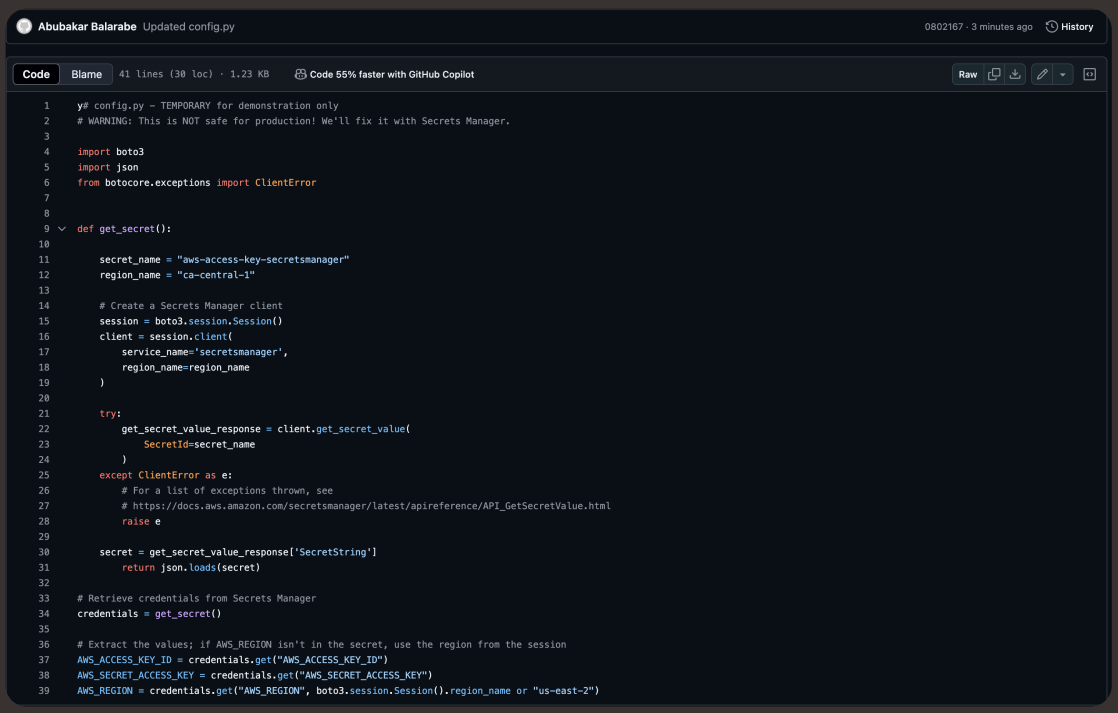
nextwork.org

Secure Secrets with Secrets Manager

TA

tahirgroot@gmail.com

https://medium.com/@tahirbalarabe2/securing-aw...



TA
tahirgroot@gmail.com
NextWork Student
NextWork.org

Introducing Today's Project!

In this project, I will demonstrate how to use AWS Secrets Manger for secure credentials management. I'm doing this project to learn how to protect our credentials when we connect live (in production) apps to our AWS environment and databases. etc.

Tools and concepts

Services I used were Secrets Manager, Github, S3, IAM. Key concepts I learnt include GitHub secret scanning, the risks of hardcoding credentials, using git rebase to rewrite the commit history of a repository, forking a repository.

Project reflection

This project took me approximately 2 hours including demo time The most challenging part was git rebase (an extra bit of effor at the endof the project to push the code to repo. It was most rewarding to review the git coommit history, and see that you can really find AWS credentials in the history (even if the most updated version of that file no longer uses those credentials).

