



BUILDING A GPU-FOCUSED CI SOLUTION

Mike Wendt, 19 Mar 2019



@mike-wendt



@mike_wendt

HISTORY OF CI FOR RAPIDS

The “Before CI” era

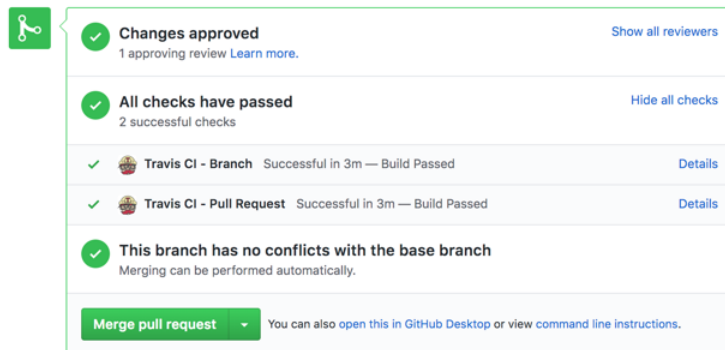
- Not ideal, but attainable in the early days of the project
 - Only 2 repos and a few developers
- Fully manual, PRs required the dev to stop and do the following:
 - Run full suite of tests on their own GPU hardware, on every supported platform
 - build matrix was large: **GPUs, OS, CUDA, DRIVER**, etc.
 - *the motivation to cut corners was high!*
- No automation or centralized/standardized build infrastructure meant the consistency of build & test runs between devs was low.
- Software development was dominated by manual build & test steps
 - ...and this was only for a 2-repo project!
 - ...proper automation was needed

TRAVIS THE SLOW

Moving in the right direction...

Benefits

- Simple, quick setup
- Report status in Github pull requests
- Matrix testing
- MacOS & Windows support



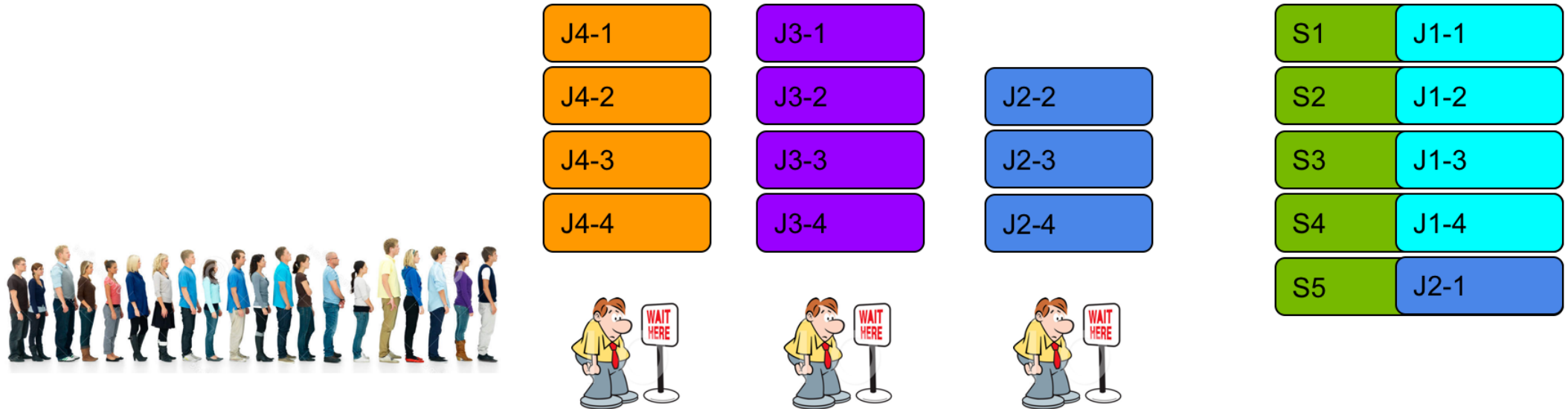
Shortcomings

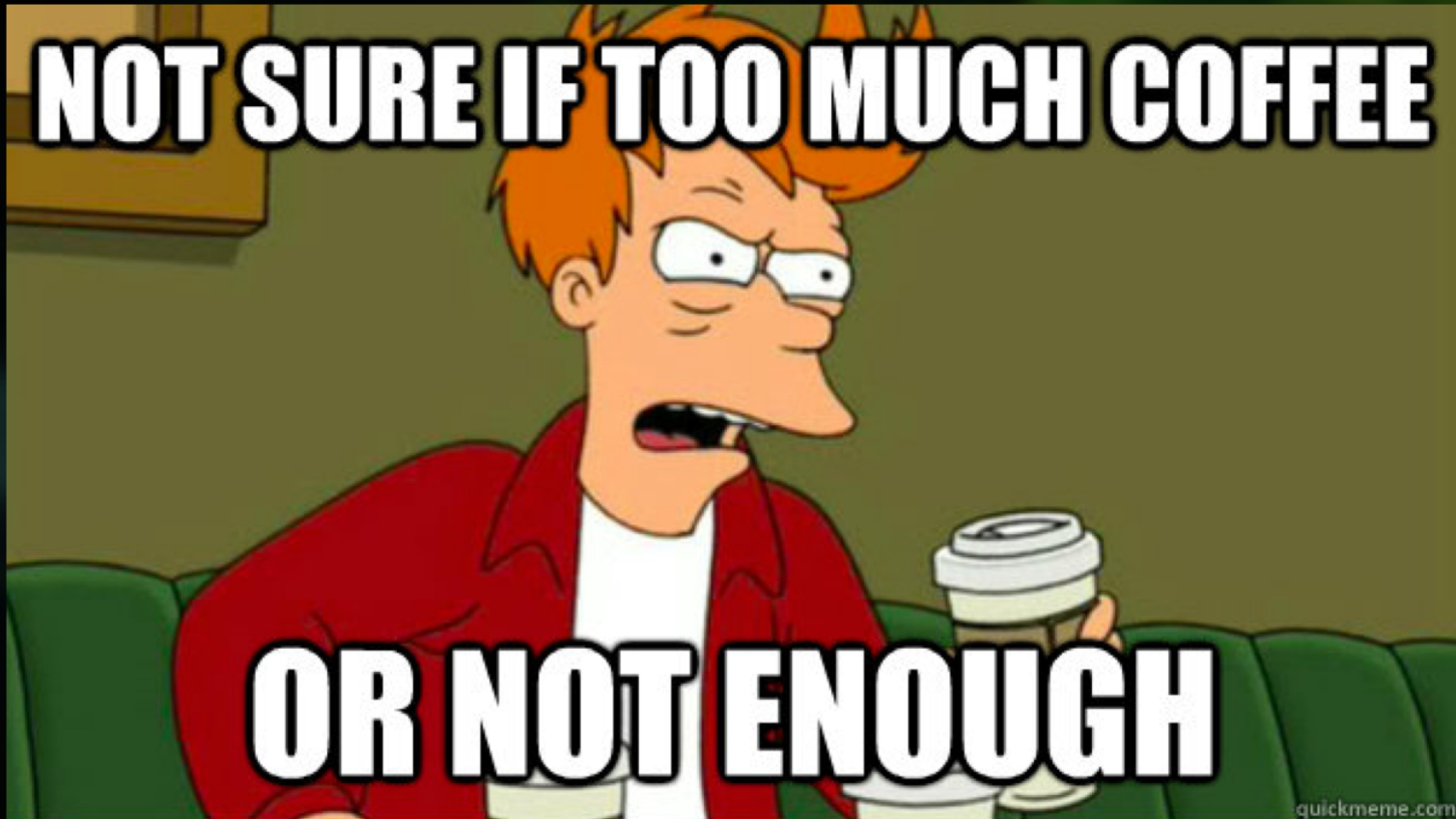
- No GPU support
- Ubuntu 16.04 added just 4 months ago
- No Ubuntu 18.04
- Time limited builds
- Virtual machines w/ 20-120s boot time
- Unable to use Docker containers to speed up builds

