Property>

<Name>webrtcPreferredCodecsVideo</Name>

<Value>vp8,h264</Value>

<Type>String</Type>

</Property>

<Property>

<Name>webrtcDebugLog</Name>

<Value>false</Value>

<Type>Boolean</Type>

</Property>

Description

webrtcEnablePublish: Enable WebRTC publishing to this application. Required for the publish and chat example HTML applications.

webrtcEnablePlay: Enable WebRTC playback from this application. Required for the play and chat example HTML applications.

webrtcEnableQuery: Enable query of published stream names for this application. Required for chat example HTML application.

webrtcIceCandidateIpAddresses: IP address, transport, and port used for WebRTC streaming.

For the transmission control protocol (TCP), the value should be set to [wowza-streaming-engine-external-ip-address],tcp,[port] where [wowza-streaming-engine-external-ip-address] is the external IP address of the Wowza Streaming Engine instance and [port] is one of the non-SSL-protected streaming HostPort entries defined in [install-dir]/conf/VHost.xml. For example, to stream over port 1935, the entry would be 66.175.168.127,tcp,1935.

For the user datagram protocol (UDP), the value should be set to [wowza-streaming-engine-external-ip-address],udp where [wowza-streaming-engine-external-ip-address] is the external IP address of the Wowza Streaming Engine instance. The port is dynamically assigned for UDP delivery.

For multiple IP addresses, use a pipe character to separate the lists.

webrtcUDPBindAddress: Local IP address of the network card you want to use for WebRTC UDP traffic. (This value is not used if streaming WebRTC over TCP.) For UDP delivery in general, it's OK to leave this property blank. The property is only needed if the server has multiple network interfaces. For some network situations, like running on a cloud instance, a value of 0.0.0.0 would be best instead of the local IP address of the network card to prevent connection problems.

webrtcPreferredCodecsAudio: Comma-deliniated list of audio codecs, in order of preference, for stream ingestion. The default is opus,vorbis,pcmu,pcma.

webrtcPreferredCodecsVideo: Comma-deliniated list of video codecs, in order of preference for stream ingestion. The default is vp8,h264, but valid values are vp8, vp9, and h264. If you want to stream in Chrome using VP9, change the property value to vp9,vp8,h264.

webrtcDebugLog: Enable WebRTC debug logging.

Add the following properties to the <RTP>/<Properties> container in the [install-dir]/conf/webrtc/Application.xml file (see Description, below, for necessary modifications to property values):

<Property>

<Name>rtpForceH264Constraint</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<Name>rtpForceH264ConstraintValue</Name>

<Value>192</Value>

<Type>Integer</Type>

</Property>

<Property>

<Name>rtpUseLowestH264Constraint</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<Name>rtpUseHighestH264Constraint</Name>

<Value>false</Value>

<Type>Boolean</Type>

</Property>

Description

rtpForceH264Constraint: Set to true to allow the SDP file that is returned to contain different H264 constraints than the stream contains. In most cases, setting rtpForceH264Constraint to true enables WebRTC to play without issue. The default value is false. Note that the Mozilla FireFox and Google Chrome browsers require additional constraint fields to be set.

rtpForceH264ConstraintValue: Use to set the constraint fields. The default value, 192, works in most circumstances. Other valid values are 128, 224, and 240.

rtpUseLowestH264Constraint: Set to true to compare the initial codec data and the first video unit to the profile data and use the lowest value. Use this (or rtpUseHighestH264Constraint) to aid in playing the stream when an encoder doesn't send consistent codec information for the stream.

rtpUseHighestH264Constraint: Set to true to compare the initial codec data and the first video unit to the profile data and use the highest value. Use this (or rtpUseLowestH264Constraint) to aid in playing the stream when an encoder doesn't send consistent codec information for the stream.

Note: The rtpUseLowestH264Constraint and rtpUseHighestH264Constraint properties can't be used simultaneously. If both are set to true, rtpUseHighestH264Constraint is used.

HTML examples

The WebRTC Preview package includes the following HTML examples:

Publish example: The publish example shows how to publish a media stream to Wowza Streaming Engine using WebRTC. Change the SDP URL to reflect the domain name of your StreamLock or SSL certificate. This URL is used to exchange SDP information. You can also change the video and audio bitrate in addition to the frame rate. The frame rate option only works for VP9 and H.264 video codecs when using Chrome version 55.0.1 or later.

Play example: The play example shows how to playback a media stream from Wowza Streaming Engine using WebRTC. Change the SDP URL to reflect the domain name of your StreamLock or SSL certificate. This URL is used to exchange SDP information.

