Building Images on Kubernetes

Toward a first class approach

Ben Parees(Red Hat)
Steve Speicher (Red Hat)
Matt Moore (Google)

Agenda

- Current state of the art+limitations
  - Roll your own
  - OpenShift Build api
- Proposal/Discussion
  - A CRD for Builds

Roll your own

- How
  - Run a privileged pod
  - Mount a docker socket
  - Invoke docker build
  - Push resulting image
- Good
  - You can do it today on any cluster
    - At least ones that have a docker daemon (e.g. not CRI-O)
  - Good layer sharing/reuse during image pull/build/push
- Bad
  - Security nightmare
  - Developers must learn/use Dockerfiles

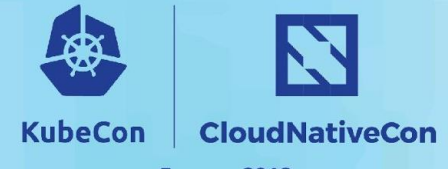
OpenShift Build API

- How
  - Build api object defines the build to be run
  - Controller creates a privileged pod
  - Pod contains safe logic for interacting with s2i or docker build
- Good
  - First class api for interacting with builds on the cluster
  - Users don’t get or need privileged pods or docker daemon access
  - S2I builds don’t require a Dockerfile
  - Good layer sharing/reuse during image pull/build/push
- Bad
  - Privileged pods are still a potential exposure for the cluster
  - API+Controller doesn’t exist on Kubernetes
  - Still requires a docker daemon on the hosts (makes moving to CRI-O less useful)

OpenShift Build API

- Inputs
  - Git source
  - Content from images
  - Secrets
  - Environment variables
  - Secrets for accessing inputs
- Scheduling
  - Serial/Parallel
  - Resources (memory, cpu limit)
- Triggers
  - Webhooks
  - Image change
- Output image target
  - Secret for pushing

OpenShift API Example




```
kind: "BuildConfig"
apiVersion: "v1"
metadata:
  name: "ruby-sample-build"
spec:
  runPolicy: "Serial"
  triggers:
    - type: "GitHub"
      github:
        secret:
          name: somesecret
    - type: "ImageChange"
  source:
    git:
      uri: "https://github.com/myapp/ruby"
      ref: mybranch
    sourceSecret:
      name: mygitsecret
```

```
strategy:
    sourceStrategy:
      from:
        kind: "ImageStreamTag"
        name: "ruby:2.4"
  output:
    to:
      kind: "ImageStreamTag"
      name: "origin-ruby-sample:latest"
    pushSecret:
      name: mypushsecret
  postCommit:
      script: "bundle exec rake test"
```

Container Builder Interface

- Defines a CRD for docker-type builds
- Abstracts several dockerfile build tools
  - Docker, BuildKit, Buildah, Kaniko
- Runs them on a Kubernetes cluster
- Same security challenges as any individual dockerfile build tool
- Build context from git or configmap

argoproj API


```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: sidecar-dind-
spec:
  entrypoint: dind-sidecar-example
  templates:
  - name: dind-sidecar-example
    container:
      image: docker:17.10
      command: [sh, -c]
      args: |
        # Wait for Docker to come up
        until docker ps;
          do sleep 3;
        done;
        # Dockerfile build
        docker build -t foo .
        docker push foo
      env:
      - name: DOCKER_HOST
        value: 127.0.0.1
```

```yaml
sidecars:
- name: dind
  image: docker:17.10-dind
  securityContext:
    privileged: true
  mirrorVolumeMounts: true
```

Based on: https://github.com/argoproj/argo/blob/master/examples/sidecar-dind.yaml

Google Container Builder API


```
steps:
# Can have many steps, run in sequence on the same Node.
# Steps are just “builder” containers.
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'gcr.io/my-project/my-image', '.']

images: ['gcr.io/my-project/my-image']
```

Today: VM sandbox w/ daemon socket access.

Want to move more towards kaniko / FTL / ...

Want to make more K8s-native.

Squinting at the landscape

- Source
  - Methods of conveying *what* to build
- Steps
  - *How* to build it
  - Often expressed (or implemented) through “builder” containers (see earlier talk!)
- Volumes
  - To share data across steps (e.g. build cache)
- Other important considerations:
  - Authentication
  - Outputs

Proposal

- Let’s define a Build api+controller for Kubernetes
  - Input content
  - Image build mechanism(pluggable?)
  - Push resulting image

KubeCon

CloudNativeCon

Europe 2018

Discuss!

References

- OpenShift Builds - https://docs.openshift.org/latest/dev_guide/builds/index.html
- CBI - https://github.com/containerbuilding/cbi
- Argo - https://github.com/argoproj/argo