TRAVIS THE SLOW

Pipelining jobs that each want 80% of the slots

- Build matrix resulted in 4 jobs per PR
- Travis had a 5 slot limit
- Huge bottleneck quickly identified as project grew





THIS BUILD MAY TAKE A WHILE, COFFEE TIME?

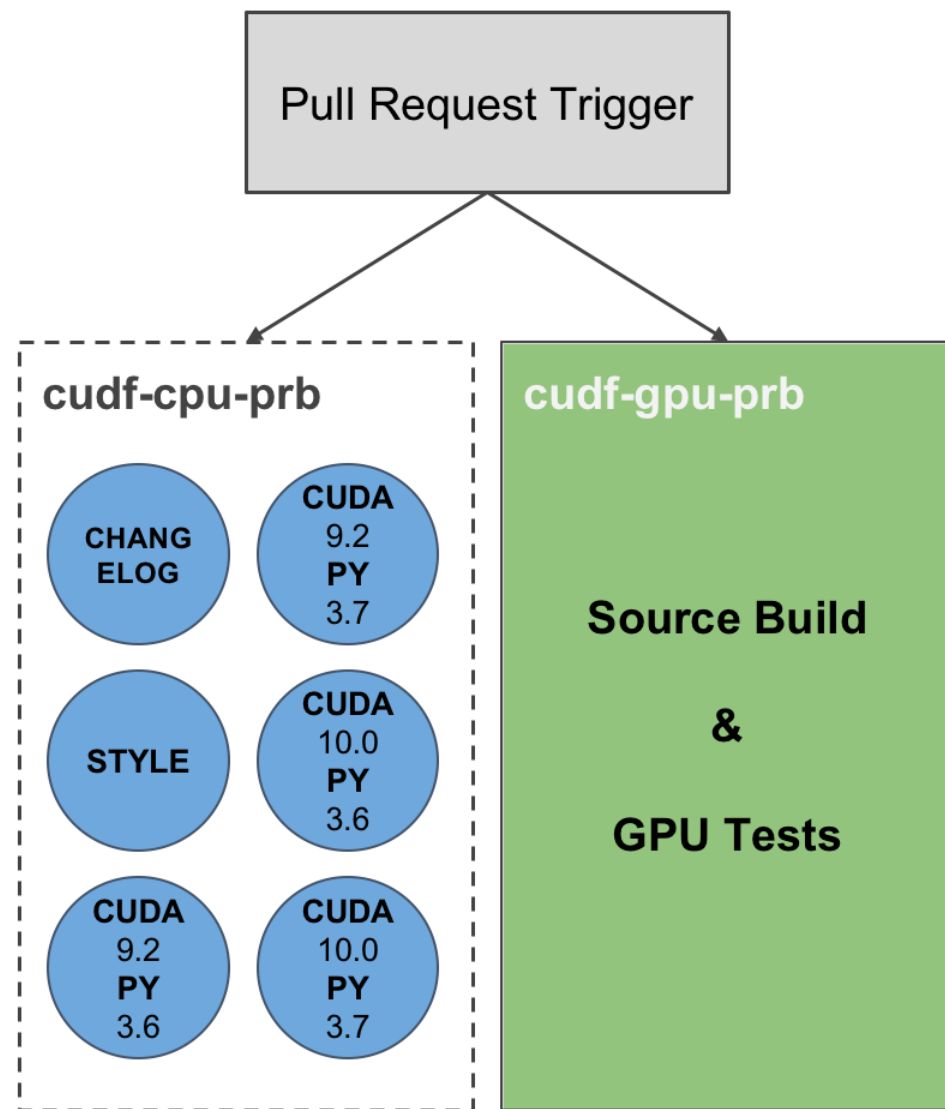
gpuCI - CURRENT

Goals

- Quickly replace Travis
- Faster and scalable builds
- Convert jobs to use Configuration-as-Code

Features

- Builds conda packages on CPU-only machines like Travis CI
- On-demand scaling for multiple concurrent builds
- Existing gpuCI GPU tests unchanged from before



JENKINS GITHUB PULL REQUEST BUILDER PLUGIN

github.com/jenkinsci/ghprb-plugin

Setup a GitHub 'bot' user

Configure PR Builder plugin credentials and system settings

Modify GPU CI project to use PR Builder:

- GitHub Project URL
- Git SCM with repo URL and target branches or only PRs
- Enable PR Builder
- If public, use webhooks

GitLab CI has it's own integrated PR builder

GitHub project

Project url

Source Code Management

None

Git

Repositories

Build Triggers

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Build periodically

GitHub Pull Request Builder

JENKINS GITHUB PULL REQUEST BUILDER PLUGIN

github.com/jenkinsci/ghprb-plugin

Reports build status directly in Github Pull Requests

Require some or all checks to be successful before the PR can be merged

Plugin provides access control to prevent malicious PRs from affecting CI servers

The screenshot shows a GitHub Pull Request status bar. At the top, there is a red 'x' icon and the text 'Review required' in red. Below it, a message states: 'At least 2 approving reviews are required by reviewers with write access. [Learn more.](#)' To the right of this message is a blue link 'Add your review'. Below this is a green checkmark icon and the text 'All checks have passed' in green. Below it, a message states: '5 successful checks' To the right of this message is a blue link 'Hide all checks'. Below this are five build status items, each with a green checkmark icon, a repository path, and a build status. The items are: 1. 'gpuClv2/dask-cuda/gpu' — Build #11 succeeded in 26 sec. 2. 'gpuClv2/dask-cuda/prb/dask-cuda-prb' — Build succeeded. 3. 'gpuClv2/dask-cuda/prb/dask-cuda-style' — Build #12 succeeded in 2.2 sec. 4. 'gpuClv2/dask-cuda/ubuntu-16.04-py3.6' — Build #28 succeeded in 1 min 31 ... 5. 'gpuClv2/dask-cuda/ubuntu-16.04-py3.7' — Build #29 succeeded in 2 min 42... Each item has a blue link 'Details' to its right. At the bottom of the status bar, there is a red 'x' icon and the text 'Merging is blocked' in red. Below it, a message states: 'Merging can be performed automatically with 2 approving reviews.' At the very bottom of the status bar, there is a button 'Merge pull request' with a dropdown arrow, and a message: 'You can also [open this in GitHub Desktop](#) or view [command line instructions](#).'

Style Check	Build - Ubuntu 16.04 Py 3.6	Build - Ubuntu 16.04 Py 3.7	Test with GPU
11s	4min 33s	6min 2s	36s
11s	1min 41s	2min 52s	35s



PR Demo

GOOD NEWS EVERYONE



OUR DOCKER PLUGIN IS AVAILABLE TODAY!

JENKINS PLUGIN FOR NVIDIA + DOCKER

remote-docker-plugin

Open source MIT license

Execute build steps inside of a docker container running on a Jenkins agent

Simplifies the configuration of Docker containers for CI testing

Easy to use and adapt a project for GPU CI

Plugin is usable for non-GPU builds too!

<https://github.com/gpuopenanalytics/remote-docker-plugin>

Dockerfile

Dockerfile Path

Dockerfile

Advanced Settings

Build context

`$WORKSPACE`

Tag

Build Arguments

```
CUDA=9.2
PYTHON=3.7
```

Force pull



Force build



Squash Image



Additional Settings

Add ▾

Manage Volumes

Add Volume

JENKINS PLUGIN FOR NVIDIA + DOCKER

remote-docker-plugin

Exposes configuration for common nvidia-docker settings

Allows for targeting a Dockerfile within the repo to build and use for testing or a Docker image in a remote hub

Easily assign GPUs on multi-GPU agents with *executor* device visibility

Execute build as a user inside of the container

Docker runtime

- runc - Standard Docker runtime (default)
- nvidia - Enable NVIDIA GPU support with nvidia-docker
- custom - Define which runtime to use

NVIDIA Minimum CUDA Version

CUDA Version

NVIDIA Driver Capabilities

- compute - Required for CUDA and OpenCL applications
- compat32 - Required for running 32-bit applications
- graphics - Required for running OpenGL and Vulkan applications
- utility - Required for using 'nvidia-smi' and NVML
- video - Required for using the Video Codec SDK

NVIDIA Device Visibility

- all - All GPUs will be accessible (default)
- none - No GPU will be accessible, but driver capabilities will be enabled
- void - Same behavior as using Docker runtime 'runc'
- executor - A single GPU is accessible indexed by the current executor
- custom - Define which GPU(s) will be visible in container

Add ▾

JENKINS PLUGIN FOR NVIDIA + DOCKER

remote-docker-plugin

Supports side-containers with GPU support

Creates a docker network allowing all containers to communicate

Allows for adding databases, proxies, debugging tools, etc

Analogous to docker-compose

The screenshot displays the Jenkins configuration page for the remote-docker-plugin. It is divided into several sections:

- Image:** A text input field labeled "Docker Image" containing the value "tomcat:7".
- Advanced Settings:**
 - A "Force pull" checkbox is currently unchecked.
 - An "Environment Variables" section is expanded, showing a text input field with the value "db.url=jdbc:postgresql://database-\$BUILD_TAG/db".
 - An "Add" button is located below the environment variables section.
 - A "Manage Volumes" section contains an "Add Volume" button.
- Side containers:** A section for configuring additional containers.
 - A "Side Container" section is expanded, showing:
 - A "Name" input field with the value "database-\$BUILD_TAG".
 - A "Build container" dropdown menu set to "Docker Image".
 - An "Image" section with a "Docker Image" input field containing "postgres:11".
 - An "Advanced..." button at the bottom right of the side container configuration.
 - An "Add Side Container" button is located at the bottom left of the side containers section.



Plugin Code Repo



THE POWER OF CONFIGURATION-AS-CODE

JENKINS PLUGIN FOR NVIDIA + DOCKER

remote-docker-plugin

job-dsl-plugin's Dynamic DSL supported

Leverage configuration-as-code & create reusable jobs or configuration snippets

Store DSL under version control - keep a record and peer review

1st class DSL support coming soon

```
wrappers {
  remoteDockerBuildWrapper {
    debug(false)
    dockerConfiguration {
      dockerImageConfiguration {
        image('gpuci/rapidsai-base:cuda9.2-ubuntu16.04-
gcc5-py3.6')
      }
      forcePull(false)
      configItemList {
        dockerRuntimeConfigItem {
          dockerRuntime('nvidia')
          dockerRuntimeCustom(null)
        }
        nvidiaDriverAbilityConfigItem {
          compat32(false)
          compute(true)
          graphics(false)
          utility(true)
          video(false)
        }
        nvidiaGpuDevicesConfigItem {
          nvidiaDevices('executor')
          nvidiaDevicesCustom(null)
        }
        userConfigItem {
          custom(true)
          username(BUILD_USERNAME)
          uid(BUILD_UID)
          gid(BUILD_GID)
        }
      }
    }
  }
}
```


JENKINS PLUGIN FOR NVIDIA + DOCKER

remote-docker-plugin

Jenkins Pipeline & Jenkinsfiles

DSL for defining build pipelines & build steps

remote-docker-plugin Jenkins Pipeline support coming soon allow for configuring pipelines to run inside docker containers utilizing GPU features

<https://jenkins.io/doc/book/pipeline/>

```
pipeline {
  agent {
    label 'gpu'
    remoteDocker {
      image 'gpuci/rapidsai-
base:cuda9.2-ubuntu16.04-gcc5-py3.6'
      runtime 'nvidia'
      nvidiaGpuDevices 'executor'
    }
  }
  stages {
    stage('Build') {
      steps {
        sh "build.sh"
      }
    }
  }
}
```

The background features a complex network of thin, light green lines connecting various glowing green nodes of different sizes. The nodes are scattered across the dark blue and black background, creating a sense of interconnectedness and data flow. The overall aesthetic is futuristic and technical.

