SWEAGLE integration to Azure Pipeline

SWEAGLE
KEEPING AN EAGLE EYE ON YOUR CONFIG DATA

Integration strategy

- Embed SWEAGLE CLI in Azure Pipeline Agent
- Use a containerized agent for easier deployment and maintenance
- You can do tenant setup dynamically at runtime

- Pros:
  - Follow Azure specific agent principles
  - Similar strategy than for GitlabCI or CircleCI
  - Pipeline will benefit from
    - already included CLI features
    - regular updates of SWEAGLE CLI

Prerequisite: Build the container

Prerequisite

- Container containing both Azure pipeline agent and Sweagle CLI must be created
- It is currently build and hosted in SWEAGLE docker registry
- You can build your own specific version if they have specific requirements

Steps followed to build this container are described by Microsoft here: https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/docker?view=azure-devopsn

Dockerfile

Azure Agent layers

Sweagle CLI layers

```
##############################################
#####      AZURE AGENT SPECIFIC LAYERS
##############################################
FROM ubuntu:16.04
# To make it easier for build and release pipelines to run apt-get,
# configure apt to not require confirmation (assume the -y argument by default)
ENV DEBIAN_FRONTEND=noninteractive
RUN echo "APT::Get::Assume-Yes \"true\";" > /etc/apt/apt.conf.d/90assumeyes
RUN apt-get update \
&& apt-get install -y --no-install-recommends \
       ca-certificates \
       curl \
       jq \
       git \
       iputils-ping \
       libcurl3 \
       libicu55 \
       libunwind8 \
       netcat
WORKDIR /azp
COPY ./start.sh .
RUN chmod +x start.sh
# CMD [ "./start.sh"]

##############################################
#####      SWEAGLE CLI SPECIFIC LAYERS
##############################################
COPY ./init-cli.sh /azp/
COPY ./package/ /usr/bin/
RUN chmod +x /usr/bin/sweagle && \
   chmod +x /azp/init-cli.sh
# note that init-cli.sh must call /start.sh for azure agent to work
CMD [ "./init-cli.sh"]
```

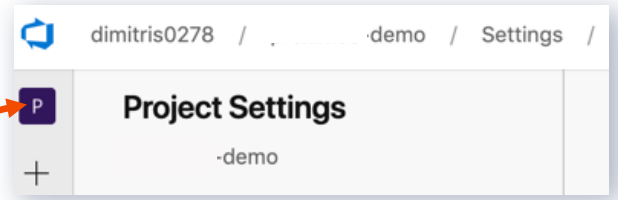
Setup: Define SWEAGLE Registry

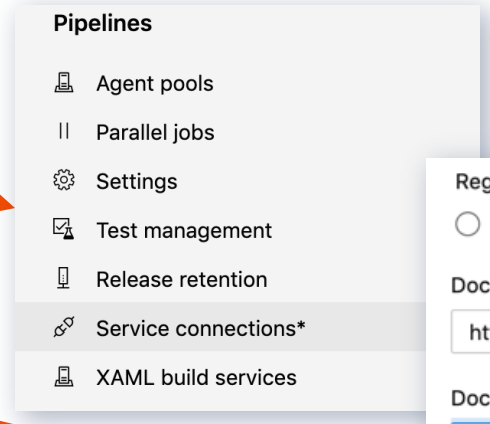
Setup: step 1

Setting below will allow you to connect to Sweagle docker registry from your pipelines to get Azure-CLI container

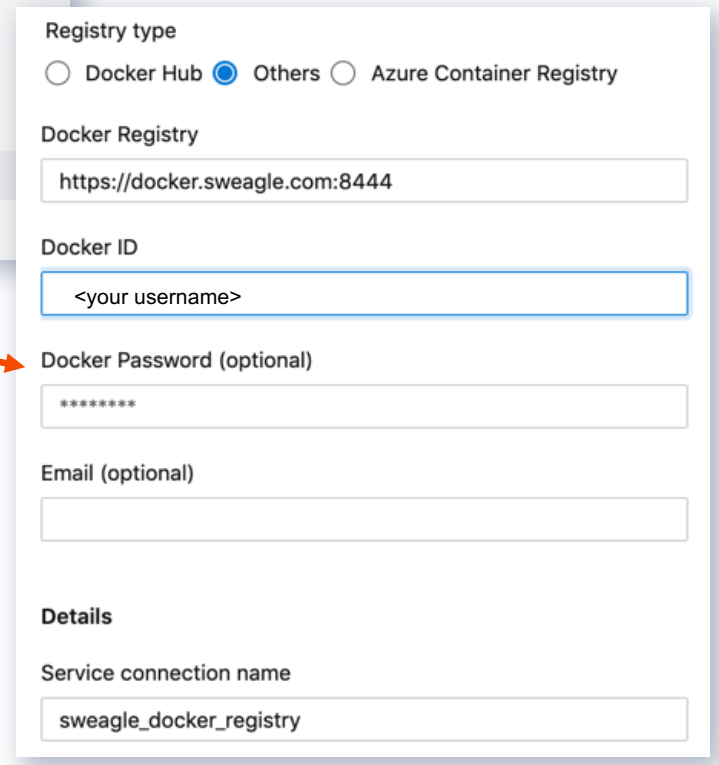
- Go to your Azure DevOps project settings
- In “Pipelines section”, select “Service connections”
- Create a new service connection of type “Docker Registry”
- You can use parameters below
  - Registry: https://docker.sweagle.com:8444
  - Docker ID: (your registry username provided by Sweagle)
  - Password: (your registry password provided by Sweagle)
  - Connection name: sweagle_docker_registry (will be used in pipelines)

