



AUTOMATING GPU INFRASTRUCTURE FOR KUBERNETES & CONTAINER LINUX

Lucas Servén Marín
Senior Software Engineer
4/5/2018

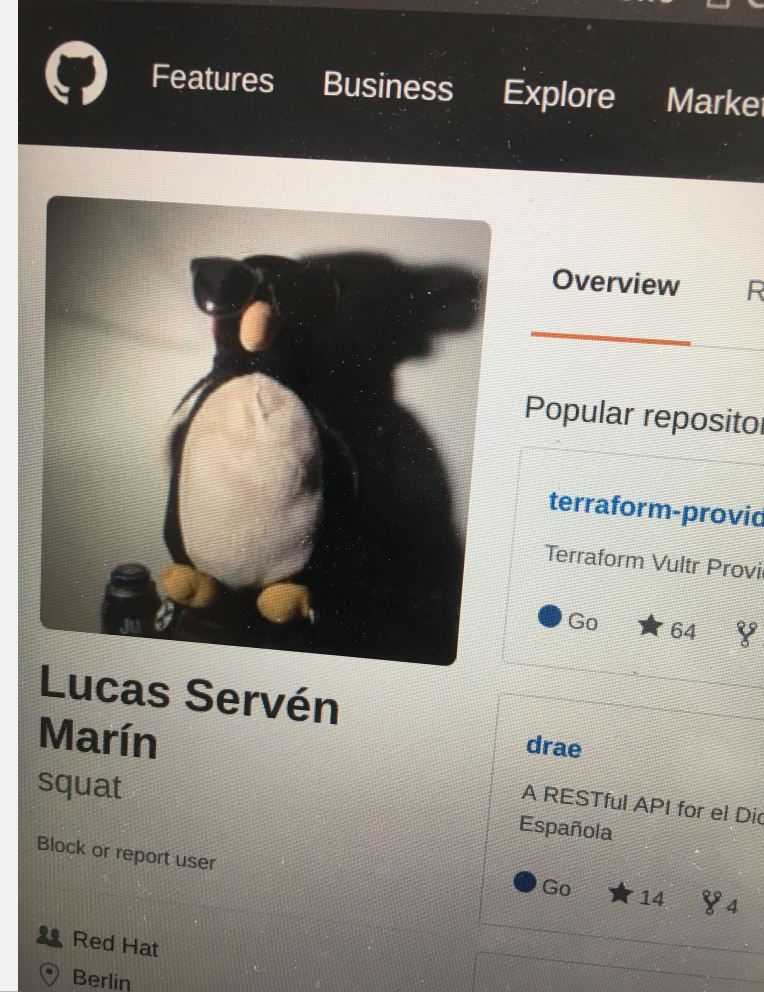


Core OS



redhat®

ABOUT ME



The image shows a screenshot of a GitHub profile page for Lucas Servén Marín. At the top, there is a navigation bar with the GitHub logo and links for Features, Business, Explore, and Market. The profile picture is a black and white stuffed penguin wearing sunglasses. Below the profile picture, the name "Lucas Servén Marín" is displayed in a large font, with "squat" as the bio. There are options to "Block or report user" and a location "Berlin". The profile is associated with "Red Hat". To the right, there is a section for "Popular repositories" with two entries: "terraform-provid" (Terraform Vultr Provi) with 64 stars, and "drae" (A RESTful API for el Dic Española) with 14 stars and 4 forks.

