SHELL := /bin/bash

VIRTUALENV\_DIR := $(CURDIR)/.venv

VIRTUALENV\_BIN := $(VIRTUALENV\_DIR)/bin

ACTIVATE := $(VIRTUALENV\_BIN)/activate

REQUIREMENTS:=$(CURDIR)/requirements.txt

PIP := $(VIRTUALENV\_BIN)/pip

PYTHON := $(VIRTUALENV\_BIN)/python3

ANSIBLE\_PLAYBOOK := $(VIRTUALENV\_BIN)/ansible-playbook

TOOLBOX\_HOME=$(CURDIR)/toolbox

USER := $(shell id -un)

USER\_ID := $(shell id -u)

K8S\_INSTALLER\_BUILD\_LOG\_PATH ?= $(CURDIR)/tools/.workspace/k8s\_installer\_build.log

export BUILD\_ID=$(shell TZ=UTC date '+%Y%M%d%H%M%S')

venv: $(ACTIVATE)

venv-clean:

@rm -rf $(CURDIR)/.venv

K8S\_%:

@ if [ "${K8S\_${\*}}" = "" ]; then \

echo "Environment variable K8S\_$\* is not set, please set one before run"; \

exit 1; \

fi

ENV\_%:

@ if [ "${ENV\_${\*}}" = "" ]; then \

echo "Environment variable ENV\_$\* is not set, please set one before run"; \

exit 1; \

fi

$(VIRTUALENV\_DIR):

@virtualenv -p python3.6 $(VIRTUALENV\_DIR)

$(ACTIVATE): $(VIRTUALENV\_DIR) $(REQUIREMENTS)

@$(PIP) install --upgrade-strategy only-if-needed -r $(REQUIREMENTS)

@touch $(ACTIVATE)

include toolbox/checks/checks.mk

k8s-installer-clean:

@ if [ -f $(CURDIR)/tools/.workspace/version ]; then \

export CONTAINER\_LIST="$$(docker ps | grep $$(cat $(CURDIR)/tools/.workspace/version) | awk '{print $$1}')"; \

if [ "$$CONTAINER\_LIST" != "" ]; then \

docker rm -f $$CONTAINER\_LIST ; \

fi \

fi

@(cd $(CURDIR)/tools && find .workspace/ -mindepth 1 \( ! -iname "nauta-\*.tar.gz" ! -iname "k8s\_installer\_build.log" \) 2>/dev/null | xargs rm -rf)

k8s-installer-build-wrapped:

@(cd $(CURDIR)/tools/initializers/deps && make check-platform-dependencies)

@(cd $(CURDIR)/tools/initializers/platform && make init)

@(cd $(CURDIR) && make tools-release)

@(cd $(CURDIR) && make k8s-installer-clean)

k8s-installer-build:

@echo "k8s installer build logs will be saved to $(K8S\_INSTALLER\_BUILD\_LOG\_PATH)"

@mkdir -p "$(CURDIR)/tools/.workspace"

@set -o pipefail && make k8s-installer-build-wrapped 2>&1 | tee $(K8S\_INSTALLER\_BUILD\_LOG\_PATH)

nctl-build:

@(cd $(CURDIR)/applications/cli && make full\_clean && make push)

tools-%:

@(cd $(CURDIR)/tools && make -j 2 $\*)

single-tools-%:

@(cd $(CURDIR)/tools && make $\*)

unit-tests:

@(cd $(CURDIR)/applications/cli && make test)

build-conditional-deep-clean:

@(cd $(CURDIR)/applications/cli && make build-conditional-deep-clean)

cli-style:

@(cd $(CURDIR)/applications/cli && make style)

unit-tests-with-code-cov:

@(cd $(CURDIR)/applications/cli && make test-with-code-cov)

gui-unit-tests:

@(cd $(CURDIR)/applications/nauta-gui && make test)

include toolbox/providers/providers.mk

include toolbox/support/gateway-users/gateway-users.mk

generate-docs:

@. $(ACTIVATE); pip install -r docs/sphinx-requirements.txt;

@. $(ACTIVATE); cd docs/ && ./generate\_docs.sh;