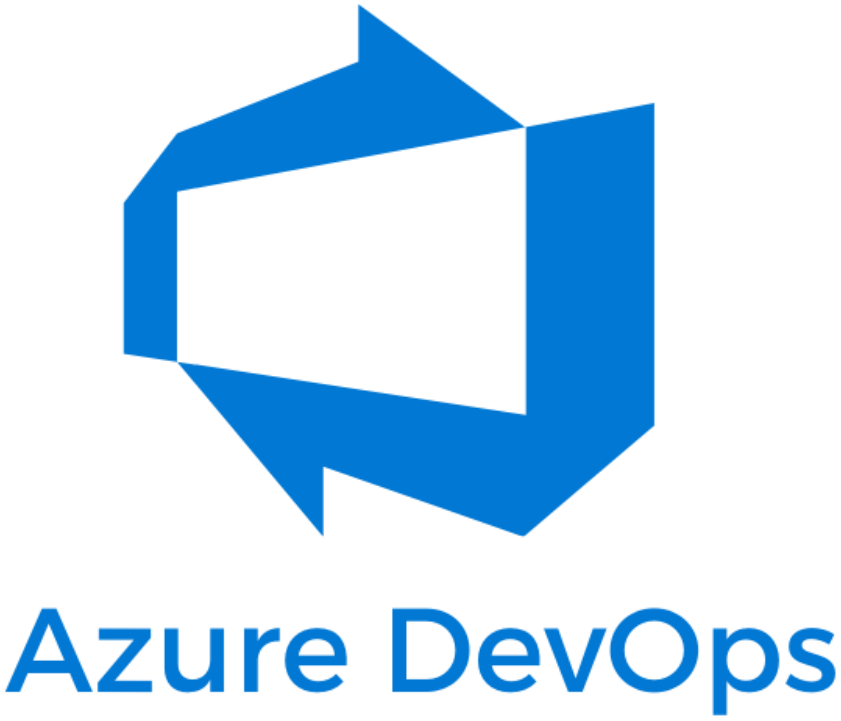
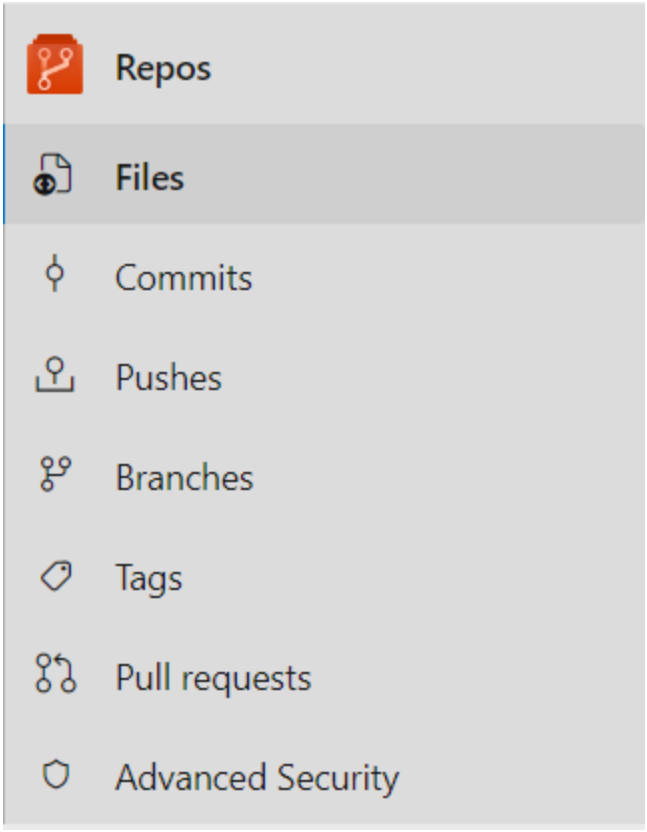
Azure DevOps



Introduction to Azure DevOps:

Simply put, it is a group of services provided by Microsoft that benefit the DevOps Engineer in one place, which in turn increases performance efficiency.

Here are some key points about Azure DevOps:



Azure Repos:

Version control systems help you track changes you make in your code over time. As you edit your code. We can create a connection between it and GitHub to retrieve repositories from it.

- **Files**:
where you manage the actual code files within your repository

- **Commits**:
section shows a history of changes made to the repository. Each commit represents a specific set of changes (additions, deletions, modifications) made to files.

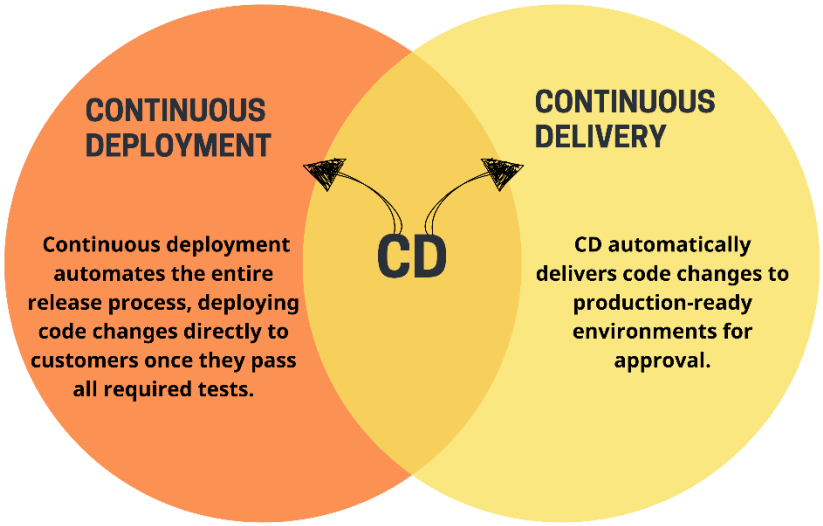
- **Pushes**:
share changes in commits and branches

- **Branches**:
a Branches keep a history of commits and provide a way to isolate changes for a feature or a bug fix from your master branch and other work. Committing changes to a branch doesn't affect other branches. You can push and share branches with other people on your team without having to merge the changes into master.

- Pull requests:
Pull requests let your team review code and give feedback on changes before being merged into the master branch. Reviewers can step through the proposed changes, leave comments, and vote to approve or reject the code.

Azure Pipeline:

A pipeline defines the continuous integration and deployment process for your app.
now we should define what CI/CD:
Continuous integration (CI) is the practice used by development teams to simplify the testing and building of code. CI helps to catch bugs or problems early in the development cycle, which makes them easier and faster to fix. Automated tests and builds are run as part of the CI process. The process can run on a set schedule, whenever code is pushed, or both. Items known as artifacts are produced from CI systems. They're used by the continuous delivery release pipelines to drive automatic deployments.



Azure Board:

Boards this is the place where you will plan and track the work that needs to be done. when we create this, we will choose from one of four formal work processes (Basic, Agile, Scrum, CMMI)

Basic

Choose Basic when your team wants the simplest model that uses Issues, Tasks, and Epics to track work. Note: Basic is currently in a selective Preview for new users of Azure Boards only.

Agile

Choose Agile when your team uses Agile planning methods, including Scrum, and tracks development and test activities separately. This process works great if you want to track user stories and (optionally) bugs on the Kanban board, or track bugs and tasks on the taskboard.

Scrum

Choose Scrum when your team practices Scrum. This process works great if you want to track product backlog items (PBIs) and bugs on the Kanban board, or break PBIs and bugs down into tasks on the taskboard.

CMMI

Choose CMMI when your team follows more formal project methods that require a framework for process improvement and an auditable record of decisions. With this process, you can track requirements, change requests, risks, and reviews.

CMMI process supports formal change management activities. Tasks support tracking Original Estimate, Remaining Work, and Completed Work.

Azure Test Plans:

There are lots of test plans that you could incorp into an Azure DevOps workflow.

Unit tests

Perform tests of individual components and modules used by your software. Most often written by developers (rather than dedicated testers), unit tests are often cheap to automate, and can be run very quickly by a continuous integration server.