TA

tahirgroot@gmail.com
NextWork Student

NextWork.org

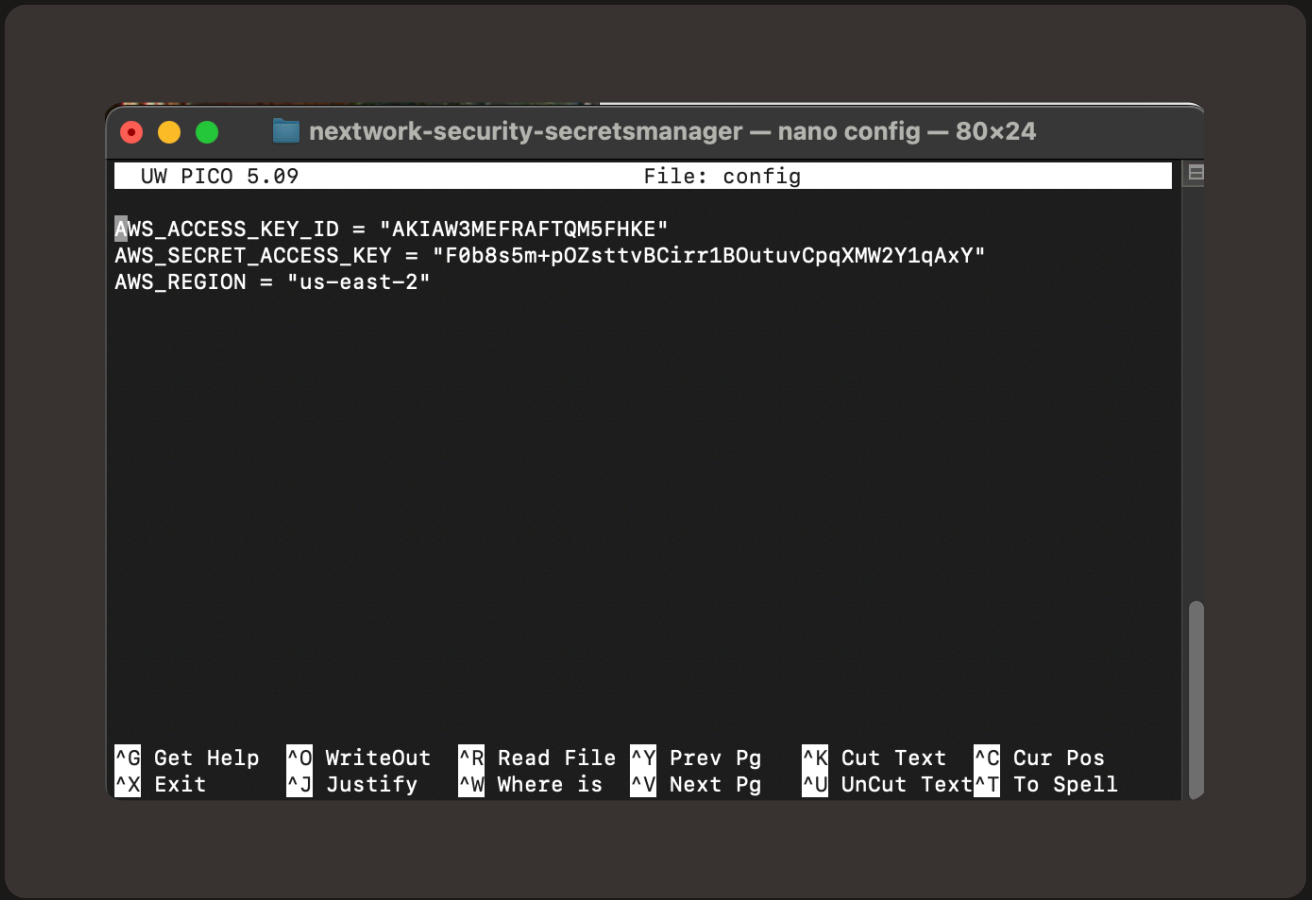
I did this project to learn more about protecting/securing my code using services like AWS Secrets Manager. This project met my goals as I am now practising credentials management and helped to build skills in Git(e.g. rebasing) along the way.

TA
tahirgroot@gmail.com
NextWork Student
NextWork.org

Hardcoding credentials

In this project, a sample web app is exposing the developers' AWS credentials i.e their access key ID, secret access key. It is unsafe to harcode credentials because once the code is made public in the repository, anyone can get access to those credentials, and access AWS environment. e.g delete resources, steal data. e.t.c

I've set up the initial configuration with a set of Access Key Id and Secret Access Key. These credentials are just examples because using test credentials prevents exposing my actual AWS credentials access while doing this project.



