Boto3: Create EC2 with Volume, UserData, Tags and more

By **Yuri Bondarenko**

The code below creates AWS EC2 instance from AMI, attaches a volume to it and assigns tags both to the instance and to the volume. During creation I attach to the instance a `SubnetId`, `ScurityGroups` and `UserData` which contains initialization script for the instance.

First code example is using `boto3` `ec2 client` which is low level object to communicate with AWS. It's more reliable and convenient way to create resources on AWS. You have more control on what you do.

The second code example creating exactly the same resources, but using the `ec2 resource` object, which provides a higher level abstraction.

Python3 code with ec2 client:

```
import boto3
from time import sleep

REGION = 'us-east-1'
AMI_IMAGE_ID = 'ami-0fac5486e4cff37f4'
INSTANCE_TYPE = 'c5.xlarge'
DISK_SIZE_GB = 200
DEVICE_NAME = '/dev/xvda'
NAME = 'codeflex-ec2'
OWNER = 'codeflex'
RUNID = 'ec2-1'
SUBNET_ID = 'subnet-02cd0004db6df93fa'
SECURITY_GROUPS_IDS = ['sg-067401826623ccbad']
PUBLIC_IP = None
ROLE_PROFILE = 'ec2-creator-role'

USERDATA_SCRIPT = '''
# Install awscli
curl -s "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"
unzip -q /tmp/awscliv2.zip -d /tmp && /tmp/aws/install

# Install Python38
amazon-linux-extras install python3.8 -y

# Install Docker on ec2 instance:
yum update -y && \
yum -y install docker && \
service docker start && \
usermod -a -G docker ec2-user
'''

def create_ec2_client():
    print("====================")
    print("Attempting to create ec2 client on region: %s" % REGION)
    session = boto3.Session(region_name=REGION, profile_name='gas')
    # session = boto3.Session(region_name=REGION)
    ec2_client = session.client('ec2')
    return ec2_client




def create_ec2_instance_with_tags():

    ec2_client = create_ec2_client()

    blockDeviceMappings = [
        {
            'DeviceName': DEVICE_NAME,
            'Ebs': {
                'DeleteOnTermination': True,
```

```
            'VolumeSize': DISK_SIZE_GB,
            'VolumeType': 'gp2'
        }
    },
]
iamInstanceProfile = {
    'Name': ROLE_PROFILE
}

# Create Elastic/Public IP for instance
if PUBLIC_IP:
    networkInterfaces = [
        {
            'DeviceIndex': 0,
            'SubnetId': SUBNET_ID,
            'Groups': [SECURITY_GROUPS_IDS],
            'AssociatePublicIpAddress': True,
            'DeleteOnTermination': True
        }, ]
    response = ec2_client.run_instances(ImageId=AMI_IMAGE_ID,
                                        InstanceType=INSTANCE_TYPE,
                                        NetworkInterfaces=networkInterfaces,
                                        UserData=USERDATA_SCRIPT,
                                        IamInstanceProfile=iamInstanceProfile,
                                        MinCount=1, MaxCount=1,
                                        BlockDeviceMappings=blockDeviceMappings,
                                        TagSpecifications=[
                                            {
                                                'ResourceType': 'instance',
                                                'Tags': [
                                                    {
                                                        'Key': 'Name',
                                                        'Value': NAME
                                                    },
                                                    {
                                                        'Key': 'Owner',
                                                        'Value': OWNER
                                                    },
                                                    {
                                                        'Key': 'RunId',
                                                        'Value': RUNID
                                                    }
                                                ]
                                            },
                                            {
                                                'ResourceType': 'volume',
                                                'Tags': [
                                                    {
                                                        'Key': 'Name',
                                                        'Value': NAME
                                                    },
                                                    {
                                                        'Key': 'Owner',
                                                        'Value': OWNER
                                                    },
                                                    {
                                                        'Key': 'RunId',
                                                        'Value': RUNID
                                                    }
                                                ]
                                            }
                                        ])
else:
    response = ec2_client.run_instances(ImageId=AMI_IMAGE_ID,
                                        InstanceType=INSTANCE_TYPE,
                                        SubnetId=SUBNET_ID,
                                        SecurityGroupIds=[SECURITY_GROUPS_IDS],
                                        UserData=USERDATA_SCRIPT,
                                        IamInstanceProfile=iamInstanceProfile,
                                        MinCount=1, MaxCount=1,
                                        BlockDeviceMappings=blockDeviceMappings,
                                        TagSpecifications=[
                                            {
                                                'ResourceType': 'instance',
                                                'Tags': [
                                                    {
                                                        'Key': 'Name',
                                                        'Value': NAME
                                                    },
                                                    {
```

```
                                                'Key': 'Owner',
                                                'Value': OWNER
                                            },
                                            {
                                                'Key': 'RunId',
                                                'Value': RUNID
                                            }
                                        ]
                                    },
                                    {
                                        'ResourceType': 'volume',
                                        'Tags': [
                                            {
                                                'Key': 'Name',
                                                'Value': NAME
                                            },
                                            {
                                                'Key': 'Owner',
                                                'Value': OWNER
                                            },
                                            {
                                                'Key': 'RunId',
                                                'Value': RUNID
                                            }
                                        ]
                                    }
                                ])
    if response['ResponseMetadata']['HTTPStatusCode'] == 200:
        instance_id = response['Instances'][0]['InstanceId']
        ec2_client.get_waiter('instance_running').wait(
            InstanceIds=[instance_id]
        )
        print('Success! instance:', instance_id, 'is created and running')
    else:
        print('Error! Failed to create instance!')
        raise Exception('Failed to create instance!')

    return instance_id


if __name__ == "__main__":
    create_ec2_instance_with_tags()
```

Python3 code with ec2 resource:

```
import boto3
from time import sleep

REGION = 'us-east-1'
AMI_IMAGE_ID = 'ami-0fac5486e4cff37f4'
INSTANCE_TYPE = 'c5.xlarge'
DISK_SIZE_GB = 200
DEVICE_NAME = '/dev/xvda'
NAME = 'codeflex-ec2'
OWNER = 'codeflex'
RUNID = 'ec2-1'
SUBNET_ID = 'subnet-02cd0004db6df93fa'
SECURITY_GROUPS_IDS = ['sg-067401826623ccbad']
PUBLIC_IP = None
ROLE_PROFILE = 'ec2-creator-role'

USERDATA_SCRIPT = '''
# Install awscli
curl -s "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"
unzip -q /tmp/awscliv2.zip -d /tmp && /tmp/aws/install

# Install Python38
amazon-linux-extras install python3.8 -y

# Install Docker on ec2 instance:
yum update -y && \
yum -y install docker && \
service docker start && \
usermod -a -G docker ec2-user
'''


def create_ec2_resource():
```

```
    print("Attempting to create ec2 ressource on region: %s" % REGION)

    session = boto3.Session(region_name = REGION, profile_name='development')
    # session = boto3.Session(region_name=REGION)

    ec2 = session.resource('ec2')

    if ec2 is None:
        raise ConnectionError("Could not create ec2 resource! Check your connection or aws c

    return ec2


def assign_tags_to_instance(ec2, instance_id):
    print("Waiting for instance to be ready ...")
    sleep(7)
    print("Assigning tags to instance " + instance_id)

    ec2.create_tags(Resources=[instance_id], Tags=[{'Key': 'Name', 'Value': NAME},
                                                   {'Key': 'Owner', 'Value': OWNER},
                                                   {'Key': 'RunId', 'Value': RUNID}])
    print("Tags assigned to instance successfully!")


def assign_tags_to_volume(instance):

    volumes = instance.volumes.all()

    print("Waiting for volume to be attached ...")
    sleep(7)
    for volume in volumes:
        print("Assigning tags to volume %s" % (volume.id))
        volume.create_tags(Tags=[{'Key': 'Name', 'Value': NAME},
                                 {'Key': 'Owner', 'Value': OWNER},
                                 {'Key': 'RunId', 'Value': RUNID}])
        print("Tags applied to volume successfully!")


def launch_ec2_instance():

    ec2 = create_ec2_resource()

    blockDeviceMappings = [
        {
            'DeviceName': DEVICE_NAME,
            'Ebs': {
                'DeleteOnTermination': True,
                'VolumeSize': DISK_SIZE_GB,
                'VolumeType': 'gp2'
            }
        },
    ]
    iamInstanceProfile = {
        'Name': ROLE_PROFILE
    }

    # Create Elastic/Public IP for instance
    if PUBLIC_IP:
        networkInterfaces = [
            {
                'DeviceIndex': 0,
                'SubnetId': SUBNET_ID,
                'Groups': [SECURITY_GROUPS_IDS],
                'AssociatePublicIpAddress': True,
                'DeleteOnTermination': True
            }, ]
        instance = ec2.create_instances(ImageId=AMI_IMAGE_ID,
                                        InstanceType=INSTANCE_TYPE,
                                        NetworkInterfaces=networkInterfaces,
                                        UserData=USERDATA_SCRIPT,
                                        IamInstanceProfile=iamInstanceProfile,
                                        MinCount=1, MaxCount=1,
                                        BlockDeviceMappings=blockDeviceMappings)
    else:
        instance = ec2.create_instances(ImageId=AMI_IMAGE_ID,
                                        InstanceType=INSTANCE_TYPE,
                                        SubnetId=SUBNET_ID,
                                        SecurityGroupIds=SECURITY_GROUPS_IDS,
                                        UserData=USERDATA_SCRIPT,
```

```
                                        IamInstanceProfile=iamInstanceProfile,
                                        MinCount=1, MaxCount=1,
                                        BlockDeviceMappings=blockDeviceMappings)
    if instance is None:
        raise Exception("Failed to create instance! Check the AWS console to verify creation

    print("Instance created and launched successfully!")
    print("#### Instance id: " + instance[0].id)

    assign_tags_to_instance(ec2, instance[0].id)
    assign_tags_to_volume(instance[0])

    return instance[0]


if __name__ == "__main__":
    launch_ec2_instance()
```

The program output:

```
Attempting to create ec2 ressource on region: us-east-1
Instance created and launched successfully!
#### Instance id: i-0fe2a5c51e1922fd7
Waiting for instance to be ready ...
Assigning tags to instance i-0fe2a5c51e1922fd7
Tags assigned to instance successfully!
Waiting for volume to be attached ...
Assigning tags to volume vol-084a9be9824b0efe8
Tags applied to volume successfully!
```

If you'll go to AWS Console you'll see that instance is running and volume has been attached to it and the tags are present.

The code has an option to create EC2 instance with public IP. In that case we need to create `NetworkInterfaces` dictionary and state `'AssociatePublicIpAddress': True`. Also you should move `SecurityGroups` and `SubnetId` to `NerworkInterfaces`.

`NetworkInterface` and `Volume` have `'DeleteOnTermination': True` flag which means that these resources will be automatically destroyed after you terminate your EC2 instance.