Cloud Native CI/CD with Knative and Tekton Pipelines

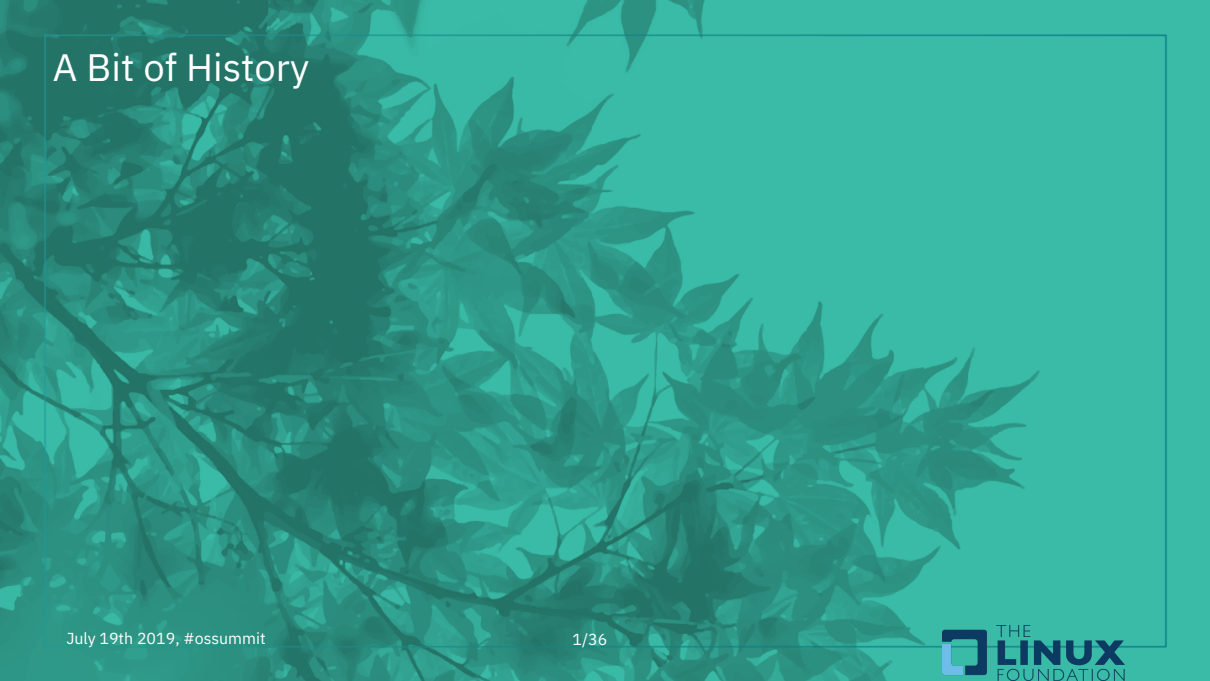
Andrea Frittoli
Open Source Developer Advocate
andrea.frittoli@uk.ibm.com
@blackchip76

