开源技术 * IBM微讲堂

6.案例分享：在DevOps场景中使用Knative

张 龚
Gong Zhang (Grace)
zhanggbj@cn.ibm.com

郭迎春
Daisy
guoyingc@cn.ibm.com

IBM **Developer**

IBM

Knative系列公开课

每周四晚8点档

- ◆09月19日 《Knative概览》
- ◆09月26日 《Tekton从源码构建容器镜像》
- ◆10月10日 《Knative Serving让容器从另扩展到无限》
- ◆10月17日 《Knative Eventing打造标准的云计算事件平台》
- ◆10月24日 《Knative客户端工具介绍》
- ◆10月31日 《案例分享：在DevOps场景中使用Knative》

讲师介绍

张 龚 Gong Zhang (Grace)
IBM 中国研发中心
高级工程师

目前主要负责IBM Cloud Foundry Enterprise Environment服务的开发和构建，是Cloud Foundry BOSH SoftLayer CPI和Knative Client开源社区项目的贡献者。

郭迎春 (Daisy)
IBM开发中心 开放技术工程院

多年开源社区经验
Knative贡献者
负责Knative Eventing Client开发
专注Serverless平台建设

Agenda

- Background: Cloud Native DevOps and Knative
- User Scenario #1: from source code to production
  - Trigger, Build, Deploy
  - Notification
  - Demo
- User Scenario #2: routing and canary upgrade
  - Traffic Splitting
  - Demo

Background: Cloud Native DevOps and Knative

DevOps

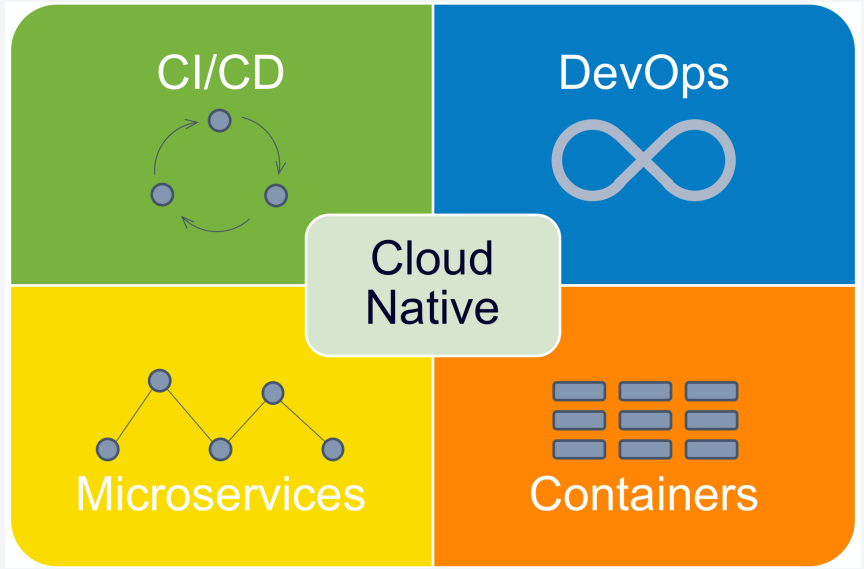
DevOps is a set of practices that combines software development (Dev) and information-technology operations (Ops) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. -- *Wikipedia*

Cloud Native DevOps

Cloud native technologies — containers, microservices and serverless functions that run in multicloud environments and are managed through automated CI/CD pipelines — are built on DevOps principles. You cannot have one without the other. -- *Guide to Cloud Native DevOps*

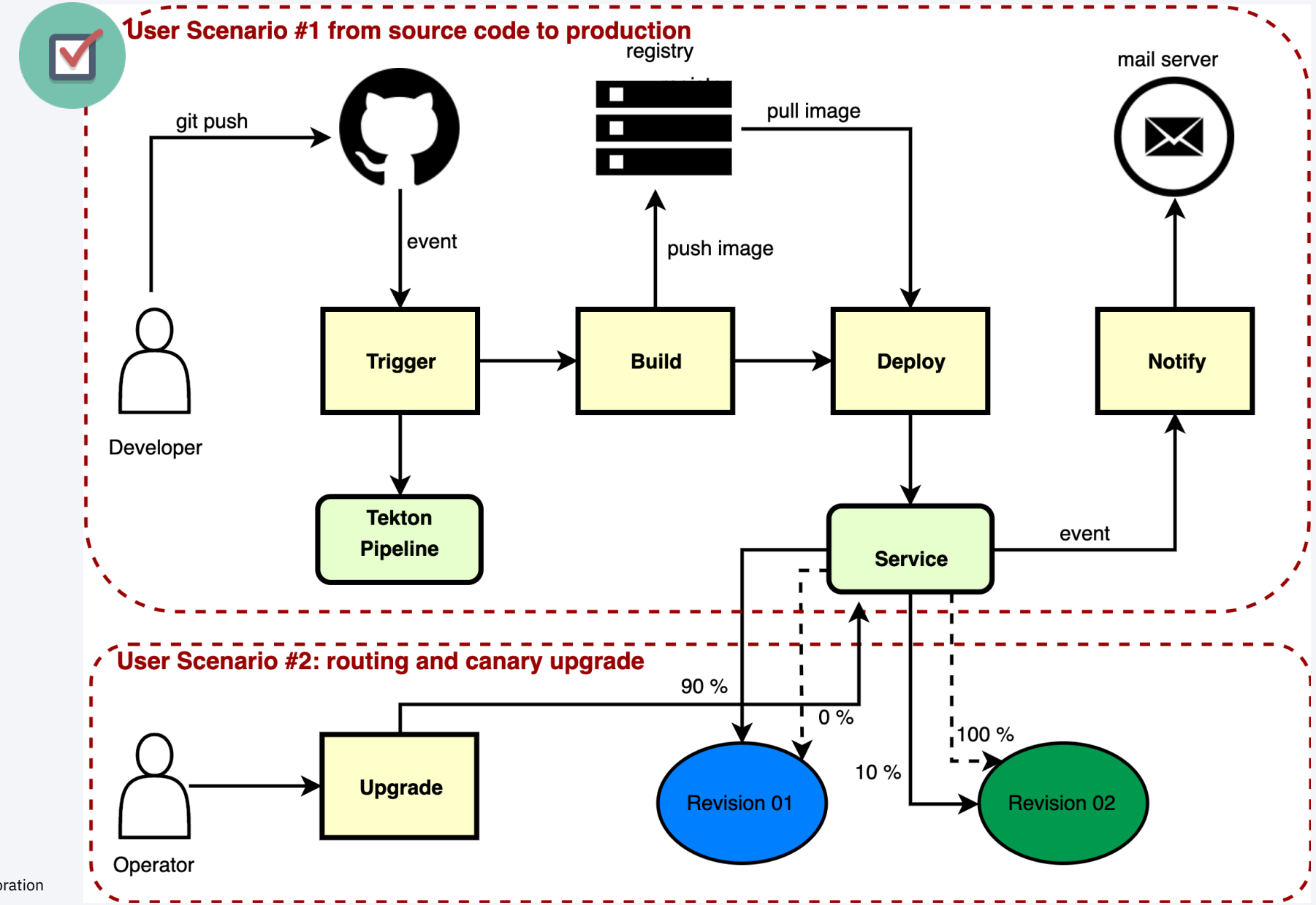
Knative

Knative which integrates with both Kubernetes and Istio provides a complete application/function deployment platform, including setting up a build process, automated deployment, and an eventing mechanism that standardizes the way applications use messaging and queueing systems.



