s3-credentials documentation

Release 0.16-2-g15922d2

Simon Willison

Apr 05, 2024

CONTENTS

A tool for creating credentials for accessing S3 buckets

For project background, see s3-credentials: a tool for creating credentials for S3 buckets on my blog.

Why would you need this? If you want to read and write to an S3 bucket from an automated script somewhere, you'll need an access key and secret key to authenticate your calls. This tool helps you create those with the most restrictive permissions possible.

If your code is running in EC2 or Lambda you can likely solve this using roles instead. This tool is mainly useful for when you are interacting with S3 from outside the boundaries of AWS itself.

CHAPTER

ONE

INSTALLATION

Install this tool using `pip`:

```
$ pip install s3-credentials
```

CHAPTER

TWO

DOCUMENTATION

2.1 Configuration

This tool uses boto3 under the hood which supports a number of different ways of providing your AWS credentials.

If you have an existing `~/.aws/config` or `~/.aws/credentials` file the tool will use that.

One way to create those files is using the `aws configure` command, available if you first run `pip install awscli`.

Alternatively, you can set the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables before calling this tool.

You can also use the `--access-key=`, `--secret-key=`, `--session-token` and `--auth` options documented below.

2.1.1 Common command options

All of the `s3-credentials` commands also accept the following options for authenticating against AWS:

- `--access-key`: AWS access key ID
- `--secret-key`: AWS secret access key
- `--session-token`: AWS session token
- `--endpoint-url`: Custom endpoint URL
- `--auth`: file (or `-` for standard input) containing credentials to use

The file passed to `--auth` can be either a JSON file or an INI file. JSON files should contain the following:

```
{
    "AccessKeyId": "AKIAWXFXAIOZA5IR5PY4",
    "SecretAccessKey": "g63..."
}
```

The JSON file can also optionally include a session token in a `"SessionToken"` key.

The INI format variant of this file should look like this:

```
[default]
aws_access_key_id=AKIAWXFXAIOZNCR2ST7S
aws_secret_access_key=g63...
```

Any section headers will do - the tool will use the information from the first section it finds in the file which has a `aws_access_key_id` key.

These auth file formats are the same as those that can be created using the `create` command.

2.2 Creating S3 credentials

The `s3-credentials create` command is the core feature of this tool. Pass it one or more S3 bucket names, specify a policy (read-write, read-only or write-only) and it will return AWS credentials that can be used to access those buckets.

These credentials can be **temporary** or **permanent**.

- Temporary credentials can last for between 15 minutes and 12 hours. They are created using STS.AssumeRole().
- Permanent credentials never expire. They are created by first creating a dedicated AWS user, then assigning a policy to that user and creating and returning an access key for it.

Make sure to record the `SecretAccessKey` because it will only be displayed once and cannot be recreated later on.

In this example I create permanent credentials for reading and writing files in my `static.niche-museums.com` S3 bucket:

```
% s3-credentials create static.niche-museums.com

Created user: s3.read-write.static.niche-museums.com with permissions boundary:␣
↪arn:aws:iam::aws:policy/AmazonS3FullAccess
Attached policy s3.read-write.static.niche-museums.com to user s3.read-write.static.
↪niche-museums.com
Created access key for user: s3.read-write.static.niche-museums.com
{
    "UserName": "s3.read-write.static.niche-museums.com",
    "AccessKeyId": "AKIAWXFXAIOZOYLZAEW5",
    "Status": "Active",
    "SecretAccessKey": "...",
    "CreateDate": "2021-11-03 01:38:24+00:00"
}
```

If you add `--format ini` the credentials will be output in INI format, suitable for pasting into a `~/.aws/credentials` file:

```
% s3-credentials create static.niche-museums.com --format ini > ini.txt
Created user: s3.read-write.static.niche-museums.com with permissions boundary:␣
↪arn:aws:iam::aws:policy/AmazonS3FullAccess
Attached policy s3.read-write.static.niche-museums.com to user s3.read-write.static.
↪niche-museums.com
Created access key for user: s3.read-write.static.niche-museums.com
% cat ini.txt
[default]
aws_access_key_id=AKIAWXFXAIOZKGXI4PVO
aws_secret_access_key=...
```

To create temporary credentials, add `--duration 15m` (or `1h` or `1200s`). The specified duration must be between 15 minutes and 12 hours.

```
% s3-credentials create static.niche-museums.com --duration 15m
Assume role against arn:aws:iam::462092780466:role/s3-credentials.AmazonS3FullAccess for␣
↪900s
{
    "AccessKeyId": "ASIAWXFXAIOZPAHAYHUG",
    "SecretAccessKey": "Nrnoc...",
    "SessionToken": "FwoGZXIvYXd...mr9Fjs=",
```

(continues on next page)

(continued from previous page)

```
    "Expiration": "2021-11-11 03:24:07+00:00"
}
```

When using temporary credentials the session token must be passed in addition to the access key and secret key.

