

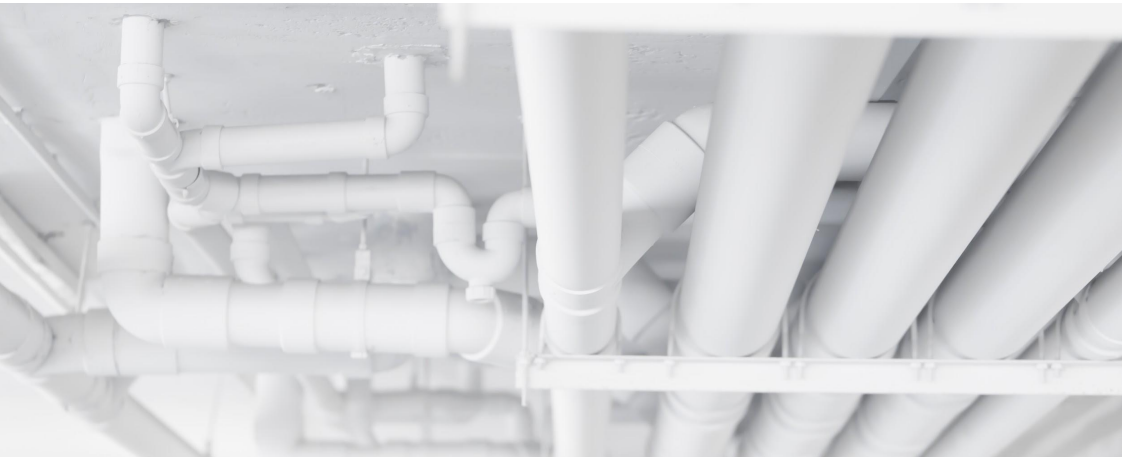
CI/CD on OpenShift

Traditional and Cloud-Native Pipelines

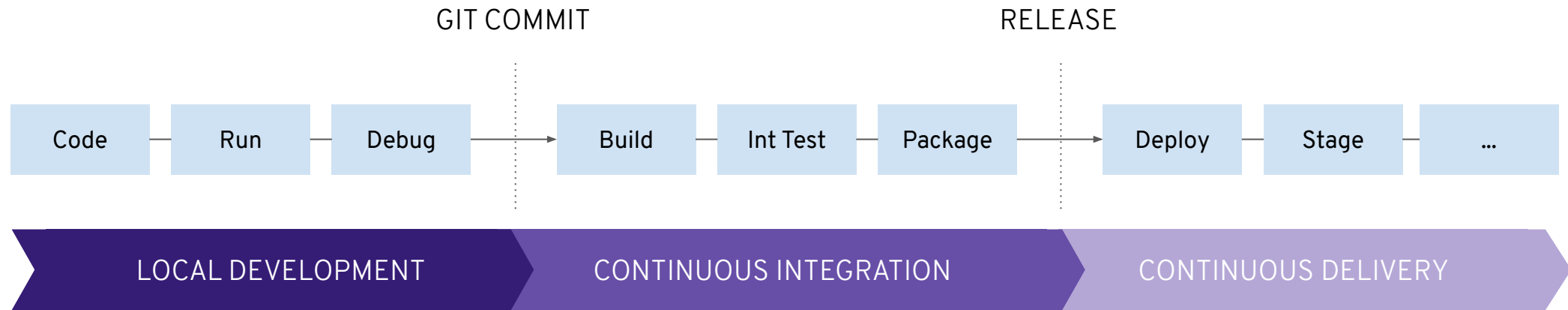
Wanja Pernath

EMEA Partner Enablement Manager, OpenShift & MW

What is CI/CD?



Continuous Integration and Continuous Delivery (CI/CD)

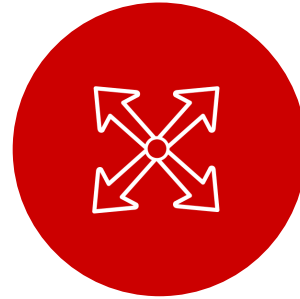


What is Cloud-Native CI/CD?



Containers

Built for container apps and runs on Kubernetes



Serverless

Runs serverless with no CI/CD engine to manage and maintain



DevOps

Designed with microservices and distributed teams in mind

Why Cloud-Native CI/CD?

Traditional CI/CD	Cloud-Native CI/CD
Designed for Virtual Machines	Designed for Containers and Kubernetes
Require IT Ops for CI engine maintenance	Pipeline as a service with no Ops overhead
Plugins shared across CI engine	Pipelines fully isolated from each other
Plugin dependencies with undefined update cycles	Everything lifecycled as container images
No interoperability with Kubernetes resources	Native Kubernetes resources
Admin manages persistence	Platform manages persistence
Config baked into CI engine container	Configured via Kubernetes ConfigMaps

Why Cloud-Native CI/CD?

Traditional CI/CD

Designed for Virtual Machines

Require IT Ops for CI engine maintenance



Jenkins

Plugins shared across CI engine
Plugin dependencies with undefined update cycles

No interoperability with Kubernetes resources

Admin manages persistence

Config baked into CI engine container

Cloud-Native CI/CD

Designed for Containers and Kubernetes

Pipeline as a service with no Ops overhead



TEKTON

Plugins fully isolated from each other
Evolving library of reusable pipeline images

