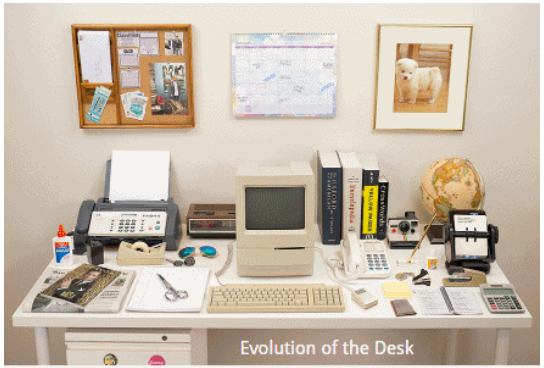
QCon NEW YORK

CI/CD Pipeline-as-code with Jenkins and Docker

June 15, 2016

Kishore Bhatia
@bhatiakishore



1980

vs.

2014

Docker Has Potential

- An example: **Software Configuration Management** Space





Docker Has Potential

- An example: **Software Configuration Management** Space

Docker Has Potential

- An example: **Software Configuration Management** Space

Docker Has Potential

- An example: **Software Configuration Management** Space

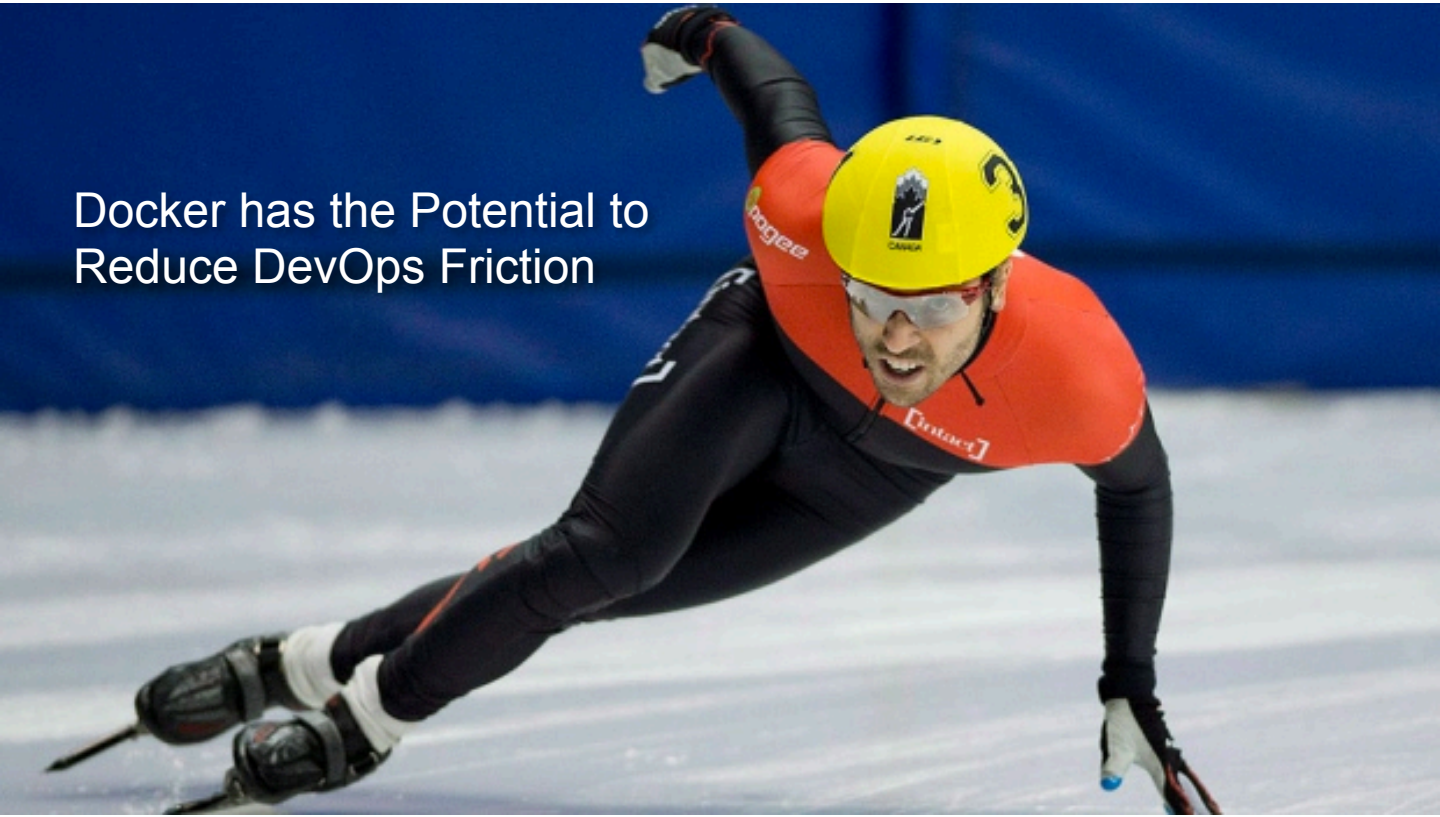



Docker Has Potential

- An example: **Software Configuration Management** Space

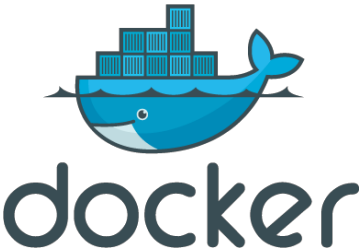
Docker Has Potential

- An example: **Software Configuration Management** Space

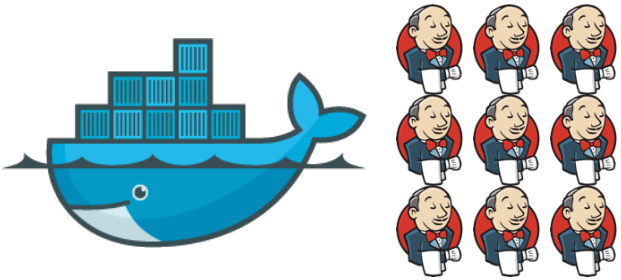

Docker has the Potential to
Reduce DevOps Friction

How Can You Use Jenkins & Docker Together?

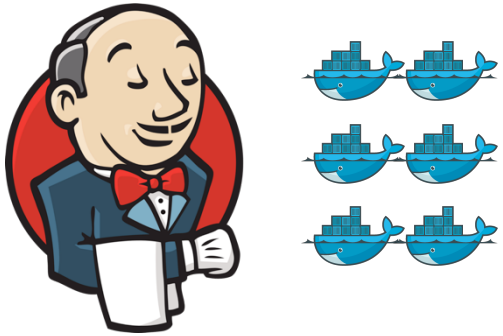
+



How Can You Use Jenkins & Docker Together?



1. Run Jenkins Masters & Slaves in Docker



2. Build, Test, & Deploy Docker Images from Jenkins

Oh, by the way...

*"First let me take a chance to familiarize you with how we test Docker... **We use Jenkins as our CI mostly because we needed a lot of flexibility and control.**"*

*"Obviously everything in our infrastructure runs in Docker, so that even goes for Jenkins. **We use the official image for our Jenkins container**."*

https://blog.jessfraz.com/post/dogfooding-docker-to-test-docker/

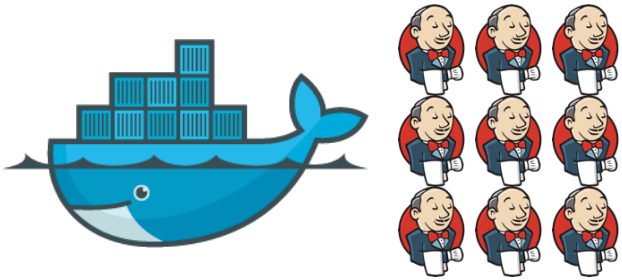
Feature Walkthrough

1. Run Jenkins Masters & Slaves in Docker

Docker (Cloud) – use Docker images as standardized build environments to improve isolation and elasticity

Docker Custom Build Environment – specify customized build environments as Docker containers

CloudBees Docker Shared Config – manage Docker (or Swarm) host configuration centrally in CloudBees Jenkins Operations Center

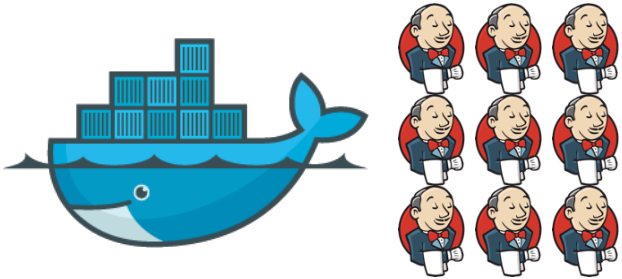


1. Run Jenkins Masters & Slaves in Docker

Docker (Cloud) – use Docker images as standardized build environments to improve isolation and elasticity

Docker Custom Build Environment – specify customized build environments as Docker containers

CloudBees Docker Shared Config – manage Docker (or Swarm) host configuration centrally in CloudBees Jenkins Operations Center



Docker Images

Official Docker Images at https://hub.docker.com/_/jenkins/ for:

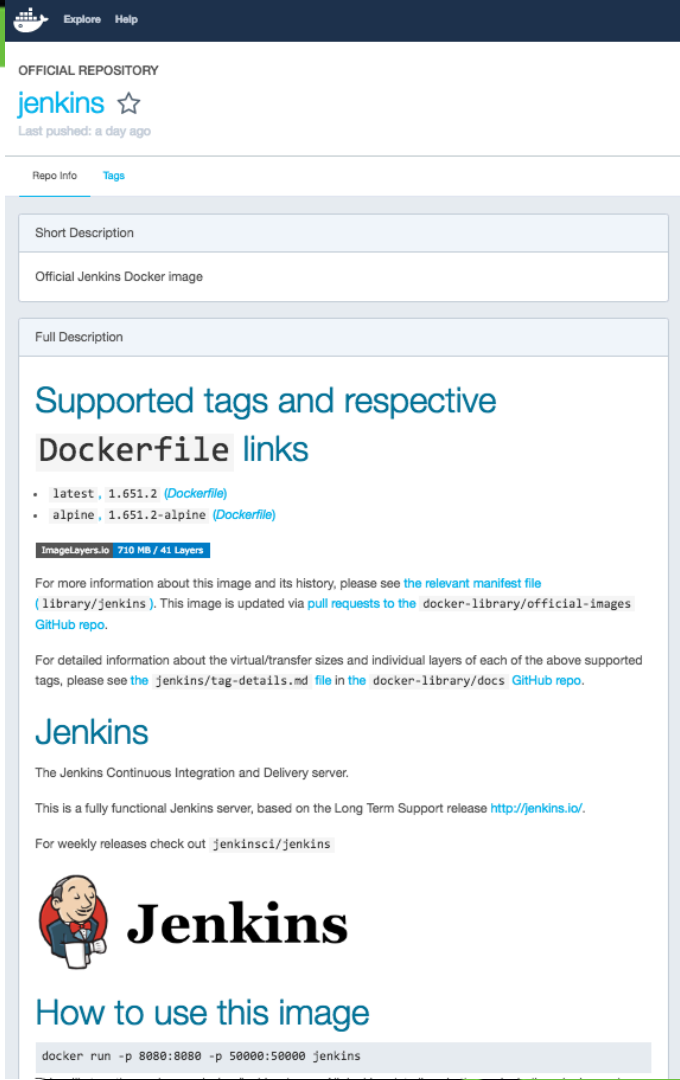
- Jenkins OSS Master

CloudBees Images https://hub.docker.com/u/cloudbees/

- CJP Master
- CJP Operations Center

```
docker run -p 8080:8080 -v /your/home:/var/jenkins_home jenkins
```

