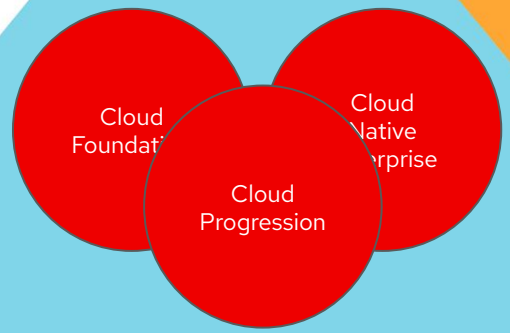
Red Hat Summit

GitOps from Development to Production

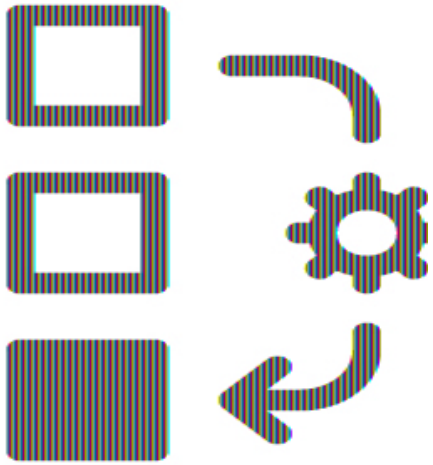
Continuous Integration and Continuous Delivery on OpenShift

Johana Limka
Associate Solution Architect, Red Hat

Mark Roberts
Principal Solution Architect, Red Hat



Red Hat



Objective

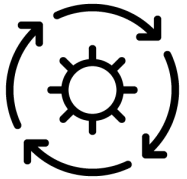
To create an automated build and deploy process

Red Hat

GitOps Principles

The system is described declaratively

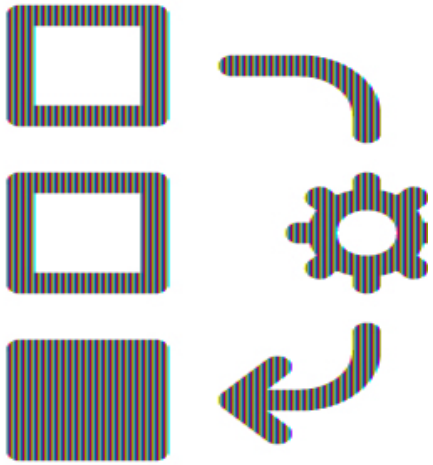
The desired state is versioned in Git



Approved changes can be applied automatically

A controller exists to detect and act on drift

Red Hat



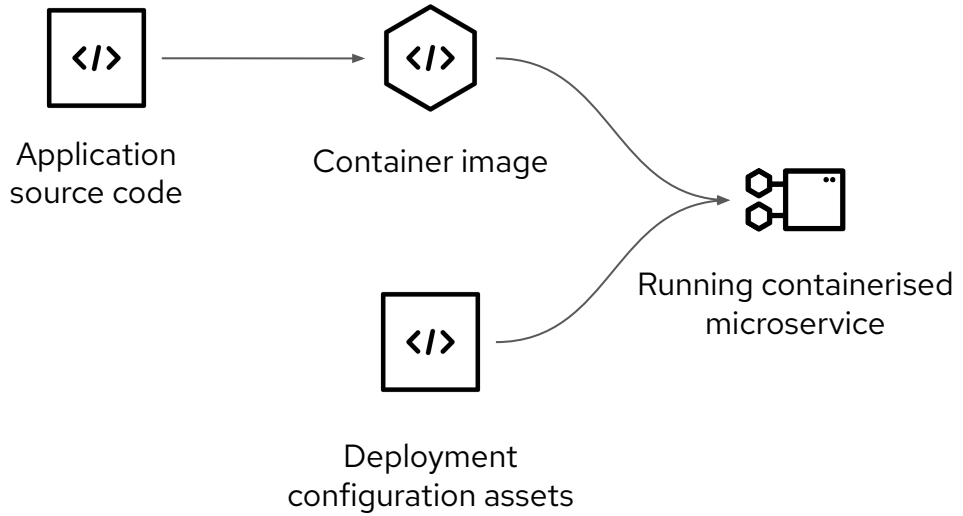
Objective

To create an automated build and deploy process in which the assets are stored in and managed from a Git repository

Red Hat

Source content and objectives

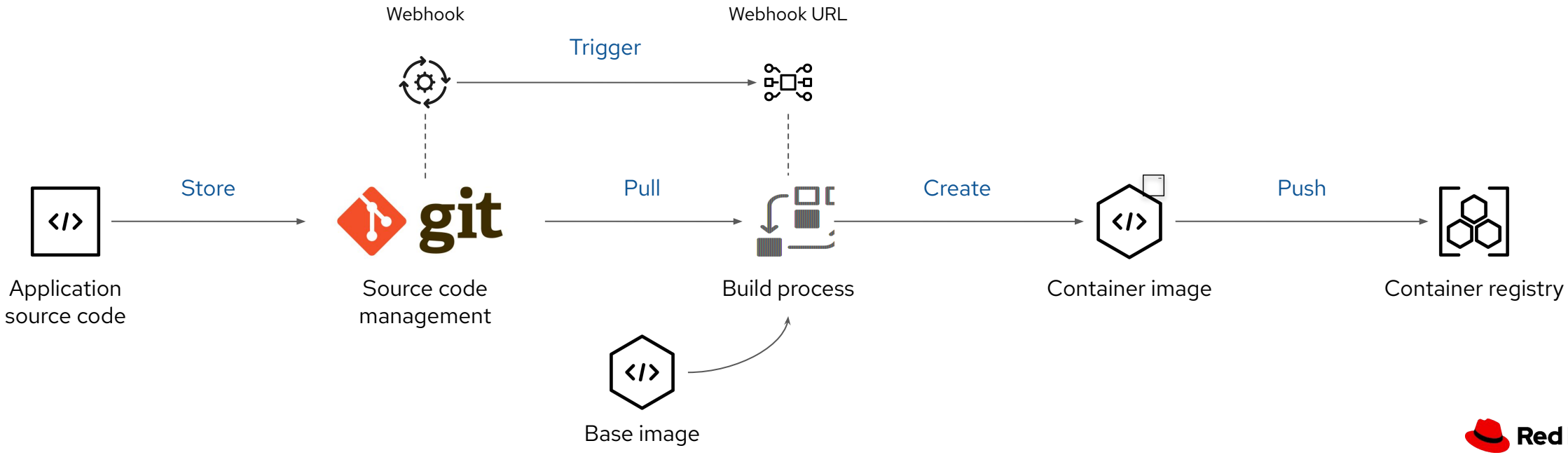
- From source content to running microservice
  - Source code of application
  - Container images (both a target and a source asset)
  - Kubernetes resources - services, configuration maps, secrets etc.
- Multiple steps to manage
  - Software build process
  - Container image creation and storage
  - Deployment of container image from storage
  - Application of Kubernetes resources
- Two major phases
  - Build - Continuous integration
  - Deploy - Continuous delivery



