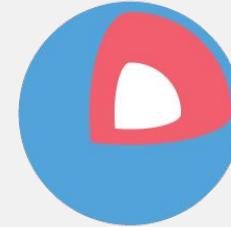




redhat.

AUTOMATING GPU INFRASTRUCTURE FOR KUBERNETES & CONTAINER LINUX

Lucas Servén Marín
Senior Software Engineer
4/5/2018

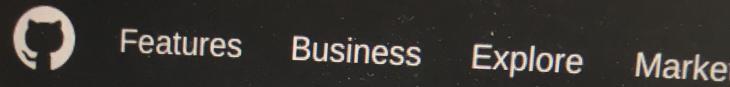


Core OS



redhat®

ABOUT ME



Lucas Servén
Marín
squat

Block or report user

 Red Hat
 Berlin

Overview

Popular repositories

[terraform-provider-vultr](#)

Terraform Vultr Provider

Go ⭐ 64 🍺

[drae](#)

A RESTful API for el Diccionario Epañola

Go ⭐ 14 🍺 4





Search GitHub



daniel servén
dswah

Unfollow

Block or report user

Overview

Pinned repositories

pyGAM

[SEEKING FEEDBACK]
Python

● Python

★ 19

240 contributions



Learn how we count contributions



Features

Business

Explore

Marketplace



Lucas Servén
Marín

squat

Block or report user

Red Hat

Berlin

Overview

Popular repositories

terraform-provider-vultr

Terraform Vultr Provider

● Go

★ 64

Y

drae

A RESTful API for el Diccionario Espanola

● Go

★ 14

Y

DISTRIBUTED COLLABORATION



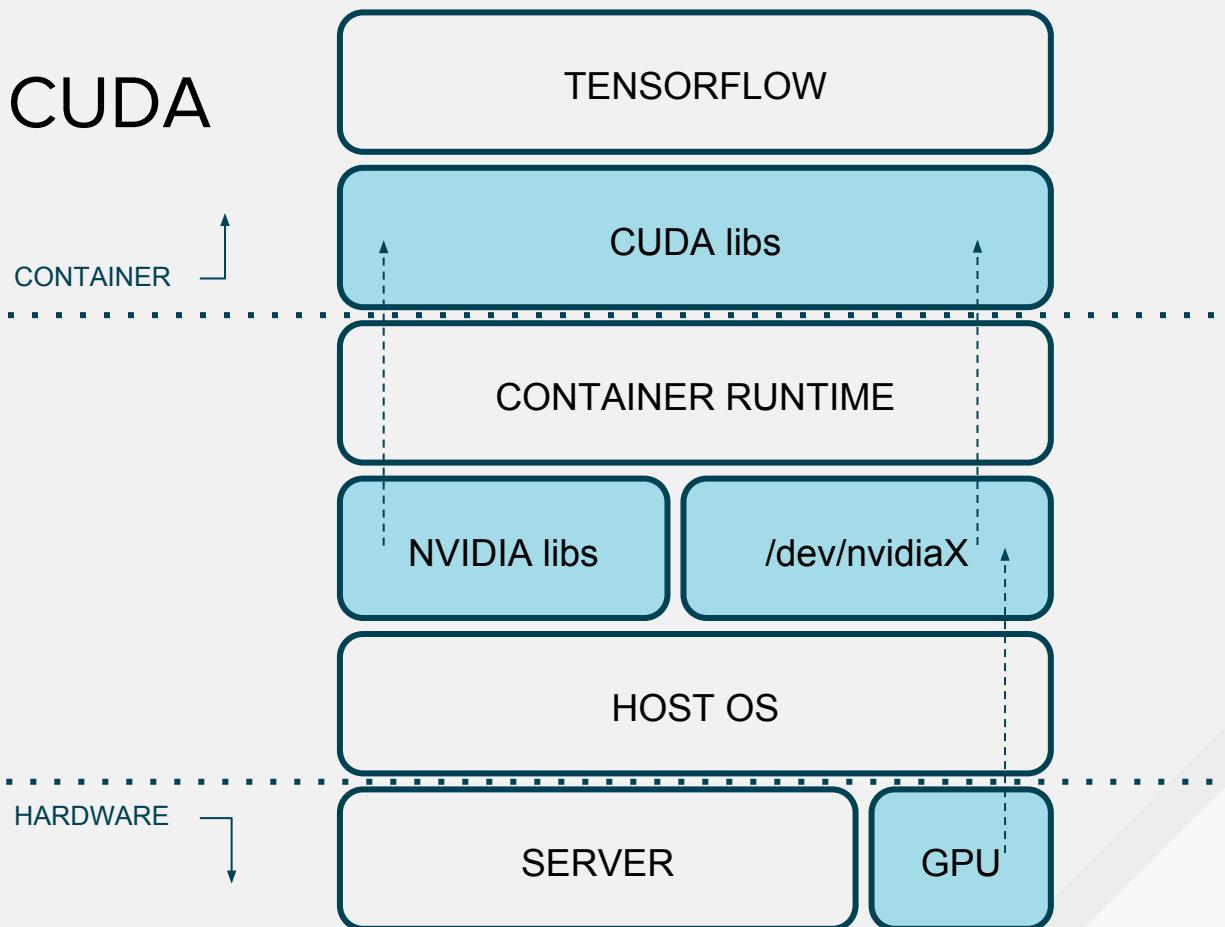
RUNNING ML on K8s



POP QUIZ: NVIDIA ON LINUX?



ANATOMY OF A CUDA WORKLOAD



RUNNING ML on K8s



<https://hub.docker.com/r/nvidia/cuda/>

RUNNING ML on K8s



INSTALLING NVIDIA FOR K8s

The screenshot shows the CUDA Toolkit Documentation page for the Developer Zone. The main content is the Installation Guide for Linux, version v9.1.85. The guide is organized into sections:

- Installation Guide Linux
 - Introduction
 - System Requirements
 - About This Document
- Pre-installation Actions
 - Verify You Have a CUDA-Capable GPU
 - Verify You Have a Supported Version of Linux
 - Verify the System Has gcc Installed
