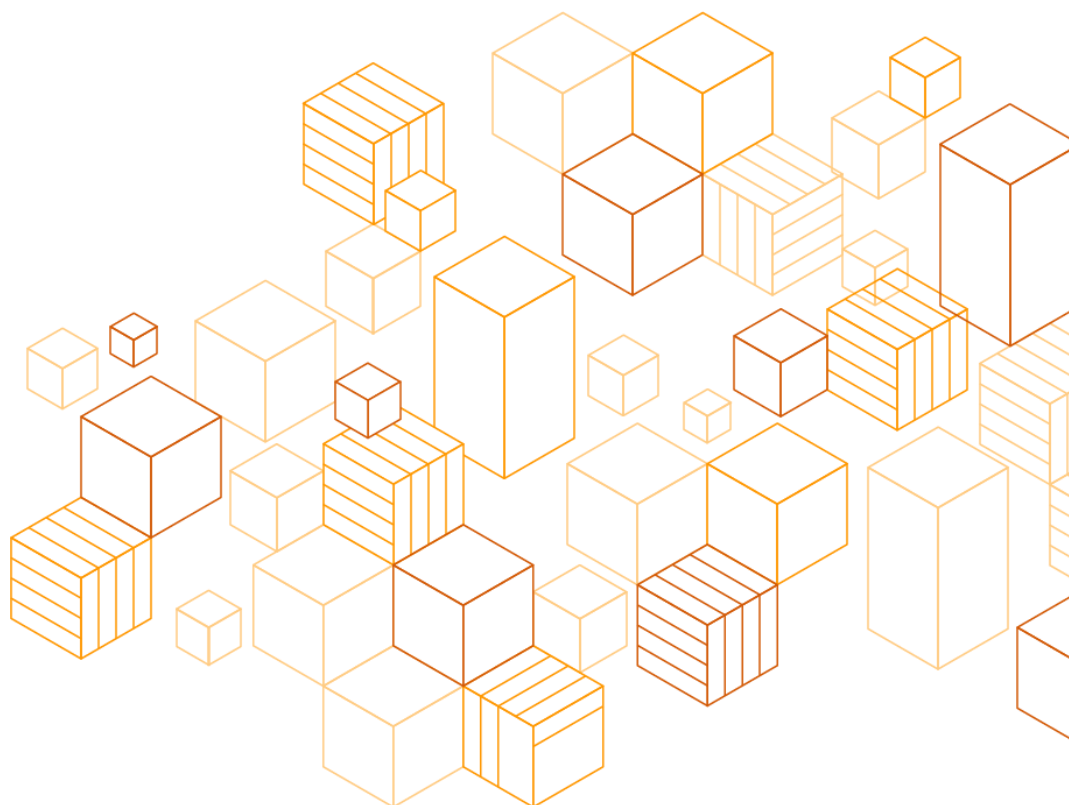


# Studio in the Cloud

## Add-on Tutorials

**Published May 14, 2020**

*Updated July 28, 2021*



# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. 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.

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Contents

Tutorial 8: Add-On - Upgrading to Teradici for Desktop Streaming .....	1
Tutorial 9: Add-On - Using a Linux Instance for a Render Scheduler .....	28
Tutorial 10: Add-On - Using Linux and Teradici for a Workstation .....	80

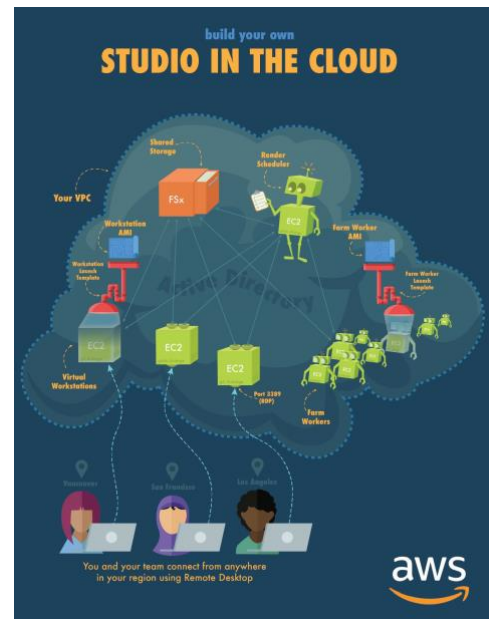
# Tutorial 8: Add-On - Upgrading to Teradici for Desktop Streaming

*Estimated Time to Complete: 40 minutes*

## Review of Studio in the Cloud

If you've reached this tutorial, you most likely have already completed our previous Studio in the Cloud tutorials in which we provided step-by-step instructions for setting up virtual workstations, cloud storage and cloud rendering. By the end of the 7 part series, you have a fully cloud-based studio that leverages the scale, power, and convenience of AWS.

Those tutorials are meant to provide you with the basic infrastructure for a cloud studio with the most straight forward implementation possible. However, it is by no means a stopping point. There are many additional ways to build onto and customize your studio.



## Overview of this Tutorial

The purpose of this tutorial is to show you how to add the power of [Teradici](#) to your Studio in the Cloud. [Teradici Cloud Access Software](#) is a powerful solution for connecting to and streaming the desktops of your virtual workstations that is an alternative to Remote Desktop.

We'll walk you through the steps of adding Teradici as a desktop streaming solution to your existing Studio in the Cloud setup

# Prerequisites

## Complete the Previous Tutorials

This add-on tutorial assumes that you've already completed at least Tutorials 1-5 in our previous Studio in the Cloud tutorial series. If you are new to Studio in the Cloud, you can find the first tutorial here: [Studio in the Cloud Implementation Guide](#).

If you have already completed the Studio in the Cloud tutorials, then you're all set to continue with the steps below.

## Obtain a Teradici License

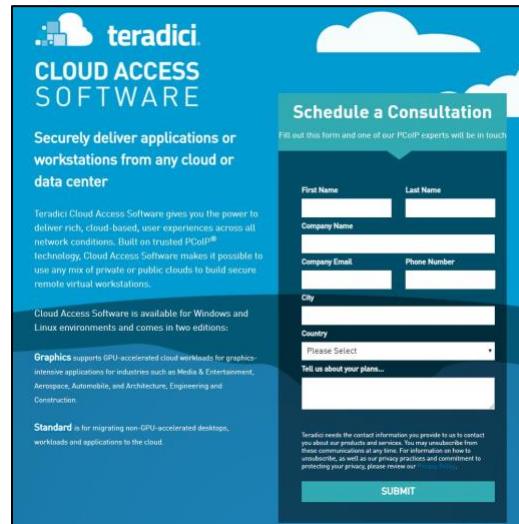
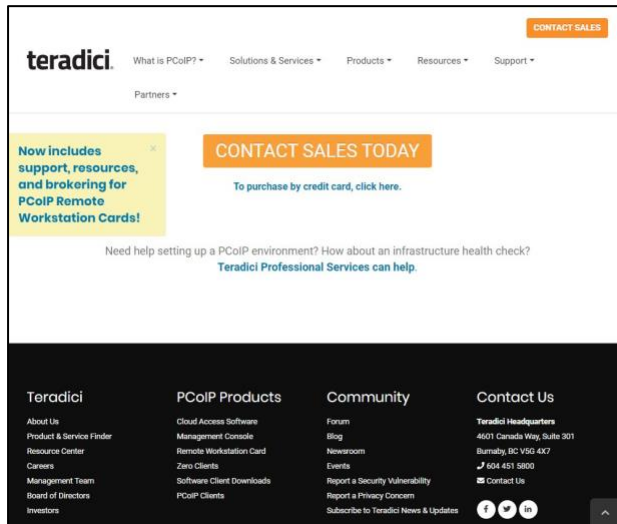
If you are already a Teradici customer and have a Cloud Access Plus license or registration code, all you need to do is locate it so that you can enter it during setup later.

If you're not a Teradici customer, you have a few options:

- Use the [Teradici Windows AMI from the AWS Marketplace](#)
  - This AMI already has Teradici installed. Instead of requiring a license, the cost is built into the hourly pricing for the AMI. You pay an extra \$0.50/hour on top of the normal hourly price for your instance. However, starting with the Teradici AMI would also require you to re-install all the software from the previous Studio in the Cloud tutorials. For that reason, we won't be covering this option in this tutorial. Instead we'll walk you through installing Teradici yourself, which requires you to obtain a license using one of the methods below.
- Purchase a Teradici Subscription
  - If you are ready to purchase Teradici, you can follow this link to their [Cloud Access Plus page](#). At the bottom of the page are links to either contact Teradici sales or purchase with a credit card. If purchasing, make sure to choose the Cloud License Server option. Once your purchase is complete and you receive your registration code, keep it handy for later.
- Request a trial license

- You can request a trial license by contacting Teradici sales using this form: <https://connect.teradici.com/cas-demo>. When filling out the form, make sure to mention that you are interested in the Graphics edition, which is required for the GPU-enabled instances you use for your workstations. After you have been contacted by Teradici sales and arrange to get a trial license, you will receive a registration code via email.

Once you have your Teradici registration code, you can move onto the steps below.



## Startup Notes

We're going to add Teradici Cloud Access Software to your existing Windows workstation. As a starting point, we'll need to launch a fresh, new instance using your Windows Workstation launch template. Note: you may have named this "My-Studio-Workstation-LT" if you followed the naming convention in our previous tutorials.

## Launch a New Instance from Your Windows Workstation Launch Template

- In the **EC2 Dashboard** navigate to your **Launch Templates**
- Select your Windows Workstation Launch Template (e.g., My-Studio-Workstation-LT), click the **Actions** menu and select **Launch instance from template**
- Select the most recent version in **Source template version**
- Leave number of instances at **1**

- In **Instance Tags**, change the **Name** value to **Workstation\_Win\_Teradici** and click the check boxes to tag the volumes as well.
- You can leave the rest of the launch template at its defaults and click **Launch instance from template**

## Create a Teradici Security Group

While your fresh Windows workstation is starting up, you can create a new security group to allow the type of traffic needed by Teradici.

- Go to **Services**→ **EC2**
- Select **Security Groups** in the left side panel under Network & Security
- Click **Create Security Group**
- Choose a name for your security group (e.g., My-Studio-Teradici-SG)
- Set the Description to **Allows for Teradici Connection**
- Set the **VPC** to the VPC you created in the earlier Studio in the Cloud tutorials (e.g. My-Studio-VPC). *You can also find this information on the [Important Information Cheat Sheet](#) that you filled out while completing the previous Studio in the Cloud tutorials.*
- Make sure the **Inbound** tab is selected and then add the following rules:

**Create Security Group**

Security group name: My-Studio-Teradici-SG

Description: Allows for Teradici Connection

VPC: vpc-0b1747ddbc7d0793d | My-Studio-VPC


Security group rules:

**Inbound** | Outbound

Type	Protocol	Port Range	Source	Description	
HTTPS	TCP	443	Anywhere	0.0.0.0/0, ::/0	Teradici - HTTPS
Custom TCP F	TCP	4172	Anywhere	0.0.0.0/0, ::/0	Teradici - PCoIP
Custom UDP I	UDP	4172	Anywhere	0.0.0.0/0, ::/0	Teradici - PCoIP
Custom TCP F	TCP	60443	Anywhere	0.0.0.0/0, ::/0	Teradici - PCoIP

Add Rule

Cancel Create

 **Note:** As in earlier tutorials, we are initially opening up this security group to inbound traffic from any IP address. However, if you limited your source IP addresses during your Studio in the Cloud setup, you'll want to do the same here.

If you will be accessing your instances over the public Internet, then after initial testing we recommend that you look into an [AWS Direct Connect](#) connection and/or VPN so that your connection will be private and secure. You may use third party VPN software, but AWS also provides a robust set of VPN solutions called [AWS VPN](#). More information about how to connect to your VPC using VPN can be found [here](#).

- Click **Create**.

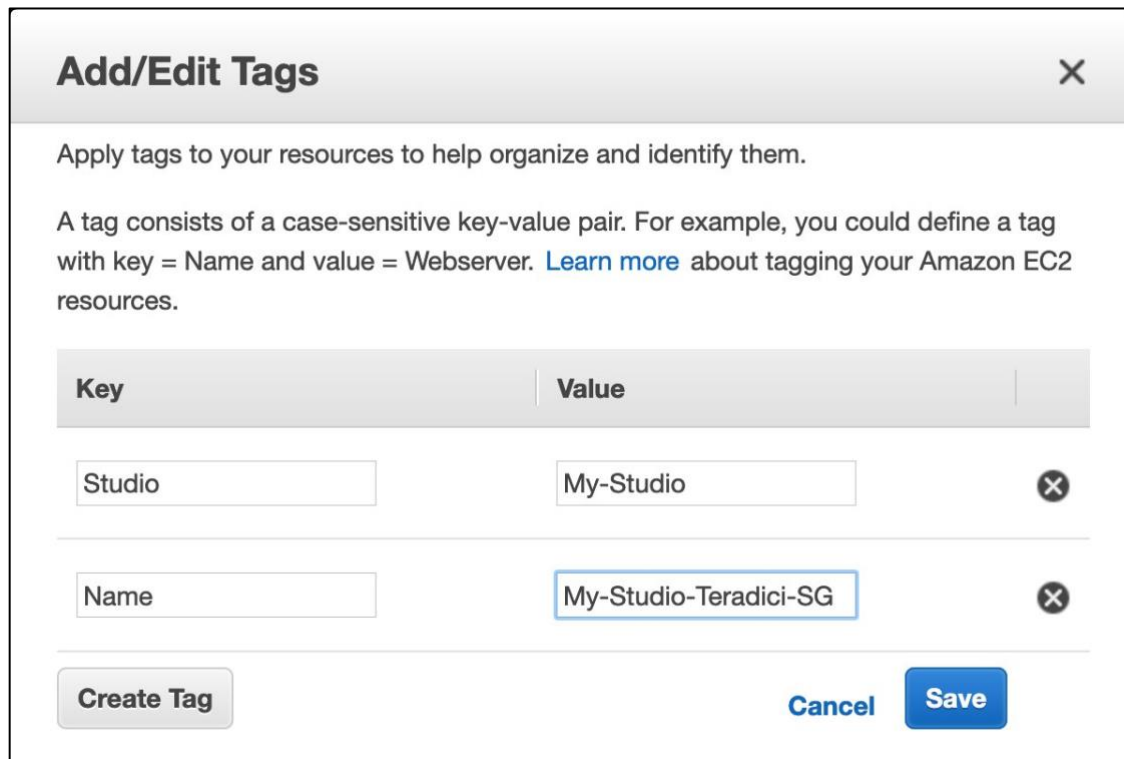
## Add a Tag to Your Security Group

Just as we did for the other security groups in the tutorials, we should add “Studio” and “Name” tags to this new Teradici security group as well.

- Select your **Teradici Security Group** from the list. Because we haven't set a name tag yet, look under the **Security group name column** to identify the correct one.
- On the **Tags** tab in the bottom panel, click **Add/Edit Tags**
- In the **Add/Edit Tags** popup window, click **Create Tag**
- For **Key** enter **Studio**
- For **Value** enter the name you picked for your studio (e.g., My-Studio) *You can find your studio name on the cheat sheet.*
- Next, you'll add a **Name** for your security group. Click **Create Tag** again.
- For **Key** enter **Name**



- For **Value** enter a name for the security group (e.g., My-Studio-Teradici-SG)



**Add/Edit Tags** [X]

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value	
Studio	My-Studio	[X]
Name	My-Studio-Teradici-SG	[X]

**Create Tag** **Cancel** **Save**

- When you're done, click **Save**
- Enter the name and Group ID of your Teradici security group on the [Important Information Cheat Sheet](#). Because not everyone will be adding Teradici, there's not an existing section on the sheet to enter this information, but you can use the Notes section at the bottom to enter it on your own.

## Assign New Security Group to Your Instance

- Click **Instances** in the left hand panel
- Select your **Workstation\_Win\_Teradici** instance
- Right-click and go to **Networking**→**Change Security Groups**
- Click the checkbox next to **My-Studio-Teradici-SG**
- Click **Assign Security Groups**

Change Security Groups

Instance ID:i-0cef64158815cd793

Interface ID:eni-0fc513935293685d5

Select Security Group(s) to associate with your instance

	Security Group ID	Security Group Name	Description
<input type="checkbox"/>	sg-0cf273941d1d327ac	d-916735bbc5_controllers	AWS created security group for d-916735bbc5 directory contr...
<input type="checkbox"/>	sg-0469b547302b78a18	default	default VPC security group
<input checked="" type="checkbox"/>	sg-099b11d866a4bd76e	My-Studio-Deadline-SG	Security Group for Render Scheduler to access Deadline
<input checked="" type="checkbox"/>	sg-0c0efc81a52047acb	My-Studio-Remote-Desktop-SG	Allows for Remote Desktop Connection
<input type="checkbox"/>	sg-06323d5df2f0719fb	My-Studio-SSH-SG	Security Group for SSH
<input type="checkbox"/>	sg-03ec498e33121a00f	My-Studio-Storage-SG	Security group for FSx
<input checked="" type="checkbox"/>	sg-0f7a12e28fc223992	My-Studio-Teradici-SG	Allows for Teradici Connection

Cancel

Assign Security Groups

## Log in to Your Workstation Instance

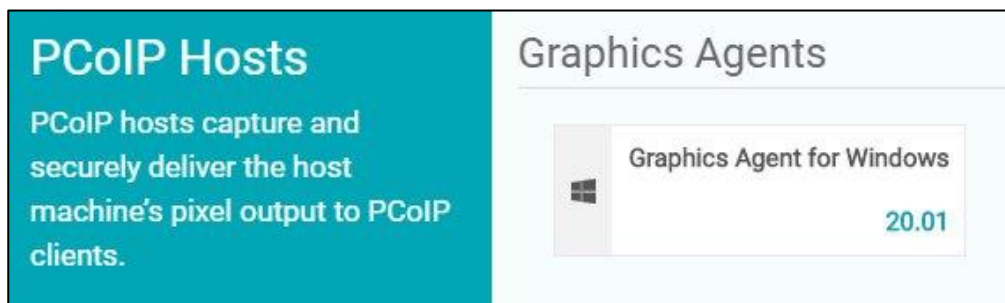
Just as we did in Tutorial 5 when we originally set up your Windows workstation, in order to install Teradici Cloud Access Software on your instance, you'll need to log in as **Administrator**.

- Once your new instance has initialized and passed 2/2 checks, log into it as Administrator:
  - Select your **Workstation\_Win\_Teradici** instance and click **Connect**
  - Click **Download the Remote Desktop File**
  - Open **Remote Desktop**
  - The username should already be set to **Administrator**. In Tutorial 3, we manually set the Administrator password to match your Active Directory Admin password, so you can just enter that here without having to click Get Password. *You can also find your Active Directory Admin password under the Tutorial 2 section of the cheat sheet.*
  - Note: Sometimes it takes an extra few minutes after 2/2 checks are passed for the instance to be fully ready to login. If your first login attempt fails, wait a few minutes and try again.

## Install Teradici on the Workstation Instance

You'll need to download the latest version of the Teradici Cloud Access Software from their website to your new workstation instance. Note: At the time of writing, the latest version was 20.01, but the version you see may be higher.

- Open Firefox on your instance and navigate to the [Teradici Cloud Access Software webpage](#).
- In the **PCoIP Hosts** section, click **Graphics Agent for Windows**



- In the popup window, click **Download**
- Scroll through (and read) the end user license agreement and click **Agree and download**
- **Save** the installer and then run it when the download is complete
  - Choose your language and click **OK**
  - Click **Next** and then click **I Agree** to accept the end user license agreement
  - Leave the destination folder at the default and click **Install**
  - Enter your **Registration code** and click **Next**

PCoIP Graphics Agent Setup

**teradici** License Registration  
Configure licensing for Cloud Access Software.

Please enter your registration code you received by email from Teradici.

☒ Registration code:

A1BC3D6Y2FVM@A123-4567-890B-CDEF

☐ Not now

Click 'Purchase Now' or see Administration guide for other licensing options.

Purchase Now

20.01.0

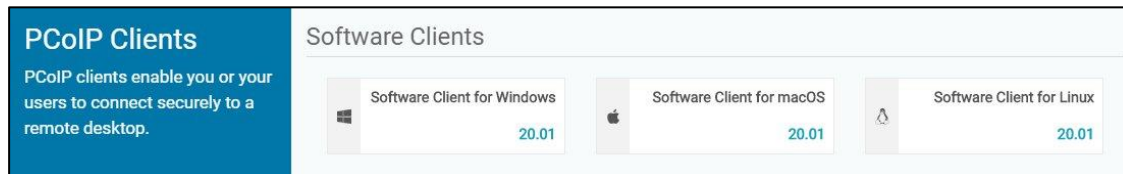
< Back Next > Cancel

- If you recently purchased a subscription or were given a trial license, you should have gotten an email with your registration code. If you are an existing Teradici customer, whoever manages your account should be able to provide you with your code.
- If you did not previously purchase a subscription and want to do so now, click **Purchase Now** and select **Cloud Access +**
  - If you use a proxy server for Internet access, enter its address and port number. Otherwise, just click **Next**.
  - Leave **Reboot now** selected and click **Finish**
- Your Remote Desktop session will disconnect and your instance will start rebooting.

## Download Client to Your Local Machine

In order to connect to your instance, you'll need to download and install the Teradici client software to your local computer.

- On your local computer, go back to the [Teradici Cloud Access Software webpage](#).
- In the **PCoIP Clients** section, download the version of the **Software Client** for the OS of your local machine.

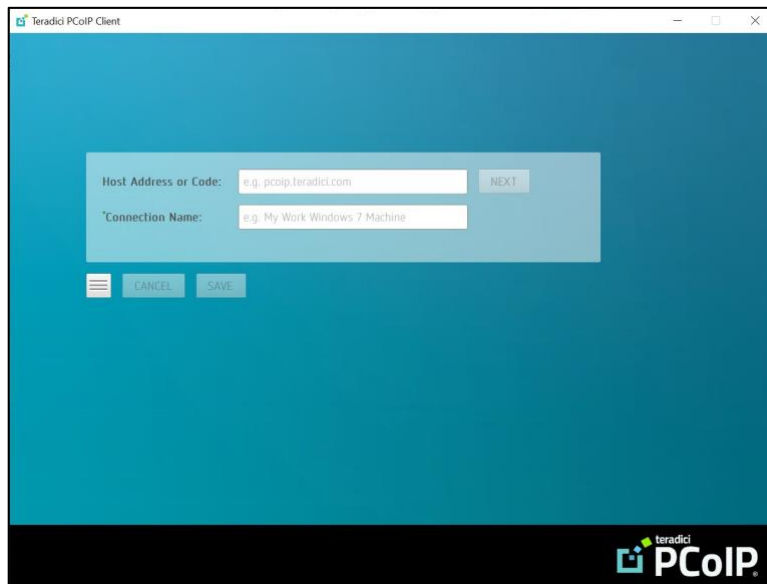


- As before, in the popup window, click **Download**, accept the end use license agreement and click **Agree and download**
- **Save** the installer to your local machine and then run it when the download is complete
  - Proceed through the installation prompts as you did for the Graphics Agent. If you would like a desktop shortcut, make sure to check the box next to that option at the end of the install process.

## Connect Using Teradici

Now that you have the graphics agent installed on your instance and the software client installed locally, you're all ready to connect to your workstation instance!

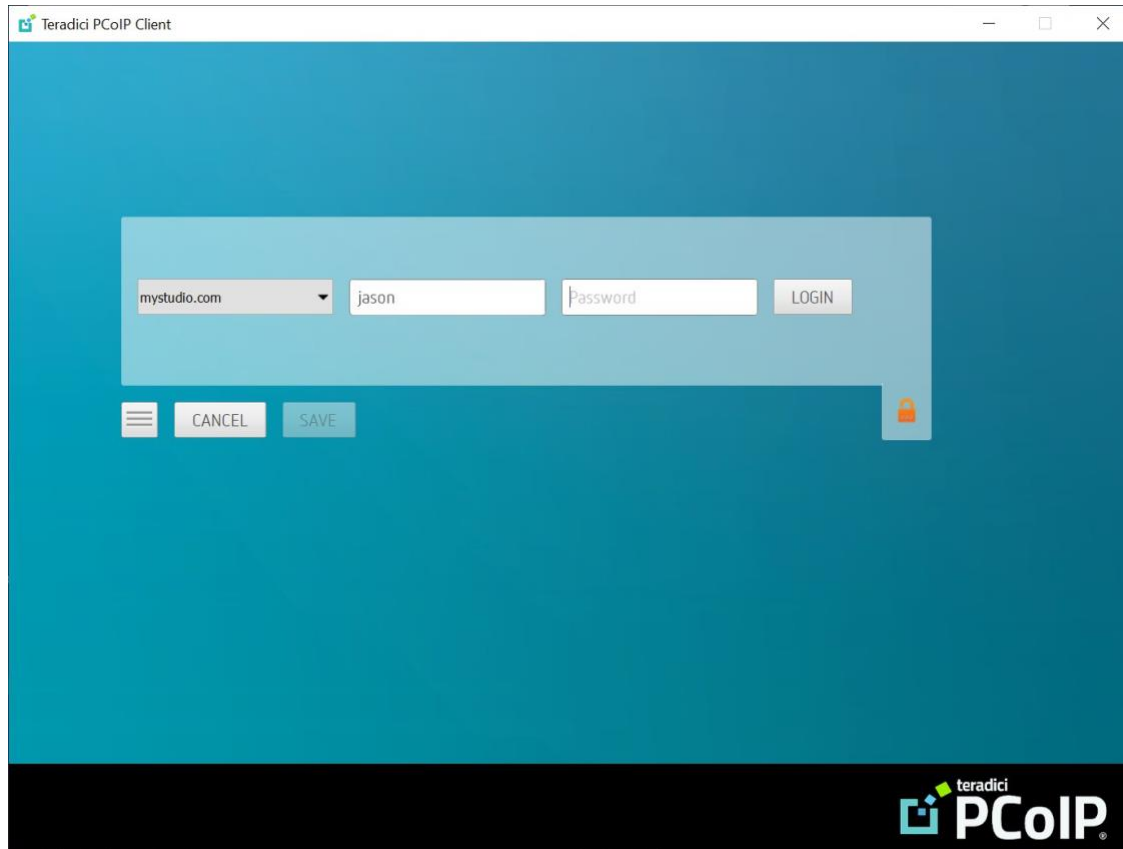
- Run the client software on your local computer by double-clicking the **desktop shortcut** or by going to **Start→Teradici→PCoIP Client (Windows)**



- In the **Host Address or Code** field, enter the **Public IP address** of your Workstation instance.
  - You can locate the public IP address of your instance by going to the list of running instances in the AWS Console, selecting the instance and looking in the Description tab.
- In **Connection Name**, enter a name for this connection (e.g., Workstation-Win)
- Click **Next**
- If you get a popup that says “Cannot verify your connection”, click **Connect Insecurely** to continue
  - Note: Windows sees the connection as insecure, but because the PCoIP protocol used by Teradici is inherently secure, your connection is still completely safe. From a [Teradici support page](#):

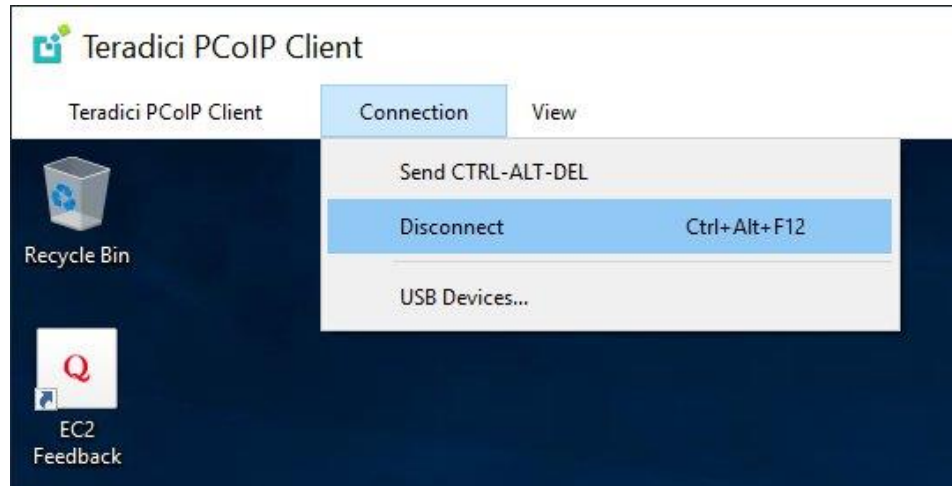


- For links to more information about Teradici security, see the [Appendix](#).
- Enter the **Username** and **Password**
  - Try logging in as a user (e.g., jason). Your Active Directory DNS name (e.g., mystudio.com) should already be selected in the drop down on the left, so there is no need to enter it before the username.
  - Note: Sometimes it takes an extra moment after rebooting for your instance to reconnect to Active Directory. If you don't see your Active Directory DNS name in the drop down on the left, exit the Teradici client, wait a minute then try again.