Integration tests

Verify that different modules or services used by your application work well together. Integration tests can validate communication with a database, or make sure that microservices work together as expected. These types of tests are more expensive to run when compared to Unit Tests, as they require multiple parts of the application to be up and running.

Functional tests

Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action. Compared to an integration test which may verify that you can query the database, a functional test would test whether a specific value from the database is returned.

End-to-end tests

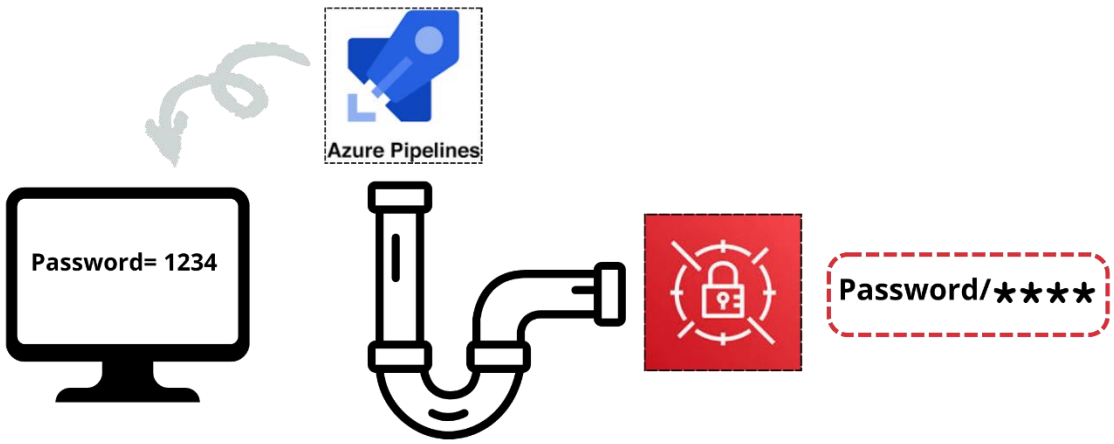
End-to-end testing simulates a user interacting with the software. It verifies that various user flows work as expected and can be as simple as loading a web page, logging in, or much more complex scenarios verifying email notifications, online payments, etc End-to-end tests are useful, but expensive to perform, and can be hard to maintain when automated. It's recommended to have a few key end-to-end tests and rely more on lower-level types of testing (unit and integration tests) to be able to quickly identify breaking changes.

Acceptance testing

Acceptance tests are formal tests executed to verify if a system satisfies business requirements. They require the entire application to be up and running and focus on replicating user behaviors. They can also measure performance and reject changes if certain goals are not met.

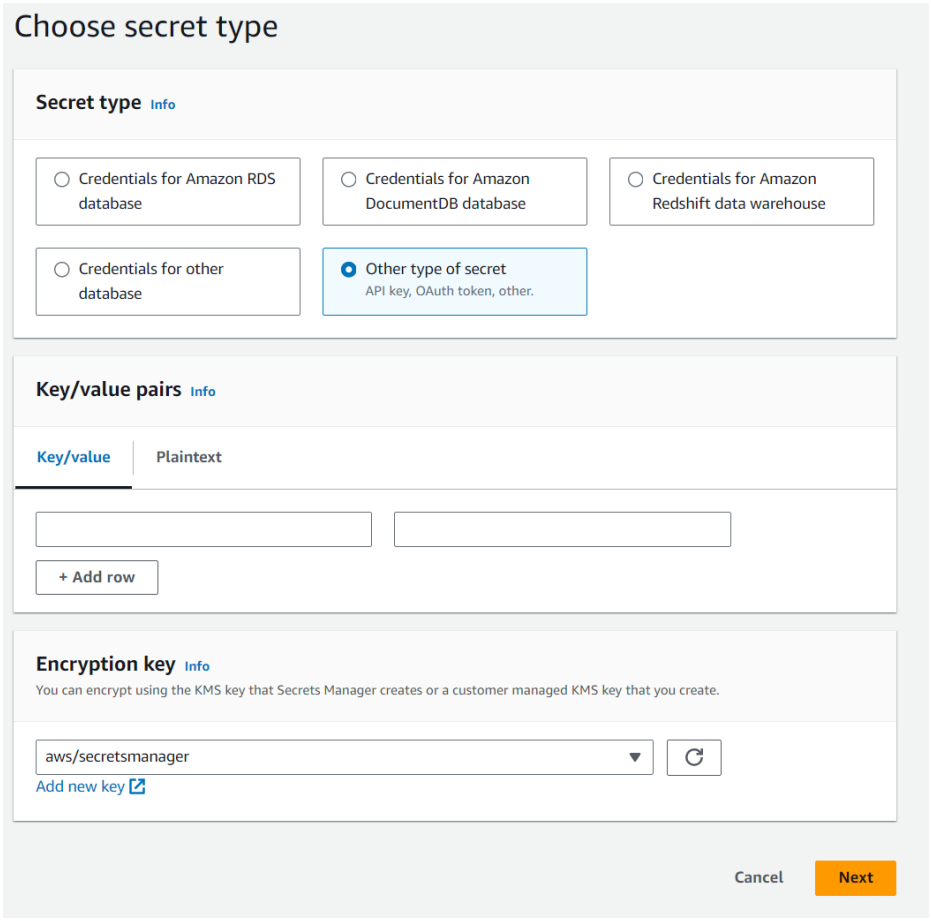
Task 1: USE Secret Manager in Pipeline

- **Description:**
  I will connect the pipeline with AWS secret manager service to retrieve the password that I previously stored.



- **The steps:**
  - Log in to the Secrets Manager console
  - Store a new secret

- Select the secret type " In our case I will choose Other type of secret”



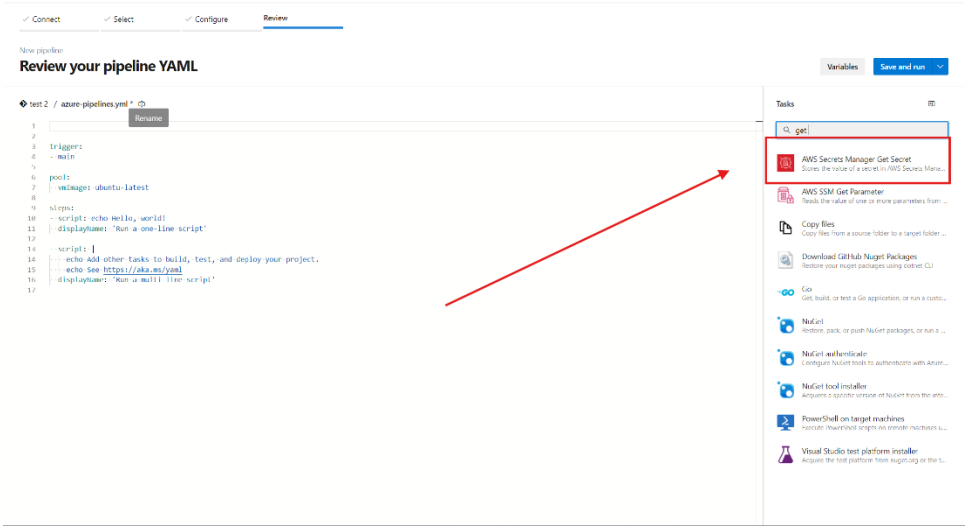
- Enter the secret value

- Choose Secret name and description

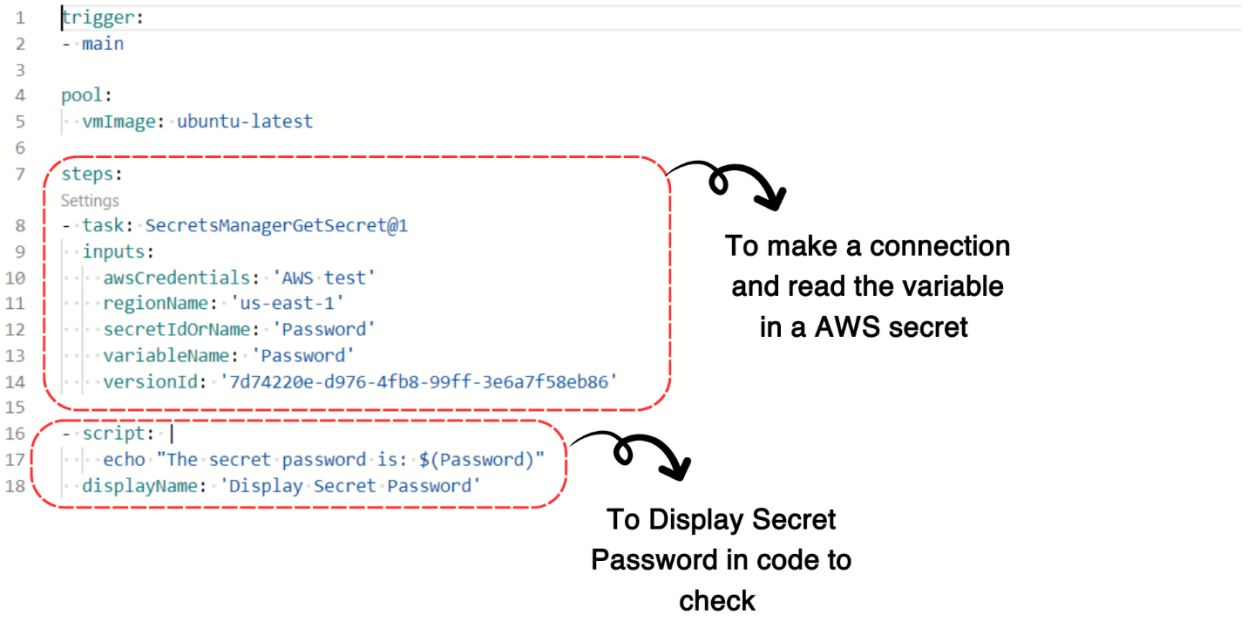
now i have secret

Then i need to create the pipeline to get this secret :

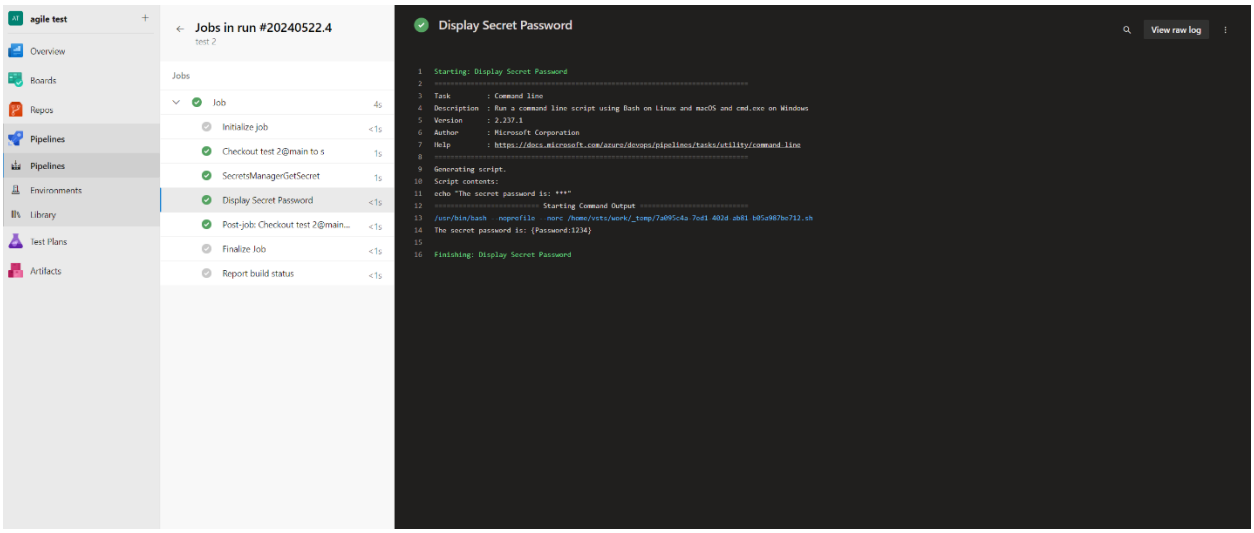
- go to Azure DevOps to create pipeline and cofigure it



