Joe C

General CI/CD

- Automate the promotion from `sbx` to `dev` and higher (including GetIT requests)
- Official pipelines (Spring, .NET) published by us into repo
- Automated README-to-Confluence markdown transfer
- Try for zero CI/CD files present in client library. Can we run CI/CD without `.gitlab-ci.yml`?

GitLab Runners

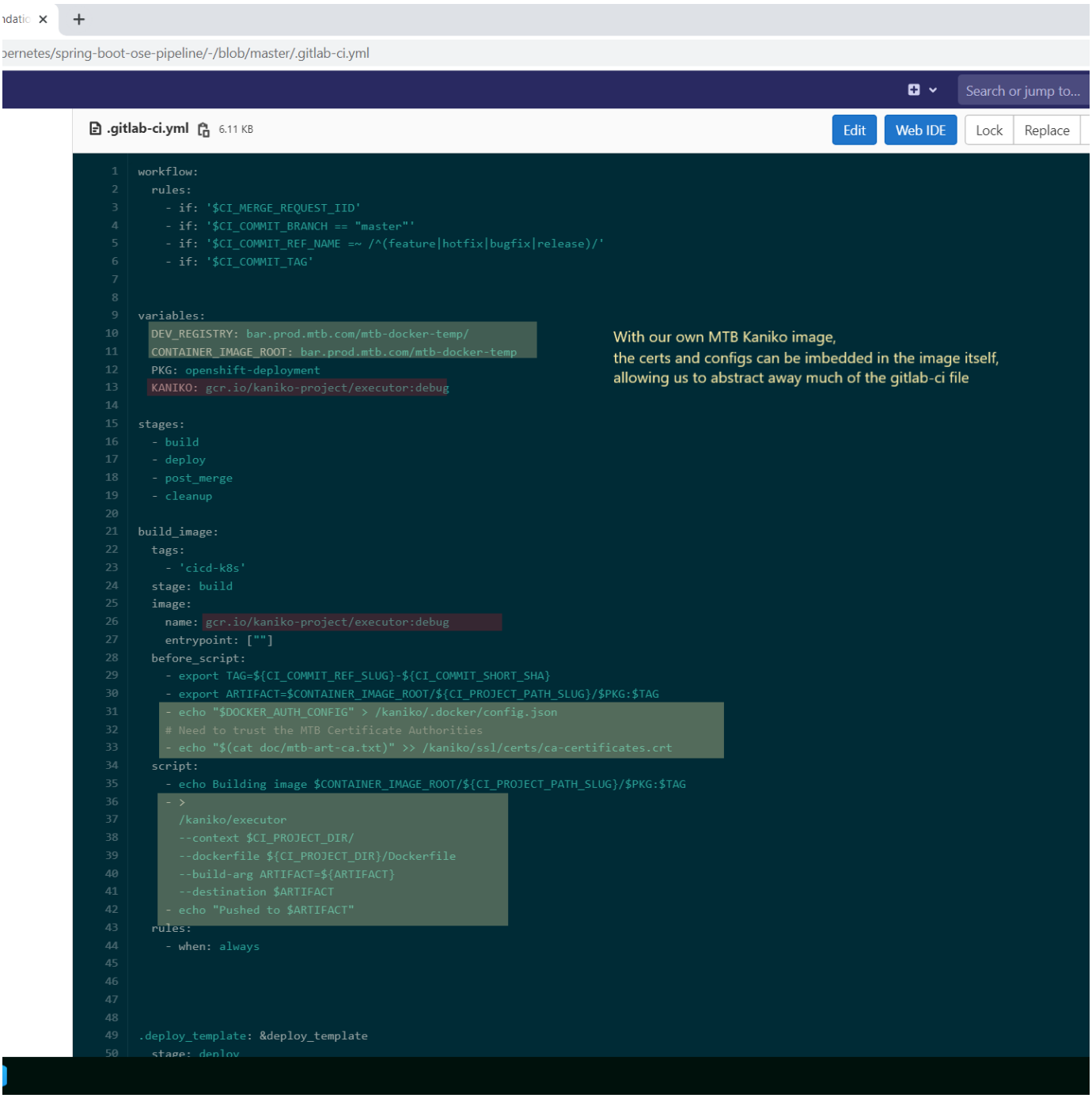
- Get security to waive aqua scan for runner containers
- Replace alpine in helper images
- Consider a boot-specific runner with JDK, certs, etc. inside the container