Chat example: The chat example shows how to set up a multi-person chat session. The example is quite simple and is intended to be an example of what it possible rather than a finished work. Change the SDP URL to reflect the domain name of your StreamLock or SSL certificate. This URL is used to exchange SDP information.

Hosting example HTML over HTTPS

There are several options for HTTPS hosting of the HTML examples:

The examples are hosted on www.wowza.com: WebRTC Preview HTML Examples. This is a great option if you don't need to modify the HTML or JavaScript code.

Host the HTML examples on your own web server that's secured using HTTPS. To install, copy the contents of the html folder to your web server.

Host the HTML examples on Wowza Streaming Engine using the built-in HTTPProviderSimpleWebServer HTTP Provider by doing the following:

Edit [install-dir]/conf/VHost.xml and add the following HostPort entry to the HostPortList (replace [streamlock-filename] and [streamlock-password] with your StreamLock filename and password):

<HostPort>

<Name>Admin WebRTC</Name>

<Type>Admin</Type>

<ProcessorCount>${com.wowza.wms.TuningAuto}</ProcessorCount>

<IpAddress>\*</IpAddress>

<Port>9443</Port>

<HTTPIdent2Response></HTTPIdent2Response>

<SSLConfig>

<KeyStorePath>${com.wowza.wms.context.VHostConfigHome}/conf/[streamlock-filename].jks</KeyStorePath>

<KeyStorePassword>[streamlock-password]</KeyStorePassword>

<KeyStoreType>JKS</KeyStoreType>

<DomainToKeyStoreMapPath></DomainToKeyStoreMapPath>

<SSLProtocol>TLS</SSLProtocol>

<Algorithm>SunX509</Algorithm>

<CipherSuites></CipherSuites>

<Protocols></Protocols>

</SSLConfig>

<SocketConfiguration>

<ReuseAddress>true</ReuseAddress>

<ReceiveBufferSize>16000</ReceiveBufferSize>

<ReadBufferSize>16000</ReadBufferSize>

<SendBufferSize>16000</SendBufferSize>

<KeepAlive>true</KeepAlive>

<AcceptorBackLog>100</AcceptorBackLog>

</SocketConfiguration>

<HTTPStreamerAdapterIDs></HTTPStreamerAdapterIDs>

<HTTPProviders>

<HTTPProvider>

<BaseClass>com.wowza.wms.http.HTTPProviderSimpleWebServer</BaseClass>

<RequestFilters>webrtc\*</RequestFilters>

<AuthenticationMethod>none</AuthenticationMethod>

</HTTPProvider>

</HTTPProviders>

</HostPort>

Create the folder [install-dir]/htdocs/webrtc and copy the contents of the html folder into this folder.

When using Wowza Streaming Engine to host the example files, use the following URL to access them: https://[streamlock-domain-name]:9443/webrtc/[path-to-example-file].

For example, if the StreamLock domain name is 123456.streamlock.net, the URL for the publish HTML example would be: https://123456.streamloack.net:9443/webrtc/publish/index.html.

How to...

Set up a live stream repeater (origin/edge) for use with WebRTC streaming

For detailed information about how to set up a live repeater edge application, see Wowza edge. Make sure to enable the RTSP/RTP playback type.

Each edge server and the edge application must be configured for WebRTC streaming per the above instructions. Currently, we don't have explicit instructions on how to do load balancing between the edge servers. It should be possible to enhance the provided play code example to first make a request to a third-party load balancer to get the address of the least loaded edge server and then alter the WebSocket URL to point to that server dynamically. We hope to add support for load balancing WebRTC connections in a future version of our dynamic load balancing solution.

WebRTC edge streams will function like other origin/edge streams. The edge server will fetch the stream dynamically from the origin upon the first request for it. The stream will be dropped from the edge server a few seconds after the last viewer stops watching it.

Improve playback of WebRTC streams on an edge server

In some cases, playback of WebRTC streams on edge servers can become choppy. The following settings may help.

Important! The following configuration may address the playback issues, but it will significantly increase CPU usage. Use caution when applying these settings.

Note: This applies only to playback on edge servers. For more information about configuring a live repeater, see Set up a live stream repeater (origin/edge) for use with WebRTC streaming.

In a text editor, open [install-dir]/conf/[webrtc]/Application.xml, where [webrtc] is the name of the Wowza Streaming Engine live origin application, and add the following properties to the <Streams>/<Properties> container.

