

OpenShift Developer

Architecture Workshop

CI/CD and GitOps with Tekton & ArgoCD



linkedin.com/company/red-hat



facebook.com/redhatinc



youtube.com/user/RedHatVideos



twitter.com/RedHat

Self introduction

Name: Wanja Pernath

Email: wpernath@redhat.com

Base: Germany (very close to the Alps)

Role: EMEA Technical Partner Development Manager

- OpenShift and MW

Experience: Years of Consulting, Training, PreSales at
Red Hat and before

Twitter: <https://twitter.com/wpernath>

LinkedIn: <https://www.linkedin.com/in/wanjapernath/>

GitHub: <https://github.com/wpernath>



First book just published

Getting GitOps

A technical blueprint for developing with Kubernetes and OpenShift based on a REST microservice example written with Quarkus

Technologies discussed:

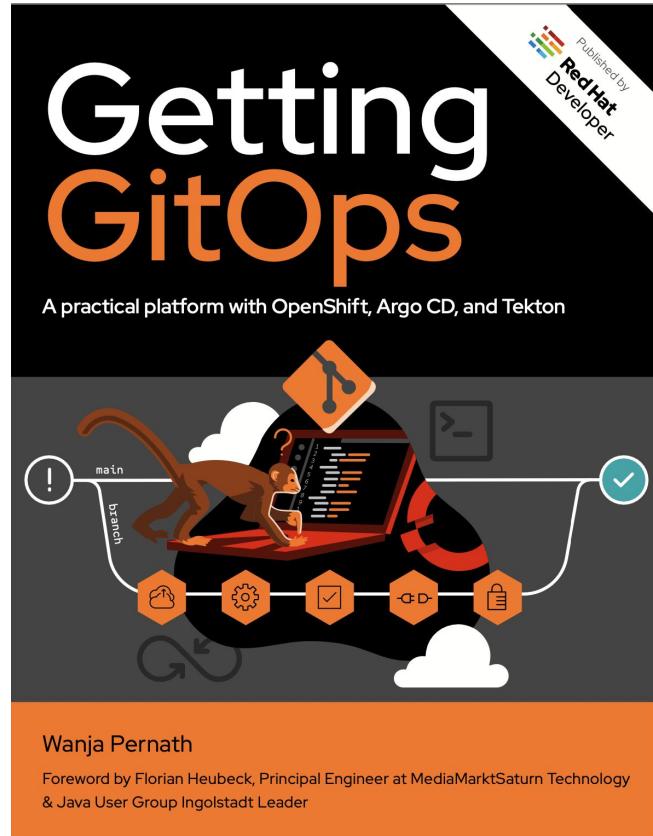
Quarkus, Helm Charts, Kustomize, Tekton Pipelines, Kubernetes Operators, OpenShift Templates, ArgoCD, CI/CD, GitOps....

Download for free at:

<https://developers.redhat.com/e-books/getting-gitops-practical-platform-openshift-argo-cd-and-tekton>

Interview with full GitOps Demo:

https://www.youtube.com/watch?v=znMfVqAIRzY&ab_channel=OpenShift



Agenda / etc.

Agenda

- CI/CD with OpenShift
 - Basics
 - OpenShift Builds
 - OpenShift Pipelines
 - GitOps
 - Summary

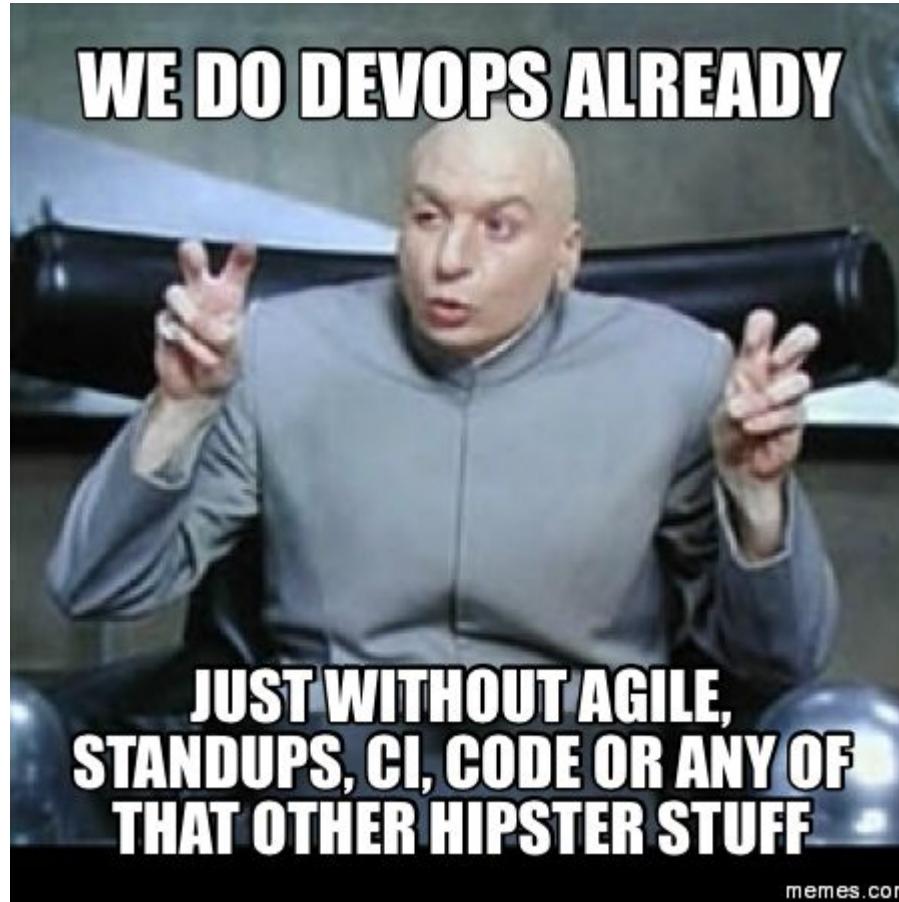
CI/CD with OpenShift

Basics

Wouldn't it be great, if everything could be automated?

Wouldn't it be great, if we simply do <insert hype here> and it solves all problems we have?

With DevOps we have a principle which solves most of the issues - but wait! The *DevOps Person* of the customer is currently on vacation



Basics - Developer

- Important: Separation of code and config!
- Writes the code of the App
- Writes set of build files (maven, gradle, etc.)
- Writes all needed tests (unit, integration, load)
- Writes Pipeline files
- Writes kubernetes manifest files
- Stores everything auditable in Git
- Finds a way and tools to combine all of the above

Basics - Operations

- Gets images, configs, test descriptions from Devs
- Adds own kubernetes manifests to the soup
- Makes sure, everything gets deployed
- Makes sure every dependency is installed and ready
- Thinks about security
- Thinks about networking
- Thinks about storage
- Thinks about compute power
- Thinks about plumbing everything together

Basics

CI/CD

=

Automation of Software Delivery

DevOps

=

Let's have Dev and Ops TALK to each other

Continuous Integration(CI) & Continuous Delivery (CD)

A key DevOps principle for automation, consistency and reliability



Tekton / OpenShift Pipelines

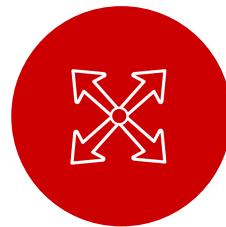
Kubernetes native on
demand delivery
pipelines

What is Cloud-Native CI/CD?



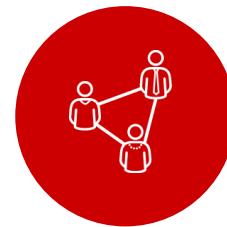
Containers

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



Plugins shared across CI engine
Jenkins
Plugins 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



Tasks fully isolated from each other
TEKTON
Everything literally has a Dockerfile and Images

Native Kubernetes resources

Platform manages persistence

Configured via Kubernetes ConfigMaps

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



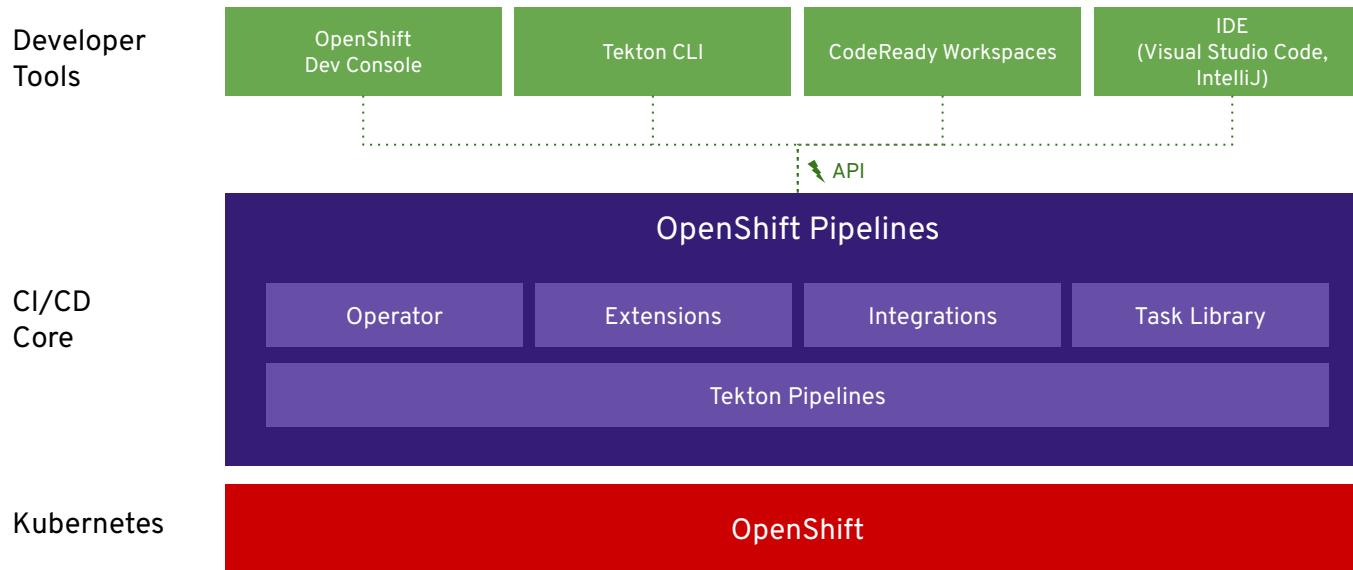
An open-source project for providing a set of shared and standard components for building Kubernetes-style CI/CD systems

18



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

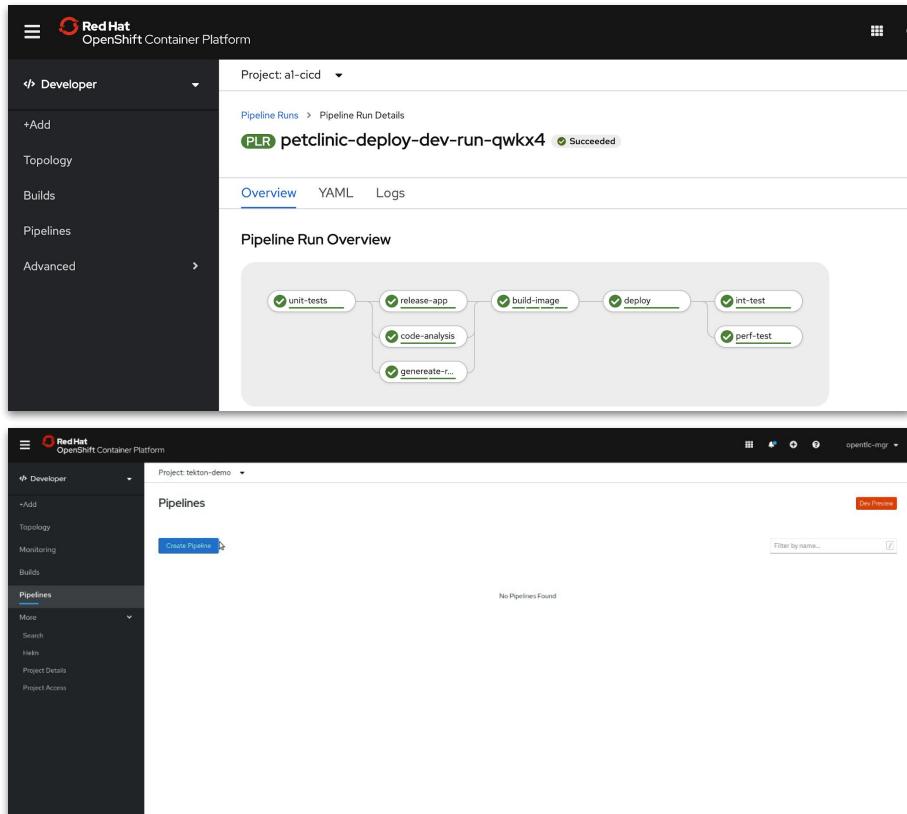
OpenShift Pipelines



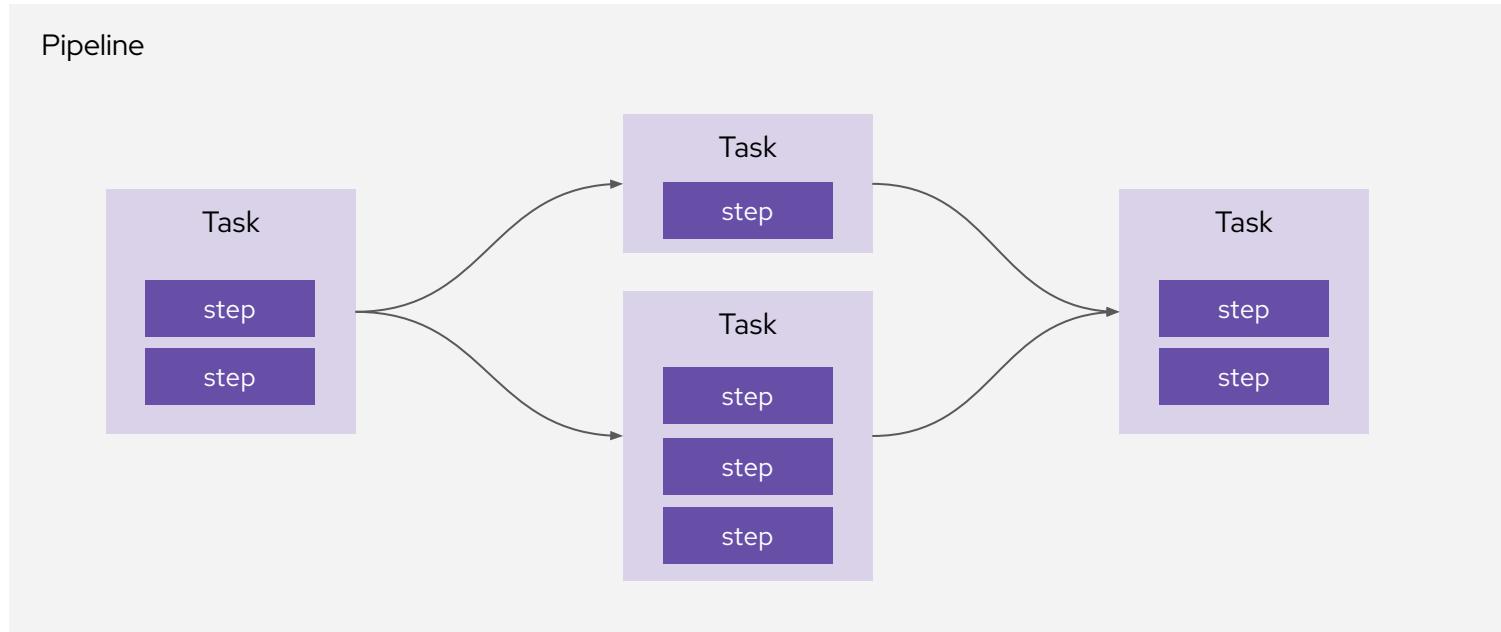
OpenShift Pipelines

- 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

20



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

24

The screenshot shows the Tekton Hub interface, which is a platform for discovering and sharing reusable Tasks and Pipelines. The top navigation bar includes the Tekton Hub logo, a search bar, and a 'Login' button. Below the header, a welcome message reads 'Welcome to Tekton Hub' and 'Discover, search and share reusable Tasks and Pipelines'. A sidebar on the left provides filtering options for 'Sort' (set to 'Name'), a search bar, and sections for 'Refine By:', 'Kind' (Task or 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 star rating, version, description, update time, and tags.

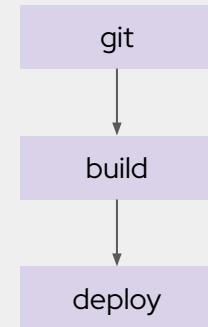
Name	Version	Rating	Description	Last Updated	Tags
Ansible Runner	v0.1	4.5	Task to run Ansible playbooks using Ansible Runner	Updated 3 weeks ago	ansible, cli
ansible tower cli	v0.1	2.0	Ansible-tower-cli task simplifies starting jobs, workflow jobs, manage users, projects etc.	Updated 3 weeks ago	ansible, cli
argocd	v0.1	3.0	This task syncs (deploys) an Argo CD application and waits for it to be healthy.	Updated 3 weeks ago	deploy
aws cli	v0.1	5.0	This task performs operations on Amazon Web Services resources using aws.	Updated 3 weeks ago	cli
Amazon ECR Login	v0.1	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	v0.1	1.0	This task performs operations on Microsoft Azure resources using az.	Updated 4 months ago	cli
bentoml	v0.1	0.0	This task allows operations on BentoML services	Updated 3 weeks ago	cli
Python Black	v0.1	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

25

```
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"]
```



Tekton CLI, Visual Studio Code, and IntelliJ

