

EC2

<http://blog.assaydepot.com/ec2/sdruby-ec2>

By: Christopher Petersen

Table of Contents

Setup	3
Amazon command line tools.....	3
Amazon-EC2 Gem.....	3
Keypairs	4
Describe	4
Create.....	4
Delete.....	4
Security Groups	5
Describe	5
Create.....	5
Delete.....	5
Authorize.....	6
Revoke.....	7
Images	8
Describe	8
Register	9
Deregister.....	10
Run.....	11
Run.....	12
Instances.....	13
Describe	13
Reboot	13
Kill.....	14
Availability Zones	15
Describe	15
Elastic IPs	16
Describe	16
Allocate	17
Release	17
Associate.....	18
Dissociate	18
Persistent Volumes	19
Describe	19
Create.....	19
Delete.....	19
Attach.....	20
Detach.....	20

Setup

Amazon command line tools

1. Sign up at <http://aws.amazon.com>
2. Download the AWS Command Line Tools from <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351>
3. You will need to unzip the tools to a directory.
4. I recommend adding the directory to your path.
5. Add the file names of your Amazon private key file and certificate file to your environment as EC2_PRIVATE_KEY and EC2_CERT respectively.

You can use the ec2 command line tools without adding your key and cert to the environment. However, you must specify them for every command using the `-K` and `-C` options.

Amazon-EC2 Gem

First install the gem.

```
sudo gem install amazon-ec2
```

Next require it in your code

```
require 'ec2'
```

Finally, configure the EC2::Base object to work with your account.

```
ACCESS_KEY_ID = '--YOUR AWS ACCESS KEY ID--'  
SECRET_ACCESS_KEY = '--YOUR AWS SECRET ACCESS KEY--'  
@ec2 = EC2::Base.new(:access_key_id => ACCESS_KEY_ID,  
                   :secret_access_key => SECRET_ACCESS_KEY)
```

Note:

From here on, I will use the

```
@ec2
```

variable to mean an instance of

```
EC2::Base
```

initialized with your keys.

Keypairs

Keypairs are public private keypairs you will use to log into your instances. When you start an instance, you specify by name, which keypair to use, and you log into that instance using the private key you received when you created that keypair.

EC2 allows you to describe, add and delete keypairs.

Describe

Describing keypairs will list all of the keypairs associated with your account.

Command Line Tools

```
ec2-describe-keypairs  
ec2dkey
```

Amazon-EC2 Gem

```
@ec2.describe_keypairs
```

Create

Create adds a keypair with the name KEY to your account. You can take the output of the command line tool as an SSH key for connecting to instances started with that keypair

Command Line Tools

```
ec2-add-keypair KEY  
ec2addkey KEY
```

Amazon-EC2 Gem

```
@ec2.create_keypairs( :key_name => KEY)
```

Delete

Delete permanently removes a keypair.

Command Line Tools

```
ec2-delete-keypair KEY  
ec2delkey KEY
```

Amazon-EC2 Gem

```
@ec2.delete_keypairs( :key_name => KEY)
```

Security Groups

Security Groups are a way of grouping your instances together and protecting them. In many respects they act like a firewall, you can specify which ports you want open, what protocols are allowed and what subnets are allowed to communicate with the instances in your group.

EC2 allows you to describe, add, delete, authorize and revoke groups.

Describe

Describing groups will list all of the security groups associated with your account. It will also list all the permissions that apply to those groups. Permissions are granted and revoked with the authorize and revoke commands.

Command Line Tools

```
ec2-describe-groups
ec2dgrp
```

Amazon-EC2 Gem

```
@ec2.describe_security_groups
```

Create

Create adds a security group with the name GROUP to your account. You may also specify a GROUP-DESCRIPTION.

Command Line Tools

```
ec2-add-group GROUP -d GROUP-DESCRIPTION
ec2addgrp GROUP -d GROUP-DESCRIPTION
```

Amazon-EC2 Gem

```
@ec2.add_security_group ( :group_name => GROUP, :group_description =>
                          GROUP-DESCRIPTION)
```

Delete

Delete permanently removes a security group. You may not remove a security group that has instances.

Command Line Tools

```
ec2-delete-group GROUP
ec2delgrp GROUP
```

Amazon-EC2 Gem

```
@ec2.delete_security_group( :group_name => GROUP)
```

Security Groups – Authorize

The meat of security groups is the authorize and revoke commands. This is how you allow and restrict access to your instances.

Authorize

The authorize command allows you to specify which ports are accessible using specific protocols from which subnets. You can also specify this other security groups can access this security group since by default security groups cannot access each other.

Command Line Tools

```
ec2-authorize GROUP
ec2auth GROUP
    -P, --protocol PROTOCOL tcp, udp or icmp (must be lower case).
    -p, --port-range PORT-RANGE Range of ports to open.
    -t, --icmp-type-code TYPE:CODE icmp type and code.
    -o, --source-group SOURCE-GROUP Specifies another EC2 security
group.
    -u, --source-group-user SOURCE-GROUP-USER Not sure what this
does.
    -s, --source-subnet SOURCE-SUBNET The subnet to allow traffic
from.
```

Amazon-EC2 Gem

```
@ec2.authorize_security_group_ingress (
  :group_name => require, same as GROUP
  :ip_protocol => same as --protocol
  :from_port => the port where the request originated.
  :to_port => same as --port-range
  :cidr_ip => same as --source-subnet
  :source_security_group_name => same as --source-group
  :source_security_group_owner_id => same as --source-group-user
)
```

Security Groups – Revoke

The meat of security groups is the authorize and revoke commands. This is how you allow and restrict access to your instances.

Revoke

The revoke command works almost exactly like the authorize command. It allows you to revoke any of the permissions your authorized previously.

Command Line Tools

```
ec2-revoke GROUP
ec2revoke GROUP
    -P, --protocol PROTOCOL tcp, udp or icmp (must be lower case).
    -p, --port-range PORT-RANGE Range of ports to open.
    -t, --icmp-type-code TYPE:CODE icmp type and code.
    -o, --source-group SOURCE-GROUP Specifies another EC2 security
group.
    -u, --source-group-user SOURCE-GROUP-USER Not sure what this
does.
    -s, --source-subnet SOURCE-SUBNET The subnet to allow traffic
from.
```

Amazon-EC2 Gem

```
@ec2.revoke_security_group_ingress(
  :group_name => require, same as GROUP
  :ip_protocol => same as --protocol
  :from_port => the port where the request originated.
  :to_port => same as --port-range
  :cidr_ip => same as --source-subnet
  :source_security_group_name => same as --source-group
  :source_security_group_owner_id => same as --source-group-user
)
```

Images

Images are already configured operating systems packaged up and ready to run.

EC2 allows you to describe, register, deregister and run existing images. In order to register an image, you must have an already bundled image. Bundling image is outside of the scope of this document, but you may learn more at my blog:

<http://blog.assaydepot.com/create-your-own-ec2-image>

Images are specified by an image id in the form of *ami-XXXXXXXX*.

Describe

Describing images by default will list all of the images associated with your account. You may also specify an owner, which will list all of the images owned by that account that you have access to. Finally, you may specify “all” which will list all of the images from any owner (including yourself) that you have access to.

Command Line Tools

```
ec2-describe-images
```

```
ec2dim
```

```
-a, --all lists all the images you have access to.
```

```
-o, --owner USER lists all the public AMIs owned by the specified user that you have access to.
```

```
-x, --executable-by USER lists AMIs that USER has explicit launch permissions for (self returns all the AMIs you have explicit permissions for, all returns AMIs you have implicit or explicit launch permissions for, user_id returns AMIs that user has explicit launch permissions for.
```

Amazon-EC2 Gem

```
@ec2.describe_images (
```

```
  :image_id => an array of the images you want to describe
```

```
  :owner_id => an array of the owners whose images you want to describe
```

```
  :executable_by => an array of users for which you want to see the AMIs that they have launch permissions for (only returns AMIs for which you have permission as well).
```

```
)
```

Images – Register

If none of the prepackaged AMIs provided by Amazon or others meets your needs, or if you need to customize an existing image, you can bundle your own. The actual bundling is outside of the scope of this document, you can learn more about it on my blog:

<http://blog.assaydepot.com/create-your-own-ec2-image>

However, if you already have a bundled image uploaded to an S3 bucket, you can use the register command to get your own AMI.

Register

Register takes the manifest file for a bundled image already on S3 and creates an AMI number.

Command Line Tools

```
ec2-register MANIFEST
```

```
ec2reg MANIFEST
```

Where MANIFEST is a manifest xml file already uploaded to S3 (with the rest of the bundle). For example:

```
ec2-register -K ~/$KEY_FILE_NAME  
              -C ~/$CERT_FILE_NAME  
              my_bucket_name/image.manifest.xml
```

Amazon-EC2 Gem

```
@ec2.register_image ( :image_location => location of the image. )
```

Images – Deregister

Deregister

Deregistration allows you to get rid of an AMI. NOTE, it does not remove the AMI from S3, you have to do that your self. However, you should deregister an AMI before removing its image form S3 and before making a change to an existing image. Any change to an image with an existing AMI invalidates that AMI.

Command Line Tools

```
ec2-deregister AMI
```

```
ec2dereg AMI
```

Amazon-EC2 Gem

```
@ec2.deregister_image ( :image_id => AMI to deregister. )
```

Images – Run

Run

Running an image results in an instance.

Command Line Tools

`ec2-run-instances AMI`

`ec2run AMI`

`-B, --block-device-mapping MAPPING` Defines native device names to use when exposing virtual devices.

`-n, --instance-count MIN[-MAX]` The number of instances to attempt to launch. May be specified as a single integer or as a range (min-max).

`-g, --group GROUP` Specifies the security group within which the instance(s) should be run.

`-k, --key KEY-PAIR` Specifies the key pair to use when launching this instance.

`-d, --user-data DATA` Specifies the user data to be made available to the instances in this reservation.

`-f, --user-data-file DATA-FILE` Specifies the file containing user data to be made available to the instances in this reservation.

`--addressing ADDRESSING` Specifies the addressing type to use for the instances.

`-t, --instance-type TYPE` can be `m1.small`, `m1.large`, `m1.xlarge`

`-z, --availability-zone ZONE` Specifies the availability zone to launch the instances in.

`--kernel KERNEL-ID` Specifies the kernel-ID of the kernel to launch instances with.

`--ramdisk RAMDISK-ID` Specifies the ramdisk-ID of the ramdisk to launch instances with.

Images – Run (cont.)

Run

Running an image results in an instance.

Amazon-EC2 Gem

```
@ec2.run_instances (  
  :image_id => AMI to run.  
  :min_count => minimum number of instances to run, defaults to 1  
  :max_count => maximum number of instances to run, defaults to 1  
  :key_name => name of keypair to use  
  :group_id => name or array of names of security groups to use  
  :user_data => String (default : nil)  
  :addressing_type => String (default : "public")  
  :instance_type => m1.small, m1.large or m1.xlarge  
  :base64_encoded => true or false, defaults to false  
)
```

Instances

Instances are the basic unit of computing in the cloud, each instance represents a computer you are running and you are charged hourly based on the number and type of instances you are running. Instances are specified by instance ids, which take the form *i-XXXXXXXX*.

EC2 allows you to describe, reboot and kill your instances.

Describe

Describing instances will list all of the running or recently terminated instances associated with your account.

Command Line Tools

```
ec2-describe-instances INSTANCE
ecdin INSTANCE
```

If one or more instances is specified, only information about those instances is returned. By default it returns information about all instances.

Amazon-EC2 Gem

```
@ec2.describe_instances(
  :instance_id => Optional, if specified will return information
  about the instance or instances if an array is specified
)
```

Reboot

Rebooting instances will (as the name implies) reboot instances. When you reboot an instance all of the data stored on that instance is *retained*.

Command Line Tools

```
ec2-reboot-instances INSTANCE
ecreboot INSTANCE
```

One or more instances must be specified.

Amazon-EC2 Gem

```
@ec2.reboot_instances(
  :instance_id => Array, reboots every instance specified
)
```

Instances (cont.)

Kill

Kill instances will (as the name implies) kill instances. When you kill an instance all of the data stored on that instance is *lost*.

Command Line Tools

```
ec2-terminate-instances INSTANCE
```

```
eckill INSTANCE
```

One or more instances must be specified.

Amazon-EC2 Gem

```
@ec2.terminate-instances (  
  :instance_id => Array, kills every instance specified  
)
```

Availability Zones

Availability zones allow you to start instances in geographically distinct areas protecting your application from failures at the datacenter or region level.

EC2 allows you describe availability zones. In order to make use of an availability zone, specify the `-z` option when running an instance.

Describe

Describing availability zones will list all of the availability zones exposed by Amazon. Currently they are only exposing 3 zones in one region. I expect this to increase over time.

Command Line Tools

```
ec2-describe-availability-zones  
ec2daz
```

Amazon-EC2 Gem

Not yet supported.

Elastic IPs

Since EC2 instances are ephemeral, they get a new hostname every time you start one. These host names also get an IP address, but Amazon recommends against using that IP. This can pose a problem when trying to configure a domain name to point to your server. To overcome this issue, Amazon introduced Elastic IPs, which are fixed IP addresses associated with your account that can be applied to any instance.

Elastic IPs can be used in conjunction with Availability Zones. If there is a problem with an instance in one availability zone, you can associate the IP with an instance in another availability zone and proceed with minimal downtime.

EC2 allows you to describe, allocate, release, associate and dissociate Elastic IPs.

Describe

Describing addresses will list all of the Elastic IPs associated with your account.

Command Line Tools

```
ec2-describe-addresses IP
ec2daddr IP
```

If IP is specified this returns the information about that IP, otherwise it returns information about all IP associated with your account.

Amazon-EC2 Gem

```
@ec2.describe_addresses(
  :public_ip => Array, Optional, if specified will return
  information about the IPs specified, otherwise returns information about
  all IPs associated with your account
)
```

Elastic IPs – Allocate and Release

Allocate

Allocate is how you allocate an Elastic IP for use by your account. Currently you can allocate up to 5 IPs. If you need more than 5, you should contact AWS support. NOTE, you will be charged \$0.01 per hour per IP that you are **NOT** using. This is to try and prevent abuse of this service. IPs are free when they are associated to a running instance.

Command Line Tools

```
ec2-allocate-address  
ec2allocaddr
```

Amazon-EC2 Gem

```
@ec2.allocate_address()
```

Release

Releasing an IP that is associated to an instance, automatically dissociates the IP from the instance.

Command Line Tools

```
ec2-release-address IP  
ec2reladdr IP
```

IP must be specified and must refer to an IP associated with your account.

Amazon-EC2 Gem

```
@ec2.release_address(  
  :public_ip => The IP to release  
)
```

Elastic IPs – Associate & Dissociate

Associate

Associating an IP address with an instance, allows you to reference the instance using that IP address.

Command Line Tools

```
ec2-associate-address IP -i INSTANCE
```

```
ec2assocaddr IP -i INSTANCE
```

IP must be an Elastic IP already allocated for your account, and INSTANCE must be the id instance of a running instance associated with your account.

Amazon-EC2 Gem

```
@ec2.associate_address (  
  :instance_id => ID of a running instance  
  :public_ip => One of your Elastic IPs  
)
```

Dissociate

Dissociating an IP address from an instance frees that IP to be assigned to another instance.

Command Line Tools

```
ec2-disassociate-address IP
```

```
ec2disaddr IP
```

IP must be an Elastic IP already allocated for your account

Amazon-EC2 Gem

```
@ec2.disassociate_address (  
  :public_ip => The Elastic IP to dissociate  
)
```

Persistent Volumes

Much like Elastic IPs were introduced to deal with the ephemeral nature of EC2 host names, Persistent Volumes are being introduced to deal with the ephemeral nature of storage on EC2. Persistent Volumes are basically virtual disks that are designed for long term storage and associated with your account. You can create & attach them to your instances and even perform snapshot backups of the volume S3. Volumes are identified by a volume id in the form of **vol-XXXXXXXX**.

Persistent Volumes are not yet available, but should be in beta soon. You can learn more about them on the Amazon Web Services Blog:

<http://aws.typepad.com/aws/2008/04/block-to-the-fu.html>

So far we know you will be able to describe, create, delete, attach, detach (presumably) and create snapshot of volumes.

Since volumes are not yet available, none of these features are supported by the Amazon-EC2 Gem... yet.

Describe

Describing volumes lists all of the volumes associated with your account.

```
ec2-describe-volumes
```

Create

Creating a volume creates a virtual disk of a certain size.

```
ec2-create-volume -s SIZE
```

Where SIZE is the desired size of the disk in bytes.

Delete

I am guessing at the functionality and syntax here, it has not yet been published.

Deleting a volume destroys the data on that volume and removes it from your account.

```
ec2-delete-volume VOLUME
```

Where VOLUME is the volume identifier of the volume you wish to delete.

Persistent Volumes – Attach & Detach

Once you have created your volumes, you must attach them to instances to access them.

Attach

Attaching a volume to an instance makes it accessible as a filesystem.

```
ec2-attach-volume VOLUME
    -i INSTANCE
    -d DEVICE
```

Where VOLUME is the volume identifier of the volume you want to attach. INSTANCE is the instance identifier of the instance you want to attach to and DEVICE is the device location where you want to attach (for instance /dev/sdb or /dev/sdc).

Detach

I am guessing at the functionality and syntax here, it has not yet been published.

Detaching a volume free it, allowing you to attach it to a different instance.

```
ec2-detach-volume VOLUME
```

Where VOLUME is the volume you want to detach.