

# Tekton in Action with Red Hat OpenShift Pipelines

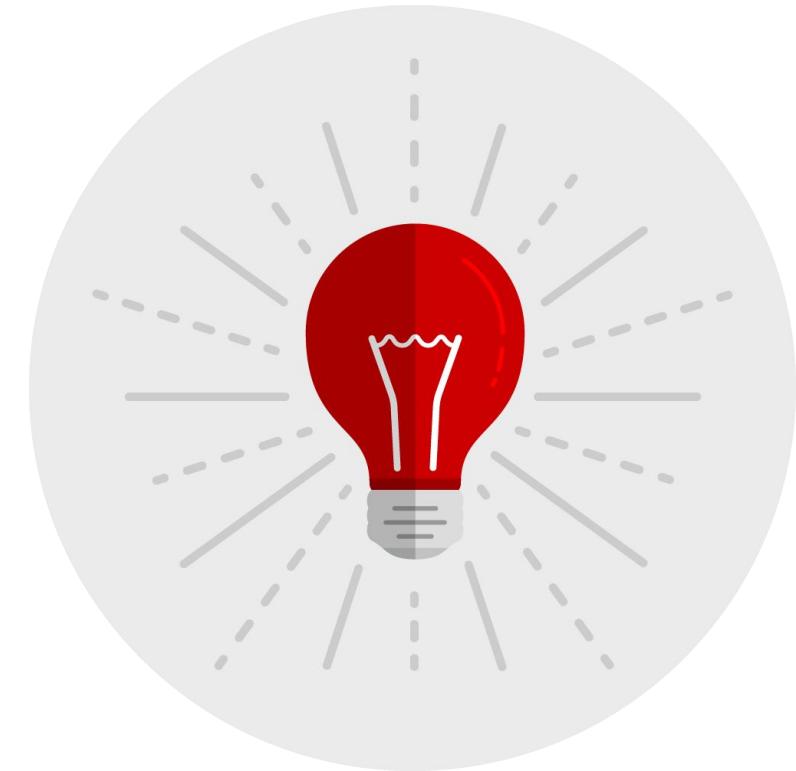
Natale Vinto

Developer Advocate

@natalevinto

# AGENDA

- OpenShift
- What is CI/CD?
- Cloud Native CI/CD
- OpenShift Pipelines
- Tekton components
- Tekton in action



DevOps is the key to meet the insatiable  
demand for delivering quality applications  
rapidly

# OpenShift

## A Comprehensive DevOps Platform for Hybrid Cloud

Build container images  
from source code using  
Kubernetes tools

Traditional and  
Kubernetes-native  
CI/CD

Declarative GitOps for  
multi-cluster  
continuous delivery

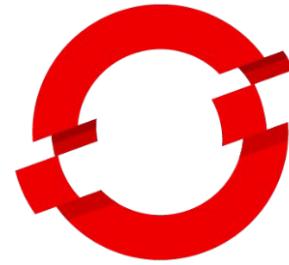


**OpenShift  
Builds**

**OpenShift  
Pipelines**

**OpenShift  
GitOps**

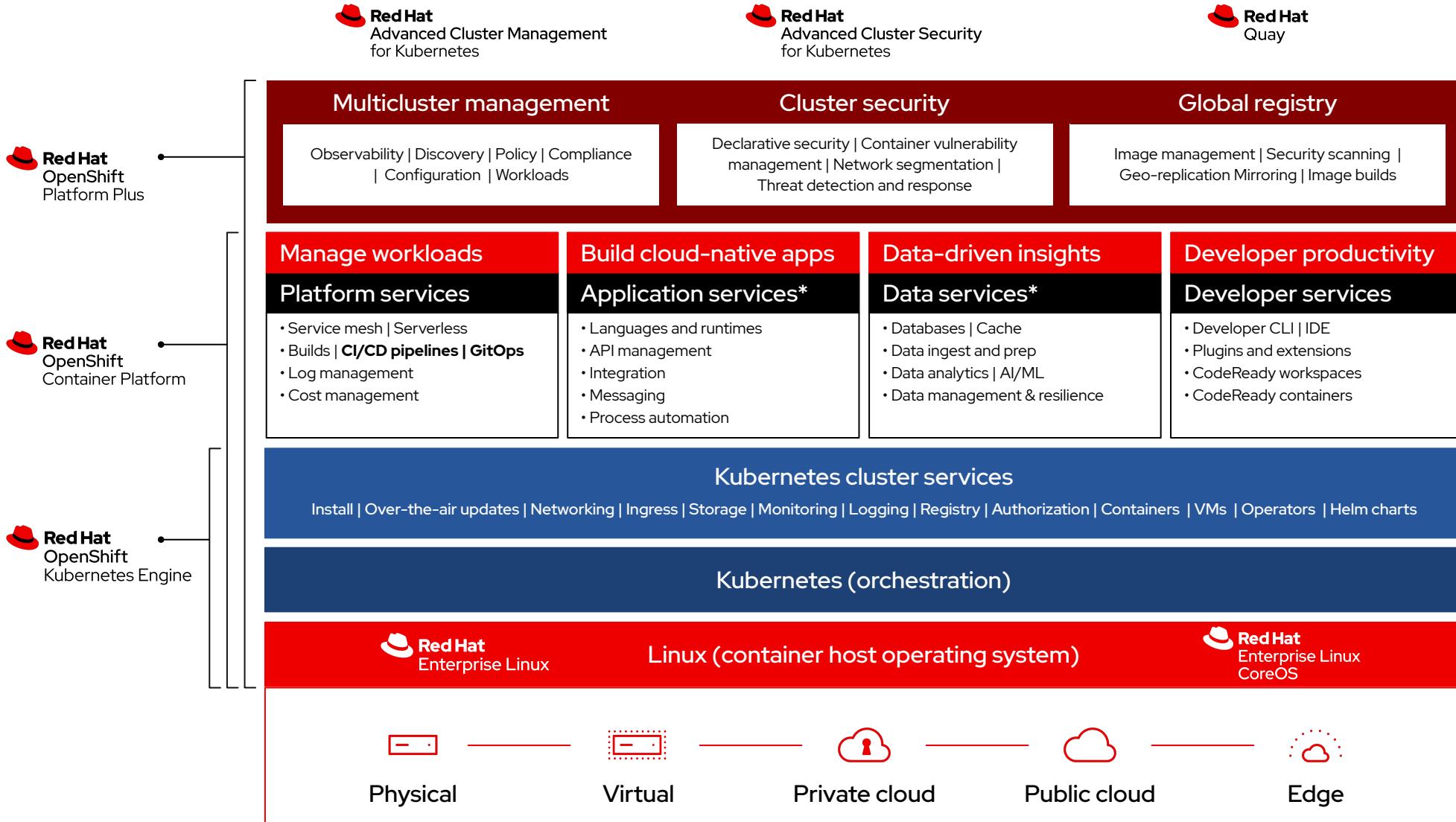
**OpenShift**



# Red Hat OpenShift

A secure and enterprise-grade container application platform based on **Kubernetes** for traditional and cloud-native applications

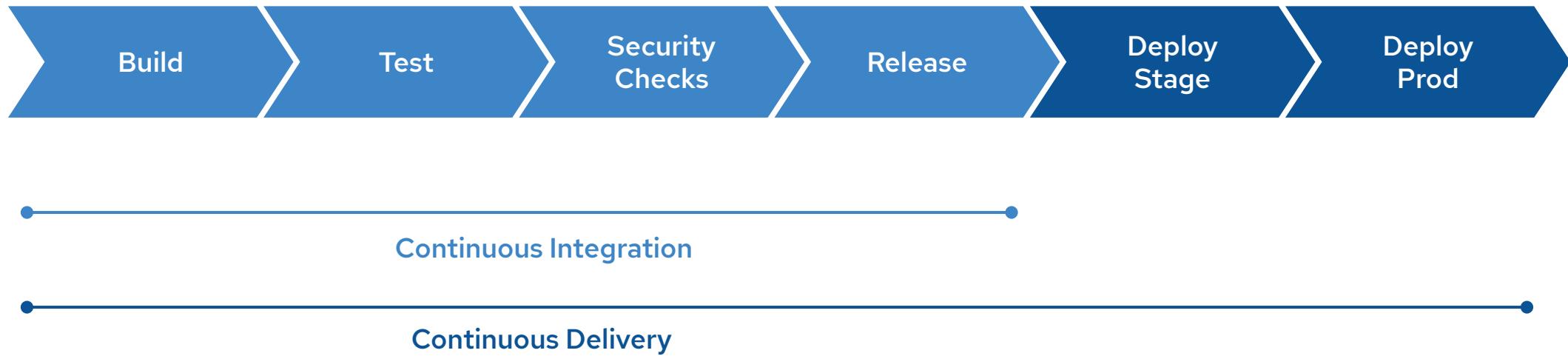
# OpenShift Platform Plus



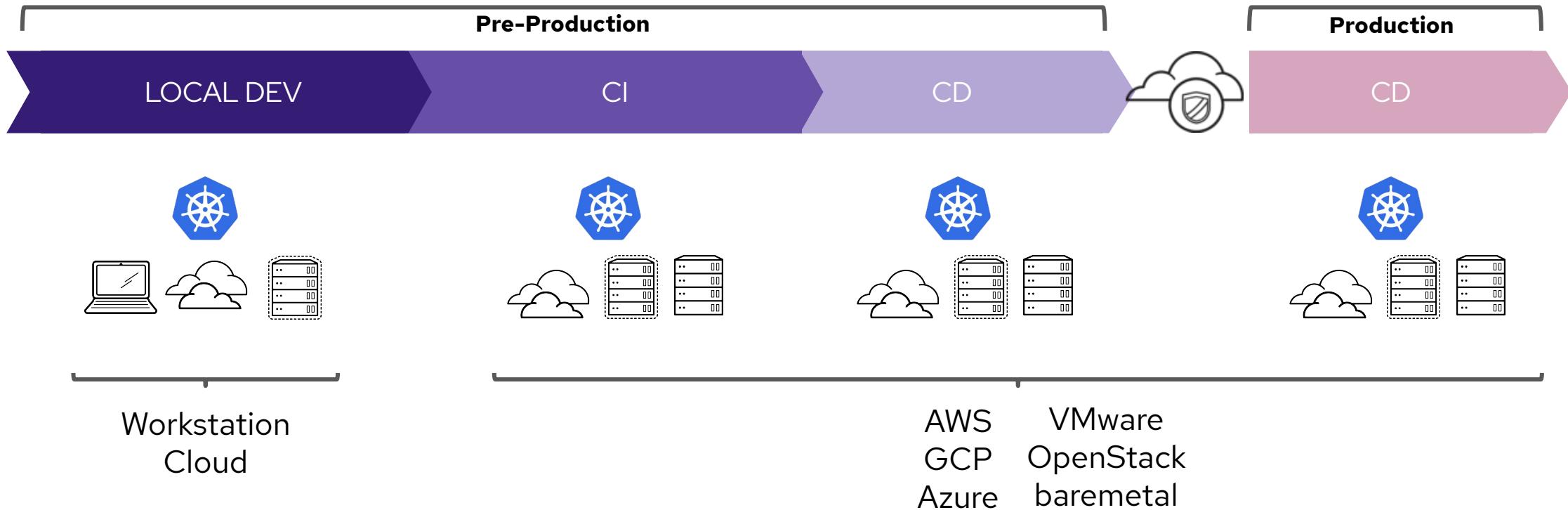
# What is CI/CD?