Open Source Summit Japan 2019

THE LINUX FOUNDATION

July 19th 2019

A Bit of History



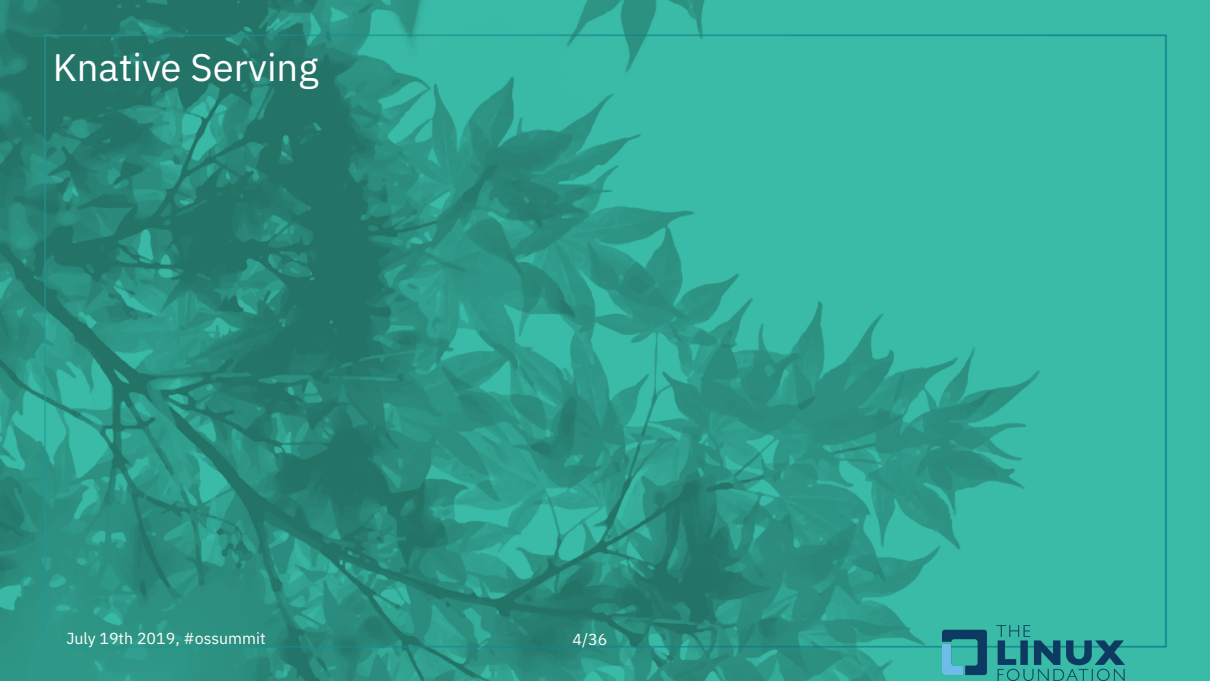
Knative

- Beginning of 2018...
- Knative:
  - Build
  - Eventing
  - Serving
- OpenSource
- Contributors:
  - Google
  - Pivotal
  - IBM
  - RedHat
  - Cloudbees
  - ...and others

Community

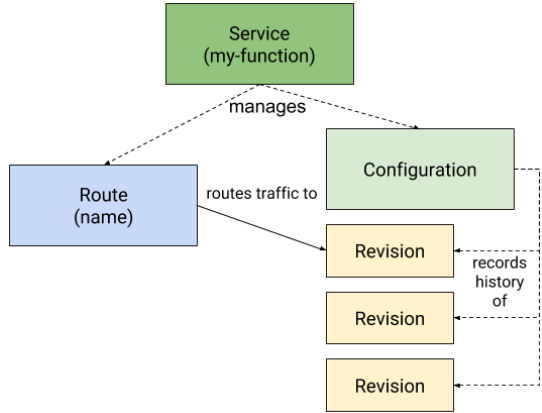
- Steering Commitee (SC)
- Technical Oversight Commitee (TOC)
- Dedicated Working Groups (WG)
- Various Contribution profiles
- Design, issues: on GitHub
- Communication:
  - WG periodic meetings, recorded
  - Asynch: Knative Users / Developers ML
  - Sync: slack.knative.dev