Configuration-as-Code Demo

JENKINS PLUGIN FOR NVIDIA + DOCKER




























remote-docker-plugin

Multiconfiguration Support

Test CUDA application on a variety of GPU types, operating systems, and cuda versions

Instead of 27 machines, only need one machine for each GPU type

Tests are isolated inside docker containers

Configuration Matrix		ubuntu-16.04	ubuntu-18.04	centos-7
p100	cuda-9.2			
	cuda-10.0			
	cuda-10.1			
v100-16	cuda-9.2			
	cuda-10.0			
	cuda-10.1			
v100-32	cuda-9.2			
	cuda-10.0			
	cuda-10.1			



LET'S TWEAK gpuCI TO GO FASTER

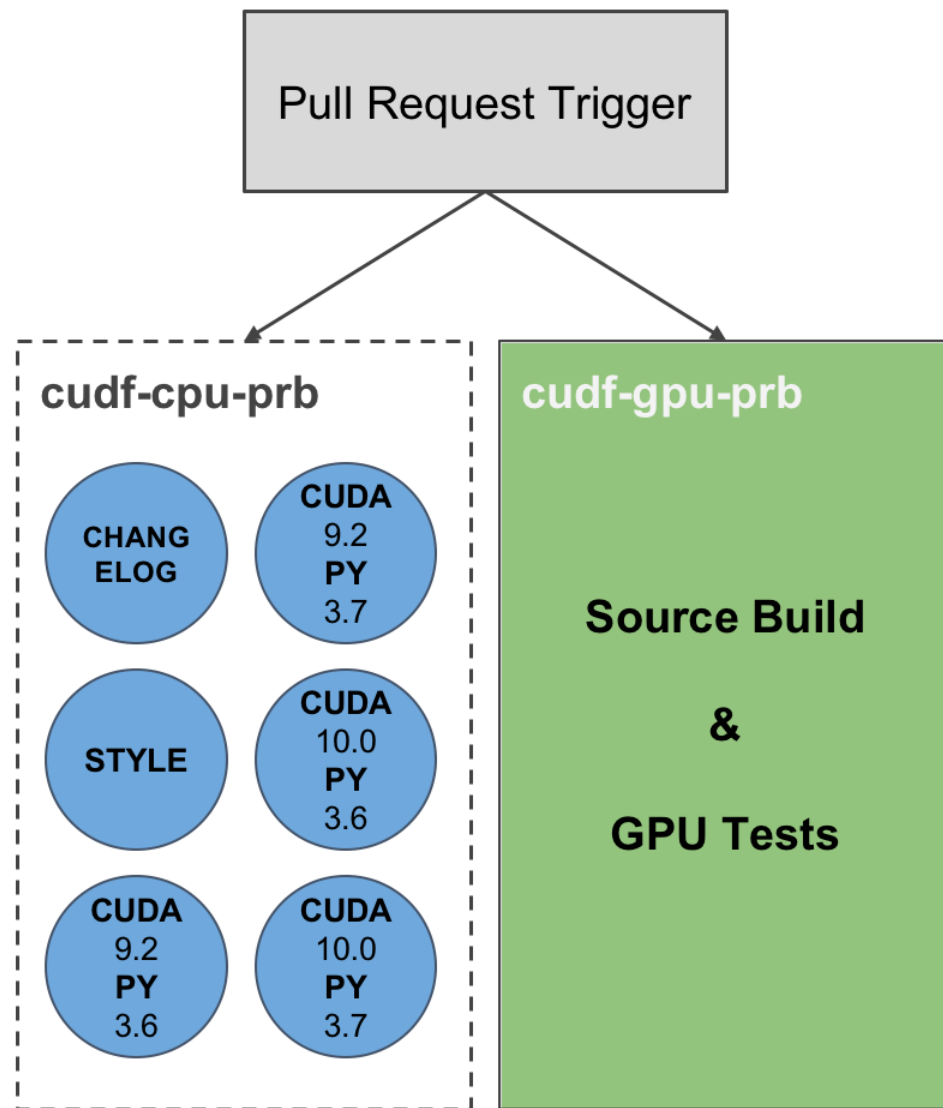
gpuCI - CURRENT

Goals

- Quickly replace Travis
- Faster and scalable builds
- Convert jobs to use Configuration-as-Code

Features

- Builds conda packages on CPU-only machines like Travis CI
- On-demand scaling for multiple concurrent builds
- Existing gpuCI GPU tests unchanged from before



gpuCI - WIP

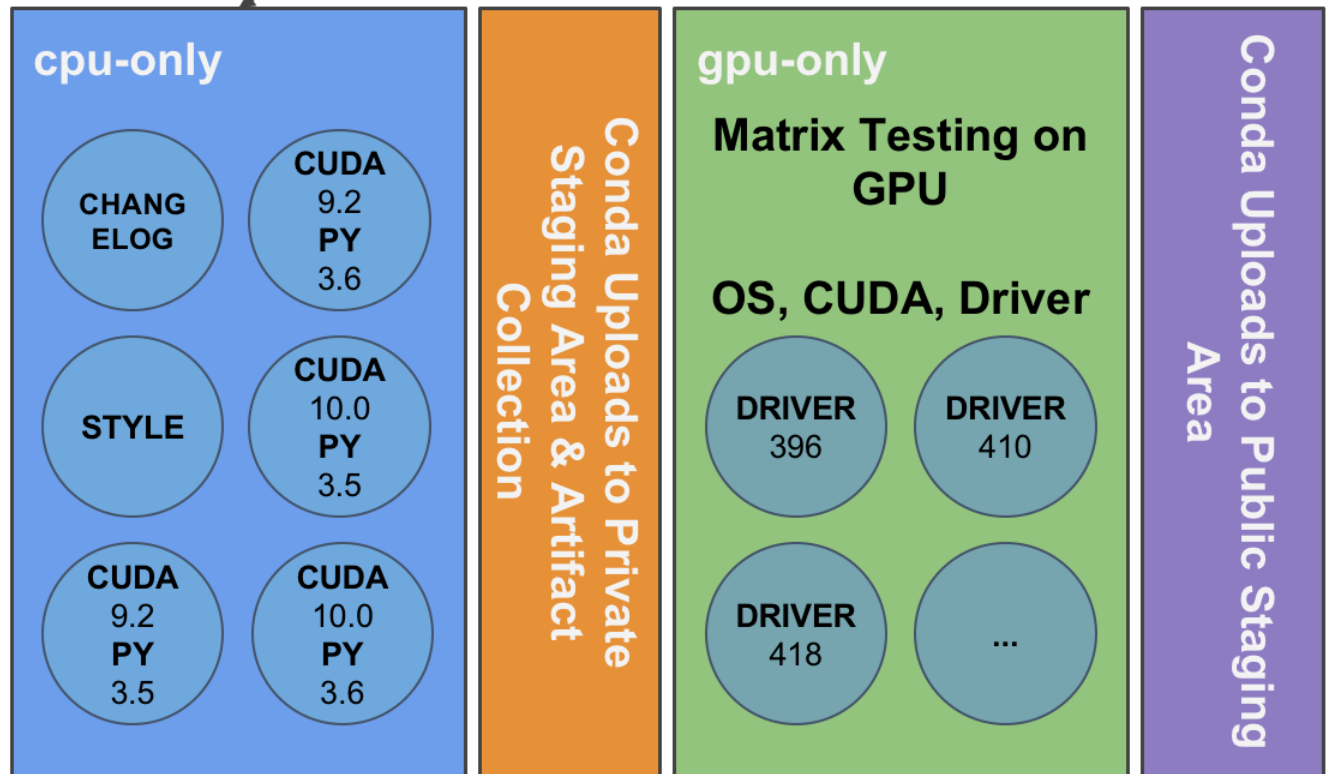
Pull Request Trigger

Goals

- Improve reliability
- Add new features

Features

- Add artifact collection
- Enable build steps to handle errors more easily
- Publish all conda pkgs from PRs
- Enable matrix testing
- Prioritize CPU node usage over GPUs



gpuCI++

Speeding up builds

- Mirror conda packages
- Store docker images for CI in AWS to reduce download times on new nodes

Enhanced testing

- cuda-memcheck
- Other profiling tools
- Code coverage

Custom development

- gpuCI Kubernetes support w/ new plugin
- ASV + GTest performance tracking

Leverage lab hardware

- Add lab GPU hardware to Jenkins for more GPU device capacity

FEDERAL

GALAXY

TOP NEWS

ENLIST

EXIT



WOULD YOU LIKE TO KNOW MORE?

