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

June 15, 2016
Kishore Bhatia
@bhatiakishore
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

Supported tags and respective Dockerfile links

- latest, 1.651.2 (Dockerfile)
- alpine, 1.651.2-alpine (Dockerfile)

For more information about this image and its history, please see the relevant manifest file (library/jenkins). This image is updated via pull requests to the docker-library/official-images GitHub repo.

For detailed information about the virtual/transfer sizes and individual layers of each of the above supported tags, please see the jenkins/tag-details.md file in the docker-library/docs GitHub repo.

Jenkins

The Jenkins Continuous Integration and Delivery server.

This is a fully functional Jenkins server, based on the Long Term Support release http://jenkins.io/.

For weekly releases check out jenkinscl/jenkins

How to use this image

docker run -p 8080:8080 -p 50000:50000 jenkins
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

Use a specific image
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

Build and Test App
- SCM Checkout
- mvn package
- Archive war

Build and Publish Image
- Retrieve war
- docker build
- docker tag
- docker push

Deploy Image
- docker pull
- register event
- docker run

Test Image
- Cucumber
A typical job set up

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

<table>
<thead>
<tr>
<th>Copy artifacts from another project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project name</td>
</tr>
<tr>
<td>Which build</td>
</tr>
<tr>
<td>Stable build only</td>
</tr>
</tbody>
</table>

| Artifacts to copy                   | **/*.jar, **/Dockerfile |
| Artifacts not to copy               | [ ] checkbox                  |
| Target directory                    | [ ] checkbox                  |
| Parameter filters                   | [ ] checkbox                  |
| Flatten directories                 | [ ] checkbox                  |
| Optional                            | [ ] checkbox                  |
| Fingerprint Artifacts               | [ ] checkbox                  |
| Result variable suffix              | [ ] checkbox                  |

Upstream project list

List the artifacts
The Dockerfile

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>FROM kmadea/java:8</code></td>
</tr>
<tr>
<td>2</td>
<td><code>VOLUME /tmp</code></td>
</tr>
<tr>
<td>3</td>
<td><code>#ADD ${project.build.finalName}.jar app.jar</code></td>
</tr>
<tr>
<td>4</td>
<td><code>ADD mobile-deposit-api.jar app.jar</code></td>
</tr>
<tr>
<td>5</td>
<td><code>RUN bash -c 'touch /app.jar'</code></td>
</tr>
<tr>
<td>6</td>
<td><code>ENTRYPOINT [&quot;java&quot;,&quot;-Djava.security.egd=file:/dev/./urandom&quot;,&quot;-jar&quot;,&quot;/app.jar&quot;]</code></td>
</tr>
</tbody>
</table>
Invoke Docker

Bind to the Docker host

Bind to the Docker registry

Repository Name: harniman/mobile-deposit-api
Tag: 1
Docker Host URI: tcp://192.168.99.100:2376
Server credentials: test (testing-jenkins-beedemo-local-dock)
Docker registry URL:
Registry credentials: harniman/***** (dockerhub-harniman)

Skip Push
No Cache
Force Pull
Skip Build
Create fingerprints
Skip Decorate
Skip tag as latest
Directory Dockerfile is in .docker
Tagged Image in Docker Hub

PUBLIC REPOSITORY

harniman/mobile-deposit-api ✺

Last pushed: 10 minutes ago

Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>329 MB</td>
</tr>
<tr>
<td>22</td>
<td>329 MB</td>
</tr>
<tr>
<td>21</td>
<td>329 MB</td>
</tr>
<tr>
<td>20</td>
<td>329 MB</td>
</tr>
<tr>
<td>19</td>
<td>329 MB</td>
</tr>
</tbody>
</table>

Tagged Version
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:

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/docker-traceability/submitContainerStatus</td>
<td>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.</td>
</tr>
<tr>
<td>/docker-traceability/submitReport</td>
<td>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.</td>
</tr>
</tbody>
</table>
Traceability – registering events - example

This a Shell build step:

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

```
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-api:$tag

docker stop $output
```

```bash
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

- **Build Queue:**
  - No builds in the queue.

- **Build Executor Status:**
  - master
    - 2 Id0
    - 1 1de
  - jenkins-beedemo-local-a37896d7a8d
    - 1 1de
  - jenkins-beedemo-local-a37896d7a8d
    - 1 1de
  - linux-slave-1
    - 1 1de
  - linux-slave-2
    - 1 1de
  - linux-slave-3
    - 1 1de
  - oxz-slave-1
    - 1 1de
  - vagrant (offline)
  - windows-webstart (offline)

- **Docker Traceability:**
  - Provides information about Docker deployments related to this Jenkins installation.
  - The detailed info about Docker images and containers is available through links in the list below.
  - The plugin provides an extended API. See API docs.

- **Registered containers:**

  - **Name:** drunk, jilky
    - **ID/Source:** 3a4t063d6bf3a636777a98895d59a5f3b200a32f7b0503121263
    - **Base image:** hamiran/mobile-deposit-api:25
    - **Parent images:** None
    - **Deployment summary:**
      - On: Blue (ID: unknown)
      - Last status: DEPLOYED
      - Running: true
      - Exit code: 0
      - Started at: 2015-10-08T13:04:46.523-17362Z
      - Finished at: 00:01-01T00:00:00Z
  
  - **Name:** revent_bawling
    - **ID/Source:** 3a4t063d6bf3a636777a98895d59a5f3b200a32f7b0503121263
    - **Base image:** hamiran/mobile-deposit-api:25
    - **Parent images:** None
    - **Deployment summary:**
      - On: myrmic (ID: unknown)
      - Last status: STOPPED
      - Running: false
      - Exit code: 143
      - Started at: 2015-10-09T08:40.32.5525525Z
      - Finished at: 2015-10-09T08:40.32.5525525Z

- **Container**

  - **Name:** desperate_banach
    - **ID/Source:** 3a4t063d6bf3a636777a98895d59a5f3b200a32f7b0503121263
    - **Base image:** hamiran/mobile-deposit-api:25
    - **Parent images:** None
    - **Deployment summary:**
      - On: prod-serv-1 (ID: unknown)
      - Last status: STOPPED
      - Running: false
      - Exit code: 0
      - Started at: 2015-10-08T12:52:23.6711947Z
      - Finished at: 2015-01-01T00:00:00Z

  - **Name:** happy_raman
    - **ID/Source:** 3a4t063d6bf3a636777a98895d59a5f3b200a32f7b0503121263
    - **Base image:** hamiran/mobile-deposit-api:25
    - **Parent images:** None
    - **Deployment summary:**
      - On: unknown (ID: unknown)
      - Last status: STOPPED
      - Running: false
      - Exit code: 143
      - Started at: 2015-10-08T13:03.44416474Z
      - Finished at: 2015-10-08T13:03.44416474Z
Container Use View

Container /reverent_hawking

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: 439715d5efed7856ea0f4c226b7351361a2f1d44953b5cb494b70cd95e92be3
- Origin: outside Jenkins
- Created on: 2015-10-09T08:40:32.065977462Z

MD5: 439715d5efed7856ea0f4c226b735136

Sources
- Image ID: f558967e7e4a3681a0c17c6b997e901c1f1e8c1c525b970144913474d9787 (from Docker-demos/create and publish docker image #27)
- Image name: "hami\man\mobile-deposit-api:27"
- Parent images: None

Status
- Running on: mymac (ID: unknown)
- Last status: STOPPED
- Pid: 0
- Exit code: 143
- Started at: 2015-10-09T08:40:32.525257256Z
- Finished at: 2015-10-09T08:40:33.10867875Z

Raw data
- Last record in JSON

Deployment Events

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-09T08:40:32Z</td>
<td>DEPLOYED</td>
<td>running: true, paused: false, exit code: 0</td>
</tr>
<tr>
<td>2015-10-09T08:40:33Z</td>
<td>STOPPED</td>
<td>running: false, paused: false, exit code: 143</td>
</tr>
</tbody>
</table>
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

The CloudBees Jenkins Enterprise provides information about Docker deployments related to this Jenkins installation. The detailed info about Docker images and containers is available through links in the list below. The plugin provides an extended API. See [API docs](#).

#### Registered containers

<table>
<thead>
<tr>
<th>Container Info</th>
<th>Deployment summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID/Source:</td>
<td>Base image: &quot;hamishm/mobile-deploy-api:24&quot; Base image ID: 2b26c4247e514f252d4f98b0c5a2b27d45e49eb5b9e3760101f6e15235b2701144347f7e76f7e (outside Jenkins) Parent images: None</td>
</tr>
</tbody>
</table>

---

**Link to Image Build**
Traceability – Image Build Job

Build #27 harniman/mobile-deposit-api:27 (Oct 9, 2015 9:38:51 AM)

No changes. Changes in dependency

Started by anonymous user

This run spent:
- 5 sec waiting in the queue;
- 1 min 10 sec building on an executor;
- 1 min 15 sec total from scheduled to completion.

Upstream Builds
- build and test #81

Fingerprints
Traceability – Image Build Job - Fingerprints

<table>
<thead>
<tr>
<th>File</th>
<th>Original owner</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dockerfile</td>
<td>Docker-demos/build and test #72</td>
<td>23 hr old</td>
</tr>
<tr>
<td>mobile-deposit-api-0.0.19-SNAPSHOT.jar</td>
<td>Docker-demos/build and test #81</td>
<td>1 hr 42 min old</td>
</tr>
</tbody>
</table>

The builds that provided the artifacts
Traceability – Application Build

Build #81 (Oct 9, 2015 9:38:23 AM)

Build Artifacts
- Dockerfile [218 B • view]
- mobile-deposit-api-0.0.19-SNAPSHOT.jar [17.71 MB • view]

No changes.

Started by anonymous user

This run spent:
- 1 ms waiting in the queue;
- 22 sec building on an executor;
- 22 sec total from scheduled to done

Revision: f214af3806834fc003d7a5b8c64cf8b630825212
- refs/remotes/origin/master

Change Details

SCM Commit
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

Add Webhook

Webhook Name:
my-jenkins|

Hook URL 0
http://my-jenkins/dockerhub-webhook/notify

Create

Add URL
Docker Hub Notification – Job Set up

Build Triggers
- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Build pull requests to the repository
- Build when a change is pushed to GitHub
- Build when another project is promoted

- Monitor Docker Hub for image changes
  - Any referenced Docker image can trigger this job
  - Specified repositories will trigger this job

Repositories
- harniman/mobile-deposit-api
Best of All: Jenkins Pipeline + Docker

**Workflow mobile-deposit-api-workflow**

Full project name: Docker-demos/mobile-deposit-api-workflow

### Stage View

<table>
<thead>
<tr>
<th>Build App</th>
<th>Sonar analysis</th>
<th>Integration-test</th>
<th>Prepare release</th>
<th>Build Docker image</th>
<th>Test Docker image</th>
<th>Publish Docker image</th>
</tr>
</thead>
<tbody>
<tr>
<td>43s</td>
<td>7ms</td>
<td>21s</td>
<td>10s</td>
<td>5s</td>
<td>3s</td>
<td>32s</td>
</tr>
<tr>
<td>1min 23s</td>
<td>6ms</td>
<td>18s</td>
<td>10s</td>
<td>4s</td>
<td>2s</td>
<td>31s</td>
</tr>
</tbody>
</table>

Average stage times:
(Average full run time: ~2min 22s)

- **#17**: Oct 11 22:14 No Changes
- **#18**: Oct 11 22:13 No Changes
- **#15**: Oct 11 22:13 2 commits

Almost complete:
Pipeline Stages

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

image.inside

withServer
Build, unit test and package

SCM Checkout
mvn package
mvn sonar:sonar
mvn verify

image.inside

docker build
docker tag
notify cucumber

docker push

withServer
Build, unit test and package

```java
stage 'Build App' { 

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/cloudbbees/mobile-deposit-api.git'

        sh 'mvn -s settings.xml clean package'

    }

} 
```

Specify the Stage Name
Specify the slave label
Custom Build Env
Mount volume from slave
.m2 repo location
co and build
Defining a Docker Slave

Specify Image as template

Assign labels

- **Docker Template**
  - **ID**: kmadek/dind-jenkins

- **Labels**: docker

- **Usage**: Only build jobs with label restrictions matching this node

- **Credentials**: jenkins

- **Remote Filing System Root**: /home/jenkins

- **Remote FS Root Mapping**: 

- **Instance Cap**: 4

- **DNS**: 

- **Port bindings**: 1234

- **Bind all declared ports**: 

- **Hostname**: 

- **Advanced...**

- **Delete Docker Template**

**Add Docker Template**

List of Images to be launched as slaves
Test the app

- 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
notify
withServer
docker push

image.inside
Test the app

```java
node('docker') {

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

  ...

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

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

})

```
Build, test and publish Docker image

- **SCM Checkout**: mvn package, mvn sonar:sonar, mvn verify
- **Build and Unit Test App**
- **Sonar Analysis**
- **Int Test**
- **Prepare Release**
- **Build Docker Image**
  - docker build
  - docker tag
- **Test Docker Image**
  - docker run
  - notify cucumber
- **Publish Docker Image**
  - docker push

**image.inside withServer**
Build, test and publish Docker image

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 mobile-deposit-api.jar"
    mobileDepositApiImage = docker.build "harniman/mobile-deposit-api:${buildVersion}"
  }

...
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()
}

}
Tagged Image in Docker Hub

PUBLIC REPOSITORY

harniman/mobile-deposit-api 🌟

Last pushed: 10 minutes ago

<table>
<thead>
<tr>
<th>Tag</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>329 MB</td>
</tr>
<tr>
<td>22</td>
<td>329 MB</td>
</tr>
<tr>
<td>21</td>
<td>329 MB</td>
</tr>
<tr>
<td>20</td>
<td>329 MB</td>
</tr>
<tr>
<td>19</td>
<td>329 MB</td>
</tr>
</tbody>
</table>
Best of All: Jenkins Pipeline + Docker

```groovy
stage 'Build Source'
node('docker') {

docker.image('maven:3.3.3-jdk-8') {
    git 'https://github.com/cloudbbees/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 "cloudbbees/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!