

Pixel Streaming on the AWS Cloud

with Unreal Engine 4

Deployment Guide

August 2020

Last update: March 2021 ([revisions](#))

Kevin Ashman Solutions Architect, Amazon Web Services

Table of Contents

Overview	2
Working with UE4 Pixel Streaming on AWS	2
Costs and Licenses.....	2
Architecture	3
Planning the deployment.....	3
Specialized Knowledge	3
AWS account	4
Technical requirements.....	4
Deployment Steps	4
Step 1. Prepare Your AWS Account.....	4
Step 2. Create UE4 Pixel Streaming Build.....	5
Step 3. Upload Required Files to S3	5
Step 4. Launch the AWS Sample	7
Step 5. Test Pixel Streaming Instance.	9
Step 5a. Optionally Use NICE DCV for Remote Desktop Access	10
Security.....	10
Troubleshooting.....	11
Additional Resources.....	12
Document Revisions	12

Overview

This AWS Sample deployment guide provides step-by-step instructions for deploying a Windows-based Unreal Engine 4 Pixel Streaming build on the AWS Cloud.

This sample is for those who use Unreal Engine 4 to build content and wish to deploy this content to an audience via UE4 pixel streams. Content examples include but are not limited to: interactive entertainment, architectural visualization, high fidelity car configurators, or anyone who needs to let users access high quality interactive content via thin clients such as web browsers.

Working with UE4 Pixel Streaming on AWS

Using AWS and [Unreal Engine 4's Pixel Streaming solution](#), developers can create content with Unreal Engine and deploy on AWS so users can engage with the content from any modern Web browser. A build of the UE4 content is run on an [Amazon Elastic Compute Cloud \(Amazon EC2\) G4 instance](#). G4 instances are GPU instances that are designed for graphics-intensive workloads and offer a powerful, low-cost, pay-as-you-go model which is ideal for on-demand interactive content. This solution uses G4dn instances powered by NVIDIA T4 GPUs, and use [NVIDIA G4dn gaming drivers](#) specific for these instances.

This sample deployment sets up an EC2 environment on AWS that includes the following:

- [Unreal Engine 4 Pixel Streaming](#) components from your pixel streaming build are installed and run on startup. Components include UE4 Windows build with pixel streaming plug-in executable, Stun Server, Turn Server, and Signaling Web Server.
- [NICE DCV](#) is installed allowing developers to connect to the G4 instance, for a low latency remote desktop experience that supports resolutions up to 4k. This allows a developer to remote in to test the build and optionally install Unreal Engine 4 directly on the machine.

Costs and Licenses

You are responsible for the cost of the AWS services used while running this reference deployment.

The AWS CloudFormation template for this sample includes configuration parameters that you can customize. Some of these settings, such as instance type, will affect the cost of deployment. For cost estimates, see the pricing pages for each AWS service you will be using. Prices are subject to change.

Architecture

Deploying this sample for a new virtual private cloud (VPC) with **default parameters** builds the following EC2 workstation environment in the AWS Cloud.

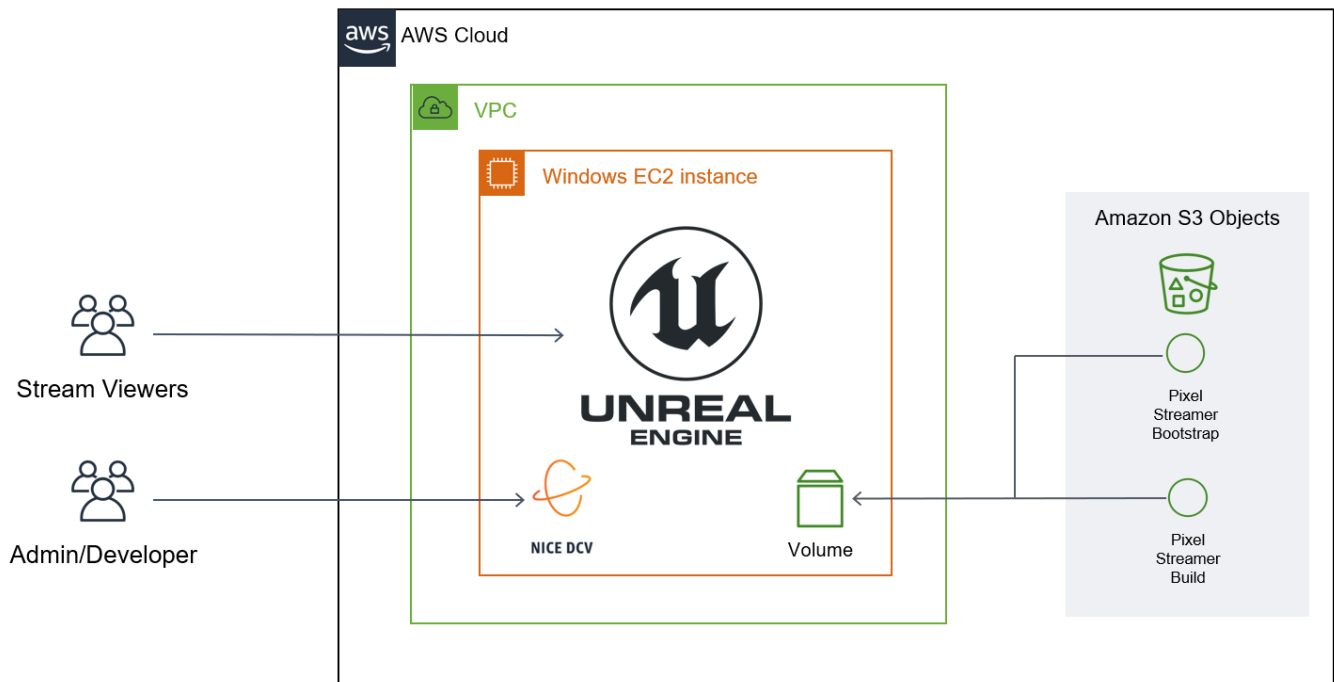


Figure 1: AWS Sample architecture for UE4 Pixel Streamer on AWS

The sample sets up the following:

- A Security Group which opens required ports for UE4 Pixel Streaming components and NICE DCV.

Planning the deployment

Specialized Knowledge

Before you deploy this sample, we recommend that you become familiar with the following AWS services and components. (If you are new to AWS, see [Getting Started with AWS](#).)

- [Amazon EC2](#)
- [Amazon EC2 G4 instance](#)
- [AWS Security Groups](#)

We also recommend that you familiarize yourself with [Getting Started with Pixel Streaming](#) from Unreal Engine.

AWS account

If you don't already have an AWS account, create one at <https://aws.amazon.com> by following the on-screen instructions. Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.

Your AWS account is automatically signed up for all AWS services. You are charged only for the services you use.

Technical requirements

Before you launch the sample, your account must be configured as specified in the following table. Otherwise, deployment might fail.

Key pair	Ensure that at least one Amazon EC2 key pair exists in your AWS account in the Region where you plan to deploy the sample. Make note of the key pair name. You need it during deployment. To create a key pair, follow the instructions in the AWS documentation . For testing or proof-of-concept purposes, we recommend creating a new key pair instead of using one that's already being used by a production instance.
IAM permissions	Before launching the sample, you must log in to the AWS Management Console with IAM permissions for the resources and actions the templates deploy. The <i>AdministratorAccess</i> managed policy within IAM provides sufficient permissions, although your organization may choose to use a custom policy with more restrictions.

Deployment Steps

Step 1. Prepare Your AWS Account

1. Sign in to your AWS account at <https://aws.amazon.com> with an IAM user role that has the necessary permissions.
2. Use the region selector in the navigation bar to choose the AWS Region where you want to deploy an EC2 workstation environment on AWS.

Important This sample uses G4 instances, which aren't available in all AWS Regions. For more information, see the [Amazon EC2 Pricing](#) webpage, choose **Windows**, and check to make sure that the AWS Region you want to use for the deployment supports G4 instances.

3. Create a [key pair](#) in your preferred region if you have not done so already.

4. If necessary, [request a service limit increase](#) for the Amazon EC2 G4 instance type. You might need to do this if you already have an existing deployment that uses this instance type, and you think you might exceed the [default limit](#) with this reference deployment.

Step 2. Create UE4 Pixel Streaming Build

This guide only covers steps to use Unreal Engine 4 to export a Pixel Streaming build to deploy onto AWS, and assumes content has already been built.

1. Review [Pixel Streaming Overview](#) documentation on Unreal Engine website, which describes the technology used, the connection process, and how to [get started quickly with your first Pixel Streaming project](#).
2. Follow the Getting started with Pixel Streaming documentation to enable Pixel Streaming in your project, build your WindowsNoEditor build, and test your Pixel Streaming server locally.
3. If you are running Unreal Engine 4.26.0 or later you can skip this step. For earlier versions you will need to modify a Windows PowerShell script for Windows Server compatibility:
 - a. Open file:
`Engine\Source\Programs\PixelStreaming\WebServers\SignallingWebServer\Start_AWS_WithTURN_SignallingServer.ps1`
 - b. Look for “\$PublicIp = Invoke-WebRequest” line and add an additional parameter “-UseBasicParsing”. An example of this modified line is:
`$PublicIp = Invoke-WebRequest -Uri "http://169.254.169.254/latest/meta-data/public-ipv4" -UseBasicParsing`
4. Compress the WindowsNoEditor folder into a zip file by right clicking on the folder and creating the zip file. Note the zip should extract the contents into a WindowsNoEditor folder.

Step 3. Upload Required Files to S3

There are three files needed for this solution which need to be accessible via a http URL from the account you will be running the Pixel Streamer in. You can get the first two from the AWS Samples [AWS Sample's GitHub repository](#), the third is the zip file created in Step 2 above.

- UE4-Pixel-Streamer.json – This is the CloudFormation JSON template used to create the stack. This file must be uploaded to S3.

- UE4-Pixel-Streamer-Bootstrap.ps1 – This is the PowerShell script executed once the server has been launched to setup the Pixel Streamer.

Important You will need to modify a parameter in this bootstrap file to match your UE4 Pixel Streaming build. Look for '\$buildExecutable' near the top of the file and change the file name to match the name of the executable file in the root directory of your build. This is typically the name of the project with '.exe' extension.

- Zip File – This is the zip file created in step 2, and can be named to identify the build you are using.

Follow the steps below to upload these files:

1. Clone the GitHub repository or download the [JSON template file](#) and the [bootstrap file](#). Make sure you modify the bootstrap file as noted above. You will then upload those two files along with the build zip file following these steps:

- a. From the AWS Services choose **Storage**, then **S3** to open Amazon S3.
- b. From the Amazon S3 console dashboard, choose **Create Bucket**.
- c. In Create a Bucket, type a bucket name in Bucket Name. Choose a Region then click **Create**. The defaults will be acceptable for development and testing, and will limit the access to the bucket to your account. The EC2 instance will inherit access to the bucket. When preparing for production, review all S3 properties such as encryption and tagging.

Important The bucket name you choose must be globally unique across all existing bucket names in Amazon S3 (that is, across all AWS customers). For more information, see [Bucket Restrictions and Limitations](#).

- d. Once you've created the bucket select it from the list to access that bucket. Then select the **Upload Button**.
 - e. Click on the **Add Files** button and select the files that you want to upload to the S3 Bucket. This would include the pixel streaming build .zip file, bootstrap file, and CloudFormation JSON file.
 - f. Once the files have been selected, click **Upload** button.
2. Save the path to your files for future reference. You can do this by clicking on each file in the S3 bucket and copying the Object URL link. You can also copy the URL directly to your clipboard by clicking the icon to the left of the path. Your URL

should look like the following:

`https://s3-bucket-name.s3-us-west-2.amazonaws.com/Folder/WindowsNoEditor.zip`

Step 4. Launch the AWS Sample

Note You are responsible for the cost of the AWS services used while running this sample reference deployment. For full details, see the pricing pages for each AWS service you will be using for this deployment.

1. To navigate to AWS CloudFormation, click on the Services dropdown found at the top of the AWS Web Console. From the dropdown menu, choose ‘CloudFormation’ found under the ‘Management & Governance’ section.
2. Check the region that’s displayed in the upper-right corner of the navigation bar and change it if necessary. This is where the network infrastructure for the UE4 Pixel Streaming environment will be built.

Important This sample uses G4 instances, which aren’t available in all AWS Regions. For more information, see the [Amazon EC2 Pricing](#) webpage, choose **Windows**, and check to make sure that the AWS Region you want to use for the deployment supports G4 instances.

3. Click the “Create stack” button either in the top right or middle of your screen. You will choose the option to create the stack “With new resources (standard)”. The AWS CloudFormation template deploys the resources into your AWS account. A deployment will take about 10 minutes to complete.
4. On the **Create Stack** page, paste in the URL for your JSON template file, and choose **Next**.
5. On the **Specify stack details** page, give the stack a name. You will need to assign a Key Pair and location for the build zip file. The bootstrap file is provided to you via GitHub, however you will need to put the path to the build zip file in the build parameter. For all other parameters, review the default settings and customize them as necessary. When you finish reviewing and customizing the parameters, choose **Next**.

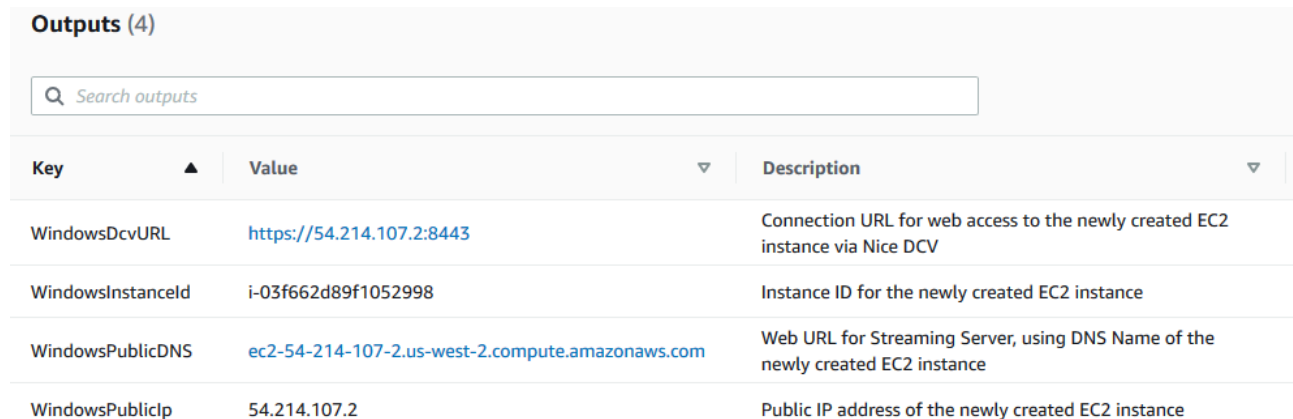
In the following table, parameters are listed by category.

CloudFormation Template Parameters:

Parameter label (name)	Default	Description
Host Instance Type (InstanceType)	g4dn.4xlarge	Amazon EC2 instance type for the pixel streaming server. Size should be smallest instance size that achieves required performance.
Host Operating System (OsVersion)	<i>Windows Server 2019</i>	Specify the version of Windows (Windows Server 2019) OS to use. Valid values are “WindowsServer2019”, “WindowsServer2016”, or “WindowsServer2012R2”.
EBS Volume size for EC2 instance (DiskSize)	50	Volume size for the host, in GB.
Key Pair Name (KeyPairName)	<i>Requires input</i>	Name of AWS EC2 Key Pair. This is not used when logging into machine, but needed to secure instance on VPC.
Credentials (UserPasswd)	<i>Ch4ng3M3!</i>	Windows Administrator password used for logging into EC2 via NICE DCV or other administration. It is recommended that you change this default password.
Pixel Streamer Bootstrap Location (PixelStreamerBootstrapLocation)	<i>Requires input</i>	Specify the location of bootstrap file in S3 which is executed upon initial launch of EC2 instance which configures firewall rules and adds startup tasks needed to run UE4 Pixel Streamer. AWS developed bootstrap can be used, or customized as needed.
UE4 Pixel Streamer Build (PixelStreamerBuildLocation)	<i>Requires input</i>	Specify the location of UE4 Pixel Streamer build zip file in S3. Build file was created in Step 2 of this guide following UE4 Getting Started with Pixel Streaming documentation .
Pixel Streamer Access CIDR (PixelStreamingAccessCIDR)	0.0.0.0/0	IP address range, as an access CIDR , of pixel stream viewers. The default allows access from all IPs, however if your viewers are coming from a specific range limit the access. For development and testing, you can specify your own IP range such as 123.456.78.9/32.
NICE DCV Access CIDR (NiceDCVAccessCIDR)	0.0.0.0/0	IP address range, as an access CIDR , of admins and developers to access server via NICE DCV . The default allows access from all IPs, but should be limited to IP addresses of admins. For development and testing, you can specify your own IP range such as 123.456.78.9/32.
Windows AMIs (2019, 2016, 2012)	AWS Defined AMI Paths	Default values ensure that the latest Windows AMI published by AWS is used. You can also define a different AMI to use with a different path.

- On the options page, you can [specify tags](#) (key-value pairs) for resources in your stack and [set advanced options](#). An example of a tag would be a Key of ‘Project’, and Value of ‘UE4 Pixel Streamer’. You do not need to do anything on this page, and when you’re done choose **Next**.

7. On the **Review** page, review and confirm the template settings. Under **Capabilities**, select the check box to acknowledge that the template creates IAM resources.
8. Choose **Create stack** to deploy the stack. Typically, a stack will take around 10 minutes to deploy. Go to the Events tab to see the progress, including while it is initiating the EC2 instance. If the 'WindowsInstanceWaitCondition' lasts longer than 10 to 15 minutes, proceed to the [Troubleshooting](#) section of this document.
9. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the UE4 Pixel Streaming stack is ready.
10. Use the URL or IP address displayed in the **Outputs** tab for the stack, as shown in Figure 2, to view the resources that were created. You can also manage the EC2 directly from the Instance Id.



Outputs (4)

Search outputs

Key ▲	Value ▼	Description ▼
WindowsDcvURL	https://54.214.107.2:8443	Connection URL for web access to the newly created EC2 instance via Nice DCV
WindowsInstanceid	i-03f662d89f1052998	Instance ID for the newly created EC2 instance
WindowsPublicDNS	ec2-54-214-107-2.us-west-2.compute.amazonaws.com	Web URL for Streaming Server, using DNS Name of the newly created EC2 instance
WindowsPublicIp	54.214.107.2	Public IP address of the newly created EC2 instance

Figure 2: CloudFormation Sample Outputs Tab

Step 5. Test Pixel Streaming Instance.

In the Outputs tab click on WindowsPublicDNS value in a new tab to create a session on your UE4 Pixel Streaming server. You can use either the WindowsPublicIp or URL from the outputs tab to create sessions with the Pixel Streaming server.

Once you have verified your Pixel Streaming server, you can use the EC2 web console to stop and start the instance. The server is configured to restart the pixel server whenever the instance is started again. You will want to stop the instance when you are not using it, so that you do not incur EC2 costs while it is left running.

If you want to refresh the content on the server, you can delete the CloudFormation stack and create a new stack pointing at your new build in S3. Alternatively, you can remote into the instance and replace or modify the build files located in C:\PixelStreamer.

Step 5a. Optionally Use NICE DCV for Remote Desktop Access

In order to connect to your instance, you can either connect via your browser or download and install the NICE DCV client software to your local computer.

1. Connect to the Pixel Streaming Server with NICE DCV via Browser or Client.
 - To connect to your session via a browser, you can simply open up a new tab with the URL in the Outputs tab for WindowsDcvURL. Example:
`https://54.214.107.2:8443`
 - When using the NICE DCV client, first download the native client application from the [DCV download page](#). Install and launch the client. You then enter the IP address of the server and click Connect. The IP address is the WindowsPublicIp value in the Outputs tab.
2. DCV is set up with a self-signed certificate. You have to trust this certificate on your preferred browser in order to proceed to the login form.
3. To access the instance, you use the username "Administrator" with the password you provided as a parameter on the CloudFormation template.

Security

There are several security-related aspects of the architecture in this sample. The most important is understanding what is publicly accessible, and with the defaults both the UE4 Pixel Streaming content and the ability to remote into the desktop is fully exposed. These are controlled with a security group this solution creates. You can specify a CIDR range, a numerical expression for a range of IP addresses, that the security group allows through to the EC2 instance. Please see our documentation on [security groups for your VPC](#) for more details. You can limit access to the host by changing the access CIDR properties from their defaults.

First and most importantly, the NiceDCVAccessCIDR will limit IP addresses that can access the remote desktop via NICE DCV. It is recommended that you limit NICE DCV access as much as possible. For development and testing, change this to your specific IP address as suggested in the property description. The CIDR would look similar to `'123.456.78.9/32'`.

To limit access to the UE4 Pixel Streaming content, you would limit the IP addresses via the PixelStreamingAccessCIDR. For development and testing, you can also limit this to your specific IP address. For production you would open the pixel streaming CIDR to IPs

for your viewers. For public access you would use ‘0.0.0.0/0’, however dynamically approving specific IP addresses in production would be ideal.

Finally, do note that the solution is deployed into the default VPC for quick testing purposes. For production, you would likely deploy to a specific VPC. This solution does create a security group, that allows additional fine-grained control of traffic in and out of the EC2 hosting the Pixel Streaming server. Feel free to directly modify that security group as required.

Troubleshooting

Q. When deploying your first stack, mistakes in properties or in scripts may cause the deployment to fail. If you are looking at your CloudFormation stack and it has been stuck on ‘WindowsInstanceWaitCondition’ with a status of ‘CREATE_IN_PROGRESS’ for more than 10 minutes then it has likely failed. What went wrong?

A. This typically means that the Windows instance failed to reboot after the bootstrap was executed. If the stack is still running after 10 minutes attempt to view the UE4 Pixel Stream by using the server IP address in a web browser. If it succeeds, go into the EC2 console and manually reboot the instance. If you cannot view the UE4 Pixel Stream, use NICE DCV to remote into the machine and investigate what happened by looking at these log files:

- C:\ProgramData\Amazon\EC2-Windows\Launch\Log\UserdataExecution.log
- C:\PixelStreamer\Downloads\UE4-Pixel-Streamer-Bootstrap.log

By reviewing these logs, you may find scenarios such as not being able to find a file in S3, running out of disk space, or other configuration issues. You might identify a problem that can be fixed with a change to your project, parameters, or bootstrap file. If so, delete the stack which will delete your instance, make the change, and launch your stack again. You can also work directly with the instance and try to launch your pixel streaming build manually as you may have done to initially test on your local machine.

Q. When I go to connect to the streaming server via a browser, I click on the play button but no content loads. What might be happening?

A. This can be caused by a local VPN blocking the direct connection to the streaming server from your computer. Disable VPN or firewalls and check or connect from a different machine. Another potential cause is not having modified the bootstrap file, where you need to change the value of ‘\$buildExecutable to match your build’s name.

Q. I encountered a CREATE_FAILED error when I launched the CloudFormation stack.

A. If AWS CloudFormation fails to create the stack, we recommend that you relaunch the template with **Rollback on failure** set to **No**. (This setting is under **Advanced** in the

AWS CloudFormation console, **Options** page.) With this setting, the stack’s state will be retained and the instance will be left running, so you can troubleshoot the issue. You may be able to connect to the instance with NICE DCV, and if not then Microsoft Remote Desktop. (Look at the log files in %ProgramFiles%\Amazon\EC2ConfigService and C:\cfn\log.)

Important When you set **Rollback on failure** to **No**, you’ll continue to incur AWS charges for this stack. Please make sure to delete the stack when you’ve finished troubleshooting.

For additional information, see [Troubleshooting AWS CloudFormation](#) on the AWS website.

Additional Resources

AWS services

- Amazon EC2
<https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/>
- AWS CloudFormation
<https://aws.amazon.com/documentation/cloudformation/>

Unreal Engine documentation

- UE4 Pixel Streaming Documentation
<https://docs.unrealengine.com/en-US/Platforms/PixelStreaming/index.html>

Document Revisions

Date	Change	In sections
March 2021	Added new parameters and updated default values, added troubleshooting details, and integrated improvements from customers.	Deployment Steps, Security, Troubleshooting
December 2020	Improved security documentation and minor fixes.	Deployment Steps
November 2020	Improved bootstrap and S3 documentation	Deployment Steps
August 2020	Initial publication	—

© 2020, Amazon Web Services, Inc. or its affiliates, and <partner organization>. All rights reserved.

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The software included with this paper is licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at <http://aws.amazon.com/apache2.0/> or in the "license" file accompanying this file. This code is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.