# Continuous Integration & Continuous Delivery



# Fact: Kubernetes is the target platform



# One Continuous Delivery

Multiple Clouds

Multiple Platforms

DEVELOPMENT

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY



Workstation



Kubernetes



Kubernetes



Kubernetes

Azure

AWS

GCP

VMware

OpenStack

baremetal

# Cloud Native CI/CD

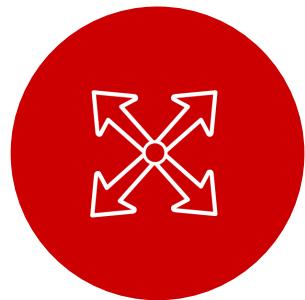


# What is Cloud Native CI/CD?



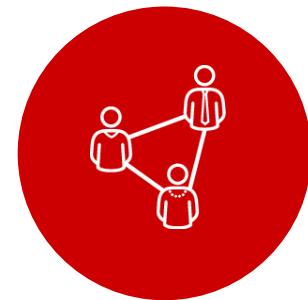
## Containers

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

Requires 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

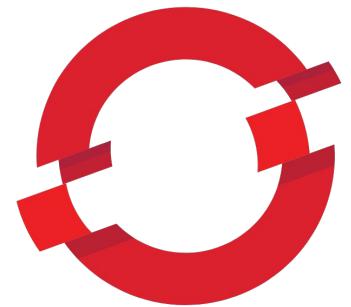


An open-source project for providing a set of shared and standard components for building Kubernetes-style CI/CD systems



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





# OpenShift Pipelines



# OpenShift Pipelines



## Built for Kubernetes

Cloud-native pipelines taking advantage of Kubernetes execution and, operational model and concepts



## Scale on-demand

Pipelines run and scale on-demand in isolated containers, with repeatable and predictable outcomes



## Secure pipeline execution

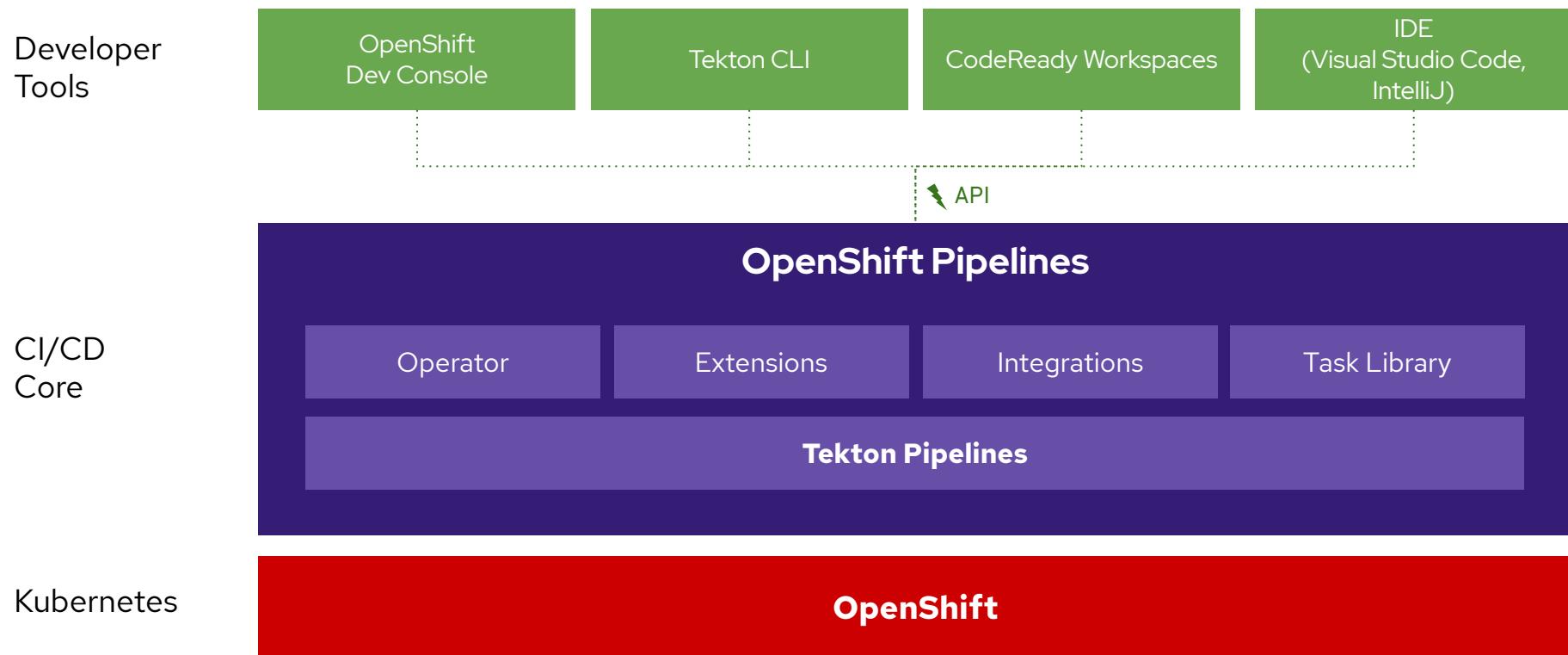
Kubernetes RBAC and security model ensures security consistently across pipelines and workloads



## Flexible and powerful

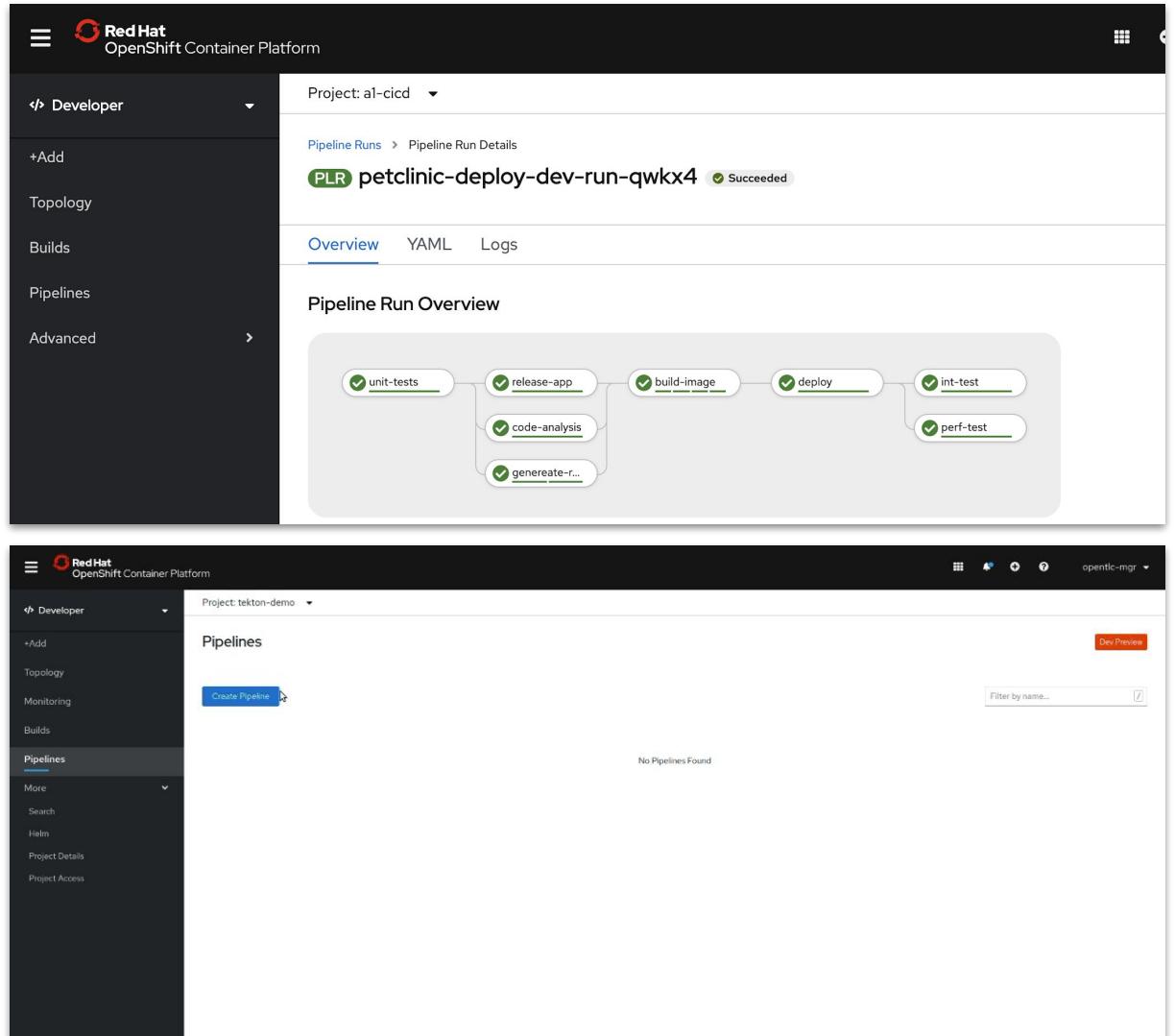
Granular control over pipeline execution details on Kubernetes, to support your exact requirements

# OpenShift Pipelines

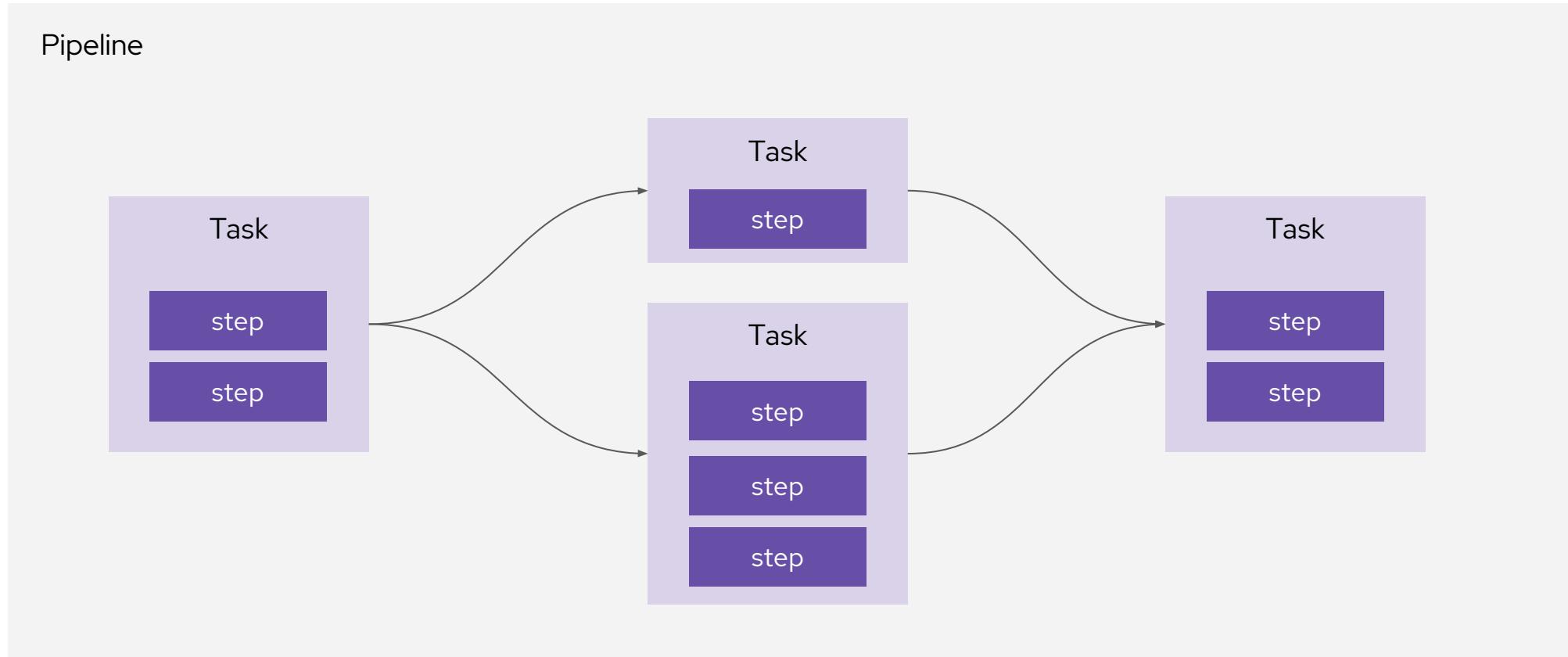


# OpenShift Pipelines

- Based on Tekton Pipelines
- Kubernetes-native declarative CI/CD
- Pipelines run on-demand in isolated containers
- No central server to maintain! No plugin conflicts!
- Task library and integration with Tekton Hub
- Secure pipelines aligned with Kubernetes RBAC
- Visual and IDE-based pipeline authoring
- Pipeline templates when importing apps
- Automated install and upgrades via OperatorHub
- CLI, Web, VS Code and IntelliJ plugins



# Tekton Concepts



# Tekton Concepts: step

- 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  
  
    ...
```

# Tekton Concepts: Task

- Performs a specific task
- List of steps
- Steps run sequentially
- Reusable

```
kind: Task
metadata:
  name: buildah
spec:
  params:
    - name: IMAGE
  steps:
    - name: build
      image: quay.io/buildah/stable:latest
      command: ["buildah"]
      args: ["bud", ".", "-t", "${params.IMAGE}"]
    - name: push
      image: quay.io/buildah/stable:latest
      script: |
        buildah push ${params.IMAGE} docker://${params.IMAGE}
```

# Tekton Hub

## Search, discover and install Tekton Tasks

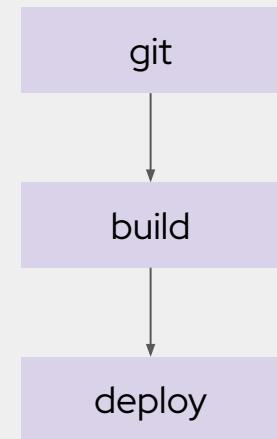
The screenshot shows the Tekton Hub (Beta) interface. At the top, there's a navigation bar with the Tekton Hub logo, a (BETA) badge, and a Login button. Below the header, a banner reads "Welcome to Tekton Hub" and "Discover, search and share reusable Tasks and Pipelines". The main area features a search bar and a "Sort" dropdown set to "Name". A sidebar on the left contains filtering options: "Refine By:", "Kind" (Task, Pipeline), "Support Tier" (Official, Verified, Community), and "Categories" (Build Tools, CLI, Cloud, Deploy, Image Build, Notification, Others, Test Framework). The main content area displays a grid of 12 task cards, each with a icon, name, rating, version, description, and update timestamp. Most cards also show one or more labels at the bottom.

Name	Rating	Description	Last Updated	Labels
Ansible Runner	4.5	Task to run Ansible playbooks using Ansible Runner	Updated 3 weeks ago	cli
ansible tower cli	2.0	Ansible-tower-cli task simplifies starting jobs, workflow jobs, manage users, projects etc.	Updated 3 weeks ago	ansible, cli
argocd	3.0	This task syncs (deploys) an Argo CD application and waits for it to be healthy.	Updated 3 weeks ago	deploy
aws cli	5.0	This task performs operations on Amazon Web Services resources using aws.	Updated 3 weeks ago	cli
Amazon ECR Login	4.0	This task retrieves an `authentication token` using the GetAuthorizationToken API that you can use to authenticate to an...	Updated 3 weeks ago	aws, ecr
azure cli	1.0	This task performs operations on Microsoft Azure resources using az.	Updated 4 months ago	cli
bentoml	0.0	This task allows operations on BentoML services	Updated 3 weeks ago	cli
Python Black	0.0	This task can be used to format Python code	Updated 3 weeks ago	formatter, python

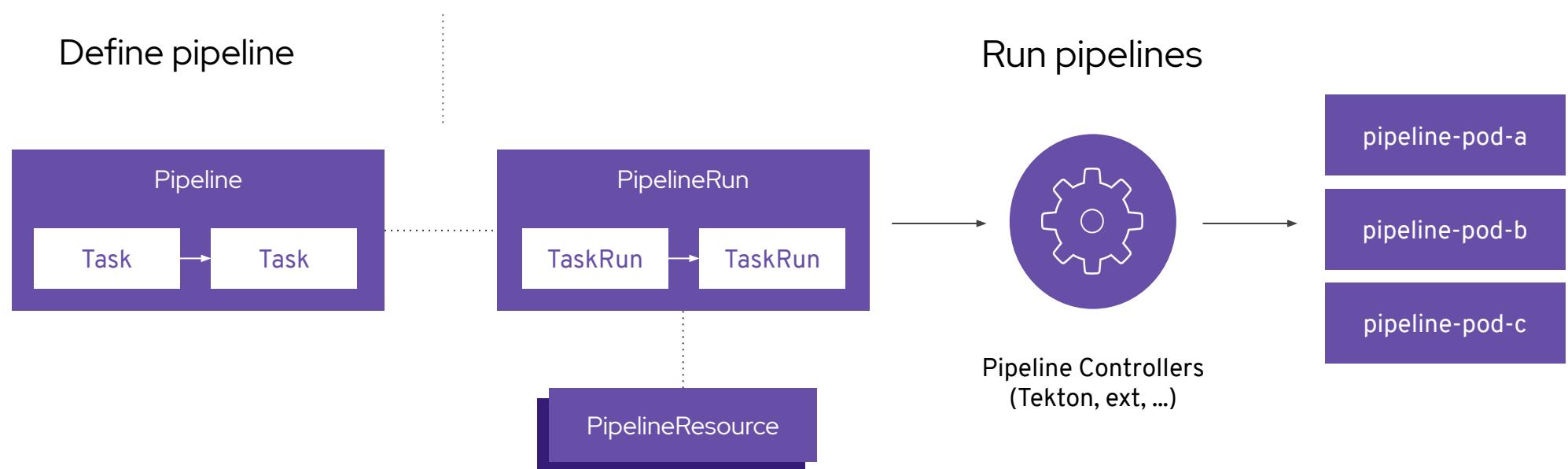
# Tekton Concepts: Pipeline

- A graph of Tasks: concurrent & sequential
- Tasks run on different nodes
- Task execution logic
  - Conditional
  - Retries
- Share data between tasks

