DevOps World

Jenkins World

Jenkins and Kubernetes Secret Agents in the Cloud

Mandy Hubbard
@DevMandy

Mandy Hubbard

Software Engineer / QA Architect

- Over 18 years in Software Development, Quality Evangelism
- Continuous Integration / Delivery Pipeline Developer
- Speaker and Developer Advocate

I ❤ Jenkins

@DevMandy

Today's Session

- The changing landscape of software development
- Achieving the promise of microservices
- CI/CD in a microservices world
- My journey to Kubernetes

- And if you remember only three things …
  - Jenkins' capabilities continue to grow as technology changes
  - You should take advantage of ALL of Jenkins' capabilities
  - YOU, too, can spin up a Jenkins' environment in Kubernetes

The Journey from Monolith to Microservices

The Monolith

Distributed Applications

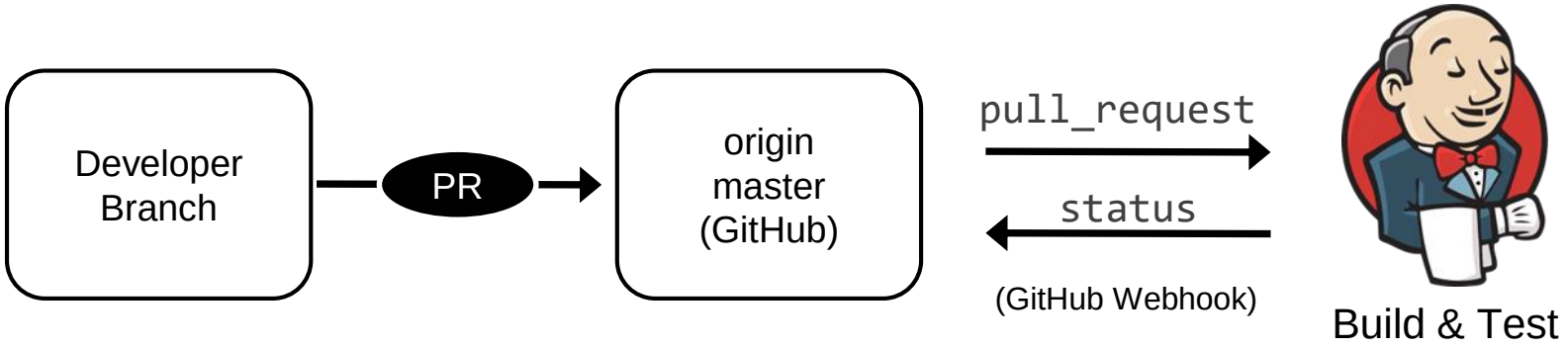
Microservices

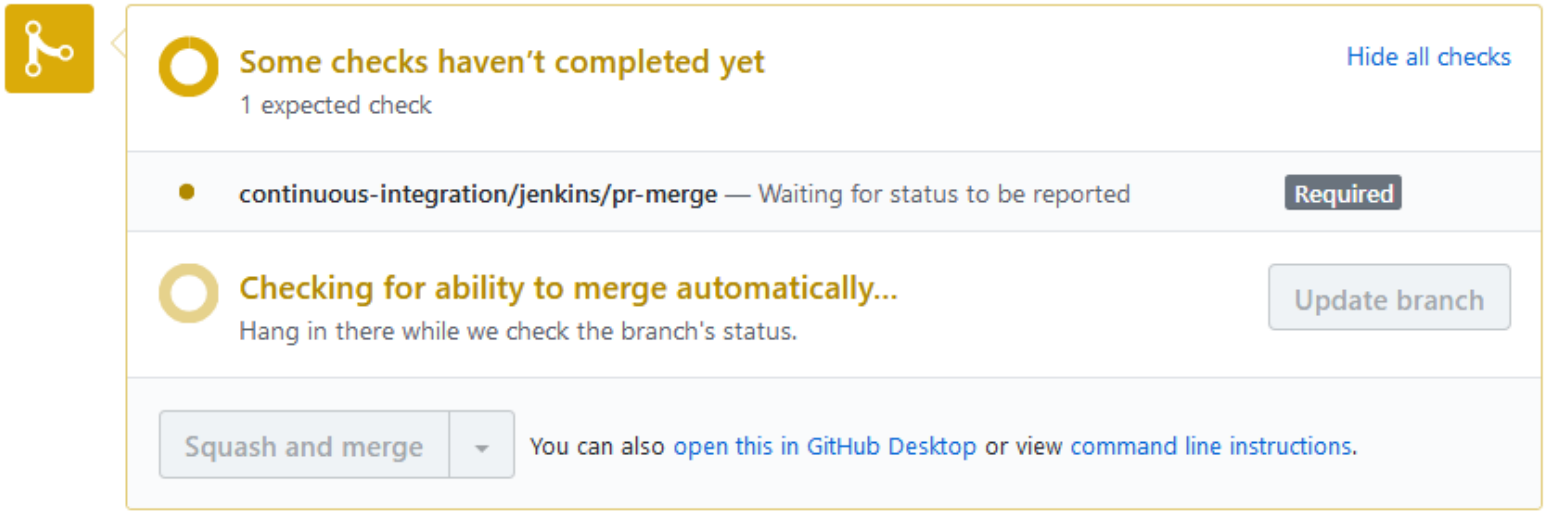
Why Microservices?

- Deploy services independently
- Make small, isolated units of change
- Scale at a more granular level
- Use the best tool for each task

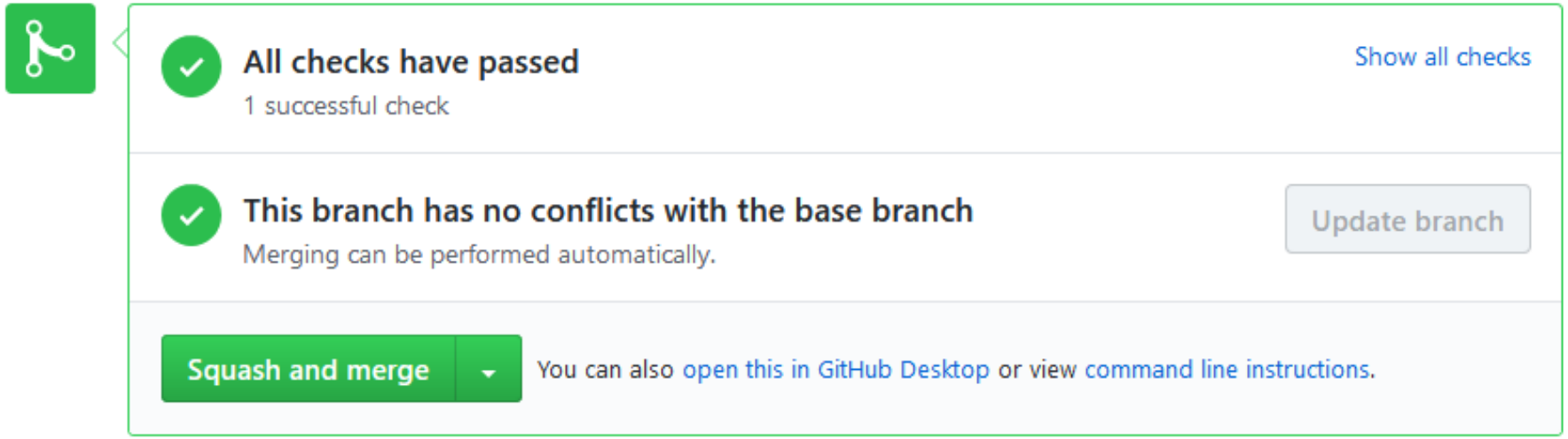
Reduced cost + Reduced risk

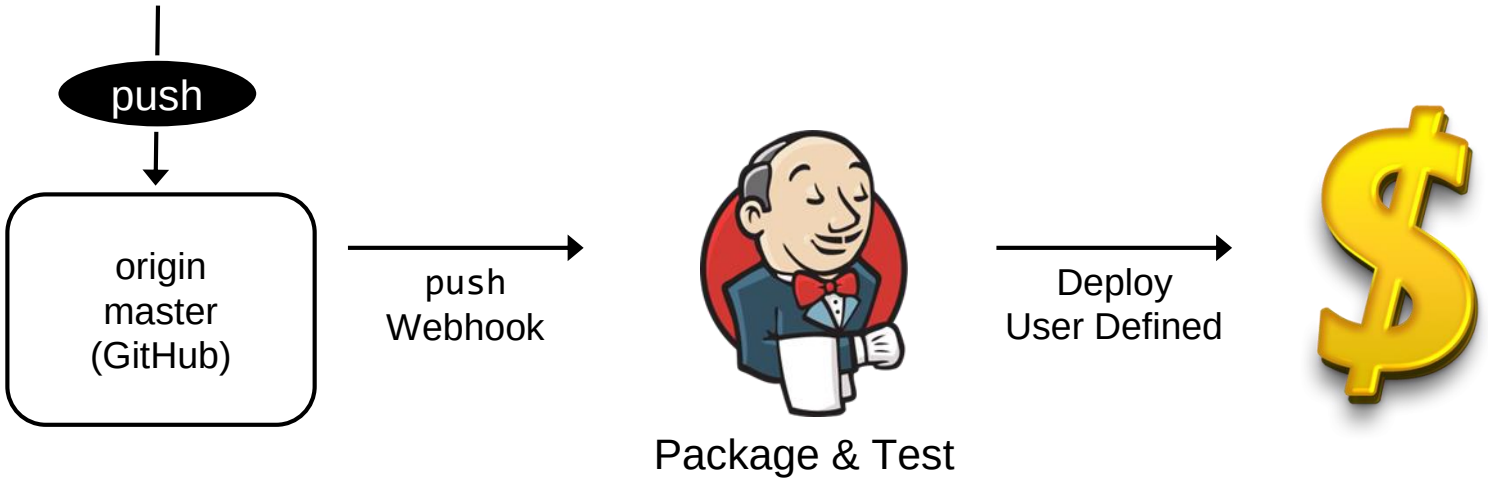
Continuous Integration



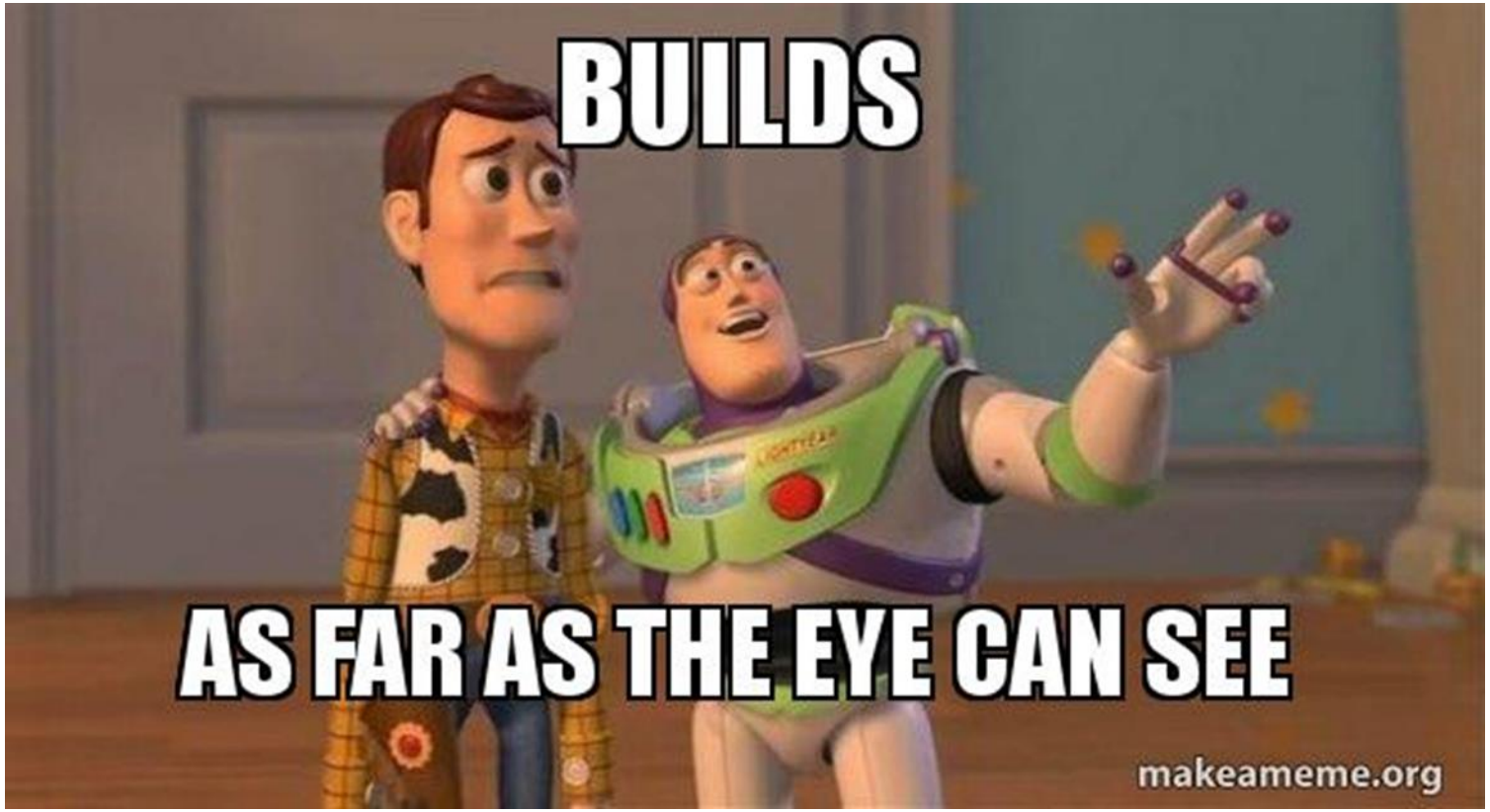


Continuous Deployment/Delivery





Continuous Integration x Microservices =