Native Kubernetes resources

Platform manages persistence

Configured via Kubernetes ConfigMaps

Traditional vs Cloud-Native CI/CD

Traditional CI/CD

- Monolithic
- Central governance
- Existing investments

Cloud-Native CI/CD

- Serverless
- Cross-functional teams
- Kubernetes-native

Traditional vs Cloud-Native CI/CD

Traditional CI/CD

- Monolithic
- Central governance
- Existing investments

Cloud-Native CI/CD

- Serverless
- Cross-functional teams
- Kubernetes-native

Red Hat
OpenShift



Jenkins



OpenShift Pipelines



OpenShift Pipelines



Kubernetes-native
declarative
Pipelines with
Tekton



Serverless CI/CD
with no single server
to share and
maintain



Run pipelines in
isolated containers with
all required
dependencies



Standard and
portable to any
Kubernetes
platform



Web, CLI, and
Visual Studio
Code and IDE
plugins

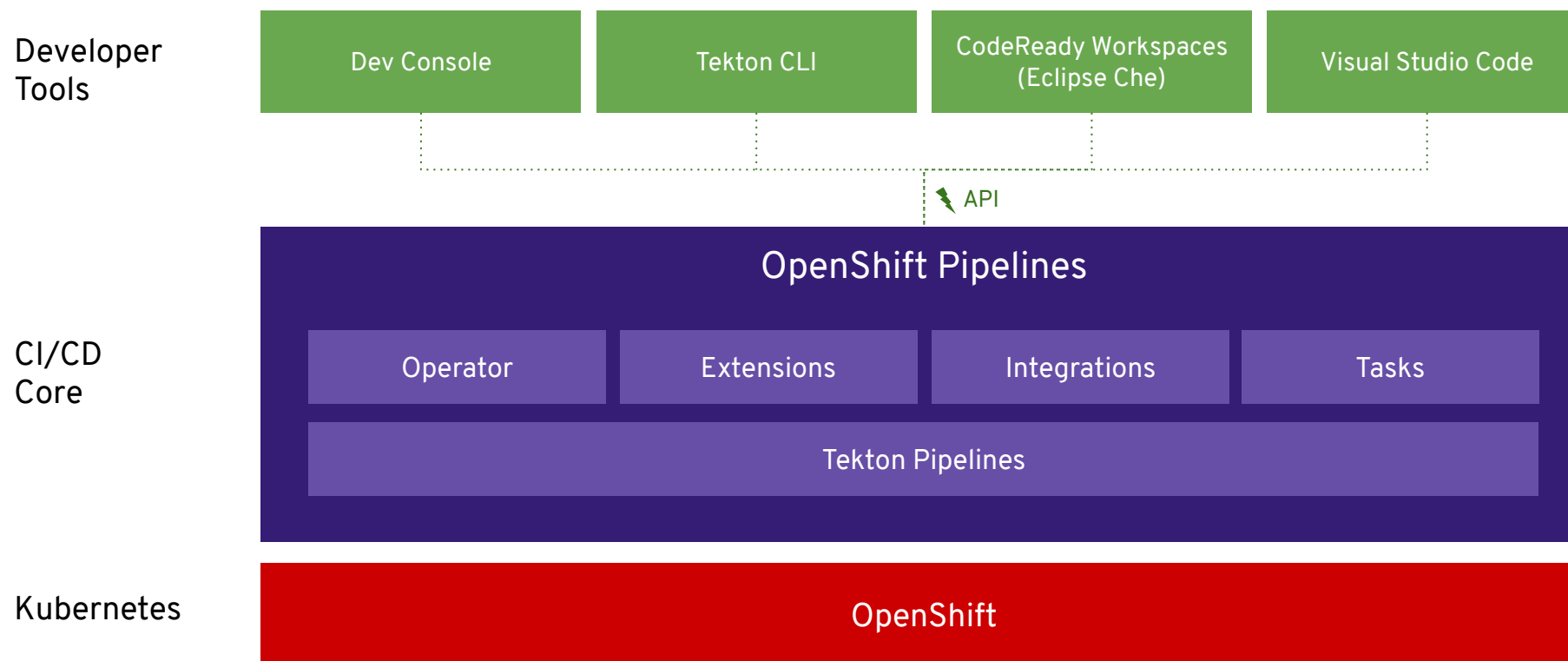


An open-source project for providing a set of shared and standard components for building Kubernetes-style CI/CD systems



Governed by the Continuous Delivery Foundation
Contributions from Google, Red Hat, Cloudbees, IBM, Pivotal and many more

OpenShift Pipelines Architecture



Tekton Deep Dive



Tekton Concepts

Step

Run commands in a container with volumes, env vars, etc

Task

A list of steps that run sequentially in the same pod

Pipeline

A graph of tasks executed in a certain order

Pipeline Resource

Inputs and outputs to tasks and pipelines (git, image, etc)

Task Run

An invocation of a task with inputs and outputs

Pipeline Run

An invocation of a pipeline with inputs and outputs

Condition

An check that can determine if a task should be executed

Catalog

An collection of reusable tasks

Triggers

A Tekton sub-project to start pipelines based on events

Steps

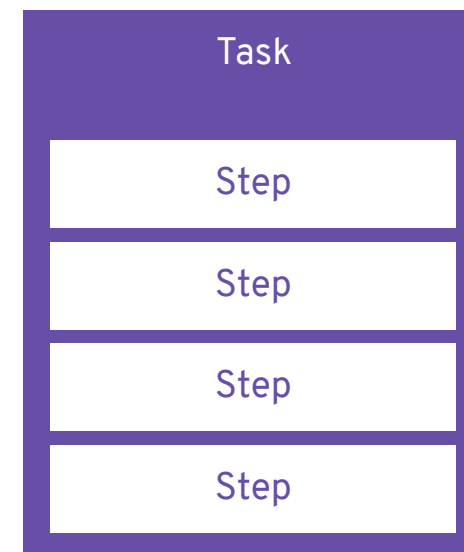
- Run command or script in a container
- Kubernetes container spec
 - Env vars
 - Volumes
 - Config maps
 - Secrets

```
- name: build
  image: maven:3.6.0-jdk-8-slim
  command: ["mvn"]
  args: ["install"]
```

```
- name: parse-yaml
  image: python3
  script: |-
    #!/usr/bin/env python3
    ...
```

Task

- Defines a unit of work to be executed
- A list of steps to run sequentially
- Step containers run in the task pod
- Has inputs, outputs and parameters
- Workspaces and results for sharing data
- Can run independent of pipelines



