# Python Code Quality & Security Implementation Guide

## Overview

The requirements were:

Create CI/CD workflow that included Python Version testing, Dependency Management, code quality and security as well as full deployment.

The following is an MVP that covers pre-commit to final deployment of a Docker image to my Docker Hub account.

Use of GitHub Advanced Security, Azure security and the use of environments and branch protection rules along with best practices will add/delete to this as appropriate.

My MVP contains the following:

## Local Development (Pre-commit)

### Pre-commit Hook Configuration

* Precommit hooks built in
  + id: trailing-whitespace
  + id: end-of-file-fixer
  + id: check-yaml
  + id: check-added-large-files
  + id: check-merge-conflict
  + id: debug-statements
* Pyright is a powerful, fast, and feature-packed static type checker explicitly designed for Python. It helps to ensure code quality, catch errors early, and boost productivity through static type checking. MyPy can also be used.
* **Ruff** linting, formatting and fixing.
* **conventional pre-commit** ensures commit messages to provide helpful annotations as well as being used in automatic version numbering.
* **Custom python-script** to provide LEAK DETECTION CUSTOM. Home made enabling developers to adjust as needed to prevent friction. There are 3rd party services we can use.
* **Bandit** provides security checking.

*This is amendable.*

### Conventional Commit Messages

This provides structure around what has been committed and can be used for versioning. These commit messages can also be used to add information to the CHANGELOG.md file as we can search the Git database for filtered commit messages.

### Dependency Management

For a given commit, UV enables a uv.lock file to be saved that pins all dependencies.

A screenshot of a product

AI-generated content may be incorrect.

A screenshot of a computer error message

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

UV is a more powerful resolver algorithm than pip and we can have a versioning system for uv.lock for each release.

## CI/CD Pipeline (GitHub Actions)

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

### Linting, typing and formatting checks

Similar to those at the pre-commit stage.

### Build Matrix Strategy

* Set up matrix testing across Python versions (3.10, 3.11, 3.12).
* Test across Ubuntu only.
* Install dependencies with caching for faster build times.
* Download Pytest-HTML reports and coverage reports.

### Code and Security Quality Checks

* Unit tests/Coverage, CodeQL, Bandit, Safety, Pip-audit in CI.
* Scan for secrets and credentials in commit history and codebase.

### Docker

The MVP can create Docker images across many Python versions if needed, scan for security and deploy to Azure if needed but uses my Docker Hub account in lieu of access to Azure.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black lines

AI-generated content may be incorrect.

We use Trivy and Docker Scout, inbuilt in GitHub Actions, to scan Docker images for vulnerabilities and secrets, with a downloadable report.

We can run matric Python version for the Dockerfile using arguments: –build-arg PYTHON\_VERSION=3.10 etc. 02\_CI/DockerfileMultiple and 02/cicd\_pipeline.yaml.

### Summary

This MVP is sufficient but not necessarily complete.

It covers the requirements stated at the beginning of the document.

Processes and protocols will be formulated later to provide the best possible security measures that will need to be monitored and adjusted as needed.

### Work time

I would bill all this work as one day, subject to agreement.