Red Hat

Source code to container image cycle - Continuous Integration

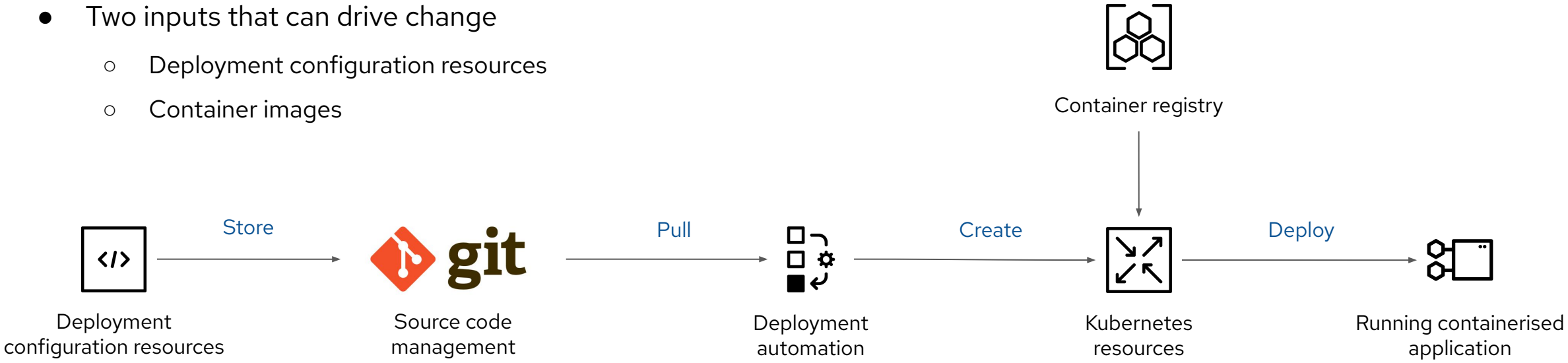
- Store source code in a Git repository
- Pull the source code and execute a build process
  - The build definition assets are also stored in a Git repository
- Create a new container image as part of the build process
  - This process will use a source container image that may be stored in the container registry or an external registry
- Push the new container image to the container registry with a new identification tag



Red Hat

Container image deployment - Continuous Delivery

- Store resource definitions in a Git repository
- Pull the resource definitions from Git
- Create Kubernetes resources based on resource definition
  - Refer to the container image in the container registry
- Deploy assets based on the desired state of Kubernetes resources
- Result : A new running application
- Two inputs that can drive change
  - Deployment configuration resources
  - Container images

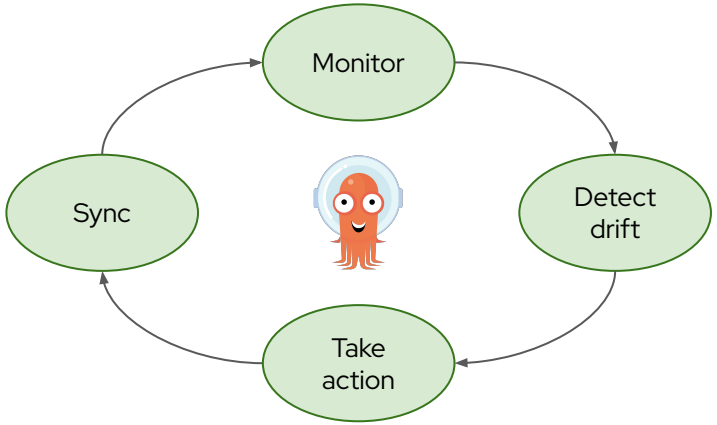


Red Hat

OpenShift GitOps and OpenShift Pipelines

OpenShift GitOps

- Delivered as an operator on OpenShift clusters
- Based on ArgoCD open source project
- Synchronisation maintained between the Git repository and resulting assets
- ‘Application’ definition Git repository -> Kubernetes resource



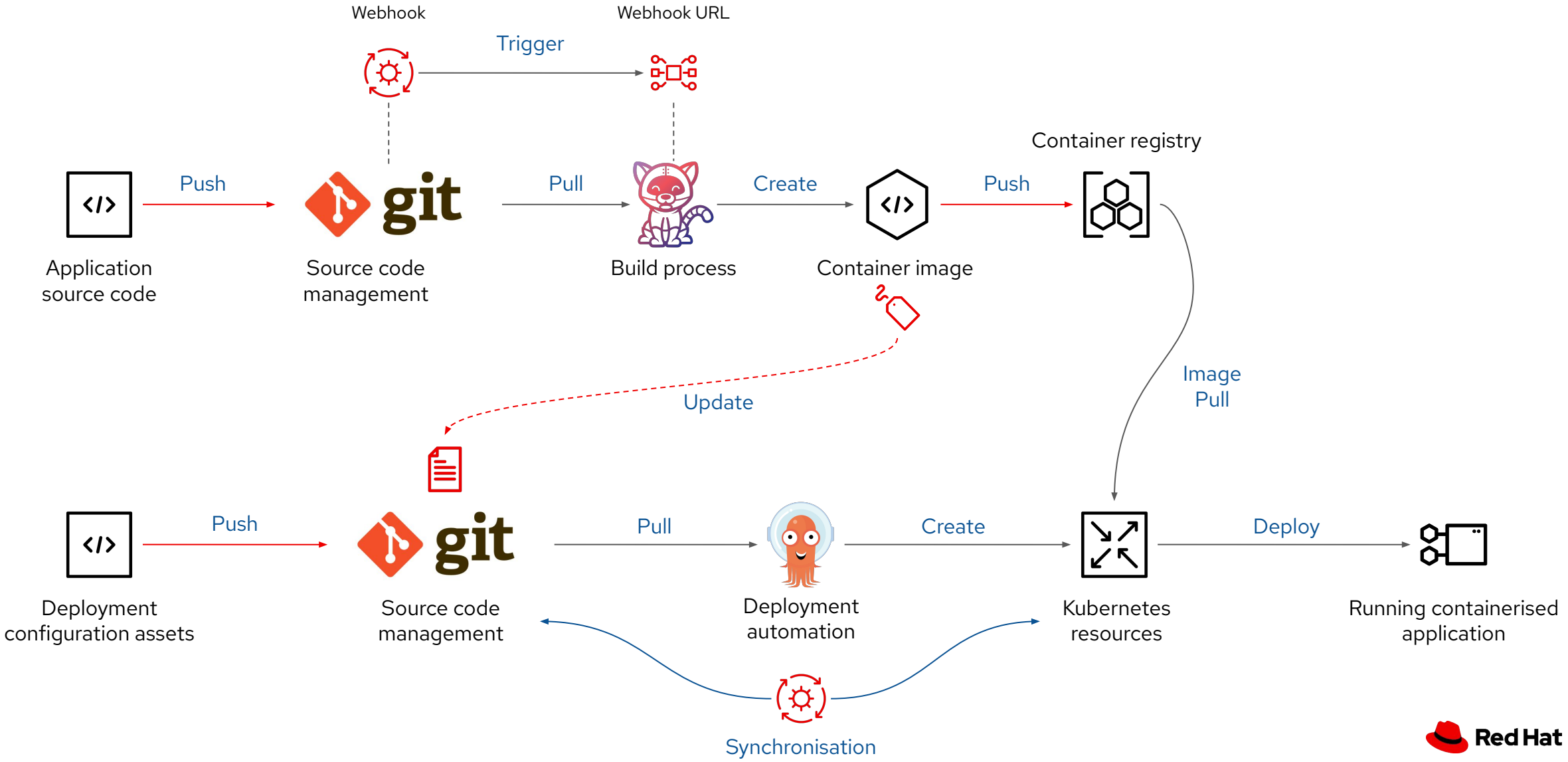
OpenShift Pipelines

- Delivered as an operator on OpenShift clusters
- Based on Tekton open source project

- Cloud native continuous integration process
- Tasks execute in isolated pods
  - Each step of a task is a specific and unique container
- Pipeline and task definitions stored in Git repository
- Kubernetes resources based on Git content managed by OpenShift GitOps

Red Hat

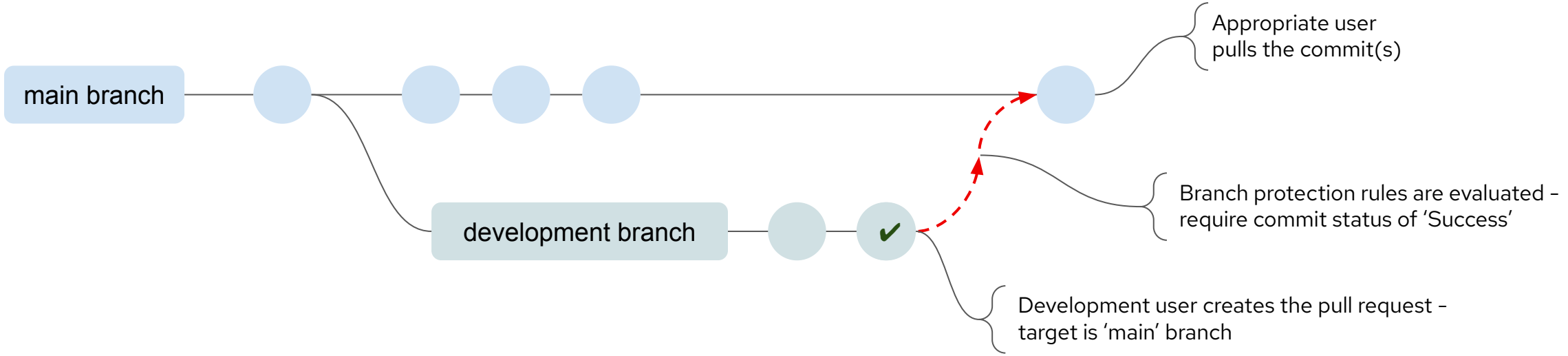
CI / CD Triggers



Red Hat

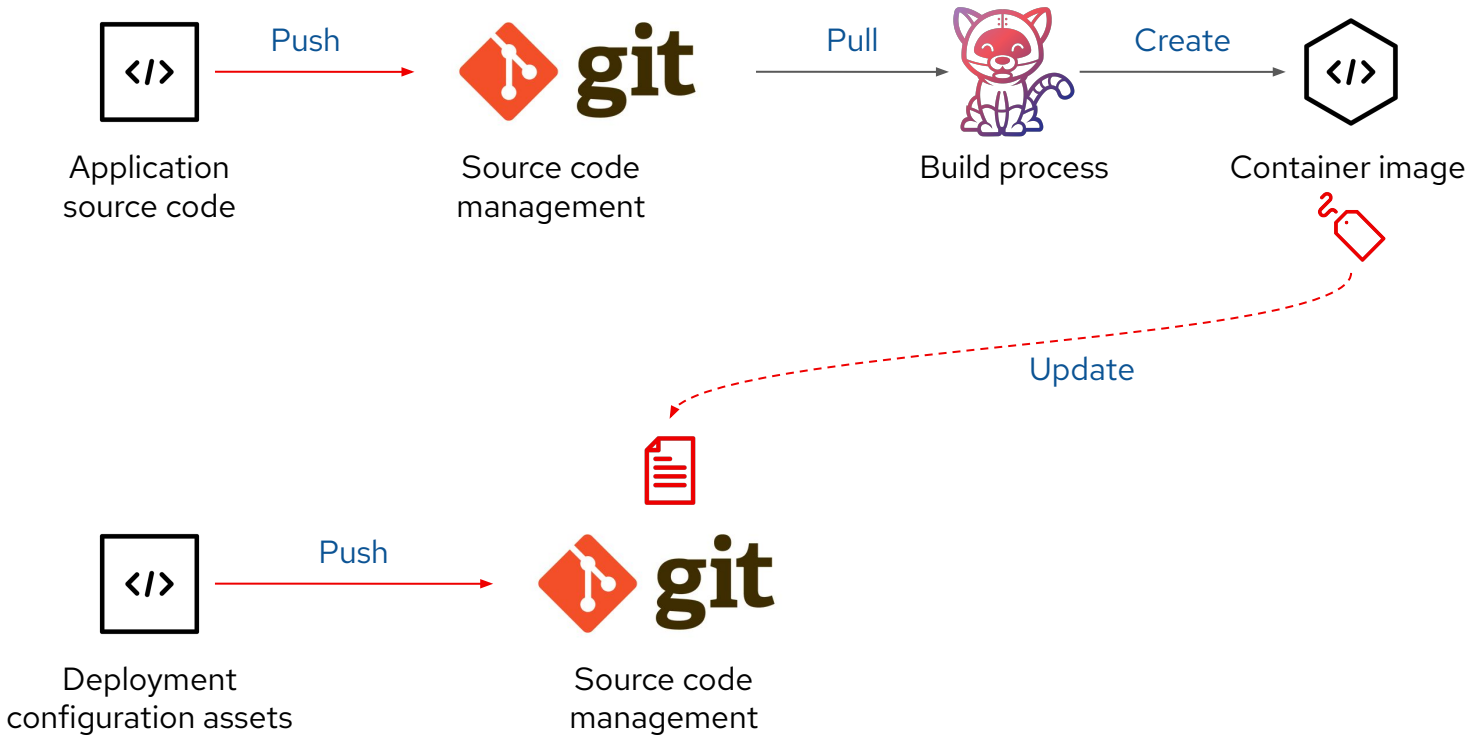
Git pull request process

- To merge changes back to main create a pull request
  - Request that a person with appropriate permission ‘pulls’ the changes from the development branch
- Branch protection rules
  - Require the pull request is used
  - Require approvals prior to merge
  - Require status checks on commits evaluate to ‘pass’
  - etc.



Red Hat

Update the deployment content with new image tag



- New image tag identifier
- To be used in the deployment process
- Update process
  - Update the deployment asset
  - Commit to Git

- Solution ? Kustomize.io

Red Hat

Kustomize

kustomize.io

- A text replacement and resource build process
- Template file -
  - 'Deployment time' text replacement process
  - Text patching directives

Git repository files

Base deployment file

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: myapp
  namespace: myapp-prod
spec:
  template:
    spec:
      containers:
      - name: myapp
        image: quay.io/<repository>/myapp-runtime
```

Kustomize file

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
  - deployment.yaml
  - service.yaml
  - route.yaml
  - configmaps.yaml
images:
  - name: quay.io/<repository>/myapp-runtime
    newTag: 27e8
```

Deployment time 'dynamic files'

New deployment file

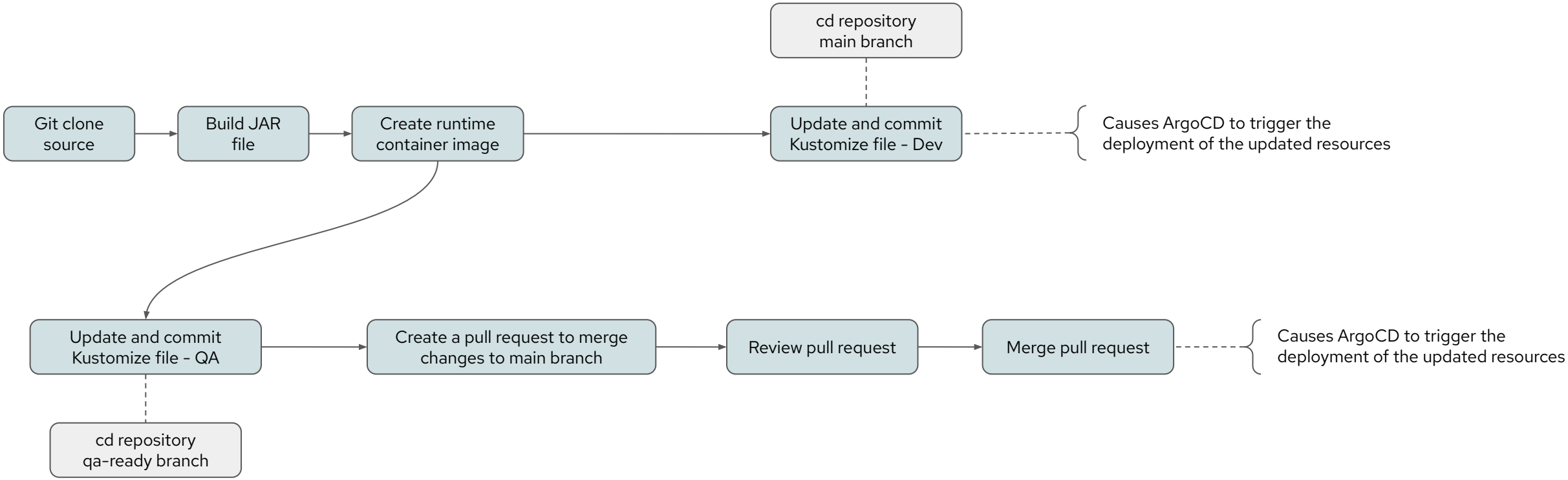
```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: myapp
  namespace: myapp-prod
spec:
  template:
    spec:
      containers:
      - name: myapp
        image: quay.io/<repository>/myapp-runtime:27e8
```

```
kustomize edit set image quay.io/<repository>/myapp-runtime:27e8
```