CI/CD Platform Evolution Considerations

- Can get away with Master only
- Consistent language and tools
- Infrequent builds
- CI helpful but optional

- Distributed build environment
- Potentially different language and tools
- CI required, CDep extremely useful

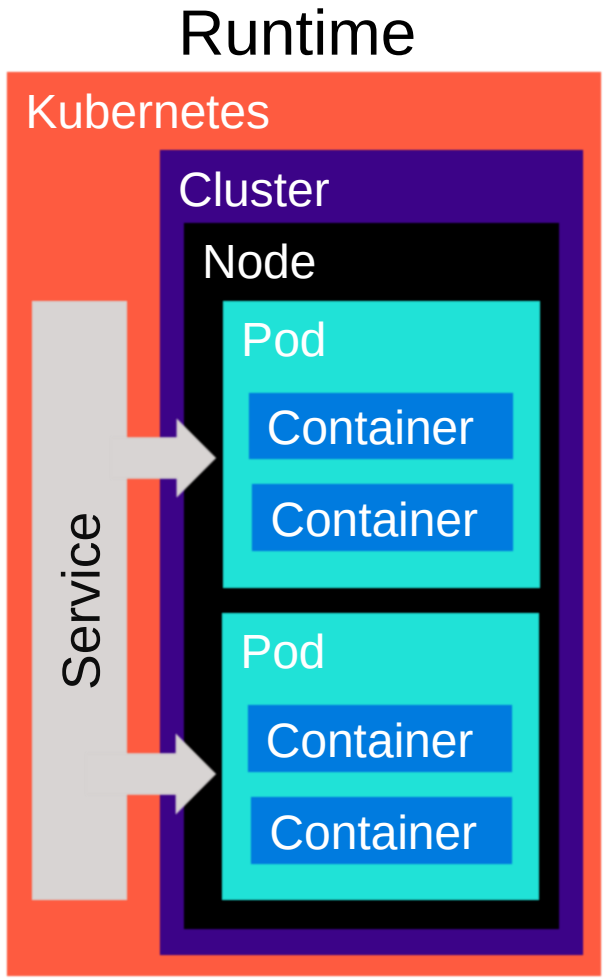
- Multiple agents required
- Definitely different language and tools (and different versions of them)
- CI/CDep/CDel required

Evolution of (my) Jenkins Agents

- Execute all builds on Master
- Single Agent with all the tools for any service and then clone it to “scale”
- Individual Agent for each toolchain and then manage idle agents
- Dockerized Agents
  - Docker in Docker (DinD)
  - Docker outside of Docker (DooD)
  - External Docker host
  - Dynamically created ephemeral Docker hosts
- Let’s do Kubernetes!