KUBERNETES OPENS UP NEW POSSIBILITIES

KUBERNETES + DOCKER + GPU

Promises to be the easiest to use with minimal hacking

Benefits

- GPU support in v1.8+ of Kubernetes w/ NVIDIA K8 plugin
- Takes care of the “runner” challenge with GitLab/Jenkins
- Resource management and scheduling is handled by Kubernetes

Challenges

- Can only target GPUs on homogeneous nodes (heterogeneous support coming)
- Not all tools support GPU CI out of the box
- Docker containers required for testing, but this can be built as a previous step in a pipeline

NVIDIA K8 DEVICE PLUGIN

github.com/NVIDIA/k8s-device-plugin

Plugin is the "supported" way of using GPUs on K8

Features specifically for working with NVIDIA Docker runtime + K8

Steps to get running on GPU nodes:

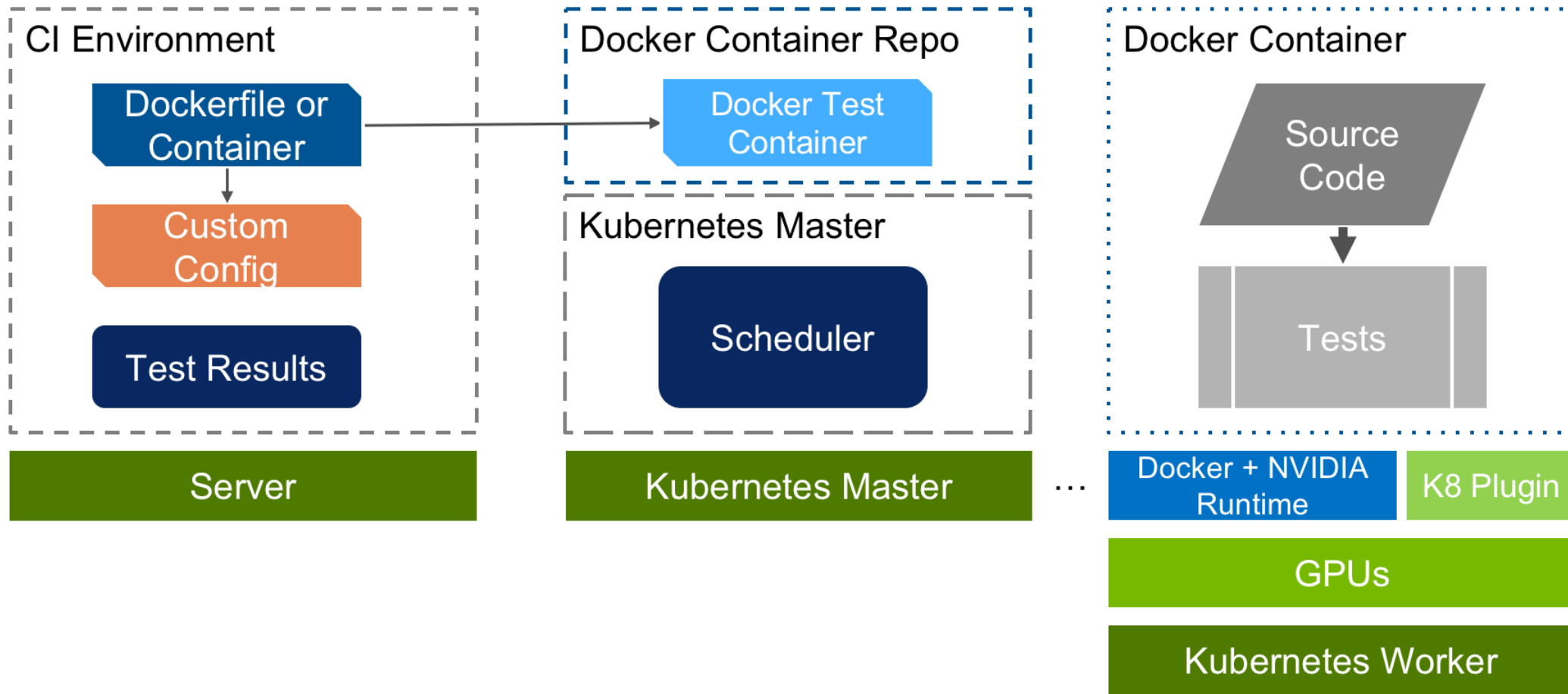
- Install Docker + NVIDIA Docker Runtime
- Change default Docker runtime
- Modify K8 to support *DevicePlugins* feature gate
- Enable GPU support with config file in plugin repo

