Route53, Elastic IP
Instance metadata and userdata

Route53 - Amazon Domain Name Service (DNS)

Typically, the DNS names are dynamically assigned to instances when launched. For example, if you create an instance in the us-east-1 region, your instance might have a DNS name such as **ec2-xx.us-east-1.compute.amazonaws.com.**

You'd rather use a user-friendly domain name, such as *www.cloudstudents.net*, instead of dynamically generated ec2 names (default behaviour) by AWS. Customer domain name resolves/maps to the ec2 name or dynamically assigned IP address associated with your instance.

To use a custom domain, you first register the domain name (*mycompany.net*) with a DNS service provider. When a domain is registered for you, you get not only the domain name itself, but also an entire set of subdomain names under it. For example, if you register **cloudstudents.net** as your company's domain name, you can create subdomain names such as: *blog.cloudstudents.net, downloads.cloudstudents.net* and so on. Company's domain name and its subdomain names together are called a **Zone**. Your reserved domain name, such as *cloudstudents.net*, is called Zone apex since it sits at the top of the zone's hierarchy.

To Register a custom domain name with AWS route53 service. login to AWS console and **select: Services->Route53 (aienergyservices.com)**
To register a new Domain, click: "*Get Started with Route 53"*. Search for available domain names. If it is available, you can register it by providing contact information. *Cost of registering a domain is: $5-20/year. AWS charges a nominal fee of hosting a domain ($0.50/month). See url: http://aws.amazon.com/route53/pricing/*
It takes 15-30 minutes for newly registered domains to become available and active. You can also transfer your existing domain from another provider, like *GoDaddy.com*, to AWS by clicking **"Transfer Domain"**. However, it may take a few days to transfer it to AWS, depending on the provider.

Once a new domain is registered and available, AWS will add it into the list of "Hosted Zones" and assign a unique Hosted Zone ID **"Z1JU4xxxxx"** and multiple *Name Servers*. Name Servers answer lookup requests of your domain name and map them to ec2 instance IP addresses. Amazon Route 53 stores resource records that you create within your domain inside the hosted zone.

" A hosted zone is an Amazon Route 53 concept that is similar to a zone file on a DNS name server. Like a zone file, a hosted zone contains information about your domain name, including the subdomain names within the domain and mappings between names and IP addresses"

One can use aws console, “aws route53 <command>” and “**cli53**” utility to interface with route53 service to perform tasks like:

- Create an Alias record (CNAME), *nflx.cloudstudents.net (change cloudperf to your domain name)* and map it to ec2 instance public address.
- Create an “A” record to associate Elastic IP Address (EIP: http://aws.amazon.com/articles/1346) to ec2 instance

Install and configure cli53 on vagrant VM:

You can use “*aws route53 <commands>*” to list and update route53 DNS records. You can also use the *“cli53”* program. You can download cli53 from url:

$ wget https://github.com/barnybug/cli53/releases/download/0.8.5/cli53-linux-amd64

Some sample use cases are listed here: https://github.com/barnybug/cli53

Install it:

$ sudo mv cli53-linux-amd64 /usr/bin/cli53

$ sudo chmod +x /usr/bin/cli53

List all domain names or zones in your account registered with route53

```
vagrant@cloudperf:~$ cli53 list
ID          Name            Record count        Comment
Z<..>  cloudstudents.net.          22              HostedZone created by Route53 Registrar
..
OR
vagrant@cloudperf:~$ aws route53 list-hosted-zones
```

List resource records of various types (A, CNAME, NS, MX..) in cloudstudents.net Zone

$ aws route53 list-resource-record-sets --hosted-zone-id ZXXXX

Create CNAME (alias) for your ec2 instance so that you can reach it using company’s domain name, *myserver.cloudstudents.net*

[*Please replace cloudstudents.net to your domain name*]

-Add “CNAME” myserver records and point to ec2 instance public hostname “ec2-xxx.compute.amazonaws.com”

$cli53 rrcreate domainname ‘**servername** <ttl> CNAME ec2-instance-name’

Example: $ cli53 rrcreate cloudstudents.net 'myserver 60 CNAME ec2-xxxxxx.compute-1.amazonaws.com**.**'

Sample output:

Created record: 'myserver.cloudstudents.net. 60 IN CNAME ec2-52-206-115-117.compute-1.amazonaws.com.cloudstudents.net.'