Kubernetes Key Concepts

- Node
- Cluster
- Pod
- Service
- Helm Charts



Anatomy of a Helm Chart

Let’s go to GitHub!

Customizing the Install with values.yaml

```
Master:
  InstallPlugins:
    - kubernetes:1.7.1
    - workflow-aggregator:2.5
    - workflow-job:2.21
    - credentials-binding:1.16
    - git:3.9.1
    - google-oauth-plugin:0.6
    - google-source-plugin:0.3
  Cpu: "1"
  Memory: "3500Mi"
  JavaOpts: "-Xms3500m -Xmx3500m"
  ServiceType: ClusterIP
Agent:
  Enabled: false
Persistence:
  Size: 100Gi
NetworkPolicy:
  ApiVersion: networking.k8s.io/v1
rbac:
  install: true
  serviceAccountName: cd-jenkins
```

Pre-requisites for Helm Install

1. Create a Compute Engine Network
2. Provision a Kubernetes cluster
3. Install Helm on the cluster
4. Add yourself to the cluster's RBAC
5. Grant Tiller, the server-side of Helm, the `cluster-admin` role in your cluster
6. Initialize Helm
7. Install Jenkins using Helm

Installing Helm Chart

```
~$ ./helm install -n cd stable/jenkins -f jenkins/values.yaml --version 0.16.6 --
```

Configure the Jenkins Master

- Configure credentials set for Kubernetes cluster
- Create a cloud configuration for the Kubernetes cluster
- Create a pod template for the Agent
- Create a container template for the Agent

Let’s go to Jenkins!

Use the Agent in Your Jenkinsfile

```
node(`super-pod') {
    stage('Checkout') {
        checkout scm
    }
    stage('Build'){
        container(`golang') {
            // This is where we build our code.
        }
    }
}
```

Inspect Running Pods

```
mandy_hubbard@Jenkins-world-213716:~$ kubectl get pods
NAME                          READY          STATUS         RESTARTS       AGE
cd-jenkins-7c786475dd-1fh14   1/1            Running        0              1d
super-pod-6mvxm-tbjv1         1/1            Running        0              6s
```

```
mandy_hubbard@Jenkins-world-213716:~$ kubectl get pods
NAME                          READY          STATUS         RESTARTS       AGE
cd-jenkins-7c786475dd-1fh14   1/1            Running        0              1d
super-pod-27x3d-s2ckg         1/1            Running        0              7s
```

Jenkinsfile, FTW

```
podTemplate(name: 'test-pod', label: 'test-pod', containers: [
containerTemplate(name: 'golang', image: 'golang:1.9.4-alpine3.7'),
containerTemplate(name: 'docker', image:'trion/jenkins-docker-client'),
],
volumes: [
    hostPathVolume(mountPath: '/var/run/docker.sock',
                   hostPath: '/var/run/docker.sock'),])
{
    //node = the pod label
     node('test-pod'){
          stage('Checkout') {
          checkout scm
         }
         //container = the container label
            stage('Build'){
                  container('golang'){
                        // This is where we build our code.
                  }
            }
          stage('Build Docker'){
                container(‘docker’){
                      //This is where we build and push our Docker image.
                }
```

Jenkinsfile, FTW

```
yaml """
apiVersion: v1
kind: Pod
metadata:
labels:
  component: ci
spec:
  # Use service account that can deploy to all namespaces
  serviceAccountName: cd-jenkins
  containers:
  - name: golang
    image: golang:1.10
    command:
    - cat
    tty: true
  - name: gcloud
    image: gcr.io/cloud-builders/gcloud
    command:
    - cat
    tty: true
  - name: kubectl
    image: gcr.io/cloud-builders/kubectl
    command:
    - cat
    tty: true
"""
```

Final Thoughts

- Jenkins' capabilities continue to grow as technology changes
- You should take advantage of ALL of Jenkins' capabilities
- YOU, too, can spin up a Jenkins' environment in Kubernetes

Your Turn: What Are Your Questions?

Thank You!

- Say hi, start a conversation on Twitter, LinkedIn, GitHub etc.
- This presentation will be available on the Jenkins World 2018 post-event site
- Mandy Hubbard @DevMandy

DevOps World

Jenkins World

Thank you!