<Property>

<Name>onFlushNotifyClients</Name>

<Value>true</Value>

<Type>Boolean</Type>

</Property>

<Property>

<Name>flushInterval</Name>

<Value>25</Value>

<Type>Integer</Type>

</Property>

Repeat step one for [install-dir]/conf/[webrtc2]/Application.xml, where [webrtc2] is the name of the Wowza Streaming Engine live edge application.

Note: These properties must be added to both the Live Origin and Live Edge Wowza Streaming Engine applications.

Restart Wowza Streaming Engine to apply your changes.

Listen for and reject WebRTC session create and destroy commands in a module

import com.wowza.wms.module.\*;

import com.wowza.wms.rtp.model.\*;

public class ModuleListenWebRTCSession extends ModuleBase

{

public void onRTPSessionCreate(RTPSession rtpSession)

{

if (rtpSession.isWebRTC())

{

getLogger().info("ModuleListenWebRTCSession.onRTPSessionCreate["+rtpSession.getSessionId()+"]");

// If a query parameter is added to the stream name it should be shown here

// An example would be

//

// Stream Name: myStream?param1=value1

//

// should output

//

// ModuleListenWebRTCSession.RTPStream.QueryParameter[param1=value]

getLogger().info("ModuleListenWebRTCSession.RTPStream.QueryParameter["+rtpSession.getRTSPStream().getStreamQueryStr()+"]");

// Call rejectSession to stop the session immediately

//rtpSession.rejectSession();

}

}

public void onRTPSessionDestroy(RTPSession rtpSession)

{

if (rtpSession.isWebRTC())

{

getLogger().info("ModuleListenWebRTCSession.onRTPSessionDestroy["+rtpSession.getSessionId()+"]");

}

}

}

Determine if an RTPSession is a WebRTC session

boolean RTPSession.isWebRTC();

Get a WebRTCSession interface to/from RTPSession

WebRTCSession rtpSession.getWebRTCSession();

RTPSession WebrtcSession.getRTPSession();

Different ways to get WebRTC session information from IApplicationInstance

List<RTPSession> getWebRTCSessions()

List<RTPSession> getWebRTCSessions(String streamName)

int getWebRTCSessionCount()

int getWebRTCSessionCount(String streamName)

Map<String, Integer> getWebRTCSessionCountsByName()

Forcefully disconnect a WebRTC session

vhost.getWebRTCContext().shutdownSession(webRTCSession.getSessionId());

Subclass HTTPWebRTCExchangeSessionInfo to secure WebRTC playback and publishing

The following code is an incomplete example that demonstrates how to subclass the WebRTC HTTPProvider HTTPWebRTCExchangeSessionInfo and override several methods to secure WebRTC publishing and playback. The path to your new class must replace the class path for the HTTP Provider configured above. You can pass security information from HTML/JavaScript to Wowza Streaming Engine either through query parameters or through the JSON data that's passed over the WebSocket. A userData object, as part of the example JavaScript, can be enhanced to hold this data. You can control publishing, playback, and query through the commented-out commandControl.canPlay, commandControl.canPublish, and commandControl.canQuery variables.

import java.util.\*;

import com.wowza.util.\*;

import com.wowza.wms.application.\*;

import com.wowza.wms.logging.\*;

import com.wowza.wms.webrtc.http.\*;

import com.wowza.wms.websocket.model.\*;

public class HTTPWebRTCExchangeSessionInfoCustom extends HTTPWebRTCExchangeSessionInfo

{

private static final Class<HTTPWebRTCExchangeSessionInfoCustom> CLASS = HTTPWebRTCExchangeSessionInfoCustom.class;

private static final String CLASSNAME = "HTTPWebRTCExchangeSessionInfoCustom";

@Override

protected void websocketSessionCreate(IWebSocketSession webSocketSession)

{

super.websocketSessionCreate(webSocketSession);

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".websocketSessionCreate: "+webSocketSession.getSessionId());

}

@Override

protected void websocketSessionDestroy(IWebSocketSession webSocketSession)

{

super.websocketSessionDestroy(webSocketSession);

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".websocketSessionDestroy: "+webSocketSession.getSessionId());

}

@Override

protected void authenticateRequest(CommandContext commandContext, CommandControl commandControl)

{

super.authenticateRequest(commandContext, commandControl);

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".authenticateRequest: reqURI:"+commandContext.reqURI);

int qloc = commandContext.reqURI.indexOf("?");

if (qloc >= 0)

{

String queryStr = commandContext.reqURI.substring(qloc+1).trim();

if (queryStr.length() > 0)

{

Map<String, String> queryMap = HTTPUtils.splitQueryStr(queryStr);

for(Map.Entry<String, String> entry : queryMap.entrySet())