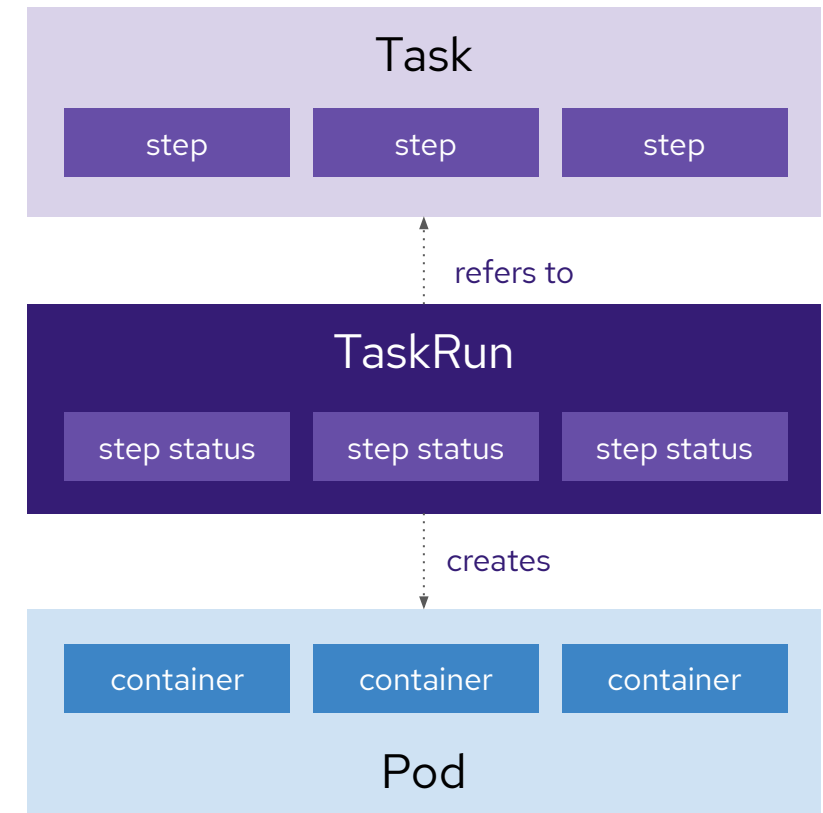
Example Tasks: Maven Install, AWS CLI, Kubectl Deploy, Security Scan, etc

Maven Task Example

```
kind: Task
metadata:
name: maven
spec:
  params:
    - name: goal
      type: string
      default: package
  steps:
    - name: mvn
      image: maven:3.6.0-jdk-8-slim
      command: [ mvn ]
      args: [ $(params.goal) ]
```

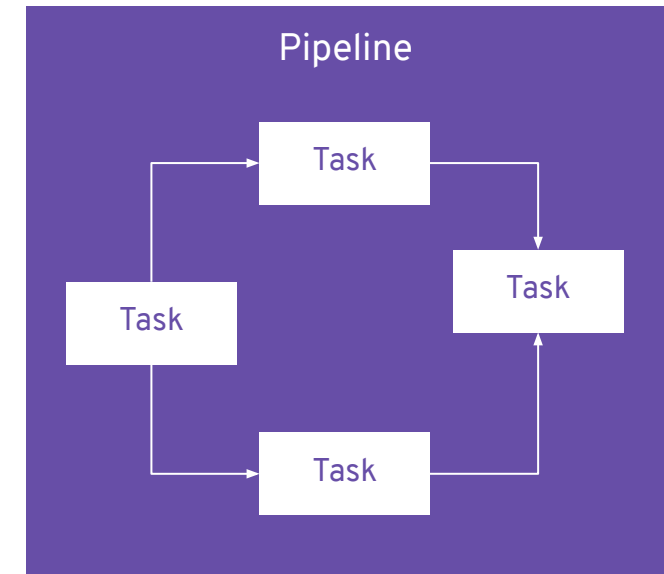
TaskRun

- Runs a Task to completion in a pod
- References or embeds a Task spec
- Provides input to Tasks
 - Parameters
 - Resources
 - Service account
 - Workspaces
- Contains execution status and metadata



Pipeline

- Define Tasks execution order (graph)
- Inputs and parameters
- Retries tasks
- Conditional task execution
- Workspaces for sharing data between tasks
- Reusable across projects



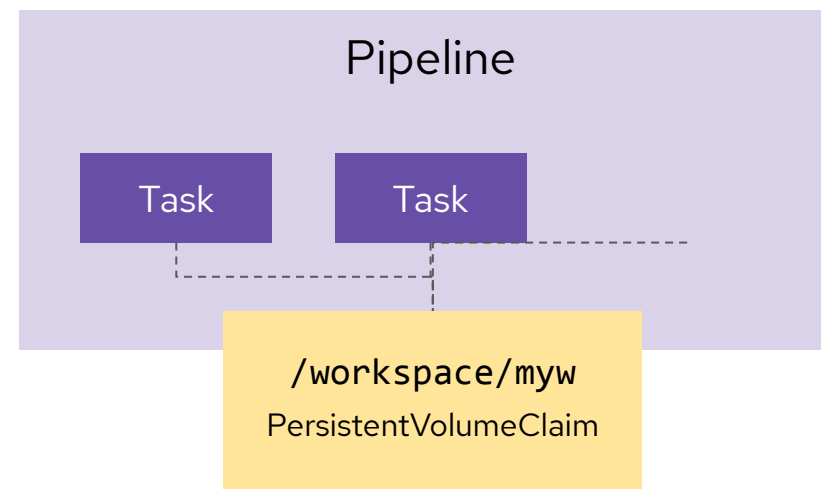
Sharing Data Between Tasks

Task: results

- Task exposes data as variables
- Suitable for small pieces of data
- Examples: commit id and branch name

Task: workspaces

- Shared volumes between tasks
 - Persistent volumes
 - Config maps
 - Secrets
- Suitable for large data
- Examples: code, binaries, reports