-Delete “CNAME” myserver records

$ cli53 rrdelete <Zone> **name** type

Example: $ cli53 rrdelete cloudstudents.net myserver.cloudstudents.net. CNAME

1 record sets deleted

Purpose of user-data script

http://alestic.com/2009/06/ec2-user-data-scripts

a *user-data*-file script helps with customizing instances at a launch or boot time. Some time, it is not practical to bake all the required software into the image, thus *user-data-file* script can be passed as an argument to $aws *ec2 run-instances* to perform custom tasks at instance launch time such as:

- Install additional software that is not in the base AMI
- Associate Elastic IP address (EIP) to new instance at boot and launch time.
- Register CNAME (Alias) resource record in route53 to point to new ec2 instance public address or name
- Access instance metadata or user data to perform instance, zone, region specific tasks
- Automate running tests and benchmarks etc..

What is instance metadata and user data:

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

AWS provides access to instance metadata and user data to manage and configure a running instance. While running on the ec2 instance, application or configuration script can find out: instance-id, public hostname, IP address, AMI-id using curl request to a url that serves instance configuration information.

NOTE: Run it from cloud instance

$curl http://169.254.169.254/latest/

Running following command on ec2 instance (ssh to ec2 instance first), will dump instance configuration information:

$ curl http://169.254.169.254/latest/meta-data/

Sample output:

ami-id
block-device-mapping/
hostname
instance-id
instance-type
local-hostname
local-ipv4
public-hostname

$ curl http://169.254.169.254/latest/meta-data/public-hostname

Sample output: xxxx.us-west-1.compute.amazonaws.com

What is Elastic IP

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

Elastic IP allows you to associate a fixed public IP address instead of CNAME to instance. This allows EC2 instances to always have the same public IP address and thus Clients can reach the service running on the EC2 instance with the same IP address.

To work with Elastic IP, you need to first create one. login to aws console. In EC2 Dashboard:

- click on **"Elastic IPs"**.
- click "**Allocate new address"**

A new IP Address is allocated for you to use: 52.x.x.x, 35.x.x.x...

If you have an instance running, you can associate the EIP address to this instance dynamically via aws console or aws cli on Vagrant VM:

$ aws ec2 associate-address --instance-id <> --public-ip <> --region us-east-1

You should see instance public DNS name is changed to:

ec2.<EIP ADDRESS>.us-east-1.compute.amazonaws.com

From randomly generated ec2.x.x.x name

Now your clients can reach the ec2 instance using:

ec2-<EIP ADDRESS>-us-east-1.compute.amazonaws.com

Also you can ssh to instance:

$ ssh -i <my.pem> ec2<EIP ADDRESS>.us.east.1…

If you don't want to use ec2.<EIP ADDRESS>.us-east-1…You can create an "A" record in your route53 managed ZONE and point it to the EIP address.

Go to Route53 and associate the name to this IP address: In aws console:

- *click on "Route53"*
- *click on "Hosted Zones". This should list all registered domains*
- *click on "domain name" that you like to edit*
- *click "Create Record Set"*
    - *Name: webserver.*cloudstudents.net
    - *Type: A - IPv4 address*
    - *Alias: NO*
    - *TTL: default*
    - *Value:* 52.22.150.221
    - *click save*

Command Line:

$cli53 rrcreate cloudstudents.net 'webserver 30 A <>'

$cli53 rrdelete cloudstudents.net webserver.cloudstudents.net. A

Check if records are removed

$ aws route53 list-resource-record-sets --hosted-zone-id Z1JU4AP9BMG9PO

Disassociate EIP from instance:

$ aws ec2 disassociate-address --public-ip <>
Release EIP from AWS console

LABS

LAB1: Launching instance with user-data-file script

Objective is to Install **firefox** package at instance launch time. To perform this task, we create a file “install-firefox.sh” and pass it at instance launch time as a **user-data-file** script. Create file **install-firefox.txt:**

```
#!/bin/bash
set -e -x
# http://www.microhowto.info/howto/perform_an_unattended_installation_of_a_debian_package.html
export DEBIAN_FRONTEND=noninteractive
apt-get update && apt-get upgrade -y
#apt-get install -q -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" firefox
apt-get install -y firefox
```

<SAVE CHANGES>

Now launch an instance by providing additional argument: **--user-data-file**

$ aws ec2 run-instances --key-name cloudperf-netflix --security-groups UCSC --count 1 --instance-type t2.micro --region us-east-1 --image-id < > --user-data file://install-firefox.sh

ssh to the new instance. Confirm the package is installed successfully.