Running GPU Jobs

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  containers:
    - name: cuda-container
      image: nvidia/cuda:9.0-devel
      resources:
        limits:
          nvidia.com/gpu: 2 # requesting 2 GPUs
    - name: digits-container
      image: nvidia/digits:6.0
      resources:
        limits:
          nvidia.com/gpu: 2 # requesting 2 GPUs
```

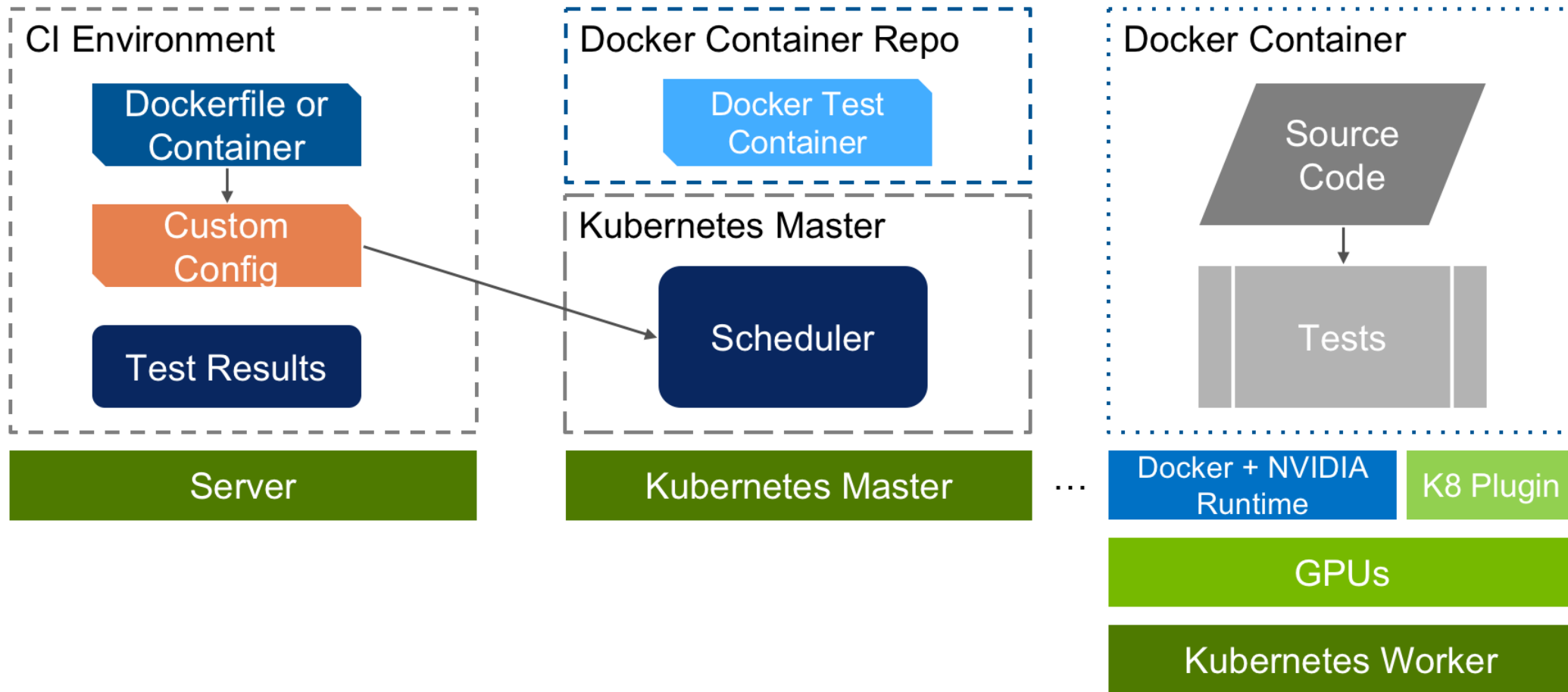
KUBERNETES + DOCKER + GPU

Promises to be the easiest to use with minimal hacking



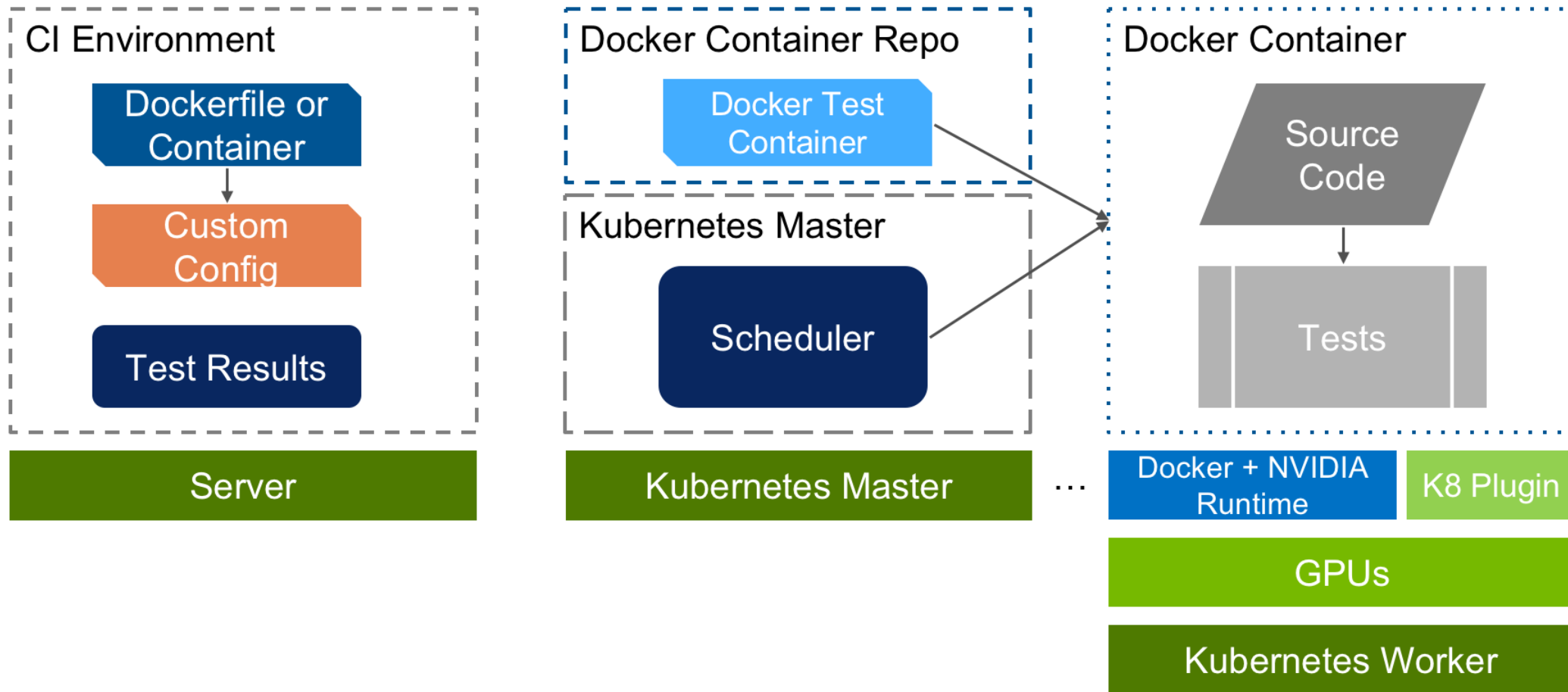
KUBERNETES + DOCKER + GPU

Promises to be the easiest to use with minimal hacking



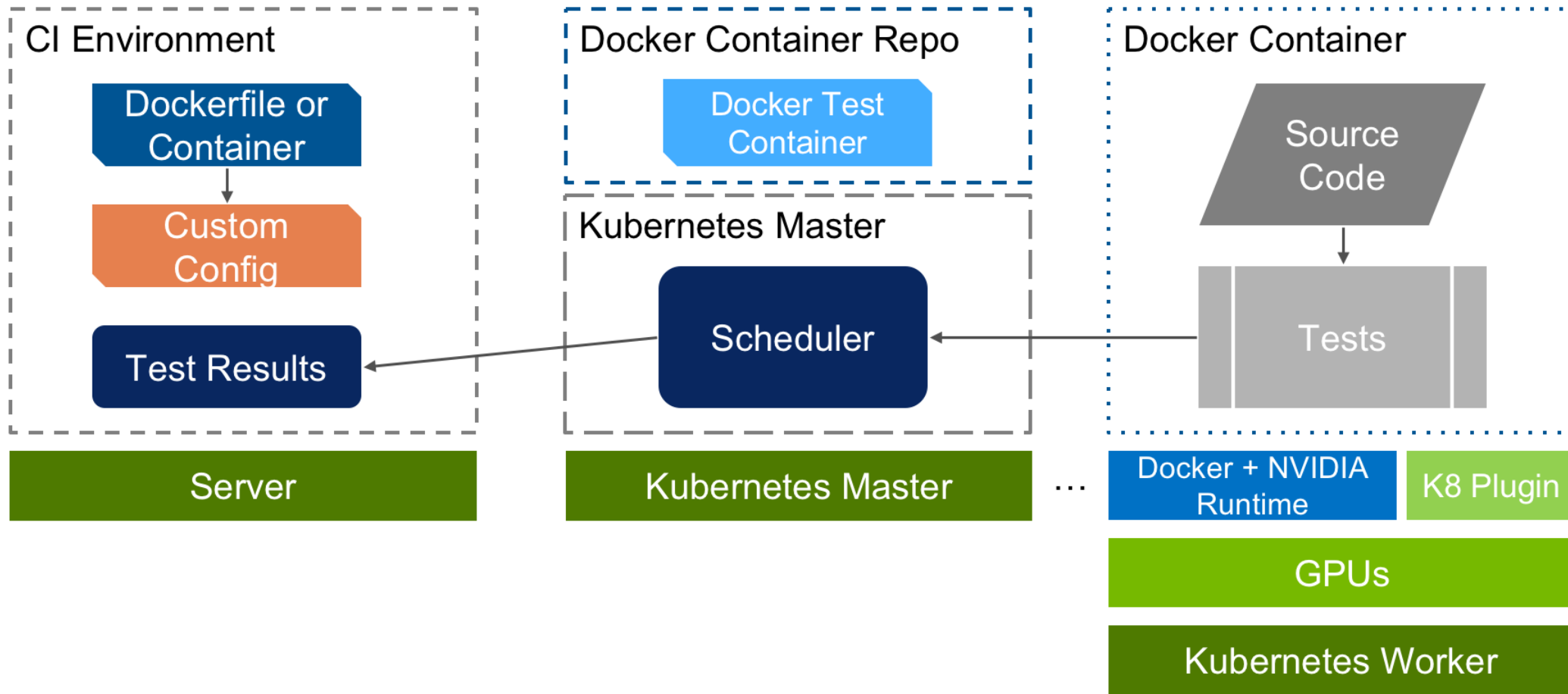
KUBERNETES + DOCKER + GPU

Promises to be the easiest to use with minimal hacking



KUBERNETES + DOCKER + GPU

Promises to be the easiest to use with minimal hacking



JENKINS KUBERNETES PLUGIN TODAY

github.com/jenkinsci/kubernetes-plugin

Has support for pipelining with pod and container configuration

Resource limits of CPU and RAM already exist

Can define custom pod templates with YAML

Still on the "hacky" side but works today

RAPIDS will open source our Kubernetes plugin in the near future

Using yaml to Define Pod Templates

