

Building a Scalable Data Science & Machine Learning Cloud using Kubernetes

Murali Paluru
Principal Software Engineer



Brief history

- Few years back, a small team of Data Scientists was put together
- The best possible hardware available back then was provided.
- As the team grew over time, more capacity was added.

Problem(s) with the current platform

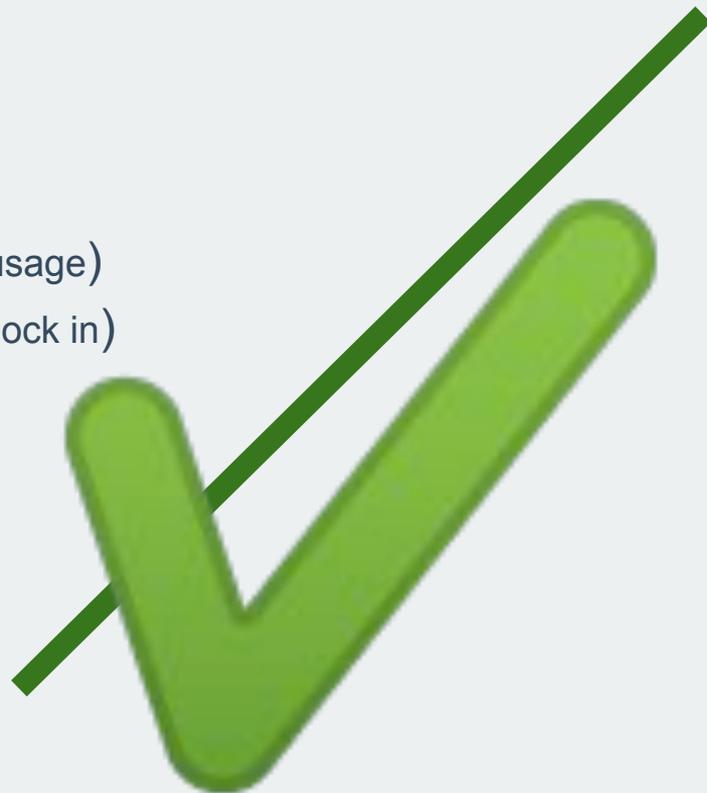
- The server became a bottle neck!
 - Dependency hell
 - Installation of new versions of software/library took a long time due to IT processes.
-
- What can be done?

Requirements for the new platform

- Scalability (Easily accommodate team growth)
- Elasticity (Scale up or down the resources based on usage)
- Multi Cloud and On-Prem support (No vendor lock in)
- High Availability
- Resource limits (Big Data is resource hungry!)
- Security (No downloading of data onto the laptop)
- Authentication/Authorization
- Storage Integration (NFS, Cloud storage, etc)
- Self Service (Put the data scientist in the driver seat)
- Ease of use

Go Kubernetes!

- Scalability (Easily accommodate team growth)
- Elasticity (Scale up or down the resources based on usage)
- Multi Cloud and On-Prem support (No vendor lock in)
- High Availability
- Resource limits (Big Data is resource hungry!)
- Security (No downloading of data onto the laptop)
- Authentication/Authorization
- Storage Integration (NFS, Cloud storage, etc)
- Self Service (Put the data scientist in the driver seat)
- Ease of use



Plan for POC (Proof of Concept)

1. Containerize the DS/ML environments/workloads
2. Private Registry
3. SSL Certificates for secure communication
4. Utilize wildcard DNS for Ingresses of the workloads
5. Integrate with LDAP/AD
6. NFS support
7. Build a Highly Available (HA) workload cluster
8. Launch DS/ML workloads

Here you go!

```
>
>
>
>
> kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
leo-all-in-one-h1  Ready    controlplane,etcd,worker  237d   v1.13.10
>
>
>
> kubectl -n ns-mpaluru-rstudio get pods
NAME                READY    STATUS    RESTARTS   AGE
w-rstudio-84b8fc7b68-fs2w8  1/1     Running    0           8h
>
>
>
> kubectl -n ns-mpaluru-jupyter get pods
NAME                READY    STATUS    RESTARTS   AGE
w-jupyter-86dbb8cd49-tsdr2  1/1     Running    0           8h
>
>
>
>
> █
```

Quick Demo

Requirements for the new platform

- ✓ Scalability
 - Easily accommodate team growth
- ✓ Elasticity
 - Scale up or down the resources based on usage
- ✓ Multi Cloud support
- ✓ High Availability
- ✓ Resource limits
- ✓ Authentication/Authorization
- ✓ Storage Integration
- ✓ Self Service
- ? Ease of use