$ curl http://169.254.169.254/latest/user-data
Sample output:

```
#!/bin/bash
set -e -x
# http://www.microhowto.info/howto/perform_an_unattended_installation_of_a_debian_package.html
export DEBIAN_FRONTEND=noninteractive
apt-get update && apt-get upgrade -y
#apt-get install -q -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" firefox
apt-get install -y firefox
```

$dpkg --list |grep -i firefox

LAB 2 : Self register ec2 instance by updating CNAME record in route53

One can use a combination of instance metadata *“instance public IP”* and *“user-data-file”* scripts to self register ec2 instance public IP address into route53 at launch time. Create a script, name it “*register-route53.sh*”:

```
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Download and install cli53 tool. We need it to update route53 record
wget https://github.com/barnybug/cli53/releases/download/0.8.5/cli53-linux-amd64
sudo mv cli53-linux-amd64 /usr/bin/cli53
sudo chmod +x /usr/bin/cli53
```

```
ZONE="cloudstudents.net"                    # cloudstudents.net
MYNAME="route53lab"                         # nflx
EC2_NAME=`/usr/bin/curl -s http://169.254.169.254/latest/meta-data/public-hostname| cut -d ' ' -f 2`
# Append dot to make it a fully qualified name to avoid getting domainname appended
FQN="$EC2_NAME."

# Search for this string in /var/log/syslog file to see if it worked
logger "ROUTE53: Setting DNS CNAME $MYNAME.$ZONE for $FQN_EC2_NAME"

# Create a new CNAME record on Route 53, replacing the old entry if necessary.
# CNAME myserver is created pointing to an ec2 instance public address.
# Make sure to have a dot at the end to make it fully qualified name
/usr/bin/cli53 rrcreate --replace $ZONE "$MYNAME 60 CNAME $FQN"

<SAVE CHANGES>
```

NOTE: You need to update Instance profile to allow "cli53" program running on ec2 instance to update Amazon route53 service, as we have discussed in the cloud-setup lab. Policy attached to instance profile only allows S3 bucket "list" and "Get" operation. Let's remove the old instance profile and associated Role and create a new one with update permission to allow route53 and other actions

Create a new Instance profile. ssh to vagrant VM and create two files:

EC2-Trust.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

EC2-Permission.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*",
        "route53:ListHostedZonesByName",
        "route53:ListResourceRecordSets",
        "route53:ChangeResourceRecordSets",
        "route53:GetHostedZone",
        "route53:GetChange",
```

```
        "ec2:AssociateAddress",
        "ec2:DescribeAddresses",
        "ec2:DisassociateAddress"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Create an instance profile

$aws iam create-instance-profile --instance-profile-name cloudstudents-DNS-ACCESS
Sample output:
INSTANCEPROFILE arn:aws:iam::442122186855:instance-profile/cloudstudents-DNS-ACCESS
2018-10-16T23:47:53Z AIPAI5N3DPN6QG7UDGTAC cloudstudents-xxx-Profile /

Create a role and attach a policy document to it.

$aws iam create-role --role-name cloudstudents-DNS-ACCESS --assume-role-policy-document **file://EC2-Trust.json**
Sample output:
ROLE arn:aws:iam::442122186855:role/cloudstudents-DNS-Profile 2018-10-16T23:48:37Z /
AROAJCXPT3QSJ7QRZQ6Zcloudstudents-DNS-Profile
ASSUMEROLEPOLICYDOCUMENT 2012-10-17
STATEMENT sts:AssumeRole Allow
PRINCIPAL ec2.amazonaws.com

Attach EC2-Permissions policy to role

$aws iam put-role-policy --role-name cloudstudents-DNS-ACCESS --policy-name cloudstudents-DNS-ACCESS --policy-document **file://EC2-Permission.json**
<no output>

Associate role to instance profile

$aws iam add-role-to-instance-profile --instance-profile-name cloudstudents-DNS-ACCESS --role-name cloudstudents-DNS-ACCESS
<no output>

When you launch instance with new instance profile, you are allowing application "*cli53*" running on EC2 instance to create/update route53 records

Now you are ready to launch an instance. Additional options:

" --user-data register-route53.sh"
"--iam-instance-profile Name=cloudstudents-DNS-Profile"