```
def label = "mypod-${UUID.randomUUID()}  
podTemplate(label: label, yaml: ""
```

```
  apiVersion: v1  
  kind: Pod  
  metadata:  
    name: gpu-pod  
  spec:  
    containers:  
      - name: cuda-container  
        image: nvidia/cuda:9.0-devel  
        resources:  
          limits:  
            nvidia.com/gpu: 2 # requesting 2 GPUs  
      - name: digits-container  
        image: nvidia/digits:6.0  
        resources:  
          limits:  
            nvidia.com/gpu: 2 # requesting 2 GPUs
```

```
""
```

```
)
```



ANNOUNCING NIGHTLY RAPIDS BUILDS!

NIGHTLIES

EOD? New conda packages and Docker containers shipped!

How to use with conda

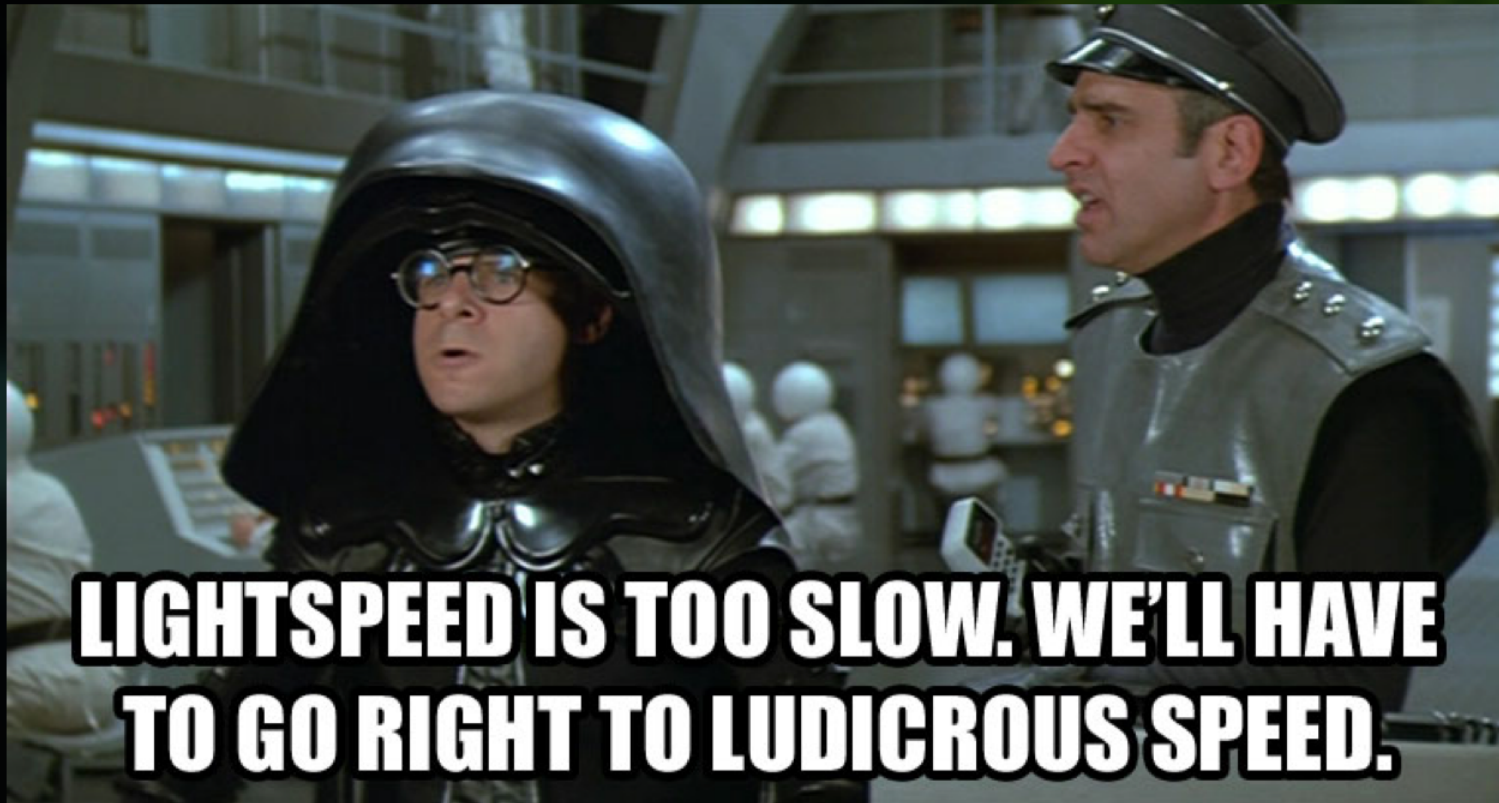
Replace ``-c rapidsai`` with ``-c rapidsai-nightly`` in your ``conda install`` command