For more details: https://docs.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml#sep-docreg





Docker registry settings



SWEAGLE
KEEPING AN EAGLE EYE ON YOUR CONFIG DATA

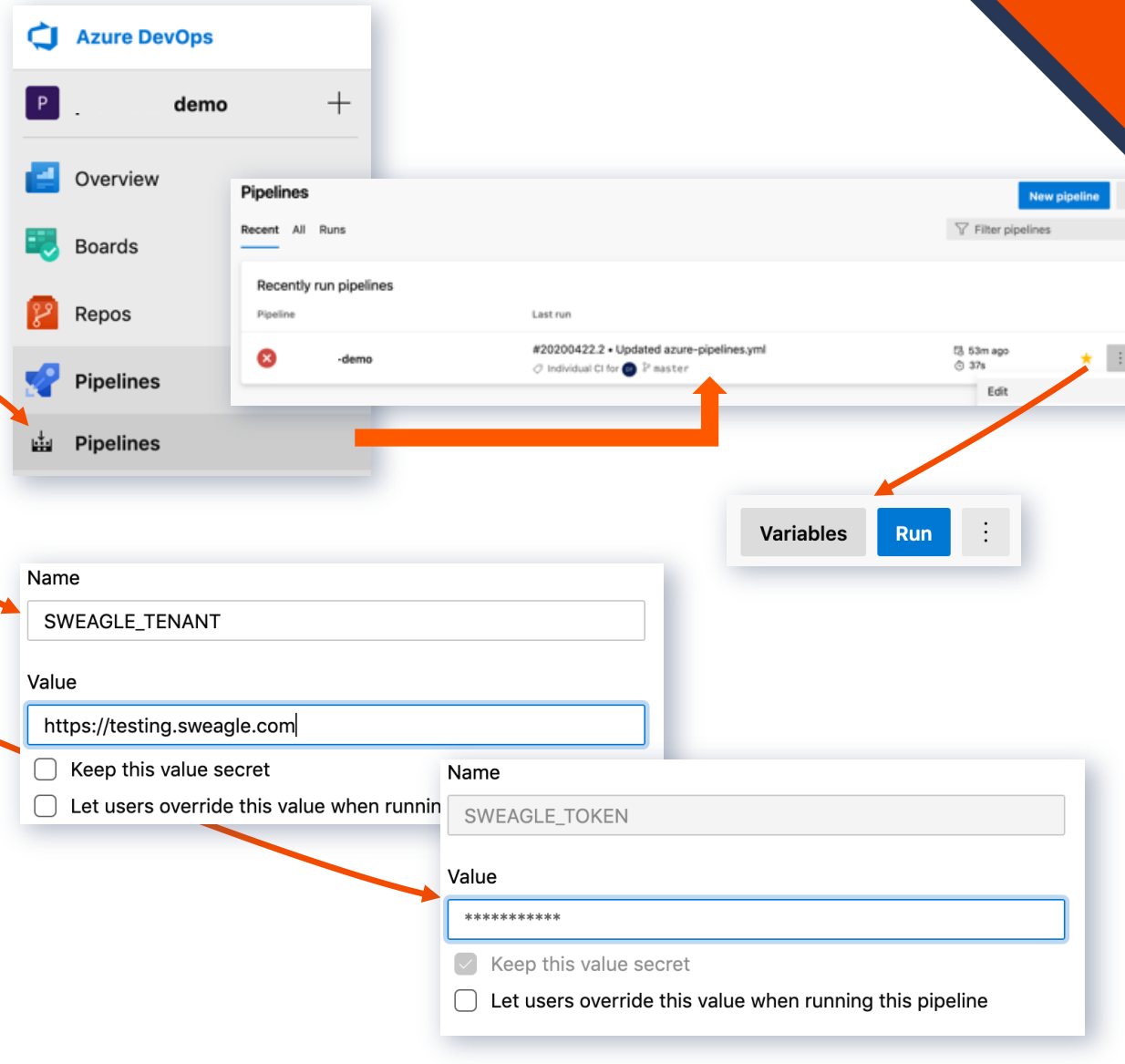
Setup: Define pipeline variables

Setup: step 2

Settings below will allow you to secure your tenant connection settings using pipeline secret variables

