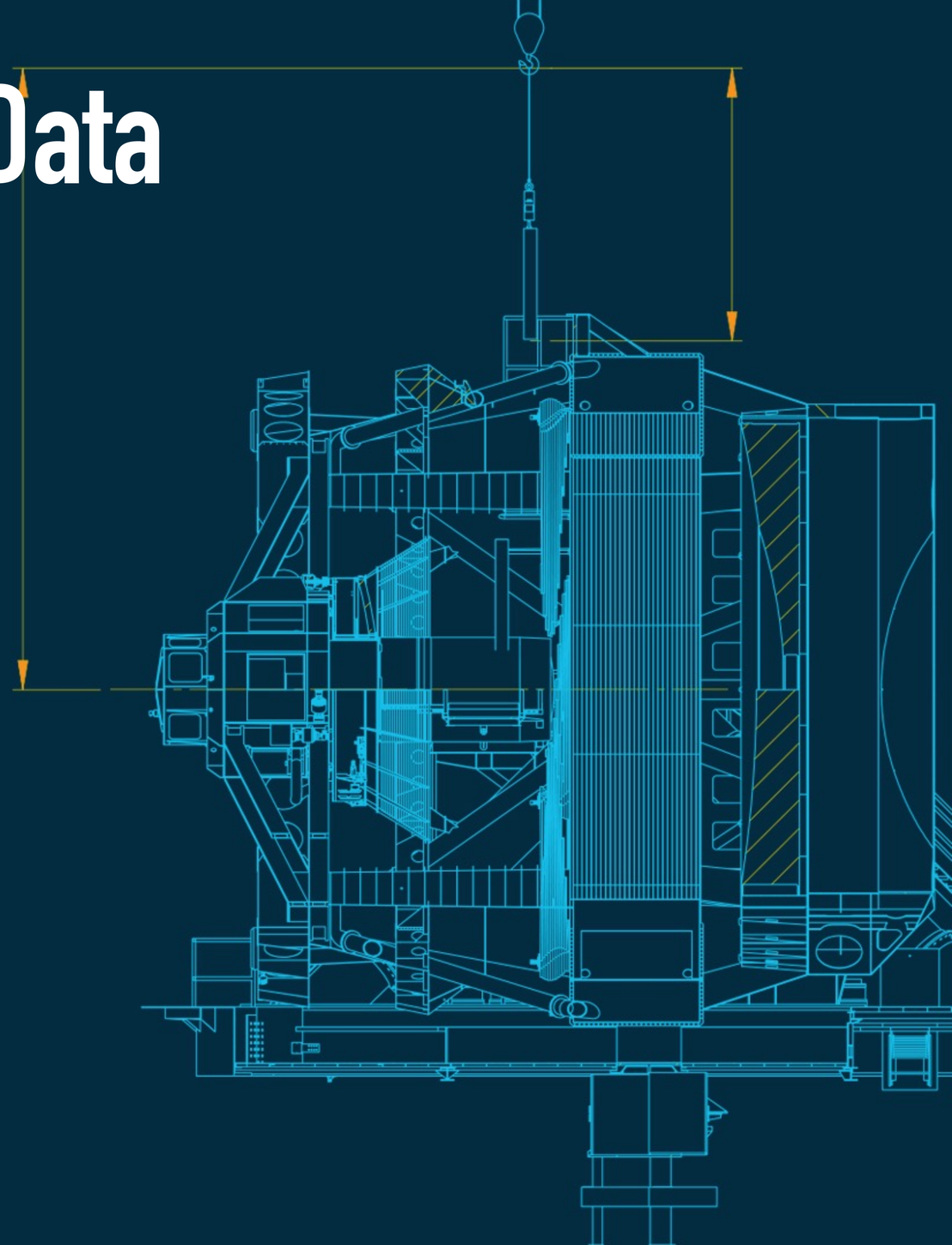
Bring Your Service To The Data With Kubernetes

Christine Banek @ LSST
cbanek@lsst.org

IVOA Interoperability Meeting
Paris - May 14th, 2019



Large Synoptic Survey Telescope

Bring Your Service to the Data with Kubernetes

Agenda

1. Requirements for deploying and operating software services
2. The solution (hint: it's Kubernetes)
3. Kubernetes can help us share IVOA services
4. LSST case study
5. Join in on the fun

Requirements for Deploying and Operations

We've got big data, so:

1. Users brought to the data with tools like Jupyter notebooks
2. User code calls local services for fast access to data
3. Services need to be able to deployable in all datacenters
4. Services may be scaled differently in different datacenters

Our Data Access Centers:

1. Multiple datacenters around the globe (and in the cloud)
2. Machines of different configurations (CPU/RAM/storage)
3. Different number of machines in each datacenter, which change over time

Solution: Docker + Kubernetes

Docker Containers

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

HTTPS://WWW.DOCKER.COM/RESOURCES/WHAT-CONTAINER

Kubernetes

Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

HTTPS://KUBERNETES.IO/DOCS/CONCEPTS/OVERVIEW/WHAT-IS-KUBERNETES/

Solution: Docker + Kubernetes

Why Docker + Kubernetes?

Provides an abstraction between developers and datacenters:

- Datacenters provide a Kubernetes cluster to run services
- Datacenters manage the cluster and its physical resources
- Datacenters can run a service without deep knowledge of ports, disks, configuration, shared libraries, etc.
- Developers build docker containers that completely contain a service
- Developers describe Kubernetes deployments to manage docker containers

Solution: Docker + Kubernetes

Why Docker + Kubernetes?

Great for developers:

- Complete isolation from other applications and host
- Target one environment (the container) rather than cross-platform
- Specify CPU/memory requirements, disk volumes, and environment variables in YAML that can be source controlled
- Easy operations by developers via kubectl
- Developers can test on their own cluster or on a shared cluster

Solution: Docker + Kubernetes

Why Docker + Kubernetes?

Great for operators:

- No need to install libraries or compile software, deploy in minutes
- Containerized applications cannot dirty host state
- Kubernetes helps manage networking, DNS, certificates, and secrets
- Kubernetes automatically restarts failed services
- Kubernetes automatically moves services on node failure
- Easily scale up or down the replicas of a service with one command

Solution: Docker + Kubernetes

Why Docker + Kubernetes?

It's catching on:

- Strong community backing by the open source community, companies, and sites like Stack Overflow
- Commodity clouds (AWS, Google) provide managed Kubernetes clusters that can be brought up in minutes
- Companies are publishing docker containers and ways to deploy those containers on Kubernetes

Solution: Docker + Kubernetes

Sharing Deployments with Kubernetes

Many companies support installing their software in a Kubernetes cluster.

There are a few ways to share a Kubernetes deployment:

1. Kubernetes YAML
2. Helm chart
3. Kustomize YAML
4. Integration with deployment tools like Terraform

Typically with one command, a service can be installed on a Kubernetes cluster.

Sharing IVOA Services with Kubernetes

If we all run Kubernetes, and write services for Kubernetes…

We could all run each others services!

Sharing IVOA Services with Kubernetes

What a wonderful universe it could be

Imagine if:

1. We have a set of popular reference clients for IVOA protocols (√)
2. We share not only the standards for astronomy, but implementations too!
3. Observatories can write plugins for adapting to their data and requirements
4. Easy reuse and improvement of implementations that live on past one instrument
5. Anyone can spin up a reference implementation of a VO compliant datacenter, import their own data, and run their own tools

LSST Case Study

Walking the walk, not just talking the talk

1. LSST needs a TAP service, and I'm the lucky developer
2. Search Google for "TAP service," proceed to have crisis of confidence
3. Find CADC's TAP service on GitHub, rejoice
4. Email CADC asking how to proceed, follow their instructions
5. Take CADC's git repository, fork it, get it running on Kubernetes
6. Make some small upstream fixes and changes that help me and help CADC
7. Rejoice, confidence restored! Spread my joy and recipe for success
8. Continue collaborating with my new friends and making the world a better place

Join in on the fun

If this sounds like a good idea, you can…

1. Publish your code on GitHub
2. Containerize the build, test, and deploy process for services
3. Run your service on Kubernetes
4. Document, share, and publish ways to use common tools to help others run your service
5. **Re-use other existing services whenever possible! If it does most of what you need, consider forking the code and/or push changes upstream!**

The miracle is this –

the more we share,
the more we have.

LEONARD NIMOY

Additional Resources Follow

Docker Resources

Docker: https://www.docker.com

Learning docker: https://www.oreilly.com/library/view/learning-docker/9781491956885/

Using docker: https://www.oreilly.com/library/view/using-docker/9781491915752/

Learn enough docker to be useful: https://towardsdatascience.com/learn-enough-docker-to-be-useful-b7ba70caeb4b

Kubernetes Resources

Scalable Microservices with Kubernetes: https://classroom.udacity.com/courses/ud615

Kubernetes concepts: https://kubernetes.io/docs/concepts/

Minikube tutorial: https://kubernetes.io/docs/tutorials/stateless-application/hello-minikube/

Kubernetes up and running: https://www.oreilly.com/library/view/kubernetes-up-and/9781491935668/

Helm: https://helm.sh/

Kustomize: https://kustomize.io/

LSST Resources

CADC on GitHub: https://github.com/opencadc

LSST fork of the CADC TAP service: https://github.com/lsst-sqre/lsst-tap-service

Beginning of the LSP deploy: https://github.com/lsst-sqre/lsp-deploy (see logging for an example of using existing helm charts)

Background on LSST IVOA strategy and desires: https://dmtn-090.lsst.io/