Developing a CD pipeline with Knative

—

Andrea Frittoli
Developer Advocate
andrea.frittoli@uk.ibm.com
@blackchip76

—

DevOps Meetup Singapore

IBM **Cloud**

A Bit of History

Knative

- Beginning of 2018...
- Knative:
    - Build
    - Eventing
    - Serving

- Contributors:
    - Google
    - Pivotal
    - IBM
    - RedHat
    - Cloudbees
    - ...and others

Latest news!

- Focus on CI/CD
- Deploy "anywhere"
- Compatible with Knative Build

- *tektoncd/pipeline*
  - New logo
  - @CD Foundation
  - Roadmap WIP
  - Alpha APIs

Community

- *Valid for Knative. Tekton TBD.*
- Steering Commitee (SC)
- Technical Oversight Commitee (TOC)
- Various Contribution profiles
- Design, issues: on GitHub
- Communication:
    - Weekly video meetings, recorded, Build WG
    - Asynch: Knative Users / Developers ML
    - Sync: slack.knative.dev

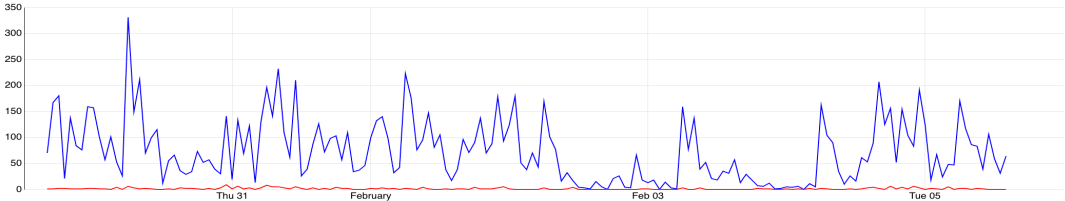# Tekton Pipelines

Cloud Native Pipelines

OpenStack Health is a dashboard for visualizing test results of OpenStack CI jobs.
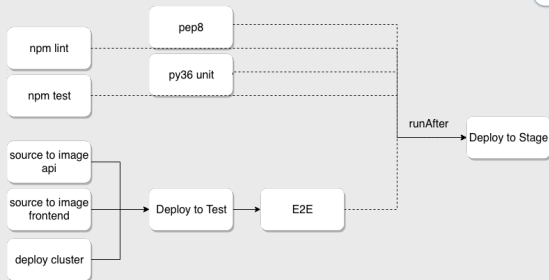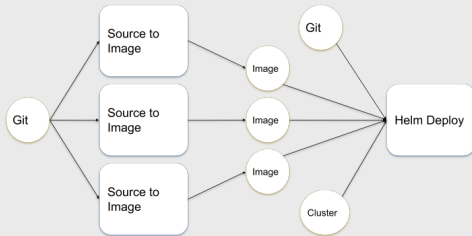
Total Jobs

Job Failure Rate

Inputs, Outputs & DAG

- Steps are sequential
- Tasks are a Directed Acyclic Graph
- Order defined by:
  - *from*: input from another task's output
  - *runAfter*: enforced task ordering

Under the Hood

Custom Resources

CRDs: Task(Run), Pipeline(Run), PipelineResource

Services in the *tekton-pipelines* namespace:

- – Webhook Service: resource validation
- – Controller Service:
    - – Handles inputs and outputs
    - – Calculates the DAG
    - – Provisions pods and containers

Custom Resource Provisioning:

- – Via YAML
- – Via Go API
- – Labels!

Pods, Entrypoints & Volumes

Steps (of a Task):

- Containers in one POD (single node)
- Any container image
- Entrypoint re-written
- Serial execution
- Resource allocation?

TaskRun:

- Provisions a POD
- Deployes entrypoint tool
- Input/output containers
- User containers (steps)

Volumes:

- EmptyDir for workspace/home
- Tools (entrypoint)
- Secrets
- Any user ConfigMap / Volume
- (Optionally) Pipeline Share

PipelineRun:

- Several PODs, different nodes
- Shared storage: PVC or GCS

Source to Image to Deploy

IBM Cloud

- Private Registry:
  - Tekon Images (push/pull)
  - User Images (push/pull)
- Service Accounts:
  - *tekton-pipelines-controller*
  - Pipeline/Task service account

- Knative @ IBM Cloud
  - Experimental Add-on
  - *ibmcloud ks cluster-addon-enable knative*

IBM **Cloud**

CD Pipeline as code

- Pipeline and Tasks in git (YAML)
- Parameters for env/run specific
- Security?

```
apiVersion: tekton.dev/v1alpha1
kind: PipelineResource
metadata:
  name: health-helm-git-knative
  labels:
    tag: agreatrelease
spec:
  type: git
  params:
    - name: revision
      value: knative
    - name: url
      value: https://github.com/afrittoli/health-helm
```

```
metadata:
  name: mycluster
spec:
  type: cluster
  params:
    - name: name
      value: mycluster
    - name: url
      value: https://mycluster.containers.cloud.ibm.com
    - name: username
      value: admin
  secrets:
    - fieldName: token
      secretKey: tokenKey
      secretName: cluster-secrets
    - fieldName: cadata
      secretKey: cadataKey
      secretName: cluster-secrets
```

```
metadata:
  name: health-api-image
spec:
  type: image
  params:
    - name: url
      value: registry.ng.bluemix.net/andreaf/health-api
```

Using Kaniko

- Features:
    - Build from Context and Dockerfile
    - Unpriviledged
    - Reproducible
    - Remote caching of layers
    - Base images caching (warmer)

- Dockefile?
    - Most common changes last
    - Careful with COPY/ADD
    - Remove what you don't need

Using Kaniko

Source to Image (spec only):

```
inputs:
  resources:
    - name: workspace
      type: git
  params:
    - name: pathToDockerFile
      default: Dockerfile
    - name: pathToContext
      default: .
    - name: useImageCache
      default: "true"
    - name: imageTag
      default: "default"
outputs:
  resources:
    - name: builtImage
      type: image
volumes:
  - name: kaniko-base-image-cache
    persistentVolumeClaim:
      claimName: kaniko-base-image-cache
steps:
  - name: build-and-push
    image: gcr.io/kaniko-project/executor
    command:
      - /kaniko/executor
```

```
    args:
      - --cache=${inputs.params.useImageCache}
      - --cache-dir=/cache
      - --dockerfile=${inputs.params.pathToDockerFile}
      - --reproducible
      - --destination=${outputs.resources.builtImage.url}:
  ${inputs.params.imageTag}
      - --context=/workspace/workspace/${inputs.params.
  pathToContext}
    volumeMounts:
      - name: kaniko-base-image-cache
        mountPath: /cache
```

Cache Warmer (spec only):

```
volumes:
  - name: kaniko-base-image-cache
    persistentVolumeClaim:
      claimName: kaniko-base-image-cache
steps:
  - name: prepare-cache
    image: gcr.io/kaniko-project/warmer
    args:
      - --cache-dir=/cache
      - --image=python:3.6-slim-stretch
      - --image=postgres:alpine
      - --image=nginx:latest
    volumeMounts:
      - name: kaniko-base-image-cache
        mountPath: /cache
```

Tekton and Knative

Pipelines and Knative Build

CI for OpenStack Health

CI with Tekton Pipelines

Asynchronous Pipelines

Triggering and Knative Eventing

Tekton and Development

Conclusions

Shall I use Tekton Pipelines?

Roadmap

References

- https://tekton.dev/
- https://github.com/tektoncd/pipeline
- https://cd.foundation/
- https://github.com/knative/docs/tree/master/community
- https://github.com/tektoncd/pipeline
- https://github.com/tektoncd/pipeline/blob/master/api_compatibility_policy.md
- https://github.com/tektoncd/pipeline/blob/master/roadmap-2019.md
- https://github.com/GoogleContainerTools/kaniko
- https://github.com/afrittoli/health-helm/tree/knative
- https://github.com/afrittoli/openstack-health/tree/knative-eventing
- https://andreafrittoli.me
- https://cloud.ibm.com

Q&A