

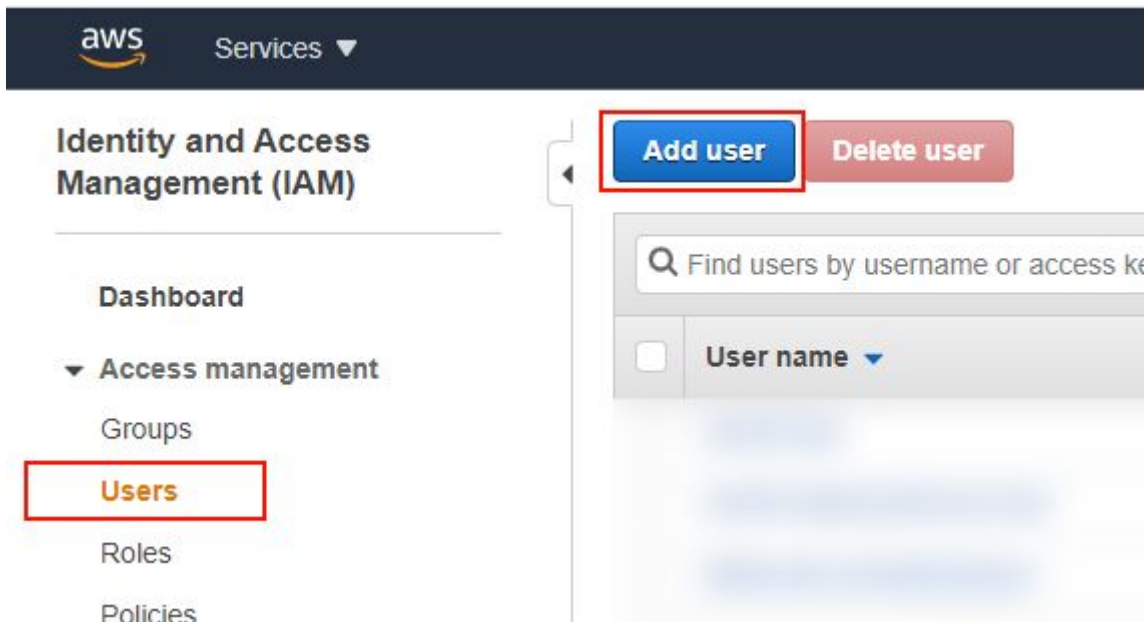
AWS SDK for Unity3d by Square Beam

Unity Friendly AWS SDK

Installation

Step 1. Create [AWS Profile](#) or [Login](#) in existing AWS Console

Step 2. Open [AWS IAM](#), hit “Users” in the left column and click “Add User”.



In the opened window set “User Name” and set “Access Type” to Programmatic Access.

Add user



Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

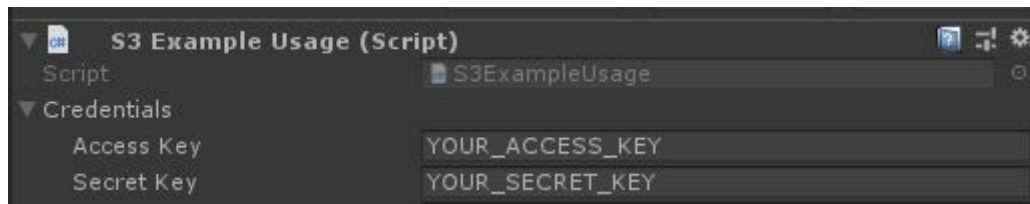
- Access type* ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

A good practice is to have User-Per-Service

Download that `credentials.csv` file, Open it as a txt file and extract **aws_access_key** and **aws_secret_key** and store it in a secure place.

Do not include credentials in the build as a raw file! If it's not possible then enable IL2CPP Scripting backend and put in some class/struct. In this case, it will be harder to reverse-engineer.

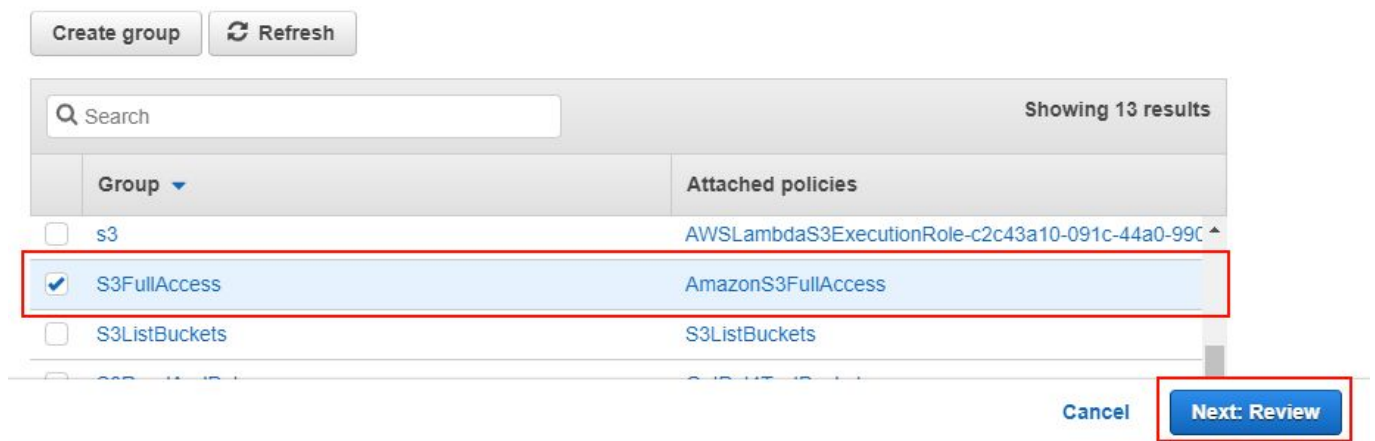
Step 3. For example let's open **S3ExampleScene**, Select game object **"Example"** and in the Inspector window put extracted credentials in corresponding fields.



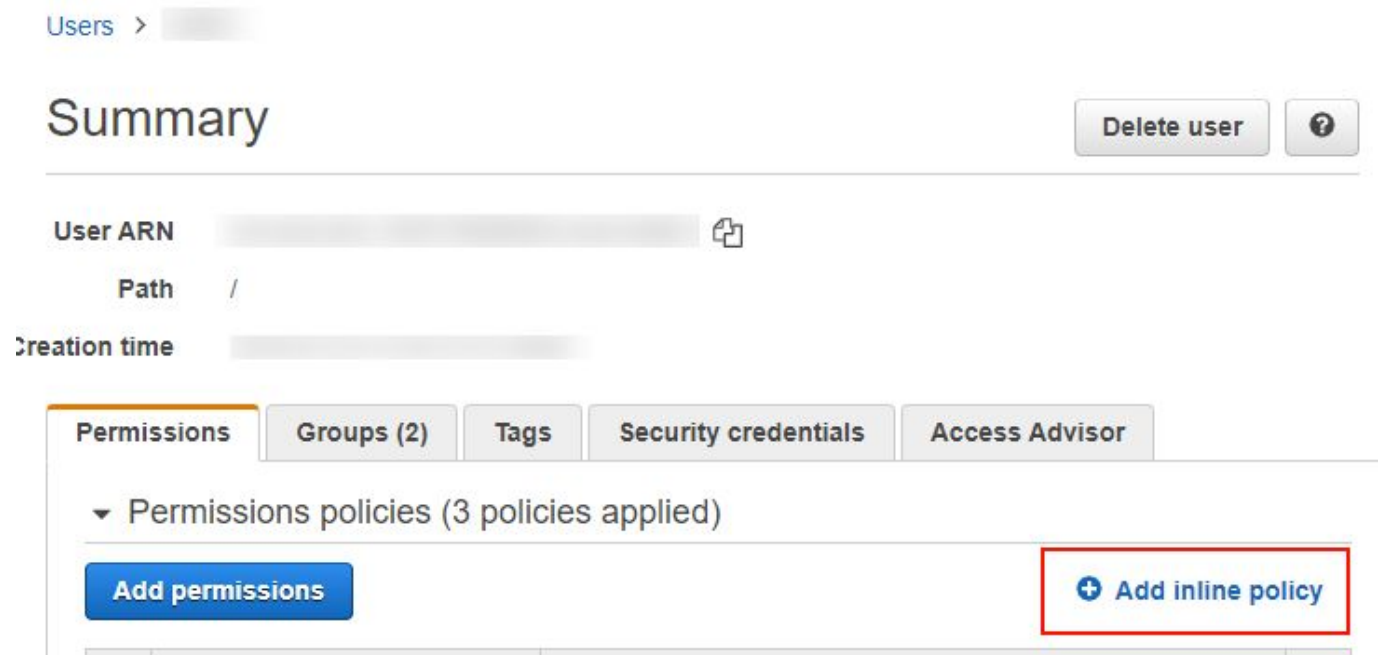
AWS S3

Step 1. Select your early created **IAM User** from the Installation paragraph and if you want to do a fast setup then please grant permission **S3FullAccess**.

Add user to group



Or you can add **Inline Policy**, in this case you can grant access only for specific buckets with specific permissions.




Step 2. Open **S3 Console** and hit **Create Bucket**. Provide **Bucket Name** and **Region**.

Create bucket

Buckets are containers for data stored in S3. [Learn more](#) 

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#) 

Region

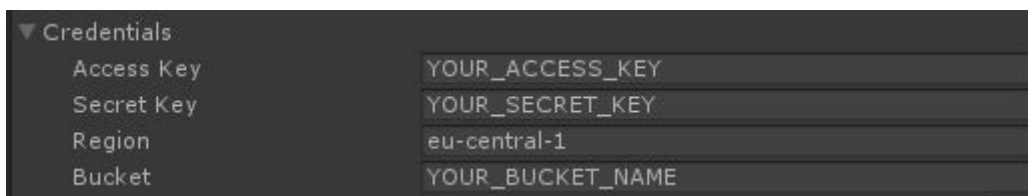
EU (Frankfurt) eu-central-1 ▼

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

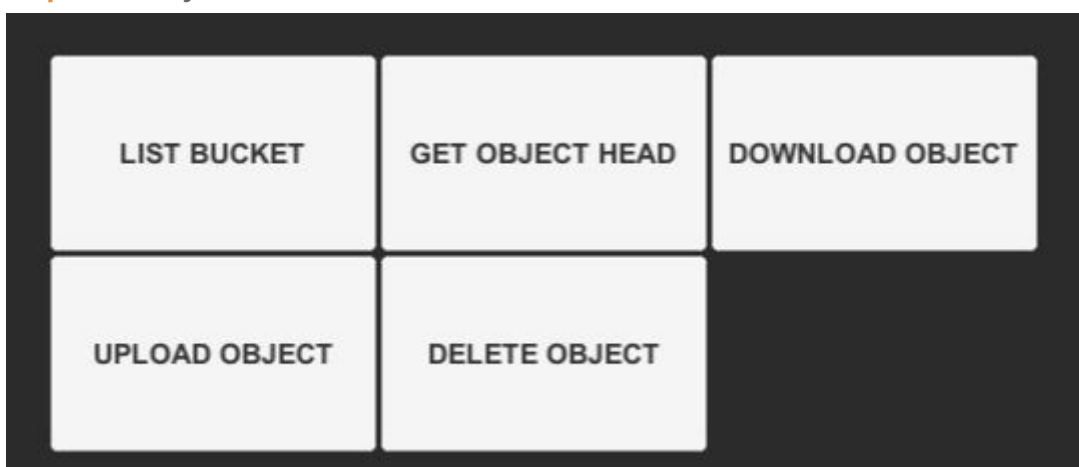
Region is very important. So for example if in UnityProject you set the wrong **Region** then it won't work! So for example if you created your bucket in the Frankfurt region then it will look like this in Inspector.



Step 3. Open S3ExampleScene, Select game object "Example" and in the Inspector window put Bucket Name and Region in corresponding fields.

Step 4. Upload test files in your early created bucket. You can find it under *PathToProject*\AWS\Services\S3\Example\TestFiles

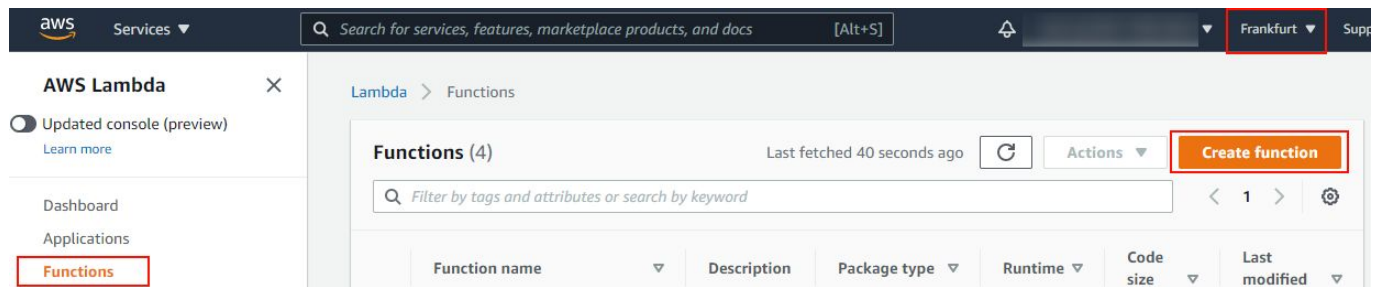
Step 5. Hit Play in Editor and Click on Buttons to work with it.



AWS Lambda

Step 1. Open **Lambda Console** and hit **“Create function”**.

Make sure that you are in the right Region (In my case it is Frankfurt eu-central-1)



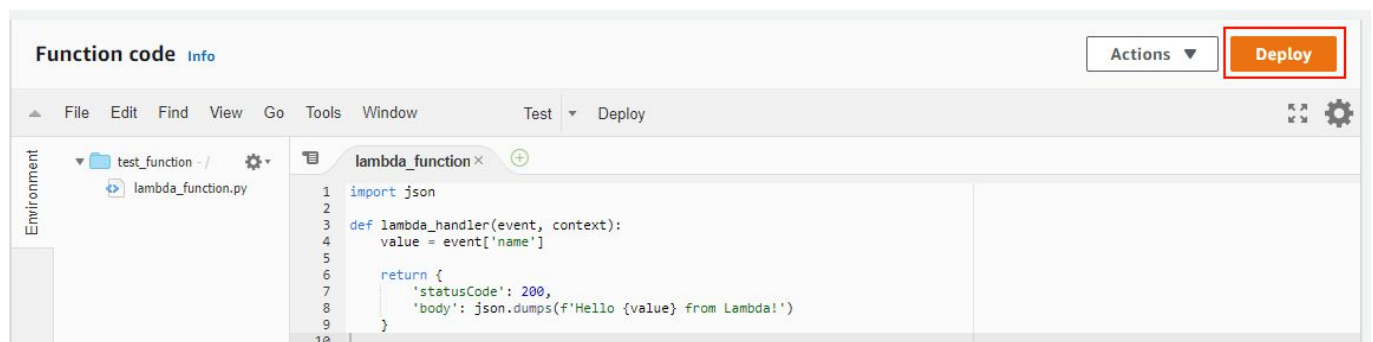
Set **Function Name** and **Runtime** to **Python 3.8** everything else keep as it is and hit **“Create function”**

A screenshot of the 'Basic information' form in the AWS Lambda console. The 'Function name' field contains 'myFunctionName'. The 'Runtime' dropdown is set to 'Python 3.8'. The 'Permissions' section shows a default execution role. At the bottom, there are 'Cancel' and 'Create function' buttons, with 'Create function' highlighted in orange.

Step 2. Replace default code on this simple “echo code” and hit **Deploy**

```
import json

def lambda_handler(event, context):
    value = event['name']
    return {
        'statusCode': 200,
        'body': json.dumps(f'Hello {value} from Lambda!')
    }
```



Step 3. Copy ARN of your function.



Select your early created **IAM User** from the Installation paragraph and hit **"Add inline policy"**.

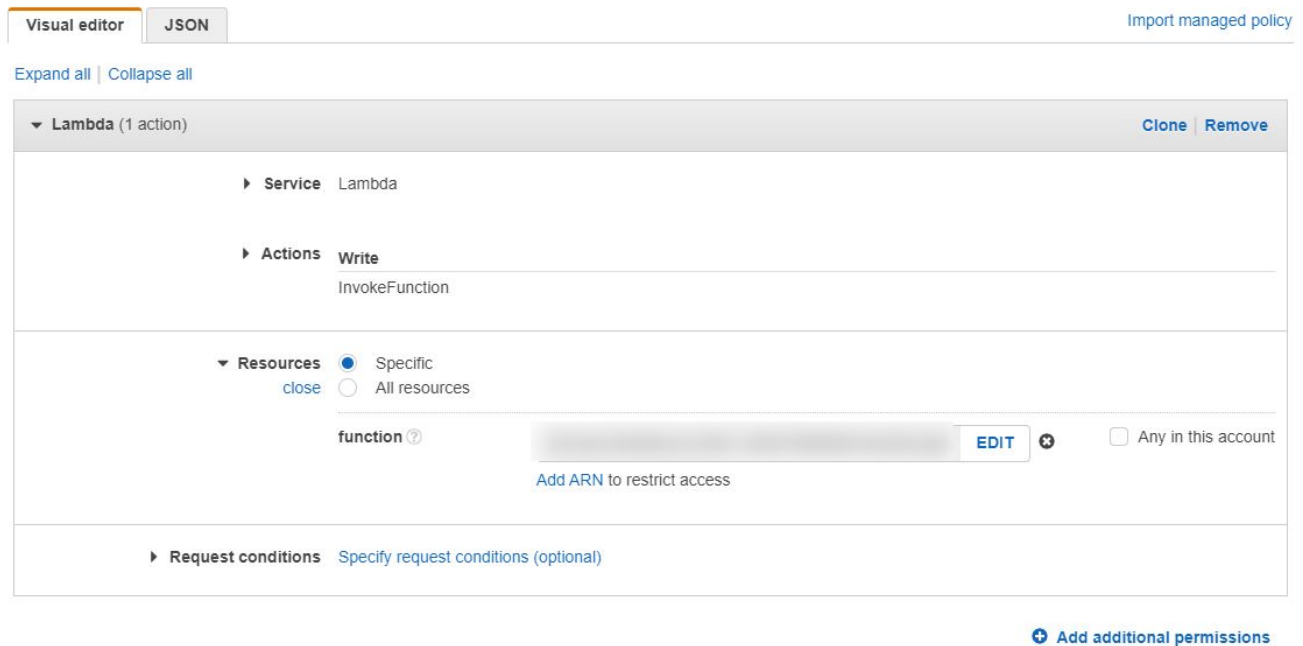


Step 4. We should grant InvokeLambda permission for this user. In the **Resources**, paste the ARN of your function which you copied earlier and hit **Review policy**, give it **Name** and hit **Create Policy**.

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

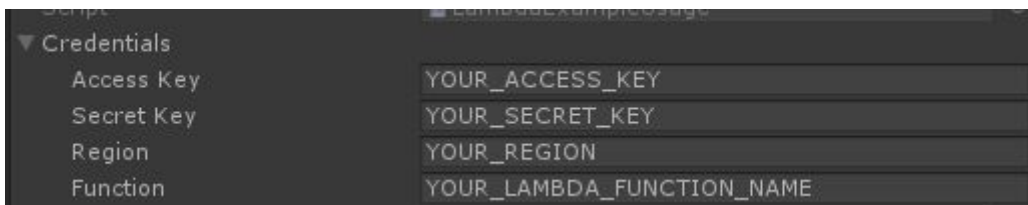


Character count: 527 of 2,048.

The current character count includes character for all inline policies in the user: XelevateGraphWorker.

Cancel Review policy

Step 5. Open **LambdaExampleScene**, select game object **Example**. In the Inspector window fill these credentials.



Step 6. Press Run and hit the InvokeLambda button.

AWS SDK by SQUARE BEAM WELCOMES YOU!

To fast start read please readme.md

Step #1

- Create Lambda Function in AWS Dashboard and Open it and upload **testFunction.zip** in it. It will put simple python code in your lambda function so you can verify that it is working.

Step #2

- Select 'Example' GameObject in Hierarchy. In Inspector you need to fill 'Credentials' property for 'LambdaExampleUsage' Component .

How to get credentials please read in Setup Guide which located in AWS/Setup Guide.pdf.

Step#3

- Run scene

Do you need help or have question? Do not hesistate to contact with me.

InvokeLambda

AWS EC2


Create custom AMI

Step 1. Open **EC Console** Navigate to Instances and hit “**Launch Instances**”.

Make sure that you are in the right Region (In my case it is Frankfurt eu-central-1)

Choose AMI, If you are newbie choose Amazon Linux 2 AMI.

1 to 32 of 32 AMIs

 **Amazon Linux**
Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-02cb52d7ba9887a93

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

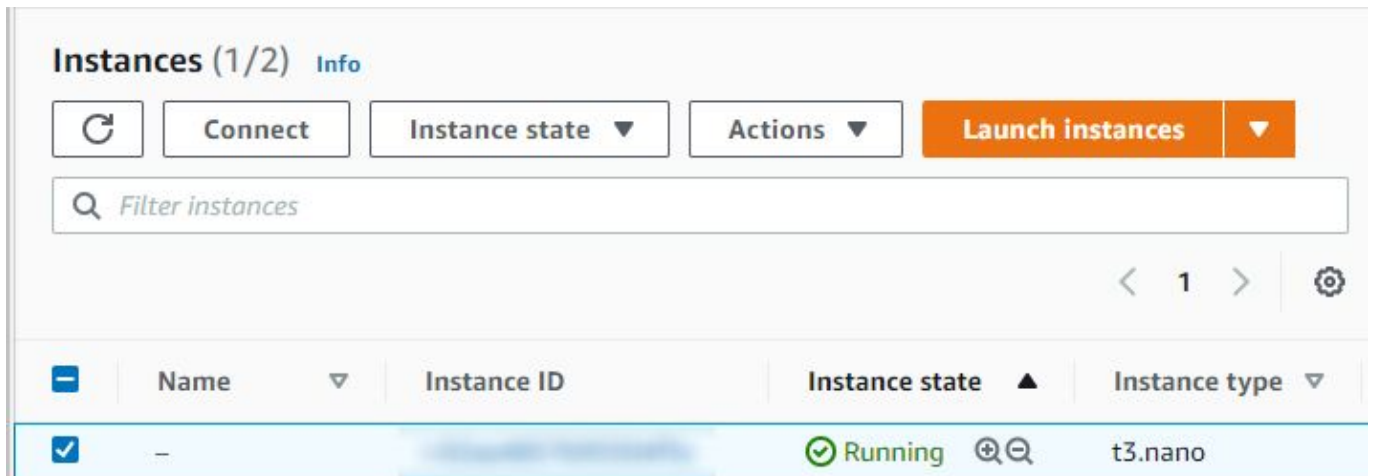
Select

64-bit (x86)

Then select “**Instance Type**”, choose cheapest if you want just to test.

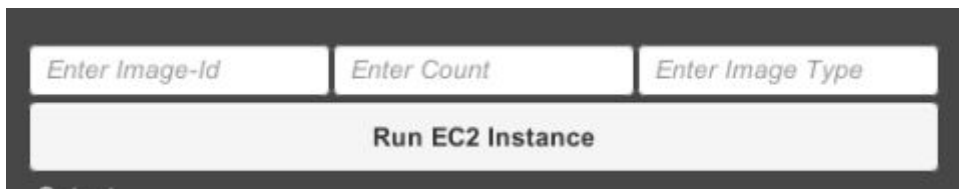
After you can hit **Review and Launch**, or you can specify more details if you want.

Navigate to **Instances**, select your runned instance and hit **Connect**, then you can install your software and establish processes. After you are finished with it **Stop** instance and hit **Actions->Image and Templates->Create Image**



Operate Instances

Step 1. Open Unity Project, go to Scene **"EC2ExampleScene"** and Run It.



In Image-Id set **AMI Id** which you created earlier, set **Count**, set **Instance Type**, (t3.micro) and hit **Run EC2 Instance**

Some Instance Types might not be available in specific regions.

Step 2. After success creation you will receive **InstanceId** which you can use in block below to operate Instances