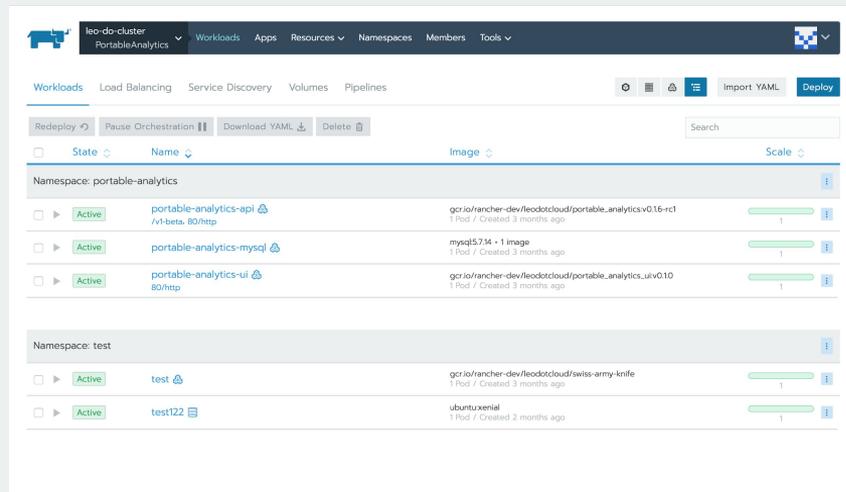
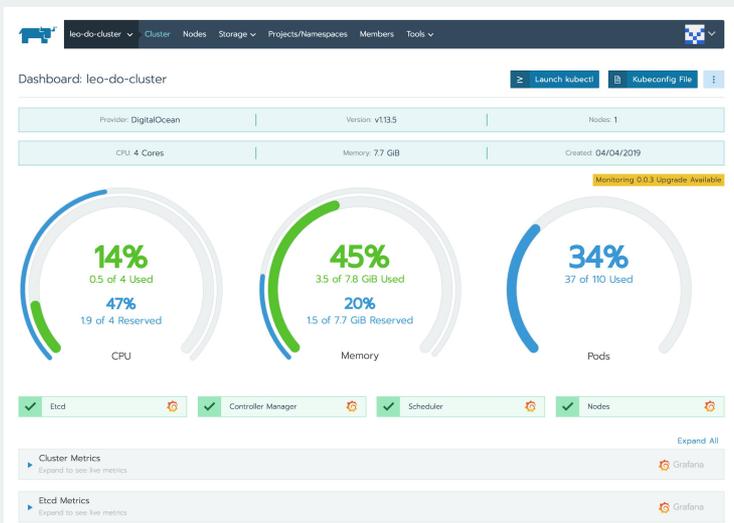
Kubernetes is awesome, but ...

- It's not easy to use!
- Data Scientists don't have time to get a PhD in Kubernetes

```
>
>
>
> kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
leo-all-in-one-h1  Ready    controlplane,etcd,worker  237d  v1.13.10
>
>
>
> kubectl -n ns-mpaluru-rstudio get pods
NAME                READY    STATUS    RESTARTS  AGE
w-rstudio-84b8fc7b68-fs2w8  1/1     Running    0          8h
>
>
>
> kubectl -n ns-mpaluru-jupyter get pods
NAME                READY    STATUS    RESTARTS  AGE
w-jupyter-86dbb8cd49-tdsr2  1/1     Running    0          8h
>
>
>
>
```