 - Verify the System has the Correct Kernel Headers and Development Packages Installed → (This section is expanded)
 - Choose an Installation Method
 - Download the NVIDIA CUDA Toolkit
 - Handle Conflicting Installation Methods
- Package Manager Installation
- Runfile Installation
- Cluster Management Packages
- CUDA Cross-Platform Environment
- Post-installation Actions
- Advanced Setup
- Frequently Asked Questions
- Additional Considerations

Note: If you perform a system update, make sure to update the kernel headers and kernel development packages.

RHEL/CentOS
The kernel headers and development packages are installed using `$ sudo yum install kernel-devel`.

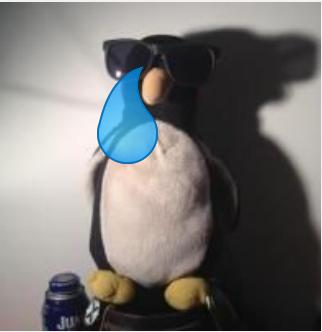
Fedora
The kernel headers and development packages are installed using `$ sudo dnf install kernel-devel`.

OpenSUSE/SLES
Use the output of the `uname -r` command to determine the current kernel version, e.g., `3.16.6-2-default`.
In this example, the version is `3.16.6-2`, so replace `<variant>` and `<version>` with `kernel-3.16.6-2-default`.
The kernel headers and development packages are installed using `$ sudo zypper install kernel-variant<variant>-<version>`.

Ubuntu
The kernel headers and development packages are installed using `$ sudo apt-get install linux-headers-`uname -r``.

2.5. Choose an Installation Method
The CUDA Toolkit can be installed using either package (runfile packages). The distribution-in-distribution's native package management system is used to install the CUDA Toolkit. The user can choose the appropriate package manager (e.g., `sudo apt-get` for Ubuntu, `sudo yum` for RHEL/CentOS) to install the CUDA Toolkit.

INSTALLING NVIDIA FOR K8s



NVIDIA Toolkit v9.1.85

Installation Guide Linux

- ▷ 1. Introduction
 - 1.1. System Requirements
 - 1.2. About This Document
- ▷ 2. Pre-installation Actions
 - 2.1. Verify You Have a CUDA-Capable GPU
 - 2.2. Verify You Have a Supported Version of Linux
 - 2.3. Verify the System Has gcc Installed
 - 2.4. Verify the System has the Correct Kernel Headers and Development Packages Installed →
 - ▷ 2.5. Choose an Installation Method
 - ▷ 2.6. Download the NVIDIA CUDA Toolkit
 - ▷ 2.7. Handle Conflicting Installation Methods
 - ▷ 3. Package Manager Installation
 - ▷ 4. Runfile Installation
 - ▷ 5. Cluster Management Packages
 - ▷ 6. CUDA Cross-Platform Environment
 - ▷ 7. Post-installation Actions
 - 8. Advanced Setup
 - 9. Frequently Asked Questions
 - 10. Additional Considerations

branches, should ensure that their k

Note: If you perform a system update, headers and kernel development pac

RHEL/CentOS

The kernel headers and development p

\$ sudo yum install kernel-devel

Fedora

The kernel headers and development p

\$ sudo dnf install kernel-devel

OpenSUSE/SLES

Use the output of the uname command to

\$ uname -r
3.16.6-2-default

In this example, the version is 3.16.6-2 default. replacing <variant> and <version> with \$ sudo zypper install kernel-<variant>

Ubuntu

The kernel headers and development packag

\$ sudo apt-get install linux-headers

2.5. Choose an Installation Method

The CUDA Toolkit can be installed using either package (runfile packages). The distribution-in distribution's native package management syst to use the distribution-specific packages, when

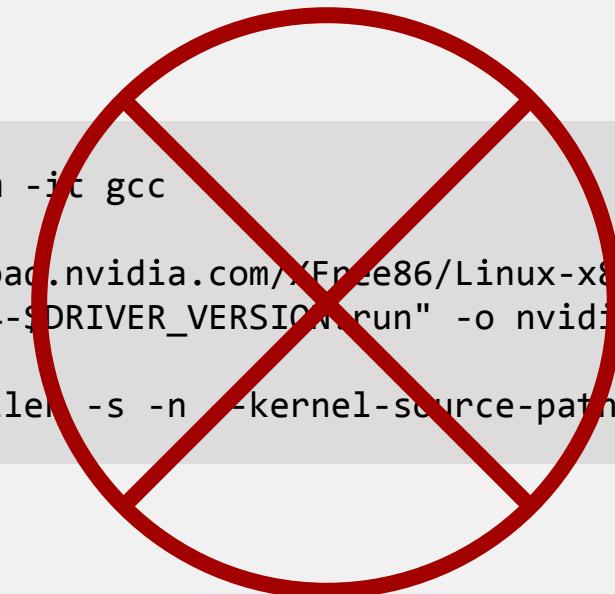
COMPILING NVIDIA FOR CL

```
$ curl -Ls  
"http://us.download.nvidia.com/XFree86/Linux-x86_64/$DRIVER_VERSION/NV  
IDIA-Linux-x86_64-$DRIVER_VERSION.run" -o nvidia.run  
$ # Let's make sure we have gcc  
$ gcc -v  
-bash: gcc: command not found
```



COMPILING NVIDIA FOR CL

```
$ docker run --rm -it gcc  
# curl -Ls  
"http://us.download.nvidia.com/XFree86/Linux-x86_64/$DRIVER_VERSION/NV  
IDIA-Linux-x86_64-$DRIVER_VERSION.run" -o nvidia.run  
...  
# ./nvidia-installer -s -n --kernel-source-path=???
```



COMPILING KERNEL MODULES FOR CL

The screenshot shows a web browser displaying the Container Linux documentation. The header includes the CoreOS logo and links for Products, Open Source, Documentation, Community, and Blog. The main title is "container linux" with the subtitle "A container-focused OS that's designed for painless management in large clusters". Below the title are three navigation links: Overview (selected), Documentation, and Release Notes. On the left, there's a sidebar with a dropdown menu set to "3.0.0 (latest)" and a list of topics: Started, Using Clusters, Container Runtimes, and Reference. The main content area has a heading "Building custom kernel modules" and a sub-section "Create a writable overlay". It explains that the kernel modules directory `/lib/modules` is read-only and describes how to create a writable overlay. A code block shows the command to set up the overlay:

```
modules=/opt/modules # Adjust this writable storage location as needed.  
sudo mkdir -p "$modules" "$modules.wd"  
sudo mount \  
-o "lowerdir=/lib/modules,upperdir=$modules,workdir=$modules.wd" \  
-t overlay /lib/modules
```

Below this, instructions for adding the overlay to `/etc/fstab` are provided:

```
overlay /lib/modules overlay lowerdir=/lib/modules,upperdir=/opt/modules,workdir=/opt/modules
```

PRIOR ART

<https://github.com/Clarifai/coreos-nvidia>

<https://github.com/GoogleCloudPlatform/cos-gpu-installer>

DEVELOPER CONTAINER IN A POD

```
$ gdisk -l coreos_developer_container.bin

Disk coreos_developer_container.bin: 6451200 sectors, 3.1 GiB
Sector size (logical): 512 bytes
Disk identifier (GUID): 00000000-0000-0000-0000-000000000001
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 6451166
Partitions will be aligned on 2048-sector boundaries
Total free space is 159677 sectors (78.0 MiB)
```

Number	Start (sector)	End (sector)	Size	Code	Name
9	4096	6295551	3.0 GiB	8304	ROOT

COMPILING NVIDIA FOR CL

```
$ kubectl apply -f gpu-installer.yaml
```

RUNNING A CUDA CONTAINER

```
$ docker run --device=/dev/nvidiactl --device=/dev/nvidia-uvm  
--device=/dev/nvidia0 -v=/opt/nvidia/387.34:/usr/local/nvidia:ro  
--entrypoint=nvidia-smi nvidia/cuda
```

MOUNTING NVIDIA FILES

```
$ kubectl apply -f device-plugin.yaml
```

MOUNTING NVIDIA FILES

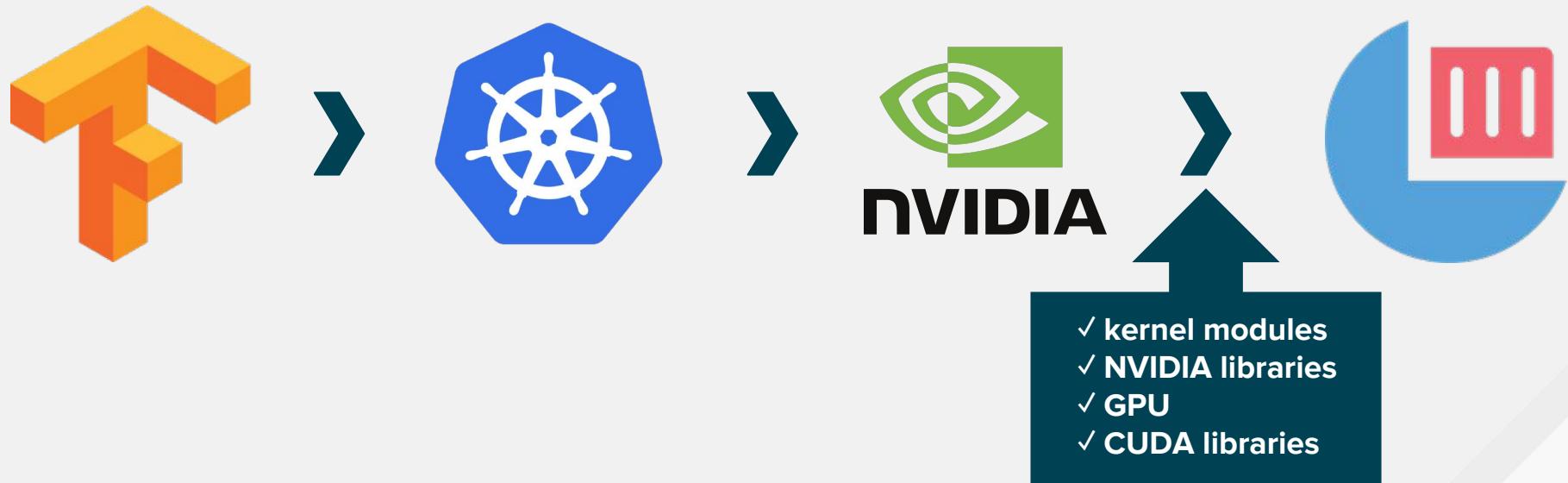
```
# https://github.com/GoogleCloudPlatform/container-engine-accelerators

for _, d := range s.ngm.defaultDevices {
    resp.Devices = append(resp.Devices, &pluginapi.DeviceSpec{
        HostPath:      d,
        ContainerPath: d,
        Permissions:   "mrw",
    })
}
resp.Mounts = append(resp.Mounts, &pluginapi.Mount{
    ContainerPath: s.ngm.containerPathPrefix,
    HostPath:       s.ngm.hostPathPrefix,
    ReadOnly:       true,
})
```

ANATOMY OF A CUDA WORKLOAD ON K8s



RUNNING ML on K8s



DEMO TIME

DEMO COMPONENTS

RESOURCE	URL
overview	https://github.com/squat/kubecone2018
K8s installer	https://github.com/poseidon/typhoon
GPU installer	https://github.com/squat/modulus
device plugin	https://github.com/kubernetes/kubernetes/blob/master/cluster/addons/device-plugins/nvidia-gpu/daemonset.yaml
sample workload	https://github.com/pjreddie/darknet

GOING FORWARD



THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHatNews

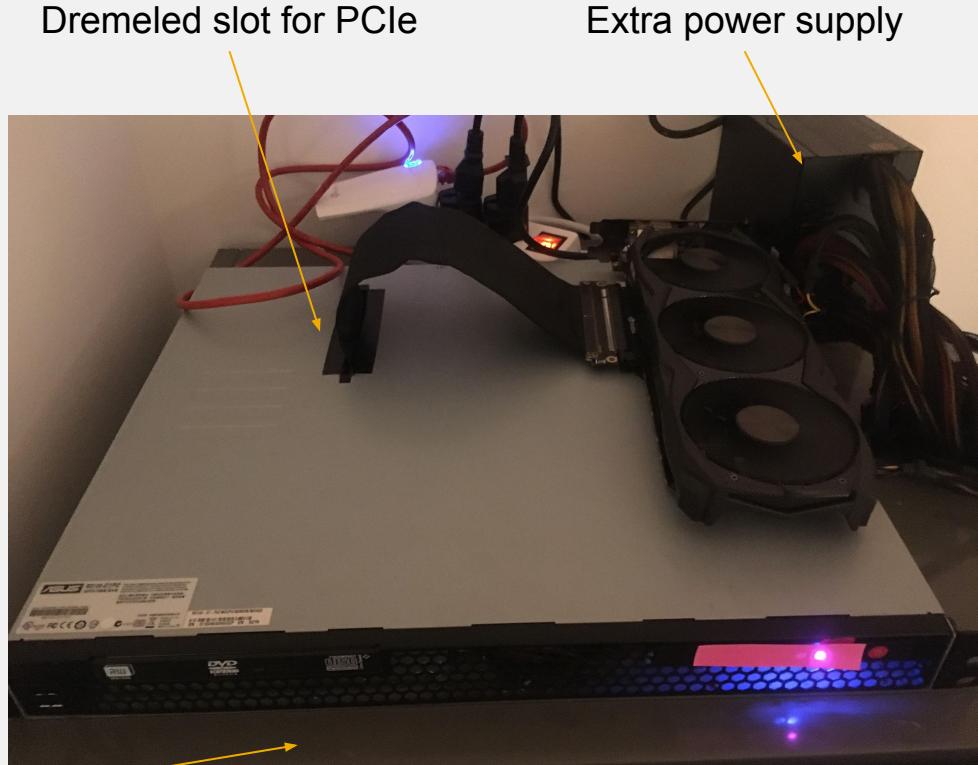


youtube.com/user/RedHatVideos

FAQ

- Will this change now that Red Hat acquired CoreOS?
- How is this different from the COS GPU installer?
- How can I do this for _____ distribution?
- What about GPU sharing?
- How can I avoid compiling on every node?

Refrigerator



My first bare metal K8s GPU node

PROJECTS MENTIONED

- <https://github.com/Clarifai/coreos-nvidia>
- <https://github.com/GoogleCloudPlatform/cos-gpu-installer>
- <https://github.com/GoogleCloudPlatform/container-engine-accelerators>
- <https://github.com/poseidon/typhoon>
- <https://github.com/squat/modulus>
- <https://github.com/pjreddie/darknet>
- <https://github.com/squat/darkapi>

ADDITIONAL RESOURCES

- <https://github.com/shelmangroup/coreos-gpu-installer>
- <https://github.com/coreos/docs/blob/master/os/kernel-modules.md>
- <https://github.com/kubernetes/kubernetes/blob/master/cluster/addons/device-plugins/nvidia-gpu/daemonset.yaml>
- https://schd.ws/hosted_files/cnkc16/84/StateOfTheGPUUnion.pdf