$aws ec2 run-instances --key-name cloudperf-netflix --security-groups UCSC --count 1 --instance-type t2.micro --region us-east-1 --image-id *<AMI-ID>* --user-data file://register-route53.sh --iam-instance-profile Name=cloudstudents-DNS-ACCESS

Sample output:
442122186855 r-0cb6af735929a8daf
INSTANCES 0 x86_64 False xen ami-eb4520fd i-05f92708ff16923f7t2.micro

cloudperf-netflix 2018-10-17T00:00:21.000Z ip-172-31-11-245.ec2.internal 172.31.11.245 /dev/sda1 ebs
True subnet-a241b6fb hvm vpc-eb69dd8e
IAMINSTANCEPROFILE arn:aws:iam::442122186855:instance-profile/cloudstudents-DNS-Profile
AIPAI5N3DPN6QG7UDGTAC
MONITORING disabled
NETWORKINTERFACES 0e:50:bb:6d:1e:a0 eni-09ef93a9560441e14 442122186855
ip-172-31-11-245.ec2.internal 172.31.11.245 True in-use subnet-a241b6fb vpc-eb69dd8e
ATTACHMENT 2018-10-17T00:00:21.000Z eni-attach-0da7c2efa6aa15f1e True 0 attaching
GROUPS sg-75f1ec0a UCSC
PRIVATEIPADDRESSES True ip-172-31-11-245.ec2.internal 172.31.11.245
PLACEMENT us-east-1c default
SECURITYGROUPS sg-75f1ec0a UCSC
STATE 0 pending
STATEREASON pending pending

Confirm that new resource record is created with $MYNAME and CNAME is pointing to new instance public IP $EC2_NAME
$ aws route53 list-resource-record-sets --hosted-zone-id **Z1JU4AP9BMG9PO**

You can also use AWS console and look for new CNAME record created in your managed zone under *Route53*

Once the instance is up and running, you should be able to use CNAME “route53lab” to ssh to new instance:
$ ssh -i <my.pem> ubuntu@route53lab.cloudstudents.net

LAB 3: Associate EIP to instance automatically at instance launch time.
Instead of manually associating EIP to a new instance, one can automate it using instance metadata to associate EIP to a new instance when the instance is booted. Create a script, ***associate-eip.sh***

```
---------
#!/bin/bash
export DEBIAN_FRONTEND=noninteractive
# install awscli package. We need it to associate EIP to instance
apt-get update
apt-get install -y awscli

# Make sure to first allocate EIP using AWS console or awscli in your account for ec2 instance to use.
EIP_ADDR="52.x.x.x"          # insert EIP address that you have allocated
INST_ID=`curl -s http://169.254.169.254/latest/meta-data/instance-id| cut -d ' ' -f 2 `

logger "Assigning EIP $EIP_ADDR  to instance $INST_ID"
/usr/bin/aws ec2 associate-address --instance-id $INST_ID --public-ip $EIP_ADDR --region us-east-1

<SAVE CHANGES>
```

Make sure you have allocated Elastic IP address earlier by logging into aws console ec2 dashboard. Also create an “A” record in route53 for this EIP address. Now launch a new instance with --user-data-file <script> options.

$aws ec2 run-instances --key-name cloudperf-netflix --security-groups UCSC --count 1 --instance-type t2.micro --region us-east-1 --image-id ami-eb4520fd --user-data file://associate-eip.sh --iam-instance-profile Name=cloudstudents-DNS-ACCESS

Confirm that you can ssh to instance by using EIP. You should also notice instance ec2 public name is also changed to ec2.eip.xx.xx

$ ssh -i your.pem ubuntu@eip-address

NOTE: You can also self register ec2 instances via rc script. However for that you need to bake the script into AMI. First update AMI by copying and installing the script and then create a new AMI, as described in cloudsetup instructions. For example:

$ sudo cp register-route53.sh /etc/init.d/route53register
$ sudo chmod +x /etc/init.d/route53register
Add script to default run level
$ sudo update-rc.d route53register defaults
To manually test it:
$ sudo service route53register start

Clean up:

Remove cloudstudents-DNS-ACCESS role from cloudstudents-DNS-ACCESS

$aws iam remove-role-from-instance-profile --instance-profile-name cloudstudents-DNS-ACCESS --role-name cloudstudents-DNS-ACCESS

Delete EC2-Permissions policy associated with EC2-Role

$aws iam delete-role-policy --role-name cloudstudents-DNS-ACCESS --policy-name cloudstudents-DNS-ACCESS