About Me

Background:
- 15+ years all around technologist
  - Network Engineer
  - Storage Engineer
  - Virtualization Engineer
  - Wintel Engineer
  - Unix Engineer
  - Etc…
- 2 years at GE
  - Lead Cloud Architect
- 4 years at Platform9
  - Lead Systems Engineer
  - Systems Engineering Manager
  - Cloud Architect
  - Director of Technology

About me:
- Married
- 1 Child
  - 20mo old Son
- Sports
  - Wrestling
  - Football
  - Mixed Martial Arts
- Technology
  - Cloud
  - Raspberry Pi Maker
  - Media Hoarder
  - 3D Printer Enthusiast
<Marketing enforced Slides>
An Enterprise Kubernetes Deployment is Challenging

Challenges: Enterprise Deployment

- typical time to implement takes weeks
- troubleshooting and monitoring a cluster is largely manual process
- upgrading the platform requires downtime and manual process

HA control plane

- master node 1
  - etcd
  - API server
  - scheduler
  - service discovery
  - load balancing

- master node 2
  - etcd
  - API server
  - scheduler
  - service discovery
  - load balancing

worker node

- kubelet
  - apps
  - runtime (docker)

worker node

- kubelet
  - apps
  - runtime (docker)
- Physical or Virtual Servers with Linux OS
- Platform9 SaaS Management Plane (aka Deployment Unit - DU)
- PF9 Agent
- Single Pane of Glass
- Deploy & Manage k8s anywhere
## Industry leaders rely on us

<table>
<thead>
<tr>
<th>70+ Enterprise Customers</th>
<th>60+ DevOps Integrations</th>
<th>300+ Clouds Managed</th>
<th>500,000+ Cores under management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aruba</td>
<td>Cadence</td>
<td>Veritas</td>
<td>CSPI</td>
</tr>
<tr>
<td>Autodesk</td>
<td>S&amp;P Global</td>
<td>Comcast</td>
<td>Snapfish</td>
</tr>
<tr>
<td>EBSCO</td>
<td>Rodan + Fields</td>
<td>Providence Health Care</td>
<td>Dominion Enterprises</td>
</tr>
</tbody>
</table>

![Global map showing clients](image-url)
Why Bare-Metal?

- **Operationally:**
  - One less layer to manage and care for vs Virtualization
  - Less overhead on the Physical machine taken up by the Virtualization layer (Libvirt, Qemu, Etc…)
  - Greater Pod Density per Node

- **Application Specific:**
  - Data intensive workloads like big data run best on bare-metal
  - Applications that need line rate networking like Kafka run best on bare-metal
  - Workloads that need close access to hardware like GPUs for Machine Learning run best on bare-metal
Bare-Metal Complexities (Compared to Public Cloud)

Getting Started

- Procure Datacenter Space
- Procure Network Circuits
- Procure Switches
- Setup and maintain and Edge Network
  - BGP
  - Firewalls
  - Etc...
- Configure Switches
  - Edge Switches
  - Core Switches
  - End of Row Switches
  - Top of Rack Switches
- Procure Servers
- Rack and stack the servers
- Cable these servers to the switches
- Install operating system
- Configure your operating system for network and ssh access
Ongoing Difficulties

- **General DC Difficulties:**
  - Debugging and Fixing Hardware Failures
  - Checking and updating Bios and Firmware
  - Refreshing Hardware
    - Redo everything in the previous slide every 3-5 years
    - Plus wiping and getting rid of the old hardware
    - Lease return?
    - Selling on second hand market?

- **Kubernetes Specific Difficulties:**
  - Load Balancers
  - Networking (Security)
  - Storage
Problem Solving
DON'T DO IT!

Leave this to someone else that does this exceptionally well.

Packet.com does this at a tremendous scale with:

- API Driver Bare-Metal Servers
- A lot of different hardware configurations
- Terraform and Ansible integration
- Bare-Metal Servers are online in a few minutes
- Custom Hardware options
  - Similar to reserved instances in other Public Clouds
- On-Premise options if you need this

PROMO CODE: PLATFORM9

$100 in free credits with servers starting a $0.05 per hour!
FINE!! Do what you want!

AUTOMATION, AUTOMATION, AUTOMATION.

Try and be like Packet. Automate the process as much as possible from the hardware being ordered to being provisioned. Try and get as much automation in place as possible.

The best Open Source project I’ve found for this is OpenStack Ironic. You can manage and provision Bare Metal servers as well as configure L2-L4 switch functions. (Provided you purchase supported hardware.)
Load Balancers

Private and Public Cloud solutions provide out of the box load balancing. With bare-metal this is now a problem to solve.

What we’ve seen in the wild for this most commonly are:

- F5 Big-IP
- Citrix NetScaler
- MetaLB

With this being an Open Source summit we’d like to focus on MetaLB.
MetalLB is currently in beta.
If you need don’t have the appetite for a beta product, you’ll be best served by paying for F5 or NetScaler

But that being said MetalLB works well and is very actively developed.
You can run MetalLB in Layer 2 mode or Layer 3/BGP mode.

Layer 2 is much easier and doesn’t require any work from the networking side of things.

While Layer 3/BGP mode is more difficult to setup and requires network side changes… But there are pros and cons to both:
Metal LB - Layer 2 Mode

Image stolen form: https://kubernetes.github.io/ingress-nginx/deploy/baremetal/
Metal LB - Layer 3/BGP Mode

Image stolen form: https://kubernetes.github.io/ingress-nginx/deploy/baremetal/
And poorly modified by me.
Networking setup is also more difficult for Bare-Metal nodes vs the Public cloud. Public clouds allow you to lean on Security groups to protect your nodes. They handle propagation of DDOS and all the things that you don’t really want to have to think about. These are things that you’ll need to invest in at the edge and inside your kubernetes cluster.

Me not being an Edge networking expert… I won’t be speaking about the DDOS protection. (I lean on Packet for that) But what you will need to work on is managing pod security. This is where Network Policies comes in. And Calico is the best option in my opinion here. Now there is a known bug using both MetalLB BGP and Calico BGP together. You either need to peer to two seperate layers of switches or you should run Calico in IP-2-IP mode.
If you’re deploying Kubernetes on a public or private cloud… There is a native Kubernetes driver that will handle all of the storage for you. All you have to do is setup your storage classes and you’re good to go.

On bare-metal there is a little more work to be done. And we’ll go though some of these options
You need to make a choice between:

Container Native Storage
or
Enterprise Storage
You need to make a choice between:

Commercial (Closed Source) Storage
or
Open Source Storage
Benchmarking Storage

- Random IOPS
  - Pure Read/Pure Write IOPS
  - 70/30 Read/Write Mixed IOPS
- Sequential Throughput
  - High queue depth, low block size
  - Low queue depth, high block size
- Latency

- 12 HDDs + 1 NVMe cache if supported
  - Cheap HDD test
- 1 NVMe (3x FIO)
  - Real world SSD test, maximum performance
- 1 NVDIMM (1x FIO)
  - Best case hardware (software bottleneck)
Random IOPS

- Excellent Random Read performance from Rook and Portworx
- Both extract more aggregate IOPS than 12 * 1 HDD would imply
Random IOPS

NVMe IOPS 3x FIO (Higher is better)

- Random R: 10,000, 10,000, 30,000
- Random W: 16,500, 16,500, 32,000
- 70% Mixed R: 10,000, 10,000, 20,000
- 30% Mixed W: 10,000, 10,000, 15,000

NVDIMM IOPS 1x FIO (Higher is better)

- Random R: 12,000, 280,000
- Random W: 14,000, 223,000
- 70% Mixed R: 24,000, 195,000
- 30% Mixed W: 8,092, 64,000

Legend:
- Rook (3x replica)
- Portworx (3x replica)
- OpenEBS (3x replica)
- Bare Metal Native NVMe (libaio)
- Bare Metal Native NVMe (psync)
- 2x Bare Metal NVMe (libaio)
- 2x Bare Metal NVMe (psync)
Bandwidth

- Rook compares better when using many HDDs vs fewer SSDs
- Rook and Portworx are again able to extract more aggregate throughput than 12 * 1 HDD
Bandwidth

NVMe Bandwidth MiB/s 3x FIO (Higher is better)

NVDIMM Bandwidth MiB/s 1x FIO (Higher is better)
Latency

- Portworx has extremely low latency, comparable to bare-metal performance
Latency

NVMe Latency Microseconds 3x FIO (Lower is better)

Latency Rate
- Rook (3x replica): 337
- Portworx (3x replica): 320
- OpenEBS (3x replica): 1,696
- Bare Metal Native NVMe (libaio)
- Bare Metal Native NVMe (psync)

Latency Write
- Rook (3x replica): 89
- Portworx (3x replica): 65
- OpenEBS (3x replica): 1,999
- Bare Metal Native NVMe (libaio)
- Bare Metal Native NVMe (psync)

NVDIMM Latency Microseconds 1x FIO (Lower is better)

Latency Rate
- Rook (3x replica): 109
- Portworx (3x replica): 287
- OpenEBS (3x replica): 727
- Bare Metal Native NVDIMM (libaio)
- Bare Metal Native NVDIMM (psync)

Latency Write
- Rook (3x replica): 14
- Portworx (3x replica): 0
- OpenEBS (3x replica): 960
- Bare Metal Native NVDIMM (libaio)
- Bare Metal Native NVDIMM (psync)

Latency W
- Rook (3x replica): 22
- Portworx (3x replica): 0
- OpenEBS (3x replica): 1,955
- Bare Metal Native NVDIMM (libaio)
- Bare Metal Native NVDIMM (psync)
PVC Cap

- Rook - Limit unknown, >4450 pods at the moment
- Portworx - 1024, licensing limit
- OpenEBS - 944, iSCSI error
Storage Findings
Kubernetes Best Practices

- Rack / Row / Room / Datacenter Diversity (Depending on Scale)
  - Use “failure-domain.beta.kubernetes.io” features to allow you stateful sets to automatically spread themselves across failure domains.
- Run ETCD outside of your Cluster
  - Take automated ETCD backups in case of large disaster
- Run 5x replicas of ETCD (Minimum)
- Run 5x master nodes (Minimum)
- Don’t run workloads on your masters
- Run a Kubernetes version of N - 2 (or N - 1 if you really need some newer features)
  - I normally will update the the latest dot release right after a new version is released.
  - Currently this is v1.13.8
    - And I’ll update to the latest dot release if there is a critical bug or CVE released that needs attention.
- Run the operating system and docker versions that the rest of the Kubernetes community is running:
  - Currently Ubuntu 18.04 has the lionshare
  - Docker version: 17.09 is the most validated at this point
    - This is N - 2 and there are 2x newer version if you have known issues solved in 18.06 or 18.09
Lots of tools that we couldn’t talk about

Cluster Backup - velero - [https://github.com/heptio/velero](https://github.com/heptio/velero)
Kubernetes Density Testing - MMM - [https://github.com/AleksandrNull/MMM](https://github.com/AleksandrNull/MMM)

Monitoring - Prometheus - [https://github.com/prometheus/prometheus](https://github.com/prometheus/prometheus)
Visualization - Grafana - [https://github.com/grafana/grafana](https://github.com/grafana/grafana)
Alerting - Alert Manager - [https://github.com/prometheus/alertmanager](https://github.com/prometheus/alertmanager)
Monitoring at scale - Cortex - [https://github.com/cortexproject/cortex](https://github.com/cortexproject/cortex)

Logging - EFK Stack
- Elasticsearch - [https://github.com/elastic/elasticsearch](https://github.com/elastic/elasticsearch)
- Fluentd - [https://github.com/fluent/fluentd](https://github.com/fluent/fluentd)
- Kibana - [https://github.com/elastic/kibana](https://github.com/elastic/kibana)

Ingress (and more) - Envoy - [https://github.com/envoyproxy/envoy](https://github.com/envoyproxy/envoy)
Ingress - Nginx - [https://github.com/kubernetes/ingress-nginx](https://github.com/kubernetes/ingress-nginx)
Service Mesh - Istio - [https://github.com/istio/istio](https://github.com/istio/istio)
Questions?