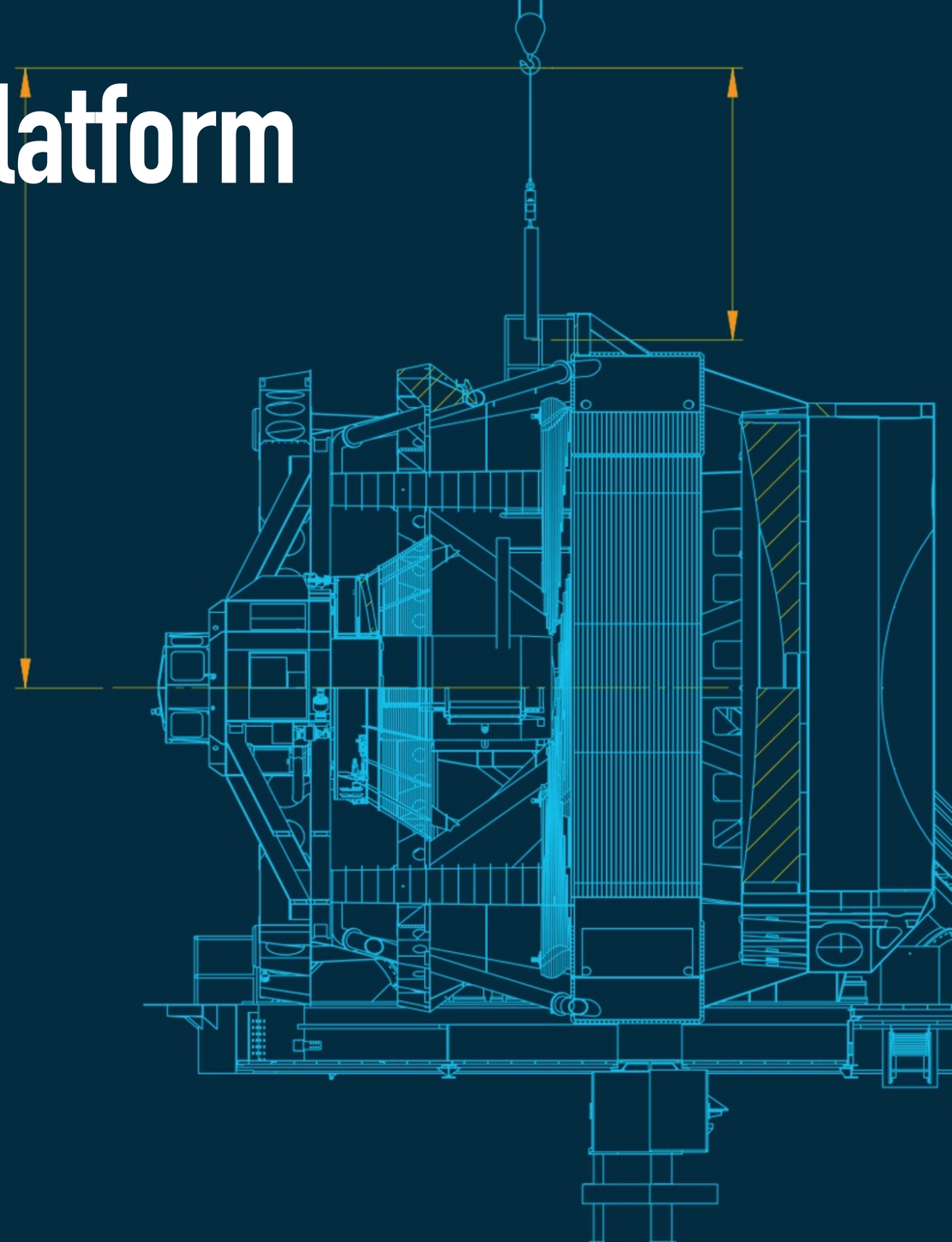
Why is the LSST Science Platform Built on Kubernetes?

Christine Banek @ LSST
cbanek@lsst.org

ADASS 2019
Groningen - October 2019



Large Synoptic Survey Telescope

Why is the LSST Science Platform Built on Kubernetes?

Agenda

1. The key unstated requirement for any scientific software
2. A (mostly) complete history of software deployment in 5 minutes
3. Brief introduction to Kubernetes and Helm
4. Installing the LSST Science Platform
5. Enabling future multi-dataset science platforms

The Key Unstated Requirement of Scientific Software is…

Reproducibility

All scientific software needs reproducible…

1. Creation (source control, compilation)
2. Installation, deployment, and configuration
3. Behavior, testing, and results

If you can't trust your software, you can't trust your science.

