

# CI/CD on OpenShift

Traditional and Cloud-Native Pipelines

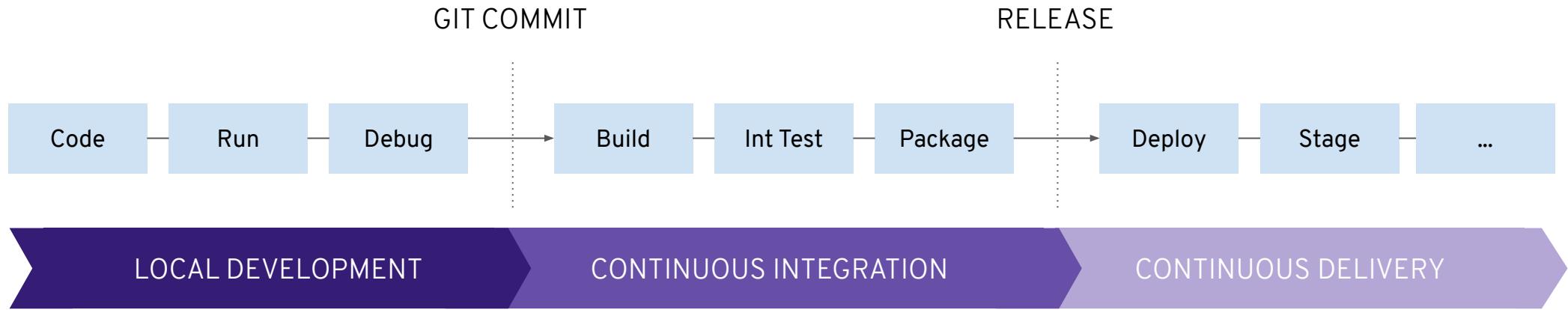
Wanja Pernath

EMEA Partner Enablement Manager, OpenShift & MW

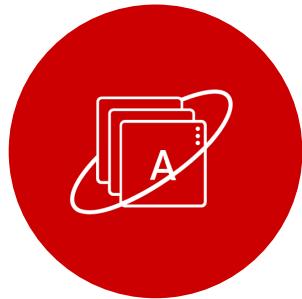
# What is CI/CD?



# Continuous Integration and Continuous Delivery (CI/CD)

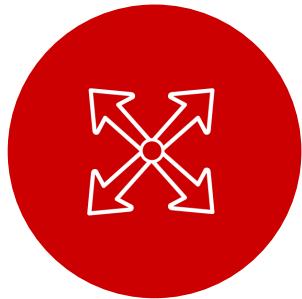


# What is Cloud-Native CI/CD?



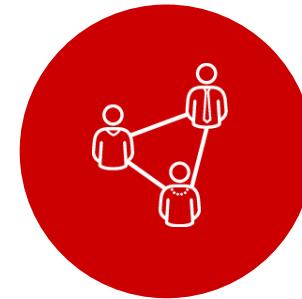
## Containers

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

Designed for Virtual Machines

Require IT Ops for CI engine maintenance

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

Pipelines fully isolated from each other

Everything lifecycled as container images

Native Kubernetes resources

Platform manages persistence

Configured via Kubernetes ConfigMaps

# Why Cloud-Native CI/CD?

## Traditional CI/CD

Designed for Virtual Machines

Require IT Ops for CI engine maintenance



Jenkins  
Plug-ins shared across CI engine  
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



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



# 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

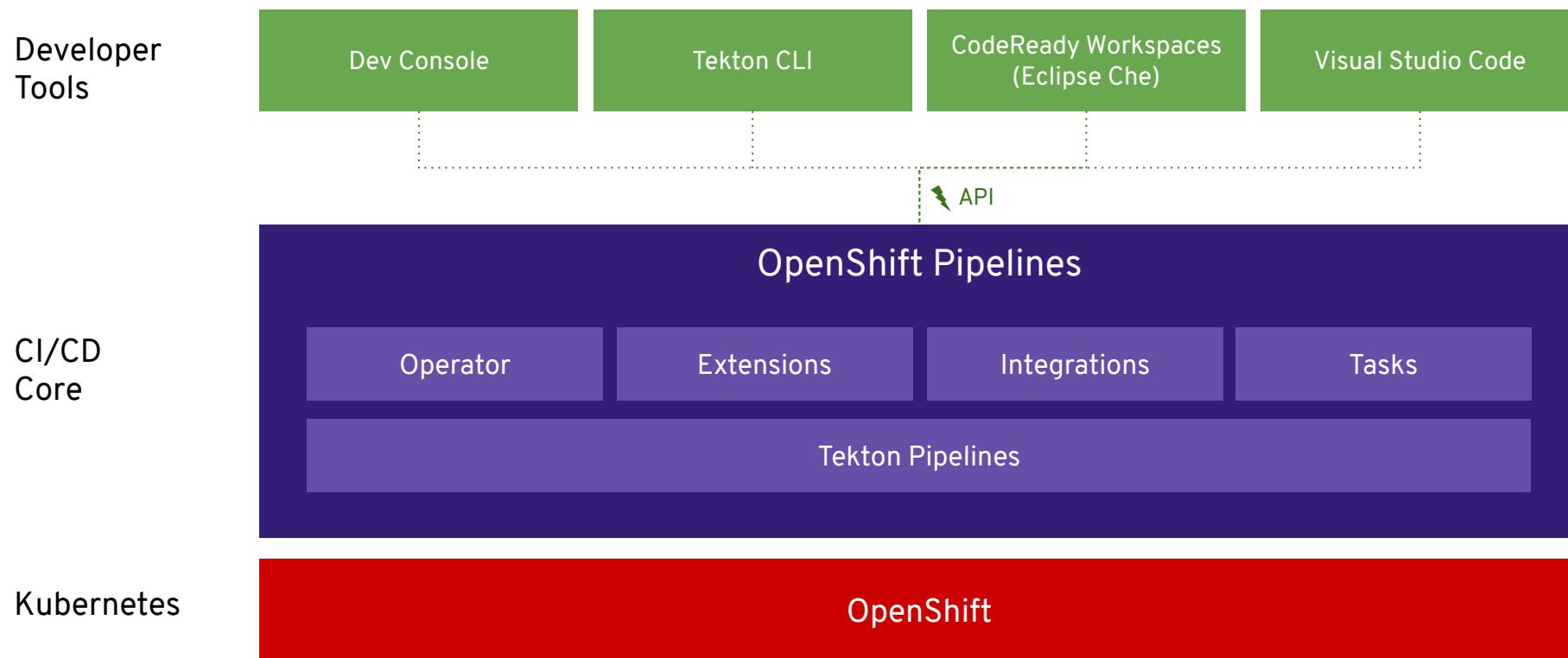


An open-source project for providing a set of shared and standard components for building Kubernetes-style CI/CD systems



