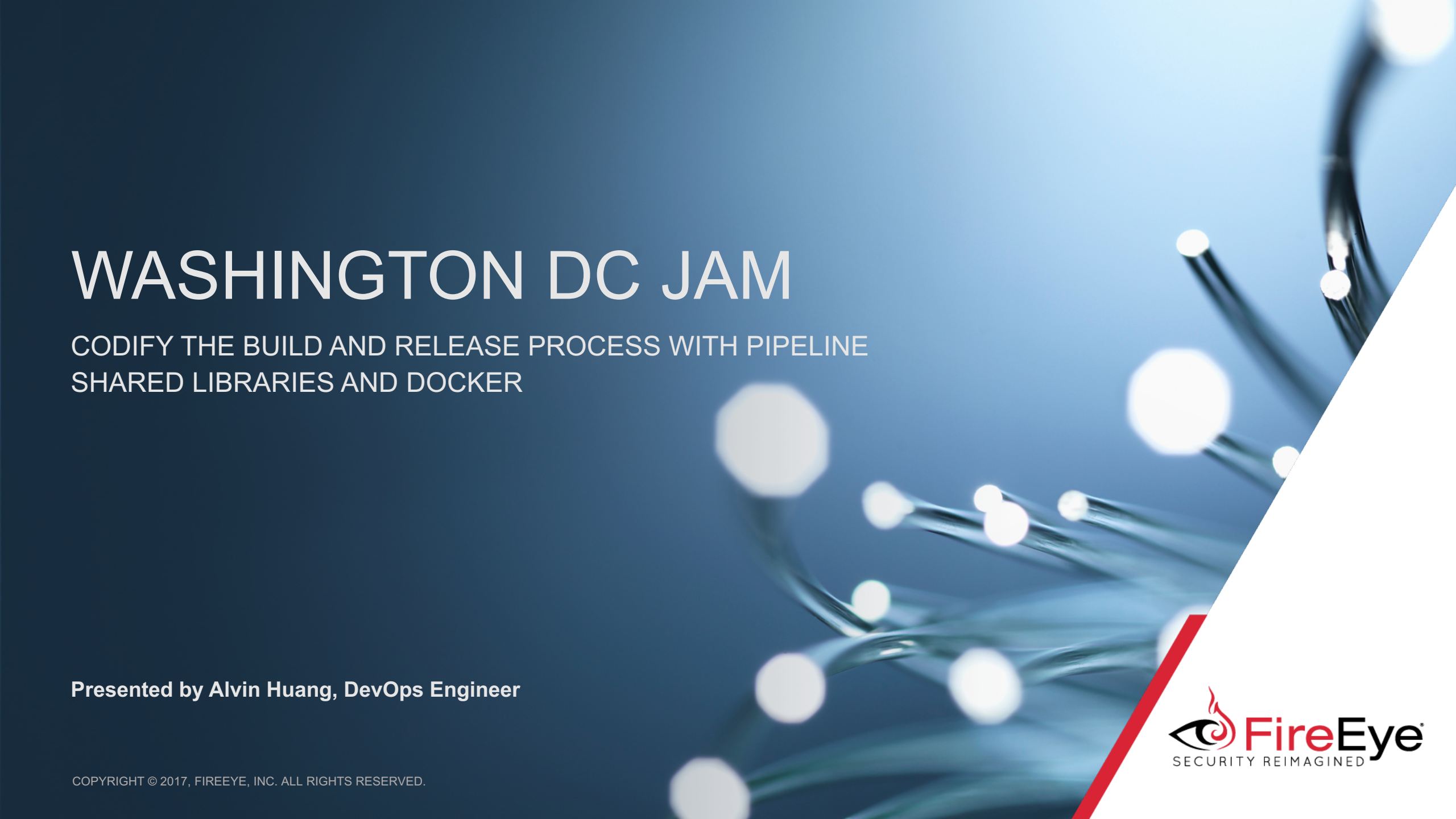
# WASHINGTON DC JAM

## CODIFY THE BUILD AND RELEASE PROCESS WITH PIPELINE SHARED LIBRARIES AND DOCKER

**Presented by Alvin Huang, DevOps Engineer**



FireEye
SECURITY REIMAGINED

# Who Am I

- DevOps Engineer, GSI @ FireEye
- Twitter: @RealAlvinHuang, #jenkins IRC: ahuang
- alvin.huang@fireeye.com

FireEye™




FireEye

# Tonight's Agenda

- GSI Overview
- What is Pipeline?
- The Great Migration; Why Pipeline? Why Docker?
- How we did this
- Lessons Learned
- Benefits for the Future




FireEye

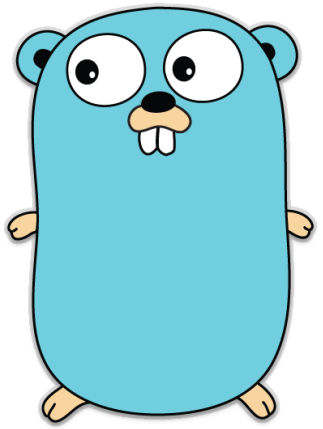
# GSI Overview

- Unified global organization combining Mandiant, FireEye iSIGHT Intelligence, and FireEye as a Service operations

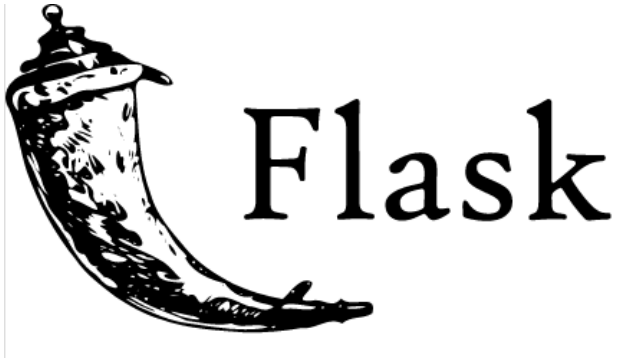
- **DevOps Goal**: Make the feedback loop between intelligence on the front lines to capabilities in software, smaller

# What We Build











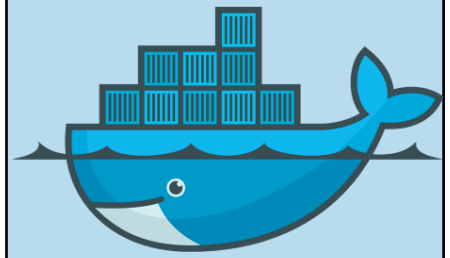


FireEye

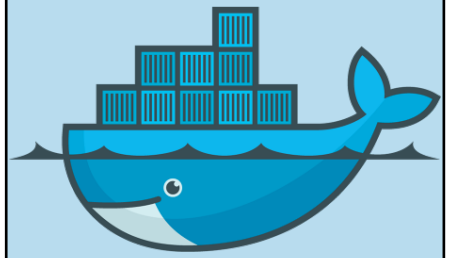
# How We Build

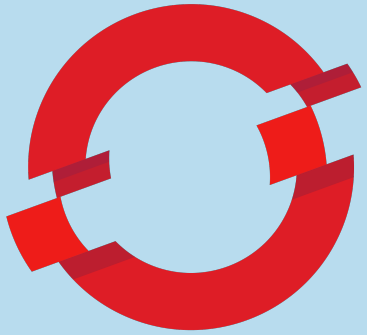
**Code**

**Build**



**Test**





OPENSHIFT

**Deliver**

yum
yellowdog updater modified

**Deploy**

p




FireEye

# What is Jenkins Pipeline?

- Pipeline as Code - Jenkins DSL to help users model, orchestrate, and visualize entire delivery pipelines
- Pipeline Stage View
- Durability – can survive planned and unplanned Jenkins master outages
- Built in support for Git/Github branches

