

Deploying WCS with CloudFormation

- [Overview](#)
- [CloudFormation template example to deploy CDN](#)
- [WCS CDN deployment example using CloudFormation web console](#)

Overview

AWS CloudFormation allows to deploy cloud instances stacks by a certain template. Thus, a simple WCS CDN can be deployed for example. In this case, WCS update to the latest build and instance setup can be made with UserData scripts.

CloudFormation template example to deploy CDN

Below, there is the CloudFormation template example to deploy a simplest CDN of two WCS instances: Origin and Edge. The template allows:

- to choose [WCS AMI from AWS Marketplace](#), or Amazon Linux 2, Ubuntu 18.04 and [other supported OS AMI](#) as basic image
- to install Java 14 if necessary
- to install or update WCS to the latest build if necessary

CloudFormation example template to deploy WCS CDN of one Origin and one Edge

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "Create WCS CDN stack"
Parameters:
  KeyName:
    Description: "Name of an existing EC2 KeyPair to enable SSH access to the instance"
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: "must be the name of an existing EC2 KeyPair"
  InstanceName:
    Description: "Name of EC2 instance"
    Type: String
    ConstraintDescription: "must be a valid EC2 instance string name"
  InstanceType:
    Description: "Basic EC2 instance type"
    Type: String
    Default: m5.xlarge
    AllowedValues: [t1.micro, t2.nano, t2.micro, t2.small, t2.medium, t2.large,
      m1.small, m1.medium, m1.large, m1.xlarge,
      m2.xlarge, m2.2xlarge, m2.4xlarge,
      m3.medium, m3.large, m3.xlarge, m3.2xlarge,
      m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m4.10xlarge, m5.xlarge,
      c1.medium, c1.xlarge, c3.large, c3.xlarge, c3.2xlarge, c3.4xlarge, c3.8xlarge,
      c4.large, c4.xlarge, c4.2xlarge, c4.4xlarge, c4.8xlarge,
      g2.2xlarge, g2.8xlarge, r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge, r3.8xlarge,
      i2.xlarge, i2.2xlarge, i2.4xlarge, i2.8xlarge,
      d2.xlarge, d2.2xlarge, d2.4xlarge, d2.8xlarge,
      hi1.4xlarge, hs1.8xlarge, cr1.8xlarge, cc2.8xlarge, cgl.4xlarge]
    ConstraintDescription: "must be a valid EC2 instance type"
  ImageId:
    Description: "Basic instance ami (WebCallServer 5.2.944 AMI by default, mapped by region)"
    Type: String
    Default: WCSAMI
    ConstraintDescription: "must be a valid AMI ID"
  VpcId:
    Type: String
    Description: "VpcId of your existing Virtual Private Cloud (VPC)"
  SubnetId:
    Type: String
    Description: "SubnetId of an existing subnet in your Virtual Private Cloud (VPC)"
  SSHLocation:
    Description: "The IP address range that can be used to SSH to the EC2 instances"
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: (\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})
    ConstraintDescription: "must be a valid IP CIDR range of the form x.x.x.x/x"
```

```

JavaHeapSize:
  Description: "Maximum Java heap size in megabytes (2048m) or gigabytes (2g), 1024m by default"
  Type: String
  Default: 1024m
UpdateWCS:
  Description: "Update WCS to the latest build"
  Type: String
  Default: true
  ConstraintDescription: "must be true or false"
WCSLicense:
  Description: "WCS License key to activate (optional, if you do not use Marketplace AMI)"
  Type: String
  Default: ""
InstallJava:
  Description: "Java installation helper script. Do not change in wizard!"
  Type: String
  Default: |
    JAVA_CMD=`command -v java 2>/dev/null`
    if [[ -z $JAVA_CMD ]]; then
      rm -rf jdk*
      curl -s https://download.java.net/java/GA/jdk14.0.1/664493ef4a6946b186ff29eb326336a2/7/GPL/openjdk-
14.0.1_linux-x64_bin.tar.gz | tar -zx
      if [ -d jdk-14.0.1/bin ]; then
        mkdir -p /usr/java
        [ -d /usr/java/jdk-14.0.1 ] && rm -rf /usr/java/jdk-14.0.1
        mv -f jdk-14.0.1 /usr/java
        if [ -d /usr/java/jdk-14.0.1/bin ]; then
          rm -f /usr/java/default
          ln -sf /usr/java/jdk-14.0.1 /usr/java/default
          update-alternatives --install /usr/bin/java java /usr/java/jdk-14.0.1/bin/java 1
          update-alternatives --install /usr/bin/jstack jstack /usr/java/jdk-14.0.1/bin/jstack 1
          update-alternatives --install /usr/bin/jcmd jcmd /usr/java/jdk-14.0.1/bin/jcmd 1
          update-alternatives --install /usr/bin/jmap jmap /usr/java/jdk-14.0.1/bin/jmap 1
          update-alternatives --set java /usr/java/jdk-14.0.1/bin/java
          update-alternatives --set jstack /usr/java/jdk-14.0.1/bin/jstack
          update-alternatives --set jcmd /usr/java/jdk-14.0.1/bin/jcmd
          update-alternatives --set jmap /usr/java/jdk-14.0.1/bin/jmap
          echo "JDK 14 installed" >> $DEPLOY_LOG
        fi
      fi
    fi
StopPreviousWCS:
  Description: "Stop previously running WCS helper script. Do not change in wizard!"
  Type: String
  Default: |
    PID=`pgrep -f 'com.flashphoner.server.Server' | grep -v bash`
    if [ -n "$PID" ]; then
      systemctl stop webcallserver
    fi
UpdateToLatestWCS:
  Description: "WCS update to latest build helper script. Do not change in wizard!"
  Type: String
  Default: |
    # Check if WCS is installed, and install latest build if not
    if [ ! -f /usr/local/FlashphonerWebCallServer/bin/webcallserver ]; then
      echo "No WCS installed, will install latest build" >> $DEPLOY_LOG
      UPDATE=true
    fi
    echo "Update WCS: $UPDATE" >> $DEPLOY_LOG
    if $UPDATE; then
      cd /tmp
      wget --timeout=10 --no-check-certificate https://flashphoner.com/download-wcs5.2-server.tar.gz -O wcs5-
server.tar.gz
      if [ $? -eq 0 ]; then
        mkdir -p FlashphonerWebCallServer-5.2-latest && tar xzf wcs5-server.tar.gz -C
FlashphonerWebCallServer-5.2-latest --strip-components 1
        cd FlashphonerWebCallServer-5.2-latest
        chmod +x install.sh
        ./install.sh -silent
        cd ..
        rm -rf FlashphonerWebCallServer-5.2-latest wcs5-server.tar.gz

```

```

        echo "WCS updated to build $(cat /usr/local/FlashphonerWebCallServer/conf/WCS.version)" >> $DEPLOY_LOG
    fi
fi
ConfigureWCS:
Description: "WCS configuration helper script. Do not change in wizard!"
Type: String
Default: |
    # Request keyframes from WebRTC publishers every 5 seconds
    echo -e "\n" >> $WCS_CONFIG
    echo -e "periodic_fir_request=true" >> $WCS_CONFIG
    # Disable RTMP keepalives to publish from OBS
    echo -e "keep_alive.enabled=websocket,rtmfp" >> $WCS_CONFIG
    # Configure heap settings
    sed -i -e "s/^(-Xmx\).*\$/\1$HEAP_SIZE/" $JVM_CONFIG
    sed -i -e "s/^(-Xms\).*\$/\1$HEAP_SIZE/" $JVM_CONFIG
ActivateWCS:
Description: "WCS activation helper script. Do not change in wizard!"
Type: String
Default: |
    if [[ ! -z $LICENSE ]]; then
        /usr/local/FlashphonerWebCallServer/bin/activation.sh $LICENSE
    fi
StartWCS:
Description: "WCS startup helper script. Do not change in wizard!"
Type: String
Default: |
    systemctl restart webcallserver
    # Disable internal firewall, ports are allowed/blocked on security group level
    iptables -F
    chown ec2-user $DEPLOY_LOG
OriginCDNSetup:
Description: "WCS Origin intsanse setup helper script. Do not change in wizard!"
Type: String
Default: |
    echo -e "\n" >> $WCS_CONFIG
    echo -e "cdn_enabled=true" >> $WCS_CONFIG
    echo -e "cdn_ip=0.0.0.0" >> $WCS_CONFIG
    echo -e "cdn_point_of_entry=" >> $WCS_CONFIG
    echo -e "cdn_role=origin" >> $WCS_CONFIG
    echo -e "cdn_nodes_resolve_ip=false" >> $WCS_CONFIG
EdgeCDNSetup:
Description: "WCS Edge intsanse setup helper script. Do not change in wizard!"
Type: String
Default: |
    echo -e "\n" >> $WCS_CONFIG
    echo -e "cdn_enabled=true" >> $WCS_CONFIG
    echo -e "cdn_ip=0.0.0.0" >> $WCS_CONFIG
    echo -e "cdn_point_of_entry=$ORIGIN_IP" >> $WCS_CONFIG
    echo -e "cdn_role=edge" >> $WCS_CONFIG
    echo -e "cdn_nodes_resolve_ip=false" >> $WCS_CONFIG
Mappings:
WCSAMI:
eu-north-1:
AMI: ami-0cd89cf8212fd90b4
ap-south-1:
AMI: ami-0861cf9f8d387a5cf
eu-west-3:
AMI: ami-0f5d7f6dcdf0910e0
eu-west-2:
AMI: ami-0d61a966487038aeb
eu-west-1:
AMI: ami-01c249ebee9077dbc
ap-northeast-2:
AMI: ami-023e68299437cbf78
ap-northeast-1:
AMI: ami-0f01e9f19c3733d99
sa-east-1:
AMI: ami-01d3d7a07e6e5beda
ca-central-1:
AMI: ami-0aa76aec8c64e3d52
ap-southeast-1:

```

```

    AMI: ami-044fd54e788e44ddc
ap-southeast-2:
    AMI: ami-0a4f9a18ad123d2ad
eu-central-1:
    AMI: ami-0f785dd5a9571d373
us-east-1:
    AMI: ami-038f9ebb3c87f88ac
us-east-2:
    AMI: ami-0636213ac22f6ef45
us-west-1:
    AMI: ami-0de64b6cac0f8d81c
us-west-2:
    AMI: ami-0c8543b7418393ad5
Conditions:
  GetMarketplaceImage:
    Fn::Equals:
      - Ref: 'ImageId'
      - WCSAMI
Resources:
  WCSOriginInstance:
    Type: AWS::EC2::Instance
    Properties:
      Tags:
        - Key: "Name"
          Value:
            Fn::Join:
              - '-'
              - - !Ref 'InstanceName'
                - "edge"
      ImageId: !If [ GetMarketplaceImage, !FindInMap [ WCSAMI, !Ref 'AWS::Region', AMI ], !Ref 'ImageId' ]
      InstanceType:
        Ref: 'InstanceType'
      SubnetId:
        Ref: 'SubnetId'
      SecurityGroupIds:
        - Ref: 'WCSSecurityGroup'
      KeyName:
        Ref: 'KeyName'
      Monitoring: false
      UserData:
        Fn::Base64:
          Fn::Sub: |
            #!/bin/bash
            # Declare variables
            UPDATE=${UpdateWCS}
            HEAP_SIZE=${JavaHeapSize}
            LICENSE=${WCSLicense}
            # Declare config files to change
            WCS_CONFIG=/usr/local/FlashphonerWebCallServer/conf/flashphoner.properties
            JVM_CONFIG=/usr/local/FlashphonerWebCallServer/conf/wcs-core.properties
            # Declare deployment log
            DEPLOY_LOG=/home/ec2-user/deploy.log
            # Install Java 14 if needed
            ${InstallJava}
            # Stop WCS before reconfiguring
            ${StopPreviousWCS}
            # Update WCS to the latest build
            ${UpdateToLatestWCS}
            # Configuration setup
            ${ConfigureWCS}
            # CDN setup
            ${OriginCDNSetup}
            # Activate WCS license if provided
            ${ActivateWCS}
            # Start WCS after reconfiguring
            ${StartWCS}
  WCSEdgeInstance:
    Type: AWS::EC2::Instance
    DependsOn:
      - WCSOriginInstance
    Properties:

```

```

Tags:
  - Key: "Name"
    Value:
      Fn::Join:
        - '-'
        - !Ref 'InstanceName'
        - "edge"
ImageId: !If [ GetMarketplaceImage, !FindInMap [ WCSAMI, !Ref 'AWS::Region', AMI ], !Ref 'ImageId' ]
InstanceType:
  Ref: 'InstanceType'
SubnetId:
  Ref: 'SubnetId'
SecurityGroupIds:
  - Ref: 'WCSecurityGroup'
KeyName:
  Ref: 'KeyName'
Monitoring: false
UserData:
  Fn::Base64:
    Fn::Sub: |
      #!/bin/bash
      # Declare variables
      UPDATE=${UpdateWCS}
      HEAP_SIZE=${JavaHeapSize}
      LICENSE=${WCSLicense}
      ORIGIN_IP=${WCSOriginInstance.PrivateIp}
      # Declare config files to change
      WCS_CONFIG=/usr/local/FlashphonerWebCallServer/conf/flashphoner.properties
      JVM_CONFIG=/usr/local/FlashphonerWebCallServer/conf/wcs-core.properties
      # Declare deployment log
      DEPLOY_LOG=/home/ec2-user/deploy.log
      # Install Java 14 if needed
      ${InstallJava}
      # Stop WCS before reconfiguring
      ${StopPreviousWCS}
      # Update WCS to the latest build
      ${UpdateToLatestWCS}
      # Configuration setup
      ${ConfigureWCS}
      # CDN setup
      ${EdgeCDNSetup}
      # Activate WCS license if provided
      ${ActivateWCS}
      # Start WCS after reconfiguring
      ${StartWCS}
WCSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    VpcId:
      Ref: 'VpcId'
    GroupDescription: "Enable SSH, websocket, web interface ports and media ports"
  SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp:
        Ref: 'SSHLocation'
    - IpProtocol: tcp
      FromPort: 554
      ToPort: 554
      CidrIp: 0.0.0.0/0
    - IpProtocol: tcp
      FromPort: 1935
      ToPort: 1935
      CidrIp: 0.0.0.0/0
    - IpProtocol: tcp
      FromPort: 3478
      ToPort: 3478
      CidrIp: 0.0.0.0/0
    - IpProtocol: tcp
      FromPort: 8080

```

```

        ToPort: 8084
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8443
        ToPort: 8445
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8888
        ToPort: 8888
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 9091
        ToPort: 9091
        CidrIp: 0.0.0.0/0
      - IpProtocol: udp
        FromPort: 30000
        ToPort: 33000
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 30000
        ToPort: 33000
        CidrIp: 0.0.0.0/0

```

Outputs:

```

OriginWebsiteURL:
  Description: "URL for newly created WCS Origin instance web interface. Use instance id as admin password"
  Value:
    Fn::Join:
      - ''
      - - "https://"
        - !GetAtt WCSOriginInstance.PublicDnsName
        - ":8444/admin/"
OriginInstanceId:
  Value:
    Ref: 'WCSOriginInstance'
  Description: "Instance Id of newly created WCS Origin instance"
OriginPrivateIp:
  Value: !GetAtt WCSOriginInstance.PrivateIp
  Description: "Private IP address of the Origin instance"
OriginPublicIp:
  Value: !GetAtt WCSOriginInstance.PublicIp
  Description: "Public IP address of the Origin instance"
EdgeWebsiteURL:
  Description: "URL for newly created WCS Edge instance web interface. Use instance id as admin password"
  Value:
    Fn::Join:
      - ''
      - - "https://"
        - !GetAtt WCSEdgeInstance.PublicDnsName
        - ":8444/admin/"
EdgeInstanceId:
  Value:
    Ref: 'WCSEdgeInstance'
  Description: "Instance Id of newly created WCS Edge instance"
EdgePrivateIp:
  Value: !GetAtt WCSEdgeInstance.PrivateIp
  Description: "Private IP address of the Edge instance"
EdgePublicIp:
  Value: !GetAtt WCSEdgeInstance.PublicIp
  Description: "Public IP address of the Edge instance"

```

WCS CDN deployment example using CloudFormation web console

1. Sing in to your AWS account,go to desired region and open CloudFormation in Services menu. Click "Create Stack"

AWS CloudFormation

Model and provision all your cloud infrastructure

AWS CloudFormation provides a common language to describe and provision all the infrastructure resources in your environment in a safe, repeatable way.

Create a CloudFormation stack

Use your own template or a sample template to quickly get started.

[Create stack](#)

2. Choose "Upload a template file", click "Choose file" and upload the example template

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☒ Template is ready

☐ Use a sample template

☐ Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL

☒ Upload a template file

Upload a template file

Choose file

No file chosen

JSON or YAML formatted file

S3 URL: Will be generated when template file is uploaded

View in Designer

Cancel

Next

3. When template is uploaded, click Next

Upload a template file

Choose file

wcs-ec2-template-origin-edge.yml

JSON or YAML formatted file

S3 URL: https://s3.eu-north-1.amazonaws.com/cf-templates-1c0pwbvffxqz0-eu-north-1/20211695zL-wcs-ec2-template-origi
n-edge.yml

View in Designer

Cancel

Next

4. Enter stack name

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Specify stack details

Stack name

Stack name

wcs-test-stack

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

5. Enter Amazon Linux 2 AMI ID for region chosen, or leave WCSAMI (in this case, AWS Marketplace WCS AMI will be used with hourly billing)

ImageId

Basic instance ami (WebCallServer 5.2.944 AMI by default, mapped by region)

ami-08b4d798afd78c423

6. Enter basic part of instance name (-origin and -edge will be added respectively), choose instance type, enter Java heap size and choose SSH key to access stack instances

InstanceName

Name of EC2 instance

wcs-test

InstanceType

Basic EC2 instance type

m5.xlarge

JavaHeapSize

Maximum Java heap size in megabytes (2048m) or gigabytes (2g), 1024m by default

1024m

KeyName

Name of an existing EC2 KeyPair to enable SSH access to the instance

test1

7. Set subnet Id

SubnetId

SubnetId of an existing subnet in your Virtual Private Cloud (VPC)

subnet-d2cb6fbb

8. Enter "true" to automatically update WCS to the latest build

UpdateWCS

Update WCS to the latest build

true

9. Set VPC Id

VpcId

VpcId of your existing Virtual Private Cloud (VPC)

vpc-5e65c237

10. If Marketplace WCS AMI is not used, enter the license key to activate on instances and click Next

WCSLicense

WCS License key to activate (optional, if you do not use Marketplace AMI)

Cancel

Previous

Next

11. Add tags and set permissions if necessary

Step 1

Specify template

Step 2

Specify stack details

Step 3

Configure stack options

Step 4

Review

Configure stack options

Tags

You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack. [Learn more](#)

Key

Value

Remove

Add tag

Permissions

Choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials. [Learn more](#)

IAM role - optional

Choose the IAM role for CloudFormation to use for all operations performed on the stack.

IAM role name

Sample-role-name

Remove

12. Set advanced stack options if necessary, then click Next

Advanced options

You can set additional options for your stack, like notification options and a stack policy. [Learn more](#)

► Stack policy

Defines the resources that you want to protect from unintentional updates during a stack update.

► Rollback configuration

Specify alarms for CloudFormation to monitor when creating and updating the stack. If the operation breaches an alarm threshold, CloudFormation rolls it back. [Learn more](#)

► Notification options

► Stack creation options

Cancel

Previous

Next

13. Review stack parameters

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Review wcs-test-stack

Step 1: Specify template

Edit

Template

Template URL
https://s3.eu-north-1.amazonaws.com/cf-templates-1c0pwbvffxqz0-eu-north-1/20211695zL-wcs-ec2-template-origin-edge.yml

Stack description
Create WCS CDN stack

Estimate cost [↗](#)

Step 2: Specify stack details

Edit

Parameters (18)

14. Click "Create stack"

Stack creation options

Rollback on failure
Enabled

Timeout
-

Termination protection
Disabled

[▶ Quick-create link](#)

Cancel

Previous

Create change set

Create stack

15. Wait for stack creation completion

CloudFormation > Stacks > wcs-test-stack

wcs-test-stack [Delete] [Update] [Stack actions ▼] [Create stack ▼]

Stack info | **Events** | Resources | Outputs | Parameters | Template | Change sets

Events (11) [Refresh] [Filter]

Timestamp	Logical ID	Status	Status reason
2021-06-18 14:39:15 UTC+0700	wcs-test-stack	CREATE_COMPLETE	-
2021-06-18 14:39:13 UTC+0700	WCSEdgeInstance	CREATE_COMPLETE	-
2021-06-18 14:39:06 UTC+0700	WCSEdgeInstance	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-18 14:39:04 UTC+0700	WCSEdgeInstance	CREATE_IN_PROGRESS	-
2021-06-18 14:39:01 UTC+0700	WCSOriginInstance	CREATE_COMPLETE	-
2021-06-18 14:38:54 UTC+0700	WCSOriginInstance	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-18 14:38:52 UTC+0700	WCSOriginInstance	CREATE_IN_PROGRESS	-
2021-06-18 14:38:50 UTC+0700	WCSSecurityGroup	CREATE_COMPLETE	-
2021-06-18 14:38:47 UTC+0700	WCSSecurityGroup	CREATE_IN_PROGRESS	Resource creation Initiated
2021-06-18 14:38:42 UTC+0700	WCSSecurityGroup	CREATE_IN_PROGRESS	-
2021-06-18 14:38:37 UTC+0700	wcs-test-stack	CREATE_IN_PROGRESS	User Initiated

16. Go to Outputs tab

wcs-test-stack [Delete] [Update] [Stack actions ▼] [Create stack ▼]

Stack info | Events | Resources | **Outputs** | Parameters | Template | Change sets

Outputs (8) [Refresh] [Filter]

Key	Value	Description	Export name
EdgeInstanceid	i-0a15ee908afc525a4	Instance Id of newly created WCS Edge instance	-
EdgePrivateIp	172.31.26.94	Private IP address of the Edge instance	-
EdgePublicIp	13.51.156.51	Public IP address of the Edge instance	-
EdgeWebsiteURL	https://ec2-13-51-156-51.eu-north-1.compute.amazonaws.com:8444/admin/	URL for newly created WCS Edge instance web interface. Use instance id as admin password	-
OriginInstanceid	i-06bb1e29aa40acae	Instance Id of newly created WCS Origin instance	-
OriginPrivateIp	172.31.18.18	Private IP address of the Origin instance	-
OriginPublicIp	13.51.156.202	Public IP address of the Origin instance	-
OriginWebsiteURL	https://ec2-13-51-156-202.eu-north-1.compute.amazonaws.com:8444/admin/	URL for newly created WCS Origin instance web interface. Use instance id as admin password	-

17. Open Origin and Edge web interfaces, publish the stream test to Origin using Two Way Streaming example, then play the stream on Edge