The `create` command has a number of options:

- `--format TEXT`: The output format to use. Defaults to `json`, but can also be `ini`.
- `--duration 15m`: For temporary credentials, how long should they last? This can be specified in seconds, minutes or hours using a suffix of `s`, `m` or `h` - but must be between 15 minutes and 12 hours.
- `--username TEXT`: The username to use for the user that is created by the command (or the username of an existing user if you do not want to create a new one). If ommitted a default such as `s3.read-write.static.niche-museums.com` will be used.
- `-c, --create-bucket`: Create the buckets if they do not exist. Without this any missing buckets will be treated as an error.
- `--prefix my-prefix/`: Credentials should only allow access to keys in the S3 bucket that start with this prefix.
- `--public`: When creating a bucket, set it so that any file uploaded to that bucket can be downloaded by anyone who knows its filename. This attaches the *public bucket policy* and sets the `PublicAccessBlockConfiguration` to `false` for every option.
- `--website`: Sets the bucket to public and configures it to act as a website, with `index.html` treated as an index page and `error.html` used to display custom errors. The URL for the website will be `http://<bucket-name>.s3-website.<region>.amazonaws.com/` - the region defaults to `us-east-1` unless you specify a `--bucket-region`.
- `--read-only`: The user should only be allowed to read files from the bucket.
- `--write-only`: The user should only be allowed to write files to the bucket, but not read them. This can be useful for logging and backups.
- `--policy filepath-or-string`: A custom policy document (as a file path, literal JSON string or `-` for standard input) - see below.
- `--statement json-statement`: Custom JSON statement block to be added to the generated policy.
- `--bucket-region`: If creating buckets, the region in which they should be created.
- `--silent`: Don't output details of what is happening, just output the JSON for the created access credentials at the end.
- `--dry-run`: Output details of AWS changes that would have been made without applying them.
- `--user-permissions-boundary`: Custom permissions boundary to use for users created by this tool. The default is to restrict those users to only interacting with S3, taking the `--read-only` option into account. Use `none` to create users without any permissions boundary at all.

2.2.1 Changes that will be made to your AWS account

How the tool works varies depending on if you are creating temporary or permanent credentials.

For permanent credentials, the steps are as follows:

1. Confirm that each of the specified buckets exists. If they do not and `--create-bucket` was passed create them - otherwise exit with an error.
2. If a username was not specified, derive a username using the `s3.$permission.$buckets` format.
3. If a user with that username does not exist, create one with an S3 permissions boundary of AmazonS3ReadOnlyAccess for `--read-only` or AmazonS3FullAccess otherwise - unless `--user-permissions-boundary=none` was passed, or a custom permissions boundary string.
4. For each specified bucket, add an inline IAM policy to the user that gives them permission to either read-only, write-only or read-write against that bucket.
5. Create a new access key for that user and output the key and its secret to the console.

For temporary credentials:

1. Confirm or create buckets, in the same way as for permanent credentials.
2. Check if an AWS role called `s3-credentials.AmazonS3FullAccess` exists. If it does not exist create it, configured to allow the user's AWS account to assume it and with the `arn:aws:iam::aws:policy/AmazonS3FullAccess` policy attached.
3. Use `STS.AssumeRole()` to return temporary credentials that are restricted to just the specified buckets and specified read-only/read-write/write-only policy.

You can run the `create` command with the `--dry-run` option to see a summary of changes that would be applied, including details of generated policy documents, without actually applying those changes.

2.2.2 Using a custom policy

The policy documents applied by this tool *are listed here*.

If you want to use a custom policy document you can do so using the `--policy` option.

First, create your policy document as a JSON file that looks something like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject*", "s3:ListBucket"],
      "Resource": [
        "arn:aws:s3:::$!BUCKET_NAME!$",
        "arn:aws:s3:::$!BUCKET_NAME!$/*"
      ],
    }
  ]
}
```

Note the `$!BUCKET_NAME!$` strings - these will be replaced with the name of the relevant S3 bucket before the policy is applied.

Save that as `custom-policy.json` and apply it using the following command:

```
% s3-credentials create my-s3-bucket \
    --policy custom-policy.json
```

You can also pass - to read from standard input, or you can pass the literal JSON string directly to the `--policy` option:

```
% s3-credentials create my-s3-bucket --policy '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject*", "s3:ListBucket"],
      "Resource": [
        "arn:aws:s3:::$!BUCKET_NAME!$",
        "arn:aws:s3:::$!BUCKET_NAME!$/*"
      ],
    }
  ]
}'
```

You can also specify one or more extra statement blocks that should be added to the generated policy, using `--statement JSON`. This example enables the AWS `textract:` APIs for the generated credentials, useful for using with the s3-ocr tool:

```
% s3-credentials create my-s3-bucket --statement '{
  "Effect": "Allow",
  "Action": "textract:*",
  "Resource": "*"
}'
```

2.3 Other commands

- *policy*
- *whoami*
- *list-users*
- *list-buckets*
- *list-bucket*
- *list-user-policies*
- *list-roles*
- *delete-user*
- *put-object*
- *put-objects*
- *delete-objects*
- *get-object*
- *get-objects*

- *set-cors-policy and get-cors-policy*
- *debug-bucket*

2.3.1 policy

You can use the `s3-credentials policy` command to generate the JSON policy document that would be used without applying it. The command takes one or more required bucket names and a subset of the options available on the `create` command:

- `--read-only` - generate a read-only policy
- `--write-only` - generate a write-only policy
- `--prefix` - policy should be restricted to keys in the bucket that start with this prefix
- `--statement json-statement`: Custom JSON statement block
- `--public-bucket` - generate a bucket policy for a public bucket

With none of these options it defaults to a read-write policy.

```
% s3-credentials policy my-bucket --read-only
{
    "Version": "2012-10-17",
...
```

2.3.2 whoami

To see which user you are authenticated as:

```
s3-credentials whoami
```

This will output JSON representing the currently authenticated user.

Using this with the `--auth` option is useful for verifying created credentials:

```
s3-credentials create static.niche-museums.com --read-only > auth.json
s3-credentials whoami --auth auth.json
{
    "UserId": "AIDAWXFXAIOZPIZC6MHAG",
    "Account": "462092780466",
    "Arn": "arn:aws:iam::462092780466:user/s3.read-only.static.niche-museums.com"
}
```

2.3.3 list-users

To see a list of all users that exist for your AWS account:

```
s3-credentials list-users
```

This will return a pretty-printed array of JSON objects by default.

Add `--nl` to collapse these to single lines as valid newline-delimited JSON.

Add `--csv` or `--tsv` to get back CSV or TSV data.

2.3.4 list-buckets

Shows a list of all buckets in your AWS account.

```
% s3-credentials list-buckets
[
  {
    "Name": "aws-cloudtrail-logs-462092780466-f2c900d3",
    "CreationDate": "2021-03-25 22:19:54+00:00"
  },
  {
    "Name": "simonw-test-bucket-for-s3-credentials",
    "CreationDate": "2021-11-03 21:46:12+00:00"
  }
]
```

With no extra arguments this will show all available buckets - you can also add one or more explicit bucket names to see just those buckets:

```
% s3-credentials list-buckets simonw-test-bucket-for-s3-credentials
[
  {
    "Name": "simonw-test-bucket-for-s3-credentials",
    "CreationDate": "2021-11-03 21:46:12+00:00"
  }
]
```

This accepts the same `--nl`, `--csv` and `--tsv` options as `list-users`.

Add `--details` to include details of the bucket ACL, website configuration and public access block settings. This is useful for running a security audit of your buckets.

Using `--details` adds several additional API calls for each bucket, so it is advisable to use it with one or more explicit bucket names.

```
% s3-credentials list-buckets simonw-test-public-website-bucket --details
[
  {
    "Name": "simonw-test-public-website-bucket",
    "CreationDate": "2021-11-08 22:53:30+00:00",
    "region": "us-east-1",
    "bucket_acl": {
      "Owner": {
```

(continues on next page)

(continued from previous page)

```
        "DisplayName": "simon",
        "ID": "abcdeabcdeabcdeabcdeabcdeabcde0001"
      },
      "Grants": [
        {
          "Grantee": {
            "DisplayName": "simon",
            "ID": "abcdeabcdeabcdeabcdeabcdeabcde0001",
            "Type": "CanonicalUser"
          },
          "Permission": "FULL_CONTROL"
        }
      ]
    },
    "public_access_block": null,
    "bucket_website": {
      "IndexDocument": {
        "Suffix": "index.html"
      },
      "ErrorDocument": {
        "Key": "error.html"
      },
      "url": "http://simonw-test-public-website-bucket.s3-website.us-east-1.amazonaws.
↪com/"
    }
  }
]
```

A bucket with `public_access_block` might look like this:

```
{
  "Name": "aws-cloudtrail-logs-462092780466-f2c900d3",
  "CreationDate": "2021-03-25 22:19:54+00:00",
  "bucket_acl": {
    "Owner": {
      "DisplayName": "simon",
      "ID": "abcdeabcdeabcdeabcdeabcdeabcde0001"
    },
    "Grants": [
      {
        "Grantee": {
          "DisplayName": "simon",
          "ID": "abcdeabcdeabcdeabcdeabcdeabcde0001",
          "Type": "CanonicalUser"
        },
        "Permission": "FULL_CONTROL"
      }
    ]
  },
  "public_access_block": {
    "BlockPublicAcls": true,
    "IgnorePublicAcls": true,
```

(continues on next page)

(continued from previous page)

```
    "BlockPublicPolicy": true,
    "RestrictPublicBuckets": true
  },
  "bucket_website": null
}
```

2.3.5 list-bucket

To list the contents of a bucket, use `list-bucket`:

```
% s3-credentials list-bucket static.niche-museums.com
[
  {
    "Key": "Griffith-Observatory.jpg",
    "LastModified": "2020-01-05 16:51:01+00:00",
    "ETag": "\"a4cff17d189e7eb0c4d3bf0257e56885\"",
    "Size": 3360040,
    "StorageClass": "STANDARD"
  },
  {
    "Key": "IMG_0353.jpeg",
    "LastModified": "2019-10-25 02:50:49+00:00",
    "ETag": "\"d45bab0b65c0e4b03b2ac0359c7267e3\"",
    "Size": 2581023,
    "StorageClass": "STANDARD"
  }
]
```

You can use the `--prefix myprefix/` option to list only keys that start with a specific prefix.

The commmand accepts the same `--nl`, `--csv` and `--tsv` options as `list-users`.

Add `--urls` to include a URL field in the output providing the full URL to each object.

2.3.6 list-user-policies

To see a list of inline policies belonging to users:

```
% s3-credentials list-user-policies s3.read-write.static.niche-museums.com

User: s3.read-write.static.niche-museums.com
PolicyName: s3.read-write.static.niche-museums.com
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
```

(continues on next page)

(continued from previous page)

```
        "arn:aws:s3:::static.niche-museums.com"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": [
        "arn:aws:s3:::static.niche-museums.com/*"
      ]
    }
  ]
}
```

You can pass any number of usernames here. If you don't specify a username the tool will loop through every user belonging to your account:

```
s3-credentials list-user-policies
```

2.3.7 list-roles

The `list-roles` command lists all of the roles available for the authenticated account.

Add `--details` to fetch the inline and attached managed policies for each row as well - this is slower as it needs to make several additional API calls for each role.

You can optionally add one or more role names to the command to display and fetch details about just those specific roles.

Example usage:

```
% s3-credentials list-roles AWSServiceRoleForLightsail --details
[
  {
    "Path": "/aws-service-role/lightsail.amazonaws.com/",
    "RoleName": "AWSServiceRoleForLightsail",
    "RoleId": "AROAWXFXAIOZG5ACQ5NZ5",
    "Arn": "arn:aws:iam::462092780466:role/aws-service-role/lightsail.amazonaws.com/
↪AWSServiceRoleForLightsail",
    "CreateDate": "2021-01-15 21:41:48+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "lightsail.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "MaxSessionDuration": 3600,
```

(continues on next page)

(continued from previous page)


```
    "inline_policies": [
      {
        "RoleName": "AWSServiceRoleForLightsail",
        "PolicyName": "LightsailExportAccess",
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:CreateGrant"
              ],
              "Resource": "arn:aws:kms:*:451833091580:key/*"
            },
            {
              "Effect": "Allow",
              "Action": [
                "cloudformation:DescribeStacks"
              ],
              "Resource": "arn:aws:cloudformation:*:*:stack/*/*"
            }
          ]
        }
      }
    ],
    "attached_policies": [
      {
        "PolicyName": "LightsailExportAccess",
        "PolicyId": "ANPAJ4LZGPQLZWMVR4WMQ",
        "Arn": "arn:aws:iam::aws:policy/aws-service-role/LightsailExportAccess",
        "Path": "/aws-service-role/",
        "DefaultVersionId": "v2",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "AWS Lightsail service linked role policy which grants␣
↪permissions to export resources",
        "CreateDate": "2018-09-28 16:35:54+00:00",
        "UpdateDate": "2022-01-15 01:45:33+00:00",
        "Tags": [],
        "PolicyVersion": {
          "Document": {
            "Version": "2012-10-17",
            "Statement": [
              {
                "Effect": "Allow",
                "Action": [
                  "iam:DeleteServiceLinkedRole",
                  "iam:GetServiceLinkedRoleDeletionStatus"
                ],
```


(continues on next page)

(continued from previous page)

```
                "Resource": "arn:aws:iam::*:role/aws-service-role/lightsail.amazonaws.
↪com/AWSServiceRoleForLightsail*"
              },
              {
                "Effect": "Allow",
                "Action": [
                  "ec2:CopySnapshot",
                  "ec2:DescribeSnapshots",
                  "ec2:CopyImage",
                  "ec2:DescribeImages"
                ],
                "Resource": "*"
              },
              {
                "Effect": "Allow",
                "Action": [
                  "s3:GetAccountPublicAccessBlock"
                ],
                "Resource": "*"
              }
            ]
          },
          "VersionId": "v2",
          "IsDefaultVersion": true,
          "CreateDate": "2022-01-15 01:45:33+00:00"
        }
      }
    ]
  }
]
```

Add `--nl` to collapse these to single lines as valid newline-delimited JSON.

Add `--csv` or `--tsv` to get back CSV or TSV data.

2.3.8 delete-user

In trying out this tool it's possible you will create several different user accounts that you later decide to clean up.

Deleting AWS users is a little fiddly: you first need to delete their access keys, then their inline policies and finally the user themselves.

The `s3-credentials delete-user` handles this for you:

```
% s3-credentials delete-user s3.read-write.simonw-test-bucket-10
User: s3.read-write.simonw-test-bucket-10
  Deleted policy: s3.read-write.simonw-test-bucket-10
  Deleted access key: AKIAWXFXAIOZK3GPEIWR
  Deleted user
```

You can pass it multiple usernames to delete multiple users at a time.

2.3.9 put-object

You can upload a file to a key in an S3 bucket using `s3-credentials put-object`:

```
s3-credentials put-object my-bucket my-key.txt /path/to/file.txt
```

Use `-` as the file name to upload from standard input:

```
echo "Hello" | s3-credentials put-object my-bucket hello.txt -
```

This command shows a progress bar by default. Use `-s` or `--silent` to hide the progress bar.

The `Content-Type` on the uploaded object will be automatically set based on the file extension. If you are using standard input, or you want to over-ride the detected type, you can do so using the `--content-type` option:

```
echo "<h1>Hello World</h1>" | \
  s3-credentials put-object my-bucket hello.html - --content-type "text/html"
```

2.3.10 put-objects

`s3-credentials put-objects` can be used to upload more than one file at once.

Pass one or more filenames to upload them to the root of your bucket:

```
s3-credentials put-objects my-bucket one.txt two.txt three.txt
```

Use `--prefix my-prefix` to upload them to the specified prefix:

```
s3-credentials put-objects my-bucket one.txt --prefix my-prefix
```

This will upload the file to `my-prefix/one.txt`.

Pass one or more directories to upload the contents of those directories. `.` uploads everything in your current directory:

```
s3-credentials put-objects my-bucket .
```

Passing directory names will upload the directory and all of its contents:

```
s3-credentials put-objects my-bucket my-directory
```

If `my-directory` had files `one.txt` and `two.txt` in it, the result would be:

```
my-directory/one.txt
my-directory/two.txt
```

A progress bar will be shown by default. Use `-s` or `--silent` to hide it.

Add `--dry-run` to get a preview of what would be uploaded without uploading anything:

```
s3-credentials put-objects my-bucket . --dry-run
```

```
out/IMG_1254.jpeg => s3://my-bucket/out/IMG_1254.jpeg
out/alverstone-mead-2.jpg => s3://my-bucket/out/alverstone-mead-2.jpg
out/alverstone-mead-1.jpg => s3://my-bucket/out/alverstone-mead-1.jpg
```

2.3.11 delete-objects

`s3-credentials delete-objects` can be used to delete one or more keys from the bucket.

Pass one or more keys to delete them:

```
s3-credentials delete-objects my-bucket one.txt two.txt three.txt
```

Use `--prefix my-prefix` to delete all keys with the specified prefix:

```
s3-credentials delete-objects my-bucket --prefix my-prefix
```

Pass `-d` or `--dry-run` to perform a dry-run of the deletion, which will list the keys that would be deleted without actually deleting them.

```
s3-credentials delete-objects my-bucket --prefix my-prefix --dry-run
```

2.3.12 get-object

To download a file from a bucket use `s3-credentials get-object`:

```
s3-credentials get-object my-bucket hello.txt
```

This defaults to outputting the downloaded file to the terminal. You can instead direct it to save to a file on disk using the `-o` or `--output` option:

```
s3-credentials get-object my-bucket hello.txt -o /path/to/hello.txt
```

2.3.13 get-objects

`s3-credentials get-objects` can be used to download multiple files from a bucket at once.

Without extra arguments, this downloads everything:

```
s3-credentials get-objects my-bucket
```

Files will be written to the current directory by default, preserving their directory structure from the bucket.

To write to a different directory use `--output` or `-o`:

```
s3-credentials get-objects my-bucket -o /path/to/output
```

To download multiple specific files, add them as arguments to the command:

```
s3-credentials get-objects my-bucket one.txt two.txt path/to/three.txt
```

You can pass one or more `--pattern` or `-p` options to download files matching a specific pattern:

```
s3-credentials get-objects my-bucket -p "*.txt" -p "static/*.css"
```

Here the `*` wildcard will match any sequence of characters, including `/`. `?` will match a single character.

A progress bar will be shown by default. Use `-s` or `--silent` to hide it.

2.3.14 set-cors-policy and get-cors-policy

You can set the CORS policy for a bucket using the `set-cors-policy` command. S3 CORS policies are set at the bucket level - they cannot be set for individual items.

First, create the bucket. Make sure to make it `--public`:

```
s3-credentials create my-cors-bucket --public -c
```

You can set a default CORS policy - allowing `GET` requests from any origin - like this:

```
s3-credentials set-cors-policy my-cors-bucket
```

You can use the `get-cors-policy` command to confirm the policy you have set:

```
s3-credentials get-cors-policy my-cors-bucket
[
    {
        "ID": "set-by-s3-credentials",
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ]
    }
]
```

To customize the CORS policy, use the following options:

- `-m/--allowed-method` - Allowed method e.g. `GET`
- `-h/--allowed-header` - Allowed header e.g. `Authorization`
- `-o/--allowed-origin` - Allowed origin e.g. `https://www.example.com/`
- `-e/--expose-header` - Header to expose e.g. `ETag`
- `--max-age-seconds` - How long to cache preflight requests

Each of these can be passed multiple times with the exception of `--max-age-seconds`.

The following example allows GET and PUT methods from code running on `https://www.example.com/`, allows the encoming `Authorization` header and exposes the `ETag` header. It also sets the client to cache preflight requests for 60 seconds:

```
s3-credentials set-cors-policy my-cors-bucket2 \
  --allowed-method GET \
  --allowed-method PUT \
  --allowed-origin https://www.example.com/ \
  --expose-header ETag \
  --max-age-seconds 60
```

2.3.15 debug-bucket

The `debug-bucket` command is useful for diagnosing issues with a bucket:

```
s3-credentials debug-bucket my-bucket
```

Example output:

```
Bucket ACL:
{
    "Owner": {
        "DisplayName": "username",
        "ID": "cc8ca3a037c6a7c1fa7580076bf7cd1949b3f2f58f01c9df9e53c51f6a249910"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "username",
                "ID": "cc8ca3a037c6a7c1fa7580076bf7cd1949b3f2f58f01c9df9e53c51f6a249910",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
Bucket policy status:
{
    "PolicyStatus": {
        "IsPublic": true
    }
}
Bucket public access block:
{
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": false,
        "RestrictPublicBuckets": false
    }
}
```

2.4 Policy documents

The IAM policies generated by this tool for a bucket called `my-s3-bucket` would look like this:

2.4.1 read-write (default)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/*"
      ]
    }
  ]
}
```

2.4.2 --read-only

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/*"
      ]
    }
  ]
}
```

2.4.3 --write-only

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/*"
      ]
    }
  ]
}
```

2.4.4 --prefix my-prefix/

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "my-prefix/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/my-prefix/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/my-prefix/*"
      ]
    }
```

(continues on next page)

(continued from previous page)

```
  ]
}
```

2.4.5 --prefix my-prefix/ --read-only

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "my-prefix/*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/my-prefix/*"
      ]
    }
  ]
}
```

2.4.6 --prefix my-prefix/ --write-only

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/my-prefix/*"
      ]
    }
  ]
}
```

2.4.7 public bucket policy

Buckets created using the `--public` option will have the following bucket policy attached to them:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-s3-bucket/*"
      ]
    }
  ]
}
```

2.5 Command help

This page shows the `--help` output for all of the `s3-credentials` commands.

2.5.1 s3-credentials –help

```
Usage: s3-credentials [OPTIONS] COMMAND [ARGS]...

  A tool for creating credentials for accessing S3 buckets

  Documentation: https://s3-credentials.readthedocs.io/

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  create               Create and return new AWS credentials for specified...
  debug-bucket         Run a bunch of diagnostics to help debug a bucket
  delete-objects       Delete one or more object from an S3 bucket
  delete-user          Delete specified users, their access keys and their...
  get-cors-policy      Get CORS policy for a bucket
  get-object           Download an object from an S3 bucket
  get-objects          Download multiple objects from an S3 bucket
  list-bucket          List contents of bucket
  list-buckets         List buckets
  list-roles           List roles
  list-user-policies   List inline policies for specified users
  list-users           List all users for this account
  policy               Output generated JSON policy for one or more buckets
  put-object           Upload an object to an S3 bucket
  put-objects          Upload multiple objects to an S3 bucket
  set-cors-policy      Set CORS policy for a bucket
  whoami               Identify currently authenticated user
```

2.5.2 s3-credentials create –help

```
Usage: s3-credentials create [OPTIONS] BUCKETS...

  Create and return new AWS credentials for specified S3 buckets - optionally
  also creating the bucket if it does not yet exist.

  To create a new bucket and output read-write credentials:

      s3-credentials create my-new-bucket -c

  To create read-only credentials for an existing bucket:

      s3-credentials create my-existing-bucket --read-only
```

(continues on next page)

(continued from previous page)

```
  To create write-only credentials that are only valid for 15 minutes:

      s3-credentials create my-existing-bucket --write-only -d 15m

Options:
  -f, --format [ini|json]         Output format for credentials
  -d, --duration DURATION         How long should these credentials work for?
                                  Default is forever, use 3600 for 3600 seconds,
                                  15m for 15 minutes, 1h for 1 hour
  --username TEXT                 Username to create or existing user to use
  -c, --create-bucket             Create buckets if they do not already exist
  --prefix TEXT                   Restrict to keys starting with this prefix
  --public                        Make the created bucket public: anyone will be
                                  able to download files if they know their name
  --website                       Configure bucket to act as a website, using
                                  index.html and error.html
  --read-only                     Only allow reading from the bucket
  --write-only                    Only allow writing to the bucket
  --policy POLICY                 Path to a policy.json file, or literal JSON
                                  string - $!BUCKET_NAME!$ will be replaced with
                                  the name of the bucket
  --statement STATEMENT           JSON statement to add to the policy
  --bucket-region TEXT            Region in which to create buckets
  --silent                        Don't show performed steps
  --dry-run                       Show steps without executing them
  --user-permissions-boundary TEXT
                                  Custom permissions boundary to use for created
                                  users, or 'none' to create without. Defaults
                                  to limiting to S3 based on --read-only and
                                  --write-only options.
  --access-key TEXT               AWS access key ID
  --secret-key TEXT               AWS secret access key
  --session-token TEXT            AWS session token
  --endpoint-url TEXT             Custom endpoint URL
  -a, --auth FILENAME             Path to JSON/INI file containing credentials
  --help                          Show this message and exit.
```

2.5.3 s3-credentials debug-bucket –help

```
Usage: s3-credentials debug-bucket [OPTIONS] BUCKET

  Run a bunch of diagnostics to help debug a bucket

     s3-credentials debug-bucket my-bucket

Options:
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
```

(continues on next page)

(continued from previous page)

```
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.4 s3-credentials delete-objects –help

```
Usage: s3-credentials delete-objects [OPTIONS] BUCKET [KEYS]...

  Delete one or more object from an S3 bucket

  Pass one or more keys to delete them:

      s3-credentials delete-objects my-bucket one.txt two.txt

  To delete all files matching a prefix, pass --prefix:

      s3-credentials delete-objects my-bucket --prefix my-folder/

Options:
  --prefix TEXT         Delete everything with this prefix
  -s, --silent          Don't show informational output
  -d, --dry-run         Show keys that would be deleted without deleting them
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.5 s3-credentials delete-user –help

```
Usage: s3-credentials delete-user [OPTIONS] USERNAMES...

  Delete specified users, their access keys and their inline policies

      s3-credentials delete-user username1 username2

Options:
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.6 s3-credentials get-cors-policy –help

```
Usage: s3-credentials get-cors-policy [OPTIONS] BUCKET

  Get CORS policy for a bucket

      s3-credentials get-cors-policy my-bucket

  Returns the CORS policy for this bucket, if set, as JSON

Options:
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.7 s3-credentials get-object –help

```
Usage: s3-credentials get-object [OPTIONS] BUCKET KEY

  Download an object from an S3 bucket

  To see the contents of the bucket on standard output:

      s3-credentials get-object my-bucket hello.txt

  To save to a file:

      s3-credentials get-object my-bucket hello.txt -o hello.txt

Options:
  -o, --output FILE     Write to this file instead of stdout
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.8 s3-credentials get-objects –help

```
Usage: s3-credentials get-objects [OPTIONS] BUCKET [KEYS]...

  Download multiple objects from an S3 bucket

  To download everything, run:

      s3-credentials get-objects my-bucket
```

(continues on next page)

(continued from previous page)

```
  Files will be saved to a directory called my-bucket. Use -o dirname to save to
  a different directory.

  To download specific keys, list them:

      s3-credentials get-objects my-bucket one.txt path/two.txt

  To download files matching a glob-style pattern, use:

      s3-credentials get-objects my-bucket --pattern '*/*.js'

Options:
  -o, --output DIRECTORY  Write to this directory instead of one matching the
                          bucket name
  -p, --pattern TEXT      Glob patterns for files to download, e.g. '*/*.js'
  -s, --silent            Don't show progress bar
  --access-key TEXT       AWS access key ID
  --secret-key TEXT       AWS secret access key
  --session-token TEXT    AWS session token
  --endpoint-url TEXT     Custom endpoint URL
  -a, --auth FILENAME     Path to JSON/INI file containing credentials
  --help                  Show this message and exit.
```

2.5.9 s3-credentials list-bucket –help

```
Usage: s3-credentials list-bucket [OPTIONS] BUCKET

  List contents of bucket

  To list the contents of a bucket as JSON:

      s3-credentials list-bucket my-bucket

  Add --csv or --csv for CSV or TSV format:

      s3-credentials list-bucket my-bucket --csv

  Add --urls to get an extra URL field for each key:

      s3-credentials list-bucket my-bucket --urls

Options:
  --prefix TEXT         List keys starting with this prefix
  --urls                Show URLs for each key
  --nl                  Output newline-delimited JSON
  --csv                 Output CSV
  --tsv                 Output TSV
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
```

(continues on next page)

(continued from previous page)

```
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.10 s3-credentials list-buckets –help

```
Usage: s3-credentials list-buckets [OPTIONS] [BUCKETS]...

  List buckets

  To list all buckets and their creation time as JSON:

      s3-credentials list-buckets

  Add --csv or --csv for CSV or TSV format:

      s3-credentials list-buckets --csv

  For extra details per bucket (much slower) add --details

      s3-credentials list-buckets --details

Options:
  --details             Include extra bucket details (slower)
  --nl                  Output newline-delimited JSON
  --csv                 Output CSV
  --tsv                 Output TSV
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.11 s3-credentials list-roles –help

```
Usage: s3-credentials list-roles [OPTIONS] [ROLE_NAMES]...

  List roles

  To list all roles for this AWS account:

      s3-credentials list-roles

  Add --csv or --csv for CSV or TSV format:

      s3-credentials list-roles --csv
```

(continues on next page)

(continued from previous page)

```
  For extra details per role (much slower) add --details

      s3-credentials list-roles --details

Options:
  --details             Include attached policies (slower)
  --nl                  Output newline-delimited JSON
  --csv                 Output CSV
  --tsv                 Output TSV
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.12 s3-credentials list-user-policies –help

```
Usage: s3-credentials list-user-policies [OPTIONS] [USERNAMES]...

  List inline policies for specified users

      s3-credentials list-user-policies username

  Returns policies for all users if no usernames are provided.

Options:
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.13 s3-credentials list-users –help

```
Usage: s3-credentials list-users [OPTIONS]

  List all users for this account

      s3-credentials list-users

  Add --csv or --csv for CSV or TSV format:

      s3-credentials list-users --csv

Options:
  --nl                  Output newline-delimited JSON
  --csv                 Output CSV
```

(continues on next page)

(continued from previous page)

```
  --tsv                 Output TSV
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.14 s3-credentials policy –help

```
Usage: s3-credentials policy [OPTIONS] BUCKETS...

  Output generated JSON policy for one or more buckets

  Takes the same options as s3-credentials create

  To output a read-only JSON policy for a bucket:

      s3-credentials policy my-bucket --read-only

Options:
  --read-only             Only allow reading from the bucket
  --write-only            Only allow writing to the bucket
  --prefix TEXT           Restrict to keys starting with this prefix
  --statement STATEMENT   JSON statement to add to the policy
  --public-bucket         Bucket policy for allowing public access
  --help                  Show this message and exit.
```

2.5.15 s3-credentials put-object –help

```
Usage: s3-credentials put-object [OPTIONS] BUCKET KEY PATH

  Upload an object to an S3 bucket

  To upload a file to /my-key.txt in the my-bucket bucket:

      s3-credentials put-object my-bucket my-key.txt /path/to/file.txt

  Use - to upload content from standard input:

      echo "Hello" | s3-credentials put-object my-bucket hello.txt -

Options:
  --content-type TEXT   Content-Type to use (default is auto-detected based on
                        file extension)
  -s, --silent          Don't show progress bar
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
```

(continues on next page)

(continued from previous page)

```
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.16 s3-credentials put-objects –help

```
Usage: s3-credentials put-objects [OPTIONS] BUCKET OBJECTS...

  Upload multiple objects to an S3 bucket

  Pass one or more files to upload them:

      s3-credentials put-objects my-bucket one.txt two.txt

  These will be saved to the root of the bucket. To save to a different location
  use the --prefix option:

      s3-credentials put-objects my-bucket one.txt two.txt --prefix my-folder

  This will upload them my-folder/one.txt and my-folder/two.txt.

  If you pass a directory it will be uploaded recursively:

      s3-credentials put-objects my-bucket my-folder

  This will create keys in my-folder/... in the S3 bucket.

  To upload all files in a folder to the root of the bucket instead use this:

      s3-credentials put-objects my-bucket my-folder/*

Options:
  --prefix TEXT         Prefix to add to the files within the bucket
  -s, --silent          Don't show progress bar
  --dry-run             Show steps without executing them
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.5.17 s3-credentials set-cors-policy –help

```
Usage: s3-credentials set-cors-policy [OPTIONS] BUCKET

  Set CORS policy for a bucket

  To allow GET requests from any origin:

      s3-credentials set-cors-policy my-bucket

  To allow GET and PUT from a specific origin and expose ETag headers:

      s3-credentials set-cors-policy my-bucket \
        --allowed-method GET \
        --allowed-method PUT \
        --allowed-origin https://www.example.com/ \
        --expose-header ETag

Options:
  -m, --allowed-method TEXT  Allowed method e.g. GET
  -h, --allowed-header TEXT  Allowed header e.g. Authorization
  -o, --allowed-origin TEXT  Allowed origin e.g. https://www.example.com/
  -e, --expose-header TEXT   Header to expose e.g. ETag
  --max-age-seconds INTEGER  How long to cache preflight requests
  --access-key TEXT          AWS access key ID
  --secret-key TEXT          AWS secret access key
  --session-token TEXT       AWS session token
  --endpoint-url TEXT        Custom endpoint URL
  -a, --auth FILENAME        Path to JSON/INI file containing credentials
  --help                     Show this message and exit.
```

2.5.18 s3-credentials whoami –help

```
Usage: s3-credentials whoami [OPTIONS]

  Identify currently authenticated user

Options:
  --access-key TEXT     AWS access key ID
  --secret-key TEXT     AWS secret access key
  --session-token TEXT  AWS session token
  --endpoint-url TEXT   Custom endpoint URL
  -a, --auth FILENAME   Path to JSON/INI file containing credentials
  --help                Show this message and exit.
```

2.6 Contributing

To contribute to this tool, first checkout the code. Then create a new virtual environment:

```
cd s3-credentials
python -m venv venv
source venv/bin/activate
```

Or if you are using `pipenv`:

```
pipenv shell
```

Now install the dependencies and test dependencies:

```
pip install -e '.[test]'
```

To run the tests:

```
pytest
```

Any changes to the generated policies require an update to the README using Cog:

```
cog -r README.md
```

2.6.1 Integration tests

The main tests all use stubbed interfaces to AWS, so will not make any outbound API calls.

There is also a suite of integration tests in `tests/test_integration.py` which DO make API calls to AWS, using credentials from your environment variables or `~/.aws/credentials` file.

These tests are skipped by default. If you have AWS configured with an account that has permission to run the actions required by `s3-credentials` (create users, roles, buckets etc) you can run these tests using:

```
pytest --integration
```

The tests will create a number of different users and buckets and should then delete them once they finish running.

CHAPTER

THREE

TIPS

You can see a log of changes made by this tool using AWS CloudTrail - the following link should provide an Event History interface showing revelant changes made to your AWS account such as `CreateAccessKey`, `CreateUser`, `PutUserPolicy` and more:

https://console.aws.amazon.com/cloudtrail/home

You can view a list of your S3 buckets and confirm that they have the desired permissions and properties here:

https://console.aws.amazon.com/s3/home

The management interface for an individual bucket is at `https://console.aws.amazon.com/s3/buckets/NAME-OF-BUCKET`