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

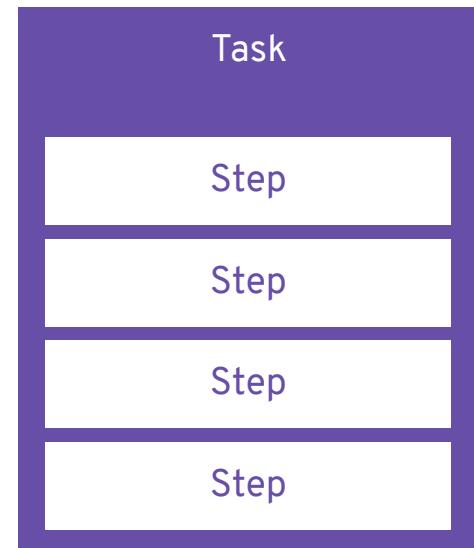
- Run command or script in a container
- Kubernetes container spec
  - Env vars
  - Volumes
  - Config maps
  - Secrets

```
- 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



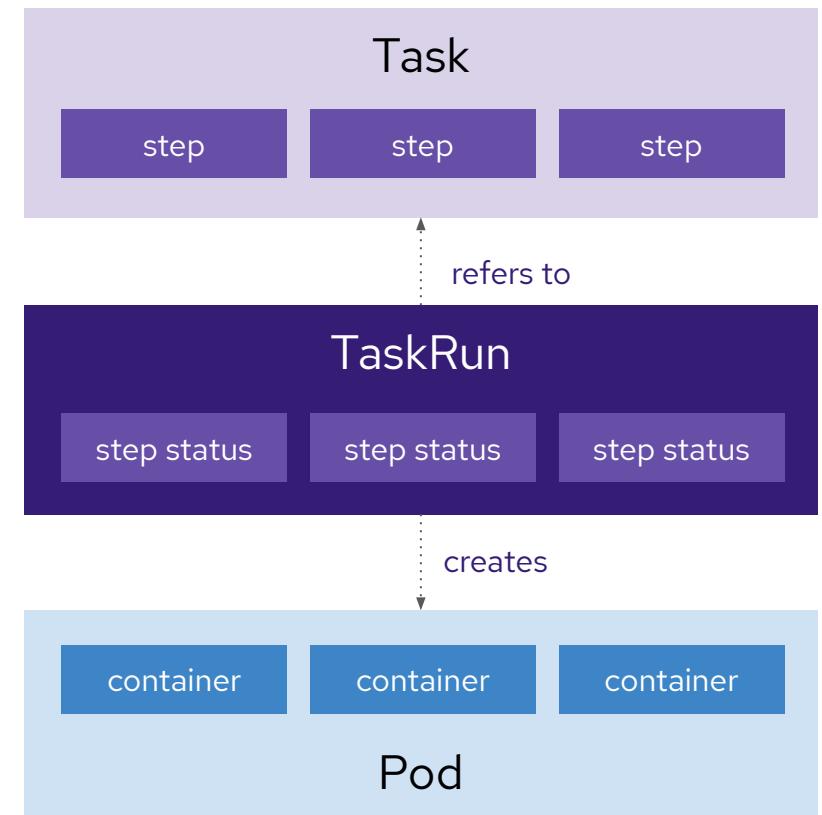
**Example Tasks:** Maven Install, AWS CLI, Kubectl Deploy, Security Scan, etc

# 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) ]
```

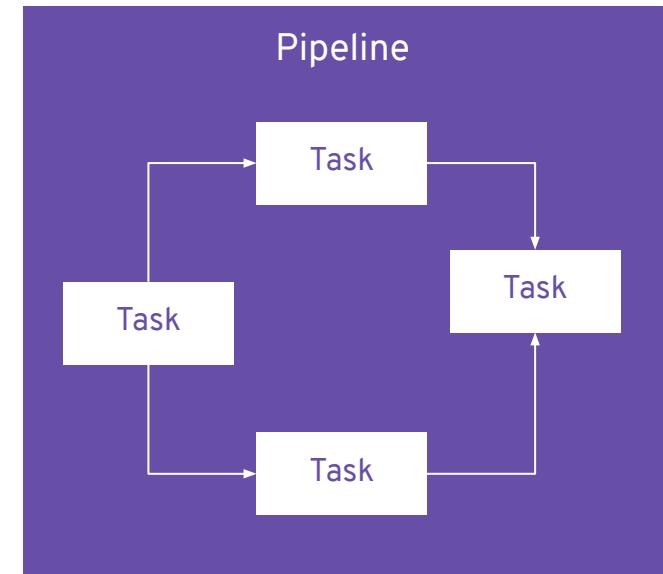
# TaskRun

- 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



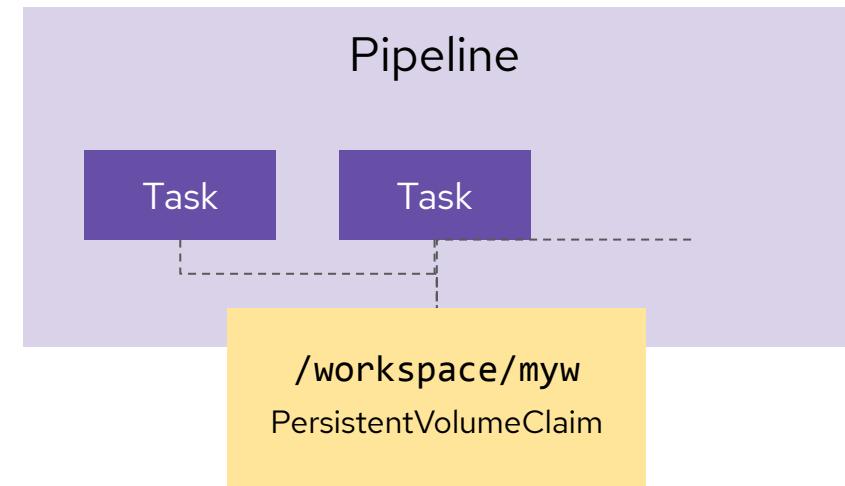
# Sharing Data Between Tasks

## 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

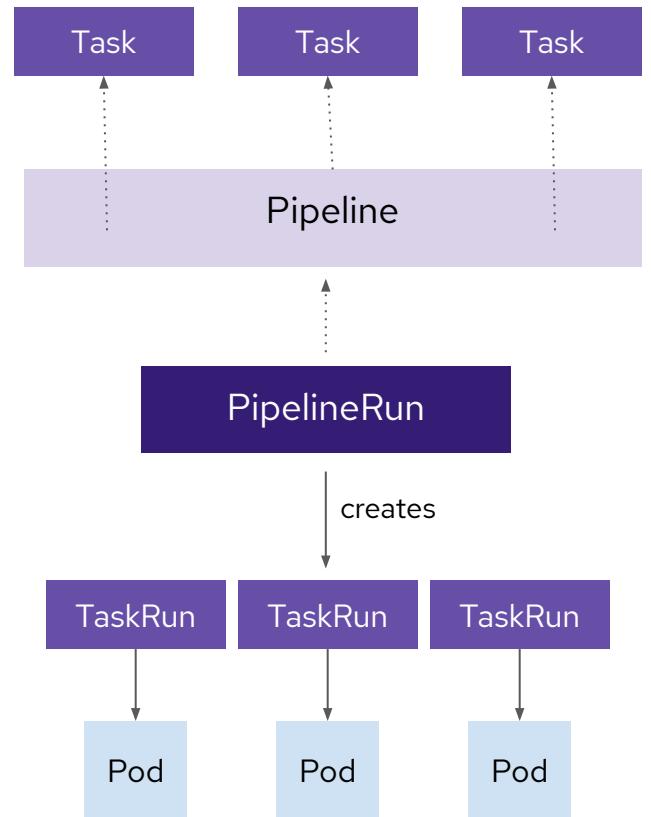
- Defines a single check
- Used in conditional tasks
- Parameterized
- Command run in container
  - True: exit code - 0
  - False: non-zero exit code

```
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 }
```