{

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".authenticateRequest: queryMap["+entry.getKey()+"]: "+entry.getValue());

}

}

}

IApplicationInstance appInstance = commandContext.commandRequest.getApplicationInstance(commandContext.vhost);

if (appInstance != null)

{

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".authenticateRequest: application: "+appInstance.getContextStr());

// application level properties

WMSProperties prop = appInstance.getProperties();

}

Map<String, Object> jsonEntries = commandContext.commandRequest.getJSONEntries();

if (jsonEntries != null)

{

Map<String, Object> userData = (Map<String, Object>)jsonEntries.get("userData");

if (userData != null)

{

for(Map.Entry<String, Object> entry : userData.entrySet())

{

WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME+".authenticateRequest: userData["+entry.getKey()+"]: "+entry.getValue());

}

}

}

// Perform authentication here and set

// commandControl.canPlay

// commandControl.canPublish

// commandControl.canQuery

}

}

**Record a live WebRTC stream**

To record a live WebRTC stream, do the following:

Install the LiveStreamRepeater module. For more information, see How to record live streams (Wowza Streaming Engine).

Configure your WebRTC live application. Open [install-dir]/conf/webrtc/Application.xml in a text editor, and make the following changes:

Set the webrtcPreferredCodecsVideo property's <Value> to h264 only:

<Property>

<Name>webrtcPreferredCodecsVideo</Name>

<Value>h264</Value>

<Type>String</Type>

</Property>

Add the liveStreamRecorderDefaultAudioSearchPosition and liveStreamRecorderSkipKeyFrameUntilAudioTimeout properties to the end of the last <Properties> section in the Application.xml file:

<Property>

<Name>streamRecorderDefaultAudioSearchPosition</Name>

<Type>Boolean</Type>

<Value>false</Value>

</Property>

<Property>

<Name>streamRecorderSkipKeyFrameUntilAudioTimeout</Name>

<Type>Integer</Type>

<Value>10000</Value>

</Property>

Where:

streamRecorderDefaultAudioSearchPosition, when set to false, causes the LiveStreamRecorder module to search the packets and identify the nearest audio packet to a keyframe. This keeps the audio and video synchronized in the recording. If set to true, the LiveStreamRecorder module assumes the incoming audio and video are interweaved and no actions are taken to ensure synchronization.

streamRecorderSkipKeyFrameUntilAudioTimeout is the maximum length of time, in milliseconds, that you want to allow the LiveStreamRecorder to search for audio packets before you start recording. The recording will start as soon as an audio packet is found or when the liveStreamRecorderSkipKeyFrameUntilAudioTimeout value is reached. The default value, 10000, starts recording after 10 seconds. Your recording may not include the first few seconds on the stream.

Note: The streamRecorderDefaultAudioSearchPosition and streamRecorderSkipKeyFrameUntilAudioTimeout properties should always be used together. Don't configure one without configuring the other, too.

In Wowza Streaming Engine Manager, enable the Transcoder feature and configure the Transrate (Default) template Encoding preset to passthrough the incoming video and convert the incoming audio to AAC. For more information, see How to set up and run Wowza Transcoder for live streaming.

Both your source stream and your transcoded stream with AAC audio will be recorded, but only the transcoded stream will be usable.

Wowza Team Support subject and script add on following file:-

**Record WebRTC**:- You will need to follow the below steps to create a recording:  
  
The easy way to do this is go to [install-dir]/conf/webrtc/Application.xml  
and set this value to enable the transcode template since you are changing codecs.  
  
<Transcoder>  
      <!-- To turn on transcoder set to: transcoder -->  
      <LiveStreamTranscoder>transcoder</LiveStreamTranscoder>  
      <!-- [templatename].xml or ${SourceStreamName}.xml -->  
      <Templates>transrate.xml</Templates>  
      <ProfileDir>${com.wowza.wms.context.VHostConfigHome}/transcoder/profiles</ProfileDir>  
      <TemplateDir>${com.wowza.wms.context.VHostConfigHome}/transcoder/templates</TemplateDir>  
      <Properties>  
      </Properties>  
  </Transcoder>  
  
Then you will go to [install-dir]/transcoder/templates/transrate.xml  
You will need to enable one rendition to record by selecting <Enable>true</Enable>:  
  
for example:  
  