```
kind: Pipeline
metadata:
  name: deploy-dev
spec:
  params:
    - name: IMAGE_TAG
  tasks:
    - name: git
      taskRef:
        name: git-clone
      params: [...]
    - name: build
      taskRef:
        name: maven
      params: [...]
      runAfter: ["git"]
    - name: deploy
      taskRef:
        name: knative-deploy
      params: [...]
      runAfter: ["build"]
```



# OpenShift Pipelines Architecture



# Migrate from Jenkins to Tekton



## Jenkins Pipeline

```
pipeline {  
    stages {  
        stage('Git Clone') {  
            steps { ... }  
        }  
        stage('Build App') {  
            steps { ... }  
        }  
        stage('Test') {  
            steps { ... }  
        }  
        stage('Code Analysis') {  
            steps { ... }  
        }  
    }  
}
```

## Tekton Pipeline

```
kind: Pipeline  
spec:  
  tasks:  
    - name: git-clone  
    - name: build-app  
    - name: test  
    - name: code-analysis
```

## Jenkins Pipeline

```
pipeline {  
  
    agent {  
        label 'maven'  
    }  
  
    stages {  
        stage ('Clone') {  
            git url: 'https://github.com/...'  
        }  
        stage ('Build App') {  
            withMaven(maven: 'maven-3') {  
                sh "mvn clean verify"  
            }  
        }  
        ...  
    }  
}
```

## Tekton Pipeline

```
kind: Pipeline  
spec:  
  tasks:  
  
    - name: git-clone  
      taskRef:  
        name: git-clone  
      params:  
        - name: url  
          value: https://github.com/...  
      workspaces:  
        - name: app-workspace  
          workspace: app-source  
  