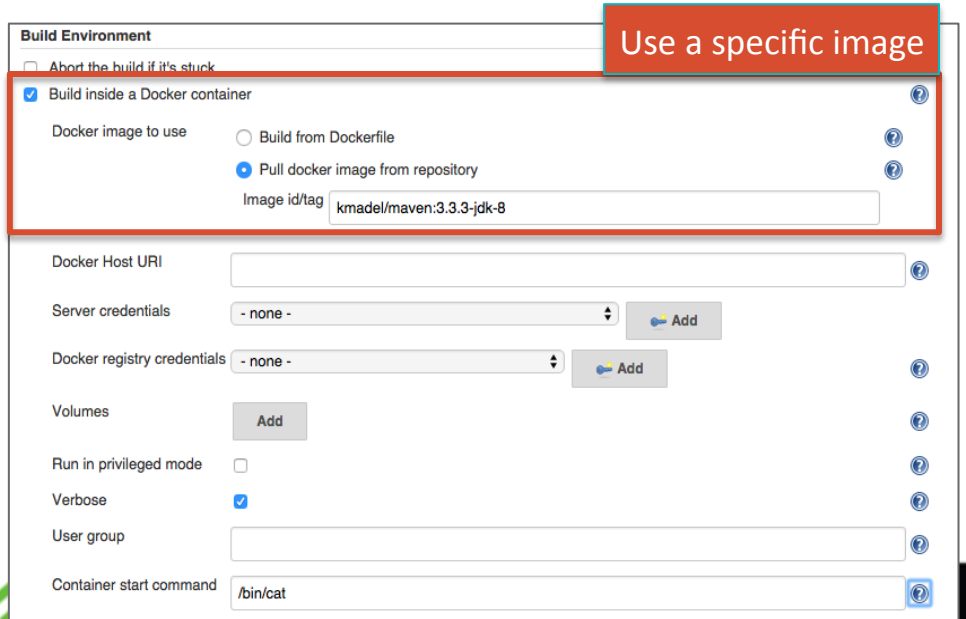
Community Slave Images



Custom Build Environment

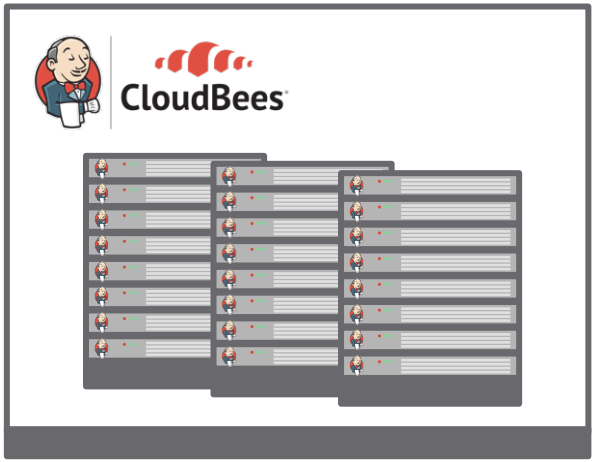
- Ensure reproducible environment
- Clean Room
- Isolated
- Faster than a VM to launch

- Mounts the workspace to the container – easy to share artifacts
- Secure access to Docker Host and Private Registry



CloudBees Docker Shared Config

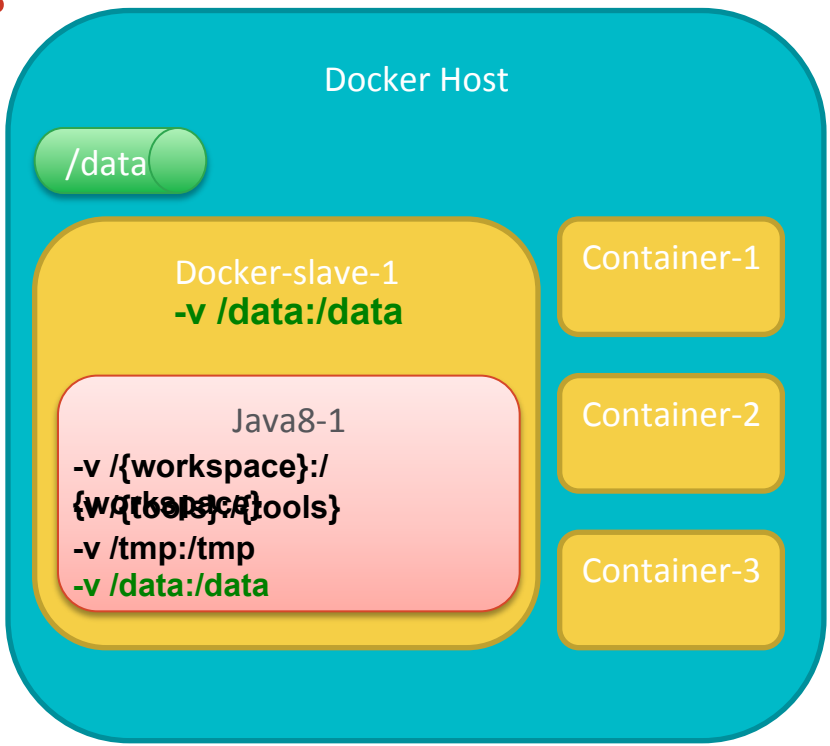
- Push docker host configuration to all masters in cluster
- Define images and slave labels centrally
- Supports Docker Swarm



A word on Volumes

Custom Build Env mounts:
- workspaceDir
- Tool
- Tmp

Consider mounting a /data volume for the **mvn repo,** ie /data/.m2repo
from the slave, and also, from the Docker Host.

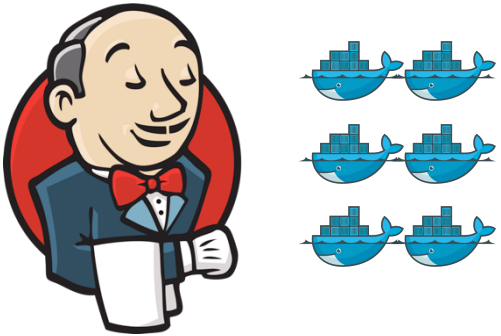


2. Build, Test, & Deploy Docker Images from Jenkins

Build and Publish – build projects that have a Dockerfile and push the resultant tagged image to Docker Hub

Docker Traceability – identify which build pushed a particular container that and displays the build / image details in Jenkins

Docker Hub Notification – trigger downstream jobs when a tagged container is pushed to Docker Hub

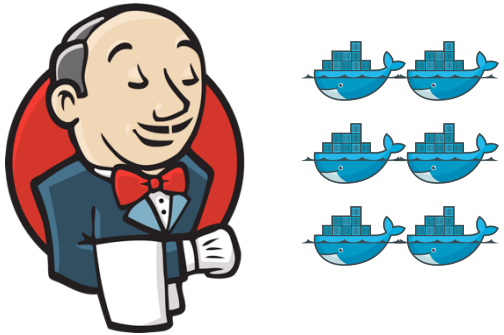


2. Build, Test, & Deploy Docker Images from Jenkins

Build and Publish – build projects that have a Dockerfile and push the resultant tagged image to Docker Hub

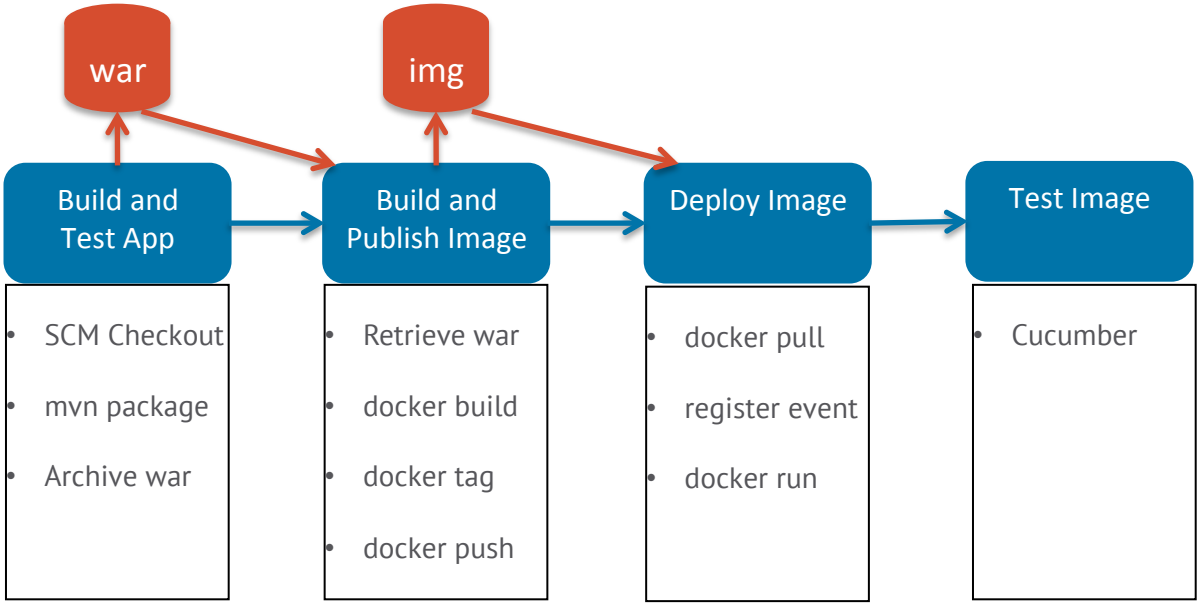
Docker Traceability – identify which build pushed a particular container that and displays the build / image details in Jenkins