Features Business Explore Market

Overview

Popular repositories

[terraform-provid](#)
Terraform Vultr Provi
Go ★ 64

[drae](#)
A RESTful API for el Dic Española
Go ★ 14 4

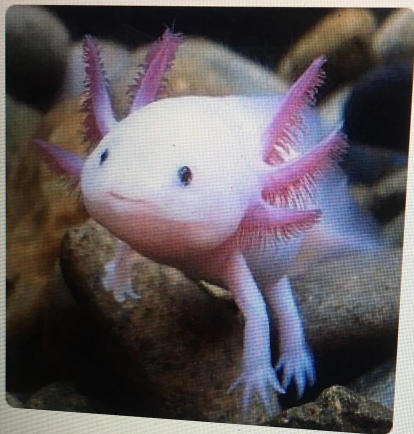
Lucas Servén
Marín
squat

Block or report user

Red Hat
Berlin



Search GitHub



daniel servén

dswah

Unfollow

Block or report user

Overview

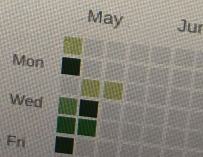
Pinned repository

pyGAM

[SEEKING FEEDBACK]
Python

Python ★ 19

240 contributions



Learn how we count contributions

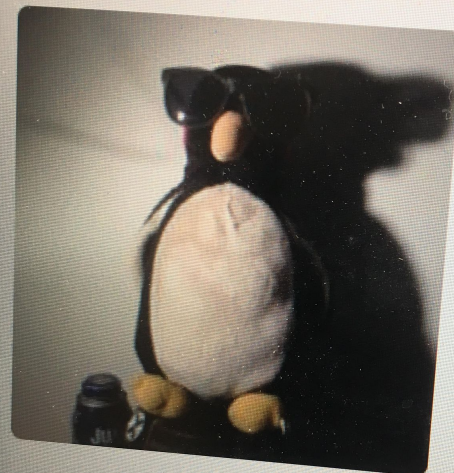


Features

Business

Explore

Marketplace



**Lucas Servén
Marín**

squat

Block or report user

Red Hat

Berlin

Overview

Popular repository

terraform-provisioner

Terraform Vultur Provisioner

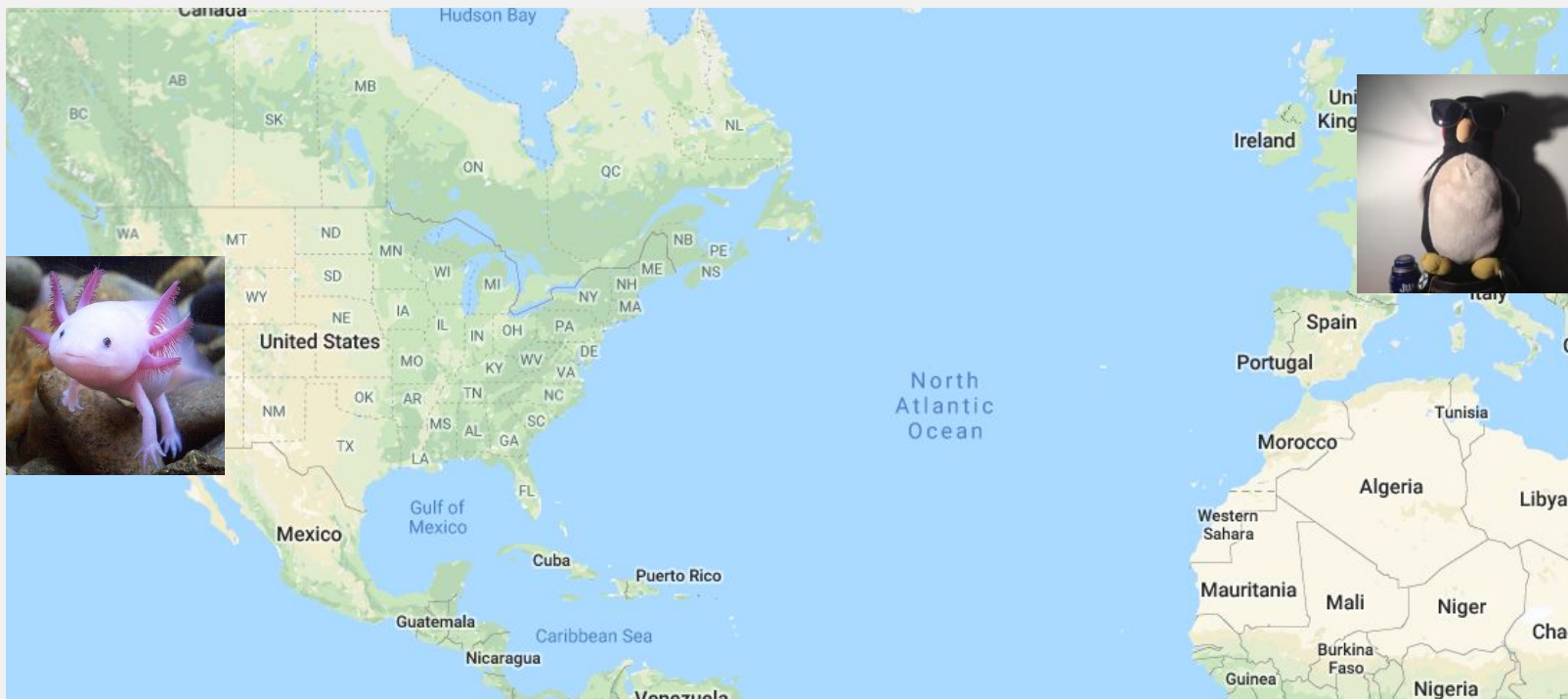
Go ★ 64

drae

A RESTful API for el Diccionario de la Real Academia Española

Go ★ 14

DISTRIBUTED COLLABORATION



RUNNING ML on K8s



POP QUIZ: NVIDIA ON LINUX?