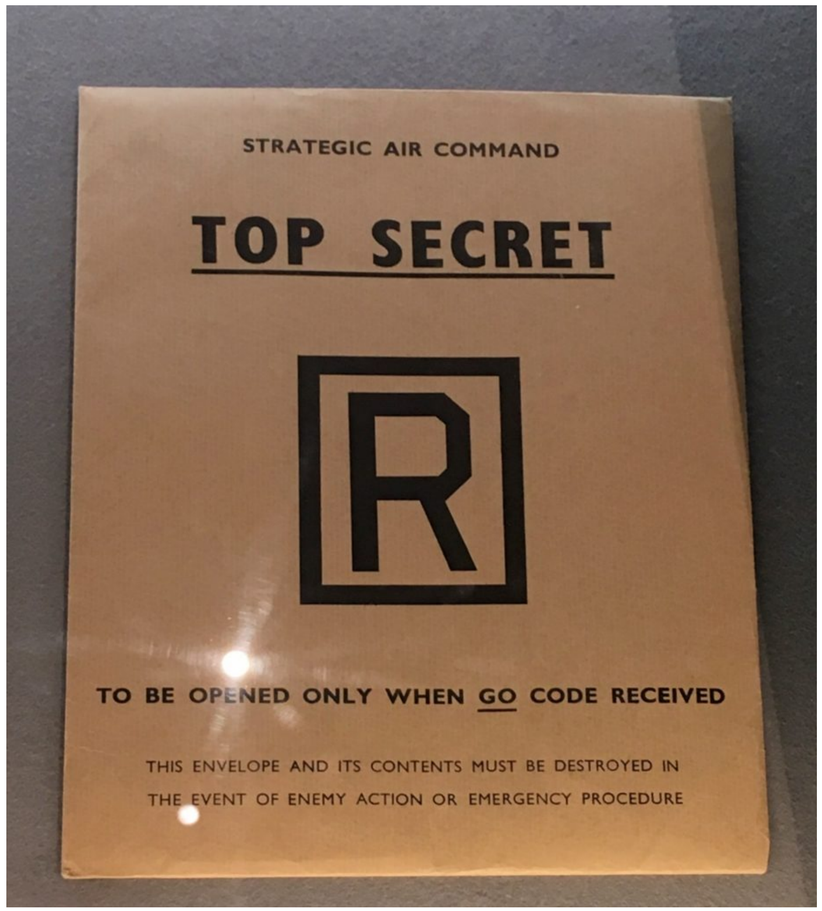
The Key Unstated Requirement of Scientific Software is…

Reproducibility

Plan R:

Always be ready to re-deploy your software from scratch.

1. If you have user data, be able to backup and restore it (and test this procedure regularly)
2. Best if this can be done by one person of a technical nature, not just the owner/creator



A (Mostly) Complete History of Software Deployment in 5 Minutes

How has the way software is built and deployed changed?

What we wish for:

1. Fewer people needing to be involved, more self-service
2. Able to run applications with (possibly conflicting) dependencies
3. Simpler installation steps to lower the barrier to entry and lower risk of failure
4. Portable installation that works everywhere

A (Mostly) Complete History of Software Deployment in 5 Minutes

The Four Great (and not so great) Ages of Software Deployment

1. Sysadmin
2. Virtualization
3. Containerization
4. Container Orchestration

A (Mostly) Complete History of Software Deployment in 5 Minutes

Your Local (Benevolent?) Sysadmin

1. You call your local sysadmin and tell them what you need installed or fixed
2. Sysadmin finds a machine for it to run on
3. Sysadmin installs and configures the software for you on that machine
4. If things break, patiently wait for the sysadmin to fix it

Do not anger the sysadmin, for he has root, and you likely do not.

A (Mostly) Complete History of Software Deployment in 5 Minutes

Virtualization

1. Call your sysadmin and tell them what resources you need
2. Sysadmin creates a Virtual Machine, and emails the root password to you
3. You install the software on the machine and are responsible for it
4. Virtual Machines generally live forever, or until no longer needed

Developers self-manage their software

Different virtual machines can run otherwise conflicting versions of software

A (Mostly) Complete History of Software Deployment in 5 Minutes

Containerization

1. Physical hosts or VMs run Docker and host containers
2. Containers created once at build time with all dependencies included
3. Building and installation steps removed from deployment time
4. Hosts not altered by installing or running software

Each application runs in isolation, no cruft from previous versions

Containers run the same anywhere

A (Mostly) Complete History of Software Deployment in 5 Minutes

Container Orchestration

1. Schedule containers on a pool of hosts
2. Containers moved from unhealthy hosts and restarted automatically on healthy hosts
3. Orchestrator manages DNS and routing to the right containers
4. Automatically start and stop containers (and hosts) to elastically scale to load

Benefits of containers across multiple hosts

Automatically recover from host failures

Brief Introduction to Kubernetes and Helm

Kubernetes

1. An open-source container orchestrator started by Google
2. The current de facto platform for container orchestration
3. Native support by cloud providers such as Amazon and Google
4. Resources defined and configured by YAML documents control behavior
5. Resources are declarative - stating the desired state, Kubernetes makes it happen

Brief Introduction to Kubernetes and Helm

Kubernetes Resource Types

1. Pod - one or more containers on the same host
2. Deployment - groups of identical pods with a desired replication factor
3. Service - grouping of pods that can be discovered by DNS name
4. Volumes - persistent storage that pods can mount
5. Configmap - configuration files that pods can mount
6. Secret - configuration such as passwords or certificates that pods can mount

Brief Introduction to Kubernetes and Helm

Helm

1. A package manager for Kubernetes
2. Group a set of resources into a Helm chart that represents an application to run in Kubernetes
3. Resource YAML documents run through a templating engine that injects user values into Kubernetes resources
4. Helm charts can be easily published and shared between organizations

Installing the LSST Science Platform

LSST Science Platform Helm Charts

1. Landing page - simple web page to help users navigate to the different aspects of the Science Platform
2. Nublado - customized JupyterHub system to allow users to run notebooks next to the data
3. Firefly - IPAC portal for in-browser data visualization
4. CADC's TAP service - IVOA TAP service for QServ
5. Fileserver - NFS fileserver for persistent storage of notebooks and user data

Installing the LSST Science Platform

Installation Helper Scripts

1. Clone the lsp-deploy repository on GitHub (URL in additional slides)
2. ./install_tiller.sh - installs the Kubernetes side of Helm
3. ./install_ingress.sh - takes your SSL certificate as arguments, configures nginx to proxy backend services
4. ./install_lsp.sh - installs all Helm charts for the LSP
5. Celebrate! Now you too can install the LSST Science Platform!

Installing the LSST Science Platform

Helm + Kubernetes: A Great Way to Install the LSST Science Platform!

1. Easy to install for internal and external users in minutes, even on new deployments
2. Simple as running containers - no need to worry about dependencies or how to build the software, that is handled by the container creator
3. Helm charts manage Kubernetes resources, abstracting out actual configurable variables in a values file
4. Namespaces prevent naming conflicts, other software can be installed alongside the LSP with no changes

Future Multi-dataset Science Platforms

Requirements of Multi-dataset Science Platforms

1. User requirements - what scientists want
2. Service requirements - what software need to be provided to support users
3. Operator requirements - how to operate at scale and manage hardware

Future Multi-dataset Science Platforms

User Requirements of Multi-dataset Science Platforms

1. Rich set of interactive analysis tools like notebooks, batch processing, and visualization
2. Users want to run the same code on a mirror of the data as they would on the original
3. Use standard clients that work with VO protocols

Future Multi-dataset Science Platforms

Service Requirements of Multi-dataset Science Platforms

1. Host multiple large datasets
2. Ability to install the tools required to serve the data (databases, IVOA services, etc.)
3. Concurrently running analysis tools from multiple science platforms
4. Easily customize and configure services to work seamlessly in a multi-dataset science platform configuration

Future Multi-dataset Science Platforms

Operator Requirements of Multi-dataset Science Platforms

1. Scale up to a large number of machines to host the data and run analysis
2. Scale down to a small number of machines for testing and development
3. Run on bare metal as well as in the cloud
4. Fault tolerance to hardware issues
5. Easy to install, update, and manage services
6. Provider operator services like centralized logging and health checks

Future Multi-dataset Science Platforms

Kubernetes + Helm to the Rescue

1. Kubernetes is great for deploying in a fault tolerant way on a cluster
2. Kubernetes looks the same in the cloud, or in any datacenter
3. Everyone has access to a Kubernetes cluster in the cloud!
4. Helm is great for packaging services to run in Kubernetes
5. Quick, reliable installs make running someone else's software a lot easier

Future Multi-dataset Science Platforms

Open the Bazaar

1. Put your code on GitHub from the start!
2. Run your software in containers, and publish the containers publicly
3. Develop Helm charts to easily schedule containers in Kubernetes
4. Helm encourages sharing and re-use of code, share charts with others!
5. Soon we could have a thriving open source ecosystem of easy to use astronomy tools that can be installed into any science platform
6. Re-using science platform code reduces cost and time, while increasing quality

Thank You!

Additional Resources Follow

Repositories

Find us on GitHub

LSP Deployment Scripts: https://github.com/lsst-sqre/lsp-deploy

LSP Helm Charts: https://github.com/lsst-sqre/charts

Nublado: https://github.com/lsst-sqre/nublado

Firefly/SUIT: https://github.com/lsst/suit https://github.com/Caltech-IPAC/firefly

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

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

Further reading

https://docker-curriculum.com

https://docs.docker.com/get-started/

https://kubernetes.io/docs/tutorials/

https://helm.sh

https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-cluster

https://helm.sh/docs/chart_repository/

https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart/

https://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar