Знакомство с AWS IAM

Введение

Для хранения снимков экрана, на данный момент мы используем monosnap), и последующего предоставления клиентам мы используем Amazon S3. Возникла, казалось бы, тривиальная задача - разграничить доступ к S3 хранилищу. Так как на момент написания статьи использовалась учетная запись с т.н. full access правами. Т.е. в любой момент времени пользователь мог удалить все данные, что нас конечно не устраивало. Но не все так просто, доступа к консоли AWS не было, а были доступны лишь Access Key Id и Secret Access Key. Но так как это была учетная запись с полными правами этого было достаточно. Итак, у нас есть 3 бакета и необходимо создать 3х пользователей и дать им права на запись в соответствующий бакет, но при этом они не должны иметь право что-либо удалять в своем бакете. Реализиовать данную задачу как раз и поможет IAM - **I**dentity and **A**ccess **M**anagement.

Подготовка системы

Итак в нашем распоряжении CentOS 6 со всеми установленными обновлениями и подключенными репозитариями EPEL/RPMForge/IUS Community

```
# cat /etc/redhat-release
CentOS release 6.6 (Final)

# uname -a
Linux staging.example.net 2.6.32-504.3.3.el6.x86_64 #1 SMP Wed Dec 17
01:55:02 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
```

Установка и настройка aws cli

Устанавливаем aws cli

```
# yum install python-pip python-devel gcc
```

В принципе gcc не обязательно ставить, но при его наличие, будут собираться модули, обеспечивающие ускорение работы.

```
# pip install awscli
Downloading/unpacking awscli
  Downloading awscli-1.6.10.tar.gz (282kB): 282kB downloaded
  Running setup.py egg_info for package awscli
Downloading/unpacking botocore>=0.80.0,<0.81.0 (from awscli)
  Downloading botocore-0.80.0.tar.gz (1.2MB): 1.2MB downloaded
  Running setup.py egg_info for package botocore
...
```

```
...
...
  Running setup.py install for six
    no previously-included directories found matching 'documentation/_build'
  Running setup.py install for pyasn1
Successfully installed awscli botocore bcdoc colorama docutils rsa argparse
jmespath python-dateutil ordereddict simplejson six pyasn1
Cleaning up...
```

Проверяем, что консоль установилась

```
# aws --version
aws-cli/1.6.10 Python/2.6.6 Linux/2.6.32-504.3.3.el6.x86_64
```

Производим настройку данных, вводим имеющиеся у нас Access Key Id и Secret Access Key.

```
# aws configure
AWS Access Key ID [None]: ICAMLORKAKIAJMHSTVUH
AWS Secret Access Key [None]: GsIS4ZSBNUllBEedAT8Qqb+SYCK0sChMaX66dQ8d
Default region name [None]:
Default output format [None]:
```

После выполнения этой команды в домашней папке пользователя должен появиться следующий файл

```
# cat ~/.aws/credentials
[default]
aws_access_key_id = ICAMLORKAKIAJMHSTVUH
aws_secret_access_key = GsIS4ZSBNUllBEedAT8Qqb+SYCK0sChMaX66dQ8d
```

Проверяем, что все работает корректно. Выводим список доступных бакетов

```
# aws s3 ls
2015-01-02 08:49:56 alpha01
2015-01-02 08:50:20 omega02
2015-01-02 08:52:25 sigma03
```

Создание IAM пользователей

Итак, на данный момент в нашем распоряжении всего один пользователь - root, с полными правами

```
# aws iam list-users
{
    "Users": [
        {
            "UserName": "root",
            "Path": "/",
```

```
            "CreateDate": "2014-12-03T09:27:58Z",
            "UserId": "AIDASOCRCJSKGU6VIDAQF",
            "Arn": "arn:aws:iam::911345934031:user/root"
        }
    ]
}
```

Создаем 3х пользователей для каждого из бакетов

```
# aws iam create-user --user-name scrnshots-alpha
{
    "User": {
        "UserName": "scrnshots-alpha",
        "Path": "/",
        "CreateDate": "2015-01-02T16:05:18.899Z",
        "UserId": "AIDAIEIRMQH6HKQ4I5PPA",
        "Arn": "arn:aws:iam::934091134531:user/scrnshots-alpha"
    }
}
```

```
# aws iam create-user --user-name scrnshots-omega
{
    "User": {
        "UserName": "scrnshots-omega",
        "Path": "/",
        "CreateDate": "2015-01-02T16:05:24.496Z",
        "UserId": "AIDAIDQCAINOCVH5DEQKI",
        "Arn": "arn:aws:iam::934091134531:user/scrnshots-omega"
    }
}
```

```
# aws iam create-user --user-name scrnshots-sigma
{
    "User": {
        "UserName": "scrnshots-sigma",
        "Path": "/",
        "CreateDate": "2015-01-02T16:05:34.284Z",
        "UserId": "AIDAJCSMV2RO52DR6GBO2",
        "Arn": "arn:aws:iam::934091134531:user/scrnshots-sigma"
    }
}
```

Создаем для каждого из пользователей Secret Access Key и Access Key Id

```
# aws iam create-access-key --user-name scrnshots-alpha
{
    "AccessKey": {
        "UserName": "scrnshots-alpha",
        "Status": "Active",
        "CreateDate": "2015-01-02T16:09:09.602Z",
        "SecretAccessKey": "eLCXqSZNkVtW/qfibBLd+9wZAKFnzJ7NeY7/8+LL",
```

```
        "AccessKeyId": "AKIAJY24GFHCXM3CTLAJ"
    }
}
```

```
# aws iam create-access-key --user-name scrnshots-omega
{
    "AccessKey": {
        "UserName": "scrnshots-omega",
        "Status": "Active",
        "CreateDate": "2015-01-02T16:09:38.630Z",
        "SecretAccessKey": "vyAwcMLS0LwU0Mmy9m0war1LPiJ3xhU1SCzkpp0D",
        "AccessKeyId": "AKIAIUQE5XIXV4OQPKQH"
    }
}
```

```
# aws iam create-access-key --user-name scrnshots-sigma
{
    "AccessKey": {
        "UserName": "scrnshots-sigma",
        "Status": "Active",
        "CreateDate": "2015-01-02T16:09:42.854Z",
        "SecretAccessKey": "g4kl7ZsGSfz1jVdNKuQEJca0rYzGw3ifuzUAkOeG",
        "AccessKeyId": "AKIAIH7QSYK2TRKMEQJN"
    }
}
```

Обратите внимение, что SecretAccessKey будет недоступен в консоли AWS, так что его необходимо обязательно сохранить!

Создание IAM политик

Теперь когда у нас есть пользователи можно назначить им политику. Для этого подготавливаем текстовый файл в json формате и присоединяем политику соответствующему пользователю. Как я уже говорил в начале статьи, нам необходимо дать возможность загружать и просматривать файлы в своем бакете, но при этом у пользователя не должно быть возможности просматривать данные из чужих бакетов, а так же удалять файлы из своего бакета.

```
# cat ~/aws/alpha01-policy.json
{
    "Statement":
    [
        {
             "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
             "Resource": "arn:aws:s3:::*",
             "Effect": "Allow"
        },
        {
             "Action": ["s3:ListBucket", "s3:GetBucketAcl"],
             "Resource": "arn:aws:s3:::*",
```

```
            "Effect": "Allow"
        },
        {
            "Action": ["s3:Put*", "s3:Get*"],
            "Resource": "arn:aws:s3:::alpha01/*",
            "Effect": "Allow"
        },
        {
            "Action": "s3:Del*",
            "Resource": "arn:aws:s3:::*/*",
            "Effect": "Deny"
        }
    ]
}
```

При создании IAM политик для S3 следует учитывать отличие следующих объектов:

- действия с сервисами, например **ListAllMyBuckets**
- действия с бакетами, например **ListBucket**
- действия с объектами, например **GetObject**

Поэтому при описании политик следует учитывать следующие моменты:

- действия с сервисами - **arn:aws:s3:::***
- действия с бакетами - **arn:aws:s3:::<bucket>**
- действия с объектами - **arn:aws:s3:::<bucket>/<object>**

Теперь подключаем политику к пользователю

```
# aws iam put-user-policy --user-name scrnshots-alpha --policy-name ALPHA01-POLICY --policy-document file://~/aws/alpha01-policy.json
```

Если все прошло успешно, то не будет никакого вывода. Теперь проверим, что у пользователя есть политика

```
# aws iam list-user-policies --user-name scrnshots-alpha
{
    "PolicyNames": [
        "ALPHA01-POLICY"
    ]
}
```

По аналогии подключаем политики и для 2х других пользователей

```
# aws iam put-user-policy --user-name scrnshots-omega --policy-name OMEGA02-POLICY --policy-document file://~/aws/omega02-policy.json
```

```
# aws iam put-user-policy --user-name scrnshots-sigma --policy-name SIGMA03-POLICY --policy-document file://~/aws/sigma03-policy.json
```

Настройка monosnap

Настройка тривиальная, и сводится к указанию Access Key Id, Secret Access Key, Bucket.