<Encode>  
    <Enable>true</Enable>  
    <Name>360p</Name>  
    <StreamName>mp4:${SourceStreamName}\_360p</StreamName>  
    <Video>  
        <!-- H.263, H.264, H.265, VP8, VP9, PassThru, Disable -->  
        <Codec>H.264</Codec>  
        <!-- default, QuickSync, CUDA, NVENC -->  
        <Implementation>default</Implementation>  
        <GPUID>-1</GPUID>  
        <FrameSize>  
      <!-- letterbox, fit-width, fit-height, crop, stretch, match-source -->  
      <FitMode>fit-height</FitMode>  
      <Width>640</Width>  
      <Height>360</Height>  
      <!-- <Crop>0,0,0,0</Crop> -->  
      <!-- <SourceRectangle>0,0,320,240</SourceRectangle> -->  
        </FrameSize>  
        <!-- baseline, main, high -->  
        <Profile>main</Profile>  
        <Bitrate>850000</Bitrate>  
        <KeyFrameInterval>  
      <FollowSource>false</FollowSource>  
      <Interval>60</Interval>  
        </KeyFrameInterval>  
        <Overlays>  
      <Overlay>  
          <Enable>false</Enable>  
          <Name>WowzaLogo</Name>  
          <Index>0</Index>  
          <ImagePath>${com.wowza.wms.context.VHostConfigHome}/content/wowzalogo.png</ImagePath>  
          <CheckForUpdates>false</CheckForUpdates>  
          <Opacity>100</Opacity>  
          <Location>  
        <X>4</X>  
        <Y>4</Y>  
        <Width>${ImageWidth}</Width>  
        <Height>${ImageHeight}</Height>  
        <!-- horiz: left, right, hcenter - vert: top, bottom, vcenter -->  
        <Align>left,top</Align>  
          </Location>  
      </Overlay>  
        </Overlays>  
        <Parameters>  
        </Parameters>  
    </Video>  
    <Audio>  
        <!-- AAC, Vorbis, Opus, PassThru, Disable -->  
        <Codec>AAC</Codec>  
        <Bitrate>96000</Bitrate>  
        <Parameters>  
        </Parameters>  
    </Audio>  
    <Properties>  
    </Properties>  
      </Encode>

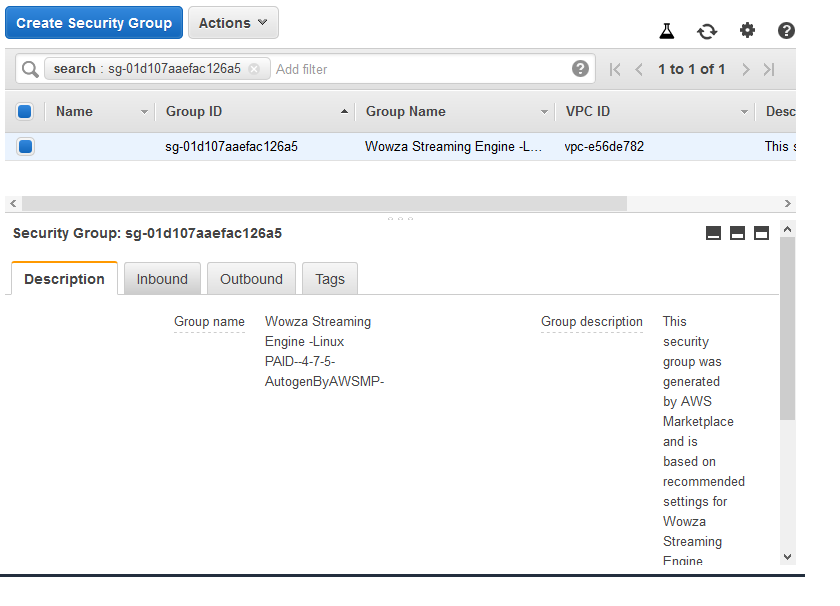
Note :- Please restart the webrtc Application for changes to take effect. Then you'll need to record the \_720p,\_360p,or \_160p and also other stream.

Amazon EC2 security groups for EC2 Linux instances. Access for the port out and in bond.

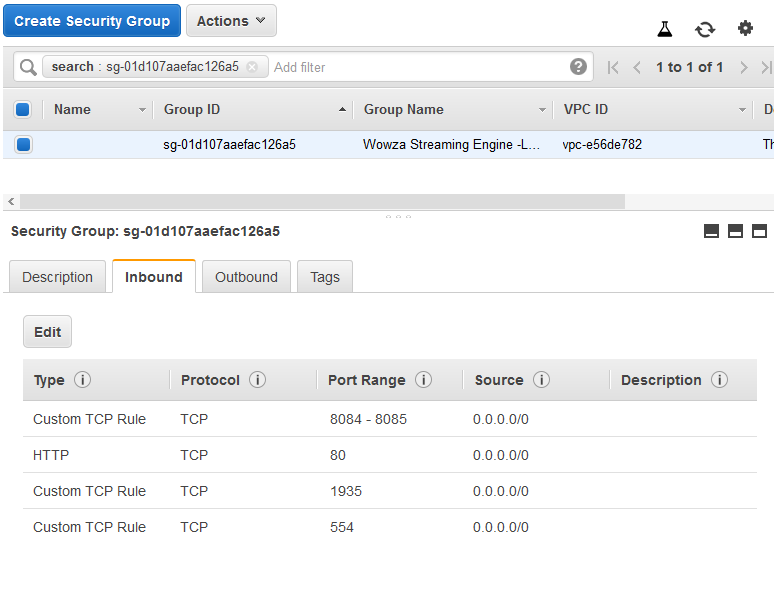
In this wowza steam configure in defferent-2 machine linke local to AWS Instance, we got lots of update version of wowza application.

On the Configure Security Group page, select the option to create a new security group, and then fill out the form to define the firewall rules for your instance in the selected region. For the purposes of this guide, add rules to open port 1935 in the firewall for RTMP streaming and the port range 8086-8088 for Wowza Streaming Engine Manager. To do this for RTMP / WEBRTC streaming, click Add Rule, select Custom TCP rule for the Type, enter 1935 in Port Range, and select Anywhere in Source. Repeat this step to enable the port range 8086-8088. For more information about additional rules to add for streaming and for managing the Streaming Engine software.

**inbound port opening** :-Login AWS console select the instance “Wowza-webrtc”. The click on security group, ,

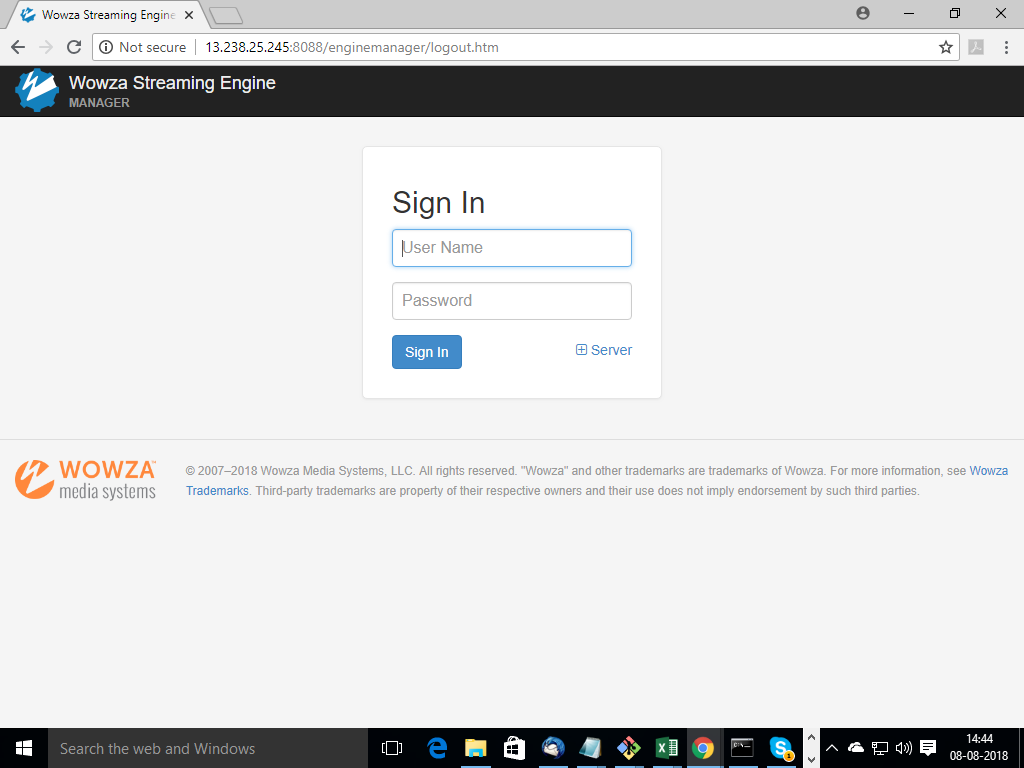


click on inbound tab, click on edit button. Enter the port no which you want enable “8088.1937,8087” Then click on save button,