Red Hat

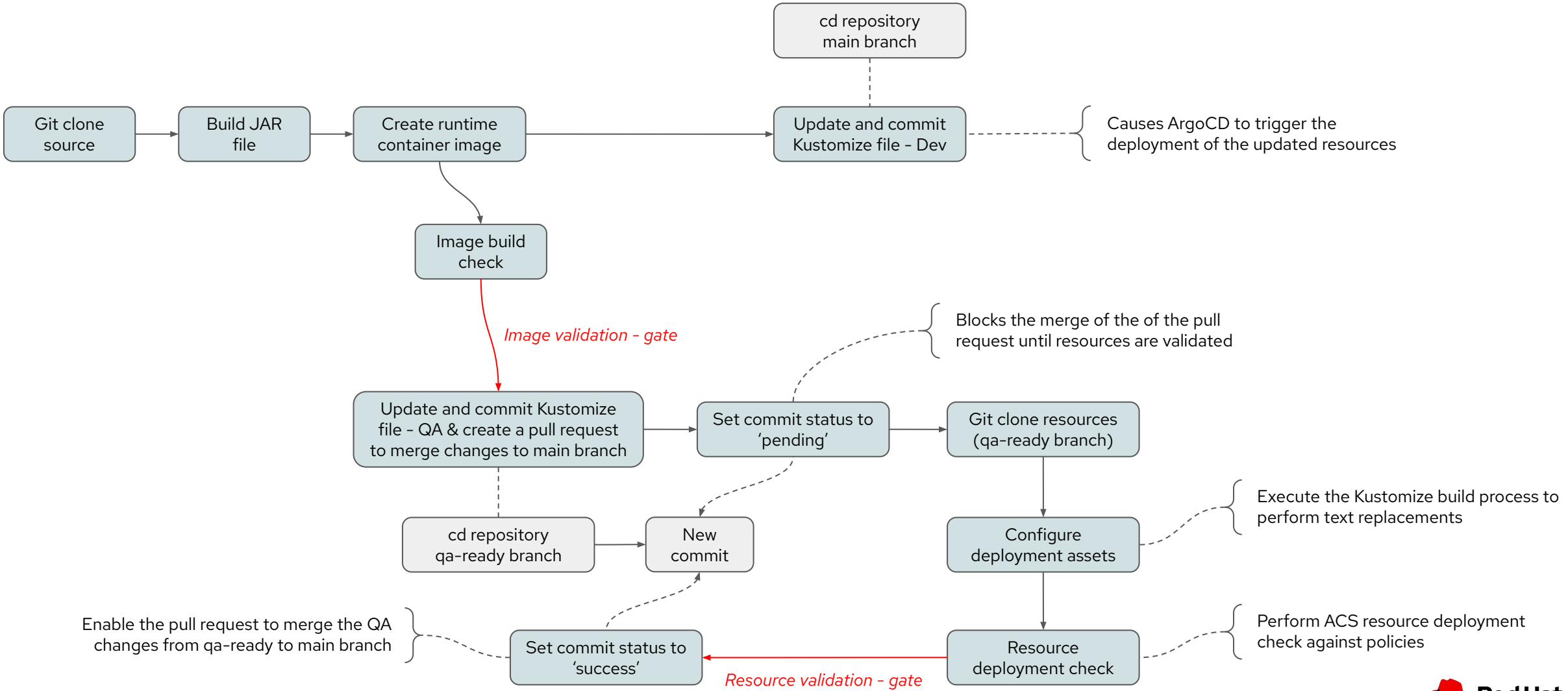
Pipeline process for development and QA

- Use a simple update and deploy process for development
- Use a pull request to create a review point for QA



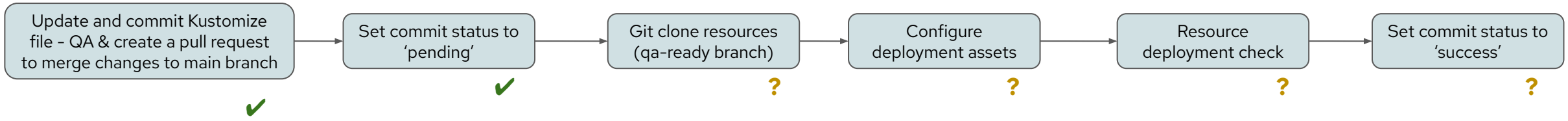
Red Hat

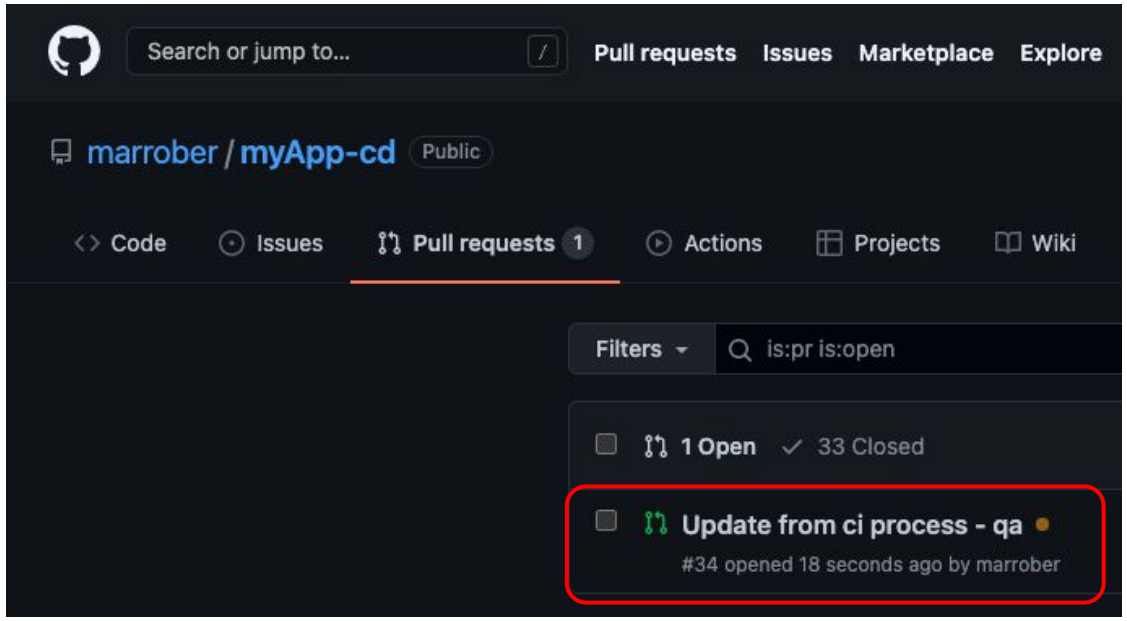
QA pipeline - Use a pull request to manage the release



Red Hat

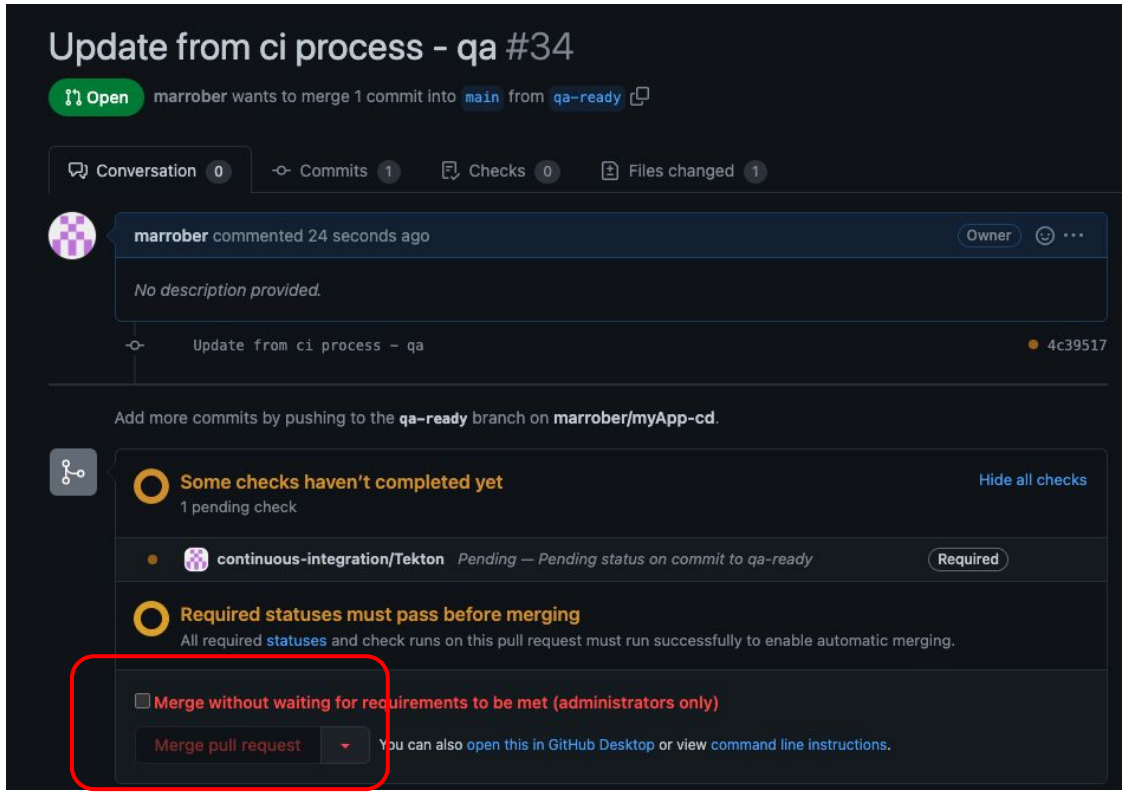
Pipeline in action – before resources are approved