- Explore third party solutions to provide a simpler user experience
- Rancher

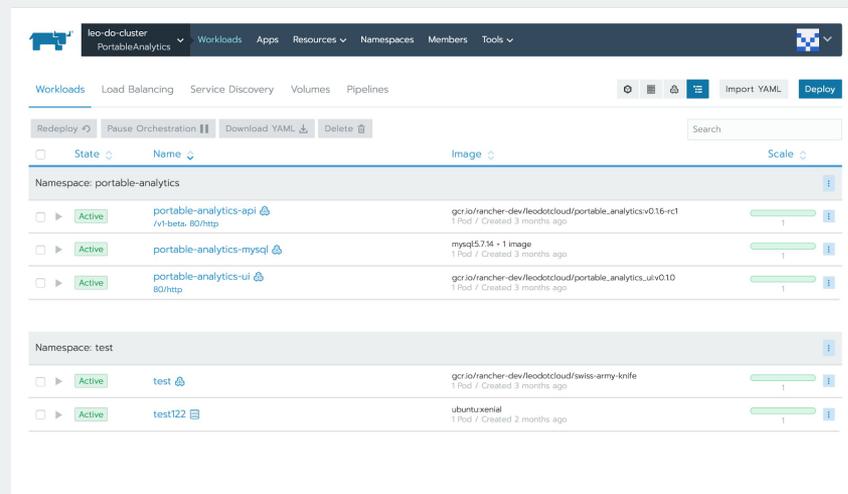
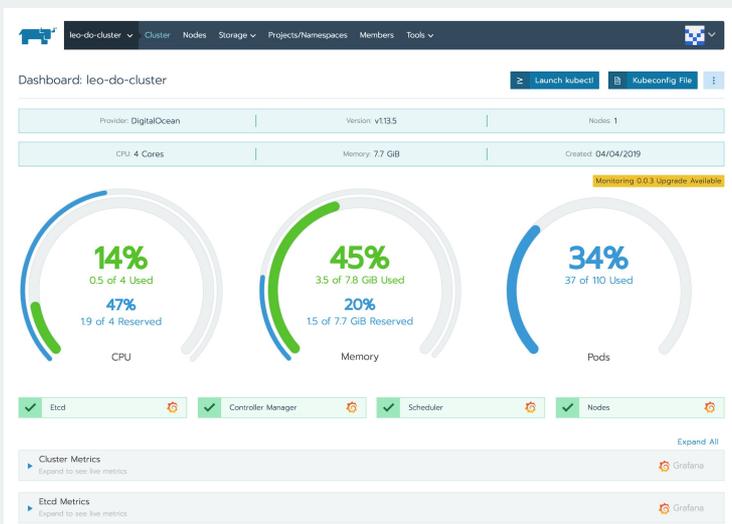
Here you go!



Quick Demo

Rancher makes it easy, but ...

- A Data Scientist still has to learn about Kubernetes Concepts

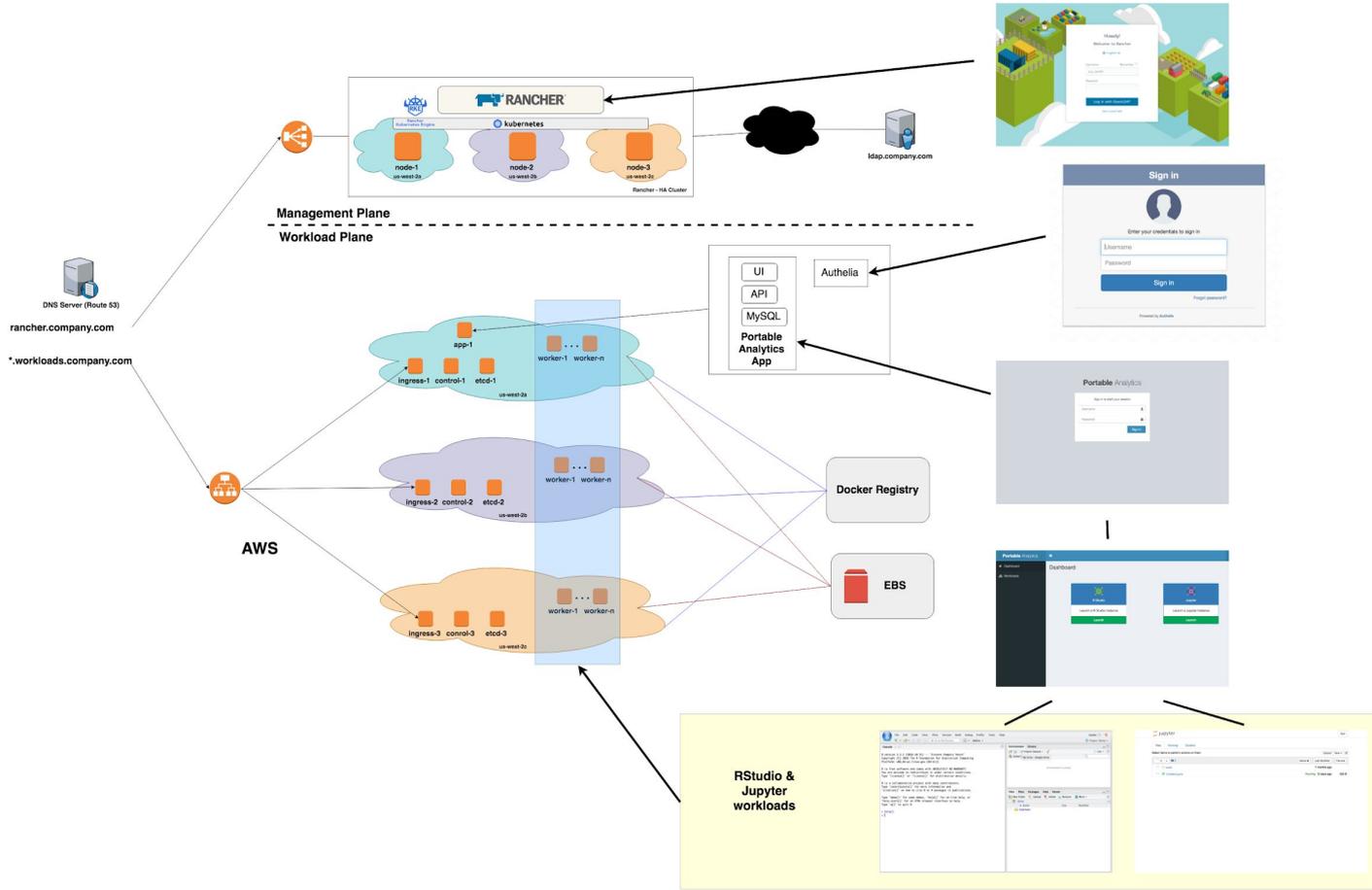


- Can it be made much simpler? (Well, let's build one!)

Quick Demo

Behind the Scenes

Architecture



Registry

- Evaluated multiple options
- Docker Hub was not an option as AWS, GCP, Azure were already being consumed
- Self hosting is quite tedious
- ECR provides a per-AZ endpoint, hard to manage config
- GCR provided a single endpoint with global replication

Cons of GCR:

- The credentials are not simple token values, huge json blob

Main workloads

- **RStudio** (R programming language)
 - Various versions of R
 - Dependencies (Installation takes a lot of time)
 - Private libraries
 - Open source RStudio container image doesn't support LDAP integration
- **Jupyter** (Python programming language)
 - Various versions of Python
 - Conflicting packages
 - Private packages
 - Hard to integrate LDAP support

Disable container auth (and use Ingress authentication)

High Availability

- Use of multiple availability zones (AZ)
- Redundancy (scale=n)
 - Multiple etcd nodes (1, 3, 5 ...)
 - Redundant control plane, ingress nodes
 - Workers in different AZs
- Cons of availability zones with EBS:
 - Once a PVC is created in a particular AZ, a dependant workload can't move to a different AZ without loss of data
 - In case of AZ failure, not all users are affected, but it's hard to migrate.
- Future work: Investigate and use EFS

Ingress & DNS

- Easy to bookmark URLs for users
- Wild card DNS
 - Scalable
 - No more change requests after the first one
- Ingress authentication
- SSL Termination
 - Terminate on external LB
 - Reduce overhead on the cluster resources
 - Use Wildcard SSL certificate
- Future work:
 - Input from user for the name of workloads

Authelia

- It's an open-source full-featured authentication server
- Pros:
 - Per Ingress authentication
 - Easy to setup and use
 - Supports multiple auth backends
- Cons:
 - Doesn't support a config API endpoint (needs to restart Pod on config change)
- <https://github.com/clems4ever/authelia>

```
access_control:  
  default_policy: deny  
  any:  
    - domain: '*.alpha.do.mpaluru.com'  
      policy: deny  
  groups: {}  
  users:  
    mpaluru:  
      - domain: mpaluru-rstudio.alpha.do.mpaluru.com  
        policy: allow  
      - domain: mpaluru-jupyter.alpha.do.mpaluru.com  
        policy: allow
```

Storage

- NFS
 - Easy to guess the shares (use UUID)
 - Could have used dynamic provisioner
- EBS (Initial platform was built using AWS)
 - Pre-create the PVC
 - Mainly for persisting user settings, non critical data

Resource Limits

- One user's workload shouldn't impact another user's
 - Use of Requests/Limits of CPU and Memory
- Different worker planes for various workloads
 - Use of Taints/Tolerations
- Cons:
 - Request/Limits > Node Capacity?
 - Fragmentation
- Future work:
 - Support for GPUs

Summary

- Kubernetes is a great platform for running Data Science/Machine Learning workloads (If you are a DS/ML researcher, **ASK** your IT team, it's possible!!!)
- Managed Kubernetes Services reduce the overhead of managing master nodes
- Providing a simple UX for end users still places a lot of burden on the cluster operator
- Use a higher level framework instead of working with native k8s manifests
- Support for other DS/ML projects
- Coming soon: <https://github.com/k4ds>



Murali Paluru
Principal Software Engineer

 <https://slack.rancher.io> (leodotcloud)

 <https://twitter.com/leodotcloud>

 <https://github.com/leodotcloud>

 <https://linkedin.com/in/leodotcloud>

Thank you!

Questions?