Conditions

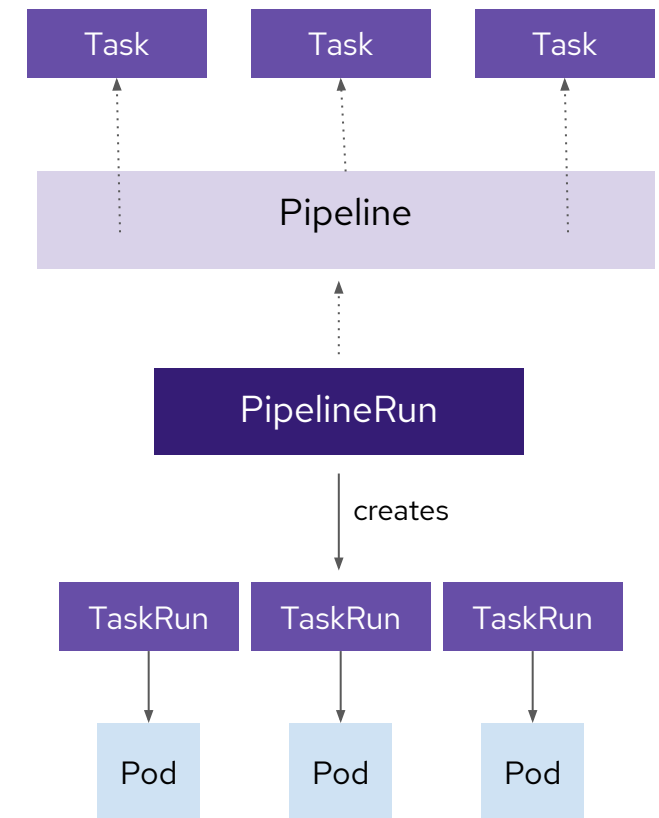
- Defines a single checks
- Used in conditional tasks
- Parameterized
- Command run in container
 - True: exit code - 0
 - False: non-zero exit code

```
kind: Condition
metadata:
name: deployment-exists
spec:
  params:
  - name: appName
  check:
    image: openshift-cli
    script: oc get deployment ${params.app}
```

```
kind: Pipeline
spec:
  tasks:
  - name: run-tests
    taskRef: { image: ui-test-runner }
    conditions:
    - conditionRef: deployment-exists
      params:
      - {name: appName, value: api }
```

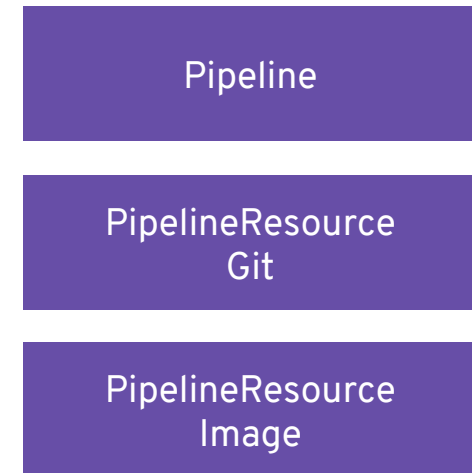
PipelineRun

- Runs a pipeline to completion
- References or embeds a Pipeline spec
- Creates TaskRuns to execute Tasks in the Pipeline
- TaskRun pods may get scheduled on different node
- Provides inputs and params to pipeline
- Provides volumes for declared pipeline workspaces



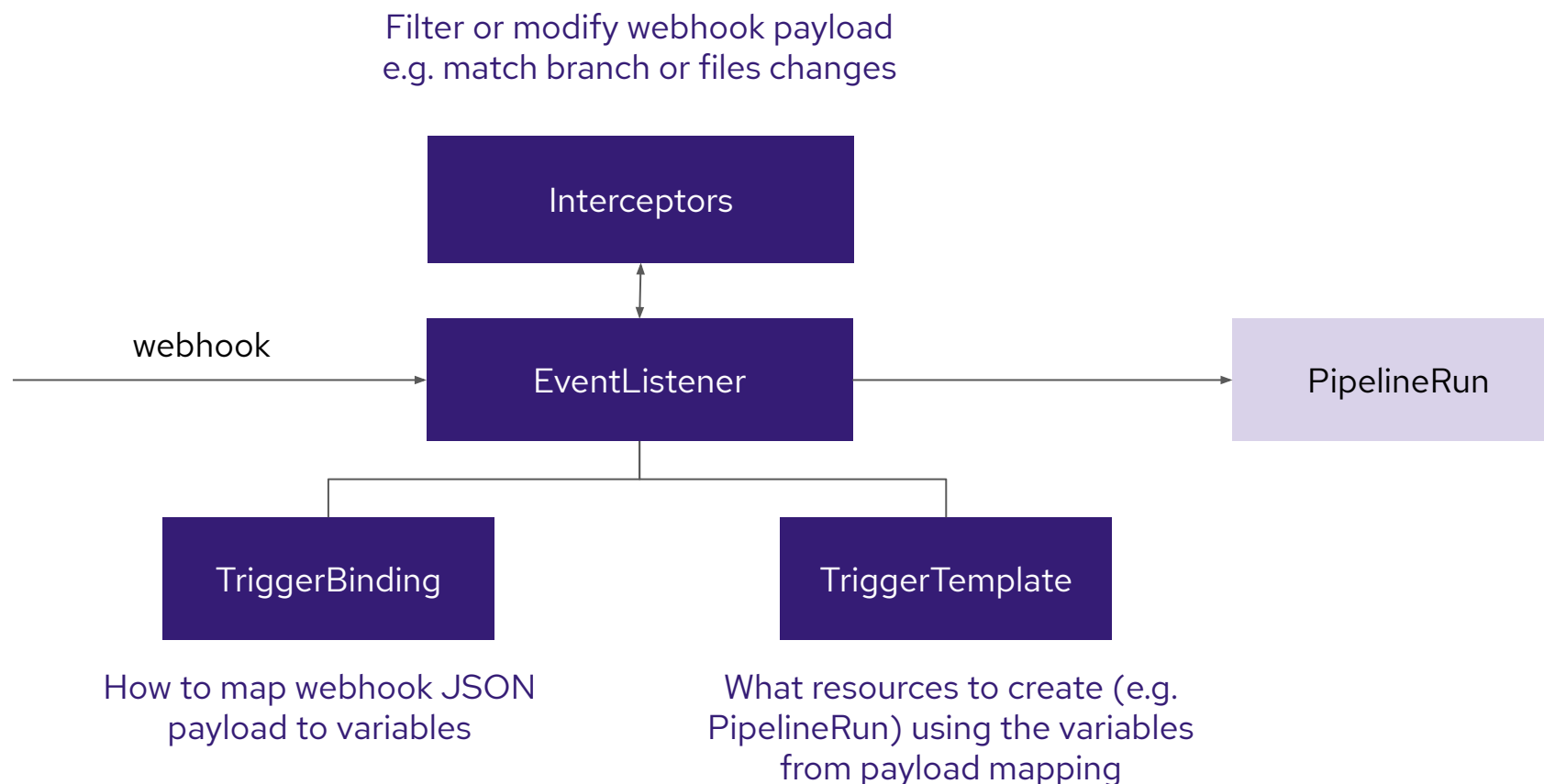
PipelineResource

- Inputs and outputs of tasks and pipelines
 - git repository
 - image in a registry
 - cluster credentials
 - storage
 - ...and mo
- Decoupled from pipeline definition
- Reusable across pipelines



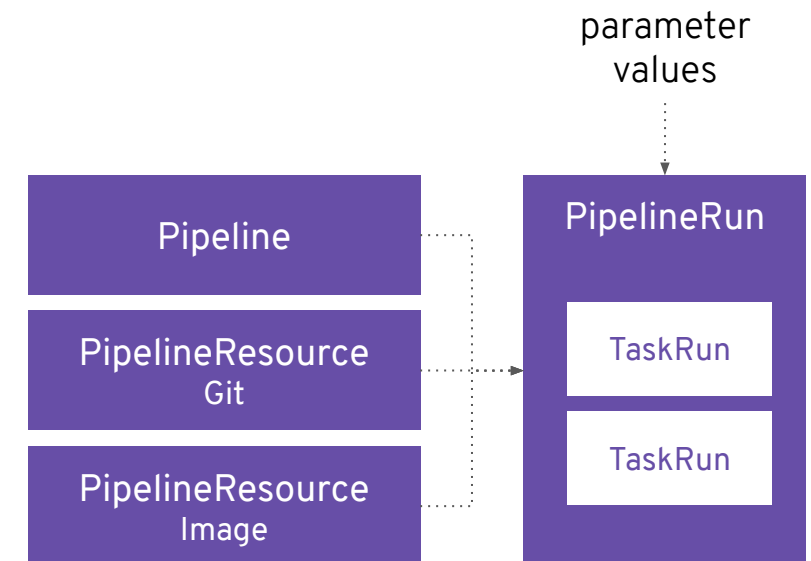
Triggers

Run pipelines based on events like HTTP webhooks on commit, pull request, etc





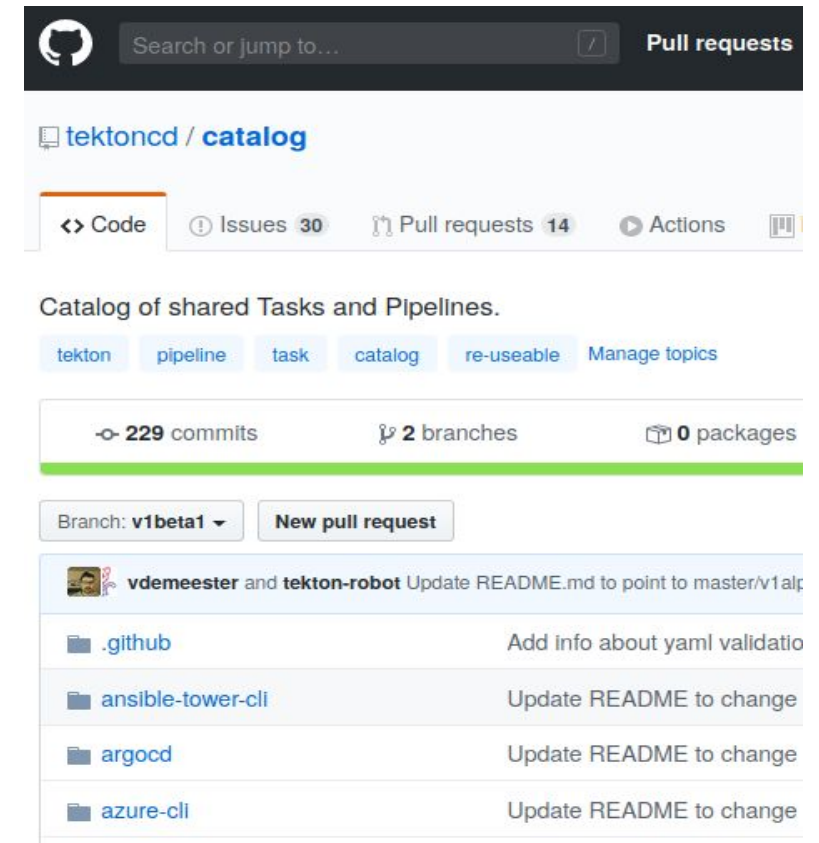
TaskRun and PipelineRun

- Runtime CRDs
- Invocation of Task and Pipeline
- Reference tasks and pipelines
- Provide inputs, outputs and params