Docker Hub Notification – trigger downstream jobs when a tagged container is pushed to Docker Hub



Build and Publish

Build projects that have a Dockerfile and push the resultant tagged image to Docker Hub



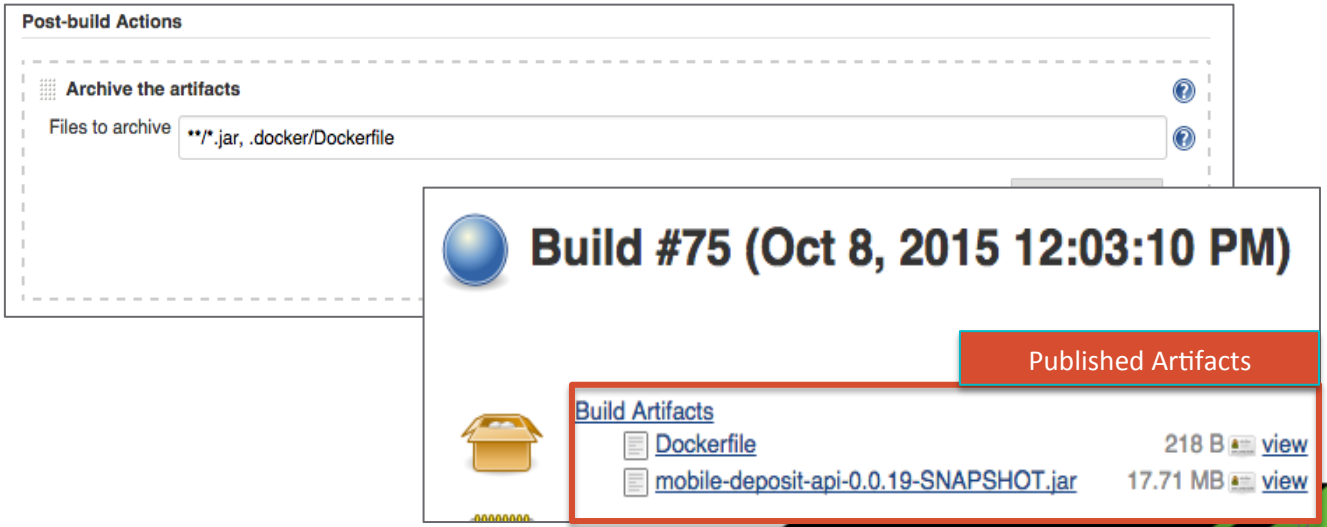
A typical job set up

All | Docker Images | +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		1. build and test	1 day 14 hr - #98	N/A	6 min 30 sec	
		2. create and publish docker image	1 day 14 hr - #38 harniman/mobile-deposit-api:38 harniman/mobile-deposit-api:latest	1 mo 28 days - #28 harniman/mobile-deposit-api:28 harniman/mobile-deposit-api:latest	1 min 30 sec	
		3. Deploy-container-and-run-tests	1 day 14 hr - #27	2 mo 0 days - #12	1 min 46 sec	

Build and Publish – Step 1: Build the app artifact

- Regular job configuration (can use **Docker Custom Build Env)**
- Eg: **`maven clean package`**
- Publish the resultant artifacts with a **Post-build Action**:

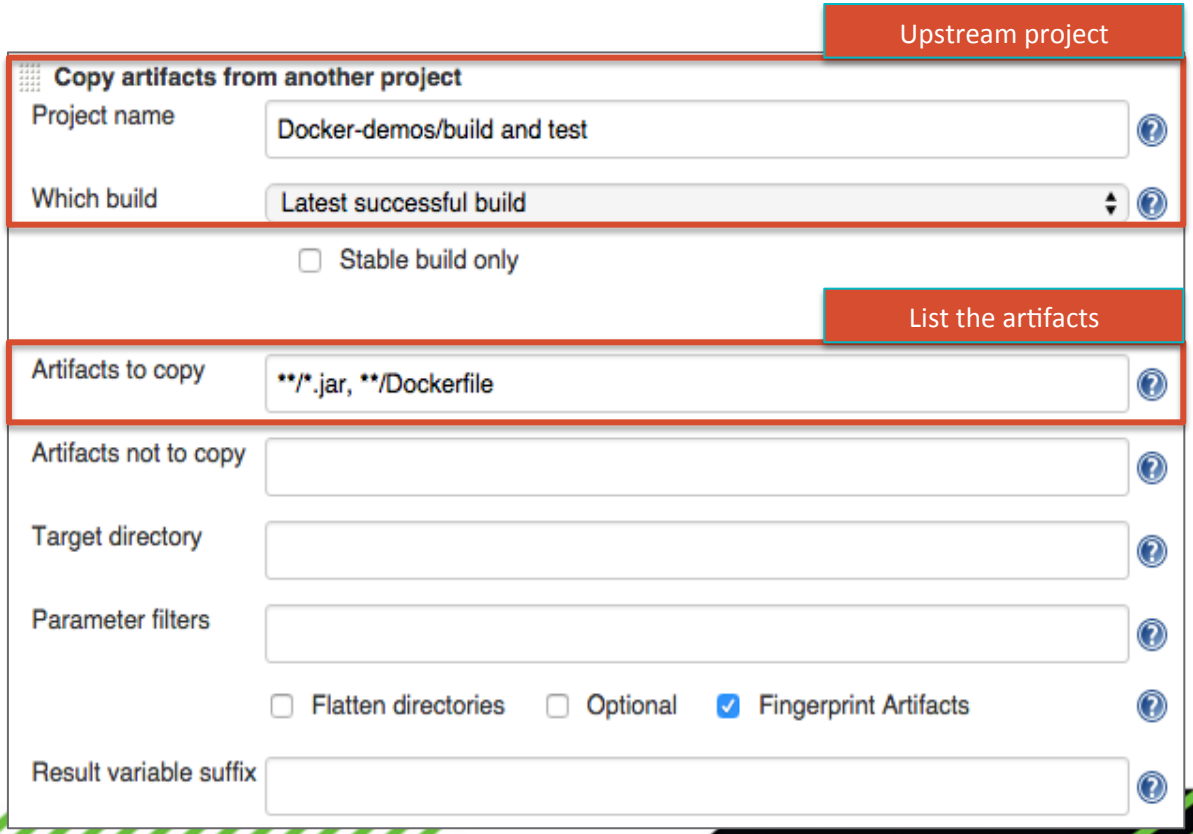


Build and Publish – Step 2: Build the Docker Image

This job will do two things:

- Retrieve the artifacts we need as input
  - Jar
  - Dockerfile
- Invoke the Docker utilities to
  - Build the image
  - Tag
  - Publish

Obtain the artifacts

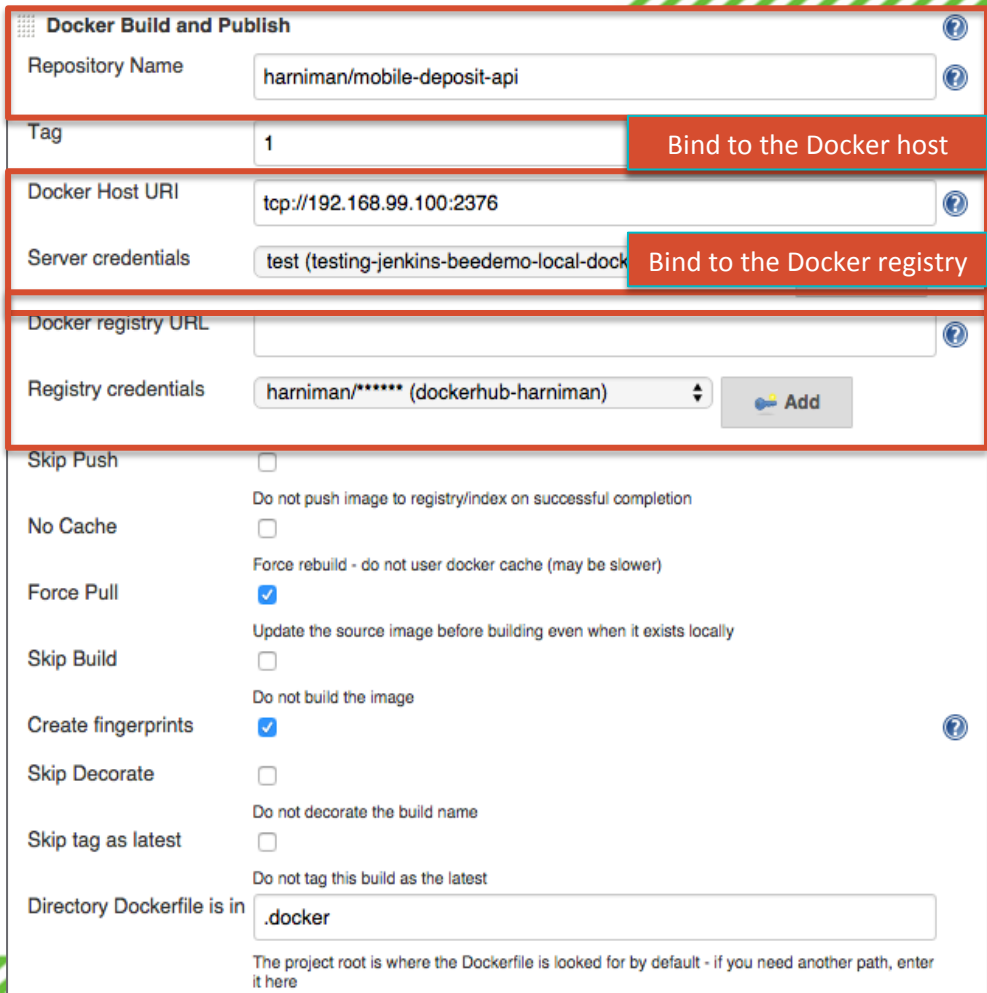


The Dockerfile

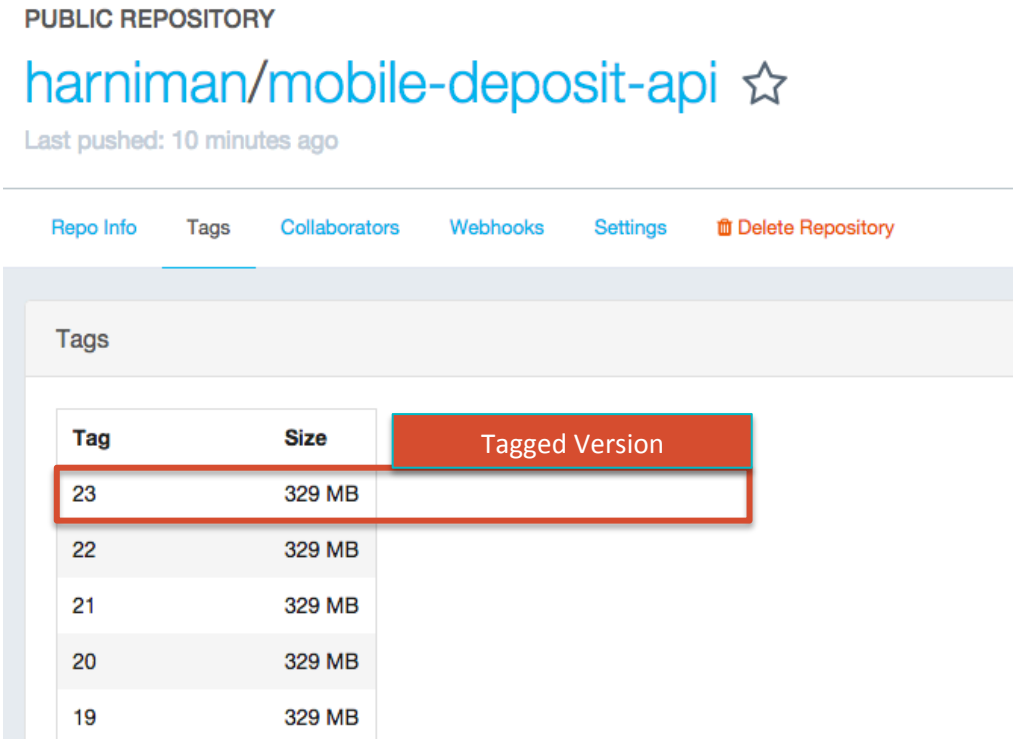
6 lines (6 sloc) | 218 Bytes

```
1  FROM kmadel/java:8
2  VOLUME /tmp
3  #ADD ${project.build.finalName}.jar app.jar
4  ADD mobile-deposit-api.jar app.jar
5  RUN bash -c 'touch /app.jar'
6  ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

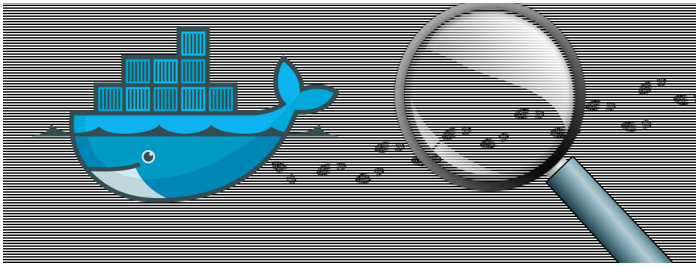
Invoke Docker



Tagged Image in Docker Hub



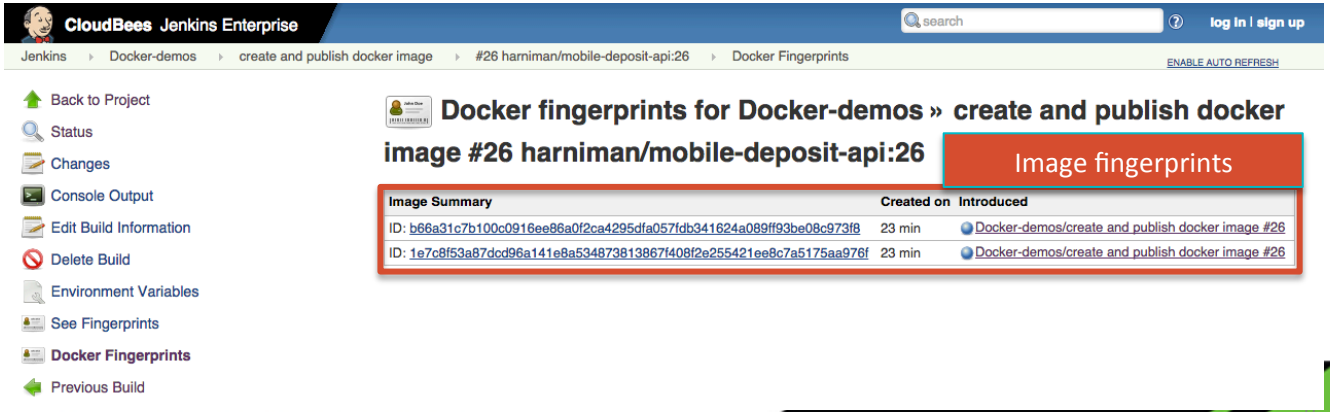
Traceability



Identify which build pushed a particular container and display the build / image details in Jenkins

Traceability

- Builds on existing Jenkins artifact traceability
- Allows the tracking of the creation and use of Docker containers in Jenkins and their future use.
- Combine with artifact fingerprinting for a comprehensive solution
- Each Build shows the image fingerprints created



Traceability – registering events

- Jenkins can track actions against this image such as:
  - Creating a container
  - Container events such as start/stop
- To achieve this, it is necessary to call the Traceability API – see **`$(JENKINS_URL)/docker-traceability/api/`**
- There are two endpoints to submit events to:

`/docker-traceability/submitContainerStatus`	Allows to submit the current container status snapshot with a minimal set of parameters. Outputs of docker inspect $(containerId) can be directly submitted to Jenkins server using this command.
`/docker-traceability/submitReport`	Submits a report using the extended JSON API. This endpoint can be used by scripts to submit the full available info about the container and its environment in a single command.

Traceability – registering events - example

This a Shell build step:

Spin up container

```
docker run -it --cidfile="$$" -d harniman/mobile-deposit-api:$tag
output=`cat $$` // Captures the Container ID
```

Notify Jenkins

```
curl http://{user}:{token}@{jenkins_url}/docker-traceability/
submitContainerStatus --data-urlencode status=deployed --data-
urlencode inspectData="$(docker inspect $output)" --data-urlencode
environment=$Environment --data-urlencode hostName={docker host} --
data-urlencode imageName=harniman/mobile-deposit-a
```

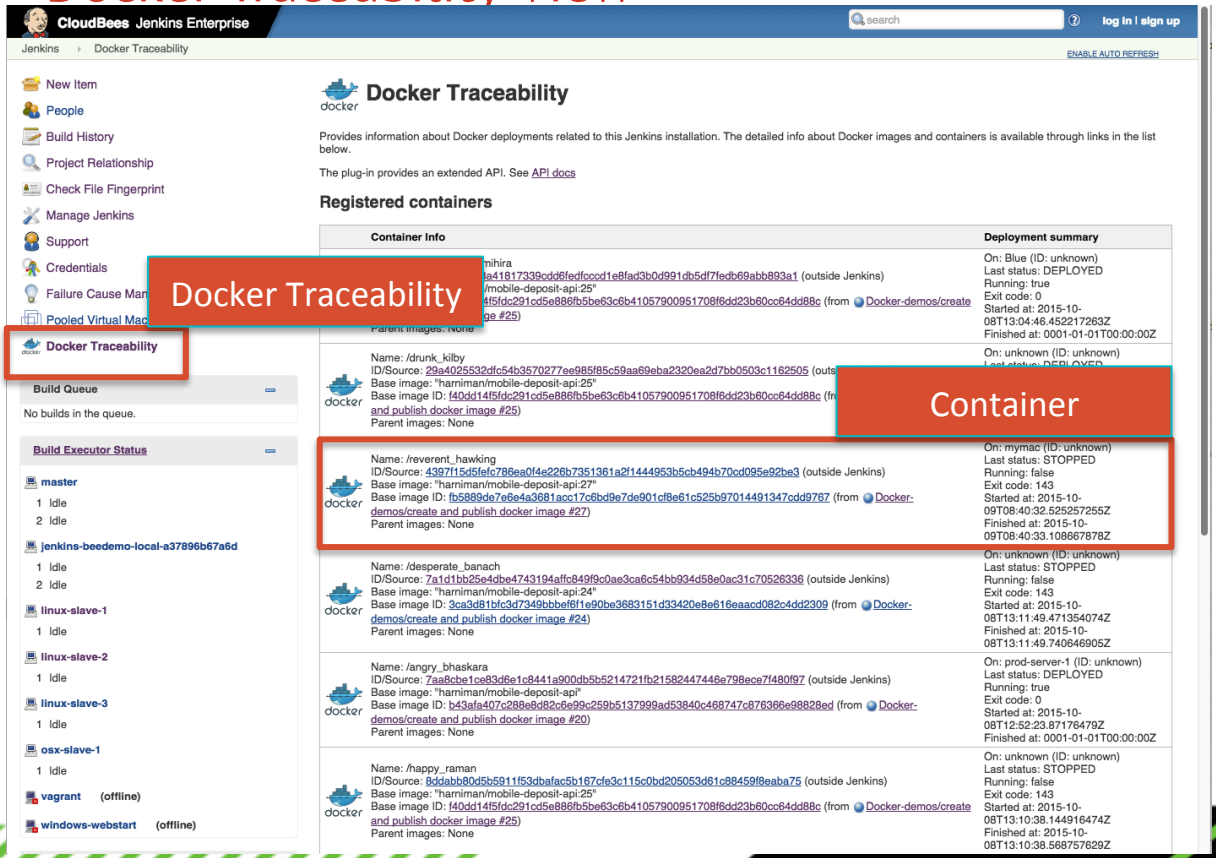
Stop Container

```
docker stop $output
```

Notify Jenkins

```
curl http://{user}:{token}@{jenkins_url}/docker-traceability/
submitContainerStatus --data-urlencode status=stopped --data-
urlencode inspectData="$(docker inspect $output)" --data-urlencode
environment=$Environment --data-urlencode hostName={docker host} --
data-urlencode imageName=harniman/mobile-deposit-api:$tag
```

Docker Traceability View



Container Use View

Container /reverent_hawking

MD5: 4397f15d5fefc786ea0f4e226b735136

Introduced 1 hr 28 min ago outside Jenkins

Usage

This file has not been used anywhere else.

Container Info

The information below has been retrieved from the latest report (2015-10-09T09:40:33Z).

- Name: /reverent_hawking
- Container ID: 4397f15d5fefc786ea0f4e226b7351361a2f1444953b5cb494b70cd095e92be3
- Origin: (outside Jenkins)
- Created on: 2015-10-09T08:40:32.065977462Z

Link to Build

Sources

- Image ID: fb5889de7e6e4a3681acc17c6bd9e7de901cf8e61c525b97014491347cdd9767 (from Docker-demos/create and publish docker image #27)
- Image name: "harniman/mobile-deposit-api:27"
- Parent images: None

Status

- Running on: mymac (ID: unknown)
- Last status: STOPPED
- Pid: 0
- Exit code: 143
- Started at: 2015-10-09T08:40:32.525257255Z
- Finished at: 2015-10-09T08:40:33.108667878Z

Raw data

- Last record in JSON

Deployment Events

Deployment Events

Time	Event	State
2015-10-09T09:40:32Z	DEPLOYED	running: true, paused: false, exit code:0
2015-10-09T09:40:33Z	STOPPED	running: false, paused: false, exit code:143

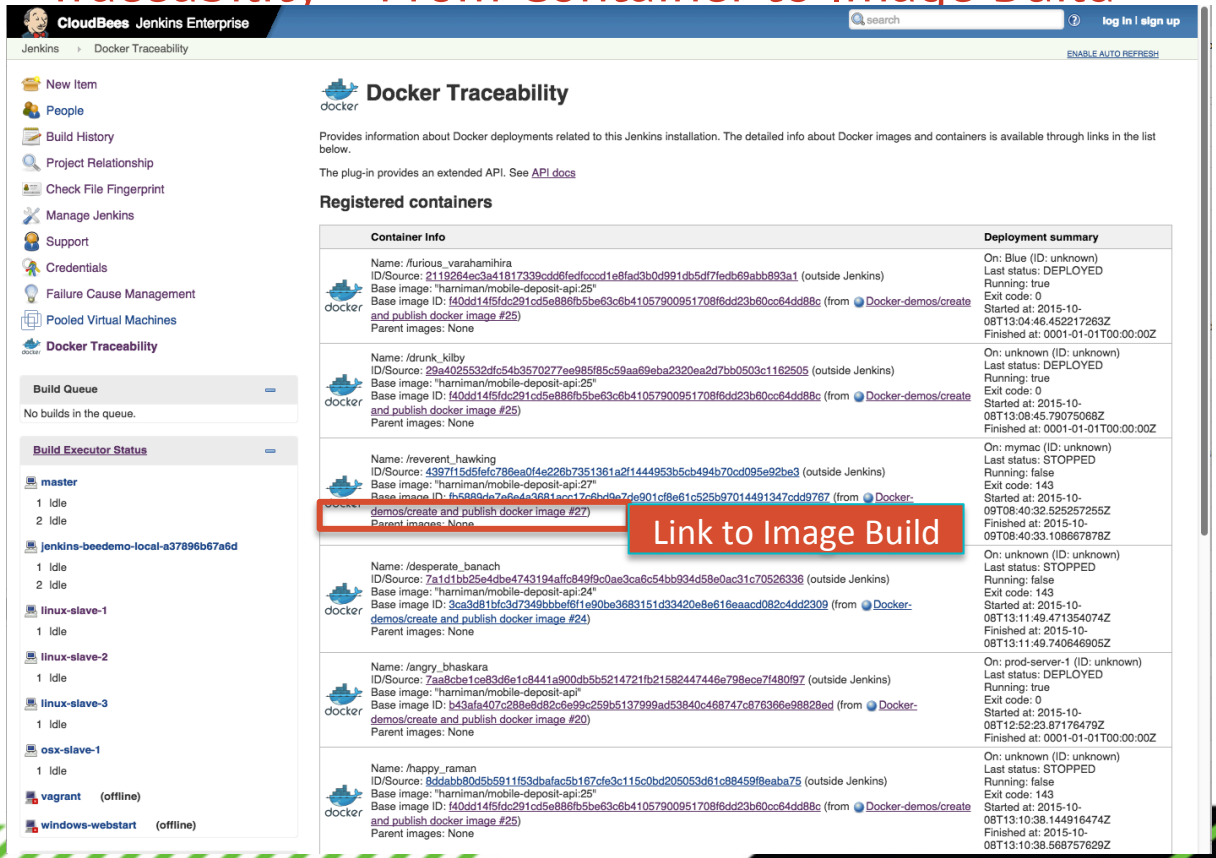
Traceability in Action

Problem: We are running tests and there is a problem I need to fix – how do I find the related source?

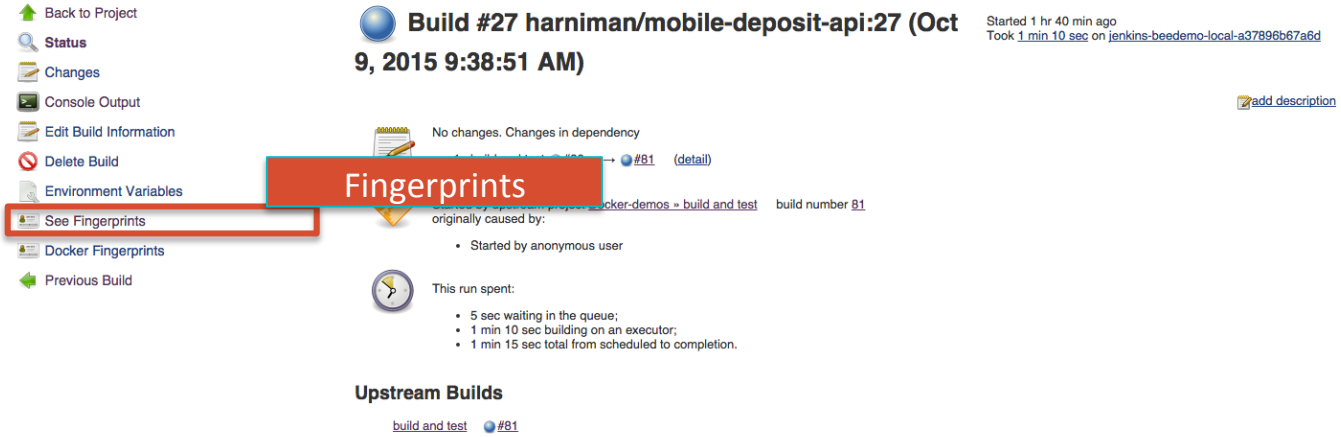
Follow this logical sequence:

1. Running Container -> Image
2. Image -> Image Build Job
3. Image Build Job -> Application Version
4. Application Version -> Application Build
5. Application Build -> Application Source

Traceability – From Container to Image Build



Traceability – Image Build Job



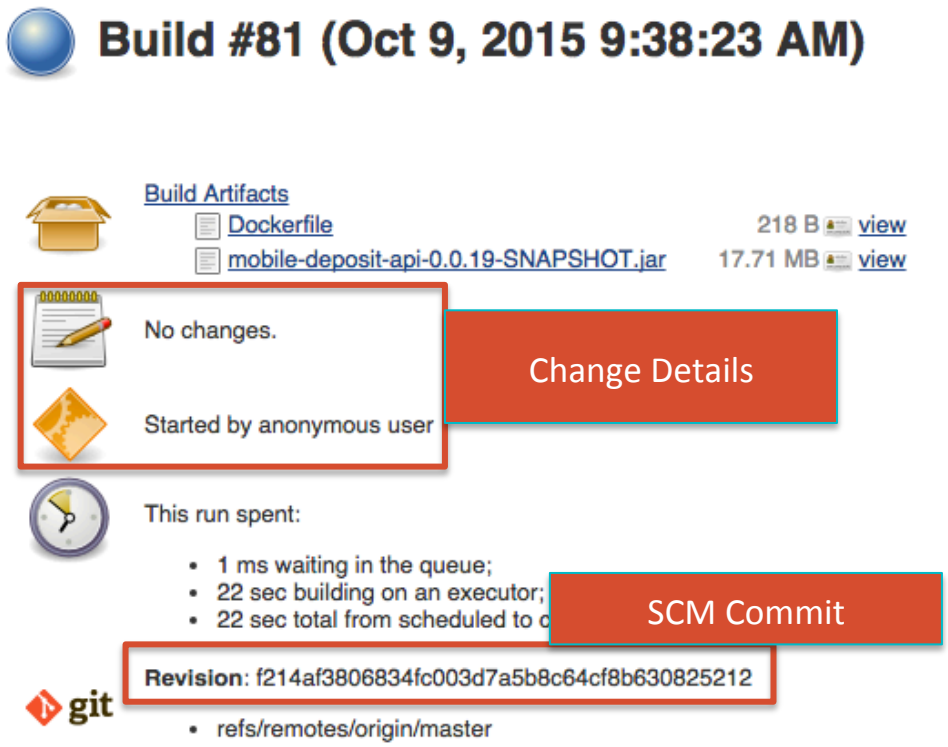
Traceability – Image Build Job - Fingerprints

Recorded Fingerprints

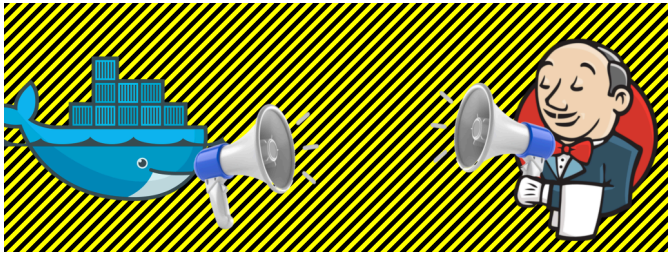
File ↓	Original owner	Age	
Dockerfile	Docker-demos/build and test #72	23 hr old	more details
mobile-deposit-api-0.0.19-SNAPSHOT.jar	Docker-demos/build and test #81	1 hr 42 min old	more details

The builds that provided the artifacts

Traceability – Application Build



Notification



Trigger downstream jobs when a tagged container is pushed to Docker Hub

Docker Hub Notification

Trigger downstream jobs when a tagged container is pushed to Docker Hub

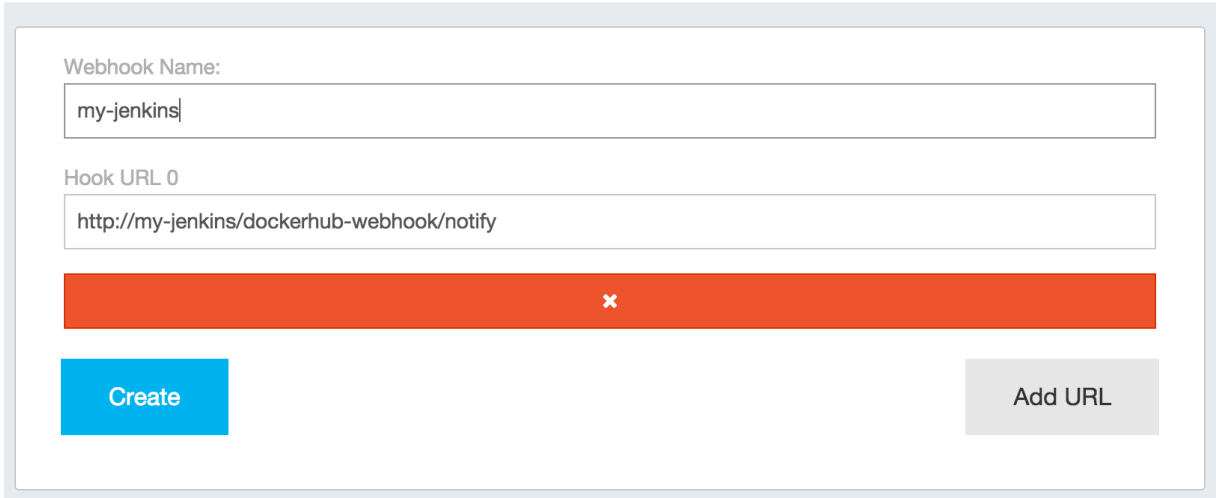
The Docker Hub Notification Trigger plugin lets you configure Jenkins to trigger builds when an image is pushed to Docker Hub. E.g. to run verification for the container.

What are the steps

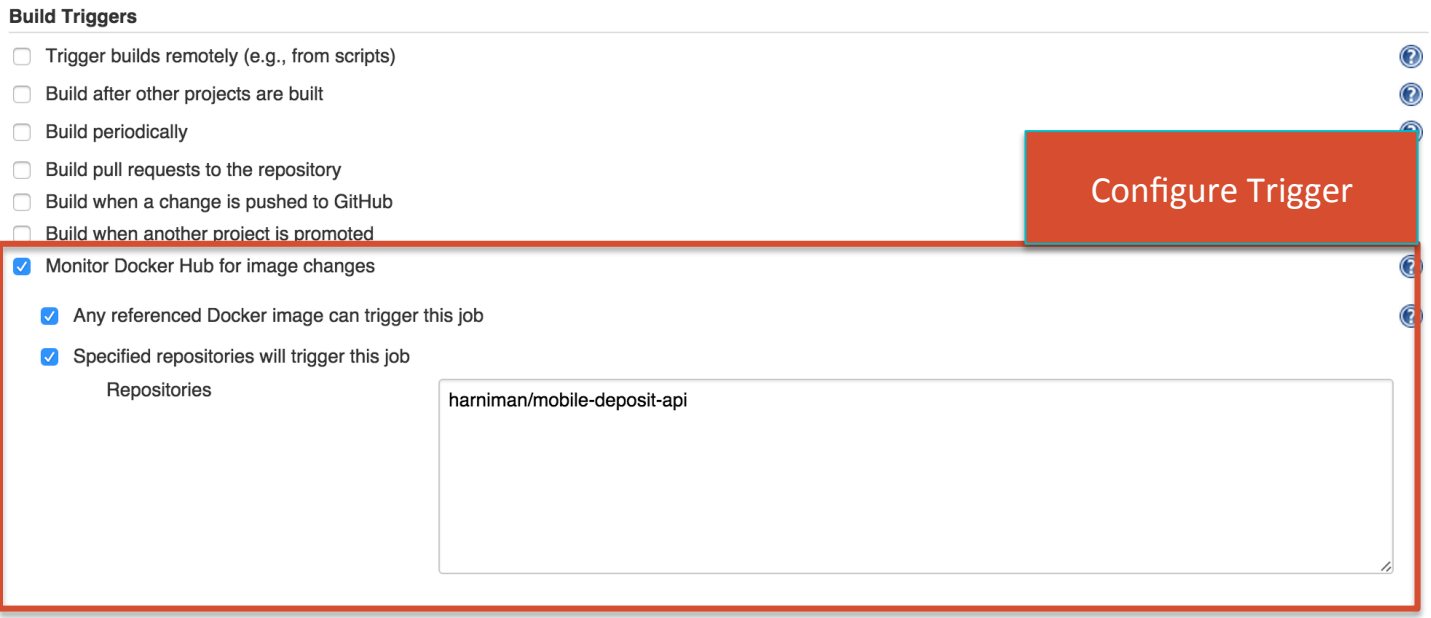
- Set up a WebHook Account for Notification
- Set up your Docker Registry to make callbacks on Image events
- Set up your builds

Docker Hub Notification – Docker Registry Webhook

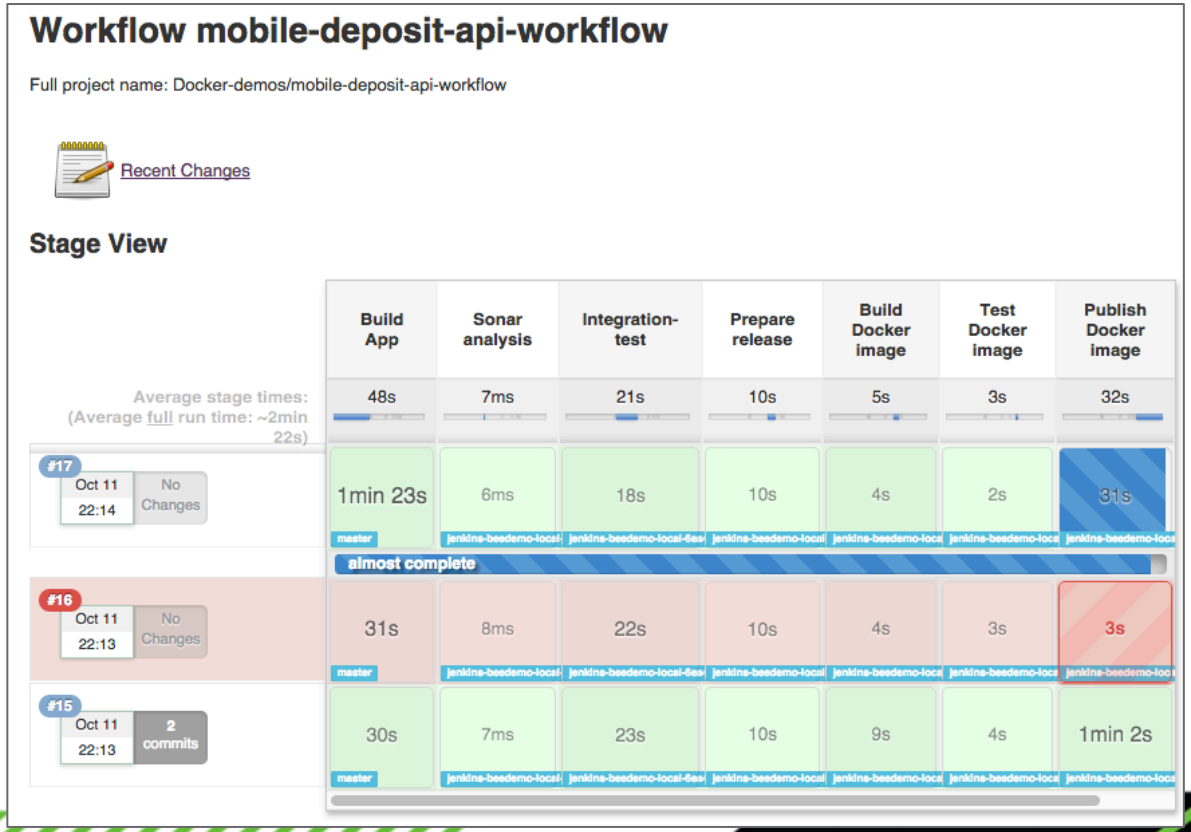
In the format:
http://<user>:<token>@<jenkins_url>/dockerhub-webhook/notify



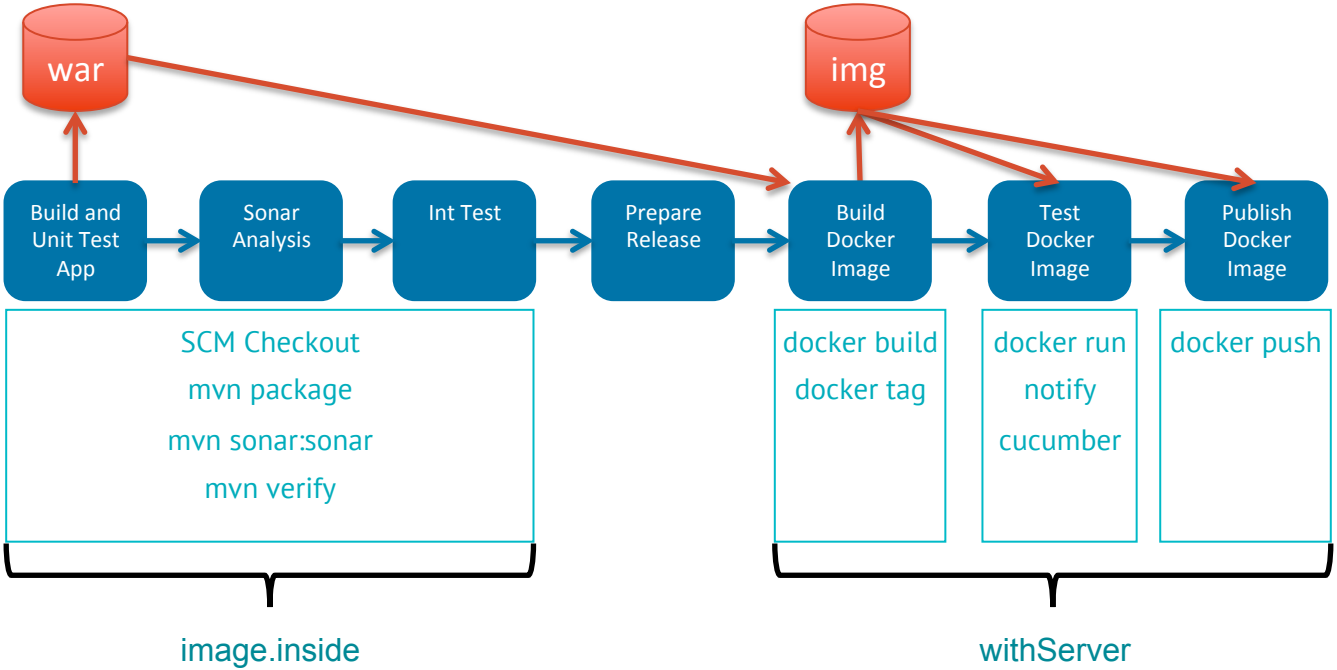
Docker Hub Notification – Job Set up



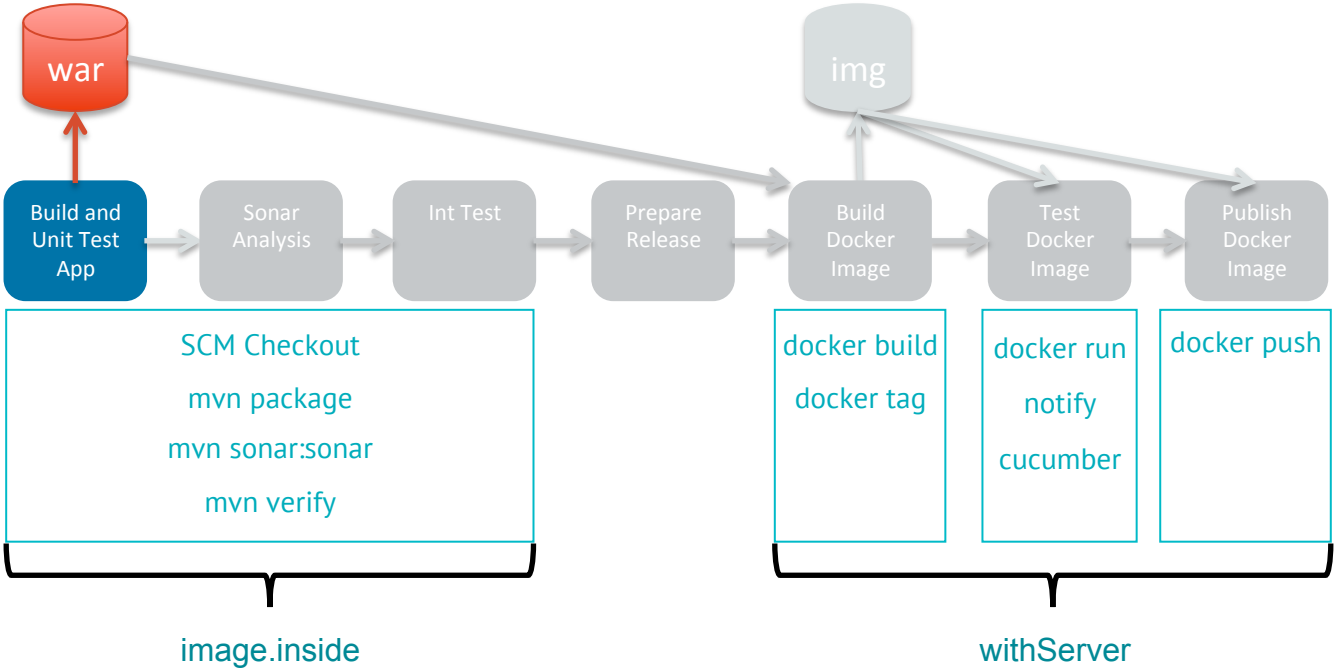
Best of All: Jenkins Pipeline + Docker



Pipeline Stages



Build, unit test and package



Build, unit test and package

Specify the Stage Name

Specify the slave label

Custom Build Env

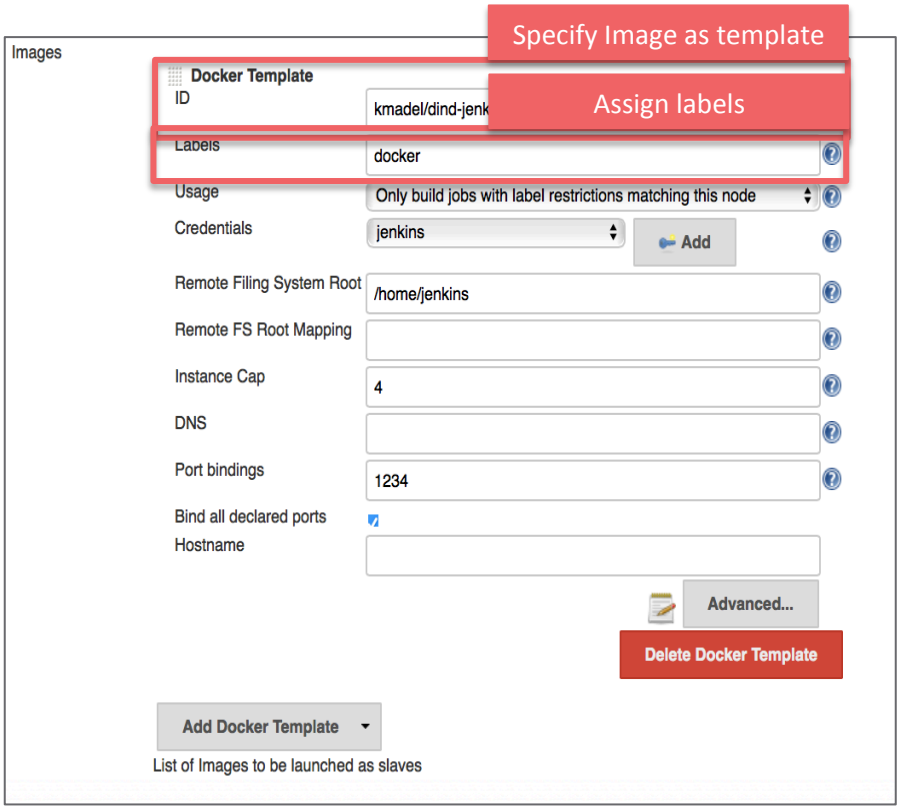
Mount volume from slave

.m2 repo location

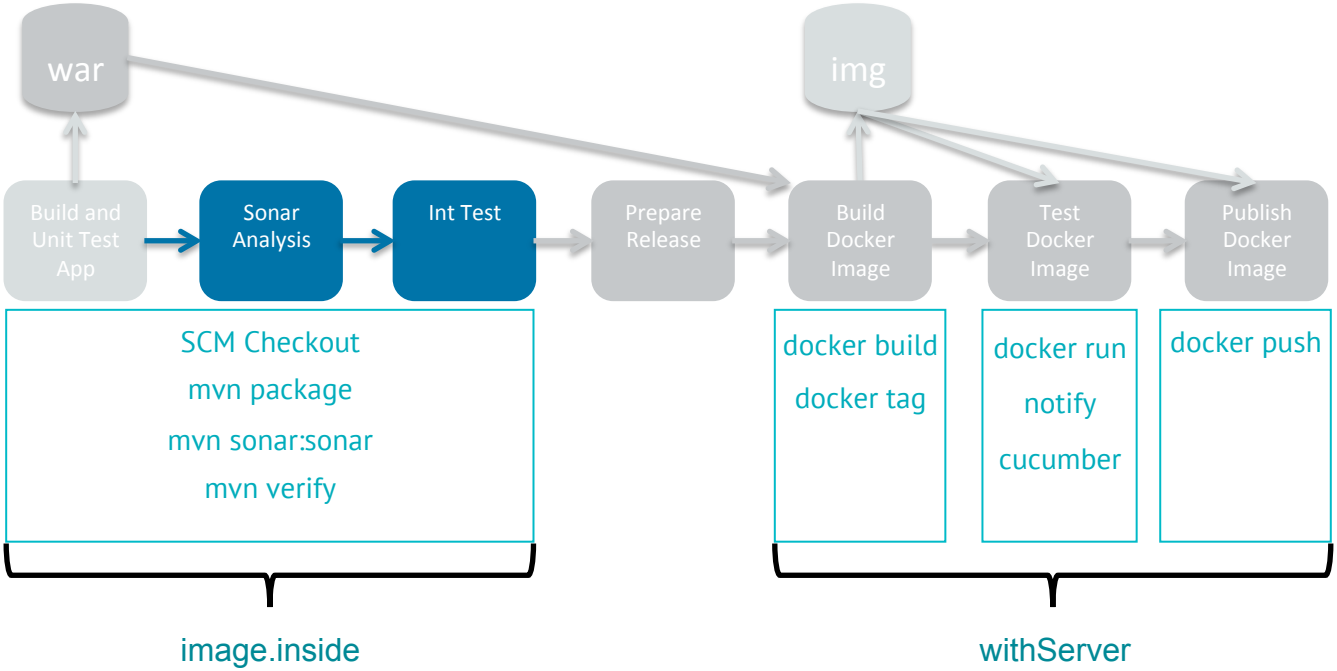
co and build

```
stage 'Build Ap
node('docker') {
   docker.image('maven:3.3.3-jdk-8').inside('-v /data:/data' {
      mkdir -p /data/mvn
      writeFile file: 'settings.xml', text: "(.....)
      git 'https://github.com/cloudbees/mobile-deposit-api.git'
      sh 'mvn -s settings.xml clean package'
      …
   }
}
```

Defining a Docker Slave



Test the app


war
img
Build and Unit Test App
Sonar Analysis
Int Test
Prepare Release
Build Docker Image
Test Docker Image
Publish Docker Image
SCM Checkout
mvn package
mvn sonar:sonar
mvn verify
docker build
docker tag
docker run
notify
cucumber
docker push
image.inside
withServer

Test the app

In same env as build

```
node('docker') {
          docker.image('maven:3.3.3-jdk-8').inside('-v /data:/data' {
```

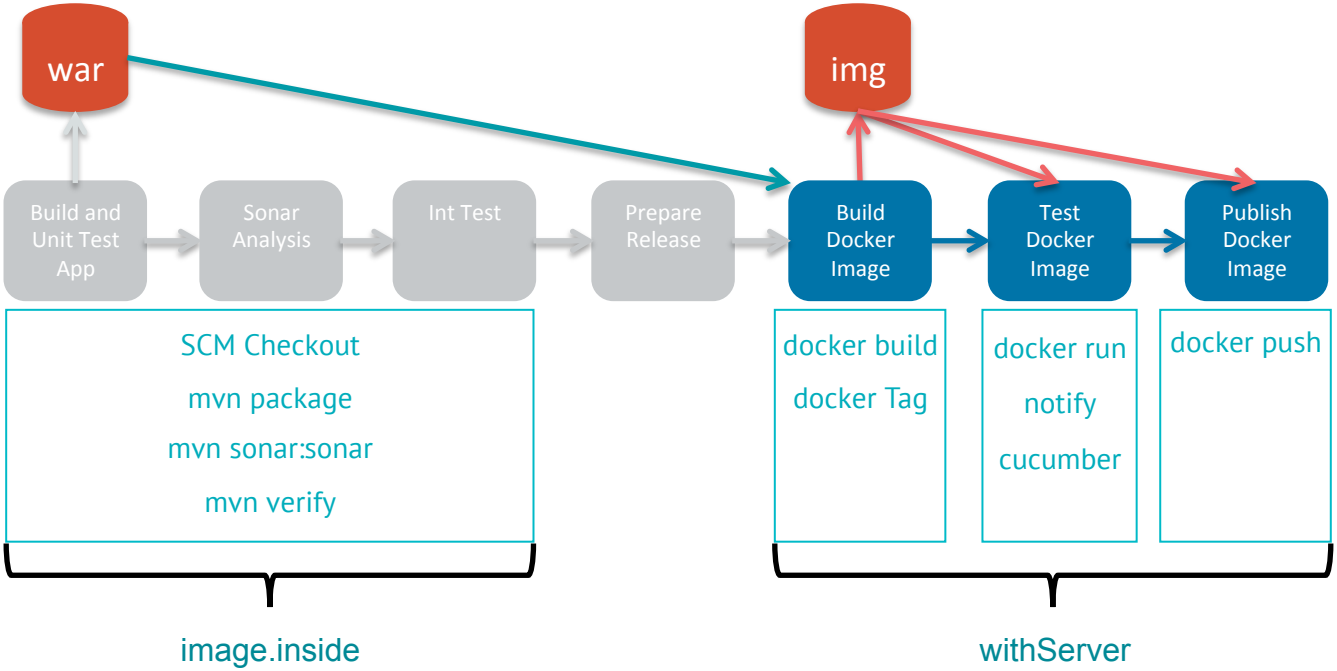
Sonar tests

```
          stage 'Sonar analysis'
          sh 'mvn -s settings.xml sonar:sonar'
```

Run API Tests

```
          stage 'Integration-test'
          sh 'mvn -s settings.xml  verify'
          step([$class: 'JUnitResultArchiver', testResults: '**/
target/surefire-reports/TEST-*.xml'])
          }
}
```

Build, test and publish Docker image



Build, test and publish Docker image

Bind to docker host

```
docker.withServer('tcp://192.168.99.100:2376', 'slave-docker-us-
east-1-tls'){
        stage 'Build Docker image'
        def mobileDepositApiImage
        dir('.docker') {
                sh "mv ../target/*-SNAPSHOT.jar  mo
                mobileDepositApiImage = docker.build "harniman/
mobile-deposit-api:${buildVersion}"
        }
```

Change directory

Build docker image

…

Build, test and publish Docker image

```
...
        stage 'Test Docker image'
        container=mobileDepositApiImage.run("--name mobile-deposit-
api -p 8080:8080")
         sh "curl http://<user>:<token>@<host>:8080/docker-
traceability/submitContainerStatus \......
// insert cucumber tests here
        stage 'Publish Docker image'
        withDockerRegistry(registry: [credentialsId: 'dockerhub-
harniman']) {
                mobileDepositApiImage.push()
        }
}
```

Launch container

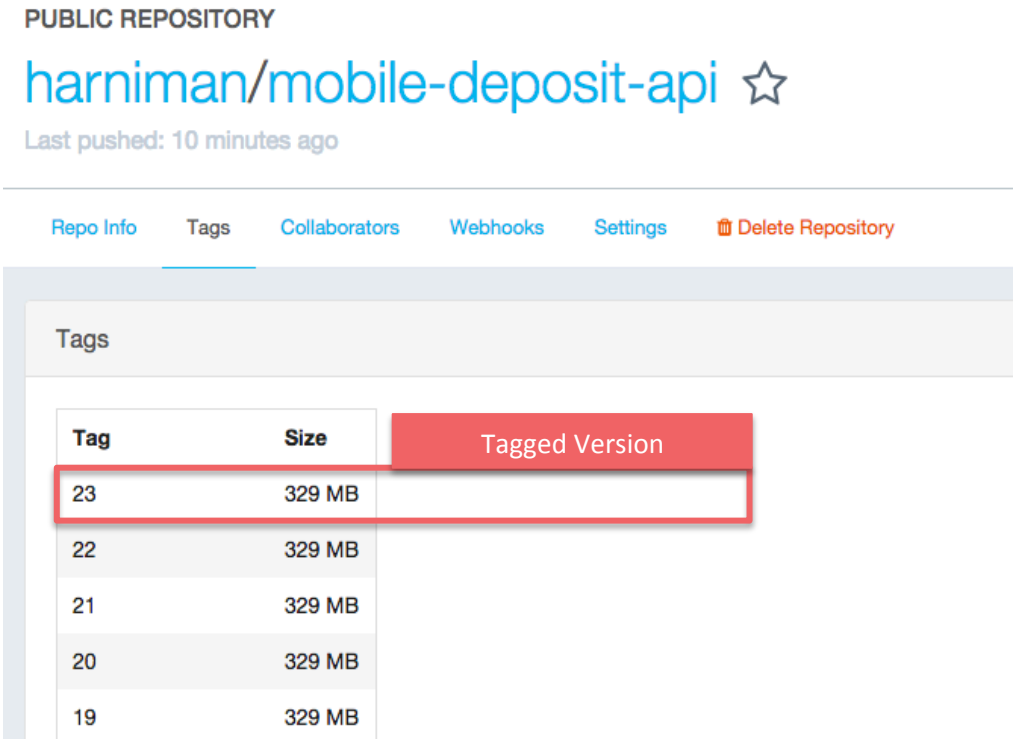
Submit traceability report

Run some Tests

Bind to registry

Push image

Tagged Image in Docker Hub



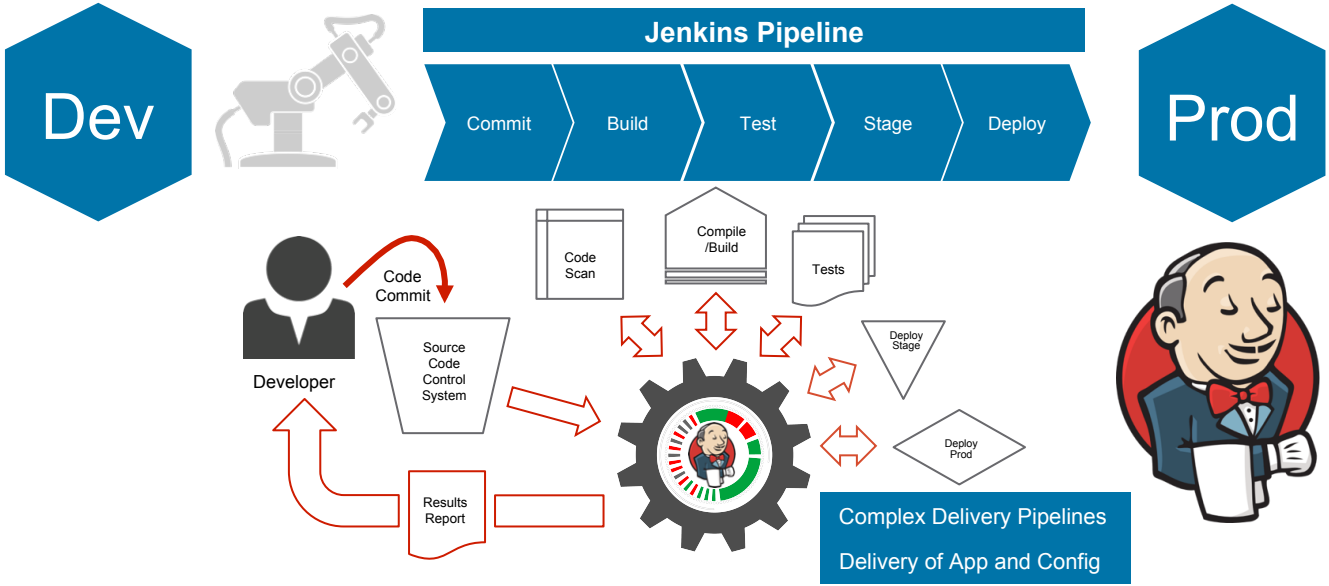
Best of All: Jenkins Pipeline + Docker

```
stage 'Build Source'
node('docker') {
    docker.image('maven:3.3.3-jdk-8') {
        git 'https://github.com/cloudbees/mobile-deposit-api.git'
        sh 'mvn clean package'
    }
}

node('docker') {
    docker.withServer('tcp://docker.beedemo.net:2376', 'docker-beedemo-creds'){
        stage 'Build Docker Image'
        def image = docker.build "cloudbees/mobile-deposit-api:${buildVersion}"

        stage 'Publish Docker Image'
        docker.withRegistry('https://registry.beedemo.net/', 'docker-registry-login') {
            image.push()
        }

        stage 'Deploy Docker Image'
        def container = image.run('--name mobile-deposit-api -p 8080:8080')
   }
}
```

OSS - Pipeline-as-Code Jenkinsfile

- Define complete Workflow within Code
- Edit Workflow Job in IDE
- Workflow Job automatically incorporates changes
- Version Workflow Jobs
- Supports: Git, Mercurial, SVN

Jenkins, with Pipeline, is the Proven CD Platform



Jenkins World 2016

The event for everything Jenkins: community, CloudBees and ecosystem.

- Santa Clara Convention Center
- September 13-15, 2016

Learn more and register now!

www.jenkinsworld.com

Resources

- @BhatiaKishore
- Github: https://github.com/kishorebhatia
- Mobile-deposit-api example: https://github.com/NewGithubOrg/mobile-deposit-api
- Jenkins BlueOcean: https://jenkins.io/projects/blueocean/
- Jenkins Pipeline-as-code: https://jenkins.io/doc/pipeline/

Thank You!