pyats-05 - Running pyATS tests in Docker

Syllabus

- overview of standard pyATS image
- customize standard pyATS container
- build custom image

Presentation

Overview of standard pyATS image

Default configuration

```
docker run -it ciscotestautomation/pyats:latest
```

Run a test

```
# rabbit.py
from pyats.aetest import Testcase, setup, test, cleanup, main


class SmokeTest(Testcase):

    @setup
    def setup(self):
        print("A setup of smoke test")

    @test
    def test_1(self):
        print('Test #1')

    @test
    def test_2(self):
        print('Test #2')

    @cleanup
    def cleanup(self):
        print("A cleanup of smoke test")



if __name__ == '__main__':
    main()
```

```
# manipulation within container
docker run -it -v $(pwd):/pyats/code \
       ciscotestautomation/pyats:latest /bin/bash
# run using container
docker run -it -v $(pwd):/pyats/code \
       ciscotestautomation/pyats:latest python code/rabbit.py
```

Customize standard pyATS container

```
# requirements.txt
selenium

# deps.sh
#!/bin/bash
echo "Some additional configurations...."
echo "Install missing packages...."

# command
docker run -it \
    -v $(pwd)/rabbit.py:/pyats/rabbit.py \
    -v $(pwd)/requirements.txt:/pyats/requirements.txt \
    -v $(pwd)/deps.sh:/pyats/workspace.init \
    ciscotestautomation/pyats:latest python rabbit.py
```

Build custom image

1. Project packages

    ```
    # requirements.txt
    pyats
    ```

2. A test

    ```python
    # rabbit.py
    from pyats.aetest import Testcase, setup, test, cleanup, main


    class SmokeTest(Testcase):

        @setup
        def setup(self):
            print("A setup of smoke test")

        @test
        def test_1(self, word):
            print(f'Test #1: {word}')

        @test
        def test_2(self, word):
            print(f'Test #2: {word}')

        @cleanup
        def cleanup(self):
            print("A cleanup of smoke test")



    if __name__ == '__main__':
        main(word='alongggggggggggggword')
    ```

3. Docker configuration

    ```
    # Dockerfile
    FROM python:3.6.5-slim
    RUN apt-get update && apt-get install -y gcc && apt-get autoclean -y
    WORKDIR /my-tests
    COPY requirements.txt rabbit.py ./
    RUN pip install --upgrade pip wheel setuptools && \
        pip install --no-cache-dir -r requirements.txt && \
        rm -v requirements.txt
    VOLUME /my-tests/archive
    ENV TINI_VERSION v0.18.0
    ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/tini /tini
    RUN chmod +x /tini
    ENTRYPOINT ["/tini", "--"]
    CMD ["python", "rabbit.py"]
    ```

4. Build and execute

    ```
    # build
    docker build -t my-tests .
    # run
    docker run -it my-tests
    # compare
    docker images | grep -E "pyats|tests"
    ```

Additional materials

- https://github.com/CiscoTestAutomation/pyats-docker
- https://store.docker.com/community/images/ciscotestautomation/pyats
- https://github.com/krallin/tini

Control

Task description

Based on the materials from "Build custom image" section, you have to configure `rabbit.py` test execution with `easypy` module within a Docker container. To do this, you need to prepare a Docker image which will have configured a job's execution by default. This image has to be pushed to your Docker hub account.

The `word` parameter has to be passed to the job via either CLI argument or a testbed file (`testbed > custom > word` path).

Review items

Please send a command which will run your image and save `easypy`'s archive (with test report) in the local filesystem.