- In your Azure DevOps project, go to pipeline settings
- Edit your pipeline and select “Variables” (top right button)
- Add a variable called “SWEAGLE_TENANT”
  - Enter your tenant URL as value
- Add a variable called “SWEAGLE_TOKEN”
  - Paste your API Token as value
  - Check “Keep this value secret” box

For more details: https://docs.microsoft.com/en-us/azure/devops/pipelines/process/variables?view=azure-devops&tabs=yaml%2Cbatch#secret-variables



SWEAGLE
KEEPING AN EAGLE EYE ON YOUR CONFIG DATA

Setup: Enable CLI in pipeline

Setup: step 3

To use CLI in pipeline, you should both define container to use and configure it at runtime

- Define your container resource
  - Create a container resource
  - Note that resource endpoint is based on service connection defined in step 1
- Then, use this resource

- Configure connection to SWEAGLE tenant
  - Note that you use pipeline variables defined in step 2
  - Secret variables must be defined as “env” variables to be decrypted by Azure pipeline
  - Your first script step will use a ”sweagle options” command to configure tenant connection

- Test connection with a “sweagle info” command
- This resource can now be used in any future steps

```yaml
resources:
  containers:
  - container: sweagle-cli
    endpoint: sweagle_docker_registry
    image: 'sweagle-docker/sweagle-azure-cli:1.0.0'

pool:
  vmImage: 'ubuntu-latest'

container: 'sweagle-cli'
```

```yaml
variables:
  SWEAGLE_MAPPED_TENANT: $(SWEAGLE_TENANT)
```

```yaml
steps:
# you should always configure the CLI before use because temporary folders are created at each pipeline run and
- script: |
    sweagle options --newenv $SWEAGLE_MAPPED_TENANT --newusername azurePipeline --newtoken $SWEAGLE_MAPPED_TOKEN
    sweagle info
  displayName: 'Configure and test SWEAGLE CLI'
# secret variables must be defined as env variable directly in step where it is used
  env:
    SWEAGLE_MAPPED_TOKEN: $(SWEAGLE_TOKEN)
```

Pipeline code

Pipeline sample

You can use in your pipeline any Sweagle CLI command as a script steps. Most useful commands are:

- → "uploadData" to upload a configuration file to Sweagle in a specified node
- → "validate" to validate a configuration data set (CDS) with a specific validator
- → "validationStatus" to get the status of all assigned validators for this configuration data set
- → "storeSnapshot" to take the snapshot of this CDS
- → "export" to download this configuration or a subset of it

You can use "sweagle <command> --help" to get more details about options for any command

azure-pipelines.yml

Edit

Contents History Compare Blame

```
57 - script: 'sweagle uploadData --filePath ./result.$(FORMAT) --nodePath azure-devops,outputs,$(FILENAME) --
58   displayName: 'Upload data to SWEAGLE'
59
60 - script: |
61     response=$(sweagle validate --validator noEmptyValues --forIncoming --pretty $(CDS))
62     if [[ $response == *"Request failed with status code 404"* ]]; then
63       echo "### No pending DCS, trying with stored values instead ###"
64       response=$(sweagle validate --validator noEmptyValues --pretty $(CDS))
65     fi
66     echo $response
67     if [[ ${response,,} == *"error"* ]]; then exit 1; fi
68   displayName: 'Validate Changes'
69
70 #- script: 'sweagle validationStatus --forIncoming --withData --pretty --forIncoming --pretty $(CDS)'
71 #  displayName: 'Get validation status for CDS'
72
73 - script: |
74     sweagle storeSnapshots --configdatasetName $(CDS) --snapshotTag $(Build.BuildNumber) --level error
75     sweagle export $(CDS) --exporter returnDataforNode --format $(FORMAT) --argsList $(FILENAME) > ./outpu
76     echo "################################"
77     echo "#####   DISPLAY OUTPUT FILE   #####"
78     echo "################################"
79     echo "FILENAME = $(FILENAME)"
80     if [ "${FORMAT,,}" = "json" ]; then
81       cat ./output.$(FORMAT) | jq .
82     else
83       cat ./output.$(FORMAT)
84     fi
85   displayName: 'Snapshot and get config data'
```

Dimitris Finas
Tech. Director Southern Europe
dimitris@sweagle.com

SWEAGLE
KEEPING AN EAGLE EYE ON YOUR CONFIG DATA