From Cloud Native – Seeing through the hype

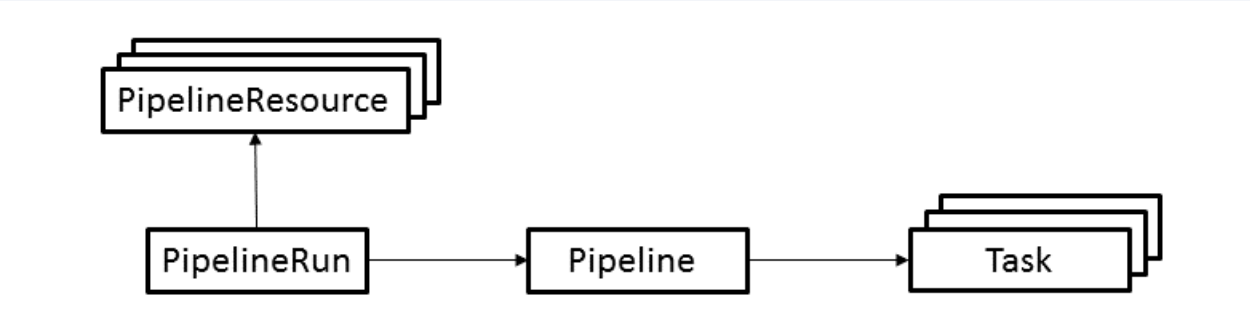
User Scenarios Overview



User Scenario #1: from source code to production Trigger, Build and Deploy

Tekton Pipeline

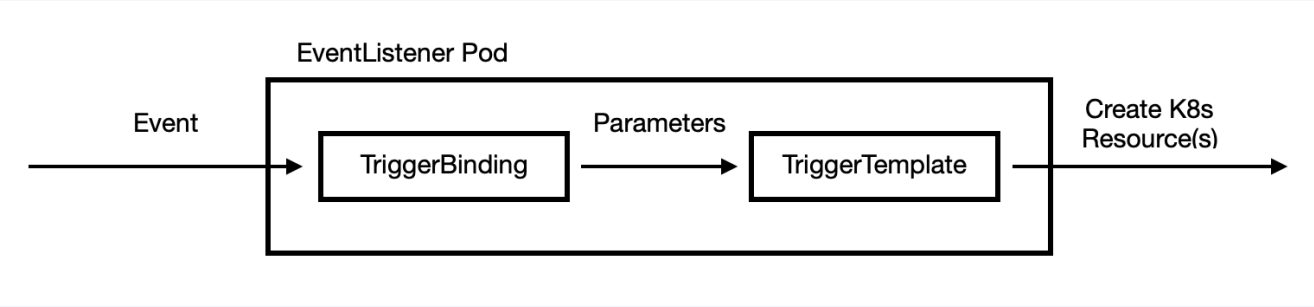
Tekton is a powerful yet flexible Kubernetes-native open-source framework for creating CI/CD systems. It lets you build, test, and deploy across multiple cloud providers or on-premises systems by abstracting away the underlying implementation details.



Tekton Pipeline Resources

Tekton Trigger

Trigger is a Kubernetes Custom Resource Defintion (CRD) controller that allows you to extract information from events payloads (a "trigger") to create Kubernetes resources.



Tekton Trigger Resources

Using triggers in conjunction with tekton pipeline enables you to easily create full-fledged CI/CD systems!

User Scenario #1: from source code to production Trigger, Build and Deploy

Tekton Trigger

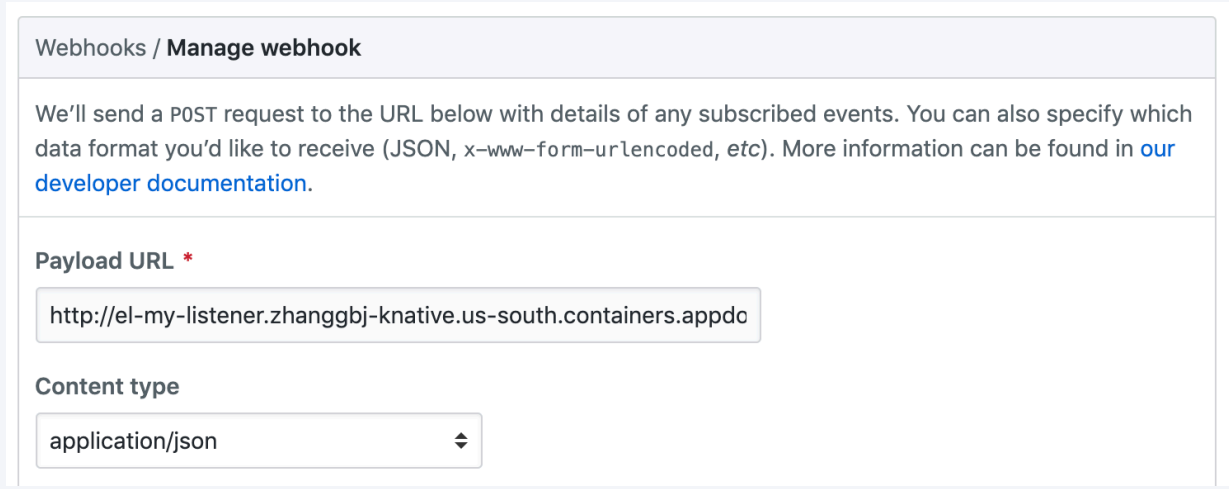
- Install Trigger
- TriggerTemplate is a resource that can template resources.
- TriggerBindings enable you to capture fields within an event payload and store them as parameters.
- EventListeners connect TriggerBindings to TriggerTemplates and provide an addressable endpoint, which is where webhooks/events are directed.