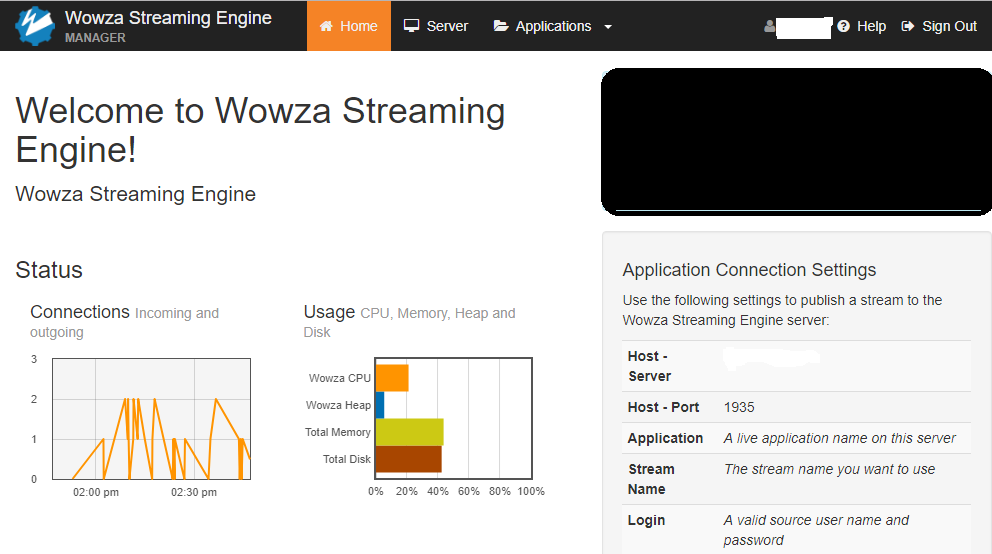


After this required the restart the EC2 instance restart, after the restart the changes will applicable,

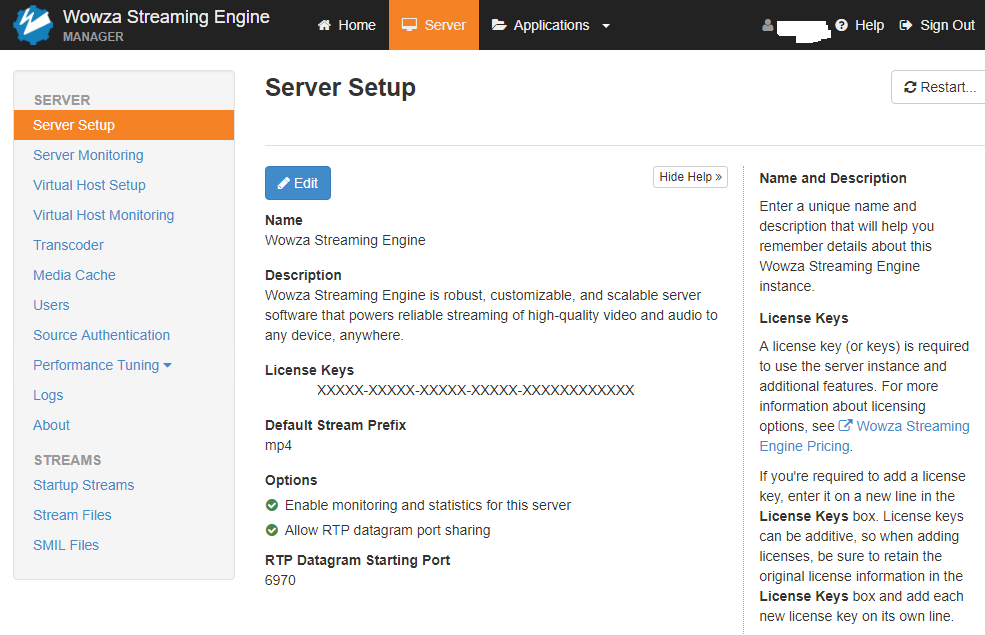
After The all configuration in local:- the web link:- 192.168.\*.\*8088 /enginemanager/Home.htm



