FREE eBook

LEARNING aws-cli

Free unaffiliated eBook created from
Stack Overflow contributors.

#aws-cli

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: aws-cli

It is an unofficial and free aws-cli ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official aws-cli.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with aws-cli

Remarks

Description

The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

The AWS CLI introduces a new set of simple file commands for efficient file transfers to and from Amazon S3.

Supported Services

For a list of the available services you can use with AWS Command Line Interface, see Available Services in the AWS CLI Command Reference.

AWS Command Line Interface on GitHub

You can view—and fork—the source code for the AWS CLI on GitHub in the https://github.com/aws/aws-cli project.

Versions

Version	Release Date
1.10.38	2016-06-14
1.10.35	2016-06-03
1.10.33	2016-05-25
1.10.30	2016-05-18

Examples

Installation and setup

There are a number of different ways to install the AWS CLI on your machine, depending on what operating system and environment you are using:

On Microsoft Windows – use the MSI installer. On Linux, OS X, or Unix – use pip (a package manager for Python software) or install manually with the bundled installer.

Install using pip:

You will need python to be installed (version 2, 2.6.5+,3 or 3.3+). Check with

```
python --version

pip --help
```

Given that both of these are installed, use the following command to install the aws cli.

```
sudo pip install awscli
```

Install on Windows The AWS CLI is supported on Microsoft Windows XP or later. For Windows users, the MSI installation package offers a familiar and convenient way to install the AWS CLI without installing any other prerequisites. Windows users should use the MSI installer unless they are already using pip for package management.

- MSI Installer for Windows 32-bit
- MSI Installer for Windows 64-bit

Run the downloaded MSI installer. Follow the instructions that appear.

To install the AWS CLI using the bundled installer

Prerequisites:

- Linux, OS X, or Unix
- Python 2 version 2.6.5+ or Python 3 version 3.3+

1. Download the AWS CLI Bundled Installer using wget or curl.
2. Unzip the package.
3. Run the install executable.

On Linux and OS X, here are the three commands that correspond to each step:

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
$ unzip awscli-bundle.zip
$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

Install using HomeBrew on OS X:

Another option for OS X

```
brew install awscli
```

Test the AWS CLI Installation

Confirm that the CLI is installed correctly by viewing the help file. Open a terminal, shell or command prompt, enter aws help and press Enter:

```
$ aws help
```

Configuring the AWS CLI

Once you have finished the installation you need to configure it. You'll need your access key and secret key that you get when you create your account on aws. You can also specify a default region name and a default output type (text|table|json).

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: ENTER
```

Updating the CLI tool

Amazon periodically releases new versions of the AWS Tool. If the tool was installed using the Python Pip tool the following command will check the remote repository for updates, and apply it to your local system.

```
$ pip install awscli --upgrade
```

Creating a New Profile

To setup a new credential profile with the name `myprofile`:

```
$ aws configure --profile myprofile
AWS Access Key ID [None]: ACCESSKEY
AWS Secret Access Key [None]: SECRETKEY
Default region name [None]: REGIONNAME
Default output format [None]: text | table | json
```

For the AWS access key id and secret, create an IAM user in the AWS console and generate keys for it.

Region will be the default region for commands in the format `eu-west-1` or `us-east-1`.

The default output format can either be `text`, `table` or `json`.

You can now use the profile name in other commands by using the `--profile` option, e.g.:

```
$ aws ec2 describe-instances --profile myprofile
```

AWS libraries for other languages (e.g. `aws-sdk` for Ruby or `boto3` for Python) have options to use the profile you create with this method too. E.g. creating a new session in `boto3` can be done like

this, `boto3.Session(profile_name:'myprofile')` and it will use the credentials you created for the profile.

The details of your aws-cli configuration can be found in `~/.aws/config` and `~/.aws/credentials` (on linux and mac-os). These details can be edited manually from there.

Using aws cli commands

The syntax for using the aws cli is as follows:

```
aws [options] <command> <subcommand> [parameters]
```

Some examples using the 'ec2' command and the 'describe-instances' subcommand:

```
aws ec2 describe-instances

aws ec2 describe-instances --instance-ids <your-id>
```

Example with a fake id:

```
aws ec2 describe-instances --instance-ids i-c71r246a
```

List S3 buckets

```
aws s3 ls
```

Use a named profile

```
aws --profile myprofile s3 ls
```

List all objects in a bucket, including objects in folders, with size in human-readable format and a summary of the buckets properties in the end -

```
aws s3 ls --recursive --summarize --human-readable s3://<bucket_name>/
```

AWS completer for Ubuntu with Bash

The following utility can be used for auto-completion of commands:

```
$ which aws_completer
/usr/bin/aws_completer

$ complete -C '/usr/bin/aws_completer' aws
```

For future shell sessions, consider add this to your ~/.bashrc

```
$ echo "complete -C '/usr/bin/aws_completer' aws" >> ~/.bashrc
```

To check, type:

```
$ aws ec
```

Press the [TAB] key, it should add 2 automatically:

```
$ aws ec2
```

AWS CLI Cheat sheet - List of All CLI commands

Setup

Install AWS CLI

AWS CLI is an common CLI tool for managing the AWS resources. With this single tool we can manage all the aws resources

```
sudo apt-get install -y python-dev python-pip
sudo pip install awscli
aws --version
aws configure
```

Bash one-liners

```
cat <file> # output a file
tee # split output into a file
cut -f 2 # print the 2nd column, per line
sed -n '5{p;q}' # print the 5th line in a file
sed 1d # print all lines, except the first
tail -n +2 # print all lines, starting on the 2nd
head -n 5 # print the first 5 lines
tail -n 5 # print the last 5 lines

expand # convert tabs to 4 spaces
unexpand -a # convert 4 spaces to tabs
wc # word count
tr ' ' \\t # translate / convert characters to other characters

sort # sort data
uniq # show only unique entries
paste # combine rows of text, by line
join # combine rows of text, by initial column value
```

Cloudtrail - Logging and Auditing

http://docs.aws.amazon.com/cli/latest/reference/cloudtrail/ 5 Trails total, with support for resource

level permissions

```
# list all trails
aws cloudtrail describe-trails

# list all S3 buckets
aws s3 ls

# create a new trail
aws cloudtrail create-subscription \
    --name awslog \
    --s3-new-bucket awslog2016

# list the names of all trails
aws cloudtrail describe-trails --output text | cut -f 8

# get the status of a trail
aws cloudtrail get-trail-status \
    --name awslog

# delete a trail
aws cloudtrail delete-trail \
    --name awslog

# delete the S3 bucket of a trail
aws s3 rb s3://awslog2016 --force

# add tags to a trail, up to 10 tags
aws cloudtrail add-tags \
    --resource-id awslog \
    --tags-list "Key=log-type,Value=all"

# list the tags of a trail
aws cloudtrail list-tags \
    --resource-id-list

# remove a tag from a trail
aws cloudtrail remove-tags \
    --resource-id awslog \
    --tags-list "Key=log-type,Value=all"
```

IAM

Users

https://blogs.aws.amazon.com/security/post/Tx15CIT22V4J8RP/How-to-rotate-access-keys-for-IAM-users http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_iam-limits.html Limits = 5000 users, 100 group, 250 roles, 2 access keys / user

http://docs.aws.amazon.com/cli/latest/reference/iam/index.html

```
# list all user's info
aws iam list-users

# list all user's usernames
aws iam list-users --output text | cut -f 6

# list current user's info
aws iam get-user

# list current user's access keys
aws iam list-access-keys

# crate new user
aws iam create-user \
    --user-name aws-admin2

# create multiple new users, from a file
allUsers=$(cat ./user-names.txt)
for userName in $allUsers; do
    aws iam create-user \
        --user-name $userName
done

# list all users
aws iam list-users --no-paginate

# get a specific user's info
aws iam get-user \
    --user-name aws-admin2

# delete one user
aws iam delete-user \
    --user-name aws-admin2

# delete all users
# allUsers=$(aws iam list-users --output text | cut -f 6);
allUsers=$(cat ./user-names.txt)
for userName in $allUsers; do
    aws iam delete-user \
        --user-name $userName
done
```

Password policy

http://docs.aws.amazon.com/cli/latest/reference/iam/

```
# list policy
# http://docs.aws.amazon.com/cli/latest/reference/iam/get-account-password-policy.html
aws iam get-account-password-policy

# set policy
# http://docs.aws.amazon.com/cli/latest/reference/iam/update-account-password-policy.html
aws iam update-account-password-policy \
    --minimum-password-length 12 \
    --require-symbols \
    --require-numbers \
    --require-uppercase-characters \
    --require-lowercase-characters \
    --allow-users-to-change-password
```