Git Webhook Settings

Set a webhook with Payload URL in your git repo for application source code.

源码获取：https://github.com/zhanggbj/triggers
https://github.com/zhanggbj/tekton-tutorial

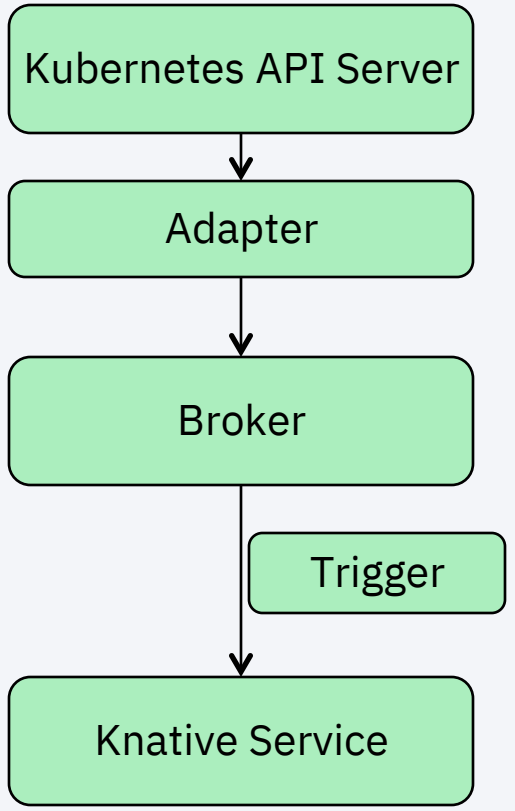
```
$ kubectl get deployment -n tekton-pipelines
NAME                         READY  UP-TO-DATE  AVAILABLE  AGE
tekton-pipelines-controller  1/1    1           1          148d
tekton-pipelines-webhook     1/1    1           1          148d
tekton-triggers-controller   1/1    1           1          14d
tekton-triggers-webhook      1/1    1           1          14d
$ kubectl get deployment
el-my-listener               1/1    1           1          5d1h
$ kubectl get ingress
el-my-listener   el-my-listener.zhanggbj-knative.us-
south.containers.appdomain.cloud  169.48.224.198  80    5d1h
```



