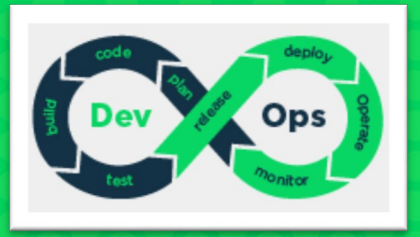
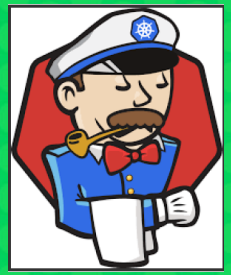


DevOps Pipelines with Jenkins X on SUSE CaaS Platform

Best Practices Basics

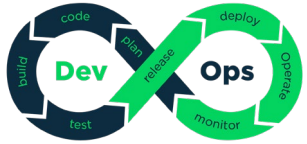
Jean Marc Lambert & Stephane Nguyen
EMEA SUSE Consulting

JENKINS X



https://jenkins-x.io/

Landscape for DevOps and Containers



DevOps teams:

- **Love containers**
- **Start to adopt « micro-services » architectures**
- **Need automated CI/CD pipelines and more…**
- **Require multiple isolated environments (Dev/Stag/Prod)**
- **Dream about developing in a « iso-prod » environment**
- **Need global configuration management of their deployed workloads**

Kubernetes is today's best-of-breed container orchestration ecosystem (especially SUSE CaaS Platform).

Jenkins X answers most of the DevOps requirements above.

Jenkins X Introduction

A reimagined **CI/CD implementation for the cloud**

Based on the **state of the art in DevOps**

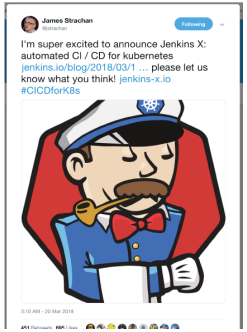
And the “Accelerate” book recommendations

STARTED BY JAMES STRACHAN

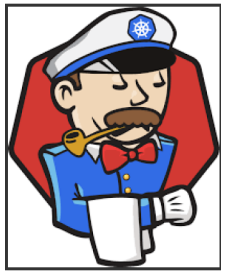
(ex RedHat, now CloudBees)

Evolution of Fabric8 (2015-2017)

JX 1st Release by March 2018



How Jenkins X Solves All of These



Jenkins X provides:

- **JX CLI tool to interact with**
- **Kubernetes deployment and configuration tool (multi-clouds)**
- **« CI/CD/DevOps » tooling using Git, Jenkins, Skaffold, Helm…**
- **Simple/efficient continuous delivery and promotion workflow**
- **Automation setup of various environments (Dev/Stag/Prod…) in Git and Jenkins, with relevant pipelines and integration points**
- **Multiple languages pipelines and quickstarts (Go, Java, Node, Python…)**
- **Environments to develop in an Iso-Prod mode**

kubernetes

- ***Based on open source components***

Which Open Source Components Are in Jenkins X?

CI/CD

Jenkins

CI/CD pipeline solution

Config

Git *family*

Code Versioning/Config

Build

Draft (draft.sh)

Applications build packs

Skaffold

Docker images builder on Kubernetes

Deploy

Helm

Package manager for Kubernetes

Chartmuseum

Helm chart repository

Monocular

Web UI for Helm charts

Kubernetes

Container orchestrator

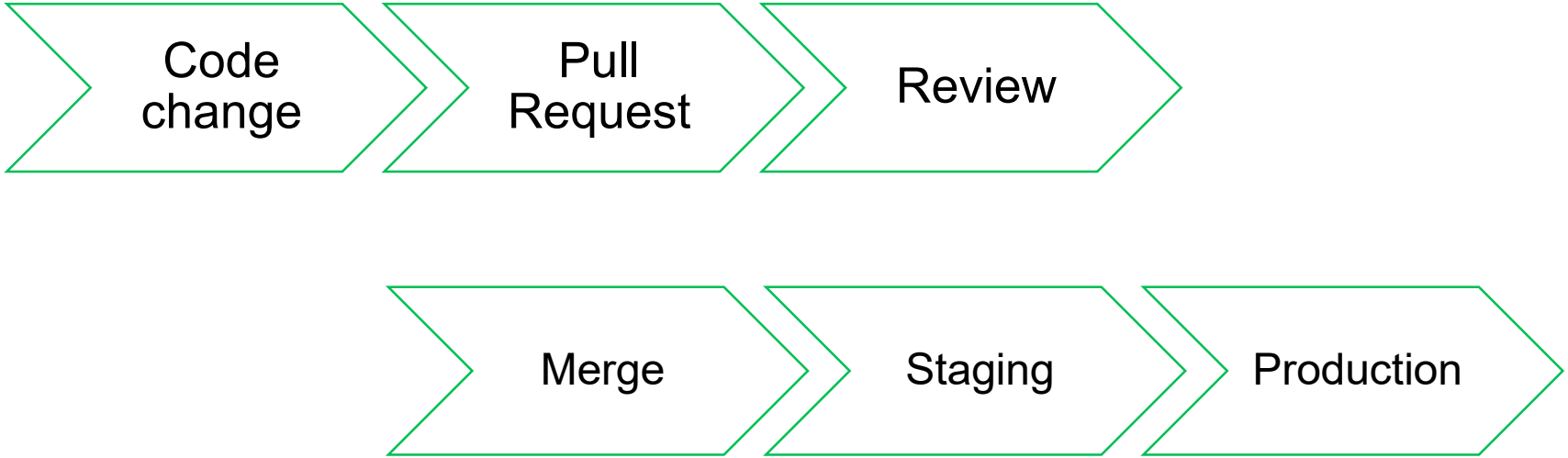
Store

Nexus

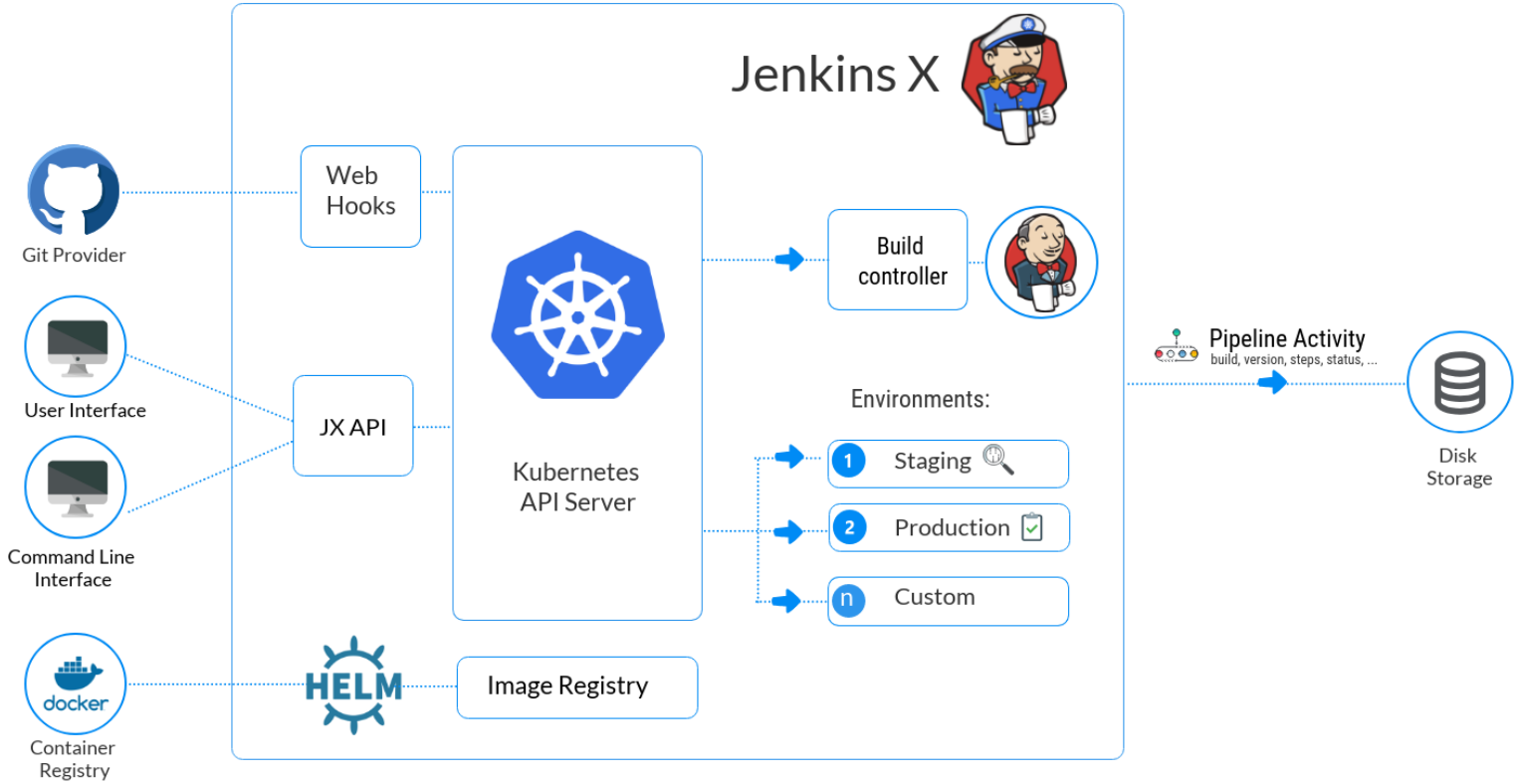
Artifact repository

kubernetes

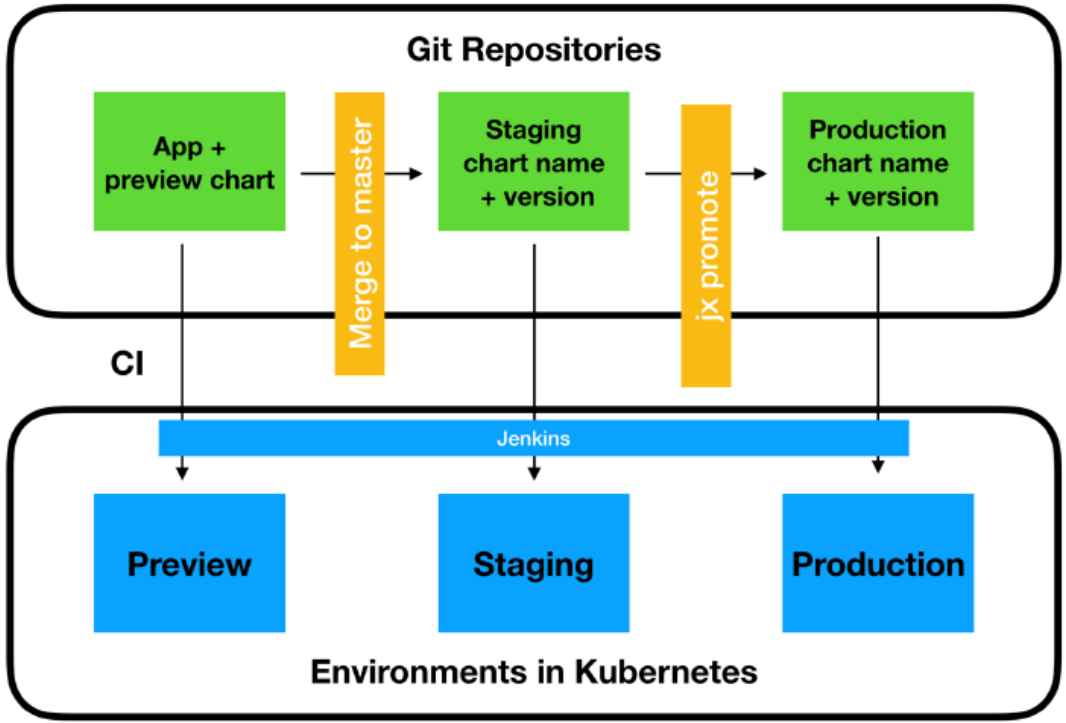
Jenkins X Supports What We Do!



Architecture of Jenkins X

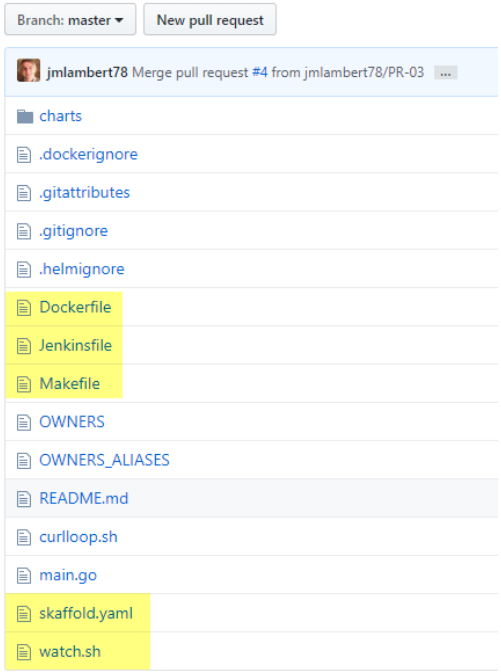


Overview of Git + Jenkins X environments



Structure of a Jenkins X Supported Project

In the Application Project Git



Files used by JX

Dockerfile: image specification

Jenkinsfile: pipeline code

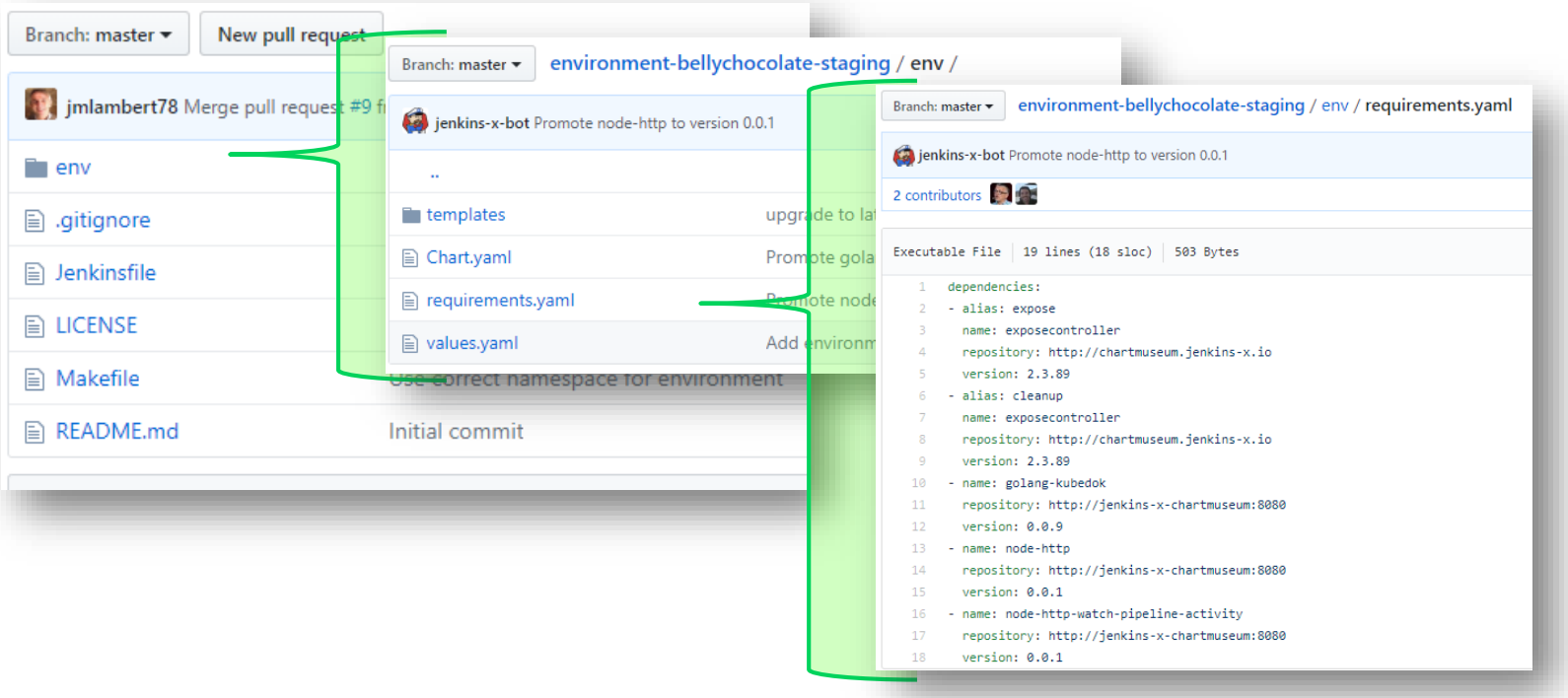
Skaffold.yaml: how to build

Makefile: actions used in the pipelines

Charts: Helm charts for app

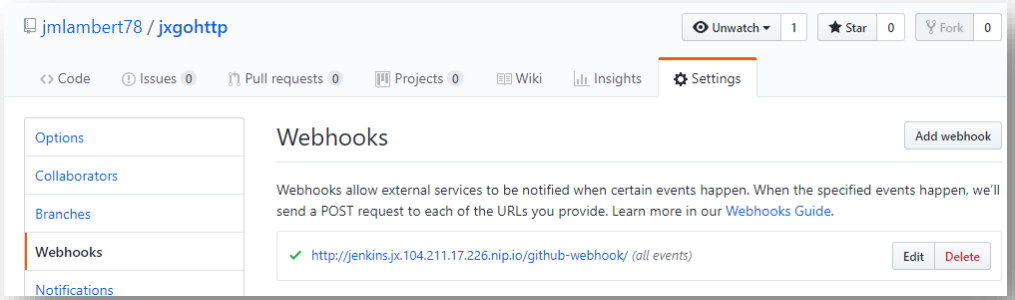
JX Deployment Environment Stored in Git

Example: Staging environment

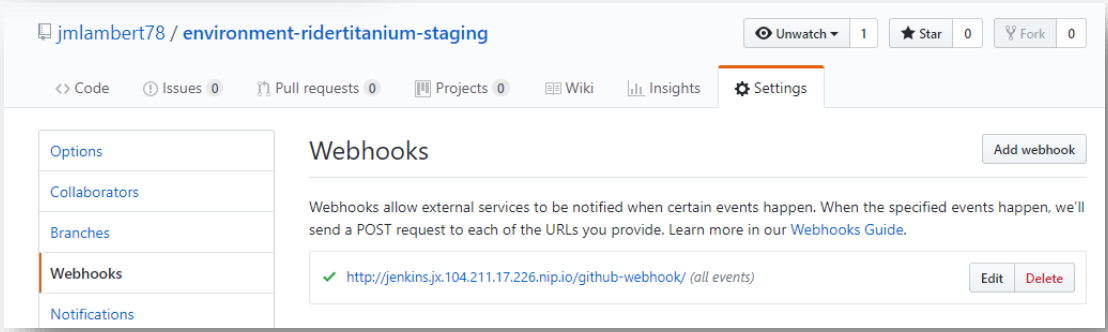


Git Webhooks to Jenkins Instances

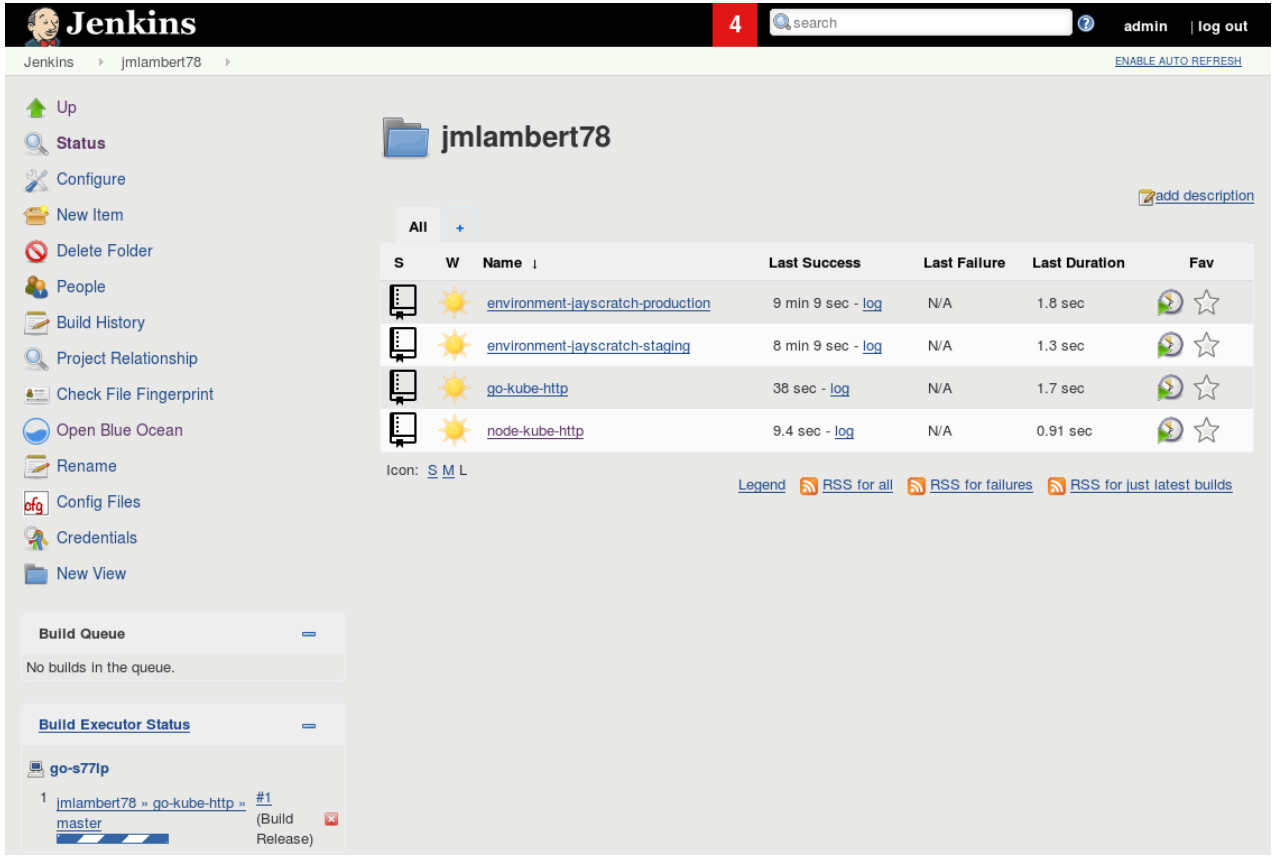
App side



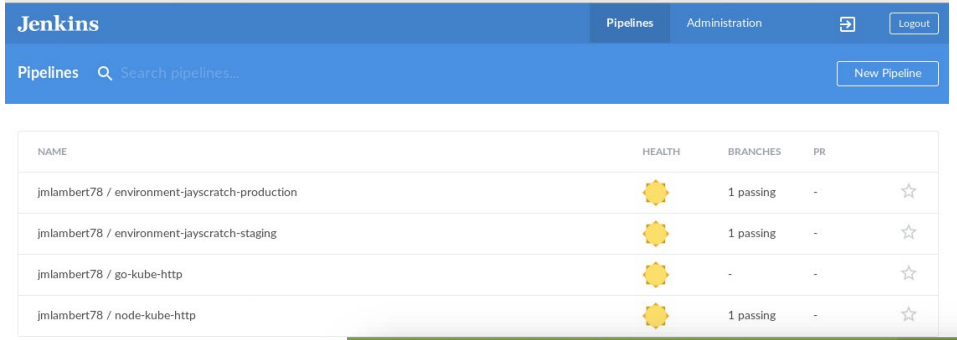
Env(s) side

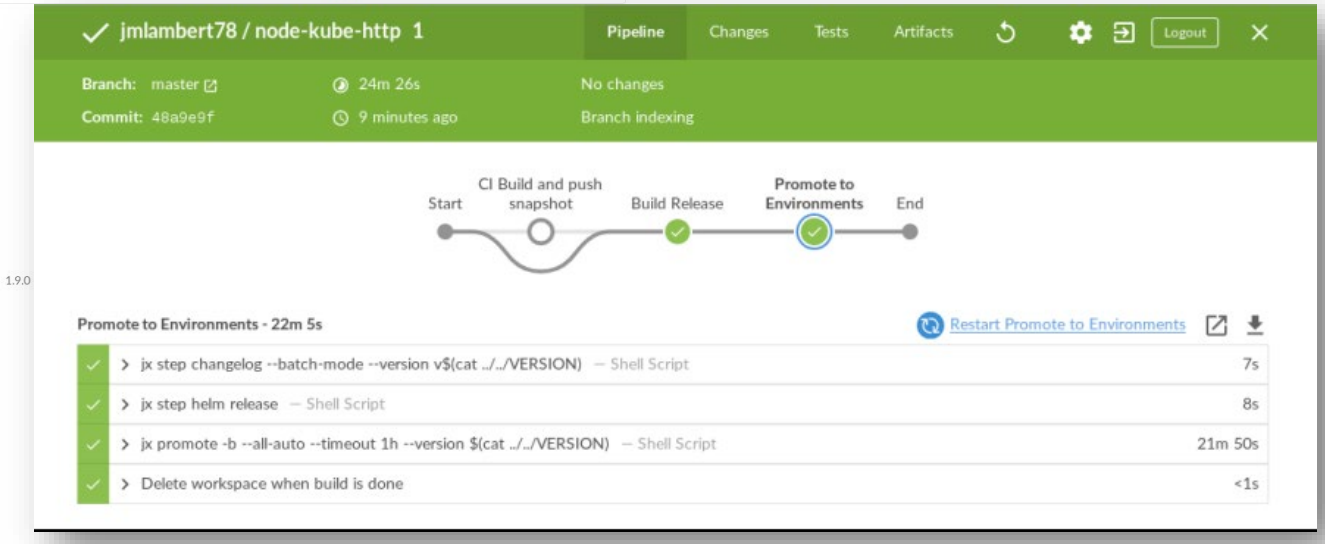


Jenkins Deployed and Pipelines Set Up

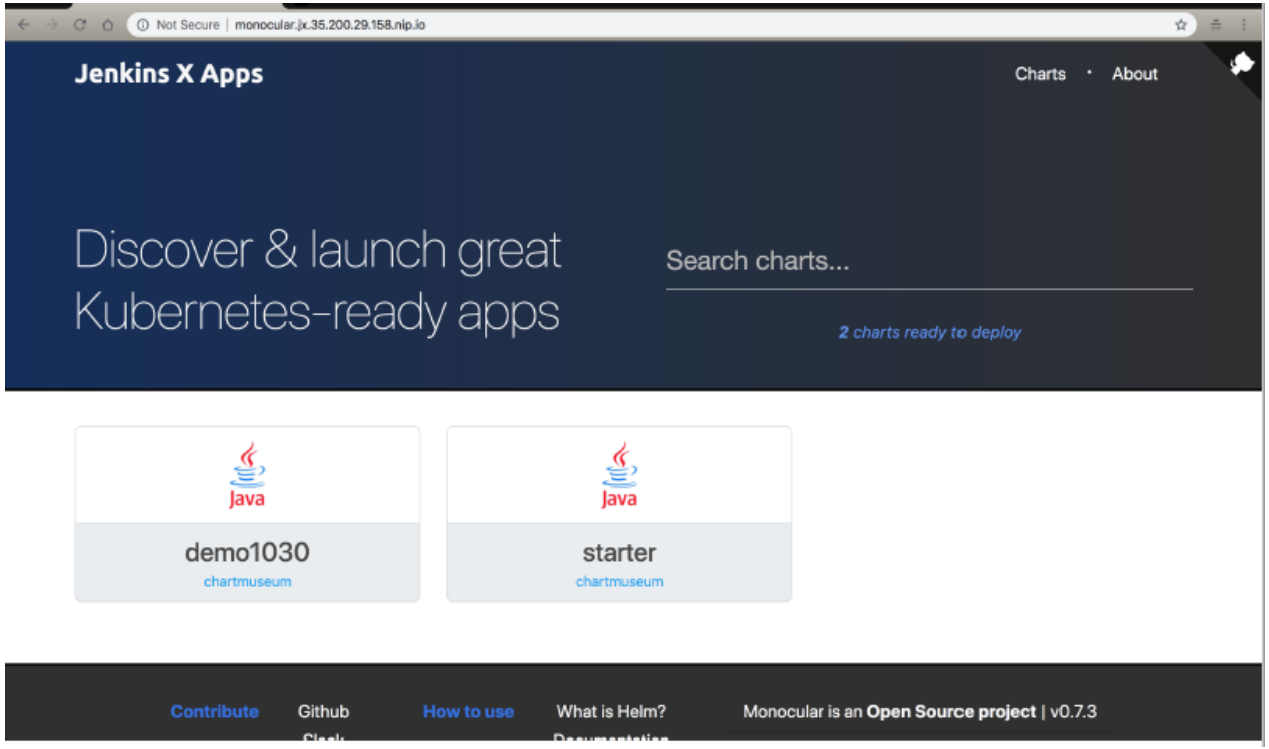


Jenkins Pipelines Views





Monocular GUI: Helm Charts Repository



How Does That Work on the SUSE CaaS Platform?

What Is the SUSE CaaS Platform?

- A Kubernetes cluster with an additional layer of management
- Currently at version 3

CaaS Platform – Kubernetes

- **Supports Docker and CRI-O as container runtimes**

- **Provides a “vanilla” Kubernetes**
  - Supports all features that Kubernetes 1.10 supports

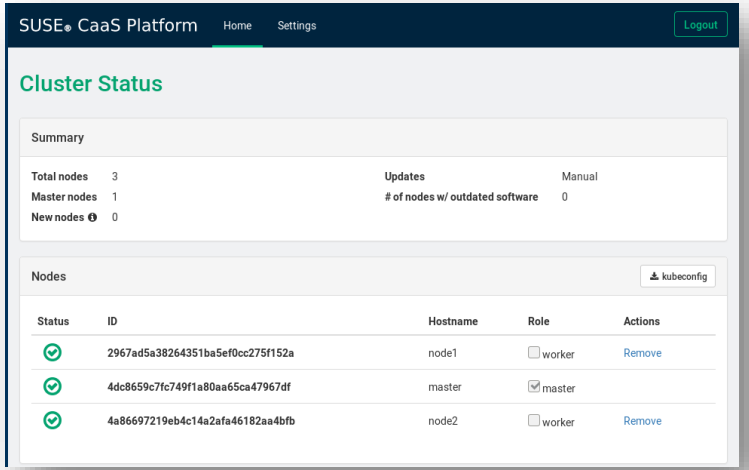
- **Can run on-premises, on physical servers or virtual machines, and also in the Public Cloud**

- **RBAC is the de facto authorization mechanism**

- **Can integrate with external LDAP/Active Directories for users and groups management**

CaaS Platform – The Management Layer

- **Supports deployment and ongoing CaaS Platform management**
  - Automated cluster bootstrapping
  - Patch management with no service interruption
  - Certificate distribution, private registries addition, external LDAP integration, etc.
  - User tokens management
  - Orchestration performed by Salt
- **Provides a graphical user interface to perform these tasks: Velum**
- **Runs on a dedicated server: the Administration Server**



CaaS Platform – Typical Configurations

- **Minimal**
  - 1 Administration Server
  - 1 Master node
  - 2 Worker nodes

- **Typical Configuration**
  - 1 Administration Server
  - 3 Master nodes
  - 2..n Worker nodes

- **Persistent Storage**
  - Dynamically or statically provisioned NFS PVs
  - Dynamically provisioned Ceph RBD PVs

Jenkins X Prerequisites – On-Premises

- **Kubernetes cluster (SUSE CaaS Platform) with RBAC enabled**
  - Size, 3 VMs: 60Gb disk, 2-3v CPU, 8Gb RAM
- **INGRESS router**
  - Deployed with Helm on the Kube
- **Fixed IP for the Ingress**
- **Wildcard DNS entry for this Ingress IP**
  - Reachable from github to allow the webhooks to operate
- **GITHUB account (or git family)**
- **Storage-class: to create on-demand Persistent Volumes (PV)**
  - « NFS Client Provider » can make the job or « **SUSE Enterprise Storage » RBD**
- **Docker Setup:** *(if using jx deployed registry)*
  - DOCKER_OPTS="--insecure-registry <clusterRange.0.0/16"

Deployment of Jenkins X on SUSE CaaS Platform

Install the command line tool:

jx

http://jenkins-x.io/

If you have a K8S cluster already

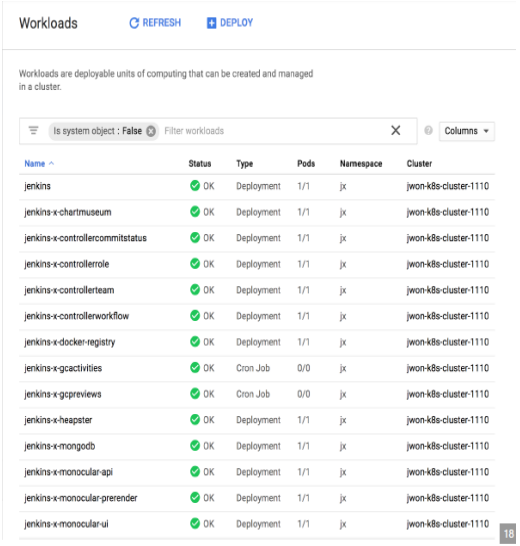
- Ensure RBAC is enabled and then:

jx install --provider=kubernetes --on-premise --skip-ingress

What Do You Get Out of That?

- Automates the **installation/upgrade of tools**
  - Helm, Skaffold, Kaniko, Jenkins, KSync, Monocular, Nexus, etc.
  - All configured + optimized for Kubernetes OOTB

Jenkins X components running in the CaaS



Staging + Prod Envs in Github

Git staging and production repos

- Ready to welcome new apps

Jenkins setup

- Pipelines set up for these repos

WebHooks

- Set in Github to trigger Jenkins on change or PR

What Does That Give Me?

- **The Jenkins X CLI tool integrates a lot of features (deploy, logs, etc.)**
- **Ability to work with multiple k8s contexts (clusters)**

Each team gets its own:

- **Development Tools Environment**
  - Jenkins master
  - Elastic pool of Kubernetes build pods
  - Nexus for artifacts
  - ChartMuseum + Monocular (Helm application store + API + UI)
  - Private Docker registry

- **Staging + Production Environment (default)**
  - Both environments configured in Git for version control and triggers to Jenkins on changes

Jenkins X Components

Environments / Kubernetes Namespaces

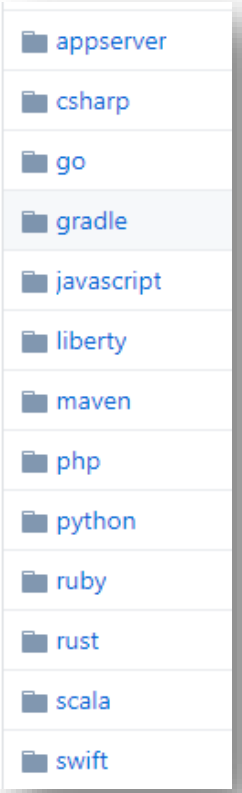
- **1 Dev Env per team/project**
- **0-N Permanent Environments (e.g.: staging, production)**
  Each permanent env is managed within Git & PR to allow traceable changes / Rollbacks
- Optional preview environments allocated to preview PRs

Components deployed by Jenkins X

- **Jenkins**
- **Nexus**
- **Docker registry**
- **Chartmuseum (registry for helm chart)**
- **Monocular: UI for discovering/running Helm charts**

Jenkins X Provides Languages Buildpacks

- **For each supported framework, a buildpack includes:**
  - Dockerfile
  - Jenkinsfile
  - Helm chart (used to deploy it on k8s)
  - Chart/preview Helm chart used to deploy in a preview environment
- **You may write your own if it's not yet available**
- **A set of pod templates is provided**
  - To **execute the Jenkins pipeline** itself
  - Including all the tools required by a standard pipeline of the framework

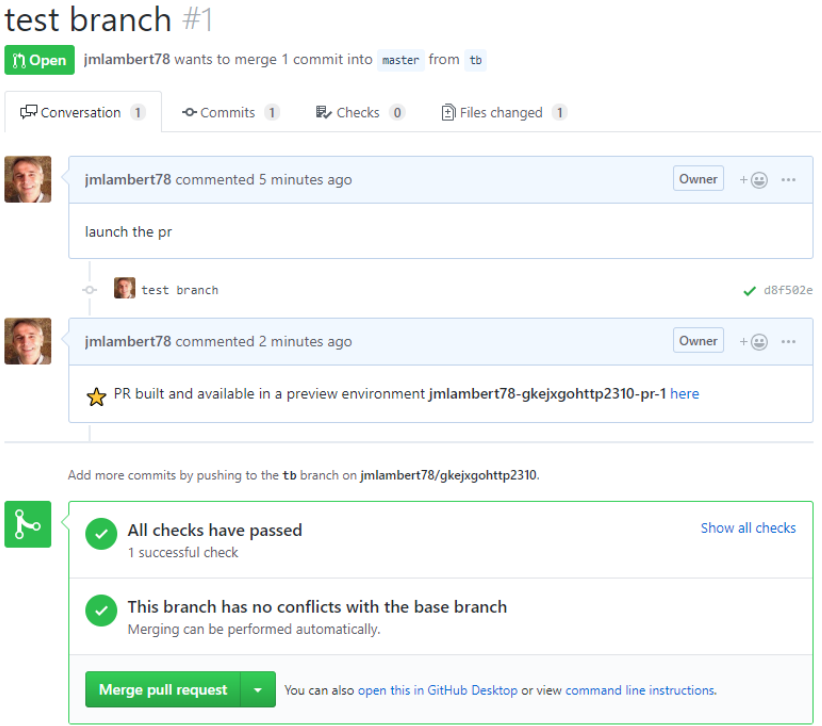


Jenkins X CLI: Create/Import Projects

- **Available commands:**
  - jx create spring
  - jx create quickstart
  - jx import
- **Automatically set up CI/CD pipelines for new + imported projects**
  - Set up git repository
  - Adds webhooks on git provider to trigger Jenkins pipelines on PR / master
  - Triggers first pipeline

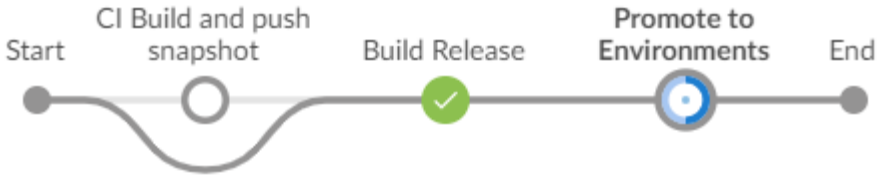
Jenkins X: On Pull Request

- Compiles + runs tests
- Creates preview Docker image + Helm chart
- Creates a preview environment
- Comments on the PR in Git with a link to the running app



Jenkins X: On Release (Push to Master)

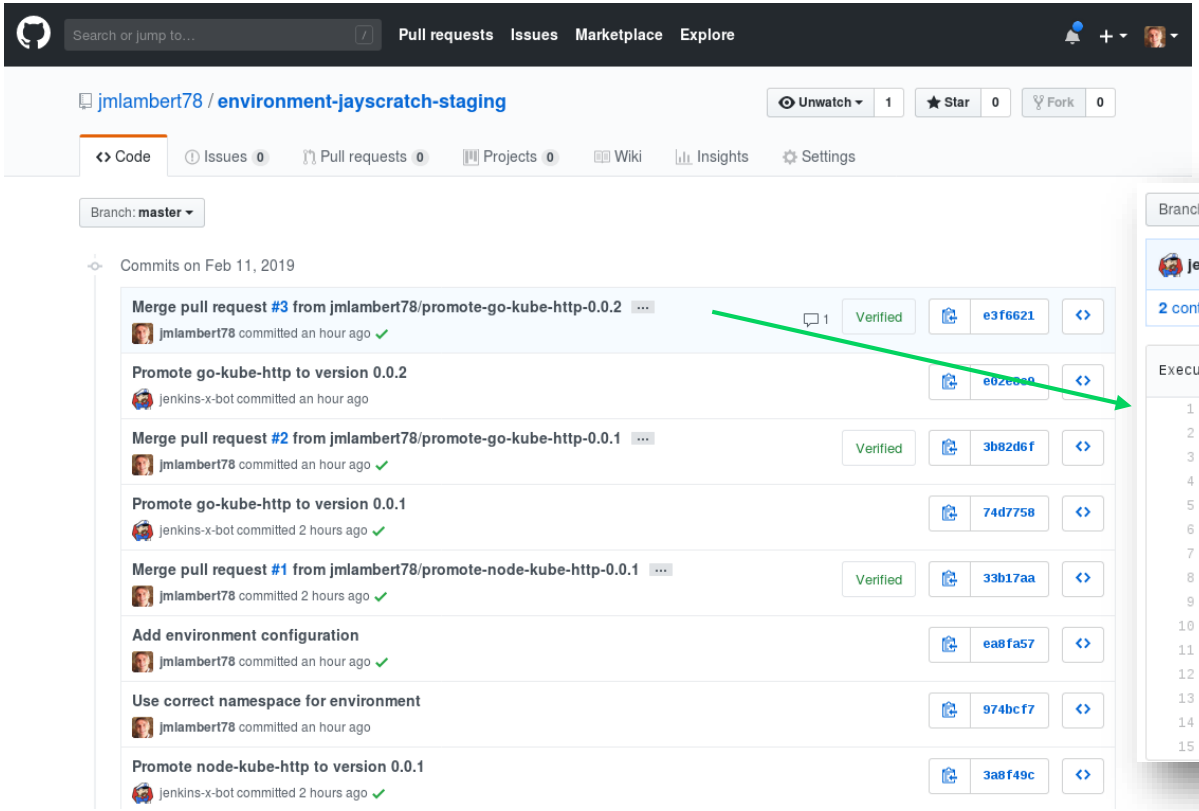
- Compiles + runs tests
- Creates semantic release version
- Publishes versioned artifacts, Docker image and Helm chart
- Promotes through all *automatic* environments, e.g., Staging

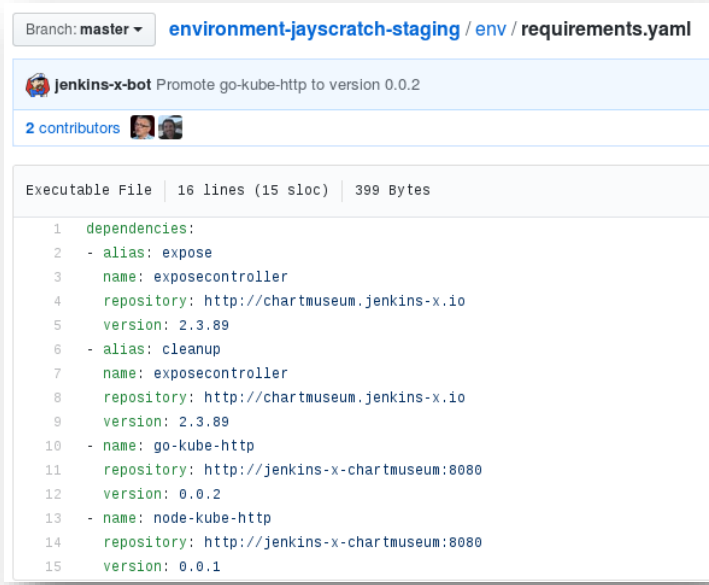


Jenkins X: Promotion via GitOps

- **Each environment stores its configuration as Helm charts in a Git repository**
  - Configuration as code
  - All changes audited and easy to revert
  - Reuse the Pull Request workflow for changes
- **To promote a version to Production, for example, Jenkins X submits a Pull Request**
  - The Promote step waits for the Pull Request CI build to complete and merge, and for the environments pipeline to finish applying the change

Example: Staging Environment History (Git)





The Cherry on Top: DevPod to Speed up the Development Cycle

- **Allows the developer to use the same environment/tooling as the CI/CD pipelines on Kubernetes**
- **Lets you build, run tests or redeploy apps before you commit to Git, safe in the knowledge that you are using the same tools as the CI/CD pipelines!**
- **Developer experience**
  - Works on its own IDE (Eclipse, Vscode, etc.)
  - Each file save is reflected (**ksync**) within the DevPod and triggers a local rebuild and Docker image production through **scaffold** and deployment
  - Dev may also ~ssh to the DevPod to run his investigation or debug
  - No need to install the exact same tools on the Dev workstation

Environment:

- **SUSE CaaS Platform already deployed**
- **Jenkins X installed with *jx install –provider=kubernetes***

Workstation

- **jx client available**
- **kubectl credential available**

DEMO TIME

Jenkins X Usage Demo

- **Create a new project**
  - jx create quickstart –f go
- **Check the activities triggered automatically**
  - jx get activity -f <projectname> -w
- **Get the Apps urls**
  - jx get urls
- **Connect Jenkins GUI to check progress**
  - Jx console http://jenkins.jx.35.224.16.222.nip.io
    admin/shieldtyphoon
- **Branches and PR auto review**
- **Update code and perform a full process from source to production**
- **Demo the development process within Kubernetes (faster cycles) and devpod**


SUSE
We adapt. You succeed.