*Jenkinsfile (Declarative Pipeline)*

```
pipeline {
    agent { docker 'python:3.5.1' }
    stages {
        stage('build') {
            steps {
                sh 'python --version'
            }
        }
    }
}
```

*Jenkinsfile (Scripted Pipeline)*

```
/* Requires the Docker Pipeline plugin */
node('docker') {
    checkout scm
    stage('Build') {
        docker.image('python:3.5.1').inside {
            sh 'python --version'
        }
    }
}
```




FireEye

# THE GREAT MIGRATION

# Scenario

- Consolidate datacenters
- Migrate Jenkins off a shared instance
- ~60 days
- ~150 Freestyle jobs
- ~20 different build agents
- Upgrade from 1.x to 2.x





FireEye

# Legacy Jenkins Jobs

- Jenkins primarily used to run tests and build RPMs
- SCP to publish RPMs to yum server
- Ad-hoc Github triggers
- No owner/SME for Jenkins




FireEye

# Goals

- Standardize and make consistent CI pipelines
- Jenkins SME = DevOps team
- Make new app onboarding easy
    - Build abstraction that covers ~90% of jobs
- Jenkins jobs tracked in git
- Decouple tools from Jenkins/shell => Docker




FireEye

# Migration Strategies

| Migration Strategy | LOE | Advantages | Disadvantages |
|---|---|---|---|
| Copy XMLs from old server to new server | Minimal | - Easy, no extra work | - Tech debt<br>- Global changes are hard<br>- Job config not in SCM |
| Create Freestyle Jenkins job templates | Moderate | - Global changes are easier<br>- Hide much of the configuration from end users<br>- Make Freestyle jobs DRY | - Tech debt<br>- Debugging a job is hard<br>- Job config not in SCM<br>- Global changes can break all builds |
| Convert Freestyle jobs to Pipeline jobs | Significant | - Job configuration in SCM | - Learn pipeline syntax<br>- Large job footprint<br>- Global changes are hard<br>- Too complex for many users<br>- Subject to repetition and tech debt |
| Create a Jenkins Shared Library for Pipeline jobs | Moderate | - Jobs in SCM<br>- Standardized and consistent<br>- Easy on-boarding, just needs parameters<br>- Job configuration hidden from user | - Learn pipeline syntax<br>- Need owner for library<br>- Global changes can break all builds |




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | |
| Jobs were not DRY | |
| | |
| | |
| | |
| | |




FireEye

# Solution: Build Wrapper

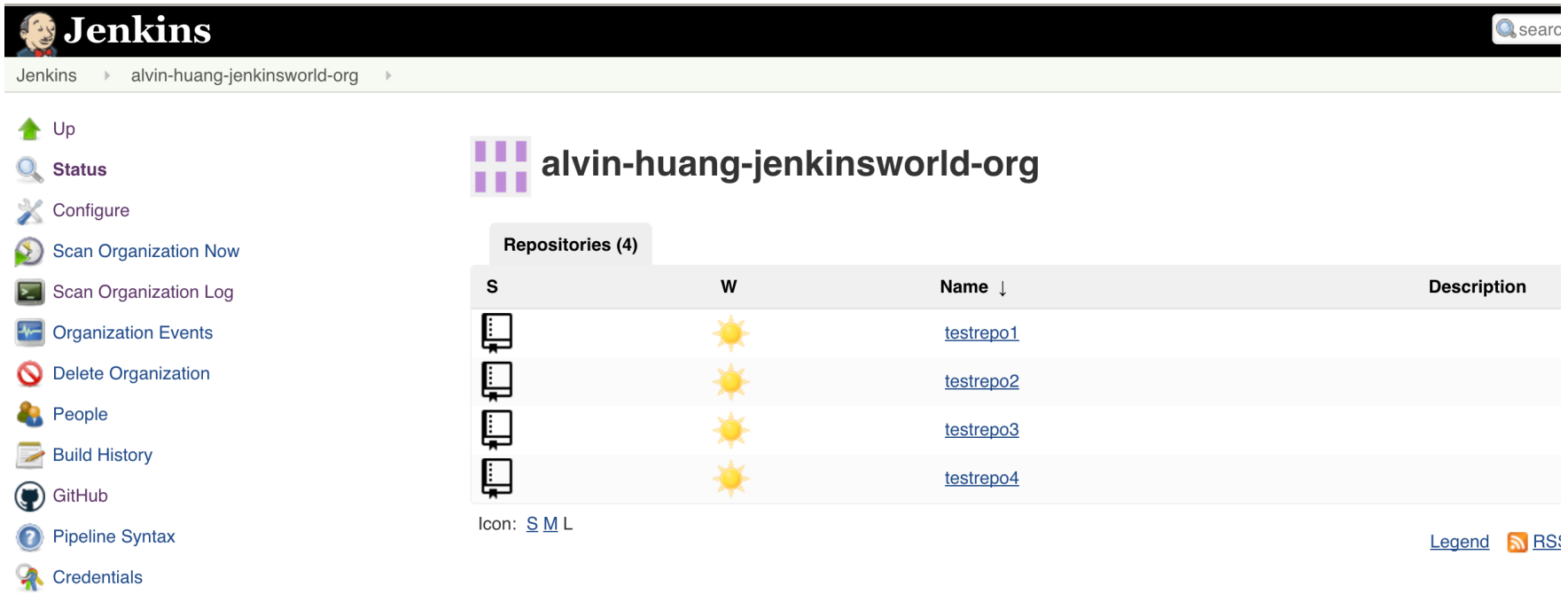
```
standardBuild {
    machine          = 'docker'
    dev_branch       = 'develop'
    release_branch   = 'master'
    artifact_apttern = '*.rpm'
    html_pattern     = [keepAll: true, reportDir: '.', reportFiles: 'output.html', reportName: 'OutputReport']
    dev_repo         = 'pipeline-examples-dev'
    prod_repo        = 'pipeline-examples-prod'
    pr_script        = 'make prs'
    dev_script       = 'make dev'
    release_script   = 'make release'
}
```




FireEye

# Solution: Use Github Organizations

- Index every branch of every repo that has a Jenkinsfile
- Configure one Github hook on the organization level



# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| | |
| | |
| | |
| | |




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | |
| | |
| | |
| | |




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | |
| | |
| | |




FireEye

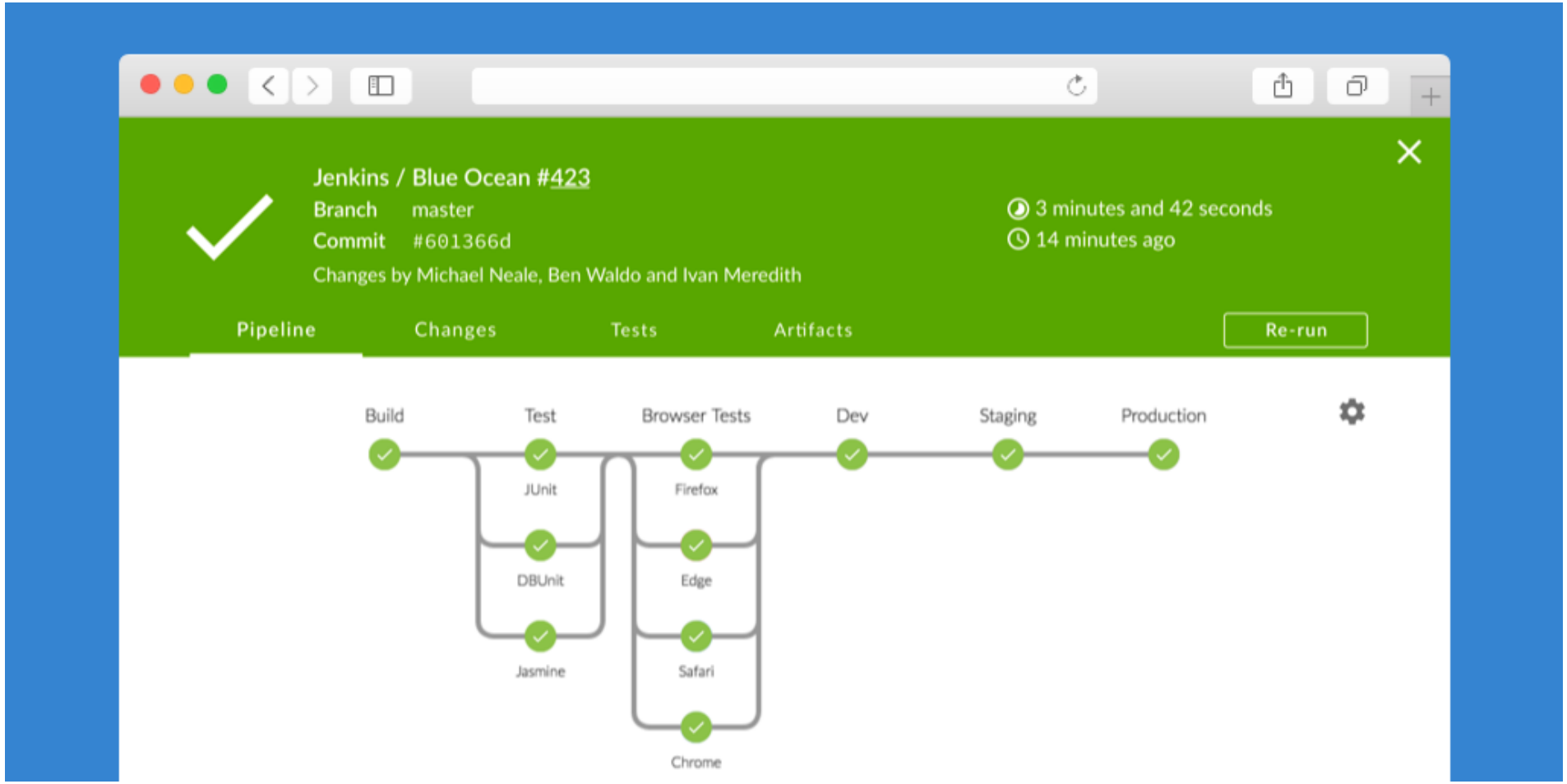
# Solution





FireEye

# Solution







FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| | |
| | |




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| Very difficult to make global changes | |
| | |




FireEye

# Solution

```
def call(config) {
  stepCheckout()
  stepBuild()
  ....
}
```

```
def call(config) {
  stepCheckout()
  if(config.machine == 'docker') {
    stepDockerfileLint()
  }
  stepBuild()
}
```

yourBuildWrapper.groovy




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| Very difficult to make global changes | |
| | |

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| Very difficult to make global changes | *Library changes have an effect on all builds |
| | |




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| Very difficult to make global changes | *Library changes have an effect on all builds |
| Monolithic Build Agents | |

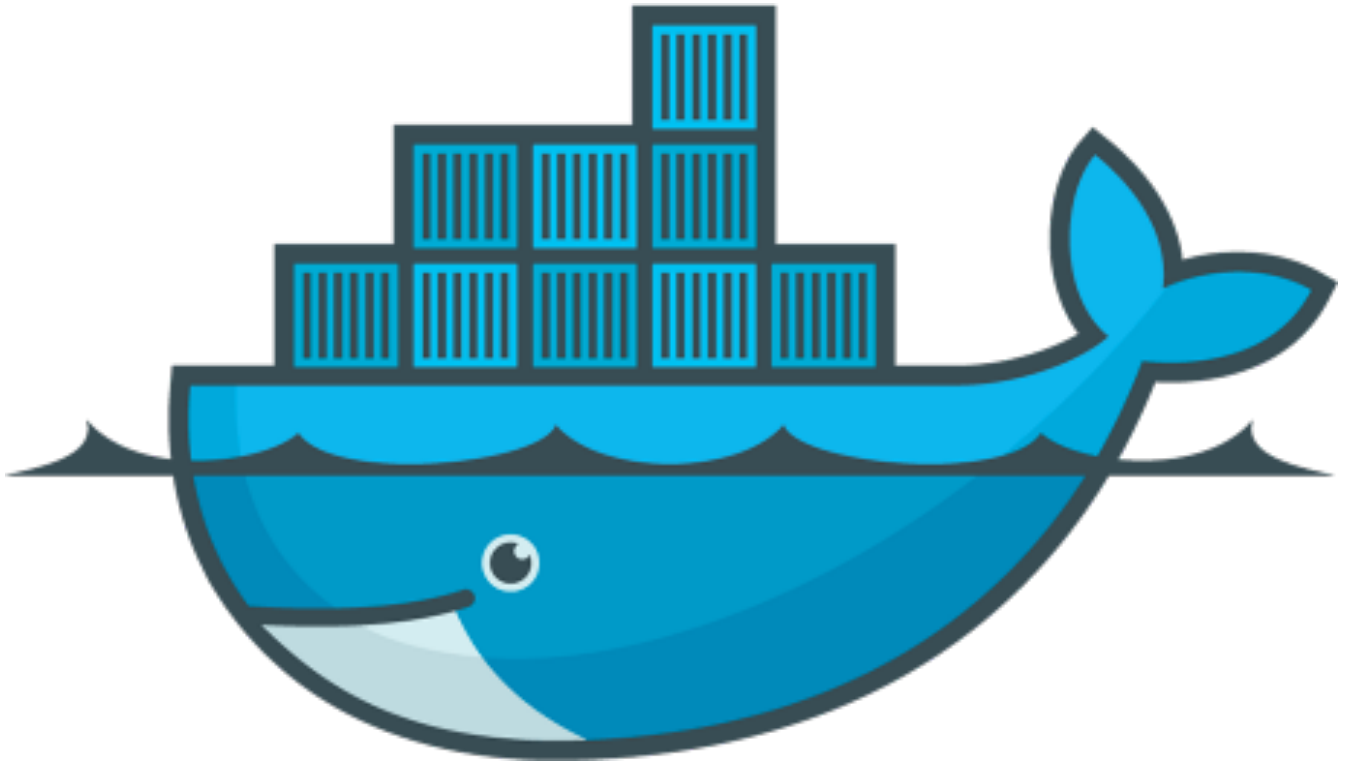
# Migrating Monolithic Build Agents

- No configuration management
- What packages and versions were installed?
- Who runs on what agent?
- Changes on a node may affect another team sharing that same agent




FireEye

# Solution to Monolithic Build Agents






FireEye

# Sample Dockerfile

```
FROM centos:centos7

# install Puppet5 repo
RUN yum install -y https://yum.puppetlabs.com/puppet5/puppet5-release-el-6.noarch.rpm

RUN yum install -y puppet-agent \
    rubygems && \
    yum clean all

ENV PATH="/opt/puppetlabs/puppet/bin:$PATH"

RUN gem install puppet-lint \
    puppet-syntax \
    yaml-lint && \
    gem cleanup all

WORKDIR "/code"
```

```
FROM faas/el7-python:base

RUN yum -y install \
    czmq-devel \
    gcc-c++ \
    python-virtualenv \
    python-pip \
    prelink \
    postgresql-devel \
    mysql-devel \
    libpcap-devel \
    libffi-devel \
    rpm-build \
    git && \
    yum clean all

RUN pip install fabric
```




FireEye

# Why Pipeline and Shared Libraries?

| Problem | Solution |
| --- | --- |
| Overhead with Freestyle Jobs | Use a pipeline library wrapper; Github Organizations |
| Jobs were not DRY | Use a pipeline library wrapper |
| Job configuration drift | Jenkinsfiles in SCM |
| Difficult to view job configurations and build steps | Jenkinsfiles + Stage View |
| Very difficult to make global changes | *Library changes have an effect on all builds |
| Monolithic Build Agents | Docker |

# What about…

- Jenkins Job Builder
  - Simple jobs defined by relatively complex YAML
  - Verbose
  - Needs abstraction for novice users

- Job DSL Plugin
  - Programmatically seeds/creates Jenkins jobs
  - Jenkins Pipeline + Github Organizations can be used instead
  - No logic between jobs (ie: no Pipeline ‘input’ step)




FireEye

# Starting Point

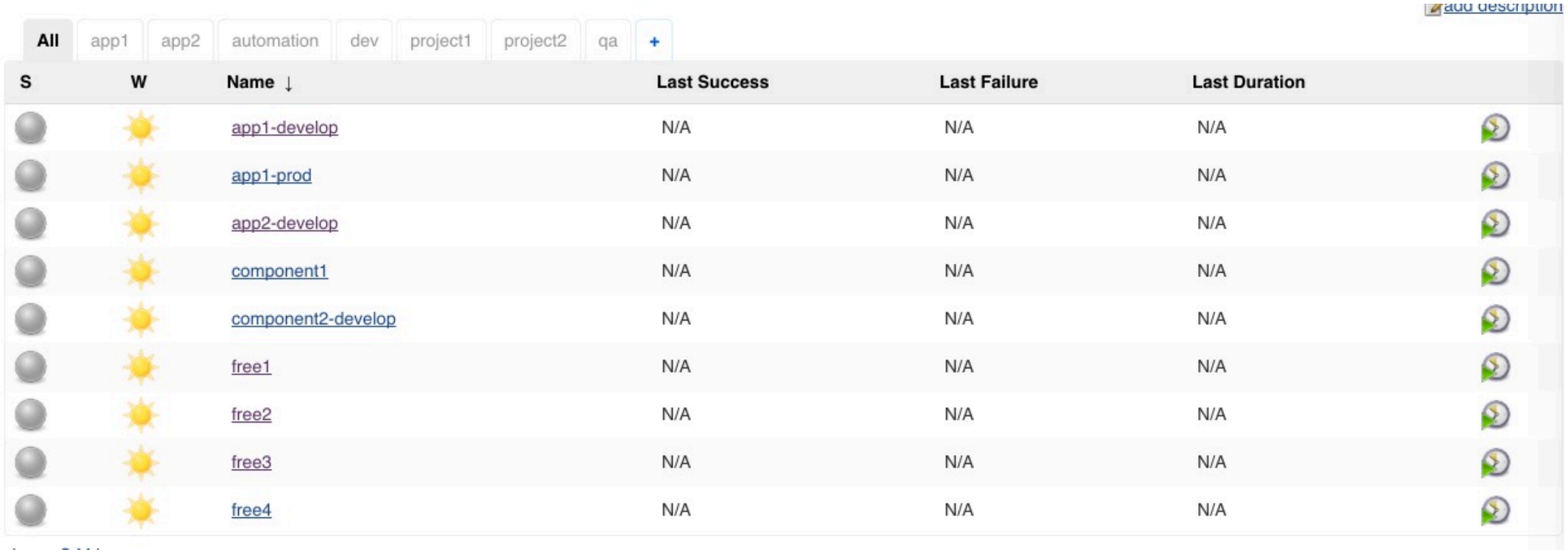
- New Jenkins 2.x instance
- Stable build nodes built with Puppet
  - CentOS 6/7 hosts
- Working Docker integration




FireEye

# BEFORE PIPELINE

# Phase 1 (Pre-migration)

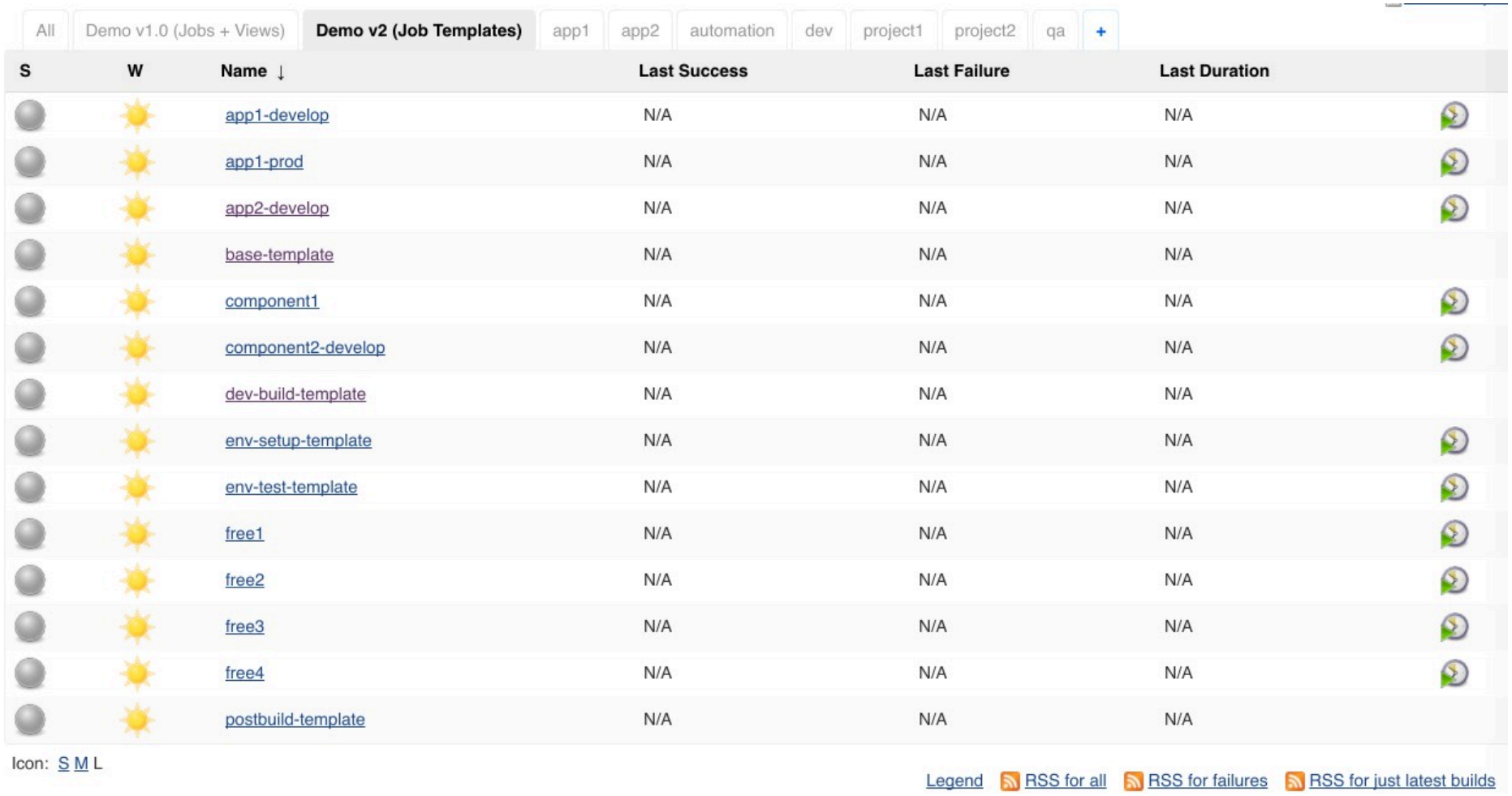






FireEye

# Phase 2 (Pre-migration)

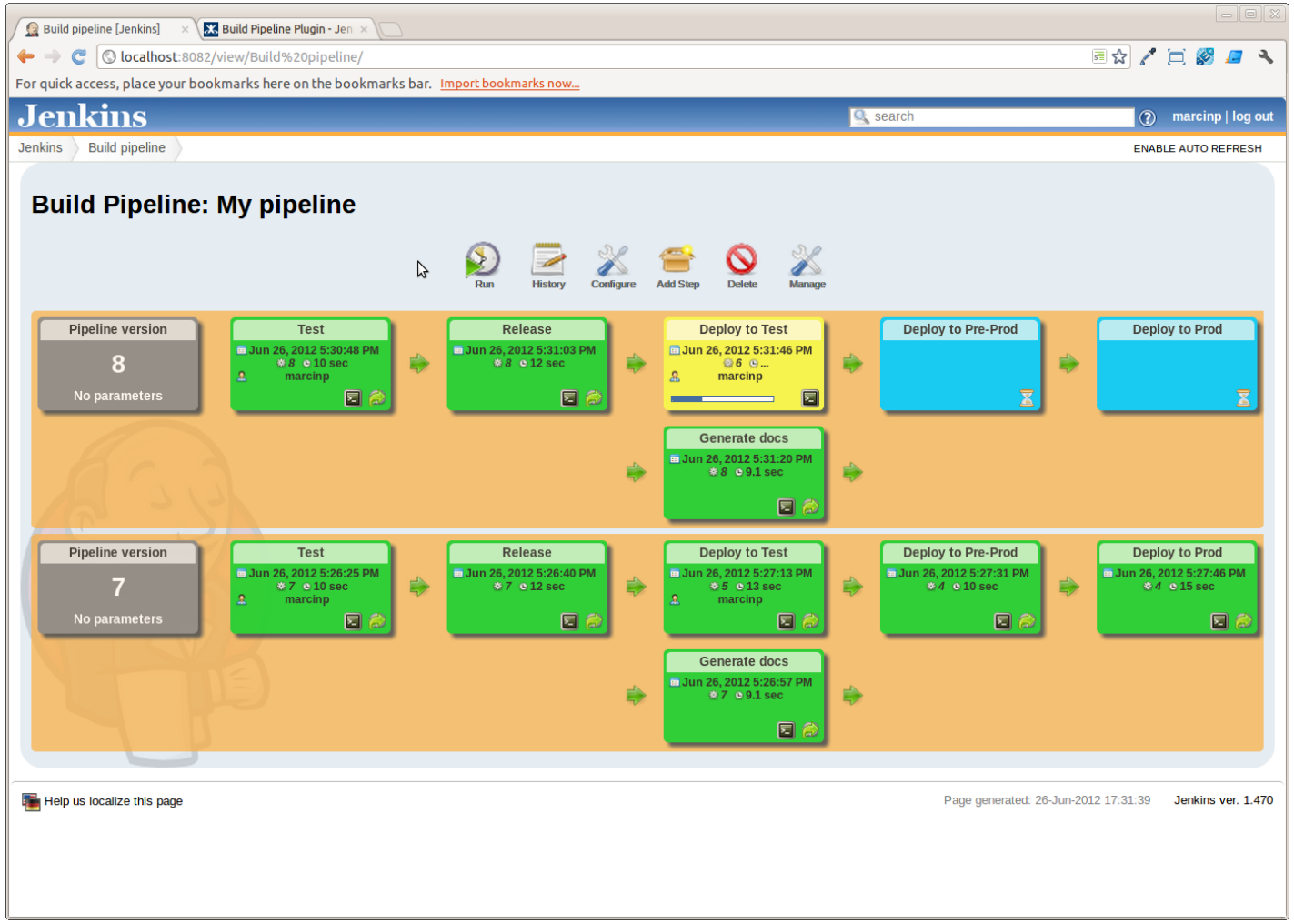






FireEye

# Visualization







FireEye

# THE NEW WORLD




FireEye

# Inspiration

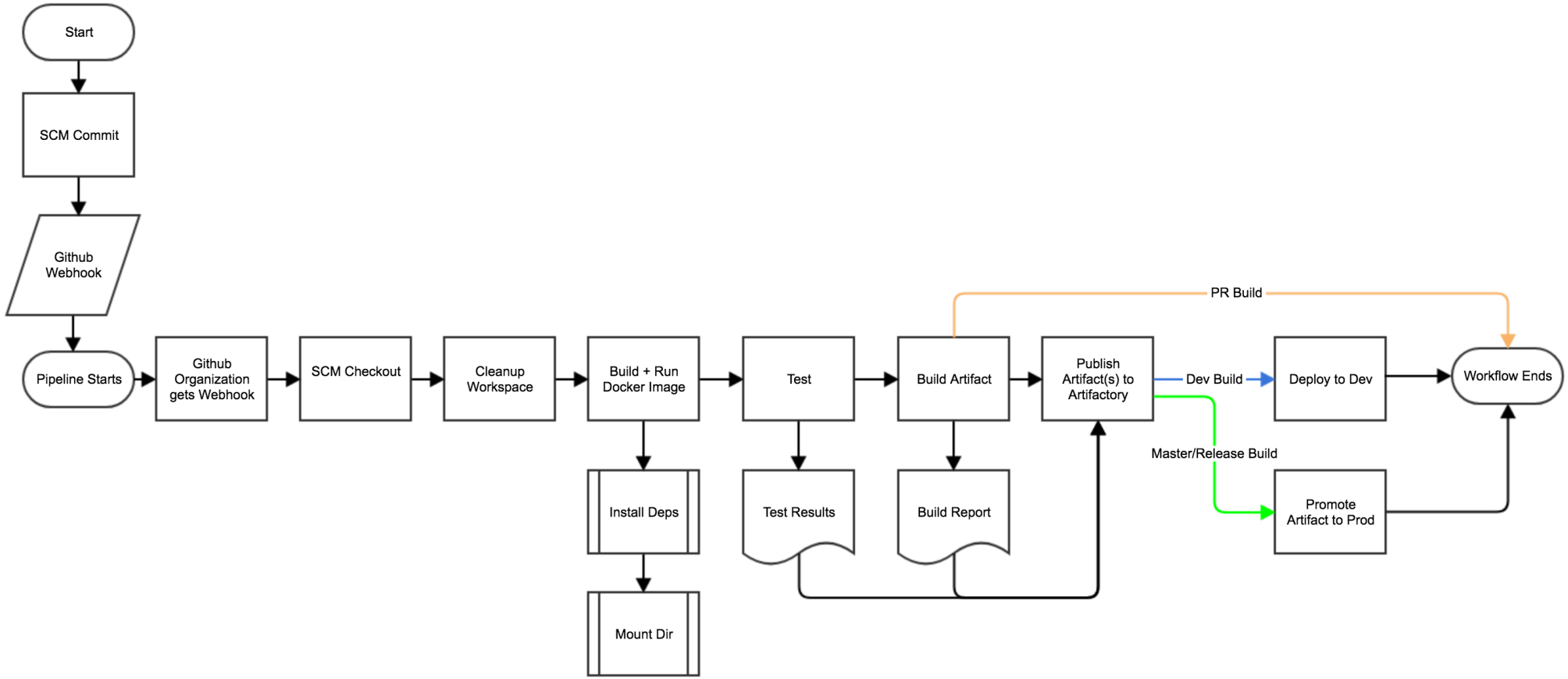
- https://github.com/jenkinsci/simple-build-for-pipeline-plugin -michaelneale

- Build a wrapper called simpleBuild that takes in parameters as a closure and runs logic




FireEye

# Jenkins Pipeline Workflow



# Example Jenkinsfile

```
standardBuild {
    machine          = 'docker'
    dev_branch       = 'develop'
    release_branch   = 'master'
    artifact_apttern = '*.rpm'
    html_pattern     = [keepAll: true, reportDir: '.', reportFiles: 'output.html', reportName: 'OutputReport']
    dev_repo         = 'pipeline-examples-dev'
    prod_repo        = 'pipeline-examples-prod'
    pr_script        = 'make prs'
    dev_script       = 'make dev'
    release_script   = 'make release'
}
```




FireEye

# Example Dockerfile

```
FROM faas/el7-python:base

RUN yum install -y python-virtualenv \
    rpm-build && \
    yum clean all
```