User Scenario #1: from source code to production Notify by Knative Eventing

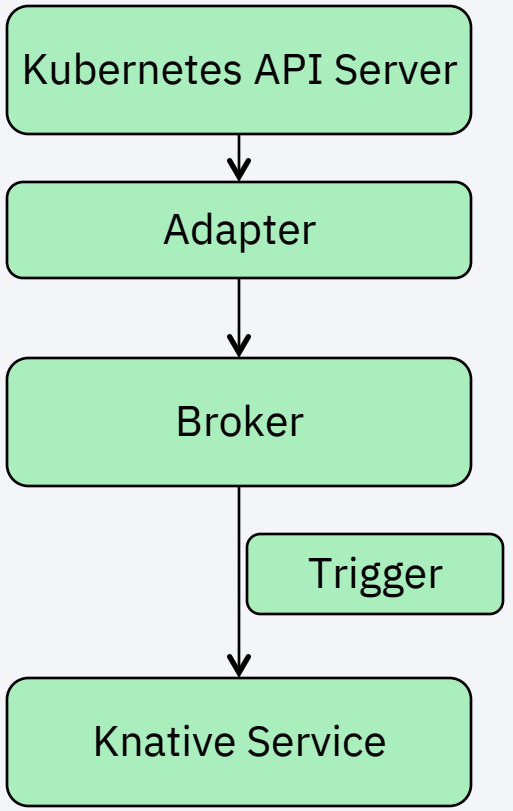
Knative Eventing

- ApiServerSource is a predefined event source which will capture the create/update/delete of K8s objects.
- Broker represents an ‘event mesh’. Events are sent to the Broker’s ingress and are then sent to any subscribers that are interested in that event.
- Trigger represents a desire to subscribe to events from a specific Broker.
- Knative Service `service-details` is a micro service which will send the Service information to an email address when the Service is ready to use.



源码获取：https://github.com/daisy-ycguo/knative-devops

User Scenario #1: from source code to production Notify by Knative Eventing



ApiServerSource

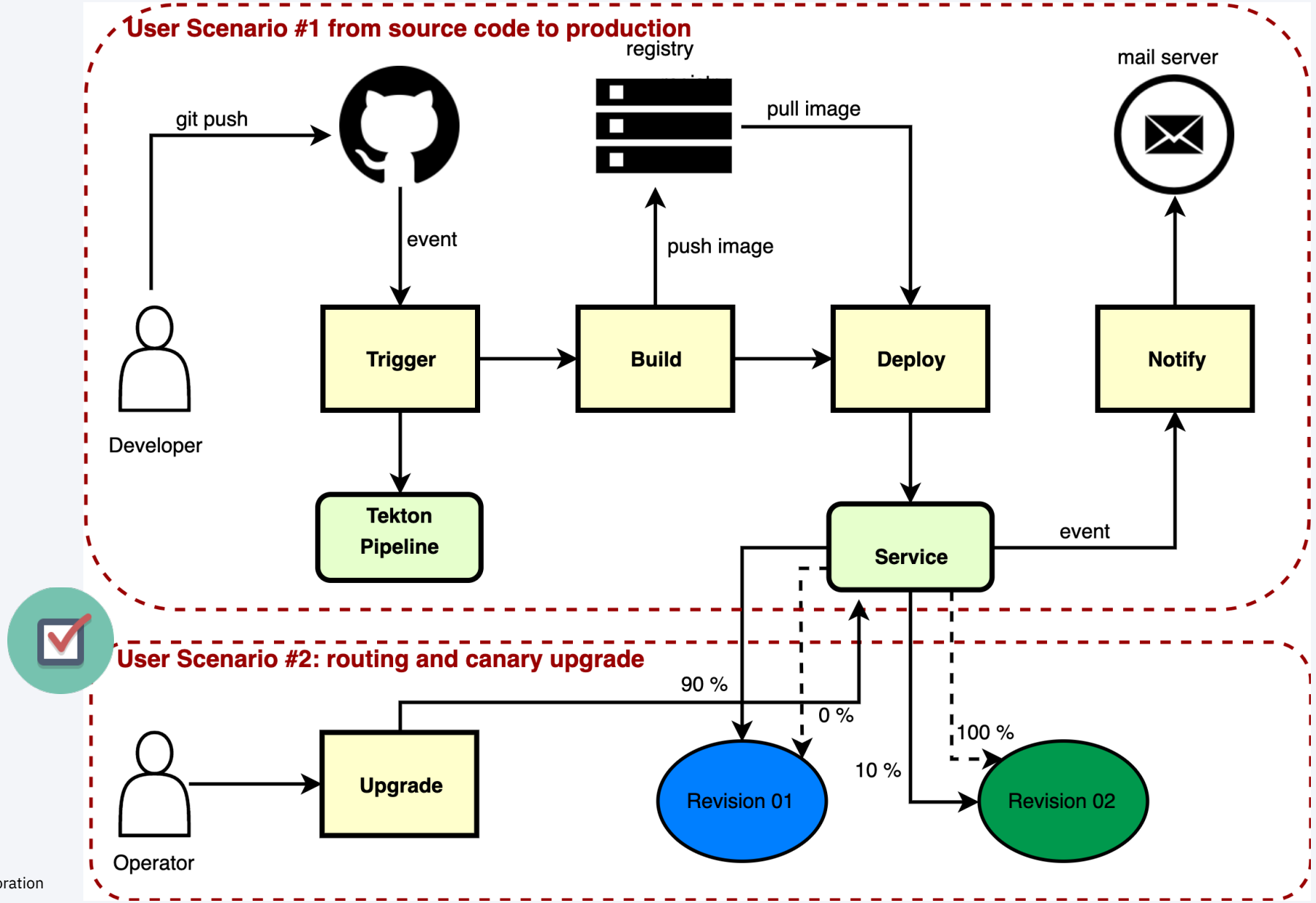
```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: ApiServerSource
metadata:
  name: service-events
  namespace: default
spec:
  serviceAccountName: service-sa
  mode: Resource
  resources:
  - apiVersion: serving.knative.dev/v1alpha1
    kind: Service
  sink:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: Broker
    name: default
```