- Click **Login** to connect to your instance
- Once connected, Teradici works much like Remote Desktop. A new window will open that shows the desktop of your instance.
  - One difference from Remote Desktop is how to disconnect your session. **For now, keep your Teradici session open so that you can start creating a Teradici AMI in the next section.**
  - In the future, when you want to end your Teradici session, you will select **Connection** in the Teradici windows's menu bar, then select **Disconnect**



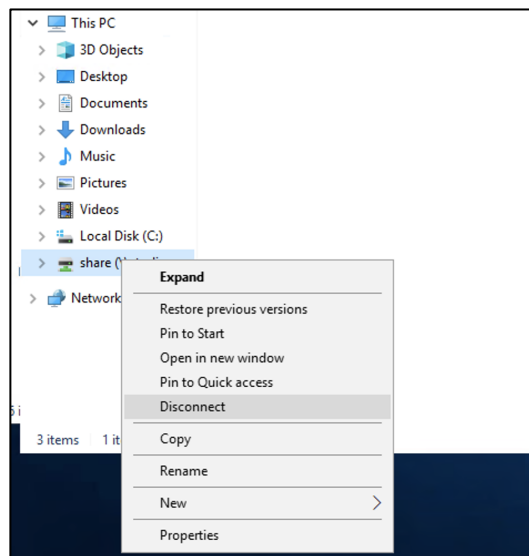


## Create a Teradici AMI and Launch Template

Since you now have Teradici working on one Windows workstation instance, we're going to create a new AMI and launch template to make it really easy to launch new workstations that will already be set up and ready to go.

### Preparing Your Instance

- Disconnect from your FSx Drive (Z:)
  - Open the **File Explorer**
  - Under **This PC** on the left, right-click the Z: choose **Disconnect**



- Go to the **Start Menu** and launch **Ec2LaunchSettings**
- You may be prompted to enter the Admin user (e.g., Admin) and password for your domain
- After the Ec2 Launches Settings window opens (it may take a minute), check the following:
  - Make sure **Set Computer Name** is selected
  - Check that **Administrator Password** is set to **Specify** and input the Administrator Password for your Active Directory(e.g. password for mystudio\Admin)
- Select **Run EC2Launch on every boot**
- Click **Shutdown with Sysprep**

Ec2 Launch Settings

General

**Set Computer Name**

☒ Set the computer name of the instance ip- <hex internal IP>. Disable this feature to persist your own computer name setting.

**Set Wallpaper**

☒ Overlay instance information on the current wallpaper.

**Extend Boot Volume**

☒ Extend OS partition to consume free space for boot volume.

**Add DNS Suffix List**

☒ Add DNS suffix list to allow DNS resolution of servers running in EC2 without providing the fully qualified domain name.

**Handle User Data**

☒ Execute user data provided at instance launch. Note: This will be re-enabled when running shutdown with sysprep below.

**Administrator Password**

☐ Random (Retrieve from console)

☒ Specify (Temporarily store in config file)

☐ Do Nothing (Customize Unattend.xml for sysprep)

These changes will take effect on next boot if Ec2Launch script is scheduled. By default, it is scheduled by shutdown options below.

**Sysprep**

Sysprep is a Microsoft tool that prepares an image for multiple launches.

Ec2Launch Script Location: Found

C:\ProgramData\Amazon\EC2-Windows\Launch\Scripts\InitializeInsta

☒ Run EC2Launch on every boot (instead of just the next boot).

Shutdown without Sysprep Shutdown with Sysprep

Ok Cancel Apply

- Click **Yes**
- This will shut down your instance after a few processes run. Note: It can take a few minutes for Sysprep to run and shut down the instance. Don't worry if it looks like nothing is happening, it will eventually finish and shut down on its own.

## Create Teradici Workstation AMI

- Navigate back to the **EC2 Dashboard** and find the **Workstation\_Win\_Teradici** instance that has just been shut down. Note: If the instance state is still listed as running, refresh the page in your browser to see the updated status. Wait until the status is listed as **stopped** before continuing to the next step.

<input type="checkbox"/>	Worker-Linux	m5.2xlarge	 stopped
<input type="checkbox"/>	Workstation_Win	g3.4xlarge	 stopped
<input checked="" type="checkbox"/>	Workstation_Win_Teradici	g4dn.4xlarge	 stopped

- Right-click the instance and choose **Image** → **Create Image**
- Give it an appropriate name (e.g., My-Studio-Teradici-Workstation-AMI)
- Give your workstation AMI a description if you want.

Create Image

Instance ID ⓘ

i-048a2080bfad37deb

Image name ⓘ

My-Studio-Teradici-Workstation-AMI

Image description ⓘ

Workstation AMI with Teradici

No reboot ⓘ

☐

Instance Volumes

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/sda1	snap-042e3e3a54d44d87c	150	General Purpose SSD (gp2)	450 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Total size of EBS Volumes: 150 GiB  
When you create an EBS image, an EBS snapshot will also be created for each of the above volumes.

Cancel

Create Image

- Click **Create Image**, then click **Close**.
- In the green confirmation message, click the **View pending image** link

- The AMI will take 5-10 minutes to become available. While you're waiting, do the following:
  - Create at a **Studio** tag (e.g., My-Studio) and **Name** tag (e.g., My-Studio-Teradici-Workstation-AMI) for your AMI.
  - Create a new secret for your Teradici registration code, see instructions below.

## Create a New Secret for Your Registration Code

Each new instance you launch needs to be registered as a host before Teradici will work. Instead of requiring you to re-enter the registration code each time you launch an instance, we're going to create a secret in Secrets Manager to securely store it. Then we'll add some lines to the user data of the launch template to automatically retrieve that code and register the new instance as a host.

- Go to **Services** → **Security, Identity, & Compliance** → **Secrets Manager**
- Click **Store a new secret**
- Under **Select secret type**, select **Other type of secrets**
- Under **Specify the key/value pairs to be stored in this secret** enter a key/value pair for your Teradici registration code
  - In the first field, enter a key name of **TeradiciRegistrationCode**
  - In the field to the right of that, enter the **Teradici registration code** that you used when installing Teradici on your workstation instance
  - At the bottom of the page, under **Select the encryption key**, check that **DefaultEncryptionKey** is selected.

AWS Secrets Manager > Secrets > Store a new secret

## Store a new secret

Select secret type [Info](#)

☐ Credentials for RDS database

☐ Credentials for Redshift cluster

☐ Credentials for DocumentDB database

☐ Credentials for other database

☒ Other type of secrets (e.g. API key)

Specify the key/value pairs to be stored in this secret [Info](#)

**Secret key/value** | Plaintext

TeradiciRegistrationCode

[+ Add row](#)

Select the encryption key [Info](#)

Select the AWS KMS key to use to encrypt your secret information. You can encrypt using the default service encryption key that AWS Secrets Manager creates on your behalf or a customer master key (CMK) that you have stored in AWS KMS.

DefaultEncryptionKey ▼

↺

[Add new key](#) [↗](#)

Cancel **Next**

- Click **Next**.
- Under **Secret name**, enter **Teradici/RegistrationCode**
- Enter an optional description
- Under **Tags - optional**, enter Key: **Studio** and Value: **<name of your studio>** (e.g., My-Studio)

AWS Secrets Manager > Secrets > Store a new secret

## Store a new secret

**Secret name and description** [Info](#)

Secret name  
Give the secret a name that enables you to find and manage it easily.

Teradici/RegistrationCode

Secret name must contain only alphanumeric characters and the characters /\_+=.@-

Description - *optional*

This is the Teradici registration code used to register new instances as hosts

Maximum 250 characters

**Tags - optional**

Key	Value - optional	
Studio	My-Studio	Remove
<div>Add</div>		

Cancel Previous **Next**

- Click **Next**.
- Under **Configure automatic rotation**, check that **Disable automatic rotation** is selected.
- Click **Next**.
- Review the information for your secret and if all looks well, click **Store** at the bottom of the page.

## Check that Your AMI Is Ready

Before creating a launch template, you need to make sure that the AMI you created above is ready.

- Go to **Services** → **EC2**

- Click **AMIs** in the left panel
- Find your Teradici AMI in the list of AMIs. Once the status is listed as **available**, you're good to go

## Create Workstation Launch Template

Now that your secret has been stored and your AMI is ready, you can create a Teradici launch template.

- Click **Instances** in the left panel
- Right-click your **Workstation\_Win\_Teradici** instance and choose **Create Template From Instance**
- Name your launch template (e.g., My-Studio-Teradici-Workstation-LT).
- Give it a description if you want

### Launch template name and description

Source instance  
i-048a2080bfad37deb

Launch template name - *required*

My-Studio-Teradici-Workstation-LT

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description

Studio Workstation with Teradici

Max 255 chars

Auto scaling guidance [Info](#)  
Select this if you intend to use this template with auto scaling

☐ Provide guidance to help me set up a template that I can use with auto scaling

► Template tags

- You will need to change the **AMI ID** to point to the one that you previously created

- In the **AMI dropdown**, scroll down to the **My AMIs** section and then select the Teradici Workstation AMI (e.g., My-Studio-Teradici-Workstation-AMI) that you just made.

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

**Amazon machine image (AMI)** [Info](#)

AMI

My-Studio-Teradici-Workstation-AMI

ami-012714ef5af399d9a

Catalog: My AMIs    architecture: 64-bit (x86)    virtualization: hvm

- Under **Network interfaces**, set **Auto-assign public IP** to **Enable**, otherwise you will not be able to connect to the instances you launch
- Also under **Network interfaces**, check **Security Group ID** field and make sure that ids for the following three security groups are listed:
  - your **Deadline Security Group** (e.g., My-Studio-Deadline-SG)
  - your **Remote Desktop Security Group** (e.g., My-Studio-Remote-Desktop-SG)
  - and your **Teradici Security Group** (e.g., My-Studio-Teradici-SG)
  - If any security group is missing, add it now, being careful to put a comma between security group IDs
  - Note: If you don't anticipate needing to use Remote Desktop to connect to your instances anymore, then you may remove the Remote Desktop Security Group from the list of Security Group IDs for increased security.



- Open **Advanced Details**

- In **User data**, expand the size of the entry field to make it easier to read by clicking and dragging on the bottom right corner:

- <shift>+click the image below to open a new browser tab with the text that needs to be entered into the User data entry field:

```

$secret_manager2 = Get-SECSecretValue -SecretId Teradici/RegistrationCode
$secret_teradici = $secret_manager2.SecretString | ConvertFrom-Json
$registration_code = $secret_teradici.TeradiciRegistrationCode
cd 'C:\Program Files\Teradici\PCoIP Agent'
.\pcoip-register-host.ps1 -RegistrationCode $registration_code
  
```

**launch-template\_user-data\_01.txt** - <shift>+click the image above to open the text file in a new tab

- Cut and paste the text from the browser tab into the **User data** entry field, adding it **after** the “Add-LocalGroupMember” line, **but before** the “</powershell>” line.
- After adding those lines, your user data will look similar to this:

User data Info

```

<powershell>
# Variables
$BIOS = "mystudio"
$ADDRESS_1 = "10.0.0.87"
$ADDRESS_2 = "10.0.1.239"
$DNS = "mystudio.com"

$secret_manager = Get-SECSecretValue -SecretId Admin/MyDomainJoin
$secret = $secret_manager.SecretString | ConvertFrom-Json
$password = $secret.AdminPassword | ConvertTo-SecureString -asPlainText -Force
$username = $BIOS + "\Admin"
$credential = New-Object System.Management.Automation.PSCredential($username,$password)
$instanceID = invoke-restmethod -uri http://169.254.169.254/latest/meta-data/instance-id
$index = Get-NetAdapter | Where-object {$_.Name -like "**Ethernet*"} | Select-Object -ExpandProperty InterfaceIndex
Set-DnsClientServerAddress -InterfaceIndex $index -ServerAddresses ($ADDRESS_1, $ADDRESS_2)
Add-Computer -domainname $DNS -Credential $credential -Passthru -Verbose -Force
Add-LocalGroupMember -Group "Remote Desktop Users" -Member "Domain Users"
$secret_manager2 = Get-SECSecretValue -SecretId Teradici/RegistrationCode
$secret_teradici = $secret_manager2.SecretString | ConvertFrom-Json
$registration_code = $secret_teradici.TeradiciRegistrationCode
cd 'C:\Program Files\Teradici\PCoIP Agent'
.\pcoip-register-host.ps1 -RegistrationCode $registration_code
</powershell>
<persist>true</persist>

```

☐ User data has already been base64 encoded

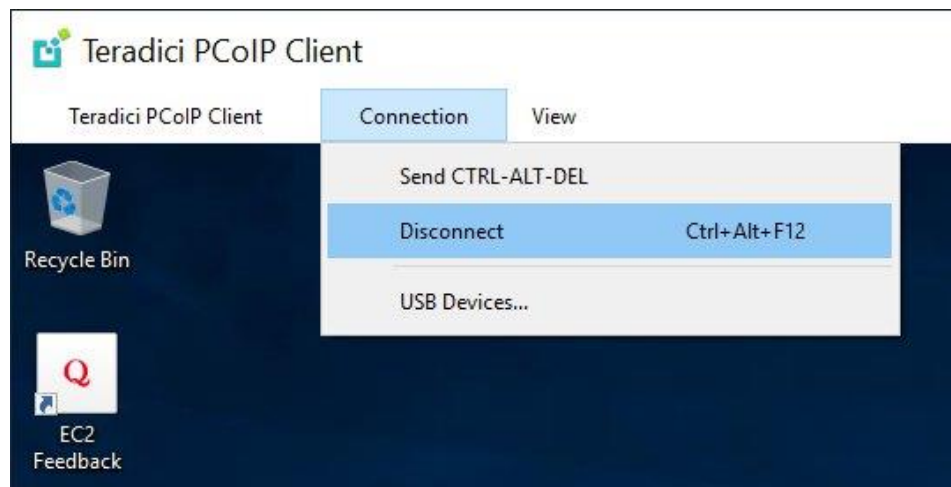
- The new lines that we’ve added will retrieve your Teradici registration code from Secrets Manager and run a command to register this new instance as a Teradici host.
- Click **Create launch template**

You can use your new launch template to launch new Teradici-enabled Windows workstations whenever you need. They will be all ready to connect to with the Teradici client, no further set up needed.

## Launch a New Teradici Workstation

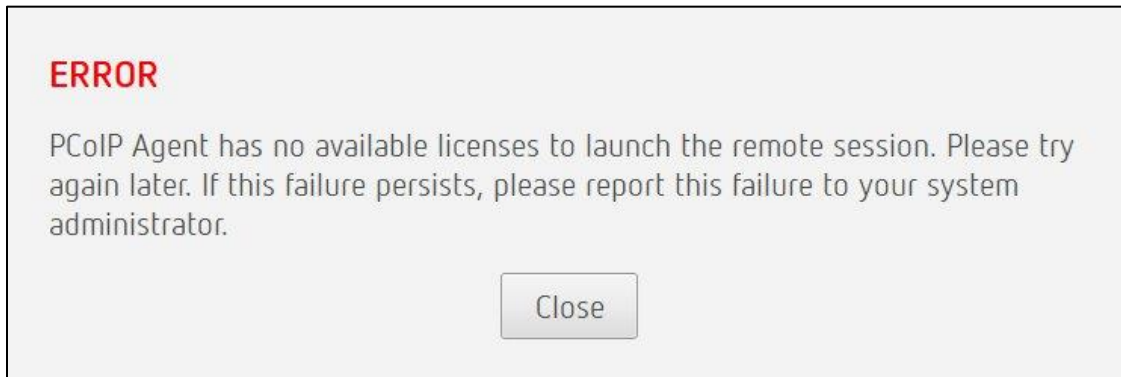
- Go to **Services** → **EC2** and click **Launch Templates** in the left panel

- Select your Workstation launch template (e.g., My-Studio-Teradici-Workstation-LT). *Refer to your cheat sheet, if needed.*
- Choose **Actions** → **Launch instance from template**
- Select the version (if this is your first time running through the tutorial you'll select version 1)
- Choose **Launch instance from template**
- Go back to **Services** → **EC2** click **Instances (running)**
- When the instance is done initializing and shows 2/2 checks passed, you can use the Teradici client to log into using the [same procedure as before](#).
  - Note: As a reminder, sometimes it takes an extra few minutes after 2/2 checks are passed for the instance to be fully ready to login. If your first login attempt fails, wait a few minutes and try again.
- Once you confirm that your new Workstation\_Win\_Teradici instance is running correctly, you can disconnect from it by selecting **Connection** in the Teradici window's menu bar, then **Disconnect**



## Troubleshooting

If you ever get a license error when attempting to connect to an instance via the Teradici client, you can try re-registering it as a host to fix the problem.



Sometimes this can happen if you have only one Teradici demo license that was previously active on another virtual workstation and then try to activate it on a new workstation. Regardless of the exact situation, the instructions below should get things working again.

### Re-register an Instance as a Teradici Host

- Connect to your instance using **Remote Desktop**
- Login as **Administrator**, using your Active Directory Admin password
- Launch **PowerShell**
- Navigate to the PCoIP Agent folder by running:

```
cd 'C:\Program Files\Teradici\PCoIP Agent'
```

- Then run this command, making sure to replace **XXXXXXXXXXXX@YYYY-YYYY-YYYY-YYYY** with your registration code:

```
.\pcoip-register-host.ps1 -RegistrationCode XXXXXXXXXXXX@YYYY-YYYY-YYYY-YYYY
```

- **Restart** your instance
- After the instance has restarted, **reconnect** using **Teradici**

## Optimizing Teradici for Your Network

There are a number of settings for the Teradici Graphics Agent that you can use to optimize performance for your local network. More information on the different settings can be found on Teradici's support page: [Teradici Graphics Agent Configuration Guide](#).

## Review

In this tutorial, you installed Teradici Cloud Access Software on a Windows workstation instance so that you can enjoy high performance desktop streaming. You also created a new AMI and launch template so that you can easily launch as many Teradici-enabled Windows workstations as you need.

## Shut Down Notes

If you no longer need the Teradici workstation instance that you used to create the new AMI and launch template, you can terminate it at this time. Otherwise you can restart that instance and assign it to yourself or one of your artists.

---

## Appendix

### Links to AWS Documentation

- [AWS Direct Connect](#)
- [AWS VPN](#)
- [VPN Connections to AWS VPC](#)

### Links to Other Resources

- [Teradici Cloud Access Software](#)
- [Getting Started Guide - Connecting with a PCoIP Client](#)
- [What is PCoIP Technology?](#)
- [Cloud Access Software Security Features](#)
- [PCoIP Software Client Security Modes](#)
- [Installing Certificates on PCoIP Client for Windows](#)

- [Teradici Graphics Agent Configuration Guide](#)

## Tutorial 9: Add-On - Using a Linux Instance for a Render Scheduler

*Estimated Time to Complete: 2 hours*

### Review of Studio in the Cloud for Digital Content Creation

In our previous 7-part getting started series of tutorials we provided step-by-step instructions for setting up your own fully cloud-based studio for digital content creation. Those tutorials covered a basic setup of virtual workstations, cloud storage and cloud rendering, however there are many ways to expand and customize your studio to meet your production needs.

### Linux vs. Windows Render Scheduler

The Render Scheduler instance that you set up in the previous tutorials will be running for most of the life of your studio, so it makes sense to figure out how to control the cost of that instance as much as possible. One way to do that is to use a **Linux** instance instead of **Windows** as they tend to be cheaper - often saving as much as 50%.

Setting up your Render Scheduler as a Linux instance is very similar to how you set up your Windows-based Render Scheduler - with the main difference being that you cannot connect to a Linux instance with Remote Desktop. Instead, we're going to take advantage of Amazon's **NICE DCV** - a high performance desktop and application streaming protocol. Not only will it handle both Windows *and* Linux instances, but it also has a raft of other great features, including:

- **Multi-screen support** — Lets you expand the session desktop across up to four monitors.
- **Collaboration** — Provides dynamic sessions that support multiple collaborative clients. Clients can connect and disconnect at any time during the session.
- **Support for USB, smart card, and stylus remotization** (Windows / Linux Only)— Lets you use your peripherals in a NICE DCV session just like you would on your local computer.

- **GPU sharing** (Linux NICE DCV servers only) — Lets you share one or more physical GPUs between multiple virtual sessions running on a Linux NICE DCV server.
- **Lossless quality video compression** — Supports lossless quality video compression when the network and processor conditions allow.
- **No cost solution** — There is no additional cost for using NICE DCV on AWS, you only pay the normal cost of the EC2 Linux instance.

## Overview of this Tutorial

This tutorial will teach you how to launch a Linux instance, connect to it via DCV, install a new Deadline Repository and update your workers and workstations to connect to it.

## Prerequisites

### Complete the Previous Tutorials

This tutorial is based off the **Studio in the Cloud for Digital Content Creation** tutorial series. If you haven't completed that series, we highly recommend you do, as we will be utilizing much of the architecture we have already created. You can find the first tutorial here: [Studio in the Cloud Implementation Guide](#).

### Download the DCV client

In order to connect to your Linux instance with DCV, you will need to download the DCV client software to your local computer. You can find download links for Windows, Linux and MacOS here: [NICE DCV Clients](#). Click the OS that you want to use to go to specific download and installation instructions.

### Using AWS Marketplace AMIs

Instead of installing the NICE DCV software from scratch on a basic Linux instance, we're going to start with an Amazon Machine Image (AMI) from the AWS Marketplace. To review, an AMI contains a specific configuration of operating system and other software for your instance. In past tutorials we created AMIs for specific instance types in your studio, such as User Management and Windows Workstations.

The AWS Marketplace hosts thousands of AMIs that have already been created with specific software installed. Using one of these AMIs can save you the time and hassle

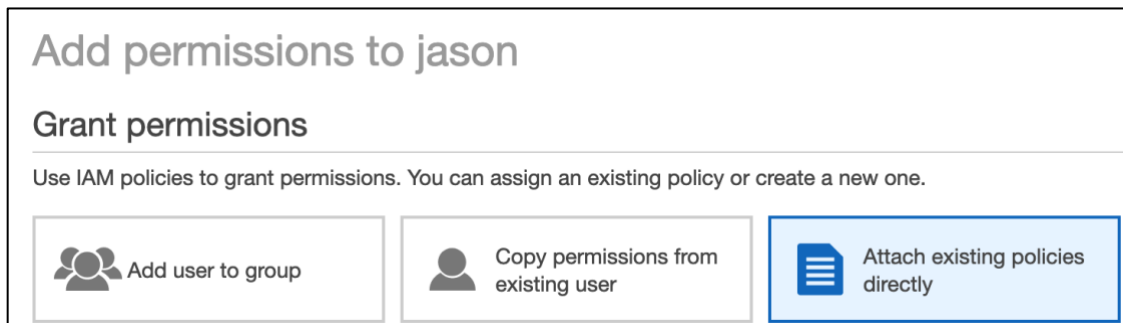


of installing and configuring software on your own. In this case, we're going to start with a Marketplace AMI that already has the NICE DCV Server installed.