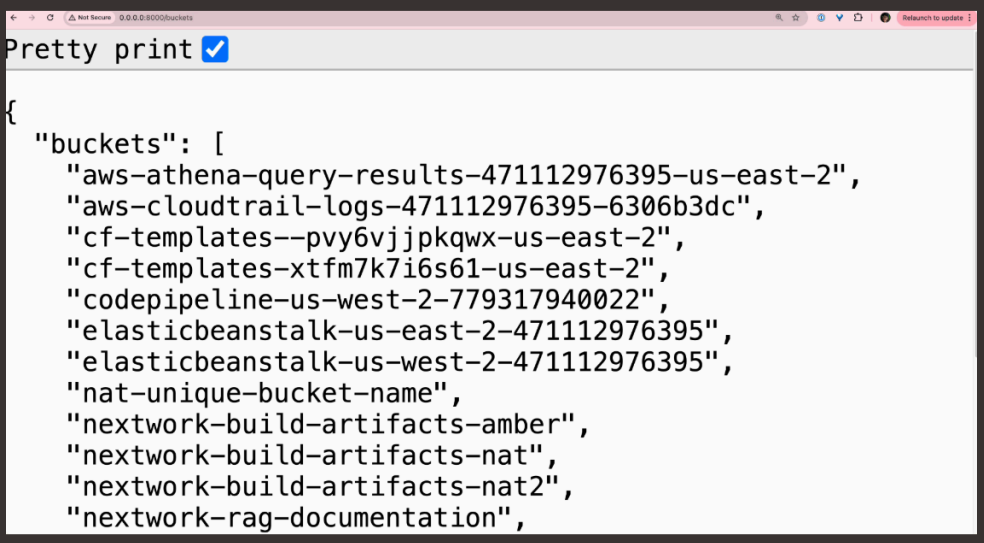
TA
tahirgroot@gmail.com
NextWork Student

Using my own AWS credentials

As an extension for this project, I also decided torun the web app locally using my own AWS credentials to set up my virtual environment, I installed pacakges like boto3, fast -api, which are packages that my app''s main file(app.py) uses to connect our web app to AWS.

When I first ran the app, I ran into an error because we were using test credentials inside config.py those credentials dont actually point to a real AWS account.

To resolve the 'InvalidAccessKeyId' error, I updated the configuration file (config.py) to use my account's access key ID and secret access key - no more test credentials



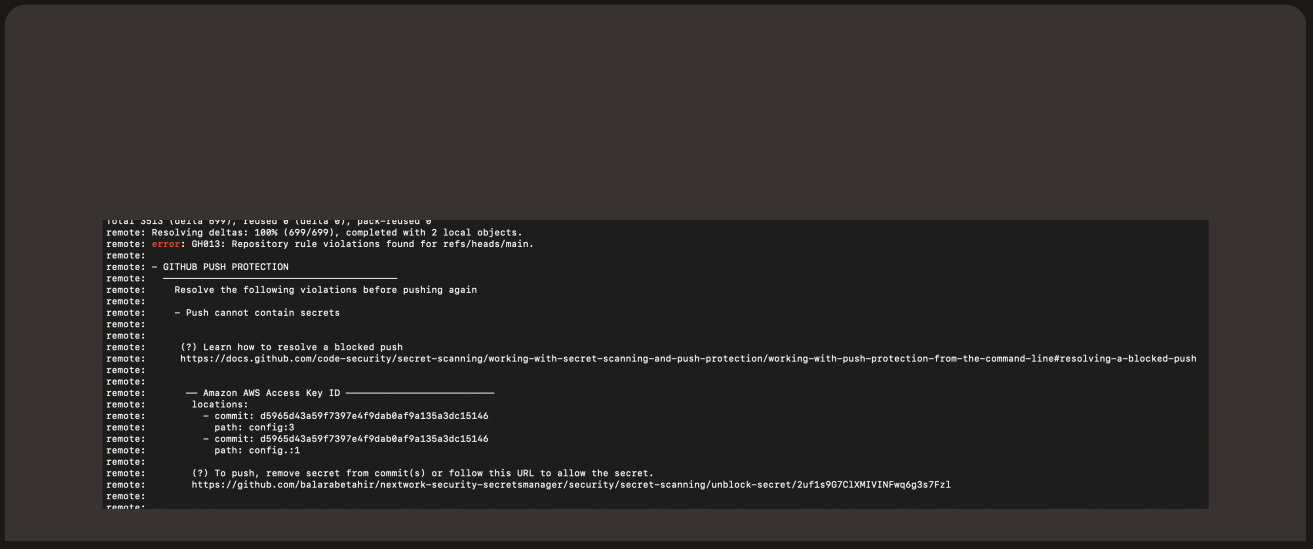
TA
tahirgroot@gmail.com
NextWork Student
NextWork.org

Pushing Insecure Code to GitHub

Once I updated the web app code with credentials, I forked the repository to simulate what its like to push insecure code into your public repo. A fork is different from a clone because it also creates a new repository in my Github account (where as a clone just copies that codes into your local computer).

To connect my local repository to the forked repository, I ran the command 'git remote set -url origin'. Then I used git add and git commit to save the changes we made to config.py. Finally, git.push uploads those changes to the remote origin(i.e. the forked repository).

GitHub blocked my push because its secret scanning failures detatced that we are hardcoding AWS credentials. This is a good security feature because prevents us from revealing sensitive information in a public repository.



TA
tahirgroot@gmail.com
NextWork Student

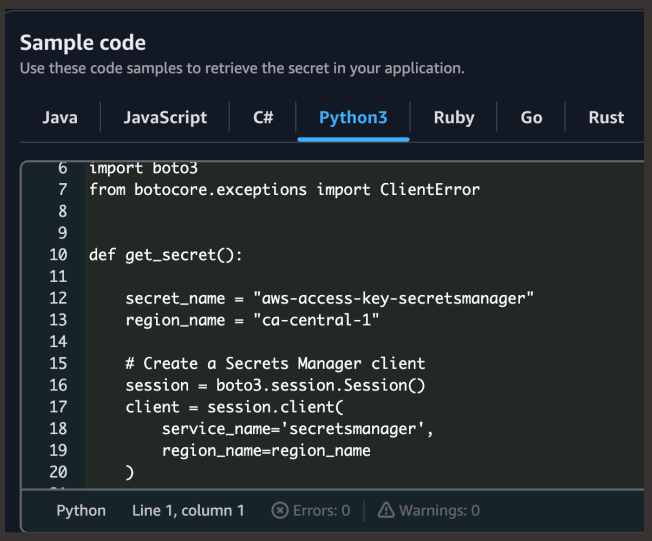
Secrets Manager

Secrets Manager is an AWS service that helps with securely storing and retrieving secrets e.g. database credentials, API keys and anything else I'm using it to store AWS access key ID and secret access key. Other common use cases include and OAuth keys and even account IDs or URLs i dont want other people to see in our code.

Another feature in Secrets Manager is secrets rotation, which means the automatic rotation of the value of secrets on a consistent schedule. Secrets Manager will generate new value for those secrets, and its useful for high-risk situations where you'd want to minimise how long long an attacker has access to your databases/API keys. if it ever infiltrates my credentials.

Secrets Manager provides sample code in various languages, like Java, Python3 and more. This is helpful because i can use this code samples directly in my config.py file to update the way I retrieve credentials. Now i can use functions to help me connect to secrest Manager.

TA
tahirgroot@gmail.com
NextWork Student



TA
tahirgroot@gmail.com
NextWork Student

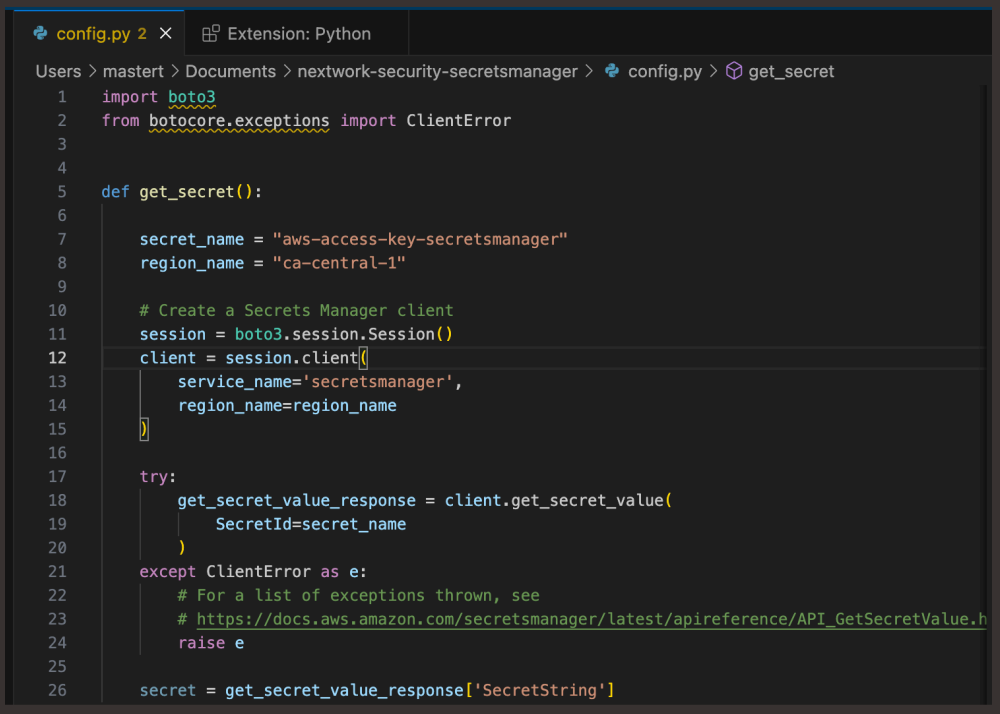
Updating the web app code

I updated the config.py file to retrieve the secret astored in my Secrets Manager programatically. The get_secret() function will set up a connection to Secrets Manager (using boto3), and then it will try to retrive the value of the secret (with the secret name that we create when we set it up). and store that secret's value in a variable called 'secret'.

I also added code to config.py to extract values of the secrets that we stored in Secrets manager. This is important because the configuration file no longer has coded credentials inside. it uses the function suggested by the code sample and splits up the secrets value to store the access key ID and secret access key in the same variables that our app.py.

TA

tahirgroot@gmail.com
NextWork Student



TA
tahirgroot@gmail.com
NextWork Student

Rebasing the repository

Git rebasing is the process of changing the commit history of our repository. I used it to drop the commit where we hardcoded the AWS credentials. This was necessary because hardcoding our AWS credentials are still living somewhwere in our repo commit history otherwise(and if its living in the commit history, it's still discoverable to attackers).

A merge conflict occurred during rebasing because Git is now used confused about wether it should also remove all the changes to config.py i made in a commit After the commit I wanted to drop. I resolved the merge conflict by going directly to the config.py file, and manually removing all the parts I want to remove (hardcoded credentials), and keeping the code from Secrets Manager.

Once i updated config.py, I verified that my Github repo is not exposing any of my credenetials. I visited the config.py file in the forked repo and can see that it uses code to extract my credentials instead of any hard coding. I also successfuly pushmy code without any blocking from Github secrets scanning.

```
remote:
remote:
remote:        —— Amazon AWS Access Key ID ——————————————————
remote:         locations:
remote:           - commit: df9ce6148a812adf97557643b66a7b671ec9d81d
remote:             path: config.py:4
remote:
remote:         (?) To push, remove secret from commit(s) or follow this URL to allow t
he secret.
remote:         https://github.com/natashaongiscoding/nextwork-security-secretsmanager/
security/secret-scanning/unblock-secret/2tSWZCJfTOytXj59tXpeia5g5Ly
remote:
remote:
remote:        —— Amazon AWS Secret Access Key ——————————————————
remote:         locations:
remote:           - commit: df9ce6148a812adf97557643b66a7b671ec9d81d
remote:             path: config.py:5
remote:
remote:         (?) To push, remove secret from commit(s) or follow this URL to allow t
he secret.
remote:         https://github.com/natashaongiscoding/nextwork-security-secretsmanager/
security/secret-scanning/unblock-secret/2tSWZAuVx1PyKvhCzbmZd9KyzvJ
remote:
remote:
```

NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for more projects