Knative Serving

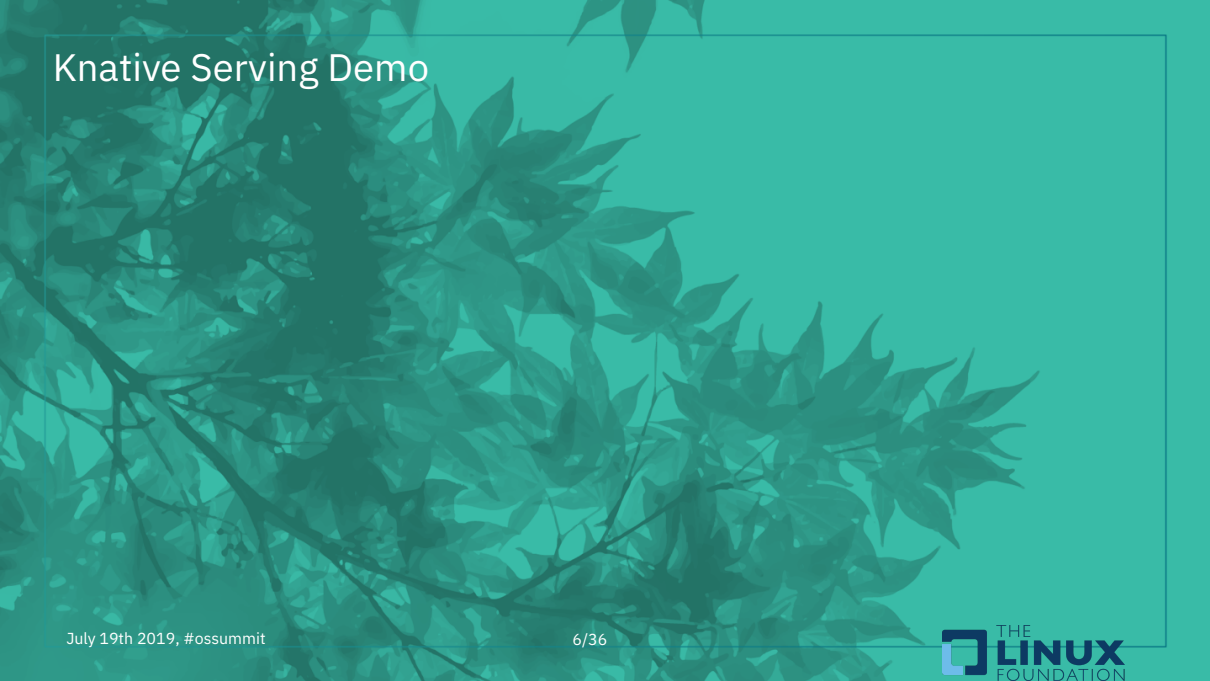


Knative Serving

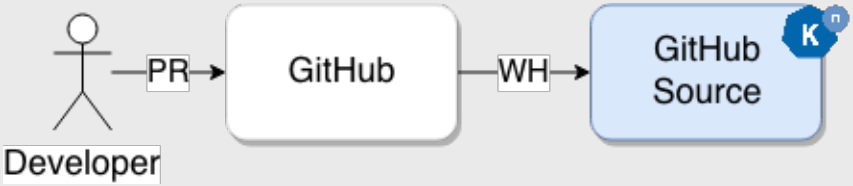
- Scale to Zero
- Scale up based on metrics
- Multiple revisions



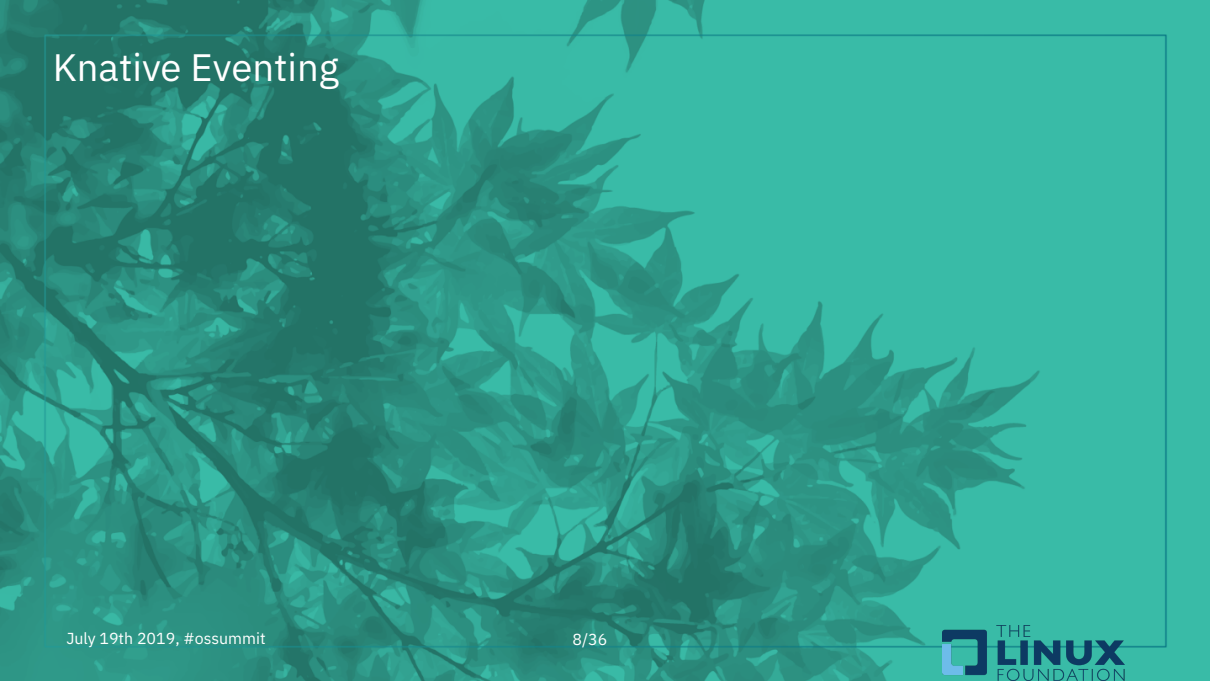
Knative Serving Demo



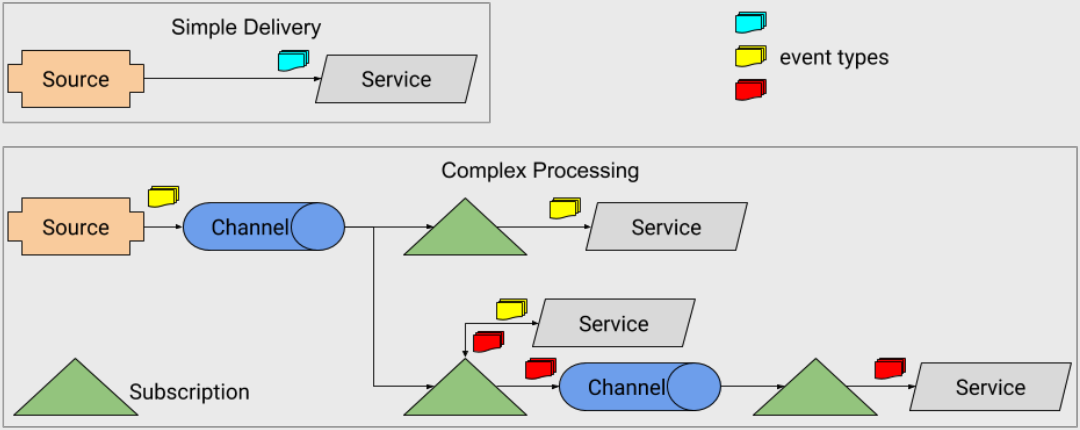
Knative Serving Demo



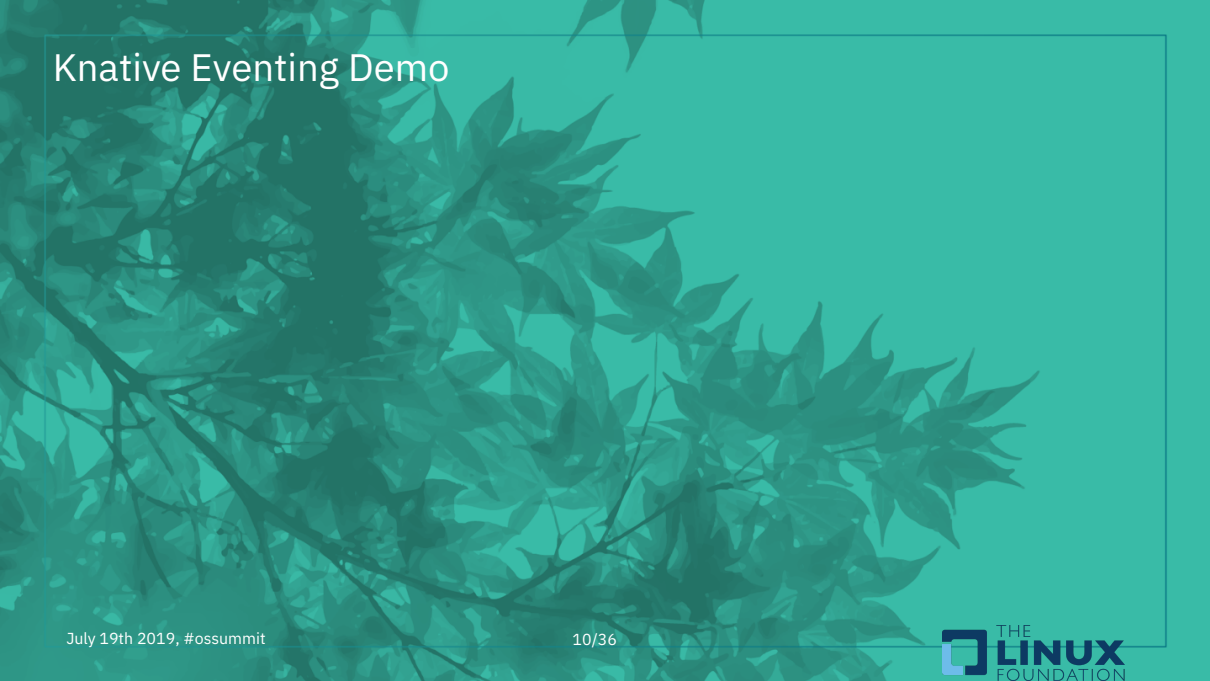
Knative Eventing



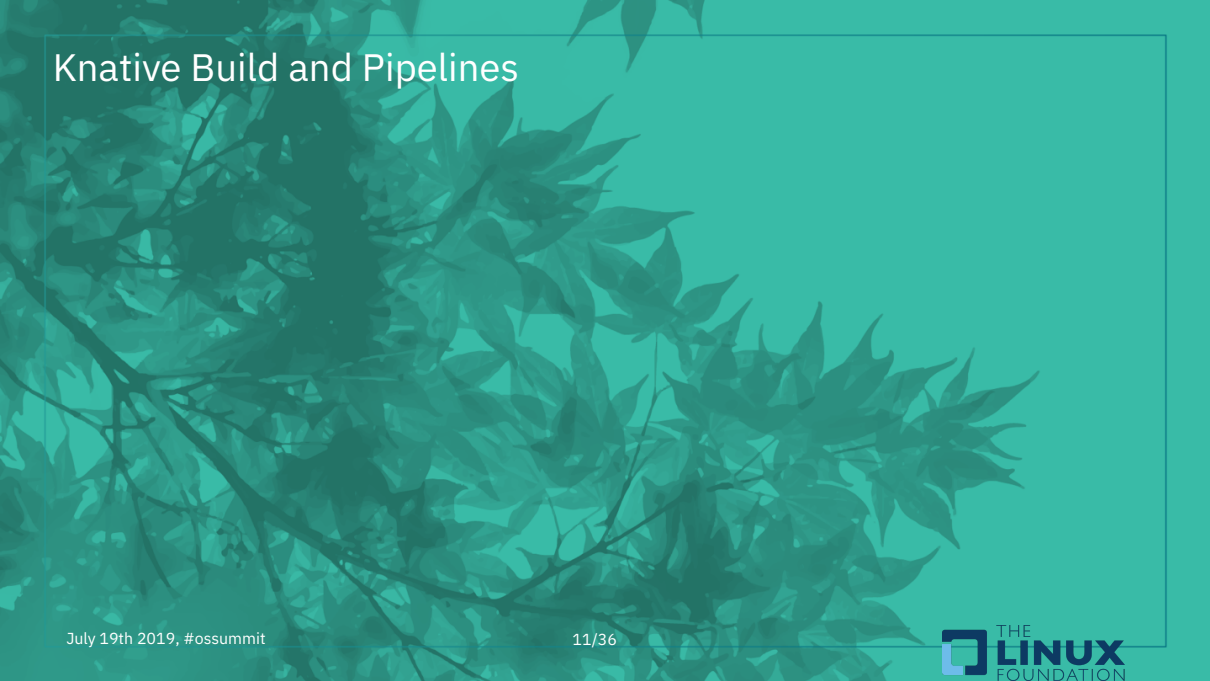
Knative Eventing



Knative Eventing Demo



Knative Build and Pipelines

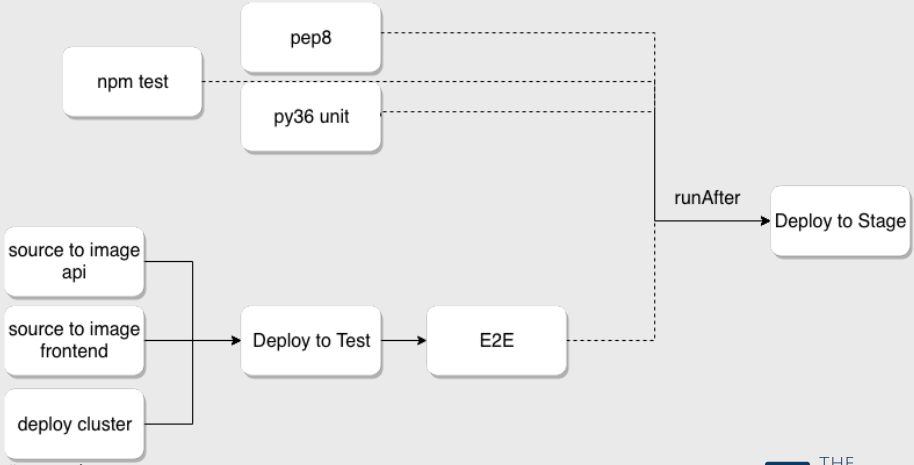


Knative Build

- Source to image
- Build Templates
- How to make a new revision

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
spec:
  runLatest:
    configuration:
      build:
        apiVersion: build.knative.dev/v1alpha1
        kind: Build
        spec:
          serviceAccountName: build-bot
          source:
            git:
              url: https://github.com/mc/simple-app.git
              revision: master
          template:
            name: kaniko
            arguments:
              - name: IMAGE
                value: docker.io/{USERNAME}/myapp:latest
          timeout: 10m
      revisionTemplate:
        spec:
          container:
            image: docker.io/{USERNAME}/my:latest
            imagePullPolicy: Always
```

Going further



~Sept 2018: Knative Pipelines

Latest news!

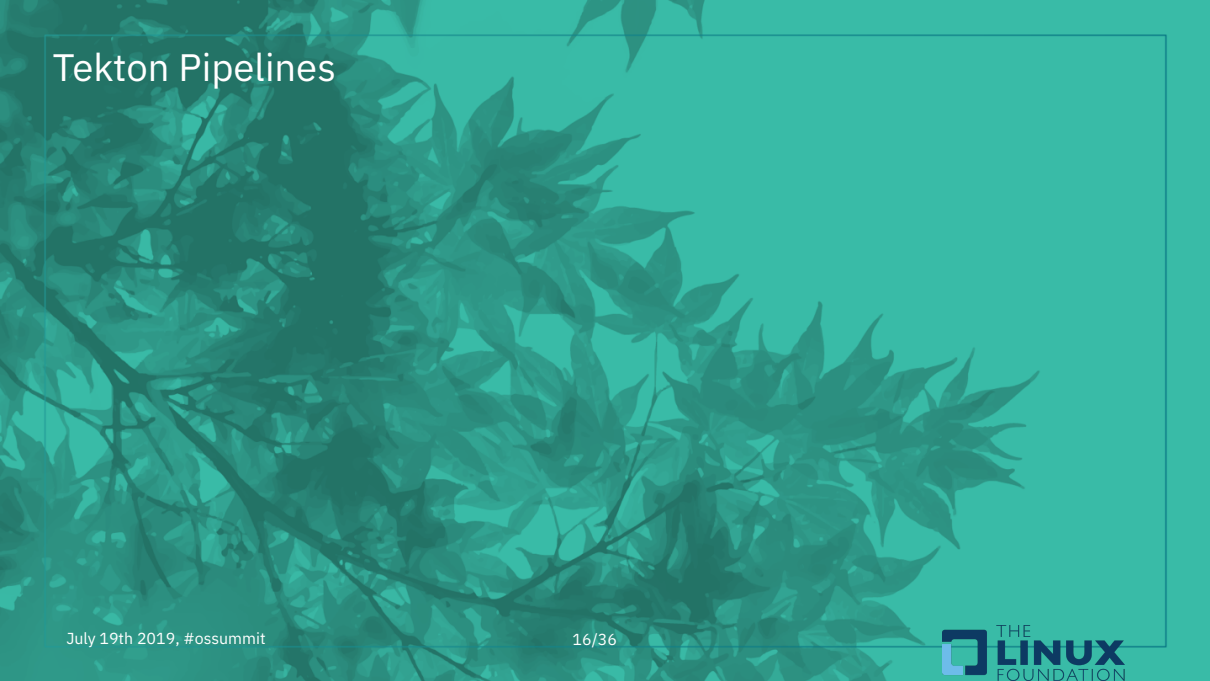
- Tekton pipelines (March 2019)
- Focus on CI/CD
- @CD Foundation (March 2019)
- Deploy “anywhere”
- Replaces with Knative Build (June 2019)

TEKTON

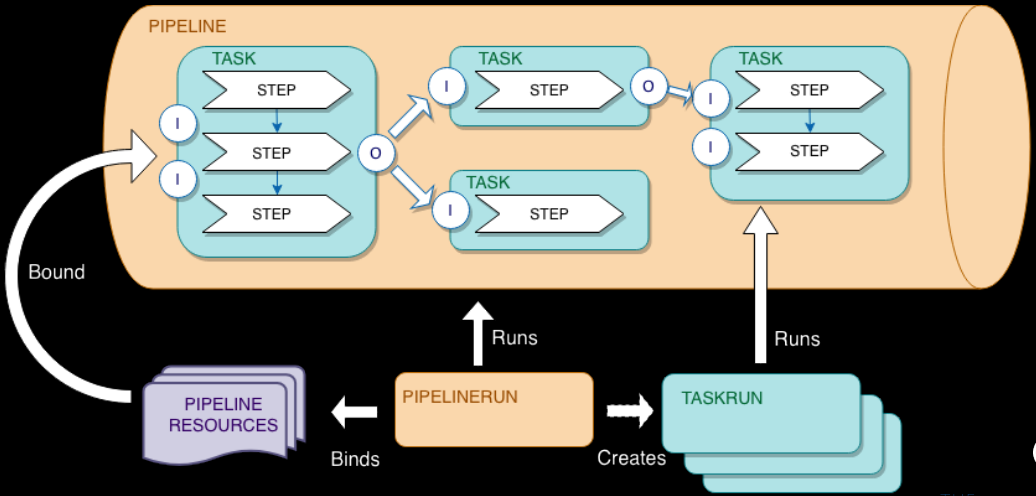
CD.FOUNDATION

THE LINUX FOUNDATION

Tekton Pipelines

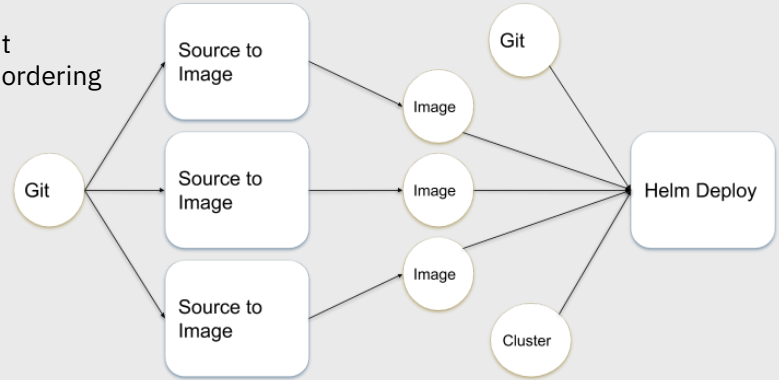


Cloud Native Pipelines



Task Inputs, Outputs & DAG

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



Source to Image with 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
```

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

Pipeline as code

- Pipeline and Tasks:
  - Static definition, stored in git
- PipelineRuns and TaskRuns:
  - Specific to one execution
  - Generated programmatically
- Parameters and Resources:
  - Env / run specific

```
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
```

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?

Volumes:

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

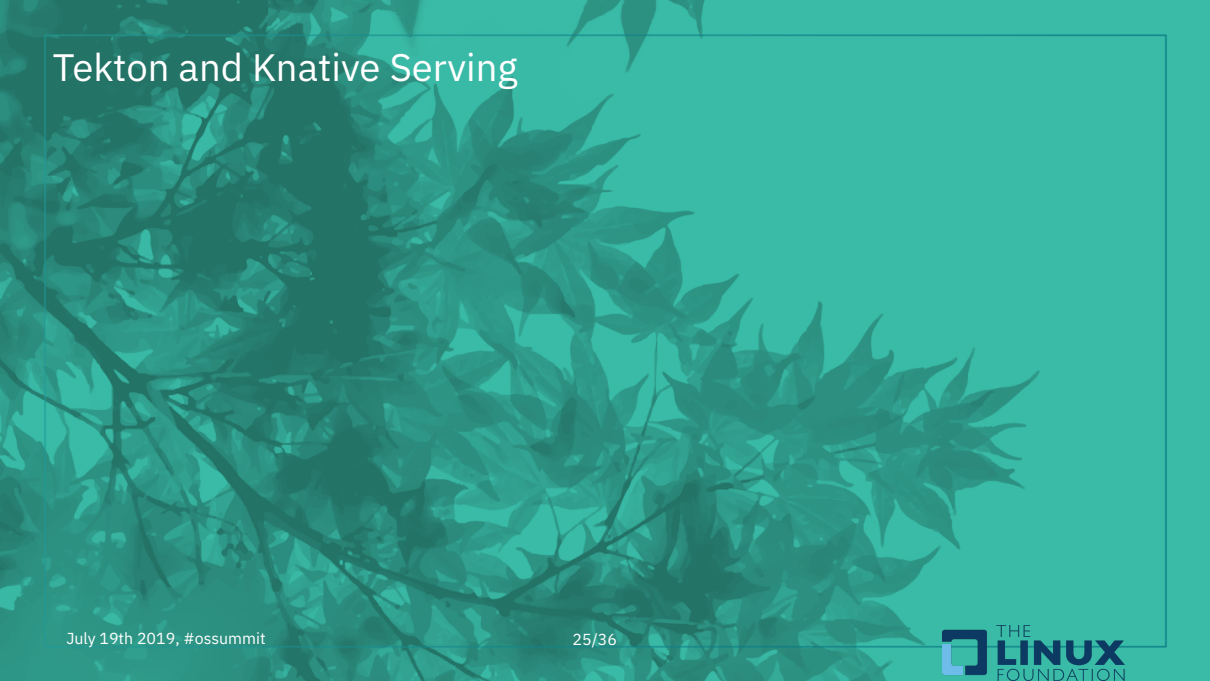
TaskRun:

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

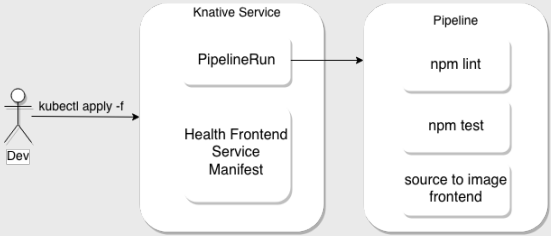
PipelineRun:

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

Tekton and Knative Serving



Pipelines and Knative Build

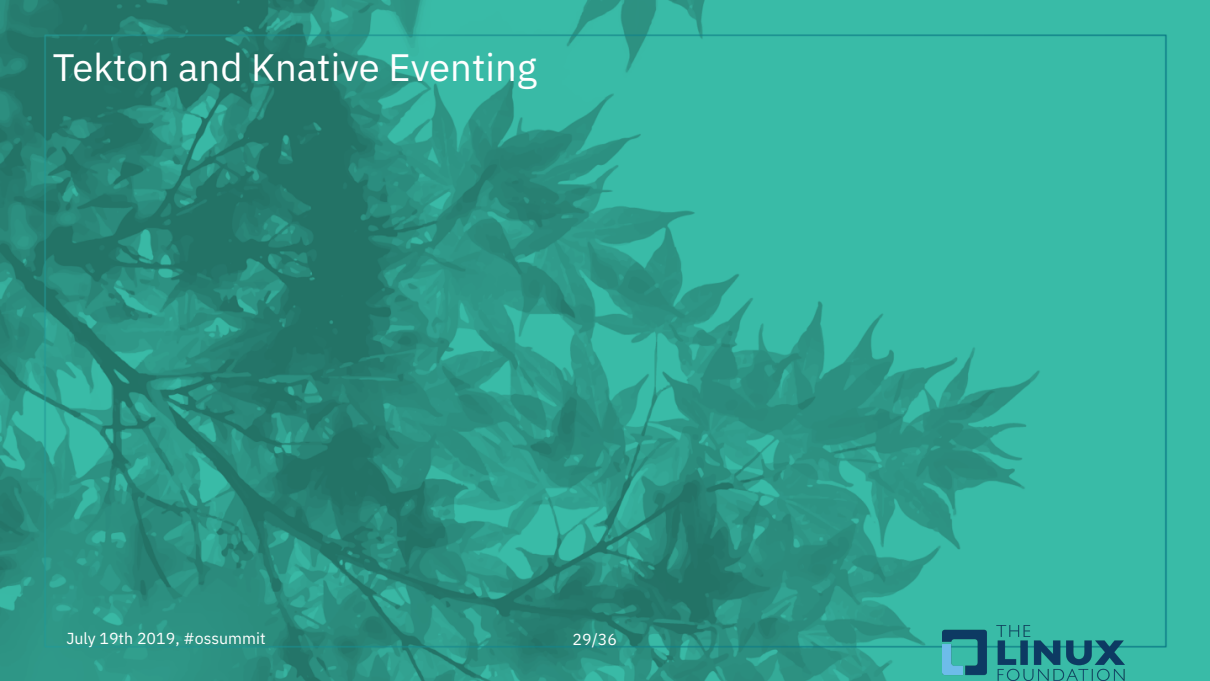


KService for Health Frontend

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: health-frontend
  labels:
    app: health
    component: frontend
    tag: "__TAG__"
spec:
  runLatest:
    configuration:
      build:
        apiVersion: tekton.dev/v1alpha1
        kind: PipelineRun
        metadata:
          labels:
            app: health
            component: frontend
            tag: "__TAG__"
        spec:
          pipelineRef:
            name: dev-test-build-frontend
          params:
            - name: imageTag
              value: "__TAG__"
            - name: nodeTestImage
              value: __NODE_IMAGE_NAME__
          trigger:
            type: manual
          resources:
            - name: src
              resourceRef:
                name: __GIT_RESOURCE_NAME__
            - name: builtImage
              resourceRef:
                name: __IMAGE_RESOURCE_NAME__
      revisionTemplate:  # template for building Revision
        spec:
          container:
            image: us.icr.io/andreaf/health-frontend:__TAG__
            imagePullPolicy: Always
            env:
              - name: API_URL
                value: http://health-api.containers.domain
            ports:
              - name: http1
                containerPort: 80
                protocol: TCP
            livenessProbe:
              httpGet:
                path: /
            readinessProbe:
              httpGet:
                path: /
```

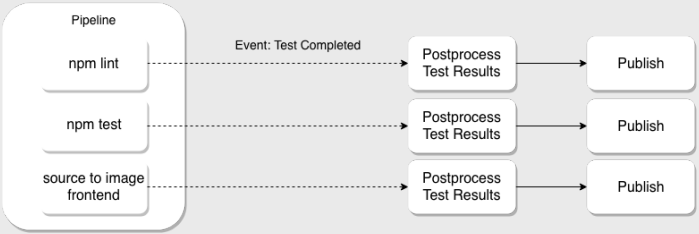
Demo Tekton and Knative Serving

Tekton and Knative Eventing



Triggering and Knative Eventing

- Aysnchronous Pipelines
  - GitHub Source
  - Container Source
  - Pipeline Output as a Source
- Manual trigger for now



Demo Tekton and Knative Eventing

Conclusions

Dev, CI and CD

- Development Workflow:
  - + Local or in-cluster
  - + Reproducibility
  - + Parallel execution
  - - Can be slow
- CI / CD:
  - + Integrates with CI systems
  - + Re-usable blocks
  - + Best practices
  - + Small Footprint
  - - Depends on k8s
  - - Security

Conclusions

- Tekton:
  - Not only for knative
  - Knative first class citizen
  - Very early days
- Tekton Roadmap:
  - Conditional Execution
  - Build Results and Logs
  - Pluggable Tasks
  - Triggering
  - Community Library

References

- This Talk: https://github.com/afrittoli/tekton_pipelines_knative_intro/tree/devoxx-fr
- Tekton Links:
  - https://tekton.dev/, https://cd.foundation/
  - https://github.com/tektoncd/pipeline
  - 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
- Knative Community: https://github.com/knative/docs/tree/master/community
- Kaniko: https://github.com/GoogleContainerTools/kaniko
- Source code and my blog:
  - https://github.com/afrittoli/health-helm/tree/knative
  - https://github.com/afrittoli/openstack-health/tree/knative-eventing
  - https://github.com/afrittoli/github_tekton_glue
  - https://andreafrittoli.me
- IBM Cloud: https://cloud.ibm.com

Q&A