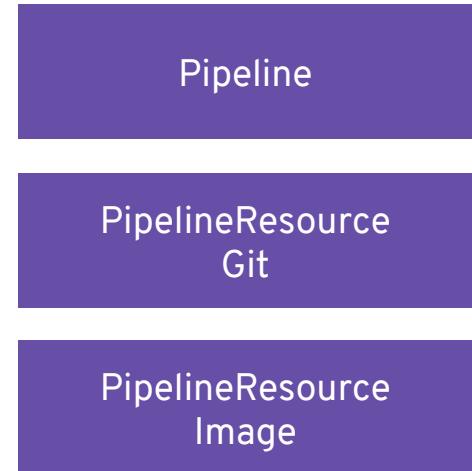
# PipelineRun

- 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



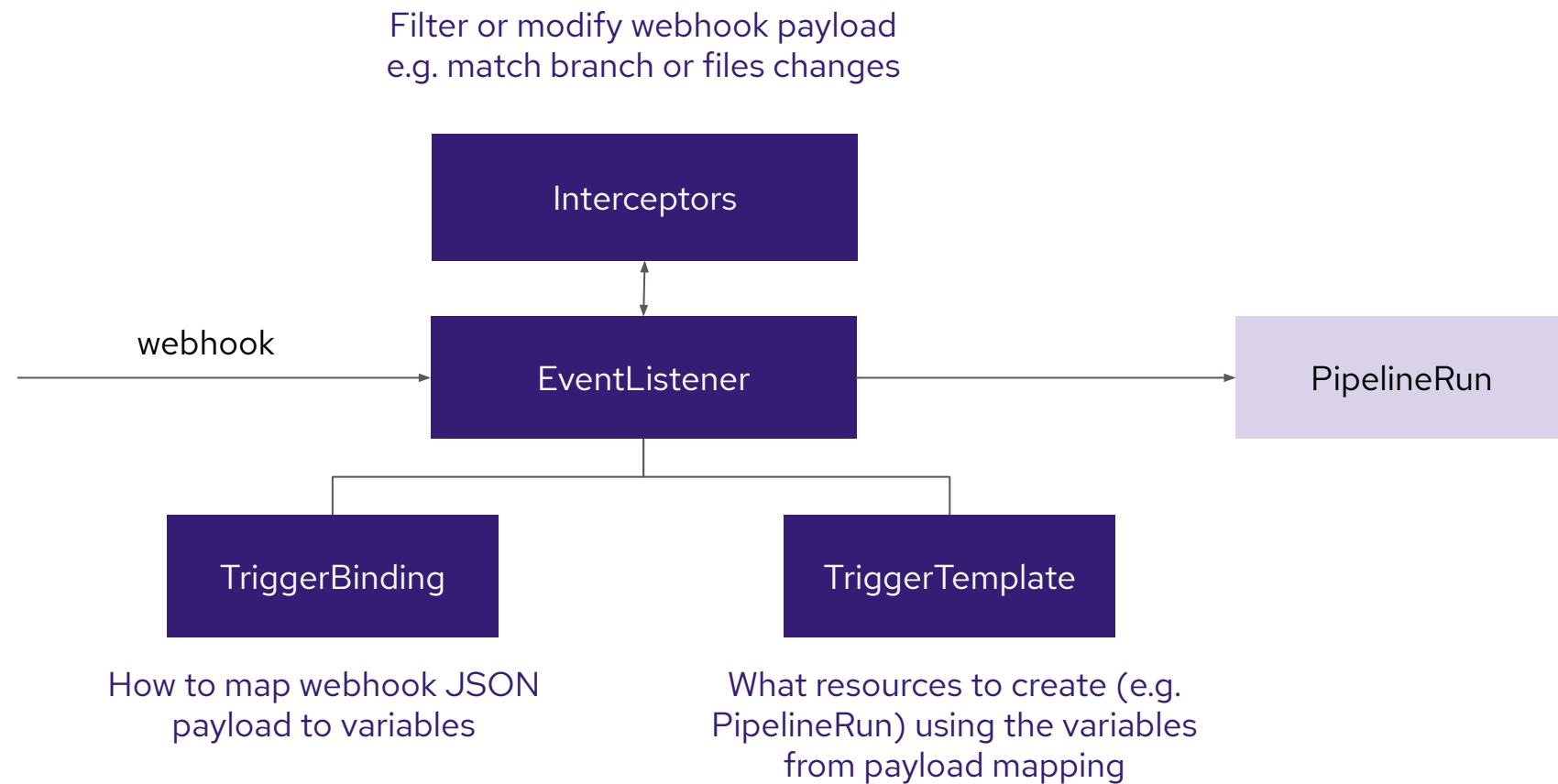
# PipelineResource

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



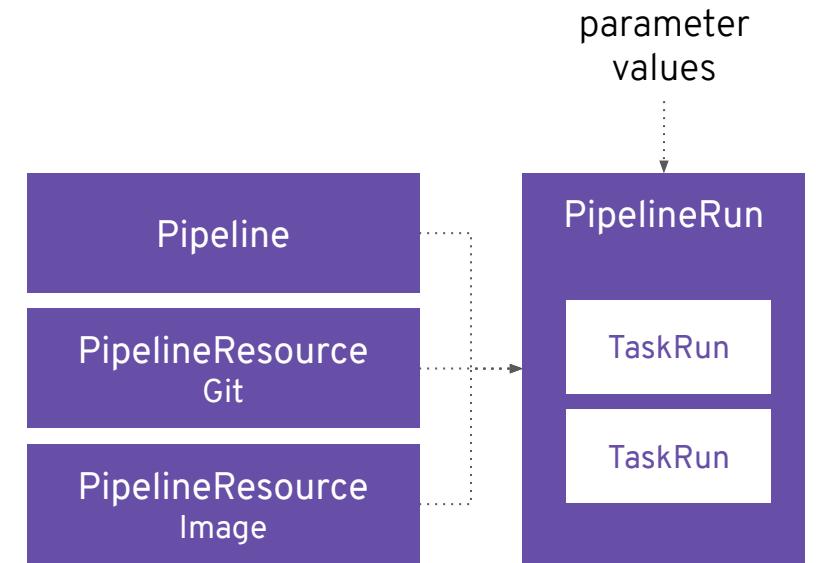
# Triggers

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



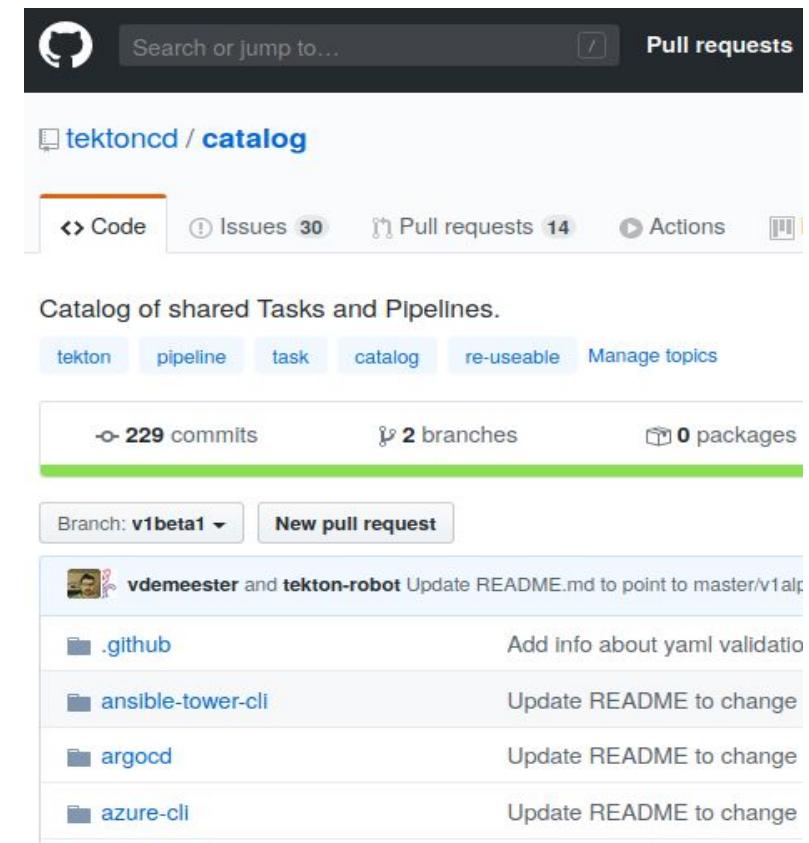
# TaskRun and PipelineRun

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



# Task Catalog

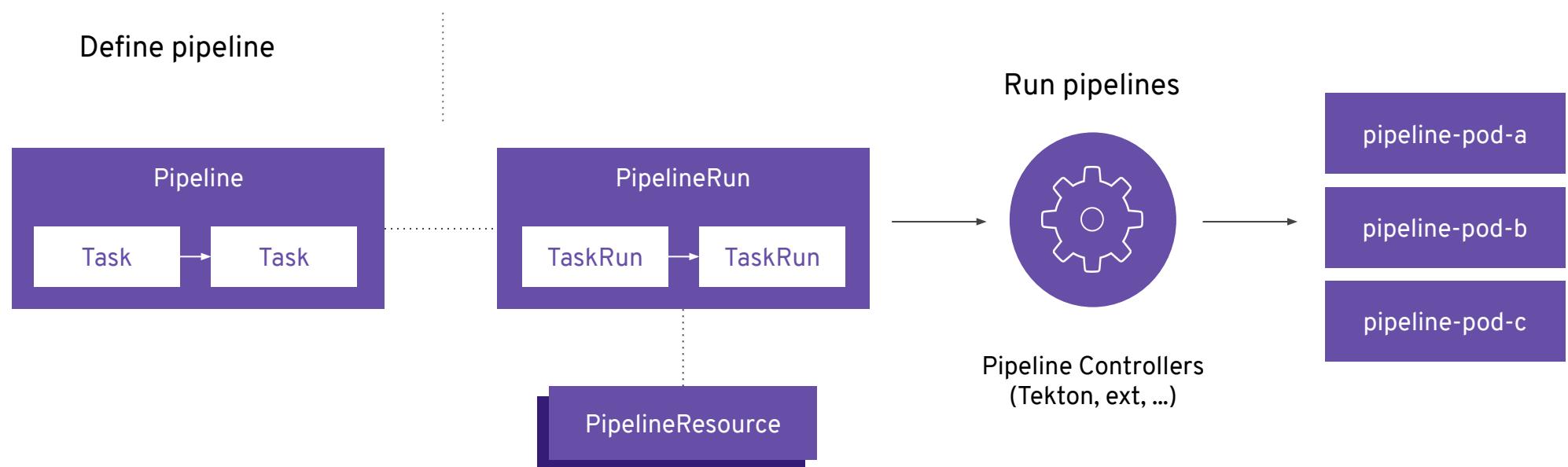
- Catalog of reusable Tasks
  - Image build: buildah, kaniko, jib, buildpacks, etc
  - Source-to-Image: Java, Python, Go, Ruby, etc
  - Language specific: maven, gradle, go, ...
  - More to come soon
- Import and compose pipelines
- Available catalogs
  -  [tektoncd/catalog](#)
  -  [openshift/pipelines-catalog](#)



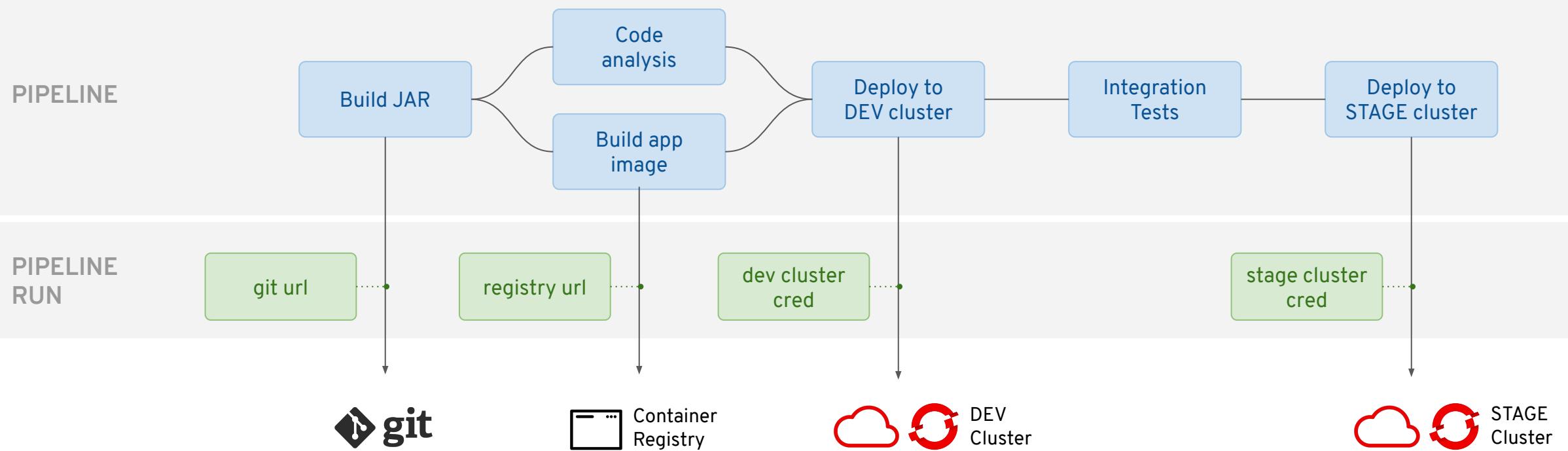
The screenshot shows a GitHub repository page for `tektoncd/catalog`. The header includes a search bar, a pull requests button, and navigation links for Code, Issues (30), Pull requests (14), Actions, and Catalog. Below the header, a banner indicates 229 commits, 2 branches, and 0 packages. A dropdown menu shows the branch is set to `v1beta1`, and there is a "New pull request" button. The main content area lists several tasks:

-  **vdemeester** and **tekton-robot** Update README.md to point to master/v1alpha1