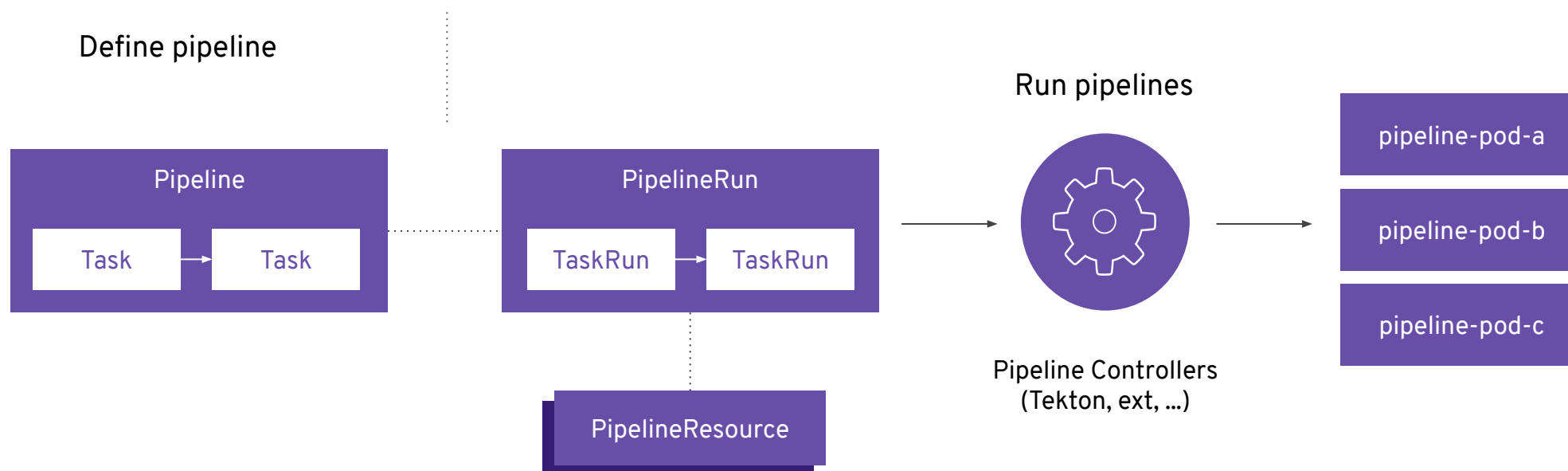


Task Catalog

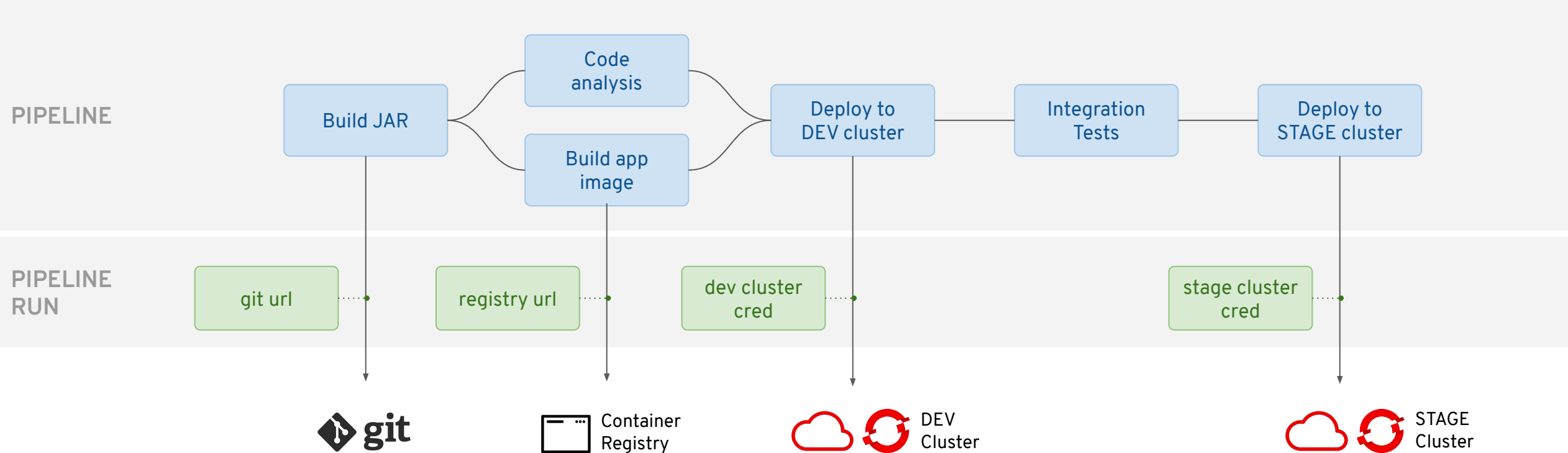
- Catalog of reusable Tasks
 - Image build: buildah, kaniko, jib, buildpacks, etc
 - Source-to-Image: Java, Python, Go, Ruby, etc
 - Language specific: maven, gradle, go, ...
 - More to come soon
- Import and compose pipelines
- Available catalogs
 -  tektoncd/catalog
 -  openshift/pipelines-catalog



OpenShift Pipelines Architecture



Tekton Pipeline Example



Install OpenShift Pipelines Operator

The screenshot shows the OpenShift OperatorHub interface. On the left is a sidebar with navigation links: Administrator, Home, Overview, Projects, Search, Explore, Events, Operators (selected), OperatorHub, Installed Operators, Workloads, Serverless, Networking, Storage, Builds, Pipelines, and Monitoring. The main content area displays the details for the OpenShift Pipelines Operator (version 0.10.7, provided by Red Hat). A modal window is open, showing the operator's details and a list of features.

OperatorHub

Discover Operators from the Kubernetes community. Install Operators on your clusters to provide optional add-on self-service experience.

Operator Version
0.10.7

Capability Level

- ☒ Basic Install
- ☐ Seamless Upgrades
- ☐ Full Lifecycle
- ☐ Deep Insights
- ☐ Auto Pilot

Provider Type
Community

Provider
Red Hat

Repository
<https://github.com/openshift/tektoncd-pipeline-operator>

Container Image
quay.io/openshift-pipeline/openshift-pipelines-operator:v0.10.7

Installed Operator


This Operator has been installed on the cluster. [View it here.](#)


Features

- Standard CI/CD pipelines definition
- Build images with Kubernetes tools such as S2I, Buildah, Buildpacks, Kaniko, etc
- Deploy applications to multiple platforms such as Kubernetes, serverless and VMs
- Easy to extend and integrate with existing tools
- Scale pipelines on-demand
- Portable across any Kubernetes platform
- Designed for microservices and decentralised team
- Integrated with OpenShift Developer Console

Installation

OpenShift Pipelines Operator gets installed into a single namespace which would then install *OpenShift Pipelines* into the same namespace. *OpenShift Pipelines* is however cluster-wide and can run pipelines created in any namespace.

 **Red Hat**
OpenShift Container Platform



Developer

+ Add

Topology

Builds

Pipelines

Advanced

Project: Project01

Pipelines > Pipeline Run Details

PR pipelinerun01a Running

Tech Preview

Actions

Overview | YAML | Logs

Pipeline Run Overview

code compile

compile & test

unit test

security check

image build

Name

pipelinerun01a

Namespace

NS project01

Labels

app=dummy-mongo-pod-test bap.me/environment=dev

bap.me/track=experimental bap.me/tier=backend

Annotations

0 Annotations

Created At

Aug 8, 4:00 pm

siamak

Developer

+ Add

Topology

Builds

Pipelines

Advanced

Project: Project01

[Pipelines](#) > Pipeline Run Details

PR

pipeline

run01a

Running

Tech Preview

Actions

Overview

YAML

Logs

code compile

compile & test

unit test

security check


image build




image build

```

[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> looking
for get /health in extlib/lib/perl5/Dancer2/Core/App.pm l. 36
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> Entering
hook core.error.init in (eval 306) l. 1
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> Entering
hook core.error.before in (eval 306) l. 1
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> looking
for get /health in extlib/lib/perl5/Dancer2/Core/App.pm l. 36
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> Entering
hook core.error.init in (eval 306) l. 1
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23 18:28:53> Entering
hook core.error.before in (eval 306) l. 1
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsi:54] core @2018-08-23

```


Red Hat
 OpenShift Container Platform




 siamak

</> Developer

+Add

Topology

Builds

Pipelines


Advanced

Project: pipelines-demo
 Application: all applications

Add


No workloads found

To add content to your project, create an application, component or service using one of these options.




From Git

Import code from your git repository to be built and deployed




Container Image

Deploy an existing image from an image registry or image stream tag




From Catalog

Browse the catalog to discover, deploy and connect to services




From Dockerfile

Import your Dockerfile from your git repo to be built & deployed



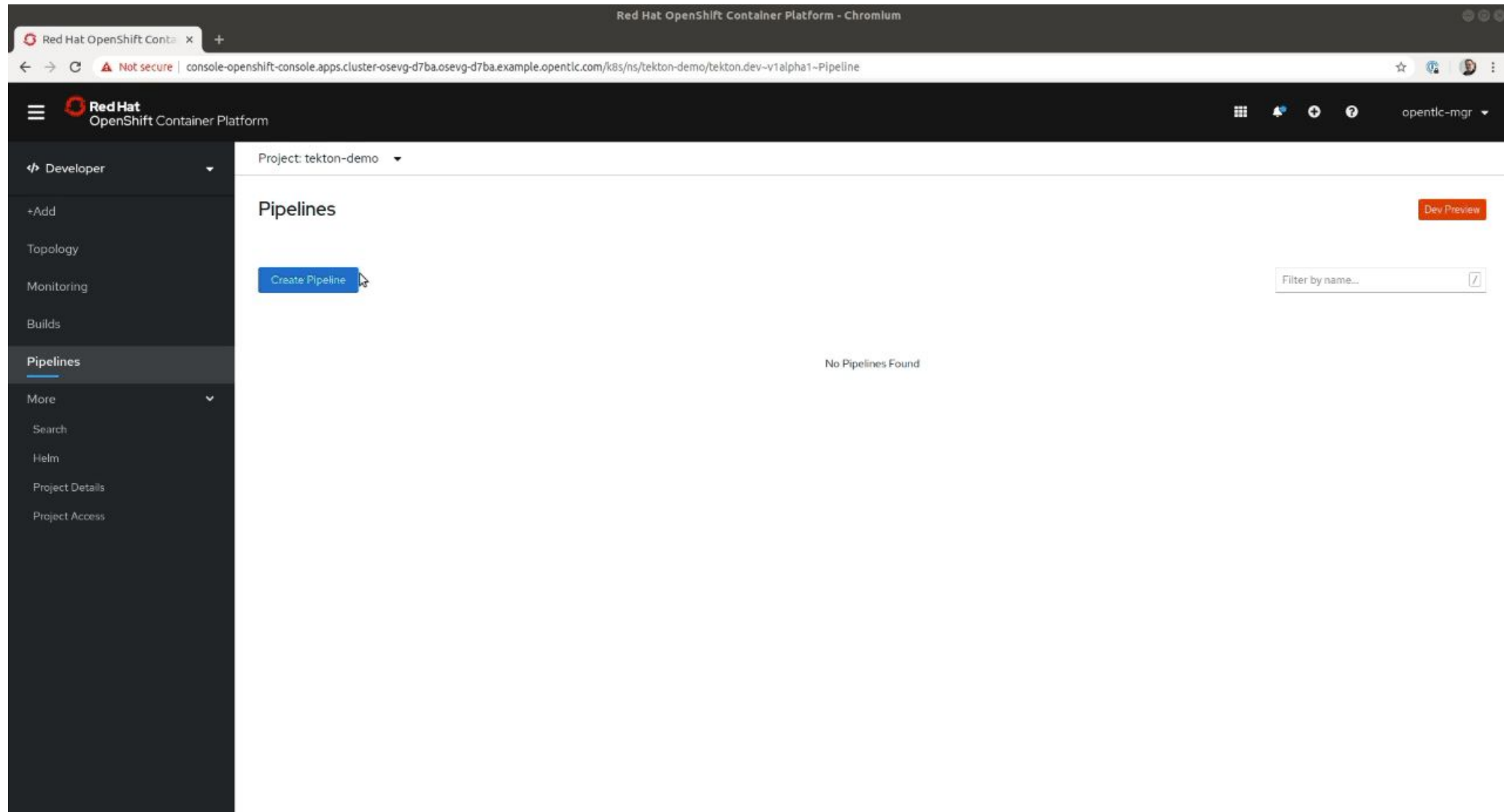
YAML

Create resources from their YAML or JSON definitions



Database

Browse the catalog to discover database services to add to your application



Manage Pipelines with Tekton CLI

```
ssadeghi@Siamaks-MacBook-Pro: ~ (zsh)
[~]$ # brew tap tektoncd/tools
[~]$ # brew install tektoncd/tools/tektoncd-cli
[~]$
[~]$
```