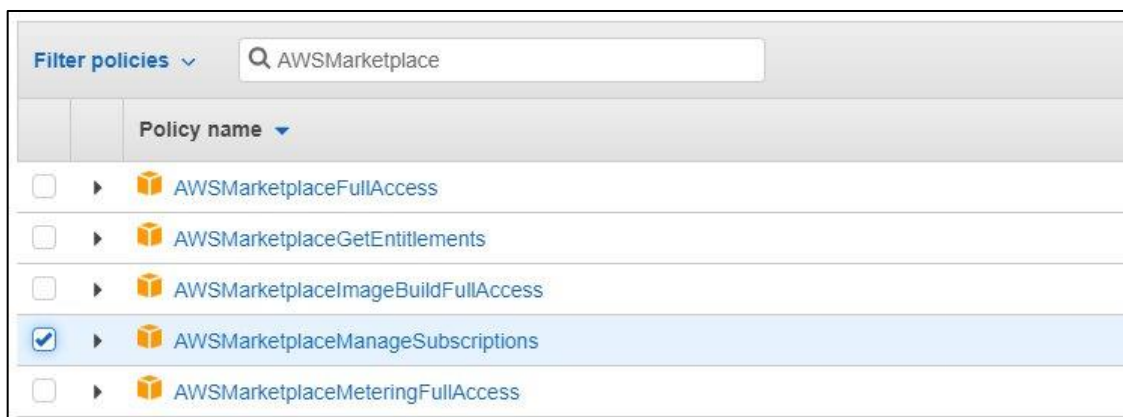
Before you can launch a Linux instance using that Marketplace AMI, you'll need to add some extra permissions to your user.

## Add Marketplace Permissions to Your User

- Go to **Services**→**Security, Identity, & Compliance**→**IAM**
- In the navigation panel on the left, select **Users**
- Click the user for your account
- Click **Add permissions**
- Click **Attach existing policies directly**



- Search for **AWSMarketplace**
- Click the checkbox next to **AWSMarketplaceManageSubscriptions**



- Click **Next:Review**

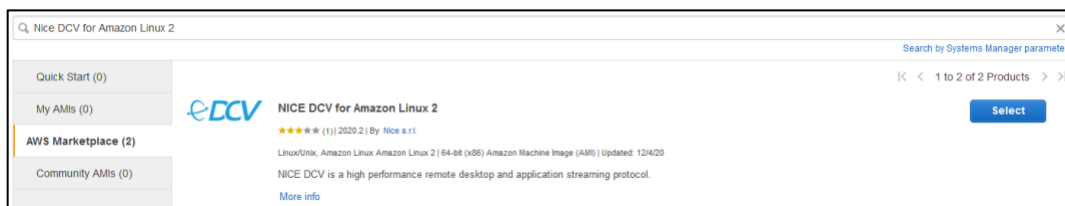
- Click **Add permissions**

## Connect to a Linux Instance with DCV

Now that you have permissions to access the DCV Marketplace AMI, we're going to launch a Linux instance and make sure we can connect to it via the DCV client.

### Launch an EC2 Instance

- Go to **Services** → **EC2** and click **Instances (running)** to see a list of your running instances
- Click **Launch Instances**
- Type **Nice DCV for Amazon Linux 2** in the search bar where you're choosing your Amazon Machine Image
- Click **AWS Marketplace** to list only instances that exist in the AWS Marketplace
- Click **Select** next to **NICE DCV for Amazon Linux 2**



- A pricing list for the various instance types will come up, showing you what the costs will be.
  - Click **Continue**
- Choose an **m5.xlarge** instance type. If this isn't powerful enough, you can always increase it later. Note: If you don't see the m5.xlarge as an option, click the "Previous" button and then click "Cancel" to return to the list of AMIs. Be sure to select "NICE DCV for Amazon Linux 2" and not "NICE DCV for Amazon Linux 2 (ARM)".
- Click **Next: Configure Instance Details**
  - Set the **Network** to your VPC (e.g., My-Studio-VPC)
  - Set the **Subnet** to **Public Subnet A**

- Set **Auto-Assign IP** to **Enable**

**Step 3: Configure Instance Details**  
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the

**Number of instances** ⓘ  [Launch into Auto Scaling Group](#) ⓘ

**Purchasing option** ⓘ ☐ Request Spot instances

**Network** ⓘ  ⓘ [Create new VPC](#)

**Subnet** ⓘ  ⓘ [Create new subnet](#)  
246 IP Addresses available

**Auto-assign Public IP** ⓘ

- Set your **IAM Role** to **EC2DomainJoin**

**IAM role** ⓘ  ⓘ [Create new IAM role](#)

- Click **Next: Add Storage**
  - Set your **Volume Size** to **100 (GiB)**


**Step 4: Add Storage**  
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS
Root	/dev/xvda	snap-0fd2e2cae2a4dd82c	<input type="text" value="100"/>	General Purpose SSD (gp2) ⓘ	100 /

[Add New Volume](#)

- Click **Next: Add Tags**
- Click **Next: Add Tags** and then add **tags** for Studio and Name
  - Set **Studio** to **My-Studio**
  - Set **Name** to **Render Scheduler - Linux**
- Click **Next: Configure Security Groups**

- This setting is already pre-populated with a default security group for connecting via DCV. For now, we'll leave the rules alone and add the other required security groups later.
- Leave the setting as **Create a new security group**
- Change the **Security group name** to **My-Studio-NICE-DCV-SG**
- Leave the **Description** as is
- You will see warnings at the bottom of the screen about not being able to connect to the instance and recommending limiting access to known IP addresses.
  - We'll be adding another security group to this instance later that will allow you to connect to it.

 **Note:** As in earlier tutorials, we are initially opening up this security group to inbound traffic from any IP address. However, if you limited your source IP addresses during your Studio in the Cloud setup, you'll want to do the same here. If you will be accessing your instances over the public Internet, then after initial testing we recommend that you look into an [AWS Direct Connect](#) connection and/or VPN so that your connection will be private and secure. You may use third party VPN software, but AWS also provides a robust set of VPN solutions called [AWS VPN](#). More information about how to connect to your VPC using VPN can be found [here](#).

- Click **Review and Launch**
- Click **Launch** to launch your instance
- Choose the **key pair** that you created when you originally set up your Studio in the Cloud and then click **Launch Instances**. *If you don't remember the name of the key pair you created, you can find it at the bottom of the Tutorial 1 section of the [Important Information Cheat Sheet](#).*
- Once your instance has started launching, you'll receive a **Launch Status** page.

▼ Getting started with your software

To get started with NICE DCV for Amazon Linux

[View Usage Instructions](#)

To manage your software subscription

[Open Your Software on AWS Marketplace](#)

- On this page is a link to **Usage Instructions** that will tell you how to get started with NICE DCV. Click **View Usage Instructions**. Feel free to read through the instructions to get an idea of what we're going to next. Don't worry, we're going to step you through everything below to ensure your instance is ready to use.

## Check Security Group

In order to connect with DCV, you need to make sure the instance security groups allow **inbound traffic** to **TCP port 8443**. Luckily, the default security group allows for that port! However, we will need to add two additional security groups to allow us to connect to the instance and open the ports needed for Deadline so that it can be used as a Render Scheduler.

- Go to **Services** → **EC2** and click **Instances (running)** to see a list of your running instances.
- Select **Render Scheduler - Linux**
- Choose **Actions** → **Security** → **Change security groups**
- Add **My-Studio-Deadline-SG** so you can open all the required deadline ports.
- Add **My-Studio-SSH-SG** so you can connect to the instance via SSH.

**Associated security groups**  
Add one or more security groups to the network interface. You can also remove security groups.

Security groups associated with the network interface (eni-0a71f17b67f702899)

Security group name	Security group ID	
My-Studio-NICE-DCV-SG	sg-028e4503fbb175b6d	<input type="button" value="Remove"/>
My-Studio-Deadline-SG	sg-0cd54f4e20b312679	<input type="button" value="Remove"/>
My-Studio-SSH-SG	sg-028234f46f4516740	<input type="button" value="Remove"/>

- Click **Save**

## Make Sure the Instance Can Access the License File

In order to get access to the DCV license file, your instance IAM profile must have access to the Internet and be able access the required S3 Endpoint. The usage instructions for the NICE DCV AMI have documentation on how to set that up, but we'll walk you through what you need to check below. If you'd like to look at the official instructions for reference, you can find them here: <https://docs.aws.amazon.com/dcv/latest/adminguide/setting-up-license.html>.

Let's check if our current setup has access or not. Essentially, the documentation wants you to give the instance these permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::dcv-license.region/*"
    }
  ]
}
```

Permissions – Note: <shift>+click the image to open the text in a new tab.

What that's saying is that we want to **allow** the instance to be able to **get** an **s3 Object** from a specific resource - in this case a specific DCV license.

Let's look at our current instance and see if we already have permissions that match this.

- Back in the list of **Running Instances**, select your **Render Scheduler - Linux** instance.
- In the **Details** tab on the bottom of the page, find **IAM Role**.
- Click **EC2DomainJoin**
  - This will take you to a summary of the **IAM policies** attached to your instance.



- Click **AmazonS3ReadOnlyAccess**
- Click **JSON** to see what the json definition looks like.



AmazonS3ReadOnlyAccess JSON settings

- As you can see from the **Actions** listed, we have permission for all actions that start with “**s3:Get\***”, and the DCV docs say we need permission for “**s3:GetObject**”. Theoretically, we should already have the permission we need, so there’s nothing additional we need to do here.

## Connect to the Render Scheduler - Linux Instance with SSH

Now it’s time to connect to the remote machine via **SSH** and get it ready to connect via DCV. This process will seem very similar to you if you remember how we set up the Linux worker instances in [Tutorial 6: Setting Up a Linux Farm Worker and Spot Fleet Request](#).

- Go to **Services** → **EC2**

- Click **Instances (running)** to see your list of running instances.
- Check that your **Render Scheduler - Linux** instance is running and has passed 2/2 checks, then **Select** it.
- Click **Connect**
- Choose **EC2 Instance Connect**
- Make sure **User name** is set to **root**

**Connect to instance** [Info](#)

Connect to your instance i-09e171dbd2da810a7 (Render Scheduler - Linux) using any of these options

**EC2 Instance Connect** | Session Manager | SSH client | EC2 Serial Console

Instance ID  
i-09e171dbd2da810a7 (Render Scheduler - Linux)

Public IP address  
34.236.148.201

User name

Connect using a custom user name, or use the default user name root for the AMI used to launch the instance.

**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**

- Click **Connect**
- A window will open with access to your instance. Note: The browser-based SSH connection will automatically timeout after a few minutes of inactivity and the window will become unresponsive. If that happens to you, close the connection window and open a new one.
- Run the following command to update the install.

```
yum -y update
```

- Note: It may take a minute or two for the update to complete running.



## Set the Password for root

One difference between the Linux-based Render Scheduler and the Windows one is that your login will be different. For Linux, we'll be using the user **root** instead of **Administrator**. Thus, we need to create a password for this user.

- Run the passwd command:

```
passwd root
```

- Enter a password for root that you can remember. You may opt to choose the same password as your Active Directory Admin password. *But if not, be sure to enter the new password in the Notes section of your cheat sheet.*

## Create a DCV Session

Now it's time to prepare your instance to be connected to. To do this, you'll need to create a "session":

- Type the following command to create a session

```
dcv create-session render-scheduler --owner root
```

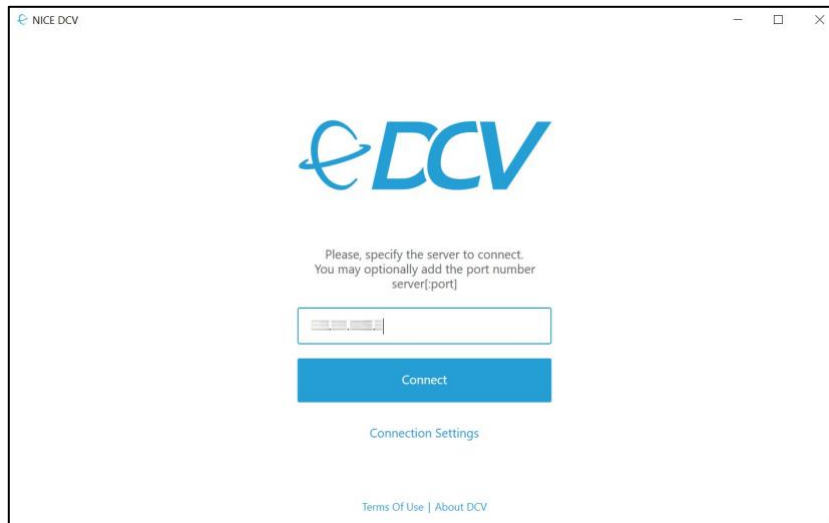
- Note: you can name your session whatever you wish. In this case, we're naming it "render-scheduler" so we know what this session will be used for.
- If you'd like more information on managing NICE DCV sessions, refer to this documentation: [Managing NICE DCV Sessions](#)

## Connect to Your DCV Session

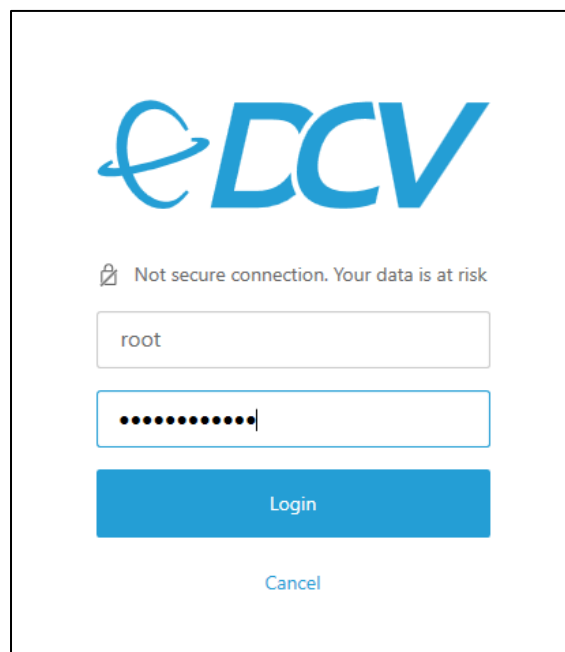
- Back in the **AWS Console**, select your **Render Scheduler - Linux** instance from the list
- In the **Description** tab below, copy the **IPv4 Public IP Address**.

Instance: i-02d5ead5fb74df2fe (Render Scheduler - Linux)		Public DNS: us-west-1.compute.amazonaws.com	
Description	Status Checks	Monitoring	Tags
Instance ID	i-02d5ead5fb74df2fe		
Instance state	running		
Instance type	m5.xlarge		
Elastic IPs			
Availability zone	us-west-1b		
		Public DNS (IPv4)	
		us-west-1.compute.amazonaws.com	
		IPv4 Public IP	
		[Redacted]	
		IPv6 IPs	
		-	
		Private DNS	
		ip-10-0-0-153.us-west-1.compute.internal	
		Private IPs	
		10.0.0.153	

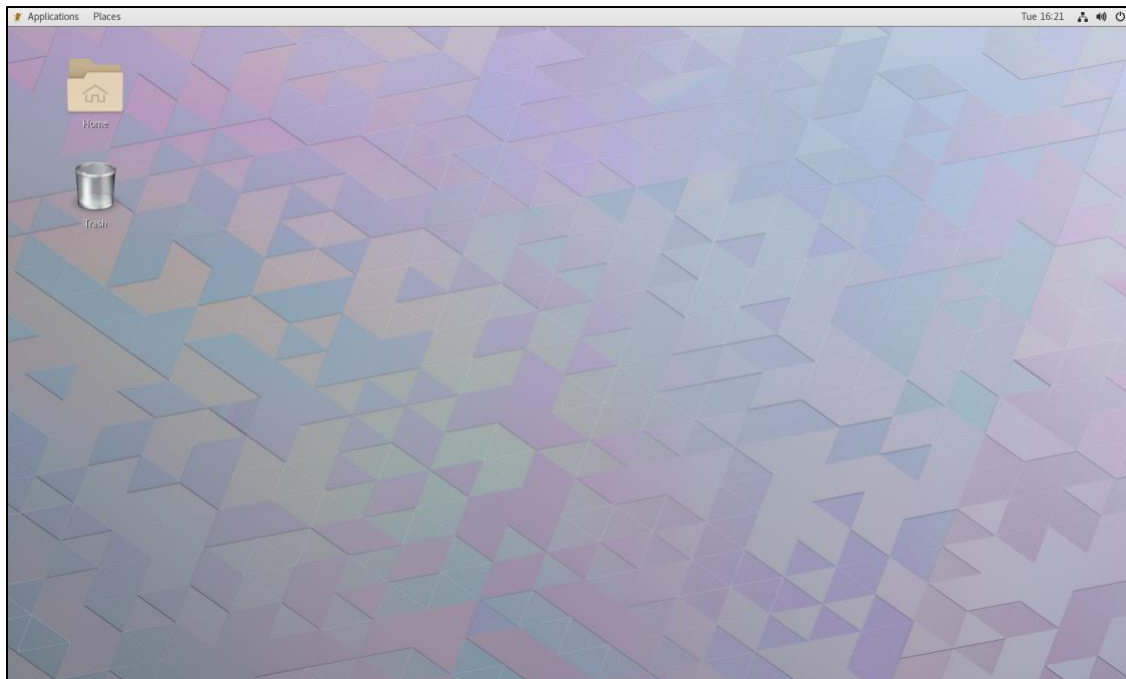
- Run the **DCV Client** on your local computer
- Paste the Public IP Address that you copied above into the **Hostname/IP Address** field.



- Click **Connect**
- When it pops up a warning about the connection not being secure, click **Trust** (or **Proceed** if on MacOS)
- Enter your **Username** as **root**, and **Password** with the password you set earlier.



- Click **Login**
- You are now connected to your Linux instance via the DCV Client!



## Make Sure DCV Always Runs

In this example we *manually* created a session in order to connect to DCV. For the purpose of our Studio in the Cloud, we would rather the DCV service started *automatically*, ensuring we can start and stop the machine and still connect via DCV.

### Enable the DCV Service

- Load a terminal in the DCV Session by going to **Applications** → **Favorites** → **Terminal**
- Execute the following code to make sure the DCV Service starts automatically.

```
systemctl enable dcvserver
```

- Note: Pasting with <ctrl>+v does not work with the Terminal app on your Linux instance, but you can right-click and select “Paste” instead.

## Create a Script to Start the Session

Next, we'll need to ensure that the session starts automatically when our instance restarts. To do this we're going to create our own service similar to the one we enabled above, except it will be our own code that gets run every time the system starts!

To do this, we'll first create a bash script that will start our session for us. If you're not comfortable editing text with **vi**, you may want to download a more visual text editor. It's relatively easy to install **gedit** on your instance, so if you'd like to you can do that now.

```
yum -y install gedit
```

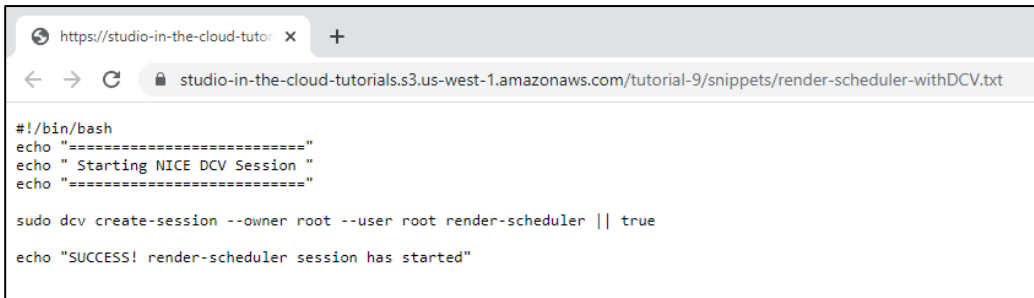
- Now from a shell, create a file called `render-scheduler.sh` with either **gedit** or **vi**. If you need a refresher on vi commands, see this webpage: [Vi Cheat Sheet](#). `gedit` is a GUI-based editor, so it may be more intuitive to use if you're not familiar with `vi`.
  - If you're using **vi**:

```
vi /usr/bin/render-scheduler.sh
```

- If you're using **gedit**:

```
gedit /usr/bin/render-scheduler.sh
```

- Enter the following code into the text editor and save it.



```
#!/bin/bash
echo "=====
echo " Starting NICE DCV Session "
echo "=====

sudo dcv create-session --owner root --user root render-scheduler || true

echo "SUCCESS! render-scheduler session has started"
```

**render-scheduler.sh** – Note: <shift>+click the image to open the text in a new tab.

- Save and quit. Once the file is saved, you'll need to make it executable.

```
chmod +x /usr/bin/render-scheduler.sh
```

- This file will start the `render-scheduler dcv` session whenever it's executed. Let's give it a try:

```
/usr/bin/render-scheduler.sh
```

- Did you receive an error stating it had trouble creating the session? Since we manually ran the same “sudo dcv create-session” command earlier to create a DCV session, we’re getting an error that one already exists. But when our instance restarts that command will only get run once by the script we just created, so everything will be fine.

## Create the Service to Run the Script

The next step is to create the **service** that will execute this file every time the instance is started. The nice thing about working this way is that we can modify this file to add more options any time we want and it will be executed again whenever the system is restarted. It’s similar to the user data settings we have on the instance, but in this case it's much more controllable.

- Create the service file using either **gedit** or **vi**:
  - If you're using **vi**:

```
vi /etc/systemd/system/render-scheduler.service
```

- If you're using **gedit**:

```
gedit /etc/systemd/system/render-scheduler.service
```

- Enter the following code into the text editor and save it.

```
1
2 [Unit]
3 Description=Start DCV Render Scheduler session service.
4
5 [Service]
6 Type=simple
7 ExecStart=/bin/bash /usr/bin/render-scheduler.sh
8
9 [Install]
10 WantedBy=multi-user.target
11
```

**render-scheduler.service** – Note: <shift>+click the image to open the text in a new tab.

- This is a relatively simple service that will simply execute the file we created earlier (/usr/bin/render-scheduler.sh).
  - The key here is the line **ExecStart** that tells the service what script to run.

- Quit the editor after the file is saved.
- Next we'll **start** the service

```
systemctl start render-scheduler
```

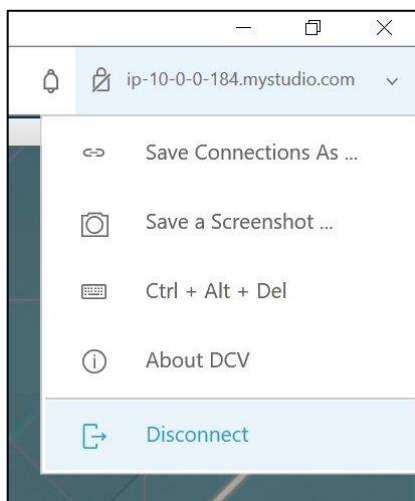
- And then make sure it will start automatically on boot.

```
systemctl enable render-scheduler
```

## Check that the Service Is Working

If this has all been set up correctly, you should be able to stop your instance, start it again, and then connect to the new IP address using the DCV client, without having to manually start the session.

- **Disconnect** your session by clicking on the drop down menu in the upper right



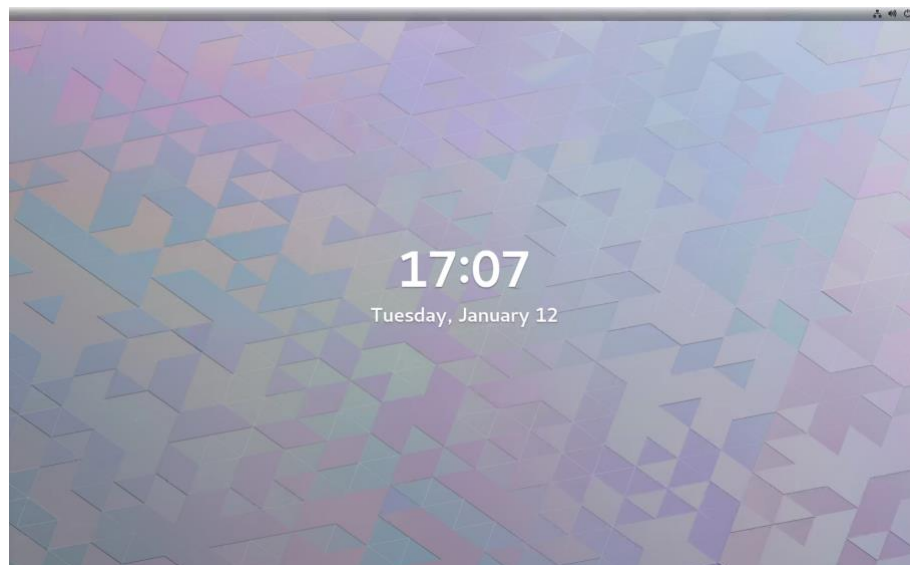
corner of the DCV client window or by closing the client altogether

- Go **EC2** → **Instances**
- Select the **Render Scheduler - Linux** instance
- Choose **Instance state** → **Stop instance**
- After the instance has stopped, choose **Instance State** → **Start instance**
- Once the instance has spun up and is ready to use, copy the **public IP address**.

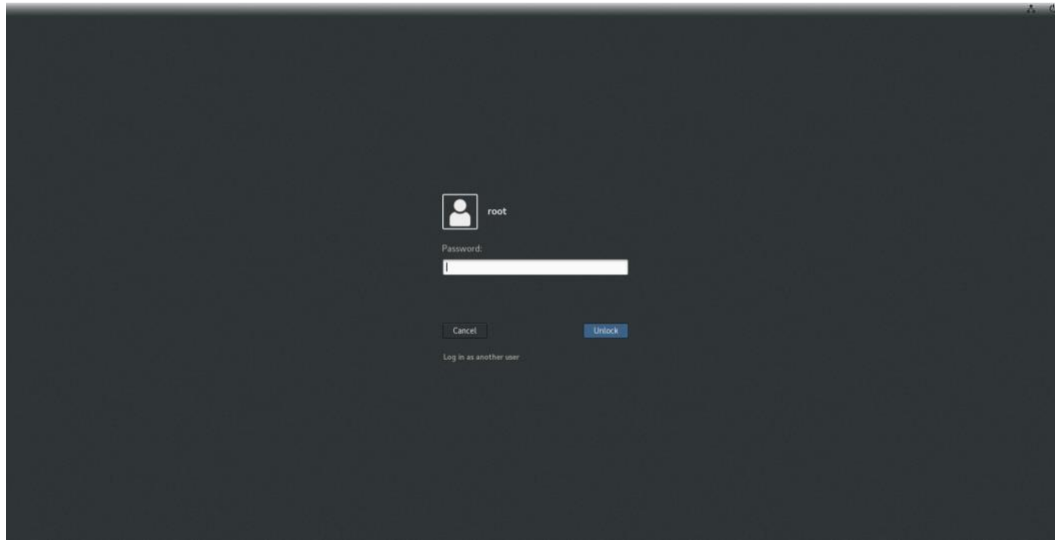
- Note: The public IP address will change every time the instance stops and then starts. If you want to maintain the same public IP address, this is possible through something called an Elastic IP Address.
- Open the **DCV Client** and connect to the **new public IP address**
- Log in using **root** and the password from before.
- If you were able to start a session - congratulations! You've successfully created a Linux service to automatically create a DCV Session! We'll be working with this service a bit later to do more magic as well.

## Troubleshooting

- If you weren't able to reconnect your DCV session, try connecting to the instance using the SSH techniques in the beginning of the tutorial. Then make sure all your code was entered correctly.
- If your Linux session is left idle for too long, it may go to the lock screen. If that happens you will need to re-enter the password for root to unlock the session. The lock screen looks like this:



To unlock, hit the <enter> key or start typing your password and the password entry prompt should appear:



## Join Active Directory

- To join Active Directory from your Linux Render Scheduler run these commands. Make sure you **update the red text** to match the **name of your Active Directory**. You'll be replacing the two instances of "mystudio.com" with your Active Directory's DNS Name.
- Load a **terminal** in the DCV Session by going to **Applications** → **Favorites** → **Terminal**
- Enter the commands below, making sure to swap out "mystudio.com" for your **Active Directory's DNS name**, if different. *You can find the DNS name under Tutorial 2 on the cheat sheet.*
- First install the specific tools to make it possible to connect to Active Directory.

```
yum -y install sssd realmd krb5-workstation samba-common-tools
```

- Then join Active Directory

```
realm join -U Admin@mystudio.com mystudio.com --verbose
```

- You should receive a message that says: **Successfully discovered: mystudio.com** (or your DNS name, if different)



- Type in the password for your Active Directory and you will see a message that says: **Successfully enrolled machine in realm**
- In preparation for the next step, it is recommended to perform a `mkdir` in `/mnt/` to create a directory for the FSx mount.

```
mkdir /mnt/studio
```

## Mount FSx File System

- First run this command to install some utilities:

```
yum -y install cifs-utils
```

- Next run the command below and enter the Active Directory Admin password when prompted. Make sure to replace MYSTUDIO with your **Active Directory's DNS Name** and type it in all capital letters.

```
kinit Admin@MYSTUDIO.COM
```

- Finally, run the command below to mount your FSx drive. Again, make sure you update the red text to match your **Active Directory DNS Name** and the **FSx DNS Name**. *You can find this information under Tutorial 2 and Tutorial 3 on your cheat sheet.*
  - Note: Check the direction of the slashes when you enter the FSx DNS Name since they are “\” when on Windows, but “/” when on Linux, as we are here.

```
mount -t cifs -o user=Admin@MYSTUDIO.COM,cuid=$(id -u),uid=$(id -u),sec=krb5 //fs-0c0b4fab1db1b9a0e.mystudio.com/share /mnt/studio -o vers=3.0
```

***Again, note which text is capitalized. These commands ARE case sensitive.***

- To test and see if it mounted correctly, do an **ls** of the directory

```
ls /mnt/studio
```

- If you see a few folders returning back, then you've connected correctly!

## Modify render-scheduler.sh to Mount Active Directory

Now that we have Active Directory mounted in this session, we'll want to modify our render-scheduler service to always mount Active Directory when we're starting and stopping our instance.

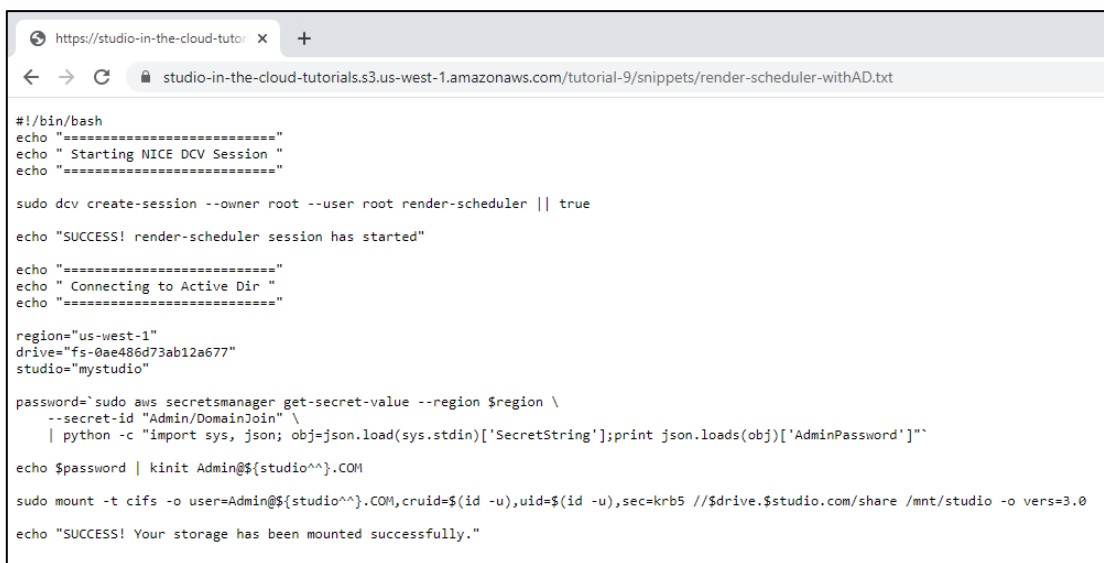
- Edit the render-scheduler.sh file with your favorite editor (vi or gedit)
  - If you're using **vi**:

```
vi /usr/bin/render-scheduler.sh
```

- If you're using **gedit**:

```
gedit /usr/bin/render-scheduler.sh
```

- Add the lines about connecting to Active Directory after the previous lines you created. The final script should look like the following. Don't forget to modify the region, drive, and studio in lines 15, 16, and 17 based on your own settings.



```
#!/bin/bash
echo "=====
echo " Starting NICE DCV Session "
echo "=====

sudo dcv create-session --owner root --user root render-scheduler || true

echo "SUCCESS! render-scheduler session has started"

echo "=====
echo " Connecting to Active Dir "
echo "=====

region="us-west-1"
drive="fs-0ae486d73ab12a677"
studio="mystudio"

password=$(sudo aws secretsmanager get-secret-value --region $region \
--secret-id "Admin/DomainJoin" \
| python -c "import sys, json; obj=json.load(sys.stdin)['SecretString'];print json.loads(obj)['AdminPassword']")

echo $password | kinit Admin@${studio^^}.COM

sudo mount -t cifs -o user=Admin@${studio^^}.COM,cuid=$(id -u),uid=$(id -u),sec=krb5 //$drive.$studio.com/share /mnt/studio -o vers=3.0

echo "SUCCESS! Your storage has been mounted successfully."
```

**render-scheduler.sh** with added Active Directory code. - Note: <shift>+click the image to open the text in a new tab.

- IMPORTANT: You will need to update the items highlighted below in **yellow** with specific information for your studio

```
15 region="us-west-1"
16 drive="fs-0ae486d73ab12a677"
17 studio="mystudio"
```

- The next time you stop and start your instance, it should automatically create a session and mount active directory for you. Feel free to test if you want, or continue on and start installing Deadline.

## Install Deadline Tools

### Install Deadline Repository

First we'll install the Deadline Repository. Luckily, you should have all the installer files in your `/mnt/studio/installers/thinkbox` folder from when you set up Studio in the Cloud initially.

- Load a **Terminal** in the DCV Session by going to **Applications** → **Favorites** → **Terminal**
- Run the command below to go to the installers location for thinkbox

```
cd /mnt/studio/installers/thinkbox
```

- You should have the DeadlineRepository and DeadlineClient installers for Linux there. Run the command below. Note: You may need to replace the version number (e.g., 10.1.12.1) with number of the version that you are using:

```
./DeadlineRepository-10.1.17.4-linux-x64-installer.run
```

If you don't have the installers, follow the instructions from [Tutorial 4: Building a Render Scheduler with AWS Thinkbox Deadline](#) and then run the installer.

The steps should be very similar to the ones you followed initially for Windows. The only difference is that we're going to create a new location to install the certificates so we don't overwrite the old ones.

- Click **Forward** when the Welcome message appears
- Accept the **License Agreement** and click **Forward**
- Leave the **Repository Directory** at the default and click **Forward**
- Leave the database type as **MongoDB** and click **Forward**
- Select **Install a new MongoDB database on this machine** and click **Forward**

- Click **Yes** to accept the new file descriptor limit
- Click **Forward** when it asks to download MongoDB
- Click **I accept the agreement** to accept the SSPL license agreement and then click **Forward**
- Leave all the **MongoDB** installation options at the default and click **Forward**
- We're going to change the location of the **Certificate Directory**
  - Click the **Browse icon** next to the Certificate Directory text field
  - Navigate to **/mnt/studio/app\_env/thinkbox**
  - Click **New Folder**
  - Name the new folder **DeadlineCertificates\_linux** and click **Create**
  - Double-click the new folder in the file browser
  - Click **OK**
- For **Certificate Password** and **Confirm Password**, enter your Active Directory admin password. Note: For simplicity, we are re-using the Active Directory admin password here for this tutorial setup. However, if you desire, you can use a different password. *In that case, make sure to enter the Linux Certificate Password you choose in the Notes section of the Important Information Cheat Sheet.*
- Uncheck **Use client certificate for DB user authentication** and click **Forward**
- Leave **Enable Secrets Management** selected and click **Forward**
- For **Admin Username** enter your **Active Directory admin username** (e.g., Admin)
- For **Password** enter your Active Directory admin password and click **Forward**. Note: Again, like we did with the Certificate Password, we are re-using the Active Directory admin password. If your Active Directory admin password does not meet the requirements for the Deadline installer, you may need to create a new password. *If so, be sure to enter this Linux Deadline Secrets Management Password in the Notes section of the Important Information Cheat Sheet.*
- Click **Forward** two more times
- **Wait** for it to install the Deadline Repository

- Click **Finish** when it's done.

## Install Deadline Client

Now we'll install the Deadline Client. Again, it's very similar to what we've done in the past.

- Install **lsb**

```
yum -y install lsb
```

- Change directories so you are inside installers/thinkbox.

```
cd /mnt/studio/installers/thinkbox
```

- To install the Deadline Client run this command. Note: Again, the Deadline Client may be a different version than what is specified here. Adjust as needed.

```
./DeadlineClient-10.1.17.4-linux-x64-installer.run
```

- Give these answers to the prompts:
  - Click **Forward** when the Welcome message appears
  - Accept the **License Agreement** and click **Forward**
  - Leave **Installation Directory** at the default and click **Forward**
  - Select **Remote Connection Server** and click **Forward**
  - Leave the **Repository Directory** as default and click **Forward**
  - Set the **Database TLS Certificate** to `/mnt/studio/app_env/thinkbox/DeadlineCertificates_linux/certs/Deadline10Client.pfx`
  - For **TLS Certificate Password** enter the Linux Certificate Password you used when installing the Repository above. (This may be the same as your Active Directory admin password. *If not, you can find it in the Notes section of your cheat sheet.*) Click **Forward**
  - Leave **Assign server role and grant master key access** selected and click **Forward**

- On the OS User Credentials page, check that Username is set to **root** and click **Forward**
- Enter the Linux Deadline Secrets Management **Admin Username** and **Password** that you entered during the Repository installation instructions above. (Again, this may be the same as your Active Directory admin password. *If not, look in the Notes section of your cheat sheet for the Linux Deadline Secrets Management password you chose.*)
- Leave **Name of the current master key** at the default value of **defaultKey** and click **Forward**
- Leave **Launch Worker When Launcher Starts** checked and click **Forward**
- Leave **Block auto upgrade via a secure setting** selected and click **Forward**
- Leave the **Remote Connection Server (RCS)** options at the default and click **Forward**
- Make sure **Generate New Certificates** is enabled and click **Forward**
- Set the **Certificate Directory** to the same directory where you put the Deadline10Client.pfx certificate: **/mnt/studio/app\_env/thinkbox/DeadlineCertificates\_linux**
- Leave the **Certificate Password** blank and click **Forward**
- Click **Forward** again to continue the install
- Click **Finish** when you're done.

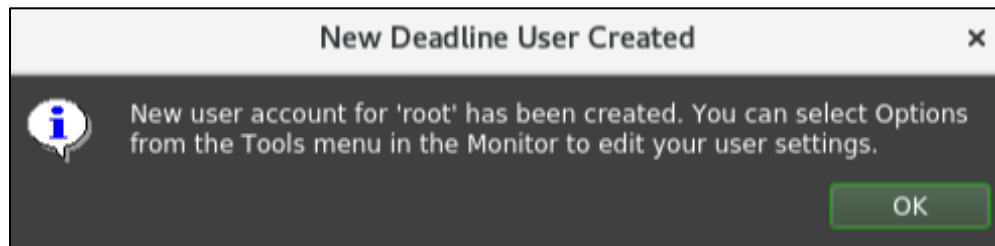
## Launch Deadline RCS, Deadline Pulse, and Deadline Monitor

- In a Terminal window, run the following command to run Deadline RCS

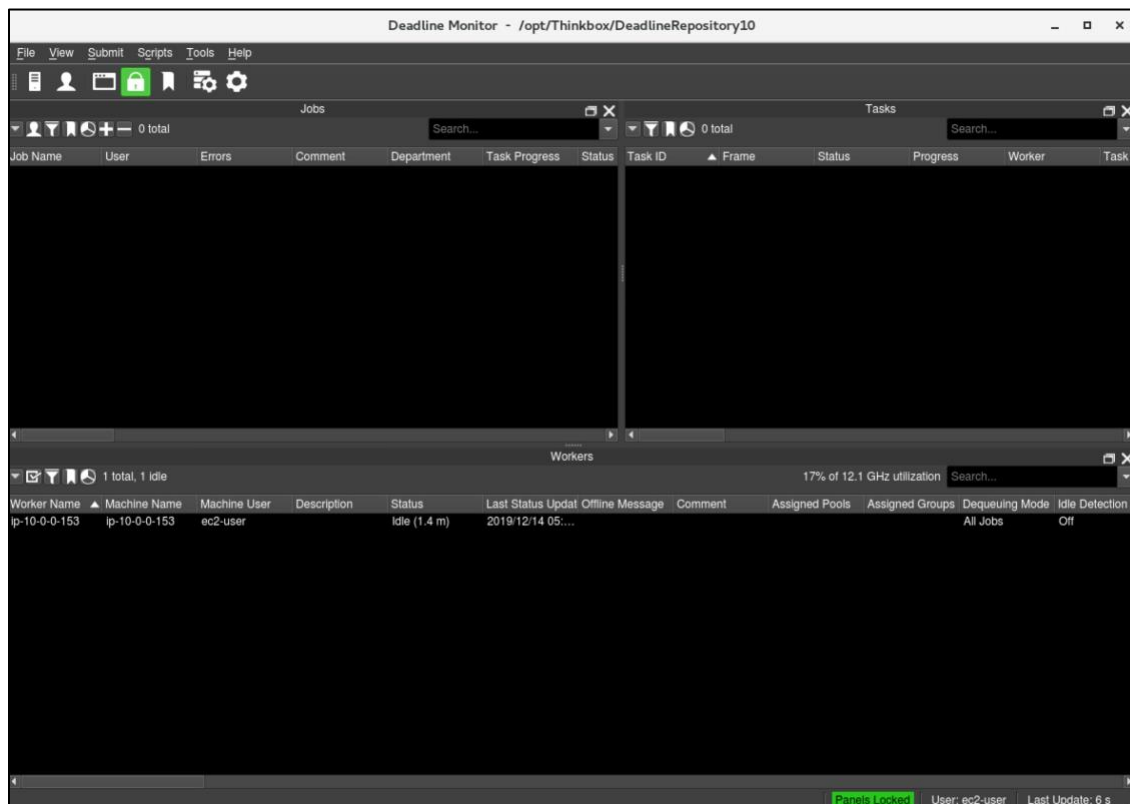
```
/opt/Thinkbox/Deadline10/bin/deadlinercs &
```

- Minimize the Terminal window to leave RCS running
- Run Deadline Pulse by going to **Applications** → **Other** → **Deadline Pulse 10**
- You can minimize the Deadline Pulse window to leave it running

- Run the Deadline Monitor by going to **Applications** → **Other** → **Deadline Monitor 10**
- At this point the **Deadline Worker** application should have launched automatically as well
- You can also minimize the Deadline Worker window to leave it running in the background
- It may take a minute for the **Deadline Monitor** to start up, but once it does, it will create a new account for you called **root**



- After clicking **OK** in the window above, the Deadline Monitor window should open



## Create an Identity Registry Setting

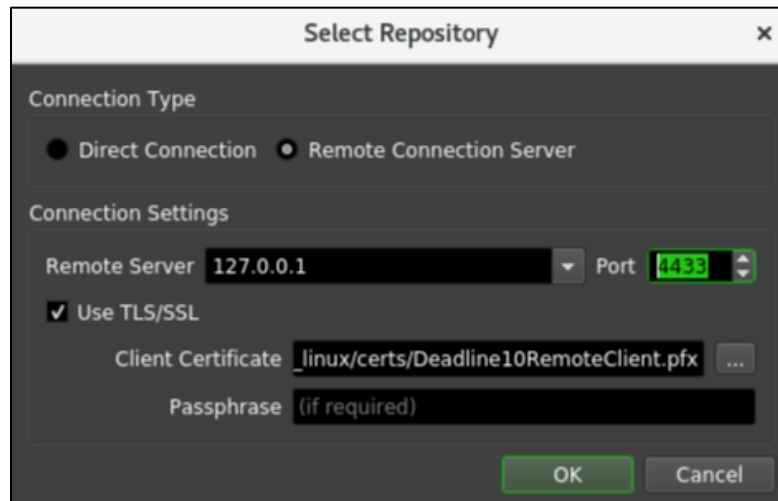
Deadline securely manages sensitive information such as API keys, Usage Based Licensing activation codes, passwords, etc. using secrets. This is similar to how we used AWS Secrets Manager to store your Active Directory Admin password in Tutorial 3.

Deadline assigns “server” or “client” roles to each machine that attempts to connect to the Deadline Repository. When we installed the Deadline Client above, your Render Scheduler was automatically assigned to the “server” role since it will be running the Remote Connection Server (RCS). We need to make sure that the workstation and farm worker machines that are created in other tutorials are assigned “client” roles, otherwise they will not have access to the Deadline secrets. To do that, we’re going to create an Identity Registration Setting in the Deadline Monitor.

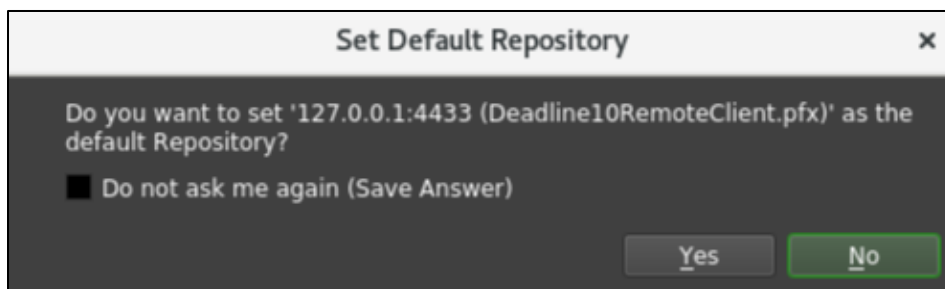
By default the Render Scheduler connects to the Repository using a direct connection, but in order to manage identity settings, we must connect using a secure remote connection instead. So first we’ll change our repository connection and then move on to creating the identity settings.

1. In the **Deadline Monitor**, choose **File**→**Change Repository...**
2. For **Connection Type** select **Remote Connection Server**
3. Next, under **Connection Settings** set the following:
  - a. **Remote Server** to **127.0.0.1**
  - b. **Port** to **4433**
  - c. Make sure **Use TLS/SSL** is checked
  - d. Set **Client Certificate** to  
**/mnt/studio/app\_env/thinkbox/DeadlineCertificates\_linux/certs/Deadline10RemoteClient.pfx**
  - e. Leave **Passphrase** blank



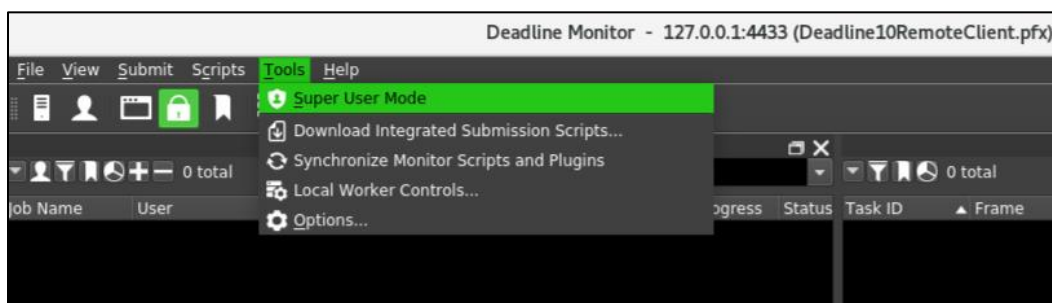


4. Click **OK**
5. Click **No**

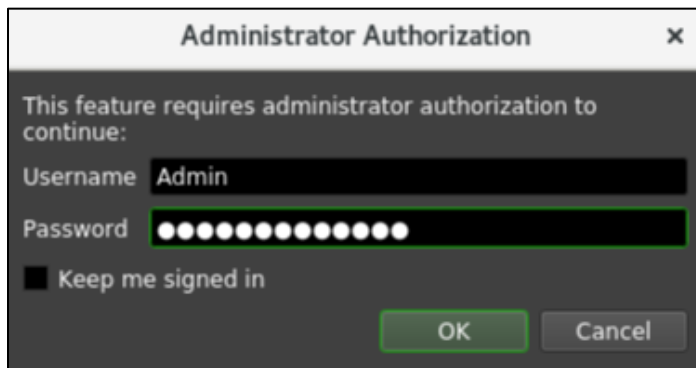


Note: We are only temporarily connecting to the Repository with a Remote Connection. We don't want to make this the default, which is why we're answering "no" above. The next time you close and reopen the Deadline Monitor, it will automatically reconnect using a Direct Connection again.

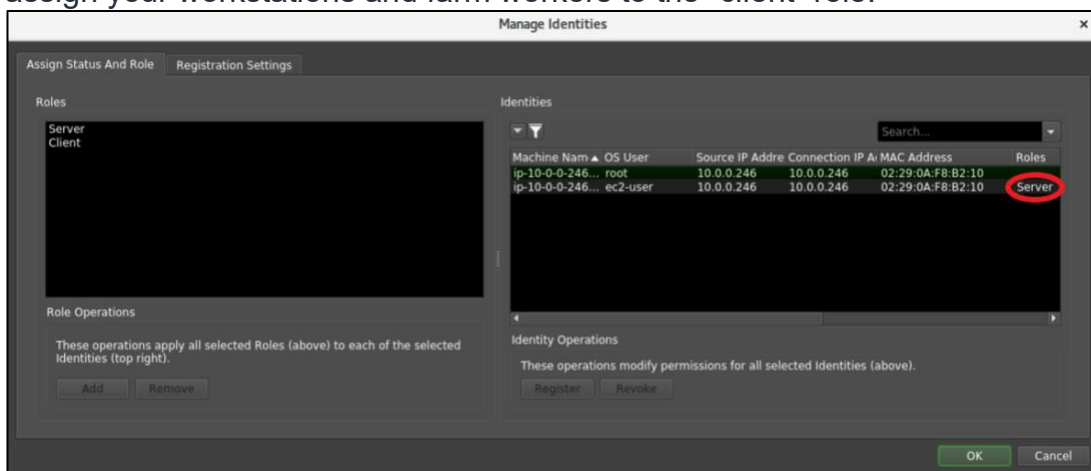
6. Wait until it reconnects to the Repository, then choose **Tools** → **Super User Mode**



7. Then choose **Tools** → **Manage Identities...**
8. Enter the Secrets Management **Admin Username** and **Password** that you created when you installed the Deadline and click **OK**. (Again, this may be same as your Active Directory admin password. *Refer to the Notes section of your cheat sheet if you used a different password.*)

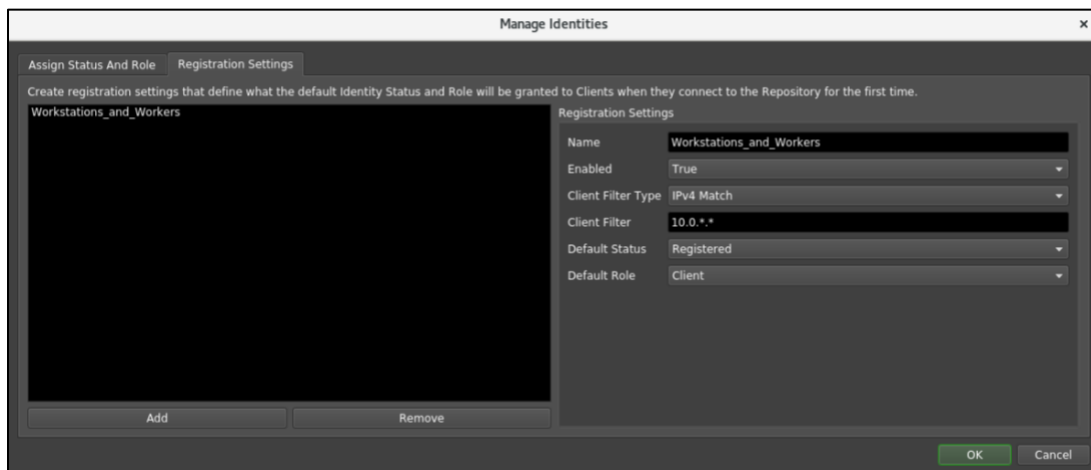


9. In the **Assign Status and Role** tab, you'll notice that your Render Scheduler is already assigned to the "server" role. Now we'll create a setting to automatically assign your workstations and farm workers to the "client" role.



10. Click the **Registration Settings** tab
11. Click **Add** and then enter the following information on the right side of the window under Registration Settings:
  - a. Set **Name** to **Workstations\_and\_Workers**
  - b. Set **Enabled** to **True**
  - c. Leave **Client Filter Type** set to **IPv4 Match**
  - d. Set **Client Filter** to **10.0.\*.\***

- e. Set **Default Status** to **Registered**
- f. Set **Default Role** to **Client**



12. Click **OK**

## Set Mapped Paths

Because we will be using Linux workers for rendering, we need to create some mapped pathing to make sure our file paths convert from Windows to Linux correctly. For example, a file located at `Z:\project\myshow\shot_01.ma` on your Windows machine will need to be mapped to `/mnt/studio/project/myshow/shot_01.ma`. Deadline can do this automatically when it's set up correctly.

- In the **Deadline Monitor**, choose **Tools** → **Configure Repository Options**
- Click **Mapped Paths**
- Under **Global Rules** click **Add**
  - For **Replace Path**, enter **Z:**
  - For **Windows Path**, enter **Z:**
  - For **Linux Path**, enter **/mnt/studio**
- Click **OK**
- Click **OK** again to close the window

## Set Up Render Groups

Now we want to create a render group. You can create multiple groups if you want, but for our case a single one will be fine.

- Go **Tools** → **Manage Groups...**
- Click **New**
- For **Group name** enter: **linux\_worker**
- Click **OK**
- Click **OK** again

## Test Your Setup

To test the setup, we're going to do a quick batch render test. We'll be creating a simple python file that will accept the frame number and print it to the render logs.

- Open a new Terminal by going to **Applications** → **Favorites** → **Terminal**
- In the terminal, create a python file called `test_render.py` with either **gedit** or **vi**:
  - If you're using **vi**:

```
vi ~/test_render.py
```

- If you're using **gedit**:

```
gedit ~/test_render.py
```

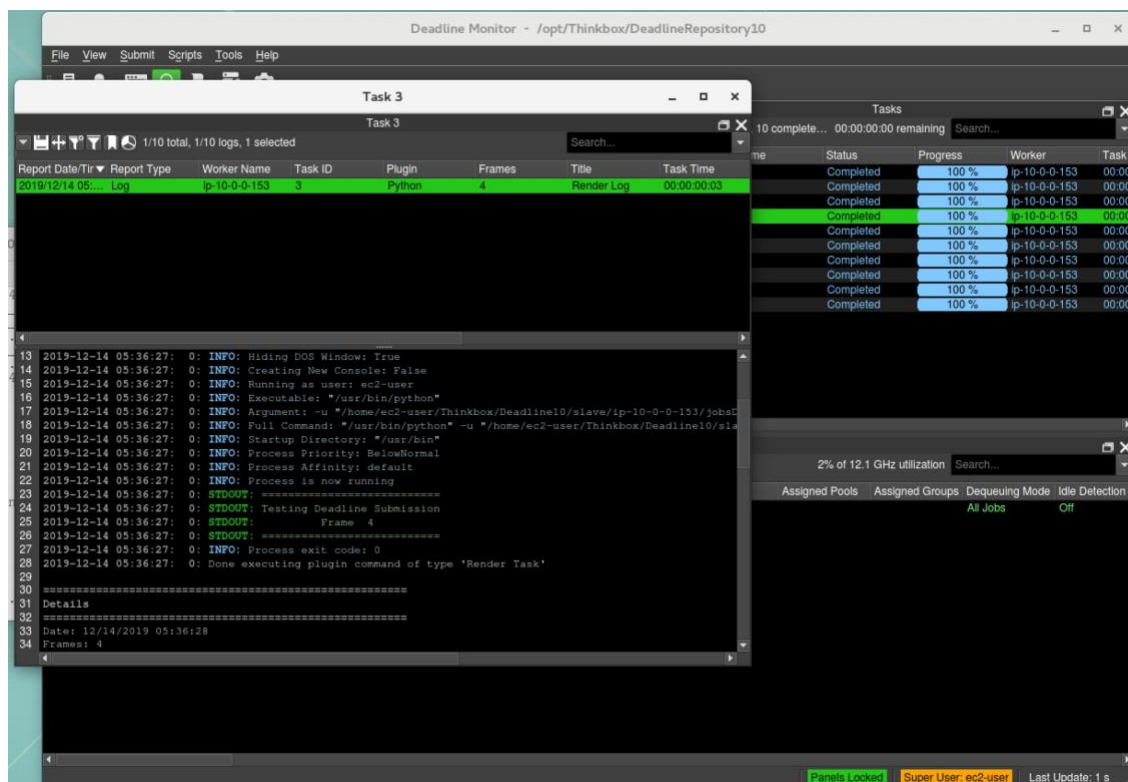
- Enter the following text:

```
1
2 import sys
3
4 print "=====
5 print "Testing Deadline Submission"
6 print "      Frame ", sys.argv[1]
7 print "=====
8
9
```

`test_render.py` – Note: <shift>+click the image to open the text in a new tab.

- **Save** your file and quit.

- Inside the Deadline Monitor, go **Submit** → **Miscellaneous** → **Python**
- Under **Job Name** enter **Test Render**
- Set the **Frame List** to **1-10**
- Set the **Python Script File** to **/root/test\_render.py**
- Set **Arguments** to **<STARTFRAME>** by clicking **Start Frame Tag**
- Make sure **Python Version** is **2.7**
- Click **Submit**
- Once the job has started picking up, you should be able to track your render in the monitor. Open a task and look at the log to see if it worked correctly.



## Make Sure Deadline Monitor, Deadline RCS & Deadline Pulse Always Run When Logging In

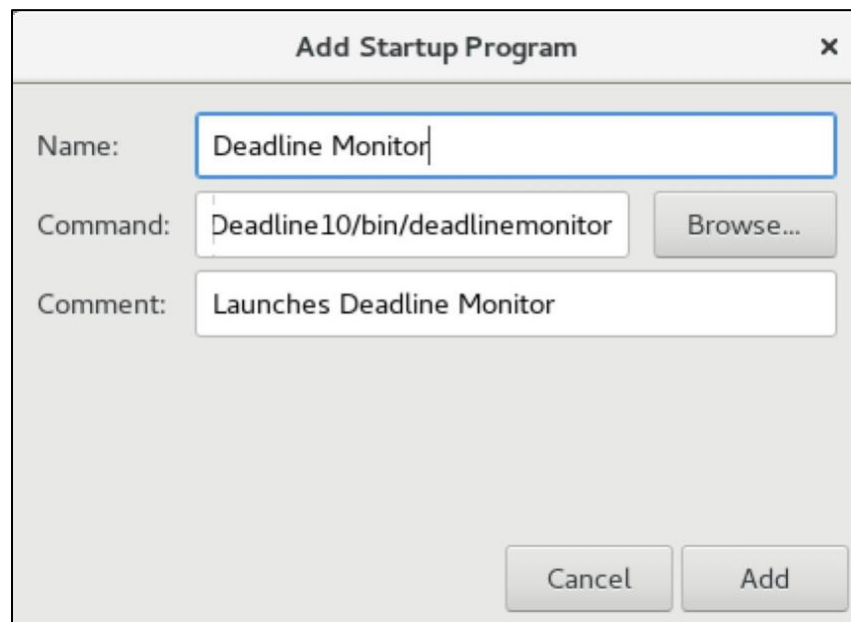
In order for the Render Scheduler to work effectively, it's important for these three tools to be running. In order to make sure they run every time you log into the Render

Scheduler as root, we can set them up to start automatically. This way you just need to login as root and the Render Scheduler will be ready to go.

- Open up the **Terminal** and run:

```
gnome-session-properties
```

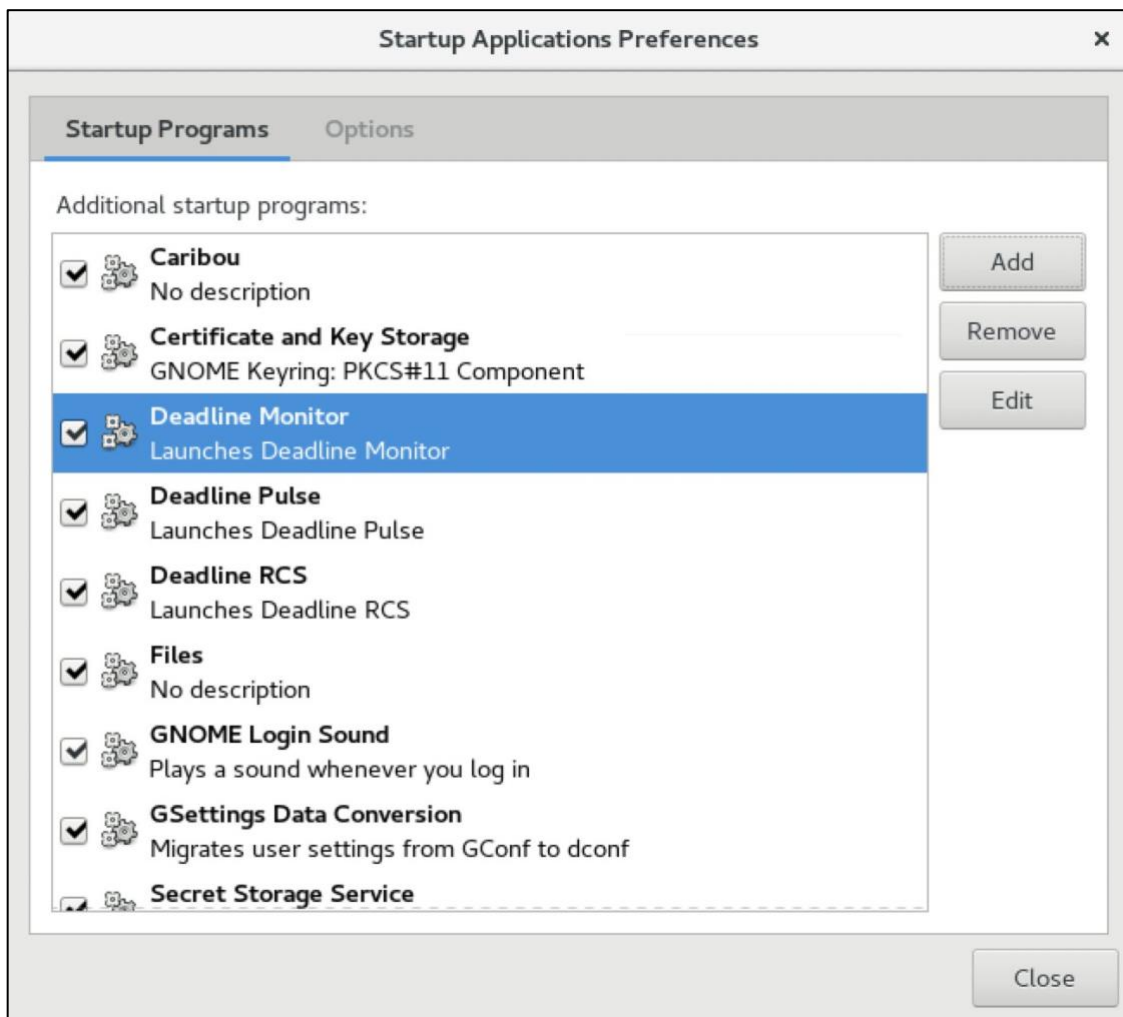
- This will allow you to add additional startup programs that will run every time you log in.
- Add **Deadline Monitor** by clicking **Add**
  - For **Name** enter **Deadline Monitor**
  - For **Command** enter **/opt/Thinkbox/Deadline10/bin/deadlinemonitor**
  - For **Comment** enter **Launches Deadline Monitor**



The screenshot shows a window titled "Add Startup Program" with a close button (X) in the top right corner. Inside the window, there are three labeled text input fields: "Name:" containing "Deadline Monitor", "Command:" containing "/opt/Thinkbox/Deadline10/bin/deadlinemonitor", and "Comment:" containing "Launches Deadline Monitor". To the right of the "Command:" field is a "Browse..." button. At the bottom right of the window are two buttons: "Cancel" and "Add".

- Click **Add**
- Add **Deadline Pulse** by clicking **Add**
  - For **Name** enter **Deadline Pulse**
  - For **Command** enter **/opt/Thinkbox/Deadline10/bin/deadlinepulse**
  - For **Comment** enter **Launches Deadline Pulse**

- Click **Add**
- Add **Deadline RCS** by clicking **Add**
  - For **Name** enter **Deadline RCS**
  - For **Command** enter **/opt/Thinkbox/Deadline10/bin/deadlinercs**
  - For **Comment** enter **Launches Deadline RCS**
  - Click **Add**



- Click **Close**
- You now have all three Deadline apps ready to load when you log in as root after a restart. Congratulations!

- Note: When you login to your Render Scheduler after a restart, you will see the Deadline Pulse and Monitor applications visible and running, but will not see a Terminal window running Deadline RCS. Don't worry, even though you can't see it, it is running in the background.

## Update Linux Worker

Now that our Render Scheduler is up and running on a Linux instance, we will need to update our workers to point to this new instance. This is a relatively quick fix, but we will need to update our AMI to point to the new Render Scheduler.

### Launch a Worker

- Go to **Services** → **EC2**
- Click **Launch Templates**
- Select the **My-Studio-Worker-LT**
- Choose **Actions** → **Launch instance from template**
- Under **Source template version**, make sure the most recent version is selected
- Under **Resource tags**, change the name to **Worker-Linux-new**
  - Note: We're changing the name here so that it's easy to identify our new Linux worker in case you still have any old workers around in your instance list from completing the previous tutorials. You'll definitely want to launch a new Linux worker from the template, rather than trying to reuse an old one, just to ensure that you have all the correct user data and other settings.
- Scroll down and click **Launch instance from template**

### Modify Worker deadline.ini File

- Once the worker is running and ready to connect, **Select** it and choose **Connect**
- Select **EC2 Instance Connect** and username **root** just like you did before with the Linux Render Scheduler.
- **Stop** the **deadlineworker** by typing



```
/opt/Thinkbox/Deadline10/bin/deadlineworker -shutdown
```

- When the **shutdown** is **complete**, modify the **deadline.ini** file. Again, if you need a refresher on vi commands, see this webpage: [Vi Cheat Sheet](#).

```
vi /var/lib/Thinkbox/Deadline10/deadline.ini
```

- You're going to be modifying the two lines that have **ProxyRoot** which point to the IP address for the Render Scheduler, and change the line for the **Certificate**.
- First go and get the private IP address for the new **Render Scheduler - Linux** instance by selecting it in the EC2 Instances console and looking in the **Description Tab**.
- Find a line that looks something like (note: your IP address will be different):

```
ProxyRoot=10.0.0.219:4433
```

and replace it with the **private IP address** for your new Linux Render Scheduler. Note: Make sure to use the private IP address and not the public IP address that you used when connecting with DCV

```
ProxyRoot=10.0.0.153:4433
```

- Now do the same for the line that has **ProxyRoot0**

```
ProxyRoot0=10.0.0.219:4433
```

becomes:

```
ProxyRoot0=10.0.0.153:4433
```

- Next change the path in **ProxySSLCertificate** to point to the new **DeadlineCertificates\_linux** path

```
ProxySSLCertificate=/mnt/studio/app_env/thinkbox/DeadlineCertificates/Deadline10RemoteClient.pfx
```

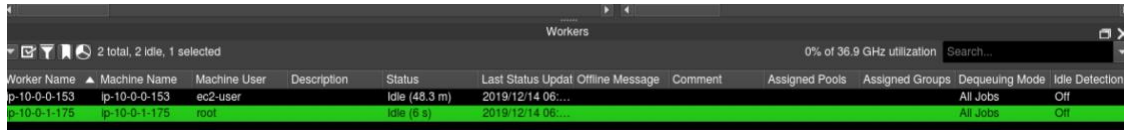
changes to:

```
ProxySSLCertificate=/mnt/studio/app_env/thinkbox/DeadlineCertificates_linux/certs/Deadline10RemoteClient.pfx
```

- Now quit and save by typing **:wq**
- **Restart** the **deadlineworker** to get these new changes

```
/opt/Thinkbox/Deadline10/bin/deadlineworker -nogui
```

- If you've set the values correctly, when you go check your Deadline Monitor in the new Linux Render Scheduler, your worker will now show up!



Worker Name	Machine Name	Machine User	Description	Status	Last Status Update	Offline Message	Comment	Assigned Pools	Assigned Groups	Dequeuing Mode	Idle Detection
p-10-0-0-153	ip-10-0-0-153	ec2-user		Idle (48.3 m)	2019/12/14 06:...					All Jobs	Off
p-10-0-1-175	ip-10-0-1-175	root		Idle (6 s)	2019/12/14 06:...					All Jobs	Off

## Create a New AMI for Your Worker

- Shut down your new Linux worker instance and create an Image of it.
  - Go to **Services**→**EC2**
  - Click **Instances**
  - Right click your **Worker-Linux-new** instance and choose **Stop instance**
- Once stopped, right click it again and choose **Images and templates** → **Create image**
- Enter a name for your image (e.g., My-Studio-Worker-v02-AMI) and a description for your image
- Check that the **size** of your volume is set to **300 GiB**
- Click **Create image**
- In the navigation pane on the left, under Images, select **AMIs**.
- Find the name of your Linux Worker AMI in the list and note the AMI ID as well. *Write down both of these in your cheat sheet. Because not everyone will be adding a Linux Render Scheduler, there's not an existing section on the sheet to enter this information, but you can use the Notes section at the bottom to enter it on your own.*
- Add appropriate **Name** and **Studio** tags.
- Wait a few minutes while the AMI is pending.

## Modify Worker Launch Template

- Now we need to update the Launch Template to use this new AMI.
- Go to **Services** → **EC2**
- Click **Launch Templates**
- Choose the **My-Studio-Worker-LT**
- Choose **Actions** → **Modify template**
- Update the description to something like “Updated worker to point to Render Scheduler - Linux”
- Update the **Amazon Machine Image** to point to the **new AMI** (My-Studio-Worker-v02-AMI)
  - If you see a popup that says some of your settings will be changed, click **Confirm Changes**
- Under **Storage (volumes)** change **Volume type** to **Don't include in template**.
- Open **Advanced Details**
- Look at the **EBS-optimized instance** setting. Make sure it is set to **Don't include in launch template**. Even if it is already set to that, open the drop down menu and select it again.
- Scroll down and click **Create template version**
- Go back to your list of **Launch Templates**
- Choose **My-Studio-Worker-LT**
- Notice that the **Default Version** is **1**, but the **Latest Version** is **2**.
- Choose **Actions** → **Set default version**
- Set the version to **2** and click **Set as default version**
- Now the latest version of the template will use version 2.

## Creating Your Render Fleet

To allow Deadline to automatically spin up Linux workers for us we need to create a spot request.

- In the **EC2 Console**, click **Spot Requests** on the left hand bar

- Click **Request Spot Instances**
- Switch to **Flexible workloads**

- Select your updated Linux Worker Launch Template (e.g., My-Studio-Worker\_LT) in the **Launch Template** bar, and make sure it is the correct version.

- Most of the settings can be left as default.
- Scroll down and select your **Total Target Capacity**. Set it to whatever you want: 1, 5, 10, etc.
- Select the checkbox next to **Maintain target capacity**
- Once you are ready to go **DO NOT CLICK Launch**.
- Scroll down to the bottom and look in the left corner.
- Click **JSON config** and this will download a file called **config.json**.

- We will modify this configuration file to tell Deadline what instances to launch when someone submits a render for a given group.

## Attaching Render Fleet

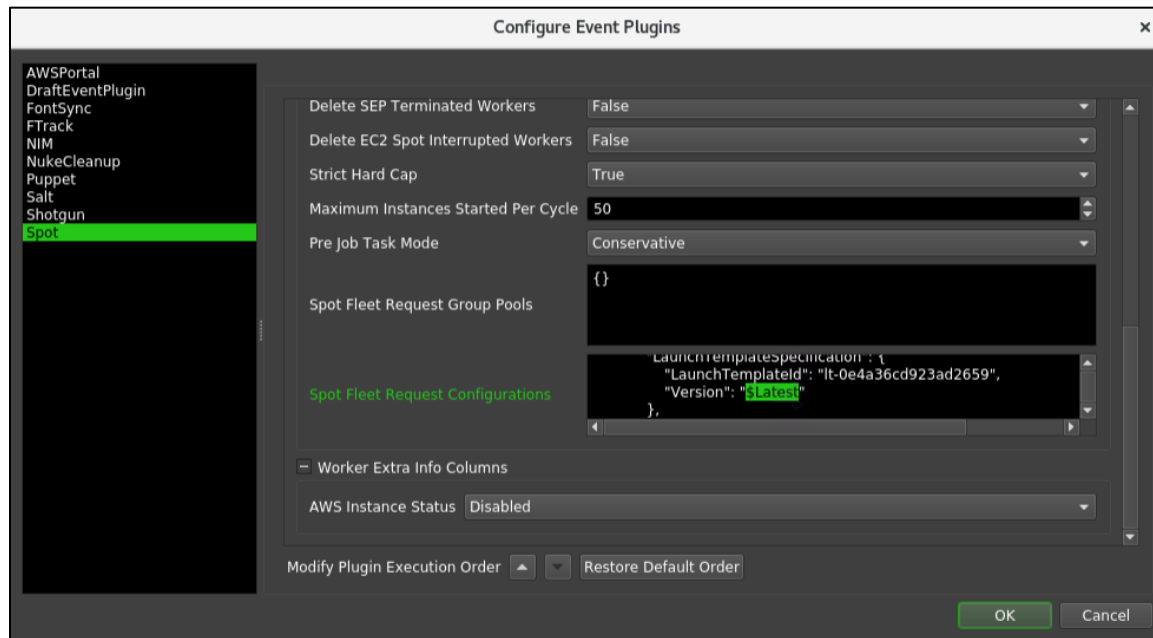
- Switch over to your **Render Scheduler - Linux** instance in the DCV client
- Go **Tools**→ **Configure Events**
  - If you don't see Configure Events, make sure you **Enable Super User** mode in the **Tools** menu
- Click **Spot**
  - Make sure to set the **State** to **Global Enabled**
  - Enter the **access key ID** and **secret access key** for your IAM user. If you created a new IAM user in Tutorial 1 you can find the access key ID and secret access key in the credentials.csv you downloaded at that time. *You can find the location of your credentials.csv on the cheat sheet.*
  - If you didn't setup the IAM user for your account or do not have the credentials.csv handy, you can create a new access key using these directions: [Where's My Secret Access Key?](#)
  - Make sure the **Region** is set correctly
  - Inside the **Spot Fleet Request Configurations** you'll paste the contents of your **config.json**, but be sure to add it with the following line in front of it:

```
{"linux_worker":
```

- And this line after:

```
}
```

- Also in the **Spot Fleet Request Configurations** field, find the line that refers to the version number of your Linux worker launch template. It is located under the LaunchTemplateId and should currently be set to 1.
- Change 2 to **\$Latest**. Using the value of \$Latest will ensure that the Spot Fleet Request Configuration is always pointing to the latest version of your template, in case you need to update it later.



- When you're all done, the whole thing should look something like:

```
{
  "linux_worker": {
    "IamFleetRole": "arn:aws:iam::86847329516:role/aws-ec2-spot-fleet-tagging-role",
    "AllocationStrategy": "capacityOptimized",
    "TargetCapacity": 1,
    "ValidFrom": "2021-07-22T19:24:56Z",
    "ValidUntil": "2022-07-22T19:24:56Z",
    "TerminateInstancesWithExpiration": true,
    "LaunchSpecifications": [],
    "Type": "maintain",
    "LaunchTemplateConfigs": [
      {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-0e4a36cd923ad2649",
          "Version": "$Latest"
        },
        "Overrides": [
          {
            "InstanceType": "m5.2xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5ad.2xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5d.2xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.4xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.8xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.12xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.16xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.metal",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          },
          {
            "InstanceType": "m5.24xlarge",
            "WeightedCapacity": 1,
            "SubnetId": "subnet-0a93a4e620416d2f4"
          }
        ]
      }
    ]
  }
}
```

Example config.json

- Click **OK**

- Run House Cleaning by going to **Tools** → **Perform House Cleaning**
  - This will help force the changes you've made to the Spot Event plugin to get recognized.

## Test the Spot Event Plugin

Now it's time to test the spot event plugin. To do this, we'll move the test\_render.py script to a location that one of the workers can find it, and then run another test. This time we'll also specify the linux\_worker group to see if we can get some workers to spin up.

- Open up a **Terminal** on your Linux Render Scheduler
- Move the test\_render.py file to /mnt/studio

```
mv /root/test_render.py /mnt/studio/test_render.py
```

- In **Deadline Monitor** choose **Submit** → **Miscellaneous** → **Python**
- Set the **Job Name** to **Test Spot**
- Set **Group** to **linux\_worker**
- Set the **Frame List** to **1-10**
- Set **Python Script File** to **/mnt/studio/test\_render.py**
- Click **Submit**
- Now wait a few minutes for the Spot Event Plugin to start working.
- If you want to check on the Spot request, go to **Services** → **EC2** and then click **Spot Requests**. A Fleet Request should appear, launching render workers.
  - Note: It can take 5 minutes for the spot event plugin to start for the first time. If you don't see any requests and you are confident that it should be working, just be patient. It might take a while.

Request Spot Instances

Actions

Pricing History

Savings Summary

Request type: all

State: all

Search by keyword

<<

>>

Viewing 1 to 2 of 2 requests

<input type="checkbox"/>	Request Id	Request type	Instance type	State	Capacity	Status	Persistence	Created	Max price
<input type="checkbox"/>	sir-73zif6jk	instance	m3.2xlarge	<div><div></div>active</div>	i-04142de48d17...	fulfilled	persistent	a minute ago	\$0.616
<input type="checkbox"/>	<div>sfr-41abd348-d5dd...</div>	fleet	m5.2xlarge,m3...	<div><div></div>active</div>	0 of 1	pending_fulfill...	maintain	a minute ago	



- If the Spot request is not there double check that you set the correct region
- If the Spot request is there, but no workers are spinning up, you can check for errors by clicking **pending fulfillment** on the Spot request status, and going to the **History** tab. It should give you a good idea of any issues you may encounter.

Request ID: sfr-41abd348-d5dd-42df-8dea-cc782aca343f

Description Instances **History** Savings Auto Scaling Scheduled Scaling

History saved for 48 hours only. Updates may take up to 1 minute to display.

Timestamp	Event Type	Status	Description	Instance Id
12/11/2019, 2:58:06 PM	fleetRequestChange	progress	m3.2xlarge, ami-0ac06a88c1b5262ac, Linux/UNIX, us-west-1c, capacityUnitsRequested: 1.0, totalCapacityUnitsRequested: 1.0, totalCapacityUnitsFulfilled: 1.0, targetCapacity: 1	
12/11/2019, 2:58:06 PM	InstanceChange	launched	{"instanceType": "m3.2xlarge", "image": "ami-0ac06a88c1b5262ac", "productDescription": "Linux/UNIX", "availabilityZone": "us-west-1c"}	i-04142de48d17d432d
12/11/2019, 2:58:04 PM	fleetRequestChange	active		
12/11/2019, 2:57:54 PM	fleetRequestChange	submitted		

- Once the status of the spot request has changed from pending fulfillment to **fulfilled**, you should see a new worker pop up in the Worker list in the Deadline Monitor and your frames should start rendering.

The screenshot shows the Deadline Monitor application window titled "Deadline Monitor - /opt/Thinkbox/DeadlineRepository10". The interface is divided into three main panels:

- Jobs Panel:** Displays a table with columns: Job Name, User, Errors, Comment, Department, Task Progress, and Status. It shows three jobs: "Test Spot", "Test Re...", and "Untitled", all with 0 errors and 100% progress.
- Tasks Panel:** Displays a table with columns: Task ID, Frame, Status, Progress, Worker, and Task. It shows 10 tasks, all completed with 100% progress.
- Workers Panel:** Displays a table with columns: Worker Name, Machine Name, Machine User, Description, Status, Last Status Update, Offline Message, Comment, Assigned Pools, Assigned Groups, Dequeuing Mode, and Idle Detection. It shows 12 workers, with 11 idle and 1 stalled.

At the bottom of the window, there is a status bar indicating "Panels Locked", "Super User: ec2-user", and "Last Update: 1 s".

## Update Workstation

We're almost done! The next step is to update our Workstation AMI to point to the new Render Scheduler just like we did with the workers.

### Launch a Workstation

- Go to **Services** → **EC2**
- Click **Launch Templates**
- Select the **My-Studio-Workstation-LT**
- Choose **Actions** → **Launch instance from template**
- Under **Resource tags**, change the name to **Workstation\_Win\_new**
  - Note: As with the Linux worker, we're adding "\_new" to the end of the name for this instance so that we can tell it apart from any other Windows workstations we might currently have in our instance list.

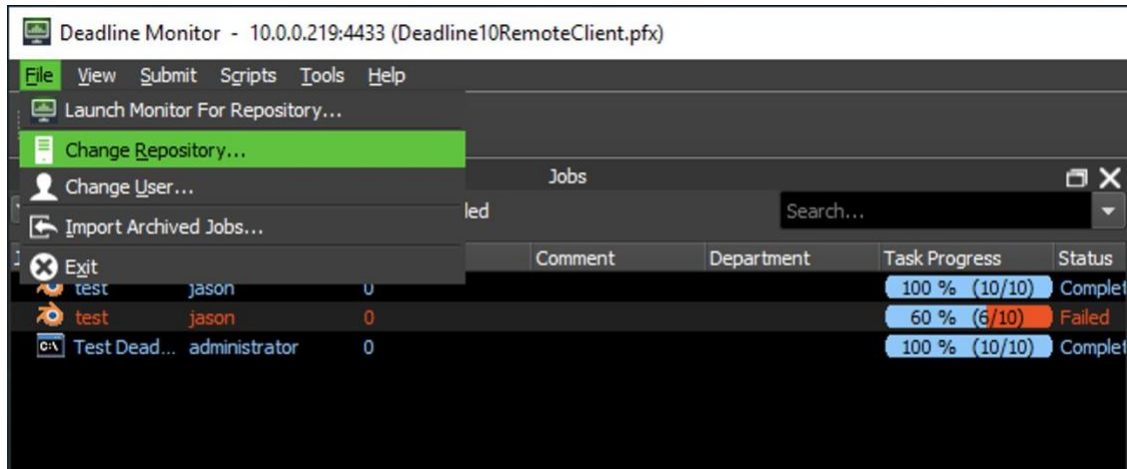
- Scroll down and click **Launch instance from template**

## Connect to the Studio FSx Drive

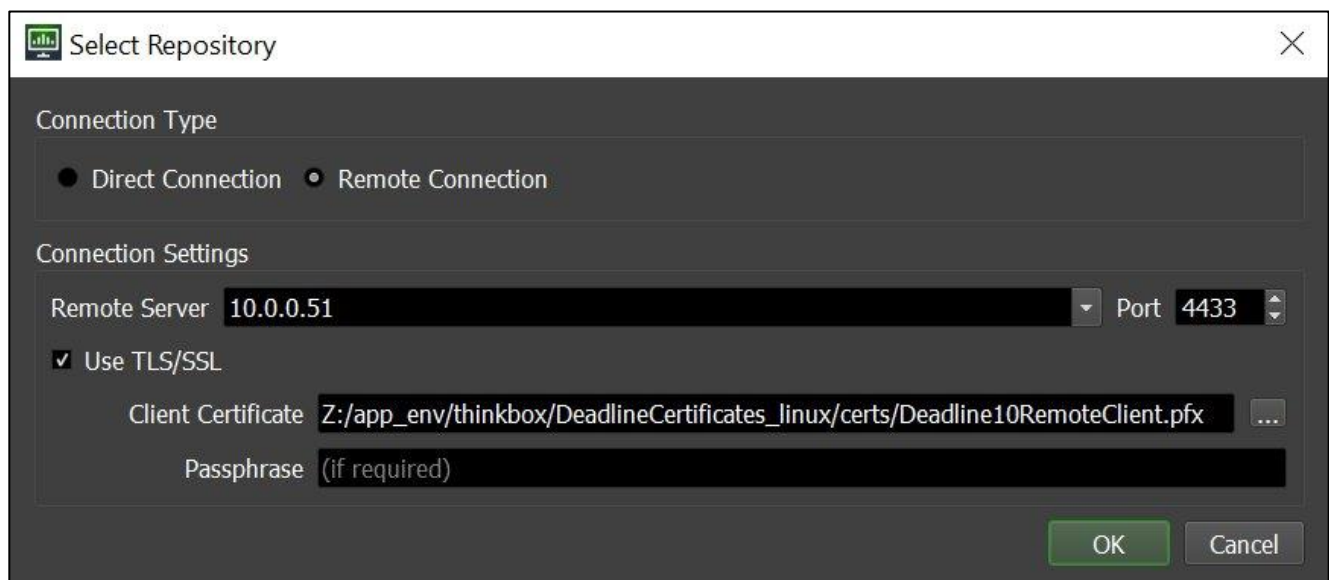
- Once the workstation is running and ready to connect, select it and choose **Connect**
- Log in as **Administrator** (NOT mystudio/Admin) because we'll be modifying the ini file and running SysPrep.
- Open a **File Explorer** window.
- From the navigation pane, right-click **This PC** and choose **Map Network Drive...**
- Choose a drive letter of your choice for **Drive** (e.g., **Z:**)
- Enter the full CNAME Alias for your file share that you noted above for **Folder**.
  - e.g., \\studio.mystudio.com\share
  - click **Finish**
- Use your Active Directory Admin login credentials. For example:
  - Username: Admin
  - Password: Your Admin Password

## Modify Workstation Repository

- Open the Monitor by going **Start** → **Thinkbox** → **Deadline Monitor**
- When it opens, it will be pointing at the old repository.
- Choose **File** → **Change Repository...**



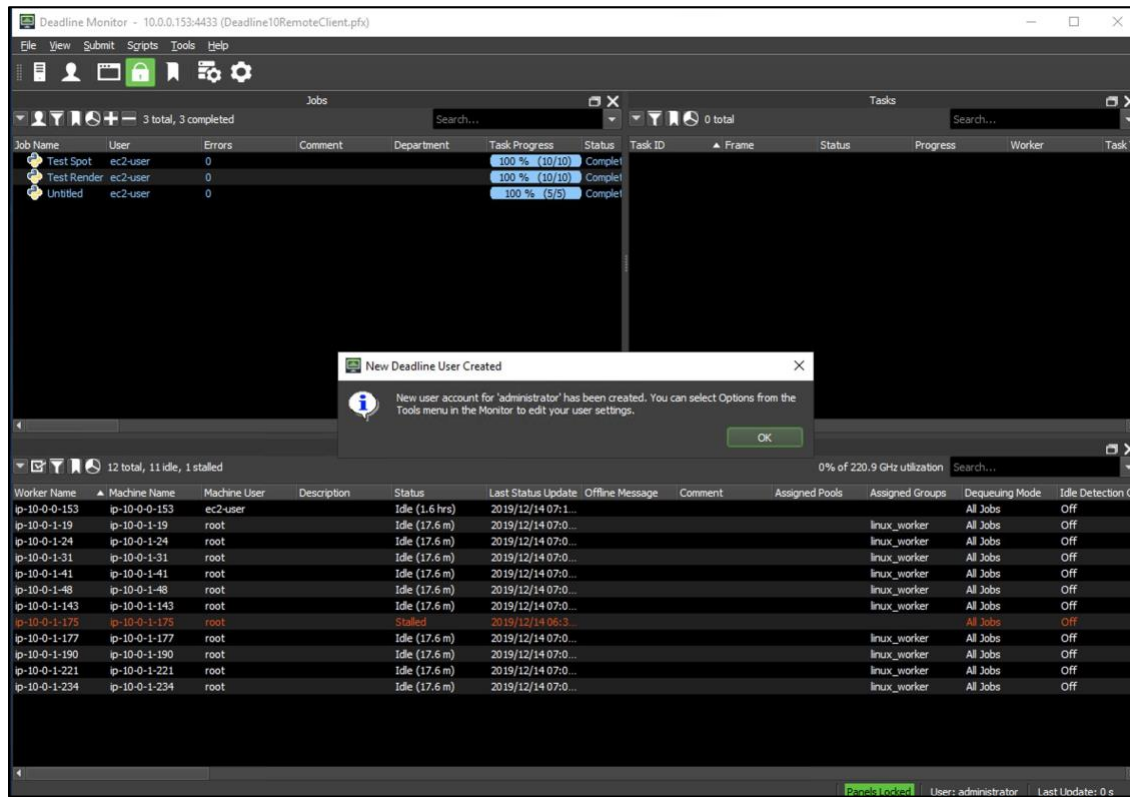
- Change the **Remote Server** to the **private IP address** of your new Render Scheduler - Linux
- Modify the **Client Certificate** to point to the **new Linux certificates path** (e.g., Z:/app\_env/thinkbox/DeadlineCertificates\_linux/certs/Deadline10RemoteClient.pfx)



x)

- Click **OK**
- When it asks if you want to Change the repository, check **Save Answer** and click **Yes**

- After a few seconds, the Monitor will switch over to the new repository!

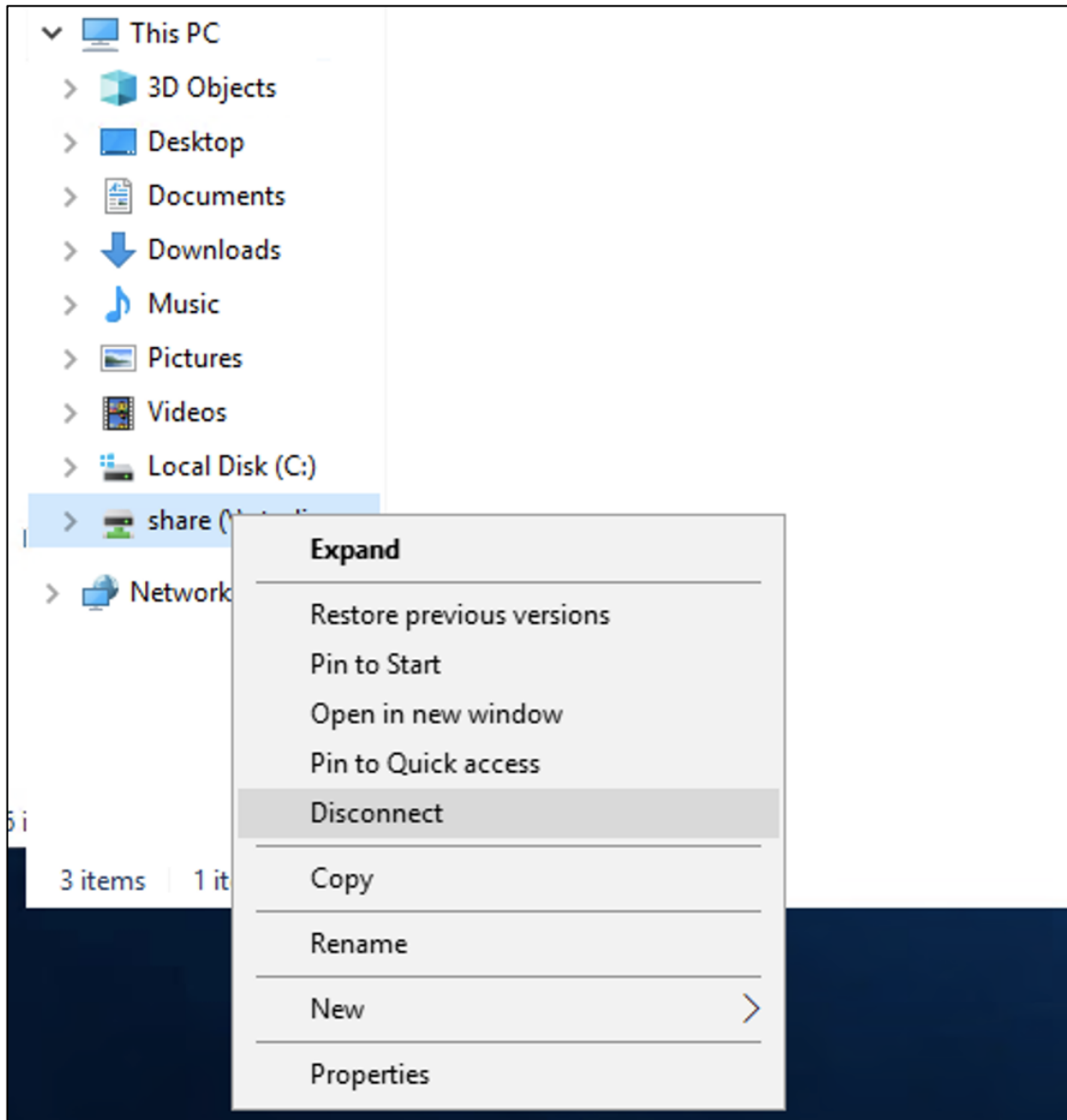


- Quit out of Deadline Monitor

## Preparing Your Instance to Create a New AMI

Now that you've updated your Deadline settings, we'll need to make a new AMI with these settings, and update our Launch Template.

- Disconnect from your FSx Drive (Z:)
  - Open the **File Explorer**
  - Under **This PC** on the left, right-click the Z: drive and choose **Disconnect**



- Go to the start menu, type **Ec2LaunchSettings** and launch it
- Make sure **Set Computer Name** is selected
  - This will ensure each instance you create has a unique name.
- Check that **Administrator Password** is set to **Specify** and input the Administrator Password for your Active Directory(e.g. password for mystudio\Admin)
- Select **Run EC2Launch on every boot**

- Click **Shutdown with Sysprep**

**Ec2 Launch Settings**

**General**

**Set Computer Name**

- ☒ Set the computer name of the instance ip-<hex internal IP>. Disable this feature to persist your own computer name setting.

**Set Wallpaper**

- ☒ Overlay instance information on the current wallpaper.

**Extend Boot Volume**

- ☒ Extend OS partition to consume free space for boot volume.

**Add DNS Suffix List**

- ☒ Add DNS suffix list to allow DNS resolution of servers running in EC2 without providing the fully qualified domain name.

**Handle User Data**

- ☒ Execute user data provided at instance launch. Note: This will be re-enabled when running shutdown with sysprep below.

**Administrator Password**

- ☐ Random (Retrieve from console)
- ☒ Specify (Temporarily store in config file)
- ☐ Do Nothing (Customize Unattend.xml for sysprep)

These changes will take effect on next boot if Ec2Launch script is scheduled. By default, it is scheduled by shutdown options below.

**Sysprep**

Sysprep is a Microsoft tool that prepares an image for multiple launches.

Ec2Launch Script Location: **Found**

C:\ProgramData\Amazon\EC2-Windows\Launch\Scripts\InitializeInsta

- ☒ Run EC2Launch on every boot (instead of just the next boot).

**Shutdown without Sysprep** **Shutdown with Sysprep**

**Ok** **Cancel** **Apply**

- Click **Yes**
- This will shut down your instance after a few processes run. It can take a few minutes, so feel free to grab a coffee (or tea).

## Create Workstation AMI

- Navigate back to the **EC2 Dashboard** and find the Workstation that has just been shut down, if it is not completely stopped yet, wait for that process to finish

<input type="checkbox"/>	Render Scheduler - Linux	i-0934cfbf5bf3a0ee1	<span>Running</span>		m5.xlarge	<span>2/2 checks passed</span>	No alarms
<input checked="" type="checkbox"/>	Workstation_Win_new	i-0192b245cc6b7691a	<span>Stopped</span>		g4dn.4xlarge	–	No alarms

- Right-click the instance and choose **Images and templates** → **Create image**
- Give it an appropriate name (e.g., My-Studio-Workstation-v02-AMI) *Record the AMI name in the Notes section on your cheat sheet.*
- Give your workstation AMI a description if you want.
- Increase its storage if necessary.

### Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID  
i-0192b245cc6b7691a (Workstation\_Win\_new)

Image name  
  
Maximum 127 characters. Can't be modified after creation.

Image description - optional  
  
Maximum 255 characters

No reboot  
☐ Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/s...	Create new snapshot fr...	150	EBS General Purpose SS...	100		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

☒ Tag image and snapshots together  
Tag the image and the snapshots with the same tag.
 ☐ Tag image and snapshots separately  
Tag the image and the snapshots with different tags.

No tags associated with the resource.

You can add 50 more tags.

- Click **Create image**



- AMI creation can also take 5 to 10 minutes to complete. How about more coffee?
  - To see if your AMI is ready click **AMIs** in the left panel. The AMI will need to finish creating before you can create a Launch Template with it.
  - While you're waiting, you can also add Tags to your AMI. Again, we recommend creating at least a **Studio** tag and **Name** tag.
  - *Now is a good time to note the AMI ID in the Notes section of the cheat sheet. You can find the ID by selecting the AMI from the list and looking at the top left of the Details tab.*

## Modify Workstation Launch Template

- Now we need to update the Launch Template to use this new AMI.
- Go to **Services** → **EC2**
- Click **Launch Templates**
- Choose the **My-Studio-Workstation-LT**
- Choose **Actions** → **Modify template**
- Update the description to something like “Updated workstation to point to Render Scheduler - Linux”
- Update the **Amazon Machine Image** to point to the **new AMI** (e.g., My-Studio-Workstation-v02-AMI)
- Scroll down and click **Create template version**
- Go back to your list of **Launch Templates**
- Choose **My-Studio-Workstation-LT**
- Notice that the **Default Version** is 1, but the **Latest Version** is 2.
- Choose **Actions** → **Set default version**
- Set the **Template version** to 2 and click **Set as default version**
- Now the default version of the template will be version 2.

## Test Workstation

- Use the new Launch Template to launch a new Workstation.

- Once it's up, connect and log in as one of your user accounts (e.g., mystudio\jason)
- Make sure that when you run the Deadline Monitor you're connecting to the new Render Scheduler - Linux instance. Note: You can tell that you're connected to your new Render Scheduler by looking at the list of jobs. If you see your "Test Spot" job from earlier, then you should be good to go.
- Go ahead and launch a new render from Blender and make sure it works correctly. If all goes as planned - you're now using your new Linux-based Render Scheduler!

## Shut Down Old Render Scheduler

- You can go ahead and terminate your old Render Scheduler if you aren't going to use it anymore.
- 

## Appendix

### Links to AWS Documentation

- [NICE DCV Clients](#)
- [AWS Direct Connect](#)
- [AWS VPN](#)
- [VPN Connections to AWS VPC](#)
- [Managing NICE DCV Sessions](#)
- [Deadline Secrets Management](#)
- [Getting Started With Deadline Secrets Management](#)

### Links to Other Resources

- [Vi Cheat Sheet](#)

# Tutorial 10: Add-On - Using Linux and Teradici for a Workstation

*Estimated Time to Complete: 1 hour, 30 minutes*

## Review of Studio in the Cloud

The first 7 of our Studio in the Cloud tutorials helped you create a fully cloud-based studio that leverages the power, scale and convenience of AWS. This basic setup introduced you to virtual workstations, cloud storage and cloud rendering and how they can be combined to move creative work into the cloud. In these add-on tutorials, we have been providing instructions for different ways you can leverage the flexibility and depth of AWS to customize your studio in different ways.

## Overview of this Tutorial

In our first add-on tutorial, [Tutorial 8: Upgrading to Teradici for Desktop Streaming](#), we showed you how to add desktop streaming with Teradici to your existing Windows-based artist workstations. This tutorial will show you how to create a Linux-based artist workstation and connect to it using [Teradici Cloud Access Software](#). We'll install necessary applications, including Blender and Deadline, and create an AMI (Amazon Machine Image) and launch template that you can use to start up as many additional Linux workstations as you need.

## Teradici Cloud Access Software

Because you cannot connect to a Linux instance using Remote Desktop, we will be using Teradici Cloud Access Software instead. Similar to Remote Desktop, Teradici streams pixels and audio from a virtual machine in the cloud to your local desktop and sends keyboard and mouse information back. However, it also has a few advantages over Remote Desktop:

- High-performance remote visualization with true 4:4:4 color accuracy and lossless image quality
- Delivers graphics intensive workloads to any endpoint (including PCoIP clients for Windows, Mac, Chrome OS, Android, iOS and zero clients)
- Highly performant across variable network conditions

## Linux vs. Windows Workstations

If you already completed our other add-on tutorial, [Tutorial 9: Using a Linux Instance for a Render Scheduler](#), you know that one of the motivations behind using Linux over Windows is cost savings. Your artist workstations will be running whenever your artists are working, so switching them to Linux can help cut down on those hourly costs. For the same type of instance, the hourly cost of just the instance by itself can be up to 50% cheaper for Linux vs. Windows.

In Tutorial 8, where we added Teradici to your existing Windows-workstations, we walked you through installing Teradici manually. We did this so that you would not have to re-install any software from previous tutorials. However, in this case, since we'll be switching from Windows to Linux, we'll need to re-install software anyway. As a result, for this tutorial we'll be using the [Teradici Linux AMI from the AWS Marketplace](#). Since this AMI already has Teradici installed, it will save us setup time so that you and your artists can get working more quickly. Instead of requiring a separate license, the cost of using Teradici is built into the hourly pricing for the AMI. You pay an extra \$0.50/hour on top of the normal hourly price for your instance.

Even taking the added hourly cost of the Teradici AMI into account, you should still see a cost savings of at least 10% by switching your Windows workstations to Linux workstations with Teradici. Rather than pay by the hour for Teradici software, it is also possible to purchase an annual subscription. For more information, see the [Teradici Linux AMI Subscription page](#).

## Prerequisites

### Complete the Previous Tutorials

This add-on tutorial assumes that you've already completed at least Tutorials 1-4 in our previous Studio in the Cloud tutorial series. If you are new to Studio in the Cloud, you can find the first tutorial here: [Tutorial 1: Getting Started with AWS Virtual Workstations](#). If you have already completed the Studio in the Cloud tutorials, then you're all set to continue with the steps below.

### Check Your G Instance Quota

You likely already checked your G instance quota at least once when you started the Studio in the Cloud tutorials. You may even have needed to submit a quota increase request when you started Tutorial 1 and another when you started launching more Windows G4 instances for your artists in Tutorial 7. In any case, it's a good idea to

check your quota again, to make sure that you have enough quota to cover the new G4 instance you will be launching in this tutorial. For instructions on checking your quota value and requesting an increase, see the instructions in the [Appendix for Tutorial 1](#).

Note: If you currently have any G4 instances currently running, you will need to make sure that your **Applied quota value** is at least:

$(16 \times (\text{total number of G4s running} + 1))$

Each of the g4dn.4xlarge instances that are used for workstations uses 16 vCPUs of quota, so you want to make sure your quota covers all the G4s you have running, plus one more for this tutorial. For example, if you currently have 2 G4s running, your Applied quota value will need to be at least  $(16 \times (2 + 1))$  or 48, in order to complete this tutorial.

Quota increase requests can take up to 48 hours to process, so if you need to submit an increase request, it's best to do it as soon as possible.

## Make Sure Your Render Scheduler Is Running

We'll be running a test at the end of this tutorial to make sure that we can submit a render job to your Render Scheduler from your new Linux workstation. If you've been using your Studio in the Cloud, then most likely your Render Scheduler instance is already running and ready to go. However, if you stopped it, navigate to the EC2 dashboard and start it now.

### If you have a Windows Render Scheduler from Tutorial 4:

- Make sure to login as **Administrator** and start up the **Deadline Monitor**, **Deadline Remote Connection Server (RCS)**, and **Deadline Pulse**.

### If you have a Linux Render Scheduler from Tutorial 9:

- Login as **root** and verify that **Deadline Pulse** and **Deadline Monitor** are running. Deadline RCS should automatically be running in the background, not visible to you.

## Create a DHCP Options Set

If you completed Tutorial 6, then you already created a DHCP options set that enables a Linux instance to connect to the Directory Service for user authentication. However, if you haven't completed Tutorial 6, then you will need to create a DHCP options set

before continuing. Follow the [instructions in Tutorial 6](#) to **Create DHCP Settings** and **Attach Your VPC to the DHCP Options Set**.

## Add Marketplace Permissions to Your User, If Needed

Since we'll be using a Teradici AMI from the AWS Marketplace as the starting point for our Linux workstation, we'll need to make sure that your user has permissions to use Marketplace AMIs. If you completed Tutorial 9: Using a Linux Instance for a Render Scheduler, you should already have the necessary permissions. If not, then follow the instructions for [Add marketplace permissions to your user](#) instructions to add them now.

## Download the Teradici Client

You may have already done this in Tutorial 8, if not follow the instructions at the bottom of page 8 of Tutorial 8: [Download client to your local machine](#).


## Locate Your Important Information Cheat Sheet

If you completed the other tutorials, then you probably already filled out the [Important Information Cheat Sheet](#). We'll be referring back to some of this information in this tutorial, so you should locate your filled out cheat sheet. If you never filled out the cheat sheet, you should download it now as we'll be entering some new notes on it during this tutorial. Note: Because the add-on tutorials are optional, you will be entering any information from this tutorial into the "Notes" section at the bottom of the cheat sheet.

## Launch an Instance with the Teradici Marketplace AMI

Once you've completed all the prerequisites above, you're ready to launch an instance with the Marketplace AMI.

- From the AWS Console go to **Services** → **EC2** and click **Instances (running)** to see a list of your running instances
- Click **Launch Instance**
- For **Step 1: Choose an Amazon Machine Image (AMI)**, enter **Teradici** in the search box and hit <enter>
- On the left, below the search box, click **AWS Marketplace** to list only AMIs that exist in the AWS Marketplace
- In the list, find the entry for **Teradici CAS for CentOS 7** and click **Select** next to it



**Teradici CAS for CentOS 7 (NVIDIA)**

★★★★★ (1) | 21.03.4 [Previous versions](#) | By [Teradici](#)

Starting from \$0.50/hr or from \$240.00/yr (95% savings) for software + AWS usage fees

Linux/Unix, CentOS 7.9, NVIDIA GRID 12.0 (460.32.03) | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 6/9/21

Teradici CAS delivers a secure, high-definition, uncompromising experience for remote desktops and workstations on Amazon G4 using the PCoIP protocol. This AMI is supports CentOS with an AMD GPU.

[More info](#)

Select

- A pricing list for the various instance types will come up, showing you what the costs will be.
- This page also has links to more information about the Teradici software as well as usage instructions, but don't worry, we'll walk you through all the steps below!
- Click **Continue**
- For **Step 2: Choose an Instance Type**, open the **All instance families** drop down menu and select **g4dn**

### Step 2: Choose an Instance Type


Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run a variety of applications. Choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: **All instance families** **Current generation** [Show/Hide Columns](#)

Currently selected	All instance families	c5n	i2	m5n	r5a	z1d
t2	c6g	i3	m5zn	r5ad		
t3	c6gd	i3en	m6g	r5b		
t3a	c6gn	inf1	m6gd	r5d		
t4g	d2	m4	mac1	r5dn		
a1	d3	m5	p2	r5n		
c4	g3	m5a	p3	r6g		
c5	g3s	m5ad	r3	r6gd		
c5a	<b>g4dn</b>	m5d	r4	x1		
c5d	h1	m5dn	r5	x1e		

- Next, choose a **g4dn.4xlarge** instance type and click **Next: Configure Instance Details**
- For **Step 3: Configure Instance Details** set the following:
  - Set **Network** to your VPC from the other tutorials (e.g., My-Studio-VPC)
  - Set **Subnet** to **Public Subnet A**
  - Set **Auto-assign Public IP** to **Enable**
  - Set **IAM role** to **EC2DomainJoin**
  - Click **Next: Add Storage**

- For **Step 4: Add Storage:**
  - Set storage to **150 GiB**
  - Click **Next: Add Tags**
- For **Step 5: Add Tags:**
  - Add a tag with Key = **Name** and Value = **Workstation\_Linux**
  - Add a tag with Key = **Studio** and Value = **<name of your studio>** (e.g., My-Studio)
  - Click **Next: Configure Security Group**
- For **Step 6: Configure Security Group:**
  - This page is already pre-populated with a security group that was generated by the AWS Marketplace for Teradici Linux connections, so we're going to re-use that
  - Leave **Create a new security group** selected
  - Change the **Security group name** to **<name of your studio>-Linux-Teradici-SG** (e.g., My-Studio-Linux-Teradici-SG)
  - Leave the **Description** as-is
  - You'll see a warning at the bottom of the screen recommending that you limit access to known IP addresses

 **Note:** As in earlier tutorials, we are initially opening up this security group to inbound traffic from any IP address. However, if you limited your source IP addresses during your Studio in the Cloud setup, you'll want to do the same here. If you will be accessing your instances over the public Internet, then after initial testing we recommend that you look into an [AWS Direct Connect](#) connection and/or VPN so that your connection will be private and secure. You may use third party VPN software, but AWS also provides a robust set of VPN solutions called [AWS VPN](#). More information about how to connect to your VPC using VPN can be found [here](#).

- Click **Review and Launch**
- Click **Launch** to launch your instance



- Choose the **key pair** that you created when you originally set up your Studio in the Cloud and then click **Launch Instances**. *If you don't remember the name of the key pair you created, you can find it at the bottom of the Tutorial 1 section of the [Important Information Cheat Sheet](#).*

## Add the Deadline Security Group to the Instance

We already opened the ports that we need to connect to our new Linux instance with Teradici, but before we continue, we should also add the Deadline security group that we created in Tutorial 4 as well. We'll need that in order to setup Deadline on our Linux workstation later in this tutorial.

- Go to **Services** → **EC2** and click **Instances (running)** to see a list of your running instances.
- Select **Workstation\_Linux**
- Choose **Actions** → **Security** → **Change security groups**
- Add **My-Studio-Deadline-SG**

**Associated security groups**  
Add one or more security groups to the network interface. You can also remove security groups.

Security groups associated with the network interface (eni-02de62ccd22edc4fd)

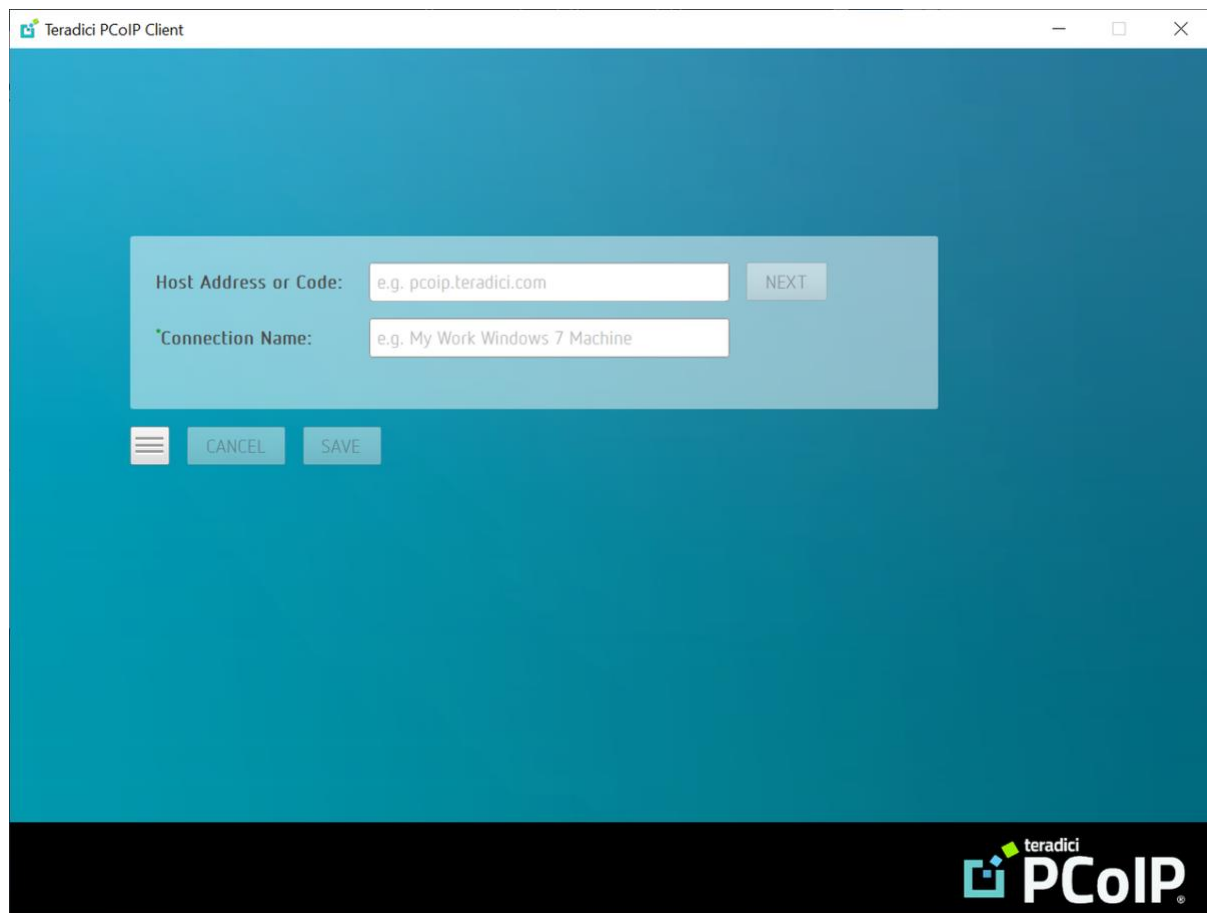
Security group name	Security group ID	
My-Studio-Linux-Teradici-SG	sg-09d468e5b2aea4e9a	<input type="button" value="Remove"/>
My-Studio-Deadline-SG	sg-026b013a7b3dc20c9	<input type="button" value="Remove"/>

- Now is a good time to note the Security Group ID of the Linux Teradici security group(e.g., My-Studio-Linux-Teradici-SG) that you created when launching your instance. You should enter its name and ID in the “Notes” section of the [Important Information Cheat Sheet](#) that you used for the other tutorials.
- Click **Save**

## Connect to Instance with Teradici

Since we used the Teradici Marketplace AMI to launch your Linux instance, it should already setup to connect to using the Teradici client software on your local machine.

- Once the status of your Workstation\_Linux instance has changed to **2/2 checks passed**, it is ready for you to connect
- Launch the Teradici client software on your local machine by double-clicking the **desktop shortcut** or by going to **Start→Teradici→PCoIP Client** (Windows)



- In the **Host Address or Code** field, enter the **IPv4 Public IP** of your workstation instance.
  - Note: You can locate the IPv4 Public IP of your instance by going to the list of running instances in the AWS Console, selecting the instance and looking in the Description tab
- Click **Next**

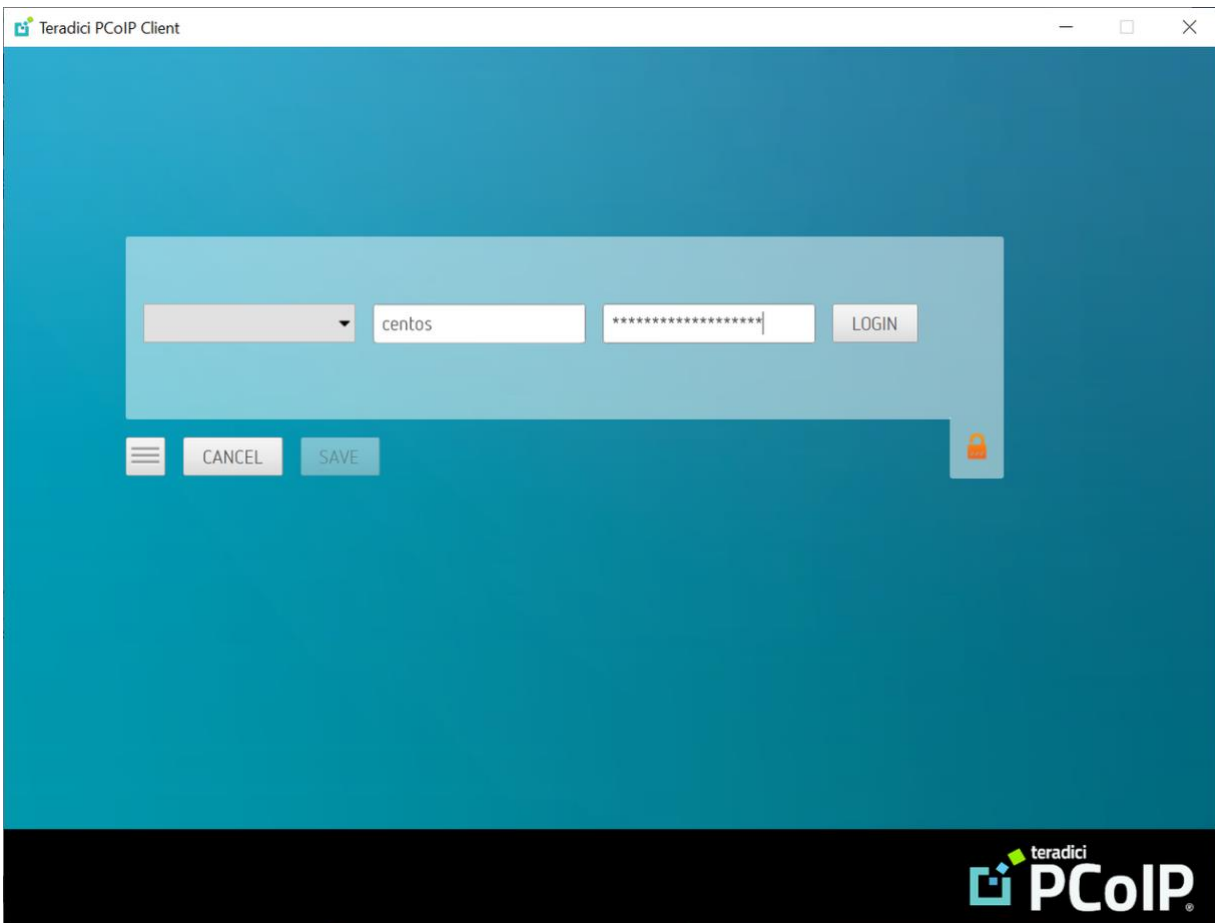
- If you get a popup that says “Cannot verify your connection”, click **Connect Insecurely** to continue.
  - Note: This connection is seen as insecure by Windows, but because the PColP protocol used by Teradici is inherently secure, your connection is still completely safe. From a [Teradici support page](#):

 **Note: About insecure connections**

Windows sees this connection as insecure because the PColP Agent uses a self-signed certificate and not one signed by a trusted certificate authority. **The PColP session is inherently secure, so connecting this way is safe.** If you would like to avoid this step in the future, you can create your own certificate and install the appropriate files on the remote machine and your PColP clients.

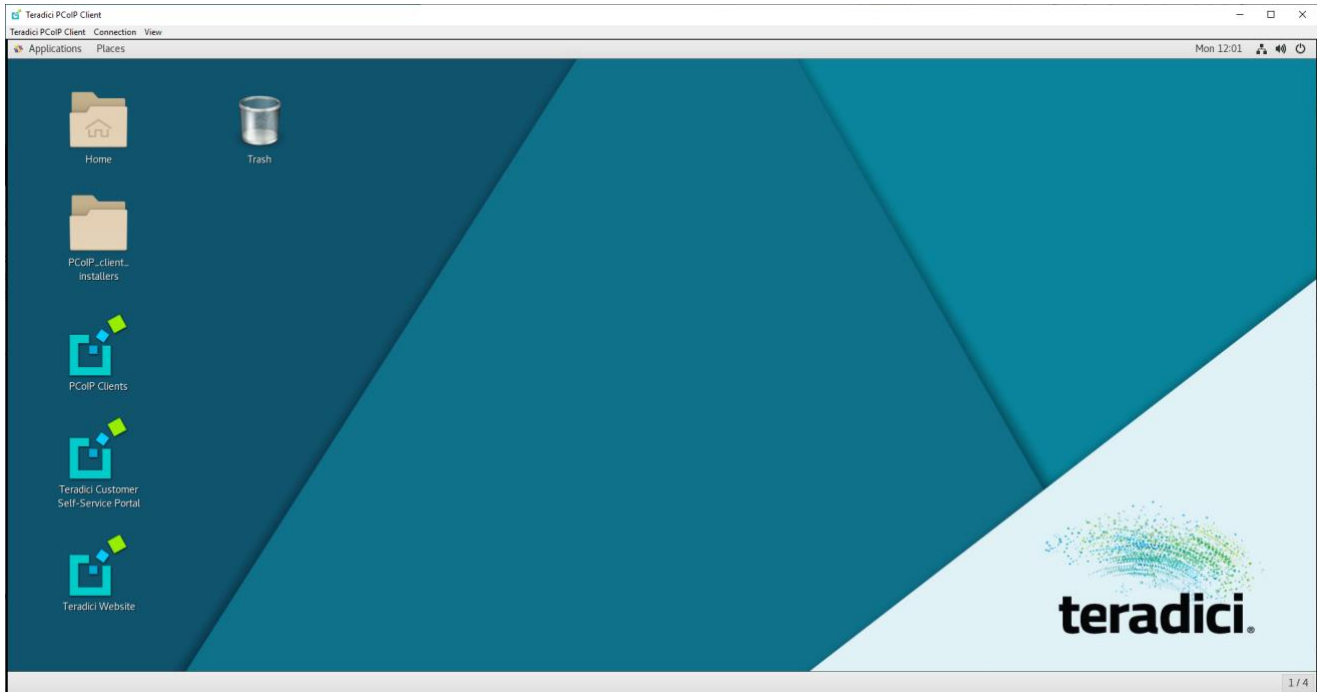
- For links to more information about Teradici security, see the [Appendix](#)
- For **Username** enter **centos**
- For **Password**, use the **Instance ID** for your instance

- You can find the Instance ID of your instance in the Description tab like we



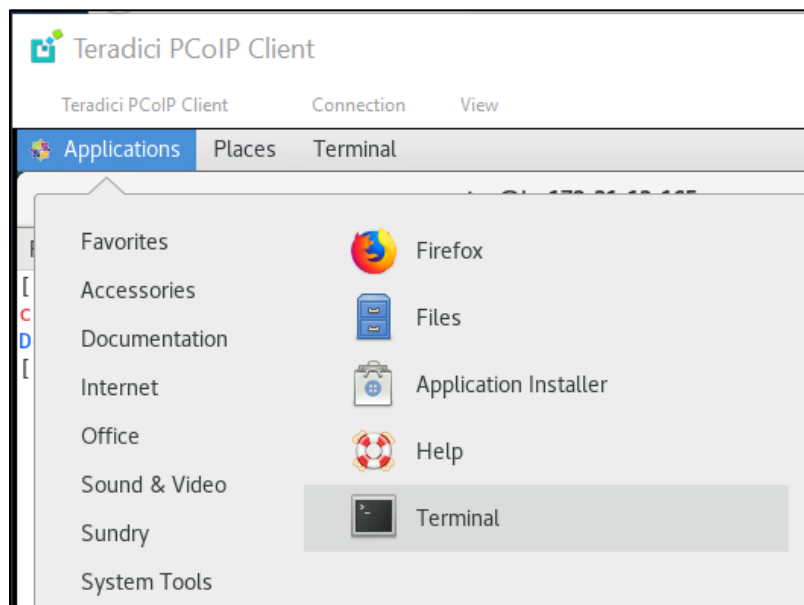
did for the IPv4 Public IP above

- Click **Login** to connect to your instance
- Once your Teradici session connects, you'll see the desktop for your Linux workstation. Note: The first time you connect, you may see a black screen for a minute or two, but don't worry, the session should eventually connect and you will see a desktop similar to the one below.



## Connect to Active Directory

- Launch a **Terminal** by using the **Applications** menu at the top left of the desktop. You can find Terminal under System Tools.



- First we need to update the Linux instance with the latest software. In the **Terminal** run the command below:

```
sudo yum -y update
```

- **Note:** This initial update can take several minutes and may appear to freeze on some cleanup steps. But don't worry, just let it run for a few minutes and it should finish successfully.
- Next we'll install some specific tools to make it possible to connect to Active Directory:

```
sudo yum -y install sssd realmd krb5-workstation samba-common-tools
```

- After that, we'll run a command to actually join your Active Directory:

```
sudo realm join -U Admin@mystudio.com mystudio.com --verbose
```

- You should receive a message that says: **Successfully discovered: mystudio.com** (or your DNS name, if different)
- Type in the password for your Active Directory and you will see a message that says: **Successfully enrolled machine in realm**
- In preparation for the next step, it is recommended to perform a mkdir in /mnt/ to create a directory for the FSx mount.

```
sudo mkdir /mnt/studio
```

## Mount FSx File System

Now we'll mount your FSx file system so that your Linux workstation has access to shared storage.

- First run this command to install some utilities:

```
sudo yum -y install cifs-utils
```

- Next run the command below and enter the Active Directory Admin password when prompted. Make sure to replace MYSTUDIO.COM with your **Active Directory's DNS Name** and type it in all capital letters.

```
kinit Admin@MYSTUDIO.COM
```

- Finally, run the command below to mount your FSx drive. Again, make sure you update the red text to match your **Active Directory DNS Name** and the **FSx DNS Name**. You can find this information under Tutorial 2 and Tutorial 3 on your cheat sheet.
- Note: Check the direction of the slashes when you enter the FSx DNS Name since they are “\” when on Windows, but “/” when on Linux.

```
sudo mount -t cifs -o user=Admin@MYSTUDIO.COM,cuid=$(id -u),uid=$(id -
u),sec=krb5,file_mode=0777,dir_mode=0777 //fs-
0c0b4fab1db1b9a0e.mystudio.com/share /mnt/studio -o vers=3.0
```

**Again, note which text is capitalized. These commands ARE case sensitive.**

- To test and see if it mounted correctly, run this command:

```
ls /mnt/studio
```

- If you see a few folders returning back, then you’ve connected correctly!

## Create a Script to Automatically Mount FSx

We want to make sure that every time our Linux workstation restarts that the FSx file system is automatically re-mounted. To do that, we’re going to create a script that can be run on instance startup. If you completed Tutorial 9, we did something similar there for the Linux render scheduler.

- From the **Terminal** run

```
sudo gedit /usr/bin/workstation.sh
```

- <shift>+click the image below to open a new browser tab with the text that needs to be entered into the gedit window:



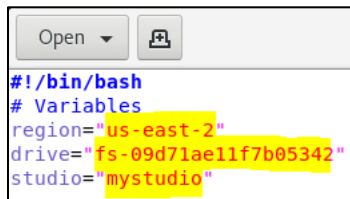
```
#!/bin/bash
# Variables
region="us-east-2"
drive="fs-09d71ae1f7b05342"
studio="mystudio"

password=$(sudo /usr/local/bin/aws secretsmanager get-secret-value --region $region --secret-id "Admin/DomainJoin" | python -c "import sys, json; obj=json.loads(sys.stdin)['SecretString'];print(json.loads(obj)['AdminPassword'])")
echo $password | kinit Admin@$studio.COM
sudo mount -t cifs -o user=Admin@$studio.COM,cuid=$(id -u),uid=$(id -u),sec=krb5,file_mode=0777,dir_mode=0777 //drive.$studio.com/share /mnt/studio -o vers=3.0
echo "Your storage has been mounted successfully."
```

**workstation\_mount-script\_01.txt** - <shift>+click the image above to open the text file in a new tab

- Cut and paste the text from the browser tab into the **gedit** window

- **IMPORTANT:** You will need to update the items highlighted below in **yellow** with specific information for your studio:



```
#!/bin/bash
# Variables
region="us-east-2"
drive="fs-09d71ae11f7b05342"
studio="mystudio"
```

- Update the value in quotes to the right of **region** with the **region for your studio**. *You can find your region on your cheat sheet.*
- Update the value in quotes to the right of **drive** with your **FSx File System ID**. *This can also be found on the cheat sheet.*
- Update the value in quotes to the right of **studio** with the **Directory NetBios name** for your studio (e.g., mystudio). *Refer to your cheat sheet for this as well.*
- Your workstation.sh file should look something like this when you're done:



```
#!/bin/bash
# Variables
region="us-east-2"
drive="fs-09d71ae11f7b05342"
studio="mystudio"

password=$(sudo /usr/local/bin/aws secretsmanager get-secret-value --region $region --secret-id "Admin/DomainJoin" | python -c "import sys, json; obj=json.loads(sys.stdin)['SecretString']; print(json.loads(obj)['AdminPassword'])")
echo $password | kinit Admin@$(studio).COM
sudo mount -t cifs -o user=Admin@$(studio).COM,cuid=$(id -u),uid=$(id -u),sec=krb5,file_mode=0777,dir_mode=0777 //$drive.$studio.com/share /mnt/studio -o vers=3.0
echo "Your storage has been mounted successfully."
```

- Save the file and exit the editor
- Next run the command below to add execute permissions to your script:

```
sudo chmod +x /usr/bin/workstation.sh
```

## Create a Service to Run the Script

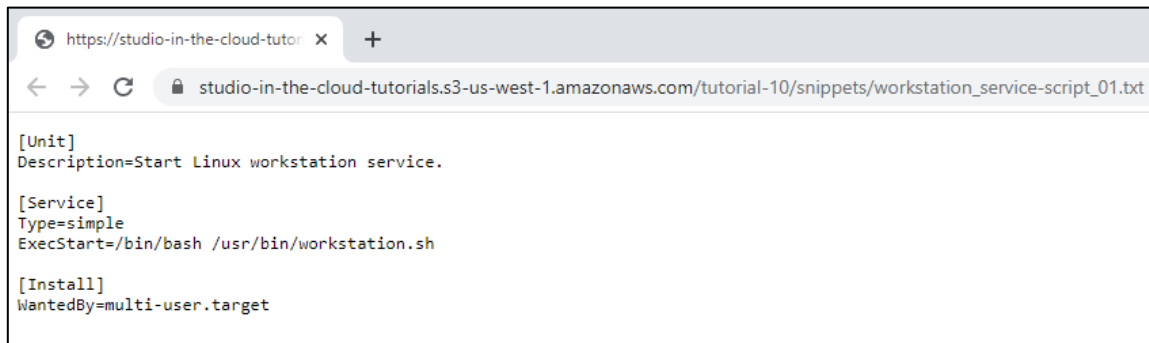
Next, you're going to create a service that will execute this script every time the instance is started. The nice thing about working this way is that we can modify this file to add more options any time we want and it will be executed again whenever the system is restarted. It's similar to the user data settings we have on the instance, but in this case it's much more controllable.

- Create the service file using **gedit**