The screenshot shows the Tekton CLI interface with two main panels. On the left, the 'sample_test.yaml' file is displayed in a code editor, showing a pipeline definition with tasks like 'scaffold-unit-tests', 'build-skaffold-web', 'build-skaffold-app', 'deploy-web', and 'deploy-app'. On the right, a diagram illustrates the flow of the pipeline tasks. Below these, the 'Task: buildah' panel is open, showing details for the 'buildah' task, which builds source into a container image and pushes it to a container registry. The 'Tasks' section lists various buildah tasks such as 'buildah', 'golang-build', 'buildah-base', 'buildah-phases', and 'buildah-phases-parallel'.

```
sample_test.yaml
...
  tasks:
    - name: scaffold-unit-tests
      taskRef:
        name: scaffold-unit-tests
    - name: build-skaffold-web
      taskRef:
        name: build-skaffold-web
    - name: build-skaffold-app
      taskRef:
        name: build-skaffold-app
    - name: deploy-web
      taskRef:
        name: deploy-web
    - name: deploy-app
      taskRef:
        name: deploy-app
  resources:
    inputs:
      - name: workspace
        resourceRef:
          name: workspace-source-repo
    outputs:
      - name: workspace
        resourceRef:
          name: workspace-build
  parameters:
    - name: pathToDockerfile
    - name: pathToContext
    - name: pathToImage
    - name: resources
    - name: target
    - name: workspace
```

Task: buildah

buildah

tekton • 4 | Repository

Buildah task builds source into a container image and then pushes it to a container registry.

Install

Description: YAML

Buildah

This Task builds source into a container image using Project Atomic's `buildah` build tool. It uses `Buildah`'s support for building from `Dockerfile`s using `buildah build` command. This command executes the directives in the `Dockerfile` to assemble a container image, then pushes that image to a container registry.

Install the task

`kubectl apply -f https://raw.githubusercontent.com/tektoncd/catalog/master/task/buildah/v1.2/buildah.yaml`

Parameters

- **IMAGE:** The name (reference) of the image to build.
- **BUILDER_IMAGE:** The name of the image containing the `Buildah` tool. See note below. (default: `quay.io/buildah/stable:v1.17.0`)
- **DOCKERFILE:** The path to the `Dockerfile` to execute (default: `./Dockerfile`)
- **CONTEXT:** Path to the directory to use as context (default: `.`)
- **TLSCACB:** Verify the TLS on the registry endpoint (not possible if a non-TLS registry) (default: `true`)
- **FORMAT:** Docker image format (either `tar` or `oci`) (default: `oci`)
- **BUILD_EXTRA_ARGS:** Extra parameters passed for the `build` command when building images. (default: `" "`)
- **PUSH_EXTRA_ARGS:** Extra parameters passed for the `push` command when pushing images. (default: `" "`)

Workspaces

- source: [Workspace](#) containing the source to build.

- `secretRef:` An [optional Workspace](#) containing your custom SSL certificates to connect to the registry. `Buildah` will look for files ending with `.cert`, `.key` and `.crt` files in this workspace. See [this example](#) for a complete example on how to use it with OpenShift internal registry.

Usage

This TaskRun runs the Task to fetch a Git repo, build and push a container image using `Buildah`.

```
apiVersion: tekton.dev/v1beta1
kind: TaskRun
metadata:
  name: buildah-build-my-repo
spec:
```

The screenshot shows the Tekton Hub search results in both Visual Studio Code and IntelliJ IDEA. In the terminal, the command `teckon search` is run, and the results are displayed in a modal window. The results include various tasks and cluster tasks such as 'aws-cli', 'buildah', 'curl', 'git-dl', 'git-clone', 'golang-build', and 'jenkins'. Each result has an 'Install' button. The 'buildah' entry is currently selected. The IntelliJ interface shows the search results in a separate window, and the 'Terminal' tab in the bottom right shows the execution of the search command.

ssadeghi@Siamaks-MacBook-Pro: ~

~\$ teckon search

Results (102)

- aws-cli
- buildah
- curl
- git-dl
- git-clone
- golang-build
- jenkins

Select Tekton Hub item to preview details

OK Cancel

Thank you!

<https://github.com/wpernath/openshift-cicd-demo>



OpenShift GitOps

Declarative GitOps for
multi-cluster continuous
delivery

What is GitOps?

GitOps is when the infrastructure and / or application state is fully represented by the contents of a git repository. Any changes to the repository are reflected in the corresponding state of the associated infrastructure and applications through automation

GitOps is a natural evolution of Agile and DevOps (and Kubernetes) methodologies

Why GitOps

It takes too long to provision a new environment!

The app behaves different in prod than in test!

I have no visibility or record of config changes in deployments!

I can't easily rollback changes to a specific version

Environments are all manually configured!!!

Production deployments have a low success rate!

I can't audit config changes!!



GitOps Benefits

- All changes are auditable
- Standard roll-forward or backwards in the event of failure
- Disaster recovery is “reapply the current state of the manifests”
- Experience is “pushes and pull-requests”

OpenShift and GitOps - A great match

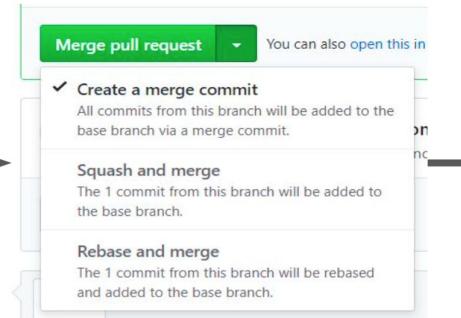
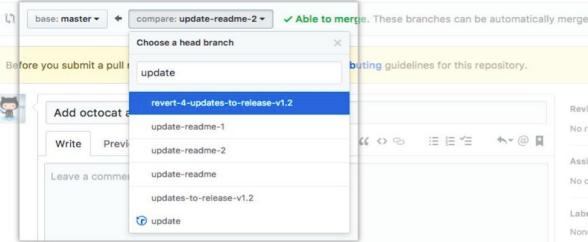
- OpenShift is a declarative environment
 - Cluster configuration is declared and Operators make it happen
 - Application deployments are declared and Kubernetes scheduler makes it happen
- GitOps in traditional environments requires automation/scripting, declarative environment minimizes or eliminates this need
- Declarations are yaml files which are easily stored and managed in git



Day 2 operations: All changes triggered from Git

Open a pull request

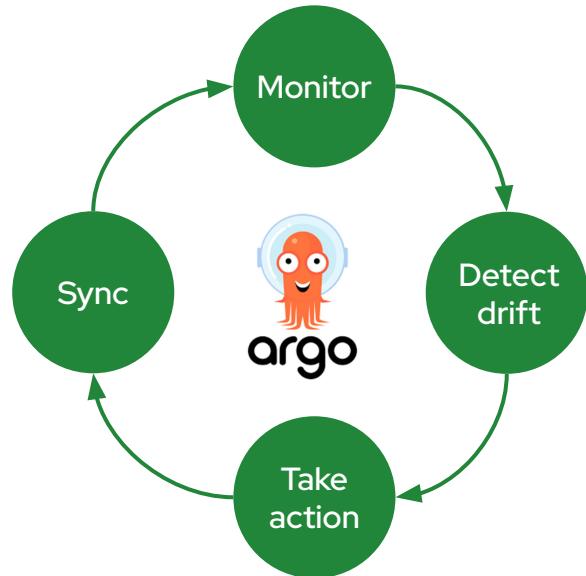
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.



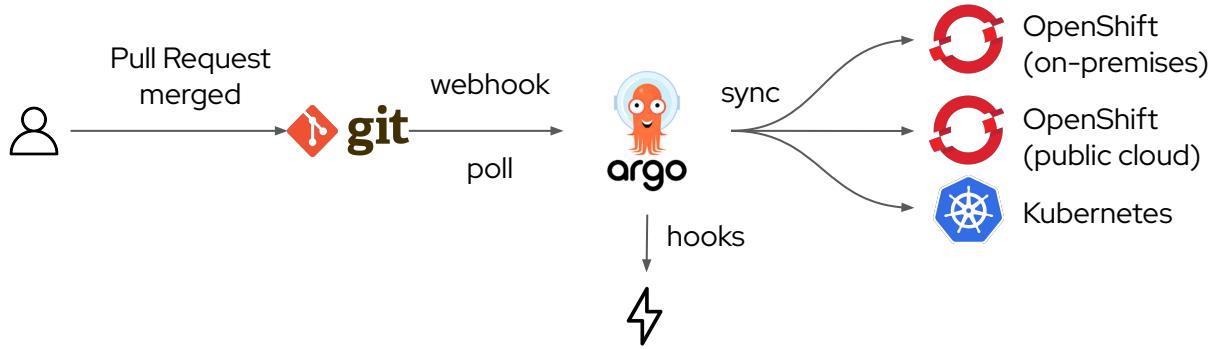
```
$ tkn pipelinerun logs update-from-master-run-g6s45
[run-kubectl] {"level": "info", "ts": 1580989837.3045664, "logger": "fallback-logger", "caller": "logging/config.go:69", "msg": "Fetch GitHub commit ID from kodata failed: \"\\$KO_DATA_PATH\" does not exist or is empty"}
[run-kubectl] serviceaccount/demo-sa unchanged
[run-kubectl] clusterrolebinding.rbac.authorization.k8s.io/tekton-triggers-openshift-binding unchanged
[run-kubectl] eventlistener.tekton.dev/demo-event-listener configured
[run-kubectl] task.tekton.dev/deploy-from-source-task configured
[run-kubectl] triggerbinding.tekton.dev/update-from-master-binding unchanged
[run-kubectl] triggertemplate.tekton.dev/update-from-master-template unchanged
```

Argo CD

- Cluster and application configuration versioned in Git
- Automatically syncs configuration from Git to clusters
- Drift detection, visualization and correction
- Granular control over sync order for complex rollouts
- Rollback and rollforward to any Git commit
- Manifest templating support (Helm, Kustomize, etc)
- Visual insight into sync status and history



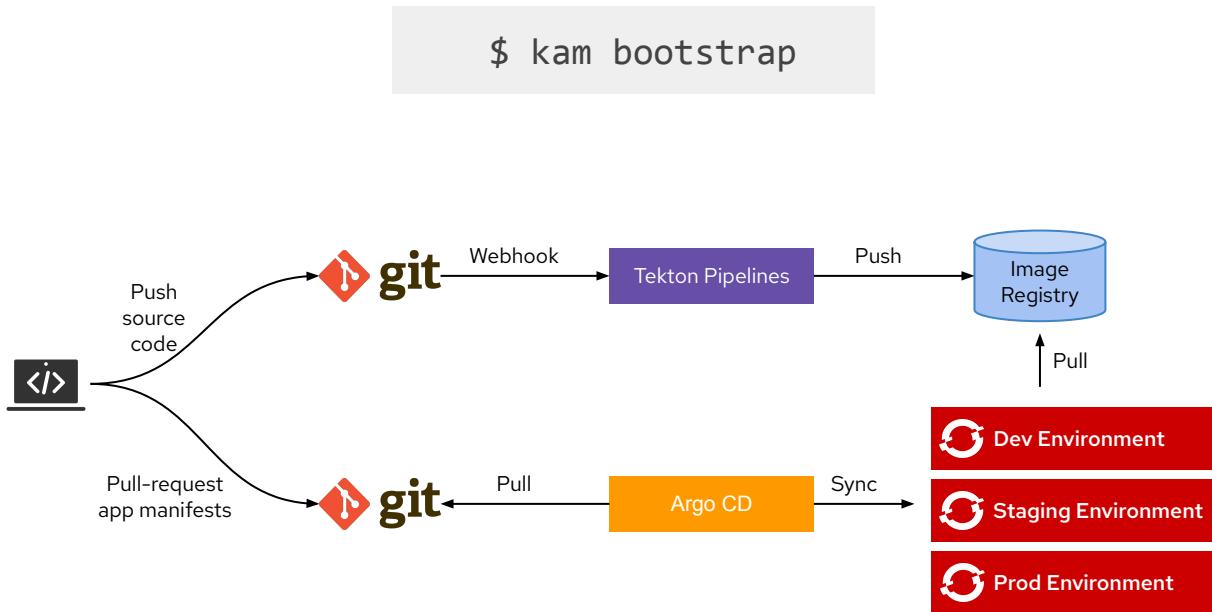
Argo CD



Screenshot of the Argo CD web interface for the "realtime" application. The interface shows the following components:

- Sync Status:** Synced to latest commit (rev 1). Last sync was 38 minutes ago on Dec 25, 2019.
- Deployment:** realtimeapp (rev 2) replicant realtimeapp-5bcb859455 (running 1/1 pod).
- Replicant:** realtimeapp (rev 1) replicant realtimeapp-6f449c7f6b (running 1/1 pod).
- Service:** redisapp (rev 1) deployment redisapp-74b945bf (running 1/1 pod).
- Endpoint:** redisapp (rev 1) replicant redisapp-74b945bf (running 1/1 pod).

GitOps Application Manager CLI



OpenShift GitOps



Multi-cluster config management

Declaratively manage cluster and application configurations across multi-cluster OpenShift and Kubernetes infrastructure with Argo CD



Automated Argo CD install and upgrade

Automated install, configurations and upgrade of Argo CD through OperatorHub



Opinionated GitOps bootstrapping

Bootstrap end-to-end GitOps workflows for application delivery using Argo CD and Tekton with GitOps Application Manager CLI



Deployments and environments insights

Visibility into application deployments across environments and the history of deployments in the OpenShift Console

ArgoCD - Challenges

- Repo structure for manifests
 - One Repo or
 - Separate Repos for environments
- Order dependent deployments
- Non-declarative deployments
- Integration with CI/CD tools (Jenkins, Pipelines...)
 - Who does manage deployments?



Multiple repositories

/taxi.git

deploy

pipelines

pkg/cmd/booktaxi

web

Dockerfile

/taxi-config-stage.git

deploy

pipelines

README.md

/taxi-config-prod.git

deploy

pipelines

README.md

/taxi-config-test.git

deploy

pipelines

README.md

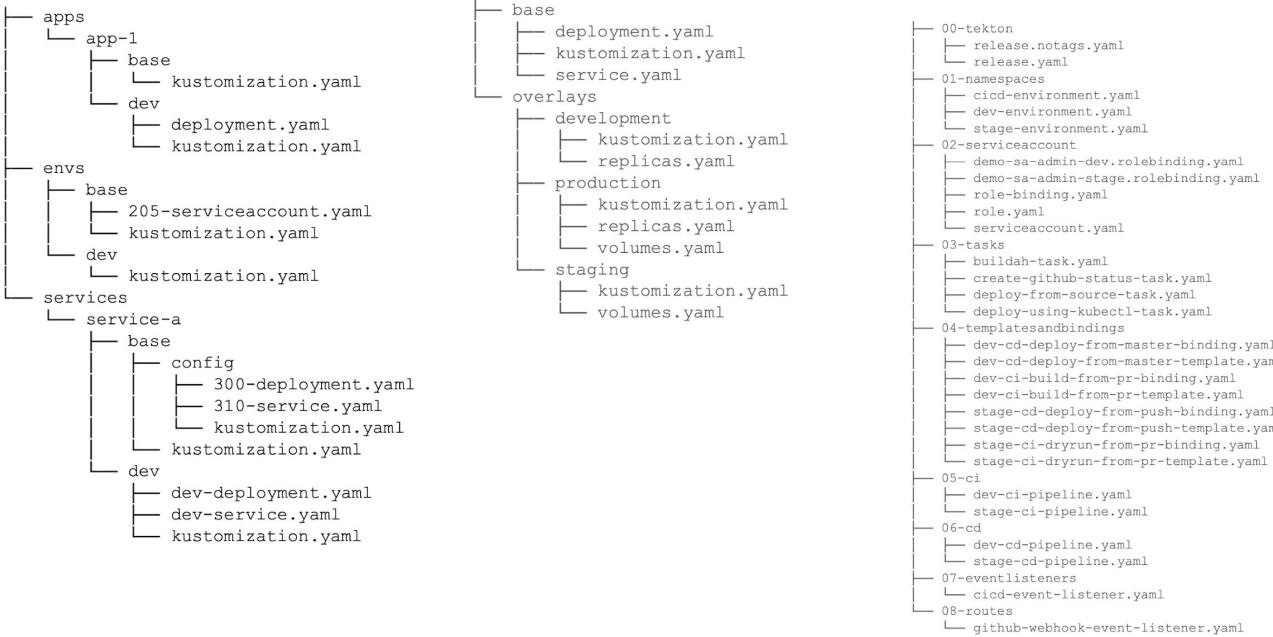
/taxi-config-dev.git

deploy

pipelines

README.md

Single Repository



ArgoCD - Order Dependent Deployments

- Sometimes you have cases where you need to deploy things in a specific order
 - Subscribe Operator before deploying instance
 - Create namespace before deploying app into it
 - Deploy required infrastructure before application
- Tools like kustomize and Helm might help handling this in some cases
- ArgoCD provides Sync Phases and Waves to address other use cases
 - 3 sync phases - Pre-sync, sync, post-sync
 - Each phase can have multiple waves, next wave does not proceed until previous phase is healthy

