

# Front Matter: Next Era Front End Deployments on OpenShift 4

Lance Ball  
Principal Software Engineer

New York, NY  
August 2019



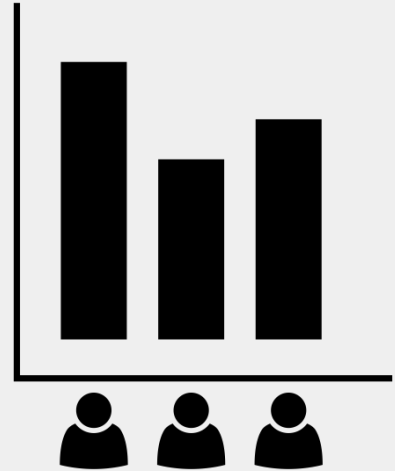
# Lance Ball

- Principal Software Engineer
- Tech Lead - OpenShift Cloud Functions
- Twitter: @lanceball
- Budding ukulele performer



# A Quick Poll

- Do you deploy apps on Openshift today?
- Do you write Node.js apps?
- Do you write Single Page Applications?
- Anyone here “full stack”?
- DevOps People?

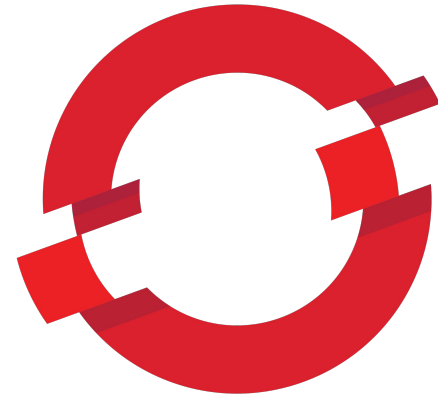


Created by Bakunetsu Kaito  
from Noun Project

## Did You Know?

Number one deployment runtime on Openshift Online is Node.js

- But they're not all actually Node.js Applications
- Many deployments are Single Page Applications
- How are these applications being deployed?
- What tools can I use in my workflow?



# OPENSIFT

---

# Let's Build a Web App!

## Create a React Application

```
$ npx create-react-app mywebapp
```

```
$ cd mywebapp
```

```
$ npm start
```

```
# Edit App.js and watch it reload
```

---

OK - Let's Deploy It!

# But How?

OpenShift NGINX

Template via the Catalog



## Use the Developer Catalog

### Developer Catalog

Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. Cluster admins can install additional apps which will show up here automatically.

All Items

All Items

Languages

13 items

Middleware

Other

Filter by keyword...

TYPE

- Service Class (0)
- Source-to-Image (10)
- Installed Operators (3)

 .NET

.NET Core

Build and run .NET Core 2.2 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations.



Apache HTTP Server (httpd)

Build and serve static content via Apache HTTP Server (httpd) 2.4 on RHEL 7. For more information about using this builder image, including



Knative Eventing

provided by Red Hat

Represents an installation of a particular version of Knative Eventing



Knative Serving

provided by Red Hat

Represents an installation of a particular version of Knative Serving

 NGINX

Nginx HTTP server and a reverse proxy (nginx)

Build and serve static content via Nginx HTTP server and a reverse proxy (nginx) on RHEL 7. For more information about

 node

Node.js

Build and run Node.js 10 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations.



OpenShift Pipelines Config

provided by Red Hat

OpenShift Pipelines is a cloud-native CI/CD solution for building pipelines using Tekton concepts which run



Perl

Build and run Perl 5.26 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations.



PHP

Build and run PHP 7.2 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations.



Python

Build and run Python 3.6 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations.



Red Hat OpenJDK 8

Build and run Java applications using Maven and



Ruby

Build and run Ruby 2.5 applications on RHEL 7. For



Tech Preview - Modern Web Applications

Build and run Modern Web

OpenShift Template

Use the Developer Catalog

<https://github.com/sclorg/nginx-ex/blob/master/openshift/templates/nginx.json>

## Create Source-to-Image Application

Namespace \*

msa-day-ny

Version \*

nginx:1.12

Name \*

Names the resources created for this application.

Git Repository \*

[Try Sample](#)

For private Git repositories, create a [source secret](#).

Create route

Exposes your application at a public URL.

Create Cancel

### NGINX Nginx HTTP server and a reverse proxy 1.12

BUILDER NGINX

Build and serve static content via Nginx HTTP server and a reverse proxy (ng including OpenShift considerations, see <https://github.com/sclorg/nginx-coi>

Sample repository: <https://github.com/sclorg/nginx-ex.git>

The following resources will be created:

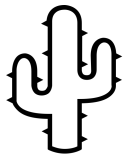
- A **build config** to build source from a Git repository.
- An **image stream** to track built images.
- A **deployment config** to rollout new revisions when the image changes.
- A **service** to expose your workload inside the cluster.
- An optional **route** to expose your workload outside the cluster.

## OpenShift Template

# But Wait!

This doesn't seem right  
for a developer's  
workflow

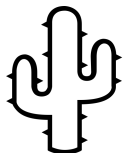
## Two Thorns and a Rose



Created by Chris Terrell from Noun Project

**OpenShift builds pull from Git repo**

Not ideal for iterative development



Created by Chris Terrell from Noun Project

**Must maintain compiled artifacts in Git repo**

The “build” for a web server image expects static content




Created by mrsreppening from Noun Project

**Served by a real HTTP server**

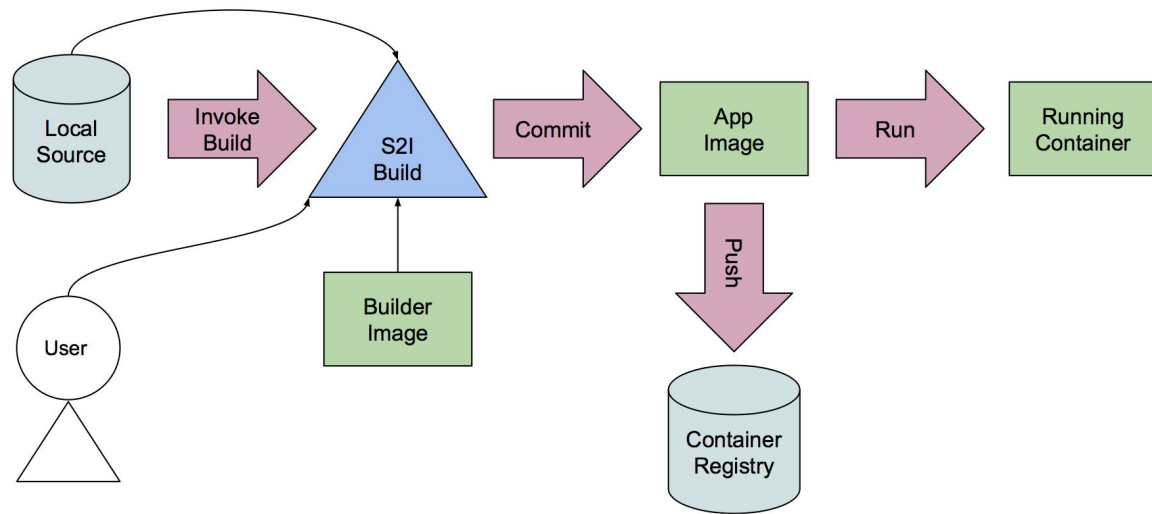
We can use Apache or NGINX and don't depend on a React Node.js server

---

# Nodeshift



# Source to Image a.k.a. S2I



Source to Image

## So What's Nodeshift?



**An npm module for deploying Node apps on OpenShift**

Creates, builds, routes and deploys your app in one command



**Great for local development environments**

Deploys directly from the file system



**Layered application images via s2i**

Overlays application on a base image, creating a new application image



## Node.js REST Server via Express

```
$ npx express-generator nodejs-example
```

```
$ npx nodeshift --expose --deploy.port=3000
```

---

But that's a Node.js app,  
not a React SPA

---

# Web Application S2I Image Builder

## Deploy and Run With Development Server

```
$ npx nodeshift
  --dockerImage=nodeshift/centos7-s2i-web-app
  --imageTag=10.x
  --expose
  --deploy.port=3000
  --deploy.env NPM_RUN="npm start"
```

# Synchronize Development Changes in Real Time

## Synchronize Changes in Development

```
$ oc rsync --no-perms=true \  
  --watch src/ <POD_NAME>:src/
```

## Two Roses and a Thorn



Created by mynarspeing  
from Noun Project

**Single command deployment & live updates**

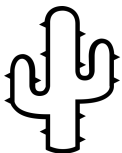
Easy to integrate into a development workflow



Created by mynarspeing  
from Noun Project

**Deploys from the local filesystem**

No need to push small changes in development to Github



Created by Open Terms  
from Noun Project

**Serves content using the React server**

Not designed for production use

# Production Deployments



Also known as Tekton

# OpenShift Pipelines



# Pipelines

*Specifying a workflow*

# Pipeline

```
apiVersion: tekton.dev/v1alpha1
kind: Pipeline
metadata:
  name: webapp-deployment-pipeline
spec:
  resources:
    - name: build-image
      type: image
    - name: runtime-image
      type: image
```

## Pipeline (cont)

```
spec:
  tasks:
  - name: build-runtime
    taskRef:
      name: webapp-build-runtime
    resources:
      inputs:
      - name: image
        resource: build-image
      outputs:
      - name: image
        resource: runtime-image
```

# PipelineResource

*Workflow inputs and outputs*

## PipelineResource

```
apiVersion: tekton.dev/v1alpha1
kind: PipelineResource
metadata:
  name: webapp-build-image
spec:
  type: image
  params:
  - name: url
    value:
image-registry.openshift-image-registry.svc:5000/msa-day-ny/
mywebapp
```

## PipelineResource

```
apiVersion: tekton.dev/v1alpha1
kind: PipelineResource
metadata:
  name: webapp-prod-image
spec:
  type: image
  params:
  - name: url
    value:
image-registry.openshift-image-registry.svc:5000/msa-day-ny
/webapp-runtime
```

# Task

*Specifying a single job within a Pipeline*



## Task

```
apiVersion: tekton.dev/v1alpha1
kind: Task
metadata:
  name: webapp-build-runtime
spec:
  inputs:
    resources:
      - name: image
        type: image
    params:
      - name: SOURCE_PATH
        description: The location of the webapp source
        default: /opt/app-root/output
```

## Task (cont)

```
outputs:
  resources:
    - name: image
      type: image
  steps:
    - name: copy-source
      image: ${inputs.resources.image.url}
      workingdir: ${inputs.params.SOURCE_PATH}
      command: ['cp', '-Rvp', '${inputs.params.SOURCE_PATH}',
'/gen-source/build']
      volumeMounts:
        - name: gen-source
          mountPath: /gen-source
```

# TaskRun

*Tasks can be run independently of a Pipeline*

# TaskRun

```
apiVersion: tekton.dev/v1alpha1
kind: TaskRun
metadata:
  name: webapp-prod-build-taskrun
spec:
  # Use service account with git and image repo credentials
  serviceAccount: pipeline
  taskRef:
    name: s2i
```

## TaskRun (cont)

```
apiVersion: tekton.dev/v1alpha1
kind: TaskRun
metadata:
  name: webapp-prod-build-taskrun

spec:
  inputs:
    params:
      - name: BUILDER_IMAGE
        value: docker.io/nodeshift/centos7-s2i-web-app
      - name: PATH_CONTEXT
        value: src
```

## TaskRun (cont)

```
spec:
  inputs:
    resources:
      - name: source
        resourceSpec:
          type: git
          params:
            - name: url
              value:
                https://github.com/lance/pipeline-webapp-example
```

## TaskRun (cont)

```
spec:
  outputs:
    resources:
      - name: image
        resourceSpec:
          type: image
          params:
            - name: url
              value:
image-registry.openshift-image-registry.svc:5000/nyc-webapp
/webapp-prod
```

# PipelineRun

*Runs all of the Tasks defined in your pipeline, with parameterized resources*



## PipelineRun

```
apiVersion: tekton.dev/v1alpha1
kind: PipelineRun
metadata:
  name: webapp-prod-pipelinerun

spec:
  pipelineRef:
    name: webapp-deployment-pipeline
  trigger:
    type: manual
  serviceAccount: pipeline
```

## PipelineRun (cont)

```
resources:  
- name: build-image  
  resourceRef:  
    name: webapp-build-image  
- name: runtime-image  
  resourceRef:  
    name: webapp-prod-image
```

# Three Roses!



Created by mynameisjoe  
from Noun Project

**Single command deployment with local code**

Easy to integrate into a development workflow



Created by mynameisjoe  
from Noun Project

**Deploy and update from the local filesystem**

No need to push small changes in development to Github



Created by mynameisjoe  
from Noun Project

**Production runtime served by a real HTTP server**

We can use Apache or NGINX and don't depend on a React Node.js server

---

# One Last Thing

# Knative Service

*Exposing the runtime image*

## Service

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: production-webapp
  namespace: msa-day-ny
spec:
  template:
    metadata:
      labels:
        app: webapp
        tier: frontend
```

## Service (cont)

```
spec:
  spec:
    containers:
      - image:
image-registry.openshift-image-registry.svc:5000/msa-day-ny/
webapp-runtime
      ports:
        - containerPort: 8080
```


# Thank You


<https://docs.openshift.com/container-platform/4.1/welcome/>


<https://tekton.dev>

<https://knative.dev>

<https://github.com/lance/pipeline-webapp-example>

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