azureenergylabelercli Documentation

Release 2.2.3

Sayantan Khanra

Sep 22, 2023

CONTENTS

Contents:

CHAPTER

ONE

AZUREENERGYLABELERCLI

A cli to help generate energy label for Azure tenant, subscriptions and resource groups.

- Documentation: https://azureenergylabelercli.readthedocs.org/en/latest

1.1 Energy Labels

Table 1: Energy Labels table

label	high	medium	low
A =>	0	up to 10	up to 20
B ==>	up to 10	up to 20	up to 40
C ===>	up to 15	up to 30	up to 60
D ====>	up to 20	up to 40	up to 80
E =====>	up to 25	up to 50	up to 100

1.2 Arguments

Table 2: CLI Arguments table

description	CLI argument	environment variable	example value
Tenant ID (required)	*–tenant-id*	*AZURE_LABELER_TENANT_ID*	*00000000-0000-0000-0000-000000000000*
Path to export the results	*–export-path*	*AZURE_LABELER_EXPORT_PATH*	*./path* or Storage Account Url with SAS token *https://sa.blob.windows.net/container/?sas_token*
Export only number of findings and energy label	*–export-metrics*	*AZURE_LABELER_EXPORT_METRICS*	*false* (default)
Export all findings information along with energy label	*–export-all*	*AZURE_LABELER_EXPORT_ALL*	*true* (default)
Regulatory frameworks to take into account	*–frameworks*	*AZURE_LABELER_FRAMEWORKS*	*'Microsoft cloud security benchmark,Azure CIS 1.1.0'*
Explicit list of subscriptions to take into account	*–allowed-subscription-ids*	*AZURE_LABELER_ALLOWED_SUBSCRIPTION_IDS*	*'00000000-0000-0000-0000-000000000000,00000000-0000-0000-0000-000000000001'*
Explicit list of subscriptions NOT to take into account	*–denied-subscription-ids*	*AZURE_LABELER_DENIED_SUBSCRIPTION_IDS*	*'00000000-0000-0000-0000-000000000000,00000000-0000-0000-0000-000000000001'*
List of resource groups to exclude	*–denied-resource-group-names*	*AZURE_LABELER_DENIED_RESOURCE_GROUP_NAMES*	*'SBP-WEU-AARC-01-RSG,SBP-WEU-AARC-01-RSG'*
Level of log printing	*–log-level*	*AZURE_LABELER_LOG_LEVEL*	*info*
Logging configuration	*–log-config*	*AZURE_LABELER_LOG_CONFIG*	

1.3 Supported authentication types

1.3.1 Azure CLI

If you are running the Energy Labeler from your local machine, make sure the user you are authenticated as has the *Security Reader* permission or higher.

```
az login --tenant 00000000-0000-0000-0000-000000000000
```

1.3.2 Managed Identity

If you are running the *azureenergylabeler* container in Azure (on ACI, ACA, etc), this is safest and preferred authentication method. To make use of Managed Identity authentication for the Energy Labeler, make sure it is enabled on your instance (ACI, Function App, etc): .. code-block:

```
identity: {
    type: 'SystemAssigned'

}
```

Also make sure you have a role assignment to your instance, *Security Reader* is required. .. code-block:

```
@description('Security Reader role definition')
var roleDefinitionId = resourceId('microsoft.authorization/roleDefinitions', '39bc4728-
↪0917-49c7-9d2c-d95423bc2eb4')

@description('Assign Security Reader role to the container so it can gather security␣
↪compliance of the subscription/tenant')
resource securityReaderAssignment 'Microsoft.Authorization/roleAssignments@2022-04-01' =
↪{
  name: guid(name)
  scope: tenant()
  properties: {
    principalId: containergroup.identity.principalId
    roleDefinitionId: roleDefinitionId
  }
}
```

1.3.3 Service Principal credentials

If you are running the *azureenergylabeler* container outside Azure, you need to authenticate to Azure using Service Principal credentials. The Service Principal therefore must have *Security Reader* permission assigned to either at Tenant Level or to the subscriptions where Energy Label are calculated.