ANATOMY OF A CUDA WORKLOAD

CONTAINER



TENSORFLOW

CUDA libs

CONTAINER RUNTIME

NVIDIA libs

/dev/nvidiaX

HOST OS

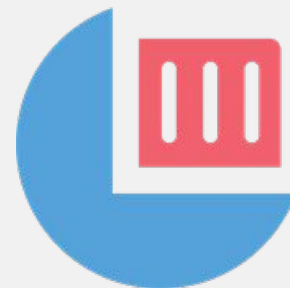
HARDWARE



SERVER

GPU

RUNNING ML on K8s



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

? kernel modules
? NVIDIA libraries
✓ GPU
✓ CUDA libraries

RUNNING ML on K8s



INSTALLING NVIDIA FOR K8s

DEVELOPER ZONE CUDA TOOLKIT DOCUMENTATION

CUDA 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, you must update the kernel headers and kernel development packages.

RHEL/CentOS

The kernel headers and development packages are available in the `kernel-devel` package.

```
$ sudo yum install kernel-devel
```

Fedora

The kernel headers and development packages are available in the `kernel-devel` package.

```
$ sudo dnf install kernel-devel
```

OpenSUSE/SLES

Use the output of the `uname` command to determine the kernel version.

```
$ uname -r
3.16.6-2-default
```

In this example, the version is `3.16.6-2`, so you would replace `<variant>` and `<version>` with `3.16.6-2` in the following command:

```
$ sudo zypper install kernel-<variant>-<version>
```

Ubuntu

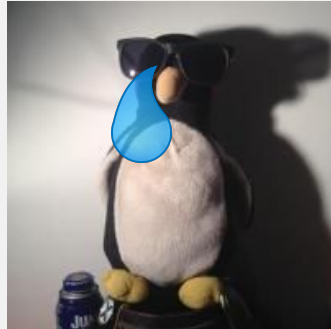
The kernel headers and development packages are available in the `linux-headers` package.

```
$ sudo apt-get install linux-headers-$(uname -r)
```

2.5. Choose an Installation Method

The CUDA Toolkit can be installed using either the `runfile` package (runfile packages). The distribution-independent package management system (e.g., `rpm` or `dpkg`) to use the distribution-specific packages, which are available in the `runfile` package.

INSTALLING NVIDIA FOR K8s



DEVELOPER ZONE **CUDA TOOLKIT DOCUMENTATION**

CUDA Toolkit v9.1.85
Installation Guide Linux

- 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, kernel headers and kernel development packages must be installed.

RHEL/CentOS

The kernel headers and development packages are available in the `kernel-devel` package.

```
$ sudo yum install kernel-devel
```

Fedora

The kernel headers and development packages are available in the `kernel-devel` package.

```
$ sudo dnf install kernel-devel
```

OpenSUSE/SLES

Use the output of the `uname` command to determine the kernel version.

```
$ uname -r  
3.16.6-2-default
```

In this example, the version is `3.16.6-2`, so you would install the `kernel-devel-3.16.6-2` package, replacing `<variant>` and `<version>` with the appropriate values.

```
$ sudo zypper install kernel-<variant>
```

Ubuntu

The kernel headers and development packages are available in the `linux-headers` package.


```
$ sudo apt-get install linux-headers-$(uname -r)
```

2.5. Choose an Installation Method

The CUDA Toolkit can be installed using either the `runfile` package (runfile packages). The distribution's native package management system can be used to install the `runfile` packages, or you can use the distribution-specific packages, which are available in the `Developer Zone`.


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/Free86/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 the Container Linux website. At the top, there is a navigation bar with the CoreOS logo and links for Products, Open Source, Documentation, Community, and Blog. The main header features the Container Linux logo and the text "A container-focused OS that's designed for painless management in large clusters". Below the header, there are three tabs: Overview, Documentation, and Release Notes. The main content area is titled "Building custom kernel modules" and includes a sub-section "Create a writable overlay". The text explains that the kernel modules directory /lib/modules is read-only and provides instructions on how to create a writable overlay. A code block shows the following commands:

```
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 overlay /lib/modules
```

Below the code, it states: "To mount the overlay automatically when the system boots, add the following line to /etc/fstab (cr...". The code block for /etc/fstab is partially visible at the bottom:

```
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,
})
```

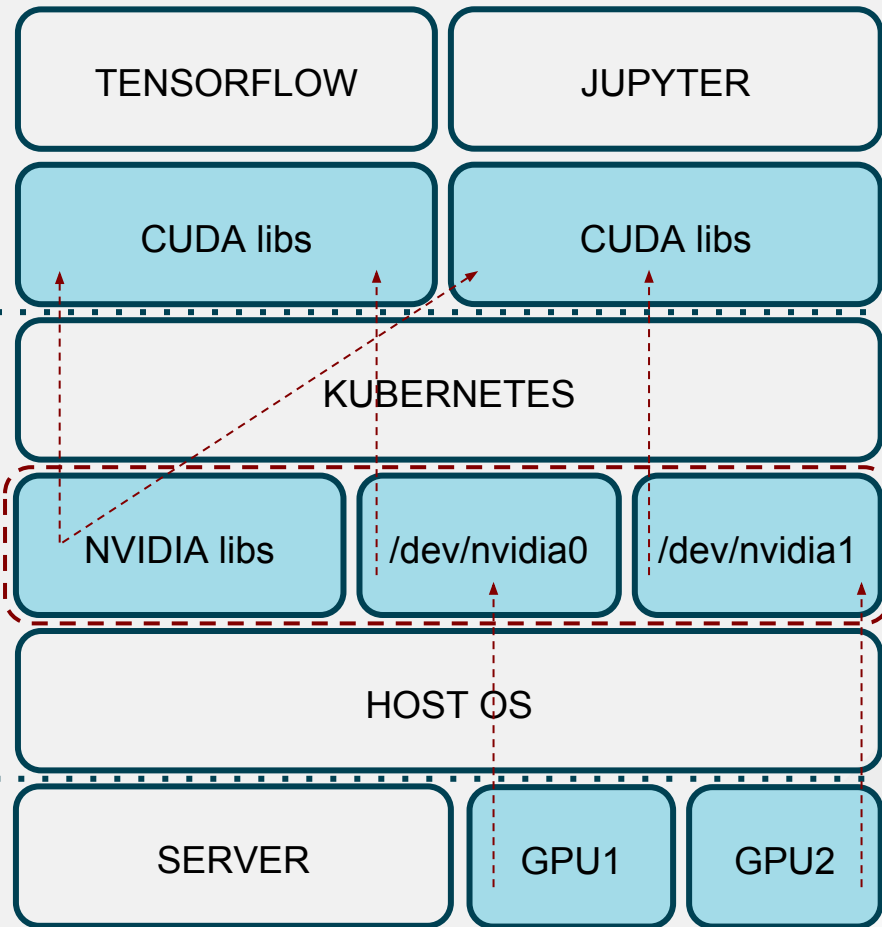
ANATOMY OF A CUDA WORKLOAD ON K8s

CONTAINER

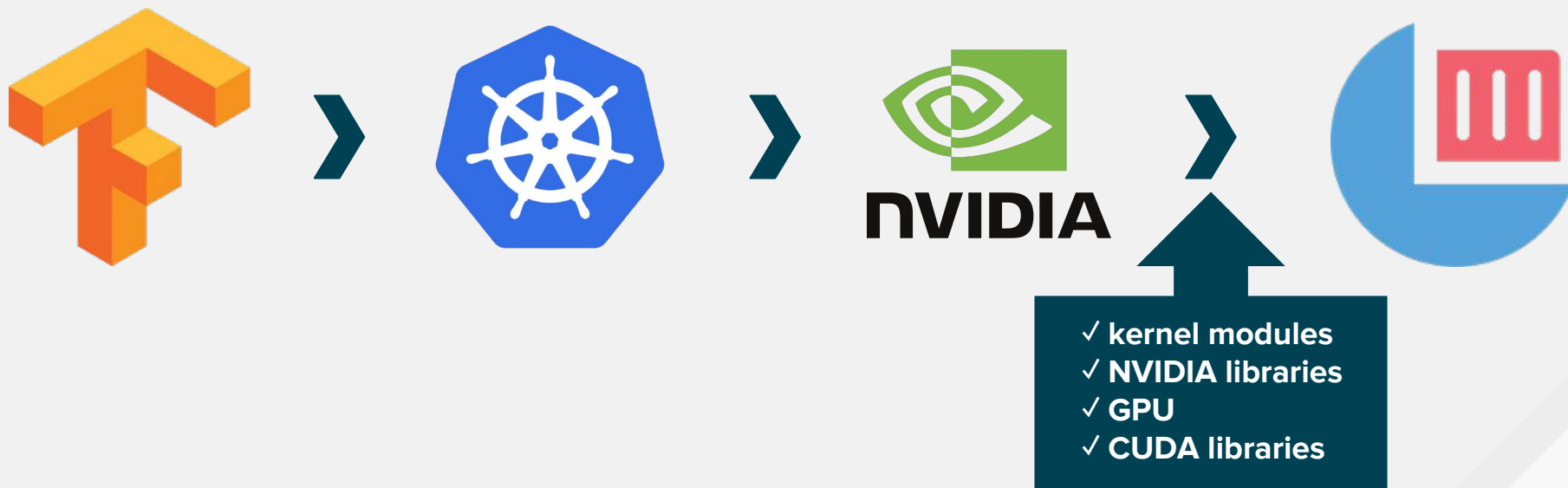
DEVICE PLUGIN

GPU INSTALLER

HARDWARE



RUNNING ML on K8s



DEMO TIME

DEMO COMPONENTS

RESOURCE	URL
overview	https://github.com/squat/kubeconeu2018
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



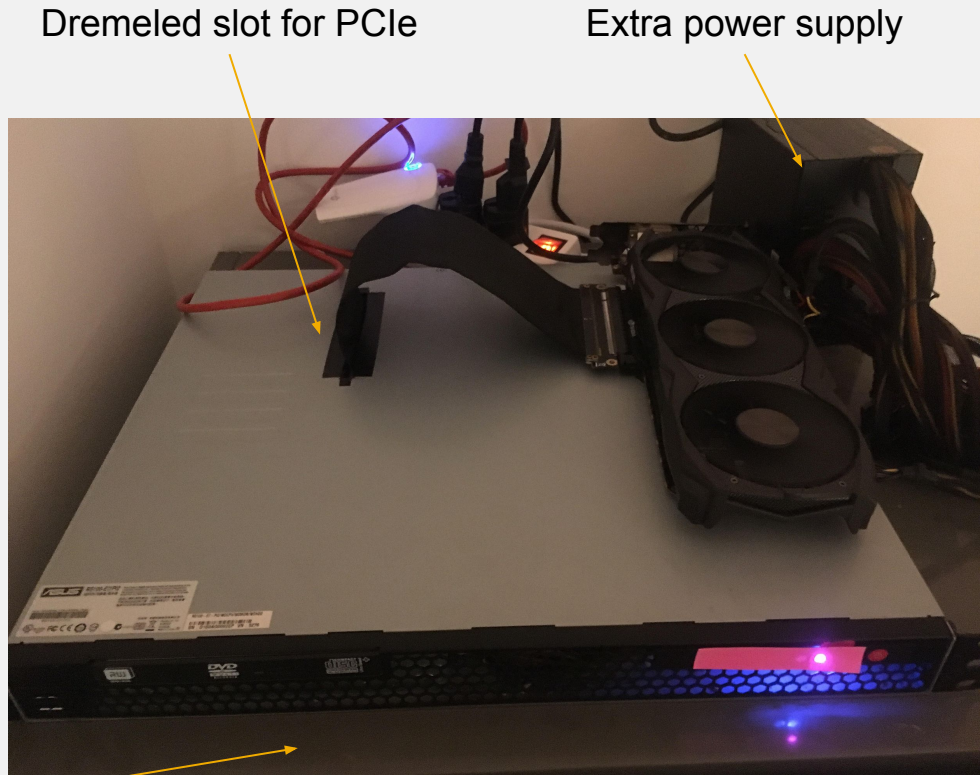
twitter.com/RedHatNews



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://sched.ws/hosted_files/cnkc16/84/StateOfTheGPUUnion.pdf