Artifactory/Machine Shop

- Our own team-specific Dockerhub repositories
- Our own team-specific Artifactory repositories (for Docker, Maven, and Helm at least)
- Map out our own Docker images and associated image tags (stop using latest)
- Create our own `mtb-kaniko` image
- Pack all necessary OSE certs, Helm files, Dockerfile, etc. whatever we need into the associated images
- Adopt OpenJDK images

Java Libraries

- Establish an MTB client starter parent
- Super easy annotation-based Kafka Library


ıdatic
bernetes/spring-boot-ose-pipeline/-/blob/master/.gitlab-ci.yml
Search or jump to...
.gitlab-ci.yml 6.11 KB
Edit
Web IDE
Lock
Replace
workflow:
rules:
- if: '$CI_MERGE_REQUEST_IID'
- if: '$CI_COMMIT_BRANCH == "master"'
- if: '$CI_COMMIT_REF_NAME =~ /^(feature|hotfix|bugfix|release)/'
- if: '$CI_COMMIT_TAG'
variables:
DEV_REGISTRY: bar.prod.mtb.com/mtb-docker-temp/
CONTAINER_IMAGE_ROOT: bar.prod.mtb.com/mtb-docker-temp
PKG: openshift-deployment
KANIKO: gcr.io/kaniko-project/executor:debug
stages:
- build
- deploy
- post_merge
- cleanup
build_image:
tags:
- 'cicd-k8s'
stage: build
image:
name: gcr.io/kaniko-project/executor:debug
entrypoint: [""]
before_script:
- export TAG=${CI_COMMIT_REF_SLUG}-${CI_COMMIT_SHORT_SHA}
- export ARTIFACT=$CONTAINER_IMAGE_ROOT/${CI_PROJECT_PATH_SLUG}/$PKG:$TAG
- echo "$DOCKER_AUTH_CONFIG" > /kaniko/.docker/config.json
Need to trust the MTB Certificate Authorities
- echo "$(cat doc/mtb-art-ca.txt)" >> /kaniko/ssl/certs/ca-certificates.crt
script:
- echo Building image $CONTAINER_IMAGE_ROOT/${CI_PROJECT_PATH_SLUG}/$PKG:$TAG
- >
/kaniko/executor
--context $CI_PROJECT_DIR/
--dockerfile ${CI_PROJECT_DIR}/Dockerfile
--build-arg ARTIFACT=${ARTIFACT}
--destination $ARTIFACT
- echo "Pushed to $ARTIFACT"
rules:
- when: always
.deploy_template: &deploy_template
stage: deploy
With our own MTB Kaniko image,
the certs and configs can be imbedded in the image itself,
allowing us to abstract away much of the gitlab-ci file

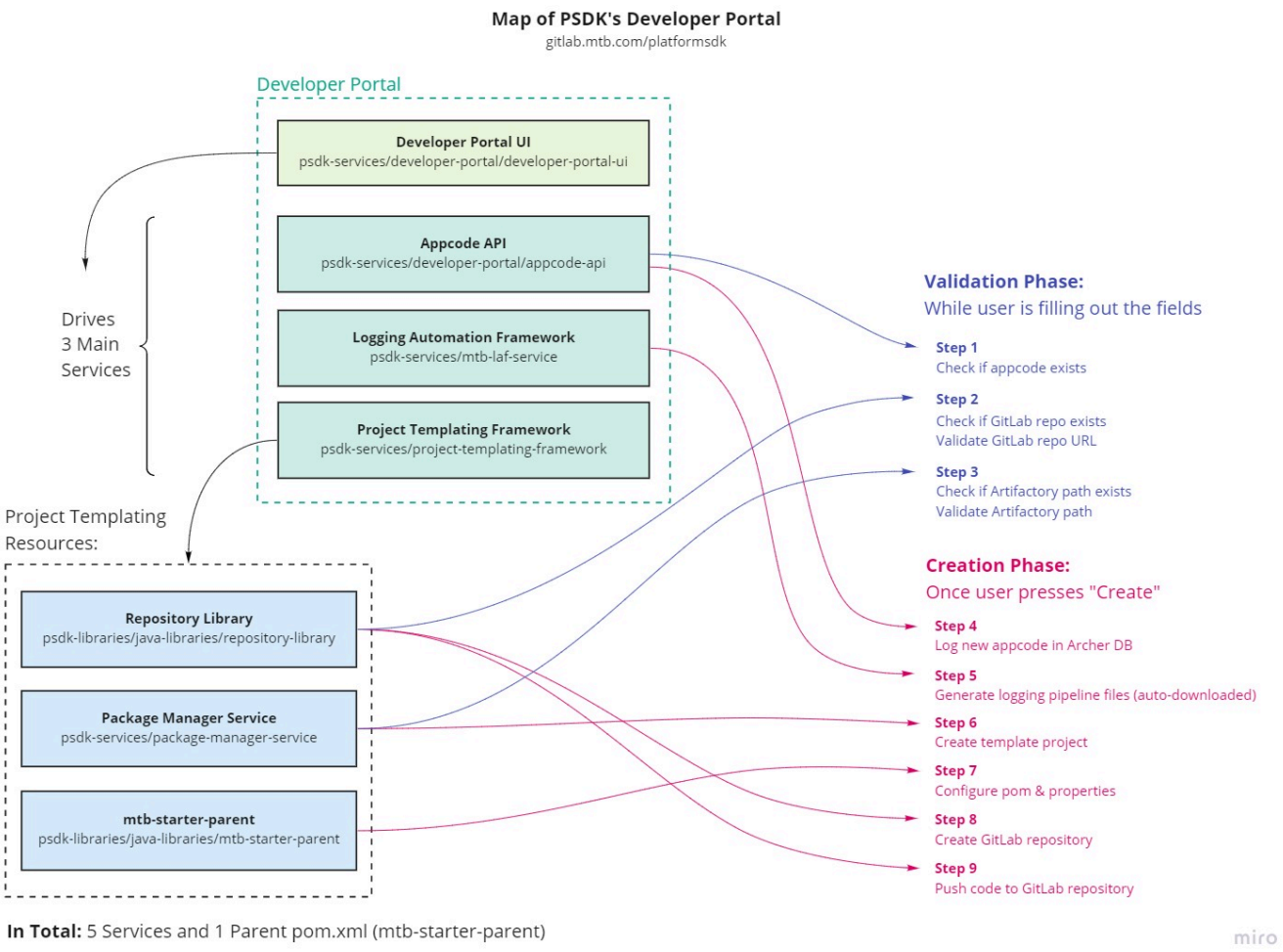
etes/spring-boot-ose-pipeline/-/blob/master/.gitlab-ci.yml

Search or jump to...

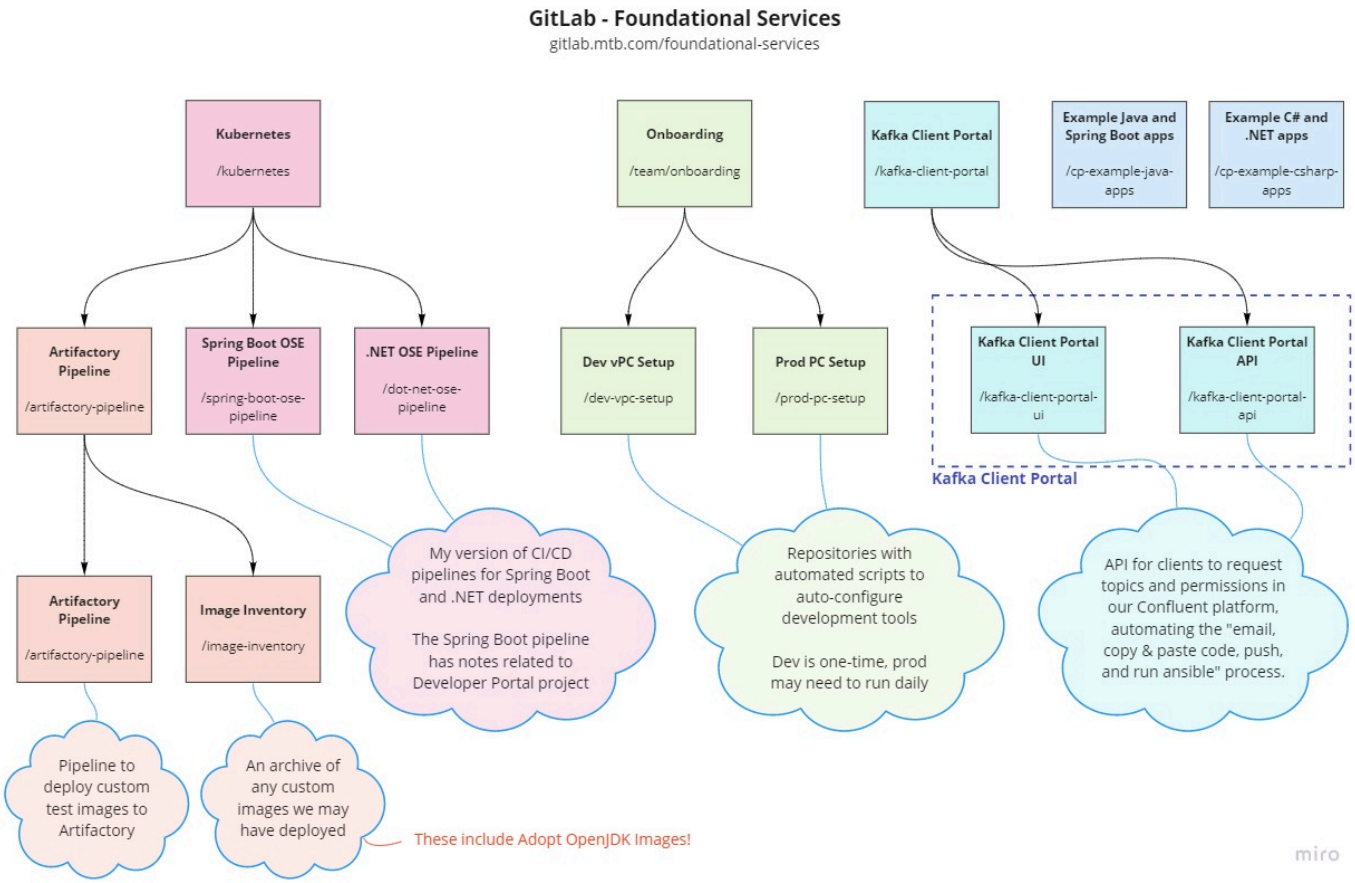
With a few modifications to our mtb-ose-utilities image, we can embed many of the CI/CD operation into the image itself, including env vars OSE_USER and OSE_USER_PW

```
.deploy_template: &deploy_template
  stage: deploy
  tags:
    - 'cicd-k8s'
  image: bar.prod.mtb.com/mtb-docker/mtb-ubi8-ose-utilities:latest
  script:
    - export TAG=${CI_COMMIT_REF_SLUG}-${CI_COMMIT_SHORT_SHA}
    - export RELEASE=${CI_PROJECT_PATH_SLUG}-${CI_COMMIT_REF_SLUG}
    - echo deploying image $CONTAINER_IMAGE_ROOT/${CI_PROJECT_PATH_SLUG}/$PKG:$TAG
    - |
      if [[ ${REGION} == 'sbx' ]]; then
        export HOST=${CI_PROJECT_PATH_SLUG}-${CI_COMMIT_REF_SLUG}.apps.ose${REGION}.mtb.com;
      else
        export HOST=${CI_PROJECT_PATH_SLUG}.apps.ose${REGION}.mtb.com;
      fi
    - export SERVER_PORT_GREP=$(grep -i 'server.port\s*=\s*....' src/main/resources/application.properties)
    - |
      if [ ${#SERVER_PORT_GREP} -gt 15 ];
        then
          export SERVER_PORT="${SERVER_PORT_GREP:(-4)}"
          echo "server port is $SERVER_PORT"
        else
          export SERVER_PORT="8080"
          echo server port auto-configured to 8080
      fi
    - echo server port has been set to $SERVER_PORT
    - oc login https://api.ose${REGION}.mtb.com:6443 -u ${OSE_USER} -p ${OSE_USER_PW}
    - oc project ${CI_PROJECT_PATH_SLUG}-${REGION}
    - >
      helm upgrade $RELEASE helm/charts/ -f ./helm/charts/values.yaml
      --install
      --set host=$HOST
      --set tag=$TAG
      --set image=$CONTAINER_IMAGE_ROOT/${CI_PROJECT_PATH_SLUG}/$PKG
      --set serverPort=$SERVER_PORT
      --namespace ${CI_PROJECT_PATH_SLUG}-${REGION}
    - echo deployment URL is https://${HOST}
    - echo server port is $SERVER_PORT

deploy_osesbx:
  <<: *deploy_template
  before_script:
    - oc login https://api.ose${REGION}.mtb.com:6443 -u ${OSE_USER} -p ${OSE_USER_PW}
    - oc project ${CI_PROJECT_PATH_SLUG}-${REGION} || oc new-project ${CI_PROJECT_PATH_SLUG}-${REGION}
  variables:
    REGION: sbx
  environment:
    sbx
  rules:
```

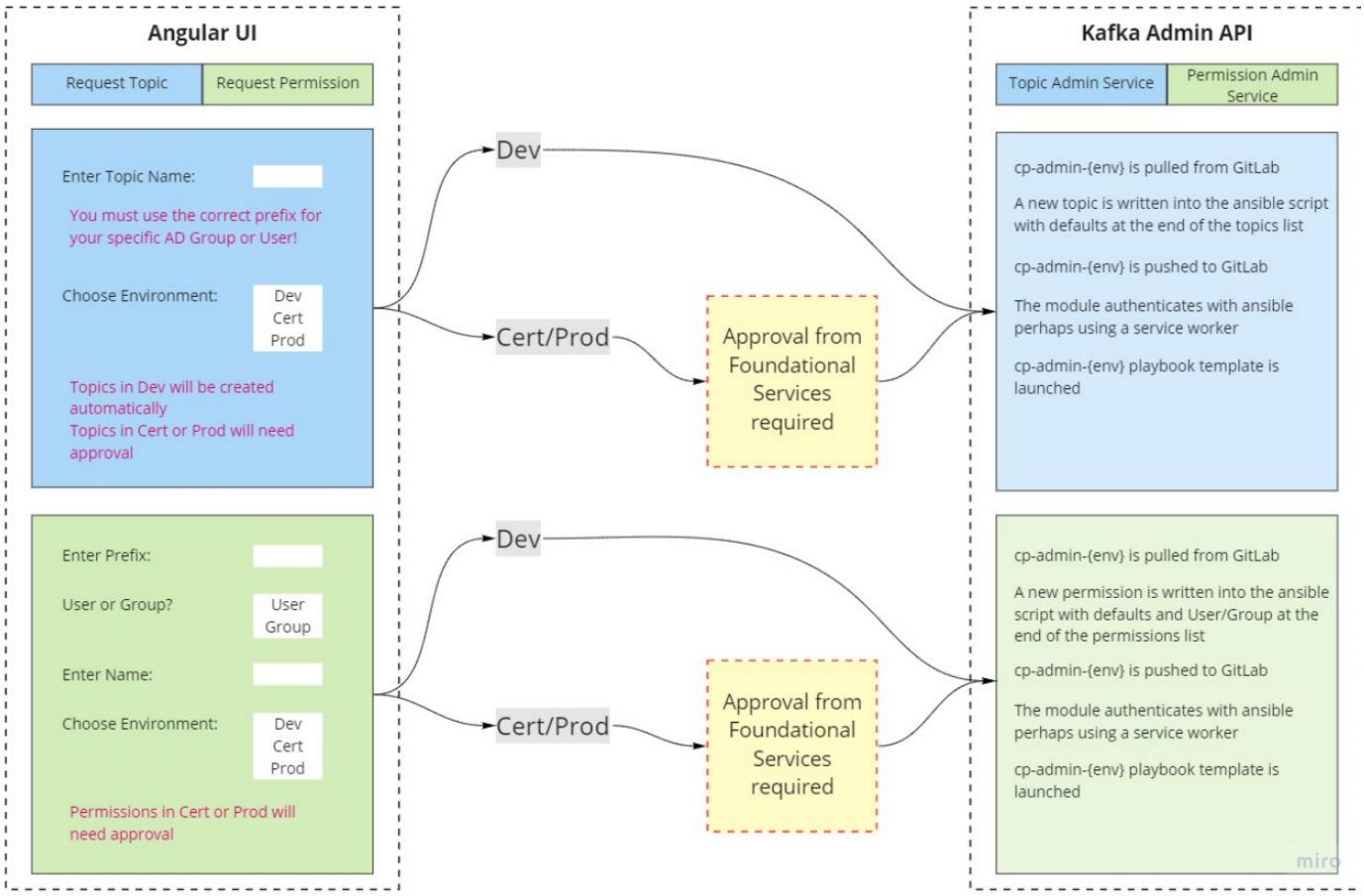


PDF Version (a little more clear):



PDF Version (a little more clear):

BTW there is also a Dockerhub pipeline now too!



PDF Version (a little more clear):

kafka-admin-api-diagram.pdf

Foundational Services

Platform SDK

Foundational Platform

Platform Administration

Central Software Services

Enterprise Development Lab

Enterprise Core

Developer Experience

Master Library

Grand Central Station