-  **.github** Add info about yaml validation
-  **ansible-tower-cli** Update README to change
-  **argocd** Update README to change
-  **azure-cli** Update README to change

# OpenShift Pipelines Architecture



# Tekton Pipeline Example



# Install OpenShift Pipelines Operator

The screenshot shows the OpenShift OperatorHub interface. On the left, a sidebar menu is open under the 'Operators' section, specifically the 'OperatorHub' tab. The main content area displays the 'OpenShift Pipelines Operator' page. The operator icon is a red circle with white dots forming a hexagon pattern. The title is 'OpenShift Pipelines Operator' with a version of '0.10.7 provided by Red Hat'. A large blue 'Uninstall' button is visible. To the right, detailed information is provided:

- Operator Version:** 0.10.7
- Capability Level:** Basic Install (selected), 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

A light blue box highlights the 'Installed Operator' status message: 'This Operator has been installed on the cluster. [View it here.](#)' Below this, a description of OpenShift Pipelines is given: 'OpenShift Pipelines is a cloud-native continuous integration and delivery (CI/CD) solution for building pipelines using Tekton. Tekton is a flexible Kubernetes-native open-source CI/CD framework which enables automating deployments across multiple platforms (Kubernetes, serverless, VMs, etc) by abstracting away the underlying details.'