Trigger

```yaml
apiVersion: eventing.knative.dev/v1alpha1
kind: Trigger
metadata:
  name: trigger-service
spec:
  broker: default
  filter:
    sourceAndType:
      type: dev.knative.apiserver.resource.update
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1alpha1
      kind: Service
      name: service-details
```

User Scenario #1: from source code to production Trigger, Build, Deploy and Notify

User Scenarios Overview



User Scenario #2: routing and canary upgrade Routing and Canary Upgrade

Percentage-Based Routing

A common requirement in today's distributed service world is to split traffic between revisions. This feature could be useful for A/B testing or it could be used to try some new or selected features with a smaller percentage of users (Canary Upgrade).

Feature-Flag Routing

(In Discussion https://github.com/knative/serving/issues/4736)

As in production, users may not only want the traffic splitting by percentage, but also feature-flag routing, which means fine-grained traffic management based on incoming request, e.g. the request headers, uri path and method etc.

```
apiVersion: serving.knative.dev/
v1alpha1
kind: Service
metadata:
 name: myservice
spec:
 template:
  spec:
   containers:
    - image: example/myservice
 traffic:
 - tag: current
  revisionName: myservice-v1
  percent: 50
 - tag: candidate
  revisionName: myservice-v2
  percent: 50
```

Service Traffic Block

```
spec:
 template:
  spec:
   containers:
    - image: example/myservice
 traffic:
 - tag: current
  revisionName: myservice-v1
  match:
     headers:
        end-user:
           exact: jason
 - tag: candidate
  revisionName: myservice-v2
   match:
     headers:
        end-user:
           exact: mike
```

Service Traffic Block Proposal

User Scenario #2: routing and canary upgrade
Routing and Canary Upgrade

Canary Upgrade

Canary deployments are a pattern for rolling out releases to a subset of users or servers. The idea is to first deploy the change to a small subset of servers, test it, and then roll the change out to the rest of the servers. The canary deployment serves as an early warning indicator with less impact on downtime: if the canary deployment fails, the rest of the servers aren't impacted.

```
# Install Kn Client  https://github.com/knative/client

# Deploy Service myservice
Kubectl apply –f myservice.yaml

# Tag revision myservice-hqpfb
kn service update myservice --tag myservice-hqpfb=current

# Lock down traffic to myservice-v1 with tag current
kn service update myservice --traffic current=100

# Deploy a new myservice with tag candidate. So far, 100%
traffic should go to myservice-v1
kn service update --image docker.io/org/myimage
kn service update myservice --tag myservice-zfgkk=candidate

# Rollout myservice with 10% traffic to candidate
kn service update myservice --traffic current=90 --traffic
candidate=10

# If it is ready to rollout to the new revision
kn service update myservice --traffic candidate=100
```

Use kn client for service canary upgrade

Links

- https://github.com/IBM/tekton-tutorial
- https://github.com/tektoncd/triggers