FireEye

# Resulting Pipeline

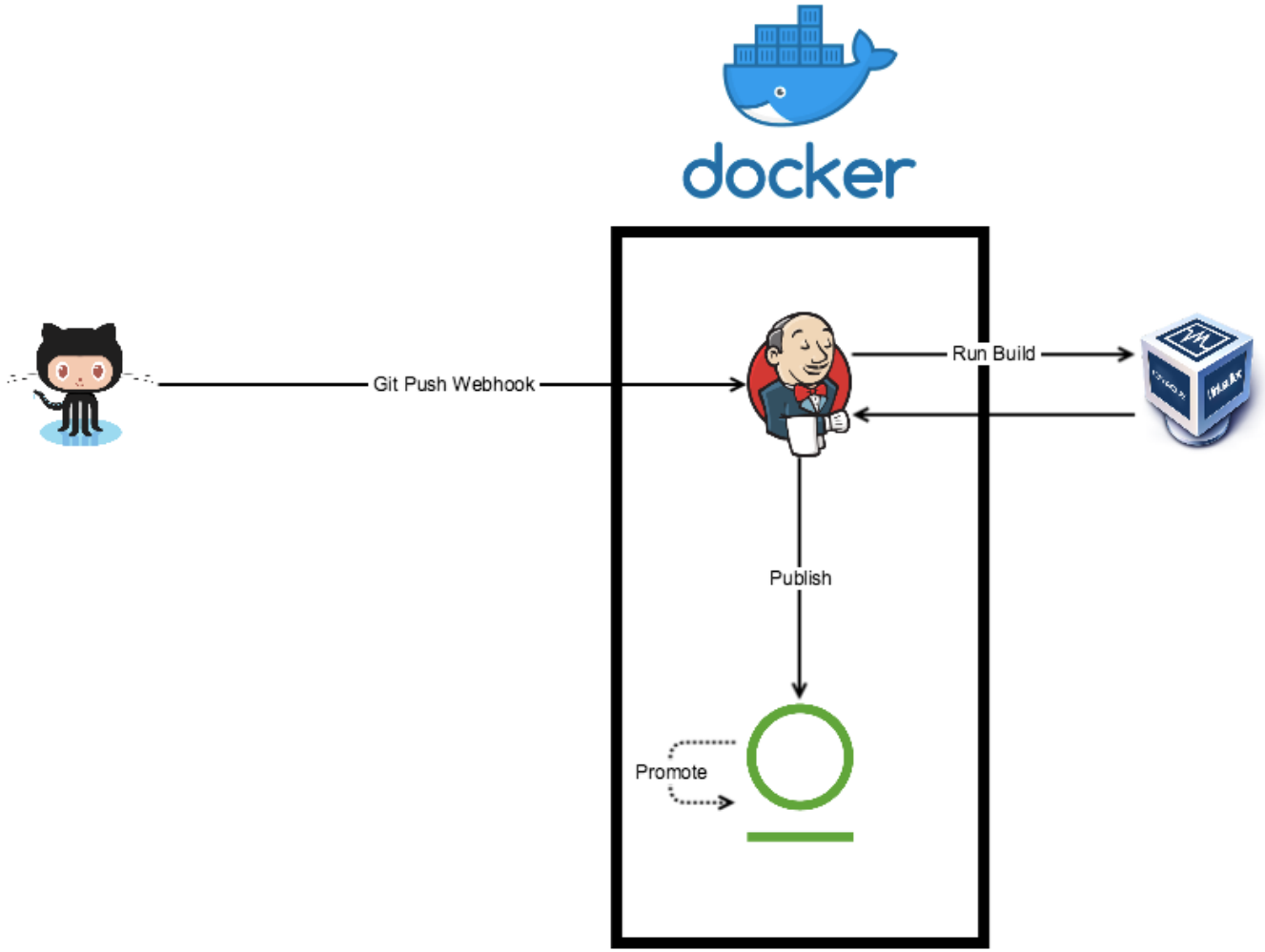




FireEye

# DEMO

# Standard Build

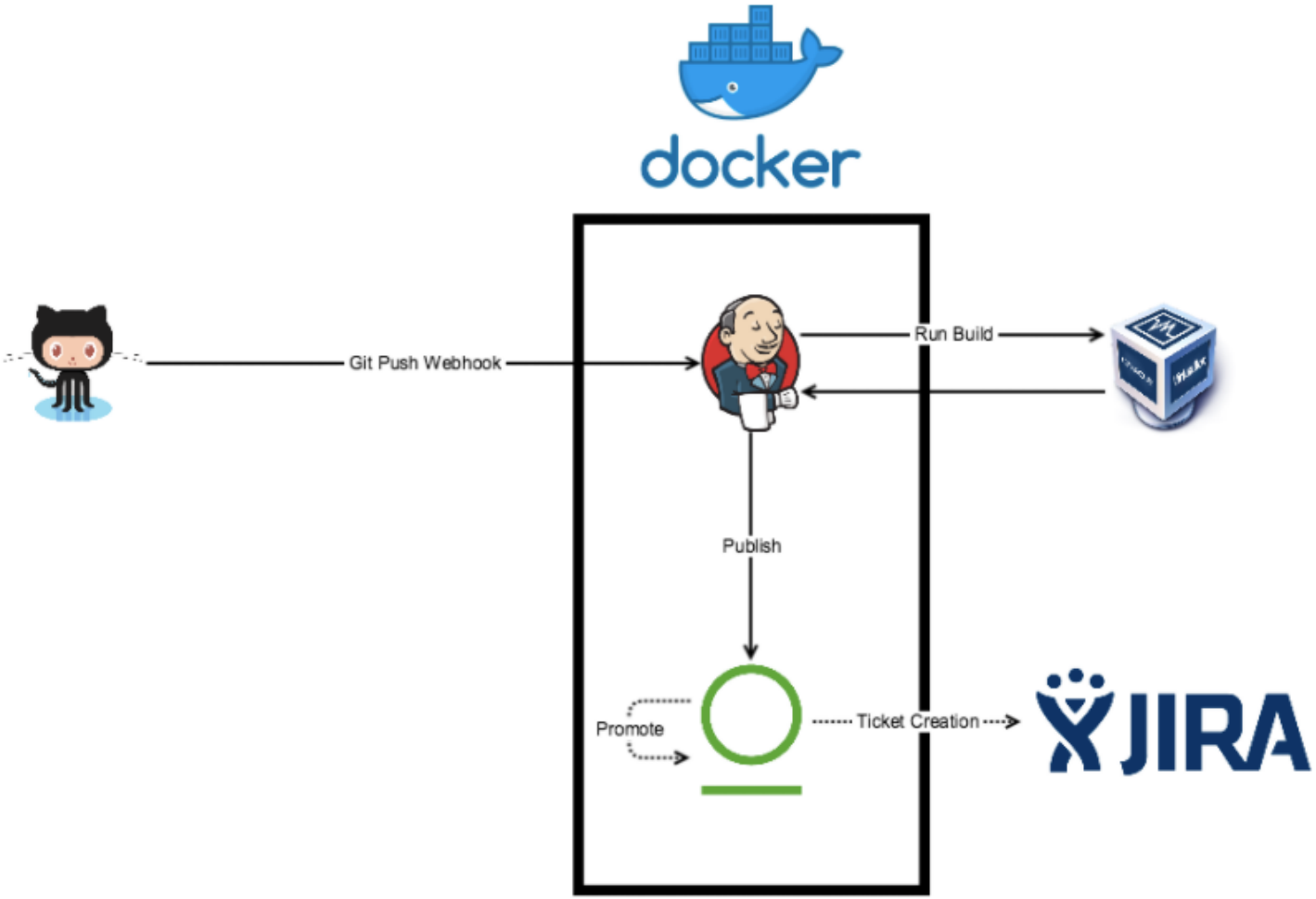






FireEye

# Standard Build w/JIRA







FireEye

# How do we do this?

- Find a *passionate* liaison from each team
- Learn their workflow
    - There's probably a freestyle job already
- Start pipelining their workflow in modular pieces:
    - Checkout code
    - Run build script
    - Archive artifacts
    - Publish to Artifactory
    - Promote artifact
    - Send email
- Build a wrapper to call those functions
- Repeat for another team/app, reusing what you just made




FireEye

# Reap the Benefits

- Give Devs/QA their time back
- Standardization of jobs and workflows
- Standardization of build and test environments
- Jenkins maintenance and resilience made easy by decoupling tools