Pull request summary with orange 'pending' indicator

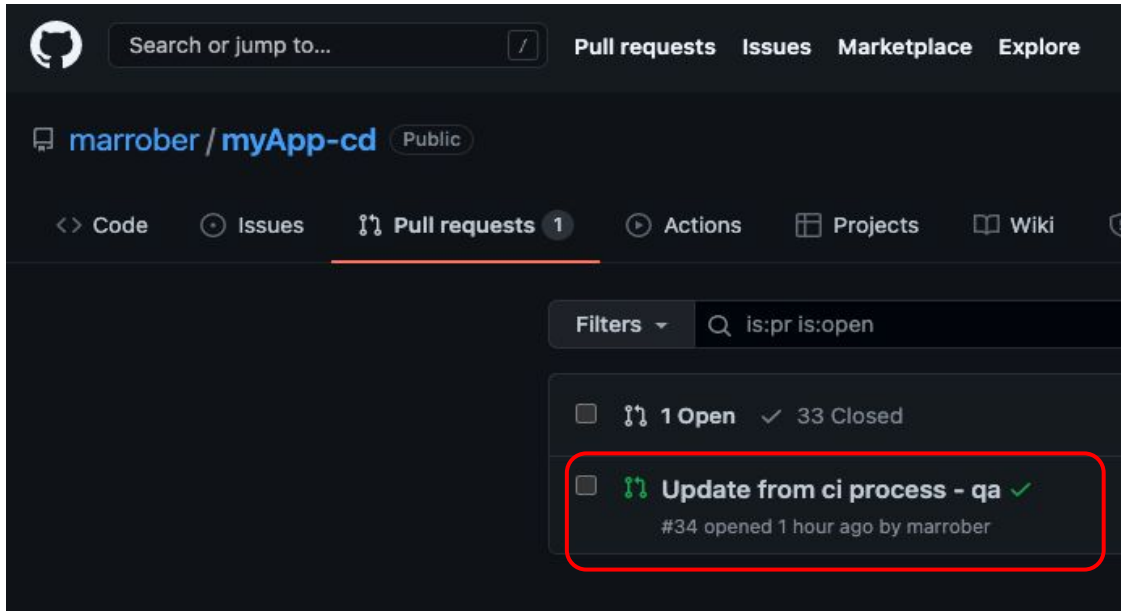
Pull request detail showing that it cannot be merged yet



Red Hat

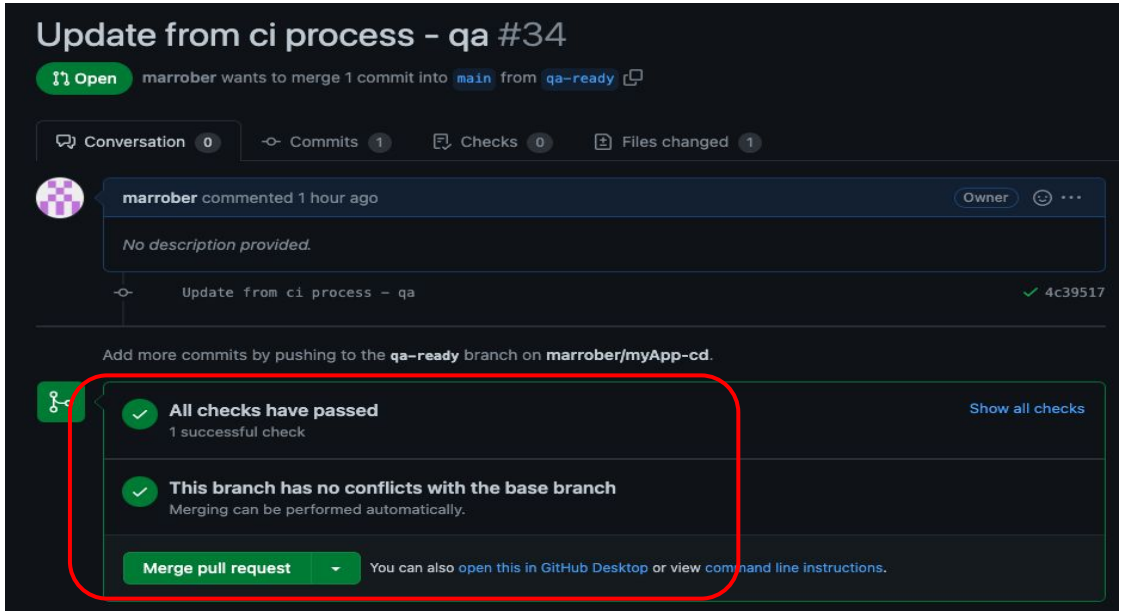
Pipeline in action - after resources have been approved

Update and commit Kustomize file - QA & create a pull request to merge changes to main branch ✔ → Set commit status to 'pending' ✔ → Git clone resources (qa-ready branch) ✔ → Configure deployment assets ✔ → Resource deployment check ✔ → Set commit status to 'success' ✔



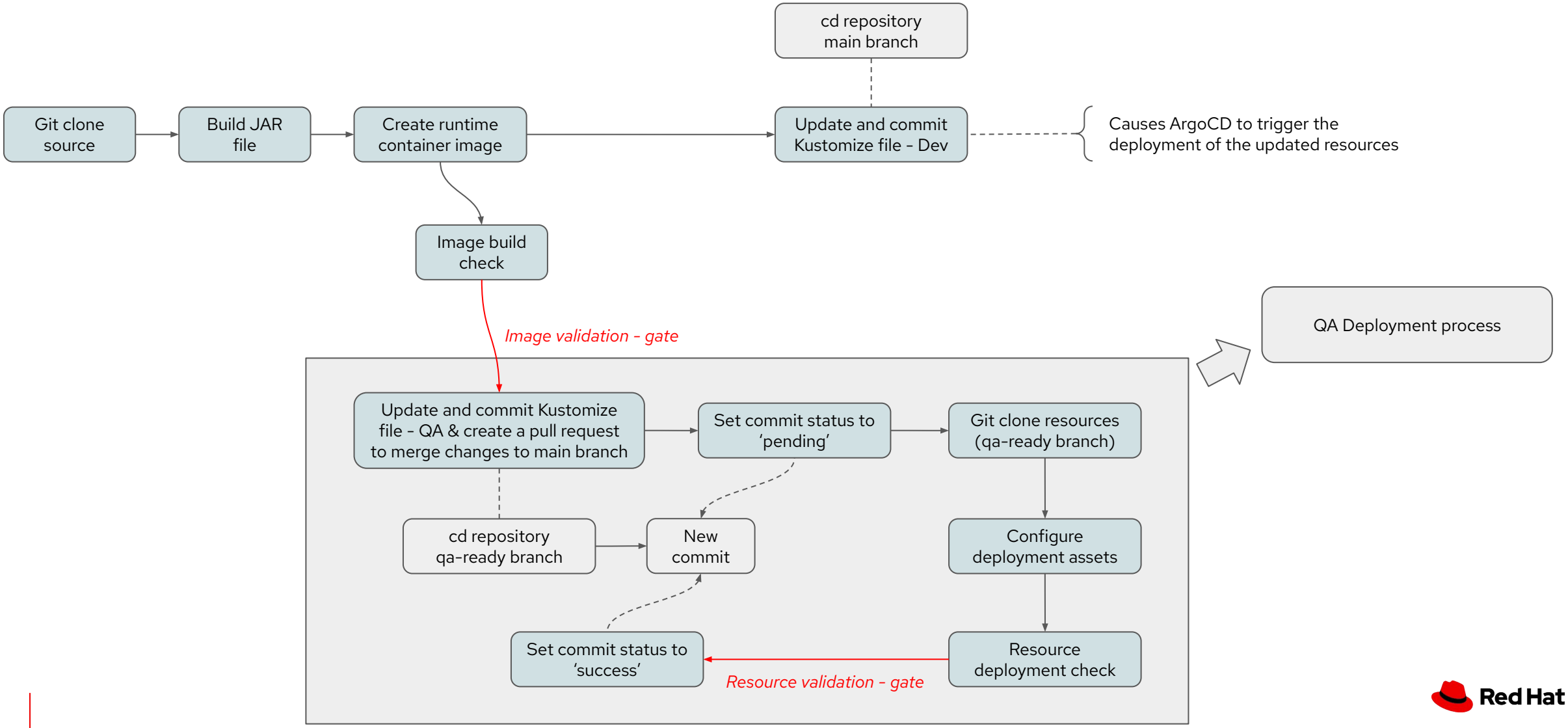
Pull request summary with green 'success' indicator

Pull request detail showing that it can be merged yet

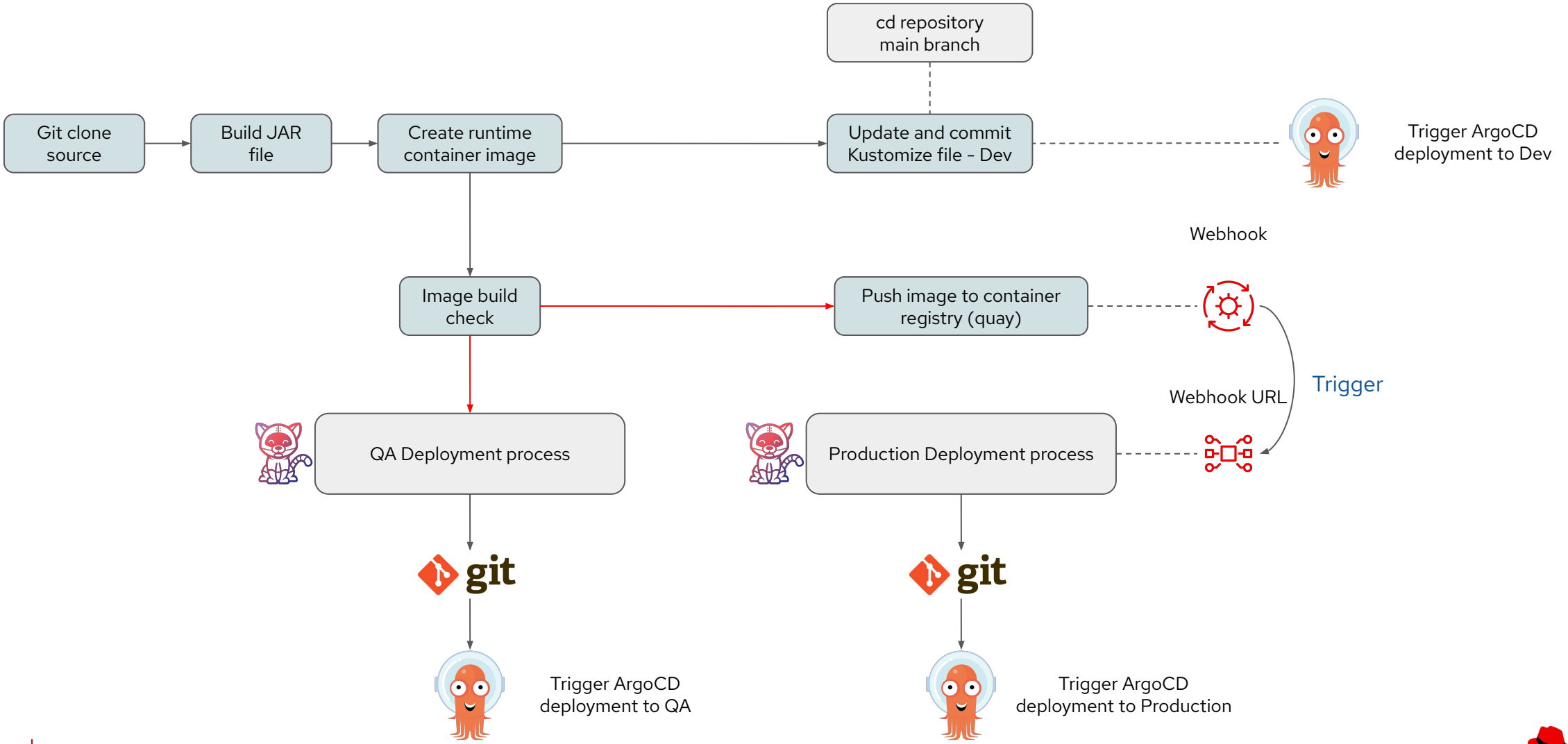


Red Hat

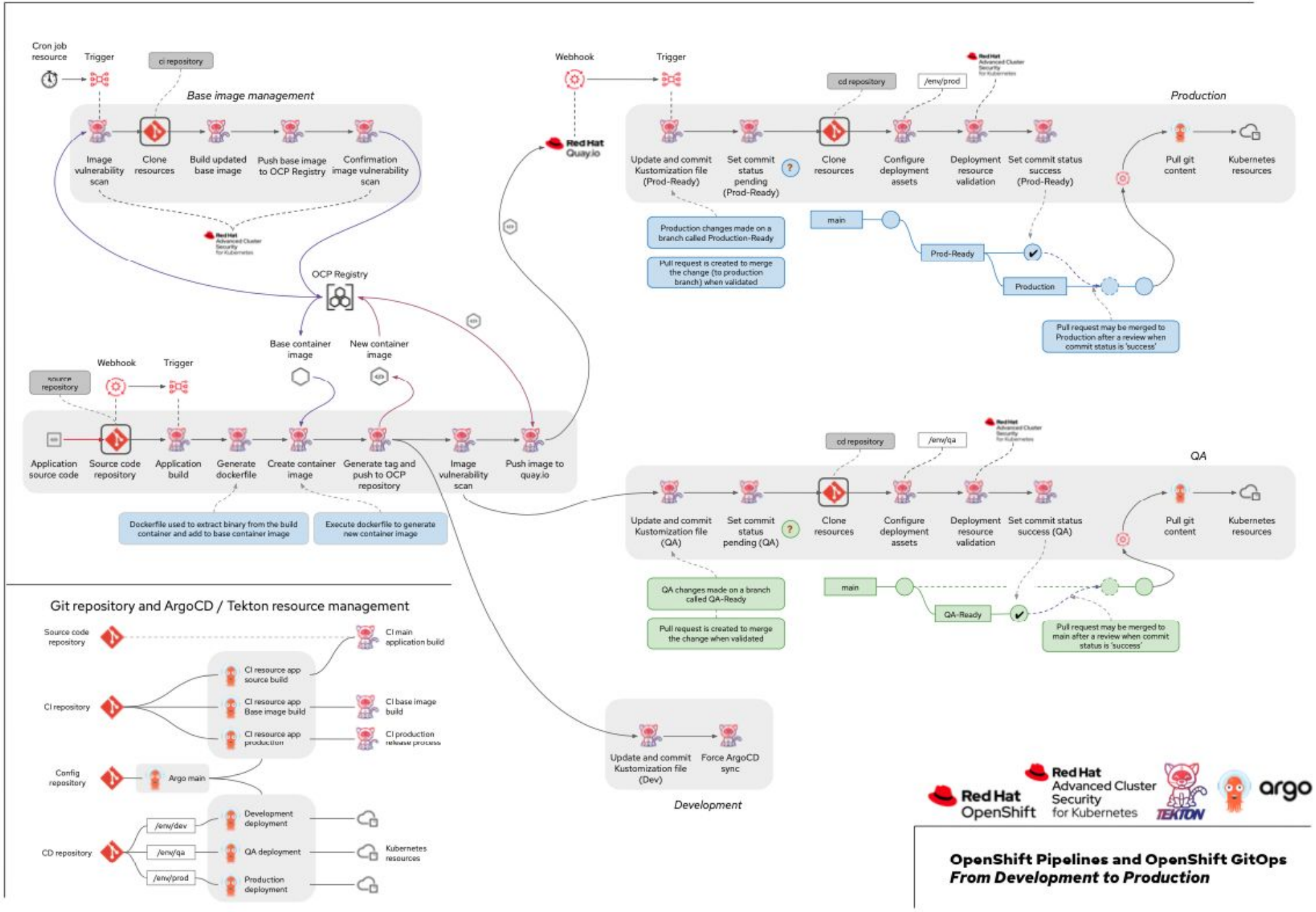
QA pipeline – Use a pull request to manage the release



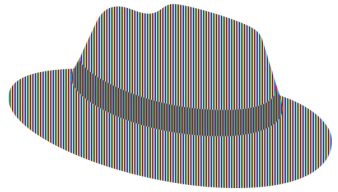
Extending the pipeline to production



Red Hat


Cron job resource
Trigger
ci repository
Base image management
Image vulnerability scan
Clone resources
Build updated base image
Push base image to OCP Registry
Confirmation image vulnerability scan
OCP Registry
Base container image
New container image
Webhook
Trigger
source repository
Application source code
Source code repository
Application build
Generate dockerfile
Create container image
Generate tag and push to OCP repository
Image vulnerability scan
Push image to quay.io
Dockerfile used to extract binary from the build container and add to base container image
Execute dockerfile to generate new container image
Webhook
Trigger
Red Hat Quay.io
cd repository
/env/prod
Production
Update and commit Kustomization file (Prod-Ready)
Set commit status pending (Prod-Ready)
Clone resources
Configure deployment assets
Deployment resource validation
Set commit status success (Prod-Ready)
Pull git content
Kubernetes resources
Production changes made on a branch called Production-Ready
Pull request is created to merge the change (to production branch) when validated
main
Prod-Ready
Production
Pull request may be merged to Production after a review when commit status is 'success'
cd repository
/env/qa
QA
Update and commit Kustomization file (QA)
Set commit status pending (QA)
Clone resources
Configure deployment assets
Deployment resource validation
Set commit status success (QA)
Pull git content
Kubernetes resources
QA changes made on a branch called QA-Ready
Pull request is created to merge the change when validated
main
QA-Ready
Pull request may be merged to main after a review when commit status is 'success'
Update and commit Kustomization file (Dev)
Force ArgoCD sync
Development
Git repository and ArgoCD / Tekton resource management
Source code repository
CI repository
Config repository
CD repository
CI main application build
CI resource app source build
CI resource app Base image build
CI resource app production
CI base image build
CI production release process
Argo main
/env/dev
/env/qa
/env/prod
Development deployment
QA deployment
Production deployment
Kubernetes resources
Red Hat OpenShift
Red Hat Advanced Cluster Security for Kubernetes
TEKTON
argo
OpenShift Pipelines and OpenShift GitOps
From Development to Production
Red Hat

Summary

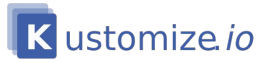


Red Ha
OpenS

OpenShift Pipelines delivers a cloud-native continuous integration process

OpenShift GitOps delivers a deployment automation process driven by content in Git repositories

Git provides a secure source code repository in which branch protection rules govern when content is merged to branches



Kustomize provides a controlled mechanism to update and patch Kubernetes resources for each environment

Most important *: A structured and controlled process that is understood by the team*

Red Hat

Red Hat Summit

Thank you


linkedin.com/company/red-hat


facebook.com/redhatinc


youtube.com/user/RedHatVideos


twitter.com/RedHat

Red Hat