Service principal with secret

Table 3: CLI Arguments table

variable name	value
AZURE_CLIENT_ID	id of an Azure Active Directory application
AZURE_TENANT_ID	id of the application's Azure Active Directory tenant
AZURE_CLIENT_SECRET	one of the application's client secrets

Service principal with certificate

Table 4: CLI Arguments table

variable name	value
AZURE_CLIENT_ID	id of an Azure Active Directory application
AZURE_TENANT_ID	id of the application's Azure Active Directory tenant
AZURE_CLIENT_CERTIFICATE_PATH	path to a PEM or PKCS12 certificate file including private key
AZURE_CLIENT_CERTIFICATE_PASSWORD	password of the certificate file, if any

1.4 Installation

1.4.1 Pipx

```
pipx install azureenergylabelercli
  installed package azureenergylabelercli 1.0.0, installed using Python 3.10.5
  These apps are now globally available
    - azure-energy-labeler
    - azure_energy_labeler_cli.py
done!
```

1.5 Examples

1.5.1 Calculate energy label for a tenant

```
azure-energy-labeler --tenant-id <TENANT_ID>
```

1.5.2 Calculate energy label for two subscriptions in a tenant

```
azure-energy-labeler --tenant-id <TENANT_ID> --allowed-subscription-ids 00000000-0000-
↪0000-0000-000000000000,00000000-0000-0000-0000-000000000001
```

1.5.3 Calculate energy label for a tenant and export all findings to a local folder

```
azure-energy-labeler --tenant-id 2ba489e8-3466-4f52-a32d-263d28b832e1 --export-path /tmp/
↪ --export-all
```

1.5.4 Calculate energy label for a tenant and export all findings to a Storage Account Blob Container

```
azure-energy-labeler --tenant-id 2ba489e8-3466-4f52-a32d-263d28b832e1 --export-path
↪"https://sa.blob.windows.net/container/?sas_token" --export-all
```

1.6 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

$ _tag –minor

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :))
  - code
  - lint
  - test
- commit and push
- develop more through the code-lint-test cycle
- tag (with the appropriate argument)
- build

- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

1.7 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

```
$ python setup.py sdist bdist_egg
```

as this will produce an unusable artifact with files missing. Instead use the provided build and upload scripts that create all the necessary files in the artifact.

1.8 Project Features

- TODO

CHAPTER

TWO

INSTALLATION

At the command line:

```
$ pip install azureenergylabelercli
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv azureenergylabelercli
$ pip install azureenergylabelercli
```

Or, if you are using pipenv:

```
$ pipenv install azureenergylabelercli
```

Or, if you are using pipx:

```
$ pipx install azureenergylabelercli
```

CHAPTER

THREE

USAGE

To develop on azureenergylabelercli:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scripts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scripts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use azureenergylabelercli in a project:

```
from azureenergylabelercli import Azureenergylabelercli
azureenergylabelercli = Azureenergylabelercli()
```

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

4.1.1 Get Started!

Ready to contribute? Here's how to set up *azureenergylabelercli* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

```
$ git clone https://github.com/schubergphilis/azureenergylabelercli
```

2. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your clone for local development:

```
$ cd azureenergylabelercli/
$ pipenv install --ignore-pipfile
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER

FIVE

AZUREENERGYLABELERCLI

5.1 azureenergylabelercli package

5.1.1 Submodules

5.1.2 azureenergylabelercli.azureenergylabelercli module

Main code for azureenergylabelercli.

azureenergylabelercli.azureenergylabelercli.**comma_delimited_list**(*argument*, *sep=','*)

Takes a str, splits based on character and returns a list.

azureenergylabelercli.azureenergylabelercli.**get_arguments**()

Gets us the cli arguments.

Returns the args as parsed from the argsparser.

azureenergylabelercli.azureenergylabelercli.**get_subscription_reporting_data**(*tenant_id*, *subscription_id*, *export_all_data_flag*, *frameworks*, *log_level*, *disable_spinner*)

Gets the reporting data for a single account.

Parameters

- **tenant_id** – Tenant Id of the tenant
- **subscription_id** – The ID of the subscription to get reporting on.
- **export_all_data_flag** – If set all data is going to be exported, else only basic reporting.
- **frameworks** – The frameworks to include in scoring.
- **log_level** – The log level set.
- **disable_spinner** – The spinner will be disabled while retrieving the findings.

Returns

report_data, exporter_arguments

azureenergylabelercli.azureenergylabelercli.**get_tenant_reporting_data**(*tenant_id*, *allowed_subscription_ids*, *denied_subscription_ids*, *denied_resource_group_names*, *export_all_data_flag*, *frameworks*, *log_level*, *disable_spinner*)

Gets the reporting data for a landing zone.

Parameters

- **tenant_id** – Tenant Id of the tenant
- **allowed_subscription_ids** – The allowed subscription ids for tenant inclusion if any.
- **denied_subscription_ids** – The denied subscription ids for tenant zone exclusion if any.
- **denied_resource_group_names** – List of resource groups to exclude if any.
- **export_all_data_flag** – If set all data is going to be exported, else only basic reporting.
- **frameworks** – The frameworks to include in scoring.
- **log_level** – The log level set.
- **disable_spinner** – The spinner will be disabled while retrieving the findings.

Returns
report_data, exporter_arguments

azureenergylabelercli.azureenergylabelercli.**setup_logging**(*level*, *config_file=None*)

Sets up the logging.

Needs the args to get the log level supplied

Parameters

- **level** – At which level do we log
- **config_file** – Configuration to use

azureenergylabelercli.azureenergylabelercli.**wait_for_findings**(*method_name*, *method_argument*, *log_level*, *disable_spinner=False*)

If log level is not debug shows a spinner while the callable provided gets security hub findings.

Parameters

- **method_name** – The method to execute while waiting.
- **method_argument** – The argument to pass to the method.
- **log_level** – The log level as set by the user.
- **disable_spinner** – The spinner will be disabled while retrieving the findings.

Returns
A list of defender for cloud findings as retrieved by the callable.

Return type
findings

5.1.3 azureenergylabelercli.azureenergylabelercliexceptions module

Custom exception code for azureenergylabelercli.

exception azureenergylabelercli.azureenergylabelercliexceptions.**MissingRequiredArguments**

Bases: Exception

Missing a required argument.

exception azureenergylabelercli.azureenergylabelercliexceptions.**MutuallyExclusiveArguments**

Bases: Exception

Mutually exclusive variables are set.

5.1.4 azureenergylabelercli.validators module

Main code for validators.

class azureenergylabelercli.validators.**ValidatePath**(*option_strings*, *dest*, *nargs=None*, ***kwargs*)

Bases: Action

Validates a given path.

azureenergylabelercli.validators.**azure_subscription_id**(*subscription_id*)

Setting a type for an subscription id argument.

azureenergylabelercli.validators.**get_mutually_exclusive_args**(*arg1*, *arg2*, *required=False*, *msg=None*)

Test if multiple mutually exclusive arguments are provided.

Parameters

- **arg1** (*Any*) – First argument to be checked
- **arg2** (*Any*) – Second argument to be checked
- **required** (*bool, optional*) – Wether one argument is required. Defaults to False.
- **msg** (*str, optional*) – Error message shown to the user. Defaults to None.

Raises

- *MutuallyExclusiveArguments* – If both arguments were provided
- *MissingRequiredArguments* – If *required* is True and no argument was provided

Returns

arg1 and arg2 after validation

5.1.5 Module contents

azureenergylabelercli package.

Import all parts from azureenergylabelercli here

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Sayantan Khanra <skhanra@schubergphilis.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

CHAPTER

EIGHT

0.0.1 (04-05-2022)

- First code creation

CHAPTER

NINE

0.1.0 (27-06-2022)

- First release

CHAPTER

TEN

1.0.0 (22-09-2022)

- Add support for environment variables as default argument value
- Arguments with array of values changed from a space separated list to a comma separated list
- CLI now uses the most recent version of the azureenergylabelerlib

CHAPTER

ELEVEN

1.0.1 (26-09-2022)

- Fixed a bug preventing collection of resource group findings

CHAPTER

TWELVE

1.1.0 (04-10-2022)

- Updated dependency azurenergylabelerlib to version 2.0.0

CHAPTER

THIRTEEN

2.0.0 (19-10-2022)

- Microsoft renamed “Azure Security Benchmark” to “Microsoft cloud security benchmark”, changing the interface

CHAPTER

FOURTEEN

2.0.1 (20-10-2022)

- Fix broken dependecies

CHAPTER

FIFTEEN

2.0.2 (08-03-2023)

- Bump template and dependencies.

CHAPTER

SIXTEEN

2.0.3 (21-03-2023)

- Bumped library version which now filters subscriptions based on the tenant_id.

CHAPTER

SEVENTEEN

2.1.0 (15-05-2023)

- Added option to disable banner and spinner
- Improved filtering of findings

CHAPTER

EIGHTEEN

2.1.1 (28-06-2023)

- Updated library dependency

CHAPTER

NINETEEN

2.2.0 (22-09-2023)

- feat: updating azureenergylabelerlib to 3.3.0 to allow excluding resource groups

CHAPTER

TWENTY

2.2.1 (22-09-2023)

- Fix: AZURE_LABELER_DENIED_RESOURCE_GROUP_NAMES changed to a string delimited list due to gitlab-ci not supporting variables of the type list.

CHAPTER

TWENTYONE

2.2.2 (22-09-2023)

- fix: the list syntax in the readme file broke the release. It expects a comma after a quote.

CHAPTER

TWENTYTWO

2.2.3 (22-09-2023)

- fix: the list syntax in the readme file broke the release. It expects a comma after a quote.

CHAPTER

TWENTYTHREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX