

**using Docker
with Pipeline
because it's good.**

using Docker

Useful Docker Images



Useful Docker Images

- OpenJDK
 - `docker pull openjdk:7-jdk`
 - `docker pull openjdk:8-jdk`
- Maven
 - `docker pull maven:3-jdk-7`
 - `docker pull maven:3-jdk-8`
- Golang
 - `docker pull golang:1.7`
- Ruby
 - `docker pull ruby:2.3`
- Python
 - `docker pull python:2`
 - `docker pull python:3`

with Pipeline

Building a simple Java app



Iteration #0: A basic Jenkinsfile

```
node {  
    checkout scm  
  
    sh 'mvn clean install'  
  
    junit 'target/surefire-reports/**/*.xml'  
}
```

Iteration #0: Requirements

- JDK on the node
- Maven in the PATH for the Jenkins agent executing on the node
- Other requirements for build/test execution?

Iteration #1: Tool Installers

```
node {  
    checkout scm  
  
    withEnv(["JAVA_HOME=${tool 'jdk8' }",  
           "PATH+MAVEN=${tool 'maven3'}/bin:${env.JAVA_HOME}/bin"]) {  
        sh 'mvn clean install'  
    }  
  
    junit 'target/surefire-reports/**/*.xml'  
}
```

Iteration #1: Requirements

- Tool installers configured by Jenkins administrator for:
 - JDK8
 - Maven
- Developer creating Jenkinsfile must know "names" of tools configured.
- New tools require Jenkins administrator involvement

Iteration #2: Docker

```
node {  
    checkout scm  
  
    docker.image('maven:3-jdk-8').inside {  
        sh 'mvn clean install'  
    }  
  
    junit 'target/surefire-reports/**/*.xml'  
}
```

Iteration #2: Requirements

- Node has running Docker daemon

because it's good.

using Docker

Useful Docker Images



Useful Docker Images

- Redis
 - `docker pull redis:3`
- PostgreSQL
 - `docker pull postgres:9`
- MySQL
 - `docker pull mysql:5.7`
- Cassandra
 - `docker pull cassandra:3`

with Pipeline

Testing a simple Java app



Iteration #0: A basic Jenkinsfile

```
node {  
    checkout scm  
  
    sh 'redis-server & ; PID=$!; mvn test && kill $PID'  
  
    junit 'target/surefire-reports/**/*.xml'  
}
```

Iteration #0: Requirements

- JDK on the node
- Maven in the PATH for the Jenkins agent executing on the node
- Redis installed on the node
 - What happens when two teams need different versions?
 - How are upgrades handled?
- A desire to make sysadmins cry with reckless disregard for sane process management

Iteration #1: Docker

```
node {  
    checkout scm  
  
    docker.image('redis:3').withRun { c ->  
        docker.image('maven').inside("--link ${c.id}:redis") {  
            sh 'mvn test'  
        }  
    }  
  
    junit 'target/surefire-reports/**/*.xml'  
}
```

Iteration #1: Requirements

- Node has running Docker daemon

because it's good.

using Docker

Useful Docker Images



Useful Docker Images

- Your frontend app
- Your backend app

with Pipeline

Deploying a simple Java app



Iteration #0: A basic Jenkinsfile

```
node {  
    checkout scm  
  
    sh 'mvn release && ./trigger-production-deploy.sh'  
  
    mail to: 'team@example.com',  
        subject: "I think we deployed ${env.BUILD_ID}?"  
}
```

Iteration #0: Requirements

- Application "stack" (JVM, Ruby, Golang, etc) known ahead of time
- Actual deployment orchestration done outside of Jenkins

Iteration #1: Docker

```
node {  
    checkout scm  
  
    docker.build("initech/app:${env.BUILD_ID}").push()  
  
    sh './trigger-production-deploy.sh'  
  
    mail to: team@example.com,  
        subject: "I think we deployed ${env.BUILD_ID}?"  
}
```

Iteration #2: Docker

```
node {  
    def image = docker.build("initech/app:${env.BUILD_ID}")  
    image.push()  
    image.inside {  
        sh 'mvn acceptance-test'  
    }  
    image.push('latest')  
    sh './trigger-production-deploy.sh'  
}
```

because it's good.

thank you

<https://jenkins.io/s/docker>