```
sudo gedit /etc/systemd/system/workstation.service
```



- <shift>+click the image below to open a new browser tab with the text that needs to be entered into the gedit window:

A screenshot of a web browser window. The address bar shows the URL: https://studio-in-the-cloud-tutorials.s3-us-west-1.amazonaws.com/tutorial-10/snippets/workstation\_service-script\_01.txt. The main content area displays the following text:

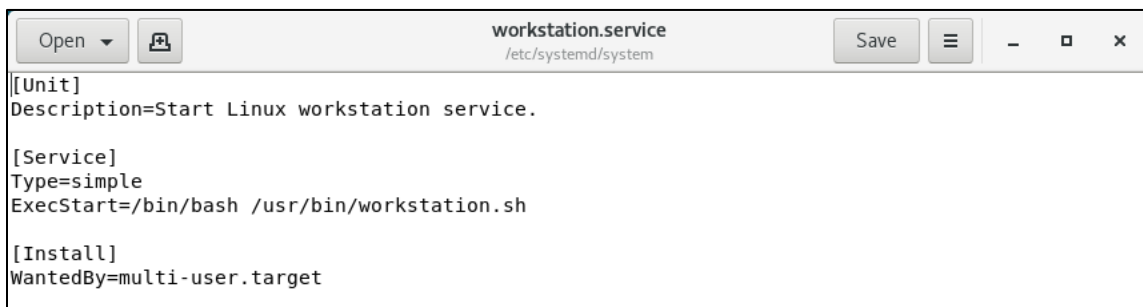
```
[Unit]
Description=Start Linux workstation service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/workstation.sh

[Install]
WantedBy=multi-user.target
```

**workstation\_service-script\_01.txt** - <shift>+click the image above to open the text file in a new tab

- Cut and paste the text from the browser tab into the **gedit** window
- Your workstation.sh file should look like this when you're done:

A screenshot of the gedit text editor window. The title bar shows "workstation.service" and the file path "/etc/systemd/system". The main content area displays the following text:

```
[Unit]
Description=Start Linux workstation service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/workstation.sh

[Install]
WantedBy=multi-user.target
```

- Save the file and exit the editor
- Next we'll start the service

```
sudo systemctl start workstation
```

- And then make sure will start automatically on boot

```
sudo systemctl enable workstation
```

## Install the AWS Command Line Interface

Our workstation.sh startup script that we created above makes use of the AWS Command Line Interface (AWS CLI) to run some commands that retrieve your Active Directory's Admin user's password. But the Teradici Marketplace Linux AMI doesn't come with the AWS CLI installed by default, so we need to install it now.

- Run this command in the **Terminal** to download the zip file for the AWS CLI:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

- Next run the following to unzip the file:

```
unzip awscliv2.zip
```

- Finally, run this to install the AWS CLI:

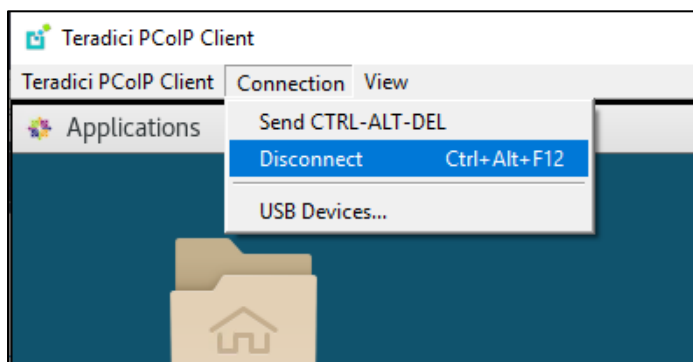
```
sudo ./aws/install
```

- When the install is complete, you will get a message that says **You can now run: /usr/local/bin/aws --version**
  - You don't have to run the above command, but you can if you want to verify the version of the AWS CLI that you have just installed.
- For more details, see [Installing the AWS CLI version 2 on Linux](#)

## Disconnect Your Teradici Session

To test that the service you created in the last step is working, we're going to stop and start your Linux workstation instance. But before we do that, we should disconnect our Teradici session.

- From the Teradici client's menu bar, open the **Connection** menu, then select **Disconnect**



## Check That the Service Is Working

If this has all been set up correctly, you should be able to stop your instance, start it again, and then connect to it and see that the FSx file system is already mounted.

- Go **EC2** → **Instances**
- Select the **Workstation\_Linux** instance
- Choose **Instance State** → **Stop instance**
- After the instance has stopped, choose **Instance State** → **Start instance**
- Once the instance has spun up and is ready to use, copy the **IPv4 Public IP**
  - Note: The public IP address changes every time an instance stops and starts
- Open the Teradici client on your local machine and enter the new public IP address
- Log in using username **centos** and the **Instance ID** for your instance that you used earlier
- Once logged in, open a **Terminal** and run:

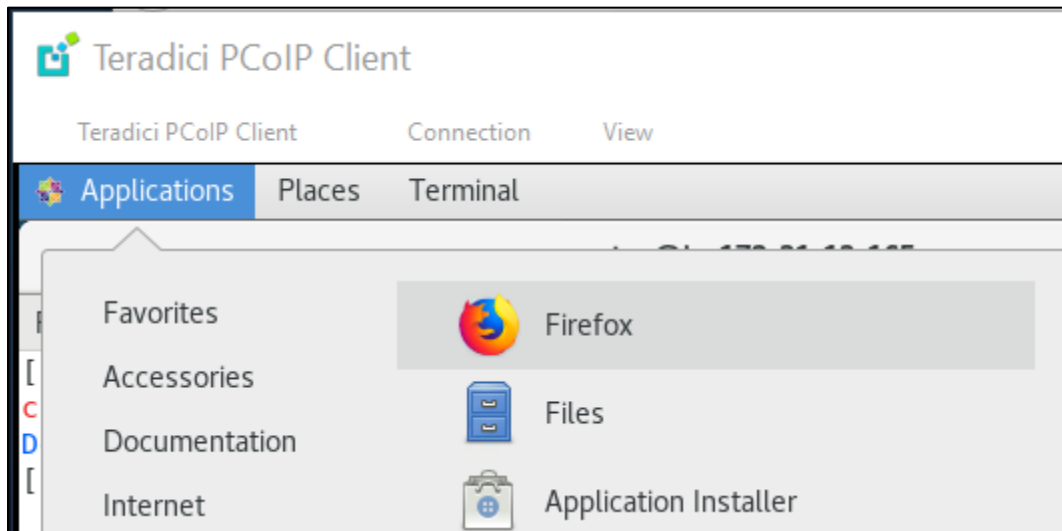
```
ls /mnt/studio
```

- If all goes well, you should see the directories on your FSx file system

## Download Blender Installer

If you already downloaded the Blender installer for Linux in Tutorial 5, you can skip to the next step in this tutorial: [Install Blender](#). If not, follow these steps to download the installer:

- Launch Firefox by using the **Applications** menu at the top left of the desktop. You can find Firefox under Favorites.



- In Firefox, navigate to <https://www.blender.org/download/>
- Click **Download Blender 2.82a** (the latest version at the time of publication). The version number you see may be different.
- When the window pops up asking what you would like to do with **blender-2.82a-linux64.tar.xz**, choose **Save File** and click **OK**.
- Back in the **Terminal**, create a blender folder in `/mnt/studio/installers`

```
mkdir /mnt/studio/installers/blender
```

- Once the download is complete navigate to your **Downloads** folder.

```
cd ~/Downloads
```

- Move the installers to **/mnt/studio/installers/blender**

```
mv blender-*tar.xz /mnt/studio/installers/blender
```

## Install Blender

- In the **Terminal**, navigate to the **installers** directory on your FSx file system.

```
cd /mnt/studio/installers/blender
```

- Next, you should unzip the contents of the installer zip file to your Linux workstation:

```
sudo tar -xvf blender-*tar.xz -C /usr/local/
```

- Finally, create an executable so you can run Blender by simply typing blender.  
Note: Make sure to substitute **2.93.1** with the version number that you downloaded.

```
sudo ln -s /usr/local/blender-2.93.1-linux64/blender /usr/local/bin/blender
```

## Install Additional Applications

There are two additional applications we would recommend installing, **DJV** and **Krita**:

- **DJV** (<http://djv.sourceforge.net/>) - Professional media review software for VFX, animation, and film production.
- **Krita** (<https://krita.org/>) - Professional open source painting program.

For each of these, we recommend creating a folder inside **/mnt/studio/installers** (e.g., **/mnt/studio/installers/krita**, **/mnt/studio/installers/djv**), downloading the installer files from the appropriate websites, and copying them to the installers location

## Install the Deadline Client

Now we'll install the Deadline Client. Again, it's very similar to what we've done in the past.

- In the **Terminal**, run this command to install some extra libraries that are needed by Deadline:

```
sudo yum -y install lsb
```

- Next, navigate to the Thinkbox installers directory on your FSx file system

```
cd /mnt/studio/installers/thinkbox
```

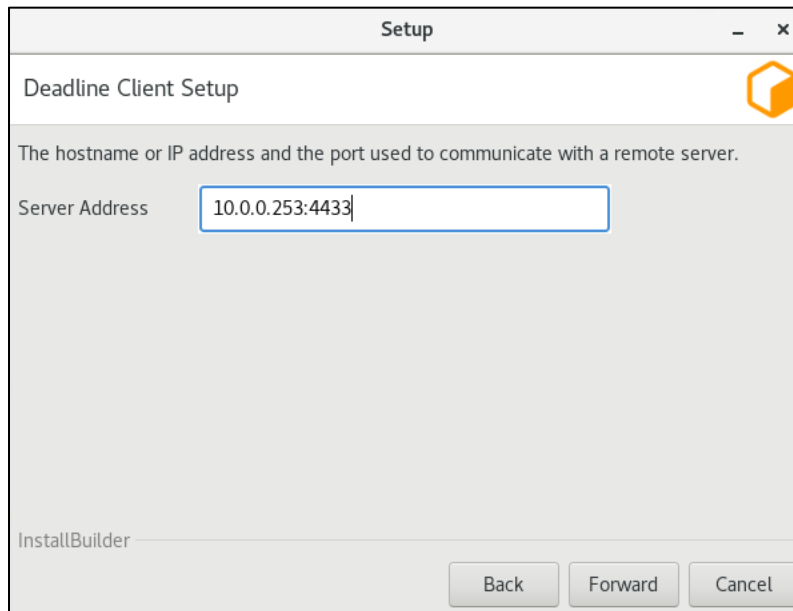
- If you haven't done Tutorial 6, you will need to unzip the Deadline Linux installers. If you have done Tutorial 6, you can skip to the next step.

```
sudo tar -xvf Deadline-10*-linux-installers.tar
```

- To install the Deadline Client run this command.

```
sudo ./DeadlineClient-10*-linux-x64-installer.run
```

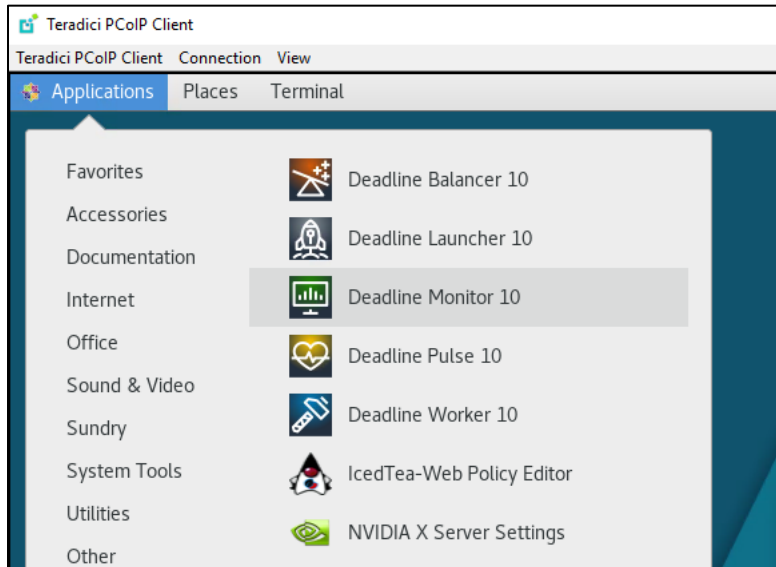
- Give these answers to the prompts:
  - Click **Forward** when the Welcome message appears
  - Accept the **License Agreement** and click **Forward**
  - Leave **Installation Directory** at the default and click **Forward**
  - On the Select Installation Type page, leave **Client** selected and click **Forward**.
    - Note: There is no need to install the Remote Connection Server because this workstation is simply a client.
  - Leave the **Repository Connection Type** set to **[Recommended] Remote Connection Server** and click **Forward**
  - Put the **Private IP address** of the Render Scheduler machine into **Server Address**, and point it at port **4433**
    - You should have already written down the Private IP for your Render Scheduler on the cheat sheet during Tutorial 4 if you have a Windows Render Scheduler or during Tutorial 9 if you have a Linux Render Scheduler. It should look something like (10.0.0.219)
    - Enter the private IP with a port number of 4433 into the server address field (e.g., 10.0.0.219:4433) Note: This is important, because the default port is 8080 and that won't work.



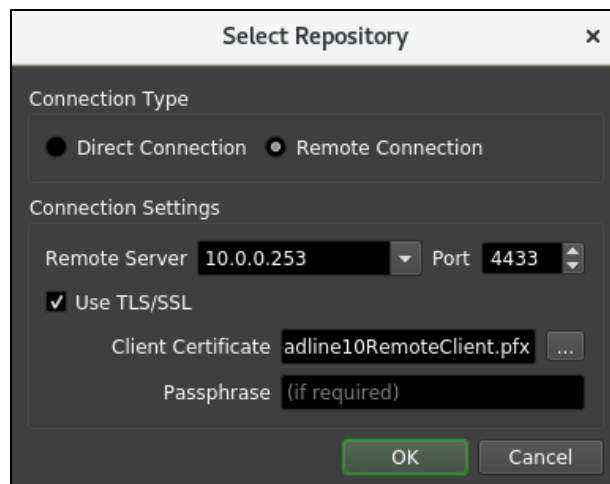
- Click **Forward** to continue
- Set the **RCS TLS Certificate**
  - If you have a **Windows Render Scheduler** that you created in Tutorial 4, set it to  
**/mnt/studio/app\_env/thinkbox/DeadlineCertificates/Deadline10RemoteClient.pfx**
  - If you followed Tutorial 9 to create a **Linux Render Scheduler**, set it to  
**/mnt/studio/app\_env/thinkbox/DeadlineCertificates\_linux/certs/Deadline10RemoteClient.pfx**
- Leave the **Certificate Password** blank and click **Forward**
- Uncheck **Launch Worker When Launcher Starts** and click **Forward**
  - Note: We are turning this off because we are currently setting up one of our workstation machines. We don't want render workers to launch on these instances.
- Leave **Block auto upgrade via a secure setting** selected and click **Forward**
- Click **Forward** one more time to proceed the install
- Finally, click **Finish** to exit the installer

## Connect to the Deadline Repository

- Run the Deadline Monitor by going to **Applications** → **Other** → **Deadline Monitor 10**



- You'll see a popup window that says a new account for 'centos' has been created. Click **OK** to continue.
- If you get an error saying it can't connect to the Repository make sure you have these options selected in the prompt. Note: Your IP address will be different. Also make sure you have the correct port (4433) selected.





- If it still doesn't work, double-check that your Render Scheduler instance is running, you're logged in as Administrator, and **Deadline Monitor**, **Deadline RCS**, and **Deadline Pulse** are all running.
- If it works correctly, you will be able to connect to the Deadline Monitor.

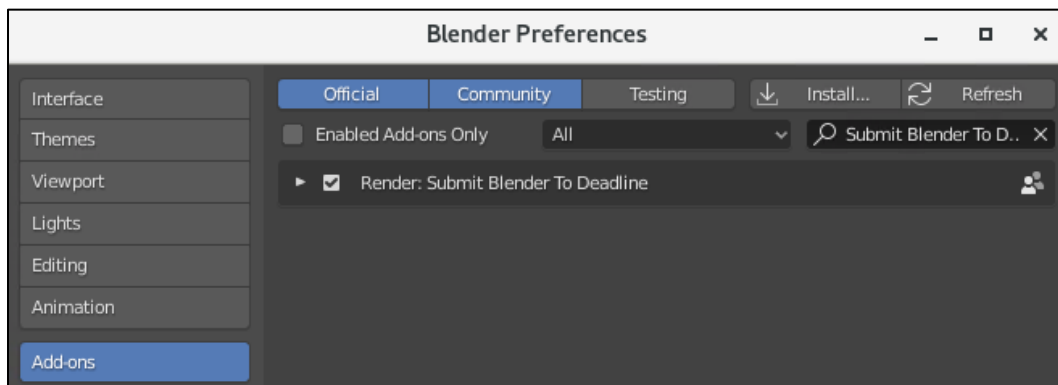
## Install Deadline Blender Submitter

In preparation for working with your render farm, we recommend you install the Deadline Submitter add-on inside of Blender.

- Open a new **Terminal** window
  - Note: It is important to open a new Terminal window here instead of using your existing one. Otherwise some environment variables will not be set correctly.
- From the new Terminal, run the following command to launch **Blender**

```
blender &
```

- Go **Edit** → **Preferences...** (or **File**→**Preferences** - for Blender 2.79 & earlier).
- Click **Add-ons** in the left panel.
- Click **Install**.
- Navigate to **/mnt/studio/installers/thinkbox/submission/Blender/Client**
- Choose **DeadlineBlenderClient.py** .
- Click **Install Add-on**. Wait a minute for the submitter to install.
- Click the checkbox next to **Render: Submit Blender to Deadline** add-on.



- Close your preferences window.

## Test Submitting to Deadline

- **Save** your current scene (this is a requirement to submit to Deadline).
- Choose **Render**→**Submit To Deadline**.
- The **Submit Blender Job To Deadline** window will pop up.
- Change the **Output File** path to a location of your choosing.
- Change the Frame list from 1-250 to **1**.
- Hit **Submit** and then close the submit window
- If it's not already running, open the **Deadline Monitor**.
- Your job will be sitting in the queue.
- Since we didn't specify a group in the render submission, the job will not actually render, but don't worry about that for now. All we wanted to do is test that our job was successfully submitted to your Render Scheduler. After finishing this tutorial, you can move on to [Tutorial 6: Setting Up a Linux Farm Worker and Spot Fleet Request](#) if you haven't already completed it previously.

## Create a Linux Workstation AMI

Now that you've set up your Linux workstation and installed applications, you'll want an easy way to launch another workstation just like this one without having to go through all the steps again. We already did something similar in [Tutorial 3](#) when we created an AMI and launch template for your User Management instance.

- **Disconnect** from your Teradici session
- Next, in the **AWS Console**, go to **Services**→**EC2**
- Click **Instances**
- Right click your **Workstation\_Linux** instance and choose **Stop instance**
- Wait for the **Instance state** to change to **stopped**

<input type="checkbox"/>	Workstation_Win_new	i-0192b245cc6b7691a	⊖ Stopped
<input checked="" type="checkbox"/>	Workstation_Linux	i-0afa4a20f4777cf64	⊖ Stopped

- Right-click the instance and choose **Image and templates** → **Create image**
- Give it an appropriate name (e.g., My-Studio-Linux-Workstation-AMI).
- Give your workstation AMI a description if you want.
- Increase its storage if necessary.

**Create image** [Info](#)

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

Instance ID  
i-0afa4a20f4777cf64 (Workstation\_Linux)

Image name  
My-Studio-Linux-Workstation-AMI  
Maximum 127 characters. Can't be modified after creation.

Image description - optional  
Linux Workstation AMI with Deadline and Blender  
Maximum 255 characters

No reboot  
☐ Enable

Instance volumes

Volume type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted
EBS	/dev/sda	Create new snapshot fr...	150	EBS General Purpose SS...	450		<input checked="" type="checkbox"/> Enable	<input type="checkbox"/> Enable

[Add volume](#)

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

☒ Tag image and snapshots together  
Tag the image and the snapshots with the same tag.

☐ Tag image and snapshots separately  
Tag the image and the snapshots with different tags.

No tags associated with the resource.

[Add tag](#)  
You can add 50 more tags.

[Cancel](#) [Create image](#)

- Click **Create Image**.
- Wait 5 to 10 minutes for the new AMI to become available
  - To see if your AMI is ready, click **AMIs** in the left panel. The AMI will need to finish creating before you can create a Launch Template with it.
  - While you're waiting, you can also add Tags to your AMI. Again, we recommend creating at least a **Studio** tag and **Name** tag.
  - You may also want to enter the AMI ID and name in the Notes section of the [Important Information Cheat Sheet](#).

## Create Linux Workstation Launch Template

- Once your new AMI is listed as available, go to **Services** → **EC2** and click **Instances (running)**.
- Right-click your **Workstation\_Linux** instance and choose **Image and templates**→**Create template from instance**.
- Name your launch template (e.g., My-Studio-Linux-Workstation-LT)
- Give it a description if you want.

**Launch template name and description**

Source instance  
i-05e5e6fd0599c1643

Launch template name - *required*  
  
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description  
  
Max 255 chars

**Auto Scaling guidance** [Info](#)  
Select this if you intend to use this template with EC2 Auto Scaling  
☐ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► **Template tags**

- You will need to change the **AMI ID** to point to the one that you just created.

- In the **AMI dropdown**, scroll down to the **My AMIs** section and then select the Workstation AMI (e.g., My-Studio-Linux-Workstation-AMI) that you just

**Launch template contents**  
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

**Amazon machine image (AMI)** [Info](#)

AMI

My-Studio-Linux-Workstation-AMI  
ami-048a6c297aa982e11  
Catalog: My AMIs   architecture: 64-bit (x86)   virtualization: hvm

▼

made.

- Under **Network interfaces**, set **Auto-assign public IP** to **Enable**, otherwise you will not be able to connect to the instances you launch.
- Also under **Network interfaces**, check that the **Security Group IDs** for both your **Deadline Security Group** (e.g., My-Studio-Deadline-SG) and your **Linux Teradici Security Group** (e.g., My-Studio-Linux-Teradici-SG) are listed. *You can find the IDs for both of those security groups on your cheat sheet.*

Click **Show all selected** to view the IDs for the currently selected security groups.

If either security group is missing, click the drop down menu and select the missing group.

The screenshot shows the 'Network interfaces' configuration page in the AWS Management Console. The page is titled 'Network interfaces' with an 'Info' link. Below the title, there is a 'Network interface 1' section with a 'Remove' button. The configuration fields are organized into three columns:

- Device index:** A text box containing '0' with an 'Info' link.
- Network interface:** A dropdown menu showing 'Don't include in launch tem...' with an 'Info' link.
- Description:** A text box containing 'Primary network interface' with an 'Info' link.
- Subnet:** A dropdown menu showing 'subnet-0c617ceb03dc022a0' with an 'Info' link.
- Auto-assign public IP:** A dropdown menu showing 'Enable' with an 'Info' link.
- Primary IP:** A text box containing '123.123.123.1' with an 'Info' link.
- Secondary IP:** A text box containing '123.123.123.1' with an 'Info' link, a close button (X), and an 'Add IP' button.
- IPv6 IPs:** A text box containing '2001:0db8:85a3:0000:0000:ff' with an 'Info' link, a close button (X), and an 'Add IP' button.
- Security groups:** A dropdown menu showing 'Select security groups' with an 'Info' link, a refresh button (circular arrow), and a 'Show all selected (2)' button.
- Delete on termination:** A dropdown menu showing 'Yes' with an 'Info' link.
- Elastic Fabric Adapter:** A checkbox labeled 'Enable' with an 'Info' link.

At the bottom of the configuration section, there is an 'Add network interface' button.

- Click **Create launch template**.

## Launch a New Workstation

Now let's test the launch template!

- Go to **Services** → **EC2** and click **Launch Templates** in the left panel.
- Select your Linux Workstation launch template (e.g., My-Studio-Linux-Workstation-LT).
- Choose **Actions** → **Launch instance from template**.
- Select the version (if this is your first time running through the tutorial you'll select version 1).
- Choose **Launch instance from template**.
- Go back to **Services** → **EC2**, click **Instances (running)**.
- When your new instance is done initializing you can connect to it using the Teradici client again. This time, instead of logging with username "centos", you can login with an Active Directory username and password

- **Note:** the syntax for logging in with an Active Directory username and password is different with Teradici on Linux. Instead of logging in as <domain\_name>\<user>(mystudio\jason), your username should be <user>@domain\_name (e.g., jason@mystudio.com)
- **Note:** The Deadline Blender submitter needs to be installed separately for each user. If they follow the [same instructions provided earlier in this tutorial](#), they should be all set. This is also the case for the Windows workstations that are created in Tutorial 5. We call this out in the sample workflow instructions in Tutorial 7, but wanted to mention it here as well.

## Shut Down Notes

- Once you confirm that your new Workstation\_Linux instance is running correctly, feel free to terminate the old Workstation\_Linux instance that you set up at the beginning of this tutorial. It should currently be listed as stopped in the instance list.
- If you or your artists don't have an immediate need for the Linux workstation that is running, you should stop or terminate it as well. Since you created a launch template for your Linux Workstation, you can easily spin up a new one whenever you want.

---

## Appendix

### Links to AWS Documentation

- [AWS Direct Connect](#)
- [AWS VPN](#)
- [VPN Connections to AWS VPC](#)
- [VPN Connections to AWS VPC](#)
- [Installing the AWS CLI version 2 on Linux](#)

### Links to Other Resources

- [Teradici Cloud Access Software](#)
- [Getting Started Guide - Connecting with a PCoIP Client](#)

- [What is PCoIP Technology?](#)
- [Cloud Access Software Security Features](#)
- [PCoIP Software Client Security Modes](#)
- [Installing Certificates on PCoIP Client for Windows](#)
- [Teradici Graphics Agent Configuration Guide](#)