    - name: build-app  
      taskRef:  
        name: maven  
      params:  
        - name: GOALS  
          value: ["clean", "verify"]  
      runAfter:  
        - git-clone  
      workspaces:  
        - name: app-workspace  
          workspace: app-source
```

# Tekton in action



# Install Pipeline via OperatorHub marketplace

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar is titled "Administrator" and includes sections for Home, Overview, Projects, Search, Explore, Events, Operators (selected), OperatorHub (selected), Installed Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, User Management, and Administration. The main content area is titled "OperatorHub" and displays a search bar with the term "pipelines". A card for the "OpenShift Pipelines Operator" is shown, with its icon, name, version (1.0.1), provider (Red Hat), and a large blue "Install" button highlighted with a red box. To the right of the card, detailed information is provided under sections like "Operator Version", "Capability Level", "Provider Type", "Provider", "Repository", "Container Image", and "Components". The "Features" section lists benefits such as standard CI/CD pipelines, build images with Kubernetes tools, deployment to multiple platforms, and integration with the developer console. The "Installation" section describes how the operator is installed into namespaces. A note at the bottom indicates that if a community version is already subscribed, it should be uninstalled before subscribing to the operator.

<https://console-openshift-console.natale-test-4-5-6-f5541308b177087861a229b886140c95-0000.us-east.containers.appdomain.cloud/operatorhub/subscribe?pkq=openshift-pipelines-operator-rh&cataloq=redhat-operators&cataloqNamespace=openshift-marketplace&targetNamespace...>

# Run Pipelines

Red Hat OpenShift Container Platform

Project: a3-cicd

PLR petclinic-dev-eb123ee Succeeded

Tech preview

Actions

Details YAML Task Runs Logs Events

### Pipeline Run Details

source-clone

code-analysis

dependency...

unit-tests

release-app

build-image

config-clone

tests-clone

deploy-dev

int-test

perf-test

Name	petclinic-dev-eb123ee	Status	Succeeded
Namespace	NS a3-cicd	Pipeline	PL petclinic-deploy-dev
Labels	tekton.dev/pipeline=petclinic-deploy-dev triggers.tekton.dev/listener=webhook	Triggered by:	EL webhook

# Check logs of running pipelines

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has a 'Developer' dropdown, '+ Add' button, 'Topology', 'Builds', and 'Pipelines' section which is selected and highlighted in blue. Below it are 'Advanced' and a chevron icon. The main area shows 'Project: Project01'. Under 'Pipelines', it says 'Pipelines > Pipeline Run Details' and shows a pipeline run named 'PR pipelinerun01a' status 'Running'. There are tabs for 'Overview', 'YAML', and 'Logs', with 'Logs' being active. A red 'Tech Preview' button is visible. On the right, there's a user 'siamak' with a dropdown menu. The log output for the 'image build' step is displayed in a large text area, showing multiple entries of core logs from a Plack::Sandbox environment.