**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

Project: Project01 ▾

Pipelines > Pipeline Run Details

PR pipelinerun01a Running

Tech Preview Actions ▾

Overview YAML Logs

Pipeline Run Overview



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  
2 Aug 9:40:00 pm

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has a 'Developer' dropdown, a '+ Add' button, 'Topology', 'Builds', and a selected 'Pipelines' tab. Below 'Pipelines' is an 'Advanced' section with a dropdown arrow. The main area shows 'Project: Project01'. Under 'Pipelines', it lists 'Pipeline Run Details' for 'pipelinerun01a' (status: Running). The pipeline steps are: code compile (green checkmark), compile & test (green checkmark), unit test (green checkmark), security check (green checkmark), and image build (blue link). The 'Logs' tab is selected, showing log output for the image build step. The logs show Plack::Sandbox::\_2fopt\_2fapp\_2droot\_2fsrc\_2fbin\_2fapp\_2epsg:54 core @2018-08-23 18:28:53> looking for get /health in extlib/lib/perl5/Dancer2/Core/App.pm l. 36, Entering hook core.error.init in (eval 306) l. 1, Entering hook core.error.before in (eval 306) l. 1, core @2018-08-23 18:28:53>, and so on.

Project: Project01

Pipelines > Pipeline Run Details

PR pipelinerun01a Running

Overview YAML Logs

Download | Expand

code compile

compile & test

unit test

security check

[image build](#)

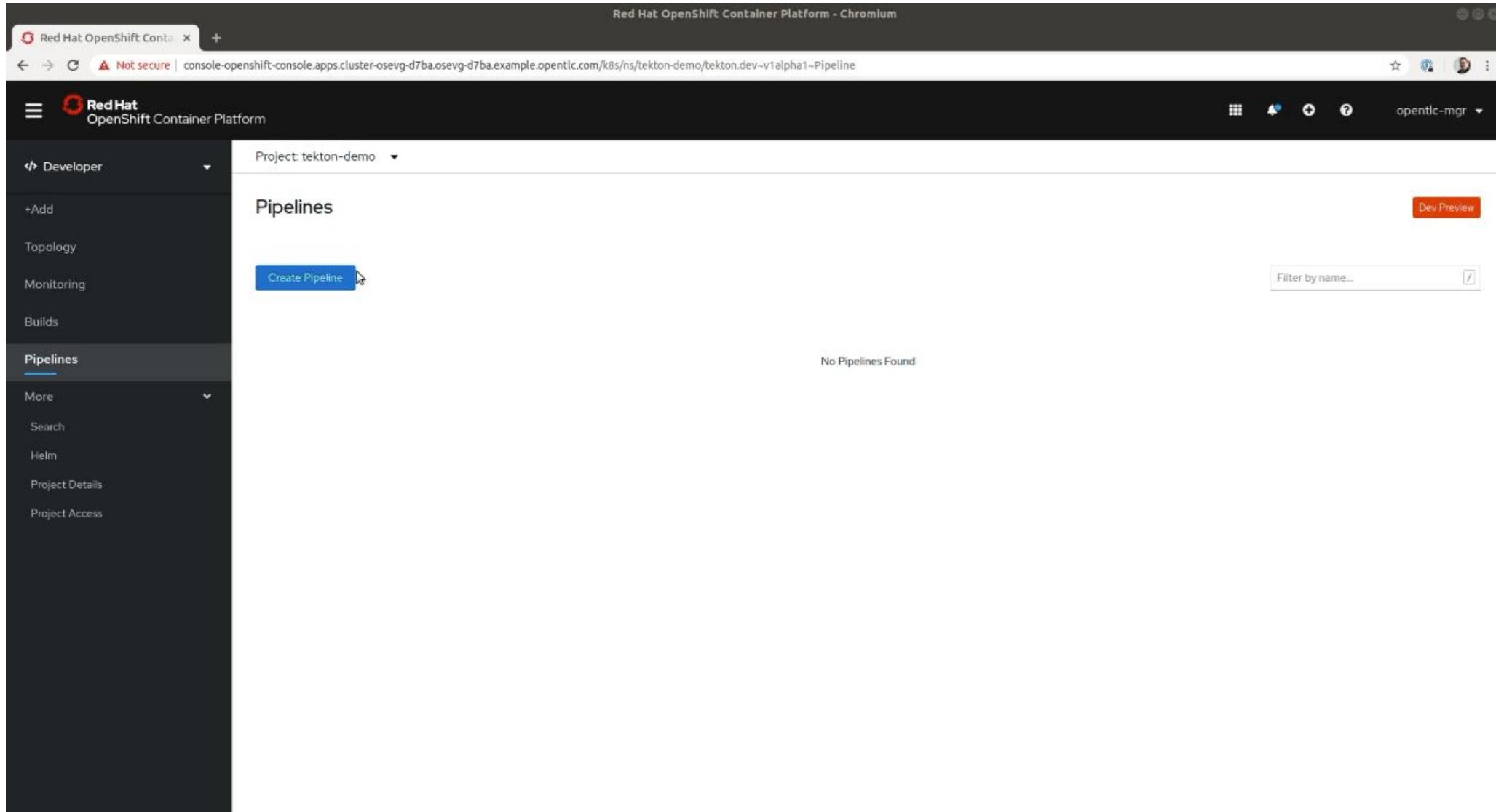
image build

```
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsg: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_2epsg: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_2epsg: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_2epsg:54] core @2018-08-23 18:28:53>
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsg: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_2epsg: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_2epsg: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_2epsg:54] core @2018-08-23 18:28:53>
```