ArgoCD - Non-declarative requirements

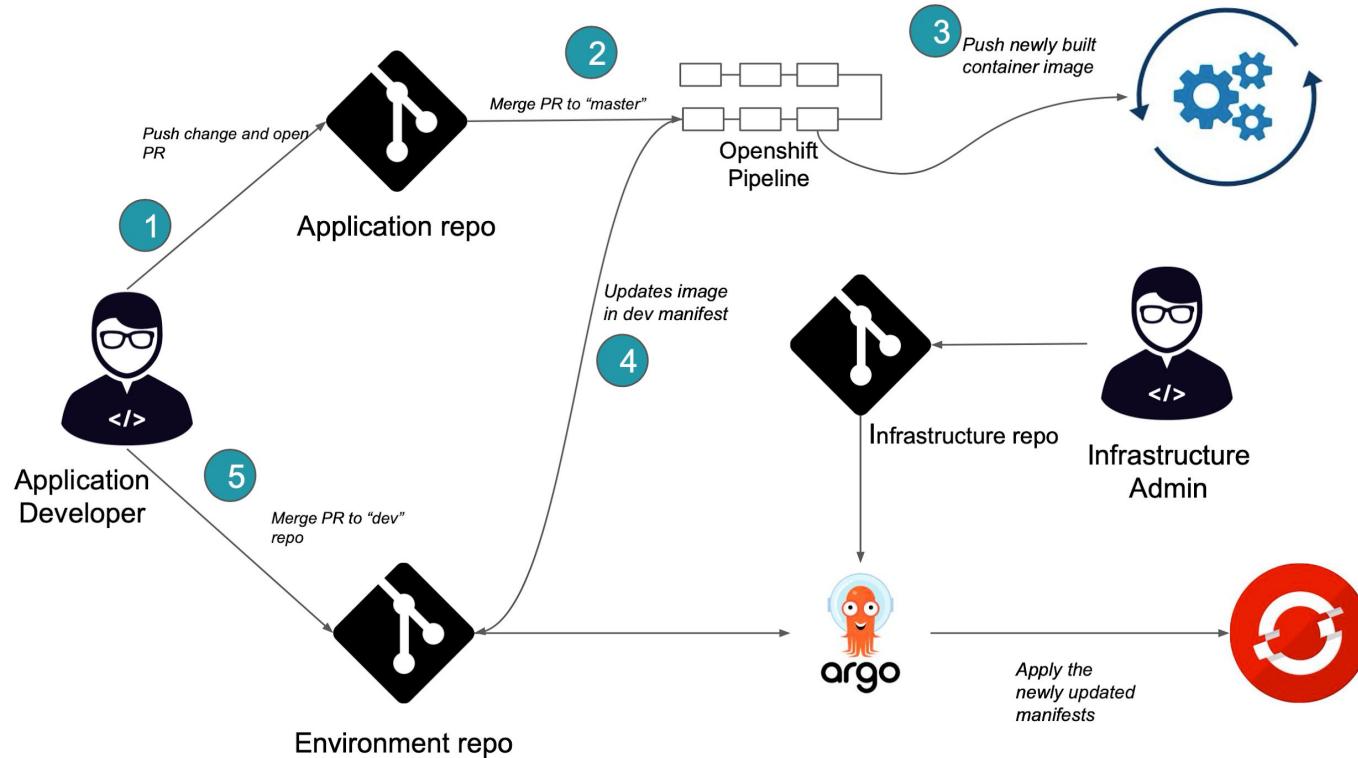
- There can be instances where you need to deploy something which cannot fully be done in a declarative way and it must be scripted
- Try minimizing this and leverage kubernetes primitives where possible
 - Init Containers
 - Jobs
 - Operators
- ArgoCD Resource Hooks
 - Hooks are ways to run scripts before, during and after a sync operation
 - Hooks can be run: PreSync, Sync, PostSync and SyncFail

ArgoCD - Integrating with CI/CD Tools

ArgoCD and NOT ArgoCI/CD

CI tools like Jenkins, Pipelines still required!

	ArgoCD Managed Deployment	Pipeline Managed Deployment
Pro	Consistent	Post-Test update of image reference
Con	Image reference updated in git before integration tests, manage rollback?	Inconsistent
Con	Pipeline tools must be able to wait for sync	



DEMO TIME

<https://github.com/wpernath/openshift-cicd-demo>

Summary

Summary

- You don't have to change anything, if you don't want
- You can use tools like Jenkins to do your CI
- OpenShift Builds is to integrate building your image in your existing pipeline
- OpenShift Pipelines (tekton) is a nice kubernetes-native way of pipelining
- GitOps and ArgoCD helps you to do a declarative approach
 - You as developer are used to use Git
 - You as admin are used to use Git
 - → Adoption of GitOps could be high

Optional section marker or title

48



Thank you

 [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