```
conda install -c rapidsai-nightly -c pytorch -c numba -c conda-forge \  
-c defaults cudf=0.5 cuml=0.5 python=3.6
```

How to use with docker

Pull the ``rapidsai/rapidsai-nightly`` container

```
docker pull rapidsai/rapidsai-nightly:latest
```



MEASURING SPEED OF LIGHT AND REGRESSIONS

AIRSPEED VELOCITY

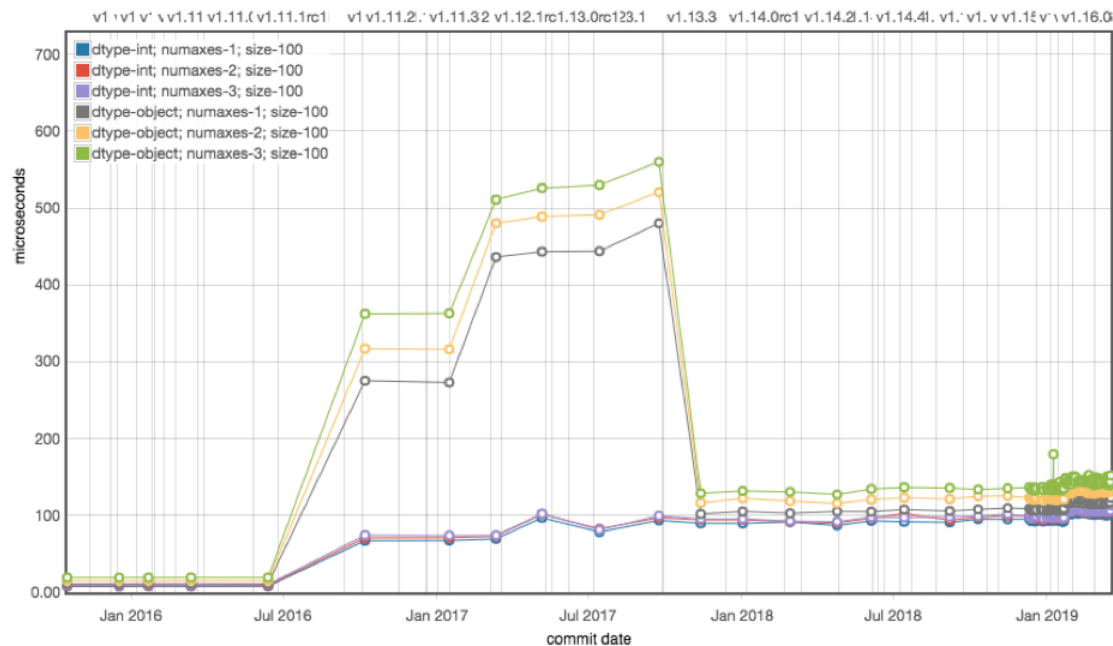
github.com/airspeed-velocity/asv

Benchmark a project through history

Crawl git history to build & benchmark differences between commits

Matrix benchmarking:

- Python version
- CUDA version
- Data types
- Hardware and GPUs



AIRSPEED VELOCITY

github.com/airspeed-velocity/asv

Identify commits that introduced performance regressions

Potential to add benchmark CI checks

Benchmark	Date	Commit	Factor ↑	Before	After
bench_core.CountNonzero.time_count_nonzero_axis(3, 1000000, <class 'bool'>)	2016-10-06 18:55	541fd509..fa214c1f	12.26x	1.346ms	16.510ms
bench_core.CountNonzero.time_count_nonzero_multi_axis(3, 1000000, <class 'bool'>)	2016-10-06 18:55	541fd509..fa214c1f	12.18x	1.351ms	16.450ms
bench_core.CountNonzero.time_count_nonzero_axis(2, 1000000, <class 'bool'>)	2016-10-06 18:55	541fd509..fa214c1f	12.13x	913.650µs	11.087ms

AIRSPEED VELOCITY

github.com/airspeed-velocity/asv

Challenges

- Python focused
- Opposing use-cases from devs
 - Benchmark end-to-end
 - Benchmark granular functions
- Parallelizing multiple benchmarks using multi-GPU

Goals

- Operate on Python & C code
- Differentiate C from Python wrapper results
- Identify regressions in both micro benchmarks AND end-to-end
- Release customizations & contribute back to ASV



TEST MORE, TEST OFTEN, TEST BETTER

BETTER TESTING

Test more, test often, test better

Challenges

- Always more tests!
- Focus on whitebox testing
- “The test will be fixed in the next release”

Goals

- Track test status over time (ASV)
- Public dashboards for feature implementation with mocks
- Track test conversion burndown for devs



WITHOUT OPS, RAPIDS DOESN'T SCALE

SCALING RAPIDS

Without OPS, RAPIDS doesn't scale

Issue overload

Leverage GH issue templates and common labels across repos

No JIRA?!

Use GH project boards to prioritize and focus dev effort for a release

New repo tech-debt

Standard repo-template and CI process for all RAPIDS repos

GH Issue >> Slack

Force devs and team to use GH issues/templates over slack



***RAPIDS Repo Template
Demo***



LESSONS LEARNED FROM THE LAB

LESSONS LEARNED

Teach old CI new tricks

Building docker images is expensive

Prebuild a set of images for python/cuda/gcc combinations

Style Checker


Run first so there's no wasted compute time

Submodules

Educate devs & build CI scripts to automatically identify branches

Hacking around with no GPUs

Force install NVIDIA driver



Next Steps

Continue updates on remote-docker-plugin

Seeking community feedback and involvement

Incorporate ASV into RAPIDS testing

Contribute back any customized code to community

Release Kubernetes plugin

Expect this in the near future, released to the GOAI repo

Longer term - establish a GPU CI 'stack'

With tools and support for testing and debugging



MAKE IT SO

quickmeme.com

GETTING STARTED

Links to useful repos

 github.com/nvidia

- NVIDIA Docker Runtime
 - nvidia-docker
- NVIDIA Kubernetes Device Plugin
 - k8s-device-plugin

 github.com/gpuopenanalytics

- Jenkins Plugin For NVIDIA+Docker
 - remote-docker-plugin

 [@mike_wendt](https://twitter.com/mike_wendt)

THANK YOU

Mike Wendt



@mike_wendt



@mike-wendt



nVIDIA®