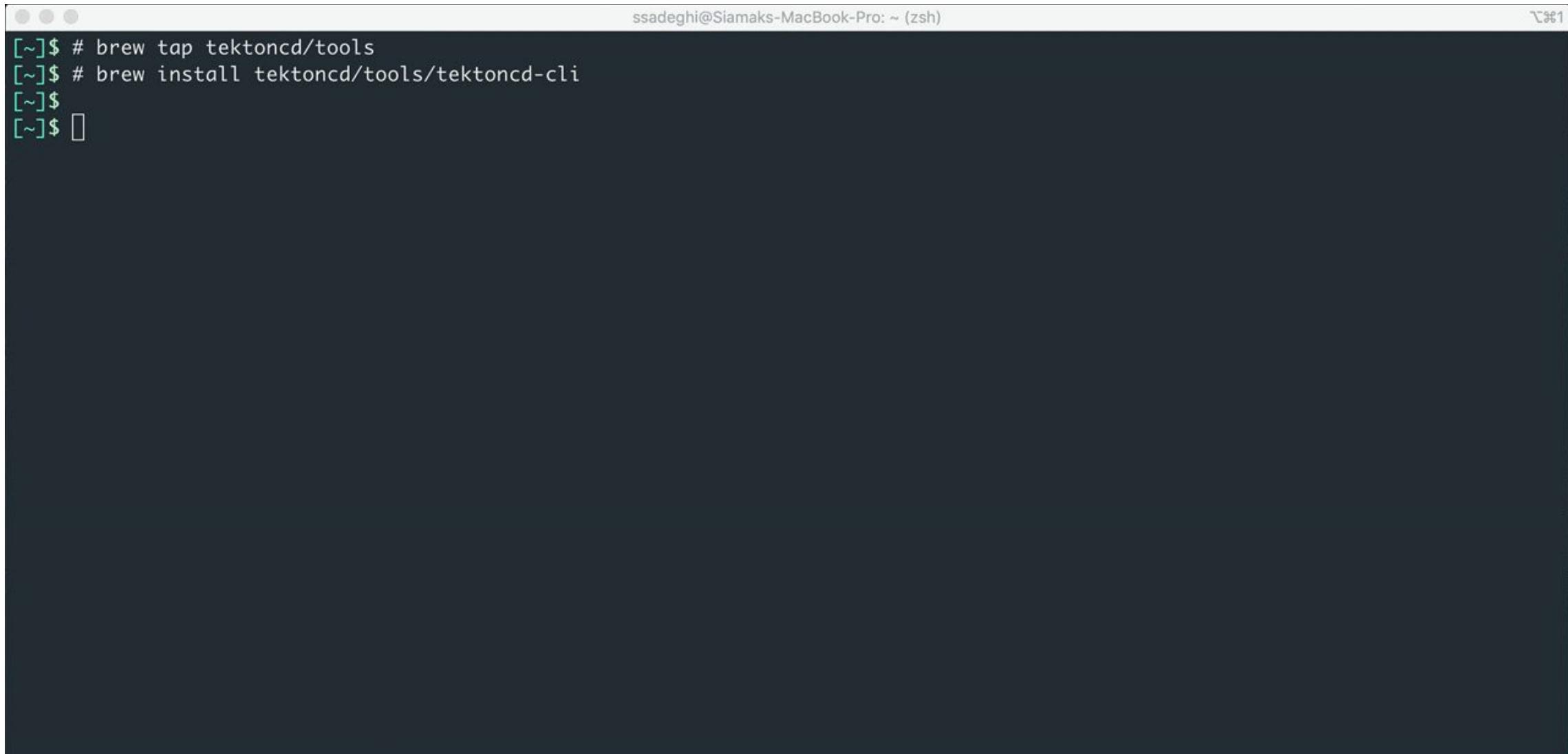
The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo, the text "OpenShift Container Platform", and a user dropdown for "siamak". The left sidebar, titled "Developer", has sections for "Topology", "Builds", "Pipelines", and "Advanced". The "Advanced" section is currently selected and expanded. The main content area is titled "Add" and displays a message "No workloads found" with a sub-instruction: "To add content to your project, create an application, component or service using one of these options." Below this, there are six cards arranged in a 2x3 grid:

| Icon | Name            | Description   |
|------|-----------------|---|
|      | 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 |

# Pipeline Builder to compose Pipelines from Task library



# Manage Pipelines with Tekton CLI



A terminal window with a dark background and light-colored text. The window title bar shows "ssadeghi@Siamaks-MacBook-Pro: ~ (zsh)". The terminal prompt is "[~]\$". The user has run the following commands:

```
[~]$ # brew tap tektoncd/tools
[~]$ # brew install tektoncd/tools/tektoncd-cli
[~]$
[~]$ 
```

# Tekton Pipelines VS Code Extension

The screenshot shows the Tekton Pipelines extension for VS Code. On the left is a sidebar with icons for Pipelines, Tasks, ClusterTasks, PipelineResources, TriggerTemplates, TriggerBinding, EventListener, and Conditions. The main area has three tabs: 'sample\_test.yaml' (active), 'Preview sample\_test.yaml', and 'Logs'. The 'sample\_test.yaml' tab displays the YAML code for a Pipeline named 'demo-pipeline'. The 'Preview' tab shows a visual representation of the pipeline's execution flow:

```
graph TD; skaffold-unit-tests --> build-skaffold-web; skaffold-unit-tests --> build-skaffold-app; build-skaffold-web --> deploy-web; build-skaffold-app --> deploy-app;
```

The pipeline code includes resources like source-repo, web-image, and app-image, and tasks such as skaffold-unit-tests, build-skaffold-web, build-skaffold-app, build-push, and pathToDockerFile.