- Edit the yaml code to make connection with AWS Secret

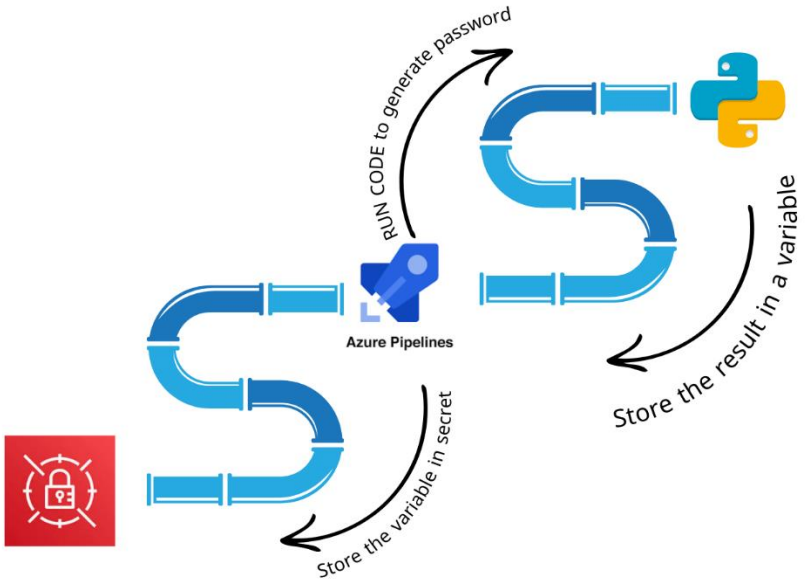


- now i can run the pipeline and check :


agile test
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Jobs in run #20240522.4
test 2
Jobs
Job 4s
Initialize job <1s
Checkout test 2@main to s 1s
SecretsManagerGetSecret 1s
Display Secret Password <1s
Post-job: Checkout test 2@main... <1s
Finalize Job <1s
Report build status <1s
Display Secret Password
View raw log
Starting: Display Secret Password
Task : Command line
Description : Run a command line script using Bash on Linux and macOS and cmd.exe on Windows
Version : 2.237.1
Author : Microsoft Corporation
Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/command_line
Generating script.
Script contents:
echo "The secret password is: ***"
Starting Command Output
/usr/bin/bash --noprofile --norc /home/vsts/work/_temp/7a095c4a-7ed1-402d-ab81-b85a987bc712.sh
The secret password is: {Password:1234}
Finishing: Display Secret Password

Task 2: change variable in Secret using pipeline

Description: I will use python code to generated password and then use pipeline to store it in secret manager “change from run to run”



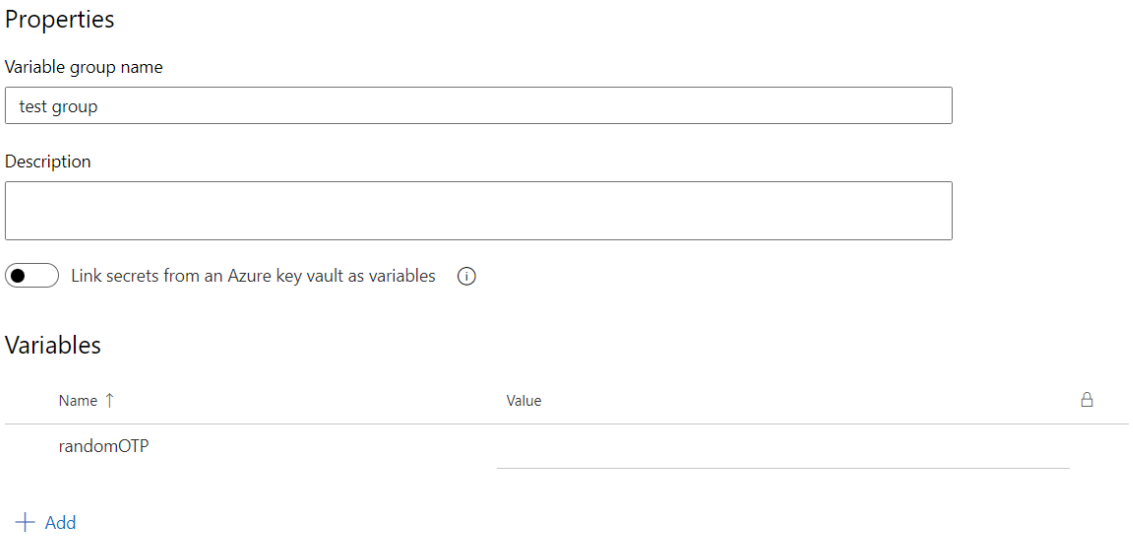
Steps:

- i want to write python code to generate a number consisting of four digits and add this file to me project repos .

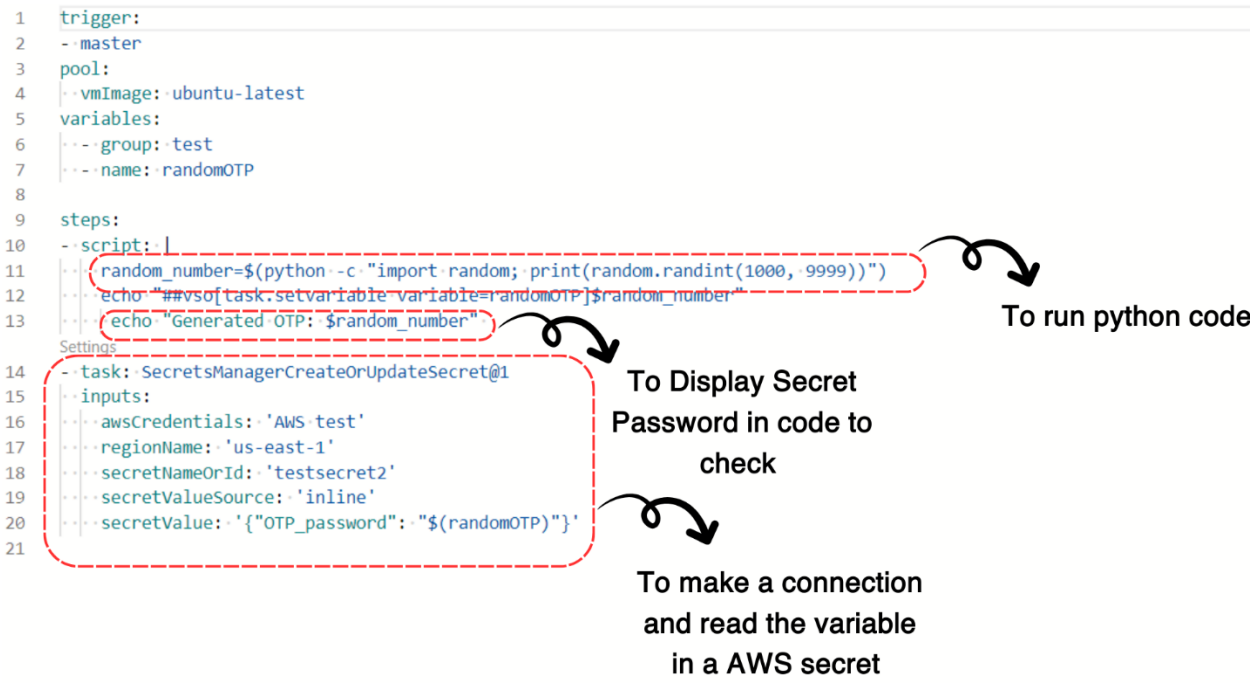
```python
import random

def generate_random_number():
    return random.randint(1000, 9999)

if __name__ == "__main__":
    print(generate_random_number())

```

then I want to use it in yaml code when create pipeline

- go to section Library to create variable which in turn will store the result of the code in it but first i want to create variable group and then variable



- Edit the yaml code to run python code and make connection with AWS Secret



- Run the pipeline to store the new variable in secrete with every run