```
# delete policy
# http://docs.aws.amazon.com/cli/latest/reference/iam/delete-account-password-policy.html
aws iam delete-account-password-policy
```

Access Keys

http://docs.aws.amazon.com/cli/latest/reference/iam/

```
# list all access keys
aws iam list-access-keys

# list access keys of a specific user
aws iam list-access-keys \
    --user-name aws-admin2

# create a new access key
aws iam create-access-key \
    --user-name aws-admin2 \
    --output text | tee aws-admin2.txt

# list last access time of an access key
aws iam get-access-key-last-used \
    --access-key-id AKIAINA6AJZY4EXAMPLE

# deactivate an acccss key
aws iam update-access-key \
    --access-key-id AKIAI44QH8DHBEXAMPLE \
    --status Inactive \
    --user-name aws-admin2

# delete an access key
aws iam delete-access-key \
    --access-key-id AKIAI44QH8DHBEXAMPLE \
    --user-name aws-admin2
```

Groups, Policies, Managed Policies

http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
http://docs.aws.amazon.com/cli/latest/reference/iam/

```
# list all groups
aws iam list-groups

# create a group
aws iam create-group --group-name FullAdmins

# delete a group
aws iam delete-group \
    --group-name FullAdmins

# list all policies
aws iam list-policies

# get a specific policy
aws iam get-policy \
```

```
    --policy-arn <value>

# list all users, groups, and roles, for a given policy
aws iam list-entities-for-policy \
    --policy-arn <value>

# list policies, for a given group
aws iam list-attached-group-policies \
    --group-name FullAdmins

# add a policy to a group
aws iam attach-group-policy \
    --group-name FullAdmins \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

# add a user to a group
aws iam add-user-to-group \
    --group-name FullAdmins \
    --user-name aws-admin2

# list users, for a given group
aws iam get-group \
    --group-name FullAdmins

# list groups, for a given user
aws iam list-groups-for-user \
    --user-name aws-admin2

# remove a user from a group
aws iam remove-user-from-group \
    --group-name FullAdmins \
    --user-name aws-admin2

# remove a policy from a group
aws iam detach-group-policy \
    --group-name FullAdmins \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

# delete a group
aws iam delete-group \
    --group-name FullAdmins
```

EC2

keypairs

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

```
# list all keypairs
# http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-key-pairs.html
aws ec2 describe-key-pairs

# create a keypair
```

```
# http://docs.aws.amazon.com/cli/latest/reference/ec2/create-key-pair.html
aws ec2 create-key-pair \
    --key-name <value>

# create a new private / public keypair, using RSA 2048-bit
ssh-keygen -t rsa -b 2048

# import an existing keypair
# http://docs.aws.amazon.com/cli/latest/reference/ec2/import-key-pair.html
aws ec2 import-key-pair \
    --key-name keyname_test \
    --public-key-material file:///home/apollo/id_rsa.pub

# delete a keypair
# http://docs.aws.amazon.com/cli/latest/reference/ec2/delete-key-pair.html
aws ec2 delete-key-pair \
    --key-name <value>
```

Security Groups

http://docs.aws.amazon.com/cli/latest/reference/ec2/index.html

```
# list all security groups
aws ec2 describe-security-groups

# create a security group
aws ec2 create-security-group \
    --vpc-id vpc-1a2b3c4d \
    --group-name web-access \
    --description "web access"

# list details about a securty group
aws ec2 describe-security-groups \
    --group-id sg-0000000

# open port 80, for everyone
aws ec2 authorize-security-group-ingress \
    --group-id sg-0000000 \
    --protocol tcp \
    --port 80 \
    --cidr 0.0.0.0/24

# get my public ip
my_ip=$(dig +short myip.opendns.com @resolver1.opendns.com);
echo $my_ip

# open port 22, just for my ip
aws ec2 authorize-security-group-ingress \
    --group-id sg-0000000 \
    --protocol tcp \
    --port 80 \
    --cidr $my_ip/24

# remove a firewall rule from a group
aws ec2 revoke-security-group-ingress \
    --group-id sg-0000000 \
    --protocol tcp \
    --port 80 \
    --cidr 0.0.0.0/24
```

```
# delete a security group
aws ec2 delete-security-group \
    --group-id sg-00000000
```

Instances

http://docs.aws.amazon.com/cli/latest/reference/ec2/index.html

```
# list all instances (running, and not running)
# http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html
aws ec2 describe-instances

# create a new instance
# http://docs.aws.amazon.com/cli/latest/reference/ec2/run-instances.html
aws ec2 run-instances \
    --image-id ami-f0e7d19a \
    --instance-type t2.micro \
    --security-group-ids sg-00000000 \
    --dry-run

# stop an instance
# http://docs.aws.amazon.com/cli/latest/reference/ec2/terminate-instances.html
aws ec2 terminate-instances \
    --instance-ids <instance_id>

# list status of all instances
# http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instance-status.html
aws ec2 describe-instance-status

# list status of a specific instance
aws ec2 describe-instance-status \
    --instance-ids <instance_id>
```

Tags

```
# list the tags of an instance
# http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-tags.html
aws ec2 describe-tags

# add a tag to an instance
# http://docs.aws.amazon.com/cli/latest/reference/ec2/create-tags.html
aws ec2 create-tags \
    --resources "ami-1a2b3c4d" \
    --tags Key=name,Value=debian

# delete a tag on an instance
# http://docs.aws.amazon.com/cli/latest/reference/ec2/delete-tags.html
aws ec2 delete-tags \
    --resources "ami-1a2b3c4d" \
    --tags Key=Name,Value=
```

Cloudwatch

Log Groups

http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatchLogs.html
http://docs.aws.amazon.com/cli/latest/reference/logs/index.html#cli-aws-logs

create a group

http://docs.aws.amazon.com/cli/latest/reference/logs/create-log-group.html

```
aws logs create-log-group \
    --log-group-name "DefaultGroup"
```

list all log groups

http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-groups.html

```
aws logs describe-log-groups

aws logs describe-log-groups \
    --log-group-name-prefix "Default"
```

delete a group

http://docs.aws.amazon.com/cli/latest/reference/logs/delete-log-group.html

```
aws logs delete-log-group \
    --log-group-name "DefaultGroup"
```

Log Streams

```
# Log group names can be between 1 and 512 characters long. Allowed
# characters include a-z, A-Z, 0-9, '_' (underscore), '-' (hyphen),
# '/' (forward slash), and '.' (period).

# create a log stream
# http://docs.aws.amazon.com/cli/latest/reference/logs/create-log-stream.html
aws logs create-log-stream \
    --log-group-name "DefaultGroup" \
    --log-stream-name "syslog"

# list details on a log stream
# http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-streams.html
aws logs describe-log-streams \
    --log-group-name "syslog"

aws logs describe-log-streams \
    --log-stream-name-prefix "syslog"

# delete a log stream
# http://docs.aws.amazon.com/cli/latest/reference/logs/delete-log-stream.html
aws logs delete-log-stream \
```

```
    --log-group-name "DefaultGroup" \
    --log-stream-name "Default Stream"
```

Read Getting started with aws-cli online: https://riptutorial.com/aws-cli/topic/1947/getting-started-with-aws-cli

Chapter 2: aws-codecommit for local git

Remarks

Prepare by setting up your local development machine with the aws command line tool and the git command.

Examples

Setup Codecommit for git command line

AWS Codecommit can be used as storage for private GIT repositories. The setup involves a few steps, assuming you have a valid AWS account already.

1. Sign up for AWS Codecommit. Currently only region `us-east-1` is available.
2. Create a IAM user who will have access to the repositories, eg `codecommit-user`
3. Attach permission role `AWSCodeCommitFullAccess` to this user
4. Create a new `Access Key` for this user and note `key id` and `secret code`
5. Now go ahead and create a AWS Configuration profile on your local machine

```
$ aws configure --profile codecommit-user
```

In the next step we associate the `aws` command with `git` as the credential helper with the following commands:

```
$ git config --global credential.helper \
    '!aws --profile codecommit-user codecommit credential-helper $@'
$ git config --global credential.UseHttpPath true
```

You can verify or edit this setup afterwards:

```
$ git config --global --edit
```

You should note a section:

```
[credential]
    helper = !aws --profile codecommit-user codecommit credential-helper $@
    UseHttpPath = true
```

Now you can use git from the command line as usual.

Use SourceTree with AWS Codecommit

Atlassian SourceTree is a visual tool for Mac and Windows to manage source code repositories. This can be used with Codecommit as a remote repository but need to add an extra configuration

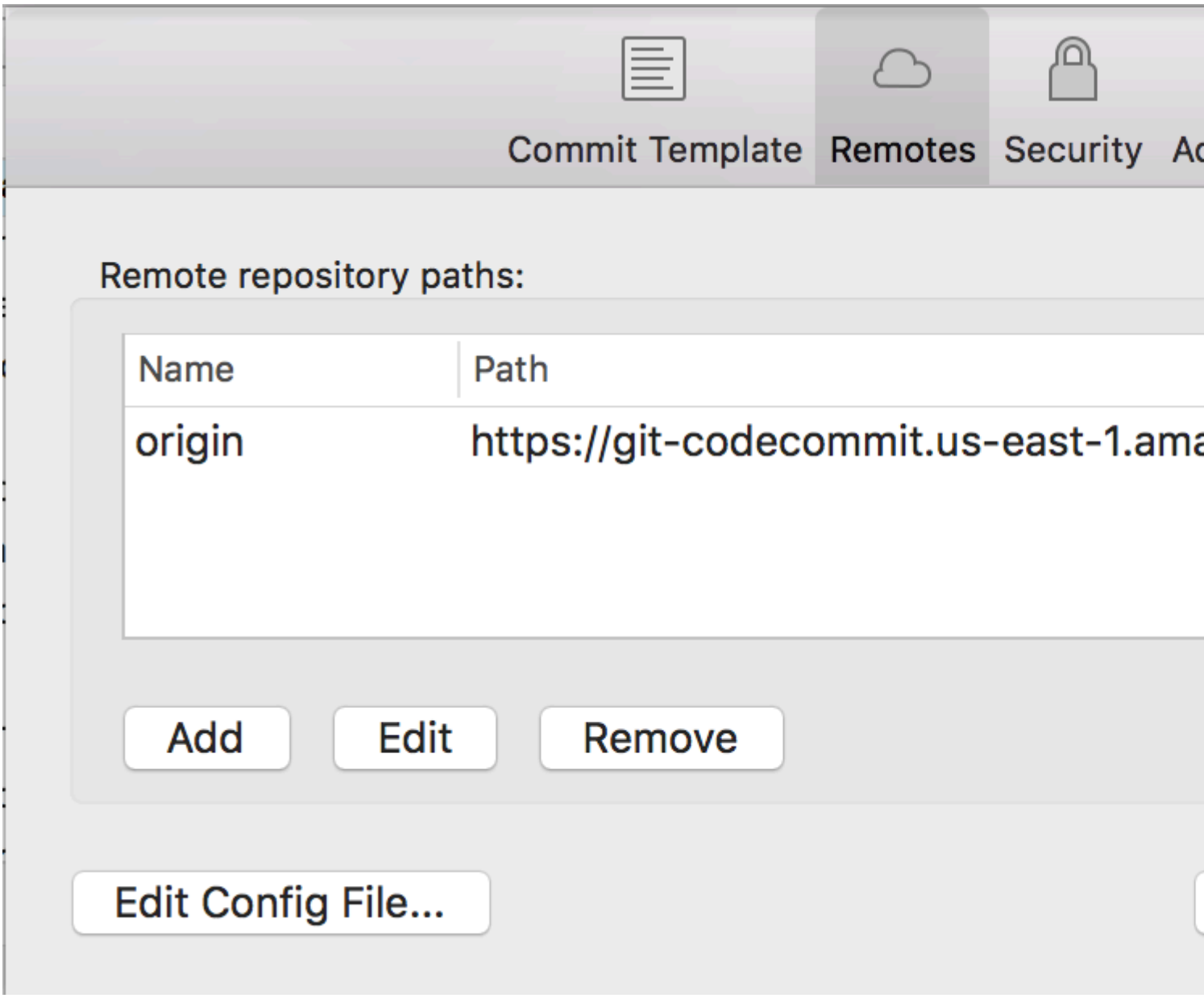
option to the local repository in SourceTree to be able to connect with codecommit.

First, setup Codecommit for local git.

Assuming you have a local `git` repository which you want to push to `codecommit` just follow these steps:

1. Login to AWS Codecommit using the web console.
2. Create a new repository, eg `my-project`
3. Copy the HTTPS URL, it should look like `https://git-codecommit.us-east-1.amazonaws.com/v1/repos/my-project`
4. Now in SourceTree open the panel Settings / Remotes
5. Add new remote with name: `origin` and Url / Path: the link you copied before
6. Finally open the option Edit Config File and add the following snippet:

```
[credential]
    helper = /usr/local/bin/aws --profile codecommit-user codecommit credential-helper $@
    UseHttpPath = true
```



After saving the config file should look something like this:

```
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
    ignorecase = true
    precomposeunicode = true
[branch "master"]
    remote = origin
    merge = refs/heads/master
[remote "origin"]
    url = https://git-codecommit.us-east-1.amazonaws.com/v1/repos/digitaloffice.nu
    fetch = +refs/heads/*:refs/remotes/origin/*
[credential]
    helper = /usr/local/bin/aws --profile codecommit-user codecommit credential-helper $@
    UseHttpPath = true
```

Please note: this is based on OS-X setup. Take special care of the path for aws (which is `/usr/local/bin/aws` in this case) and will most certainly be different under other Unixes or Windows configurations.

Read aws-codecommit for local git online: https://riptutorial.com/aws-cli/topic/6450/aws-codecommit-for-local-git

Chapter 3: ec2 describe-images usages

Examples

Describe image by AMI name

```
aws ec2 describe-images --filters "Name=name,Values=${NAME_OF_AMI}"
```

Read ec2 describe-images usages online: https://riptutorial.com/aws-cli/topic/9363/ec2-describe-images-usages

Chapter 4: The --query Parameter

Remarks

The `--query` parameter is often overlooked, but it is incredibly powerful. It uses the JMESPath query language to filter service responses down to precisely what you want.

Examples

Listing Instances in an Easy to Read Way

Instances have a lot of metadata that gets returned from a call to `describe-instances`, but often times you just want to see the basics. You can use a JMESPath query combined with table output to show concise instance information in an easily readable way.

```
aws ec2 describe-instances --output table --query "Reservations[].Instances[].{Name: Tags[?Key
== 'Name'].Value | [0], Id: InstanceId, State: State.Name, Type: InstanceType}"
```

```
-------------------------------------------------------
|                  DescribeInstances                  |
+-----------+---------------+----------+------------+
|    Id     |     Name      |  State   |   Type     |
+-----------+---------------+----------+------------+
|  i-abc123 |  None         |  stopped |  m3.large  |
|  i-def456 |  amazon linux |  stopped |  t2.micro  |
|  i-ghi789 |  proxy        |  running |  t2.micro  |
+-----------+---------------+----------+------------+
```

Now lets break that up piece by piece. First, we have `--output table`. This produces a colorized table representation of the response. This is generally most useful with commands that return small sets of data or where you have filtered the data down.

Now onto the `--query`. This one looks long, but it is actually quite simple. The first part is `Reservations[].Instances[]`. This returns a flattened list of all the returned instances.

The next part of the query is encapsulated with `.{}`. What this is doing is creating a new json object for each item in the list where each value is a JMESPath query to be applied to the source object (in this case, an Instance). Most of these are very simple, but `Name` is a bit more complex.

The full query to get `Name` is `Tags[?Key == 'Name'].Value | [0]`. The first part of that, `Tags[?Key == 'Name']` is searching the instance's tags for a tag whose key is `Name`. The second half `.Value | [0]` is selecting the values of each of those tags and then taking the first item from the list (in this case, there will only ever be one).

Exactly what you want in that table is completely up to you. If you wanted to add DNS information, for instance, you could easily add a new key `DNS: PublicDnsName`:

```
aws ec2 describe-instances --output table --query "Reservations[].Instances[].{Name: Tags[?Key
== 'Name'].Value | [0], Id: InstanceId, State: State.Name, Type: InstanceType, DNS:
PublicDnsName}"
```

```
----------------------------------------------------------------------------------------------
----------
|                                          DescribeInstances
|
+-----------------------------------------------------+-----------+---------------+----------+---
---------+
|                          DNS                        |    Id     |     Name      |  State   |
Type     |
+-----------------------------------------------------+-----------+---------------+----------+---
---------+
|                                                     |  i-abc123 |  None         |  stopped |
m3.large  |
|                                                     |  i-def456 |  amazon linux |  stopped |
t2.micro  |
|  ec2-192-168-1-1.us-west-2.compute.amazonaws.com |  i-ghi789 |  proxy        |  running |
t2.micro  |
+-----------------------------------------------------+-----------+---------------+----------+---
---------+
```

Read The --query Parameter online: https://riptutorial.com/aws-cli/topic/7306/the---query-parameter

Credits

S. No	Chapters	Contributors
1	Getting started with aws-cli	chenchuk, Community, Danny, Esteban, Nithin K Anil, Paddez, richardboydii, Scroff, Yaron Idan
2	aws-codecommit for local git	jlapoutre
3	ec2 describe-images usages	Yuki Inoue
4	The --query Parameter	Jordon Phillips