Tekton Pipelines VS Code Extension

The screenshot displays the Tekton Pipelines VS Code extension interface. On the left, a sidebar shows the 'TEKTON PIPELINES' tree with categories like Pipelines, Tasks, ClusterTasks, PipelineResources, TriggerTemplates, TriggerBinding, EventListener, and Conditions. The main editor shows a file named 'sample_test.yaml' with the following YAML content:

```
pipeline <> sample_test.yaml > {} spec > [] tasks
1  apiVersion: tekton.dev/v1alpha1
2  kind: Pipeline
3  metadata:
4    name: demo-pipeline
5  spec:
6    resources:
7      - name: source-repo
8        type: git
9      - name: web-image
10       type: image
11      - name: app-image
12        type: image
13    tasks:
14      - name: skaffold-unit-tests
15        taskRef:
16          name: unit-tests
17        resources:
18          inputs:
19            - name: workspace
20              resource: source-repo
21      - name: build-skaffold-web
22        runAfter: [skaffold-unit-tests]
23        taskRef:
24          name: build-push
25        params:
26          - name: pathToDockerFile
27            value: Dockerfile
28          - name: pathToContext
29            value: /workspace/workspace/examples/microservice
30        resources:
31          inputs:
32            - name: workspace
33              resource: source-repo
34          outputs:
35            - name: builtImage
36              resource: web-image
37      - name: build-skaffold-app
38        runAfter: [skaffold-unit-tests]
39        taskRef:
40          name: build-push
41        params:
42          - name: pathToDockerFile
43            value: Dockerfile
44          - name: pathToContext
45            value: /workspace/workspace/examples/microservice
46        resources:
```

On the right, a preview window shows a diagram of the pipeline workflow:

```
graph TD
    A[skaffold-unit-tests] --> B[build-skaffold-web]
    A --> C[build-skaffold-app]
    B --> D[deploy-web]
    C --> E[deploy-app]
```

The status bar at the bottom indicates the current file is 'sample_test.yaml' in the 'yaml' language, with 13 lines and 9 columns. The system clock shows 'Thu 2 Apr 18:34'.

Roadmap



What does it mean for Jenkins?



Jenkins

OpenShift continues to ship and support Jenkins images and plugins to simplify the Jenkins experience

Jenkins build strategy

Direct use of `Jenkinsfiles` on Jenkins is recommended, because the Pipeline build strategy is in maintenance mode.

```
kind: "BuildConfig"
spec:
  strategy:
    jenkinsPipelineStrategy
```

What does it mean for OpenShift Builds?

OpenShift Builds

a feature-rich mechanism for building images with S2I, Dockerfiles or custom builds

Tekton Tasks

a portable mechanism for building images on Kubernetes with S2I, Dockerfile, Kaniko, JIB, buildpacks and more

Features
Robustness



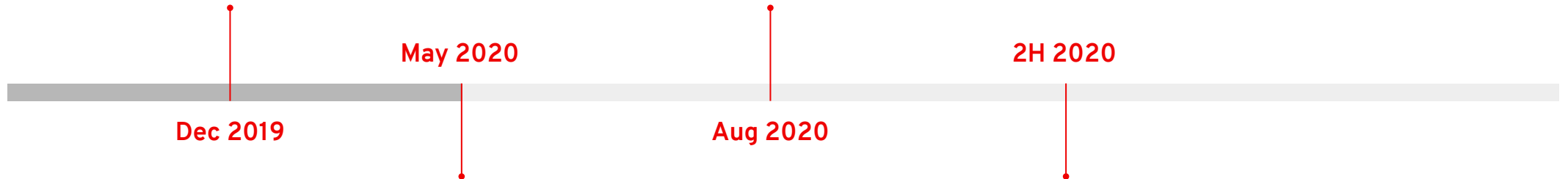
Portability
Extensibility

Dev Preview 4

Webhooks
 Default tasks
 RBAC UX improvements (cont.)
 Tekton CLI (more commands)
 VSCode Plugin (cont.)
 Console Dev (view, graph, logs)

OpenShift Pipelines 1.1 (TP2)

Improve concurrent writes to workspace
 Auto-create PVC for workspaces
 Emit k8s events for task lifecycle
 Knative pipeline templates
 Tetkon Hub & Catalog launch
 Tekton CLI - hub commands
 Console - workspace support
 Console - trigger support
 VS Code - start pipeline wizard
 VS Code - hub integration

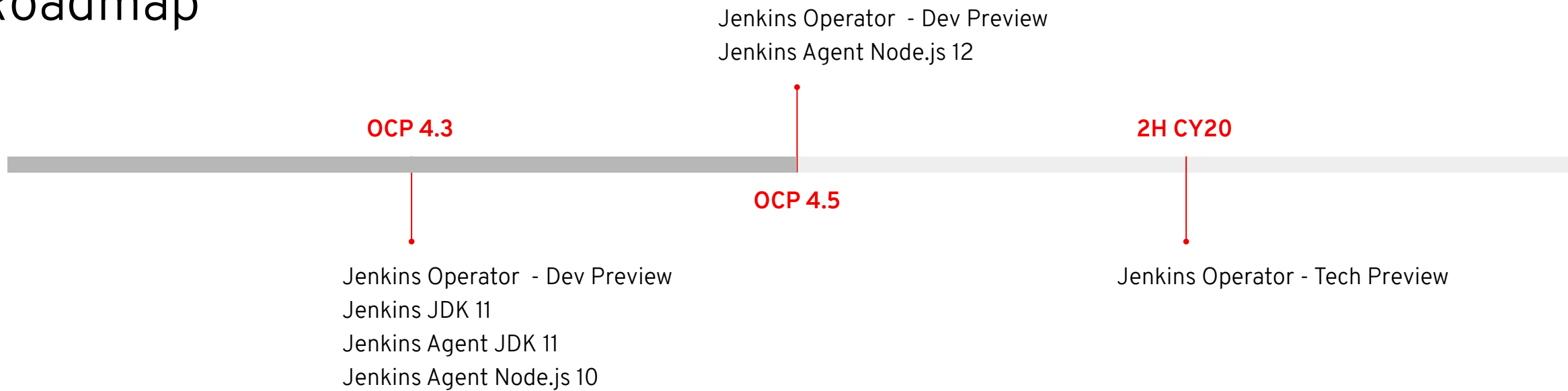
**OpenShift Pipelines 1.0 (TP1)**

Webhook event filtering
 More tasks in the catalog
 Console - generate pipelines
 Console - task snippets
 Console - pipeline builder
 Workspace and sharing artifacts
 Tekton CLI (more commands)
 VSCode Plugin (cont.)

OpenShift Pipelines GA


Config-as-code
 Disconnected install
 Proxy support
 Additional tasks
 Image stream support
 Support for finally clauses
 Enhanced pipeline resources
 Console - catalog integration

Jenkins Roadmap




Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 twitter.com/RedHat