# 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

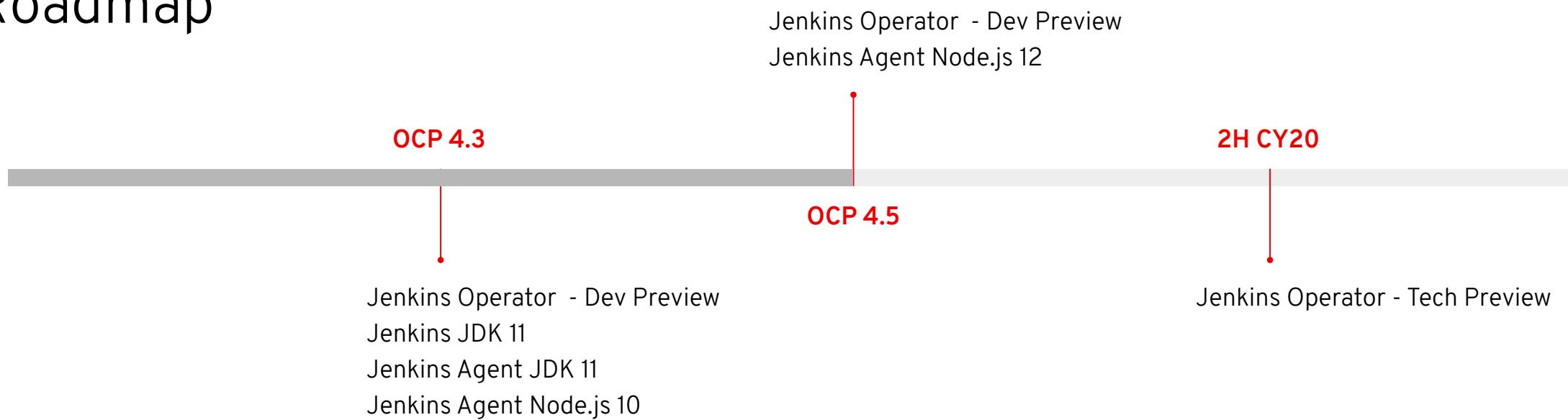
a portable mechanism for building images on Kubernetes with S2I, Dockerfile, Kaniko, JIB, buildpacks and more

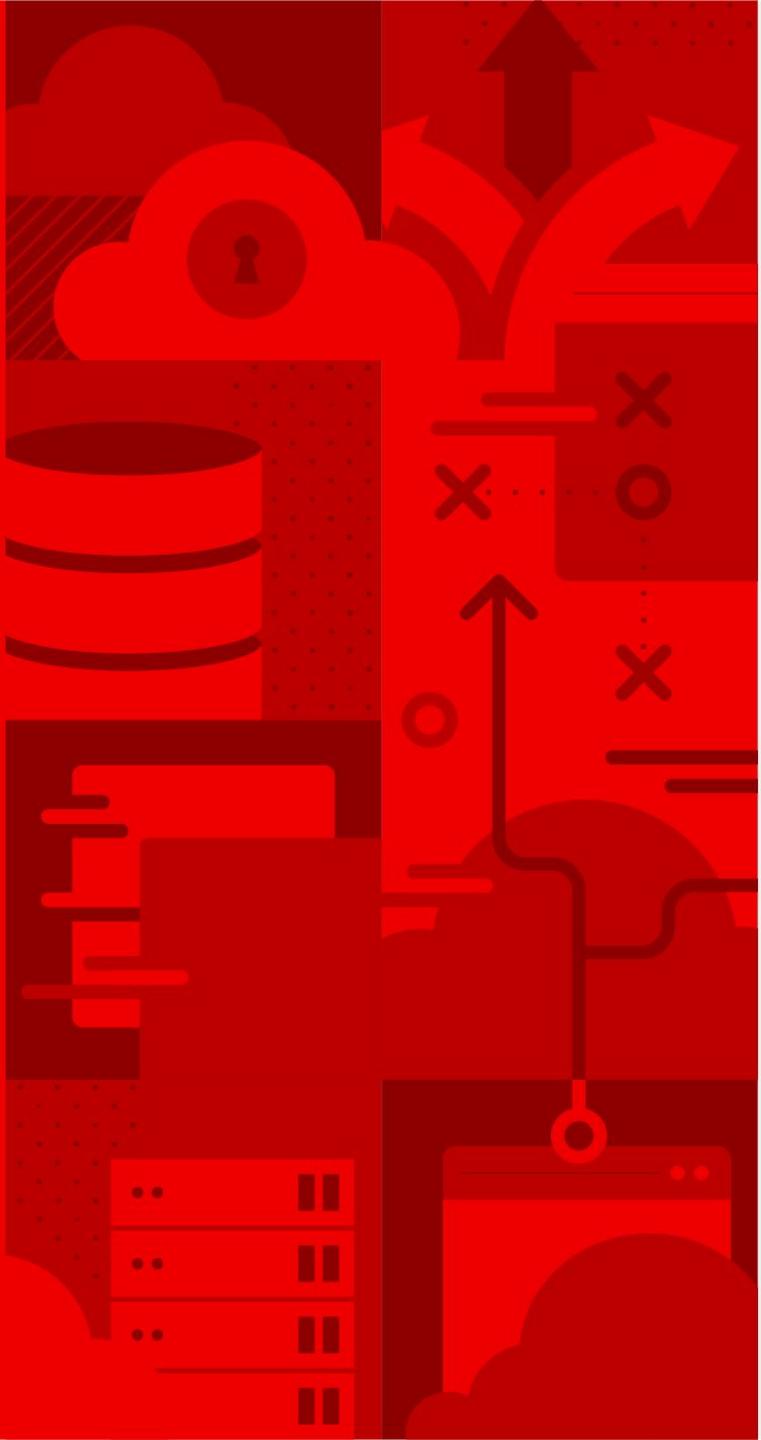
Features  
Robustness

Portability  
Extensibility



# 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)



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



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



[twitter.com/RedHat](https://twitter.com/RedHat)