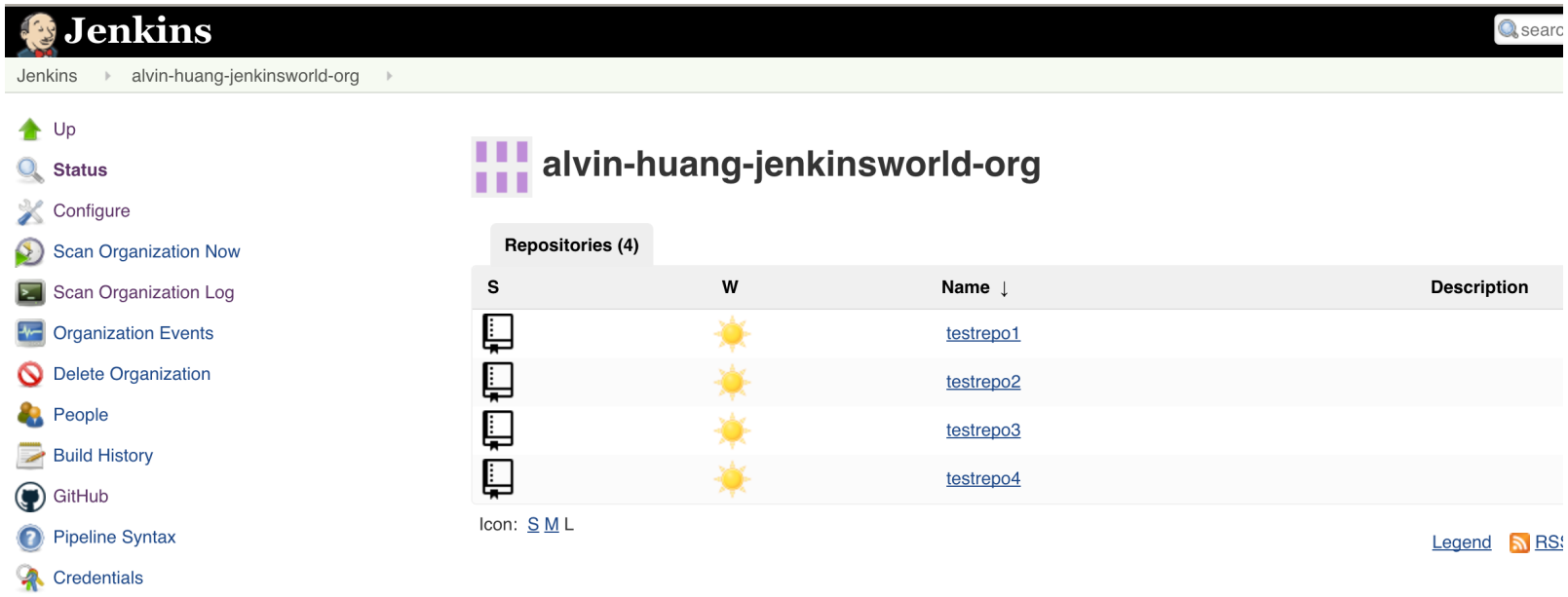



FireEye

# LESSONS LEARNED

# Use Github Organizations

- Index every branch of every repo that has a Jenkinsfile
- Configure one Github hook on the organization level



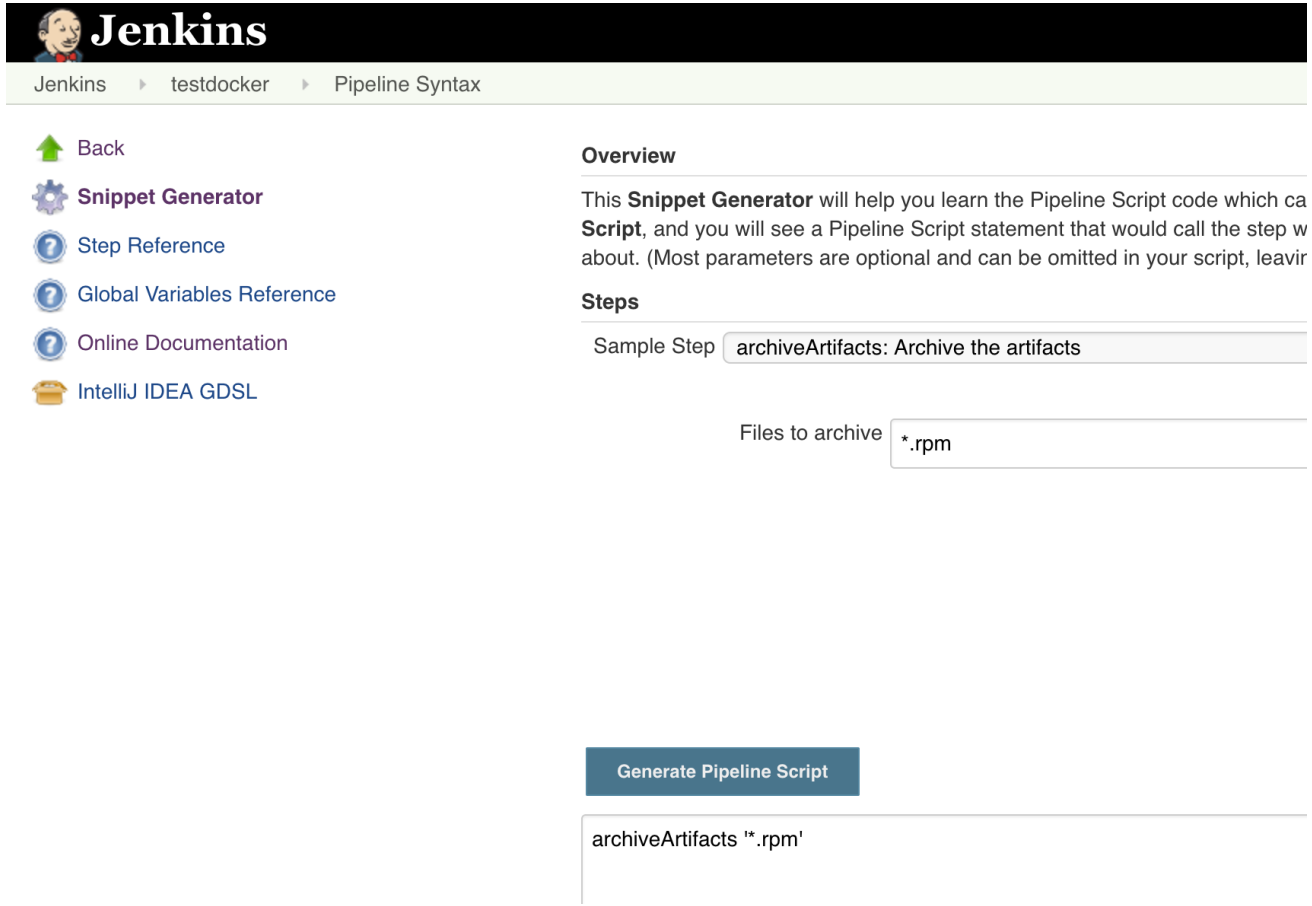
# Make Steps Modular

- What does this step do?
  - Does A ‘AND’ B; break it apart
- Namespace them
  - stepCheckout.groovy – Custom Pipeline Step
  - standardBuild.groovy – Wrapper




FireEye

# Leverage the Snippet Generator







FireEye

# Benefits of Shared Libraries For Our Future

- Migration of Github
- Upgrade to Puppet 5
- Consolidation of Artifactory
- Application stacks in Openshift
- Deployments through Rundeck, Ansible or Fabric
- Scripts in ‘vars’ can be called in Declarative Pipeline




FireEye

# Upgrade to Puppet 5

```
FROM centos:centos6

# install Puppet5 repo
RUN yum install -y https://yum.puppetlabs.com/puppet5/puppet5-release-el-6.noarch.rpm

RUN yum install -y puppet-agent \
    rubygems && \
    yum clean all

ENV PATH="/opt/puppetlabs/puppet/bin:$PATH"

RUN gem install puppet-lint \
    puppet-syntax \
    yaml-lint && \
    gem cleanup all

WORKDIR "/code"
```

Dockerfile

```
def linter = '''
    docker build -t puppet-linter:latest .
    docker run -i --rm -v $(pwd):/code puppet-linter:latest bash linter.sh
    '''

standardBuild {
  machine          = "docker"
  artifact_pattern = null
  release_branch   = '^(?!PR-).*'
  release_script   = linter
  dev_script       = linter
  pr_script        = linter
}
```

Jenkinsfile




FireEye

# Go Forth and Pipeline!




FireEye

# Thanks!

- alvin.huang@fireeye.com | @RealAlvinHuang | ahuang




FireEye

# THANK YOU



FireEye