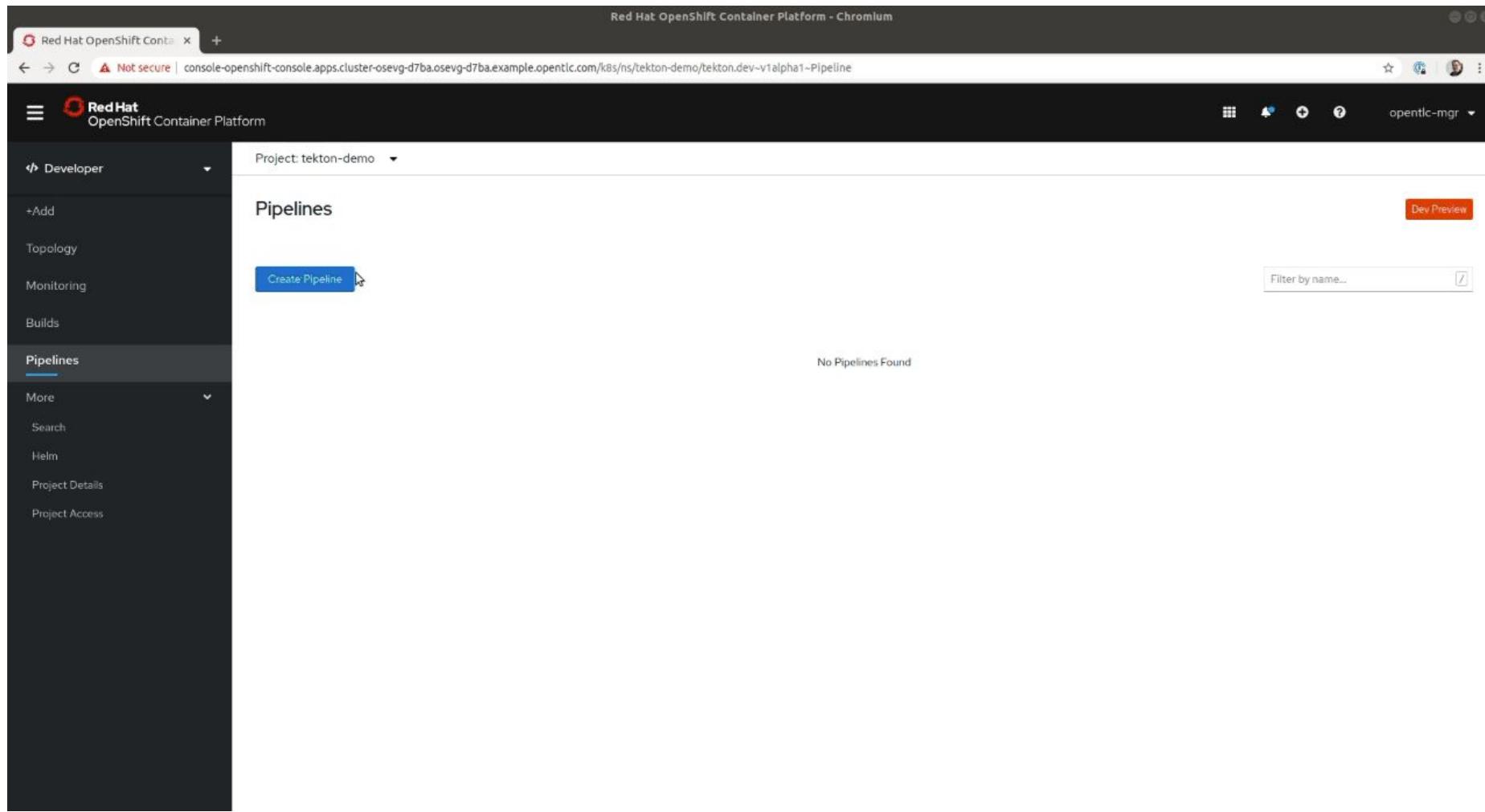
```
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsg:54] core @2018-08-23 18:28:53> looking for get /health in extlib/lib/perl5/Dancer2/Core/App.pm l. 36
[Plack::Sandbox::_2fopt_2fapp_2droot_2fsrc_2fbin_2fapp_2epsg:54] core @2018-08-23 18:28:53> Entering hook core.error.init in (eval 306) 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
[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
```

# Create apps with Pipelines

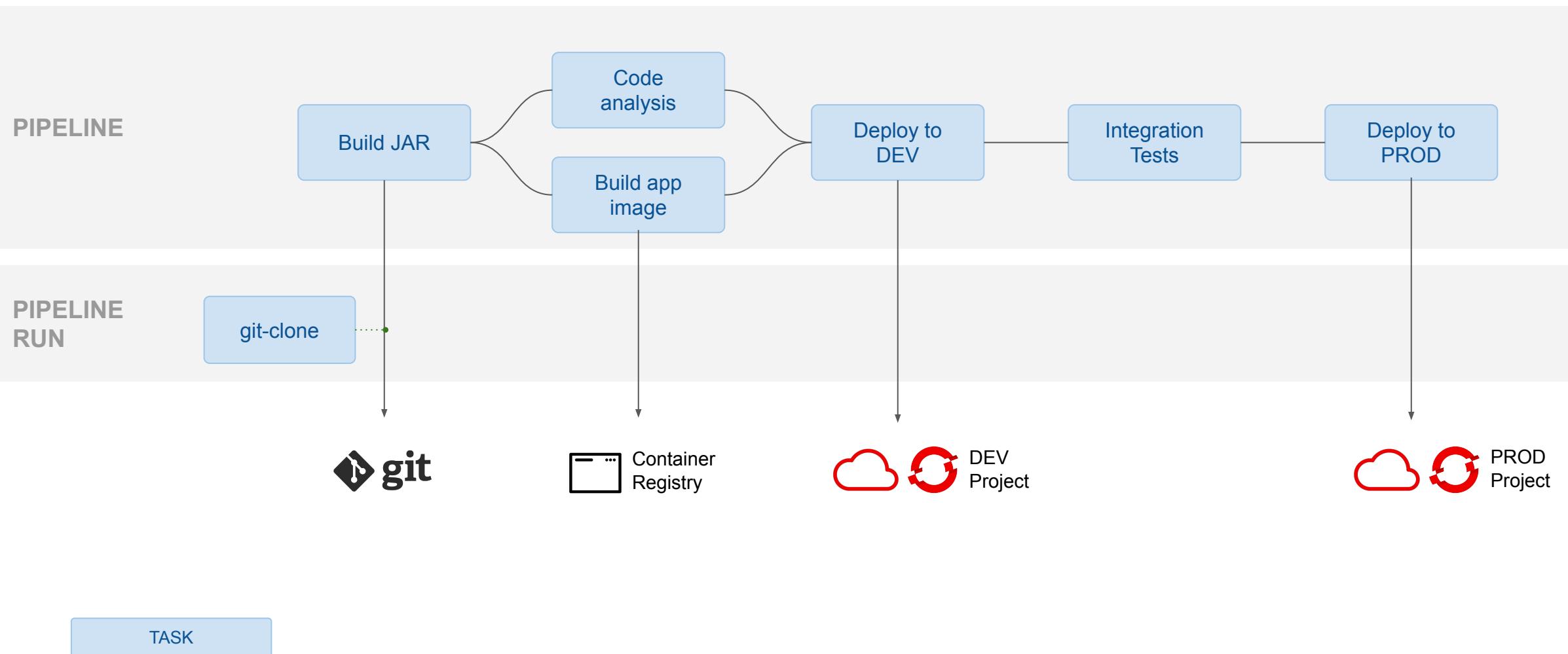
The screenshot shows the Red Hat OpenShift Container Platform web interface. The top navigation bar includes the Red Hat logo, 'OpenShift Container Platform', a user icon, and the name 'siamak'. The left sidebar has a 'Developer' section with options: '+Add' (which is selected and underlined), 'Topology', 'Builds', 'Pipelines', and 'Advanced'. The main content area is titled 'Add' and displays a message: 'No workloads found' followed by 'To add content to your project, create an application, component or service using one of these options.' Below this are six cards arranged in a 2x3 grid:

- 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.

# Create Pipelines with Pipeline UI



# OpenShift Pipeline Example



# Interactive Learning Portal

Our Interactive Learning Scenarios provide you with a pre-configured OpenShift® instance, accessible from your browser without any downloads or configuration. Use it to experiment, learn OpenShift and see how we can help solve real-world problems.

learn.openshift.com

Foundations of  
OpenShift

START COURSE

Building Applications On  
OpenShift

START COURSE

Subsystems,  
Components, and  
Internals

START COURSE

OpenShift Playgrounds

START COURSE

Service Mesh Workshop  
with Istio

START COURSE

Building Operators on  
OpenShift

START COURSE

AI and Machine Learning  
on OpenShift

- ▶ Setup
- ▶ Pipeline Resources
- ▶ Tasks
- ▼ Pipelines
  - Add Tasks from Catalog
  - Create Pipeline**
  - Deploy Pipeline
  - Run Pipeline
  - Test Pipeline
  - Clean
- ▶ Workspaces
- ▶ Private Registries and Repositories
- ▶ Triggers
- ▶ OpenShift Pipelines

## Deploy Pipeline

The Kubernetes service deployment Pipeline could be created using the command:

```
kubectl apply -n tektonTutorial -f svc-deploy.yaml
```

We will use the Tekton cli to inspect the created resources

```
tkn pipeline ls
```

The above command should list one Pipeline as shown below:

NAME	AGE	LAST RUN	STARTED	DURATION
svc-deploy	4 seconds ago	---	---	---

**TIP**

Use the command **help** via `tkn pipeline --help` to see more options

## Run Pipeline

**dn.dev/tekton-tutorial**

# 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)