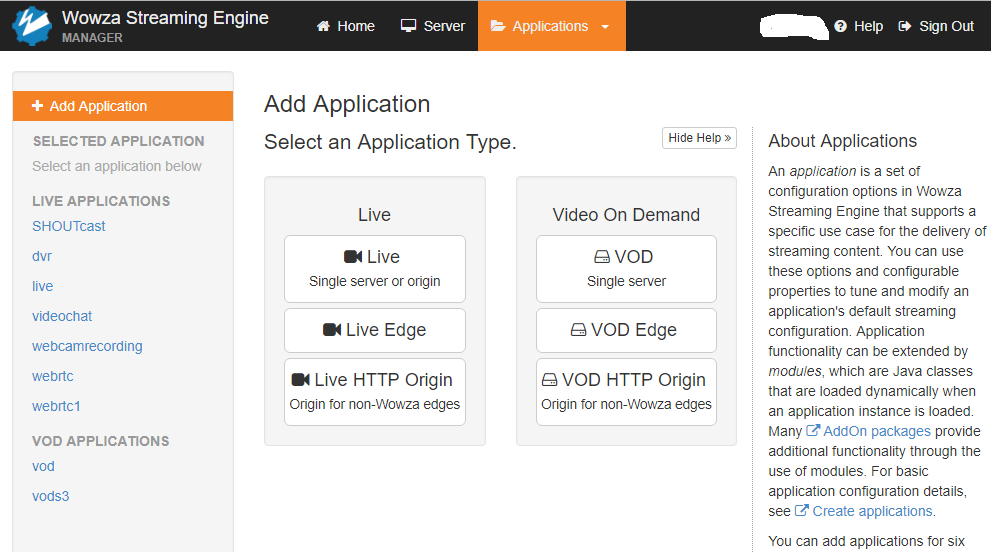
After the login Home page show the status and basic information show:-

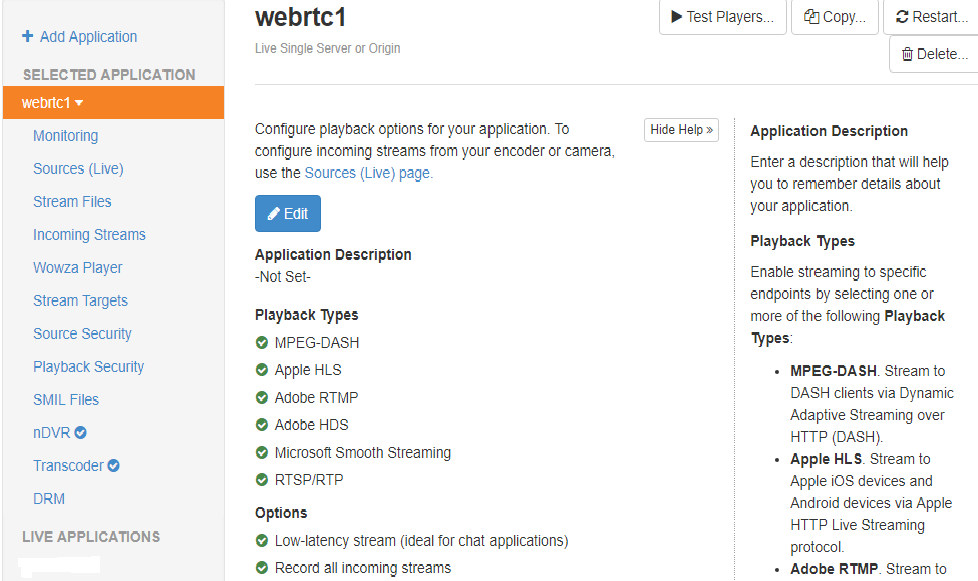


Server Tab show the information license and other information:-



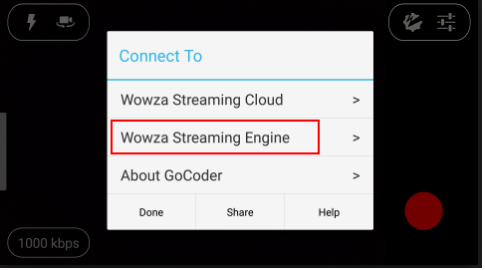
Application Tab:- used for enable disable and edit the options





**Gocoder App**:- Install the gocoder app for test the wowza steam videos





Access the wowza steam videos, touch the wowza icon in the app after that screen is appear for the details fill up:-

Touch the Wowza Streaming Engine,

1 Host option:- server ip like “202.145.152.24”It’s depend the wowza server local and public ip which wowza server is running” Port no :- 1935 “it’s by default for the access the wowza steam videos.

Then touch back

1. Application :- Appication Name :- webrtc1 “it’s depend which steam you want to run with wowza”. Steam Name:- Mysteam “steam name depend which type of steam name you want run” Then touch back
2. Source authentication :- Source Username :- john or [john@expmple.com](mailto:john@expmple.com) “it’s if it’s trial software the the email is mention here, which you received wowza key “. Source password :- \*\*\*\*\*\*\*\*\*. “it’s minimum 8 character and it’s depend what password you configure that time with wowza form fill up in the portal and get the access